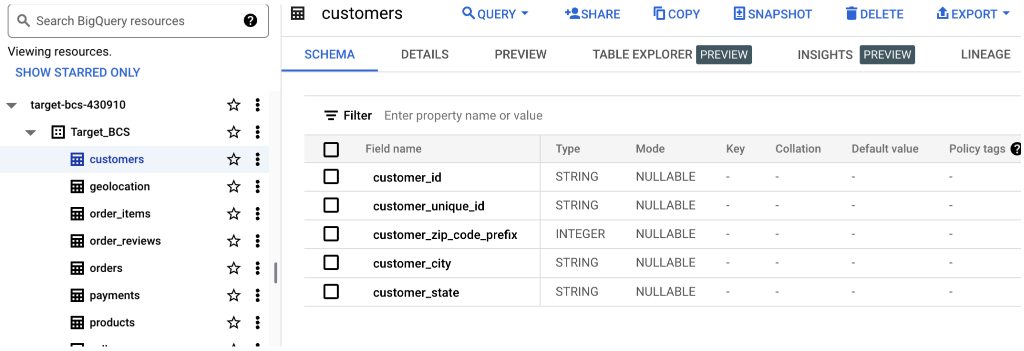
TARGET - Business Case Study

-By Suhasini M Mottannavar

1.1 Data type of all columns in the "customers" table



☐	Field name	Type	Mode	Key	Collation	Default value	Policy tags
☐	customer_id	STRING	NULLABLE	-	-	-	-
☐	customer_unique_id	STRING	NULLABLE	-	-	-	-
☐	customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-
☐	customer_city	STRING	NULLABLE	-	-	-	-
☐	customer_state	STRING	NULLABLE	-	-	-	-

Fig: Data type of all the columns from “customer” table.

From the above fig we know that datas are stored in “string” fromat for the columns represinting “customer_id”, “customer_unique_id”, “customer_city” and “customer_state”, where as data stored in “integer” format for the column “customer_zip_code_prefix”.

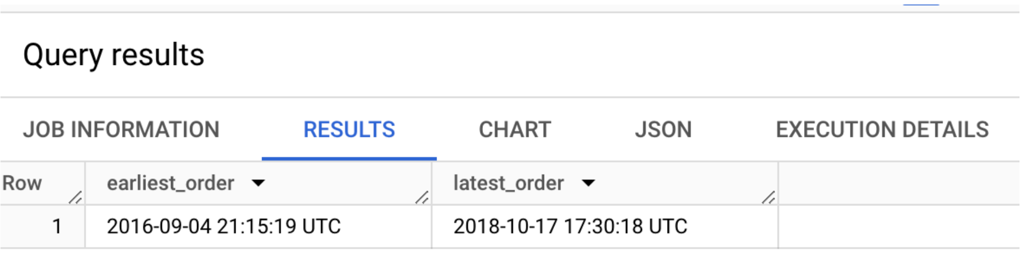
We can also use the following query to fetch the data type of all the cloumns in the customer table

```
SELECT
    column_name,
    data_type
FROM
    `target-bcs-430910.Target_BCS.INFORMATION_SCHEMA.COLUMNS`
WHERE
    table_name = 'customers';
```

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

From the query

```
SELECT
    MIN(order_purchase_timestamp) AS earliest_order,
    MAX(order_purchase_timestamp) AS latest_order
FROM Target_BCS.orders;
```



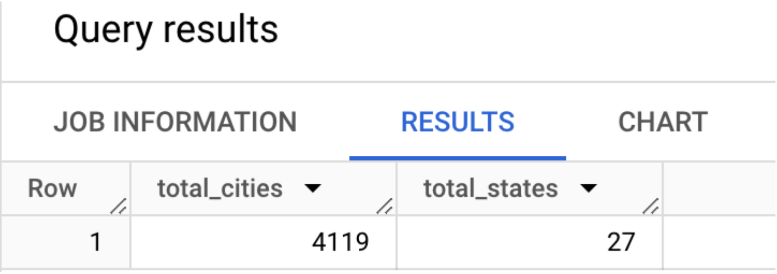
Row	earliest_order	latest_order
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Fig: date & time of the first and last orders purchased .

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

From the query

```
SELECT
    COUNT(DISTINCT c.customer_city) AS total_cities,
    COUNT(DISTINCT c.customer_state) AS total_states
FROM
    Target_BCS.customers c
JOIN
    Target_BCS.orders o on o.customer_id = c.customer_id
WHERE
    o.order_purchase_timestamp BETWEEN '2016-09-04' AND
'2018-10-17';
```



Row	total_cities	total_states
1	4119	27

Fig: total number of cities & states where orders were placed by the customers

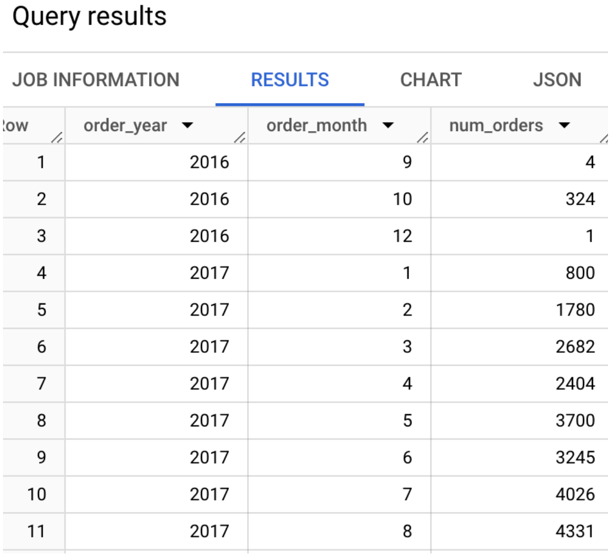
2.1 Is there a growing trend in the no. of orders placed over the past years?

From the query

```
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS
order_year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS
order_month,
    COUNT(*) AS num_orders
FROM Target_BCS.orders
GROUP BY 1,2
ORDER BY 1,2;
```

From the query

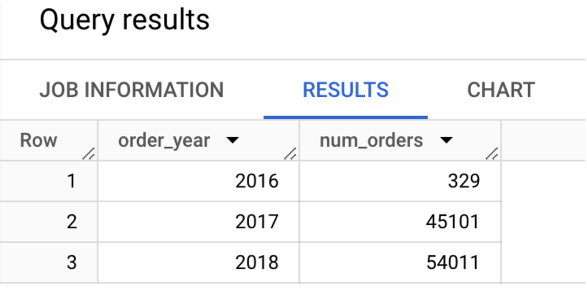
```
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS
order_year,
    COUNT(*) AS num_orders
FROM Target_BCS.orders
GROUP BY order_year
ORDER BY order_year;
```



Query results

tow	order_year	order_month	num_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
11	2017	8	4331

Fig 2.1.1: no. of orders placed in each month, over the past years.



Query results

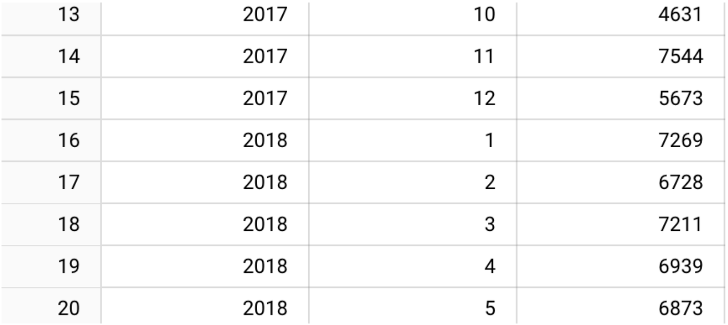
Row	order_year	num_orders
1	2016	329
2	2017	45101
3	2018	54011

Fig 2.1.2: no. of orders placed in each year, over the past years.

From the fig 2.1.1 and fig 2.1.2 we can conclude that there is a gradually growing trend in the no. of orders placed over the months and the past years.

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

From the table 2.1.1



13	2017	10	4631
14	2017	11	7544
15	2017	12	5673
16	2018	1	7269
17	2018	2	6728
18	2018	3	7211
19	2018	4	6939
20	2018	5	6873

Fig: no. of orders placed are at peak during certain months.

From the above fig we can see that there is a monthly seasonality in terms of the no.of orders being placed. The no. of orders made during the month of 11(November), 2017 and the month of 1(January), 3(March), 2018 are in the peak crossing the no.of orders over 7000 compared to that of other months of the years. The heighst no.of orders are made in the month of 11,2017.

We can also use the following query to fetch the data of monthly seasonality in terms of the no. of orders being placed, this information can be valuable for inventory management, marketing strategies, and staffing to align with expected order volumes.

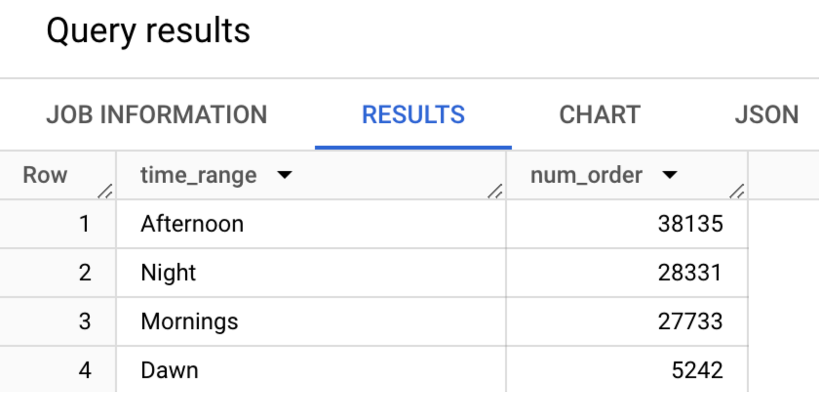
```
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
    COUNT(order_id) AS total_orders
FROM
    `Target_BCS.orders`
GROUP BY
    year, month
ORDER BY
    year, month;
```

consistent peaks in certain months, it indicates a seasonality pattern, suggesting those months have higher demand or sales activity

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

From the query

```
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp)
BETWEEN 0 AND 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp)
BETWEEN 7 AND 12 THEN 'Mornings'
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp)
BETWEEN 13 AND 18 THEN 'Afternoon'
    WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp)
BETWEEN 19 AND 23 THEN 'Night'
    END as time_range,
  COUNT(o.order_id) as num_order
FROM
  Target_BCS.orders o
JOIN
  Target_BCS.customers c ON o.customer_id = c.customer_id
GROUP BY
  time_range
ORDER BY
  2 desc;
```

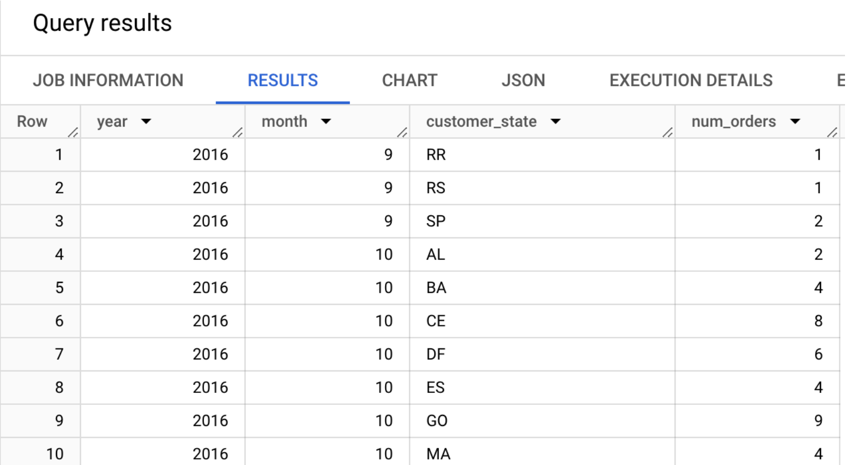


Row	time_range	num_order
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Fig: no. of orders classified on the bases of time period of the day

From the above fig we can conclude that most Brazilian customers place their orders in the afternoon.

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



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

Fig: no. of orders placed in each state, in each month by customers.

From the 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_BCS.orders o
JOIN
  Target_BCS.customers c
```

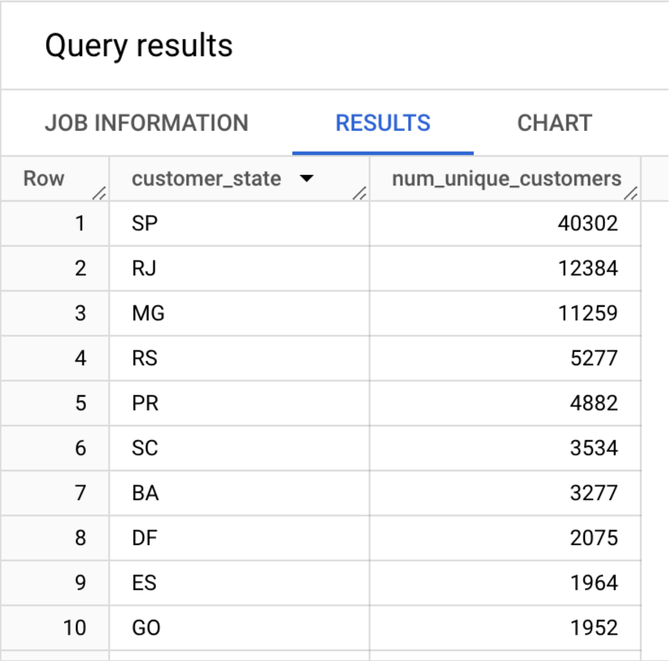
```
USING
  (customer_id)
GROUP BY
  year, month, c.customer_state
ORDER BY
  year, month, c.customer_state;
```

Tracking the month-on-month number of orders placed in each state provides essential insights into regional demand, operational efficiency, and strategic planning. It enables businesses to respond effectively to trends, optimize resources, and tailor marketing efforts, ultimately supporting better decision-making and growth.

3.2 How are the customers distributed across all the states?

From the query

```
SELECT
  customer_state,
  COUNT(DISTINCT customer_unique_id) AS num_unique_customers
FROM
  `Target_BCS.customers`
GROUP BY
  customer_state
ORDER BY
  num_unique_customers DESC;
```



Row	customer_state	num_unique_customers
1	SP	40302
2	RJ	12384
3	MG	11259
4	RS	5277
5	PR	4882
6	SC	3534
7	BA	3277
8	DF	2075
9	ES	1964
10	GO	1952

Fig: no. of unique customers present in each state.

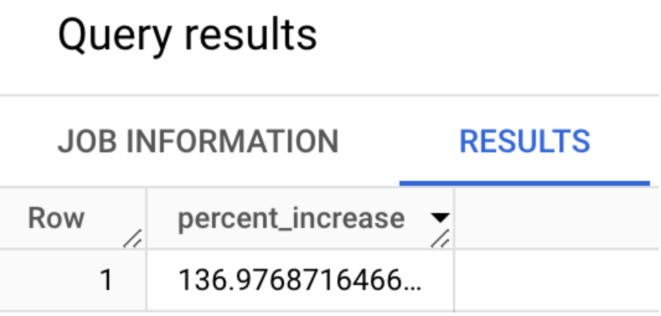
From the table we know that customer state “SP” has the heighest no. of unique customers over 40,302 followed by “RJ” and “MJ” with the maximum no. of unique customers compared to other states.

4.1 Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

From the query

```
WITH orders_2017_18 AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
    p.payment_value
  FROM
    `Target_BCS.orders` o
  JOIN
    `Target_BCS.payments` p
  using
    (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
),
sum_paymentvalue AS (
  SELECT
    year,
    SUM(payment_value) AS total_payment
  FROM
    orders_2017_18
  GROUP BY
    year
)

SELECT
  (SUM(CASE WHEN year = 2018 THEN total_payment ELSE 0 END) -
   SUM(CASE WHEN year = 2017 THEN total_payment ELSE 0
END)) /
   SUM(CASE WHEN year = 2017 THEN total_payment ELSE 0 END) *
100 AS percent_increase
FROM
  sum_paymentvalue;
```



Row	percent_increase
1	136.9768716466...

Fig: percentage increase in the cost of orders from year 2017 to 2018 (months included between Jan to Aug only).

From the above table we can say that the percentage increase in the total payment value from January to August 2017 to the same period in 2018 gives us a measure of growth in sales. A percentage increase of 136.978% indicates that the value has more than doubled since that year. This substantial increase points to significant growth or improvement in the cost of orders.

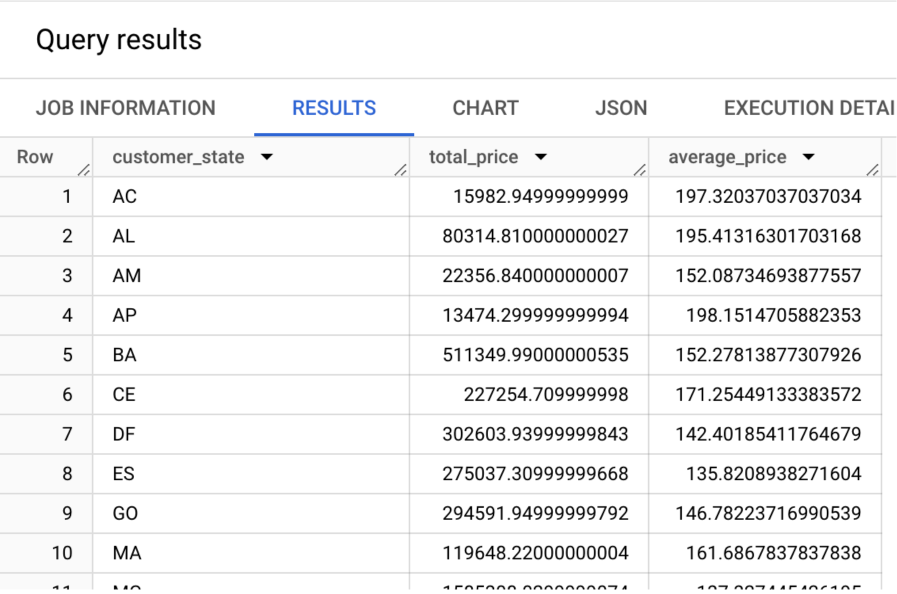
4.2 Calculate the Total & Average value of order price for each state.

From the query

```
WITH OrderPrices AS (
    SELECT
        o.order_id,
        SUM(oi.price) AS total_order_price,
        c.customer_state
    FROM
        `Target_BCS.orders` o
    JOIN
        `Target_BCS.order_items` oi ON o.order_id =
oi.order_id
    JOIN
        `Target_BCS.customers` c ON o.customer_id =
c.customer_id
    GROUP BY
        o.order_id, c.customer_state
)

SELECT
    customer_state,
    SUM(total_order_price) AS total_price,
    AVG(total_order_price) AS average_price
FROM
    OrderPrices
GROUP BY
```

```
    customer_state
ORDER BY
    customer_state ASC;
```


Query results

Row	customer_state	total_price	average_price
1	AC	15982.94999999999	197.32037037037034
2	AL	80314.810000000027	195.41316301703168
3	AM	22356.840000000007	152.08734693877557
4	AP	13474.299999999994	198.1514705882353
5	BA	511349.99000000535	152.27813877307926
6	CE	227254.709999998	171.25449133383572
7	DF	302603.93999999843	142.40185411764679
8	ES	275037.30999999668	135.8208938271604
9	GO	294591.94999999792	146.78223716990539
10	MA	119648.22000000004	161.6867837837838
11	MG	[illegible]	[illegible]

Fig: total price and the average price of orders for each state.

From the above table we can identify the total price and the average price of orders of each state. Here after dividing the total price by average price we get the decent amount of orders placed which suggest that there is a moderate volume of orders.
In the future to increase the average price considering the strategies like offering discounts on larger purchases, cross-selling, or upselling will be usefull, and for the goal to increase total revenue, boosting the number of orders through marketing campaigns, promotions, or expanding customer reach might be necessary.

4.3Calculate the Total & Average value of order freight for each state.

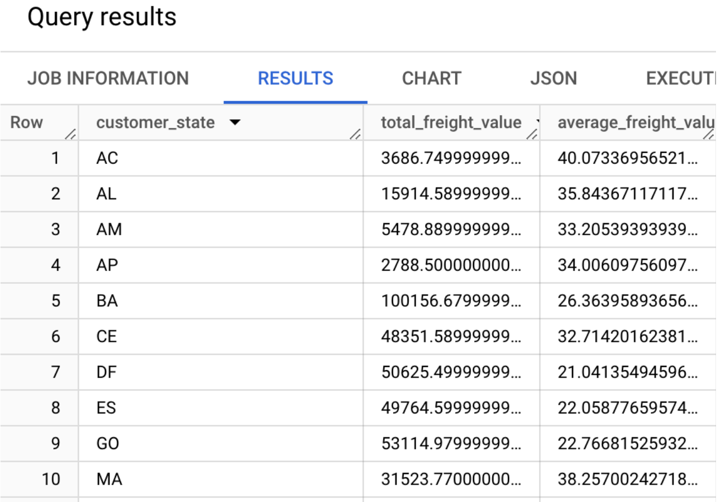
From the query

```
WITH Orderfreight_value AS (
    SELECT
        o.order_id,
        oi.freight_value AS total_freight_value,
        c.customer_state
    FROM
        `Target_BCS.orders` o
    JOIN
```

```
        `Target_BCS.order_items` oi ON o.order_id =
oi.order_id
    JOIN
        `Target_BCS.customers` c ON o.customer_id =
c.customer_id

)

SELECT
    customer_state,
    SUM(total_freight_value) AS total_freight_value,
    AVG(total_freight_value) AS average_freight_value
FROM
    Orderfreight_value
GROUP BY
    customer_state
ORDER BY
    customer_state ASC;
```



Query results

Row	customer_state	total_freight_value	average_freight_valu
1	AC	3686.749999999...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.889999999...	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...
10	MA	31523.77000000...	38.25700242718...

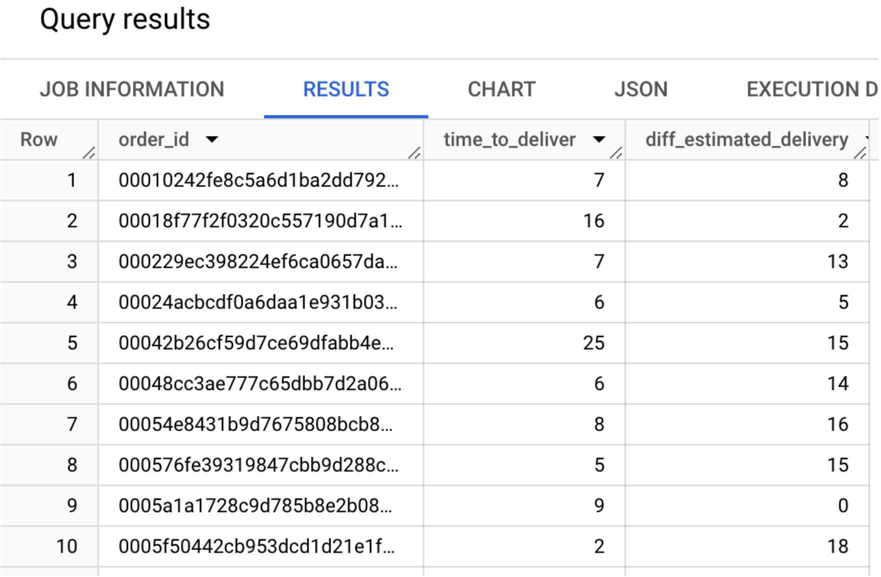
Fig: total freight value and the average freight value of orders for each state.

From the query we know that the total freight value which is the cumulative cost across all orders is high which indicate a large number of orders being shipped to that states, or that shipping costs for that state are generally high.
By dividing total freight value by average freight vale we get the total shipments of orders for each state.

5.1 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.

From the query

```
SELECT
    order_id,
    DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) AS time_to_deliver,
    DATE_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY) AS
diff_estimated_delivery
FROM
    `Target_BCS.orders`
WHERE
    order_delivered_customer_date IS NOT NULL
ORDER BY
    order_id ASC;
```



Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18

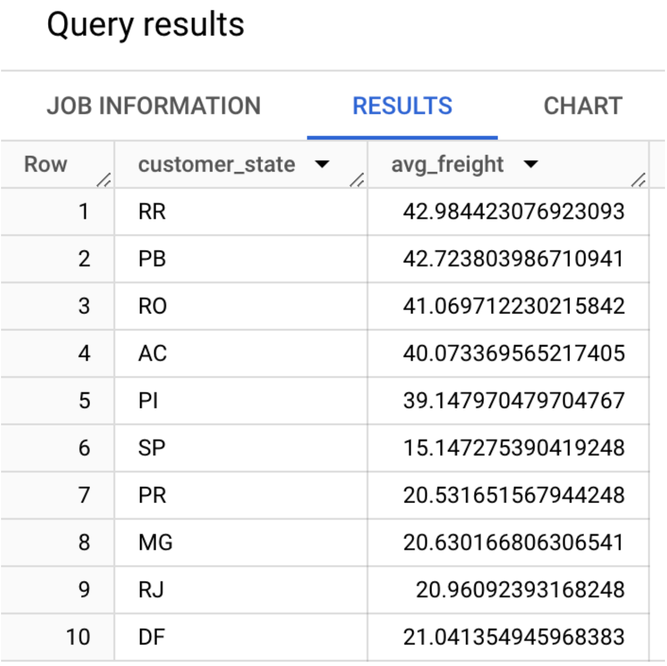
Fig: delivery time and the difference between the estimated & actual delivery date

From the above table we can see the deliver time and the estimated deliver time, maximum of the orders are delivered within the the estimated delivery days.

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

From the query

```
WITH StateFreight AS (
  SELECT
    c.customer_state,
    AVG(oi.freight_value) AS avg_freight
  FROM Target_BCS.customers c
  JOIN Target_BCS.orders o ON c.customer_id = o.customer_id
  join Target_BCS.order_items oi on o.order_id = oi.order_id
  GROUP BY
    c.customer_state
)
(SELECT
  customer_state,
  avg_freight
FROM
  StateFreight
ORDER BY
  avg_freight DESC
LIMIT
  5)
UNION ALL
(SELECT
  customer_state,
  avg_freight
FROM (
  SELECT
    customer_state,
    avg_freight
  FROM
    StateFreight
  ORDER BY
    avg_freight ASC
  LIMIT
    5 ));
```



Row	customer_state	avg_freight
1	RR	42.984423076923093
2	PB	42.723803986710941
3	RO	41.069712230215842
4	AC	40.073369565217405
5	PI	39.147970479704767
6	SP	15.147275390419248
7	PR	20.531651567944248
8	MG	20.630166806306541
9	RJ	20.96092393168248
10	DF	21.041354945968383

Fig: top 5 states with the highest & lowest average freight value.

From the table we can see that the first five rows are the top 5 sates with the average highest freight value where in the last five rows are the states with the lowest average freight value.
Here the heighest freight value may indicate premium services, longer distances, heavier weights, or urgent deliveries. Low freight values often suggest cost-effective shipments, possibly smaller or lighter items, or less urgent deliveries.

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

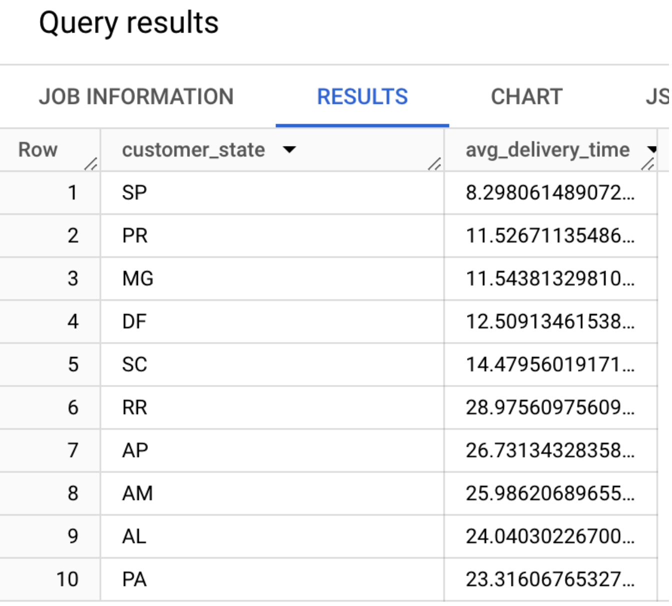
From the query

```
WITH DeliveryTime AS (
    SELECT
        c.customer_state,
        DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY) AS delivery_time
    FROM
        `Target_BCS.orders` o
    JOIN
        `Target_BCS.customers` c ON o.customer_id =
c.customer_id
    WHERE
        o.order_delivered_customer_date IS NOT NULL
)

(SELECT
    customer_state,
    AVG(delivery_time) AS avg_delivery_time
FROM
    DeliveryTime
GROUP BY
    customer_state
ORDER BY
    avg_delivery_time ASC
LIMIT 5)

UNION ALL

(SELECT
    customer_state,
    AVG(delivery_time) AS avg_delivery_time
FROM
    DeliveryTime
GROUP BY
    customer_state
ORDER BY
    avg_delivery_time DESC
LIMIT 5);
```



Row	customer_state	avg_delivery_time
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...
6	RR	28.97560975609...
7	AP	26.73134328358...
8	AM	25.98620689655...
9	AL	24.04030226700...
10	PA	23.31606765327...

Fig: top 5 states with the highest & lowest average delivery time.

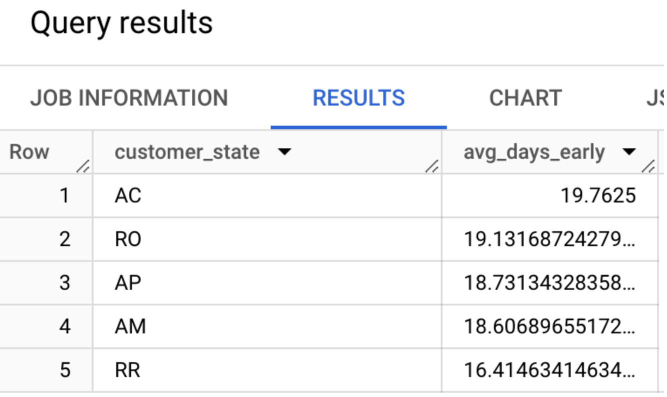
From the tabel we can we can see that the first five rows are the top 5 sates with the average delivery time where in the last five rows are the states with the lowest average delivery time.
Highest average delivery time may indicate delays, inefficiencies, complex shipments or even longer distances or challenging locations. Lowest average delivery time may indicate shorter distance or good logistics which result in effective process and routing and reliable services.

5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

From the query

```
WITH DeliveryDifference AS (
    SELECT
        c.customer_state,
        AVG(DATE_DIFF(o.order_estimated_delivery_date, 
o.order_delivered_customer_date, DAY)) AS avg_days_early
    FROM
        `Target_BCS.orders` o
    JOIN
        `Target_BCS.customers` c ON o.customer_id = 
c.customer_id
    WHERE
        o.order_delivered_customer_date IS NOT NULL
    GROUP BY
        c.customer_state
)
```

```
SELECT
    customer_state,
    avg_days_early
FROM
    DeliveryDifference
ORDER BY
    avg_days_early DESC
LIMIT 5;
```



Row	customer_state	avg_days_early
1	AC	19.7625
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

Fig: top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

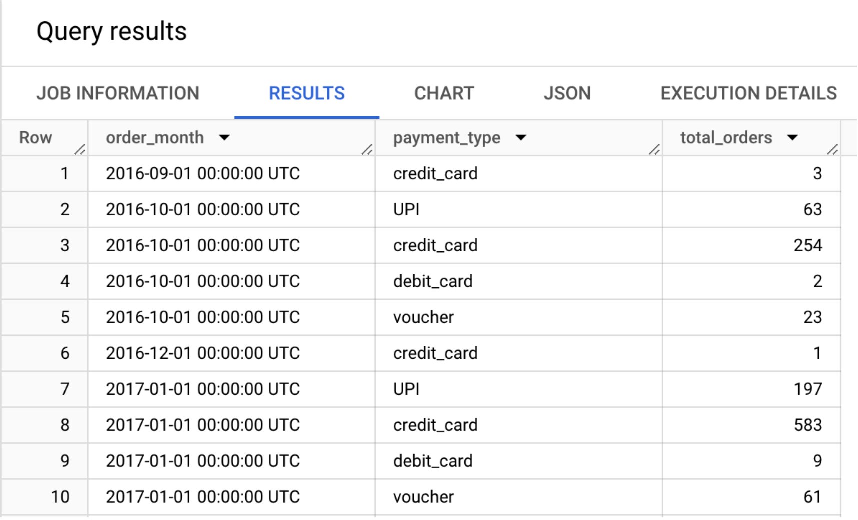
From the above table we see the top 5 states that have the order delivered really fast, this are the states with efficient logistics, optimized routes and has the accurate and up-to-date information and faster handling.

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

From the query

```
WITH OrderPayments AS (
    SELECT
        o.order_id,
        o.order_purchase_timestamp,
        p.payment_type
    FROM
        `Target_BCS.orders` o
    JOIN
        `Target_BCS.payments` p using (order_id)
)
SELECT
    DATE_TRUNC(order_purchase_timestamp, MONTH) AS
order_month,
    payment_type,
    COUNT(order_id) AS total_orders
FROM
```

```
    OrderPayments
GROUP BY
    order_month, payment_type
ORDER BY
    order_month ASC, payment_type ASC;
```



Query results

Row	order_month	payment_type	total_orders
1	2016-09-01 00:00:00 UTC	credit_card	3
2	2016-10-01 00:00:00 UTC	UPI	63
3	2016-10-01 00:00:00 UTC	credit_card	254
4	2016-10-01 00:00:00 UTC	debit_card	2
5	2016-10-01 00:00:00 UTC	voucher	23
6	2016-12-01 00:00:00 UTC	credit_card	1
7	2017-01-01 00:00:00 UTC	UPI	197
8	2017-01-01 00:00:00 UTC	credit_card	583
9	2017-01-01 00:00:00 UTC	debit_card	9
10	2017-01-01 00:00:00 UTC	voucher	61

Fig: month on month no. of orders placed using different payment types.

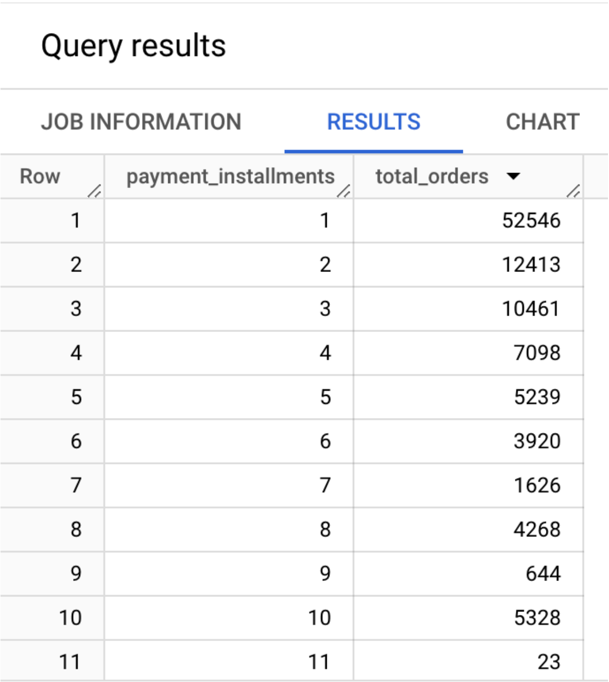
From the above table we can see the order month and the mode of payment with total no. of orders in each payment modes. We can conclude that the is acceptance of multiple payments to cater a broad customer base and enhance the customer experiences. There is also promotional offers given like vouchers to promote customer loyaly.

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

From the query

```
SELECT
    p.payment_installments,
    COUNT(o.order_id) AS total_orders
FROM
    `Target_BCS.orders` o
JOIN
    `Target_BCS.payments` p ON o.order_id = p.order_id
WHERE
    p.payment_installments > 0
GROUP BY
    p.payment_installments
ORDER BY
```

```
p.payment_installments ASC;
```



Row	payment_installments	total_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328
11	11	23

Fig: no. of orders placed on the basis of the payment installments that have been paid.

From the above table we can see the no. of orders placed based on the no. of payment installments where at least one installment has been successfully paid.

Orders placed on an installment basis reflect consumer preferences for financial flexibility and affordability, as well as business strategies to increase sales and customer satisfaction. Installment payments help manage cash flow, enable access to higher-value items, and provide a competitive edge for businesses.

Thank you