Introduction to the End-to-End Data Analytics Project with ETL Process For Retail Order Dataset

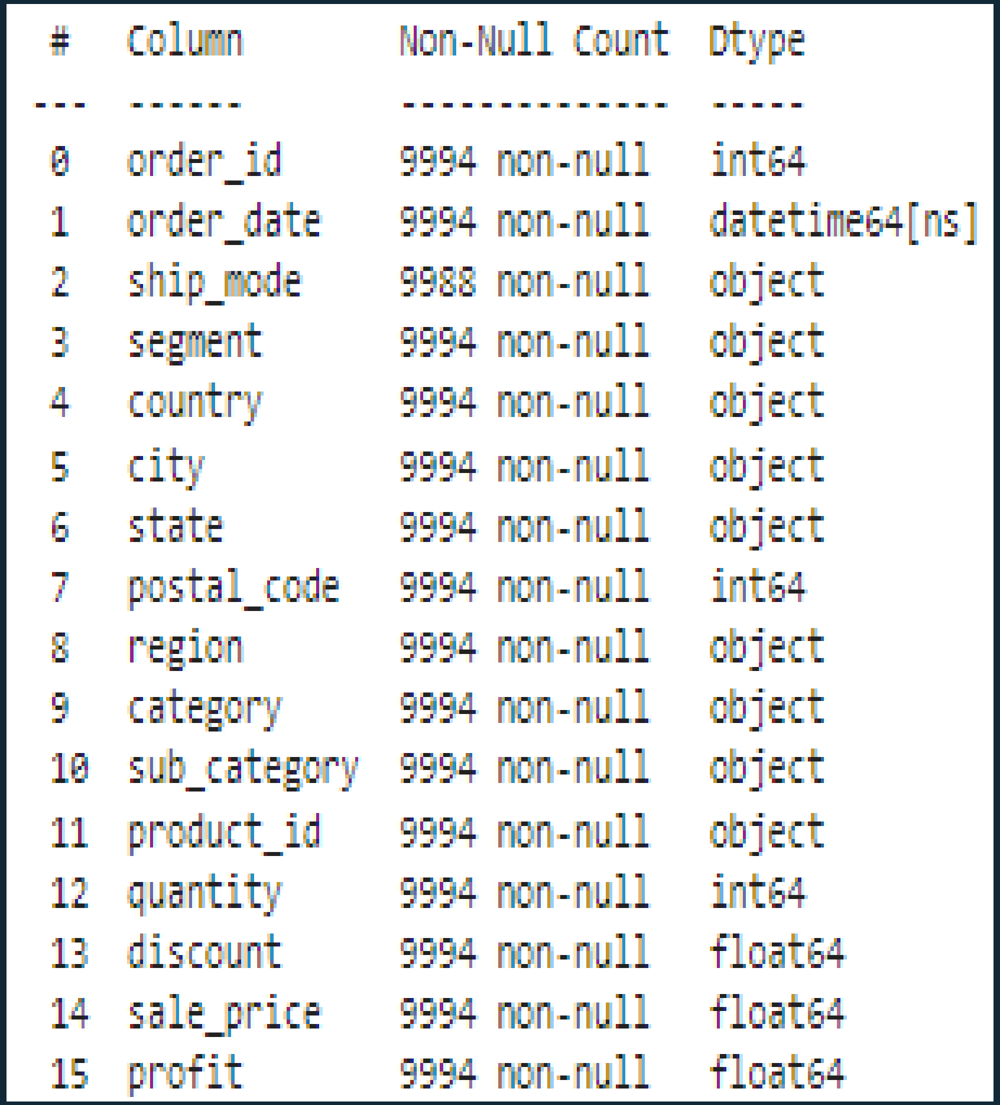
Welcome to our comprehensive data analytics project focusing on a retail orders dataset from Kaggle. In this presentation, we'll take you through the entire journey - from data extraction and transformation using Python and SQL, to in-depth data analysis and insights.

by Saurabh Shah

Overview of the Retail Orders Dataset from Kaggle

The Retail Orders dataset from Kaggle is a comprehensive dataset containing information about online orders placed by customers of a major retail company. The dataset includes details such as order date, ship date, ship mode, customer name, segment, country, city, state, postal code, region, product ID, category, sub-category, product name, sales, quantity, and discount.

This dataset provides a wealth of data that can be can be leveraged to uncover valuable insights insights about customer behavior, sales trends, trends, and supply chain performance for the the retail business.



```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   order_id      9994 non-null   int64
 1   order_date    9994 non-null   datetime64[ns]
 2   ship_mode     9988 non-null   object
 3   segment       9994 non-null   object
 4   country       9994 non-null   object
 5   city          9994 non-null   object
 6   state         9994 non-null   object
 7   postal_code   9994 non-null   int64
 8   region        9994 non-null   object
 9   category      9994 non-null   object
 10  sub_category  9994 non-null   object
 11  product_id    9994 non-null   object
 12  quantity      9994 non-null   int64
 13  discount      9994 non-null   float64
 14  sale_price    9994 non-null   float64
 15  profit        9994 non-null   float64
```

ETL Process using Python

1

Extract the Data

To extract data from Kaggle using the Kaggle API in Python, use the kaggle library to library to download datasets and competition files, making it ready for further processing. processing.

2

Transform Data

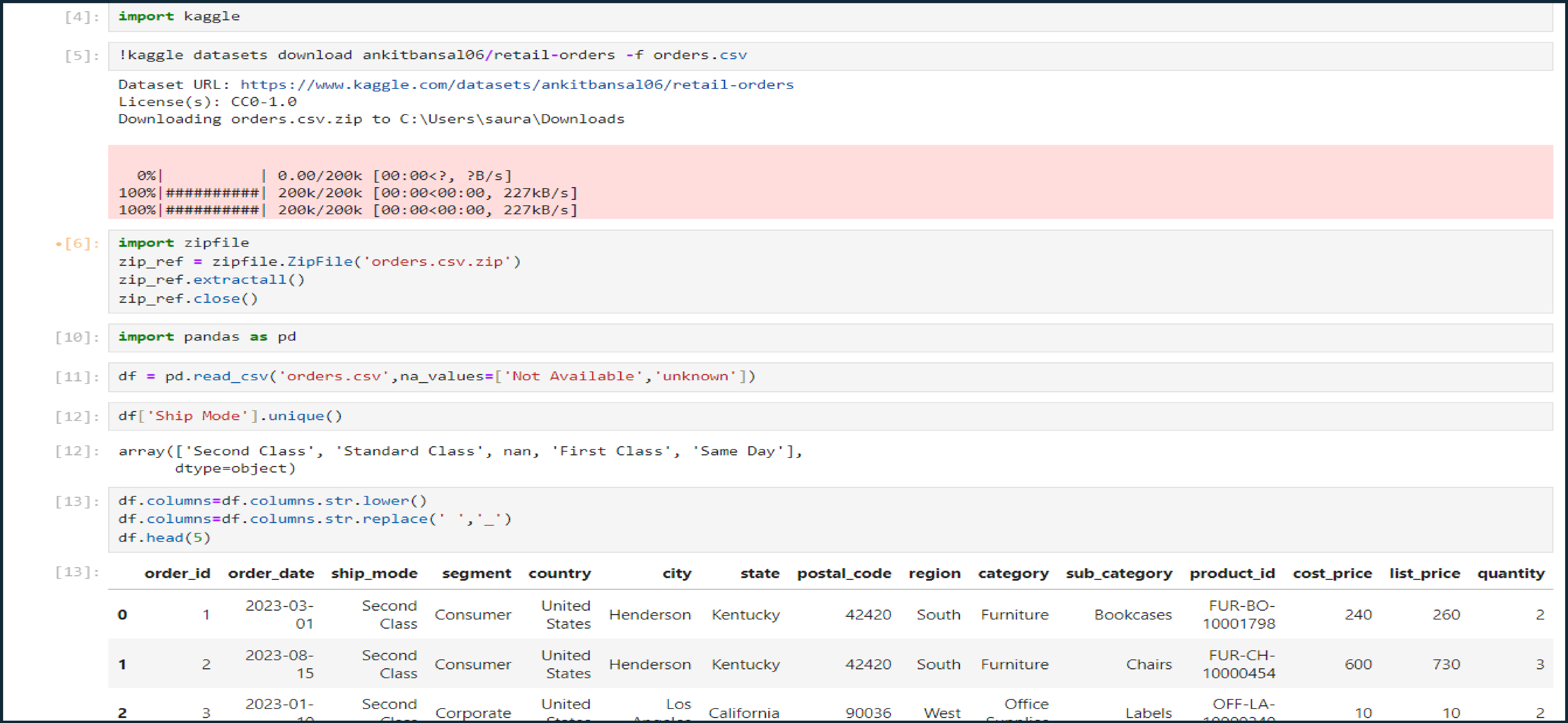
Examine the column names, data types, transform data using the pandas library in Python, Rename the column name, add some calculated column.

3

Load the Data

load data to SQL Server using Python's SQLAlchemy library and the to_sql() function.

ETL Process using Python(1)



[4]:
```
import kaggle
```

[5]:
```
!kaggle datasets download ankitbansal06/retail-orders -f orders.csv
```

```
Dataset URL: https://www.kaggle.com/datasets/ankitbansal06/retail-orders
License(s): CC0-1.0
Downloading orders.csv.zip to C:\Users\saura\Downloads
```

```
  0%|          | 0.00/200k [00:00<?, ?B/s]
100%|##########| 200k/200k [00:00<00:00, 227kB/s]
100%|##########| 200k/200k [00:00<00:00, 227kB/s]
```

•[6]:
```
import zipfile
zip_ref = zipfile.ZipFile('orders.csv.zip')
zip_ref.extractall()
zip_ref.close()
```

[10]:
```
import pandas as pd
```

[11]:
```
df = pd.read_csv('orders.csv',na_values=['Not Available','unknown'])
```

[12]:
```
df['Ship Mode'].unique()
```

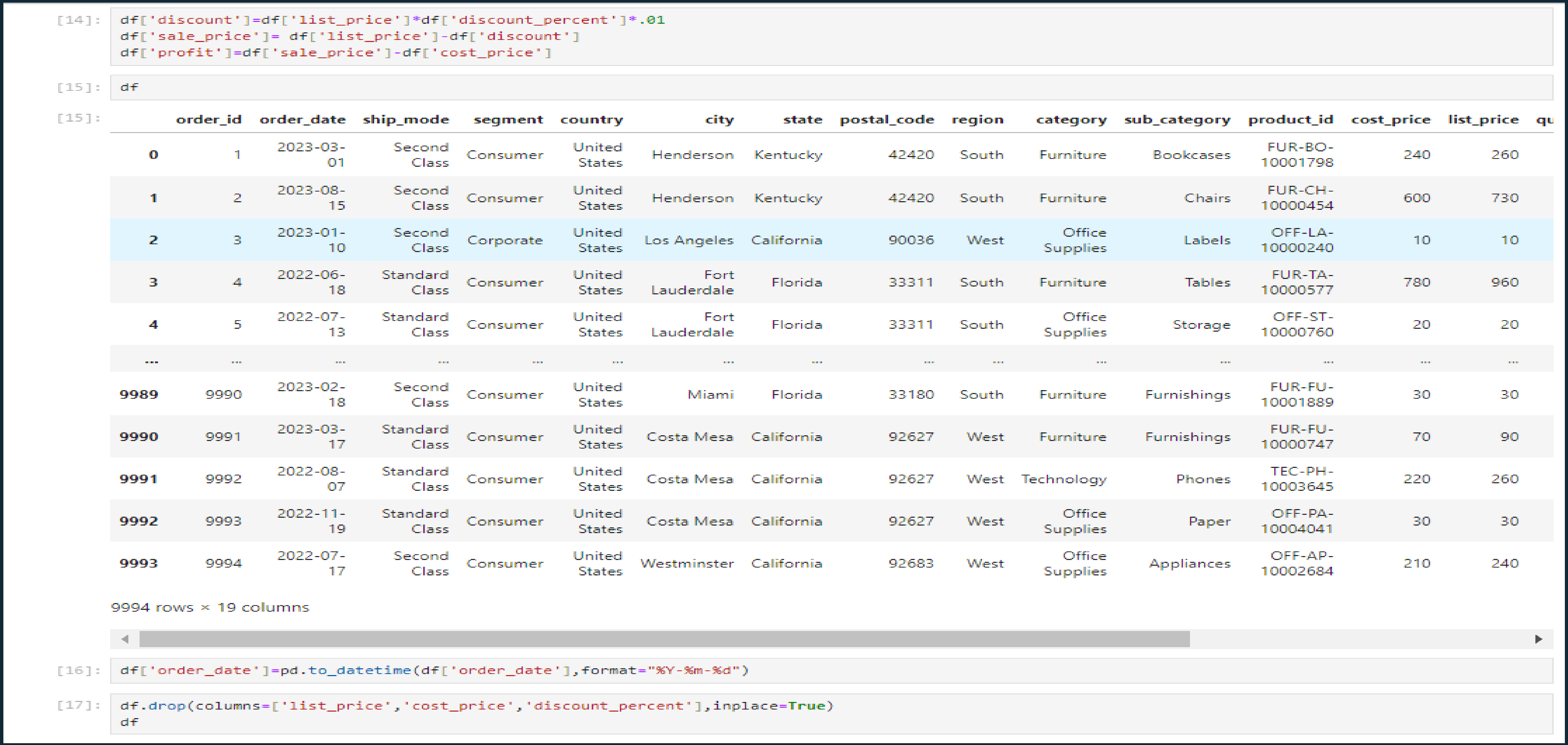
[12]:
```
array(['Second Class', 'Standard Class', nan, 'First Class', 'Same Day'],
      dtype=object)
```

[13]:
```
df.columns=df.columns.str.lower()
df.columns=df.columns.str.replace(' ','_')
df.head(5)
```

[13]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	cost_price	list_price	quantity
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3
2	3	2023-01-	Second	Corporate	United	Los	California	90036	West	Office	Labels	OFF-LA-	10	10	2

ETL Process using Python(2)



```
[14]: 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']
```

```
[15]: df
```

[15]:

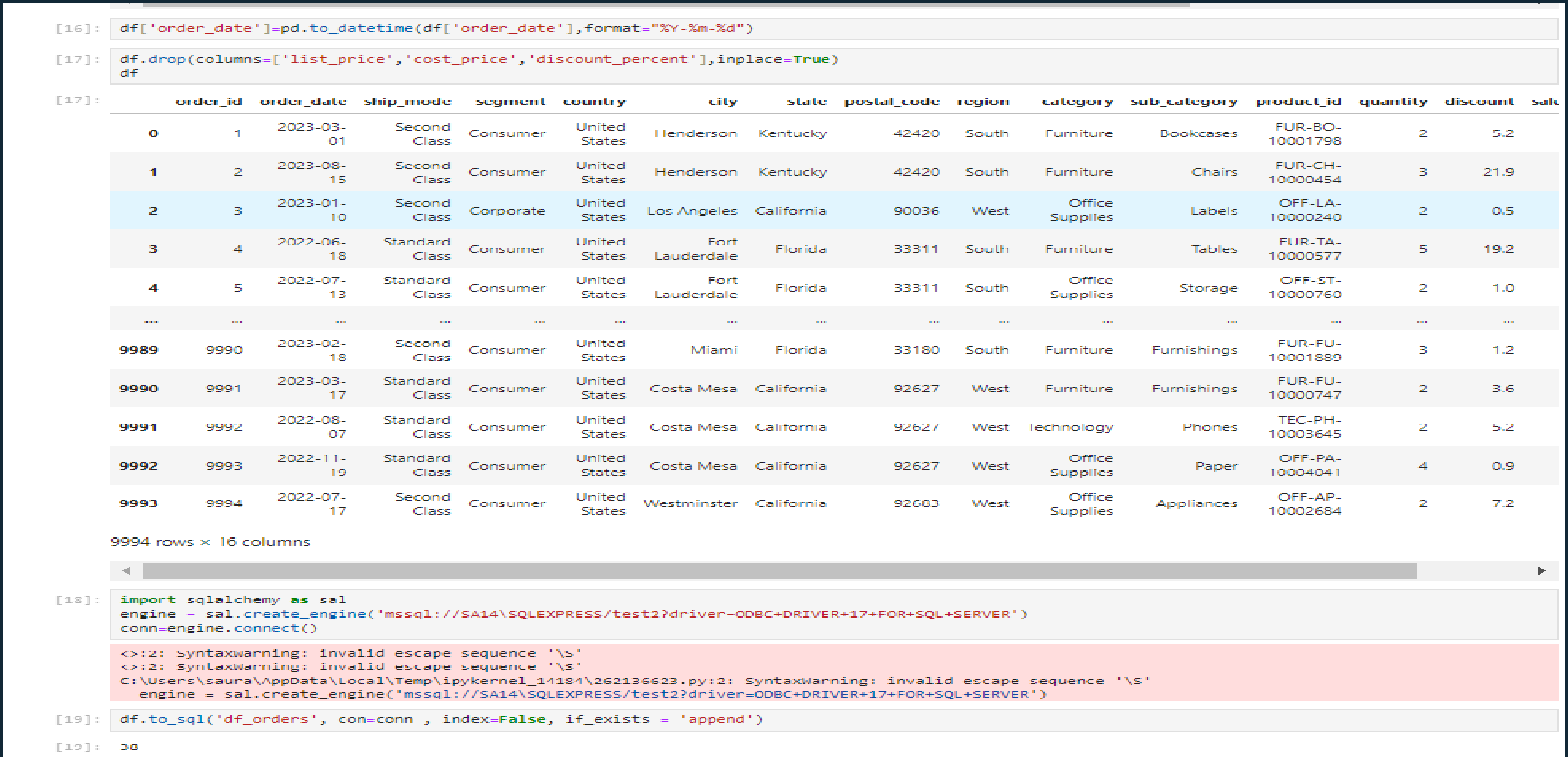
	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	cost_price	list_price	qu
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST-10000760	20	20	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9989	9990	2023-02-18	Second Class	Consumer	United States	Miami	Florida	33180	South	Furniture	Furnishings	FUR-FU-10001889	30	30	
9990	9991	2023-03-17	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Furniture	Furnishings	FUR-FU-10000747	70	90	
9991	9992	2022-08-07	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Technology	Phones	TEC-PH-10003645	220	260	
9992	9993	2022-11-19	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Office Supplies	Paper	OFF-PA-10004041	30	30	
9993	9994	2022-07-17	Second Class	Consumer	United States	Westminster	California	92683	West	Office Supplies	Appliances	OFF-AP-10002684	210	240	

9994 rows × 19 columns

```
[16]: df['order_date']=pd.to_datetime(df['order_date'],format="%Y-%m-%d")
```

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

ETL Process using Python(3)



```
[16]: df['order_date']=pd.to_datetime(df['order_date'],format="%Y-%m-%d")
```

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

[17]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	quantity	discount	sale
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9989	9990	2023-02-18	Second Class	Consumer	United States	Miami	Florida	33180	South	Furniture	Furnishings	FUR-FU-10001889	3	1.2	
9990	9991	2023-03-17	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Furniture	Furnishings	FUR-FU-10000747	2	3.6	
9991	9992	2022-08-07	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Technology	Phones	TEC-PH-10003645	2	5.2	
9992	9993	2022-11-19	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Office Supplies	Paper	OFF-PA-10004041	4	0.9	
9993	9994	2022-07-17	Second Class	Consumer	United States	Westminster	California	92683	West	Office Supplies	Appliances	OFF-AP-10002684	2	7.2	

9994 rows × 16 columns

```
[18]: import sqlalchemy as sal
      engine = sal.create_engine('mssql://SA14\SQLEXPRESS/test2?driver=ODBC+DRIVER+17+FOR+SQL+SERVER')
      conn=engine.connect()
```

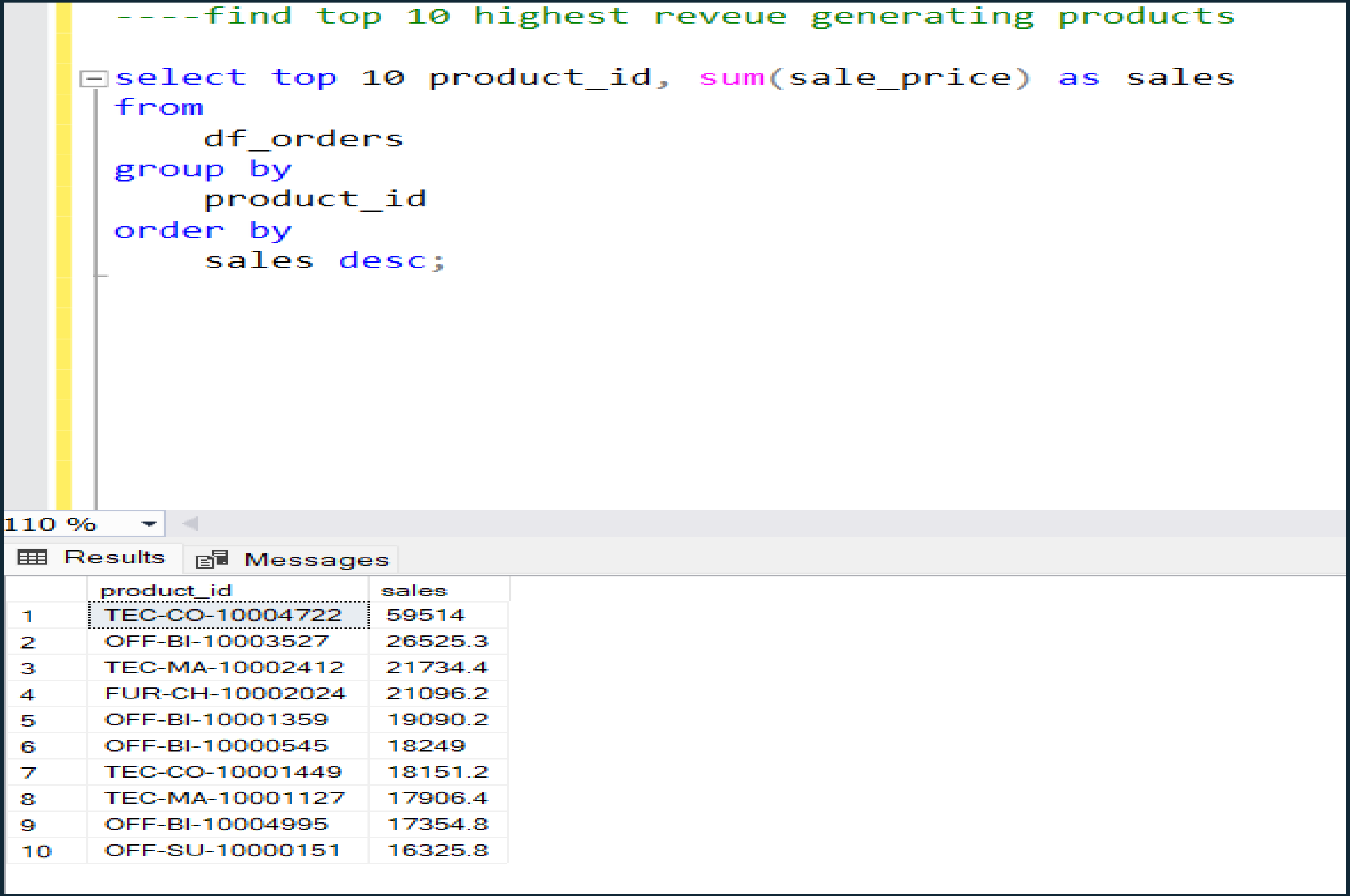
```
<>:2: SyntaxWarning: invalid escape sequence '\S'
<>:2: SyntaxWarning: invalid escape sequence '\S'
C:\Users\saura\AppData\Local\Temp\ipykernel_14184\262136623.py:2: SyntaxWarning: invalid escape sequence '\S'
  engine = sal.create_engine('mssql://SA14\SQLEXPRESS/test2?driver=ODBC+DRIVER+17+FOR+SQL+SERVER')
```

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

```
[19]: 38
```

SQL QUERIES

Find top 10 highest revenue generating products.

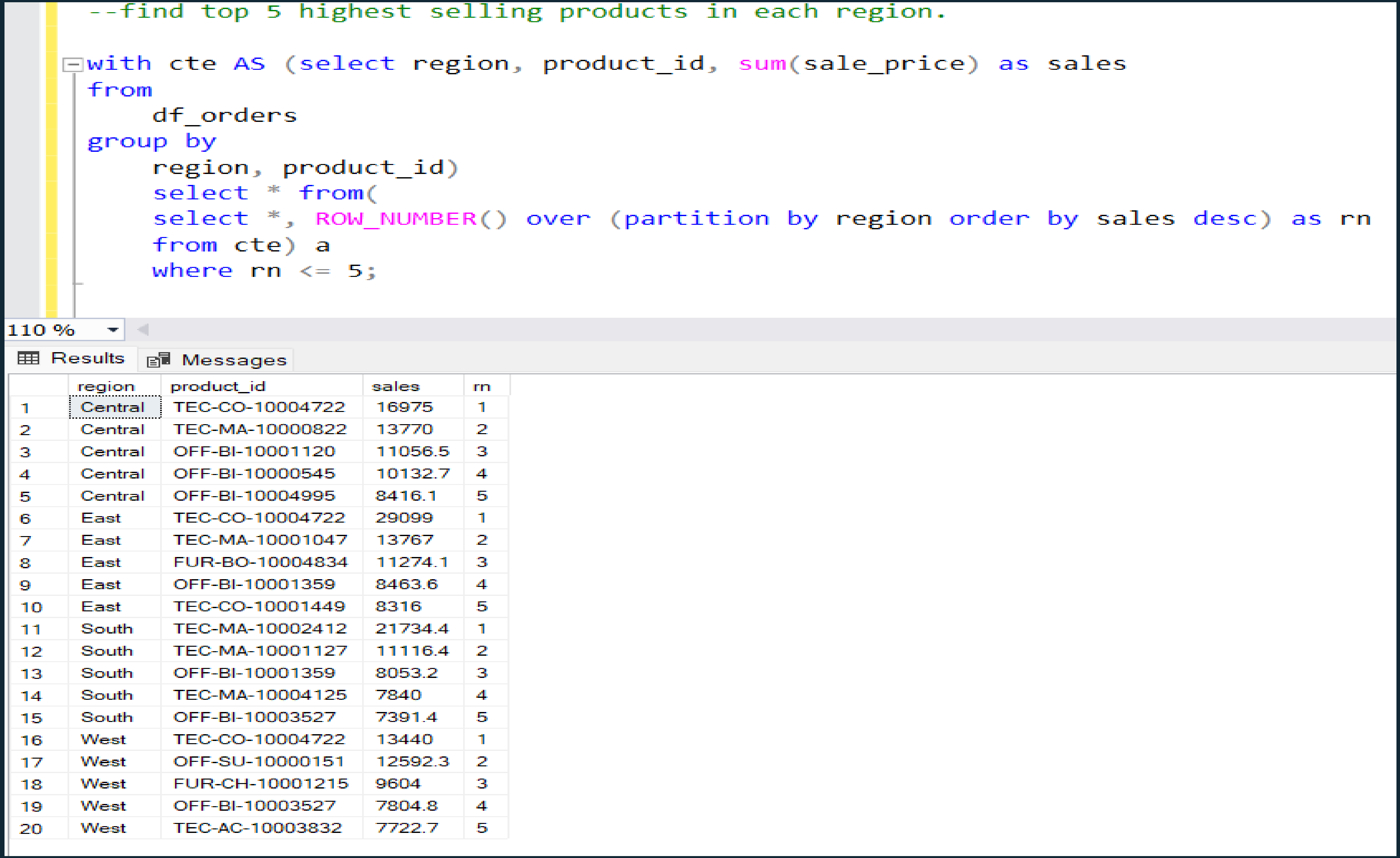


```
----find top 10 highest reveue generating products

select top 10 product_id, sum(sale_price) as sales
from
    df_orders
group by
    product_id
order by
    sales desc;
```

	product_id	sales
1	TEC-CO-10004722	59514
2	OFF-BI-10003527	26525.3
3	TEC-MA-10002412	21734.4
4	FUR-CH-10002024	21096.2
5	OFF-BI-10001359	19090.2
6	OFF-BI-10000545	18249
7	TEC-CO-10001449	18151.2
8	TEC-MA-10001127	17906.4
9	OFF-BI-10004995	17354.8
10	OFF-SU-10000151	16325.8

Find top 5 highest selling products in each region.

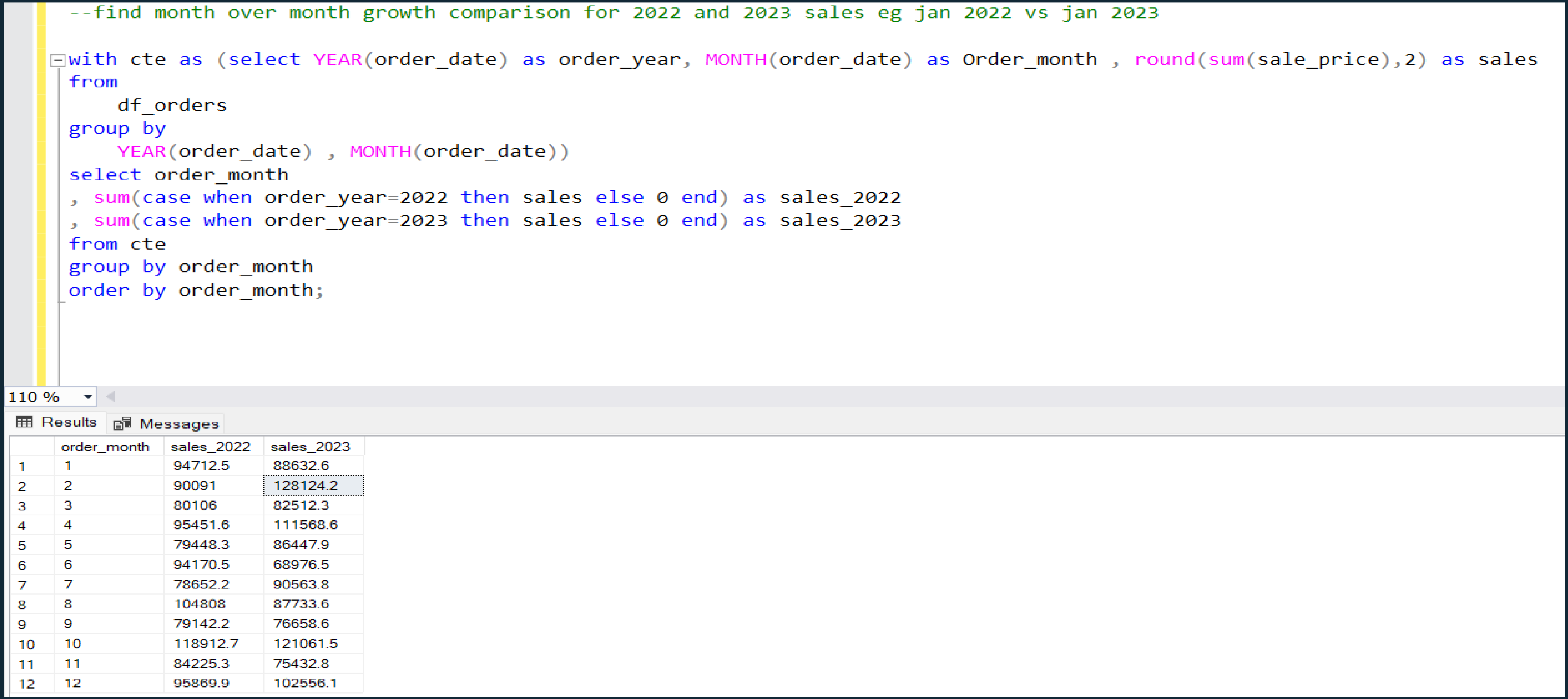


```
--find top 5 highest selling products in each region.

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) a
    where rn <= 5;
```

	region	product_id	sales	rn
1	Central	TEC-CO-10004722	16975	1
2	Central	TEC-MA-10000822	13770	2
3	Central	OFF-BI-10001120	11056.5	3
4	Central	OFF-BI-10000545	10132.7	4
5	Central	OFF-BI-10004995	8416.1	5
6	East	TEC-CO-10004722	29099	1
7	East	TEC-MA-10001047	13767	2
8	East	FUR-BO-10004834	11274.1	3
9	East	OFF-BI-10001359	8463.6	4
10	East	TEC-CO-10001449	8316	5
11	South	TEC-MA-10002412	21734.4	1
12	South	TEC-MA-10001127	11116.4	2
13	South	OFF-BI-10001359	8053.2	3
14	South	TEC-MA-10004125	7840	4
15	South	OFF-BI-10003527	7391.4	5
16	West	TEC-CO-10004722	13440	1
17	West	OFF-SU-10000151	12592.3	2
18	West	FUR-CH-10001215	9604	3
19	West	OFF-BI-10003527	7804.8	4
20	West	TEC-AC-10003832	7722.7	5

Find month over month growth comparison for 2022 and 2023 sales eg jan 2022 vs jan 2023.

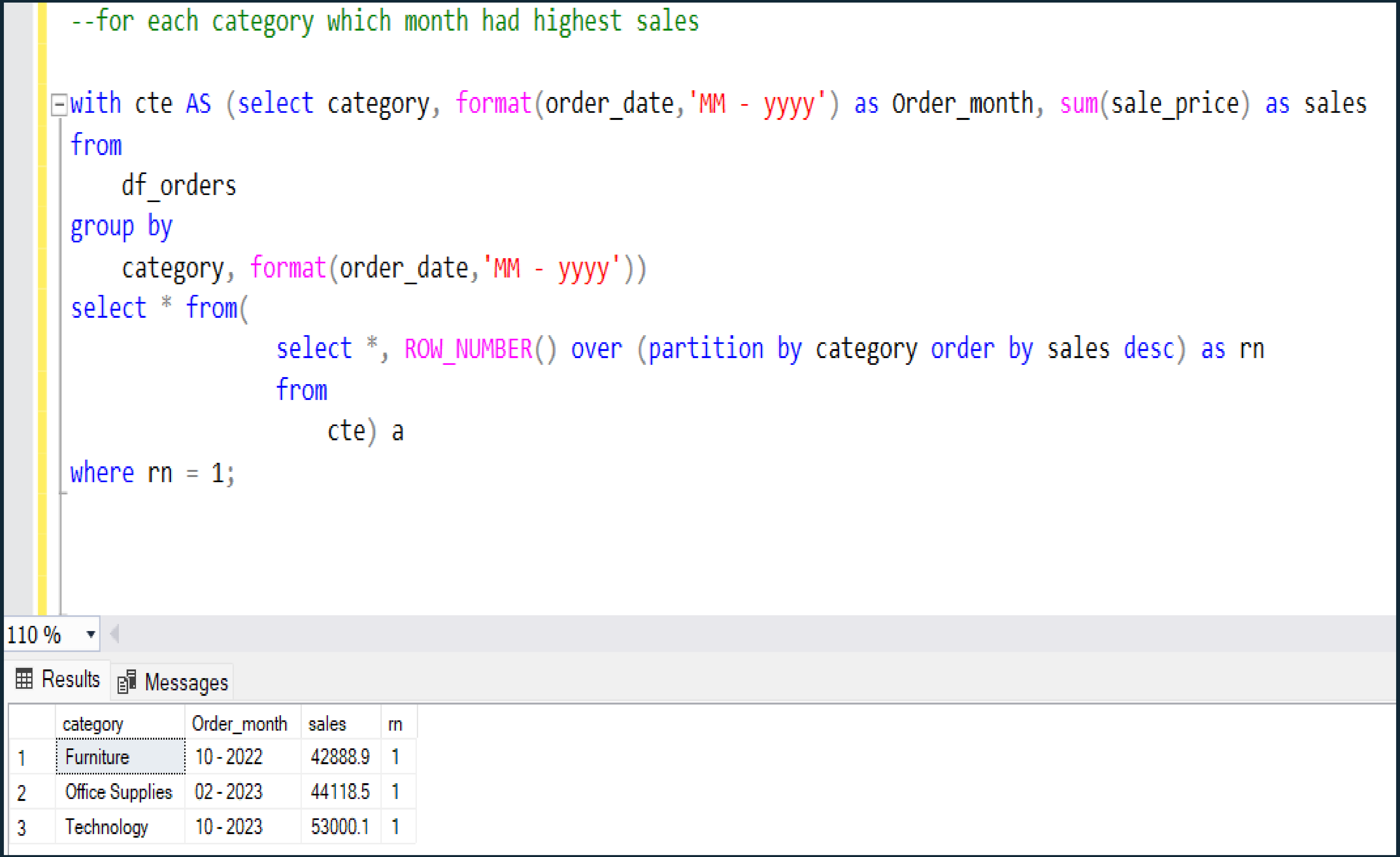


```
--find month over month growth comparison for 2022 and 2023 sales eg jan 2022 vs jan 2023

with cte as (select YEAR(order_date) as order_year, MONTH(order_date) as Order_month , round(sum(sale_price),2) as sales
from
    df_orders
group by
    YEAR(order_date) , MONTH(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	sales_2022	sales_2023
1	1	94712.5	88632.6
2	2	90091	128124.2
3	3	80106	82512.3
4	4	95451.6	111568.6
5	5	79448.3	86447.9
6	6	94170.5	68976.5
7	7	78652.2	90563.8
8	8	104808	87733.6
9	9	79142.2	76658.6
10	10	118912.7	121061.5
11	11	84225.3	75432.8
12	12	95869.9	102556.1

For each category which month had highest sales.

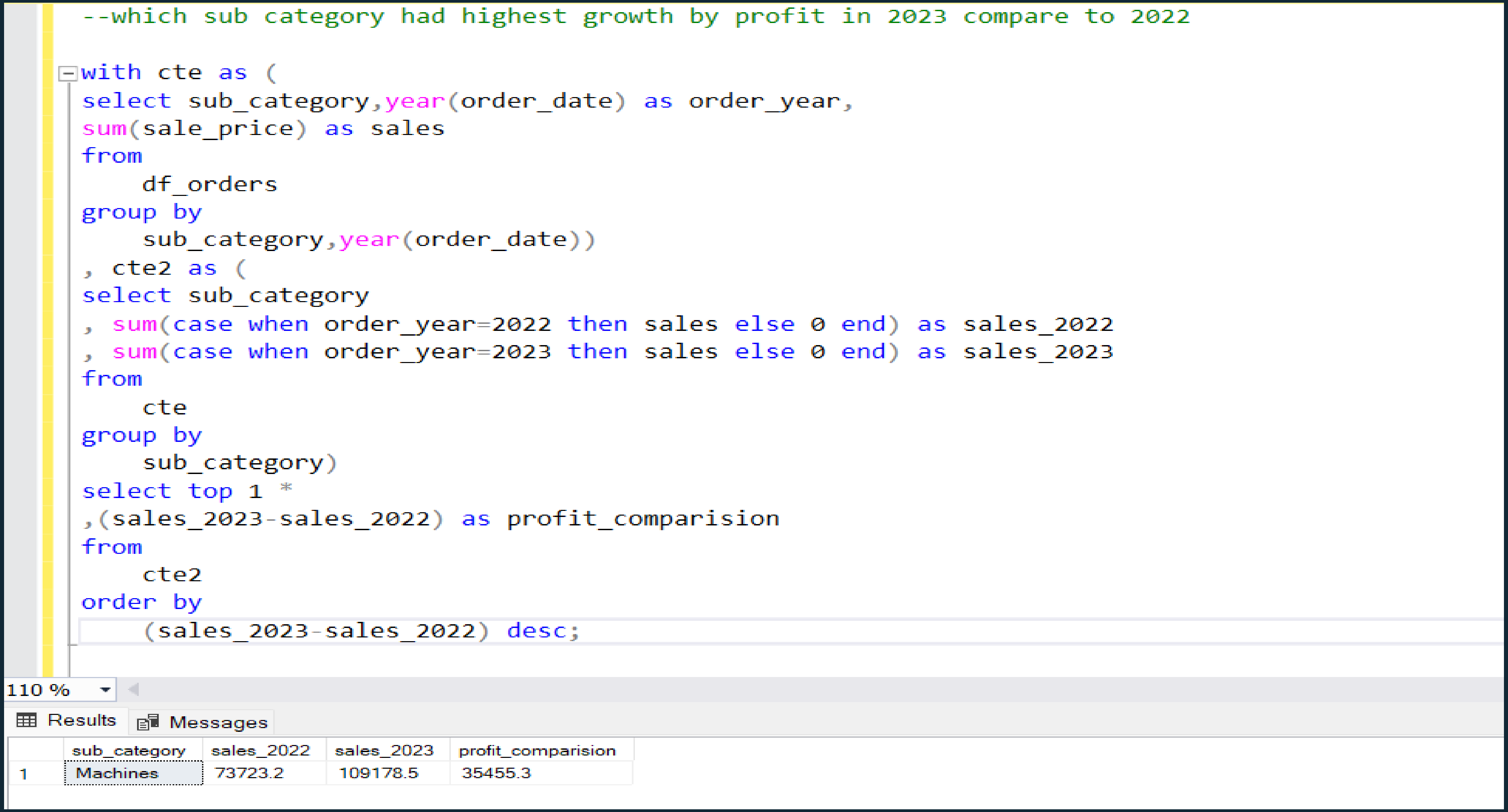


```
--for each category which month had highest sales

with cte AS (select category, format(order_date,'MM - yyyy') as Order_month, sum(sale_price) as sales
from
    df_orders
group by
    category, format(order_date,'MM - yyyy'))
select * from(
                select *, ROW_NUMBER() over (partition by category order by sales desc) as rn
                from
                    cte) a
where rn = 1;
```

	category	Order_month	sales	rn
1	Furniture	10 - 2022	42888.9	1
2	Office Supplies	02 - 2023	44118.5	1
3	Technology	10 - 2023	53000.1	1

Which sub category had highest growth by profit in 2023 compare to 2022.



```
--which sub category had highest growth by profit in 2023 compare to 2022

with cte as (
select sub_category,year(order_date) as order_year,
sum(sale_price) as sales
from
    df_orders
group by
    sub_category,year(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 top 1 *
,(sales_2023-sales_2022) as profit_comparision
from
    cte2
order by
    (sales_2023-sales_2022) desc;
```

	sub_category	sales_2022	sales_2023	profit_comparision
1	Machines	73723.2	109178.5	35455.3

Recommendations and Actionable Insights

Optimize Inventory Management

Analyze historical sales data to identify best-best-selling products and optimize inventory inventory levels accordingly. This can help help reduce stockouts and overstocking. overstocking.

Enhance Customer Engagement

Utilize customer purchase patterns to personalize product recommendations and offers. This can improve customer satisfaction and encourage repeat business.

Streamline Logistics

Identify bottlenecks in the supply chain and implement process improvements to enhance delivery times and reduce shipping costs.

Leverage Data-Driven Decisions

Continually analyze sales data, customer feedback, and market trends to make informed, data-driven decisions that drive drive business growth.

Conclusion and Next Steps

In this end-to-end data analytics project, we've demonstrated a comprehensive workflow for extracting, transforming, and analyzing retail order data. By leveraging the power of Python and SQL, we've uncovered valuable insights that can drive strategic decision-making.

THANK YOU