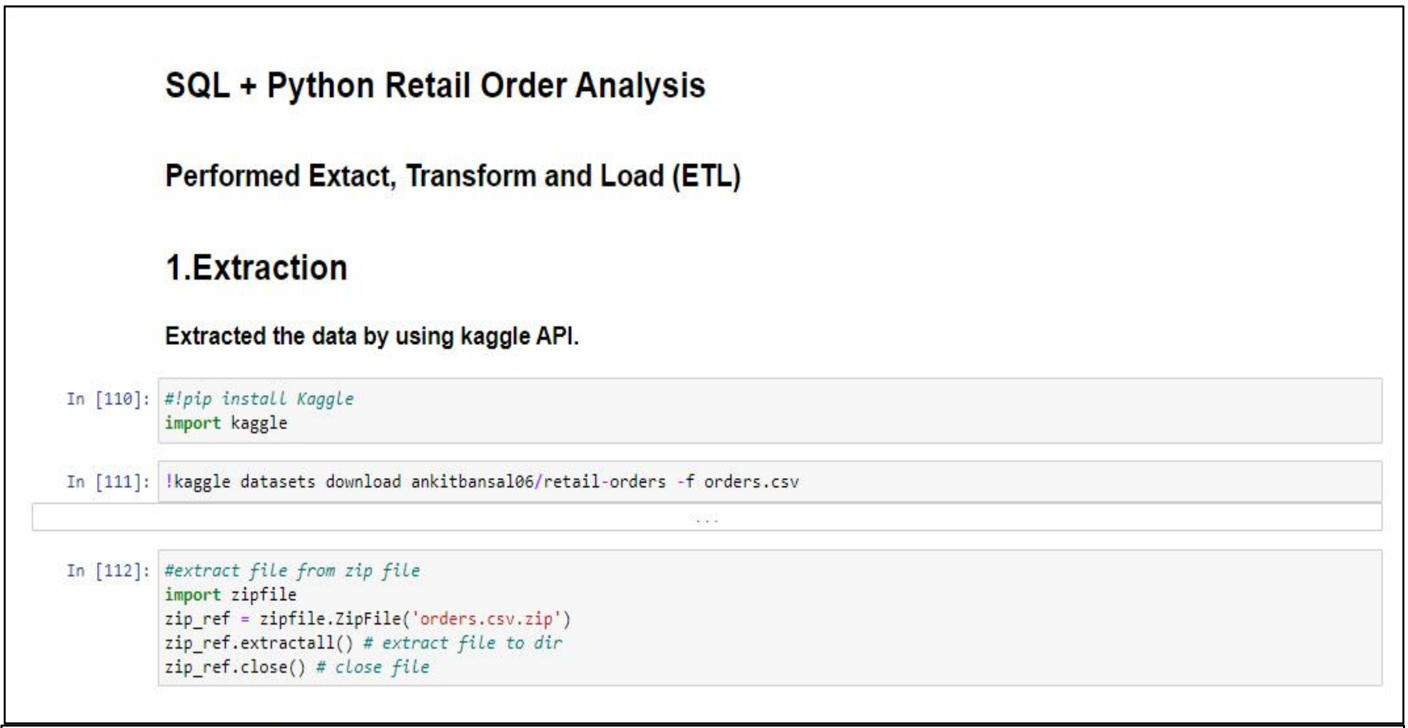


SQL + Python Retail Order Analysis

Performed Extact, Transform and Load (ETL)

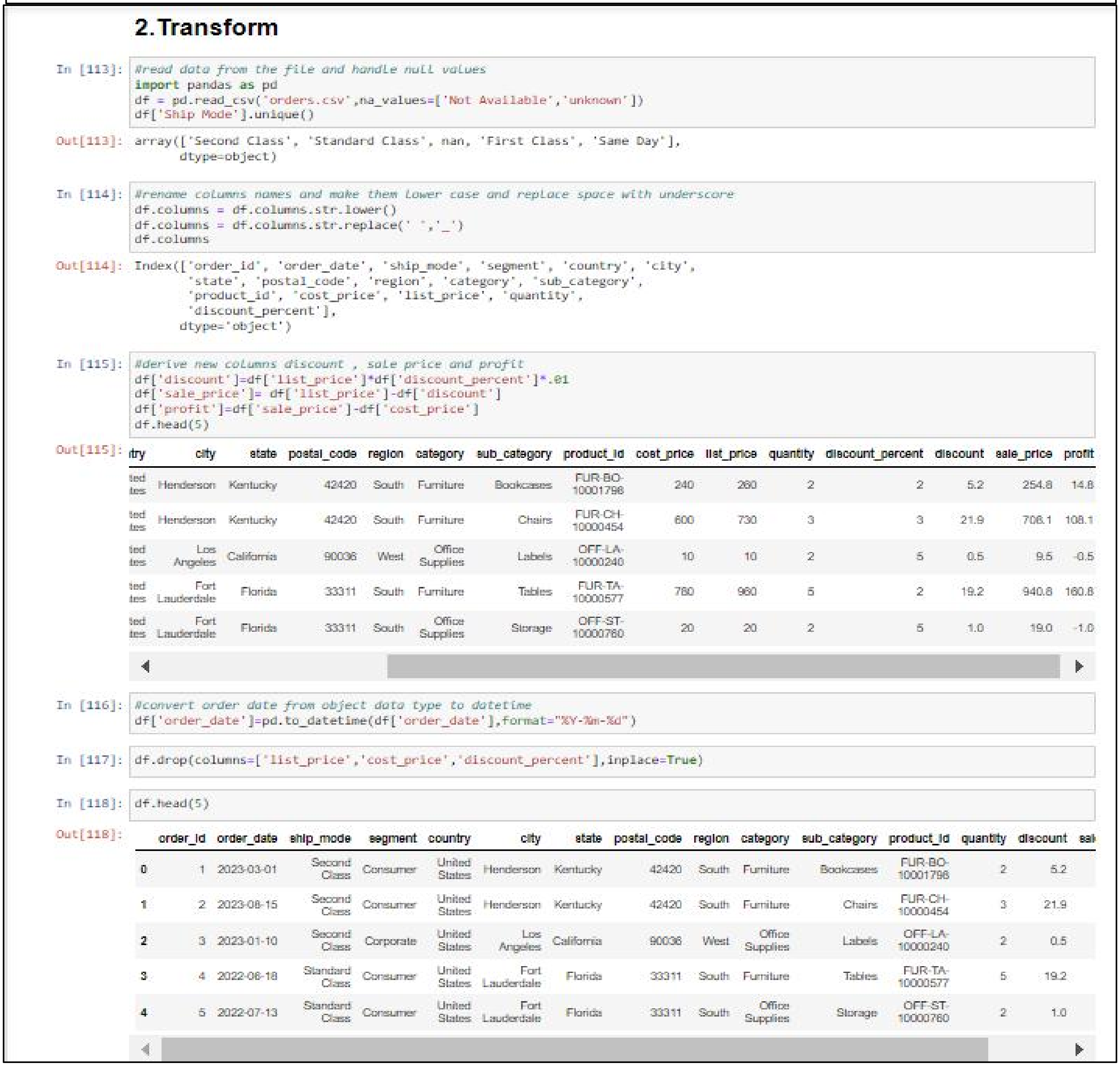
1.Extraction

Extracted the data by using kaggle API.

```
In [110]: #!pip install Kaggle
          import kaggle
```

```
In [111]: !kaggle datasets download ankitbansal06/retail-orders -f orders.csv
```

```
In [112]: #extract file from zip file
          import zipfile
          zip_ref = zipfile.ZipFile('orders.csv.zip')
          zip_ref.extractall() # extract file to dir
          zip_ref.close() # close file
```



2.Transform

```
In [113]: #read data from the file and handle null values
          import pandas as pd
          df = pd.read_csv('orders.csv',na_values=['Not Available','unknown'])
          df['Ship Mode'].unique()

Out[113]: array(['Second Class', 'Standard Class', nan, 'First Class', 'Same Day'],
                dtype=object)
```

```
In [114]: #rename columns names and make them lower case and replace space with underscore
          df.columns = df.columns.str.lower()
          df.columns = df.columns.str.replace(' ','_')
          df.columns

Out[114]: Index(['order_id', 'order_date', 'ship_mode', 'segment', 'country', 'city',
                 'state', 'postal_code', 'region', 'category', 'sub_category',
                 'product_id', 'cost_price', 'list_price', 'quantity',
                 'discount_percent'],
                dtype='object')
```

```
In [115]: #derive new columns discount , sale price and profit
          df['discount']=df['list_price']*df['discount_percent']*.01
          df['sale_price']= df['list_price']-df['discount']
          df['profit']=df['sale_price']-df['cost_price']
          df.head(5)
```

Out[115]:

itry	city	state	postal_code	region	category	sub_category	product_id	cost_price	list_price	quantity	discount_percent	discount	sale_price	profit
ted tes	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2	2	5.2	254.8	14.8
ted tes	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3	3	21.9	708.1	108.1
ted tes	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2	5	0.5	9.5	-0.5
ted tes	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5	2	19.2	940.8	160.8
ted tes	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	20	20	2	5	1.0	19.0	-1.0

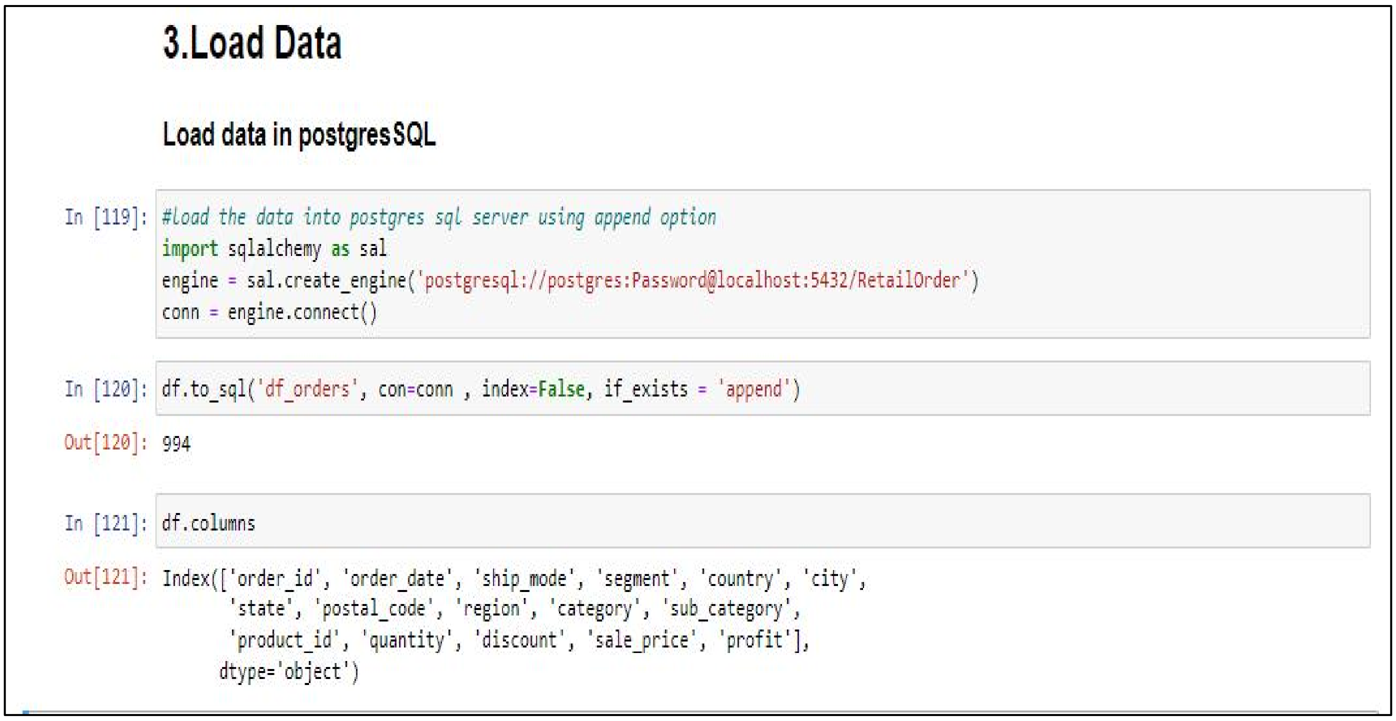
```
In [116]: #convert order date from object data type to datetime
          df['order_date']=pd.to_datetime(df['order_date'],format="%Y-%m-%d")
```

```
In [117]: df.drop(columns=['list_price','cost_price','discount_percent'],inplace=True)
```

```
In [118]: df.head(5)
```

Out[118]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	quantity	discount	sal
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	2	5.2	
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	3	21.9	
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	2	0.5	
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	5	19.2	
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	2	1.0	



3.Load Data

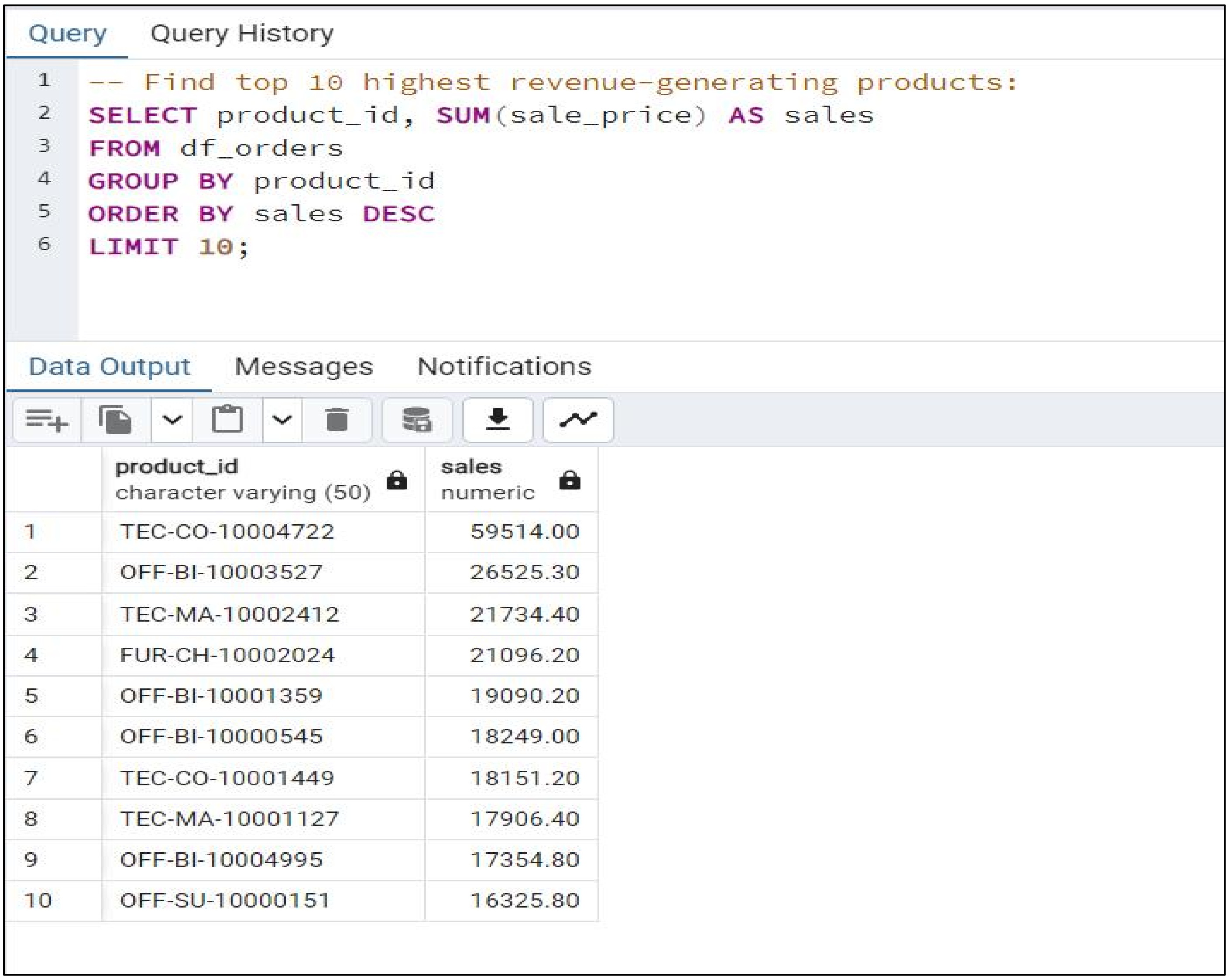
Load data in postgresSQL

```
In [119]: #load the data into postgres sql server using append option
          import sqlalchemy as sal
          engine = sal.create_engine('postgresql://postgres:Password@localhost:5432/RetailOrder')
          conn = engine.connect()
```

```
In [120]: df.to_sql('df_orders', con=conn , index=False, if_exists = 'append')

Out[120]: 994
```

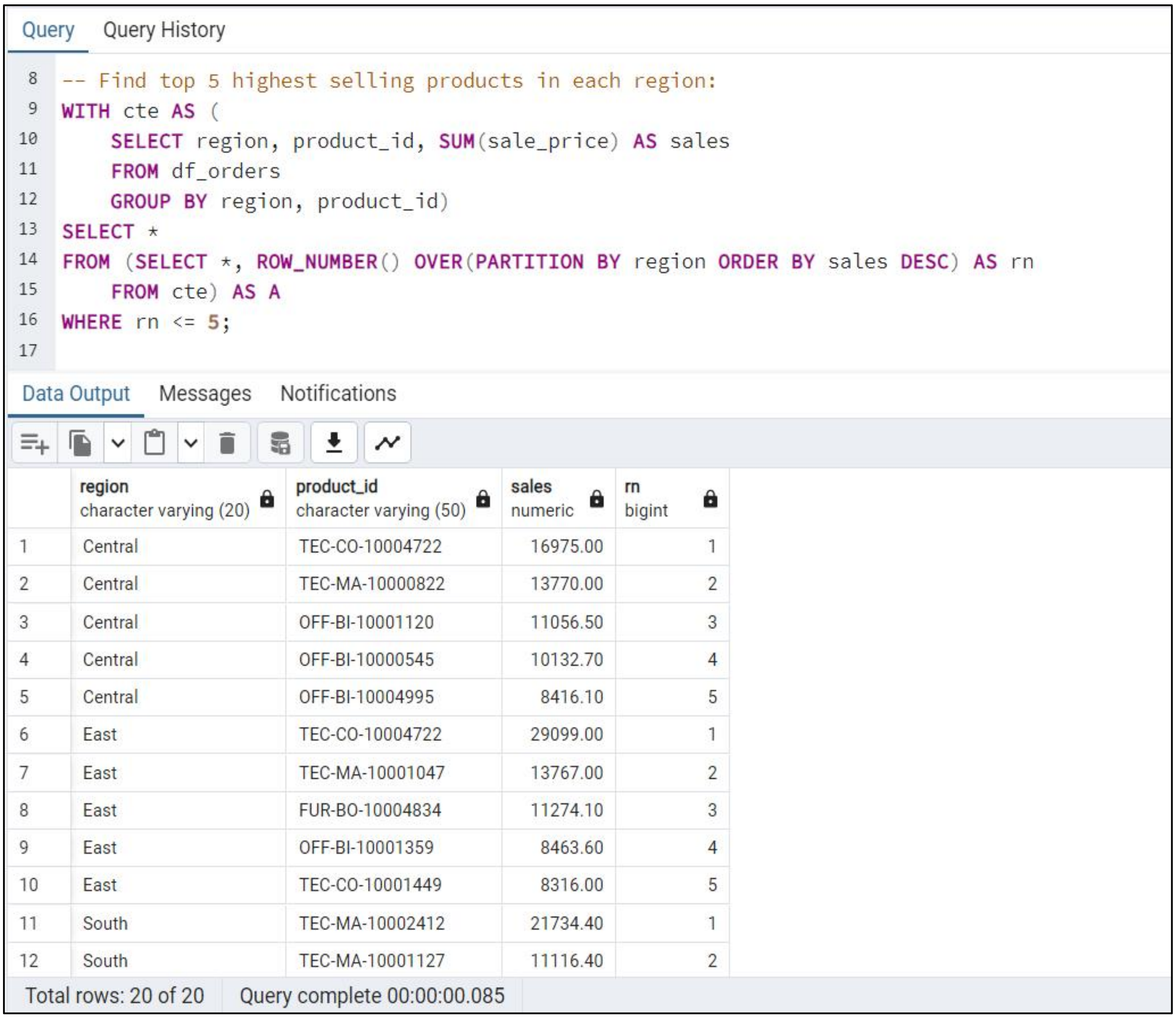
```
In [121]: df.columns

Out[121]: Index(['order_id', 'order_date', 'ship_mode', 'segment', 'country', 'city',
                 'state', 'postal_code', 'region', 'category', 'sub_category',
                 'product_id', 'quantity', 'discount', 'sale_price', 'profit'],
                dtype='object')
```

Performed queries in PostgresSQL to get Insights form available data



```
-- Find top 10 highest revenue-generating products:
SELECT product_id, SUM(sale_price) AS sales
FROM df_orders
GROUP BY product_id
ORDER BY sales DESC
LIMIT 10;
```

	product_id character varying (50)	sales numeric
1	TEC-CO-10004722	59514.00
2	OFF-BI-10003527	26525.30
3	TEC-MA-10002412	21734.40
4	FUR-CH-10002024	21096.20
5	OFF-BI-10001359	19090.20
6	OFF-BI-10000545	18249.00
7	TEC-CO-10001449	18151.20
8	TEC-MA-10001127	17906.40
9	OFF-BI-10004995	17354.80
10	OFF-SU-10000151	16325.80



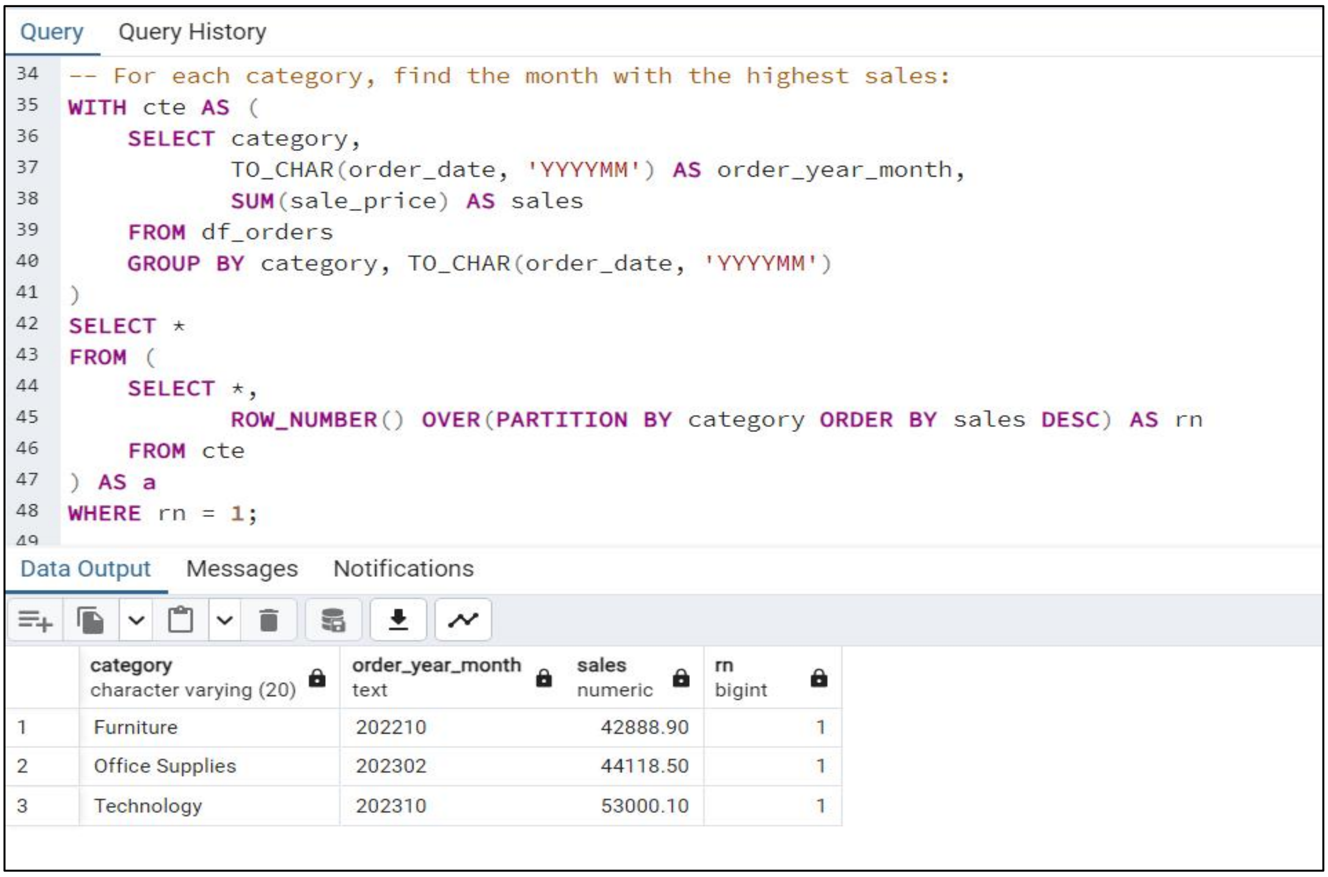
Query Query History

```sql
-- Find top 5 highest selling products in each region:
WITH cte AS (
    SELECT region, product_id, SUM(sale_price) AS sales
    FROM df_orders
    GROUP BY region, product_id)
SELECT *
FROM (SELECT *, ROW_NUMBER() OVER(PARTITION BY region ORDER BY sales DESC) AS rn
    FROM cte) AS A
WHERE rn <= 5;
```

Data Output Messages Notifications

	region character varying (20)	product_id character varying (50)	sales numeric	rn bigint
1	Central	TEC-CO-10004722	16975.00	1
2	Central	TEC-MA-10000822	13770.00	2
3	Central	OFF-BI-10001120	11056.50	3
4	Central	OFF-BI-10000545	10132.70	4
5	Central	OFF-BI-10004995	8416.10	5
6	East	TEC-CO-10004722	29099.00	1
7	East	TEC-MA-10001047	13767.00	2
8	East	FUR-BO-10004834	11274.10	3
9	East	OFF-BI-10001359	8463.60	4
10	East	TEC-CO-10001449	8316.00	5
11	South	TEC-MA-10002412	21734.40	1
12	South	TEC-MA-10001127	11116.40	2

Total rows: 20 of 20 Query complete 00:00:00.085

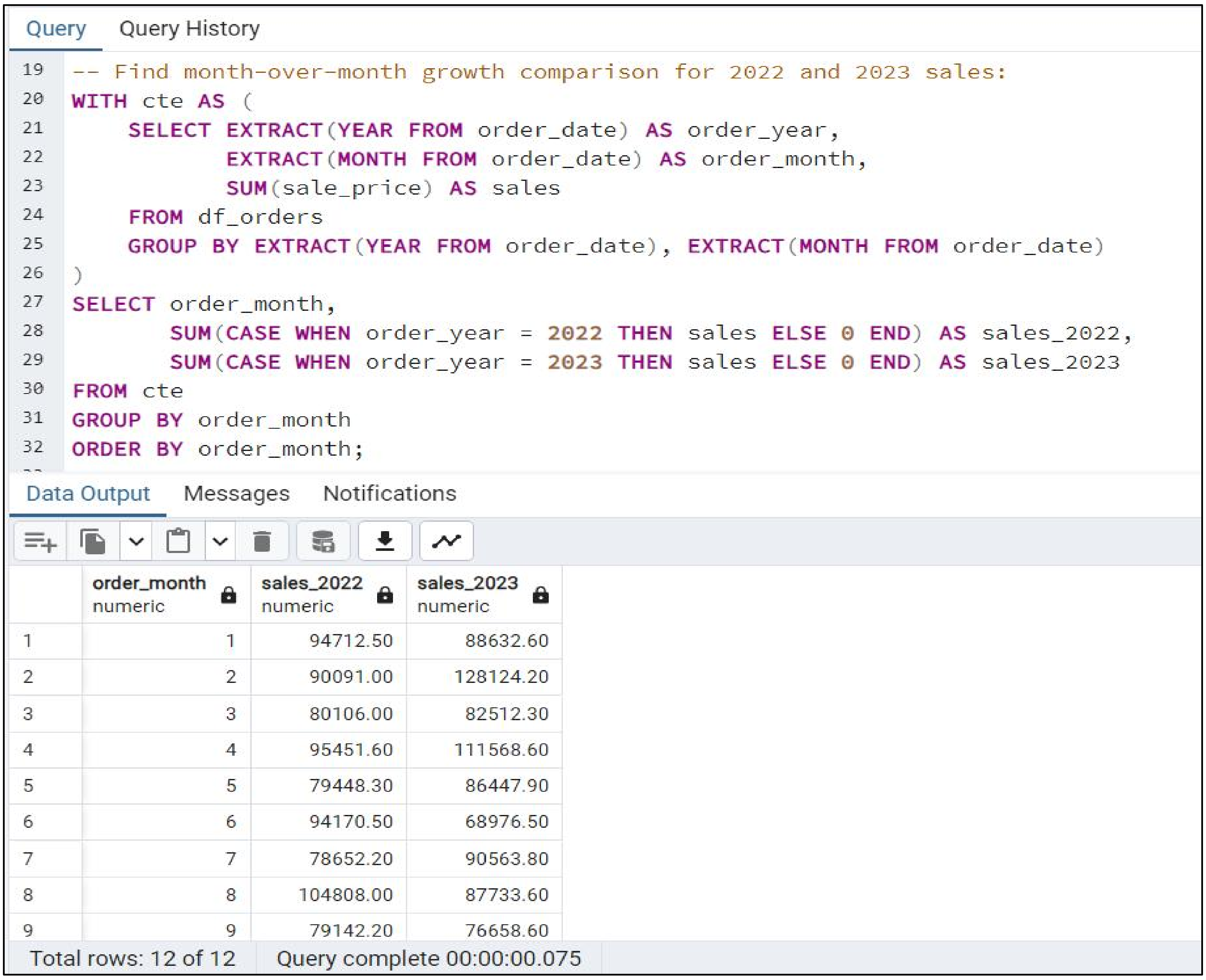


Query Query History

```sql
-- For each category, find the month with the highest sales:
WITH cte AS (
    SELECT category,
           TO_CHAR(order_date, 'YYYYMM') AS order_year_month,
           SUM(sale_price) AS sales
    FROM df_orders
    GROUP BY category, TO_CHAR(order_date, 'YYYYMM')
)
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER(PARTITION BY category ORDER BY sales DESC) AS rn
    FROM cte
) AS a
WHERE rn = 1;
```

Data Output Messages Notifications

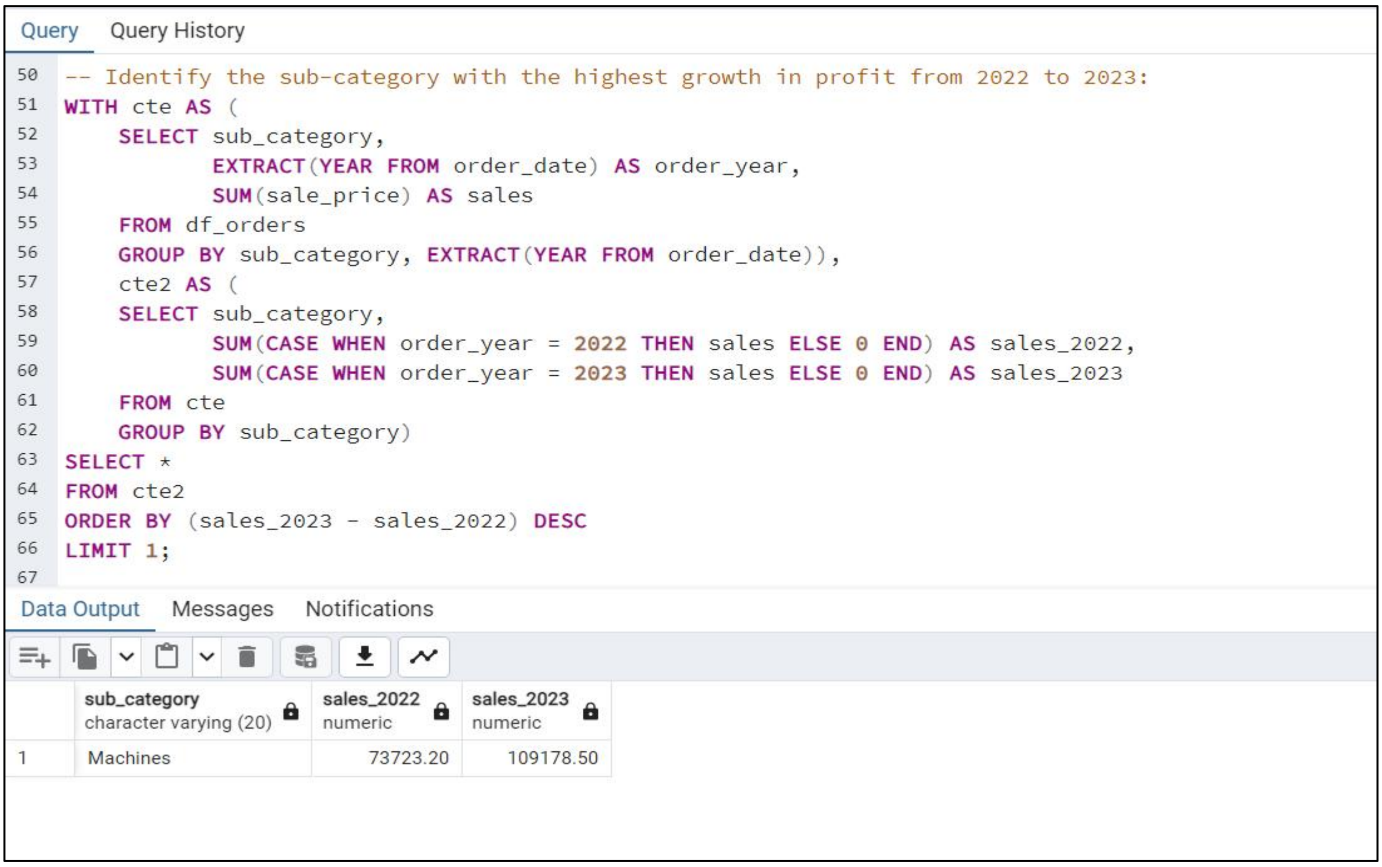
	category character varying (20)	order_year_month text	sales numeric	rn bigint
1	Furniture	202210	42888.90	1
2	Office Supplies	202302	44118.50	1
3	Technology	202310	53000.10	1



```sql
-- Find month-over-month growth comparison for 2022 and 2023 sales:
WITH cte AS (
    SELECT EXTRACT(YEAR FROM order_date) AS order_year,
           EXTRACT(MONTH FROM order_date) AS order_month,
           SUM(sale_price) AS sales
    FROM df_orders
    GROUP BY EXTRACT(YEAR FROM order_date), EXTRACT(MONTH FROM order_date)
)
SELECT order_month,
       SUM(CASE WHEN order_year = 2022 THEN sales ELSE 0 END) AS sales_2022,
       SUM(CASE WHEN order_year = 2023 THEN sales ELSE 0 END) AS sales_2023
FROM cte
GROUP BY order_month
ORDER BY order_month;
```

	order_month numeric	sales_2022 numeric	sales_2023 numeric
1	1	94712.50	88632.60
2	2	90091.00	128124.20
3	3	80106.00	82512.30
4	4	95451.60	111568.60
5	5	79448.30	86447.90
6	6	94170.50	68976.50
7	7	78652.20	90563.80
8	8	104808.00	87733.60
9	9	79142.20	76658.60

Total rows: 12 of 12   Query complete 00:00:00.075



```sql
-- Identify the sub-category with the highest growth in profit from 2022 to 2023:
WITH cte AS (
    SELECT sub_category,
           EXTRACT(YEAR FROM order_date) AS order_year,
           SUM(sale_price) AS sales
    FROM df_orders
    GROUP BY sub_category, EXTRACT(YEAR FROM order_date)),
    cte2 AS (
    SELECT sub_category,
           SUM(CASE WHEN order_year = 2022 THEN sales ELSE 0 END) AS sales_2022,
           SUM(CASE WHEN order_year = 2023 THEN sales ELSE 0 END) AS sales_2023
    FROM cte
    GROUP BY sub_category)
SELECT *
FROM cte2
ORDER BY (sales_2023 - sales_2022) DESC
LIMIT 1;
```

	sub_category character varying (20)	sales_2022 numeric	sales_2023 numeric
1	Machines	73723.20	109178.50