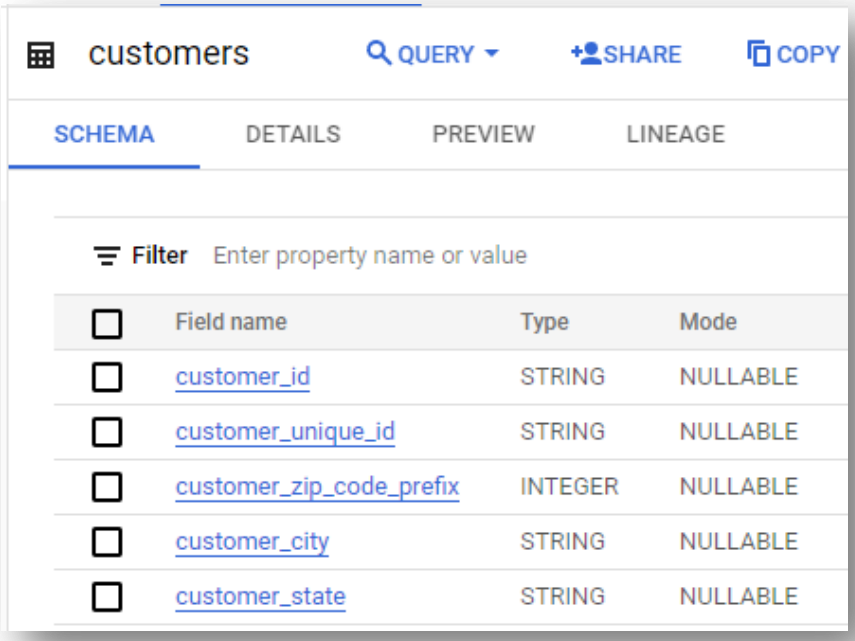
SQL PROJECT – (Target's Dataset)

~ Yogesh More

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

A. Data type of columns in a table.



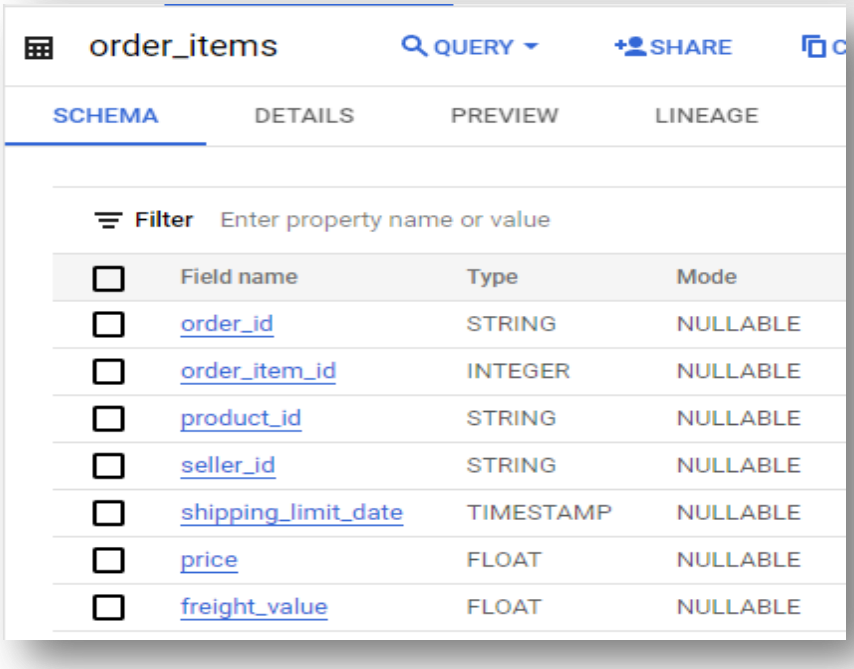
customers

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

Field name	Type	Mode
customer_id	STRING	NULLABLE
customer_unique_id	STRING	NULLABLE
customer_zip_code_prefix	INTEGER	NULLABLE
customer_city	STRING	NULLABLE
customer_state	STRING	NULLABLE



order_items

QUERY SHARE

SCHEMA DETAILS PREVIEW LINEAGE

Filter Enter property name or value

Field name	Type	Mode
order_id	STRING	NULLABLE
order_item_id	INTEGER	NULLABLE
product_id	STRING	NULLABLE
seller_id	STRING	NULLABLE
shipping_limit_date	TIMESTAMP	NULLABLE
price	FLOAT	NULLABLE
freight_value	FLOAT	NULLABLE

Observation:

In the given dataset we have many columns having different data types. Like: string, integer, float, date-time, location, etc.

B. Time period for which the data is given.

```
SELECT
  MIN(order_purchase_timestamp) AS start_date,
  MAX(order_purchase_timestamp) AS end_date
FROM
  `scaler-dsml-sql-2412.target_sql.orders`
LIMIT
  10
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

C. Cities and States of customers ordered during the given period.

```
SELECT
DISTINCT c.customer_city,
 c.customer_state
FROM
  `target_sql.customers` c
JOIN
  `target_sql.orders` o
ON
  c.customer_id = o.customer_id
LIMIT
  10
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_city	customer_state
1	rio de janeiro	RJ
2	sao leopoldo	RS
3	general salgado	SP
4	brasilia	DF
5	paranavai	PR
6	cuiaba	MT
7	sao luis	MA
8	maceio	AL
9	hortolandia	SP
10	varzea grande	MT

2) In-depth Exploration:

A. Is there a growing trend in e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

```
SELECT
  Month,
  COUNT(product_id) AS month_wise_product_purchased
FROM (
  SELECT
    *,
    FORMAT_DATETIME('%B', o.order_purchase_timestamp) AS Month
  FROM
    `target_sql.orders` o
  JOIN
    `target_sql.order_items` oi
  ON
    o.order_id = oi.order_id
  ORDER BY
    o.order_purchase_timestamp )
GROUP BY
  Month
ORDER BY
  Month ASC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

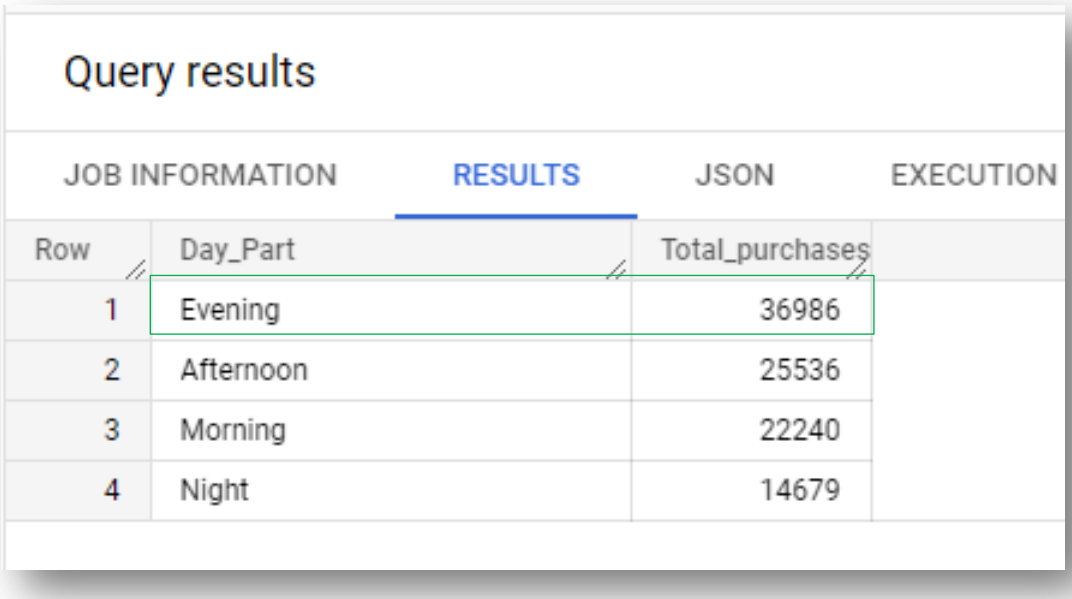
Row	Month	month_wise_pro	
1	April	10659	
2	August	12158	Max sales
3	December	6309	
4	February	9623	
5	January	9163	
6	July	11611	
7	June	10661	
8	March	11217	
9	May	12061	
10	November	8665	
11	October	5685	
12	September	4838	Min sales

Conclusion:

From the above results, we can conclude that sales in Brazil are at their peak during **August** and the least during **September**.

B. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon, or Night)?

```
SELECT
  Day_Part,
  COUNT(*) AS Total_purchases_made
FROM (
  SELECT
    purchase_hour,
    CASE
      WHEN purchase_hour >= 6 AND purchase_hour < 12 THEN "Morning"
      WHEN purchase_hour >= 12
    AND purchase_hour < 16 THEN "Afternoon"
      WHEN purchase_hour >= 16 AND purchase_hour < 22 THEN "Evening"
    ELSE
    "Night"
  END
    AS Day_Part
  FROM (
    SELECT
      *,
      EXTRACT(hour
      FROM
        order_purchase_timestamp) AS purchase_hour
    FROM
      `target_sql.orders` ) )
GROUP BY
  Day_Part
ORDER BY
  Total_purchases_made DESC
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	Day_Part	Total_purchases
1	Evening	36986
2	Afternoon	25536
3	Morning	22240
4	Night	14679

Conclusion:

Here we can say that Brazilian mostly prefer to buy during the **Evening time between 4 pm to 10 pm.**

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

A. Get month-on-month orders by state.

```
SELECT
  DISTINCT customer_state,
  Month,
  COUNT(order_id) AS total_purchase
FROM (
  SELECT
    *,
    FORMAT_DATETIME("%B", o.order_purchase_timestamp) AS Month
  FROM
    `target_sql.customers` c
  JOIN
    `target_sql.orders` o
  ON
    c.customer_id = o.customer_id ) x
GROUP BY
  customer_state,
  Month
ORDER BY
  total_purchase DESC
LIMIT
  10
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECU

Row	customer_state	Month	total_purchase
1	SP	August	4982
2	SP	May	4632
3	SP	July	4381
4	SP	June	4104
5	SP	March	4047
6	SP	April	3967
7	SP	February	3357
8	SP	January	3351
9	SP	November	3012
10	SP	December	2357

Conclusion:

SP is the state which has recorded the overall highest purchases, especially during the month of **August.**

B. Distribution of customers across the states in Brazil.

```
SELECT
  DISTINCT customer_state,
  COUNT(customer_id) AS total_customers
FROM
  `target_sql.customers`
GROUP BY
  customer_state
ORDER BY
  total_customers DESC
LIMIT
  10
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_state	total_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

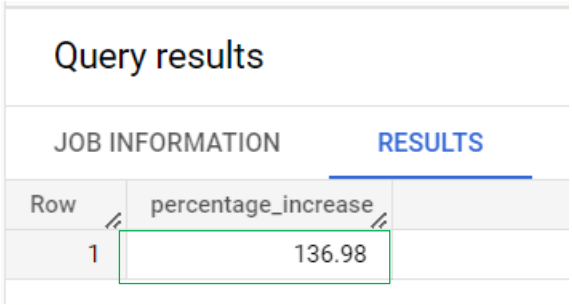
Conclusion:

There are a total of 99441 customers within 27 states in Brazil out of which **41746 customers from SP state** have recorded a **higher count of purchases** within the time period of **2 years.**

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

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

```
WITH
  Y AS (
  SELECT
    EXTRACT(year
    FROM
      o.order_purchase_timestamp) AS year,
    SUM(p.payment_value) AS total_payment_value
  FROM
    `target_sql.payments` AS p
  JOIN
    `target_sql.orders` AS o
  ON
    o.order_id=p.order_id
  WHERE
    EXTRACT(month
    FROM
      o.order_purchase_timestamp) BETWEEN 1
    AND 8
  GROUP BY
    1
  HAVING
    year IN (2017,
      2018))
SELECT
  ROUND(((Y1.total_payment_value / Y2.total_payment_value ) -1)*100,2) AS percentage_increase
FROM
  Y AS Y1,
  Y AS Y2
WHERE
  Y1.year=2018
  AND Y2.year=2017
```



Conclusion:

In the results, we can see that the cost of orders increased by **136.98 %** from 2017 to 2018 between January to August.

B. Mean & Sum of price and freight value by a customer state.

```
SELECT
  customer_state,
  ROUND(AVG(price),2) AS Avg_Price,
  ROUND(SUM(price),2) AS Total_Price,
  ROUND(AVG(freight_value),2) AS Avg_Freight_Value,
  ROUND(SUM(freight_value),2) AS Total_Freight_Value
FROM
  `target_sql.customers` c
JOIN
  `target_sql.orders` o
ON
  c.customer_id=o.customer_id
JOIN
  `target_sql.order_items` oi
ON
  o.order_id=oi.order_id
GROUP BY
  customer_state
LIMIT
  10
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	Avg_Price	Total_Price	Avg_Freight_Value	Total_Freight_Value
1	RN	156.97	83034.98	35.65	18860.1
2	CE	153.76	227254.71	32.71	48351.59
3	RS	120.34	750304.02	21.74	135522.74
4	SC	124.65	520553.34	21.47	89660.26
5	SP	109.65	5202955.05	15.15	718723.07
6	MG	120.75	1585308.03	20.63	270853.46
7	BA	134.6	511349.99	26.36	100156.68
8	RJ	125.12	1824092.67	20.96	305589.31
9	GO	126.27	294591.95	22.77	53114.98
10	MA	145.2	119648.22	38.26	31523.77

Conclusion:

Using the above query we can find the sum and mean (avg) of the column price and freight based on a customer state.

5) Analysis of sales, freight, and delivery time.

A. Calculate days between purchasing, delivering, and estimated delivery

```
SELECT
  DISTINCT DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) AS delivery_diff,
  DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) AS estimated_delivery_diff
FROM
  `target_sql.orders`
LIMIT
  10
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	delivery_diff	estimated_delivery_diff
1	null	16
2	null	33
3	null	36
4	null	25
5	null	24
6	null	27
7	null	31
8	null	15
9	null	13
10	null	28

Conclusion:

Easily we can conclude that there is no day difference between the purchase date to the delivery date because the order was delivered within the estimated time of delivery. Hence, the second result column shows the days difference.

B. Find time_to_delivery & diff_estimated_delivery. The formula for the same is given below:

i. time_to_delivery=order_purchase_timestamp-order_delivered_customer_date

```
SELECT
  DISTINCT DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date,day)
  AS time_to_delivery
FROM
  `target_sql.orders`
LIMIT
  10
```

Query results

JOB INFORMATION | RESULTS

Row	time_to_delivery
1	null
2	-7
3	-30
4	-10
5	-35
6	-23
7	-12
8	-1
9	-6
10	-21

ii. diff_estimated_delivery=order_estimated_delivery_date-order_delivered_customer_date

```
SELECT
  DISTINCT DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,day)
  AS diff_estimted_delivery
FROM
  `target_sql.orders`
LIMIT
  10
```

Query results

JOB INFORMATION | RESULTS

Row	diff_estimted_delivery
1	null
2	45
3	-12
4	28
5	44
6	41
7	16
8	9
9	-5
10	12

C. Group data by state, take the mean of freight_value, time_to_delivery, and diff_estimated_delivery.

```
SELECT
  distinct customer_state,
  round(AVG(freight_value),2) AS avg_freight_value,
  round(AVG(DISTINCT DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date,
  day)),2) AS time_to_delivery,
  round(AVG(DISTINCT DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_
  date, day)),2) AS diff_estimted_delivery
FROM
  `target_sql.orders` o
JOIN
  `target_sql.order_items` oi
ON
  o.order_id=oi.order_id
JOIN
  `target_sql.customers` c
ON
  c.customer_id=o.customer_id
GROUP BY
  customer_state
LIMIT
  10
```

Query results

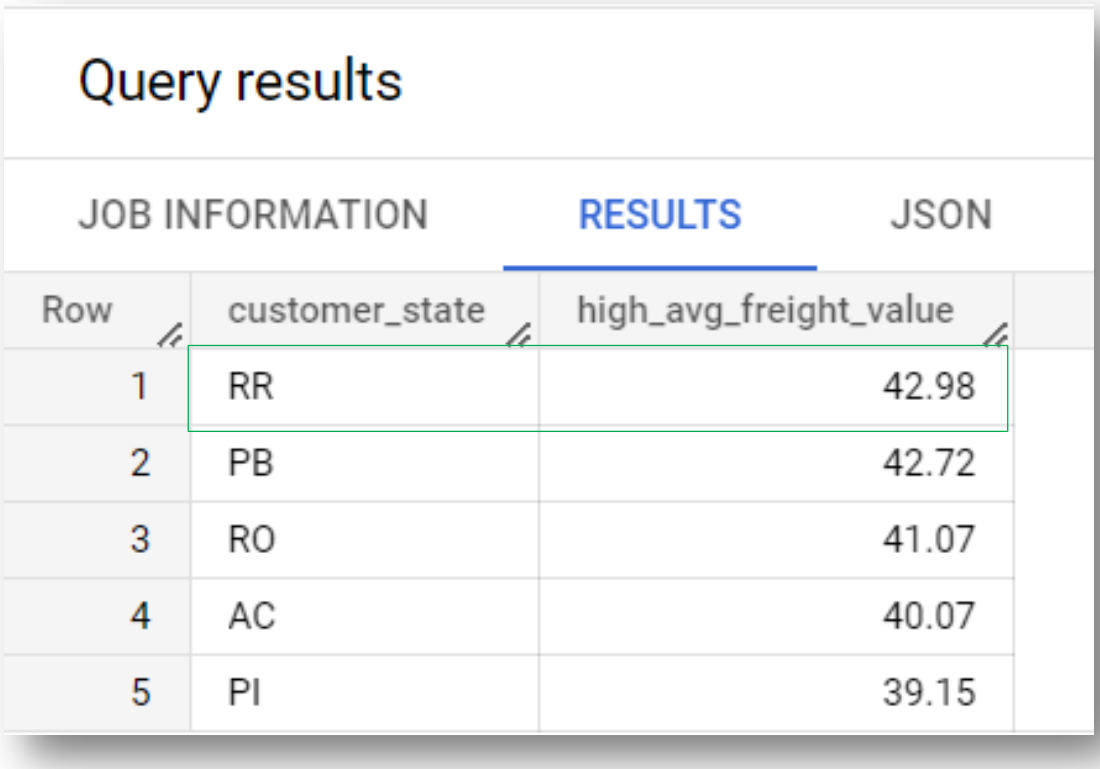
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	customer_state	avg_freight_value	time_to_delivery	diff_estimted_delivery
1	MT	28.17	-30.19	10.43
2	MA	38.26	-33.9	7.3
3	AL	35.84	-30.42	7.36
4	SP	15.15	-54.96	-9.24
5	MG	20.63	-37.67	5.0
6	PE	32.92	-35.81	3.74
7	RJ	20.96	-52.87	-7.25
8	DF	21.04	-27.24	11.04
9	RS	21.74	-37.77	5.76
10	SE	36.65	-36.51	1.88

D. Sort the data to get the following:

E. Top 5 states with highest average freight value - sort in desc limit 5.

```
SELECT
  customer_state,
  ROUND(AVG(freight_value),2) AS high_avg_freight_value,
FROM
  `target_sql.customers` c
JOIN
  `target_sql.orders` o
ON
  c.customer_id=o.customer_id
JOIN
  `target_sql.order_items` oi
ON
  o.order_id=oi.order_id
GROUP BY
  customer_state
ORDER BY
  high_avg_freight_value DESC
LIMIT
  5
```



Query results

JOB INFORMATION | RESULTS | JSON

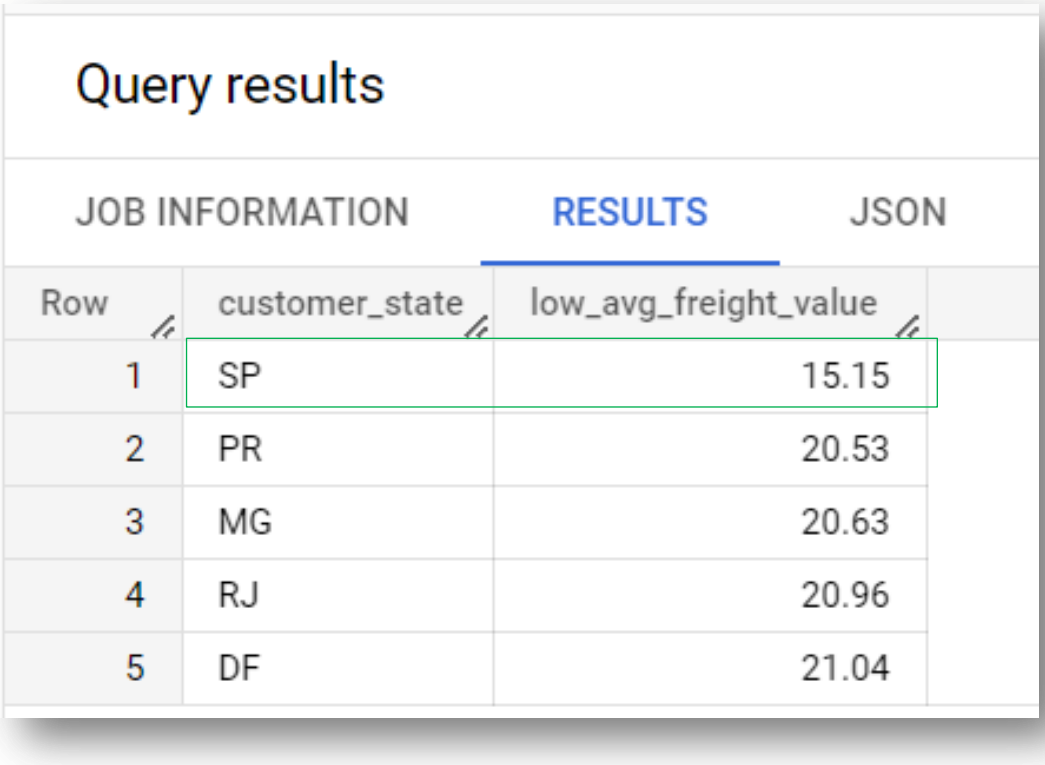
Row	customer_state	high_avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

Conclusion:

We can say that **RR** is the state in Brazil which has **higher average freight charges** i.e. **42.98.**

F. Top 5 states with lowest average freight value - sort in asc limit 5.

```
SELECT
  customer_state,
  ROUND(AVG(freight_value),2) AS low_avg_freight_value,
FROM
  `target_sql.customers` c
JOIN
  `target_sql.orders` o
ON
  c.customer_id=o.customer_id
JOIN
  `target_sql.order_items` oi
ON
  o.order_id=oi.order_id
GROUP BY
  customer_state
ORDER BY
  low_avg_freight_value
LIMIT
  5
```



Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_state	low_avg_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

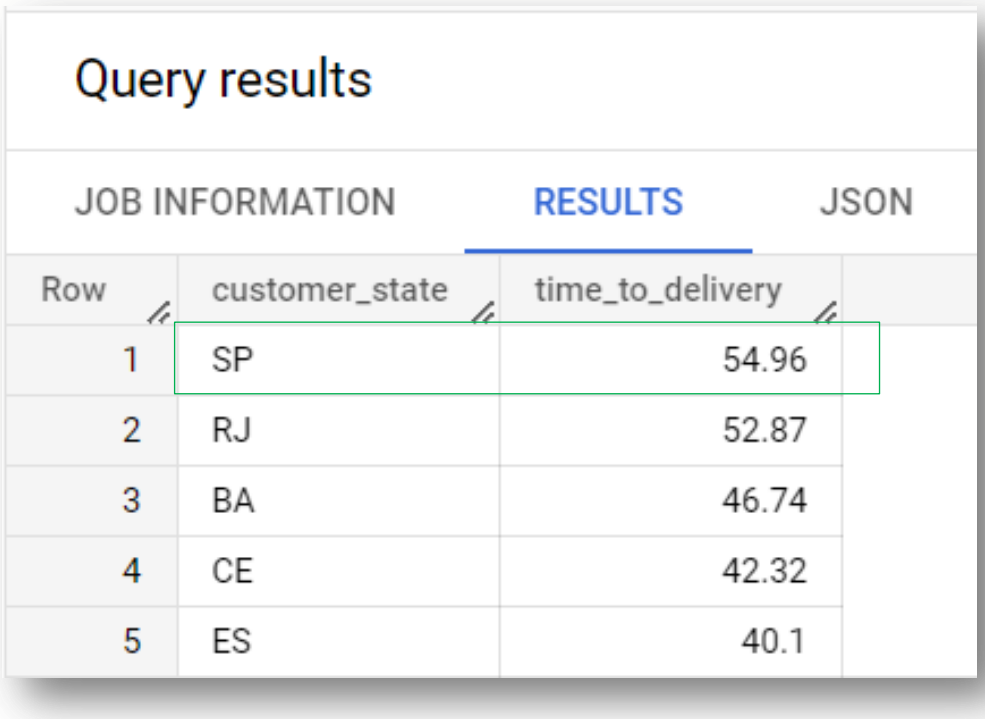
Conclusion:

Here we can see that **SP** is the state in Brazil which has **lowest freight charges** as compared to other states i.e. **15.15**

G. Top 5 states with a highest average time to delivery.

```
SELECT
  customer_state,
  round(AVG(DISTINCT DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
  day)),2) AS time_to_delivery

FROM
  `target_sql.customers` c
JOIN
  `target_sql.orders` o
ON
  c.customer_id=o.customer_id
JOIN
  `target_sql.order_items` oi
ON
  o.order_id=oi.order_id
GROUP BY
  customer_state
ORDER BY
  time_to_delivery desc
LIMIT
  5
```



Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_state	time_to_delivery
1	SP	54.96
2	RJ	52.87
3	BA	46.74
4	CE	42.32
5	ES	40.1

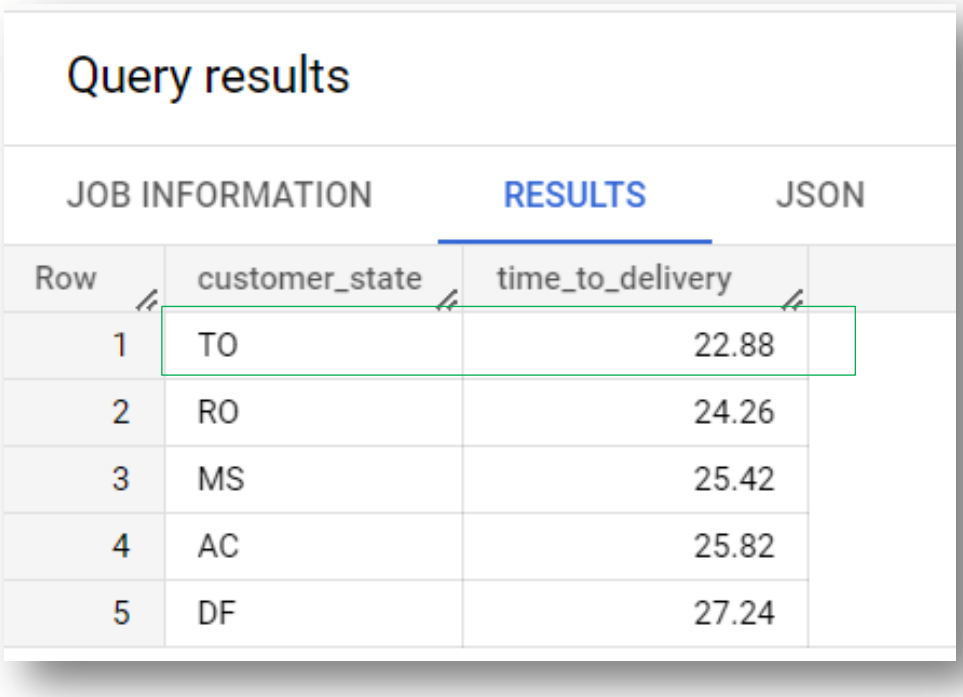
Conclusion:

SP is the state in Brazil which experiences **54.96** on an **average higher delivery days** after placing the order date.

H. Top 5 states with a lowest average time to delivery.

```
SELECT
  customer_state,
  round(AVG(DISTINCT DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
  day)),2) AS time_to_delivery

FROM
  `target_sql.customers` c
JOIN
  `target_sql.orders` o
ON
  c.customer_id=o.customer_id
JOIN
  `target_sql.order_items` oi
ON
  o.order_id=oi.order_id
GROUP BY
  customer_state
ORDER BY
  time_to_delivery
LIMIT
  5
```



Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_state	time_to_delivery
1	TO	22.88
2	RO	24.26
3	MS	25.42
4	AC	25.82
5	DF	27.24

Conclusion:

TO is the state in Brazil which experiences **22.88** on an **average lowest delivery days** after placing the order date.

I. Top 5 states where delivery is really fast/ not so fast compared to the estimated date.

```
SELECT
  customer_state,
  order_delivered_customer_date,
  order_estimated_delivery_date
FROM
  `target_sql.customers` c
JOIN
  `target_sql.orders` o
ON
  c.customer_id=o.customer_id
WHERE
  order_delivered_customer_date < order_estimated_delivery_date
ORDER BY
  order_delivered_customer_date DESC
LIMIT
  5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION

Row	customer_state	order_delivered_customer_date	order_estimated_delivery_date
1	PB	2018-08-31 07:31:51 UTC	2018-09-13 00:00:00 UTC
2	PE	2018-08-31 05:20:37 UTC	2018-09-03 00:00:00 UTC
3	MG	2018-08-31 03:11:38 UTC	2018-09-13 00:00:00 UTC
4	SP	2018-08-31 02:36:23 UTC	2018-09-04 00:00:00 UTC
5	CE	2018-08-31 02:32:36 UTC	2018-09-26 00:00:00 UTC

Conclusion:

The top 5 states where delivery is faster than the estimated delivery. The states are **PB**, **PE**, **MG**, **SP**, and **CE**.

6) Payment type analysis:

A. Month over Month count of orders for different payment types.

```
SELECT
  month,
  payment_type,
  COUNT(*) AS total_orders
FROM (
  SELECT
    *,
    EXTRACT(month
    FROM
      o.order_purchase_timestamp) AS month
  FROM
    `target_sql.orders` o
  JOIN
    `target_sql.payments` p
  ON
    o.order_id=p.order_id )
GROUP BY
  month,
  payment_type
ORDER BY
  month,
  total_orders DESC
LIMIT
  10
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

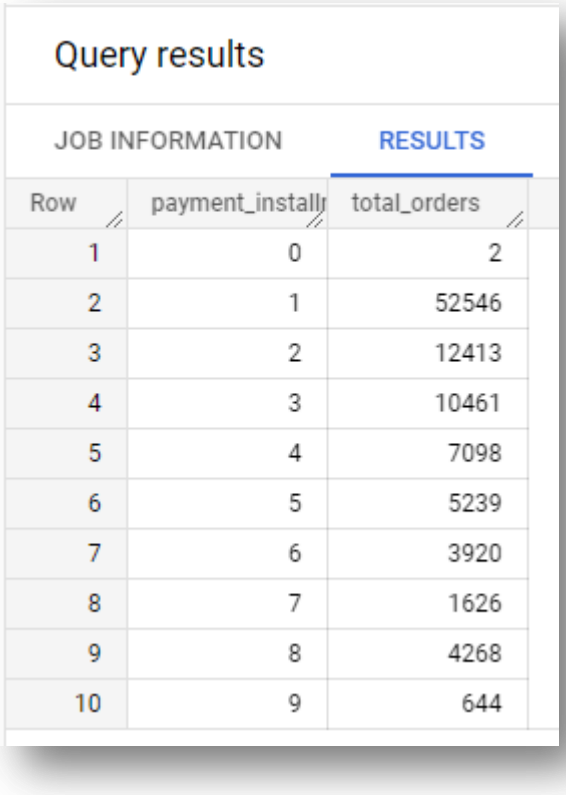
Row	month	payment_type	total_orders
1	1	credit_card	6103
2	1	UPI	1715
3	1	voucher	477
4	1	debit_card	118
5	2	credit_card	6609
6	2	UPI	1723
7	2	voucher	424
8	2	debit_card	82
9	3	credit_card	7707
10	3	UPI	1942

Conclusion:

Here we can observe the month-wise total count of orders based on different payment types. In the first 2 months, most orders were purchased using a credit card, but in the third month, UPI was used by many people to make the payments.

B. Count of orders based on the no. of payment installments.

```
SELECT
  payment_installments,
  COUNT(order_id) AS total_orders
FROM
  `target_sql.payments`
GROUP BY
  payment_installments
ORDER BY
  payment_installments,
  total_orders DESC
LIMIT
  10
```



Query results

JOB INFORMATION | RESULTS

Row	payment_installr	total_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

Conclusion:

After analyzing the results we can say that 2 customers haven't done any payment in installments (might have done full payment in advance or COD), and other customers have chosen installments from 1 to 24 months resp.