

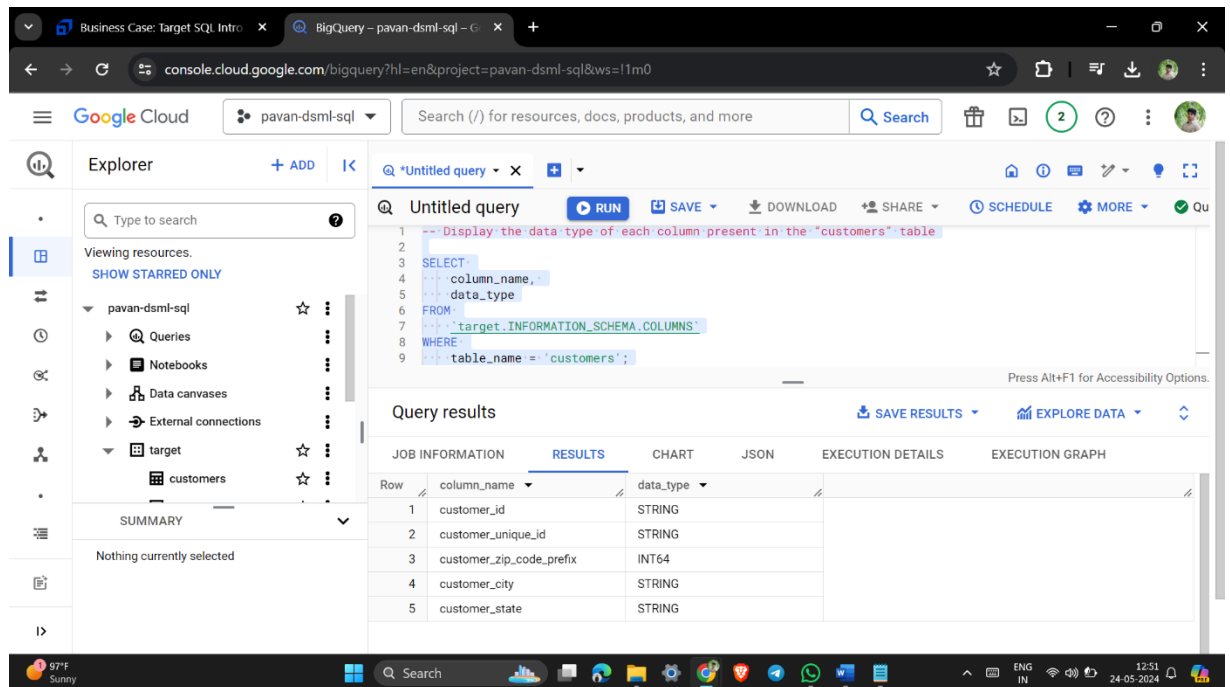
Problem Statement

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';
```



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`;
```

The screenshot shows the Google Cloud BigQuery console interface. The left sidebar displays the Explorer view with the project 'pavan-dsml-sql' and the dataset 'target' expanded, showing the 'customers' table. The main panel shows an 'Untitled query' editor with the following SQL code:

```
1 -- Get the time range between which the orders were placed.
2
3 SELECT
4   MIN(order_purchase_timestamp) AS first_order_date,
5   MAX(order_purchase_timestamp) AS last_order_date
6 FROM
7   `target.orders`;
```

The query has been executed, and the results are displayed in a table. The table has two columns: 'first_order_date' and 'last_order_date'. The results show a single row with the following values:

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`;
```

The screenshot shows the Google Cloud BigQuery console interface. On the left, the Explorer pane displays the project structure for 'pavan-dsml-sql', including 'Queries', 'Notebooks', 'Data canvases', 'External connections', and a dataset named 'target' with a table 'customers'. The main editor shows an 'Untitled query' with the following SQL code:

```

1  -- Count the number of unique cities & states where orders were placed by the customers
2  SELECT
3  ... COUNT(DISTINCT customer_city) AS unique_cities,
4  ... COUNT(DISTINCT customer_state) AS unique_states
5  FROM
6  ... `target.customers`;
7

```

Below the query editor, the 'Query results' section is visible, showing a table with the following data:

Row	unique_cities	unique_states
1	4119	27

The bottom of the image shows a Windows taskbar with the date and time as 12:57 on 24-05-2024.

2. In-depth Exploration:

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

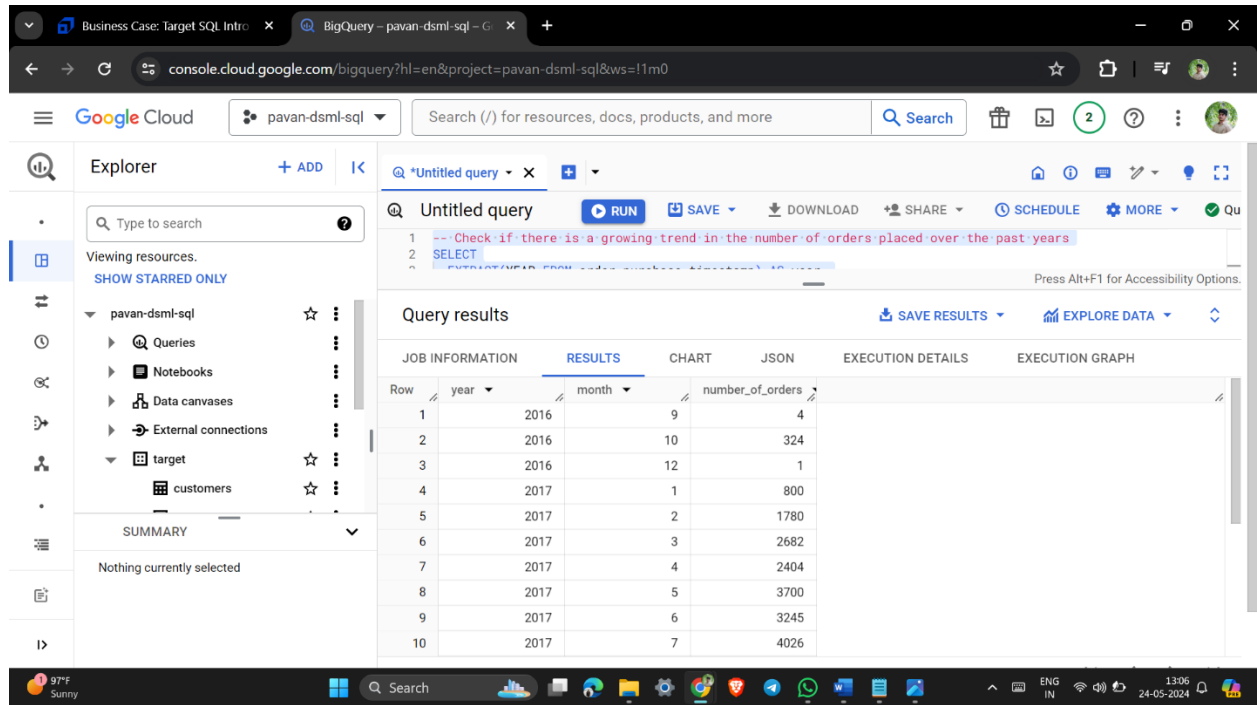
`target.orders`

GROUP BY

year, month

ORDER BY

year, month;



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

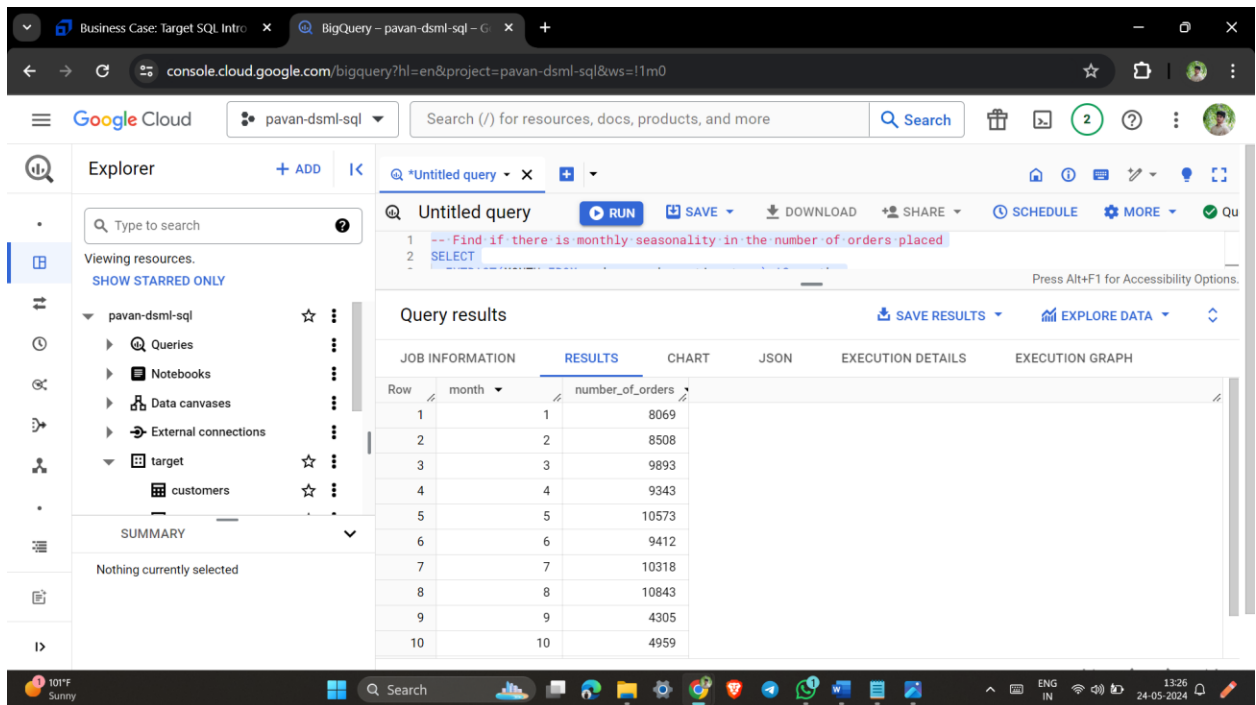
`target.orders`

GROUP BY

month

ORDER BY

month;



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;

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the following SQL code:

```

4 WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
5 WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
6 WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
7 WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
8 END AS time_of_day,
9 COUNT(order_id) AS number_of_orders
10 FROM
11 `target.orders`
12 GROUP BY
13 time_of_day
14 ORDER BY

```

The query results are displayed in a table with the following data:

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.

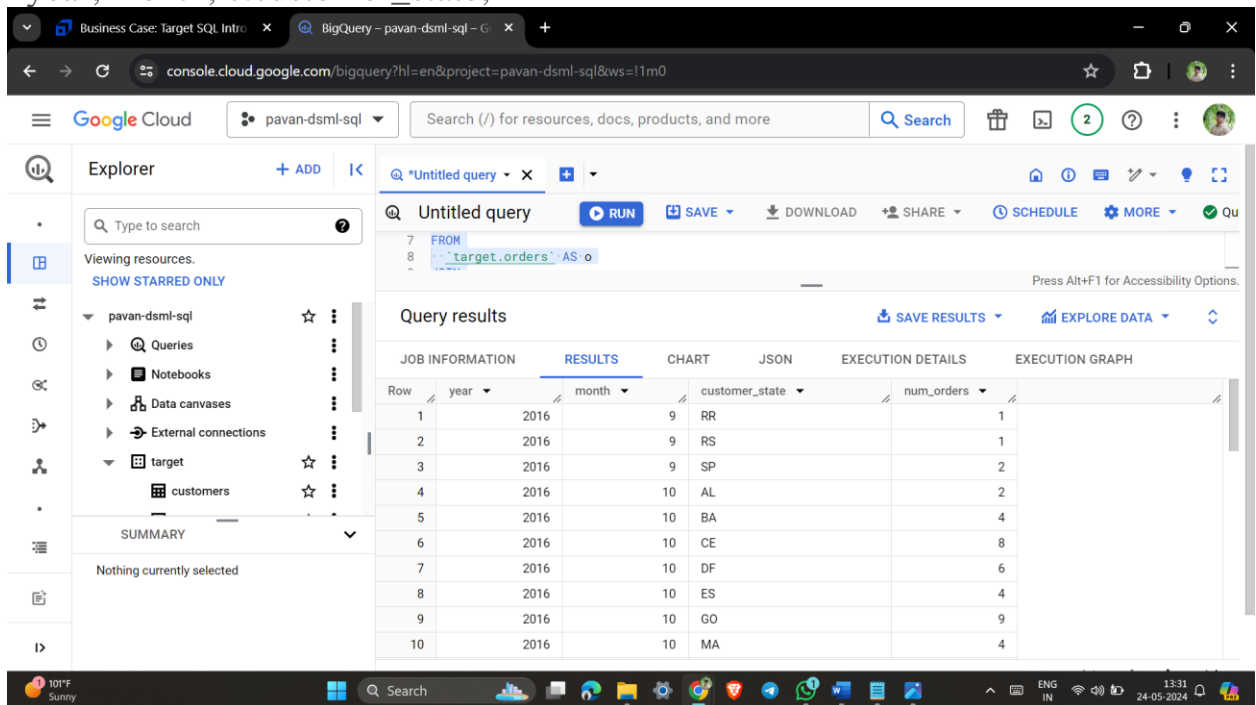
QUERY:

```

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;



The screenshot shows the Google Cloud BigQuery console interface. On the left, the Explorer pane displays the project structure for 'pavan-dsml-sql', including 'Queries', 'Notebooks', 'Data canvases', 'External connections', and a dataset named 'target' containing a table 'customers'. The main editor area shows an 'Untitled query' with the following SQL code:

```
7 FROM
8 `target.orders` AS o
```

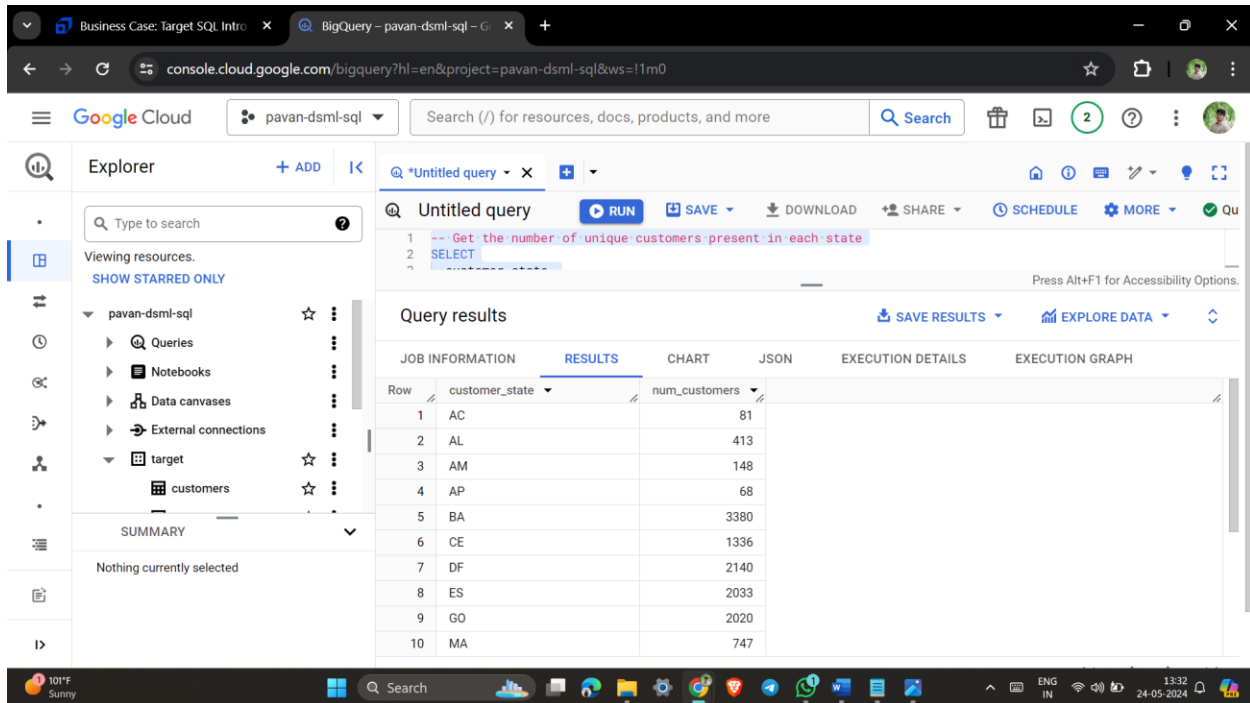
The 'Query results' pane displays a table with the following data:

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;
```



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;

```

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the following SQL query:

```

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

```

The query results are displayed in a table with 8 rows and 2 columns: 'Row' and 'percent_increase'.

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;

```

The screenshot shows the Google Cloud BigQuery console. The query editor displays the following SQL query:

```

-- 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
FROM
  `target.orders` oi
  JOIN `target.customers` c ON oi.customer_id = c.customer_id

```

The query results are displayed in a table with the following columns: Row, customer_state, total_order_price, and avg_order_price. The results are as follows:

Row	customer_state	total_order_price	avg_order_price
1	AC	15982.949999999...	173.7277173913...
2	AL	80314.809999999...	180.8892117117...
3	AM	22356.840000000...	135.49599999999...
4	AP	13474.299999999...	164.3207317073...
5	BA	511349.990000000...	134.6012082126...
6	CE	227254.709999999...	153.7582611637...
7	DF	302603.939999999...	125.7705486284...
8	ES	275037.309999999...	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;

```

The screenshot shows the Google Cloud BigQuery console interface. On the left, there's a sidebar with navigation options like 'Queries', 'Notebooks', 'Data canvases', and 'External connections'. The main area displays an 'Untitled query' with the following SQL code:

```

4 SUM(oi.freight_value) AS total_freight_value,
5 AVG(oi.freight_value) AS avg_freight_value
6 FROM
7 `target.order_items` AS oi
8

```

Below the query editor, the 'Query results' section is visible, showing a table with the following data:

Row	customer_state	total_freight_value	avg_freight_value
1	AC	3686.750000000...	40.07336956521...
2	AL	15914.589999999...	35.84367117117...
3	AM	5478.890000000...	33.20539393939...
4	AP	2788.500000000...	34.00609756097...
5	BA	100156.6799999...	26.36395893656...
6	CE	48351.589999999...	32.71420162381...
7	DF	50625.499999999...	21.04135494596...
8	ES	49764.599999999...	22.05877659574...
9	GO	53114.979999999...	22.76681525932...

The bottom of the console shows the 'Job history' section and a status bar with system information like '101°F Sunny' and '24-05-2024'.

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:

$$\text{i. time_to_deliver} = \text{order_delivered_customer_date} - \text{order_purchase_timestamp}$$

ii. $\text{diff_estimated_delivery} = \text{order_delivered_customer_date} - \text{order_estimated_delivery_date}$

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';
```

The screenshot shows the Google Cloud BigQuery console interface. On the left, there's a sidebar with navigation options like 'Queries', 'Notebooks', 'Data canvases', 'External connections', 'target', and 'customers'. The main area displays a query titled 'Untitled query' with the following SQL code:

```
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
```

Below the query, the 'Query results' section shows a table with 10 rows. The columns are 'order_id', 'delivery_time', and 'diff_estimated_delivery'. The data is as follows:

Row	order_id	delivery_time	diff_estimated_delivery
1	635c894d068ac37e6e03dc54e...	30	1
2	3b97562c3aee8bdecb5c2e45...	32	0
3	68f47f50f04c4cb6774570cfde...	29	1
4	276e9ec344d3bf029ff83a161c...	43	-4
5	54e1a3c2b97fb0809da548a59...	40	-4
6	fd04fa4105ee8045f6a0139ca5...	37	-1
7	302bb8109d097a9fc6e9cfc5...	33	-5
8	66057d37308e787052a32828...	38	-6
9	19135c945c554eebfd7576c73...	36	-2
10	4493e45e7ca1084efcd38ddeb...	34	0

At the bottom, there's a 'Job history' section with a 'REFRESH' button. The status bar at the very bottom shows the system time as 14:05 on 24-05-2024.

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;

```

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'pavan-dsml-sql' project is selected, showing a list of resources including 'queries', 'notebooks', 'data canvases', 'external connections', and a 'target' dataset with a 'customers' table. The main area displays an 'Untitled query' with the following SQL code:

```

1 with highest_freight as(
2 select customer_state,avg(freight_value) high_av,
3 row_number() over(order by avg(freight_value) desc) rhf
4 from `target.order_items` oi join `target.orders` o on oi.order_id=o.order_id
5 join `target.customers` c on c.customer_id=o.customer_id
6 group by customer_state
7 order by high_av desc
8 limit 5),
9 lowest_freight as(
10 select customer_state,avg(freight_value) low_av,
11 row_number() over(order by avg(freight_value) asc) rlf
12 from `target.order_items` oi join `target.orders` o on oi.order_id=o.order_id

```

Below the query, the 'Query results' section is visible, showing a table with 5 rows of data. The table has columns for 'customer_state', 'high_av', 'customer_state_1', and 'low_av'. The results are as follows:

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...

The bottom of the console shows the 'Job history' section with a 'REFRESH' button. The Windows taskbar at the very bottom shows the date as 28-05-2024 and the time as 13:05.

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;

```

The screenshot shows the Google Cloud BigQuery console interface. The left sidebar displays the project structure for 'pavan-dsml-sql', including 'Queries', 'Notebooks', 'Data canvases', 'External connections', and a 'target' dataset with a 'customers' table. The main area shows an 'Untitled query' with the following SQL code:

```

5 `target.orders`o join `target.customers`c on
6 o.customer_id=c.customer_id
7 group by customer_state),
8 cte2 as(
9 select *,
10 row_number() over(order by av_del_time desc) row_h,
11 from cte),
12 cte3 as(
13 select *,
14 row_number() over(order by av_del_time asc) row_l
15 from cte

```

Below the query editor, the 'Query results' section is visible, showing a table with 5 rows of data. The table has columns for 'customer_state', 'av_del_time', 'customer_state_1', and 'av_del_time_1'. The 'Job history' section at the bottom shows a single job with a status of 'SUCCEEDED'.

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;
```

The screenshot shows the Google Cloud BigQuery console. On the left, there's a sidebar with a search bar and a list of resources: geolocation, order_items, order_reviews, orders, payments, products, and sellers. The main area displays an 'Untitled query' with the following SQL code:

```

1 WITH fast_delivery_states AS (
2   SELECT
3     c.customer_state,
4     AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS
5     avg_delivery_difference
6   FROM
7     `target.orders` AS o
8   JOIN
9     `target.customers` AS c
10    ON
11     o.customer_id = c.customer_id
12   WHERE

```

Below the query editor, the 'Query results' section is visible, showing a table with 5 rows and 2 columns: 'customer_state' and 'avg_delivery_difference'.

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

At the bottom, there's a 'Job history' section with a 'REFRESH' button.

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;

```


The screenshot shows the Google Cloud BigQuery console interface. On the left, there's a sidebar with a search bar and a list of resources including 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main area displays an 'Untitled query' with the following SQL code:

```

1 SELECT
2   EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
3   EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
4   p.payment_type,
5   COUNT(DISTINCT o.order_id) AS num_orders
6 FROM

```

Below the query editor, the 'Query results' section is visible, showing a table with columns: 'Row', 'year', 'month', 'payment_type', and 'num_orders'. The table contains 8 rows of data. At the bottom, there's a 'Job history' section with a 'REFRESH' button.

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;

```

Business Case: Target SQL Intro xBigQuery - pavan-dsml-sql - G x

console.cloud.google.com/bigquery?hl=en&project=pavan-dsml-sql&wsl=11m0

Google Cloudpavan-dsml-sqlSearch (/) for resources, docs, products, and moreSearch

Explorer+ ADD<

Type to search?

Viewing resources.
SHOW STARRED ONLY

geolocation ☆

order_items ☆

order_reviews ☆

orders ☆

payments ☆

products ☆

sellers ☆

SUMMARY

Nothing currently selected

Untitled query x

SELECT
payment_installments,
COUNT(*) AS num_orders

Query results

JOB INFORMATIONRESULTSCHARTJSONEXECUTION DETAILSEXECUTION GRAPH

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

101°F Sunny

Search

ENG IN14:1324-05-2024