Target-Sql Business Case

Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

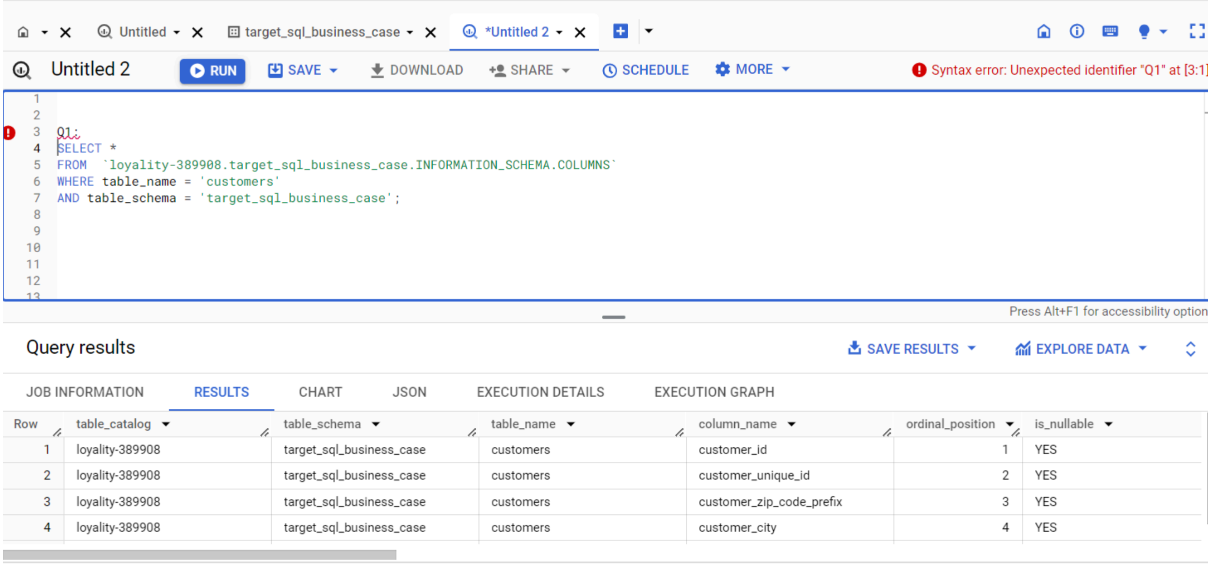
1 . Data type of all columns in the "customers" table.

Ans :

- Query :

```
        SELECT *
        FROM
`loyality-389908.target_sql_business_case.INFORMATION_SCHEMA.COLUMNS`
        WHERE table_name = 'customers'
        AND table_schema = 'target_sql_business_case';
```

- Result-ScreenShot :



Syntax error: Unexpected identifier "Q1" at [3:1]

```


Q1;
SELECT *
FROM  `loyality-389908.target_sql_business_case.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers'
AND table_schema = 'target_sql_business_case';
```

Row	table_catalog	table_schema	table_name	column_name	ordinal_position	is_nullable
1	loyality-389908	target_sql_business_case	customers	customer_id	1	YES
2	loyality-389908	target_sql_business_case	customers	customer_unique_id	2	YES
3	loyality-389908	target_sql_business_case	customers	customer_zip_code_prefix	3	YES
4	loyality-389908	target_sql_business_case	customers	customer_city	4	YES

2 . Get the time range between which the orders were placed.

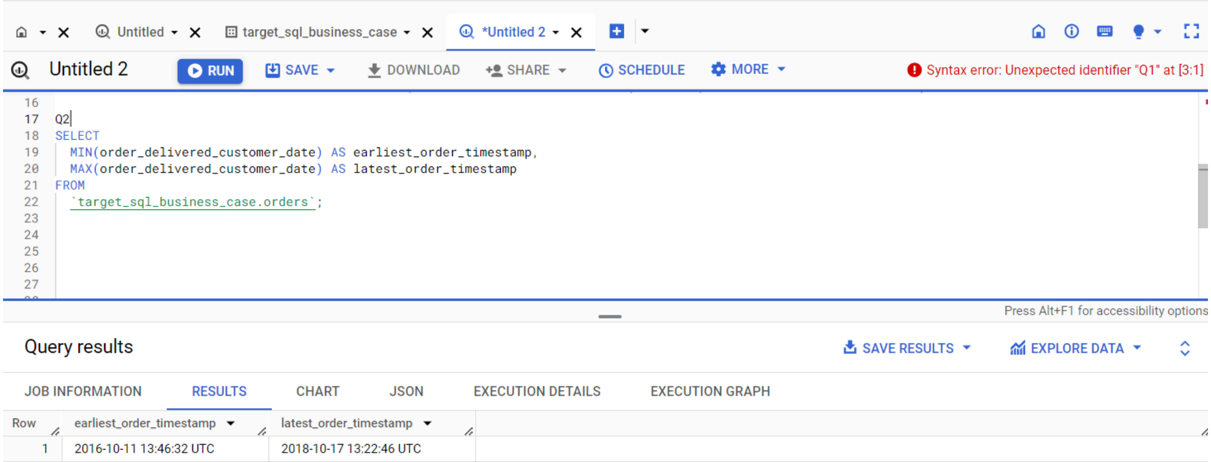
Ans :

- Query :

```
              SELECT
                MIN(order_delivered_customer_date) AS
             earliest_order_timestamp,
                MAX(order_delivered_customer_date) AS latest_order_timestamp
```

```
        FROM
          `target_sql_business_case.orders`;
```

- Result-ScreenShot :



Syntax error: Unexpected identifier "Q1" at [3:1]

```

Q2
SELECT
  MIN(order_delivered_customer_date) AS earliest_order_timestamp,
  MAX(order_delivered_customer_date) AS latest_order_timestamp
FROM
  `target_sql_business_case.orders`;
```

Row	earliest_order_timestamp	latest_order_timestamp
1	2016-10-11 13:46:32 UTC	2018-10-17 13:22:46 UTC

Insight:

Orders were placed over a period spanning from October 11, 2016, to October 17, 2018. The dataset covers approximately two years of order data.

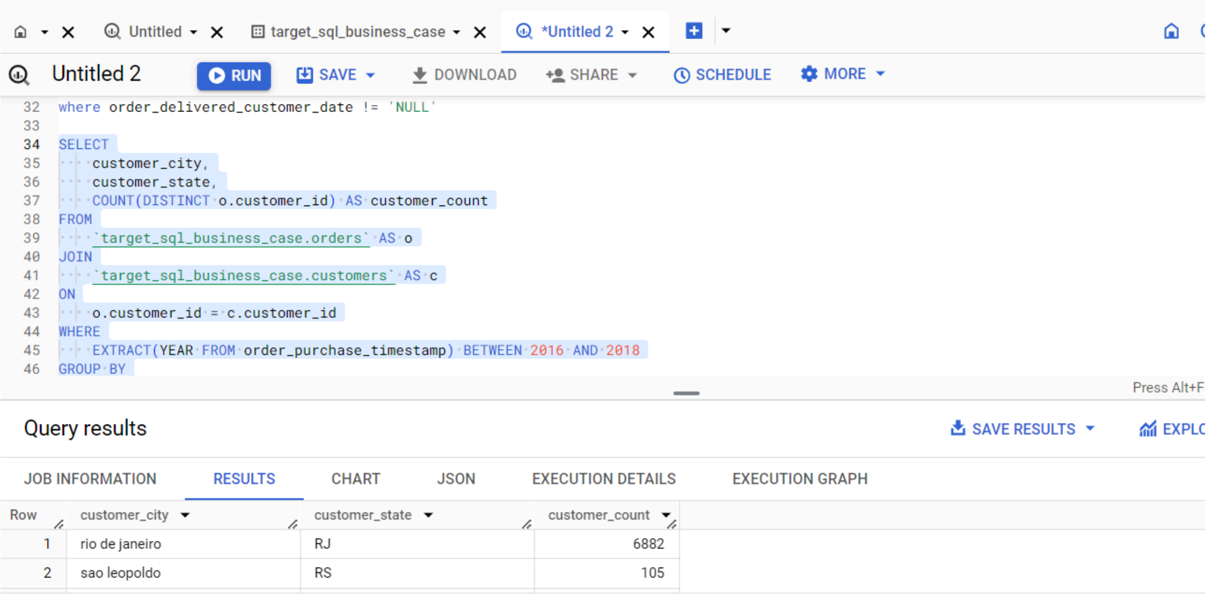
3. Count the Cities & States of customers who ordered during the given period.

Ans :

- Query :

```
  SELECT
      customer_city,
      customer_state,
      COUNT(DISTINCT o.customer_id) AS customer_count
FROM
    `target_sql_business_case.orders` AS o
JOIN
    `target_sql_business_case.customers` AS c
ON
    o.customer_id = c.customer_id
WHERE
    EXTRACT(YEAR FROM order_purchase_timestamp) BETWEEN 2016 AND 2018
GROUP BY
    customer_city,
        Customer_state;
```

- Result-ScreenShot :



```
where order_delivered_customer_date != 'NULL'

SELECT
    customer_city,
    customer_state,
    COUNT(DISTINCT o.customer_id) AS customer_count
FROM
    `target_sql_business_case.orders` AS o
JOIN
    `target_sql_business_case.customers` AS c
ON
    o.customer_id = c.customer_id
WHERE
    EXTRACT(YEAR FROM order_purchase_timestamp) BETWEEN 2016 AND 2018
GROUP BY
```

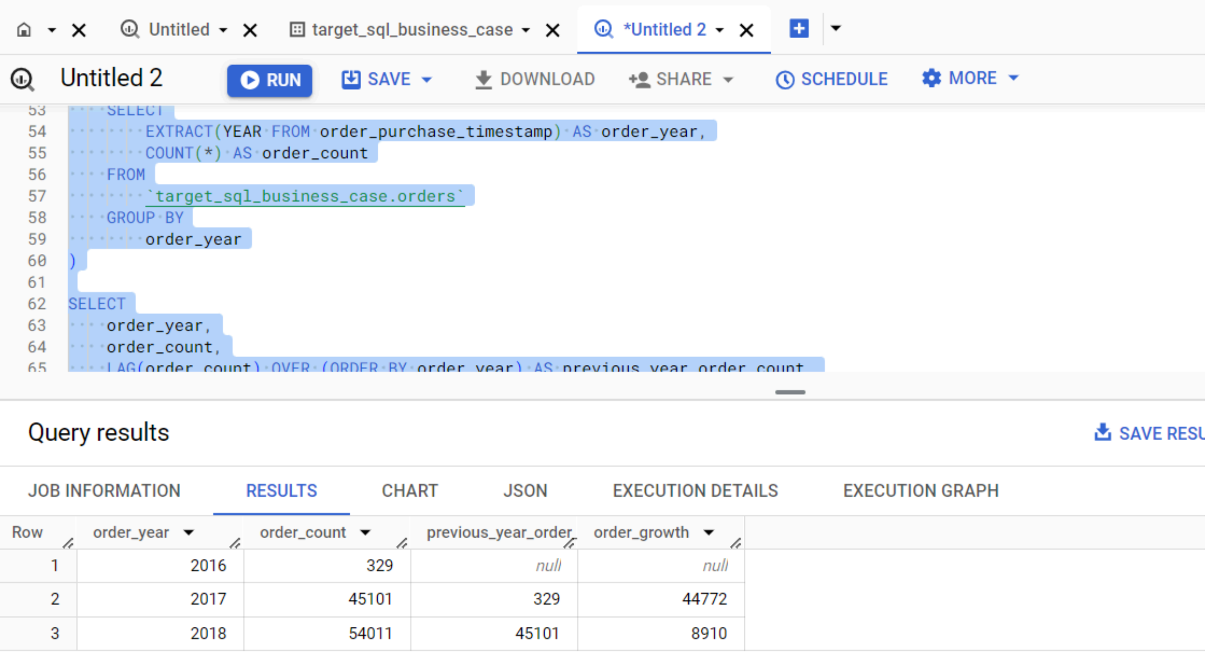
Row	customer_city	customer_state	customer_count
1	rio de janeiro	RJ	6882
2	sao leopoldo	RS	105

In-depth Exploration:

1. Is there a growing trend in the no. of orders placed over the past years?

Solution:

```
WITH OrderYears AS (
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        COUNT(*) AS order_count
    FROM
        `target_sql_business_case.orders`
    GROUP BY
        order_year
)

SELECT
    order_year,
    order_count,
    LAG(order_count) OVER (ORDER BY order_year) AS previous_year_order_count,
    (order_count - LAG(order_count) OVER (ORDER BY order_year)) AS order_growth
FROM
    OrderYears
ORDER BY
    order_year;
```



```
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        COUNT(*) AS order_count
    FROM
        `target_sql_business_case.orders`
    GROUP BY
        order_year
)

SELECT
    order_year,
    order_count,
    LAG(order_count) OVER (ORDER BY order_year) AS previous_year_order_count
```

Query results

Row	order_year	order_count	previous_year_order_	order_growth
1	2016	329	null	null
2	2017	45101	329	44772
3	2018	54011	45101	8910

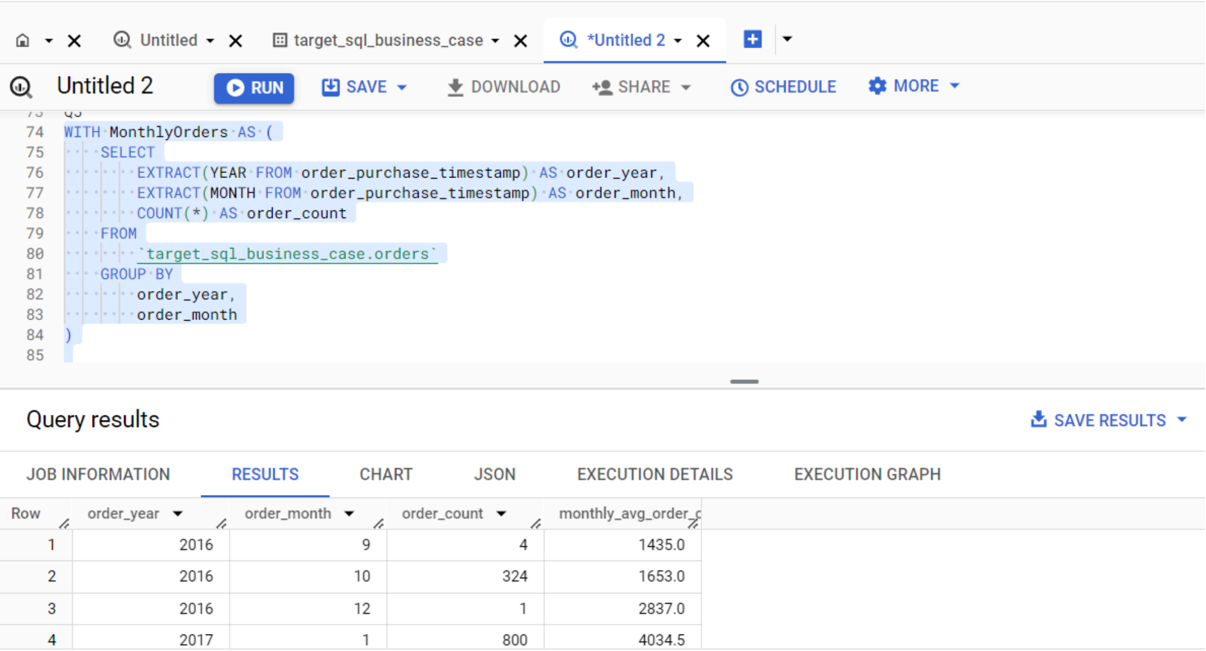
insight :

The analysis indicates a significant increase in orders from 2016 to 2017, with a substantial growth of 44,772 orders. While the growth continued into 2018, it slowed down, with an increase of 8,910 orders compared to the previous year. Understanding the factors driving growth and monitoring market dynamics will be crucial for sustaining and potentially accelerating growth in the future.

2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Ans:

```
 WITH MonthlyOrders AS (
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
        COUNT(*) AS order_count
    FROM
        `target_sql_business_case.orders`
    GROUP BY
        order_year,
        order_month
)

SELECT
    order_year,
    order_month,
    order_count,
    AVG(order_count) OVER (PARTITION BY order_month) AS monthly_avg_order_count
FROM
    MonthlyOrders
ORDER BY
    order_year,
    order_month;
```



```
Q5
WITH MonthlyOrders AS (
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
        COUNT(*) AS order_count
    FROM
        `target_sql_business_case.orders`
    GROUP BY
        order_year,
        order_month
)

```

Row	order_year	order_month	order_count	monthly_avg_order_c
1	2016	9	4	1435.0
2	2016	10	324	1653.0
3	2016	12	1	2837.0
4	2017	1	800	4034.5

insight:

- There is a noticeable increase in the number of orders from October 2016 to November 2017, with consistent growth month over month.
- There is a seasonal trend, with the number of orders peaking in the middle months of the year (May to August) and decreasing towards the end and beginning of each year.
- The monthly average order count fluctuates but generally shows higher values in the middle months of the year and lower values towards the beginning and end of each year.
- There are anomalies in September and October 2016 and 2018, where the number of orders is significantly lower compared to other months, possibly due to specific events or factors affecting order volume during those periods.

Overall, there is a clear monthly seasonality pattern in the number of orders being placed, with peaks and troughs evident throughout the years analyzed.

Recommendation :

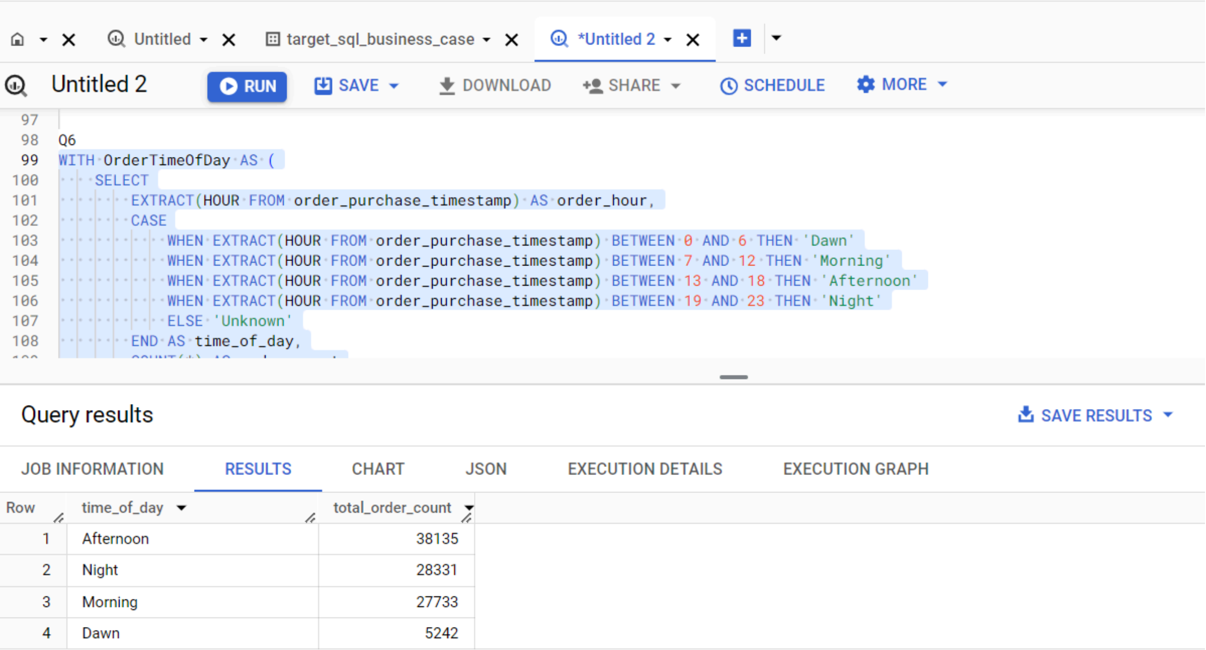
1. Sustain Growth: Maintain strategies to continue the observed growth trend from October 2016 to November 2017.
2. Seasonal Marketing: Launch targeted campaigns during peak months (May to August) to maximize order volume.
3. Monthly Analysis: Conduct monthly analysis for proactive planning and resource allocation.
4. Investigate Anomalies: Explore reasons behind order volume dips in September and October 2016 and 2018.
5. Operational Efficiency: Optimize operations to handle fluctuations in order volume effectively.
6. Customer Engagement: Enhance customer experience to retain and attract customers amid changing order trends.

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- ○ 0-6 hrs : Dawn
- ○ 7-12 hrs : Mornings
- ○ 13-18 hrs : Afternoon
- ○ 19-23 hrs : Night

Ans:

```
WITH OrderTimeOfDay AS (
    SELECT
        EXTRACT(HOUR FROM order_purchase_timestamp) AS order_hour,
        CASE
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN
'Dawn'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN
'Morning'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'Afternoon'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN
'Night'
            ELSE 'Unknown'
        END AS time_of_day,
        COUNT(*) AS order_count
    FROM
        `target_sql_business_case.orders`
    GROUP BY
        order_hour,
        time_of_day
)

SELECT
    time_of_day,
    SUM(order_count) AS total_order_count
FROM
    OrderTimeOfDay
GROUP BY
    time_of_day
ORDER BY
    total_order_count DESC;
```



```

Q6
WITH OrderTimeOfDay AS (
    SELECT
        EXTRACT(HOUR FROM order_purchase_timestamp) AS order_hour,
        CASE
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
            WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
            ELSE 'Unknown'
        END AS time_of_day,
```

Row	time_of_day	total_order_count
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

insight:

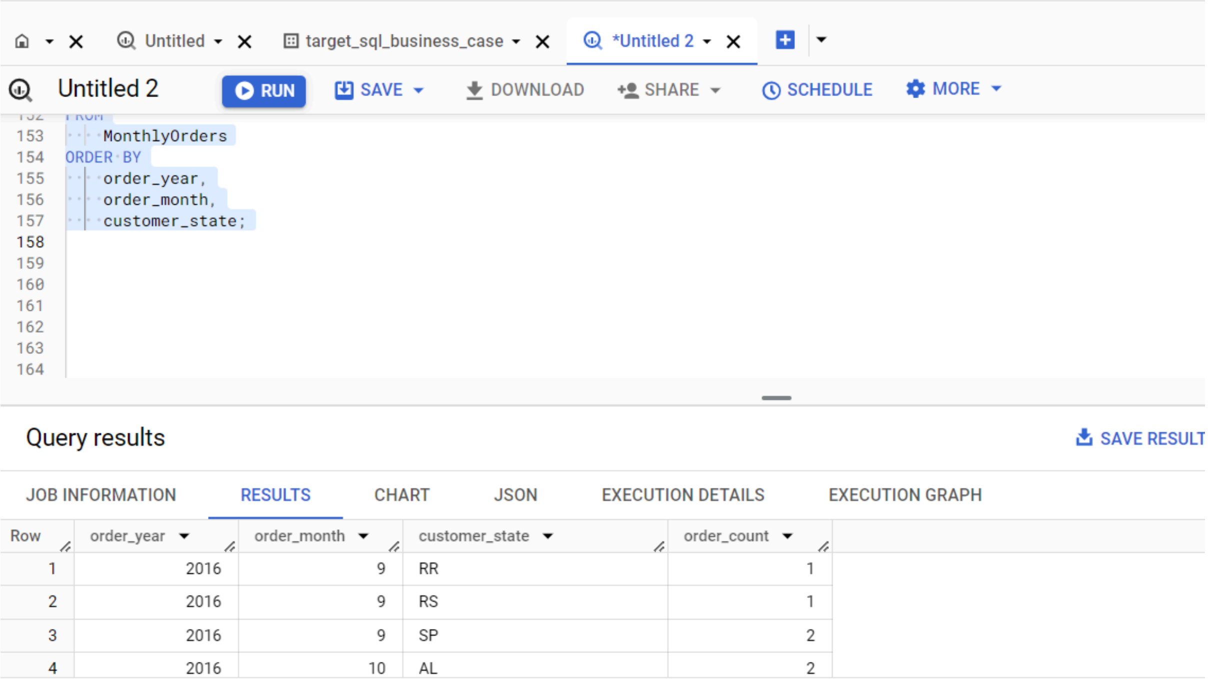
- Brazilian customers mostly place their orders in the afternoon, with a total order count of 38,135.
- Nighttime follows closely behind, with a total order count of 28,331.
- Morning orders are slightly lower than nighttime, with a total order count of 27,733.
- Dawn has the lowest order count among the time slots, with a total order count of 5,242.

In short, Brazilian customers prefer to place their orders during the afternoon and nighttime, with morning orders being slightly less frequent, and dawn orders being the least common.

Evolution of E-commerce orders in the Brazil region:

1. Get the month on month no. of orders placed in each state.

```
Ans:
WITH MonthlyOrders AS (
    SELECT
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
        c.customer_state,
        COUNT(*) AS order_count
    FROM
        `target_sql_business_case.orders` AS o
    JOIN
        `target_sql_business_case.customers` AS c
    ON
        o.customer_id = c.customer_id
    GROUP BY
        order_year,
        order_month,
        c.customer_state
)

SELECT
    order_year,
    order_month,
    customer_state,
    order_count
FROM
    MonthlyOrders
ORDER BY
    order_year,
    order_month,
    customer_state;
```

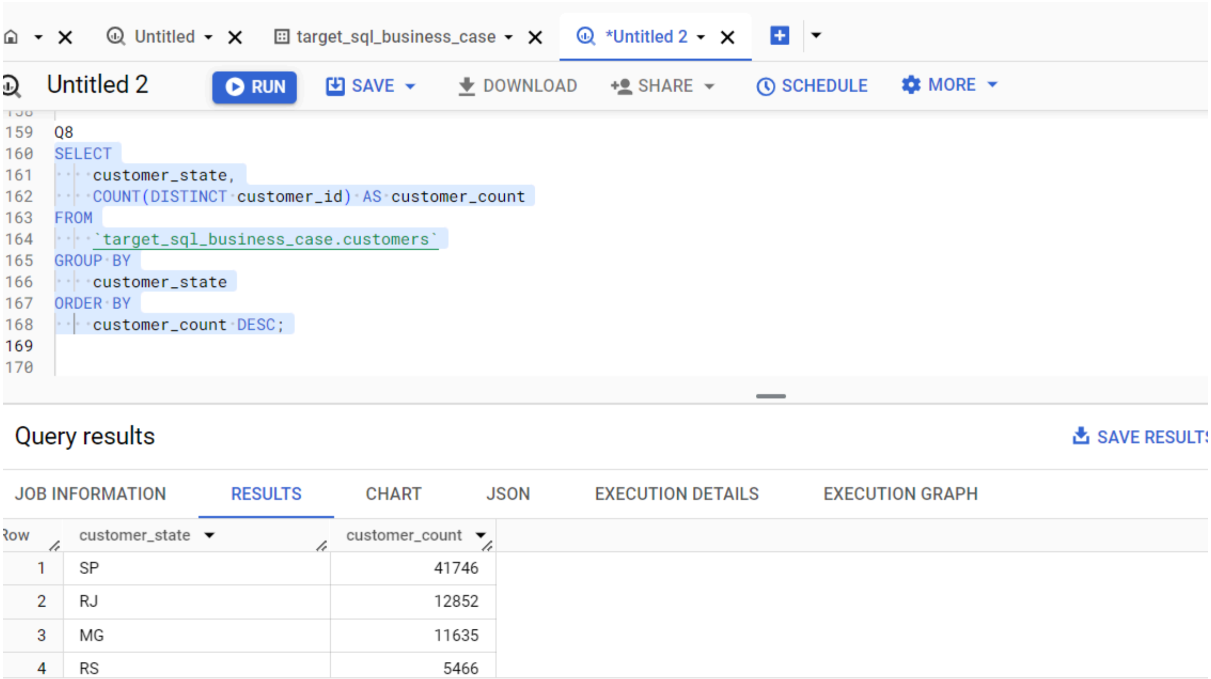


```
FROM
    MonthlyOrders
ORDER BY
    order_year,
    order_month,
    customer_state;
```

Row	order_year	order_month	customer_state	order_count
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2

2. How are the customers distributed across all the states?

```
SELECT
    customer_state,
    COUNT(DISTINCT customer_id) AS customer_count
FROM
    `target_sql_business_case.customers`
GROUP BY
    customer_state
ORDER BY
    customer_count DESC;
```



```
Q8
SELECT
    customer_state,
    COUNT(DISTINCT customer_id) AS customer_count
FROM
    `target_sql_business_case.customers`
GROUP BY
    customer_state
ORDER BY
    customer_count DESC;
```

Row	customer_state	customer_count
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466

insight:

- The state with the highest number of customers is São Paulo (SP) with 41,746 customers.
- Rio de Janeiro (RJ) follows with 12,852 customers, and Minas Gerais (MG) is next with 11,635 customers.
- Other states have varying numbers of customers, with smaller states like Roraima (RR) and Amapá (AP) having fewer customers.

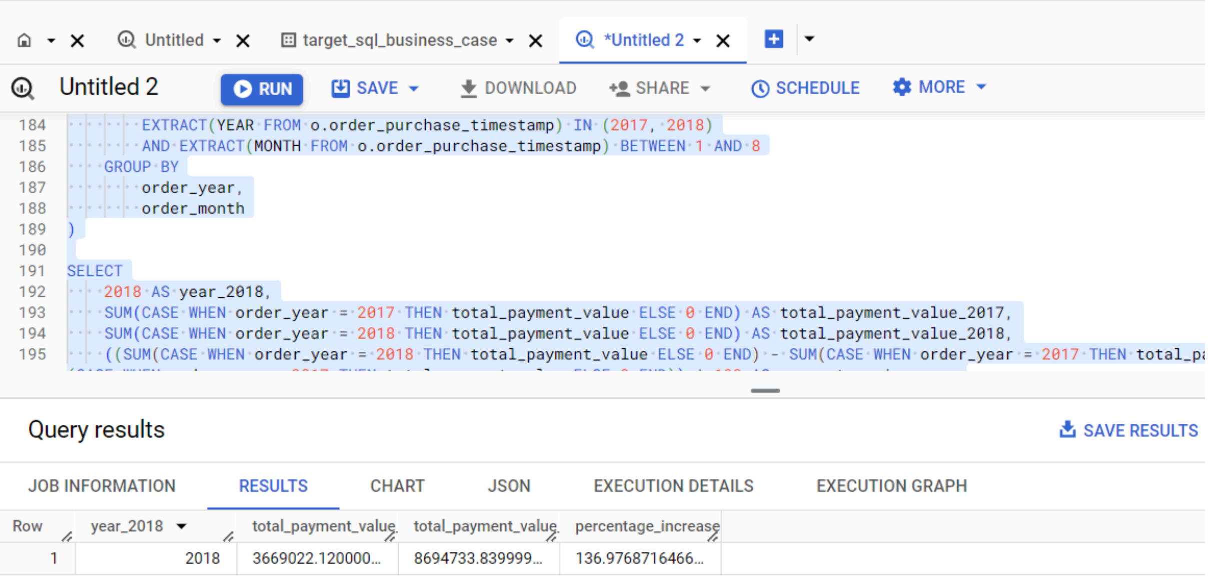
In short, São Paulo has the largest customer base, followed by Rio de Janeiro and Minas Gerais, while other states have fewer customers comparatively.

Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
You can use the "payment_value" column in the payments table to get the cost of orders.

Ans:

```
WITH OrderCost AS (
    SELECT
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
        SUM(p.payment_value) AS total_payment_value
    FROM
        `target_sql_business_case.orders` AS o
    JOIN
        `target_sql_business_case.payments` AS p
    ON
        o.order_id = p.order_id
    WHERE
        EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)
        AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
    GROUP BY
        order_year,
        order_month
)

SELECT
    2018 AS year_2018,
    SUM(CASE WHEN order_year = 2017 THEN total_payment_value ELSE 0 END) AS
total_payment_value_2017,
    SUM(CASE WHEN order_year = 2018 THEN total_payment_value ELSE 0 END) AS
total_payment_value_2018,
    ((SUM(CASE WHEN order_year = 2018 THEN total_payment_value ELSE 0 END) -
SUM(CASE WHEN order_year = 2017 THEN total_payment_value ELSE 0 END)) / SUM(CASE
WHEN order_year = 2017 THEN total_payment_value ELSE 0 END)) * 100 AS
percentage_increase
FROM
    OrderCost;
```



```sql
        EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)
        AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
    GROUP BY
        order_year,
        order_month
)

SELECT
    2018 AS year_2018,
    SUM(CASE WHEN order_year = 2017 THEN total_payment_value ELSE 0 END) AS total_payment_value_2017,
    SUM(CASE WHEN order_year = 2018 THEN total_payment_value ELSE 0 END) AS total_payment_value_2018,
    ((SUM(CASE WHEN order_year = 2018 THEN total_payment_value ELSE 0 END) - SUM(CASE WHEN order_year = 2017 THEN total_pa
```

Row	year_2018	total_payment_value	total_payment_value	percentage_increase
1	2018	3669022.120000...	8694733.839999...	136.9768716466...

insight:

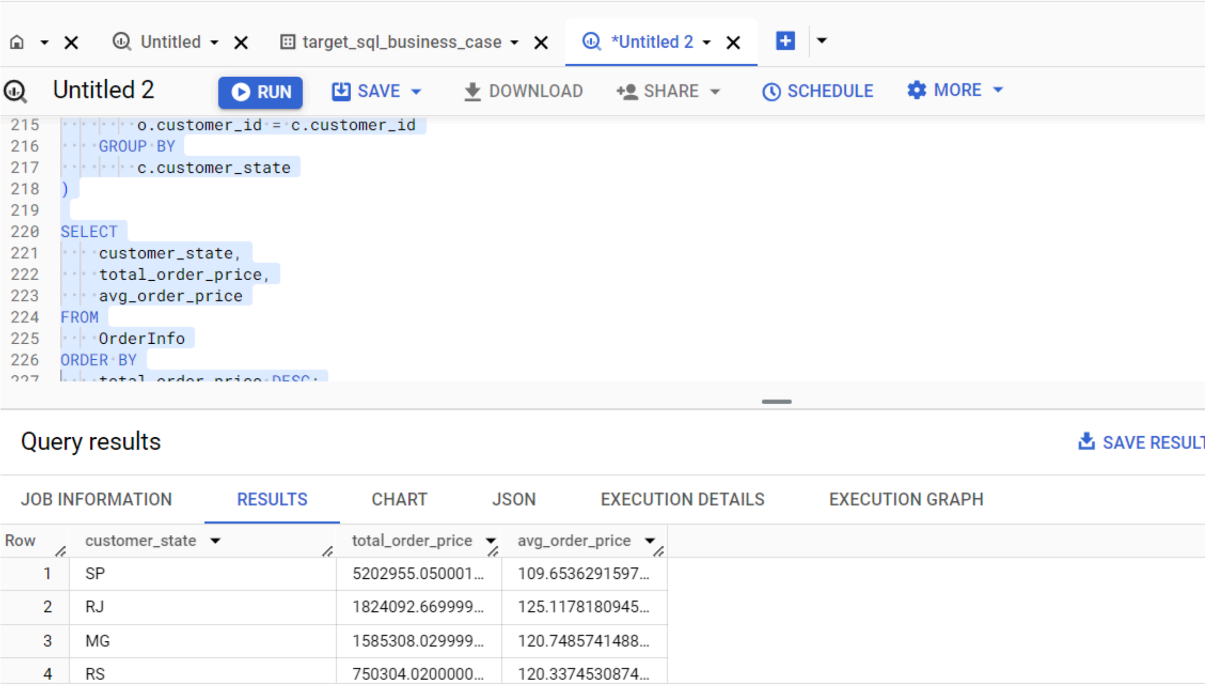
Total payment value for orders in 2017: 3,669,022.12
Total payment value for orders in 2018: 8,694,733.84
Percentage increase in the cost of orders from 2017 to 2018: Approximately 136.98%
The percentage increase indicates that the cost of orders increased by approximately 136.98% from 2017 to 2018 during the specified months (January to August). This suggests a significant growth in the money movement by e-commerce within this period

2. Calculate the Total & Average value of order price for each state.

Ans:

```sql
WITH OrderInfo AS (
    SELECT
        c.customer_state,
        SUM(oi.price) AS total_order_price,
        AVG(oi.price) AS avg_order_price
    FROM
        `target_sql_business_case.orders` AS o
    JOIN
        `target_sql_business_case.order_items` AS oi
    ON
        o.order_id = oi.order_id
    JOIN
```

```
        `target_sql_business_case.customers` AS c
    ON
        o.customer_id = c.customer_id
    GROUP BY
        c.customer_state
)

SELECT
    customer_state,
    total_order_price,
    avg_order_price
FROM
    OrderInfo
ORDER BY
    total_order_price DESC;
```



```
        o.customer_id = c.customer_id
    GROUP BY
        c.customer_state
)

SELECT
    customer_state,
    total_order_price,
    avg_order_price
FROM
    OrderInfo
ORDER BY
    total_order_price DESC;
```

Query results

Row	customer_state	total_order_price	avg_order_price
1	SP	5202955.050001...	109.6536291597...
2	RJ	1824092.669999...	125.1178180945...
3	MG	1585308.029999...	120.7485741488...
4	RS	750304.0200000...	120.3374530874...

insight:

1. São Paulo (SP) has the highest total order price of approximately $5,202,955.05, with an average order price of $109.65.
2. Rio de Janeiro (RJ) follows with a total order price of around $1,824,092.67 and an average order price of $125.12.
3. Minas Gerais (MG) has a total order price of approximately $1,585,308.03, with an average order price of $120.75.

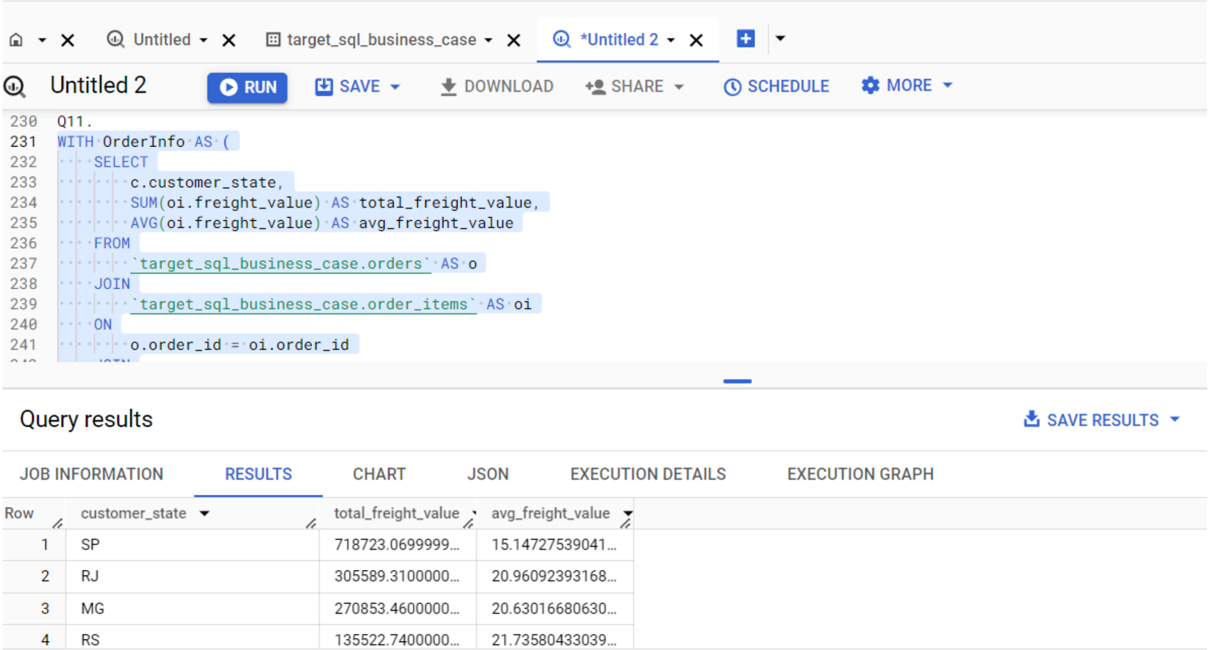
4. States like Bahia (BA), Pernambuco (PE), and Ceará (CE) exhibit higher average order prices compared to other states, indicating potentially higher-value purchases or different purchasing behaviors.
5. Smaller states like Roraima (RR), Amapá (AP), and Acre (AC) show lower total order prices and average order prices compared to larger states.

In summary, São Paulo leads in both total and average order prices, while other states exhibit varying levels of order prices, reflecting diverse economic activities and consumer behaviors across regions.

3. Calculate the Total & Average value of order freight for each state.

Ans:

```
WITH OrderInfo AS (
    SELECT
        c.customer_state,
        SUM(oi.freight_value) AS total_freight_value,
        AVG(oi.freight_value) AS avg_freight_value
    FROM
        `target_sql_business_case.orders` AS o
    JOIN
        `target_sql_business_case.order_items` AS oi
    ON
        o.order_id = oi.order_id
    JOIN
        `target_sql_business_case.customers` AS c
    ON
        o.customer_id = c.customer_id
    GROUP BY
        c.customer_state
)

SELECT
    customer_state,
    total_freight_value,
    avg_freight_value
FROM
    OrderInfo
ORDER BY
    total_freight_value DESC;
```



```
Q11.
WITH OrderInfo AS (
    SELECT
        c.customer_state,
        SUM(oi.freight_value) AS total_freight_value,
        AVG(oi.freight_value) AS avg_freight_value
    FROM
        `target_sql_business_case.orders` AS o
    JOIN
        `target_sql_business_case.order_items` AS oi
    ON
        o.order_id = oi.order_id
```

Row	customer_state	total_freight_value	avg_freight_value
1	SP	718723.0699999...	15.14727539041...
2	RJ	305589.3100000...	20.96092393168...
3	MG	270853.4600000...	20.63016680630...
4	RS	135522.7400000...	21.73580433039...

insight:

1. São Paulo (SP) has the highest total freight value of approximately $718,723.07, with an average freight value of $15.15.
2. Rio de Janeiro (RJ) follows with a total freight value of around $305,589.31 and an average freight value of $20.96.
3. States like Bahia (BA), Pernambuco (PE), and Ceará (CE) exhibit higher average freight values compared to other states, indicating potentially higher shipping costs or different shipping methods.
4. Smaller states like Roraima (RR), Amapá (AP), and Acre (AC) show lower total freight values and average freight values compared to larger states.

In summary, São Paulo leads in both total and average freight values, while other states exhibit varying levels of freight costs, reflecting diverse shipping expenses across regions.

Analysis based on sales, freight and delivery time.

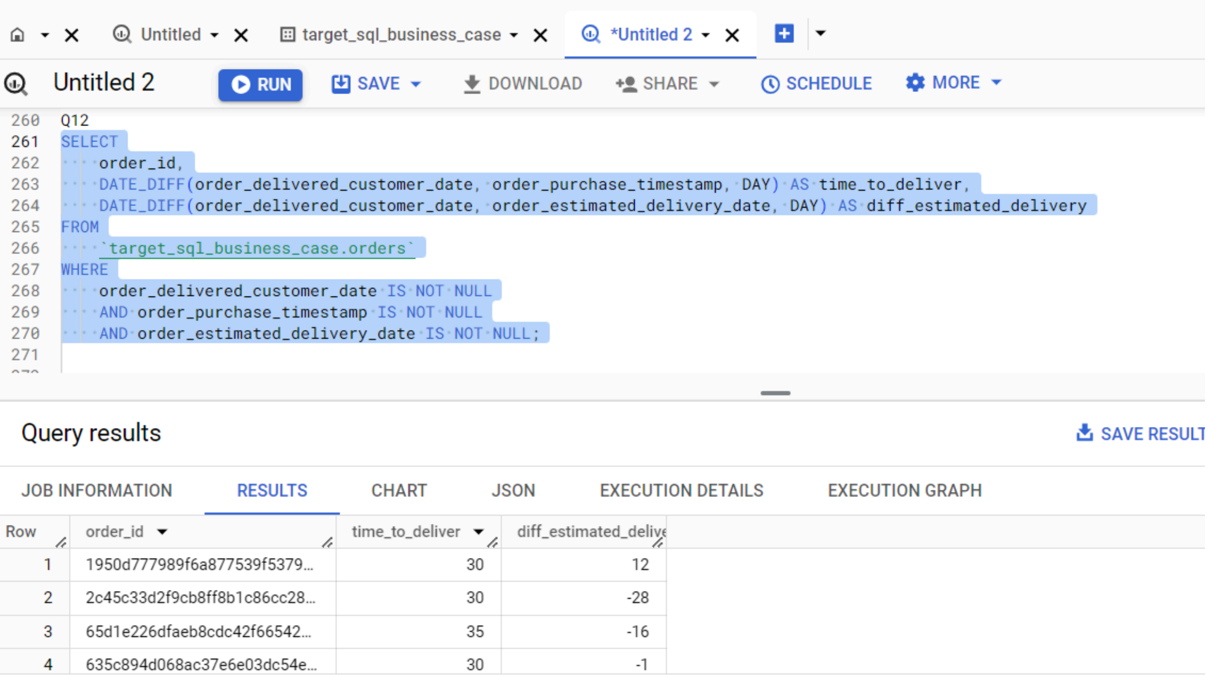
1. Find the no. of days taken to deliver each order from the order’s purchase date as delivery time.
   Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
   Do this in a single query.

   You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

   ○ time_to_deliver = order_delivered_customer_date - order_purchase_timestamp

   ○ diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

```
        Ans:
        SELECT
    order_id,
    DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
time_to_deliver,
    DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) AS
diff_estimated_delivery
FROM
    `target_sql_business_case.orders`
WHERE
    order_delivered_customer_date IS NOT NULL
    AND order_purchase_timestamp IS NOT NULL
    AND order_estimated_delivery_date IS NOT NULL;
```



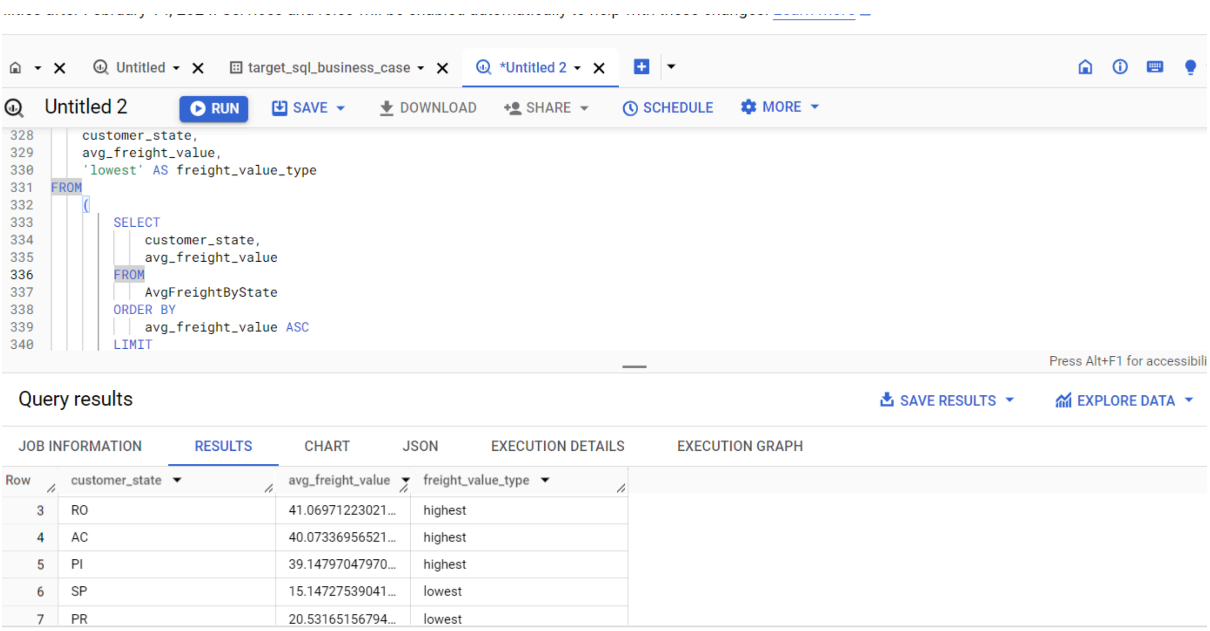
```
Q12
SELECT
    order_id,
    DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS time_to_deliver,
    DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) AS diff_estimated_delivery
FROM
    `target_sql_business_case.orders`
WHERE
    order_delivered_customer_date IS NOT NULL
    AND order_purchase_timestamp IS NOT NULL
    AND order_estimated_delivery_date IS NOT NULL;
```

Row	order_id	time_to_deliver	diff_estimated_delive
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1

2. Find out the top 5 states with the highest & lowest average freight value.

Ans:

```
WITH AvgFreightByState AS (
    SELECT
        c.customer_state,
        AVG(oi.freight_value) AS avg_freight_value
    FROM
        `target_sql_business_case.order_items` AS oi
    JOIN
        `target_sql_business_case.orders` AS o ON oi.order_id = o.order_id
    JOIN
        `target_sql_business_case.customers` AS c ON o.customer_id = c.customer_id
    GROUP BY
        c.customer_state
)

SELECT
    customer_state,
    avg_freight_value,
    'highest' AS freight_value_type
FROM
    (
        SELECT
            customer_state,
            avg_freight_value
        FROM
            AvgFreightByState
        ORDER BY
            avg_freight_value DESC
        LIMIT
            5
    )

UNION ALL

SELECT
    customer_state,
    avg_freight_value,
    'lowest' AS freight_value_type
FROM
    (
        SELECT
            customer_state,
            avg_freight_value
        FROM
            AvgFreightByState
        ORDER BY
            avg_freight_value ASC
        LIMIT
            5
    );
```



```
    customer_state,
    avg_freight_value,
    'lowest' AS freight_value_type
FROM
    (
        SELECT
            customer_state,
            avg_freight_value
        FROM
            AvgFreightByState
        ORDER BY
            avg_freight_value ASC
        LIMIT
```

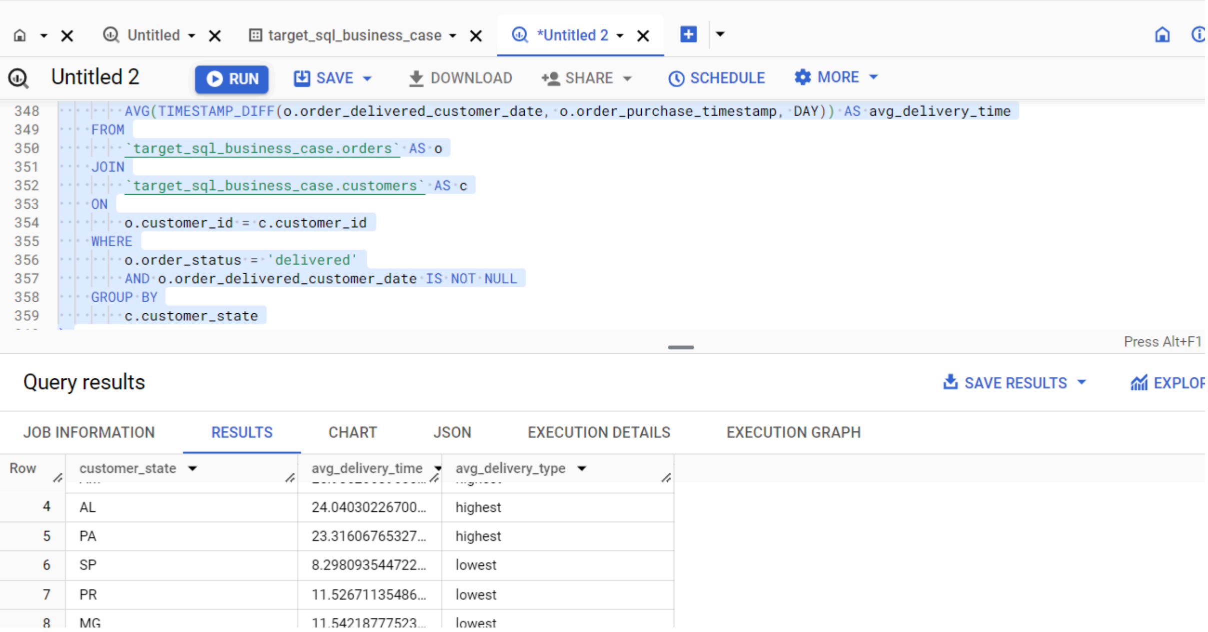
Row	customer_state	avg_freight_value	freight_value_type
3	RO	41.06971223021...	highest
4	AC	40.07336956521...	highest
5	PI	39.14797047970...	highest
6	SP	15.14727539041...	lowest
7	PR	20.53165156794...	lowest

3. Find out the top 5 states with the highest & lowest average delivery time.

Ans:

```
WITH DeliveryTimes AS (
    SELECT
        c.customer_state,
        AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)) AS avg_delivery_time
    FROM
        `target_sql_business_case.orders` AS o
    JOIN
        `target_sql_business_case.customers` AS c
    ON
        o.customer_id = c.customer_id
    WHERE
        o.order_status = 'delivered'
        AND o.order_delivered_customer_date IS NOT NULL
    GROUP BY
        c.customer_state
)

SELECT
    customer_state,
    avg_delivery_time,
    'highest' AS avg_delivery_type
FROM (
      SELECT
          customer_state,
          avg_delivery_time
      FROM
          DeliveryTimes
      ORDER BY
          avg_delivery_time DESC
      LIMIT
          5
      )

UNION ALL

SELECT
    customer_state,
    avg_delivery_time,
    'lowest' AS avg_delivery_type
```

```
FROM (
    SELECT
      customer_state,
      avg_delivery_time
    FROM
      DeliveryTimes
    ORDER BY
      avg_delivery_time ASC
    LIMIT
      5
    );
```



```
        AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS avg_delivery_time
    FROM
        `target_sql_business_case.orders` AS o
    JOIN
        `target_sql_business_case.customers` AS c
    ON
        o.customer_id = c.customer_id
    WHERE
        o.order_status = 'delivered'
        AND o.order_delivered_customer_date IS NOT NULL
    GROUP BY
        c.customer_state
```

Row	customer_state	avg_delivery_time	avg_delivery_type
4	AL	24.04030226700...	highest
5	PA	23.31606765327...	highest
6	SP	8.298093544722...	lowest
7	PR	11.52671135486...	lowest
8	MG	11.54218777523...	lowest

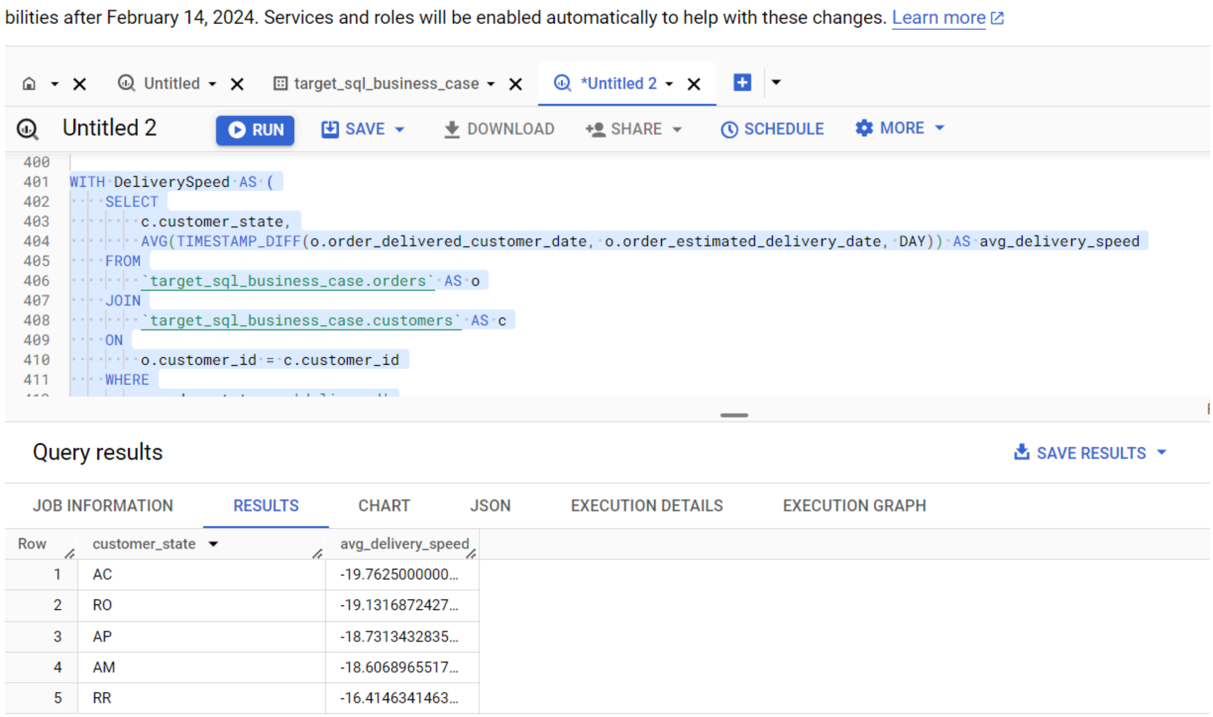
```
insight:
- States with the highest average delivery time:
  1. RR (Roraima) - Average delivery time: 28.98 days
  2. AP (Amapá) - Average delivery time: 26.73 days
  3. AM (Amazonas) - Average delivery time: 25.99 days
  4. AL (Alagoas) - Average delivery time: 24.04 days
  5. PA (Pará) - Average delivery time: 23.32 days

- States with the lowest average delivery time:
  1. SP (São Paulo) - Average delivery time: 8.30 days
  2. PR (Paraná) - Average delivery time: 11.53 days
  3. MG (Minas Gerais) - Average delivery time: 11.54 days
  4. DF (Distrito Federal) - Average delivery time: 12.51 days
  5. SC (Santa Catarina) - Average delivery time: 14.48 days
```

The sorting order is based on the `avg_delivery_time` field in ascending order for the lowest average delivery time, and descending order for the highest average delivery time.

4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Ans:

```
WITH DeliverySpeed AS (
    SELECT
        c.customer_state,
        AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS avg_delivery_speed
    FROM
        `target_sql_business_case.orders` AS o
    JOIN
        `target_sql_business_case.customers` AS c
    ON
        o.customer_id = c.customer_id
    WHERE
        o.order_status = 'delivered'
        AND o.order_delivered_customer_date IS NOT NULL
        AND o.order_estimated_delivery_date IS NOT NULL
    GROUP BY
        c.customer_state
)

SELECT
    customer_state,
    avg_delivery_speed
FROM
    DeliverySpeed
ORDER BY
    avg_delivery_speed ASC
LIMIT
    5;
```



bilities after February 14, 2024. Services and roles will be enabled automatically to help with these changes. Learn more

```
WITH DeliverySpeed AS (
  SELECT
    c.customer_state,
    AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS avg_delivery_speed
  FROM
    `target_sql_business_case.orders` AS o
  JOIN
    `target_sql_business_case.customers` AS c
  ON
    o.customer_id = c.customer_id
  WHERE
```

Row	customer_state	avg_delivery_speed
1	AC	-19.7625000000...
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6068965517...
5	RR	-16.4146341463...

insight:

1. AC:
   - Average delivery speed: -19.76 days
   - This indicates that, on average, orders in Acre are delivered approximately 19.76 days earlier than the estimated delivery date.

2. RO :
   - Average delivery speed: -19.13 days
   - Orders in Rondônia are delivered approximately 19.13 days earlier than the estimated delivery date, on average.

3. AP :
   - Average delivery speed: -18.73 days
   - In Amapá, orders are delivered about 18.73 days earlier than the estimated delivery date, on average.

4. AM:
   - Average delivery speed: -18.61 days
   - Amazonas experiences delivery speeds where orders are delivered roughly 18.61 days earlier than expected.

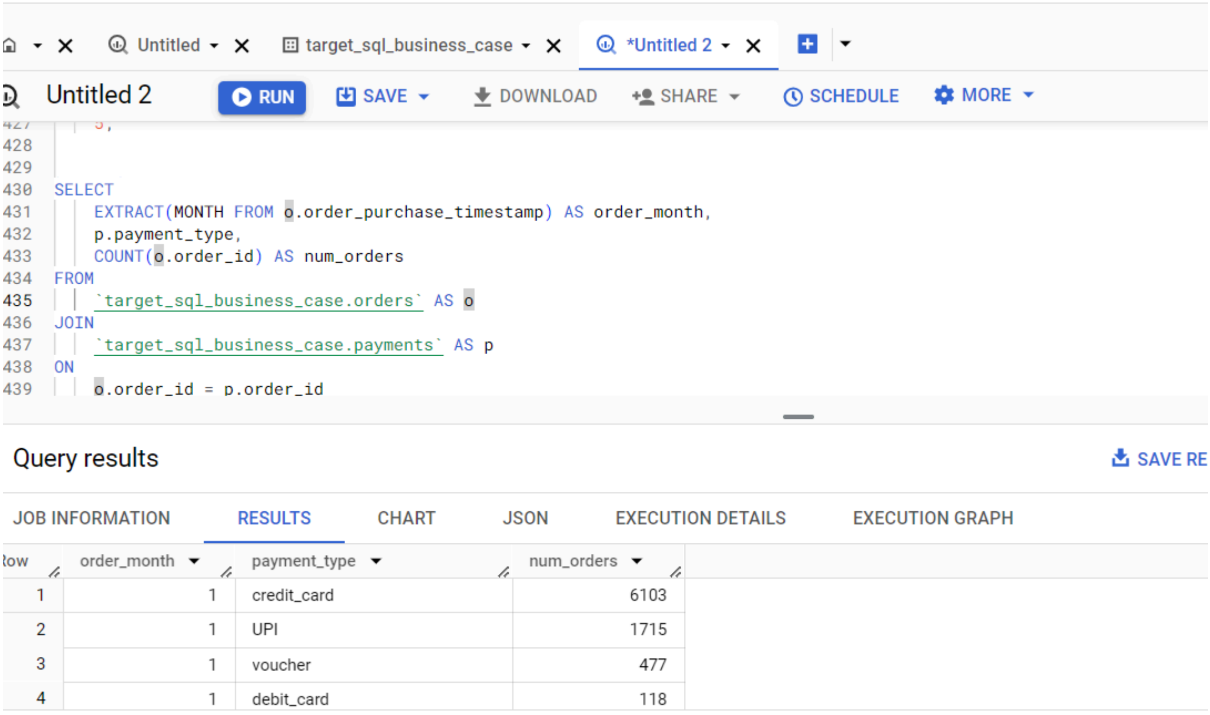
5. RR :
   - Average delivery speed: -16.41 days
   - Orders in Roraima are delivered approximately 16.41 days earlier than the estimated delivery date, on average.

Analysis based on the payments:

1. Find the month on month no. of orders placed using different payment types.

Ans:

```
SELECT
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
    p.payment_type,
    COUNT(o.order_id) AS num_orders
FROM
    `target_sql_business_case.orders` AS o
JOIN
    `target_sql_business_case.payments` AS p
ON
    o.order_id = p.order_id
GROUP BY
    order_month,
    p.payment_type
ORDER BY
    order_month,
    num_orders DESC;
```



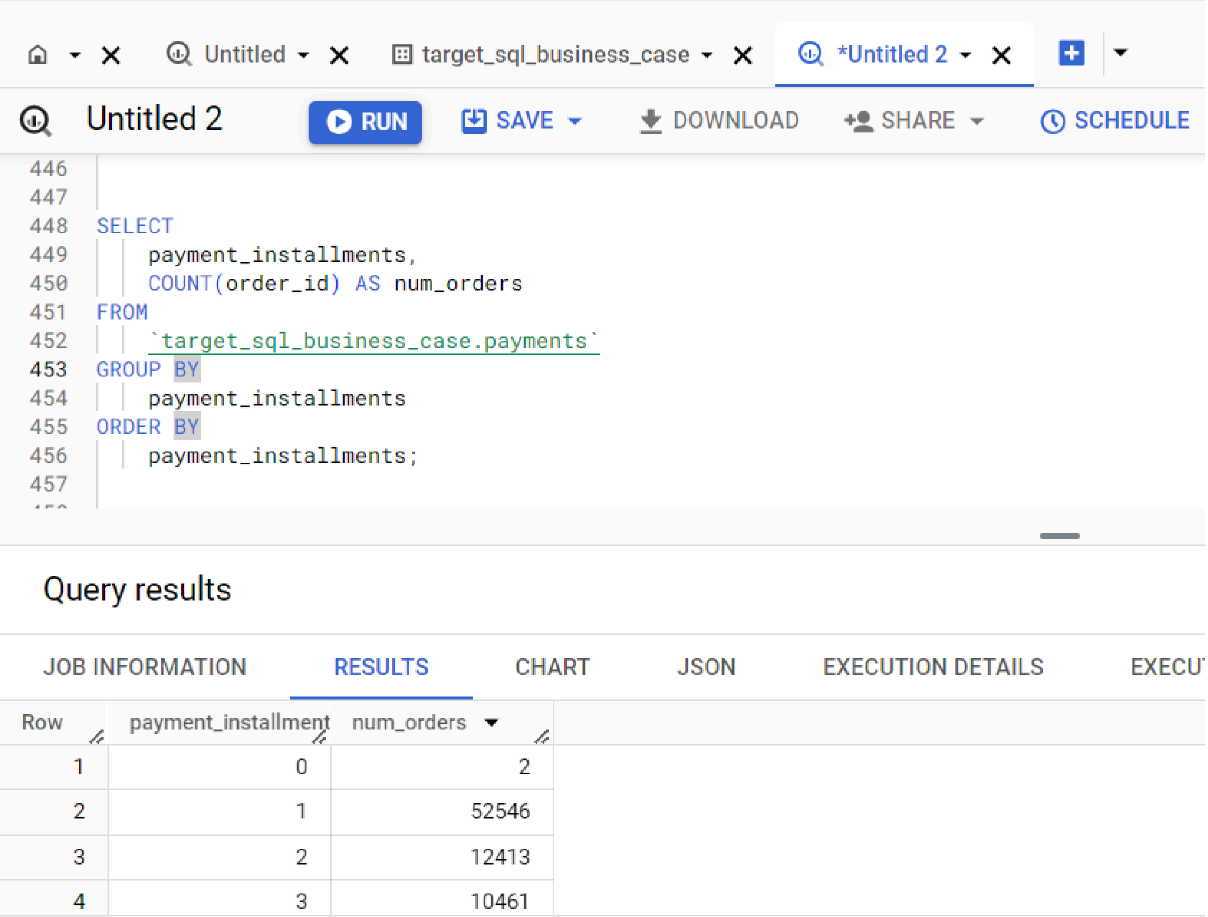
```


SELECT
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
    p.payment_type,
    COUNT(o.order_id) AS num_orders
FROM
    `target_sql_business_case.orders` AS o
JOIN
    `target_sql_business_case.payments` AS p
ON
    o.order_id = p.order_id
```

Row	order_month	payment_type	num_orders
1	1	credit_card	6103
2	1	UPI	1715
3	1	voucher	477
4	1	debit_card	118

2. Find the no. of orders placed on the basis of the payment installments that have been paid.

Ans:

```
SELECT
    payment_installments,
    COUNT(order_id) AS num_orders
FROM
    `target_sql_business_case.payments`
GROUP BY
    payment_installments
ORDER BY
    payment_installments;
```



```

SELECT
    payment_installments,
    COUNT(order_id) AS num_orders
FROM
    `target_sql_business_case.payments`
GROUP BY
    payment_installments
ORDER BY
    payment_installments;

```

Row	payment_installment	num_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461