

Tarea 8.2

Esta labor consiste en la elaboración de un informe de Power BI, aprovechando las capacidades analíticas de Python. Se utilizarán los scripts de Python creados previamente en la Tarea 1 para generar visualizaciones personalizadas con las bibliotecas Seaborn y Matplotlib. Estas visualizaciones estarán integradas en el informe de Power BI para ofrecer una comprensión más profunda de la capacidad del lenguaje de programación en la herramienta Power BI.

- Objetivos:
 - ✓ Integrar los scripts de Python desarrollados en la Tarea 1 con Power BI para la creación de visualizaciones avanzadas.
 - ✓ Documentar cada paso del proceso de creación del informe con scripts para facilitar la reproducibilidad y mantenimiento.
- Entrega
 - ✓ Almacena en un repositorio de tu GitHub una carpeta que contenga:
 - ✓ El archivo .pbix de Power BI con el informe solicitado.
 - ✓ Un PDF del informe exportado.
 - ✓ Un archivo .ipynb de Jupyter Notebook que contenga cada script correspondiente a cada visualización del informe junto al enunciado correspondiente.
 - ✓ En la entrega, coloca el link de la carpeta.

Nivel 1

Los 7 ejercicios del nivel 1 de la tarea 8.1

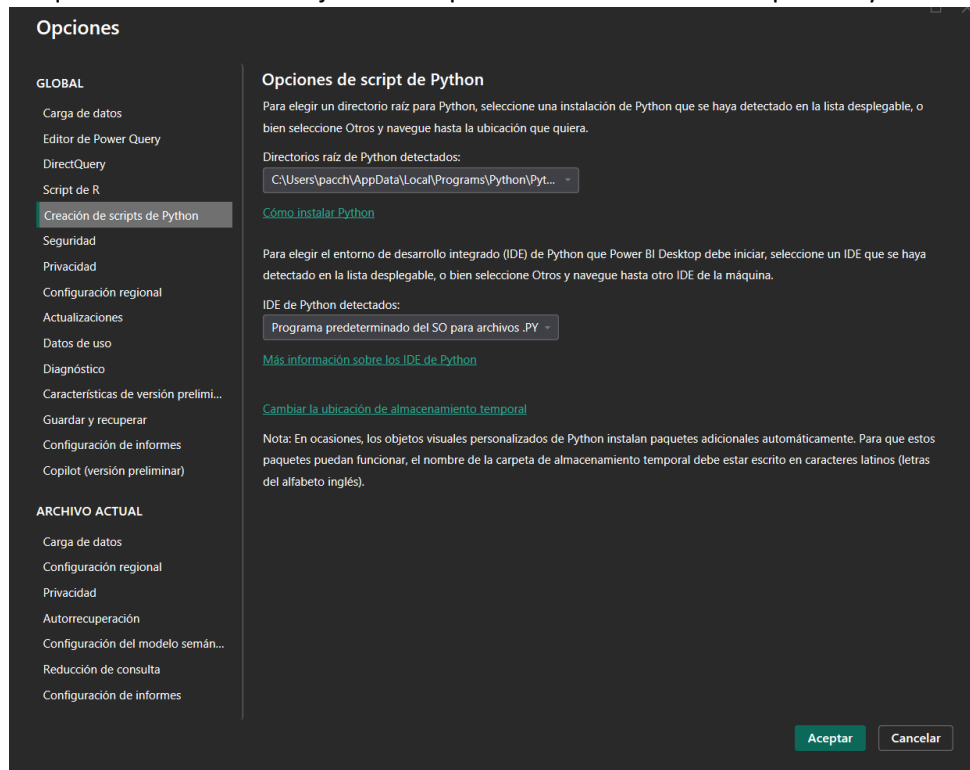
Nivel 2

Los 2 ejercicios del nivel 2 de la tarea 8.1

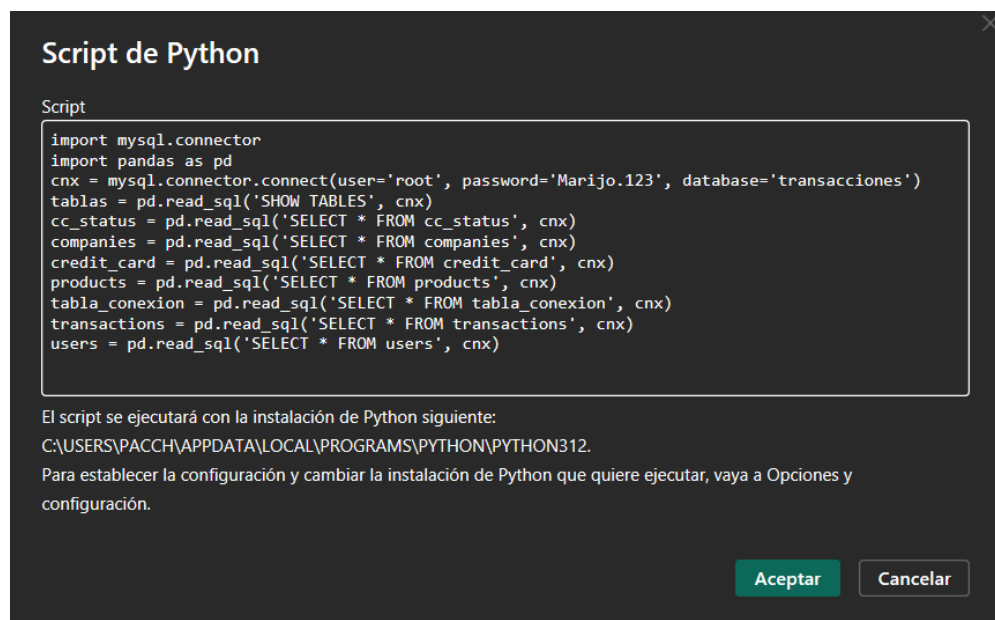
Nivel 3

Los 2 ejercicios del nivel 3 de la tarea 8.1

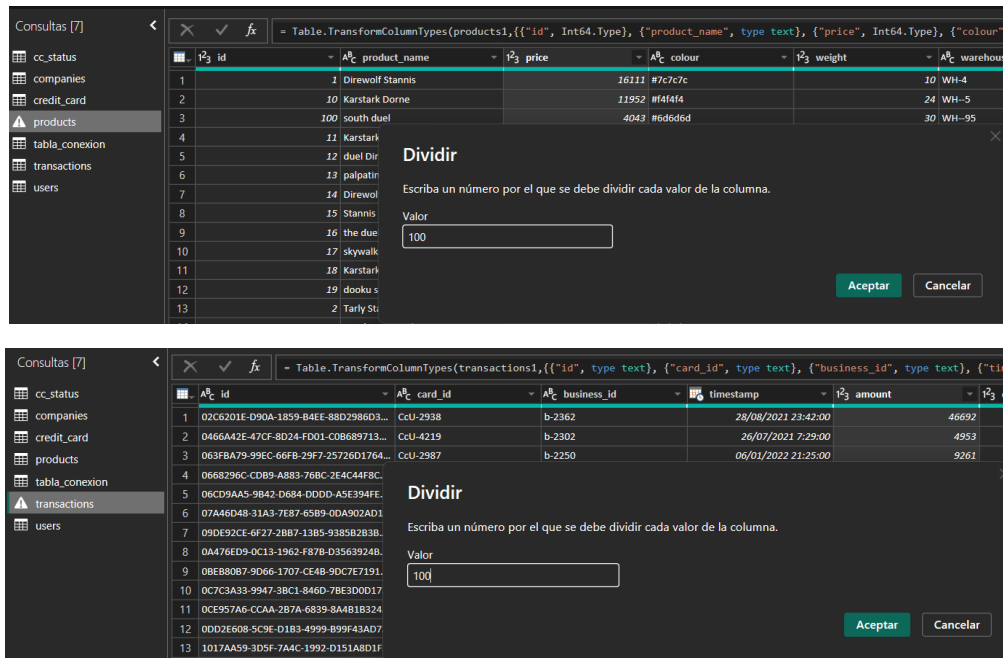
Lo primero que he hecho ha sido ajustar las opciones de Creación de scripts de Python en power bi



Luego en las opciones de obtener datos seleccioné “script de Python” y escribí el código de importación de la base de datos:



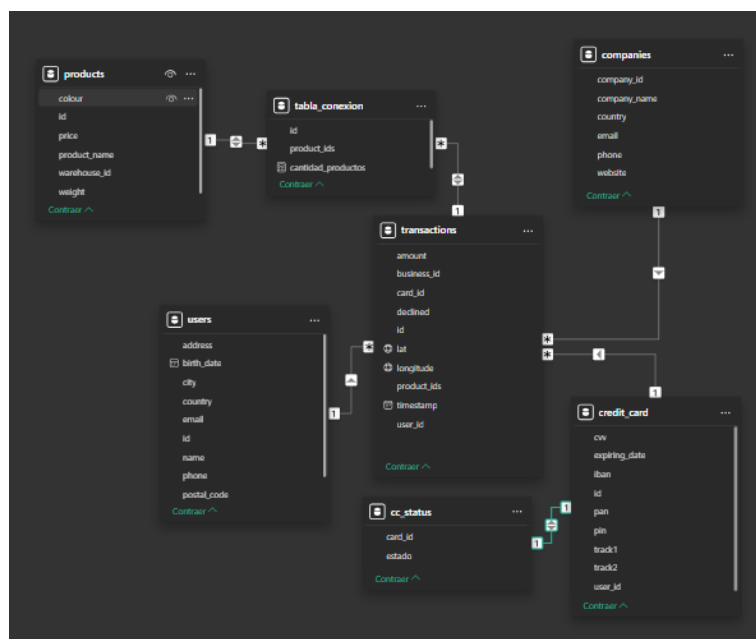
Luego seleccioné las tablas correspondientes y verifiqué que los datos estuvieran en el formato correcto. Encontré algunos errores los cuales solucioné en la sección de transformación de datos, para la tabla products y transactions fue necesario modificar la columna 'price', dividiendo el monto entre 100 para obtener los datos correctos:



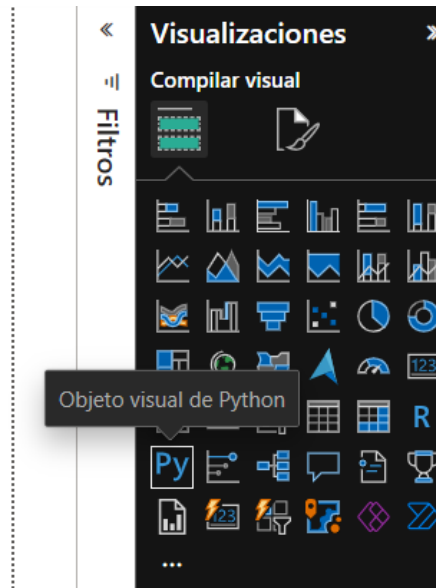
The first screenshot shows the Databricks SQL interface with a query editor. The query is: `Table.TransformColumnTypes(products1,({{"id", Int64.Type}, {"product_name", type text}, {"price", Int64.Type}, {"colour", type text}, {"weight", Int64.Type}, {"warehouse", type text}))`. A dialog box titled "Dividir" (Divide) is open, asking for a value to divide the 'price' column by. The value "100" is entered in the input field.

The second screenshot shows the same interface but with a query for the 'transactions' table: `Table.TransformColumnTypes(transactions1,({{"id", type text}, {"card_id", type text}, {"business_id", type text}, {"timestamp", type text}, {"amount", type text}, {"declined", type text}))`. A similar "Dividir" dialog box is open, with the value "100" entered.

Seguidamente configuré las conexiones de las tablas hasta tenerlas de la siguiente manera:



Luego de tener los datos listos, pasé a crear los gráficos que ya tenía hechos en Visual Studio Code, seleccionando la opción de “Objeto visual de Python” en los tipos de visualizaciones:



Luego seleccioné las columnas que usaría en cada gráfico y finalmente copie y pegué el código, modificando la fuente de los datos, que para este caso se modifica por 'dataset', de la siguiente manera:

```
Editor de scripts de Python

⚠ Las filas duplicadas se quitarán de los datos.

4 # dataset = pandas.DataFrame(price)
5 # dataset = dataset.drop_duplicates()
6
7 # Pegue o escriba aquí el código de script:
8 import seaborn as sns
9 import matplotlib.pyplot as plt
10
11 sns.set_theme()
12 plot = sns.displot(
13     data=dataset,
14     x='price',
15     color='#5865F2')
16
17 plt.title("Distribución de Precios(1)", fontsize=18)
18 plt.xlabel("Precio", fontsize=12)
19 plt.ylabel("Frecuencia", fontsize=12)
20 plot.fig.tight_layout(rect=[0, 0, 1, 0.95])
21 plt.show()
```

Y seguí el mismo proceso para los demás gráficos, teniendo como resultado los siguientes gráficos:

