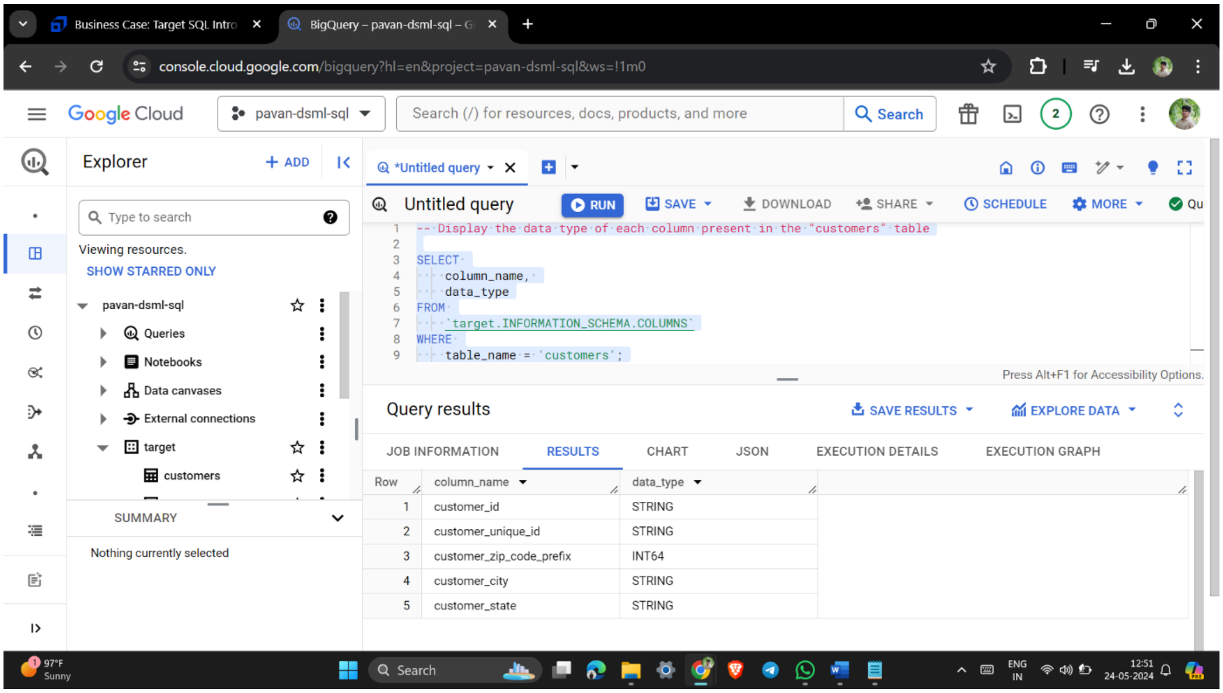
Problem Statement

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

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

QUERY:

```
SELECT
    column_name,
    data_type
FROM
    `target.INFORMATION_SCHEMA.COLUMNS`
WHERE
    table_name = 'customers';
```



```
-- Display the data type of each column present in the "customers" table

SELECT
    column_name,
    data_type
FROM
    `target.INFORMATION_SCHEMA.COLUMNS`
WHERE
    table_name = 'customers';
```

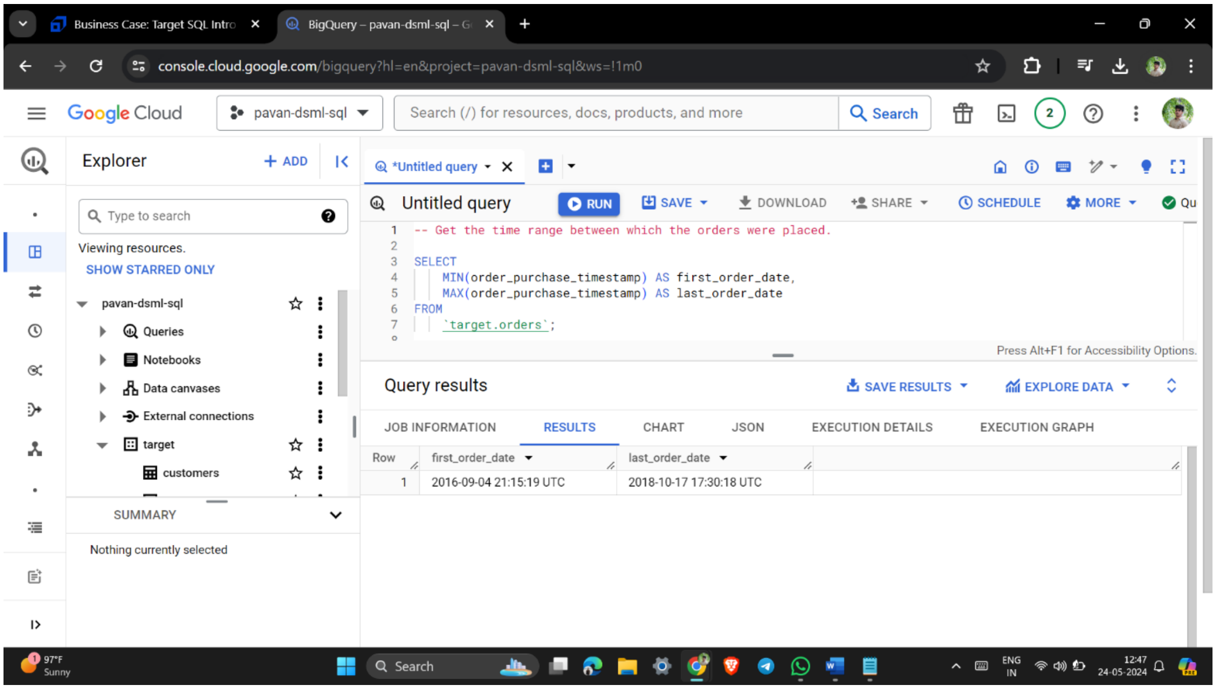
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

QUERY:

```
SELECT
    MIN(order_purchase_timestamp) AS first_order_date,
    MAX(order_purchase_timestamp) AS last_order_date
```

```
FROM
  `target.orders`;
```



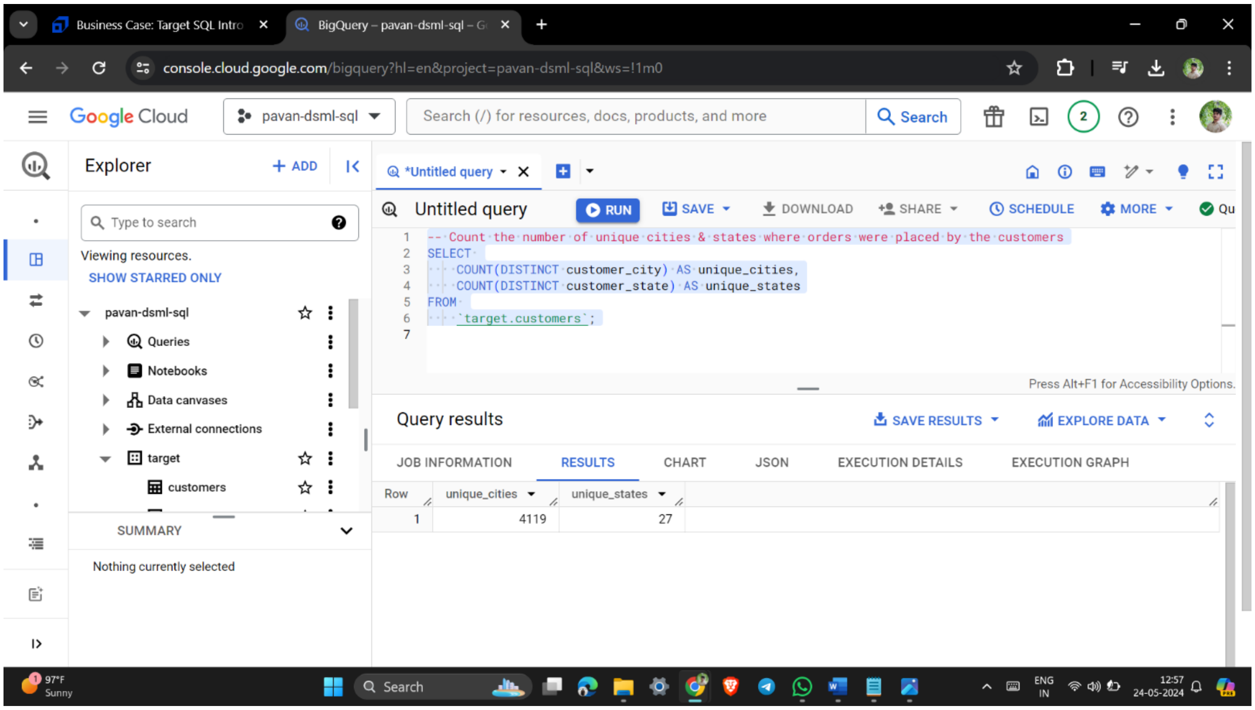
```
-- Get the time range between which the orders were placed.

SELECT
  MIN(order_purchase_timestamp) AS first_order_date,
  MAX(order_purchase_timestamp) AS last_order_date
FROM
  `target.orders`;
```

Row	first_order_date	last_order_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

QUERY:

```
SELECT
  COUNT(DISTINCT customer_city) AS unique_cities,
  COUNT(DISTINCT customer_state) AS unique_states
FROM
  `target.customers`;
```



```
-- Count the number of unique cities & states where orders were placed by the customers
SELECT
  COUNT(DISTINCT customer_city) AS unique_cities,
  COUNT(DISTINCT customer_state) AS unique_states
FROM
  `target.customers`;
```

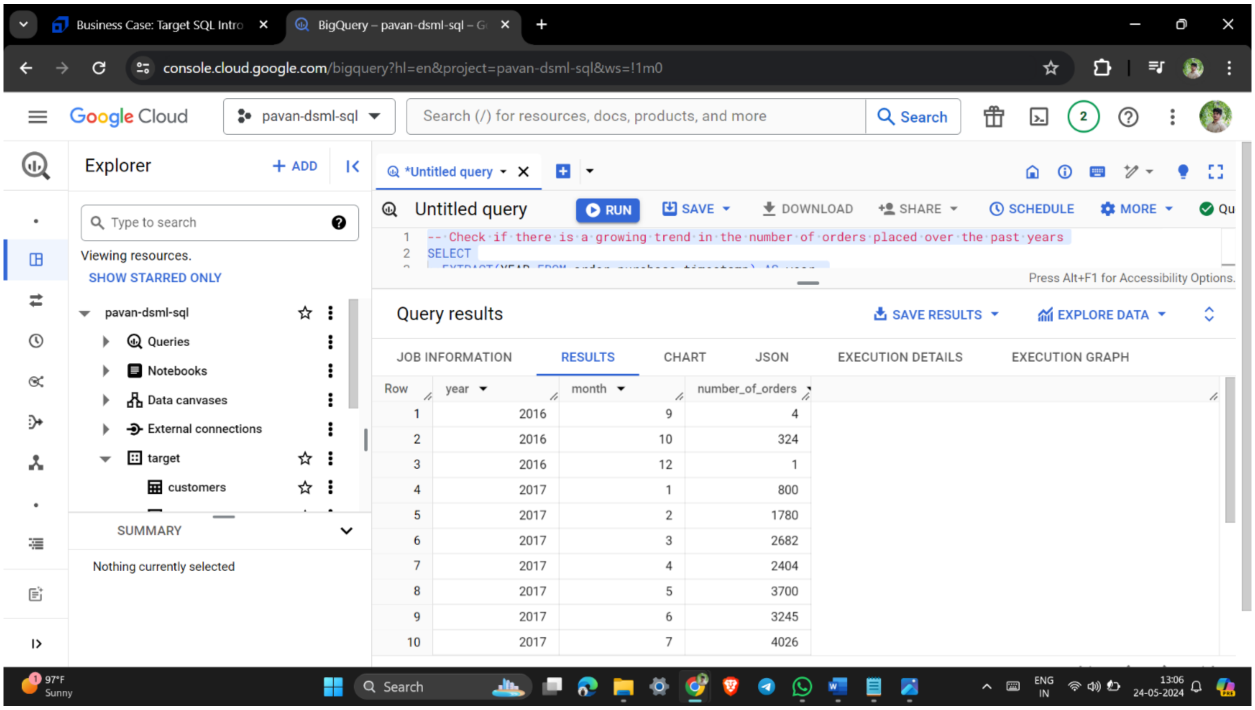
Row	unique_cities	unique_states
1	4119	27

2. In-depthExploration:

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

QUERY:

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(order_id) AS number_of_orders
FROM
  `target.orders`
GROUP BY
  year, month
ORDER BY
  year, month;
```



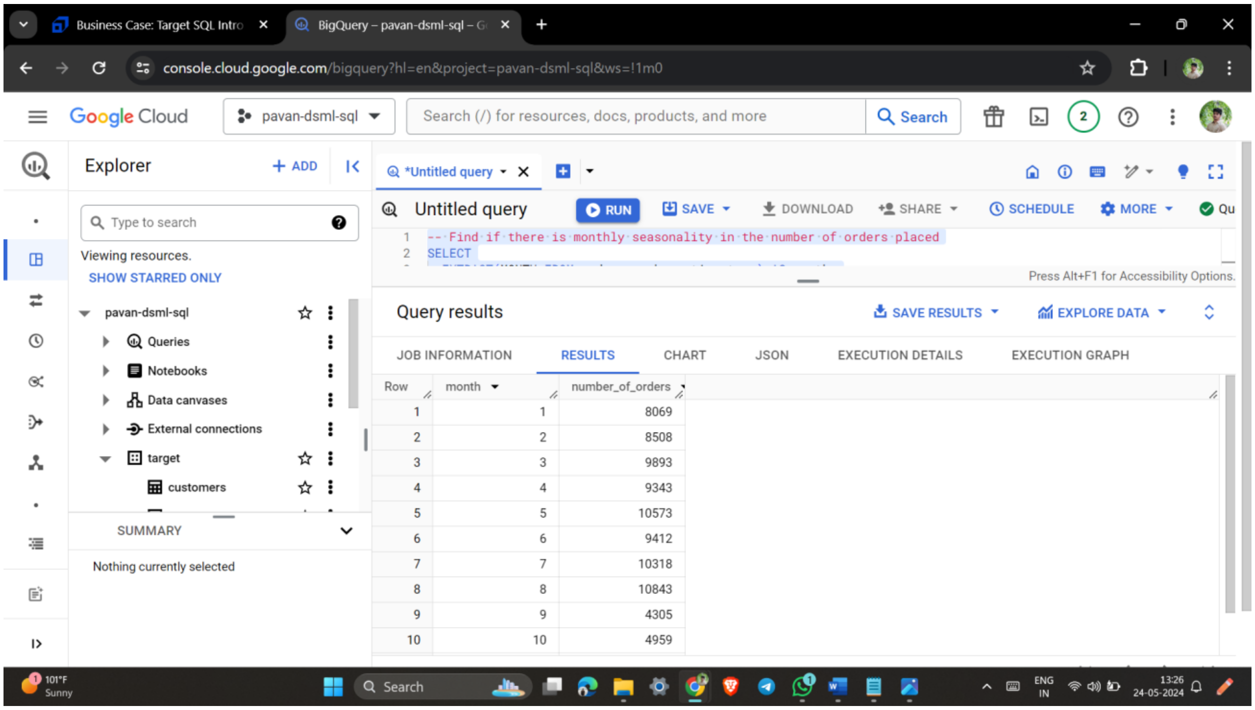
```
-- Check if there is a growing trend in the number of orders placed over the past years
SELECT
```

Row	year	month	number_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

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

QUERY:

```
SELECT
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(order_id) AS number_of_orders
FROM
  `target.orders`
GROUP BY
  month
ORDER BY
  month;
```



```
-- Find if there is monthly seasonality in the number of orders placed
SELECT
```

Row	month	number_of_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

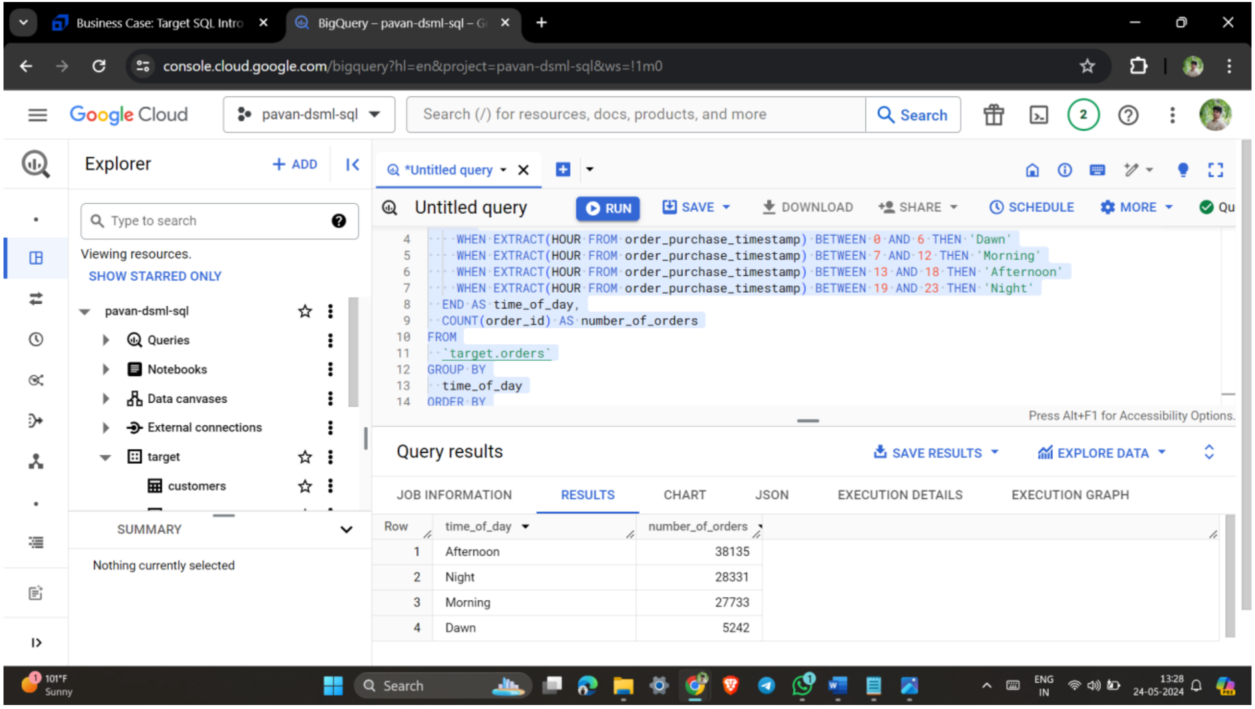
C. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

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

QUERY:

```
SELECT
  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'
  END AS time_of_day,
  COUNT(order_id) AS number_of_orders
FROM
  `target.orders`
GROUP BY
```

```
  time_of_day
ORDER BY
  number_of_orders DESC;
```



```
      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'
  END AS time_of_day,
  COUNT(order_id) AS number_of_orders
FROM
  `target.orders`
GROUP BY
  time_of_day
ORDER BY
```

Query results

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

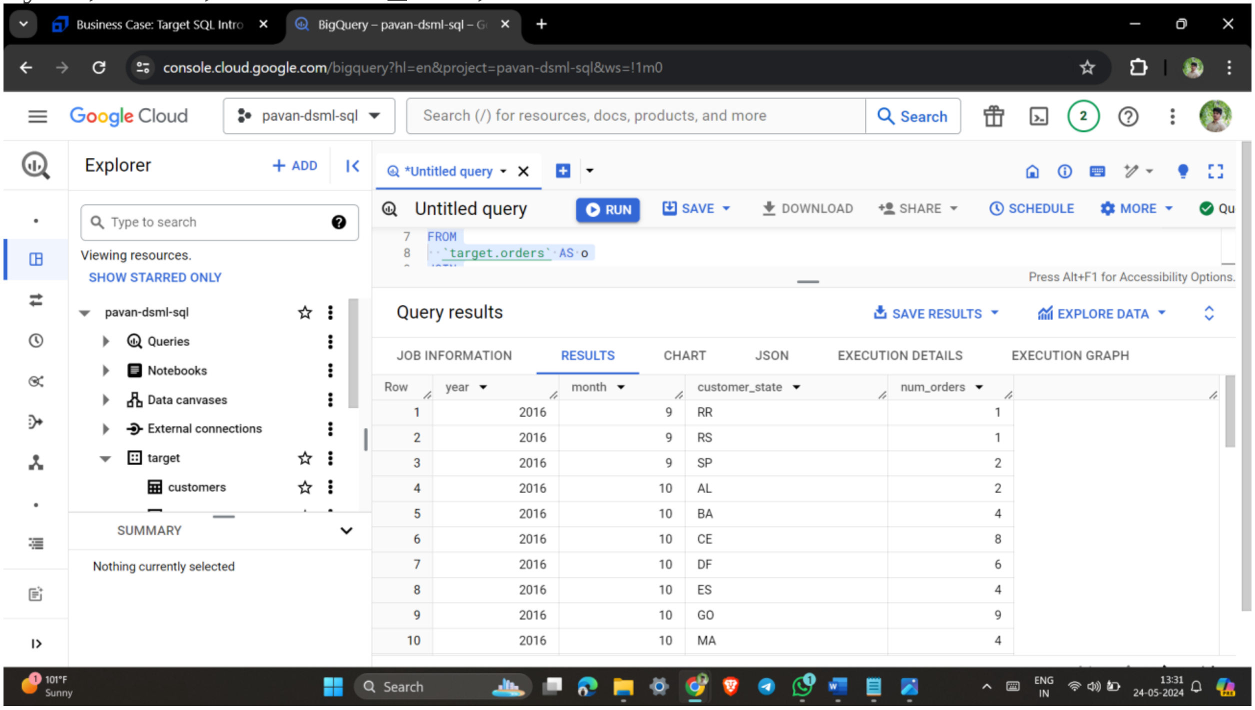
3. Evolution of E-commerce orders in the Brazil region:

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

QUERY:

```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  c.customer_state,
  COUNT(o.order_id) AS num_orders
FROM
  `target.orders` AS o
JOIN
  `target.customers` AS c
ON
  o.customer_id = c.customer_id
GROUP BY
  year, month, c.customer_state
ORDER BY
```

year, month, c.customer_state;



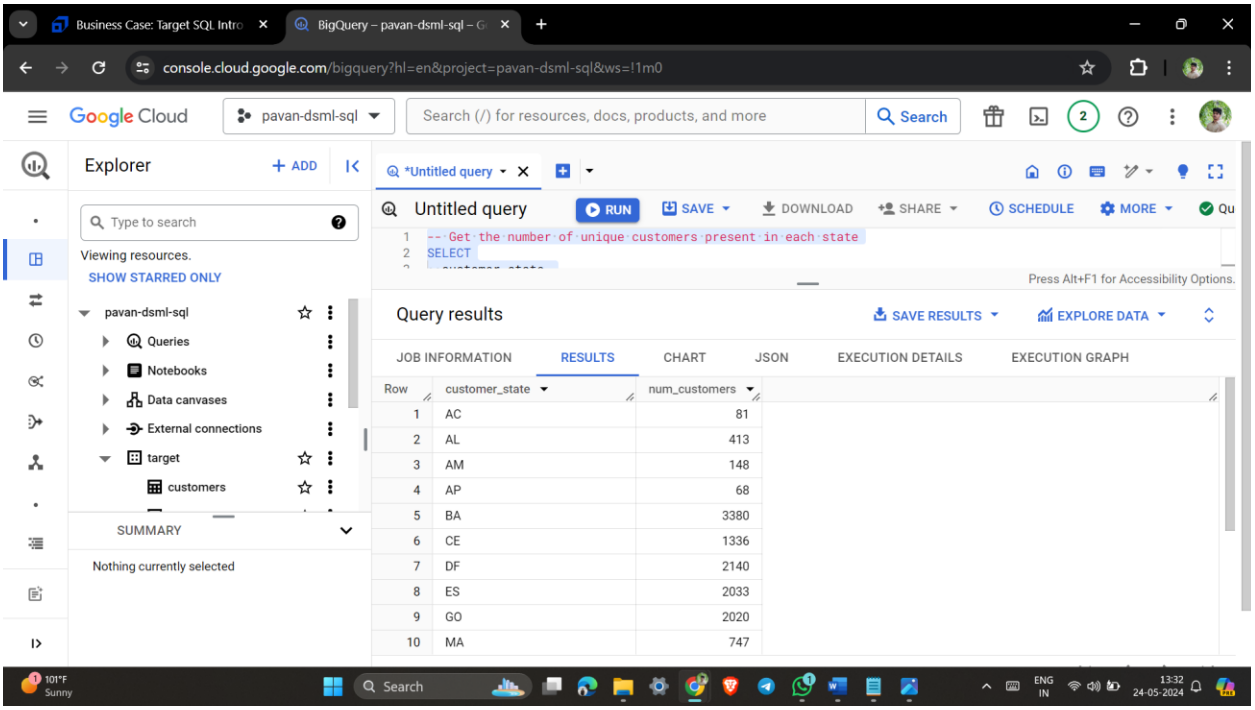
```
FROM
  `target.orders` AS o
```

Row	year	month	customer_state	num_orders
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2
5	2016	10	BA	4
6	2016	10	CE	8
7	2016	10	DF	6
8	2016	10	ES	4
9	2016	10	GO	9
10	2016	10	MA	4

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

QUERY:

```
SELECT
  customer_state,
  COUNT(DISTINCT customer_id) AS num_customers
FROM
  `target.customers`
GROUP BY
  customer_state
ORDER BY
  customer_state;
```



```
-- Get the number of unique customers present in each state
SELECT
  customer_state
```

Press Alt+F1 for Accessibility Options.

Query results

Row	customer_state	num_customers
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

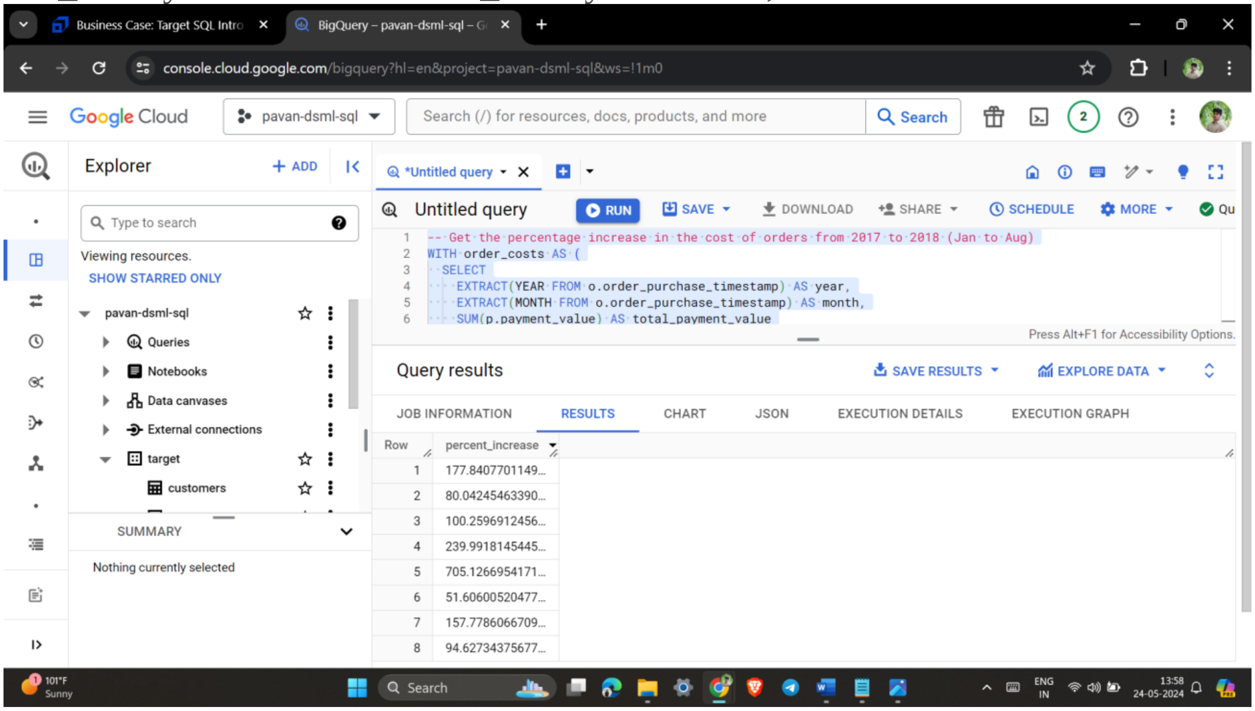
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

A. 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.

QUERY:

```
WITH order_costs AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    SUM(p.payment_value) AS total_payment_value
  FROM
    `target.orders` AS o
  JOIN
    `target.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
    year, month
)
SELECT
  (oc_2018.total_payment_value - oc_2017.total_payment_value) /
oc_2017.total_payment_value * 100 AS percent_increase
FROM
  order_costs oc_2017
JOIN
  order_costs oc_2018
ON
  oc_2017.month = oc_2018.month
WHERE
  oc_2017.year = 2017 AND oc_2018.year = 2018;
```



```
-- Get the percentage increase in the cost of orders from 2017 to 2018 (Jan to Aug)
WITH order_costs AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    SUM(p.payment_value) AS total_payment_value
```

Query results

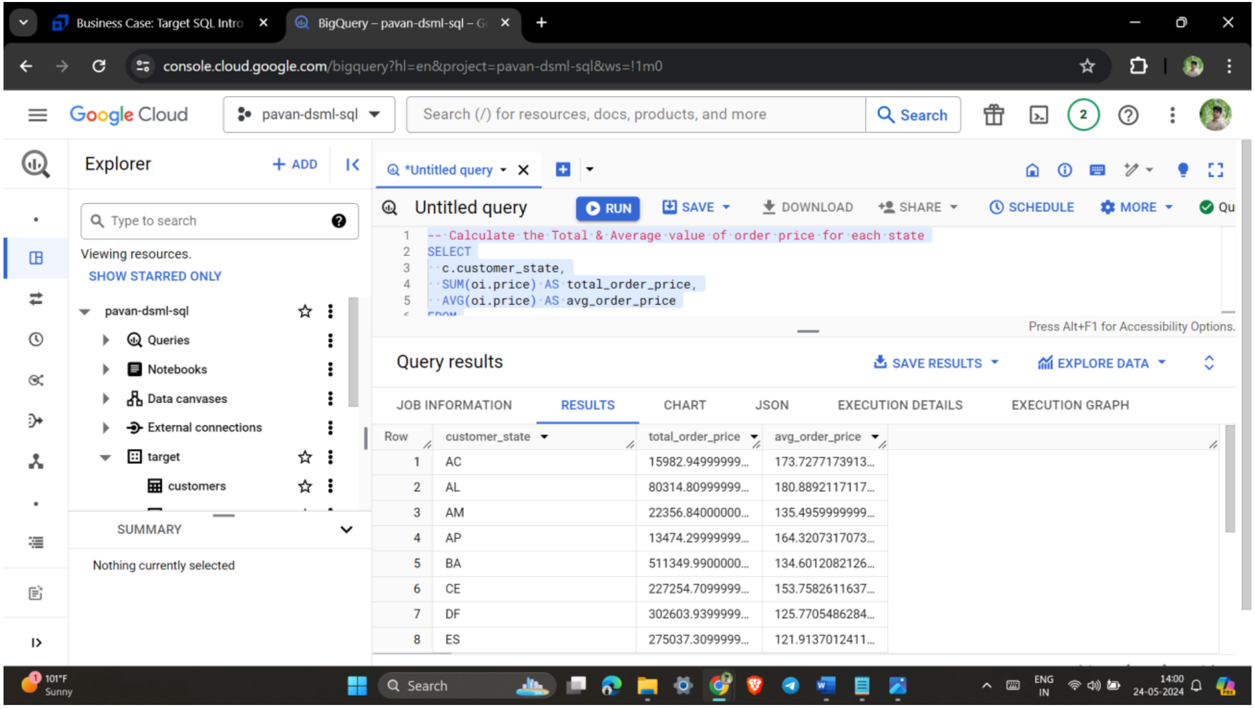
Row	percent_increase
1	177.8407701149...
2	80.04245463390...
3	100.2596912456...
4	239.9918145445...
5	705.1266954171...
6	51.60600520477...
7	157.7786066709...
8	94.62734375677...

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

QUERY:

```
SELECT
  c.customer_state,
  SUM(oi.price) AS total_order_price,
  AVG(oi.price) AS avg_order_price
FROM
  `target.order_items` AS oi
```

```
JOIN
  `target.orders` AS o
ON
  oi.order_id = o.order_id
JOIN
  `target.customers` AS c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state;
```



```
-- Calculate the Total & Average value of order price for each state
SELECT
  c.customer_state,
  SUM(oi.price) AS total_order_price,
  AVG(oi.price) AS avg_order_price
```

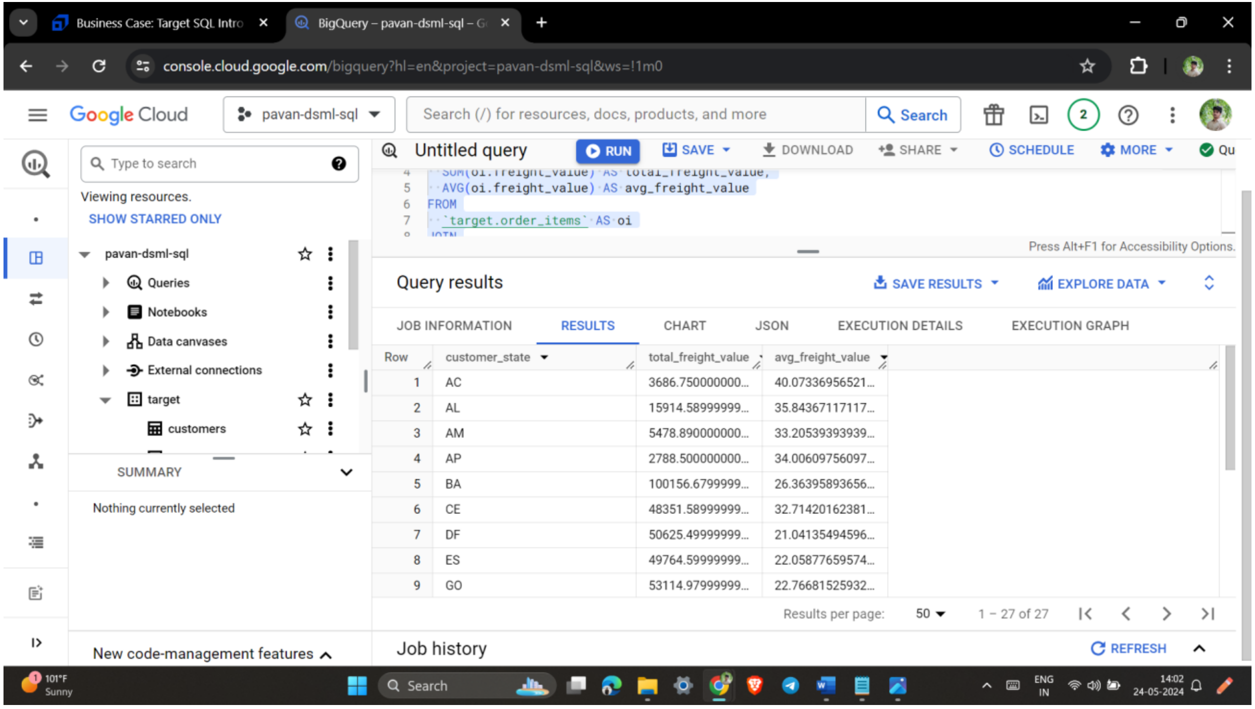
Row	customer_state	total_order_price	avg_order_price
1	AC	15982.94999999...	173.7277173913...
2	AL	80314.80999999...	180.8892117117...
3	AM	22356.84000000...	135.4959999999...
4	AP	13474.29999999...	164.3207317073...
5	BA	511349.9900000...	134.6012082126...
6	CE	227254.7099999...	153.7582611637...
7	DF	302603.9399999...	125.7705486284...
8	ES	275037.3099999...	121.9137012411...

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

QUERY:

```
SELECT
  c.customer_state,
  SUM(oi.freight_value) AS total_freight_value,
  AVG(oi.freight_value) AS avg_freight_value
FROM
  `target.order_items` AS oi
JOIN
  `target.orders` AS o
```

```
ON
  oi.order_id = o.order_id
JOIN
  `target.customers` AS c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state;
```



```
  SUM(oi.freight_value) AS total_freight_value,
  AVG(oi.freight_value) AS avg_freight_value
FROM
  `target.order_items` AS oi
JOIN
```

Query results

Row	customer_state	total_freight_value	avg_freight_value
1	AC	3686.750000000...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.890000000...	33.20539393939...
4	AP	2788.500000000...	34.00609756097...
5	BA	100156.6799999...	26.36395893656...
6	CE	48351.58999999...	32.71420162381...
7	DF	50625.49999999...	21.04135494596...
8	ES	49764.59999999...	22.05877659574...
9	GO	53114.97999999...	22.76681525932...

Results per page: 50 1 – 27 of 27

5. Analysis based on sales, freight and delivery time.

A. 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.

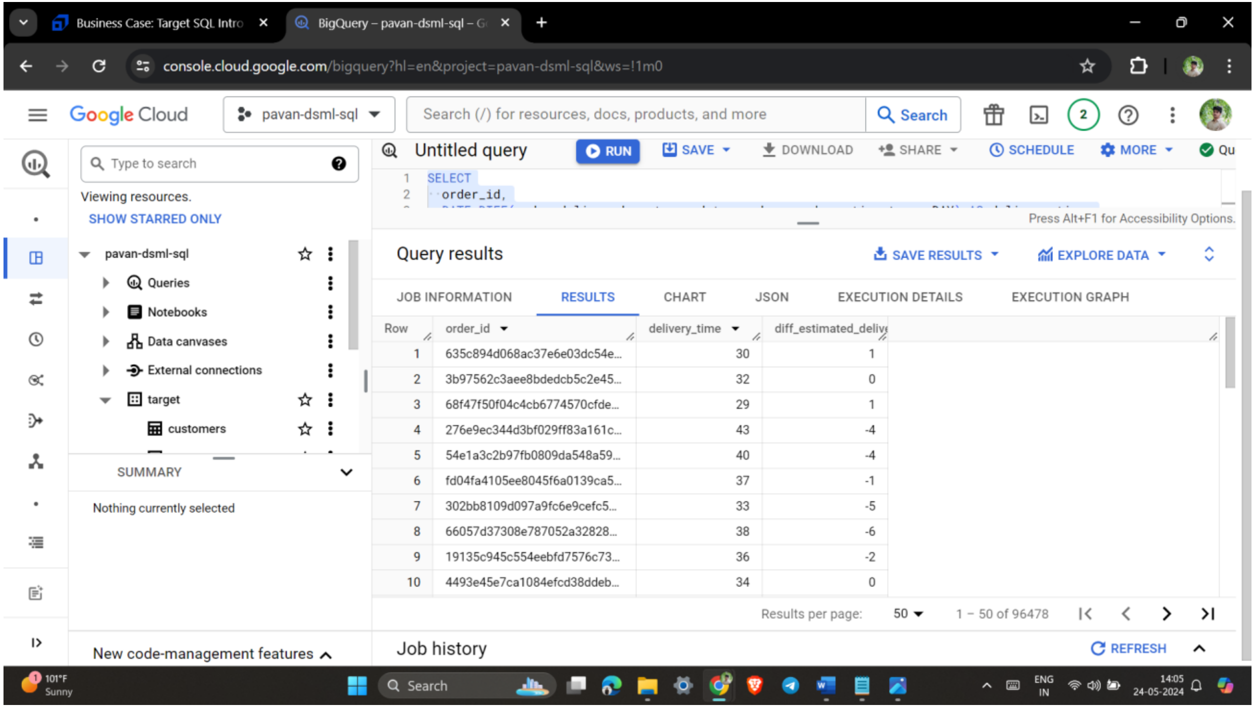
You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

i. time_to_deliver = order_delivered_customer_date - order_purchase_timestamp

ii. diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

QUERY:

```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS delivery_time,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) AS diff_estimated_delivery
FROM
  `target-project.target_brazil.orders`
WHERE
  order_status = 'delivered';
```



```
SELECT
  order_id,
```

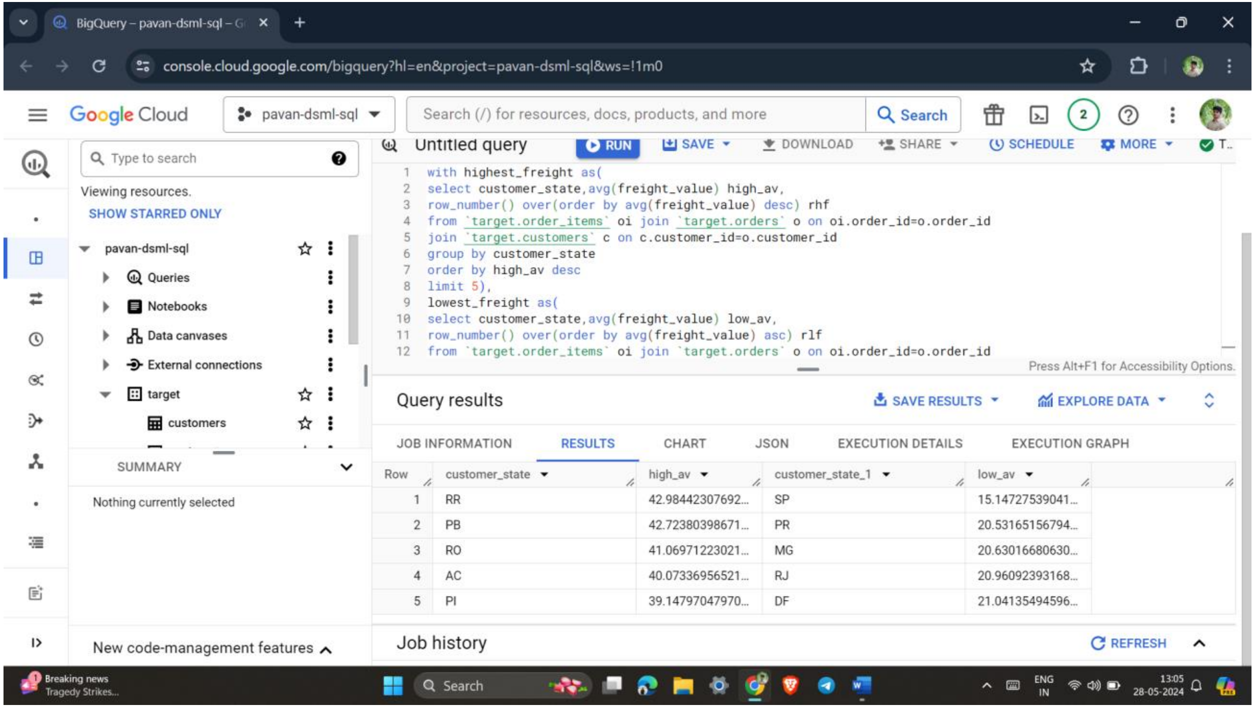
Row	order_id	delivery_time	diff_estimated_delive
1	635c894d068ac37e6e03dc54e...	30	1
2	3b97562c3aee8bdedcb5c2e45...	32	0
3	68f47f50f04c4cb6774570cfde...	29	1
4	276e9ec344d3bf029ff83a161c...	43	-4
5	54e1a3c2b97fb0809da548a59...	40	-4
6	fd04fa4105ee8045f6a0139ca5...	37	-1
7	302bb8109d097a9fc6e9cefc5...	33	-5
8	66057d37308e787052a32828...	38	-6
9	19135c945c554eebfd7576c73...	36	-2
10	4493e45e7ca1084efcd38ddeb...	34	0

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

QUERY:

```
with highest_freight as(
select customer_state,avg(freight_value) high_av,
row_number() over(order by avg(freight_value) desc) rhf
```

from `target.order_items` oi join `target.orders` o on oi.order_id=o.order_id
join `target.customers` c on c.customer_id=o.customer_id
group by customer_state
order by high_av desc
limit 5),
lowest_freight as(
select customer_state,avg(freight_value) low_av,
row_number() over(order by avg(freight_value) asc) rlf
from `target.order_items` oi join `target.orders` o on oi.order_id=o.order_id
join `target.customers` c on c.customer_id=o.customer_id
group by customer_state
order by low_av asc
limit 5)
select hf.customer_state,hf.high_av,lf.customer_state,lf.low_av from
highest_freight hf join lowest_freight lf
on hf.rhf=lf.rlf;



```
with highest_freight as(
select customer_state,avg(freight_value) high_av,
row_number() over(order by avg(freight_value) desc) rhf
from `target.order_items` oi join `target.orders` o on oi.order_id=o.order_id
join `target.customers` c on c.customer_id=o.customer_id
group by customer_state
order by high_av desc
limit 5),
lowest_freight as(
select customer_state,avg(freight_value) low_av,
row_number() over(order by avg(freight_value) asc) rlf
from `target.order_items` oi join `target.orders` o on oi.order_id=o.order_id
```

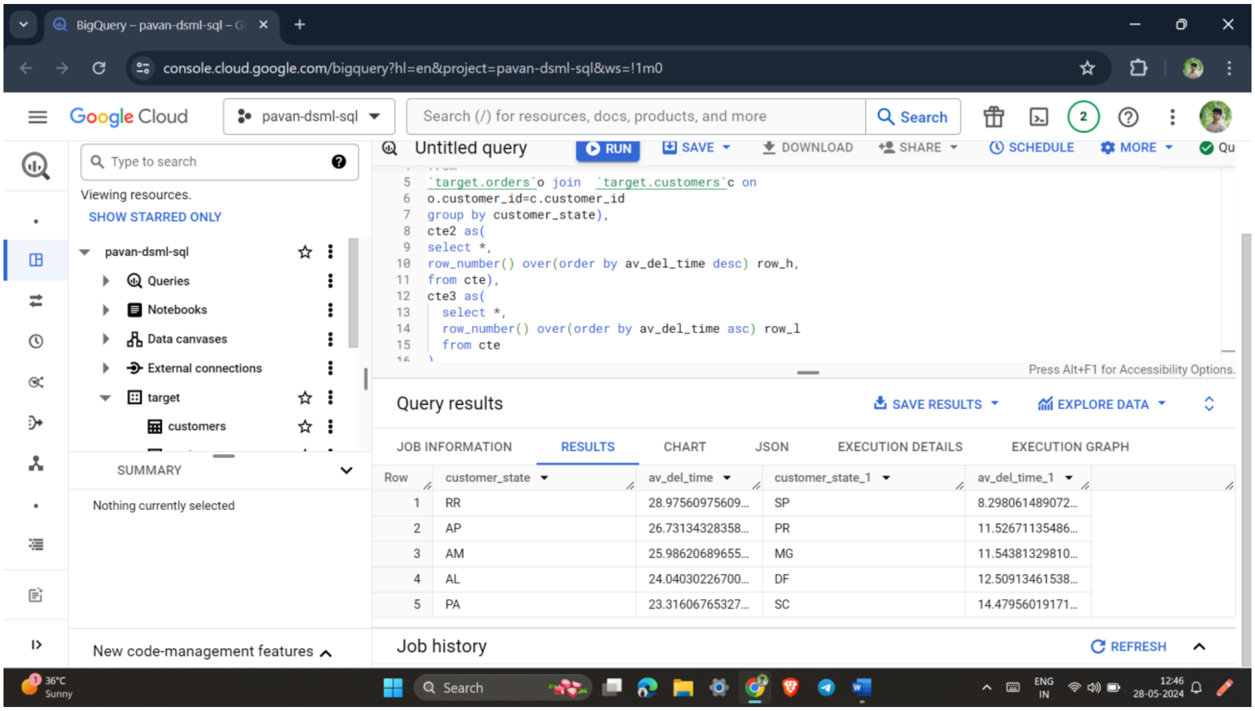
Row	customer_state	high_av	customer_state_1	low_av
1	RR	42.98442307692...	SP	15.14727539041...
2	PB	42.72380398671...	PR	20.53165156794...
3	RO	41.06971223021...	MG	20.63016680630...
4	AC	40.07336956521...	RJ	20.96092393168...
5	PI	39.14797047970...	DF	21.04135494596...

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

QUERY:

with cte as(select
customer_state,

```
avg(datetime_diff(order_delivered_customer_date,order_purchase_timesta
mp,day ) ) av_del_time,
from
`target.orders`o join  `target.customers`c on
o.customer_id=c.customer_id
group by customer_state),
cte2 as(
select *,
row_number() over(order by av_del_time desc) row_h,
from cte),
cte3 as(
  select *,
  row_number() over(order by av_del_time asc) row_l
  from cte
)
select
cte2.customer_state,cte2.av_del_time,cte3.customer_state,cte3.av_del_time
from
cte2 join cte3
on cte2.row_h=cte3.row_l
limit 5;
```



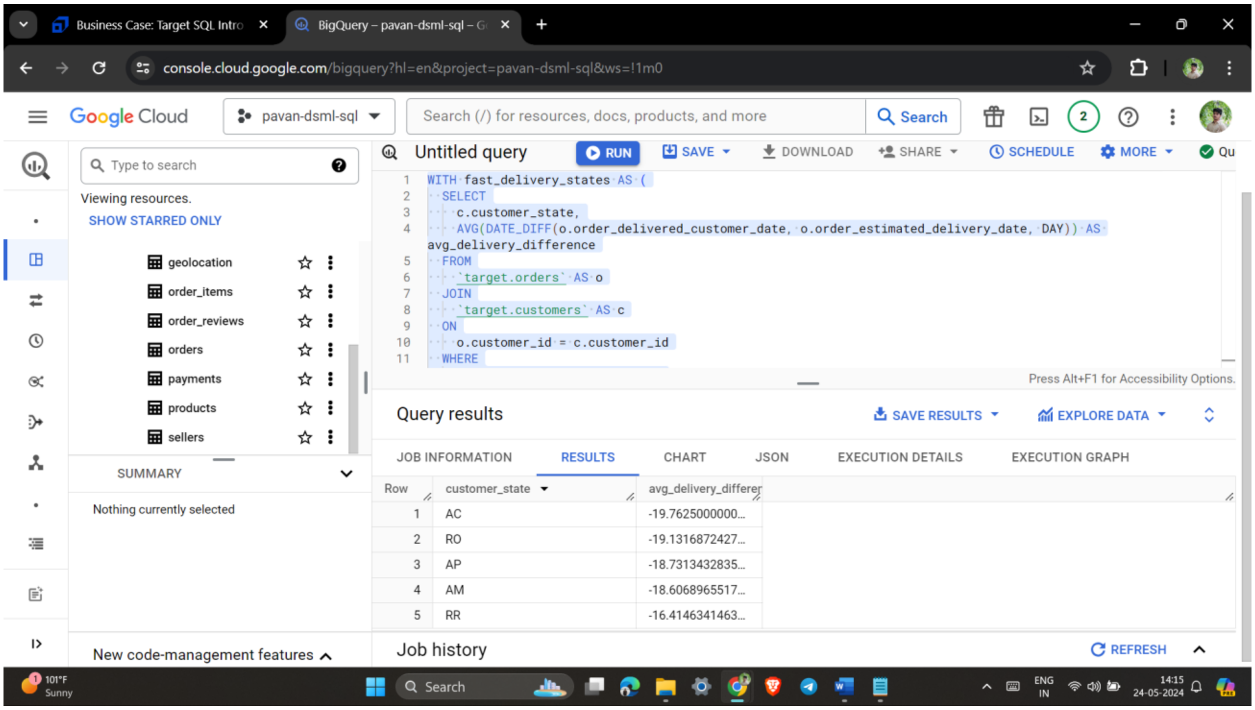
```
`target.orders`o join  `target.customers`c on
o.customer_id=c.customer_id
group by customer_state),
cte2 as(
select *,
row_number() over(order by av_del_time desc) row_h,
from cte),
cte3 as(
  select *,
  row_number() over(order by av_del_time asc) row_l
  from cte
```

Row	customer_state	av_del_time	customer_state_1	av_del_time_1
1	RR	28.97560975609...	SP	8.298061489072...
2	AP	26.73134328358...	PR	11.52671135486...
3	AM	25.98620689655...	MG	11.54381329810...
4	AL	24.04030226700...	DF	12.50913461538...
5	PA	23.31606765327...	SC	14.47956019171...

D. 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.

QUERY:

```
WITH fast_delivery_states AS (
  SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)) AS avg_delivery_difference
  FROM
    `target.orders` AS o
  JOIN
    `target.customers` AS c
  ON
    o.customer_id = c.customer_id
  WHERE
    o.order_status = 'delivered'
  GROUP BY
    c.customer_state
)
SELECT
  customer_state,
  avg_delivery_difference
FROM
  fast_delivery_states
ORDER BY
  avg_delivery_difference ASC
LIMIT
  5;
```



```
WITH fast_delivery_states AS (
  SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS 
avg_delivery_difference
  FROM
    `target.orders` AS o
  JOIN
    `target.customers` AS c
  ON
    o.customer_id = c.customer_id
  WHERE
```

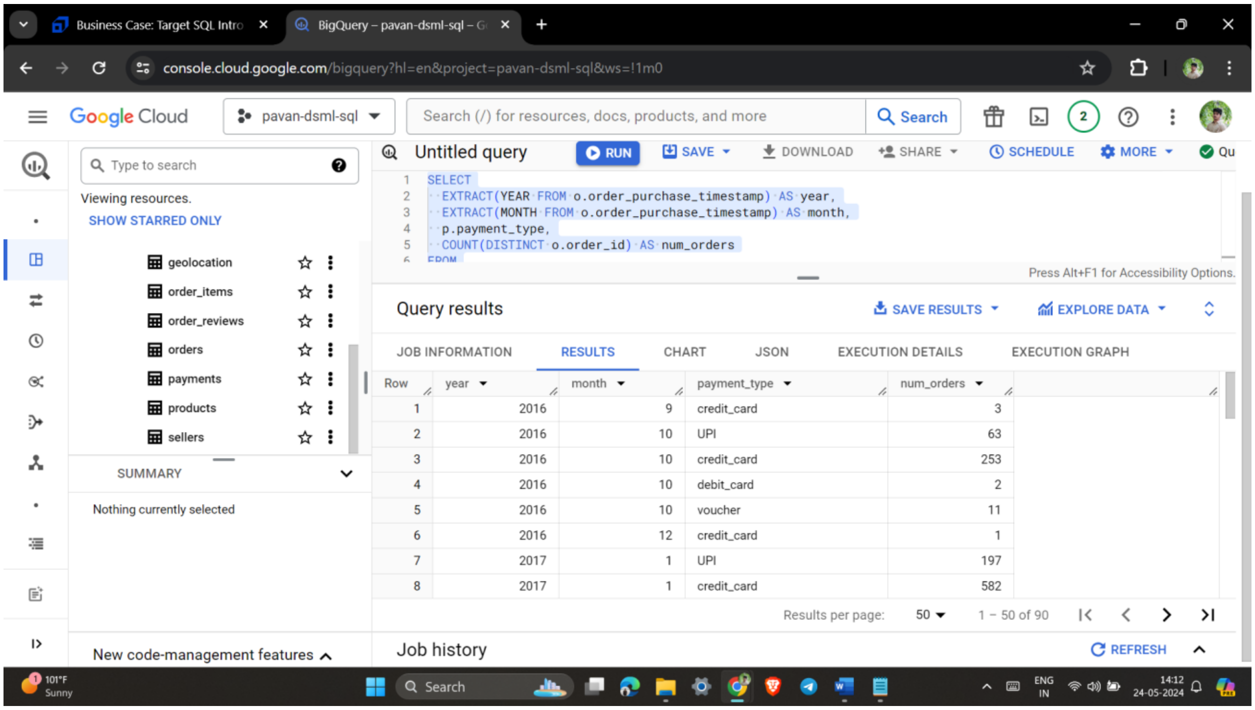
Row	customer_state	avg_delivery_differen
1	AC	-19.7625000000...
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6068965517...
5	RR	-16.4146341463...

6. Analysis based on the payments:

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

QUERY:

```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS num_orders
FROM
  `target.orders` AS o
INNER JOIN
  `target.payments` AS p
ON
  o.order_id = p.order_id
GROUP BY
  year, month, payment_type
ORDER BY
  year, month, payment_type;
```



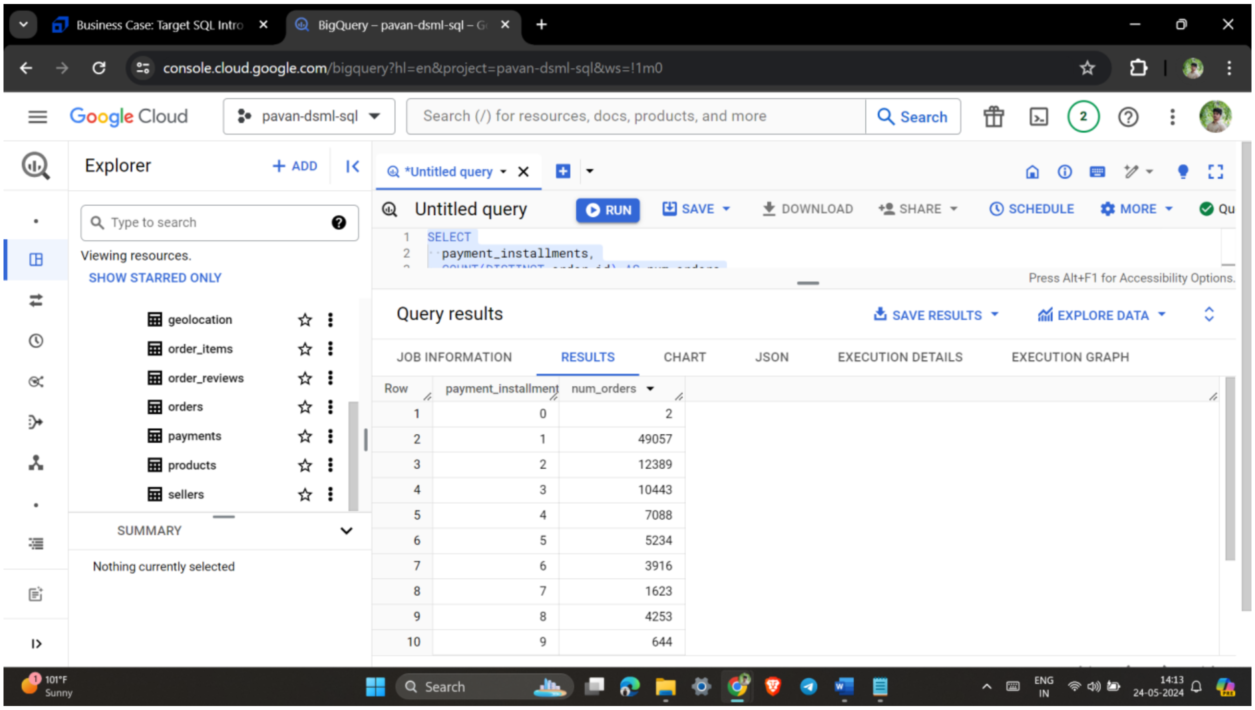
```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS num_orders
FROM
```

Row	year	month	payment_type	num_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	253
4	2016	10	debit_card	2
5	2016	10	voucher	11
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	582

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

QUERY:

```
SELECT
  payment_installments,
  COUNT(DISTINCT order_id) AS num_orders
FROM
  `target.payments`
WHERE
  payment_value > 0
GROUP BY
  payment_installments
ORDER BY
  payment_installments;
```



```
SELECT
  payment_installments,
  COUNT(DISTINCT order_id) AS num_orders
```

Row	payment_installment	num_orders
1	0	2
2	1	49057
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644