MY SQL QUERIES

COFFEE SHOP SALES PROJECT

ALTER DATE (transaction_date) COLUMN TO DATE DATA TYPE

```
ALTER TABLE coffee_shop_sales  Alter column transaction_date  type  date

Using transaction_date :: date
```

ALTER TIME (transaction_time) COLUMN TO DATE DATA TYPE

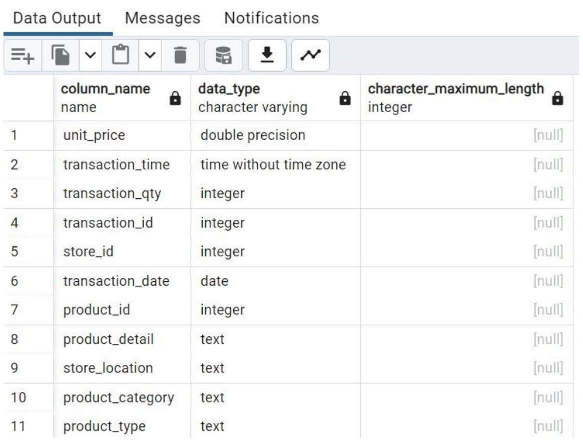
```
ALTER TABLE coffee_shop_sales  Alter column transaction_time type time

Using transaction_time::time
```

ALTER PRICE (unit_price) COLUMN TO DATE DATA TYPE

```
Alter Table coffee_shop_sales alter column unit_price type numeric
```

DATA TYPES OF DIFFERENT COLUMNS

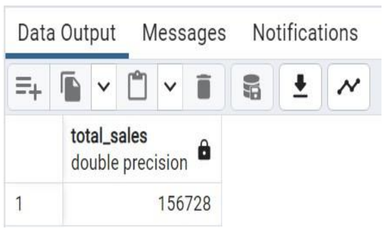
```
SELECT column_name, data_type, character_maximum_length

FROM information_schema.columns

WHERE table_name = 'coffee_shop_sales';
```



	column_name name	data_type character varying	character_maximum_length integer
1	unit_price	double precision	[null]
2	transaction_time	time without time zone	[null]
3	transaction_qty	integer	[null]
4	transaction_id	integer	[null]
5	store_id	integer	[null]
6	transaction_date	date	[null]
7	product_id	integer	[null]
8	product_detail	text	[null]
9	store_location	text	[null]
10	product_category	text	[null]
11	product_type	text	[null]

TOTAL SALES

```
SELECT CONCAT(ROUND(SUM(unit_price * transaction_qty)/1000,2),'k') as Total_Sales

FROM coffee_shop_sales

WHERE EXTRACT(MONTH FROM transaction_date) = 5 -- for month of (CM-May)
```

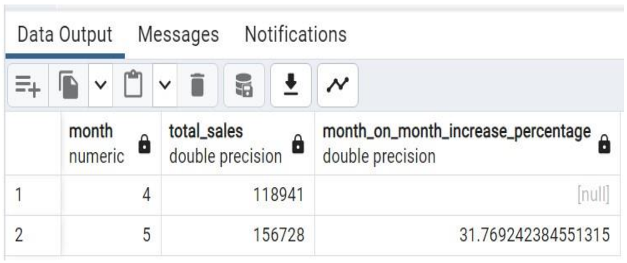


Data Output | Messages | Notifications

	total_sales double precision
1	156728

TOTAL SALES KPI - MOM DIFFERENCE AND MOM GROWTH

```
SELECT

   Extract(MONTH from transaction_date) AS month,

   ROUND(SUM(unit_price * transaction_qty)) AS total_sales,

   (SUM(unit_price * transaction_qty) - LAG(SUM(unit_price * transaction_qty), 1)

   OVER (ORDER BY Extract(MONTH from transaction_date))) / LAG(SUM(unit_price * transaction_qty), 1)

   OVER (ORDER BY Extract(MONTH from transaction_date)) * 100 AS
month_on_month_increase_percentage

FROM

   coffee_shop_sales

WHERE

   Extract(MONTH from transaction_date) IN (4, 5) -- for months of April and May

GROUP BY

   Extract(MONTH from transaction_date)

ORDER BY

   Extract(MONTH from transaction_date);
```



Data Output | Messages | Notifications

	month numeric	total_sales double precision	month_on_month_increase_percentage double precision
1	4	118941	[null]
2	5	156728	31.769242384551315

Explaination

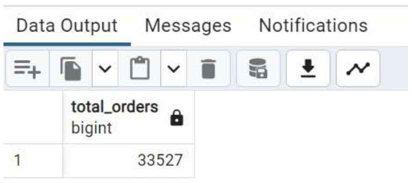
SELECT clause:

- Extract(MONTH from transaction_date): Extracts the month from the transaction_date column and renames it as month.
- ROUND(SUM(unit_price * transaction_qty)) AS total_sales: Calculates the total sales by multiplying unit_price and transaction_qty, then sums the result for each month. The ROUND function rounds the result to the nearest integer.
- (SUM(unit_price * transaction_qty) - LAG(SUM(unit_price * transaction_qty), 1) OVER (ORDER BY Extract(MONTH from transaction_date))) / LAG(SUM(unit_price * transaction_qty), 1) OVER (ORDER BY MONTH(transaction_date)) * 100 AS mom_increase_percentage with the functions used:
    - SUM(unit_price * transaction_qty): This calculates the total sales for the current month. It multiplies the unit_price by the transaction_qty for each transaction and then sums up these values.
    - LAG(SUM(unit_price * transaction_qty), 1) OVER (ORDER BY Extract(MONTH from transaction_date)): This function retrieves the value of the total sales for the previous month. It uses the LAG window function to get the value of the SUM(unit_price * transaction_qty) from the previous row (previous month) ordered by the transaction_date.
    - (SUM(unit_price * transaction_qty) - LAG(SUM(unit_price * transaction_qty), 1) OVER (ORDER BY Extract(MONTH from transaction_date))): This part calculates the difference between the total sales of the current month and the total sales of the previous month.
    - LAG(SUM(unit_price * transaction_qty), 1) OVER (ORDER BY Extract(MONTH from transaction_date)): This function retrieves the value of the total sales for the previous month again. It's used in the denominator to calculate the percentage increase.
    - (SUM(unit_price * transaction_qty) - LAG(SUM(unit_price * transaction_qty), 1) OVER (ORDER BY Extract(MONTH from transaction_date))) / LAG(SUM(unit_price * transaction_qty), 1) OVER (ORDER BY Extract(MONTH from transaction_date)): This calculates the ratio of the difference in sales between the current and previous months to the total sales of the previous month. It represents the percentage increase or decrease in sales compared to the previous month.
    - 100: This part multiplies the ratio by 100 to convert it to a percentage.

- FROM clause:
  coffee_shop_sales: Specifies the table from which data is being selected.
- WHERE clause:
  Extract(MONTH from transaction_date) IN (4, 5): Filters the data to include only transactions from April and May.
- GROUP BY clause:
  Extract(MONTH from transaction_date): Groups the results by month.
- ORDER BY clause:
  Extract(MONTH from transaction_date)): Orders the results by month.

TOTAL ORDERS

```
SELECT COUNT(transaction_id) as Total_Orders

FROM coffee_shop_sales

WHERE Extract(MONTH from transaction_date)= 5 -- for month of (CM-May)
```

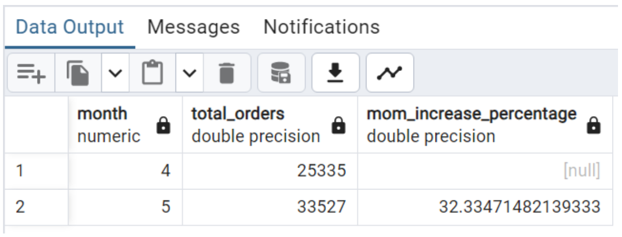


Data Output Messages Notifications

	total_orders bigint
1	33527

TOTAL ORDERS KPI - MOM DIFFERENCE AND MOM GROWTH

```
SELECT

  Extract(MONTH from transaction_date) AS month,

   ROUND(COUNT(transaction_id)) AS total_orders,

   ROUND(COUNT(transaction_id) - LAG(COUNT(transaction_id), 1)

   OVER (ORDER BY Extract(MONTH from transaction_date))) / LAG(COUNT(transaction_id), 1)

   OVER (ORDER BY Extract(MONTH from transaction_date)) * 100  AS mom_increase_percentage

FROM

   coffee_shop_sales

WHERE

  Extract(MONTH from transaction_date) IN (4, 5) -- for April and May

GROUP BY

     Extract(MONTH from transaction_date)

ORDER BY

     Extract(MONTH from transaction_date);
```

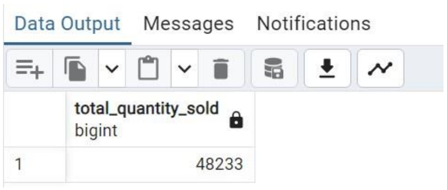


Data Output Messages Notifications

	month numeric	total_orders double precision	mom_increase_percentage double precision
1	4	25335	[null]
2	5	33527	32.33471482139333

TOTAL QUANTITY SOLD

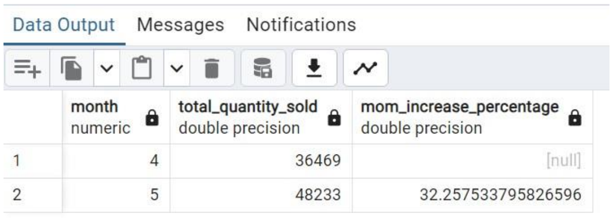
```
SELECT SUM(transaction_qty) as Total_Quantity_Sold

FROM coffee_shop_sales

WHERE EXTRACT(MONTH FROM transaction_date) = 5 -- for month of (CM-May)
```



	total_quantity_sold bigint
1	48233

TOTAL QUANTITY SOLD KPI - MOM DIFFERENCE AND MOM GROWTH

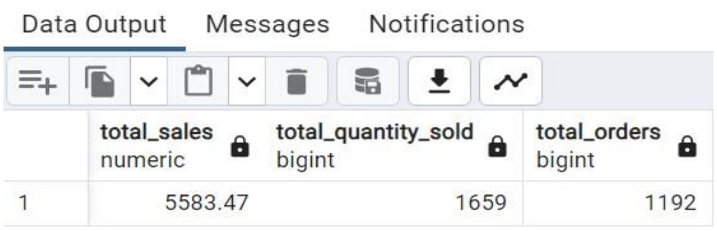
```
SELECT

    Extract(MONTH from transaction_date) AS month,

    ROUND(SUM(transaction_qty)) AS total_quantity_sold,

   ROUND(SUM(transaction_qty) - LAG(SUM(transaction_qty), 1)

    OVER (ORDER BY Extract(MONTH from transaction_date))) / LAG(SUM(transaction_qty), 1)

    OVER (ORDER BY Extract(MONTH from transaction_date)) * 100  AS mom_increase_percentage

FROM

    coffee_shop_sales

WHERE

    Extract(MONTH from transaction_date) IN (4, 5)   -- for April and May

GROUP BY

Extract(MONTH from transaction_date)

ORDER BY

    Extract(MONTH from transaction_date);
```



	month numeric	total_quantity_sold double precision	mom_increase_percentage double precision
1	4	36469	[null]
2	5	48233	32.257533795826596

CALENDAR TABLE – DAILY SALES, QUANTITY and TOTAL ORDERS

```
SELECT
    SUM(unit_price * transaction_qty) AS total_sales,
    SUM(transaction_qty) AS total_quantity_sold,
    COUNT(transaction_id) AS total_orders
FROM
    coffee_shop_sales
WHERE
    transaction_date = '2023-05-18'; --For 18 May 2023
```



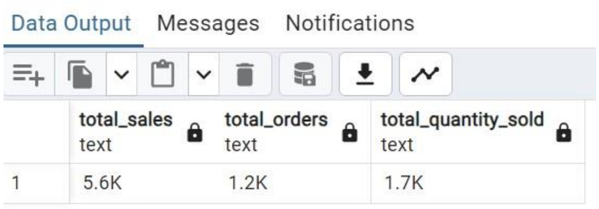
	total_sales numeric	total_quantity_sold bigint	total_orders bigint
1	5583.47	1659	1192

If you want to get exact Rounded off values then use below query to get the result:

```
SELECT
    CONCAT(ROUND(SUM(unit_price * transaction_qty) / 1000.0, 1), 'K') AS total_sales,
    CONCAT(ROUND(COUNT(transaction_id) / 1000.0, 1), 'K') AS total_orders,
    CONCAT(ROUND(SUM(transaction_qty) / 1000.0, 1), 'K') AS total_quantity_sold
FROM
    coffee_shop_sales
WHERE
    transaction_date = '2023-05-18';
```

or

```
SELECT
    CONCAT(ROUND(SUM(unit_price * transaction_qty) / 1000::numeric, 1), 'K') AS total_sales,
    CONCAT(ROUND(COUNT(transaction_id) / 1000::numeric, 1), 'K') AS total_orders,
    CONCAT(ROUND(SUM(transaction_qty) / 1000::numeric, 1), 'K') AS total_quantity_sold
FROM
    coffee_shop_sales
WHERE
    transaction_date = '2023-05-18';
```



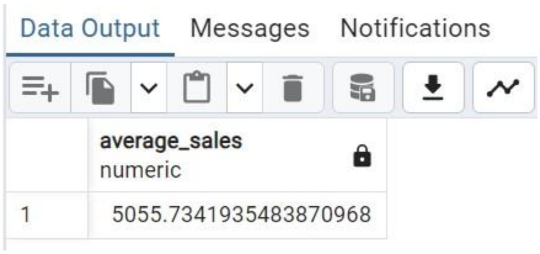
	total_sales text	total_orders text	total_quantity_sold text
1	5.6K	1.2K	1.7K

SALES TREND OVER PERIOD

```
SELECT AVG(total_sales) AS average_sales
FROM (
  SELECT
    SUM(unit_price * transaction_qty) AS total_sales
  FROM
    coffee_shop_sales
  WHERE
    EXTRACT(month FROM transaction_date) = 5  -- Filter for May
  GROUP BY
    transaction_date
) AS internal_query;
```

Query Explanation:

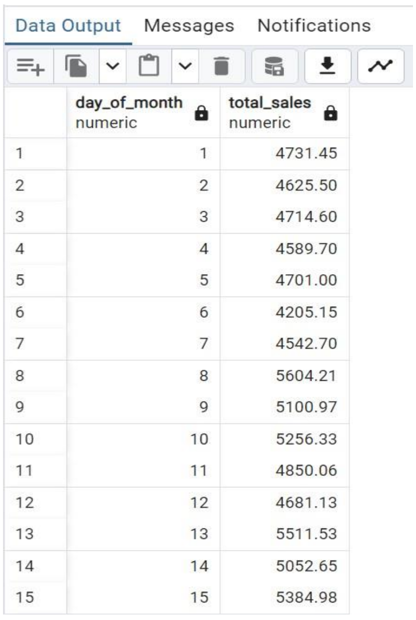
- This inner subquery calculates the total sales (unit_price * transaction_qty) for each date in May. It filters the data to include only transactions that occurred in May by using the MONTH() function to extract the month from the transaction_date column and filtering for May (month number 5).
- The GROUP BY clause groups the data by transaction_date, ensuring that the total sales are aggregated for each individual date in May.
- The outer query calculates the average of the total sales over all dates in May. It references the result of the inner subquery as a derived table named internal_query.
- The AVG() function calculates the average of the total_sales column from the derived table, giving us the average sales for May.



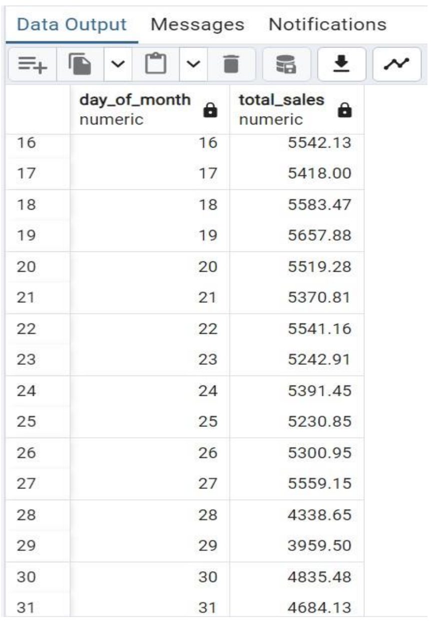
	average_sales numeric
1	5055.7341935483870968

DAILY SALES FOR MONTH SELECTED

```
select

extract (day from transaction_date) as day_of_month,

sum(unit_price*transaction_qty) as total_sales

from coffee_shop_sales

where extract (month from transaction_date)=5

group by extract (day from transaction_date)

order by extract (day from transaction_date)
```



	day_of_month numeric	total_sales numeric
1	1	4731.45
2	2	4625.50
3	3	4714.60
4	4	4589.70
5	5	4701.00
6	6	4205.15
7	7	4542.70
8	8	5604.21
9	9	5100.97
10	10	5256.33
11	11	4850.06
12	12	4681.13
13	13	5511.53
14	14	5052.65
15	15	5384.98

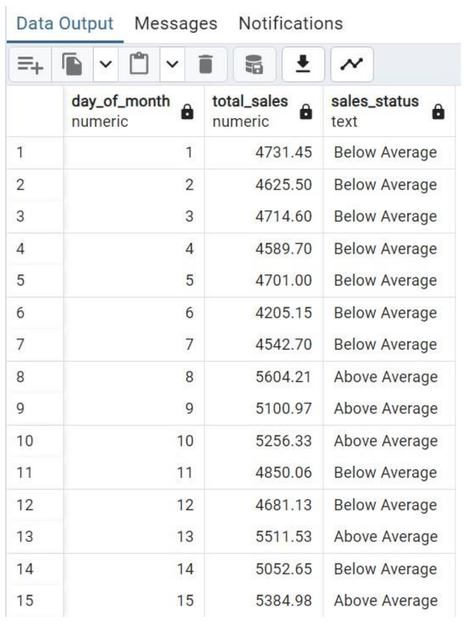


	day_of_month numeric	total_sales numeric
16	16	5542.13
17	17	5418.00
18	18	5583.47
19	19	5657.88
20	20	5519.28
21	21	5370.81
22	22	5541.16
23	23	5242.91
24	24	5391.45
25	25	5230.85
26	26	5300.95
27	27	5559.15
28	28	4338.65
29	29	3959.50
30	30	4835.48
31	31	4684.13

COMPARING DAILY SALES WITH AVERAGE SALES – IF GREATER THAN "ABOVE AVERAGE" and LESSER THAN "BELOW AVERAGE"

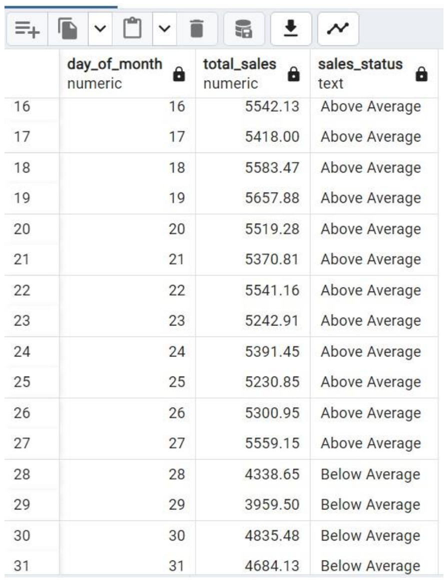
```
SELECT day_of_month,total_sales,

   CASE

      WHEN total_sales > avg_sales THEN 'Above Average'

      WHEN total_sales < avg_sales THEN 'Below Average'

      ELSE 'Average'

END AS sales_status

FROM ( SELECT

      extract(day from transaction_date) AS day_of_month,

      SUM(unit_price * transaction_qty) AS total_sales,

      AVG(SUM(unit_price * transaction_qty)) OVER () AS avg_sales

FROM
```

```
    coffee_shop_sales
  WHERE extract(MONTH from transaction_date) = 5  -- Filter for May
  GROUP BY extract(day from transaction_date)
  ORDER BY day_of_month) as internal_query
```



Data Output Messages Notifications

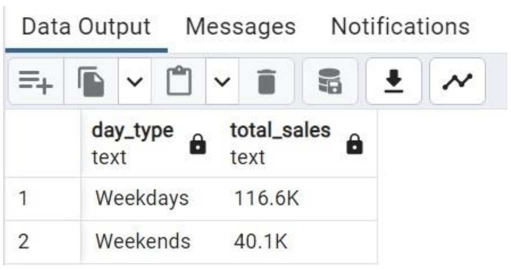
	day_of_month numeric	total_sales numeric	sales_status text
1	1	4731.45	Below Average
2	2	4625.50	Below Average
3	3	4714.60	Below Average
4	4	4589.70	Below Average
5	5	4701.00	Below Average
6	6	4205.15	Below Average
7	7	4542.70	Below Average
8	8	5604.21	Above Average
9	9	5100.97	Above Average
10	10	5256.33	Above Average
11	11	4850.06	Below Average
12	12	4681.13	Below Average
13	13	5511.53	Above Average
14	14	5052.65	Below Average
15	15	5384.98	Above Average



	day_of_month numeric	total_sales numeric	sales_status text
16	16	5542.13	Above Average
17	17	5418.00	Above Average
18	18	5583.47	Above Average
19	19	5657.88	Above Average
20	20	5519.28	Above Average
21	21	5370.81	Above Average
22	22	5541.16	Above Average
23	23	5242.91	Above Average
24	24	5391.45	Above Average
25	25	5230.85	Above Average
26	26	5300.95	Above Average
27	27	5559.15	Above Average
28	28	4338.65	Below Average
29	29	3959.50	Below Average
30	30	4835.48	Below Average
31	31	4684.13	Below Average

SALES BY WEEKDAY / WEEKEND:

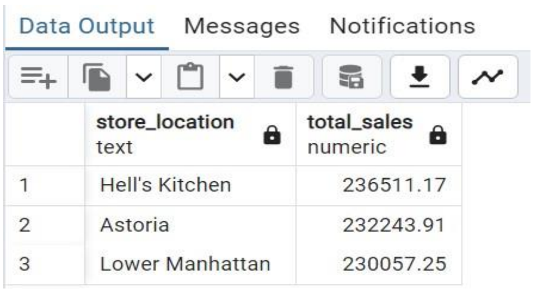
```
SELECT
  CASE
    WHEN extract(dow from transaction_date) IN (0, 6) THEN 'Weekends'
    ELSE 'Weekdays'
  END AS day_type,
  CONCAT(ROUND(SUM(unit_price * transaction_qty)/1000.0,1),'K') AS total_sales
FROM
  coffee_shop_sales
WHERE
  extract(month from transaction_date) = 5
GROUP BY
  CASE
    WHEN extract(dow from transaction_date) IN (0, 6) THEN 'Weekends'
 ELSE 'Weekdays'
 END;
```



	day_type text	total_sales text
1	Weekdays	116.6K
2	Weekends	40.1K

SALES BY STORE LOCATION

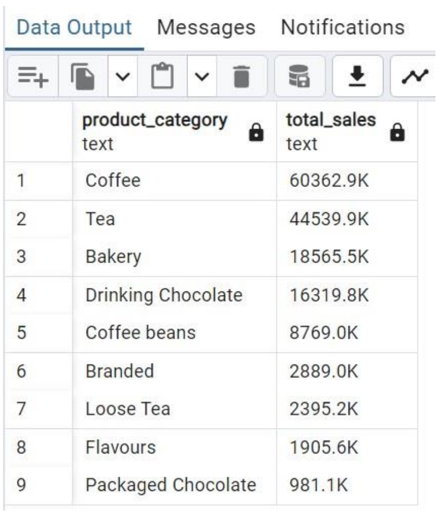
```
SELECT
        store_location,
        SUM(unit_price * transaction_qty) as Total_Sales
FROM coffee_shop_sales
WHERE
        Extract(MONTH from transaction_date) =5
GROUP BY store_location
ORDER BY Total_Sales DESC
```



	store_location text	total_sales numeric
1	Hell's Kitchen	236511.17
2	Astoria	232243.91
3	Lower Manhattan	230057.25

SALES BY PRODUCT CATEGORY

```
SELECT
        product_category,
        concat(ROUND(SUM(unit_price * transaction_qty),1),'K') as Total_Sales
FROM coffee_shop_sales
WHERE
Extract(month  from transaction_date) = 5
GROUP BY product_category
ORDER by ROUND(SUM(unit_price * transaction_qty),1) desc
```

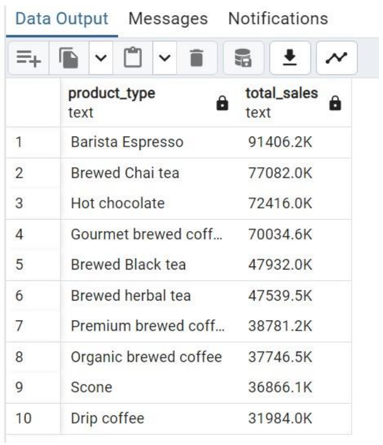


Data Output Messages Notifications

	product_category text	total_sales text
1	Coffee	60362.9K
2	Tea	44539.9K
3	Bakery	18565.5K
4	Drinking Chocolate	16319.8K
5	Coffee beans	8769.0K
6	Branded	2889.0K
7	Loose Tea	2395.2K
8	Flavours	1905.6K
9	Packaged Chocolate	981.1K

SALES BY PRODUCTS (TOP 10)

```
SELECT
        product_type,
        concat(ROUND(SUM(unit_price * transaction_qty),1),'K') as Total_Sales
FROM coffee_shop_sales
GROUP BY product_type
ORDER by ROUND(SUM(unit_price * transaction_qty),1) desc
Limit 10;
```

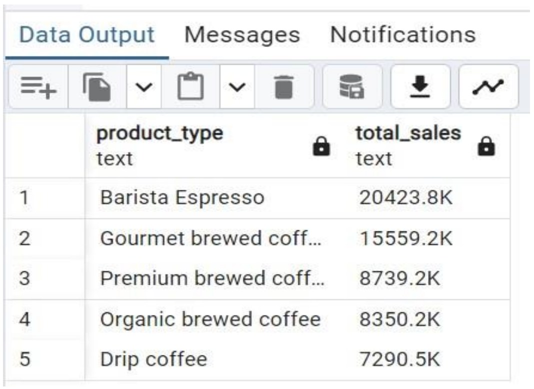


Data Output Messages Notifications

	product_type text	total_sales text
1	Barista Espresso	91406.2K
2	Brewed Chai tea	77082.0K
3	Hot chocolate	72416.0K
4	Gourmet brewed coff...	70034.6K
5	Brewed Black tea	47932.0K
6	Brewed herbal tea	47539.5K
7	Premium brewed coff...	38781.2K
8	Organic brewed coffee	37746.5K
9	Scone	36866.1K
10	Drip coffee	31984.0K

Sales by Product type on category

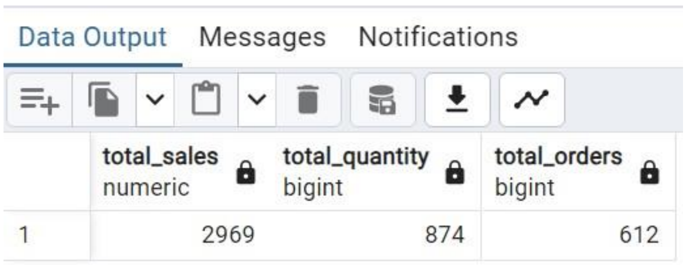
```
SELECT
    product_type,
    CONCAT(ROUND(SUM(unit_price * transaction_qty), 1), 'K') AS Total_Sales
FROM
    coffee_shop_sales
WHERE
    EXTRACT(month FROM transaction_date) = 5
    AND lower(product_category) = 'coffee'  -- Ensure case-insensitive match
GROUP BY
    product_type
ORDER BY
    round(SUM(unit_price * transaction_qty),1) DESC
```



	product_type text	total_sales text
1	Barista Espresso	20423.8K
2	Gourmet brewed coff...	15559.2K
3	Premium brewed coff...	8739.2K
4	Organic brewed coffee	8350.2K
5	Drip coffee	7290.5K

SALES BY DAY | HOUR

```
SELECT
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales,
    SUM(transaction_qty) AS Total_Quantity,
    COUNT(*) AS Total_Orders
FROM
    coffee_shop_sales
WHERE
     extract(month from transaction_date)= 5 -- Filter for May (month number 5)
and extract(DOW from transaction_date)=2 -- Filter for Tuesday (0 is Sunday, 1 is Monday, ..., 6 is Saturday)
and extract(hour from transaction_time)=8;  -- Filter for hour number 8
```

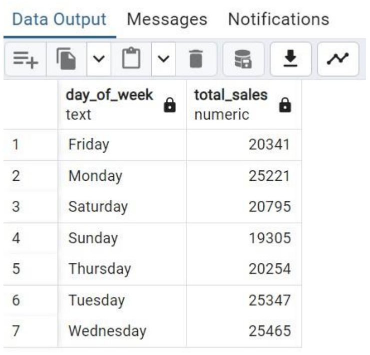


	total_sales numeric	total_quantity bigint	total_orders bigint
1	2969	874	612

TO GET SALES FROM MONDAY TO SUNDAY FOR MONTH OF MAY

```
SELECT
    CASE
        WHEN extract(Dow from transaction_date) = 1 THEN 'Monday'
        WHEN extract(Dow from transaction_date) = 2 THEN 'Tuesday'
        WHEN extract(Dow from transaction_date) = 3 THEN 'Wednesday'
        WHEN extract(Dow from transaction_date) = 4 THEN 'Thursday'
        WHEN extract(Dow from transaction_date) = 5 THEN 'Friday'
        WHEN extract(Dow from transaction_date) = 6 THEN 'Saturday'
        ELSE 'Sunday'
    END AS Day_of_Week,
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM
    coffee_shop_sales
WHERE
```

```
    extract(month from transaction_date) = 5 -- Filter for May (month number 5)
GROUP BY
    CASE
        WHEN extract(Dow from transaction_date) = 1 THEN 'Monday'
        WHEN extract(Dow from transaction_date) = 2 THEN 'Tuesday'
        WHEN extract(Dow from transaction_date) = 3 THEN 'Wednesday'
        WHEN extract(Dow from transaction_date) = 4 THEN 'Thursday'
        WHEN extract(Dow from transaction_date) = 5 THEN 'Friday'
        WHEN extract(Dow from transaction_date) = 6 THEN 'Saturday'
        ELSE 'Sunday'
End;
```



	day_of_week text	total_sales numeric
1	Friday	20341
2	Monday	25221
3	Saturday	20795
4	Sunday	19305
5	Thursday	20254
6	Tuesday	25347
7	Wednesday	25465

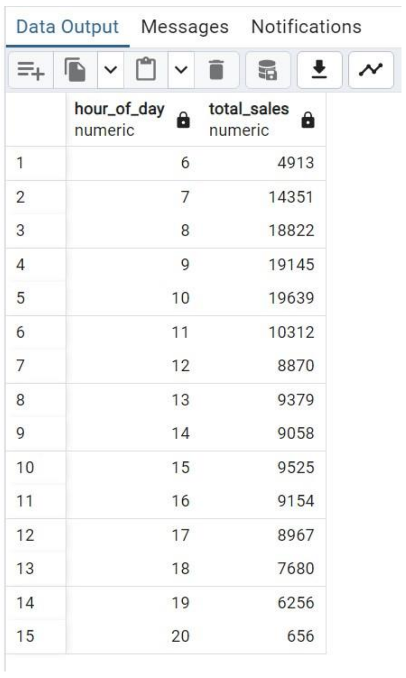
TO GET SALES FOR ALL HOURS FOR MONTH OF MAY

```
SELECT
    Extract (hour from transaction_time) AS Hour_of_Day,
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM
    coffee_shop_sales
WHERE
    Extract(MONTH from transaction_date) = 5 -- Filter for May (month number 5)
GROUP BY
    Extract (hour from transaction_time)
ORDER BY
    Extract (hour from transaction_time);
```

OR

```
FROM coffee_shop_sales
WHERE
```

```
    Extract(MONTH from transaction_date) = 5 -- Filter for May (month number 5)
GROUP BY 1
 ORDER by 1;
```



Data Output | Messages | Notifications

	hour_of_day numeric	total_sales numeric
1	6	4913
2	7	14351
3	8	18822
4	9	19145
5	10	19639
6	11	10312
7	12	8870
8	13	9379
9	14	9058
10	15	9525
11	16	9154
12	17	8967
13	18	7680
14	19	6256
15	20	656

Here group by “1” =   Extract(MONTH from transaction_date)

Group by “2” = Total_Sales and so on ….