TARGET (Brazil Operations)

Exploratory data analysis using SQL

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

Q1. Data type of all columns in the "customers" table.
Ans:

Query (BigQuery):

```
SELECT
  table_name,
  column_name,
  data_type
FROM
  `target_brazil.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name = "customers"
```

Query result:

Query results

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

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

Observation:

There are **5 columns** in the 'customers' table which store the basic information about the customer, mainly their unique id for the store, and their location (city, state and zip code).

Q2. Get the time range between which the orders were placed.
Ans:

Query (BigQuery):

```
SELECT
  MIN(order_purchase_timestamp) AS first_order,
  MAX(order_purchase_timestamp) AS latest_order
FROM
  `target_brazil.orders`
```

Query result:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS

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

Observation:

From the imported dataset, we observe that the **first order was placed on 04th September 2016**, at 21:15 hours UTC to be exact and the **latest order was placed on 17th October 2018** at 17:30 hours UTC to be exact.

Therefore, we are looking at a dataset of orders that were placed between the time-range of 2**016 September to 2018 October.**

Q3. Count the Cities & States of customers who ordered during the given period
Ans:

Query (BigQuery):

```
SELECT
  COUNT(DISTINCT c1.customer_city) AS customer_city,
  COUNT(DISTINCT c1.customer_state) AS customer_state
FROM
  `target_brazil.orders` o1
JOIN
  `target_brazil.customers` c1
ON
  o1.customer_id = c1.customer_id
```

Query result:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	customer_city	customer_state
1	4119	27

Observation:

We observe that, customers from 4119 different cities and 27 different states have ordered during the given time period

---

2. In-depth Exploration:

Q1. Is there a growing trend in the no. of orders placed over the past years?
Ans:

Query (BigQuery):

```
SELECT
  DISTINCT EXTRACT(year FROM order_purchase_timestamp) AS year,
  COUNT(*) OVER (PARTITION BY EXTRACT(year FROM order_purchase_timestamp)) AS order_count
FROM
  `target_brazil.orders`
ORDER BY
  year
```

Query result:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	year	order_count
1	2016	329
2	2017	45101
3	2018	54011

Observation:

Referring to the above query result we can see that there is a growing trend in the no. or orders placed over the past years.

There is a sharp rise between 2016 and 2017, where we see a massive increase in the no. of orders, and between 2017 and 2018 we see a linear growth in the no. of orders placed.

Insight:

Due to the massive increase in popularity of the company, we can see the sharp rise in the no. of orders and thereby making profit in the country.

Recommendation:

Expanding the workforce will definitely cater to more growth and sales.

Q2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
Ans:

Query (BigQuery):

```
select
      DISTINCT EXTRACT(year FROM order_purchase_timestamp) as year,
      FORMAT_DATETIME("%b", order_purchase_timestamp) as month,
      count(*) over (partition by EXTRACT(year FROM order_purchase_timestamp), EXTRACT(month FROM
order_purchase_timestamp)) AS count_order
from `target_brazil.orders`
order by year desc, count_order desc
```

Query result:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	year	month	count_order
1	2018	Jan	7269
2	2018	Mar	7211
3	2018	Apr	6939
4	2018	May	6873
5	2018	Feb	6728
6	2018	Aug	6512
7	2018	Jul	6292
8	2018	Jun	6167
9	2018	Sep	16
10	2018	Oct	4
11	2017	Nov	7544
12	2017	Dec	5673

Observation:

From the above query result, we can see a monthly seasonality in terms of the no. of orders that are placed. There seems to be an increase in the number of orders, roughly during the **months November to March** over the past years.

Insights:

1. The country of Brazil experiences **summer season** during the months of **November to March** as it is in the southern hemisphere of the earth, also some of the most popular events and sales go on during this period for example, **Black Friday sales, Christmas Holidays, New Year Celebration, Carnival Event etc**. due to which we see a rise in the no. of orders and therefore the incoming cash flow.
2. However we saw a sharp decrease in the no. of orders during the month of September and October of 2018.

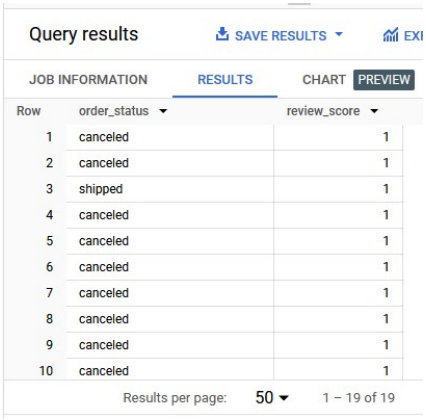
We found out that this could be because of 2 reasons:

a. We tried to analyze the data of the order_status and order_reviews of the orders placed at that time.

Query:

```
select
      o1.order_status,
      or1.review_score
from `target_brazil.orders` o1
join `target_brazil.order_reviews` or1
on o1.order_id = or1.order_id
where extract(year from o1.order_purchase_timestamp) = 2018 and extract(month from
o1.order_purchase_timestamp) between 9 and 10
```

Query result:



Observation:

We found out that almost all of the orders placed, were eventually canceled during that time, alongwith less review scores which is not good for the company's reputation.

b. The country faced some controversial political issues / disturbances, also elections were happening during the period of September and October 2018, as per reports. So that could be the reason for less sales.

Recommendation:

1. As we see a monthly seasonality/rise in orders during the months of November to March, it is recommended to keep the stocks of the relevant/popular products updated and re-filled before they go out-of-stock, during that period
2. Extensive attention should be given to the logistics and delivery department during the peak period, so that the deliveries are done successfully. It is recommended to hire more delivery partners during the period to facilitate faster & more deliveries.

3. Keep the company's e-commerce website updated all the time and monitor for any server issues/crashes, as there will be a lot of traffic into the e-commerce website, during the period.

4. To help increase sales & attract more customers during the downtime of September and October 2018, we can provide special discounts/offers for the products.

5. We also need to take a deeper look by contacting the customers why they canceled the orders and try to fix the issues.

Q3. 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:

Query (BigQuery):

```
select
        sum(if(order_hour between 0 and 6, count_order_per_hour,0)) as dawn_order_count,
        sum(if(order_hour between 7 and 12, count_order_per_hour,0)) as morning_order_count,
        sum(if(order_hour between 13 and 18, count_order_per_hour,0)) as afternoon_order_count,
        sum(if(order_hour between 19 and 23, count_order_per_hour,0)) as night_order_count,
from (
  select
      distinct extract(hour from order_purchase_timestamp) as order_hour,
      count(*) over (partition by extract(hour from order_purchase_timestamp)) as count_order_per_hour
from `target_brazil.orders`
order by count_order_per_hour desc
) as X
```

Query result:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS

Row	dawn_order_count	morning_order_coun	afternoon_order_cou	night_order_count
1	5242	27733	38135	28331

Observation/Insight:

From the above query we can see the total order count from different shifts of the day (Dawn, Morning, Afternoon, Night), over the past years.

Therefore, from the above data we notice that the Brazilian customers like to place their orders **mostly in the Afternoon time**, as the total number of orders placed during that shift is the maximum.

Recommendation:

1. Keep the E-commerce website updated and address any server issues/crashes quickly, especially during the Afternoon and night shifts, as the traffic will be more during that time.

2. Assign more backend developers/Dev-ops/Customer care people during the peak order times, or align the working hours accordingly.

---

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

Q1. Get the month on month no. of orders placed in each state.
Ans:

Query (BigQuery):

```
select
      c1.customer_state,
      extract(year from o1.order_purchase_timestamp) as Year,
      extract(month from o1.order_purchase_timestamp) as month_no,
      --FORMAT_DATETIME("%b", order_purchase_timestamp) as Month,
      count(*) as order_count
from `target_brazil.customers` c1
join `target_brazil.orders` o1
on c1.customer_id = o1.customer_id
group by c1.customer_state, Year, month_no
order by c1.customer_state, Year, month_no
```

Query result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS | EXECU

Row	customer_state	Year	month_no	order_count
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6

Results per page: 50 | 1 – 50 of 565

Observation:

From the above query we get the table containing the data of month on month no. of orders placed in each state.

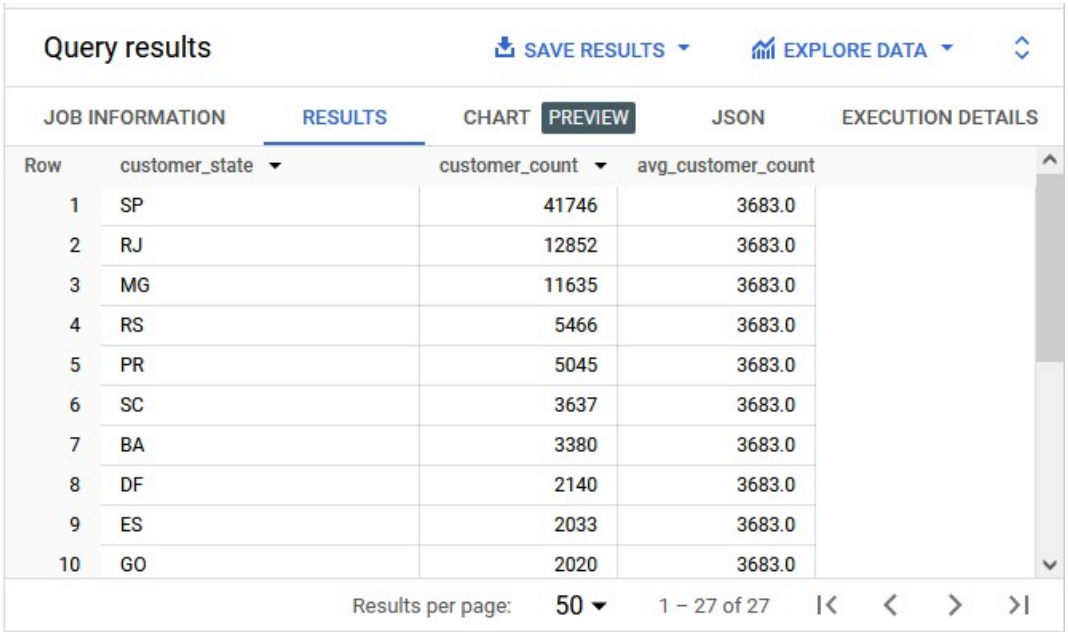
Q2. How are the customers distributed across all the states?
Ans:

Query (BigQuery):

```
select customer_state,
       customer_count,
       round(avg(customer_count) over ()) as avg_customer_count
from (select
            customer_state,
            count(*) as customer_count
```

```
from `target_brazil.customers`
group by customer_state) as X
order by customer_count desc
```

Query result:



Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS

Row	customer_state	customer_count	avg_customer_count
1	SP	41746	3683.0
2	RJ	12852	3683.0
3	MG	11635	3683.0
4	RS	5466	3683.0
5	PR	5045	3683.0
6	SC	3637	3683.0
7	BA	3380	3683.0
8	DF	2140	3683.0
9	ES	2033	3683.0
10	GO	2020	3683.0

Results per page: 50 | 1 – 27 of 27

Observation:

From the above table we are able to see the data of how the customers are distributed across all the states in Brazil.

The table is ordered from the highest number of customers to the lowest across the states along with the average customer count.

State with the highest customer count is **SP**

State with the lowest customer count is **RR**

---

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

Q1. 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:

Query (BigQuery):

```
select
      X.Year,
      X.avg_payment_value,
      round(((X.avg_payment_value-(lag(X.avg_payment_value,1) over (order by
X.Year)))/(lag(X.avg_payment_value,1) over (order by X.Year)))*100,2) as percent_increase
from (select
      distinct extract(year from o1.order_purchase_timestamp) as Year,
      round(avg(p1.payment_value) over (partition by extract(year from
o1.order_purchase_timestamp)), 2) as avg_payment_value
from `target_brazil.payments` p1
join `target_brazil.orders` o1
on p1.order_id = o1.order_id
```

```
where (extract(year from o1.order_purchase_timestamp) between 2017 and 2018) and (extract(month
from o1.order_purchase_timestamp) between 1 and 8)
order by Year) X
order by X.Year
```

Query result:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	Year	avg_payment_value	percent_increase
1	2017	150.43	null
2	2018	155.28	3.22

Observation:

From the above query we have calculated that the percent increase in the cost of order is **3.22 %** from 2017 to 2018 (including months from January to August only).

We have taken the average of the "payment_value" from the payments table to calculate the percentage increase.

Q2. Calculate the Total & Average value of order price for each state.
Ans:

Query (BigQuery):

```
select
      c1.customer_state,
      round(sum(oi1.price),2) as Total_order_price,
      round(avg(oi1.price),2) as Average_order_price
from `target_brazil.order_items` oi1
join `target_brazil.orders` o1
on o1.order_id = oi1.order_id
join `target_brazil.customers`c1
on o1.customer_id = c1.customer_id
group by c1.customer_state
order by Average_order_price desc
```

Query result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DET.

Row	customer_state	Total_order_price	Average_order_price
1	PB	115268.08	191.48
2	AL	80314.81	180.89
3	AC	15982.95	173.73
4	RO	46140.64	165.97
5	PA	178947.81	165.69
6	AP	13474.3	164.32
7	PI	86914.08	160.36
8	TO	49621.74	157.53
9	RN	83034.98	156.97
10	CE	227254.71	153.76

Results per page: 50 | 1 – 27 of 27

Observation:

From the above table we get the data of the total and average order price in each state.
The Average_order_price is ordered in descending order in the above table.
State with **Highest Average_order_price** is **PB.**
State with **Lowest Average_order_price** is **SP.**

Q3. Calculate the Total & Average value of order freight for each state.
Ans:

Query (BigQuery):

```
select
      c1.customer_state,
      round(sum(oi1.freight_value),2) as Total_freight_value,
      round(avg(oi1.freight_value),2) as Average_freight_value
from `target_brazil.order_items` oi1
join `target_brazil.orders` o1
on o1.order_id = oi1.order_id
join `target_brazil.customers`c1
on o1.customer_id = c1.customer_id
group by c1.customer_state
order by Average_freight_value desc
```

Query result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS

Row	customer_state	Total_freight_value	Average_freight_valu
1	RR	2235.19	42.98
2	PB	25719.73	42.72
3	RO	11417.38	41.07
4	AC	3686.75	40.07
5	PI	21218.2	39.15
6	MA	31523.77	38.26
7	TO	11732.68	37.25
8	SE	14111.47	36.65
9	AL	15914.59	35.84
10	PA	38699.3	35.83

Results per page: 50 | 1 – 27 of 27

Observation:

From the above table we get the data of the total and average freight value in each state.
The Average_freight_value is ordered in descending order in the above table.
State with **Highest Average_ freight_value** is **RR.**
State with **Lowest Average_ freight_value** is **SP.**

---

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

Q1. 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.

Ans:

Query (BigQuery):

```
select
      order_id,
      order_purchase_timestamp,
      order_delivered_customer_date,
      order_estimated_delivery_date,
      timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
time_to_deliver,
      timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as
diff_estimated_delivery
from `target_brazil.orders`
where order_delivered_customer_date is not NULL
order by time_to_deliver, diff_estimated_delivery desc
```

Query result:

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_deliv
1	8339b608be0d84fca9d8da68b...	2018-06-26 20:48:33 UTC	2018-06-27 17:31:53 UTC	2018-07-25 00:00:00 UTC	0	27
2	bb5a519e352b45b714192a02f...	2017-05-31 11:11:55 UTC	2017-06-01 08:34:36 UTC	2017-06-27 00:00:00 UTC	0	25
3	434cecee7d1a65fc65358a632...	2017-05-29 13:21:46 UTC	2017-05-30 08:06:56 UTC	2017-06-19 00:00:00 UTC	0	19
4	38c1e3d4ed6a13cd0cf612d4c...	2018-02-02 15:26:38 UTC	2018-02-03 15:05:56 UTC	2018-02-20 00:00:00 UTC	0	16
5	f349cdb62f69c3fae5c4d7d3f3...	2018-06-28 14:34:48 UTC	2018-06-29 14:12:18 UTC	2018-07-12 00:00:00 UTC	0	12
6	d3ca7b82c922817b06e5ca211...	2017-11-16 13:54:08 UTC	2017-11-17 13:49:40 UTC	2017-11-29 00:00:00 UTC	0	11
7	f3c6775ba3d2d9fe2826f93b71...	2017-07-04 11:37:47 UTC	2017-07-05 08:09:26 UTC	2017-07-17 00:00:00 UTC	0	11
8	21a8ffca665bc7a1087d31751...	2017-05-31 12:00:35 UTC	2017-06-01 10:28:24 UTC	2017-06-13 00:00:00 UTC	0	11
9	1d893dd7ca5f77ebf5f59f0d20...	2017-06-19 08:19:45 UTC	2017-06-19 21:07:52 UTC	2017-06-30 00:00:00 UTC	0	10
10	e65f1eeee1f52024ad1dcd034...	2018-05-18 15:03:19 UTC	2018-05-19 12:28:30 UTC	2018-05-29 00:00:00 UTC	0	9

Results per page: 50 | 1 – 50 of 96476

Observation:

From the above table we are able to get the no. of days taken to deliver an order to the customer as well as the difference in the actual and estimated delivery date.

The table is ordered according to the number of days taken to deliver each order, the fastest deliveries are listed on the top followed by the deliveries which took more time or which have a negative difference between the estimated and actual delivery date (delivery happened later the estimated date).

Q2. Find out the top 5 states with the highest & lowest average freight value.

Ans:

Query (BigQuery):

```
with avg_freight as (
  select
      c1.customer_state,
      round(avg(oi1.freight_value),2) as Avg_freight_value
  from `target_brazil.order_items` oi1
  join `target_brazil.orders` o1
  on o1.order_id = oi1.order_id
  join `target_brazil.customers`c1
```

```
  on o1.customer_id = c1.customer_id
  group by c1.customer_state
),

rank_avg_freight as (
  select avg_freight.customer_state,
         avg_freight.Avg_freight_value,
         dense_rank() over (order by avg_freight.Avg_freight_value desc) as rank_highest,
         dense_rank() over (order by avg_freight.Avg_freight_value) as rank_lowest
  from avg_freight
)

select r1.customer_state as Highest_avg_freight_state,
       r1.Avg_freight_value as Highest_avg_freight_value,
       r2.customer_state as Lowest_avg_freight_state,
       r2.Avg_freight_value as Lowest_avg_freight_value
from rank_avg_freight r1
join rank_avg_freight r2
on r1.rank_highest = r2.rank_lowest
where r1.rank_highest between 1 and 5
order by r1.rank_highest
```

Query result:

Query results

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

Row	Highest_avg_freight_state	Highest_avg_freight_	Lowest_avg_freight_state	Lowest_avg_freight_
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04

Observation:

From the above table we get the data of the Top 5 states with the highest and lowest average freight values.

Q3. Find out the top 5 states with the highest & lowest average delivery time.
Ans:

Query (BigQuery):

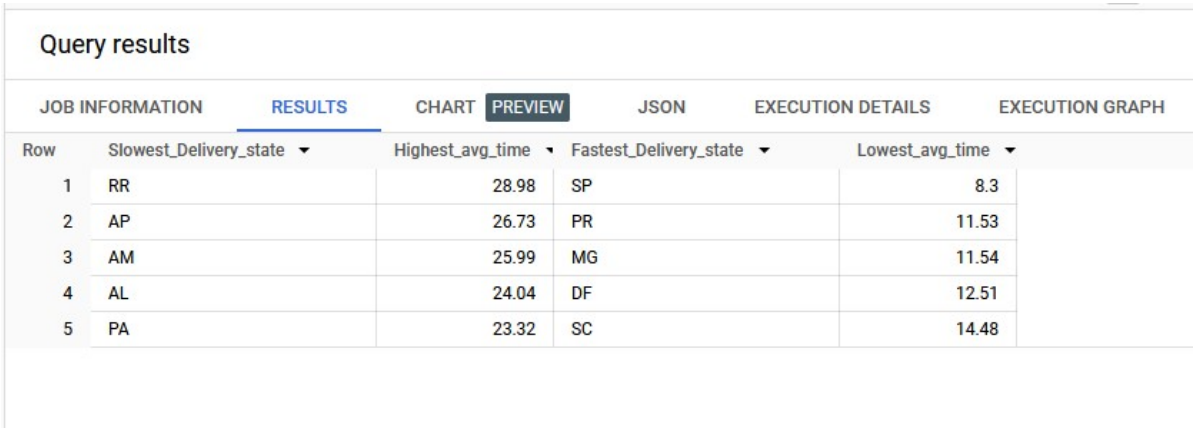
```
with avg_delv_time as (
   select
         c1.customer_state,
         round(avg(timestamp_diff(order_delivered_customer_date, order_purchase_timestamp,
day)),2) as   avg_time_to_deliver
   from `target_brazil.orders` o1
   join `target_brazil.customers`c1
   on o1.customer_id = c1.customer_id
   group by c1.customer_state
),

rank_delv_time as (
  select avg_delv_time.customer_state,
         avg_delv_time.avg_time_to_deliver,
```

```
        dense_rank() over (order by avg_delv_time.avg_time_to_deliver desc) as rank_highest,
        dense_rank() over (order by avg_delv_time.avg_time_to_deliver) as rank_lowest
  from avg_delv_time
)

select r1.customer_state as Slowest_Delivery_state,
       r1.avg_time_to_deliver as Highest_avg_time,
       r2.customer_state as Fastest_Delivery_state,
       r2.avg_time_to_deliver as Lowest_avg_time
from rank_delv_time r1
join rank_delv_time r2
on r1.rank_highest = r2.rank_lowest
where r1.rank_highest between 1 and 5
order by r1.rank_highest
```

Query result:



Query results

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

Row	Slowest_Delivery_state	Highest_avg_time	Fastest_Delivery_state	Lowest_avg_time
1	RR	28.98	SP	8.3
2	AP	26.73	PR	11.53
3	AM	25.99	MG	11.54
4	AL	24.04	DF	12.51
5	PA	23.32	SC	14.48

Observation:

From the above table we get the data of the top 5 states with the highest(slowest) and lowest(fastest) average delivery time (in days)

Insight:

From the above data including the no. of customers , average freight values and delivery times in each state, we found out that, states with higher average freight value and in turn higher average delivery time, tend to have lesser customer count.

For example, SP has the lowest average freight value, along with the lowest average freight value, and it has the highest no. of customers.

For the state of RR it is just the vice versa of SP.

Recommendation:

Opening of more local stores/hubs in states where delivery times, freight values are high, can definitely help attracting new customers and increasing the number of orders and sales.

---

6. **Analysis based on the payments:**

Q1. Find the month on month no. of orders placed using different payment types.
Ans:

Query (BigQuery):

```
select
      extract(year from o1.order_purchase_timestamp) as Year,
      extract(month from o1.order_purchase_timestamp) as Month_no,
      p1.payment_type,
      count(*) as order_count
from `target_brazil.orders` o1
join `target_brazil.payments` p1
on o1.order_id = p1.order_id
group by Year, Month_no, p1.payment_type
order by Year, Month_no, order_count desc
```

Query result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION D

Row	Year	Month_no	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9

Results per page: 50 | 1 – 50 of 90

Observation:

From the above table we get the data of the different payment types used for the order purchases per month.

The data indicates that the customers mostly use 4 different payment types for their purchases, **credit card, UPI, voucher, and debit card**.

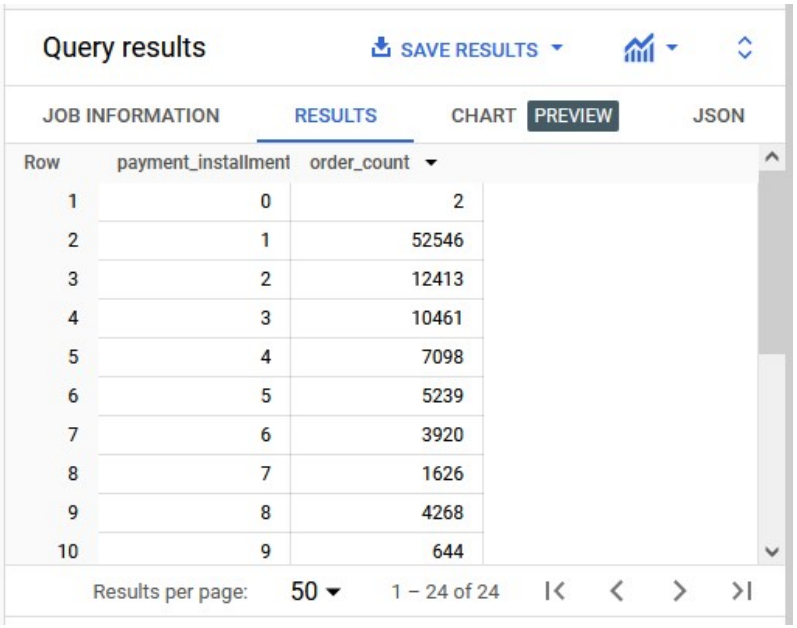
The **most popular** payment mode each month, with the highest number of purchases is **credit cards**.

Q2. Find the no. of orders placed on the basis of the payment installments that have been paid.
Ans:

Query (BigQuery):

```
select
      payment_installments,
      count(order_id) as order_count
from `target_brazil.payments`
group by payment_installments
order by payment_installments
```

Query result:



Row	payment_installment	order_count
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

Results per page: 50 1 – 24 of 24

Observation:

From the above table we get the data of the no. of orders placed based on the payment installments that have been paid.

The payment_installment having the highest no. of orders is 1.

==========