UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Scripts para automatización de procesos en análisis de datos y generación de dashboards

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Maximiliano Martínez Contreras

ASESOR DE INFORME

Ing. Carlos Omar Calieca Romero

Ciudad Universitaria, Cd. Mx., 2026

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado SCRIPTS PARA AUTOMATIZACION DE PROCESOS EN ANALISIS DE DATOS Y GENERACION DE DASHBOARDS. que presenté para obtener el titulo de INGENIERO MECATRÓNICO es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

MAXIMILIANO MARTINEZ CONTRERAS
Número de cuenta: 421092309

Índice

Scripts para automatización de procesos en análisis de datos y generación de dashboards.

1. Acerca de la empresa

Schneider Electric es un líder mundial en tecnología energética que impulsa la eficiencia y la sostenibilidad mediante la electrificación, automatización y digitalización de industrias, empresas y hogares.

Sus tecnologías permiten que edificios, centros de datos, fábricas, infraestructuras y redes funcionen como ecosistemas abiertos e interconectados, para mejorar así el rendimiento, la resiliencia y la sostenibilidad. Su cartera de productos incluye dispositivos inteligentes, arquitecturas definidas por software, sistemas impulsados por la IA, servicios digitales y asesoramiento experto. [33]

En el siglo XIX, Schneider & Cie fue pionero en los procesos de fabricación de hierro utilizando máquinas alimentadas con carbón y vapor, estableció plantas siderúrgicas y entró al emergente sector eléctrico.

A principios del siglo XX, los avances técnicos dieron paso a los sistemas de energía eléctrica y las líneas de producción. La demanda de materiales y equipos electrotécnicos reemplaza al carbón y al vapor, lo que le permite a Schneider diversificarse y expandirse.

La era del control y la automatización comienza en la segunda mitad del siglo XX, fomentando el uso generalizado de la electrónica, telecomunicaciones, computadoras y energía nuclear. Las adquisiciones estratégicas cierran el futuro de Schneider Electric con experiencia en distribución eléctrica y automatización.

El crecimiento exponencial de Internet y la Industria 4.0 posicionan a Schneider Electric como el socio tecnológico de la energía, liderando la convergencia de la electrificación, la automatización y la inteligencia digital. No nos limitamos a conectar sistemas; creamos ecosistemas en los que la IA, los datos y las personas trabajan juntos a la perfección. [34]

2. Introducción

Se presenta a continuación un reporte de las actividades realizadas durante mi estancia como becario en una empresa multinacional con enfoque en tecnología industrial, en el puesto de NEST Intern (North America Engineering Support Team), destacando mis actividades principales dentro del corporativo, definiendo aquellos puntos importantes en mi desarrollo profesional dentro de este ámbito y como éstos impactaron en la agilización de los procesos y la toma de decisiones por parte del equipo de trabajo NEST en donde me encontraba.

Mis actividades comprendían lo relacionado a implementar una estrategia para agilizar el **proceso de obtención, limpieza, normalización y filtrado de datos**, los cuales nos permiten proceder un análisis más detallado de las medidas estadísticas. Además, era necesario desarrollar y **mejorar los dashboards (gráficas)** para su posterior visualización usando software especializado. Dichos dashboards los generé para poder presentar datos de manera más ordenada y relevante, y de esta forma poder presentar información contundente acerca del desempeño de los colaboradores de NEST.

Además, como actividad complementaria a la implementación de la estrategia de mejora, realicé la documentación donde se detallaba el procedimiento para hacer uso de las herramientas que diseñé, programas para automatizar el proceso de análisis de datos, y la organización de las bases de datos, que resultó relevante para su seguimiento y reutilización en el futuro.

3. Objetivo

- Desarrollar y codificar scripts para la automatización de procesos en la obtención, manipulación y filtrado de datos de facturación, producción, calidad, etc. para el análisis de los Indicadores Clave de Rendimiento (KPI, por sus siglas en ingles).
- Generar dashboards interactivos donde se muestren gráficas relevantes de manera organizada.
- Crear manuales de operación para el uso e implementación de las automatizaciones.

4. Antecedentes

4.1 Metodología Agile

Dentro de las empresas donde existe una cultura de organización basada en la transformación digital, han encontrado atractivo y han optado por implementar metodologías tales que sea posible optimizar los parámetros más relevantes con el objetivo de reducir costos de producción y tiempos de entrega, además de mantener siempre los estándares de calidad que se les ofrece a los clientes.

De esta manera, las metodologías ágiles permiten cambiar las prioridades de cada fase de un proyecto, según los objetivos y necesidades actuales del cliente, el cual tiene como principal estrategia la colaboración continua entre los miembros desarrolladores y coordinadores de un proyecto.

Cuando se piensa en las metodologías ágiles, no debe limitare a hablar únicamente de una herramienta, sino una estrategia integral que impulsa a las grandes organizaciones a gestionar los proyectos con rapidez y flexibilidad.

No es una mentira que día a día el mercado laboral exige mayor flexibilidad y adaptación a las necesidades de los clientes, las cuales, se presentan ante un panorama incierto y cambiante, por esta razón, las empresas deben responder con urgencia ante estas demandas. No es casualidad que las organizaciones estén cada vez más preocupadas por contar con profesionales con una maestría o especialización en dirección de proyectos.

Las metodologías ágiles nos ayudan en el desarrollo de proyectos que necesitan de rapidez y flexibilidad, manteniendo siempre los estándares de calidad y el nivel de servicio, para adecuarse a las necesidades del cliente. [1]

Figura 1. Etapas de metodologías Agile

Los principios básicos que componen la Metodología mostrados en la *Figura 1*:

- Satisfacer al cliente mediante la entrega temprana y continua.
- Aceptar que los requisitos cambien, incluso en etapas tardías del desarrollo. Estos cambios se aprovechan para proporcionar ventaja competitiva al cliente.
- Los responsables de negocio y los desarrolladores trabajan juntos de forma continua durante todo el desarrollo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos de ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Las premisas anteriores nacen para privilegiar la interacción en los procesos y facilita la colaboración con el cliente ante la necesidad de cambios durante el desarrollo del producto. [1]

4.2 Indicadores de rendimiento

Al analizar actividades y procesos en los diferentes campos, es prudente considerar parámetros que son útiles para la toma de decisiones, de esta manera, podemos asegurar que el proceso cumpla con los objetivos definidos por la organización. El seguimiento de estos parámetros permite evaluar el cumplimiento de los objetivos a través de KPI's, los cuales se definen como parámetros digitales que brindan

información acerca del estado actual de los factores clave relacionado con las expectativas definidas por una organización.

Al hacer comparaciones del valor de un KPI con un cierto nivel de referencia, podemos detectar desviaciones, lo que nos permite tener información y poder tomar decisiones o acciones proactivas, preventivas o correctivas para poder mantener las acciones de una organización dentro de los niveles deseados de desempeño, y de esta manera conservar un proceso dentro del rango operativo adecuado.

El objetivo principal de un KPI es medir el nivel de servicio realizando un diagnóstico de la situación en la que se encuentra un proceso dentro de un estándar ya previamente establecido, es decir, los KPI's logran mostrar en qué estado se encuentra un proceso en tiempo real y determinar si estos se cumplen, con esta información, se tiene evidencia tangible del estado de los procesos para comunicar y tomar decisiones en las áreas requeridas. [2, p. 14]

Alguno de los KPI's más relevantes son los siguientes:

1. Seguridad: Mantener a los miembros de la organización seguros en el área laboral
2. Calidad: Definir un estándar de funcionamiento de los productos hacia los clientes
3. Nivel de servicio: Mantener los tiempos de entrega de acuerdo con lo acordado
4. Productividad: Mantener los niveles de producción en un rango previamente establecido

Figura 2. Ejemplos de KPI's

4.3 Datos

En la actualidad, vivimos en un mundo que produce una gran cantidad de datos que son almacenados diariamente, pero ¿qué son los datos? Los datos son una recolección o conjunto de hechos, números, palabras, observaciones u otro tipo de información. Vivimos en una era de la información, en donde diferentes dispositivos generan estos datos masivamente a partir de negocios, de la sociedad, ciencia e ingeniería, medicina y por casi todos los aspectos de la vida cotidiana. Este crecimiento exponencial es resultado de la capacidad computacional que se tiene en los diferentes ámbitos, lo cual nos permite generar y almacenar estos datos utilizando las herramientas adecuadas. Estos datos actualmente nos permiten conocer más a profundidad aspectos relevantes dentro de los diferentes sectores, ya que conociendo más del tema podemos tomar decisiones informadas, tratando de obtener resultados esperados a partir de un análisis de estos, esto se puede lograr con herramientas computacionales y matemáticas para determinar desviaciones, promedios, distribuciones, etc. Que nos ayudarán a representar los datos en su conjunto y comprender los sucesos actuales y anteriores.

Sin embargo, para poder almacenar datos relevantes y posteriormente usarlos para su análisis, es necesario realizar algunos métodos de selección para que estos datos sean relevantes dentro de un sistema, no queremos almacenar información que únicamente ocupe memoria y con ello recursos computacionales, por ello es necesario realizar una serie de procesamiento que nos optimice la información para una situación en específico, para ello se tiene que seguir los siguientes pasos:

- **Limpieza de datos:** Remover ruido y datos inconsistentes.
- **Integración de datos:** Donde se puede combinar diferentes fuentes de datos.
- **Selección de datos:** Seleccionar datos relevantes para la tarea de análisis de datos.
- **Transformación de los datos:** Convertir datos en formatos apropiados.
- **Minería de datos:** Métodos inteligentes para extraer datos siguiendo ciertos patrones.
- **Evaluación de patrones:** Identificar patrones que representan conocimiento basado en medidas.
- **Presentación del conocimiento:** Técnicas usadas para la presentación del conocimiento a los usuarios.

Estos pasos preparan los datos para etapas siguientes, donde puede seguir otro procesamiento para objetivos más específicos y así generar más datos o que interactúen con el usuario. [3]

4.4 Análisis de datos

El campo de la estadística se encarga de manejar las colecciones de datos, presentarlos, analizarlos y usar estos datos para tomar decisiones, resolver problemas y diseñar productos y procesos, tales que nos permitan obtener resultados esperados, en términos más simples, la estadística es la ciencia de los datos. Esta ciencia es una herramienta en el campo del análisis de datos ya que nos permite usar los datos para obtener parámetros propios de la estadística, por nombrar algunos, promedio, valores totales, desviaciones, variabilidad, etc.

Debido a que muchos aspectos dentro de la ingeniería se relacionan con los datos, siempre conocer acerca de estadística resulta una habilidad tan importante como aquellas de la ciencia de la ingeniería. Específicamente, los métodos estadísticos pueden ser poderosas si es que las enfocamos al diseño de nuevos productos y sistemas, mejorando diseños ya existentes, e implementar nuevos procesos de producción.

Dentro del ámbito de la ingeniería, muchos de los problemas y decisiones se encuentran directamente ligados al análisis de datos, por lo que el conocimiento de estadística se vuelve una competencia esencial, al mismo nivel que las disciplinas tradicionales de la ciencia de la ingeniería. Se destacan que los métodos estadísticos ya que permiten

diseñar experimentos, mejorar procesos existentes, evaluar la calidad de productos y optimizar sistemas complejos, reduciendo la dependencia de suposiciones empíricas y favoreciendo decisiones basadas en evidencia cuantitativa.

Asimismo, la aplicación de la estadística en la ingeniería moderna no se limita únicamente al análisis descriptivo, sino que se extiende hacia enfoques predictivos y de control, tales como el control estadístico de procesos, el análisis de capacidad, la modelación probabilística y la mejora continua. Estas herramientas permiten anticipar comportamientos futuros, identificar causas raíz de variaciones no deseadas y proponer estrategias de mejora con un respaldo matemático sólido. [4]

4.5 Lenguajes de programación

Las computadoras se han aplicado en una infinidad de áreas, desde el control de plantas de energía nuclear hasta la provisión de videojuegos en teléfonos móviles. Debido a esta gran diversidad en el uso de las computadoras, se han desarrollado lenguajes de programación con objetivos muy diferentes. A continuación, se discutirán brevemente algunas de las áreas de aplicación de las computadoras y los lenguajes asociados a cada una de ellas. [5]

4.5.1 Aplicaciones científicas

Las primeras computadoras digitales, que aparecieron a fines de la década de 1940 y principios de la de 1950, fueron inventadas y utilizadas para aplicaciones científicas. Típicamente, las aplicaciones científicas de esa época usaban estructuras de datos relativamente simples, pero requerían una gran cantidad de cálculos aritméticos en punto flotante.

Las estructuras de datos más comunes eran los arreglos y matrices; las estructuras de control más empleadas eran los bucles contadores y las sentencias de selección.

Los primeros lenguajes de programación de alto nivel inventados para aplicaciones científicas fueron diseñados para satisfacer esas necesidades. Su competencia directa era el lenguaje ensamblador, por lo que la eficiencia era una preocupación primordial. El primer lenguaje para aplicaciones científicas fue Fortran. [5, p. 38]

- **Simulación de trayectorias balísticas**: Los primeros computadores como el ENIAC se usaron para calcular tablas balísticas para el ejército de EE. UU. Requerían millones de operaciones en punto flotante y usaban matrices y ciclos contadores como estructuras principales, ejemplo ilustrativo en *Figura 3* [9]

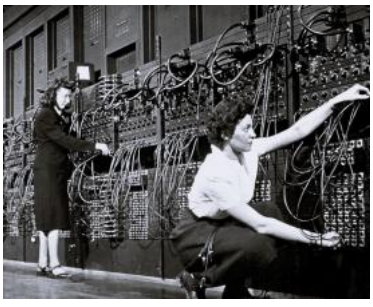

Figura 3. Fotografía de ENIAC [22]

- **Cálculo de órbitas para proyectos espaciales:** El lenguaje FORTRAN se empleó para modelar trayectorias orbitales en los programas Mercury, Gemini y Apollo. Estos modelos requerían intensivo cálculo matemático y uso de arreglos multidimensionales. [10]
- **Métodos numéricos y análisis estructural**: Se usaban lenguajes como FORTRAN para resolver ecuaciones diferenciales, análisis de vibraciones, simulaciones de dinámica de fluidos y métodos como elementos finitos, se muestra un ejemplo en la *Figura 4*. [11]

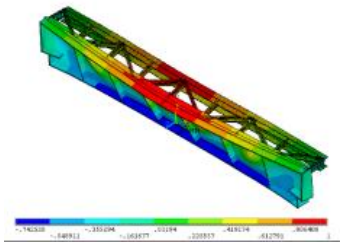

Figura 4. Ejemplo de análisis estructural [23]

4.5.2 Aplicaciones en los negocios

Los lenguajes empresariales se caracterizan por incluir herramientas para la generación de reportes detallados, métodos precisos para describir y almacenar números decimales y datos de caracteres, y la capacidad de especificar operaciones aritméticas decimales. [5, p38]

- **Sistemas bancarios:** Los primeros sistemas de procesamiento de transacciones usaban COBOL para registrar depósitos, retiros y balances con precisión decimal. [12]

4.5.3 Inteligencia artificial

La inteligencia artificial es un área amplia de aplicaciones informáticas caracterizada por el uso de cálculos simbólicos en lugar de numéricos. El cálculo simbólico implica manipular símbolos, formados por nombres más que por números, y suele realizarse de manera más conveniente con listas enlazadas en lugar de arreglos. [5, p39]

- **Sistemas expertos:** Lenguajes como LISP y Prolog se usaron para representar conocimiento mediante reglas, árboles y listas enlazadas. Ejemplo: MYCIN, un sistema experto para diagnóstico médico. [13]
- **Procesamiento del lenguaje natural:** LISP permitió implementar analizadores semánticos y sintácticos para investigación en comprensión del lenguaje humano. [14]

Figura 5. Logo de Prolog [24]

Figura 6. Logo de LISP [25]

4.5.4 Software para la web

La World Wide Web (Red Mundial) se sustenta en una colección diversa de lenguajes, que van desde los lenguajes de marcado, como HTML (que no es un lenguaje de programación), hasta lenguajes de propósito general como Java. [5, p39]

- **Sitios web estáticos con HTML y CSS:** HTML se utiliza para estructura, CSS para estilo; ninguno de los dos es un lenguaje de programación. Para dotar a los sitios web de interactividad, lógica de negocio y capacidad de procesamiento, es necesario integrar lenguajes de programación de propósito general. En el lado del cliente, lenguajes como JavaScript permiten responder a eventos del usuario, validar formularios y actualizar contenido sin recargar la página. En el lado del servidor, lenguajes como Java, PHP o Python posibilitan el acceso a bases de datos, el procesamiento de información y la generación dinámica de contenido. [17]

Figura 7. Logo de HTML y CSS [26]

4.5.5 Programación de sistemas

El sistema operativo y las herramientas de soporte de programación de un sistema informático se conocen colectivamente como software de sistemas. Este software se utiliza de manera casi continua, por lo que debe ser altamente eficiente. Además, debe contar con características de bajo nivel que permitan escribir las interfaces de software con los dispositivos externos. [5, p41]

Figura 8. Logo de C y C++ [27]

- **Creación de sistemas operativos:** El lenguaje **C** fue diseñado específicamente para escribir Unix. Permitía manipular memoria, punteros, registros y manejar dispositivos mediante interfaces de bajo nivel. [15]
- **Compiladores e intérpretes:** Herramientas como GCC y Clang están escritas en C y C++ debido a su eficiencia. [16]

Figura 9. Logo de GCC [28]

4.6 Scripts

Los scripts son programas generalmente pequeños y diseñados para automatizar tareas repetitivas, interactuar con el sistema operativo o enlazar componentes de software. A diferencia de los programas compilados, los scripts suelen ejecutarse mediante un intérprete, lo que permite una edición y prueba más rápida sin necesidad de recompilar. Esto los vuelve esenciales en áreas como administración de sistemas, análisis de datos, desarrollo web, automatización de procesos y pruebas de software.

La programación mediante scripts se caracteriza por su simplicidad sintáctica, la flexibilidad para manipular archivos y procesos del sistema, y la facilidad para integrar herramientas externas. Lenguajes como Python, Bash, PowerShell, Perl y JavaScript se consideran lenguajes de scripting debido a su capacidad para escribir rutinas breves que cumplen tareas muy específicas.

En el ámbito industrial, científico y empresarial, los scripts se utilizan para realizar operaciones como manipulación de datos, extracción de información, automatización de pipelines, ejecución de comandos del sistema operativo, procesamiento de logs y generación de reportes. [18]

4.7 Python para análisis de datos

Para muchas personas, Python es un lenguaje de programación sencillo y accesible. Desde su primera aparición en 1991, Python se ha vuelto uno de los lenguajes de programación interpretados más populares de los últimos años, este lenguaje es comúnmente llamado **scripting** ya que puede ser usado para escribir rápidamente pequeños programas o **scripts** para automatizar algunas otras tareas. En los últimos años Python ha desarrollado una gran comunidad destinada al cómputo científico y al análisis de datos.

Figura 10. Logo de Python [29]

Para el análisis de datos, la computación interactiva y la visualización de datos, inevitablemente se comparará a Python con otros lenguajes y herramientas de programación, tanto de código abierto como comerciales, ampliamente utilizados, como R, MATLAB, SAS, Stata y otros. En los últimos años, las bibliotecas de código abierto de Python han mejorado, lo que lo ha convertido en una opción popular para las tareas de análisis de datos. Combinado con la fortaleza general de Python para la ingeniería de software de propósito general, es una opción excelente como lenguaje principal para crear aplicaciones de datos.

Este lenguaje de programación cuenta con una gran cantidad de librerías dedicadas a diferentes ámbitos de la computación, sin embargo, el análisis de datos utiliza librerías como pandas y numpy que son muy conocidas por la comunidad. [6, p2]

4.7.1 NumPy

NumPy, cuyo nombre proviene de la abreviación de *Numerical Python*, se ha consolidado como una herramienta fundamental para el cómputo numérico dentro del ecosistema de Python. Esta biblioteca proporciona estructuras de datos eficientes, como arreglos multidimensionales, así como un amplio conjunto de funciones y algoritmos optimizados para realizar operaciones matemáticas y estadísticas de alto desempeño. Este posee:

Figura 11. Logo de NumPy [30]

- Un arreglo multidimensional que es rápido y eficiente.
- Funciones para realizar operaciones elemento a elemento con matrices u operaciones entre matrices.
- Herramientas para leer y escribir datos basados en matrices.
- Operaciones de algebra lineal, Transformada de Fourier y generación de números aleatorios.

Más allá de las capacidades de procesamiento rápido de arreglos que NumPy añade a Python, uno de sus usos principales en el análisis de datos es servir como contenedor de datos que se pasan entre algoritmos y bibliotecas.

Para datos numéricos, los arreglos de NumPy son altamente eficientes para almacenar y manipular datos. Además, las bibliotecas escritas en un lenguaje de más bajo nivel, como C o FORTRAN, pueden operar sobre los datos almacenados en un arreglo de NumPy sin copiar los datos a otra representación en memoria. Por lo tanto, muchas herramientas de cómputo numérico para Python suponen a los arreglos de NumPy como estructura de datos principal o, en su defecto, apuntan a la interoperabilidad con NumPy. [6, p. 4]

4.7.2 Pandas

Figura 12. Logo de Pandas [31]

Pandas proporciona estructuras de datos de alto nivel y funciones diseñadas para que trabajar con datos estructurados o tabulares sea intuitivo y flexible. Desde su aparición en 2010, ha ayudado a que Python sea un entorno de análisis de datos potente y productivo. Los objetos principales de Pandas que se utilizarán en esta librería son el DataFrame, una estructura tabular orientada a columnas con etiquetas tanto de filas como de columnas, y la Series, un arreglo unidimensional con etiquetas.

Pandas combina las ideas de cómputo con arreglos de NumPy con las capacidades de manipulación de datos que se encuentran en las hojas de cálculo y en las bases de datos relacionales (como SQL). Ofrece una indexación conveniente que permite

reformatear, segmentar, realizar agregaciones y seleccionar subconjuntos de datos. Dado que la manipulación, preparación y limpieza de datos son habilidades tan importantes en el análisis de datos, pandas es uno de los enfoques principales en esta área. [6, p. 5]

4.8 JupyterLab

JupyterLab es una aplicación web que ofrece todo un entorno de trabajo interactivo ideal para trabajar en el ámbito científico. Su utilidad más impactante es la posibilidad de crear Jupyter Notebooks, que combinan diversos elementos como código interactivo, textos, ecuaciones, imágenes y otras fuentes de datos que se pueden manejar en proyectos científicos de todo tipo.

JupyterLab es extremadamente potente y flexible, ya que permite usar una increíble cantidad de fuentes desde texto, imágenes, hojas de cálculo, HTML, PDF, LaTeX y mucho más. Paralelamente permite trabajar con código escrito en cantidad de lenguajes populares como Python, R o Scala, editando y ejecutando el código sin salirnos de la propia aplicación. Además, es capaz de usar diversos tipos de consolas de comandos y almacenar scripts para su ejecución.

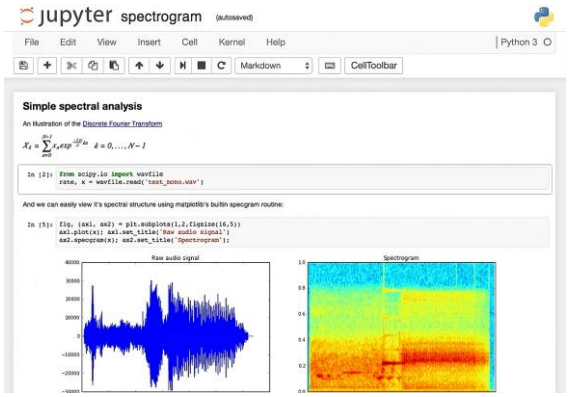

Figura 13. Ejemplo de reporte en JupyterLab [32]

Con todo, es una herramienta excelente para integrarla en diversos flujos de trabajo de data-scicence, ya que es capaz de ajustarse a las necesidades más diversas de los profesionales. Además, integra toda una serie de herramientas modernas que permiten incrementar la colaboración entre los profesionales, publicar y revisar código entre personas de todo el mundo, en flujos como los que nos proporcionan GitHub o GitLab. [7]

4.9 Tableau

Tableau es una aplicación de Inteligencia Empresarial y visualización de datos, fue diseñado para realizar análisis de datos de manera accesible e intuitiva para todo tipo de usuarios de diferentes niveles. Al ser tan sencilla, el individuo y organizaciones tienen la capacidad de transformar los datos en dashboards que son interactivas y que pueden ser compartidas para su difusión, esto resulta relevante ya que nos permite mantener informado a un grupo de persona acerca de la información más reciente, favoreciendo de esta manera la toma de decisiones.

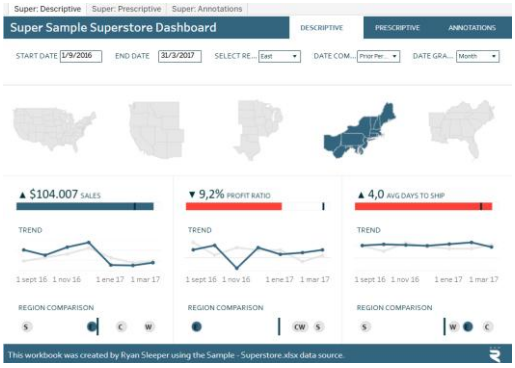

Figura 14. Ejemplo de dashboard en Tableau [21]

A diferencia de otras herramientas BI (Inteligencia Empresarial en sus siglas en español) que requieren de personas calificadas para manejar tal tipo de conocimiento técnico, Tableau prioriza la facilidad y comodidad, lo cual permite que aquellas ambas personas, técnicas y no técnicas, puedan desarrollan y crear visualizaciones complejas en el análisis de manera dinámica y rápida. Tableau soporta un gran rango de bases de datos para alimentar sus dashboards, desde speadsheets y bases de datos hasta servicios en la nube, asegurando flexibilidad y conectividad. [8]

5. Mis responsabilidades

El rol fundamental de un ingeniero dentro de una organización consiste en optimizar procesos, aumentar la confiabilidad de las operaciones y promover la escalabilidad de las soluciones tecnológicas. Para cumplir con dichos objetivos, es necesario diseñar métodos y sistemas que resuelvan problemas específicos, reduzcan la carga operativa y entreguen información clara para la toma de decisiones tanto a nivel interno como hacia el cliente.

Bajo esta premisa, mis actividades dentro del corporativo estuvieron orientadas al desarrollo de scripts que facilitaran el análisis de los KPI's permitiendo conocer el estado operativo del equipo de trabajo y de la empresa. La generación de estas herramientas contribuyó directamente a mejorar los tiempos de ejecución y la robustez dentro del

análisis de datos, así como a respaldar decisiones estratégicas que impactan en el rendimiento y la calidad de los productos o servicios.

Como parte de mis responsabilidades, se me asignaron las siguientes tareas:

- Analizar e interpretar datos provenientes de diversas áreas para obtener indicadores clave de desempeño de ingeniería.
- Realizar comparaciones temporales de los KPI's para identificar tendencias, comportamientos atípicos o áreas de mejora.
- Diseñar y codificar sistemas que automatizaran la obtención, limpieza, filtrado y normalización de datos utilizados para generar KPI's.
- Validar y verificar la información procesada por los sistemas desarrollados, asegurando su integridad antes de su implementación en diferentes etapas del proceso de ingeniería.
- Elaborar gráficas, tablas y diagramas basados en datos depurados, con el fin de disponer de herramientas visuales organizadas y enfocadas en los indicadores más relevantes.
- Crear manuales, guías y documentación técnica que permitieran a otros ingenieros comprender, utilizar y dar mantenimiento a los scripts.

6. Definición de la problemática

La empresa en la que realicé mis actividades profesionales, al ser una organización multinacional dedicada al desarrollo de soluciones tecnológicas enfocadas en la gestión de la energía y automatización industrial, como son el desarrollo de tableros de distribución de la energía y tableros de control de procesos industriales; donde el capital humano y los procesos operativos se distribuyen entre diversas áreas, la comunicación efectiva y el flujo oportuno de información son elementos esenciales para garantizar la continuidad y mejora de los procesos. Cada departamento debe ser capaz de reportar su desempeño, medir la eficiencia de sus actividades y proporcionar información confiable que respalde la toma de decisiones estratégicas.

En este contexto, los KPI's representan una herramienta fundamental, ya que permiten cuantificar el comportamiento de los procesos, evaluar el cumplimiento de objetivos y detectar desviaciones que requieren acciones correctivas. No obstante, la utilidad de los KPI's depende directamente de la calidad, consistencia y oportunidad de los datos a partir de los cuales son generados.

Durante el desarrollo de mis actividades como integrante del equipo NEST, se identificaron tres problemáticas principales, las cuales fueron definidas a través de sesiones continuas usando metodología Agile con diferentes miembros del equipo

NEST; que afectaban tanto la eficiencia del análisis de datos, como la confiabilidad y la visualización clara de los KPI's. Estas problemáticas son:

- Deficiencias en la obtención y procesamiento de datos.
- Tiempos excesivos en la generación de indicadores derivados del manejo manual de la información.
- Pocos dashboards para la visualización de KPI's.

6.1 Deficiencia en la obtención y procesamiento de datos

La organización, debido a su carácter internacional y al amplio número de colaboradores que la conforman; genera grandes volúmenes de información provenientes de áreas como producción, calidad, facturación e ingeniería. Para que esta información pueda ser utilizada en la generación de KPI's, resulta indispensable someterla a procesos sistemáticos y repetitivos de obtención, limpieza, filtrado, normalización y selección de parámetros relevantes.

En mi estancia como becario, alrededor de un mes analizando los procesos dentro de la empresa relacionada al flujo de datos y procesos, pude resaltar que las actividades eran realizadas de **manera** predominantemente **manual** por el personal encargado del análisis de datos en ingeniería y por los miembros del equipo de NEST, quienes no eran especialistas en análisis de datos sino ingenieros del producto, por ello, dichas actividades de análisis de datos se veían sujetas a la disponibilidad en tiempos y esfuerzo de los miembros involucrados. A causa de un especialista directamente encargado del análisis de datos para el equipo de NEST, se empleaban herramientas convencionales y sencillas de usar, como lo son las hojas de cálculo. Esta forma de trabajo carecía de mecanismos que ayudaran con a la estandarización de criterios y procedimientos recurrentes, lo que provocaba una **alta dependencia de la intervención del operad**or y una **variabilidad** significativa **en los resultados obtenidos**.

La ejecución manual de cada etapa **incrementaba** considerablemente **la susceptibilidad a errores**, tales como omisiones, inconsistencias en los criterios de filtrado o diferencias en la selección de variables. **Como consecuencia, los KPI's** generados **presentaban problemas de confiabilidad y reproducibilidad**, dificultando la comparación entre distintos periodos y reduciendo la certeza de que los indicadores reflejaran el estado real de los procesos evaluados.

Adicionalmente, la carga operativa asociada a estas tareas superaba la capacidad del personal disponible, lo que limitaba la realización de revisiones cruzadas o procesos de validación sistemática. Esta falta de verificación contribuía a consolidar un esquema de análisis poco robusto, en el cual los KPI's dependían en gran medida del esfuerzo manual y del criterio individual del analista.

6.2 Tiempos excesivos en la generación de indicadores

Otro factor crítico identificado fue el tiempo requerido para la generación de los KPI's. En el área de análisis de datos en la que fui asignado para realizar mis actividades, la dinámica operativa exigía contar con indicadores constantemente actualizados, recurrentemente cada mes o en juntas directivas, ya que estos eran utilizados para evaluar el desempeño del equipo y el estado de los procesos en periodos cortos de tiempo.

Sin embargo, al no contar con sistemas automatizados, **cada actualización** de información **implicaba repetir de manera manual todas las etapas del procesamiento de datos**. Este enfoque **hacía que la generación de KPI's fuera lenta y poco flexible**, limitando la frecuencia de actualización y **provocando que los indicadores disponibles no siempre correspondieran al estado más reciente de la operación**.

Como consecuencia de los procesos adoptados por la empresa, la generación de indicadores se veía afectado limitando el seguimiento oportuno de los KPI's, la visualización de su comportamiento y tendencias mediante dashboards y la toma de decisiones basadas en información actualizada se veían retrasados. Esta situación limitaba la capacidad del equipo para reaccionar de manera ágil e informada, ante posibles desviaciones en el desempeño, reduciendo además el valor práctico de los indicadores como herramientas para la gestión, el monitoreo y el control de los procesos.

6.3 Visualización de información y KPI's

Entre los principales problemas identificados durante el desarrollo de las actividades realizadas en el área fue la **limitada disponibilidad de herramientas de visualización de información**, particularmente dashboards (panel de visualización de datos) que permitieran concentrar, interpretar y dar seguimiento a indicadores clave de desempeño (KPI's) de forma clara y accesible.

La escasez de dashboards provocaba que los KPI's relevantes, tales como tiempos de operación, eficiencia de procesos, incidencias, uso de recursos o cumplimiento de objetivos, **no fueran monitoreados de manera continua, sino de forma reactiva y en intervalos irregulares**, mostrando información muy escasa en cuanto a las necesidades de la organización, limitando la capacidad de los responsables para detectar desviaciones, identificar áreas de mejora y tomar decisiones informadas en tiempo real.

Adicionalmente, **la ausencia de visualizaciones gráficas claras y específicas para cada KPI (gráficas, tablas dinámicas, entre otros)** generaba una dependencia excesiva de aquellos ya creados, los cuales no siempre contenían los casos particulares para cada KPI, que facilitara una comprensión rápida del estado general del sistema o del desempeño de los procesos evaluados. Como consecuencia, se incrementaba el riesgo de errores en la interpretación de la información y se reducía la eficiencia en la toma de decisiones.

7. Propuesta de solución

Una vez que visualicé las problemáticas existentes dentro de los procesos previamente establecidos en la organización, las cuales estaban relacionadas principalmente con la realización de procedimientos manuales, el tiempo de ejecución y la limitada visualización de indicadores clave de desempeño, fue posible identificar áreas específicas de oportunidad para la mejora de dichos procesos. Estas problemáticas, descritas en el apartado anterior, evidenciaron la necesidad de replantear la forma en que se gestionaba, procesaba y presentaba la información.

La definición de la solución la llevé a cabo, junto con mi supervisora a cargo, quien me ayudó a entender el impacto de las problemáticas dentro de la organización y la definición de posibles soluciones que se alinearan a los objetivos y recursos disponibles, a partir de un análisis estructurado de las problemáticas detectadas, evaluando sus causas, impacto y recurrencia dentro de la operación diaria. A partir de este análisis, establecí como **criterio principal la optimización de los procesos repetitivos y automatizables, con el objetivo de reducir la carga operativa manual, minimizar errores humanos y mejorar la eficiencia general del sistema.**

Para ello, fue necesario realizar una revisión de las tecnologías disponibles que permitieran automatizar y agilizar dichos procesos, considerando herramientas que facilitaran la integración de datos, el procesamiento automático de la información y su visualización mediante dashboards e indicadores clave (KPI's). Este proceso de **evaluación tecnológica se enfocó en seleccionar soluciones que fueran compatibles con la infraestructura existente, escalables y de fácil adopción por parte de los usuarios finales.**

En la tabla siguiente (Tabla 1) encontraremos una comparación de las tecnologías disponibles, denotando ventajas y limitaciones de cada una de ellas para nuestros objetivos.

Herramienta	Tipo de herramienta	Capacidades de análisis de datos	Capacidades de visualización	Integración de datos	Ventajas	Limitaciones
Tableau	Plataforma de visualización y BI	Permite análisis exploratorio	Dashboards interactivos	Conexión con múltiples bases de datos	Alta calidad visual. Implementación en empresa	No conocía la herramienta
Microsoft Excel	Hoja de cálculo	Análisis mediante fórmulas y funciones estadísticas	Gráficas, tablas dinámicas y reportes básicos	Compatible con diversas fuentes de datos	Amplio uso en entornos empresariales	Limitado en grandes volúmenes de datos y funciones avanzadas
Power BI	Plataforma de inteligencia de negocios	Transformación y modelado de datos	Dashboards interactivos	Alta integración con Excel, bases de datos y servicios cloud	Alta capacidad de visualización	No se contaba con la licencia para usar el software
Python	Lenguaje de programación para análisis de datos	Manipulación, limpieza y análisis avanzado de grandes volúmenes de datos	Visualización mediante librerías adicionales	Integración con bases de datos	Gran flexibilidad, automatización y librerías en análisis de datos	Conocimiento muy especializado para generar dashboards
SQL	Lenguaje de consulta para bases de datos	Extracción, filtrado y agregación eficiente de grandes volúmenes de datos	Visualización limitada	Alta integración con bases de datos relacionales	Alta eficiencia en manejo de datos	No está orientado directamente a visualización de datos

Tabla 1. Tabla comparativa de tecnologías para el análisis y visualización de datos

Como resultado de este análisis, definí una solución orientada a la automatización de tareas y la implementación de mecanismos de visualización clara y accesible, permitiendo un seguimiento continuo del desempeño de los procesos. Tomando en cuenta las ventajas ofrecidas y los recursos disponibles dentro de la empresa se decidió utilizar Python para el análisis de datos, ya que muestra ventajas en el área de automatización y procesamiento de datos que resaltan sobre las demás tecnologías, y Tableau para la visualización de datos, el cual permite crear dashboards interactivos, además que ya estaba implementado dentro de la empresa y se tenía la licencia; esta solución propuesta responde directamente a las problemáticas identificadas, ya que las necesidades requerían un sistema flexible para la automatización del análisis de datos y al mismo tiempo la visualización con software especializado de los datos procesados. Es importante mencionar que la base de datos usada para Tableau, era en Excel, pero el procesamiento era realizado con Python.

7.1 Scripts para la automatización

Los scripts de automatización constituyeron el núcleo técnico de la solución desarrollada. Dichos sistemas los diseñé para realizar de manera autónoma diversas etapas del procesamiento de datos, entre ellas:

- **Obtención de datos** desde las fuentes originales, ya fuera mediante archivos, repositorios internos o bases de datos proporcionadas por las áreas correspondientes.
- **Limpieza y normalización**, eliminando errores, duplicidades y valores atípicos, y estandarizando formatos para su posterior análisis.
- **Filtrado y selección** de variables relevantes, con el fin de obtener únicamente la información necesaria para generar los KPI's definidos por el equipo NEST.
- **Generación automática de archivos** procesados, organizados y listos para ser consumidos por motores de visualización.

La implementación de estos scripts me permitió reducir significativamente el tiempo de ejecución, garantizar la consistencia metodológica y minimizar errores que anteriormente surgían por la manipulación manual de grandes volúmenes de información.

Además, cada script fue acompañado de documentación técnica detallada, donde se explicaba el proceso de uso, la estructura de los datos necesarios, los parámetros configurables y los flujos de trabajo recomendados. Esto aseguró la reproducibilidad del sistema, así como su mantenimiento y mejora futura por parte de otros ingenieros.

7.2 Elaboración de informes visuales

Como complemento a la automatización del procesamiento de datos, desarrollé dashboards interactivos utilizando Tableau para la representación gráfica de información. El propósito de estos tableros fue convertir los datos ya procesados en herramientas visuales que facilitaran:

- La interpretación inmediata de tendencias, comportamientos y variaciones relevantes en el desempeño del equipo y de los procesos monitoreados.
- La comparación temporal de KPI's, permitiendo evaluar mejoras, detectar desviaciones y respaldar decisiones basadas en evidencia.
- La presentación clara y estructurada de la información, adaptada a los distintos niveles de interés dentro del corporativo.

Los dashboards incorporaron elementos como gráficas de líneas, barras, tablas comparativas, histogramas, diagramas de distribución y métricas clave, organizados de forma que permitieran un análisis intuitivo y eficiente. Asimismo, aseguré mecanismos de actualización para que, al ejecutarse los scripts de automatización, los paneles reflejaran la información más reciente sin intervención manual adicional.

En conjunto, los **dashboards se convirtieron en herramientas estratégicas para la toma de decisiones**, al proporcionar una visión general y analítica del rendimiento operativo del equipo NEST y de los procesos evaluados.

8. Metodología del proyecto

La metodología aplicada se estructuró a partir de un enfoque iterativo y cíclico, en el cual las tareas no se desarrollaron de manera aislada, sino de forma complementaria y continua.

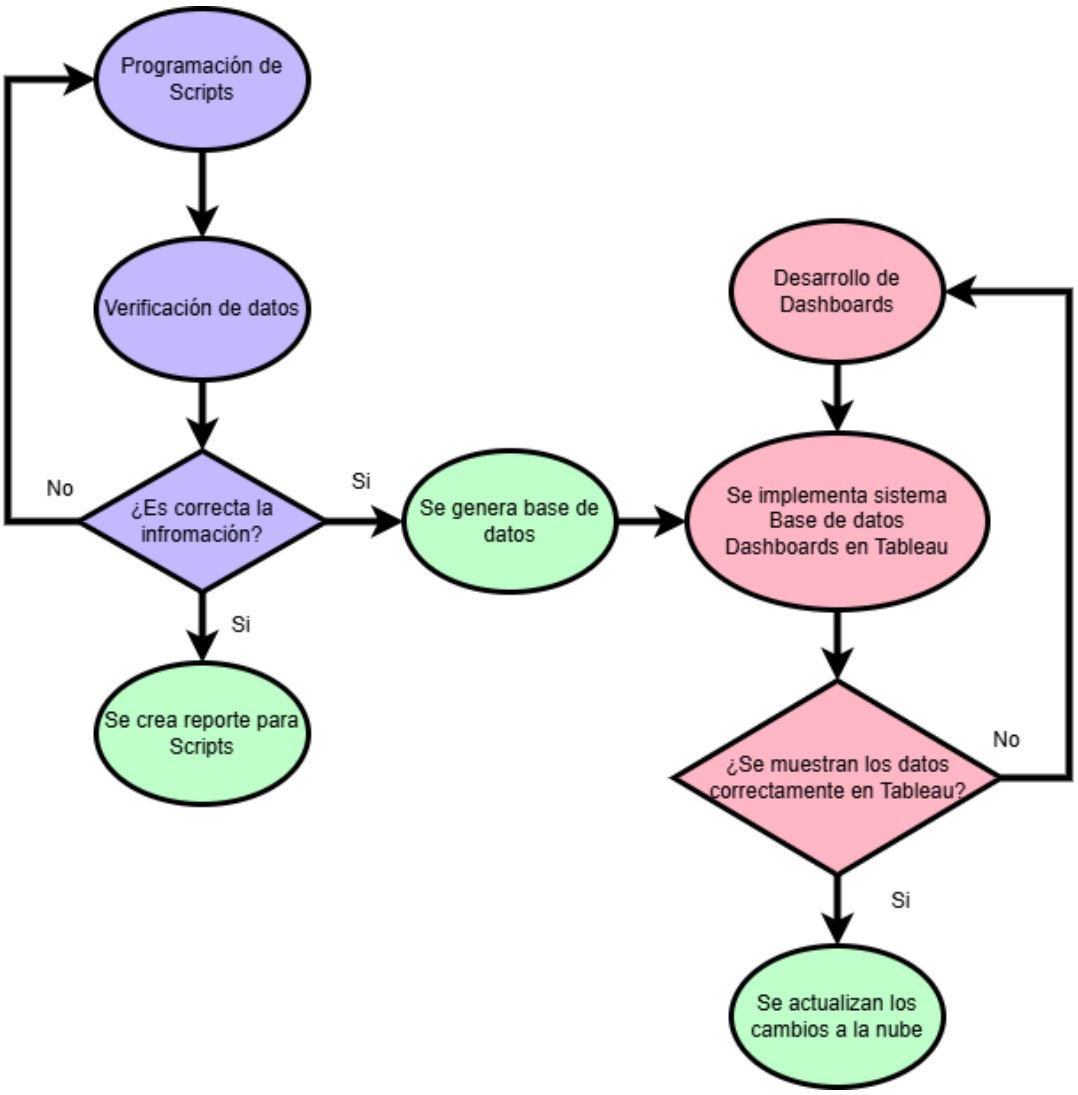

Figura 15. Flujo del desarrollo y verificación para la generación de scripts de automatización y dashboards

Este enfoque permitió que, mientras se diseñaban nuevas soluciones, se evaluara simultáneamente su funcionamiento, brindando soporte, corrigiendo posibles errores en el procesamiento de datos y optimizando su desempeño. De esta manera, la metodología adoptada facilitó una mejora progresiva del sistema, asegurando la confiabilidad de la información y la correcta operación de las soluciones implementadas. Dichas actividades pueden agruparse en dos grandes conjuntos: por un lado, la **programación e implementación de scripts orientados a la automatización de procesos (*Figura 15*) y la generación de dashboards,** incluyendo **el manual para la correcta ejecución de los scripts**, y por otro, la **validación, prueba y mejora continua de los procesos ya desarrollados,** a partir de la retroalimentación continua de los miembros del equipo NEST.

8.1 Desarrollo

La etapa de desarrollo, la cual involucra la programación de Scripts para la automatización y la creación de dashboards para KPI's, se estructuró en distintas etapas, las cuales se describen a continuación y fueron necesarias para comprender en su totalidad las necesidades y el alcance del proyecto.

8.1.1 Análisis inicial del proceso y levantamiento de información.

Como primera etapa, llevé a cabo un análisis del flujo de trabajo existente para la obtención y análisis de los KPI's. En esta fase identifiqué que el proceso se realizaba de forma manual, involucrando múltiples pasos repetitivos como la recopilación de datos, limpieza, filtrado y organización de la información. Esta etapa fue la más importante, ya que me dio la pauta que me permitió identificar aquellos datos relevantes para la generación de KPI's y los pasos a seguir para poder obtener todas las bases de datos necesarias las cuales se encontraban en diferentes fuentes y direcciones.

En conjunto con mi supervisora y el equipo NEST, definí los indicadores clave de desempeño que resultaban relevantes para evaluar las actividades del área, y que eran parte fundamental de la empresa para poder medir el desempeño de las tareas realizadas por los colaboradores y su efecto en los diferentes campos de interés, como lo es facturación, tiempos de entrega, errores de ingeniería, entre otros. A partir de esta etapa se obtuvieron las necesidades y se definieron los siguientes indicadores:

- **Facturación y generación**, éstos nos indicaban cuales de los proyectos producían un valor a la empresa y aquellos los cuales no generaban valor pero que debían de tener un costo de ingeniería.
- **Nivel de servicio**, nos mostraba los tiempos de ejecución y de entrega de los proyectos de acuerdo con lo establecido por los lideres de los proyectos.
- **Calidad**, este parámetro nos permite conocer la relación de errores o cambios de ingeniería con respecto al número de proyectos
- **Productividad**, esta métrica nos permite conocer el desempeño y ocupabilidad del recurso humano.

Una vez obtenidos los parámetros a cuantificar y aquellos necesarios para un análisis más detallado de KPI's de NEST, pude establecer los requerimientos técnicos de los sistemas que desarrollé posteriormente y comenzar con la etapa de programación de scripts.

8.1.2 Estrategia de automatización.

La estrategia basada en la automatización del procesamiento de datos. Consiste en una búsqueda de las herramientas más completas y sencillas para el análisis de datos. Esta herramienta debía de tener las siguientes características para que fuera viable:

- Aceptar formatos con extensión .xlsx (Archivos de Excel).
- Mínima intervención humana para su ejecución.
- Alto grado de automatización.
- Herramientas o librerías para dedicados al análisis de datos.
- Fácil implementación y desarrollo.
- Escalable.

Tecnología	Aceptar formato de Excel como base de datos	Mínima intervención humana	Alto grado de automatización	Herramienta o librerías para el análisis de datos	Fácil implementación y desarrollo	Escalable
Excel	Sí	No	Limitado	Parcial	Sí	No
Python	Sí	Sí	Sí	Sí	Parcial	Sí
SQL	Sí	Sí	Sí	Parcial	Parcial	Sí

Tabla 2. Cuadro comparativo sobre herramientas de análisis de datos

Una vez realizado este análisis y la búsqueda de herramientas, en conjunto con el personal de análisis de datos, pude llegar a la concusión de continuar el desarrollo de scripts con Python, ya que este nos brinda muchas herramientas especializadas para el análisis de datos, la velocidad de aprendizaje de este lenguaje de programación es relativamente rápida y que esta herramienta ya estaba implementada dentro de la organización para la automatización para otros equipos de trabajo, de esta manera, opté por usar este herramienta principalmente para la ejecución de los scripts de automatización.

Una vez definida la herramienta, se tenía que comenzar a conocer y explorar las capacidades y alcances de Python, por ello comencé a investigar aquellas librerías que me pudieran servir para desarrollar actividades de análisis de datos, así pude identificar aquellas fuentes que me servirían dados los requerimientos de diseño del sistema, se llegaron a las siguientes librerías:

- **Pandas**
- **NumPy**

En estas librerías pude encontrar funciones dedicadas al análisis de datos que son fundamentales para el desarrollo de los scripts, el manejo de grandes cantidades de datos, procesamiento eficiente de información estructurada, la realización de cálculos necesarios a los datos y, adicionalmente, existe una gran comunidad de científicos de datos y expertos que dan soporte a esta herramienta.

En esta etapa, dedicada a la estrategia de automatización, gran parte fue destinada al aprendizaje de las herramientas y del entorno Jupyter, ya que era necesario crear reportes donde se pudiera tener en un sistema el código y las instrucciones detalladas de la ejecución de estos Scripts.

8.1.3 Desarrollo e implementación de scripts de automatización.

Tan pronto como definí las herramientas tecnológicas y los requerimientos técnicos necesarios para la obtención de los KPI's, comencé formalmente con el desarrollo de los scripts orientados a la automatización del análisis de datos dentro del equipo de NEST. Esta etapa representó la fase operativa del proyecto, en la cual transformé los lineamientos estratégicos previamente definidos en soluciones técnicas funcionales.

Durante todo el proceso de desarrollo, mis actividades como analista de datos incluyeron la extracción, limpieza, transformación y estructuración de información proveniente de diferentes bases de datos en formato Excel, así como la validación de consistencia de los registros y la estandarización de criterios de medición. Cada script fue diseñado bajo una lógica modular, permitiendo su actualización y mantenimiento.

Para la programación de estos scripts, pude identificar un proceso básico que me permitió abordar todos los problemas y encontrar una solución a cada caso en específico, este proceso sigue el siguiente diagrama, el cual forma parte del diseño de cada script y engloba aquellos pasos fundamentales en la generación de bases de datos utilizables para la generación de reportes:

Figura 16. Diagrama de flujo del proceso

A partir de esta secuencia de diseño, pude crear diferentes soluciones que daban como resultado un script que me permitiera obtener bases de datos con la información necesaria para la visualización de KPI's.

Dentro de este proceso de desarrollo, pude crear los siguientes códigos que me permitieron obtener los KPI's definidos inicialmente. Los scripts son los siguientes:

• **Facturación**

Este script se encarga de procesar y filtrar los datos de facturación correspondientes al equipo NEST, considerando ambas etapas de ingeniería. Para ello, identifica y organiza la información asociada tanto a los ingenieros mecánicos como a los ingenieros eléctricos, permitiendo obtener una base de datos estructurada y lista para su análisis posterior. El reporte que incluye el código detallado se encuentra en el Apéndice 11.1

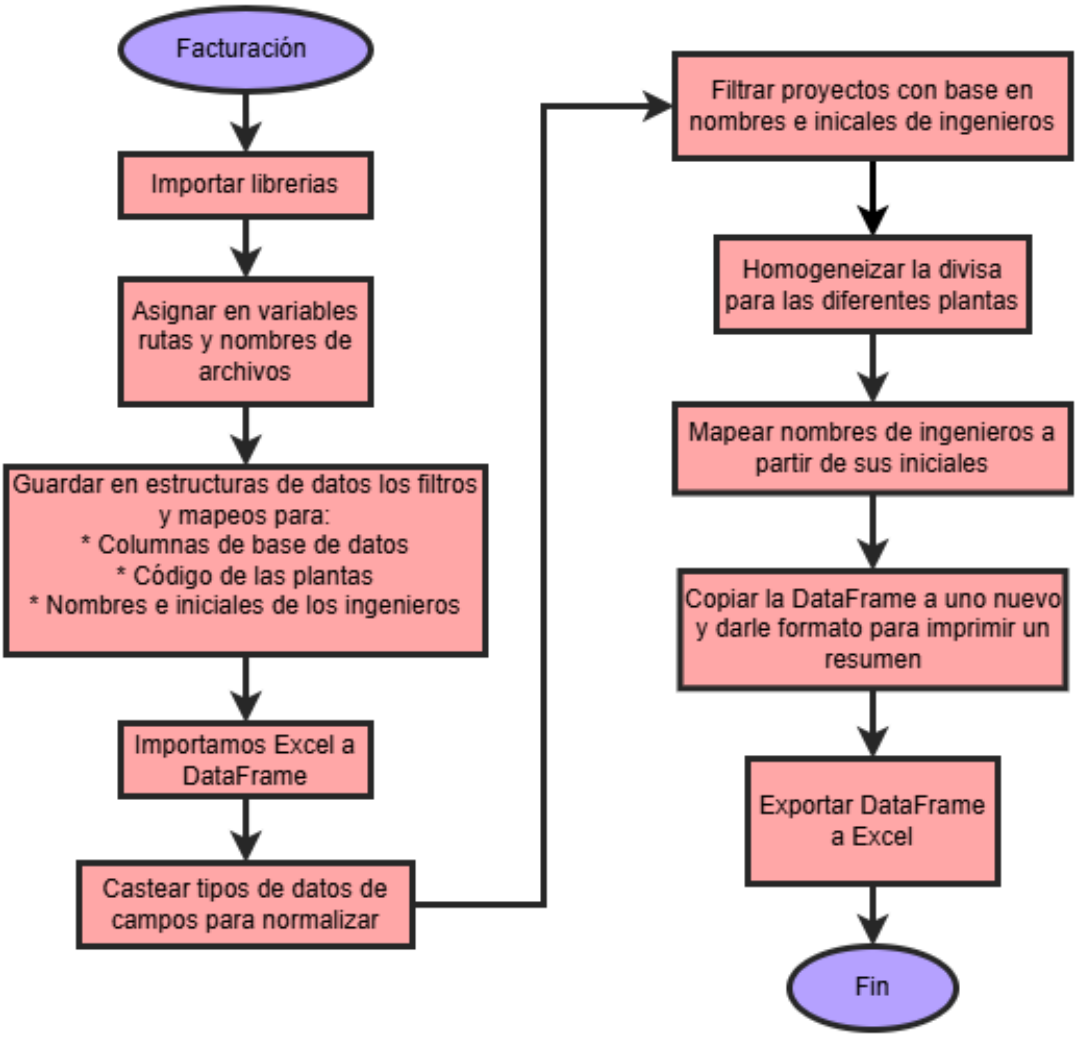

Figura 17. Diagrama de flujo de programación de script de facturación

• **OCN (Order Change Notification)**

Este script filtra y clasifica los registros de OCN en función del ingeniero responsable y la etapa de ingeniería correspondiente. Los OCN representan cambios o correcciones dentro del proceso de ingeniería, por lo que el script también incorpora la descripción del cambio solicitado, para facilitar su análisis y seguimiento por parte del equipo. El reporte que incluye el código detallado se encuentra en el Apéndice 11.2

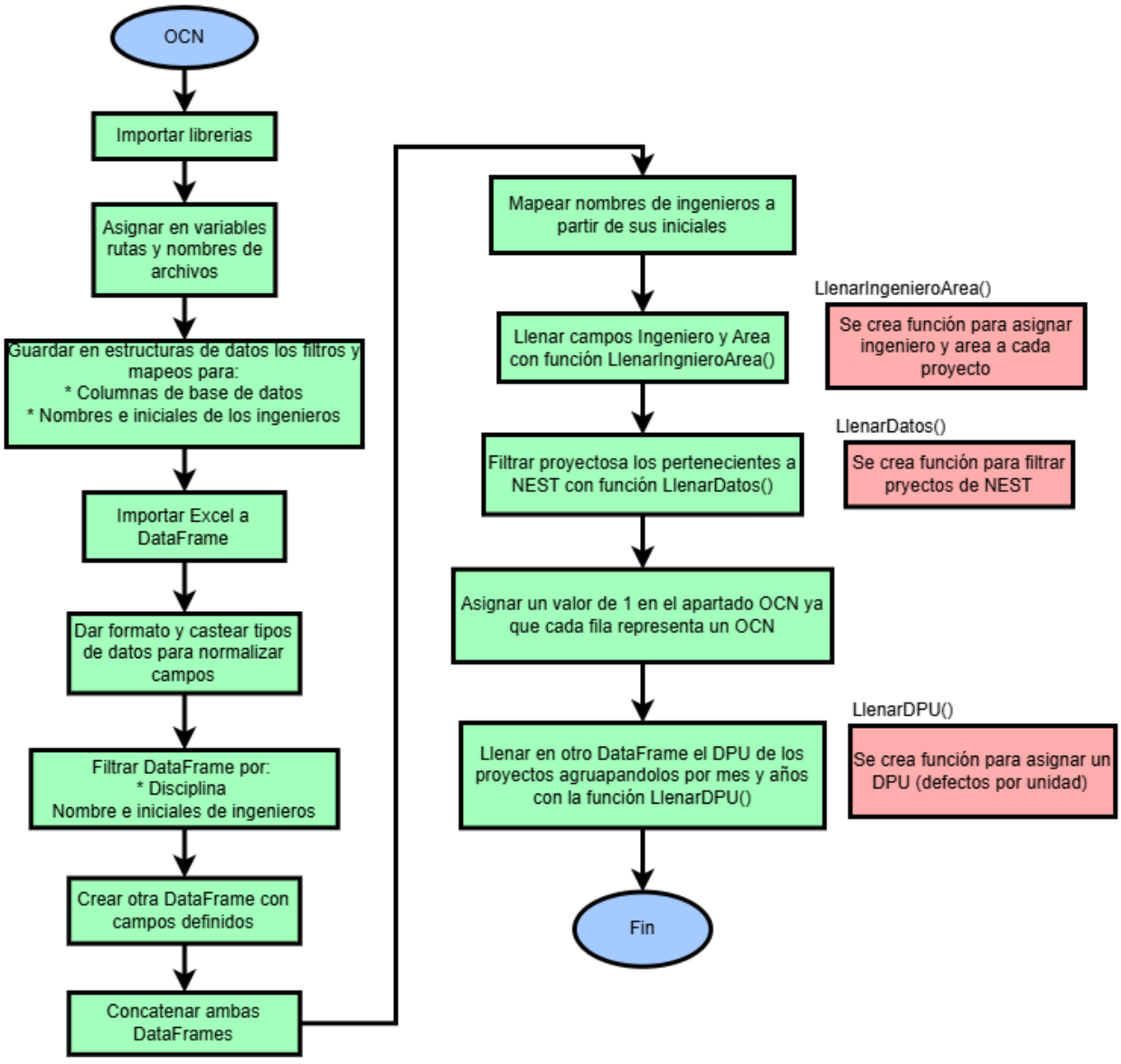

Figura 18. Diagrama de flujo de programación de script de OCN

• Nivel de servicio para los ingenieros mecánicos

Este script permite evaluar el desempeño en las etapas de ingeniería mecánica mediante el análisis de los tiempos de entrega reales en comparación con los tiempos previamente acordados. A partir de esta comparación, se calcula un indicador de cumplimiento o convergencia que permite determinar el grado en el que se respetan los plazos establecidos, facilitando así el monitoreo del nivel de servicio y la identificación de posibles desviaciones en el proceso. El reporte que incluye el código detallado se encuentra en el Apéndice 11.3

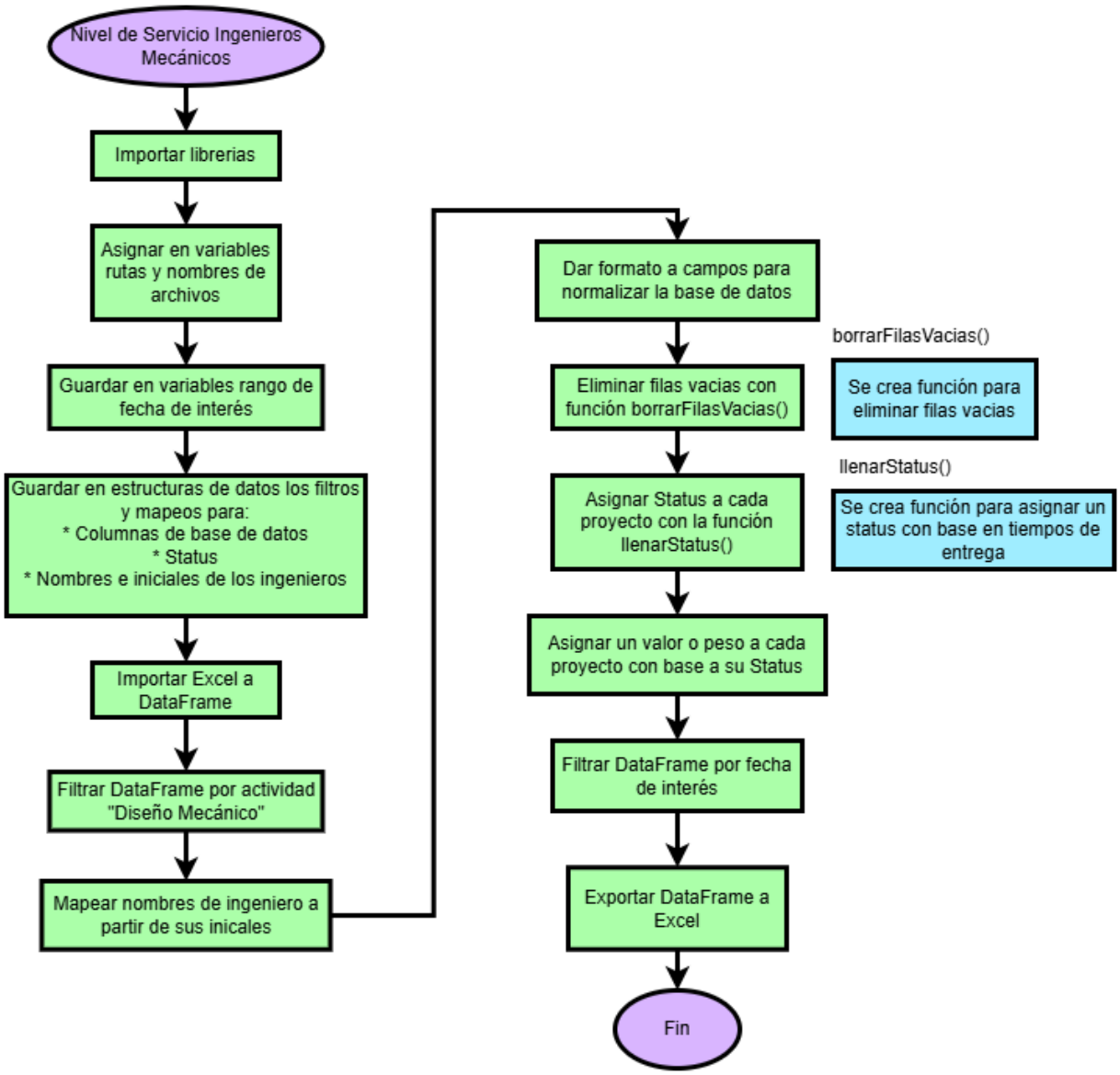

Figura 19. Diagrama de flujo de programación de script de nivel de servicio para los ingenieros mecánicos

• Nivel de servicio para los ingenieros eléctricos

Este script permite evaluar el desempeño en las etapas de ingeniería eléctrica mediante el análisis de los tiempos de entrega reales en comparación con los tiempos previamente acordados. A partir de esta comparación, se calcula un indicador de cumplimiento o convergencia que permite determinar el grado en el que se respetan los plazos establecidos, facilitando así el monitoreo del nivel de servicio y la identificación de posibles desviaciones en el proceso. El reporte que incluye el código detallado se encuentra en el Apéndice 11.4

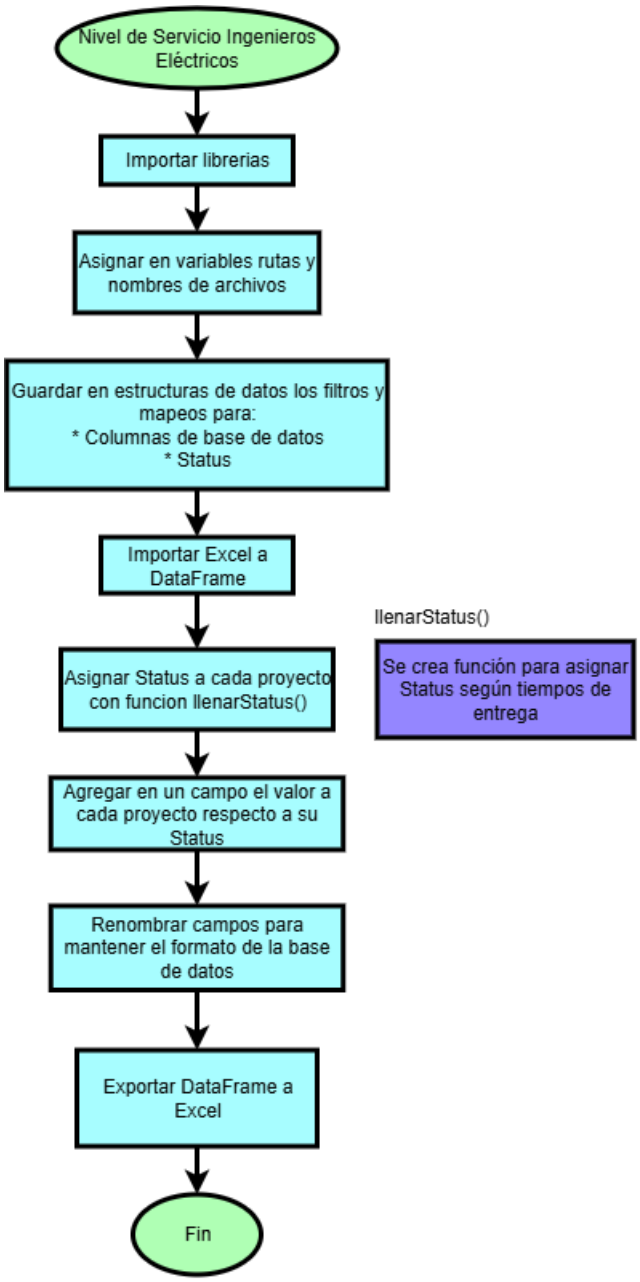

Figura 20. Diagrama de flujo de programación de script de nivel de servicio para los ingenieros

• Generado para ingenieros mecánicos

Este script procesa la base de datos de proyectos para identificar y desglosar las distintas etapas de ingeniería mecánica asociadas a cada uno. A partir de esta información, asigna el valor correspondiente al trabajo de ingeniería realizado en cada etapa. El reporte que incluye el código detallado se encuentra en el Apéndice 11.5

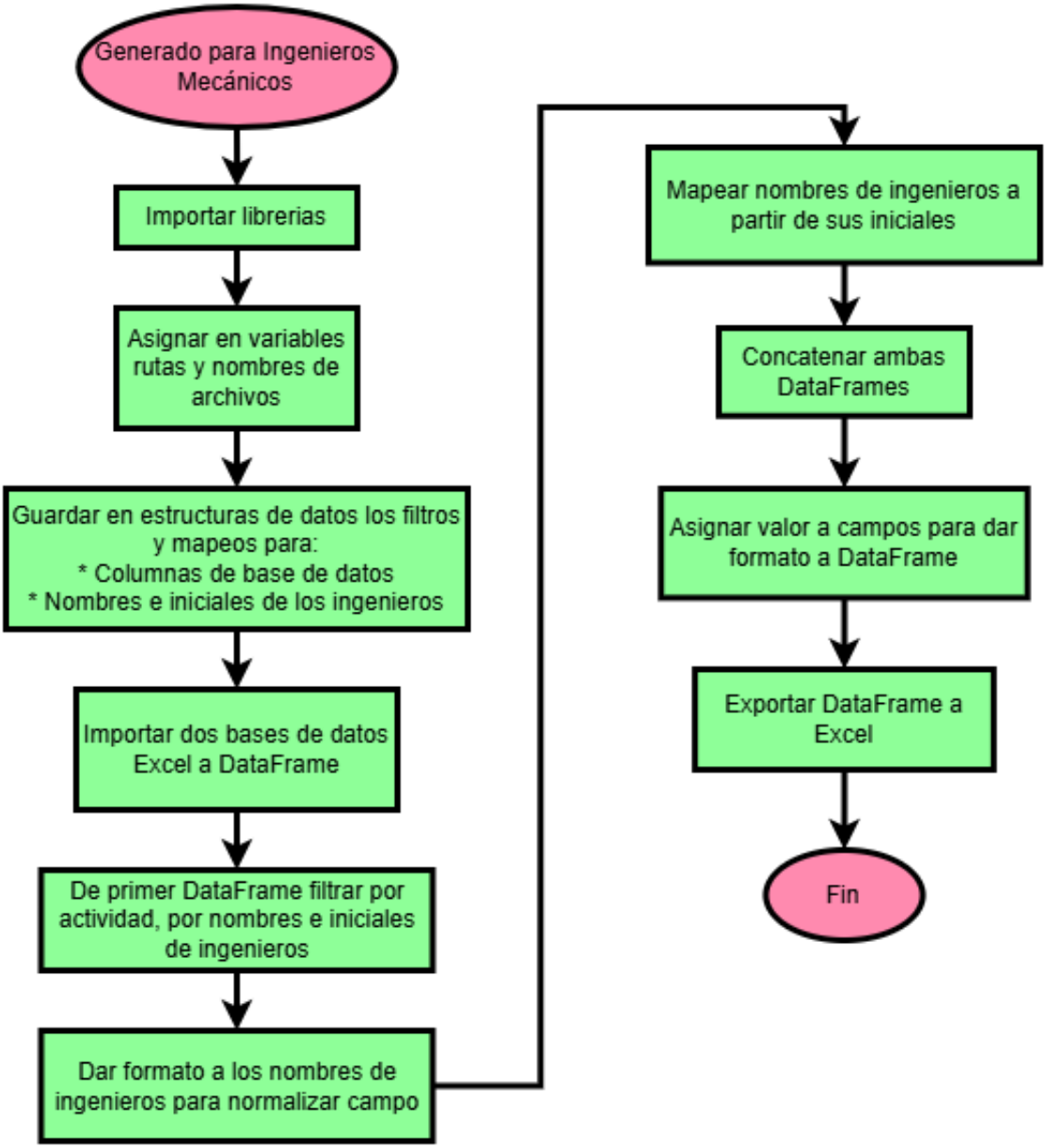

Figura 21. Diagrama de flujo de programación de script de generado para ingenieros mecánicos

• Generado para la planta 1

Este script integra y procesa dos bases de datos distintas con el objetivo de identificar y consolidar las etapas de ingeniería correspondientes al equipo NEST de ingenieros eléctricos dentro de los proyectos asociados a la planta 1. A partir de esta integración, se filtran las etapas relevantes y se calcula el valor del trabajo de ingeniería generado en cada una de ellas. El reporte que incluye el código detallado se encuentra en el Apéndice 11.6

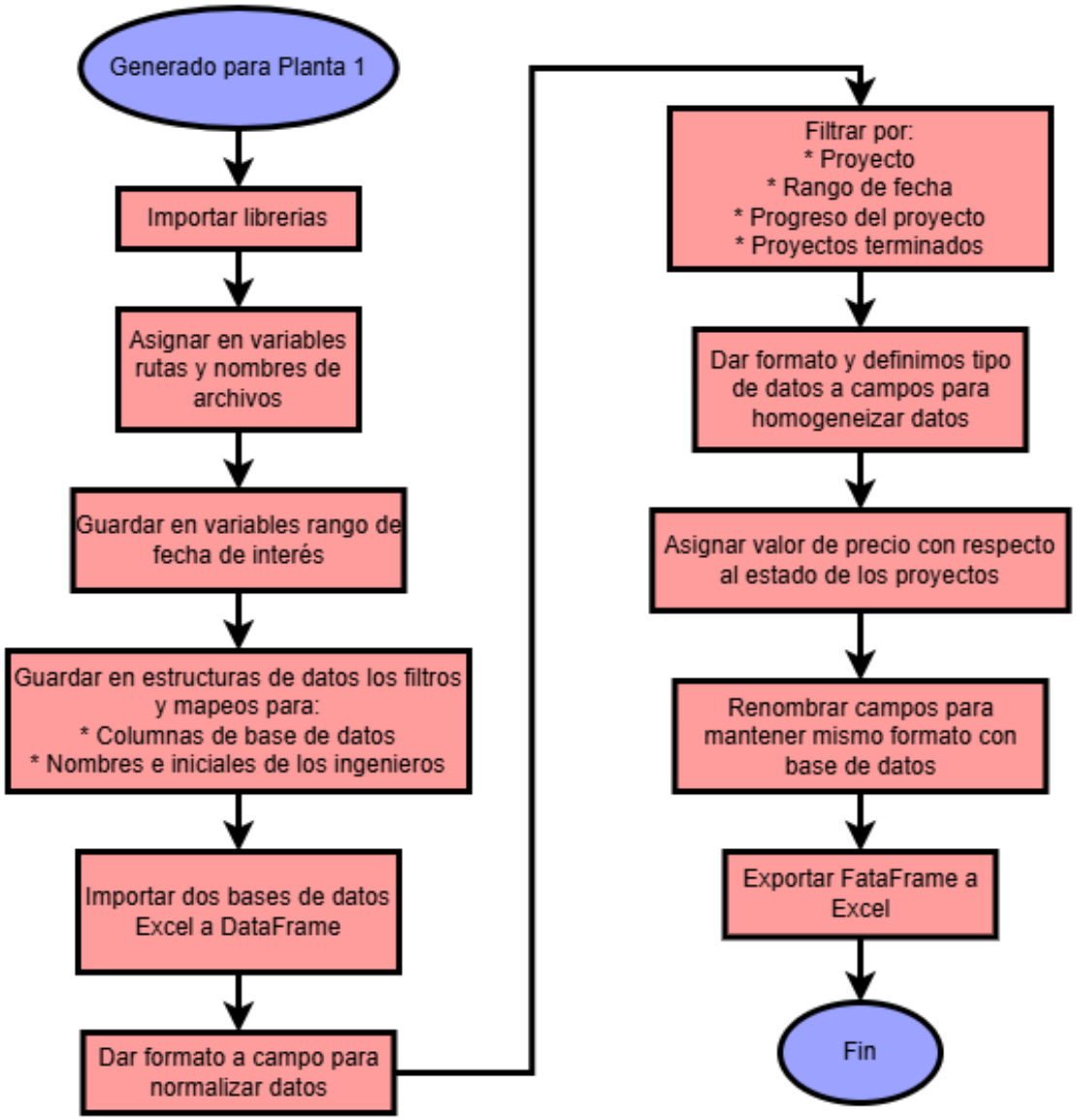

Figura 22. Diagrama de flujo de programación de script de generado para la planta 1

• Generado para la planta 2

Este script procesa la base de datos de los proyectos correspondientes a la planta 2 con el objetivo de identificar y filtrar las etapas de ingeniería asociadas al equipo NEST. A partir de este análisis, se calcula el valor del trabajo de ingeniería generado por cada ingeniero en las distintas etapas del proyecto. El reporte que incluye el código detallado se encuentra en el Apéndice 11.7

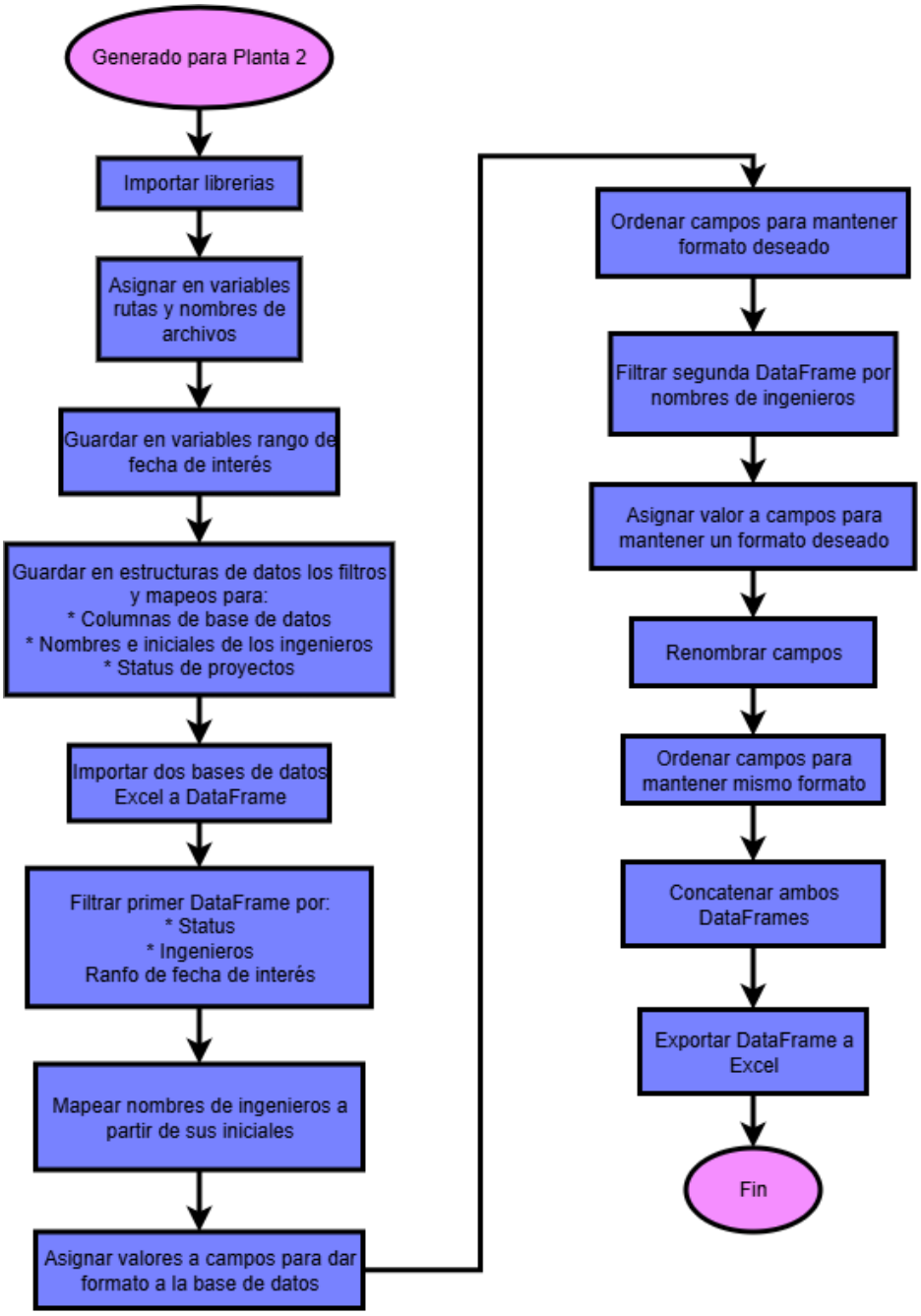

Figura 23. Diagrama de flujo de programación de script de generado para la planta 2

• Generación para la planta 3

No existe script asociado a esta planta ya que no fue necesario durante el desarrollo, un ingeniero a cargo proporcionaba la base de datos completamente filtrada para los ingenieros de esta planta, en el Apéndice correspondiente solo se observará un reporte el cual muestra los pasos a seguir para integrar la base de datos a la nube y dar formato. El reporte que incluye el código detallado se encuentra en el Apéndice 11.8

Descripción de Scripts

Cada uno de estos scripts fue diseñado con un objetivo específico alineado a los indicadores estratégicos del área.

Dentro de este proceso, en todos los casos era necesario importar y exportar una base de datos, para poder trabajar con dichos datos en Python, este proceso se realizaba de la siguiente manera, donde era necesario definir una ruta y nombres de archivos de las bases de datos fuente y el resultado. A continuación, se describen etapas repetitivas o básicas para el funcionamiento de los Scripts.

Variables de rutas y archivos

```
#Declaramos en variables las cadenas de caracteres para las rutas donde se extraeran y guardaran las bases de datos
rootdir = "RutaParaArchivoFuente"
destinationDir = "RutaParaArchivoDestino"
rootOneDrive ="RutaDeArchivoEnLaNube"
```

Una vez teníamos definidas las rutas, lo que hacíamos era usar una función de Pandas para poder importar bases de datos a DataFrames, se hizo de la siguiente manera

Importar archivo de Excel

```
#Exportamos datos de un archivo de excel
manualOrdersData = pd.read_excel(manual_orders_fullpath,sheet_name="Main data",
engine="openpyxl")
```

Este proceso es muy general y me permite englobar la manera en que importamos los datos a Python en la mayoría de los casos, los ejemplos completos se encuentran en el Apéndice. De igual manera, el proceso de exportación de DataFrame a Excel se repite en todos los scripts, que son los procesos iniciales y finales que se muestran en la *Figura 16*.

Exportar a archivo de Excel

```
#Exporta DataFrame a Excel
finalData.to_excel(destination_fullpath_xls,index=False)
```

Exportamos a partir de un archivo de Excel a nuestro entorno para su procesamiento

Para cada caso particular, los pasos intermedios; Normalización de Datos, Procesamiento de Datos y la Generación de Datos Limpios y Relevantes son diferentes y suelen cambiar dependiendo del tratamiento necesario para cada base de datos y las necesidades de cada Script, sin embargo, englobare los procesos más comunes para el procesamiento de información con Python.

Normalizar datos

```
#Cambiamos tipo de dato de parametros
manualOrdersData["sects"] = manualOrdersData["sects"].astype(str)
manualOrdersData["itm_unit_list"] = manualOrdersData["itm_unit_list"].astype(str)
manualOrdersData["pelt"] = manualOrdersData["pelt"].astype(str)
```

Parte fundamental del procesamiento es la normalización, ya que de esta manera aseguramos que los datos dentro de un apartado sean congruentes, por ello, es necesario definir tipos de datos a las columnas.

Guardar en estructuras de datos los filtros

```
#Arreglo que guarda nombres e iniciales de ingenieros de NEST
nameEng = ['Nombres de Ingenieros de NEST']
initialEng = ['Iniciales de Ingenieros de Nest']

#Arreglo que guarda nombres e iniciales de ingenieros mecanicos de NEST
mechEng = ["Nombres de ingenieros mecanicos de NEST"]
mechInitials = ["Iniciales de ingenieros mecanicos de NEST"]
```

Posteriormente, al obtener bases de datos que engloban diferentes personas de NEST y fuera de éste, es necesario guardar en arreglos y librerías los nombres e iniciales de los ingenieros de NEST

Filtrar

```
#Filtramos los datos por nombre e inicial de los ingenieros
finalData = finalData.query("appl_engr == @initialEng | appl_engr_nm == @nameEng | mech_engr ==
@initialEng | mech_engr_nm == @nameEng")
finalData["mech_engr_nm"] = finalData.apply(lambda row : None if row["mech_engr_nm"] not in
nameEng else row["mech_engr_nm"], axis=1)
finalData["appl_engr_nm"] = finalData.apply(lambda row : None if row["appl_engr_nm"] not in
nameEng else row["appl_engr_nm"], axis=1)
finalData = finalData.drop_duplicates()
```

Filtrar los datos a aquellos de interés para el equipo NEST a partir de los filtros declarados anteriormente.

Librería y mapeo de nombres

```
#Libreria para mapear las iniciales con los nombres de cada ingeniero
mapNames ={
"Libreria que mapea nombres de ingenieros a traves de sus iniciales"
}

#Sustituye nombres en el apartado correcto y elimina las columnas auxiliares, asegura tipo de dato de columna
finalData["name_2"] = finalData["appl_engr"].map(mapNames)
finalData["appl_engr_nm"] = finalData["appl_engr_nm"].fillna(finalData["name_2"])
finalData.drop("name_2", axis=1, inplace=True)
```

Además, como parte de la normalización, también de procesaban los nombres de los ingenieros para poder estandarizar y mapear a un solo elemento por ingeniero, por ello se generaba una librería para poder mapearlos

Una vez que tenemos filtrados y normalizados los datos y campos de interés, podemos seguir con el procesamiento de información, si es necesario, ya que en muchos de los casos únicamente era indispensable filtrar los datos para obtener una base de datos funcional, esto dependía del tipo de información que debíamos de extraer de las bases de datos fuente. Si desea conocer el proceso detallado puede consultar el Apéndice.

8.1.4 Generación de dashboards para el análisis de KPI's

Cuando desarrollé los scripts para el procesamiento de datos, se procedió a la creación de dashboards utilizando Tableau, software ya implementado para la presentación de datos dentro de la organización, los cuales empleaban las bases de datos generadas como fuente de información. Estos dashboards permitieron representar gráficamente los datos y calcular distintos parámetros estadísticos, con el objetivo de presentar la información de forma clara y comprensible para los usuarios. Asimismo, fueron diseñados para ser interactivos y dinámicos, permitiendo a los usuarios explorar la información desde diferentes perspectivas. De esta manera, los tableros facilitaron la visualización estructurada de los datos, así como la interpretación de tendencias y la realización de comparaciones temporales.

Los dashboards incorporaron diferentes tipos de gráficas y elementos visuales, resaltando principalmente información referente a los KPI's propuestos, como:

- Gráficas de desempeño a lo largo del tiempo.
- Comparaciones entre periodos o áreas específicas.
- Tablas dinámicas con métricas clave.
- Indicadores visuales para identificar desviaciones relevantes.

También permitían filtrar la información de acuerdo con diferentes parámetros, los cuales eran clave para mostrar datos específicos de interés, estos filtros eran:

- Fecha
- Ingeniero
- Proyecto
- Rol (Ingeniero mecánico o eléctrico)
- Nivel de servicio
- Planta

Algunos de estos filtros fueron aplicados de manera general a todos los dashboards, mientras que otros se implementaron de forma específica en determinados tableros, de acuerdo con los requerimientos, alcance y la funcionalidad particular de cada uno. Además, integré la actualización de los dashboards directamente con los scripts de automatización, permitiendo que la información se refrescara de manera rápida y sin intervención manual, esto permitió generar dashboards actualizados para la disponibilidad de mi supervisora y líderes del área.

Como se ilustra en el diagrama de la *Figura 15*, el proceso de desarrollo de los diferentes dashboards implicó un trabajo cíclico, donde se tenía un flujo de actividades con retroalimentación constante, es decir, desarrollaba un prototipo y a partir de sesiones de revisión se definían o aclaraban nuevos parámetros para completar los dashboards a las necesidades del equipo de ingeniería.

A lo largo de mi estancia como becario, desarrolle los siguientes dashboards:
- Nivel de servicio
- Facturado de ingenieros mecánicos y eléctricos
- Generado de ingenieros mecánicos y eléctricos
- OCN

Con ellos, pude englobar los KPI's que permitían hacer análisis directos acerca del desempeño de las actividades de los ingenieros de NEST.

Nota. Debido a motivos de confidencialidad de datos de la empresa no se colocó una imagen real de los dashboards.

Facturado y Generado

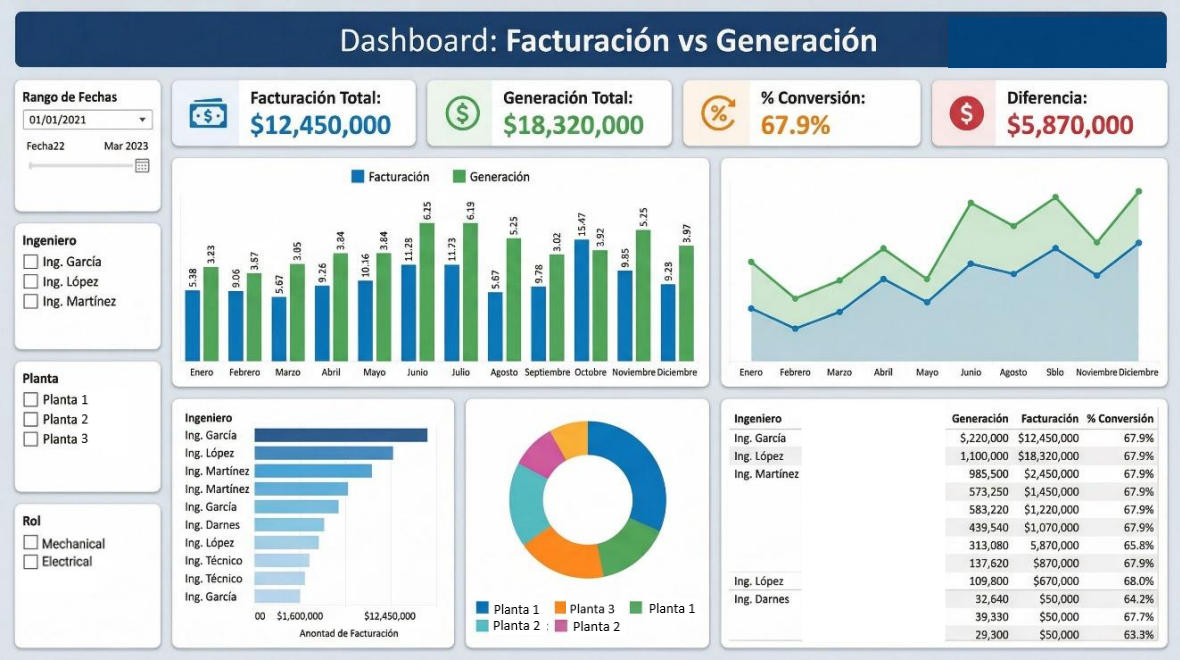

Figura 24. Ejemplo de dashboard para facturado y generado

Nivel de servicio

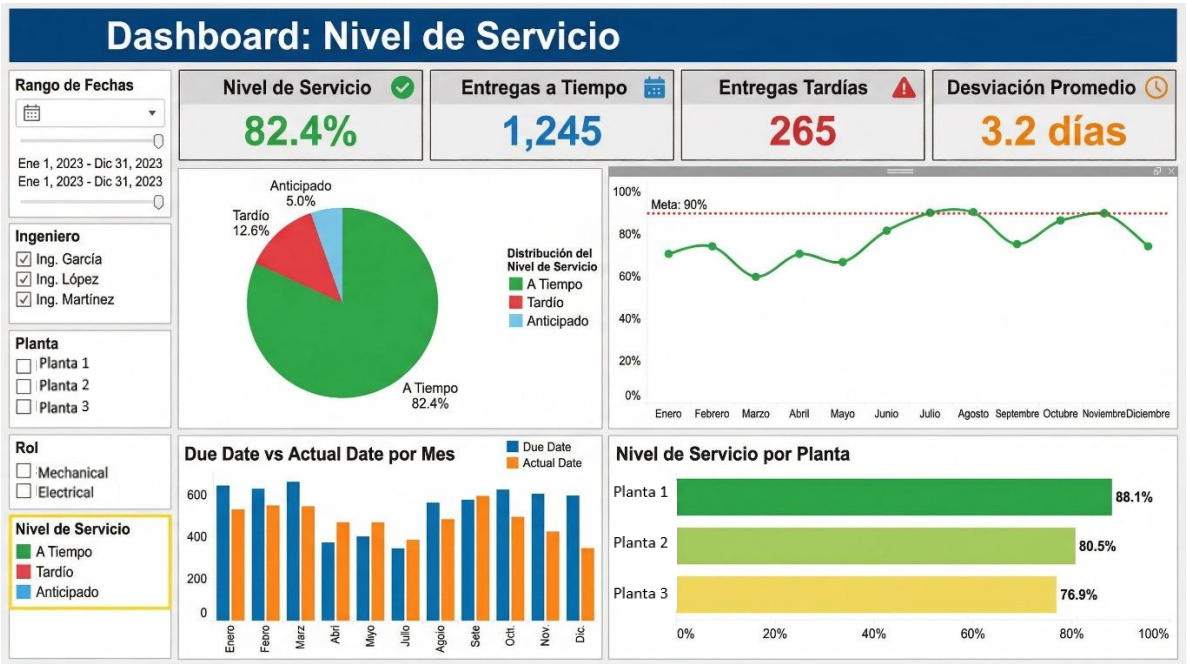

Figura 25. Ejemplo de dashboard para nivel de servicio

OCN

Figura 26. Ejemplo de dashboard para OCN

8.1.5 Manuales y estandarización del proceso.

Como parte complementaria al desarrollo técnico, elaboré documentación detallada para cada uno de los sistemas implementados. Esta documentación incluyó instrucciones de uso para poder acceder a las diferentes bases de datos, descripción de la estructura de datos y directorios, parámetros configurables y recomendaciones para su mantenimiento.

De esta manera, plasmé los procesos, nombres de archivos y rutas a directorios necesarios para poder utilizar completamente los Scripts que elaboré, esto sirvió como guía para mí mismo e incluso para alguien más que no estuviera familiarizado con el proceso; de esta manera cualquiera que tuviera acceso a esta documentación, fuera capaz de generar las bases de datos y seguir actualizando los dashboards. Además, los scripts y manuales fueron organizados y almacenados en un repositorio remoto, asegurando su disponibilidad, control de versiones y reutilización futura por parte del equipo de ingeniería.

Algunas de las instrucciones que componen los reportes pueden verse en la siguiente lista:

1. Pasos para acceder y descargar a la base de datos fuente
2. Nombre con el cual se debía guardar la base de datos fuente
3. Ruta donde se debía almacenar dicha base de datos
4. En caso necesario, indicaciones de edición de código para su correcta ejecución
5. Nombre con el cual se debía guardar la base de datos generado

6. Ruta donde se debía almacenar la base de datos generado
7. Indicaciones adicionales para poder ejecutar los códigos tomando consideraciones excepcionales.

A continuación, se presenta un ejemplo ilustrativo para comprender la descripción de los procesos.

Generación de reporte de lo facturado del NEST

Para poder actualizar la data relacionado a lo facturado, este al ser actualizado mes con mes, por esta razón se debe de actualizar la carpeta donde se encuentran todos los extractos de reportes, primero describiré el proceso de obtención de reportes, este es decrito de la siguiente manera:

1. Pasos para poder acceder y descargar la base de datos en específico a traves del software utilizado por la empresa

De esta manera podemos obtener, para cada una de las plantas, el reporte de lo facturado, solo queda descargarlo en formato de excel y guardarlo en la carpeta **Ruta completa de la carpeta donde guardaremos la base de datos** junto con los demás extractos, es recomendable guardar cada uno de los archivos con el siguiente formato **Shipped (Planta) (Mes) (Año)**, de esta manera podemos trackear de mejor manera cada uno de los reportes. Adicionalmente, a cada uno de los nuevos reportes descargados, se deberá renombrar la columna "*gid_act*" como "*act_ship_dt*". Ya que se tengan los tres archivos (Planta1, Planta2 y Planta3) correctamente guardados, ya es posible ejecutar el script completo

//Códiogo de ejecución//

Ya que se generó el archivo de excel con todos los datos de Shipped, ahora en necesario sustituir todos los datos del archivos de **Nombre de archivo en la nube.xlsx** localizado en la carpeta *OneDrive* **Ruta completa del directorio en one drive** con los nuevos que se encuentran en el archivo nombrado **Nombre de archivo generado.xlsx** que se encuentra en la carpeta **Ruta completa al archivo generado**, previamente verificar los datos generados en la tabla anterior donde se muestra un resumen por planta y mes.

Este reporte forma parte del **Apéndice 11.1**, se usó como ejemplo para poder visualizar la estructura general que se siguió en la creación de estos manuales. Para mayor detalle puede consultar la sección de **Apéndice**

8.2 Resultados y aportaciones

En esta sección se presentan los principales resultados obtenidos a partir del desarrollo de las actividades descritas previamente, así como las aportaciones realizadas durante la estancia en el área.

8.2.1 Indicadores de rendimiento

Una de las aportaciones más relevantes de mi trabajo fue el desarrollo de métodos ágiles y automatizados para la obtención de indicadores clave de desempeño (KPI's). A partir de esta necesidad, implementé soluciones basadas en programación enfocada al análisis de datos, mediante las cuales diseñé sistemas capaces de recibir bases de datos en bruto y generar conjuntos de datos depurados que contenían únicamente la información relevante para el cálculo de los indicadores definidos.

Estos sistemas automatizados **facilitaron el análisis del rendimiento tanto a nivel local (dentro de la misma planta) como a nivel regional e internacional, permitiendo comparar el desempeño de los equipos ubicados en distintos estados de la República Mexicana e incluso con plantas situadas en otros países**. Esta capacidad de comparación aportó una visión más amplia y objetiva del estado operativo del equipo NEST.

Los principales indicadores de rendimiento implementados y monitoreados a partir de las necesidades del equipo de NEST, fueron:

- **Nivel de servicio**, para evaluar el cumplimiento de tiempos, atención y respuesta hacia los clientes.
- **Productividad**, para medir la rentabilidad del trabajo realizado. Se mide a través de la facturación y valor del trabajo (generado) de los ingenieros.
- **Calidad**, orientada a identificar desviaciones, retrabajos o incidencias dentro de los procesos atendidos.

Estos indicadores se convirtieron en el pilar principal del análisis, ya que la totalidad de los datos procesados y filtrados giraban en torno a su correcta medición y seguimiento, permitiendo evaluar de manera objetiva el desempeño del equipo y del servicio ofrecido.

Con la implementación de Scripts para la evaluación de KPI´s, nos permitimos conocer aspectos clave en el desarrollo de las actividades de los ingenieros, tanto mecánicos como eléctricos, y nos brindaba una manera objetiva y cuantificable de medir el desempeño del equipo NEST bajo parámetros internacionales. Es decir, se logró sustentar con datos estructurados y verificables la aportación del equipo NEST a la empresa, fortaleciendo la toma de decisiones basada en evidencia.

Mediante el uso de esta herramienta, fue posible incrementar la visibilidad del equipo NEST dentro de la empresa, así como respaldar con evidencia la confiabilidad y rentabilidad del trabajo realizado por el equipo de ingeniería. Asimismo, contribuyó a

fortalecer una cultura orientada a resultados y a la mejora continua de la calidad en los procesos.

8.2.2 Visualización dinámica

Los tableros de control (dashboards) permitieron visualizar de forma inmediata el comportamiento de los KPI's mediante gráficas, tablas comparativas y métricas clave, facilitando la identificación de tendencias, áreas de oportunidad y desviaciones en el desempeño. La disponibilidad de información visual y actualizada contribuyó significativamente a agilizar los procesos, de horas a tan solo minutos, de análisis y toma de decisiones, tanto por parte de los líderes como de los ingenieros involucrados.

Gracias a la integración entre los scripts de automatización y los dashboards, fue posible contar con información actualizada de forma continua, eliminando la necesidad de reprocesar datos manualmente y asegurando la consistencia de los resultados mostrados.

De esta manera se aseguró una reducción del **tiempo de ejecución** donde el proceso era **desarrollado durante horas (de 8 a 12 horas)** con las herramientas disponibles en hojas de cálculos, **y con gran susceptibilidad a errores**, comunes en los procesos **manuales y repetitivos, a tan solo fracciones de hora (alrededor de 30 minutos)**, donde el proceso era verificado y validado constantemente; **y existía gran confiabilidad** en los resultados mostrados, como consecuencia, se obtuvo una **reducción del más del 90% en tiempos de ejecución y validación**. Los errores más comunes que se tenían eran:

- Existían nombres duplicados
- Tipo de divisa
- Perdida de información
- Datos duplicados
- No se contaba con filtros estandarizados

De manera que a través de las automatizaciones se reducían errores y se tenía una confiabilidad muy alta en los dashboards y su información.

8.2.3 Documentación

Como parte fundamental del proceso de desarrollo e implementación de las soluciones, elaboré documentación técnica detallada con el objetivo de garantizar la reproducibilidad, mantenimiento y correcta utilización de los sistemas desarrollados.

La creación de manuales de uso, en los cuales se describieron de manera clara los pasos necesarios para ejecutar los programas, la estructura de las bases de datos requeridas, los parámetros configurables y las consideraciones técnicas relevantes.

Asimismo, documenté el procedimiento para la correcta integración de las bases de datos procesadas en las plataformas de visualización (Tableau), permitiendo que los

dashboards fueran visibles y actualizados correctamente, de esta manera, se facilitó que esta información pudiera ser escalada, completada y actualizada por el ingeniero de datos, conservar el trabajo realizado durante mi estancia en la empresa y sobre todo mantener un estándar en la generación y presentación de datos, válido y confiable para la empresa y los procesos internos.

La documentación fue cargada y guardada en una carpeta en la nube a través de GitHub, para el libre acceso a la documentación para aquellas personas con autorización a la nube, adjuntando el archivo README, el cual tiene el propósito de dar información general del manejo correcto del repositorio con archivos de **jupyter notebook (.ipynb),** este incluye:

- Requerimientos previos para la ejecución de Scripts
- Árbol de carpetas
- Instrucciones para clonar repositorio

README

Folder Tree

Before starting, you have to make sure you have everything in order, to accomplish this, create in the File Explorer the next folder tree in the Documents folder, as shown.

```
Árbol de carpetas
```

Requirements

- Python 3 installed
- Previous access to Git remote repository
- Access to IT - Project Engineer
- Git installed

Instructions

1. Firstly, you have to create a folder named pythonEnvironments where venv (virtual environment) is going to be allocated
2. Get into the folder previously created and type the next command line to make a venv `python -m venv NEST_venv`
3. Place into the folder created by the previous command and you must run the next line to activate the virtual environment: `Scripts\activate`
4. Clone the remote repository, you have to run the next command line: `link a repositorio remoto`
5. To install jupyter Lab, run the next line (the venv must be activated): `pip install --proxy [gateway definido por IT] -U jupyterlab`
6. When everything is set, you just have to run Jupyter lab by typing `jupyter lab` in the command line.
7. You are going to need to add some libraries to properly run the scripts, just run the command from instruction '5' but substitute `-U jupyterlab` by `<library>`. In this case: pandas, pyodbc and openpyxl .
8. To create NEST Dashboard folder, you have to request it since it was provided by the IT Team, once you have it you must place it into [Carpeta] IT folder
9. Open the script named "Nombre de carpeta" and change the root directory to your own, since it is set for another account.

Este archivo README permitió definir un estándar en la instalación y reducir errores o mal funcionamiento de los Scripts de automatización.

8.3 Verificación

A partir de los resultados obtenidos, los cuales incluyen:

- Reportes de scripts de automatización
- Dashboards para presentación de datos

Durante su desarrollo, identifiqué la necesidad de implementar etapas formales de verificación y validación en conjunto con el equipo de trabajo durante y después del desarrollo de los Scripts, particularmente con la supervisora a cargo. Durante estas sesiones detecté ciertas inconsistencias en los datos procesados, lo cual evidenció la importancia de revisar tanto la lógica de los scripts como la información proveniente de las fuentes originales.

Como parte del proceso de desarrollo, llevé a cabo reuniones periódicas para recabar requisitos, definir procedimientos y delimitar los parámetros relevantes conforme a las necesidades del equipo NEST. Estas sesiones me permitieron ajustar los scripts y dashboards, asegurando que los datos recopilados fueran los adecuados para el análisis de los KPI's, así como contemplar casos extraordinarios y escenarios no previstos inicialmente.

La fase de verificación también resultó fundamental para contrastar la información generada por los sistemas automatizados con los registros operativos mantenidos por los responsables de cada área, lo que permitió identificar posibles errores o discrepancias en las bases de datos. Este proceso contribuyó a mejorar la confiabilidad del sistema y garantizó la transparencia y consistencia de la información presentada en los dashboards, fortaleciendo así la toma de decisiones basada en datos.

9 Conclusiones

Durante el desarrollo de mi estancia profesional dentro del sector industrial, tuve la oportunidad de aplicar de manera práctica los conocimientos adquiridos a lo largo de mi formación como ingeniero, particularmente en el análisis de datos, automatización de procesos y visualización de información para la toma de decisiones.

El objetivo principal de mi participación fue desarrollar soluciones que permitieran optimizar la obtención y análisis de indicadores clave de desempeño (KPI's), reduciendo la dependencia de procesos manuales y mejorando la confiabilidad de la información. A partir del diseño e implementación de scripts de automatización, fue posible transformar bases de datos en bruto en información estructurada, depurada y orientada al análisis, lo que contribuyó significativamente a la eficiencia del proceso y a la reducción de tiempos de ejecución.

Asimismo, la creación de dashboards dinámicos permitió presentar los resultados de manera clara y organizada, facilitando la interpretación de los indicadores tanto para líderes de equipo como para los ingenieros involucrados. Estas herramientas visuales fortalecieron la capacidad de análisis del equipo, permitiendo identificar tendencias, evaluar el desempeño operativo y respaldar la toma de decisiones con información actualizada y confiable.

Otro aspecto relevante de mi aportación fue la elaboración de documentación técnica y manuales de uso, los cuales aseguraron la reproducibilidad, mantenimiento y continuidad de las soluciones desarrolladas. Esta documentación permitió que los sistemas implementados pudieran ser utilizados y escalados por otros ingenieros, incluso después de concluida mi estancia profesional. Por lo tanto, se cumplieron los objetivos del presente trabajo y se solucionó la problemática de la empresa dando como resultado un sistema completo, estructurado y escalable.

En conjunto, las actividades realizadas no solo generaron un impacto positivo en la optimización de procesos y en la gestión de información dentro del área donde colaboré, sino que también fortalecieron mi criterio profesional, mi capacidad de análisis y mi experiencia en el desarrollo de soluciones de ingeniería orientadas a la mejora continua. Asimismo, esta experiencia representó un paso decisivo en mi formación, pues me permitió enfrentar desafíos reales en un entorno corporativo y aplicar de manera práctica los conocimientos adquiridos. Al analizar problemáticas concretas y diseñar soluciones técnicas eficientes, consolidé mi habilidad para integrar teoría y práctica, desarrollar pensamiento crítico y adaptarme a las exigencias de un contexto laboral dinámico.

Durante mi experiencia como becario, comprendí profundamente el papel fundamental que la Facultad de Ingeniería tuvo en mi formación. No solo me proporcionó las bases teóricas y las herramientas prácticas necesarias para desenvolverme en el ámbito profesional, sino que también me impulsó a descubrir y fortalecer mi vocación en áreas que realmente me apasionan. La preparación académica recibida fue decisiva para que pudiera adaptarme con confianza a un entorno laboral real, aplicando conocimientos en programación, análisis de datos y resolución de problemas. Más allá de lo técnico, la Facultad representó para mí un espacio de crecimiento personal, donde aprendí a enfrentar retos con disciplina y creatividad, convirtiéndose en un pilar esencial de mi desarrollo y en una etapa que marcó mi vida de manera significativa.

10 Oportunidades de mejora

Como parte del desarrollo realizado, se identificaron diversas vertientes susceptibles de mejora que permitirían alcanzar un mayor grado de automatización y autonomía en los procesos implementados. Una de las principales áreas de oportunidad corresponde al desarrollo de interfaces de usuario más robustas e intuitivas, las cuales facilitarían la interacción entre los usuarios y los sistemas de análisis de datos desarrollados.

La implementación de estas interfaces permitiría simplificar considerablemente la ejecución de los scripts, ya que los usuarios podrían ingresar parámetros, seleccionar rangos de información y configurar distintos criterios de análisis sin necesidad de modificar directamente el código fuente. De esta manera, se reduciría la dependencia técnica para la operación de las herramientas y se incrementaría su accesibilidad para distintos perfiles de usuario dentro de la organización.

Asimismo, el desarrollo de interfaces gráficas contribuiría a la creación de sistemas más flexibles y escalables, permitiendo la incorporación dinámica de nuevos KPI's, métricas y fuentes de información conforme evolucionen las necesidades operativas del equipo NEST.

11 Apéndice

11.1 Facturación

Generación de reporte de lo facturado del NEST

Para poder actualizar la data relacionado a lo facturado, este al ser actualizado mes con mes, por esta razón se debe de actualizar la carpeta donde se encuentran todos los extractos de reportes, primero describiré el proceso de obtención de reportes, este es decrito de la siguiente manera:

1. Pasos para poder acceder y descargar la base de datos en específico a traves del software utilizado por la empresa

De esta manera podemos obtener, para cada una de las plantas, el reporte de lo facturado, solo queda descargarlo en formato de excel y guardarlo en la carpeta **Ruta completa de la carpeta donde guardaremos la base de datos** junto con los demás extractos, es recomendable guardar cada uno de los archivos con el siguiente formato **Shipped (Planta) (Mes) (Año)**, de esta manera podemos trackear de mejor manera cada uno de los reportes. Adicionalmente, a cada uno de los nuevos reportes descargados, se deberá renombrar la columna "*gid_act*" como "*act_ship_dt*". Ya que se tengan los tres archivos (Planta1, Planta2 y Planta3) correctamente guardados, ya es posible ejecutar el script completo

```
#Incluimos las librerias
import pandas as pd
import pyodbc
import sys
import os
import re
```

```
#Declaramos en variables las cadenas de caracteres para las rutas donde se extraeran y guardaran las bases de datos
rootdir = "RutaParaArchivoFuente"
destinationDir  = "RutaParaArchivoDestino"
rootOneDrive ="RutaDeArchivoEnLaNube"
destination_path  = rootdir

#Declaramos variables para nombrar las bases de datos resultados
destination_filename_xls  = "NombreArchivoResultado.xlsx"
destination_filename_csv  = "NombreArchivoResultado.csv"
destination_filename_drive  = "NombreArchivoFuente.xlsx"
manualOrders_drive  = "NombreArchivoFuenteNube.xlsx"

#Creamos las paths que apuntan a las carpetas deseadas
destination_fullpath_xls  = os.path.join(destinationDir,destination_filename_xls)
destination_fullpath_csv  = os.path.join(destinationDir,destination_filename_csv)
destination_fullpath_drive  = os.path.join(rootOneDrive,destination_filename_drive)
manual_orders_fullpath  = os.path.join(rootOneDrive,manualOrders_drive)


#Definimos un path vacio
logPath = ''
#Creamos la libreria para mapear las plantas de NEST
mapPlants = {"Libreria que mapea plantas por codigo"}
#Arreglo para seleccionar las columnas de interes
columnsArr =
["sale_order","commit_dt","act_ship_dt","appl_engr","appl_engr_nm","mech_engr","mech_engr_nm","sects","itm_unit_
net","itm_unit_list","plant_id","pelt",  "intl_crncy_cd"]
#Constante de cambio de Dolares Canadienses a Dolares Americanos
CADtoUSD = 0.7356

#Creamos dos DataFrames para almacenar datos exportados
finalData = pd.DataFrame(columns=columnsArr)
tempData = pd.DataFrame()

#Exportamos datos de un archivo de excel
manualOrdersData  = pd.read_excel(manual_orders_fullpath,sheet_name="Main  data", engine="openpyxl")
#Cambiamos tipo de dato de parametros
manualOrdersData["sects"]  = manualOrdersData["sects"].astype(str)
manualOrdersData["itm_unit_list"]  = manualOrdersData["itm_unit_list"].astype(str)
manualOrdersData["pelt"]  = manualOrdersData["pelt"].astype(str)
#Arreglo que guarda nombres e iniciales de ingenieros de NEST
nameEng = ['Nombres de Ingenieros  de NEST']
initialEng = ['Iniciales de Ingenieros de Nest']

#Libreria para mapear las iniciales con los nombres de cada ingeniero
mapNames ={
"Libreria que mapea nombres de ingenieros  a traves de sus iniciales"
}

#Arreglo que guarda nombres e iniciales de ingenieros mecanicos de NEST
mechEng = ["Nombres de ingenieros  mecanicos de NEST"]
mechInitials = ["Iniciales de ingenieros  mecanicos de NEST"]

#Libreria para mapear ingeniero mecanico de NEST
mapMech = {
"Libreria que mapea nombres de ingenieros  mecanicos a traves de sus iniciales"
}
```

```
#Importamos datos de Excel a DataFrame y normalizamos
for subdir, dirs, files in os.walk(rootdir):
  for file in files:
    toolName = os.path.splitext(file)[0]
    ext = os.path.splitext(file)[1]
    logPath = os.path.join(subdir, file)
    tempData = pd.read_excel(logPath,usecols=columnsArr,dtype={'plant_id': object})
    tempData['plant_id'] = tempData['plant_id'].map(mapPlants)
    tempData['appl_engr'] = tempData['appl_engr'].str.split('/').str[0]
    tempData['mech_engr'] = tempData['mech_engr'].str.split('/').str[0]

    finalData = pd.concat([finalData, tempData])

#Filtramos los datos por nombre e inicial de los ingenieros
finalData = finalData.query("appl_engr == @initialEng | appl_engr_nm == @nameEng | mech_engr == @initialEng
| mech_engr_nm == @nameEng")
finalData["mech_engr_nm"] = finalData.apply(lambda row : None if row["mech_engr_nm"] not in nameEng else
row["mech_engr_nm"], axis=1)
finalData["appl_engr_nm"] = finalData.apply(lambda row : None if row["appl_engr_nm"] not in nameEng else
row["appl_engr_nm"], axis=1)
finalData = finalData.drop_duplicates()

#Convertimos datos a tipo hora y fecha
finalData[["commit_dt", "act_ship_dt"]] = finalData[["commit_dt", "act_ship_dt"]].apply(pd.to_datetime)

#Hacemos la conversion de dolares CAD a dolares USA
filtered_data = finalData.loc[finalData["intl_crncy_cd"] == 'CAD', ['itm_unit_net', 'itm_unit_list']]
finalData.loc[finalData["intl_crncy_cd"] == 'CAD', ['itm_unit_net', 'itm_unit_list']] = filtered_data * CADtoUSD
finalData = finalData.drop("intl_crncy_cd", axis=1)

#Hace un filtro booleano para identificar celdas no nulas
finalData = finalData[finalData.act_ship_dt.notnull()]
#Crea una lista de dos DataFrames
frames = [finalData, manualOrdersData]
#Concatena ambas Dataframes
finalData =  pd.concat(frames)

#Sustituye nombres en el apartado correcto y elimina las columnas auxiliares, asegura tipo de dato de columna
finalData["name_2"] = finalData["appl_engr"].map(mapNames)
finalData["appl_engr_nm"] = finalData["appl_engr_nm"].fillna(finalData["name_2"])
finalData.drop("name_2", axis=1, inplace=True)
finalData["name_3"] = finalData["mech_engr"].map(mapNames)
finalData["mech_engr_nm"] = finalData["mech_engr_nm"].fillna(finalData["name_3"])
finalData.drop("name_3", axis=1, inplace=True)
finalData["sects"] = finalData["sects"].astype(int)
finalData["itm_unit_net"] = finalData["itm_unit_net"].astype(float)
finalData["itm_unit_list"] = finalData["itm_unit_list"].astype(float)
finalData["pelt"] = finalData["pelt"].astype(str)
#Imprime la DataFrame
display(finalData)
#Se crea una copia de Dataframe
finalData_2 = finalData.copy()
#Se cambia parametro de fecha a otro tipo de fecha
finalData_2["year_month"] = finalData_2["act_ship_dt"].dt.to_period("M")
```

```
    # Filtrar filas donde 'itm_unit_net' y 'itm_unit_list' no están en blanco
filteredData = finalData_2.dropna(subset=['appl_engr_nm'])
# Agrupar y sumar los valores
dfSummary = filteredData.groupby(["plant_id",  "year_month"])[["itm_unit_net", "itm_unit_list"]].sum().reset_index()
dfSummary["itm_unit_net"] = dfSummary["itm_unit_net"].apply(lambda x: f"${x:,.2f}")
dfSummary["itm_unit_list"] = dfSummary["itm_unit_list"].apply(lambda x: f"${x:,.2f}")

display(dfSummary)
#Exporta DataFrame a Excel
finalData.to_excel(destination_fullpath_xls,index=False)
```

Ya que se generó el archivo de excel con todos los datos de Shipped, ahora en necesario sustituir todos los datos del archivos de **Nombre de archivo en la nube.xlsx** localizado en la carpeta *OneDrive* **Ruta completa del directorio en one drive** con los nuevos que se encuentran en el archivo nombrado **Nombre de archivo generado.xlsx** que se encuentra en la carpeta **Ruta completa al archivo generado**, previamente verificar los datos generados en la tabla anterior donde se muestra un resumen por planta y mes.

11.2 OCN (Order Change Notification)

Reporte de OCN's de NEST

Para obtener los datos relacionados a los OCN's que corresponden al equipo de NEST se tienen que descargar los archivos correpsondientes a cada planta; Planta1, Planta2 y Planta3, esto se logra a través de Symmetry Analytics siguiendo el siguiente proceso:

1. Pasos para acceder a la sección de OCN's

Una vez que se abre la pestaña para ingresar los diferentes parámetros para filtrar,vamos a seleccionar las caracteristicas para cada planta de la sigueinet forma:

1. Filtros para poder obtener la base de datos de interes

Ya que se despegó la data de manera corroborar con los filtros que se encuentren en las orillas del reporte, verificar que en **STATUS** solo esten marcadas las ocpiones de "*Open*" y "*Closed*", en el filtro de **ENG CAUSE** marcar la casilla de "*Yes*" y por último el filtro de **PRODUCT TYPE** seleccionar "*ETO*" y "*AUTO RELEASE*".

Ya que se tienen todos los filtros Set, nos vamos a la pestaña de "*Workflow detail*" y descargar el archivo en formato Excel,este debe ser almanecando como **OCN(Planta)(Año)(Mes)**

```
#Importamos librerias
import pandas as pd
import numpy as np
import pyodbc
import sys
import os
import re
#Declaramos en variables las cadenas de caracteres para las rutas donde se extraeran y guardaran las bases de datos
rootdir = "RutaParaArchivoFuente"
destinationDir = "RutaParaArchivoDestino"
oneDriveData = "RutaArchivoNube"
destination_path = rootdir

#Declaramos nombre de archivo para resultado
destination_filename_xls = "NombreArchivoGenerado.xlsx"

#Definimos ruta completa de archivo resultado
destination_fullpath_xls = os.path.join(destinationDir,destination_filename_xls)

#Función para filtrar DataFrame por planta, roles y area
def fillEngineerArea(x):
  if x["Plant"] == "Planta1" and x["Discipline"] in ["Electrical", "Electrical - ED"]:
    x["Engineer"] = x["Application Engineer"]
    x["Area"] = x["Discipline"]

  elif x["Plant"] == "Planta1" and x["Discipline"] == "Mechanical":
    x["Engineer"] = x["Mechanical Engineer"]
    x["Area"] = x["Discipline"]

  elif x["Plant"] in ["Planta2","Planta3"]:
    if x["Discipline"] in ["Electrical","Electrical - ED"]:
      x["Engineer"] = x["Application Engineer"]
      x["Area"] = x["Discipline"]
    else:
      if pd.isna(x["Mechanical Engineer"]):
        x["Engineer"] = x["Application Engineer"]
        x["Area"] = x["Discipline"]
      else:
        x["Engineer"] = x["Mechanical Engineer"]
        x["Area"] = x["Discipline"]
  return x

#Funcion para filtrar filas por nombres de ingenieros de NEST
def llenarData(row):
  global nameEng
  global mechEng
  if row["Engineer"] in nameEng or row["Engineer"] in mechEng:
    return row
  else:
    return None
```

```
#Funcion para llenar parametro DPU (defectos por unidad) de DataFrame
def llenarDPU(ocn ,shipped):
  global dpu

  shipped["year_month"]  = shipped["act_ship_dt"].dt.to_period("M")
  filtered_shipped  = shipped.groupby(["plant_id",  "year_month"])[["sects"]].sum().reset_index()
  filtered_shipped  = filtered_shipped.rename(columns  = {"plant_id":"Plant"})


  ocn["Group Date"] = pd.to_datetime(ocn["Group Date"]).dt.strftime('%Y-%m')
  filtered_shipped["year_month"]  = filtered_shipped["year_month"].astype(str)

  filtered_shipped_renamed  = filtered_shipped.rename(columns={"year_month":  "Group Date"})
  ocn = ocn.merge(filtered_shipped_renamed[["Group  Date", "Plant", "sects"]], on=["Group Date", "Plant"], how="left")
  ocn["DPU"] = ocn.apply(lambda row : row["OCN"] / row["sects"], axis=1)
  return con

#Definimos un path vacio
logPath = ''
#Arreglo para seleccionar las columnas de interes en diferentes DataFrames
disciplinesArr  = ["Mechanical", "Electrical", "Electrical - ED"]
columnsArr = ["Plant", "Product", "Product Type", "Status", "Sale Order", "Units", "Root Cause", "Discipline", "Application
Engineer", "Mechanical Engineer", "Recorded Date", "Workflow Detail"]
columnsDPU = ["Plant", "Date", "Total Sects", "OCN Sects", "DPU"]
columnSort = ["Plant", "Product", "Product Type", "Status", "Sale Order", "Units", "Root Cause", "Discipline", "Application
Engineer", "Mechanical Engineer", "Recorded Date", "Group Date", "Workflow Detail", "Engineer", "Area", "OCN", "DPU"]
#Creamos DataFrames
finalData = pd.DataFrame(columns=columnsArr)
tempData = pd.DataFrame()
#Varibale que almacena nombre de planta
assignPlanta3  = "Planta3"
#Creamos DataFrame para exportar archivo Excel de shippment
df_sh = pd.DataFrame()
#Exportamos base de datos
df_sh = pd.read_excel(oneDriveData,  sheet_name="Sheet1")
#Creamos DataFrame con columnas de arregloDPU
dpu = pd.DataFrame(columns=columnsDPU)

#Arreglos para almacenar datos de nombre e iniciales de ingenieros
nameEng = ['Nombres de ingenieros  eléctricos']
initialEng  = ['Iniciales de ingenieros  eléctricos']
#Libreria para mapear las iniciales con los nombres de cada ingeniero
mapNames ={
"Libreria  para  mapear  nombres de ingenieros  eléctricos a partir  de sus iniciales"
}
#Arreglo para almacenar nombres e iniciales de ingeneiros mecanicos de NEST
mechEng = ["Nombres de ingenieros  mecánicos"]
mechInitials = ["Iniciales de ingenieros  mecánicos"]
#Libreria para mapear las iniciales con los nombres de cada ingeniero mecanico
mapMech = {
"Libreria  para  mapear  nombres de ingenieros  mecánicos a partir  de sus iniciales"
}
```

```
#Importamos datos de Excel a DataFrame y normalizamos
for subdir, dirs, files in os.walk(rootdir):
   for file in files:
      toolName = os.path.splitext(file)[0]
      ext = os.path.splitext(file)[1]
      logPath = os.path.join(subdir, file)
      try:
         tempData = pd.read_excel(logPath,usecols=columnsArr,dtype={'Plant': object})
      except Exception as e:
         print(e)
      tempData["Application Engineer"] = tempData["Application Engineer"].str.split('/').str[0]
      tempData["Mechanical Engineer"] = tempData["Mechanical Engineer"].str.split('/').str[0]
      finalData = pd.concat([finalData, tempData])
      print(logPath)
#Filtramos DataFrame por disciplina y por nombres de ingeniero
finalData = finalData.query("Discipline in @disciplinesArr")
finalData = finalData.query("`Application Engineer` in @initialEng | `Application Engineer` in @nameEng | `Mechanical Engineer`
in @mechInitials | `Mechanical Engineer` in @mechEng")

#Libreria con parametros de interes
data2 = {
   "Engineer": [],
   "Area": []
}
#Creamos DataFrame
data = pd.DataFrame(data2)
#Concatenamos dos DataFrames
finalData = pd.concat([finalData, data],axis=1)
#Estandarizamos tipo de dato
finalData["Engineer"] = finalData["Engineer"].astype(str)
finalData["Engineer"] = finalData["Area"].astype(str)
#Normalizamos datos
finalData["Plant"] = finalData["Plant"].str.title()
finalData["Application Engineer"] = finalData["Application Engineer"].apply(lambda x : x.strip() if isinstance(x, str) else x)
finalData["Mechanical Engineer"] = finalData["Mechanical Engineer"].apply(lambda x : x.strip() if isinstance(x, str) else x)
finalData["Application Engineer"] = finalData["Application Engineer"].apply(lambda x : mapNames[x] if x in initialEng else x)
finalData["Mechanical Engineer"] = finalData["Mechanical Engineer"].apply(lambda x : mapMech[x] if x in mechInitials else x)
#Mapeamos datos de nombres de ingeniero
finalData["Engineer"] = finalData["Engineer"].map(mapNames)
finalData["Engineer"] = finalData["Engineer"].map(mapMech)
#Llenamos campos de ingeniero y area
finalData = finalData.apply(lambda row : fillEngineerArea(row), axis=1)
finalData = finalData.apply(lambda row : llenarData(row), axis=1)
#Eliminamos filas vacias
lastData = finalData.dropna(how='all')
#Asignamos pesos a cada fila, fila = 1 OCN
lastData["OCN"] = 1
#Renombramos aquelllas plantas que cumplen con el parametro
lastData["Plant"] = lastData["Plant"].apply(lambda x : "Planta3" if x == "Planta3" else x)
#Damos fromato al parametro de fecha y hora
lastData["Group Date"] = lastData["Recorded Date"].dt.strftime('%Y-%m')
#Llenamos DataFrame con DPU
DPU_df = llenarDPU(lastData, df_sh)
#Acomodamos columnas
DPU_df = DPU_df[columnSort]
#Impirmimos DataFrames
display(lastData)
display(DPU_df)


#Exportamos DataFrame a Excel
DPU_df.to_excel(destination_fullpath_xls,index=False)
```

11.3 Nivel de servicio para ingenieros mecánicos

Obtención de datos acerca de los MD de NEST

Para poder ejecutar correcatemnete en script llamado *NEST_GENERADO_MECANICOS* es necesario obtener los datos por mes de interés de los mecánicos, esto se hace de la siguiente manera: Primeramente, se tiene que descargar el reporte por mes de cada uno de los MD, este reporte puede ser generado En la sección de Engineering llamado "*SecciónEngineering*", una vez que estemos en esta pestaña debemos de ingresar los siguientes filtros

1. Engineer: Ingresar el nombre cdel ingeniero
2. From: Fecha de inicio de interés
3. To: Fecha de fin de interés

Y dar click en el botón de *Refresh*

Exportar los datos obtenidos a un formato en excel y guardarlo con el nombre de **(NombreDeIngeniero)_(Año)(Mes).xlsx**en la siguiente ruta de carpetas **Ruta de carpetas**, este proceso de debe de repetir para cada uno de los MD filtrando el mes correspondiente.

```
#Importamos librerias
import pandas as pd
import os
import numpy as np
```

Para filtrar correctamente los datos por fecha correspondiente, es necesario modificar las variables "*Inicio*" y "*Fin*" que se encuentran en el siguiente diagrama de código, este debe de ser modificado para cubiri el mes de interés conservando el formato de fechas *mes/dia/año*, es recomendable solo cambiar el mes al que queramos.

```
#Declaramos en variables las cadenas de caracteres para las rutas donde se extraeran y guardaran las bases de datos
rootdir = "RutaArchivoFuente"
destination = "RutaArchivoDestino"
destination_filename_xls = "NombreArchivoGenerado.xlsx"
#Definimos formato de fecha
Inicio = "06/01/2025"
Fin = "06/30/2025"

#Convertimos a tipo de dato con formato especifico
fechaInicio = pd.to_datetime(Inicio, format="%m/%d/%Y")
fechaFin = pd.to_datetime(Fin, format="%m/%d/%Y")
```

```
    #Funcion para borrar filas vacias
def borrarFilasVacias(row):
  if isinstance(row["latest_end_dt"], pd.Timestamp):
    return row
  else:
    return None
#Funcion para llenar status: on-time, past-due
def llenarStatus(row):
  from datetime import datetime
  global status_past_due
  global status_on_time
  if pd.isna(row["actual_end_dt"]):
    if row["latest_end_dt"] >= datetime.now():
      return status_on_time[1]
    else:
      return status_past_due[1]
  else:
    if row["latest_end_dt"] >= row["actual_end_dt"]:
      return status_on_time[0]
    else:
      return status_past_due[0]
#Arreglo para almacenar columnas de interes
columnsArr = ["sale_order", "latest_end_dt", "actual_end_dt", "mech_engr", "mech_engr_nm", "activity_desc", "sects"]
#Variable para definir rol del ingeniero
activityFilter = "MECHANICAL DESIGN"
#Arreglo que almacena status del proyecto
status_past_due = ["Closed - Past Due", "Past Due"]
status_on_time = ["Closed - On Time", "On Time"]
#Creamos DataFrame con columnas especificas
df = pd.DataFrame(columns = columnsArr)
#Creamos DataFrame
df_temp = pd.DataFrame()

#Librerua para mapear nomrbes de ingenieros
mapNames = {
"Libreria para mapear nombres de ingeneiros mecanicos a partir de sus iniciales"
}

#Importamos base de datos de Excel a DataFrame
for root, dirs, files in os.walk(rootdir):
  for file in files:
    file_path = os.path.join(root, file)
    df_temp = pd.read_excel(file_path, usecols = columnsArr)
    df = pd.concat([df, df_temp], ignore_index = True)


#Filtramos por rol de ingeniero
df = df.query("activity_desc == @activityFilter")
#Mapeamos los nombres de los ingenieros
df["mech_engr_nm"] = df.apply(lambda row : mapNames[row["mech_engr"]] if pd.isna(row["mech_engr_nm"]) else
row["mech_engr_nm"], axis=1)
#Normalizamos datos
df["mech_engr_nm"] = df["mech_engr_nm"].apply(lambda x : x.title())
df["latest_end_dt"] = df["latest_end_dt"].fillna("")
```

```
#Borramos filas vacias
df = df.apply(lambda row : borrarFilasVacias(row), axis=1).dropna(how = "all")
#Asignamos status a cada proyecto
df["status"] = df.apply(lambda row : llenarStatus(row), axis=1)
#Asignamos valor dependiendo del status
df["service_level"] = df.apply(lambda row : 1 if row["status"] in status_on_time else 0, axis=1)
#Eliminamos duplicados
df = df.drop_duplicates()
#Filtramos por fecha de interes
df = df.query("@fechaInicio <= latest_end_dt <= @fechaFin")
#Imprimimos el DataFrame
display(df)


#Exportamos DataFrame a Excel
df.to_excel(destination + destination_filename_xls, index=False)
```

Finalmente, ya una vez ordenada y limpiada la data, este se alojara en la ruta **Ruta de archivo generado** la cual será utilizada por el proceso siguiente para obtener lo generado de cada no de los proyectos de los MD.

11.4 Nivel de servicio para ingenieros eléctricos

Generación de reporte para Service Level del NEST

En esta sección vamos a depurar los datos para que podamos obtener aquella relacionada al nivel de servicio de cada uno de los ingenieros del NEST, primero que nada es sumamante importante mencionar que para que al momento de ejecutar este script se tome en cuenta todos los datos relacionados hasta el mes de interes, se debera agregar en su totalidad los datos de todos las categorias que existen del NEST, con esto me refiero a que ya hayamos agregado los datos de lo generado por parte de Planta1, Planta2 y Planta3, este ultimo toma en cuenta tanto a los Eléctricos como a los Mecánicos, estos datos se deberán añadir al archivo **NombreArchivoNube.xlsx** en la carpeta *OneDrive* con ruta **Ruta archivo nube**, por lo que deberá ser posterior a actualizar este archivo, una vez actualizado este se puede ejecutar sin ningun inconveniente.

```
#Importamos librerias
import pandas as pd
import numpy as np
import os
#Declaramos en variables las cadenas de caracteres para las rutas donde se extraeran y guardaran las bases de datos
rootDir_generado  = "RutaCompletaACarpetaOrigen"
destinationDir = "RutaCompletaACarpetaDestino"
#Declaramos nombre de archivos
destination_filename_xls  = "NombreDeArchivoDestino.xlsx"
filename_generado  = "NombreDeArchivoFuente.xlsx"
```

En este bloque de código maneja exepciones las cuales se dan cuando el proyecto aparece como atrasado, pero esto se debe a que tuvo un *Workflow* que pudo atrasar la entrega de este proyecto, para estos casos debes de buscar aquellos que salgan como atrasados y consultar con los ingenieros si es que tuvo o no *Workflow*, en el caso del Tracker de Planta1, que es donde es más usual se dan estos casos, vamos a buscar el proyecto en el archivo fuente, si es que se encuentra que tiene *Workflow* su numero de proyecto deberá ser colocado en el arreglo de abajo con el formato correspondiente, de esta manera será asignado **On - Time** automáticamente, tener cuidado y corroborar las fechas y los retrasos con el equipo.

```
#Guardamos en un arreglo aquellas excepciones
excepciones = ["Códigos de proyectos"]
#Guardamos en un arreglo las columnas que queremos en nuestro DataFrame
columnsArr_generado  = ["Q2C", "Promise Finish End Date", "Real Finish End Date", "Resource Initials", "Secciones", "Activity", "Plant"]
#Creamos DataFrame
data_gen  = pd.DataFrame()
#Creamos ruta completa para arhcivo
file_path_gen  = os.path.join(rootDir_generado,  filename_generado)
#Importamos de excel a DataFrame
data_gen  = pd.read_excel(file_path_gen,  usecols=columnsArr_generado)
#Guardamos en arreglos los status de interes
status_past_due  = ["Closed - Past Due", "Past Due"]
status_on_time  = ["Closed - On Time", "On Time"]
#Funcion para llenar el status en relacion a otro parametro
def llenarStatus(row):
    from datetime import datetime
    global status_past_due
    global status_on_time
    if row["Q2C"] in excepciones:
        return status_on_time[0]
    else:
        if pd.isna(row["Real  Finish End Date"]):
            if row["Promise  Finish End Date"] >= datetime.now():
                return status_on_time[1]
            else:
                return status_past_due[1]
        else:
            if row["Promise  Finish End Date"] >= row["Real  Finish  End Date"]:
                return status_on_time[0]
            else:
                return status_past_due[0]
```

```
#Asignamos status a campo de DataFrame
data_gen["status"] = data_gen.apply(lambda row : llenarStatus(row), axis=1)
#Asignamos peso de nivel de serivio de acuerdo a status
data_gen["service_level"] = data_gen.apply(lambda row : 1 if row["status"] in status_on_time else 0, axis=1)
#Renombramos columnas de DataFrame
data_gen.rename(columns={"Q2C":"sale_order", "Promise Finish End Date":"promise_end_dt", "Real Finish End
Date":"actual_end_dt",
                "Resource Initials":"engr_nm", "Secciones":"sects", "Activity":"activity", "Plant":"plant"}, inplace=True)
#Mostramos DataFrame
display(data_gen)
#Exportamos DataFrame a archivo de Excel
data_gen.to_excel(destinationDir + destination_filename_xls, index=False)
```

Una vez se haya generado el archivo correspondiente al Nivel de Servicio, este se hallará en la carpeta **Ruta archivo generado**, todos los datos deberán ser copiados y pegados en el archivo **NombreArchivoNube.xlsx** que se encuentra en la carpeta de *OneDrive* **Ruta archivo nube**. Los datos generados de este script deberan sustituir en su totalidad los datos del archivo de *OneDrive*, ya que estos se van acumulando en el archivo fuente.

11.5 Generado para Ingenieros mecánicos

Creación de reporte para lo Generado de los Mechanical Designers del NEST

Para poder generar los datos correspondientes a la cantidad generada por parte de los mecánicos, se debe de generar y actualizar primeramente los datos que se encuentran en la carpeta **Ruta de archivo fuente**, este proceso está descrito en el script llamado **SERVICE_LEVEL_MD**, una vez que esta data este actualizada en el mes más reciente ya que este contendra datos unicamente del mes correspondiente, por otra parte, se debe de obtener el reporte para sacar el precio neto a partir de Symmetry, este se puede obtener con el siguiente flujo:

1. Pasos para descargar la base de datos fuente

Una vez tengamos la data descargada se deberá de guardar en en la carpeta **Ruta para almacenar archivo fuente** bajo el nombre de **Mecanicos(Año del reporte)(Mes del reporte)**, misma que esta referenciada en la variable *"rootDir"* del script para la ruta del archivo (esta debe ser cambiada de la

variable "*rootDir*" del script para la ruta del archivo (esta debe ser cambiada de la misma manera cada que se ejecute un nuevo proceso) y el nombre del archivo que se encuentra en la variable "*filename_xlsx*" y debe debe ser cambiada de acuerdo al formato propuesto en este reporte.

De esta manera ya tenemos los datos necesarios para poder generar los datos correspondientes a un mes en específico. Los siguientes códigos se deberán ejecutar tomando en consideración cambios solo si se mencionan en un bloque de texto.

```
#Importamos librerias
import pandas as pd
```

Esta seccion de código se deberá actualizar cada vez que se quiera hacer un reporte mensual, ya que actualizaremos las rutas a carpetas al mes y año corrrespondiente, dichas carpetas se deberan crear para almacenar los archivos correspondientes y tener todos los reportes generados en diferentes secciones.

```
#Declaramos en variables las cadenas de caracteres para las rutas donde se extraeran y guardaran las bases de datos
rootDir = "Ruta de archivo fuente"
rootDirMD = "Ruta de archivo destino"
#Definimos nombres de archivo
filenameMD_xlsx = "NombreArchivoFuente.xlsx"
filename_xlsx = "NombreArchivoFuente.xlsx"  #Modificar nombre de filename de acuerdo al nombre del archivo
generatedfile_xlsx  = "FuenteArchivoGenerado.xlsx"
#Guardamos en arreglos las columnas que vamos a necesitas de las bases de datos
columnArr = ["sale_order", "latest_end_dt", "actual_end_dt", "mech_engr", "mech_engr_nm", "activity_desc", "sects",
"itm_unit_net"]
columnArrMD = ["sale_order", "latest_end_dt", "actual_end_dt", "mech_engr_nm", "activity_desc", "sects"]
sortColumn = ["sale_order", "latest_end_dt", "actual_end_dt", "mech_engr_nm", "activity_desc", "sects", "itm_list_net",
"itm_unit_net", "plant"]
#Nombres de los ingenieros de NEST
mapEngr = ["Nombres e inciales de ingenieros  mecánicos"]
#Libreria para mapear nombre de ingeneiros con sus iniciales
mapNames = {
"Libreria para mapear nombres de ingenieros  mecánicos a partir de sus iniciales"
}
#Creamos dos DataFrames
df_mech = pd.DataFrame()
df_gen = pd.DataFrame()
#Importamos ambas bases de datos y las guardamos en DataFrames
df_mech = pd.read_excel(rootDir + filename_xlsx, usecols=columnArr)
df_mechMD = pd.read_excel(rootDirMD  + filenameMD_xlsx, usecols=columnArrMD)
```

```
    #Normalizamos datos
df_mech["mech_engr"] = df_mech["mech_engr"].apply(lambda x: str(x).split("/")[0] if pd.notna(x) and "/" in str(x) else x)
#Filtramos de aguerdo a rol y nombre de los ingenieros
df_mech = df_mech.query("activity_desc == 'MECHANICAL DESIGN'")
df_mech = df_mech.query("mech_engr_nm in @mapEngr | mech_engr in @mapEngr")
#Damos formato al campo a aquellos no vacios
df_mech["mech_engr_nm"] = df_mech["mech_engr_nm"].apply(lambda x : x.title() if pd.notna(x) and str(x).strip() != "" else x)
#Mapeamos nombres de ingeniero a un campo auxiliar y posteriormente lo copiamos al original
df_mech["aux_name"] = df_mech.apply(lambda row : mapNames[row["mech_engr"]] if pd.isna(row["mech_engr_nm"]) else row["mech_engr_nm"], axis=1)
df_mech["mech_engr_nm"] = df_mech["aux_name"]
#Eliminamos campo auxiliar
df_mech = df_mech.drop(columns=["aux_name"])
#Mezclamos campos de un DataFrame
df_merge = df_mechMD.merge(df_mech[["sale_order", "itm_unit_net"]], on="sale_order", how="left")
#Asignamos valor a un campo
df_merge["itm_list_net"] = 0
#Asignamos planta a un campo
df_merge["plant"] = "Planta"
#Ordenamos las columnas de acuerdo a un patron
df_merge = df_merge[sortColumn]
#Imprimimos DataFrame
display(df_merge)
```

En la tabla anterior, podemos observar una vista previa de la esctructura de la tabla y algunos datos generales de ella

---

```
#Exportamos FataFrame a archivo de Excel
df_merge.to_excel(rootDir + generatedfile_xlsx,  index=False)
```

- Cuando se haya generado el archivo de excel con los datos correctos, este se almacenará en la misma carpeta pero bajo el nombre que hayamos asignado en la variable "generatedfile_xlsx", sin embargo, es sugerible cambiarlo de la misma manera al formato de "generadoMecanicos(Año)(Mes).xlsx", de esta manera sabremos la fecha y el proposito de cada archivo generado
- Es importante mencionar que los datos creados por este proceso debes ser colocados en el archivo de **NombreArchivoNube.xlsx** que se encuentra en la carpeta de **Ruta Archivo Nube**

11.6 Generado para planta 1

Generado de Planta NEST

Para esta sección, obtendremos los datos de lo generado para la planta de Planta, por esta razón, es necesario obtener el *Tracker* el cual puede ser proporcionado por cualquier ingeniero de Planta, una vez que se tiene el archivo, deberás de almacenarlo con el nombre de **Planta.xlsx** en una ruta de carpeta en específico, si es que no existe la ruta suguerida, debes de ir creando las carpetas necesarias manteniendo los formatos de fechas, el formato de carpetas donde se debe localizar el archivo fuente es **Ruta archivo fuente**, una vez que se haya localizado en dicha carpeta podremos ejecutar el código haciendo pequeñas modificaciones a éste.

```
#Importamos librerias
import pandas as pd
import pyodbc
import sys
import os
import re
from datetime import datetime
```

Para poder ejecutar el siguiente bloque de código correspondiente al mes y año solicitados, se deben de hacer modificaciones a los nombres de variables que estan relacionados a nombres de archivos y/o carpetas. Primero que nada se deben de cambiar las variables de "*rootDir*" y "*destinationDir*" con la ruta de las carpetas creadas que almacenan los archivos, en pocas palabras, cambiar fecha de cada subcarpeta creada manteniendo el formato de las carpetas originales, adicionalmente, se debe de cambiar el rango de fechas de las variables "*Inicio*" y "*Fin*", donde en la primera se indicará el inicio del rango de fechas de interes y la segunda variable indicara el ultimo dia del rango de fechas de interés, note que se mantuvo un formato de **mes/dia/año**. De esta manera se puede limpicar el archivo fuente para que solo obtengamos el correpsondiente a un mes en específico.

```
#Declaramos en variables las cadenas de caracteres para las rutas donde se extraeran y guardaran las bases de datos
rootDir = "Ruta archivo fuente"
destinationDir = "Ruta archivo destino"

#Declaramos variables con nombres de archivos
destination_filename_xls = "NombreArchivoGenerado.xlsx"
destination_filename_csv = "NombreArchivoGenerado.csv"
manualOrder = "NombreArchivoFuente.xlsx"
```

```
#Creamos la ruta completa donde se guardaran archivos
destination_fullpath_xls  = os.path.join(destinationDir,destination_filename_xls)
destination_fullpath_csv  = os.path.join(destinationDir,destination_filename_csv)
manual_order_fullpath  = os.path.join(rootDir,manualOrder)

#Definimos un rango de fecha de interes
Inicio = "06/01/2025"
Fin = "06/30/2025"

#Damos fromato de fecha y hora a nuestras variables
fechaInicio = pd.to_datetime(Inicio, format="%m/%d/%Y")
fechaFin = pd.to_datetime(Fin, format="%m/%d/%Y")
#Guardamos en un arreglo las columnas
columnsArr = ["Q2C","Promise Finish End Date","Real Finish End Date","Resource Initials","Activity","Secciones","Precio
Lista","Precio Neto","Plant"]
acceptedProgress = ["Completed not RM (Waiting for oneshuttle WF to be completed)", "Completed/RM"]
#Creamos DataFrame con columnas nombradas como en el arreglo columnsArr
finalData = pd.DataFrame(columns=columnsArr)
#Guardamos en un arreglo los nombres de ingenieros
nameEng = ["Nombre ingenieros electricos"]

##Importamos archivo de Excel a DataFrame
manualOrderData  = pd.read_excel(manual_order_fullpath, engine="openpyxl")
#Damos formato a columnas
manualOrderData.columns = manualOrderData.columns.str.strip().str.lower()
#Filtramos a partir de nombres de ingeniero
manualOrderData  = manualOrderData.query("`ae  of the project` == @nameEng")
#Filtramos a partir de rango de fecha
manualOrderData  = manualOrderData.query("@fechaInicio <= `current completion date` <= @fechaFin | `current completion
date`.isnull()")
#Filtramos por campos no vacios
manualOrderData  = manualOrderData.query("`current completion date`.notnull() or `actual completion date`.notnull()")
#Filtramos a partie de status
manualOrderData  = manualOrderData.query("`progress  point` in @acceptedProgress")
#Normalizamos campo
manualOrderData["ae  of the project"] = manualOrderData["ae  of the project"].apply(lambda x : x.title())
#Damos formato y tipo de dato a columnas
manualOrderData["title"]  = manualOrderData["title"].astype(int)
manualOrderData["line  item #"] = manualOrderData["line  item #"].astype(int)
manualOrderData["sale_order"]  = manualOrderData.apply(
   lambda row : str(row["title"]) + "-" + ("0000" if len(str(row["line item #"])) == 2 else "000" if len(str(row["line item #"])) == 3
else "00")  + str(row["line item #"]), axis =1)
#Damos fromato y tipo de dato a campos de fecha
manualOrderData["current  completion date"] = manualOrderData["current  completion date"].apply(lambda x :
x.strftime("%m/%d/%Y") if pd.notnull(x) else x)
manualOrderData["actual  completion date"] = manualOrderData["actual  completion date"].apply(lambda x :
x.strftime("%m/%d/%Y") if pd.notnull(x) else x)
#Damos formato de entero a la columna
manualOrderData["section  number"] = manualOrderData["section  number"].apply(lambda x : int(x) if pd.notnull(x) else 0)
#Damos formato float a datos de la columna
manualOrderData["net  price"] = manualOrderData["net  price"].apply(lambda x : float(x) if pd.notnull(x) else x)
#Asignamos valor nulo si cumple la condicion
manualOrderData["net  price"] = manualOrderData.apply(lambda  row : 0 if row["project status"] == "Change Order" else row["net
price"], axis = 1)
#Llenamos valores vacios
manualOrderData["actual  completion date"] = manualOrderData["actual  completion date"].fillna(manualOrderData["current
completion date"])
```

```
#Renombramos las columnas de acuerdo a otro DataFrame y asignamos valores
finalData["Q2C"] = manualOrderData["sale_order"]
finalData["Promise Finish End Date"] = manualOrderData["current completion date"]
finalData["Real Finish End Date"] = manualOrderData["actual completion date"]
finalData["Resource Initials"] = manualOrderData["ae of the project"]
finalData["Activity"] = manualOrderData["project status"]
finalData["Secciones"] = manualOrderData["section number"]
finalData["Precio Lista"] = 0
finalData["Precio Neto"] = manualOrderData["net price"]
finalData["Plant"] = "Planta"
#Mostramos DataFrame
display(finalData)
#Exportamos DataFrame a archivo de Excel
finalData.to_excel(destination_fullpath_xls,index=False)
```

Ya que tenemos el archivo generado por el script en la misma carpeta donde fue almacenado el archivo fuente **Ruta archivo denstino**, los datos deberán ser copiados y pegados en el archivo de **NombreArchivoNube.xlsx** que se encuentra en la carpeta de *OneDrive* **Ruta archivo nube**

11.7 Generado para planta 2

Reporte generado Planta del NEST

Para poder obtener la data de lo generado por parte de la planta de Planta, es necesario descargar dos bases de datos, una será en Symmetry y la segunda será en Symmetry Analytics. Primeramente, descargar la data de RM o RFD (son lo mismo) para la planta de Planta, se tiene que seguir el siguiente proceso, a partir de Symmetry, en la sección de Engineering, seleccionar Reports, posteriormente seleccionar *Engineering Raw Data*, una vez en ese apartado, ingresar los siguientes parámetros:

1. Flujo para acceder a base de datos fuente

Una vez se haya desplegado los datos, se deberá exportar en formato excel y guardarlo con el nombre de **PlantaS.xlsx** en una carpeta específica para lo generado del mes correspondiente, debe de ser una ruta de carpetas de la siguiente manera **Ruta archivo fuente**

Posteriormente, para las demás etapas debemos de utilizar Symmetry Analytics, donde se podrá descargar la data relacionada a los Approvals y a los Redraw, se debe seguir el siguiente diagrama de flujo para poder descargar dicha data:

Flujo para acceder a archivo fuente. Después se debe de seleccionar los filtros de la data con la información correspondiente a cada necesaidad:

1. Colocar datos correspondientes en filtros

Después seleccionar la pestaña de Order Detail, y descargar a un archivo Excel de esa pestaña en específico, este debe ser guardado con el nombre de archivo **PlantaSA.xlsx** en la misma ruta de carpetas que el proceso anterior de esta manera, ya tenemos los datos necesarios para generar el reporte del mes en cuestión.

```
#Importamos libreria
import pandas as pd
```

Para guardar todo de la manerra correcta, es necesario modificar algunas variables del script, primeramente vamos a modificar la variable llamada *"root"* con la ruta de las carpetas donde se almacenaron los archivos fuente, además, en las variables *"Inicio"* y *"Fin"* vamos a cambiar el numero del mes del que necesitamos extraer los datos, respetando el formato de mes/dia/año (se suguiere uicamente cambiar el numero que corresponde al mes).

```
#Declaramos en variables las cadenas de caracteres para las rutas donde se extraeran y guardaran las bases de datos
root = "Ruta archivos fuente"
#Definimos nombre de archivos a importar
filename1 = "PlantaS.xlsx"
filename2 = "PlantaSA.xlsx"
#Declaramos nombre de archivo a exportar
filename_gen = "NombreArchivoDestino.xlsx"

#Definimos rango de fechas de interes
Inicio = "06/01/2025"
Fin = "06/30/2025"

#Damos formato y tipo de dato a fechas
fechaInicio = pd.to_datetime(Inicio, format="%m/%d/%Y")
fechaFin = pd.to_datetime(Fin, format="%m/%d/%Y")
#Guardamos en arreglo las columnas que tendra DataFrame
columnsS = ["sale_order", "latest_end_dt", "actual_end_dt", "appl_engr", "appl_engr_nm", "eng_status", "sects",
"itm_unit_list", "itm_unit_net", "activity_desc"]
columnsSA = ["Sale Order", "Drawing Schedule - Sched", "Drawing Schedule - Actual", "Engineer", "Sections", "List",
"Net"]
sortColumnsS = ["sale_order", "latest_end_dt", "actual_end_dt", "appl_engr_nm", "eng_status", "sects", "itm_unit_list",
"itm_unit_net", "plant"]
```

```
    #Creamos DataFrames
df_S = pd.DataFrame()
df_SA = pd.DataFrame()
#Importamos archivos de Excel a Dataframes
df_S = pd.read_excel(root + filename1, usecols=columnsS)
df_SA = pd.read_excel(root + filename2, usecols=columnsSA)

#Guardamos en arreglos los nombres y las iniciales de los ingenieros
nameEng = ['Nombres ingenieros eléctricos']
initialEng = ['Iniciales ingenieros eléctricos']
#Libreria para mapear los nomrbes a partir de iniciales
mapNames ={
"Libreria para mapear nombres de ingenieros a partir de sus iniciales"
}
#Guardamos en un arreglo los status de interes
status = ["RM", "AE COMPLETE"]
#Filtramos DataFrame a partir de status, nombres e iniciales de ingenieros y por rango de fechas
df_S = df_S.query("activity_desc in @status")
df_S = df_S.query("appl_engr in @initialEng | appl_engr_nm in @nameEng")
df_S = df_S.query("@fechaInicio <= latest_end_dt <= @fechaFin")
#Mapemaos nombres de ingeniero en un campo auxiliar
df_S["aux_name"] = df_S["appl_engr"].apply(lambda x : mapNames[x] if isinstance(x, str) else x)
#Asignamos valores a aquellos campos vacios
df_S["appl_engr_nm"] = df_S["appl_engr_nm"].fillna(df_S["aux_name"])
#Filtramos a partie de status
df_S = df_S.query("eng_status in @status")
#Asignamos valores a campos
df_S["plant"] = "Planta"
df_S["itm_unit_list"] = 0
#Eliminamos columnas que ya no son necesarias
df_S.drop("aux_name", axis=1, inplace=True)
df_S.drop("activity_desc", axis=1, inplace=True)
df_S.drop("appl_engr", axis=1, inplace=True)
#Ordenamos columnas de acuerdo a patron
df_S = df_S[sortColumnsS]
#Filtramos DataFrame a partir de nombre de ingeneiros
df_SA = df_SA.query("Engineer in @nameEng")
#Asignamos valor
df_SA["plant"] = "Planta"
#Renombramos las columnas
df_SA1 = df_SA.rename(columns={"Sale Order" : "sale_order",
                    "Drawing Schedule - Sched" : "latest_end_dt",
                    "Drawing Schedule - Actual" : "actual_end_dt",
                    "Engineer" : "appl_engr_nm",
                    "Sections" : "sects",
                    "List" : "itm_unit_list",
                    "Net" : "itm_unit_net"})
#Asignamos peso a columna
df_SA1["itm_unit_list"] = 0
#Asignamos valor a columna
df_SA1["eng_status"] = "Approvals"
#Ordenamos columnas de DataFrame de acuerdo a patron
df_SA1 = df_SA1[sortColumnsS]
#Concatenamos ambos DataFrames
df_merged = pd.concat([df_S, df_SA1], ignore_index=True)
#Mostramos DataFrame ya despues del merge
display(df_merged)
```

```
#Exportamos DataFrame a archivo de Excel
df_merged.to_excel(root + filename_gen, index=False)
```

Ya que se tiene el archivo de lo generado de la planta de Smyrna, la cual se encuentra en la misma carpeta donde se almacenaron los archivos fuente, los datos se deberán copiar y pegar complementando el archivo **NombreArchivoNube.xlsx** que se encuentra en la siguiente ruta **Ruta archivo nube**

11.8 Generado para planta 3

Obtención de datos de lo generado de Planta del NEST

En esta sección se explicará la manera de obtener los datos de Planta, para ello vamos a tener que solicitar a un ingeniero del NEST de Planta que nos proporcione los datos de lo generado de su equipo (en mi caso se lo pedía a NombreDeIngeniero pero comentarlo con Supervisor), una vez que nos entrega los datos tendremos que estandarizar las columnas de acuerdo a las que se tienen en el archivo de **NombreArchivoNube.xlsx**, para que unicamente copiemos y peguemos los datos en este archivo que se encuentra en la siguiente ruta **Ruta archivo nube**, es decir, manipular y completar los datos con las siguientes columnas en este orden:

1. Q2C
2. Promise Finish End Date
3. Real Finish End Date
4. Resocurce Initial -> Se tiene que color en formato nombre (Mayúscula la primer letra de cada nombre o apellido)
5. Activity
6. Secciones
7. Precio Lista -> Completar la columna con ceros
8. Precio Neto
9. Planta -> Colocar "Planta"

12. Bibliografía

[1] “Metodologías ágiles: Qué son y cuáles son las más utilizadas”. ADEN International Business School. Accedido el 24 de septiembre de 2025. [En línea]. Disponible: https://www.aden.org/business-magazine/metodologias-agiles/

[2] V. Ortiz y H. F. Pardo, *Importancia y ventajas de los KPI en los proyectos*. 2021.

[3] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 4th ed., Cambridge, MA: Morgan Kaufmann, 2022.

[4] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 7th ed., Hoboken, NJ: John Wiley & Sons, 2018.

[5] R. W. Sebesta, *Concepts of Programming Languages*, 12th ed., Hoboken, NJ: Pearson, 2022.

[6] W. McKinney, *Python for Data Analysis*, 3rd ed. Sebastopol, CA: O’Reilly Media, 2022.

[7] F. Fuentes, “Primeros pasos para usar JupyterLab,” *Blog de Arsys*, Oct. 18, 2024. [Online]. Available: https://www.arsys.es/blog/primeros-pasos-para-usar-jupyterlab. Disponible: Oct. 5, 2025.

[8] W. Gittleson, “What is Tableau? The Complete Guide to Tableau,” *DataCamp Blog*, Oct. 20, 2023. [Online]. Available: https://www.datacamp.com/blog/all-about-tableau. Accessed: Oct. 5, 2025.

[9] R. Sebesta, *Concepts of Programming Languages*, 12th ed., Pearson, 2022.

[10] T. Haigh, M. Priestley y C. Rope, “ENIAC in Action,” *IEEE Annals of the History of Computing*, vol. 36, no. 3, pp. 41–62, 2014.

[11] NASA, “FORTRAN in Apollo Guidance Programming,” NASA Technical Reports, 1969.

[12] R. Sebesta, *Concepts of Programming Languages*, 12th ed., Pearson, 2022.

[13] S. Russell y P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed., Pearson, 2020.

[14] E. Feigenbaum, “The Art of Artificial Intelligence,” *Stanford Heuristics Lab Report*, 1977.

[15] B. W. Kernighan y D. Ritchie, *The C Programming Language*, 2nd ed., Prentice Hall, 1988.

[16] D. Ritchie, “The Evolution of the Unix Time-Sharing System,” *Bell Labs Technical Journal*, 1979.

[17] T. Berners-Lee, “Information Management: A Proposal,” CERN, 1989.

[18] R. W. Sebesta, *Concepts of Programming Languages*, 12th ed. Pearson, 2022.

[21] “6 ejemplos del mundo real de dashboards de inteligencia de negocios”. tableau. Accedido el 12 de enero de 2026. [Imagen]. Disponible: https://www.tableau.com/es-mx/learn/articles/business-intelligence-dashboards-examples

[22] “ENIAC: El primer computador que Marcó un Antes y un Después en la Historia de la Tecnología”. Radio Canal 95. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://www.canal95.cl/magazine/eniac-computador-marco-historia-tecnologia

[23] “Cálculo de estructuras por el Método de Elementos Finitos - EADIC”. EADIC. Accedido el 19 de enero de 2026. [Imagen].Disponible: https://eadic.com/blog/entrada/calculo-de-estructuras-por-el-metodo-de-elementos-finitos/

[24] “SWI-Prolog”. SWI-Prolog. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://www.swi-prolog.org/

[25] “Lisp”. Released Blog. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://blog.released.info/2023/12/10/Lisp.html

[26] “Lenguaje HTML y CSS”. Informatica & TIC. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://infoytic.blogspot.com/2017/12/lenguaje-html-y-css.html

[27] P. D. Francesco. “#programmingtips #codinginsights #cplusplus #developerlife #programmingwisdom #coding #developer #c #cpp #gamedevelopment #gamedev | Pino De Francesco”. LinkedIn: Log In or Sign Up. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://www.linkedin.com/posts/gdefrancesco_programmingtips-codinginsights-cplusplus-activity-7081581466500911104-pZ74/

[28] “Archivo:GCC logo.png - Guía Ubuntu”. Guía Ubuntu. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://www.guia-ubuntu.com/index.php?title=Imagen:GCC_logo.png&redirect=no

[29] Contributors to Wikimedia projects. “Python (programming language) - Wikipedia”. Wikipedia, the free encyclopedia. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://en.wikipedia.org/wiki/Python_(programming_language)

[30] P. Das. “Numpy”. LinkedIn: Log In or Sign Up. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://www.linkedin.com/pulse/numpy-piyush-das-zcinc/

[31] Contributors to Wikimedia projects. “pandas (software) - Wikipedia”. Wikipedia, the free encyclopedia. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://en.wikipedia.org/wiki/Pandas_(software)

[32] “Project Jupyter Documentation — documentación de Jupyter Documentation - 4.1.1 alpha”. Project Jupyter Documentation — Jupyter Documentation 4.1.1 alpha documentation. Accedido el 19 de enero de 2026. [Imagen]. Disponible: https://docs.jupyter.org/es/latest/

[33] “Acerca de Schneider Electric”. Schneider Electric. Accedido el 08 de marzo del 2026. [En línea]. https://www.se.com/mx/es/about-us/

[34] “Siempre en el corazón de la innovación”. Schneider Electric. Accedido el 08 de marzo del 2026. [En línea]. https://www.se.com/co/es/about-us/company-profile/history/schneider-electric-history/