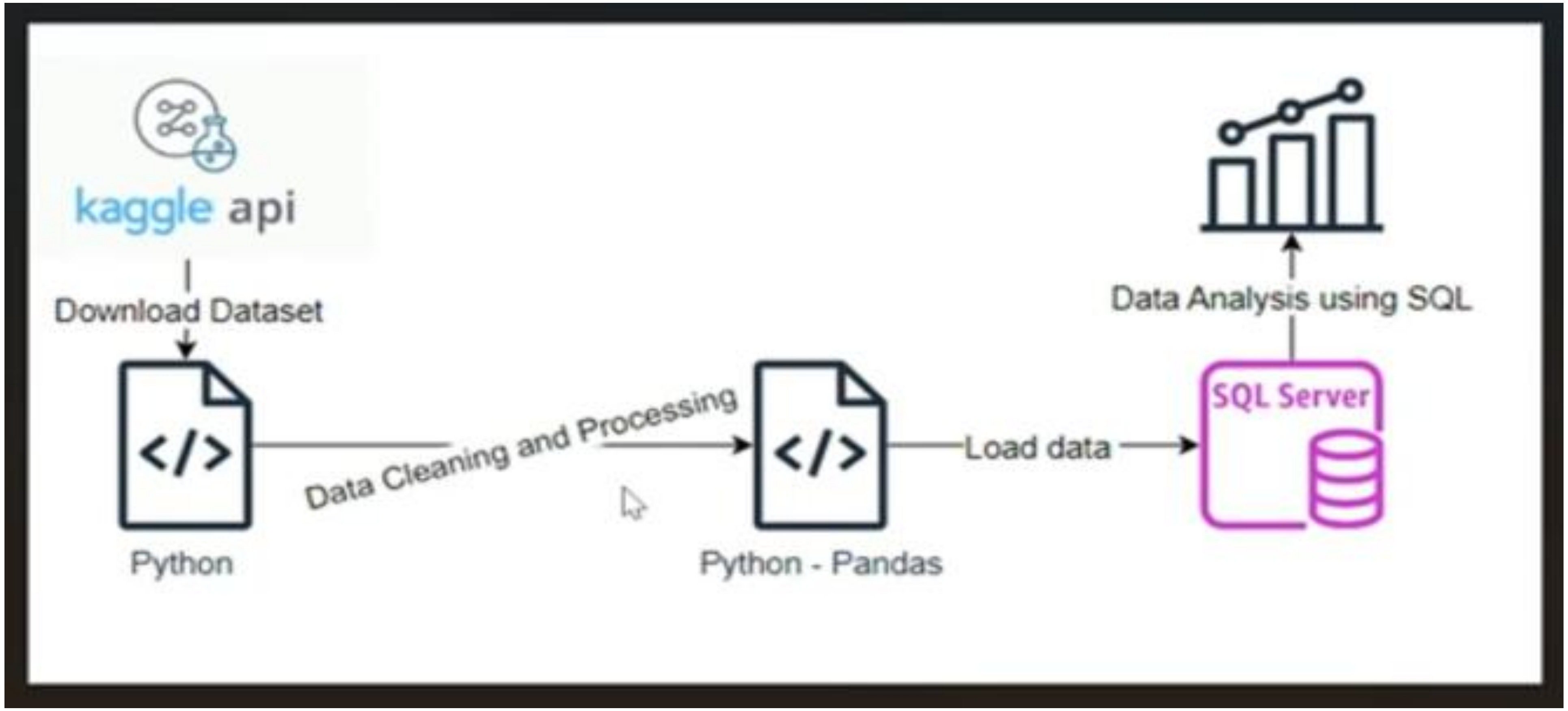
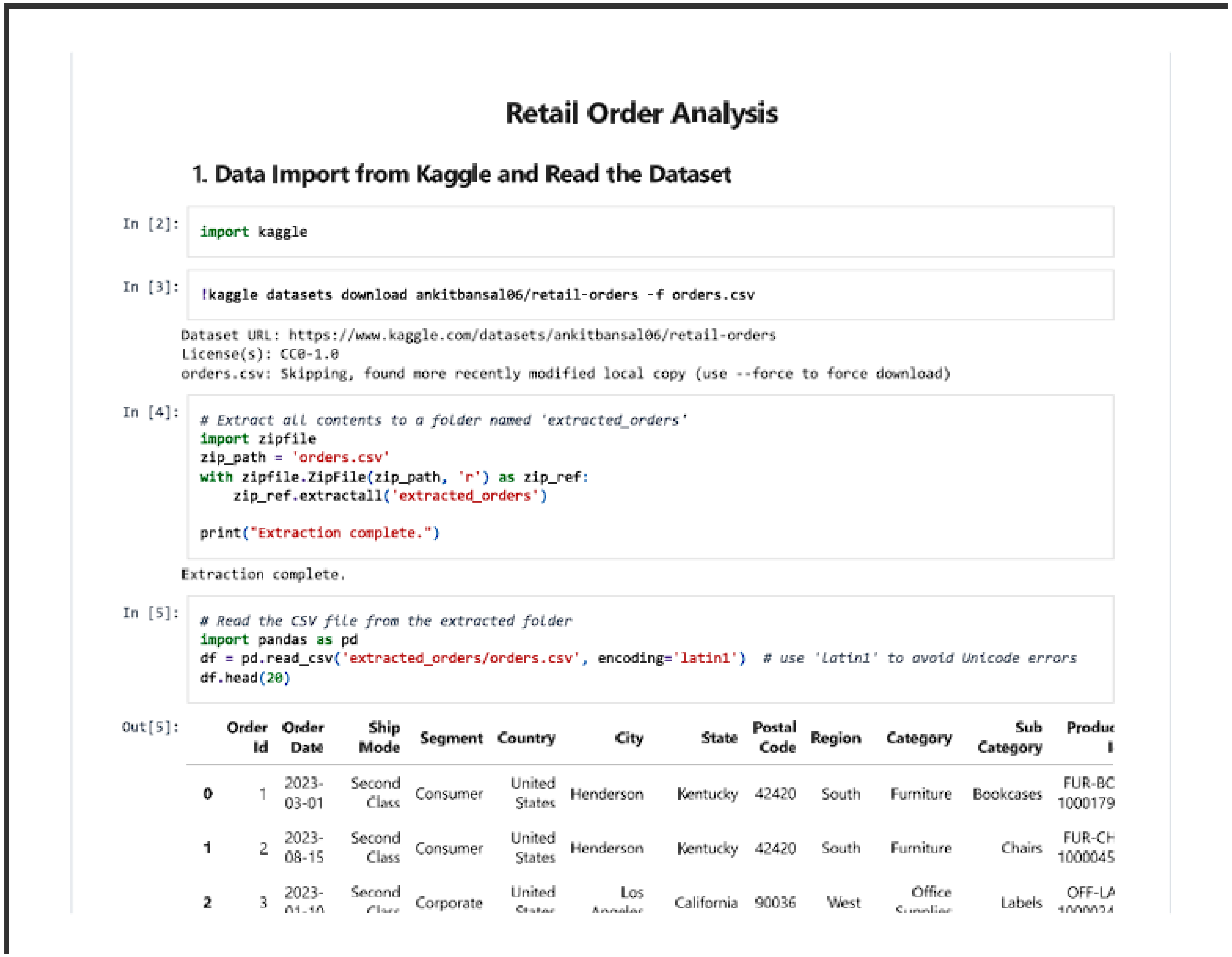

kaggle api
Download Dataset
Python
Data Cleaning and Processing
Python - Pandas
Load data
SQL Server
Data Analysis using SQL



Retail Order Analysis

1. Data Import from Kaggle and Read the Dataset

In [2]:
```
import kaggle
```

In [3]:
```
!kaggle datasets download ankitbansal06/retail-orders -f orders.csv
```

```
Dataset URL: https://www.kaggle.com/datasets/ankitbansal06/retail-orders
License(s): CC0-1.0
orders.csv: Skipping, found more recently modified local copy (use --force to force download)
```

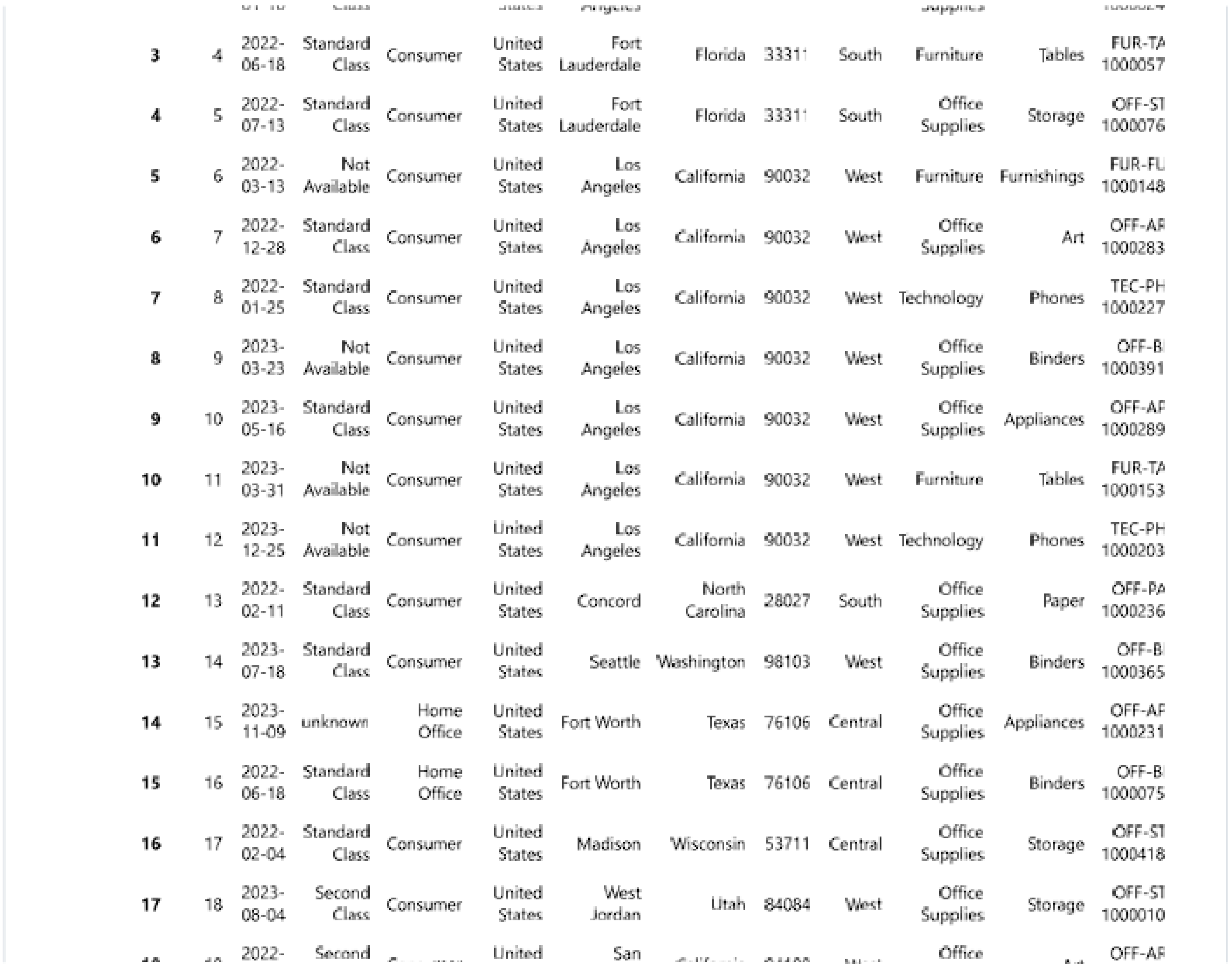
In [4]:
```
# Extract all contents to a folder named 'extracted_orders'
import zipfile
zip_path = 'orders.csv'
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall('extracted_orders')

print("Extraction complete.")
```

```
Extraction complete.
```

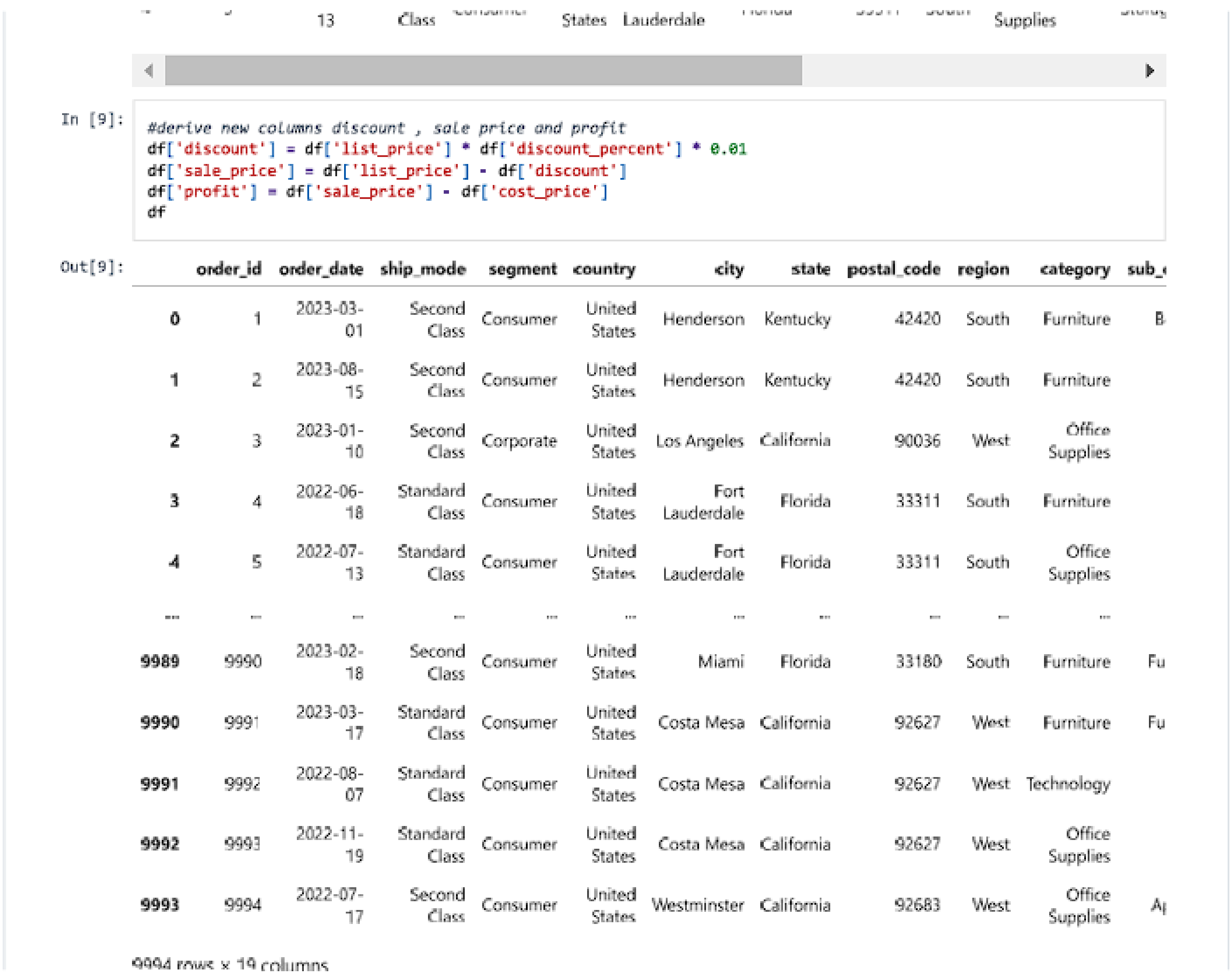
In [5]:
```
# Read the CSV file from the extracted folder
import pandas as pd
df = pd.read_csv('extracted_orders/orders.csv', encoding='latin1')  # use 'latin1' to avoid Unicode errors
df.head(20)
```

Out[5]:

	Order Id	Order Date	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub Category	Produc I
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BC 1000179
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH 1000045
2	3	2023-	Second	Corporate	United	Los	California	90036	West	Office	Labels	OFF-LA



3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA 1000057
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	OFF-ST 1000076
5	6	2022-03-13	Not Available	Consumer	United States	Los Angeles	California	90032	West	Furniture	Furnishings	FUR-FU 1000148
6	7	2022-12-28	Standard Class	Consumer	United States	Los Angeles	California	90032	West	Office Supplies	Art	OFF-AR 1000283
7	8	2022-01-25	Standard Class	Consumer	United States	Los Angeles	California	90032	West	Technology	Phones	TEC-PH 1000227
8	9	2023-03-23	Not Available	Consumer	United States	Los Angeles	California	90032	West	Office Supplies	Binders	OFF-BI 1000391
9	10	2023-05-16	Standard Class	Consumer	United States	Los Angeles	California	90032	West	Office Supplies	Appliances	OFF-AP 1000289
10	11	2023-03-31	Not Available	Consumer	United States	Los Angeles	California	90032	West	Furniture	Tables	FUR-TA 1000153
11	12	2023-12-25	Not Available	Consumer	United States	Los Angeles	California	90032	West	Technology	Phones	TEC-PH 1000203
12	13	2022-02-11	Standard Class	Consumer	United States	Concord	North Carolina	28027	South	Office Supplies	Paper	OFF-PA 1000236
13	14	2023-07-18	Standard Class	Consumer	United States	Seattle	Washington	98103	West	Office Supplies	Binders	OFF-BI 1000365
14	15	2023-11-09	unknown	Home Office	United States	Fort Worth	Texas	76106	Central	Office Supplies	Appliances	OFF-AP 1000231
15	16	2022-06-18	Standard Class	Home Office	United States	Fort Worth	Texas	76106	Central	Office Supplies	Binders	OFF-BI 1000075
16	17	2022-02-04	Standard Class	Consumer	United States	Madison	Wisconsin	53711	Central	Office Supplies	Storage	OFF-ST 1000418
17	18	2023-08-04	Second Class	Consumer	United States	West Jordan	Utah	84084	West	Office Supplies	Storage	OFF-ST 1000010
		2022-	Second		United	San				Office		OFF-AR



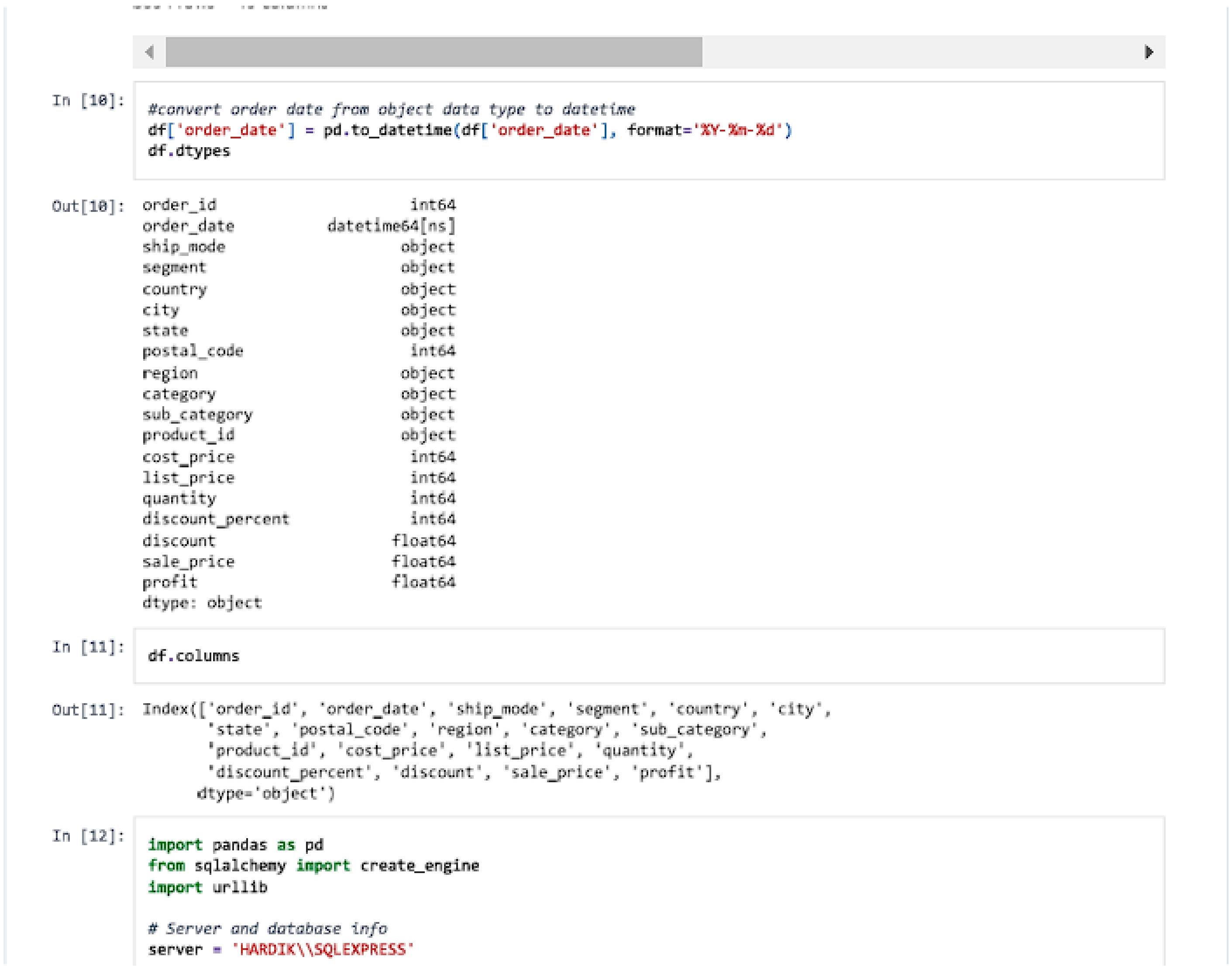
13 Class States Lauderdale Supplies

In [9]:
```
#derive new columns discount , sale price and profit
df['discount'] = df['list_price'] * df['discount_percent'] * 0.01
df['sale_price'] = df['list_price'] - df['discount']
df['profit'] = df['sale_price'] - df['cost_price']
df
```

Out[9]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	B
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	
4	5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	
...	...	...	...	...	...	...	...	...	...	...	
9989	9990	2023-02-18	Second Class	Consumer	United States	Miami	Florida	33180	South	Furniture	Fu
9990	9991	2023-03-17	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Furniture	Fu
9991	9992	2022-08-07	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Technology	
9992	9993	2022-11-19	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Office Supplies	
9993	9994	2022-07-17	Second Class	Consumer	United States	Westminster	California	92683	West	Office Supplies	A

9994 rows × 19 columns



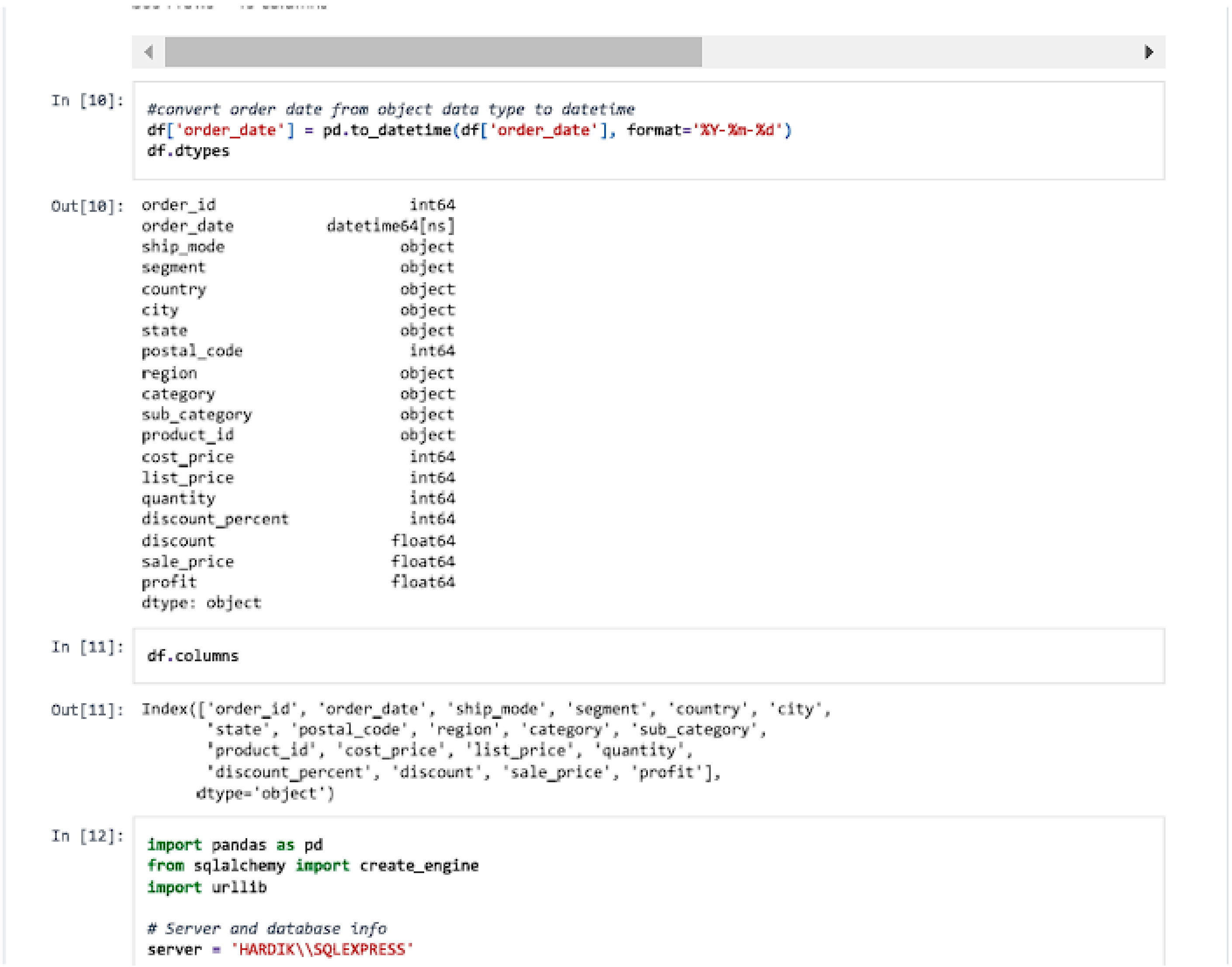
In [10]:
```
#convert order date from object data type to datetime
df['order_date'] = pd.to_datetime(df['order_date'], format='%Y-%m-%d')
df.dtypes
```

Out[10]:
```
order_id                      int64
order_date           datetime64[ns]
ship_mode                    object
segment                      object
country                      object
city                         object
state                        object
postal_code                   int64
region                       object
category                     object
sub_category                 object
product_id                   object
cost_price                    int64
list_price                    int64
quantity                      int64
discount_percent              int64
discount                    float64
sale_price                  float64
profit                      float64
dtype: object
```

In [11]:
```
df.columns
```

Out[11]:
```
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', 'discount', 'sale_price', 'profit'],
      dtype='object')
```

In [12]:
```
import pandas as pd
from sqlalchemy import create_engine
import urllib

# Server and database info
server = 'HARDIK\\SQLEXPRESS'
```



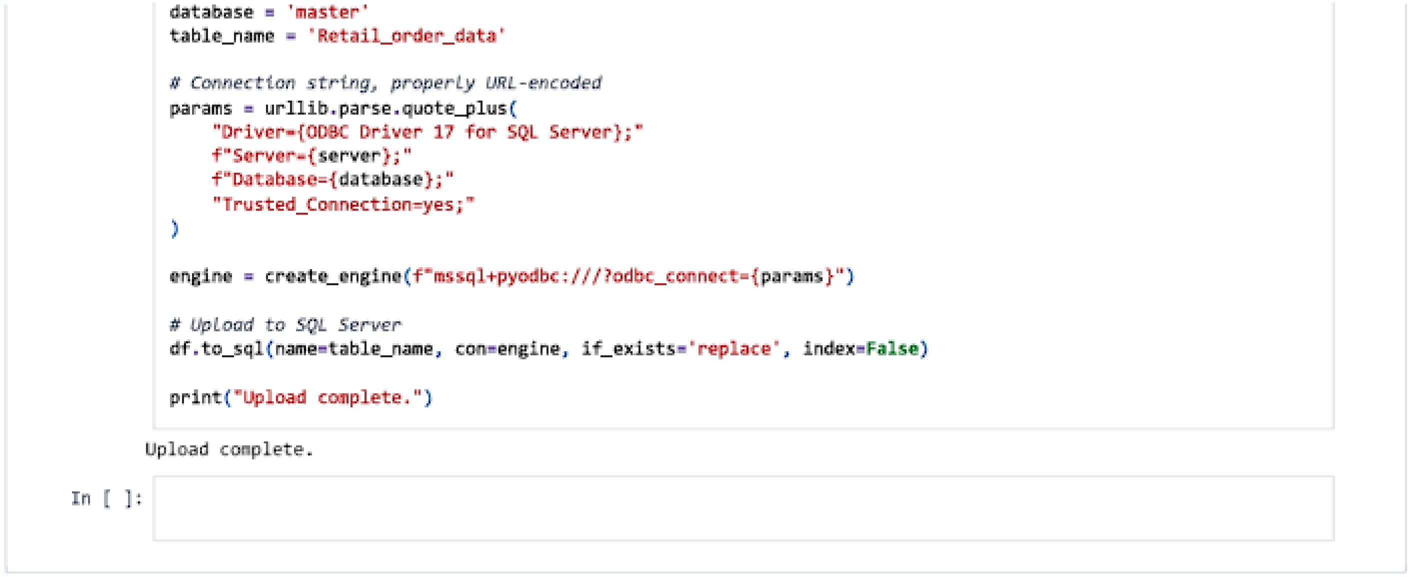
In [10]:
```
#convert order date from object data type to datetime
df['order_date'] = pd.to_datetime(df['order_date'], format='%Y-%m-%d')
df.dtypes
```

Out[10]:
```
order_id                     int64
order_date          datetime64[ns]
ship_mode                   object
segment                     object
country                     object
city                        object
state                       object
postal_code                  int64
region                      object
category                    object
sub_category                object
product_id                  object
cost_price                   int64
list_price                   int64
quantity                     int64
discount_percent             int64
discount                   float64
sale_price                 float64
profit                     float64
dtype: object
```

In [11]:
```
df.columns
```

Out[11]:
```
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', 'discount', 'sale_price', 'profit'],
      dtype='object')
```

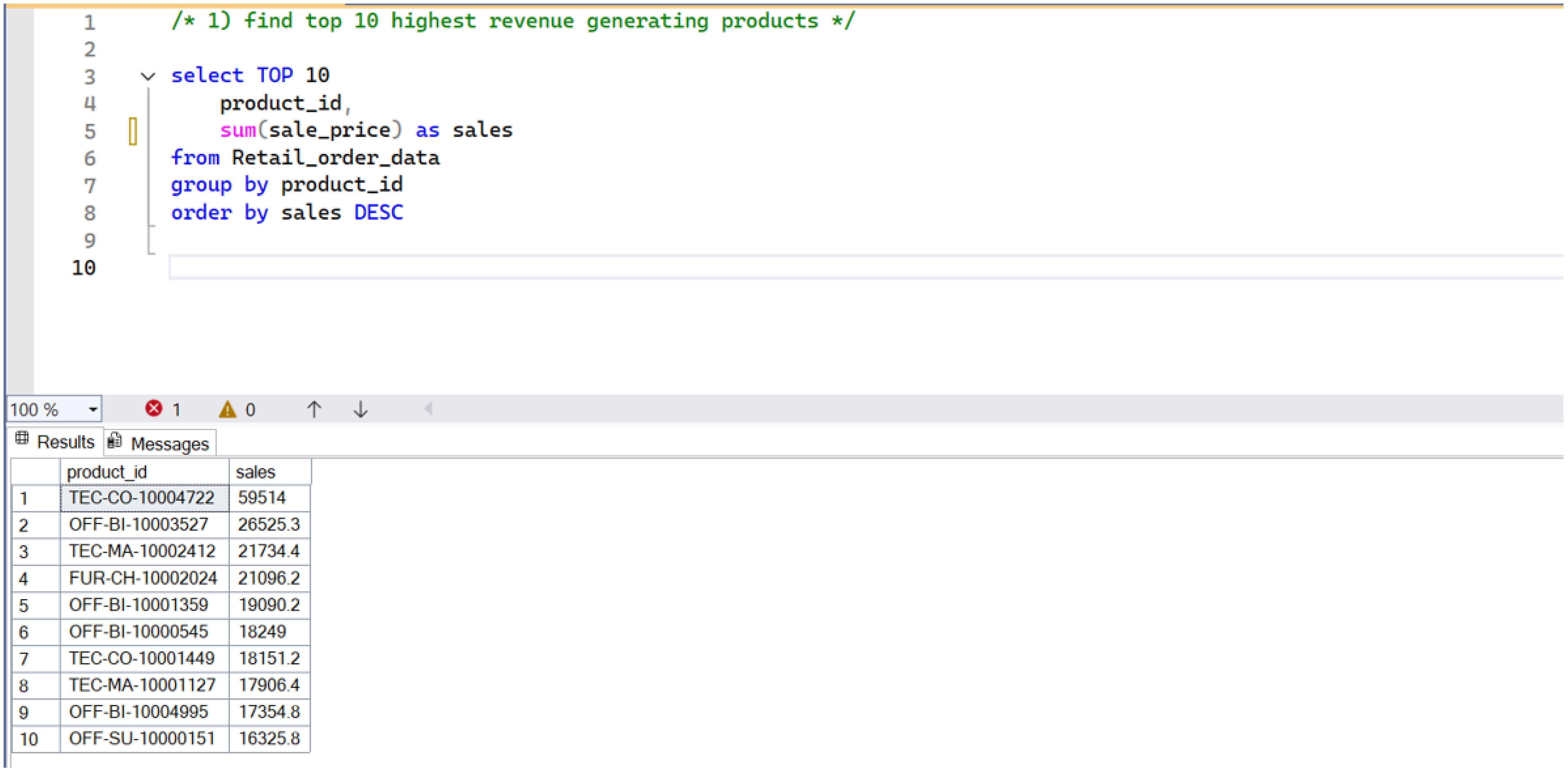
In [12]:
```
import pandas as pd
from sqlalchemy import create_engine
import urllib

# Server and database info
server = 'HARDIK\\SQLEXPRESS'
```



```
database = 'master'
table_name = 'Retail_order_data'

# Connection string, properly URL-encoded
params = urllib.parse.quote_plus(
    "Driver={ODBC Driver 17 for SQL Server};"
    f"Server={server};"
    f"Database={database};"
    "Trusted_Connection=yes;"
)

engine = create_engine(f"mssql+pyodbc:///?odbc_connect={params}")

# Upload to SQL Server
df.to_sql(name=table_name, con=engine, if_exists='replace', index=False)

print("Upload complete.")
```

```
Upload complete.
```

```
In [ ]:
```

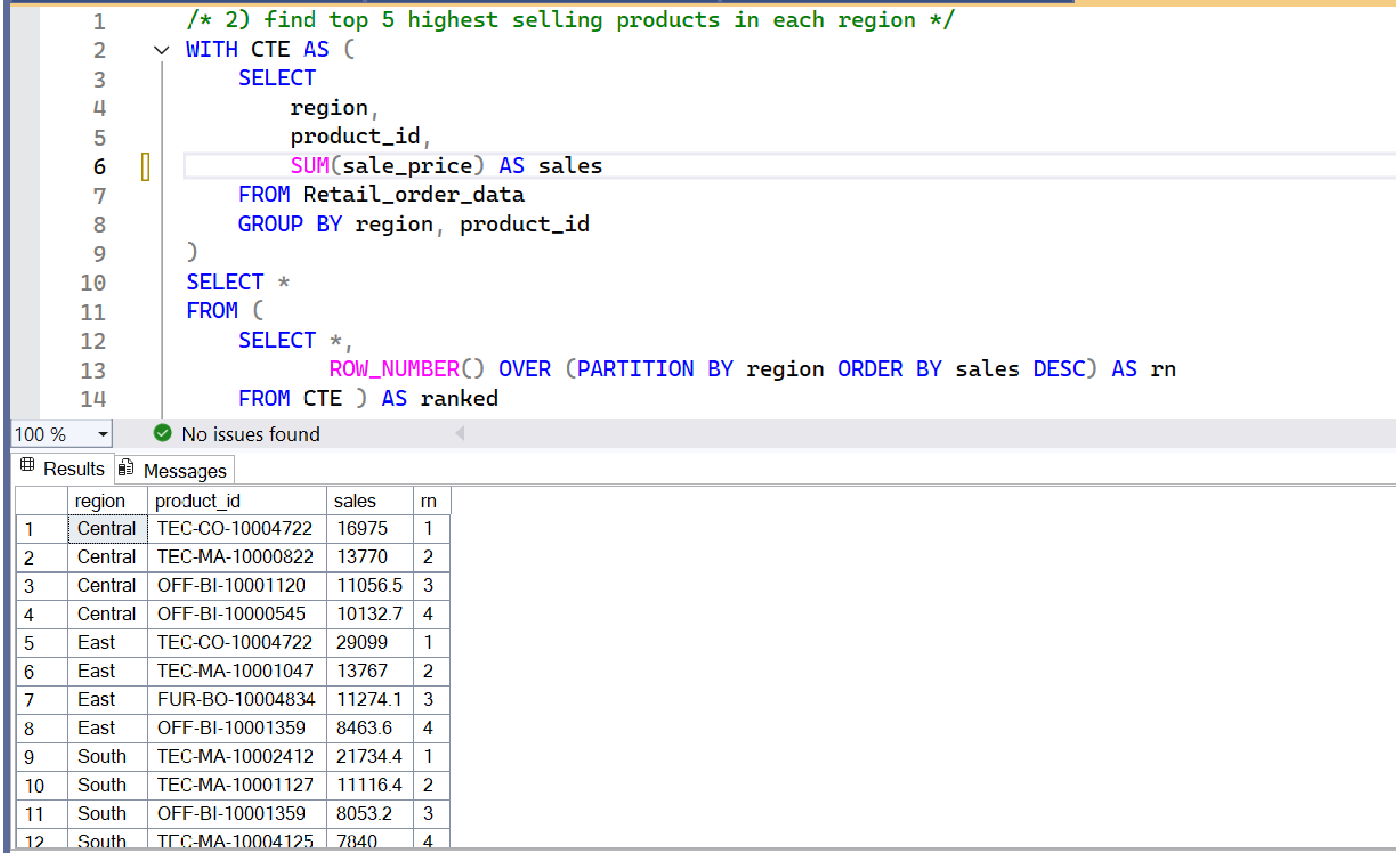


```sql
/* 1) find top 10 highest revenue generating products */

select TOP 10
    product_id,
    sum(sale_price) as sales
from Retail_order_data
group by product_id
order by sales DESC

```

100 %   1   0

Results   Messages

	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

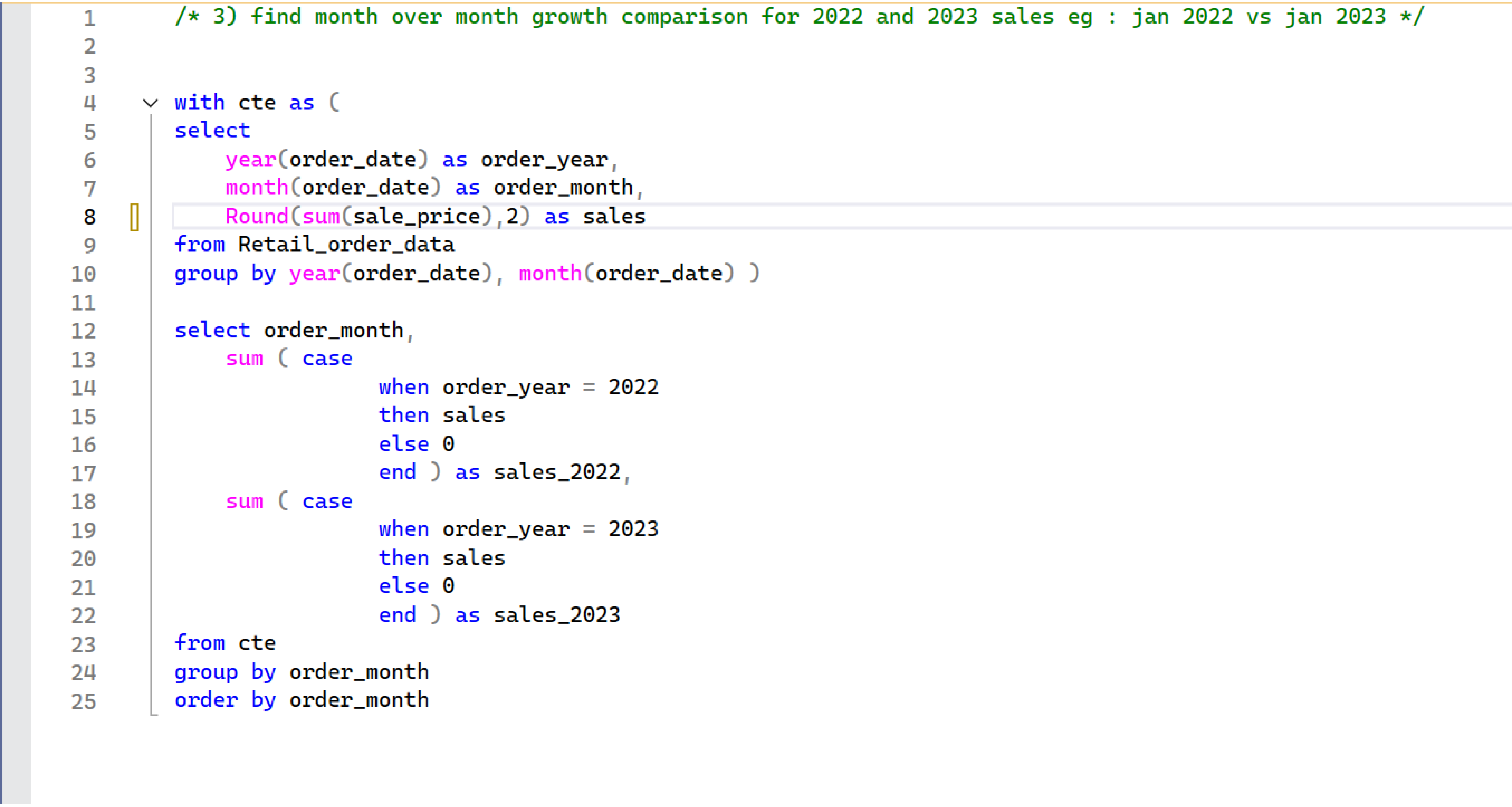


```sql
/* 2) find top 5 highest selling products in each region */
WITH CTE AS (
    SELECT
        region,
        product_id,
        SUM(sale_price) AS sales
    FROM Retail_order_data
    GROUP BY region, product_id
)
SELECT *
FROM (
    SELECT *,
          ROW_NUMBER() OVER (PARTITION BY region ORDER BY sales DESC) AS rn
    FROM CTE ) AS ranked
```

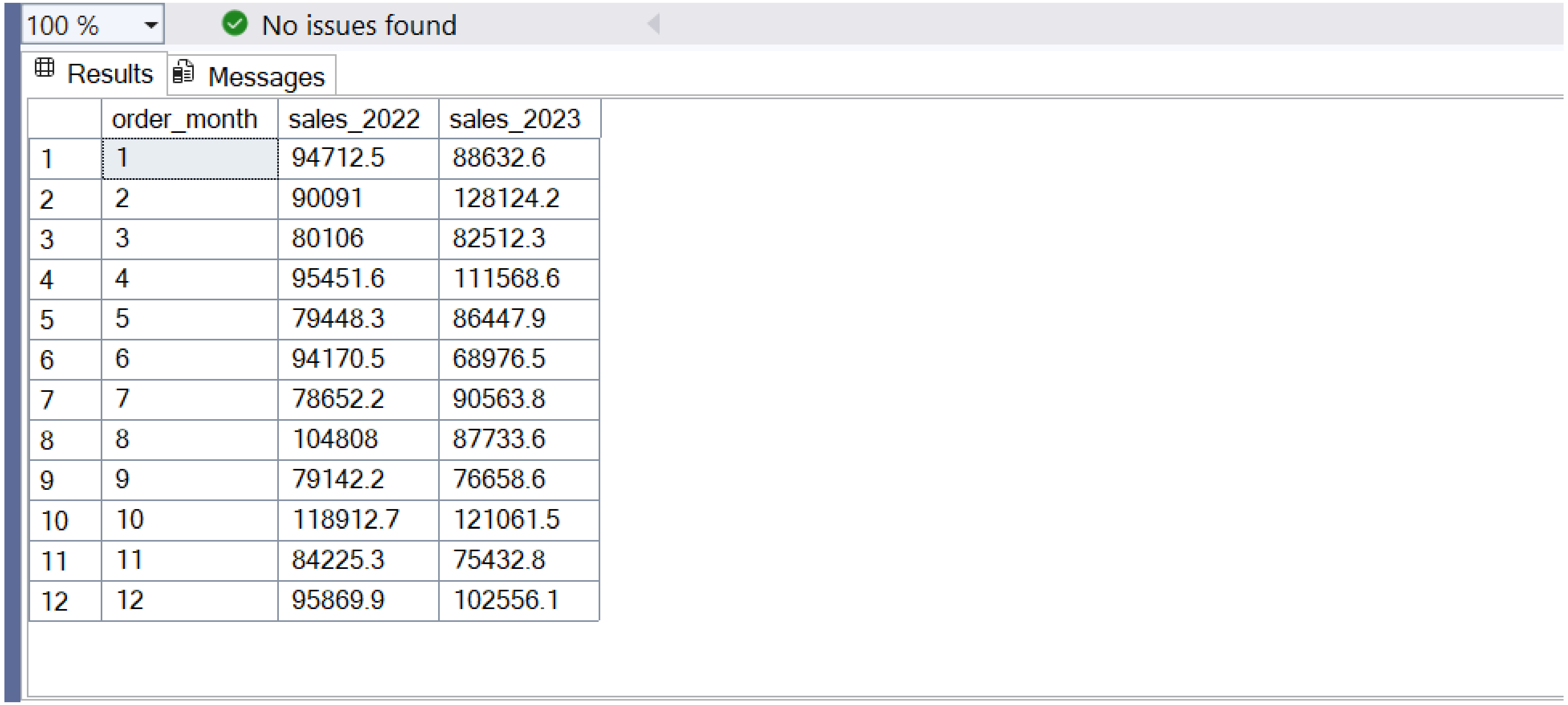
100 % No issues found

Results Messages

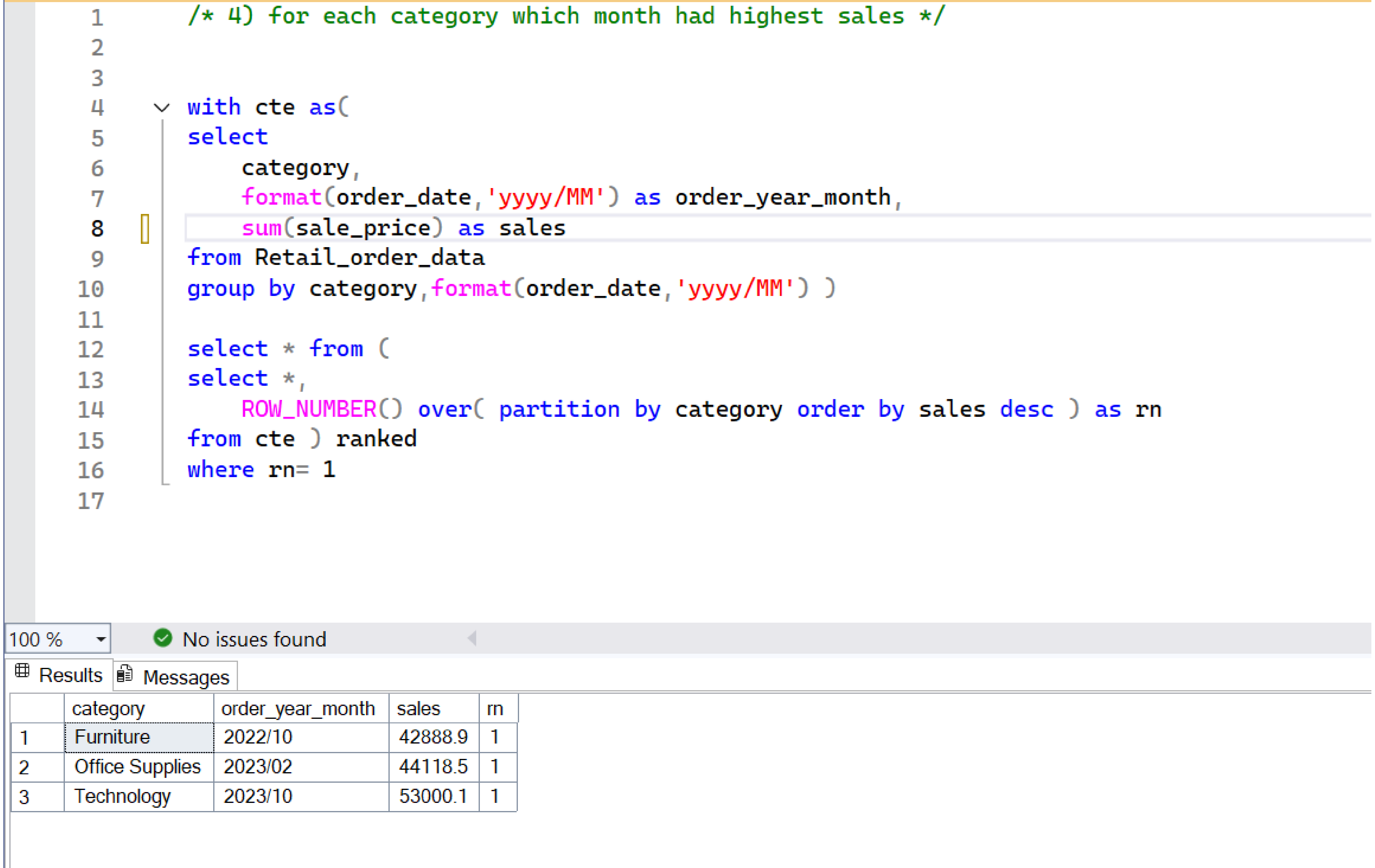
	region	product_id	sales	rn
1	Central	TEC-CO-10004722	16975	1
2	Central	TEC-MA-10000822	13770	2
3	Central	OFF-BI-10001120	11056.5	3
4	Central	OFF-BI-10000545	10132.7	4
5	East	TEC-CO-10004722	29099	1
6	East	TEC-MA-10001047	13767	2
7	East	FUR-BO-10004834	11274.1	3
8	East	OFF-BI-10001359	8463.6	4
9	South	TEC-MA-10002412	21734.4	1
10	South	TEC-MA-10001127	11116.4	2
11	South	OFF-BI-10001359	8053.2	3
12	South	TEC-MA-10004125	7840	4



```sql
/* 3) 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 Retail_order_data
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
```

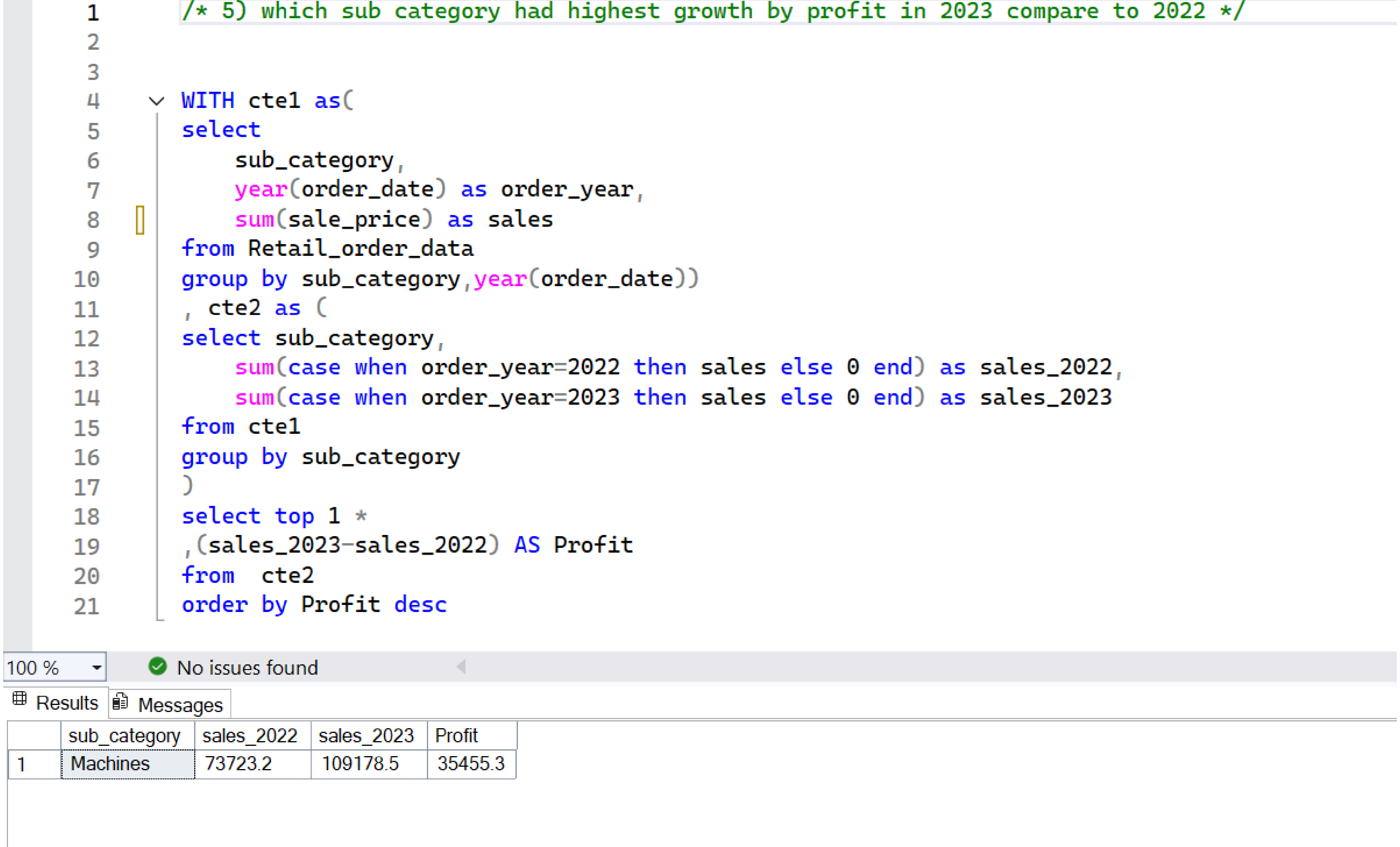


	order_month	sales_2022	sales_2023
1	1	94712.5	88632.6
2	2	90091	128124.2
3	3	80106	82512.3
4	4	95451.6	111568.6
5	5	79448.3	86447.9
6	6	94170.5	68976.5
7	7	78652.2	90563.8
8	8	104808	87733.6
9	9	79142.2	76658.6
10	10	118912.7	121061.5
11	11	84225.3	75432.8
12	12	95869.9	102556.1



```sql
/* 4) for each category which month had highest sales */


with cte as(
select
    category,
    format(order_date,'yyyy/MM') as order_year_month,
    sum(sale_price) as sales
from Retail_order_data
group by category,format(order_date,'yyyy/MM') )

select * from (
select *,
    ROW_NUMBER() over( partition by category order by sales desc ) as rn
from cte ) ranked
where rn= 1
```

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



```
/* 5) which sub category had highest growth by profit in 2023 compare to 2022 */


WITH cte1 as(
select
    sub_category,
    year(order_date) as order_year,
    sum(sale_price) as sales
from Retail_order_data
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 cte1
group by sub_category
)
select top 1 *
,(sales_2023-sales_2022) AS Profit
from  cte2
order by Profit desc
```

100 % No issues found

Results Messages

	sub_category	sales_2022	sales_2023	Profit
1	Machines	73723.2	109178.5	35455.3