Juan José Gallego – Juan José Hernández – Nicolas Rojas

Explicación del proceso y las transformaciones realizadas.

```
import pandas as pd
import psycopg2
from sqlalchemy import create_engine
```
✓ 5.2s

```
# Extracción: Cargar datos desde el archivo movies.csv
print("Extrayendo datos del CSV...")
csv_file = "proyecto.csv"  # Ruta del archivo CSV
df = pd.read_csv(csv_file)
print("Datos extraídos correctamente.")
df.head()  # Muestra las primeras filas para verificar
```
✓ 0.3s

Primero se importa pandas, psycopg2, create_engine. Luego se extrae el csv file llamado proyecto.csv.

```
df.rename(columns={'price': 'total_price'}, inplace=True)
df.rename(columns={'category': 'product'}, inplace=True)
df
```

Se hace el debido cambio de nombre en las columnas debido a un error encontrado en el nombre y cambiamos el nombre category para un mejor manejo.

```
product = df[['product']]
product['unit_price'] = df['total_price'] / df['quantity']
product = product.copy().drop_duplicates(subset=['product']).reset_index(drop=True)
product['product_id'] = product.index + 1
df = df.merge(product, on='product', how='left')
df = df.drop(columns=['product','unit_price'])
df


product
```

Se crea la tabla “product” con la columna de df llamada igual, se define la columna “unit_price” haciendo la división del precio total con las cantidades para definir el precio unitario. Luego se eliminan los productos duplicados y se le agrega un identificador. Después se hace la unión de la tabla product y df para luego eliminar el “produc”t y “unit_price” y que así solo quede el “product_id” con el resto de la tabla.

```
date = df[['invoice_date']]
date
```

Se extrae la columna invoice_date de df y se crea un nuevo DataFrame llamado date

```
date [['day','month','year']] = date['invoice_date'].str.split('/',expand=True)
date
```

Se divide la columna invoice_date en 3 nuevas columnas que son ‘day’,’month’,year’. Esto se hace tomando cada una de las partes que estan separadas por “/”.

```
date = date.drop_duplicates(subset=['day','month','year']).reset_index(drop=True)
date['date_id'] = date.index + 1
date
```

Aquí se eliminan las filas duplicadas, luego se restablecen los índices y se crea una nueva columna llamada date_id que asigna un número único a cada fila comenzando desde 1.

```
df = df.merge(date, on='invoice_date', how='left')
df
```

Esto combina el DataFrame df con date y se hace con la finalidad de añadir la columna date_id a df.

```
df = df.drop(columns=['invoice_date','day','month','year'])
df
```

Se eliminan las columnas invoice_date, day, month, year, del DataFrame df

```
date = date.drop(columns=['invoice_date'])
date
```

Se elimina la columna invoice_date del DataFrame date.

```
shopping_mall = df[['shopping_mall']]
shopping_mall = shopping_mall.drop_duplicates(subset=['shopping_mall']).reset_index(drop=True)
shopping_mall['shopping_mall_id'] = shopping_mall.index + 1
shopping_mall
```

Se crea un nuevo DataFrame con la columna shopping_mall que se encuentra en df, posteriormente se eliminan los duplicados, se resetean los indices, se añade una nueva columna llamada shopping_mall_id con un id unico.

```
df = df.merge(shopping_mall, on='shopping_mall', how='left')
df = df.drop(columns=['shopping_mall'])
df
```

Se combina df con shopping_mall y se elimina la columna del DataFrame df llamada shopping_mall con el fin de dejar en df la columna shopping_mall_id.

```
client = df[['customer_id','gender','age']].copy()
client
```

Se crea un nuevo DataFrame con la columnas customer_id, gender, age pertenecientes al DataFrame df.

```
df = df.drop(columns=['gender','age'])
df
```

Se eliminan las columnas gender y age del DataFrame df.

Se crea la tabla “payment_method” con la columna de df, se eliminan los duplicados y se crea su debido id: Luego se hace la unión con la tabla de hechos y se elimina la columna “payment_method” para que solo quede su identificador.

```
payment_method = df[['payment_method']]
payment_method = payment_method.drop_duplicates(subset=['payment_method']).reset_index(drop=True)
payment_method['payment_id'] = payment_method.index + 1
payment_method
```

✓ 0.0s

	payment_method	payment_id
0	Credit Card	1
1	Debit Card	2
2	Cash	3

```
df = df.merge(payment_method, on='payment_method', how='left')
df = df.drop(columns=['payment_method'])
df
```

```
from sqlalchemy import create_engine
```

✓ 0.0s

```
engine = create_engine("postgresql://neondb_owner:npg_CEr1lwtZ8fnR@ep-flat-dust-a5ouuo41-pooler.us-east-2.aws.neon.tech/neondb?sslmode=require")
```

✓ 0.1s

```
df.to_sql('invoice', engine, if_exists='replace', index=False)
product.to_sql('product', engine, if_exists='replace', index=False)
date.to_sql('date', engine, if_exists='replace', index=False)
shopping_mall.to_sql('shopping_mall', engine, if_exists='replace', index=False)
payment_method.to_sql('payment_method', engine, if_exists='replace', index=False)
client.to_sql('client', engine, if_exists='replace', index=False)
```

Se utiliza la URL dada por Neon Serveless Postgres y luego se cargan las tablas con sus debidos nombres.

Comprobación de que los datos han sido correctamente insertados en la base de datos.

```
1  select * from invoice
```

Data Output | Messages | Notifications

	invoice_no text	customer_id text	quantity bigint	total_price double precision	product_id bigint	date_id bigint	shopping_mall_id bigint	payment_id bigint
1	I138884	C241288	5	1500.4	1	1	1	1
2	I317333	C111565	3	1800.51	2	2	2	2
3	I127801	C266599	1	300.08	1	3	3	3
4	I173702	C988172	5	3000.85	2	4	4	1

```
1  select * from client
```

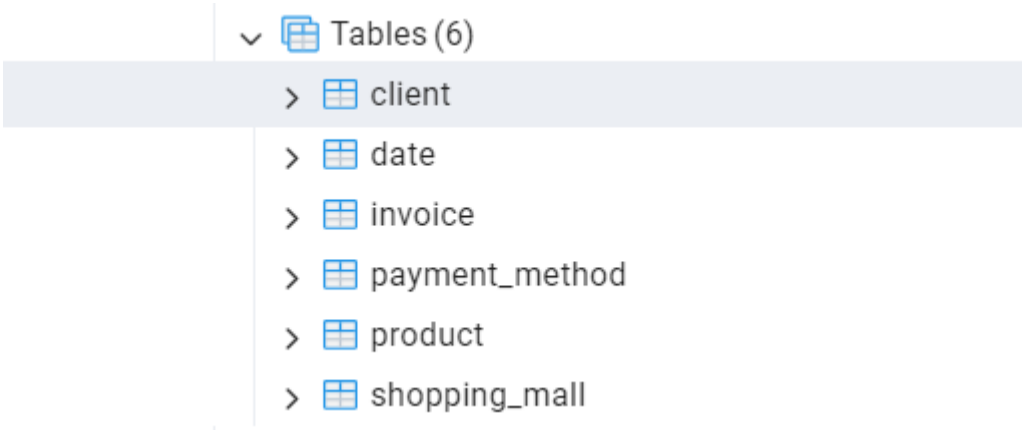
Data Output | Messages | Notifications

	customer_id text	gender text	age bigint
1	C241288	Female	28
2	C111565	Male	21
3	C266599	Male	20

```
1   select * from shopping_mall
```

Data Output | Messages | Notifications

	shopping_mall text	shopping_mall_id bigint
1	Kanyon	1
2	Forum Istanbul	2
3	Metrocity	3
4	Metropol AVM	4
5	Istinye Park	5
6	Mall of Istanbul	6
7	Emaar Square Mall	7
8	Cevahir AVM	8
9	Viaport Outlet	9
10	Zorlu Center	10




```
1   select * from payment_method
```

Data Output | Messages | Notifications

	payment_method text	payment_id bigint
1	Credit Card	1
2	Debit Card	2
3	Cash	3

```
1  select * from product
```

Data Output | Messages | Notifications

	product text	unit_price double precision	product_id bigint
1	Clothing	300.08000000000004	1
2	Shoes	600.17	2
3	Books	15.15	3
4	Cosmetics	40.66	4
5	Food & Beverage	5.23	5
6	Toys	35.84	6
7	Technology	1050	7
8	Souvenir	11.73	8

```
1  select * from date
```

Data Output | Messages | Notifications

	day text	month text	year text	date_id bigint
1	5	8	2022	1
2	12	12	2021	2
3	9	11	2021	3
4	16	05	2021	4
5	24	10	2021	5