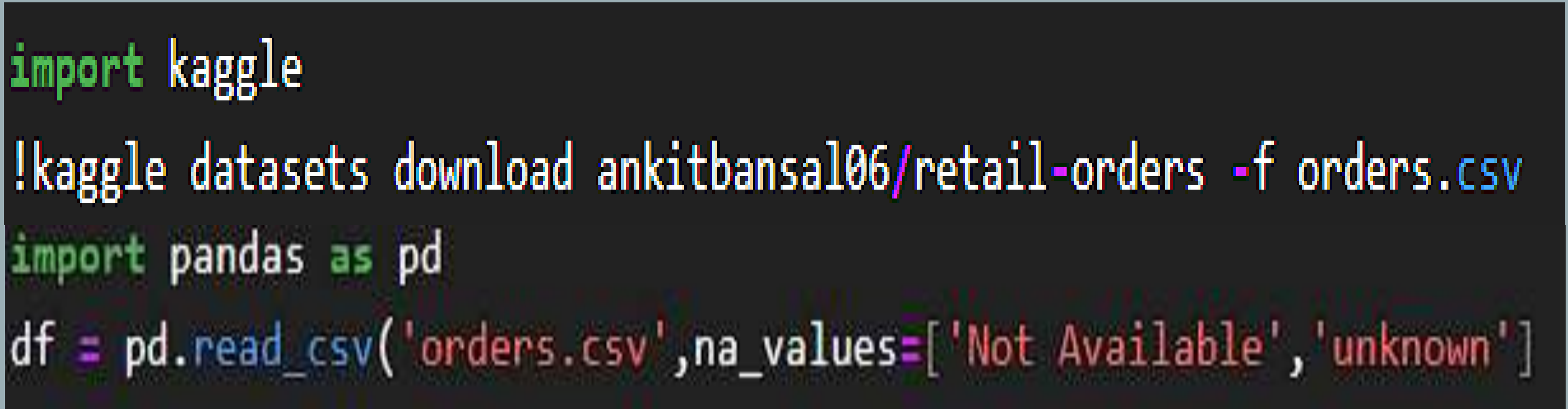
DATA ANALYSIS PROJECT USING (SQL AND PYTHON)

BY- SARTHAK TEGTA

1. EXTRATION

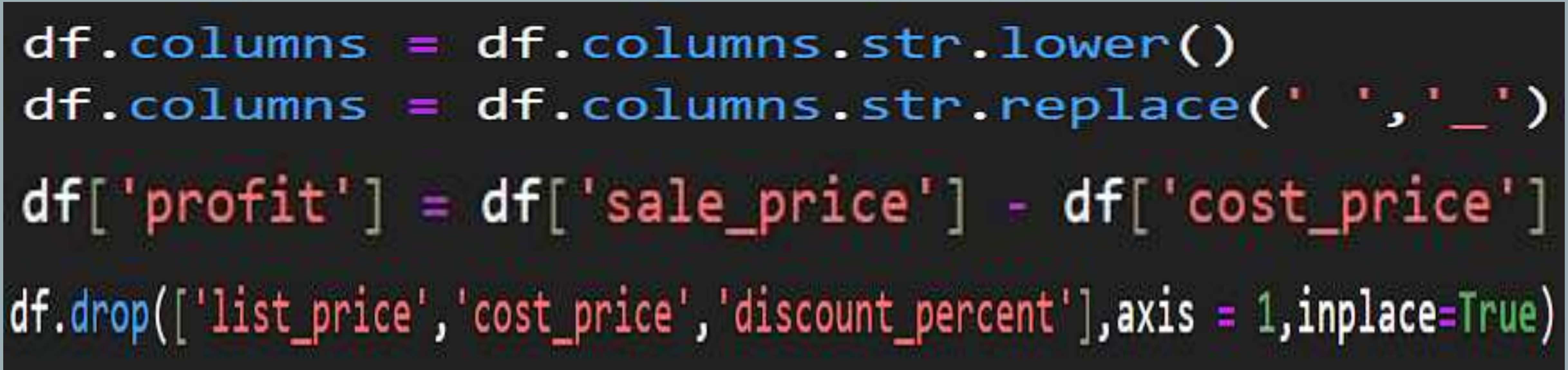
UTILIZED THE KAGGLE API TO PROGRAMMATICALLY EXTRACT DATA FROM A DATASET HOSTED ON KAGGLE. THE EXTRACTED DATA WAS THEN IMPORTED INTO A JUPYTER NOTEBOOK ENVIRONMENT USING THE PANDAS LIBRARY. THIS SETUP FACILITATED ADVANCED DATA TRANSFORMATION AND CLEANING OPERATIONS.



```
import kaggle
!kaggle datasets download ankitbansal06/retail-orders -f orders.csv
import pandas as pd
df = pd.read_csv('orders.csv',na_values=['Not Available','unknown']
```

2 TRANSFORMATION

PERFORMED A SERIES OF DATA PREPROCESSING STEPS IN THE JUPYTER NOTEBOOK USING THE PANDAS LIBRARY. THIS INCLUDED RENAMING COLUMNS TO IMPROVE READABILITY AND CONSISTENCY, ADDING NEW COLUMNS TO CAPTURE DERIVED METRICS OR ADDITIONAL INFORMATION, AND REMOVING COLUMNS DEEMED UNNECESSARY FOR THE SUBSEQUENT ANALYSIS TO STREAMLINE THE DATASET.



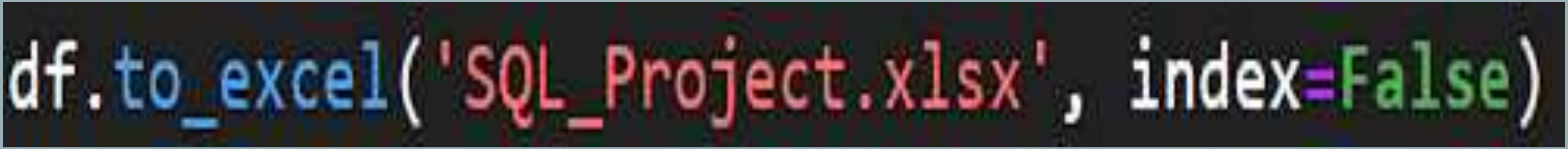
```
df.columns = df.columns.str.lower()
df.columns = df.columns.str.replace(' ','_')

df['profit'] = df['sale_price'] - df['cost_price']

df.drop(['list_price','cost_price','discount_percent'],axis = 1,inplace=True)
```

3 EXPORTED THE DATA TO AN EXCEL FILE

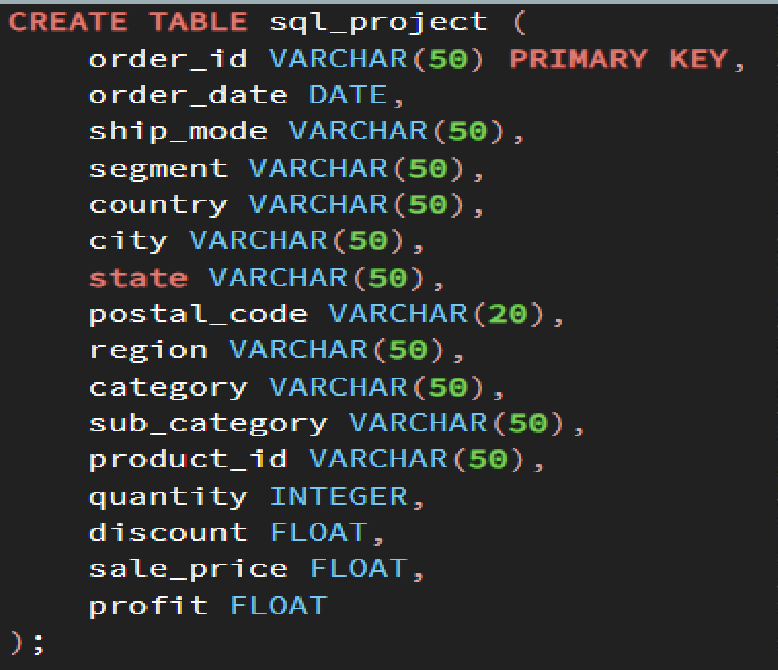
EXPORTED THE PROCESSED DATASET TO AN EXCEL FILE USING THE PANDAS .TO_EXCEL METHOD, ENABLING FURTHER IMPORT INTO POSTGRESQL FOR DATABASE MANAGEMENT AND QUERYING



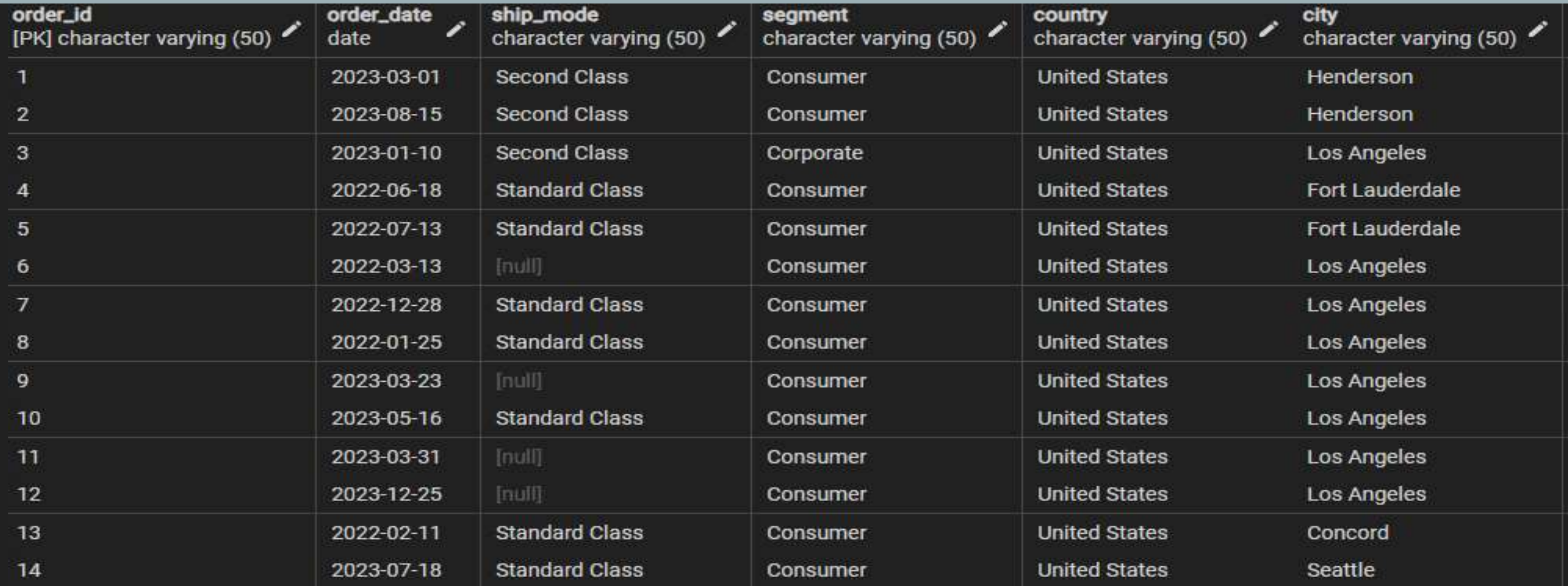
```
df.to_excel('SQL_Project.xlsx', index=False)
```

5 IMPORTED TO SQL

CREATED A TABLE IN POSTGRESQL WITH COLUMN NAMES MATCHING THOSE IN THE EXCEL FILE. SUBSEQUENTLY, THE DATA WAS IMPORTED INTO THIS POSTGRESQL TABLE TO COMPLETE THE DATA TRANSFER AND ENSURE CONSISTENCY BETWEEN THE EXCEL DATASET AND THE POSTGRESQL DATABASE SCHEMA.



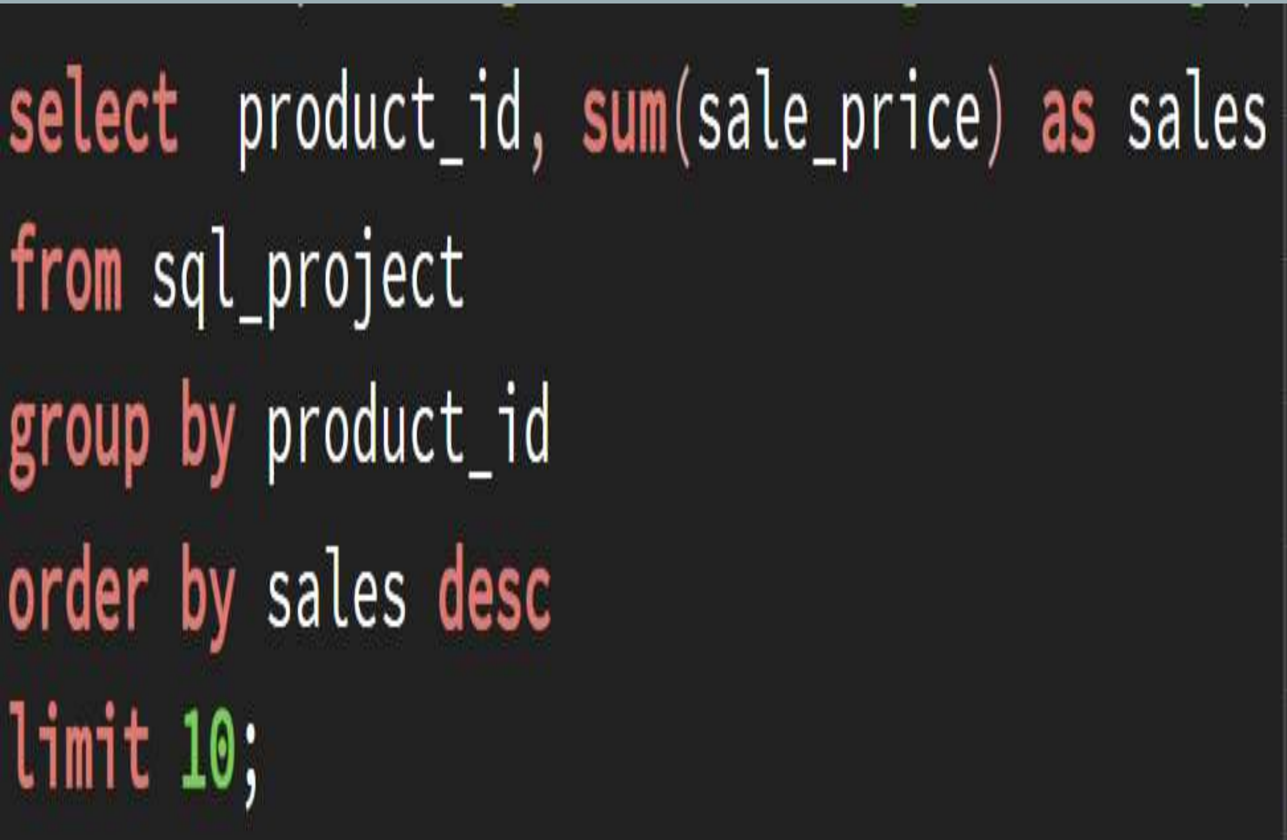
```
CREATE TABLE sql_project (
    order_id VARCHAR(50) PRIMARY KEY,
    order_date DATE,
    ship_mode VARCHAR(50),
    segment VARCHAR(50),
    country VARCHAR(50),
    city VARCHAR(50),
    state VARCHAR(50),
    postal_code VARCHAR(20),
    region VARCHAR(50),
    category VARCHAR(50),
    sub_category VARCHAR(50),
    product_id VARCHAR(50),
    quantity INTEGER,
    discount FLOAT,
    sale_price FLOAT,
    profit FLOAT
);
```



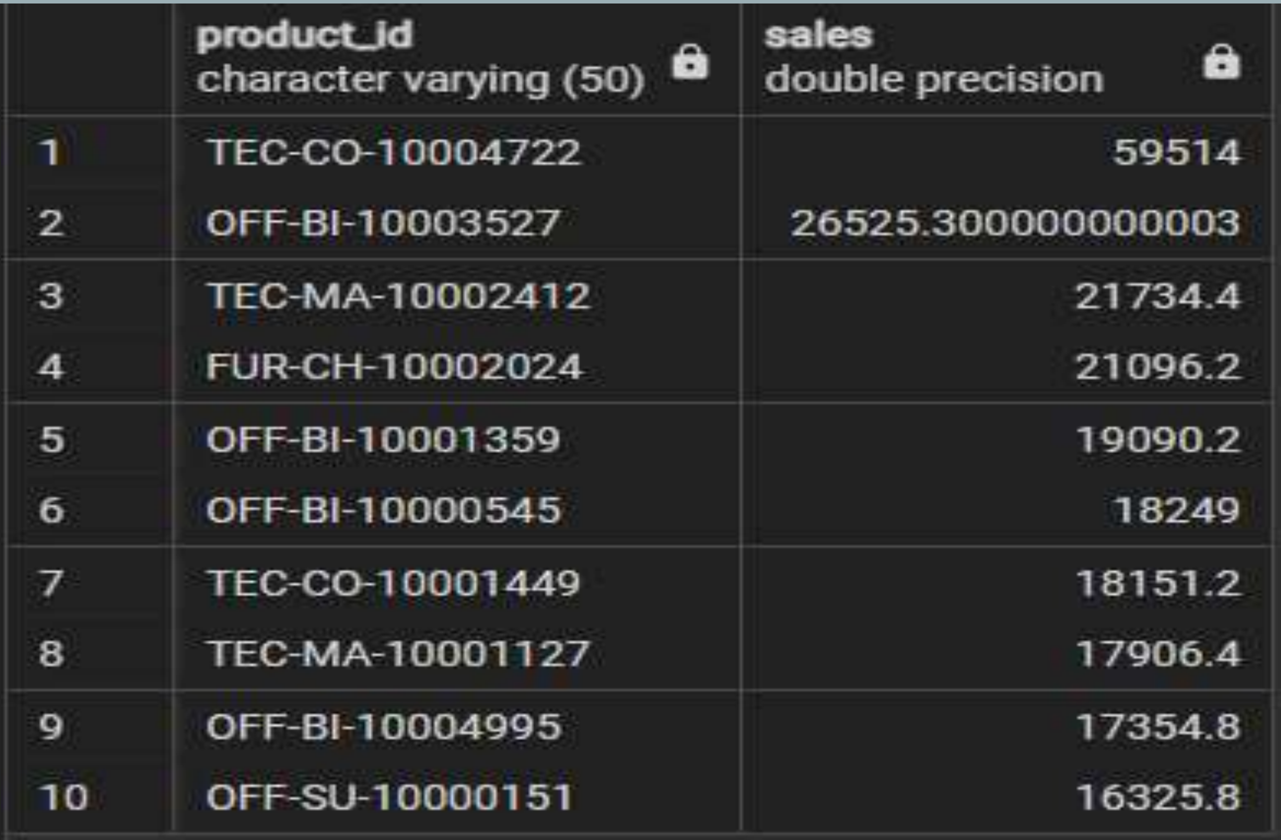
order_id	order_date	ship_mode	segment	country	city
[PK] character varying (50)	date	character varying (50)	character varying (50)	character varying (50)	character varying (50)
1	2023-03-01	Second Class	Consumer	United States	Henderson
2	2023-08-15	Second Class	Consumer	United States	Henderson
3	2023-01-10	Second Class	Corporate	United States	Los Angeles
4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale
5	2022-07-13	Standard Class	Consumer	United States	Fort Lauderdale
6	2022-03-13	[null]	Consumer	United States	Los Angeles
7	2022-12-28	Standard Class	Consumer	United States	Los Angeles
8	2022-01-25	Standard Class	Consumer	United States	Los Angeles
9	2023-03-23	[null]	Consumer	United States	Los Angeles
10	2023-05-16	Standard Class	Consumer	United States	Los Angeles
11	2023-03-31	[null]	Consumer	United States	Los Angeles
12	2023-12-25	[null]	Consumer	United States	Los Angeles
13	2022-02-11	Standard Class	Consumer	United States	Concord
14	2023-07-18	Standard Class	Consumer	United States	Seattle

6 QUERY 1

FIND TOP 10 HIGHEST REVENUE GENERATIING PRODUCTS



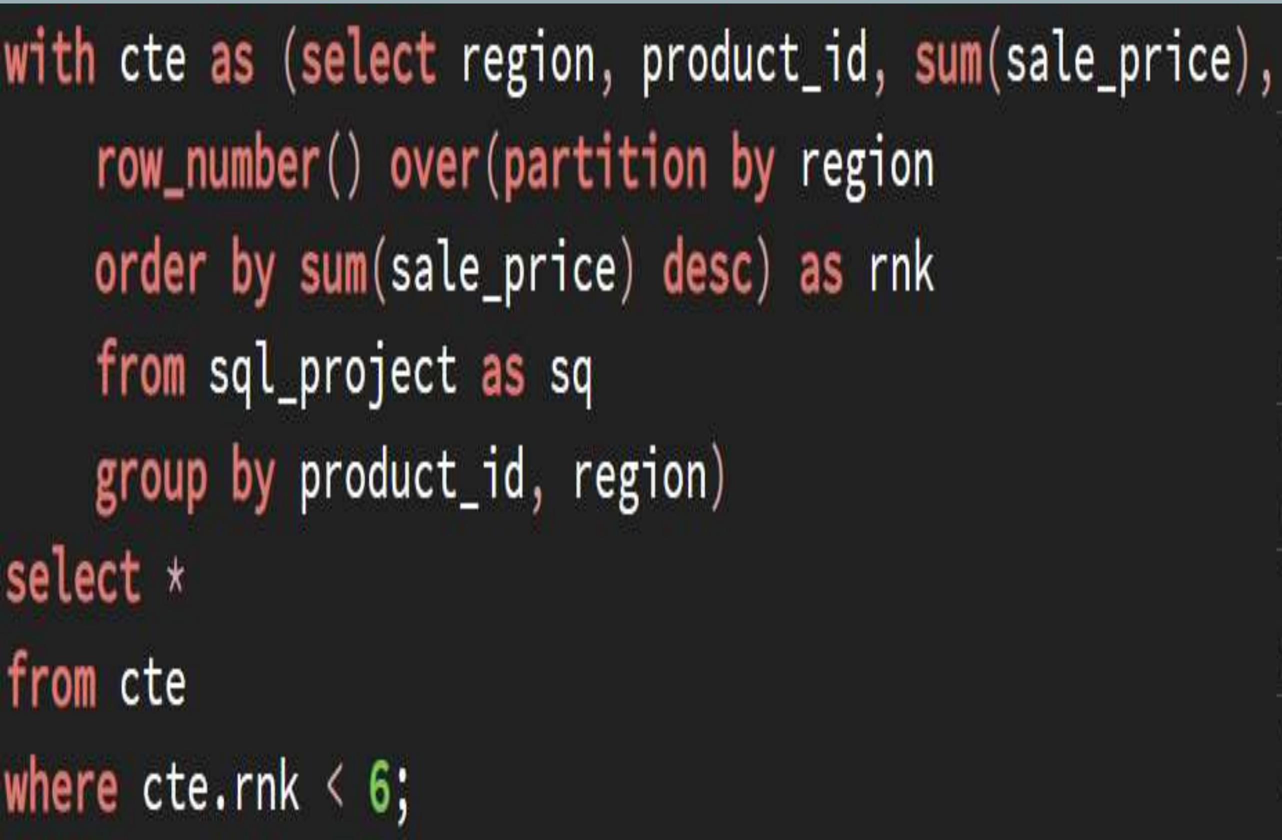
```
select  product_id, sum(sale_price) as sales
from sql_project
group by product_id
order by sales desc
limit 10;
```



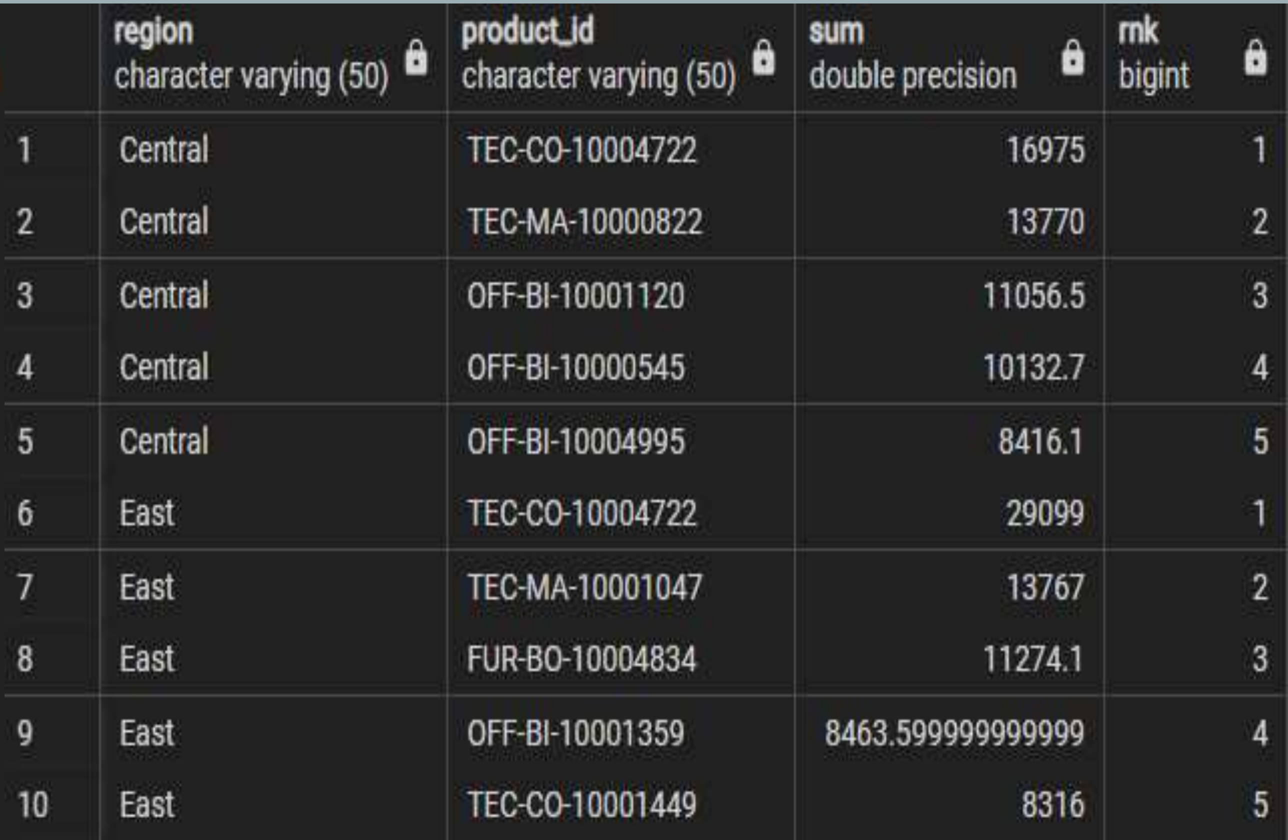
	product_id character varying (50)	sales double precision
1	TEC-CO-10004722	59514
2	OFF-BI-10003527	26525.300000000003
3	TEC-MA-10002412	21734.4
4	FUR-CH-10002024	21096.2
5	OFF-BI-10001359	19090.2
6	OFF-BI-10000545	18249
7	TEC-CO-10001449	18151.2
8	TEC-MA-10001127	17906.4
9	OFF-BI-10004995	17354.8
10	OFF-SU-10000151	16325.8

7 QUERY II

FIND TOP 5 HIGHEST SELLING PRODUCTS IN EACH REGION



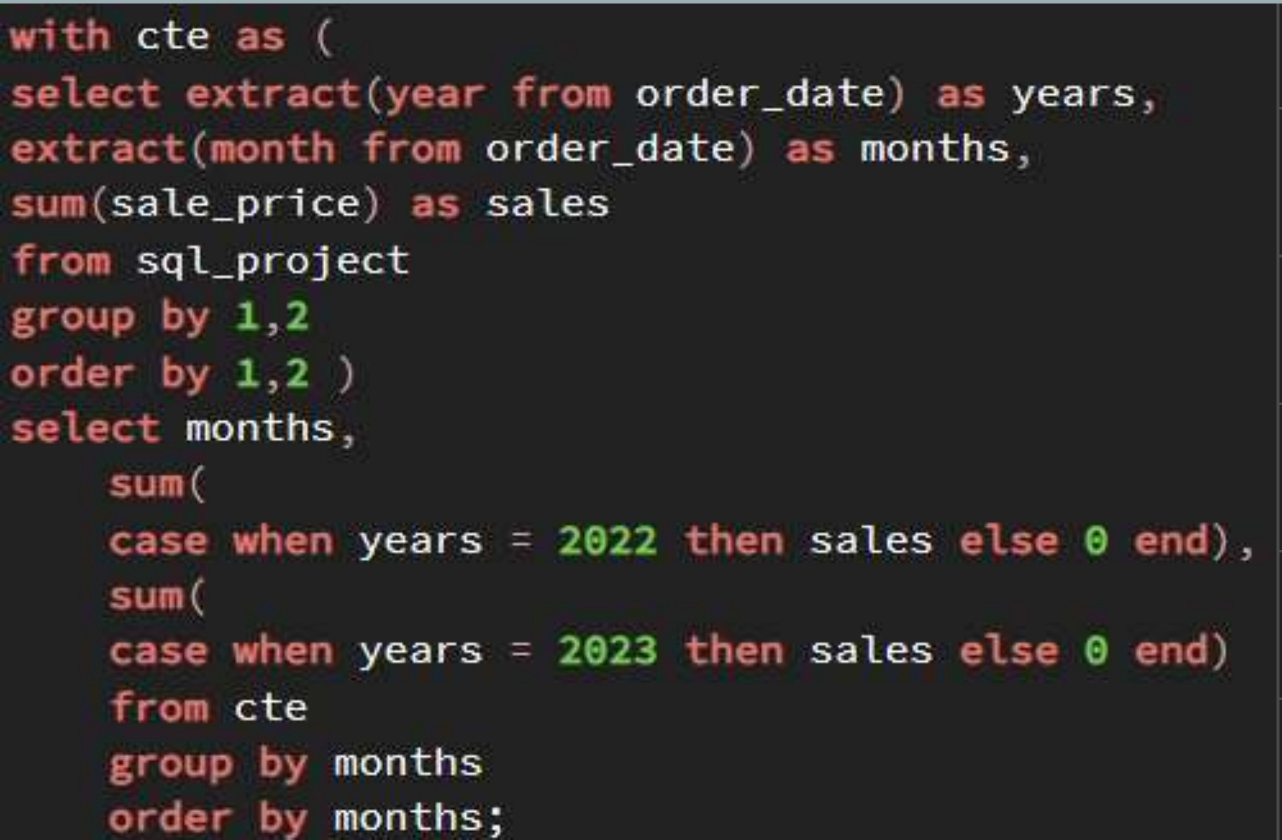
```
with cte as (select region, product_id, sum(sale_price),
    row_number() over(partition by region
    order by sum(sale_price) desc) as rnk
    from sql_project as sq
    group by product_id, region)
select *
from cte
where cte.rnk < 6;
```



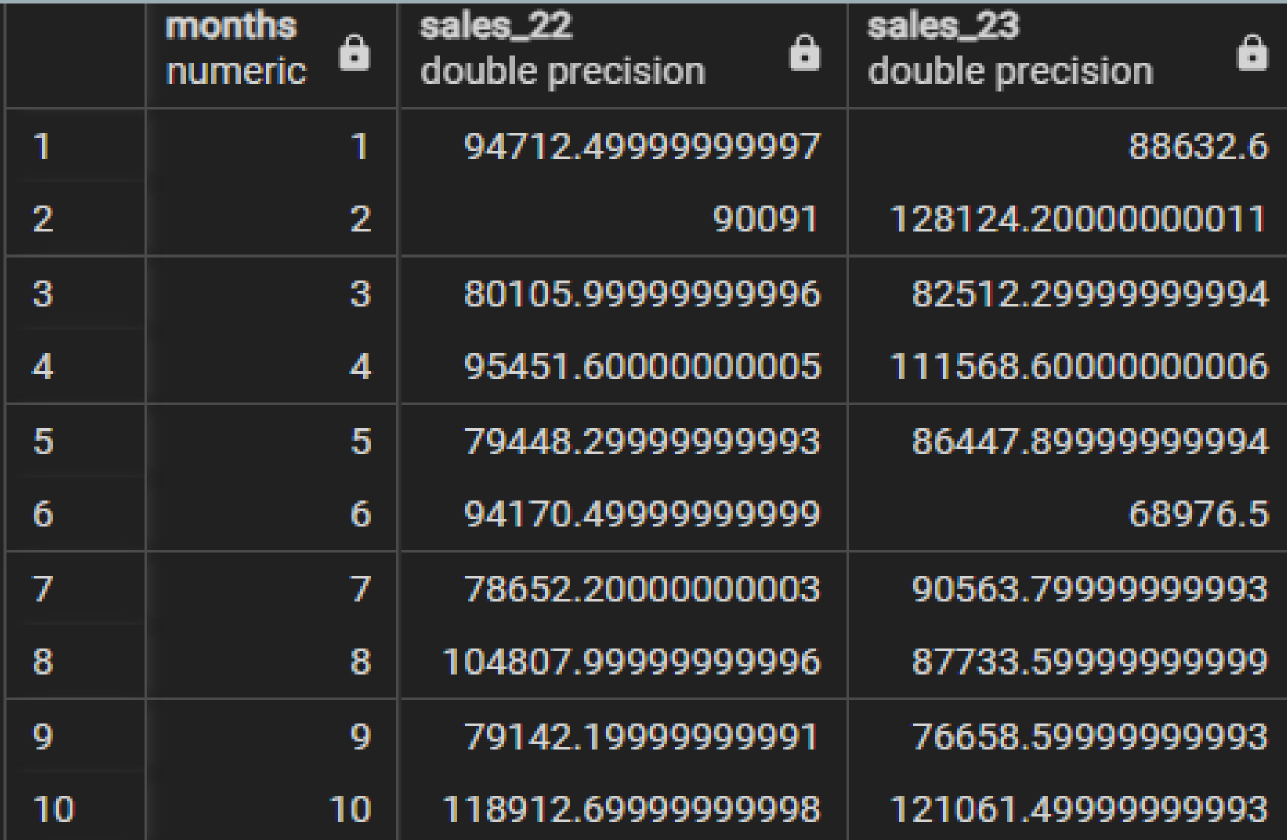
	region character varying (50)	product_id character varying (50)	sum double precision	rnk bigint
1	Central	TEC-CO-10004722	16975	1
2	Central	TEC-MA-10000822	13770	2
3	Central	OFF-BI-10001120	11056.5	3
4	Central	OFF-BI-10000545	10132.7	4
5	Central	OFF-BI-10004995	8416.1	5
6	East	TEC-CO-10004722	29099	1
7	East	TEC-MA-10001047	13767	2
8	East	FUR-BO-10004834	11274.1	3
9	East	OFF-BI-10001359	8463.599999999999	4
10	East	TEC-CO-10001449	8316	5

8 QUERY III

FIND MONTH OVER MONTH COMPARISION FOR 2022 AND 2023 SALES



```
with cte as (
select extract(year from order_date) as years,
extract(month from order_date) as months,
sum(sale_price) as sales
from sql_project
group by 1,2
order by 1,2 )
select months,
    sum(
    case when years = 2022 then sales else 0 end),
    sum(
    case when years = 2023 then sales else 0 end)
    from cte
    group by months
    order by months;
```



	months numeric	sales_22 double precision	sales_23 double precision
1	1	94712.49999999997	88632.6
2	2	90091	128124.2000000011
3	3	80105.99999999996	82512.29999999994
4	4	95451.60000000005	111568.60000000006
5	5	79448.29999999993	86447.89999999994
6	6	94170.49999999999	68976.5
7	7	78652.20000000003	90563.79999999993
8	8	104807.99999999996	87733.59999999999
9	9	79142.19999999991	76658.59999999993
10	10	118912.69999999998	121061.49999999993

9. SUMMARY

THIS PROJECT WAS AN END-TO-END DATA ANALYTICS PROJECT UTILIZING SQL AND PYTHON. THE PROJECT INVOLVED PERFORMING ETL TASKS WITH THE KAGGLE API, PANDAS, AND POSTGRESQL. THE OUTCOMES OF THE PROJECT DELIVERED VALUABLE INSIGHTS AND FACILITATED DATA-DRIVEN DECISION-MAKING FOR ANY COMPANY. THIS COMPREHENSIVE PROCESS ENSURED DATA INTEGRITY AND OPTIMIZED THE ANALYSIS WORKFLOW