**SQL QUERIES**

**1. Count of transactions provided.**

**SQL Query:**

SELECT

COUNT(DISTINCT transaction\_id) count\_transactions

FROM

coffee\_shop\_sales;

**OUTPUT:**



**2. Store id and their corresponding locations.**

**SQL QUERY:**

SELECT

DISTINCT store\_id, store\_location

FROM

coffee\_shop\_sales;

**OUTPUT:**

****

**3. Category of products available in each store.**

**SQL QUERY:**

SELECT

DISTINCT store\_id, product\_category

FROM

coffee\_shop\_sales

ORDER BY

store\_id;

**OUTPUT:**

****

**4. Product category and their corresponding types.**

**SQL QUERY:**

SELECT

DISTINCT Product\_category, product\_type

FROM

coffee\_shop\_sales

ORDER BY

Product\_category;

**OUTPUT:**

****

**5. count of product categroies and product type in each store.**

**SQL QUERY:**

SELECT

COUNT(DISTINCT Product\_category) count\_product\_category,

COUNT(DISTINCT product\_type) count\_product\_type

FROM

coffee\_shop\_sales;

**OUTPUT:**

****

**6. Identify the daily total revenue and number of transactions.**

**SQL QUERY:**

SELECT

transaction\_date, COUNT(transaction\_id) transactions,

SUM(transaction\_qty\*unit\_price) daily\_revenue

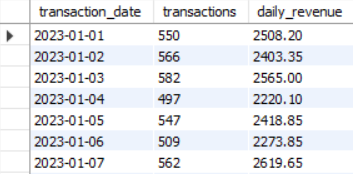
FROM

coffee\_shop\_sales

GROUP BY

transaction\_date;

**OUTPUT:**

****

**7. Find the hour with the highest sales across all transactions.**

**SQL QUERY:**

SELECT

EXTRACT(HOUR FROM transaction\_time) transaction\_hour,

SUM(transaction\_qty\*unit\_price) Hourly\_revenue FROM coffee\_shop\_sales

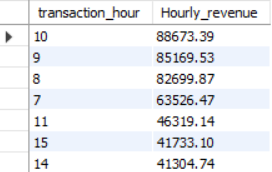
GROUP BY

EXTRACT(HOUR FROM transaction\_time)

ORDER BY

Hourly\_revenue DESC;

**OUTPUT:**

****

**8. List the top 5 best-selling products by revenue.**

**SQL QUERY:**

SELECT

product\_id, product\_type, SUM(transaction\_qty\*unit\_price) total\_revenue

FROM

coffee\_shop\_sales

GROUP BY

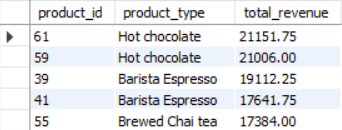
product\_id, product\_type

ORDER BY

total\_revenue DESC

LIMIT 5;

**OUTPUT:**

****

**9. List the top 5 best-selling products by quantity.**

**SQL QUERY:**

SELECT

product\_id, product\_type, SUM(transaction\_qty) total\_qty

FROM

coffee\_shop\_sales

GROUP BY

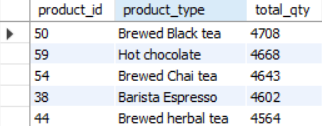
product\_id, product\_type

ORDER BY

total\_qty DESC

LIMIT 5;

**OUTPUT:**

****

**10. Calculate the total revenue generated by each product category.**

**SQL QUERY:**

SELECT

product\_category, SUM(transaction\_qty\*unit\_price) total\_revenue

FROM

coffee\_shop\_sales

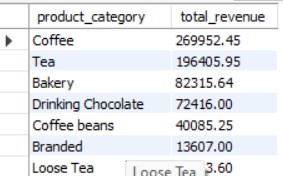
GROUP BY

product\_category

ORDER BY

total\_revenue DESC;

**OUTPUT:**

****

**11. Compare the total revenue and number of transactions for each store.**

**SQL QUERY:**

SELECT

store\_id, SUM(transaction\_qty\*unit\_price) total\_revenue,

SUM(transaction\_qty) total\_qty

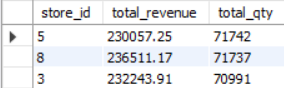
FROM

coffee\_shop\_sales

GROUP BY

store\_id;

**OUTPUT:**

****

**12. Calculate total sales for each respective month.**

**SQL QUERY:**

SELECT

EXTRACT(MONTH FROM transaction\_date) month\_name,

SUM(transaction\_qty\*unit\_price) total\_sales

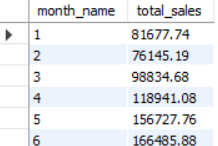
FROM

coffee\_shop\_sales

GROUP BY

EXTRACT(MONTH FROM transaction\_date);

**OUTPUT:**

****

**13. Determine month-on-month decrease or increase in sales.**

**SQL QUERY:**

SELECT

MONTH(transaction\_date) month\_no, SUM(transaction\_qty\*unit\_price)

total\_sales, ROUND((SUM(transaction\_qty\*unit\_price)-

LAG(SUM(transaction\_qty\*unit\_price)) OVER(ORDER BY

MONTH(transaction\_date) ASC))/LAG(SUM(transaction\_qty\*unit\_price))

OVER(ORDER BY MONTH(transaction\_date) ASC) \* 100,2) MOM\_Percentage

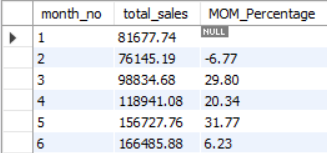
FROM

coffee\_shop\_sales

GROUP BY

MONTH(transaction\_date);

**OUTPUT:**

****

**14. Determine month on month decrease or increase in total quantity.**

**SQL QUERY:**

SELECT MONTH(transaction\_date) month\_no, SUM(transaction\_qty) total\_qty,

ROUND((SUM(transaction\_qty)-LAG(SUM(transaction\_qty)) OVER(ORDER BY

MONTH(transaction\_date) ASC))/LAG(SUM(transaction\_qty)) OVER(ORDER BY

MONTH(transaction\_date) ASC) \* 100,2) MOM\_Percentage

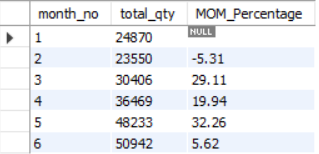
FROM

coffee\_shop\_sales

GROUP BY

MONTH(transaction\_date);

**OUTPUT:**

****

**15. Daily sales for a selected month.**

**SQL QUERY:**

SELECT

DAY(transaction\_date) day\_no, SUM(transaction\_qty\*unit\_price) total\_sales

FROM

coffee\_shop\_sales

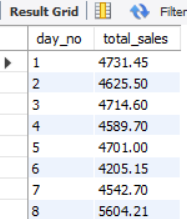
WHERE

MONTH(transaction\_date) =5

GROUP BY

DAY(transaction\_date);

**OUTPUT:**

****

**16. Comparing daily sales with average sales. If greater than print "Above Average" less than "Below Average".**

**SQL QUERY:**

SELECT day\_no,totalsales, avgsales,

CASE WHEN totalsales>avgsales THEN "Above Average"

WHEN totalsales<avgsales THEN "Below Average"

END sales\_status

FROM

(SELECT DAY(transaction\_date) day\_no,SUM(transaction\_qty\*unit\_price) totalsales, AVG(SUM(transaction\_qty\*unit\_price)) OVER() avgsales

FROM

coffee\_shop\_sales

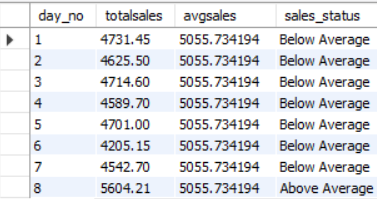
WHERE

MONTH(transaction\_date) =5

GROUP BY

DAY(transaction\_date)) sales\_data;

**OUTPUT:**

****

**17. Total sales on weekdays and weekends.**

**SQL QUERY:**

SELECT

CASE

WHEN DAYOFWEEK(transaction\_date) IN (1,7) THEN "Weekend"

ELSE "Weekday"

END day\_type

, SUM(transaction\_qty\*unit\_price) sales

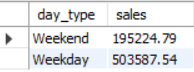
FROM

coffee\_shop\_sales

GROUP BY

day\_type;

**OUTPUT:**

****

**18. Sales from each store location on a particular month.**

**SQL QUERY:**

SELECT

store\_location, SUM(transaction\_qty\*unit\_price) totalsales

FROM

coffee\_shop\_sales

WHERE

MONTH(transaction\_date) =5

GROUP BY

store\_location**;**

**OUTPUT:**

****

**19. To view sales in a week in a particular month.**

**SQL QUERY:**

SELECT

CASE

WHEN DAYOFWEEK(transaction\_date) = 2 THEN 'Monday'

WHEN DAYOFWEEK(transaction\_date) = 3 THEN 'Tuesday'

WHEN DAYOFWEEK(transaction\_date) = 4 THEN 'Wednesday'

WHEN DAYOFWEEK(transaction\_date) = 5 THEN 'Thursday'

WHEN DAYOFWEEK(transaction\_date) = 6 THEN 'Friday'

WHEN DAYOFWEEK(transaction\_date) = 7 THEN 'Saturday'

ELSE 'Sunday'

END AS Day\_of\_Week,

ROUND(SUM(unit\_price \* transaction\_qty)) AS Total\_Sales

FROM

coffee\_shop\_sales

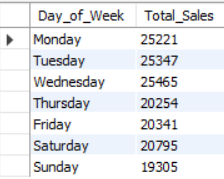
WHERE

MONTH(transaction\_date) = 5

GROUP BY

Day\_of\_Week;

**OUTPUT:**

****

**20. calculates total sales by product and then selects only those products where sales exceed the average sales.**

**SQL QUERY:**

WITH Product\_sales AS(

SELECT

product\_id, product\_type, SUM(unit\_price \* transaction\_qty) AS Total\_Sales

FROM

coffee\_shop\_sales

GROUP BY

product\_id, product\_type

)

SELECT

product\_id, product\_type, Total\_Sales

FROM

Product\_sales

WHERE

Total\_Sales>(SELECT AVG(Total\_Sales) AS Avg\_Sales FROM Product\_sales);

**OUTPUT:**

****