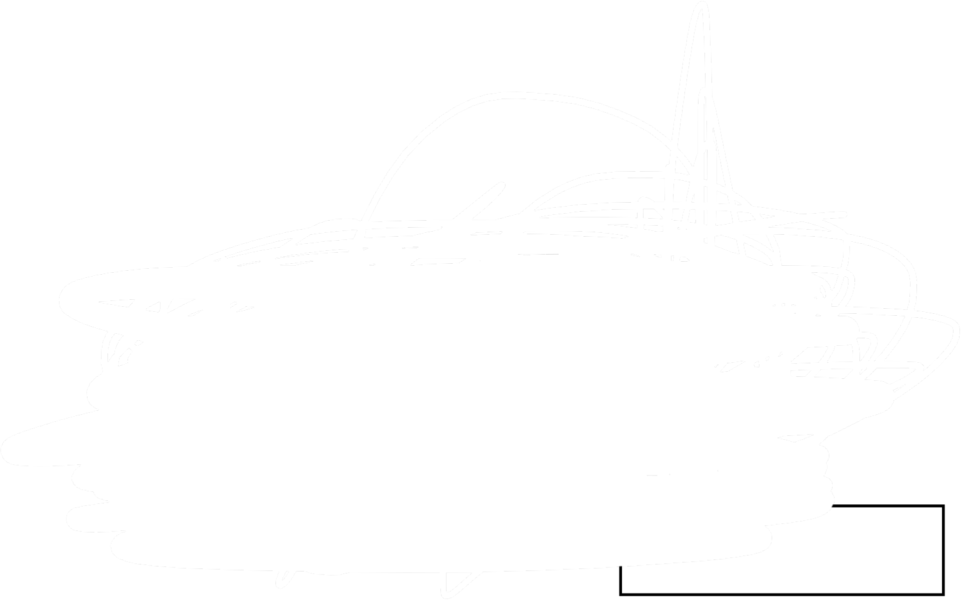
SQL Portfolio Project

Project-1



25th February 2023

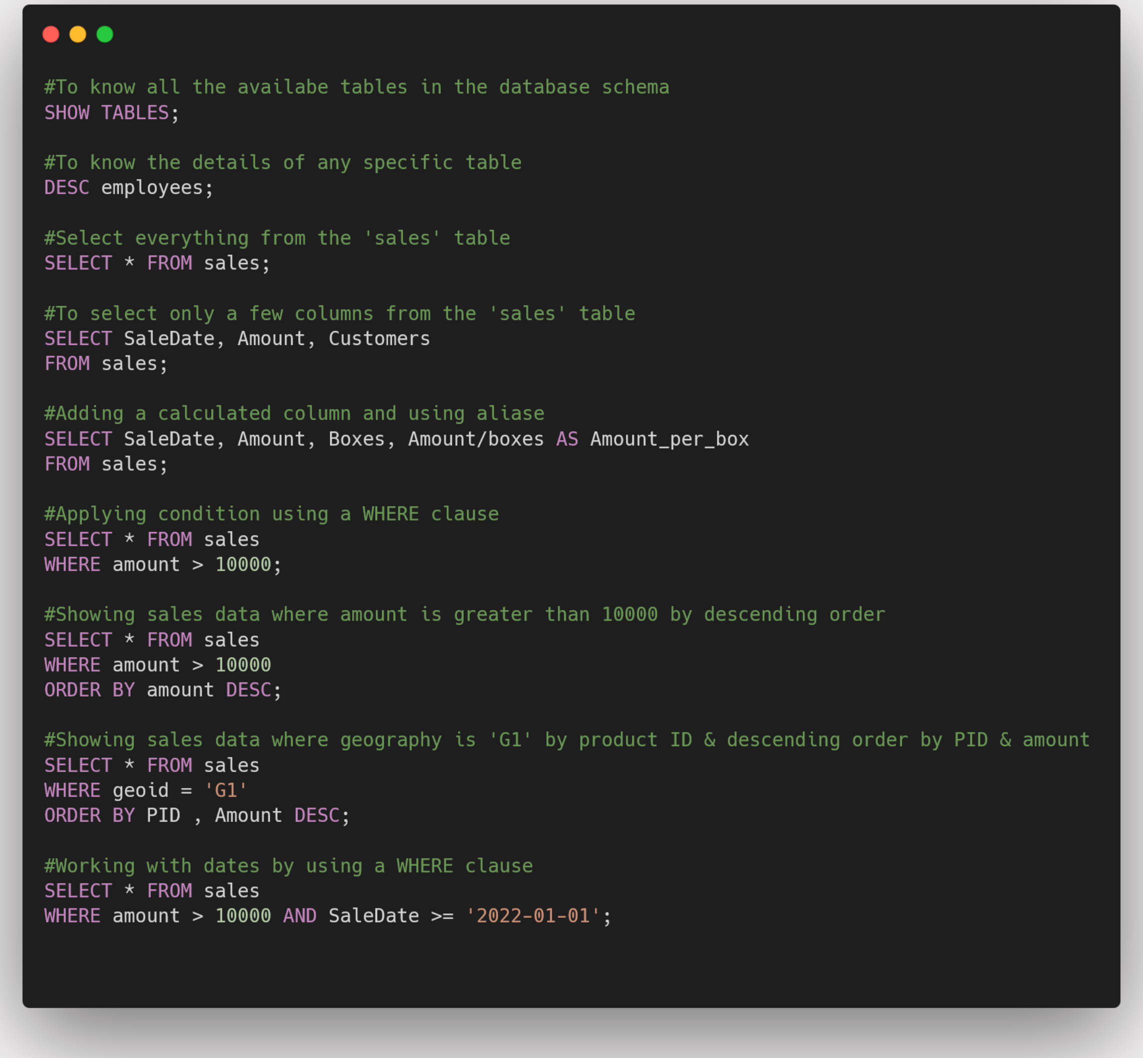
Table of contents

Introduction

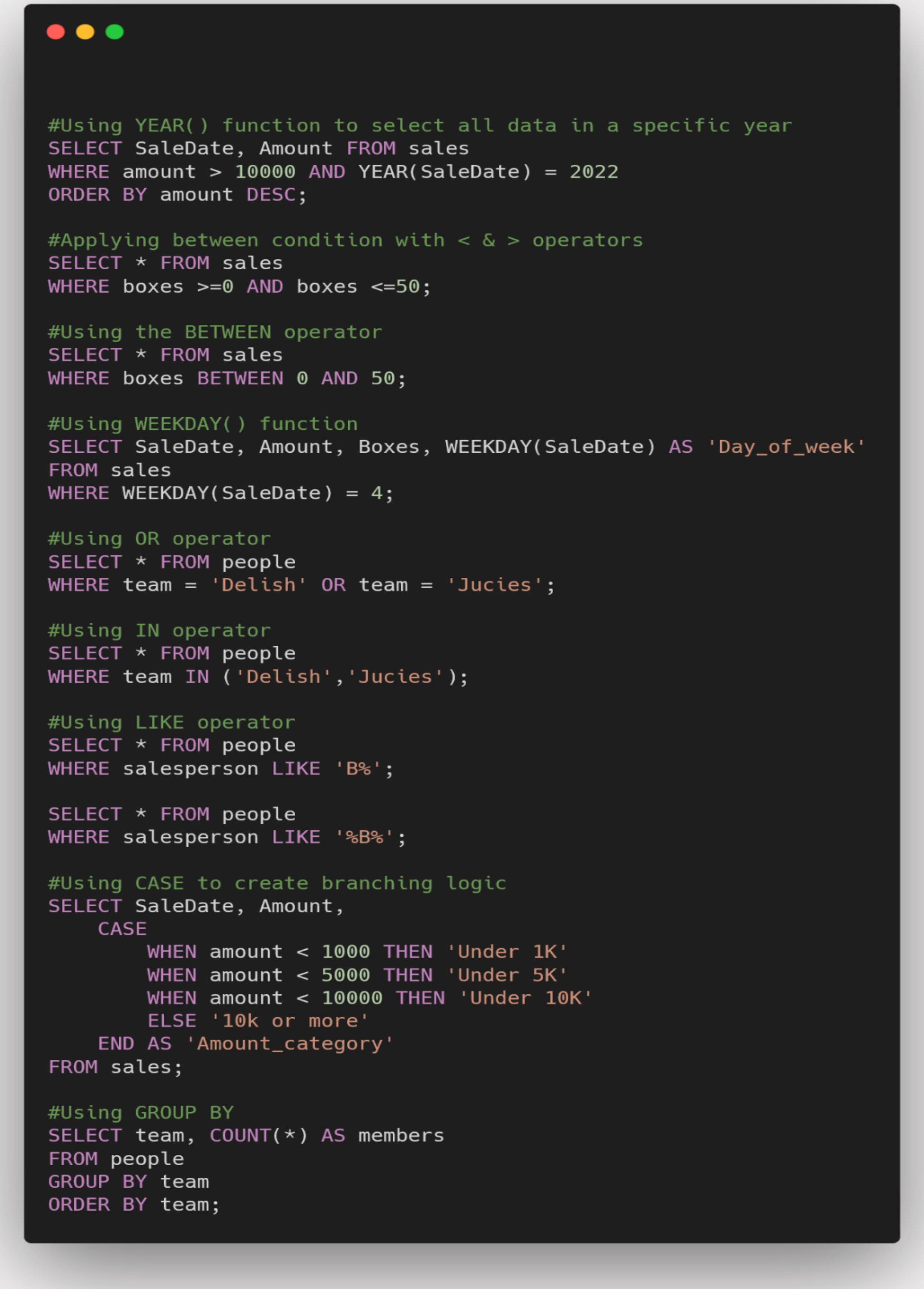
Welcome to my SQL project, which showcases my proficiency in using SQL for data analysis. Although I have used SQL in my work and completed projects during my master's program, this project marks my first SQL project for my portfolio. I have explored basic and advanced concepts of SQL and gained a solid foundation in writing complex queries to extract insights from a database. And I have used MySQL workbench for this project as I'm familiar with it. I am excited to share my work with you and hope you find it informative and engaging.

Note: Please note that only output snippets for advanced concepts have been included for brevity.

Part-1 | Basic Concepts & Queries

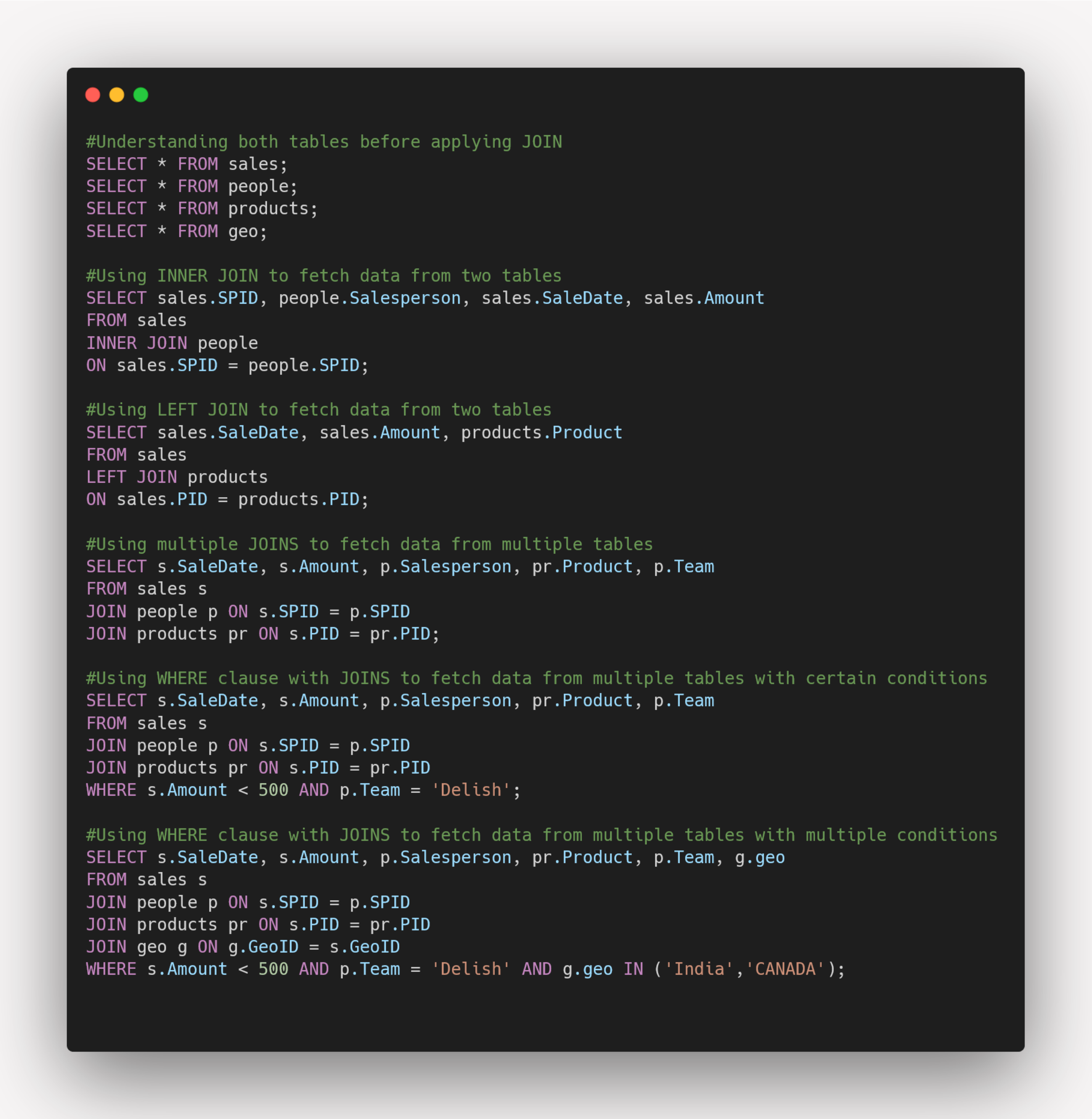


```
#To know all the availabe tables in the database schema
SHOW TABLES;

#To know the details of any specific table
DESC employees;

#Select everything from the 'sales' table
SELECT * FROM sales;

#To select only a few columns from the 'sales' table
SELECT SaleDate, Amount, Customers
FROM sales;

#Adding a calculated column and using aliase
SELECT SaleDate, Amount, Boxes, Amount/boxes AS Amount_per_box
FROM sales;

#Applying condition using a WHERE clause
SELECT * FROM sales
WHERE amount > 10000;

#Showing sales data where amount is greater than 10000 by descending order
SELECT * FROM sales
WHERE amount > 10000
ORDER BY amount DESC;

#Showing sales data where geography is 'G1' by product ID & descending order by PID & amount
SELECT * FROM sales
WHERE geoid = 'G1'
ORDER BY PID , Amount DESC;

#Working with dates by using a WHERE clause
SELECT * FROM sales
WHERE amount > 10000 AND SaleDate >= '2022-01-01';
```

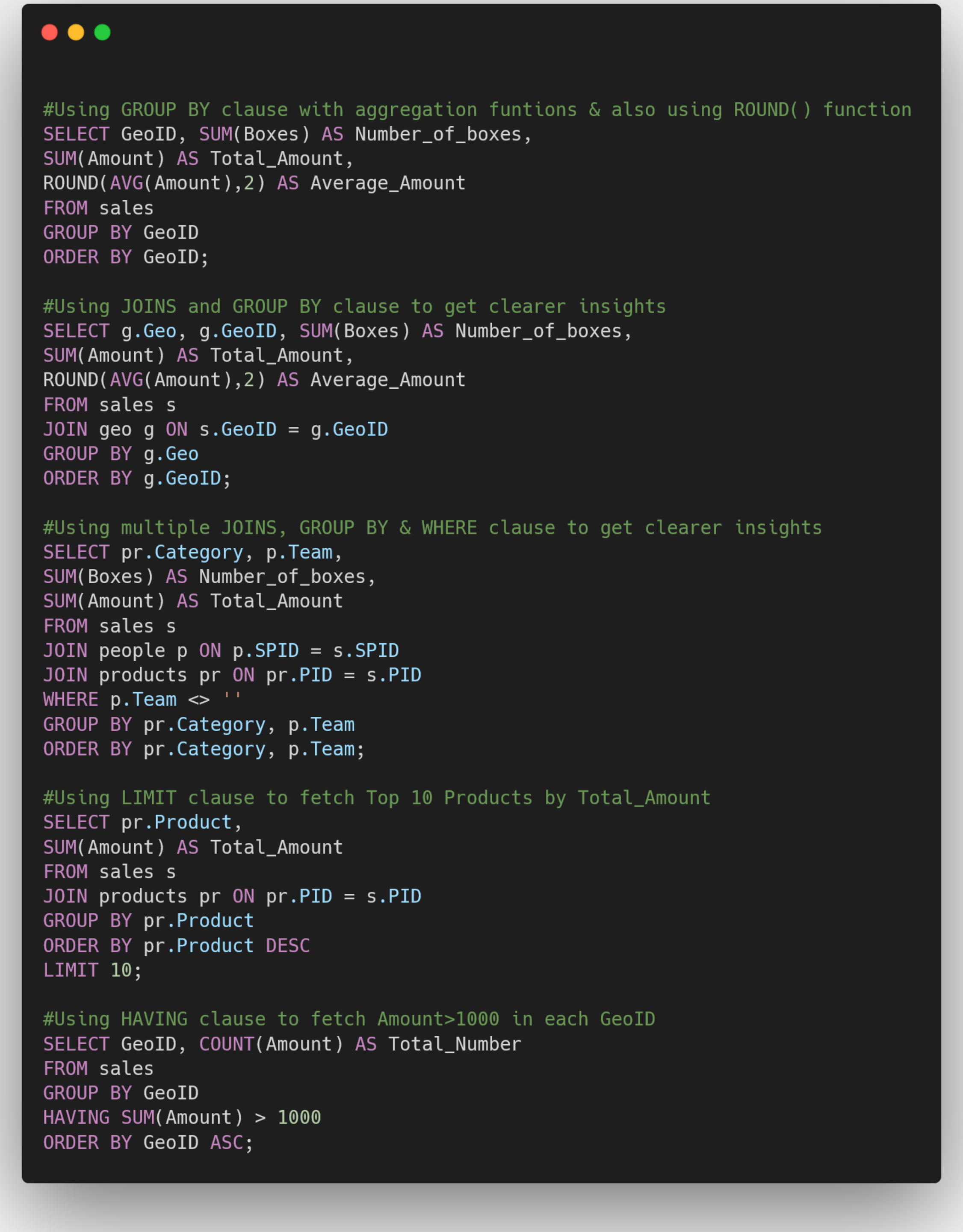


```
#Using YEAR() function to select all data in a specific year
SELECT SaleDate, Amount FROM sales
WHERE amount > 10000 AND YEAR(SaleDate) = 2022
ORDER BY amount DESC;

#Applying between condition with < & > operators
SELECT * FROM sales
WHERE boxes >=0 AND boxes <=50;

#Using the BETWEEN operator
SELECT * FROM sales
WHERE boxes BETWEEN 0 AND 50;

#Using WEEKDAY() function
SELECT SaleDate, Amount, Boxes, WEEKDAY(SaleDate) AS 'Day_of_week'
FROM sales
WHERE WEEKDAY(SaleDate) = 4;

#Using OR operator
SELECT * FROM people
WHERE team = 'Delish' OR team = 'Jucies';

#Using IN operator
SELECT * FROM people
WHERE team IN ('Delish','Jucies');

#Using LIKE operator
SELECT * FROM people
WHERE salesperson LIKE 'B%';

SELECT * FROM people
WHERE salesperson LIKE '%B%';

#Using CASE to create branching logic
SELECT SaleDate, Amount,
    CASE
        WHEN amount < 1000 THEN 'Under 1K'
        WHEN amount < 5000 THEN 'Under 5K'
        WHEN amount < 10000 THEN 'Under 10K'
        ELSE '10k or more'
    END AS 'Amount_category'
FROM sales;

#Using GROUP BY
SELECT team, COUNT(*) AS members
FROM people
GROUP BY team
ORDER BY team;
```

Part-2 | Intermediate Concepts & Queries

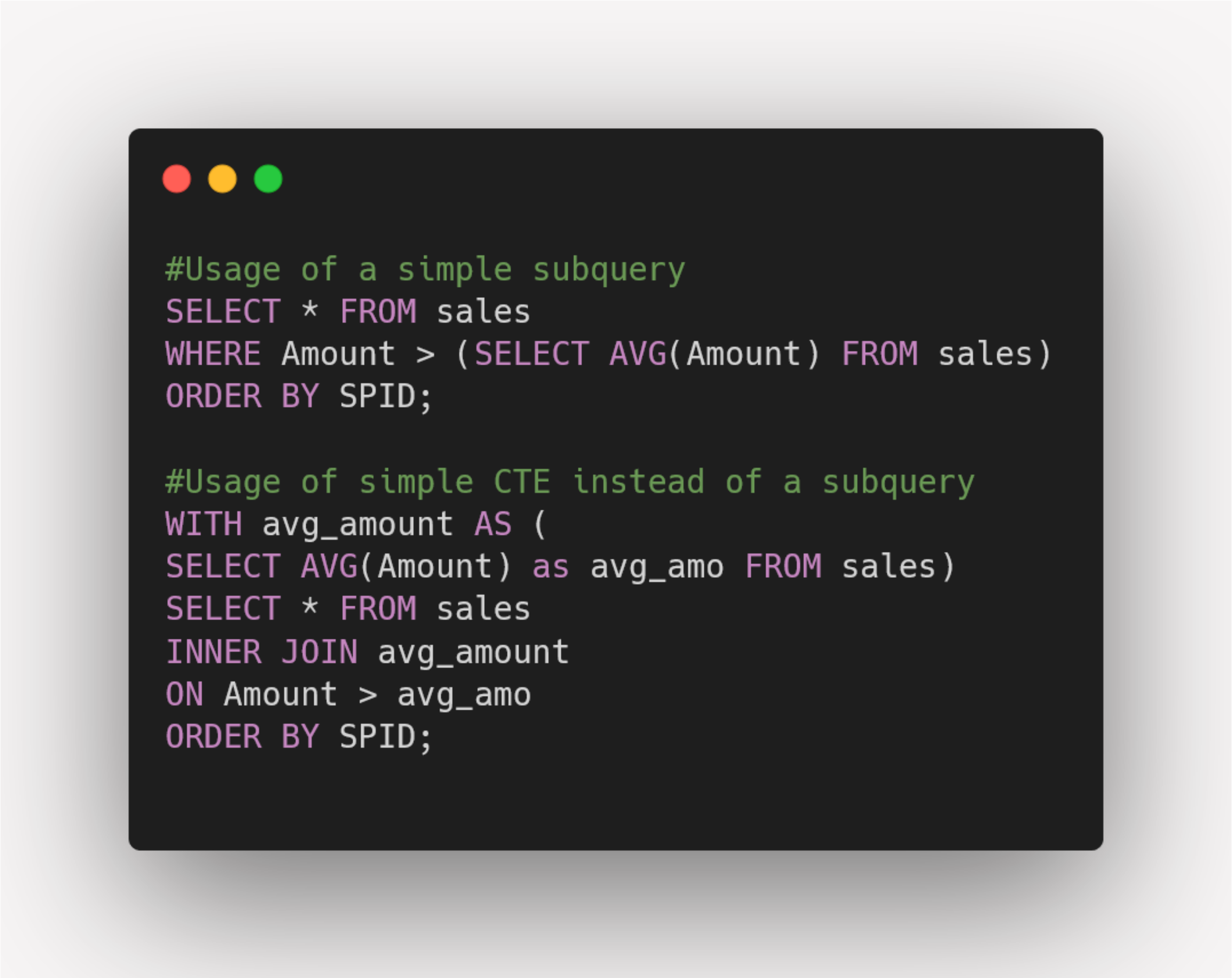


```
#Understanding both tables before applying JOIN
SELECT * FROM sales;
SELECT * FROM people;
SELECT * FROM products;
SELECT * FROM geo;

#Using INNER JOIN to fetch data from two tables
SELECT sales.SPID, people.Salesperson, sales.SaleDate, sales.Amount
FROM sales
INNER JOIN people
ON sales.SPID = people.SPID;

#Using LEFT JOIN to fetch data from two tables
SELECT sales.SaleDate, sales.Amount, products.Product
FROM sales
LEFT JOIN products
ON sales.PID = products.PID;

#Using multiple JOINS to fetch data from multiple tables
SELECT s.SaleDate, s.Amount, p.Salesperson, pr.Product, p.Team
FROM sales s
JOIN people p ON s.SPID = p.SPID
JOIN products pr ON s.PID = pr.PID;

#Using WHERE clause with JOINS to fetch data from multiple tables with certain conditions
SELECT s.SaleDate, s.Amount, p.Salesperson, pr.Product, p.Team
FROM sales s
JOIN people p ON s.SPID = p.SPID
JOIN products pr ON s.PID = pr.PID
WHERE s.Amount < 500 AND p.Team = 'Delish';

#Using WHERE clause with JOINS to fetch data from multiple tables with multiple conditions
SELECT s.SaleDate, s.Amount, p.Salesperson, pr.Product, p.Team, g.geo
FROM sales s
JOIN people p ON s.SPID = p.SPID
JOIN products pr ON s.PID = pr.PID
JOIN geo g ON g.GeoID = s.GeoID
WHERE s.Amount < 500 AND p.Team = 'Delish' AND g.geo IN ('India','CANADA');
```

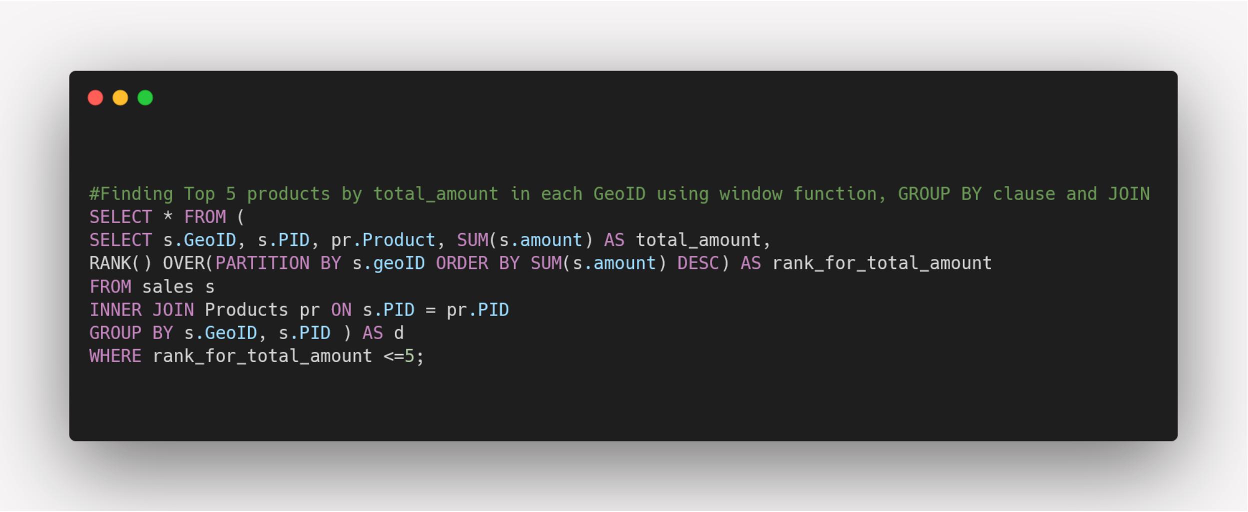


```sql
#Using GROUP BY clause with aggregation funtions & also using ROUND() function
SELECT GeoID, SUM(Boxes) AS Number_of_boxes,
SUM(Amount) AS Total_Amount,
ROUND(AVG(Amount),2) AS Average_Amount
FROM sales
GROUP BY GeoID
ORDER BY GeoID;

#Using JOINS and GROUP BY clause to get clearer insights
SELECT g.Geo, g.GeoID, SUM(Boxes) AS Number_of_boxes,
SUM(Amount) AS Total_Amount,
ROUND(AVG(Amount),2) AS Average_Amount
FROM sales s
JOIN geo g ON s.GeoID = g.GeoID
GROUP BY g.Geo
ORDER BY g.GeoID;

#Using multiple JOINS, GROUP BY & WHERE clause to get clearer insights
SELECT pr.Category, p.Team,
SUM(Boxes) AS Number_of_boxes,
SUM(Amount) AS Total_Amount
FROM sales s
JOIN people p ON p.SPID = s.SPID
JOIN products pr ON pr.PID = s.PID
WHERE p.Team <> ''
GROUP BY pr.Category, p.Team
ORDER BY pr.Category, p.Team;

#Using LIMIT clause to fetch Top 10 Products by Total_Amount
SELECT pr.Product,
SUM(Amount) AS Total_Amount
FROM sales s
JOIN products pr ON pr.PID = s.PID
GROUP BY pr.Product
ORDER BY pr.Product DESC
LIMIT 10;

#Using HAVING clause to fetch Amount>1000 in each GeoID
SELECT GeoID, COUNT(Amount) AS Total_Number
FROM sales
GROUP BY GeoID
HAVING SUM(Amount) > 1000
ORDER BY GeoID ASC;
```

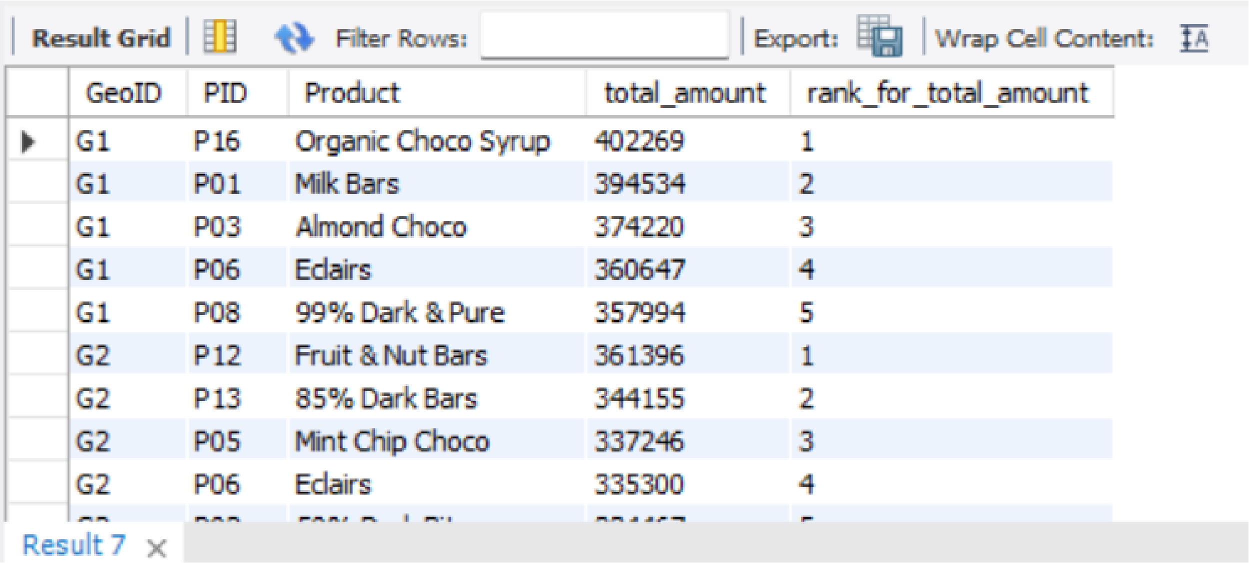
Part-3 | Advanced Concepts & Queries



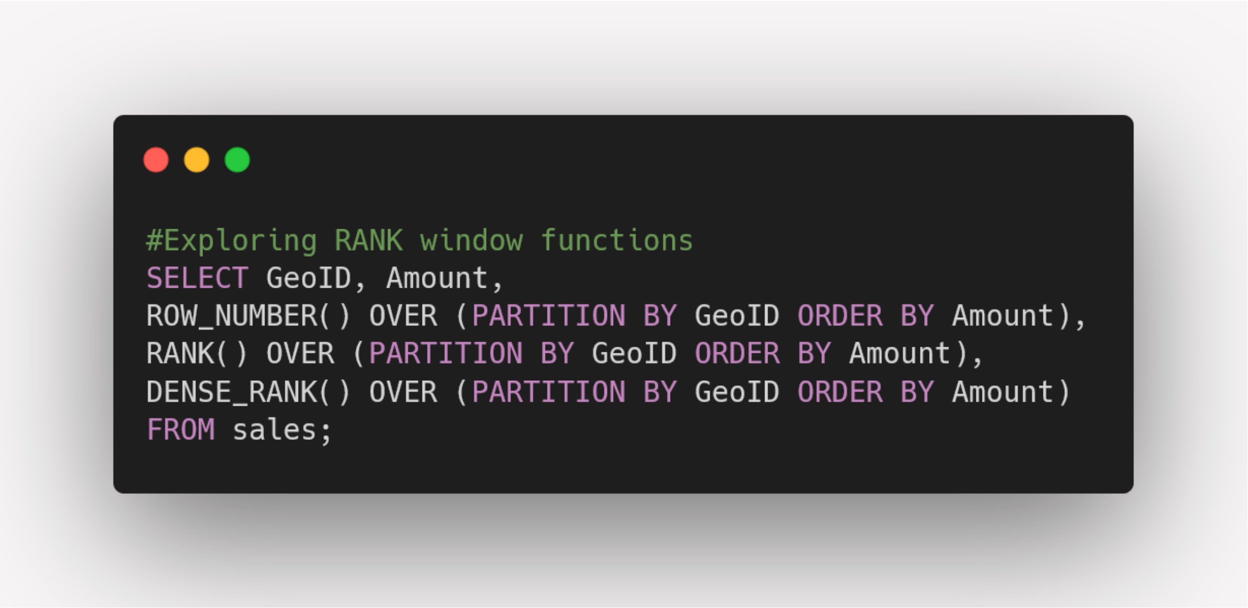
```
#Usage of a simple subquery
SELECT * FROM sales
WHERE Amount > (SELECT AVG(Amount) FROM sales)
ORDER BY SPID;

#Usage of simple CTE instead of a subquery
WITH avg_amount AS (
SELECT AVG(Amount) as avg_amo FROM sales)
SELECT * FROM sales
INNER JOIN avg_amount
ON Amount > avg_amo
ORDER BY SPID;
```



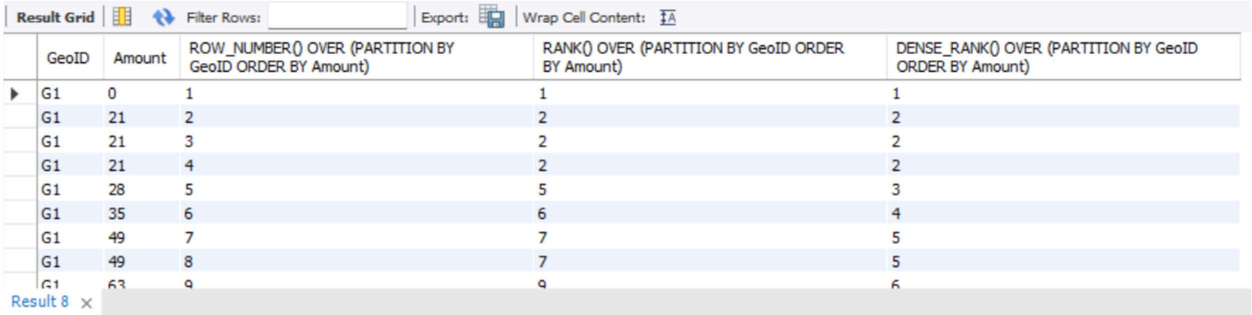
```
#Finding Top 5 products by total_amount in each GeoID using window function, GROUP BY clause and JOIN
SELECT * FROM (
SELECT s.GeoID, s.PID, pr.Product, SUM(s.amount) AS total_amount,
RANK() OVER(PARTITION BY s.geoID ORDER BY SUM(s.amount) DESC) AS rank_for_total_amount
FROM sales s
INNER JOIN Products pr ON s.PID = pr.PID
GROUP BY s.GeoID, s.PID ) AS d
WHERE rank_for_total_amount <=5;
```



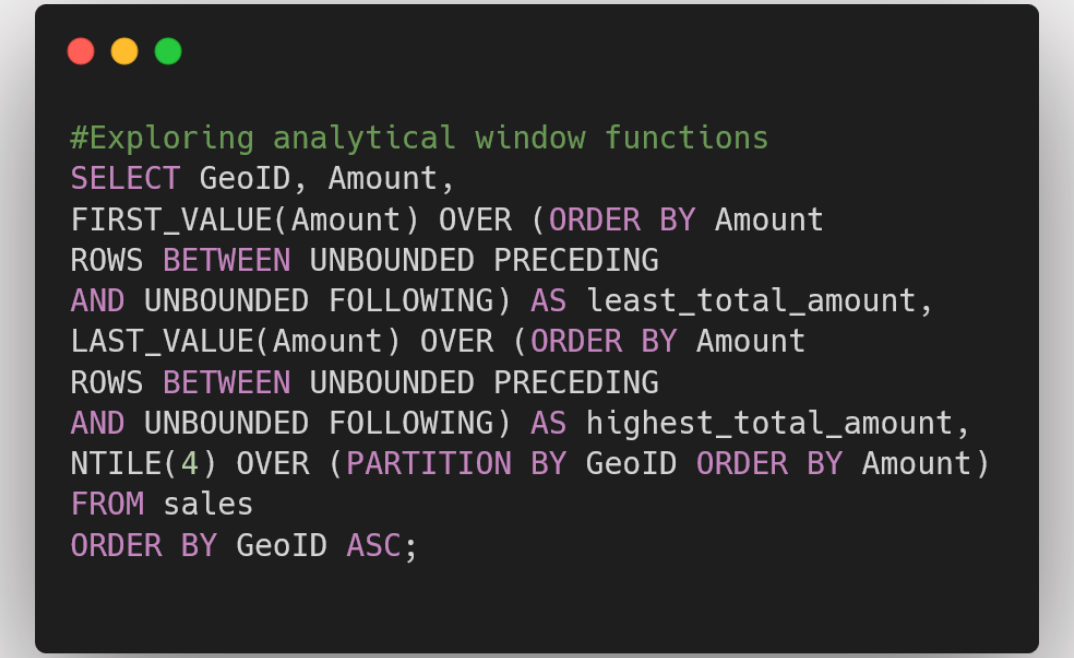
GeoID	PID	Product	total_amount	rank_for_total_amount
G1	P16	Organic Choco Syrup	402269	1
G1	P01	Milk Bars	394534	2
G1	P03	Almond Choco	374220	3
G1	P06	Eclairs	360647	4
G1	P08	99% Dark & Pure	357994	5
G2	P12	Fruit & Nut Bars	361396	1
G2	P13	85% Dark Bars	344155	2
G2	P05	Mint Chip Choco	337246	3
G2	P06	Eclairs	335300	4



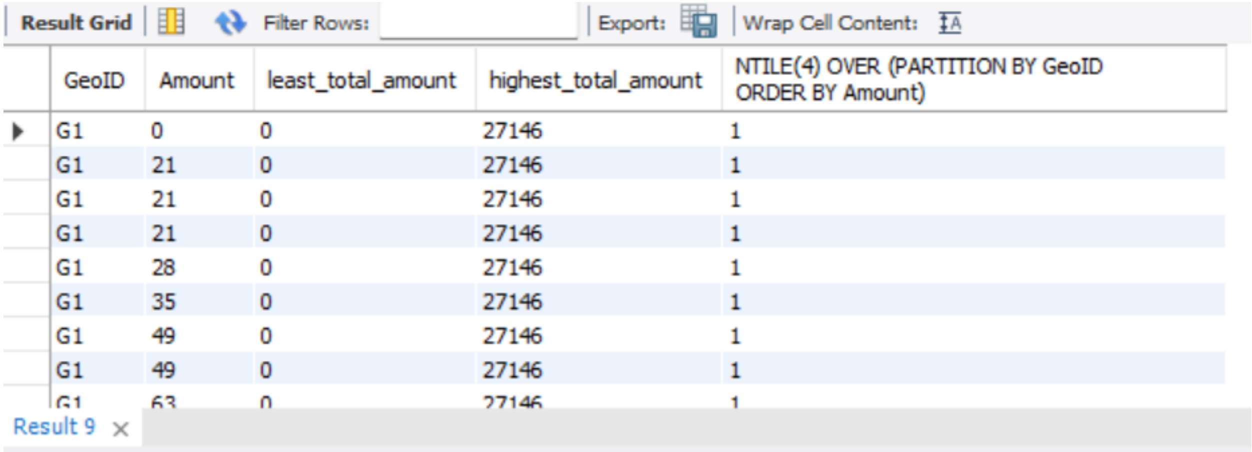
```
#Exploring RANK window functions
SELECT GeoID, Amount,
ROW_NUMBER() OVER (PARTITION BY GeoID ORDER BY Amount),
RANK() OVER (PARTITION BY GeoID ORDER BY Amount),
DENSE_RANK() OVER (PARTITION BY GeoID ORDER BY Amount)
FROM sales;
```



GeoID	Amount	ROW_NUMBER() OVER (PARTITION BY GeoID ORDER BY Amount)	RANK() OVER (PARTITION BY GeoID ORDER BY Amount)	DENSE_RANK() OVER (PARTITION BY GeoID ORDER BY Amount)
G1	0	1	1	1
G1	21	2	2	2
G1	21	3	2	2
G1	21	4	2	2
G1	28	5	5	3
G1	35	6	6	4
G1	49	7	7	5
G1	49	8	7	5
G1	63	9	9	6



```
#Exploring analytical window functions
SELECT GeoID, Amount,
FIRST_VALUE(Amount) OVER (ORDER BY Amount
ROWS BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) AS least_total_amount,
LAST_VALUE(Amount) OVER (ORDER BY Amount
ROWS BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) AS highest_total_amount,
NTILE(4) OVER (PARTITION BY GeoID ORDER BY Amount)
FROM sales
ORDER BY GeoID ASC;
```



GeoID	Amount	least_total_amount	highest_total_amount	NTILE(4) OVER (PARTITION BY GeoID ORDER BY Amount)
G1	0	0	27146	1
G1	21	0	27146	1
G1	21	0	27146	1
G1	21	0	27146	1
G1	28	0	27146	1
G1	35	0	27146	1
G1	49	0	27146	1
G1	49	0	27146	1
G1	63	0	27146	1

Conclusion

In conclusion, this SQL project has been a valuable learning experience for me, allowing me to showcase my skills in using SQL for data analysis. I have gained a solid understanding of the fundamental concepts of SQL and have progressed to advanced concepts. Through this project, I have demonstrated my ability to write complex queries to extract insights from a database, which is an essential skill for any data analyst. I am proud of the work I have done and excited to continue honing my skills in SQL to tackle even more complex projects in the future. Thank you for taking the time to review my project.