Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus tempus porttitor turpis non consequat. Donec at lobortis est, non consectetur sapien. Proin non libero finibus, mattis est id, imperdiet metus. Maecenas convallis erat sit amet varius elementum. Ut tristique accumsan urna, sed congue ex sagittis ut. Nam posuere ante nibh, at consectetur arcu viverra non. Aliquam bibendum tincidunt tristique. Aenean hendrerit hendrerit massa in tincidunt. Proin sodales dolor maximus, tempus risus vitae, ornare quam. Duis at massa rhoncus, semper nisl at, sagittis felis. Morbi quis bibendum nisl, vel interdum magna.

Aliquam metus dolor, posuere sed arcu id, accumsan hendrerit quam. Pellentesque non sodales libero. Phasellus eleifend viverra aliquet. Etiam sit amet dapibus turpis. Nulla quis sollicitudin ipsum. Nam volutpat dolor in est tempor, et gravida dui sollicitudin. Duis viverra, massa nec dictum mollis, nisl libero hendrerit ex, sit amet ultricies elit ex ac quam. Sed ornare dolor vel hendrerit aliquet. Suspendisse faucibus orci id urna feugiat, nec vestibulum dui rhoncus. Quisque laoreet malesuada massa, et convallis urna. Pellentesque diam dui, congue eu aliquet eget, lacinia nec enim. Morbi pulvinar, felis vel ullamcorper porttitor, ex arcu consequat erat, efficitur consectetur turpis ex sit amet felis. Integer a est libero. Suspendisse potenti. Phasellus rhoncus fringilla euismod. Ut efficitur tortor hendrerit, tincidunt metus sit amet, consectetur enim.

Authors:

- Adrian Flores
- Andrea Ramirez

Import Libraries ⬇️

In [ ]:
```
# Data manipulation and visualization
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mticker
import ipywidgets as widgets
import plotly.graph_objs as go

# Standard libraries
import warnings
warnings.filterwarnings('ignore')

# ===== ===== Reproducibility Seed ===== =====
# Set a fixed seed for the random number generator for reproducibility
```

```
random_state = 42

# Set matplotlib inline
%matplotlib inline

# Set default figure size
plt.rcParams['figure.figsize'] = (16, 8)

# Define custom color palette
palette = ['#648FFF', '#775EF0', '#DD2680', '#FE6100', '#FFB001']

# Set the style of seaborn
sns.set_theme(style="whitegrid")
```

In [ ]:
```
# Create a slider widget to allow users to select a floating point value, la
slider = widgets.FloatSlider(description='$x$')

# Create a text widget to display the squared value of the slider's input, l
# This widget is disabled to prevent user input (it's only for displaying ou
text = widgets.FloatText(disabled=True, description='$x^2$')

# Define a function that computes the square of the slider's value
# and updates the text widget with the result
def compute(*ignore):
    text.value = str(slider.value ** 2)

# Link the slider to the compute function; this ensures that any change
# in the slider's value triggers a re-calculation of the square
slider.observe(compute, 'value')

# Set an initial value for the slider (4 in this case), which will also trig
slider.value = 4

# Display the slider and the text output in a vertical layout (VBox)
widgets.VBox([slider, text])
```

```
VBox(children=(FloatSlider(value=4.0, description='$x$'), FloatText(value=1
6.0, description='$x^2$', disabled=…
```

Data Upload

In [ ]:
```
def read_and_process_excel(file_names):
    dfs = []  # Initialize an empty list to store DataFrames

    for file_name in file_names:
        # Read the Excel file while skipping the first six rows of headers
        df = pd.read_excel(file_name, skiprows=6)

        # Drop the last three rows from the DataFrame to remove any unwanted
        df = df.iloc[:-3]

        # Convert the 'Fecha' column to datetime format
        df['Fecha'] = pd.to_datetime(df['Fecha'])

        # Set the 'Fecha' column as the index of the DataFrame
```

```
        df.set_index('Fecha', inplace=True)

        # Select only the specified columns and create a new column 'Diesel'
        df['Diesel'] = df['Diesel alto azufre'].fillna(0) + df['Diesel bajo

        # Select only the relevant columns: Gasolina regular, Gasolina super
        df = df[['Gasolina regular', 'Gasolina superior', 'Gas licuado de pe

        # Append the processed DataFrame to the list
        dfs.append(df)

    return dfs  # Return the list of DataFrames
```

In [ ]:
```
# List of Excel file names to be processed
file_names = ["consumo.xlsx", "importacion.xlsx"]
dataset_names = ["Consumo", "Importacion", "Precios"]

# Call the function to read and process the Excel files, storing the result
dataframes = read_and_process_excel(file_names)
```

In [ ]:
```
def read_price_dfs(sheetname, skip):
  # Read the Excel file while skipping the first six rows of headers
  df = pd.read_excel("precios.xlsx", skiprows=skip, sheet_name=sheetname)
  # Drop the last three rows from the DataFrame to remove any unwanted data
  df = df.iloc[1:-3]
  # Convert the 'Fecha' column to datetime format
  df['Fecha'] = pd.to_datetime(df['FECHA'])
  # Set the 'Fecha' column as the index of the DataFrame
  df.set_index('Fecha', inplace=True)
  # Remove last column
  df = df.iloc[:, :-1]
  # Rename the columns correctly
  df.rename(columns={
      'FECHA': 'Fecha',
      'Tipo de Cambio': 'Tipo de Cambio',
      'Superior': 'Gasolina superior',
      'Regular': 'Gasolina regular',
      'Diesel': 'Diesel',
      'Bunker': 'Bunker',
      'Glp Cilindro 25Lbs.': 'Gas licuado de petróleo'
  }, inplace=True)
  # Select only the relevant columns: Gasolina regular, Gasolina superior, G
  df = df[['Gasolina regular', 'Gasolina superior', 'Gas licuado de petróleo
  # Drop NaN values from the final DataFrame
  df.dropna(inplace=True)
  return df
```

In [ ]:
```
list_price_params = [("2021", 6), ("2022", 6), ("2023", 7), ("2024", 7)]
```

In [ ]:
```
# Initialize an empty list to hold DataFrames
df_list = []

# Loop through each parameter to read and append DataFrames to the list
for year, skip in list_price_params:
    df = read_price_dfs(year, skip)
```

```
    df_list.append(df)

# Concatenate all DataFrames in the list into a single DataFrame
df = pd.concat(df_list)
# Optionally, sort the index if necessary
df.sort_index(inplace=True)
# Display the final DataFrame
dataframes.append(df)
```

Exploratory Analysis 🔎

(1) Descripción General de los Datos

In [ ]:
```
for i, df in enumerate(dataframes):
  # Get the number of rows in the merged DataFrame
  rows_num = df.shape[0]
  # Print the number of records in the DataFrame
  print(f"The given dataset {dataset_names[i]} has", rows_num, "registers.")
```

```
The given dataset Consumo has 293 registers.
The given dataset Importacion has 281 registers.
The given dataset Precios has 1302 registers.
```

Observaciones 💡 -->

> - El primer conjunto de datos se centra en la recopilación de información histórica sobre el consumo nacional de petróleo y productos petroleros en Guatemala. Este conjunto abarca entradas desde enero de 2000 hasta la fecha actual. La información fue obtenida de la página oficial del Ministerio de Energía y Minas. Cuenta con alrededor de 293 registros y 4 columnas.

> - El segundo conjunto de datos se centra en la recopilación de información histórica sobre la importación de productos derivados del petróleo en Guatemala. Este conjunto abarca entradas desde enero de 2001 hasta la fecha actual. La información fue obtenida de la página oficial del Ministerio de Energía y Minas. Cuenta con alrededor de 281 registros y 4 columnas, de manera similar al conjunto anterior.

> - El tercer conjunto de datos se centra en la recopilación de información histórica sobre los precios de productos derivados del petróleo en Guatemala (GTQ/Galón). Este conjunto abarca entradas desde enero de 2021 hasta la fecha actual. La información fue obtenida de la página oficial del Ministerio de Energía y Minas. Cuenta con 1302 registros y 4 columnas.

In [ ]:
```
for df in dataframes:
  # Basic information about the dataset
  print(df.info(), "\n")
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 293 entries, 2000-01-01 to 2024-05-01
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Gasolina regular         293 non-null    float64
 1   Gasolina superior        293 non-null    float64
 2   Gas licuado de petróleo  293 non-null    float64
 3   Diesel                   293 non-null    float64
dtypes: float64(4)
memory usage: 11.4 KB
None

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 281 entries, 2001-01-01 to 2024-05-01
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Gasolina regular         281 non-null    float64
 1   Gasolina superior        281 non-null    float64
 2   Gas licuado de petróleo  281 non-null    float64
 3   Diesel                   281 non-null    float64
dtypes: float64(4)
memory usage: 11.0 KB
None

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1302 entries, 2021-01-01 to 2024-07-28
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Gasolina regular         1302 non-null   object
 1   Gasolina superior        1302 non-null   object
 2   Gas licuado de petróleo  1302 non-null   object
 3   Diesel                   1302 non-null   object
dtypes: object(4)
memory usage: 50.9+ KB
None
```

Observaciones 💡 -->

> - En este análisis observamos que no es necesario realizar alteraciones en los tipos de las variables en ninguno de los conjuntos de datos, ya que todas están definidas de manera adecuada. Es relevante destacar que, aunque no se observan valores nulos, existe la posibilidad de que algunos de ellos estén codificados de manera diferente. Por lo tanto, es importante tener esto en cuenta durante el proceso de limpieza. Sin embargo, a medida que se avance en el análisis exploratorio, se podrá

> obtener información más detallada al respecto.

(2) Clasificación de las Variables

Conjunto de Datos de Consumo -->

Nombre	Descripción	Clasificación
Fecha	Fecha de registro de los datos de consumo de combustibles.	Categórica
Gasolina regular	Consumo nacional de gasolina regular en barriles de 42 galones.	Numérica
Gasolina superior	Consumo nacional de gasolina superior en barriles de 42 galones.	Numérica
Gas licuado de petróleo	Consumo nacional de gas licuado de petróleo en barriles de 42 galones.	Numérica
Diesel	Consumo nacional de diesel en barriles de 42 galones.	Numérica

Conjunto de Datos de Importación -->

Nombre	Descripción	Clasificación
Fecha	Fecha de registro de la importación de productos derivados del petróleo.	Categórica
Gasolina regular	Importación nacional de gasolina regular en barriles de 42 galones.	Numérica
Gasolina superior	Importación nacional de gasolina superior en barriles de 42 galones.	Numérica
Gas licuado de petróleo	Imporatación nacional de gas licuado de petróleo en barriles de 42 galones.	Numérica
Diesel	Importación nacional de diesel en barriles de 42 galones.	Numérica

Conjunto de Datos de Precio -->

Nombre	Descripción	Clasificación
Fecha	Fecha de registro del precio en quetzales.	Categórica
Gasolina regular	Precio de gasolina regular en quetzales por galón.	Numérica
Gasolina superior	Precio de gasolina superior en quetzales por galón.	Numérica
Gas licuado de petróleo	Precio del gas licuado de petróleo en quetzales por galón.	Numérica
Diesel	Precio del diesel en quetzales por galón.	Numérica

(3) Exploración y Limpieza Inicial de los Datos

🖊️ Modificación de Etiquetas de Variables -->

Para facilitar la comprensión y el manejo del conjunto de datos, se procederá a modificar los nombres de las variables. Este cambio permitirá una organización más clara y una interpretación más precisa de la información.

In [ ]:
```
# Dictionary to rename columns for better readability
rename_col = {
    'Gasolina regular': 'gasoline_regular',    # Renaming 'Gasolina regular'
    'Gasolina superior': 'gasoline_superior',  # Renaming 'Gasolina superior
    'Gas licuado de petróleo': 'liquefied_gas', # Renaming 'Gas licuado de p
    'Diesel': 'diesel'                         # Renaming 'Diesel' to 'diesel
}
```

In [ ]:
```
for i, df in enumerate(dataframes):
  # Use a pandas function to rename the current function
  df = df.rename(columns = rename_col)
  # Change the index name from 'Fecha' to 'date'
  df.rename_axis('date', inplace=True)
  # Ensure all columns are numeric
  df = df.astype('float64')
  # Save changes
  dataframes[i] = df
  print(df.head(2), "\n")
```

```
            gasoline_regular  gasoline_superior  liquefied_gas     diesel
date
2000-01-01         202645.20          308156.82  194410.476190  634667.06
2000-02-01         205530.96          307766.31  174710.552381  642380.66

            gasoline_regular  gasoline_superior  liquefied_gas     diesel
date
2001-01-01         177776.50          373963.96  194065.738095  566101.99
2001-02-01         123115.99          243091.07  170703.380952  489525.80

            gasoline_regular  gasoline_superior  liquefied_gas  diesel
date
2021-01-01             21.11              21.91           99.0   17.61
2021-01-02             21.11              21.91           99.0   17.61
```

(4) Análisis Visual Preliminar de los Datos

In [ ]:
```
# Combine the DataFrames for the interactive selection
dataframe_mapping = {
    "Consumption": dataframes[0],
    "Importation": dataframes[1],
    "Pricing": dataframes[2]
}
```

(1) Visualización de Series de Tiempo

In [ ]:
```
# Define a mapping of dataset names to y-axis labels
y_label_mapping = {
    "Consumption": "Consumption (Barrel, 42 gallons)",
    "Importation": "Importation (Barrel, 42 gallons)",
    "Pricing": "Price (GTQ/Gal)"
}

# Create a dropdown widget for selecting the DataFrame
df_selector = widgets.Dropdown(
    options=list(dataframe_mapping.keys()),
    description='Dataset:',
)

# Function to plot the selected DataFrame
def plot_dataframe(name):
    # Get the DataFrame based on the name
    df = dataframe_mapping.get(name, None)  # Default to None if the name is

    # Create the line plots
    sns.lineplot(data=df, x=df.index, y='gasoline_regular', label='Gasoline
    sns.lineplot(data=df, x=df.index, y='gasoline_superior', label='Gasoline
    sns.lineplot(data=df, x=df.index, y='liquefied_gas', label='Liquefied Ga
    sns.lineplot(data=df, x=df.index, y='diesel', label='Diesel', color=pale

    # Add titles and labels with improved formatting
    plt.title(f'Trends in Petroleum Product {name} in Guatemala', fontsize=1
    plt.xlabel('Date', fontsize=14)

    # Set the y-axis label based on the selected dataset
    y_label = y_label_mapping.get(name, 'Price')
    plt.ylabel(y_label, fontsize=14)

    plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
    plt.yticks(fontsize=12)  # Increase font size for y-ticks
    plt.legend(title='Fuel Type', fontsize=12)

    # Set y-axis tick formatting to include thousand separators
    ax = plt.gca()  # Get the current axis
    ax.yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))  #

    plt.grid(visible=True, linestyle='--', alpha=0.7)  # Add a grid for bett
    plt.tight_layout()  # Adjust layout to make room for rotated x-axis labe
    plt.show()
```

In [ ]:
```
# Create an interactive output using the df_selector
widgets.interactive(plot_dataframe, name=df_selector)
```

```
interactive(children=(Dropdown(description='Dataset:', options=('Consumptio
n', 'Importation', 'Pricing'), valu…
```

(2) Top Distribución por Mes o Año

In [ ]:
```
# Function to plot maximums interactively
def top_months_interactive(name, detail_level):
    # Create a new DataFrame that aggregates data by the chosen detail level
```

```
    if detail_level == 'Monthly':
        # Aggregate by month and get max for each month
        monthly_max = dataframe_mapping[name].groupby(dataframe_mapping[name
    elif detail_level == 'Yearly':
        # Aggregate by year and get max for each year
        yearly_max = dataframe_mapping[name].resample('Y').max()
        # If you want to display only the years as labels
        monthly_max = yearly_max

    # Plotting
    monthly_max.plot(kind='bar', color=palette, edgecolor='black', width=0.8

    # Set proper names for the legend
    fuel_labels = ['Gasoline Regular', 'Gasoline Superior', 'Liquefied Gas',

    # Formatting the plot
    plt.title(f'Maximum Fuel {name} ({detail_level})', fontsize=16, fontweig
    if detail_level == 'Monthly':
        plt.xlabel('Month', fontsize=14)
        plt.xticks(ticks=range(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', '
    else:
        plt.xlabel('Year', fontsize=14)
        plt.xticks(ticks=range(len(monthly_max.index)), labels=monthly_max.i

    plt.ylabel(f'Maximum {name}', fontsize=14)
    plt.legend(fuel_labels, title='Fuel Type', fontsize=12)

    # Set y-axis tick formatting to include thousand separators
    ax = plt.gca()  # Get the current axis
    ax.yaxis.set_major_formatter(mticker.StrMethodFormatter('{x:,.0f}'))  #

    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()
```

In [ ]:
```
# Use the interactive function with the dropdowns
df_selector = widgets.Dropdown(options=dataframe_mapping.keys(), description
detail_selector = widgets.Dropdown(options=['Monthly', 'Yearly'], descriptio

# Create interactive widgets
widgets.interactive(top_months_interactive, name=df_selector, detail_level=d
```

```
interactive(children=(Dropdown(description='Dataset:', options=('Consumptio
n', 'Importation', 'Pricing'), valu…
```

(3) Gráficos Interactivos de Distribución

In [ ]:
```
def interactive_histogram(df, name, units):
    # Create a dropdown to select the fuel type
    fuel_dropdown = widgets.Dropdown(
        options=[
            ('Gasoline Regular', 'gasoline_regular'),
            ('Gasoline Superior', 'gasoline_superior'),
            ('Liquefied Gas', 'liquefied_gas'),
            ('Diesel', 'diesel')
        ],
```

```
        value='diesel',  # Default value
        description='Fuel Type:',
    )

    # Create an empty figure widget for dynamic updates
    fig = go.FigureWidget()

    # Initialize the plot with the default fuel type
    trace = fig.add_trace(go.Histogram(
        x=df['diesel'],
        name='Diesel',
        marker=dict(color=palette[0])
    ))

    # Set up the initial layout
    fig.update_layout(
        title=f'Diesel {name} Distribution',
        xaxis_title=f'{name} {units}',
        yaxis_title='Count',
        hovermode='x unified'
    )

    # Display the dropdown and the figure
    display(fuel_dropdown)
    display(fig)

    # Function to update the chart when the fuel type is changed
    def update_chart(change):
        fuel_type = change['new']

        # Update the trace data
        with fig.batch_update():
            fig.data[0].x = df[fuel_type]
            fig.data[0].name = fuel_type.replace('_', ' ').capitalize()
            fig.data[0].marker.color = palette[0]

        # Update the layout with the new title and x-axis label
        fig.update_layout(
            title=f'{fuel_type.replace("_", " ").title()} {name} Distributio
            xaxis_title=f'{name} {units}'
        )

    # Set up an observer to call update_chart when the dropdown value change
    fuel_dropdown.observe(update_chart, names='value')
```

In [ ]:
```
interactive_histogram(dataframes[0], "Consumption", "(Barrel, 42 gallons)")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
```

```
FigureWidget({
    'data': [{'marker': {'color': '#648FFF'},
              'name': 'Diesel',
              'type': 'histogram',
              'uid': '7ff3dfc4-7859-4a84-a7b2-0bd0547fb6a1',
              'x': array([ 634667.06,  642380.66,  699807.25, ..., 1393324.5
2, 1428143.44,
                           1401052.37])}],
    'layout': {'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Consumption Distribution'},
               'xaxis': {'title': {'text': 'Consumption (Barrel, 42 gallon
s)'}},
               'yaxis': {'title': {'text': 'Count'}}}
})
```

In [ ]:
```
interactive_histogram(dataframes[1], "Importation", "(Barrel, 42 gallons)")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
FigureWidget({
    'data': [{'marker': {'color': '#648FFF'},
              'name': 'Diesel',
              'type': 'histogram',
              'uid': '4b9a3ab4-d3ca-4cc1-b40d-bb0bbe9a6300',
              'x': array([ 566101.99,  489525.8 ,  575559.68, ..., 1477038.
, 1294706.12,
                           1470870.09])}],
    'layout': {'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Importation Distribution'},
               'xaxis': {'title': {'text': 'Importation (Barrel, 42 gallon
s)'}},
               'yaxis': {'title': {'text': 'Count'}}}
})
```

In [ ]:
```
interactive_histogram(dataframes[2], "Pricing", "(GTQ/Gal)")
```

```
Dropdown(description='Fuel Type:', index=3, options=(('Gasoline Regular', 'g
asoline_regular'), ('Gasoline Supe…
FigureWidget({
    'data': [{'marker': {'color': '#648FFF'},
              'name': 'Diesel',
              'type': 'histogram',
              'uid': '9f374efd-a504-4fef-9a00-cf2a274a32c3',
              'x': array([17.61, 17.61, 17.61, ..., 28.09, 28.09, 28.09])}],
    'layout': {'hovermode': 'x unified',
               'template': '...',
               'title': {'text': 'Diesel Pricing Distribution'},
               'xaxis': {'title': {'text': 'Pricing (GTQ/Gal)'}},
               'yaxis': {'title': {'text': 'Count'}}}
})
```

(4) Comparación Entre Modelos