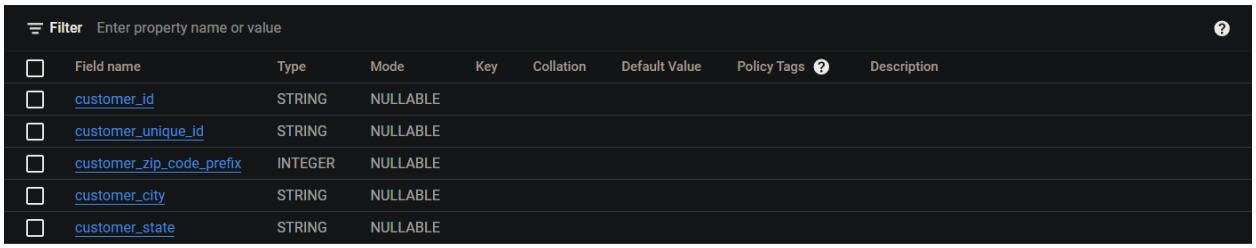
TARGET SQL BUSINESS CASE

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

Query :

```
select column_name, data_type
From targetsql-403810.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'customers';
```

Query Results :



Filter Enter property name or value

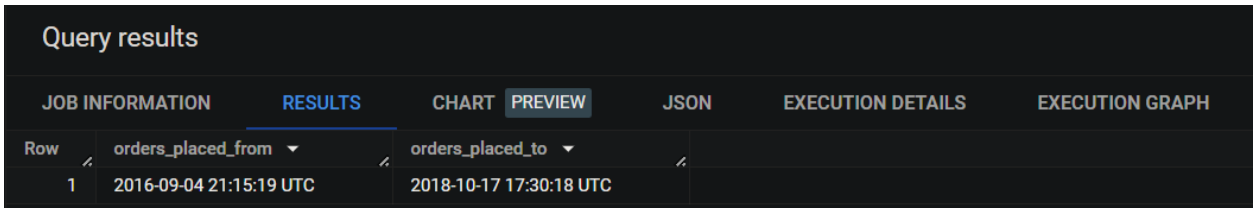
Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
customer_id	STRING	NULLABLE					
customer_unique_id	STRING	NULLABLE					
customer_zip_code_prefix	INTEGER	NULLABLE					
customer_city	STRING	NULLABLE					
customer_state	STRING	NULLABLE					

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

Query :

```
select
  min(order_purchase_timestamp) as `orders_placed_from`,
  max(order_purchase_timestamp) as `orders_placed_to`
from `target_sql.orders`;
```

Query Results :

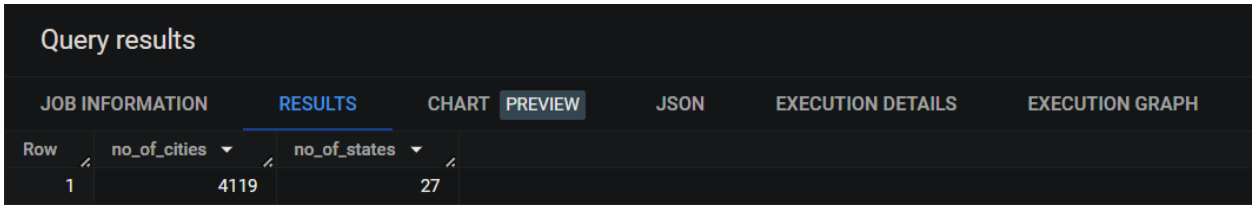


#1.3 Count the Cities & States of customers who ordered during the given period.

Query :

```
select
  count(distinct(customer_city)) no_of_cities,
  count(distinct(customer_state)) no_of_states
from
  `target_sql.customers`
```

Query Results :



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

Query :

```
select
  EXTRACT(YEAR from order_purchase_timestamp) as ordYear,
  count(distinct(order_id)) orders_per_year
from
  `target_sql.orders`
group by
  ordYear
order by
  ordYear;
```

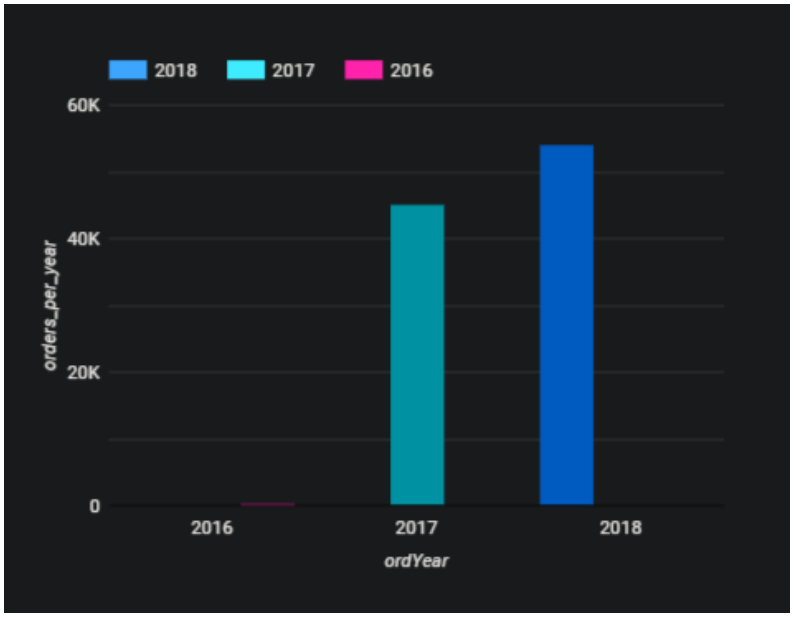
Query Results :

Query results

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

Row	ordYear	orders_per_year
1	2016	329
2	2017	45101
3	2018	54011

Chart :



Insights :

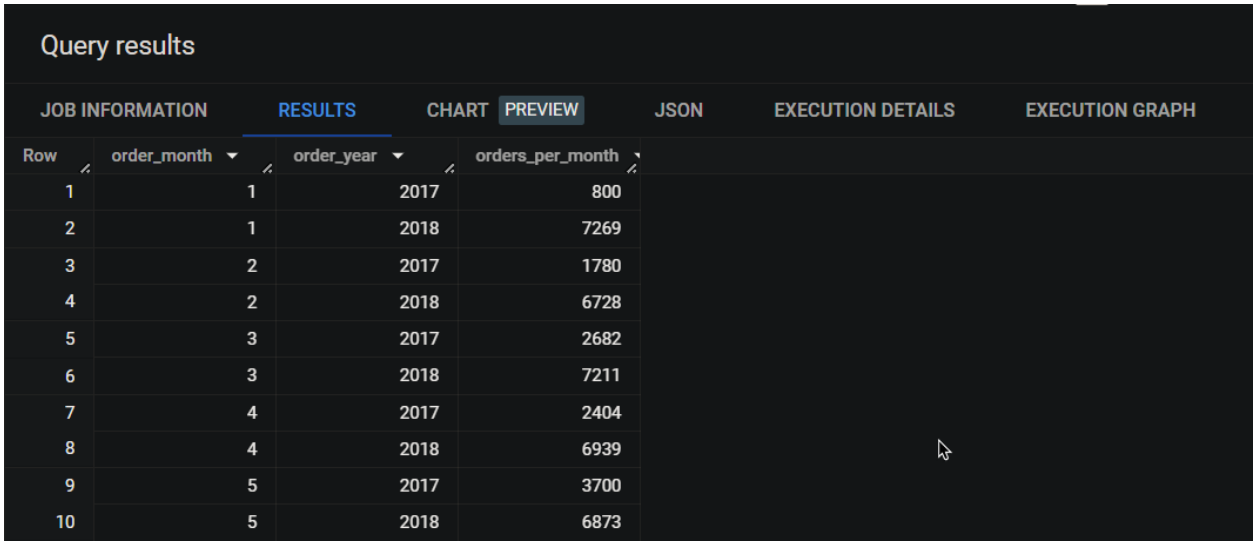
Yes, there is a year on year growth in the number of orders placed.

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

Query :

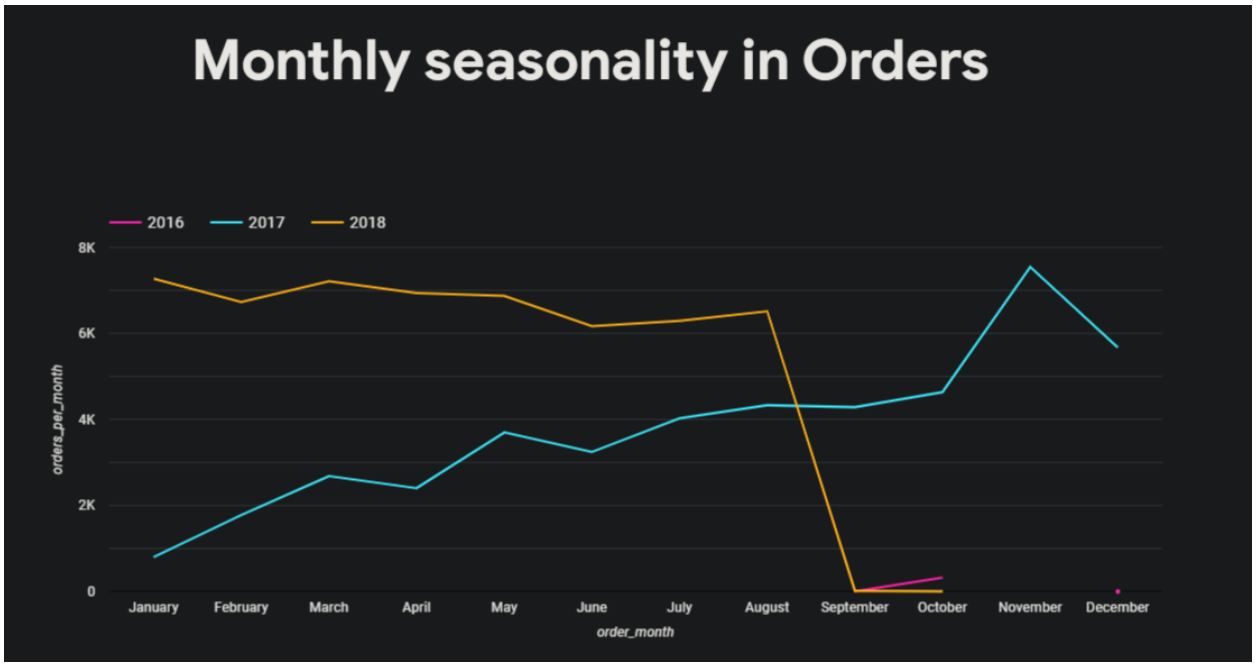
```
select
  EXTRACT(MONTH from order_purchase_timestamp) as order_month,
  EXTRACT(YEAR from order_purchase_timestamp) as order_year,
  count(distinct(order_id)) orders_per_month
from
  `target_sql.orders`
group by
  order_month, order_year
order by
  order_month, order_year;
```

Query Results :



Row	order_month	order_year	orders_per_month
1	1	2017	800
2	1	2018	7269
3	2	2017	1780
4	2	2018	6728
5	3	2017	2682
6	3	2018	7211
7	4	2017	2404
8	4	2018	6939
9	5	2017	3700
10	5	2018	6873

Chart :



Insights :

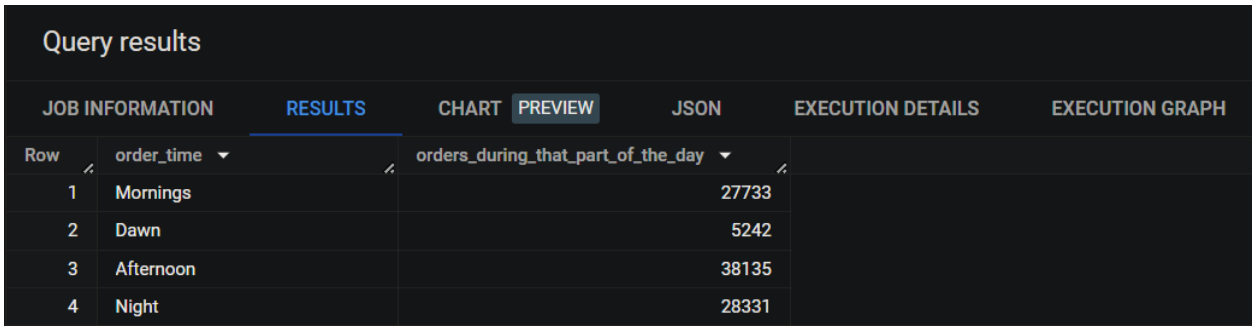
From the Results and the chart shown above we can clearly infer that there was a major drop in the number of orders placed in the month of September 2018.

#2.3 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 :

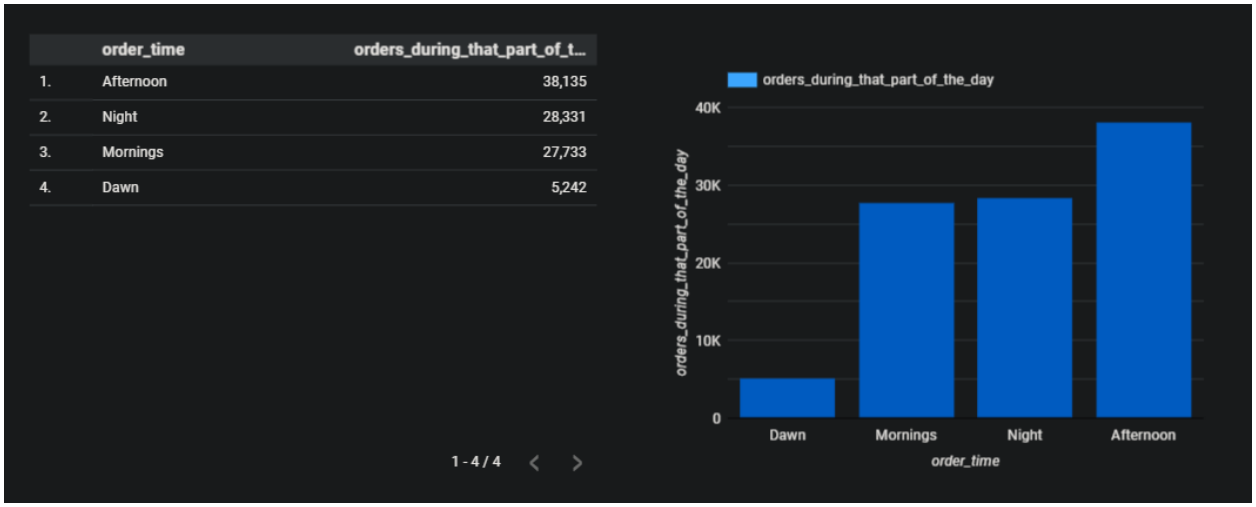
```sql
with hrwise_no_of_orders as
(select
  EXTRACT(HOUR from order_purchase_timestamp) as order_hour,
  count(distinct(order_id)) no_of_orders
from
  `target_sql.orders`
group by
  order_hour
order by
  order_hour)

select
  case when order_hour between 0 and 6 then 'Dawn'
       when order_hour between 7 and 12 then 'Mornings'
       when order_hour between 13 and 18 then 'Afternoon'
       else 'Night'
  end as order_time,
  sum(no_of_orders) as orders_during_that_part_of_the_day
from hrwise_no_of_orders
group by order_time;
```

Query Results :



Row	order_time	orders_during_that_part_of_the_day
1	Mornings	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

Chart :



Insights :

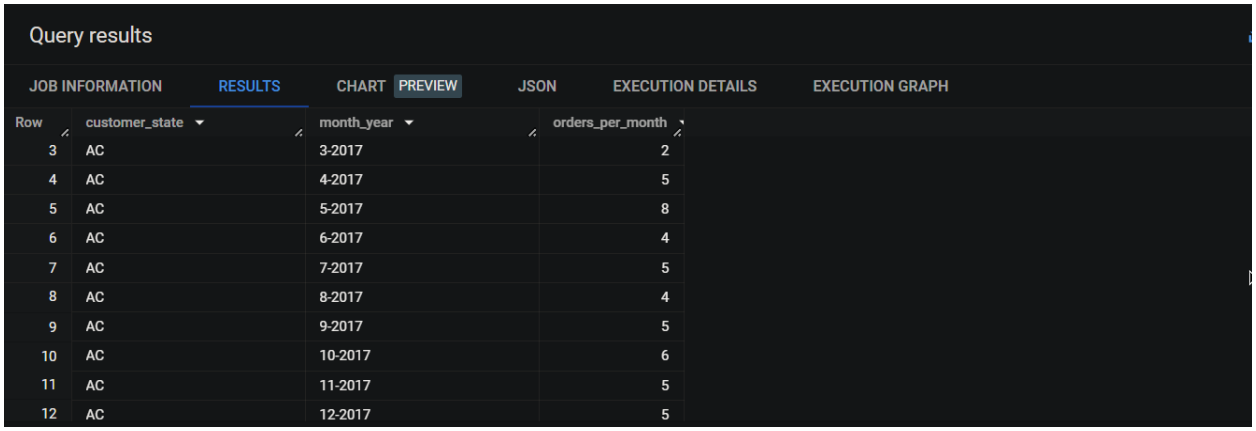
We can infer that the most number of orders are placed during the Afternoon i.e.between 13 to 18 hrs.

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

Query :

```
with mom_state as
(select
  cust.customer_state,
  format_datetime('%m-%Y', order_purchase_timestamp) mmyy,
  EXTRACT(YEAR from ord.order_purchase_timestamp) order_year,
  count(distinct(ord.order_id)) orders_per_month
from
  `target_sql.orders` ord
join
  `target_sql.customers` cust
using
  (customer_id)
group by
  customer_state, mmyy, order_year
order by
  customer_state, mmyy, order_year)

select
  customer_state,
  mmyy,
  order_year,
  orders_per_month
from
  mom_state;
```

Query Results :



Row	customer_state	month_year	orders_per_month
3	AC	3-2017	2
4	AC	4-2017	5
5	AC	5-2017	8
6	AC	6-2017	4
7	AC	7-2017	5
8	AC	8-2017	4
9	AC	9-2017	5
10	AC	10-2017	6
11	AC	11-2017	5
12	AC	12-2017	5

#3.2 How are the customers distributed across all the states?

Query :

```
select
  customer_state,
  count(customer_id) as no_of_customers
from
  `target_sql.customers`
group by
  customer_state
order by
  Customer_state;
```

Query Results:

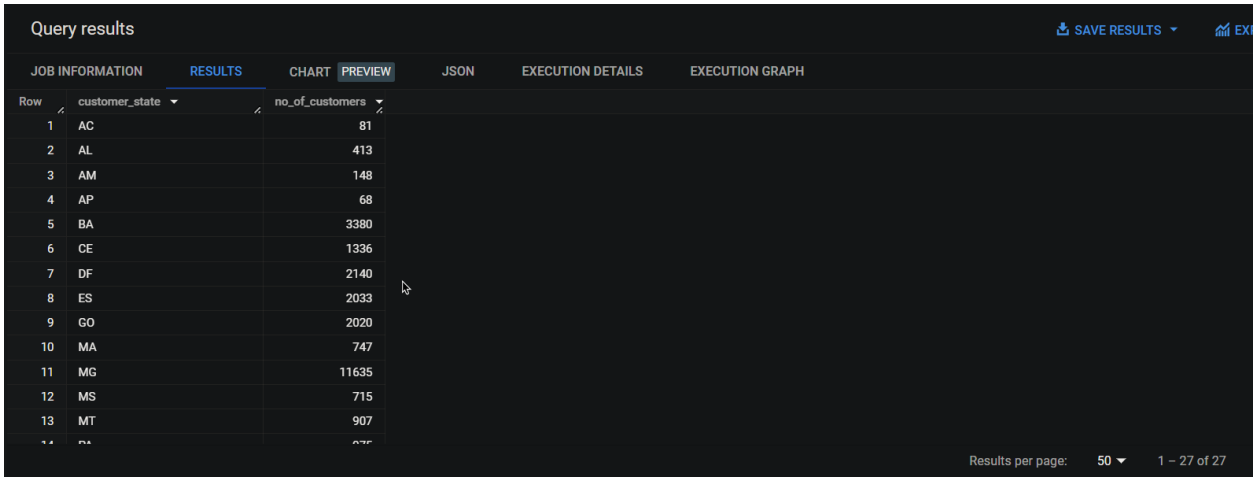
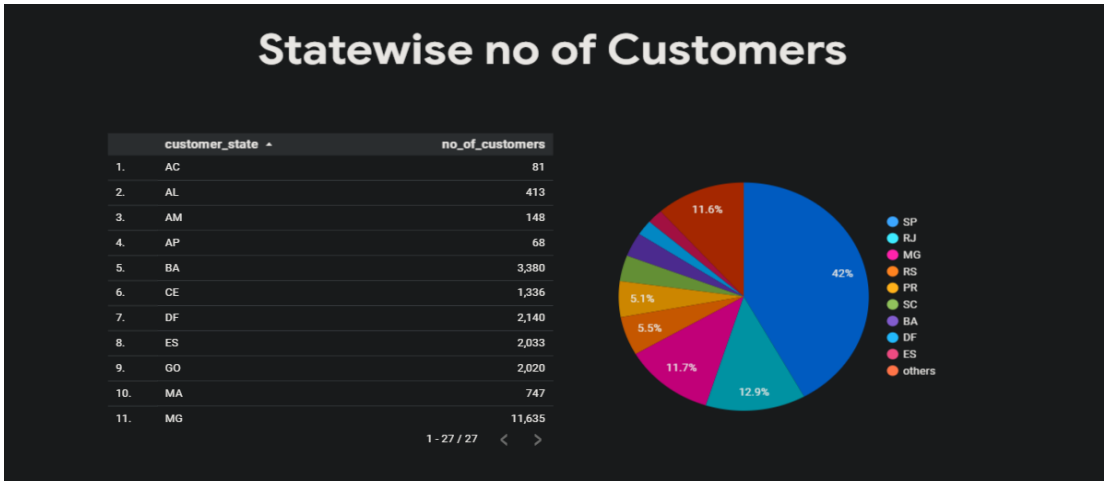


Chart :



Insights :

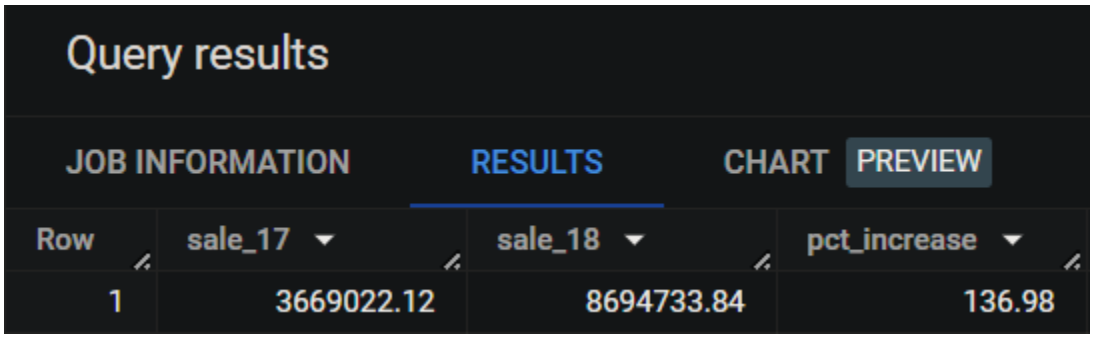
We can see that the people from the state of SP shop the most at Target Brazil, the state of SP is alone responsible for 42% of the customer base of Target Brazil.

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

Query :

```
with ms17 as
(select
  EXTRACT(YEAR from ord.order_purchase_timestamp) year_17,
  round(sum(pay.payment_value),2) sale_17
from
  `target_sql.orders` ord
join
  `target_sql.payments` pay
using
  (order_id)
where
  (EXTRACT(YEAR from ord.order_purchase_timestamp) = 2017) and (EXTRACT(MONTH from
ord.order_purchase_timestamp) between 01 and 08)
group by
  year_17),

ms18 as
(select
  EXTRACT(YEAR from ord.order_purchase_timestamp) year_18,
  round(sum(pay.payment_value),2) sale_18
from
  `target_sql.orders` ord
join
  `target_sql.payments` pay
using
  (order_id)
where
  (EXTRACT(YEAR from ord.order_purchase_timestamp) = 2018) and (EXTRACT(MONTH from
ord.order_purchase_timestamp) between 01 and 08)
group by
  year_18)

select
  sale_17, sale_18,
  round(((sale_18-sale_17)/sale_17)*100,2) as pct_increase
from  ms17 cross join ms18;
```

Query Results :



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

Query :

```
select
  cust.customer_state,
  round(sum(pay.payment_value),2) total_order_price,
  round(avg(pay.payment_value),2) avg_order_price
from
  `target_sql.customers` cust
left join
  `target_sql.orders` ord using (customer_id)
join
  `target_sql.payments` pay using (order_id)
group by
  cust.customer_state
order by
  Cust.customer_state
```

Query Results:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	customer_state	total_order_price	avg_order_price
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86
11	MG	1872257.26	154.71

Chart :

Insights :

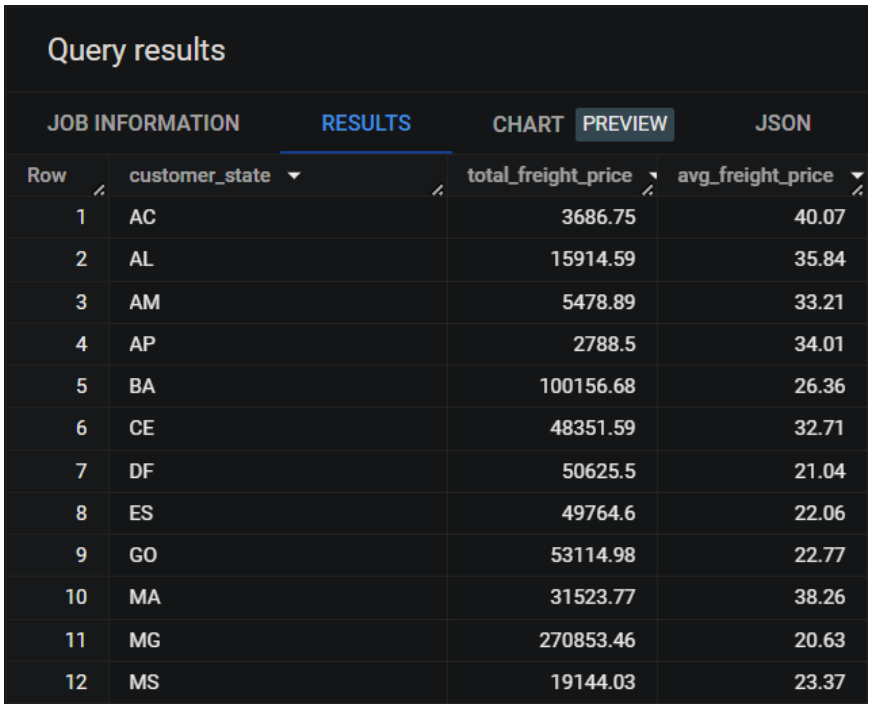
On an average, customers from each state shop for more than 120 Reals.

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

Query :

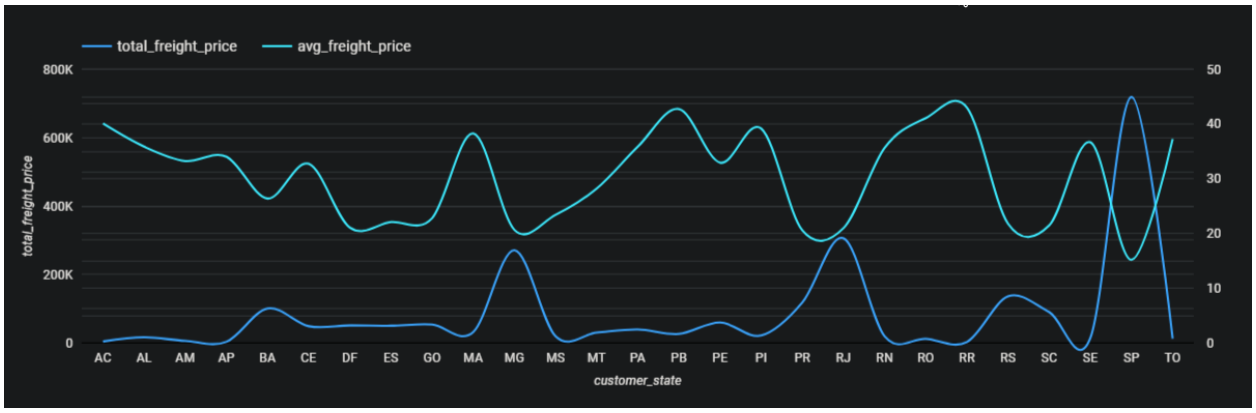
```
select
  cust.customer_state,
  round(sum(oi.freight_value),2) total_freight_price,
  round(avg(oi.freight_value),2) avg_freight_price
from
  `target_sql.customers` cust
left join
  `target_sql.orders` ord using (customer_id)
join
  `target_sql.order_items` oi using (order_id)
group by
  cust.customer_state
order by
  Cust.customer_state
```

Query Results :



Row	customer_state	total_freight_price	avg_freight_price
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37

Chart:



Insights :

On an average, customers from each state a freight value of for more than 25 Reals.

#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.

Query :

```
select
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)
time_to_deliver,
  DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date, DAY)
diff_estimated_delivery
from
  `target_sql.orders`
```

Query Results :

Query results

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

Row	order_id	time_to_deliver	diff_estimated_delive
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5

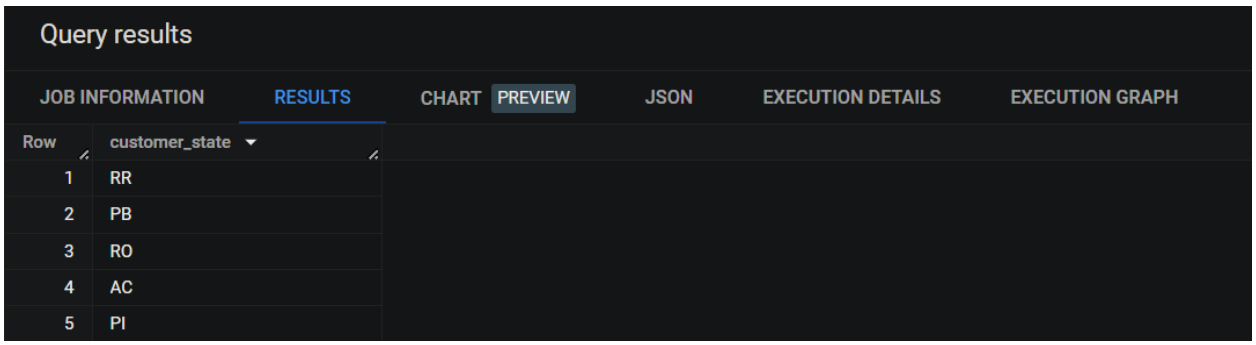
#5.2 Find out the top 5 states with the highest & lowest average freight value.

TOP 5 STATES WITH HIGHEST FREIGHT VALUE :

Query :

```
with highest_fr as
(select
  cus.customer_state,
  dense_rank() over(order by round(avg(ordi.freight_value),2) desc) state_rank
from
  `target_sql.customers` cus
join
  `target_sql.orders` ord using (customer_id)
join
  `target_sql.order_items` ordi using (order_id)
group by
  cus.customer_state
order by
  state_rank)
select customer_state from highest_fr where state_rank between 1 and 5;
```

Query Results :

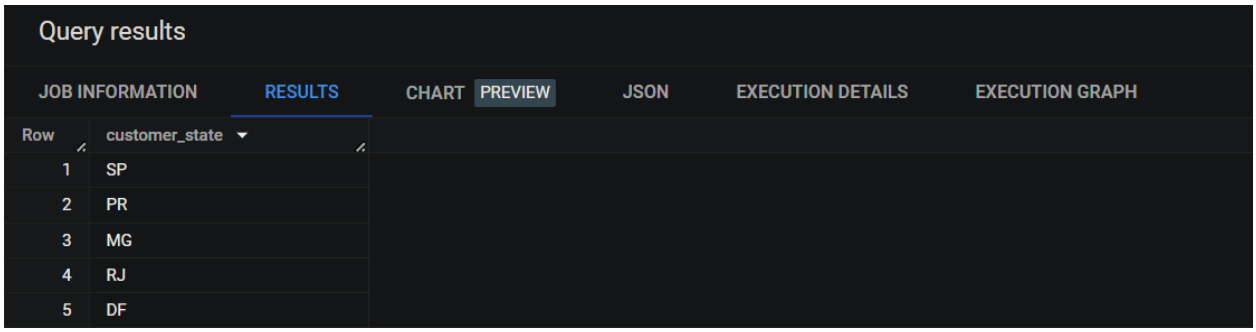


TOP 5 STATES WITH LOWEST FREIGHT VALUE :

Query :

```sql
with lowest_fr as
(select
  cus.customer_state,
  dense_rank() over(order by round(avg(ordi.freight_value),2) asc) state_rank
from
  `target_sql.customers` cus
join
  `target_sql.orders` ord using (customer_id)
join
  `target_sql.order_items` ordi using (order_id)
group by
  cus.customer_state
order by
  state_rank)
select customer_state from lowest_fr where state_rank between 1 and 5;
```

Query Results :



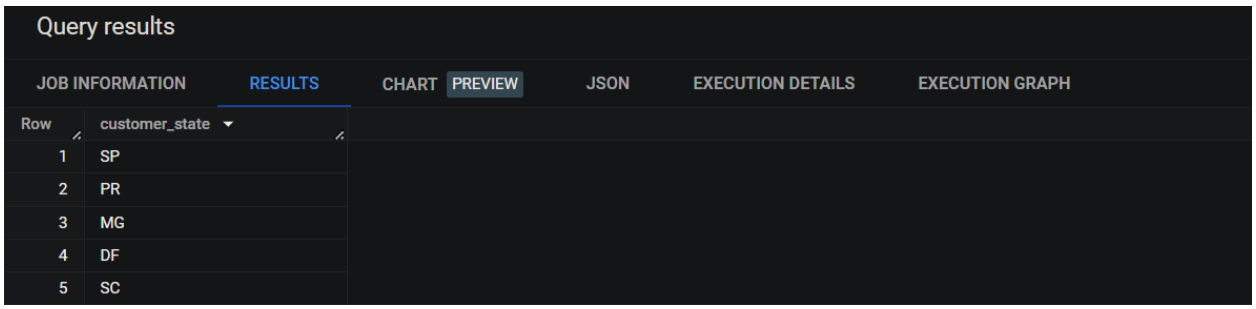
#5.3 Find out the top 5 states with the highest & lowest average delivery time.

TOP 5 STATES WITH LOWEST DELIVERY TIMES :

Query :

```
with lowest_dt_states as
(select
  cus.customer_state,
  round(avg(DATE_DIFF(ord.order_delivered_customer_date, ord.order_purchase_timestamp,
DAY)),2) time_to_deliver,
  dense_rank() over(order by round(avg(DATE_DIFF(ord.order_delivered_customer_date,
ord.order_purchase_timestamp, DAY)),2) ) delivery_time_rank
from
  `target_sql.customers` cus
join
  `target_sql.orders` ord using (customer_id)
group by
  cus.customer_state
order by
  delivery_time_rank)
select customer_state from lowest_dt_states where delivery_time_rank between 1 and 5;
```

Query Results :



Insights :

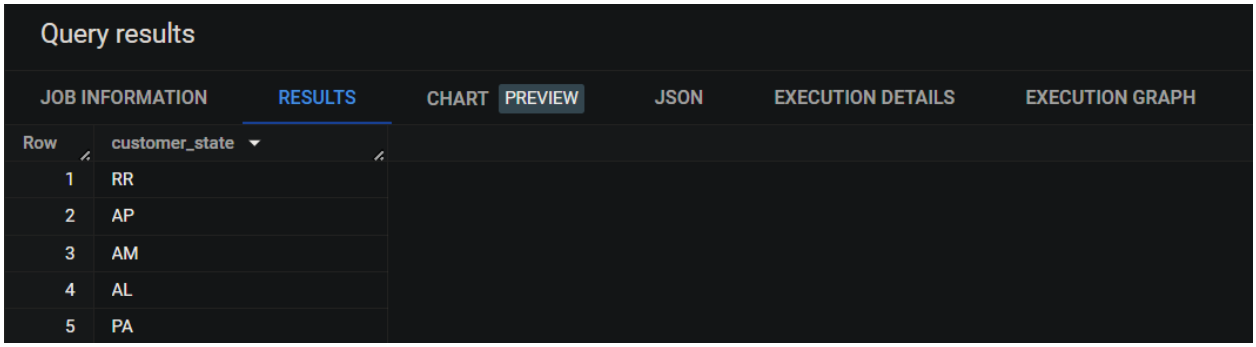
On an average, the orders are delivered the quickest in the state of SP.

TOP 5 STATES WITH HIGHEST DELIVERY TIMES :

Query :

```
with highest_dt_states as
(select
  cus.customer_state,
  round(avg(DATE_DIFF(ord.order_delivered_customer_date, ord.order_purchase_timestamp,
DAY)),2) avg_time_to_deliver,
  dense_rank() over(order by round(avg(DATE_DIFF(ord.order_delivered_customer_date,
ord.order_purchase_timestamp, DAY)),2) desc) delivery_time_rank
from
  `target_sql.customers` cus
join
  `target_sql.orders` ord using (customer_id)
group by
  cus.customer_state
order by
  delivery_time_rank)
select customer_state from highest_dt_states where delivery_time_rank between 1 and 5;
```

Query Results :



Insights :

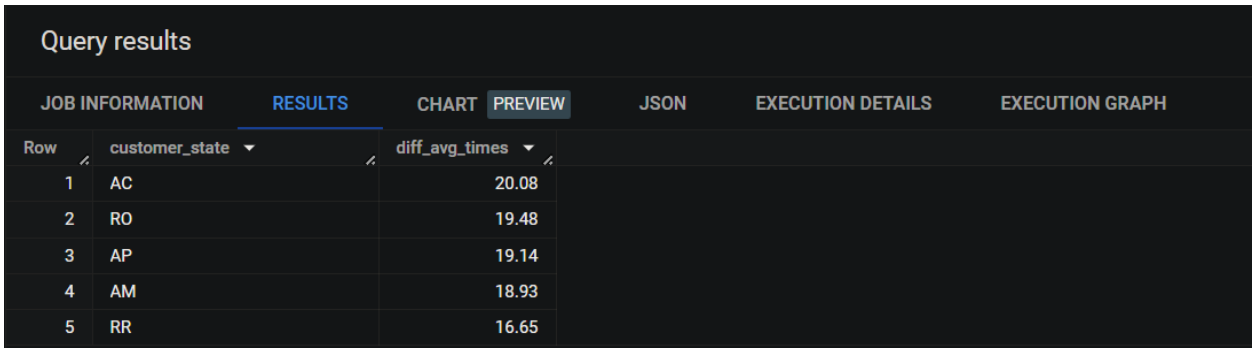
On an average, the orders are delivered the slowest in the state of RR.

#5.4 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 statewise_avg_times as
(select
  cus.customer_state,
  round(avg(DATE_DIFF(ord.order_delivered_customer_date, ord.order_purchase_timestamp,
DAY)),2) avg_act_time_to_deliver,
  round(avg(DATE_DIFF(ord.order_estimated_delivery_date, ord.order_purchase_timestamp,
DAY)),2) avg_est_time_to_deliver,
from
  `target_sql.customers` cus
join
  `target_sql.orders` ord using (customer_id)
where
  ord.order_status = 'delivered'
group by
  cus.customer_state
order by
  cus.customer_state)
select customer_state,
round((statewise_avg_times.avg_est_time_to_deliver-statewise_avg_times.avg_act_time_to
_deliver),2) diff_avg_times
from statewise_avg_times order by diff_avg_times desc limit 5;
```

Query Results :



Row	customer_state	diff_avg_times
1	AC	20.08
2	RO	19.48
3	AP	19.14
4	AM	18.93
5	RR	16.65

Insights :

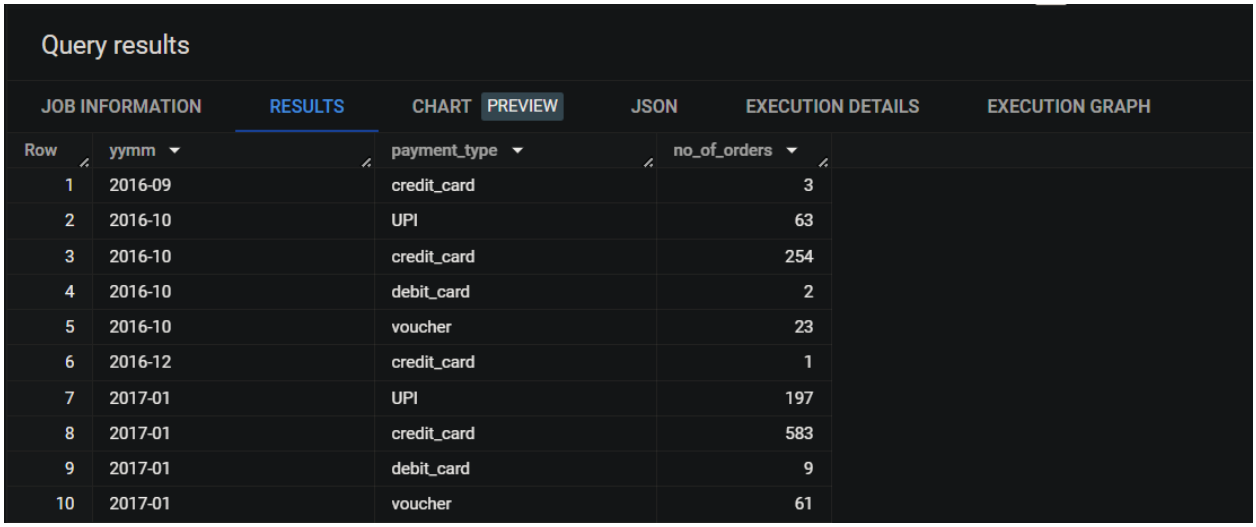
On an average, usually the orders are delivered the quickest relative to the estimated delivery dates in the Brazilian state of AC.

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

Query :

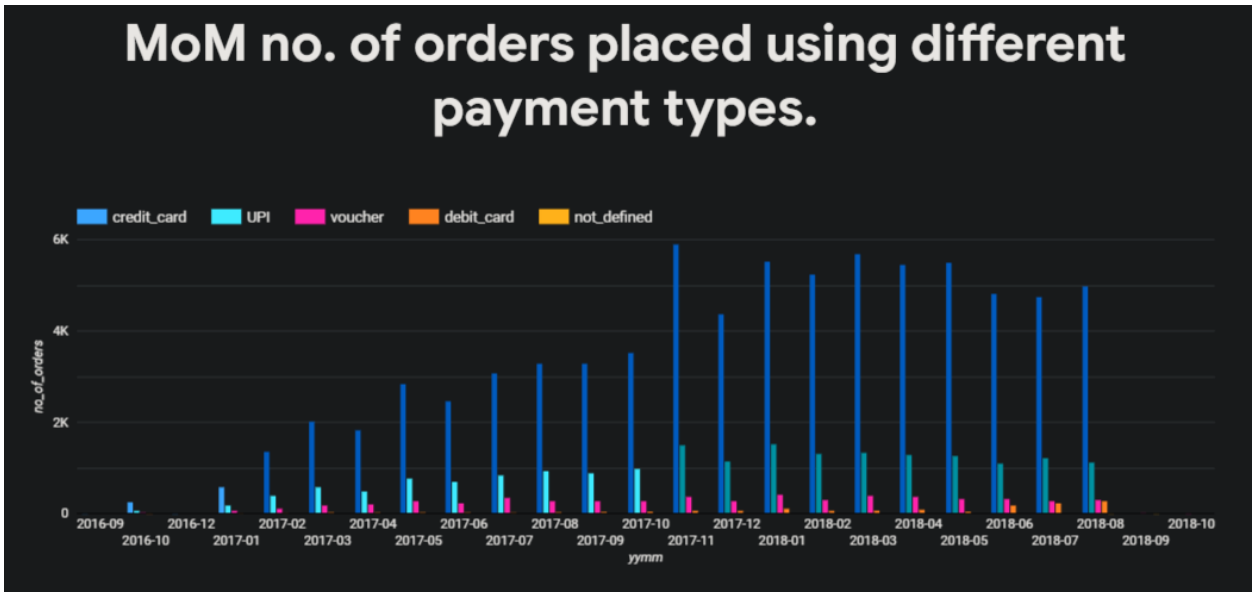
```
select
  format_datetime('%Y-%m', order_purchase_timestamp) yymm,
  payment_type,
  count(order_id) no_of_orders
from
  `target_sql.orders` ord
join
  `target_sql.payments` pay using (order_id)
group by
  1,2
order by
  1,2;
```

Query Results :



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

Chart :



Insights :

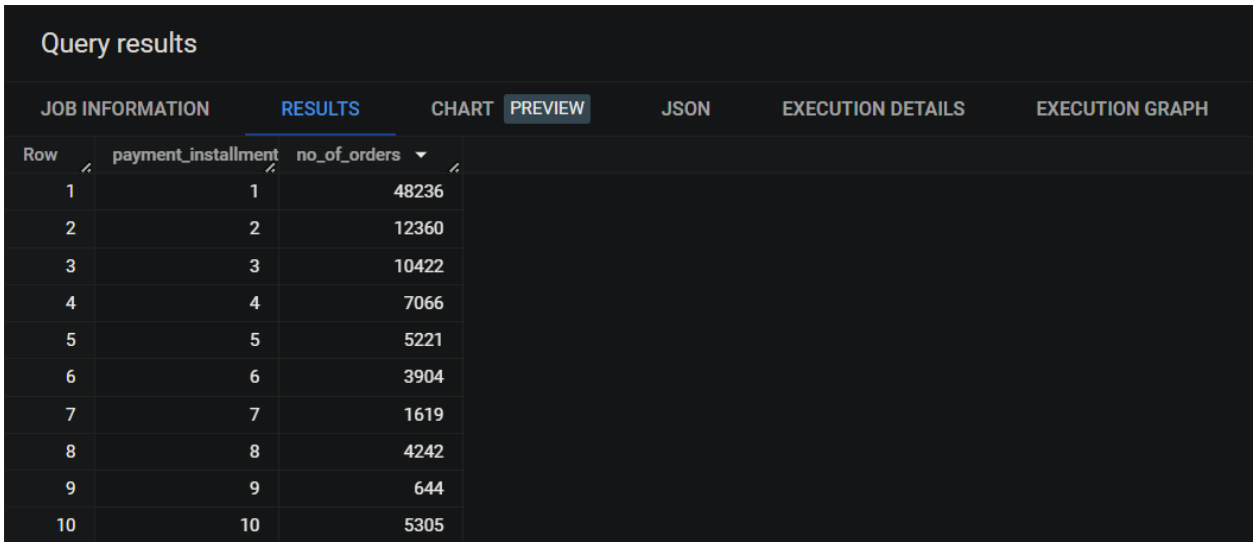
From the data generated above, it is evident that the shoppers at Target Brazil prefer to use CREDIT CARDS as the mode of payment.

#6.2 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)) no_of_orders
from target_sql.payments
where payment_installments >=1 and payment_sequential = 1
group by payment_installments
```

Query Results :



Query results

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

Row	payment_installment	no_of_orders
1	1	48236
2	2	12360
3	3	10422
4	4	7066
5	5	5221
6	6	3904
7	7	1619
8	8	4242
9	9	644
10	10	5305