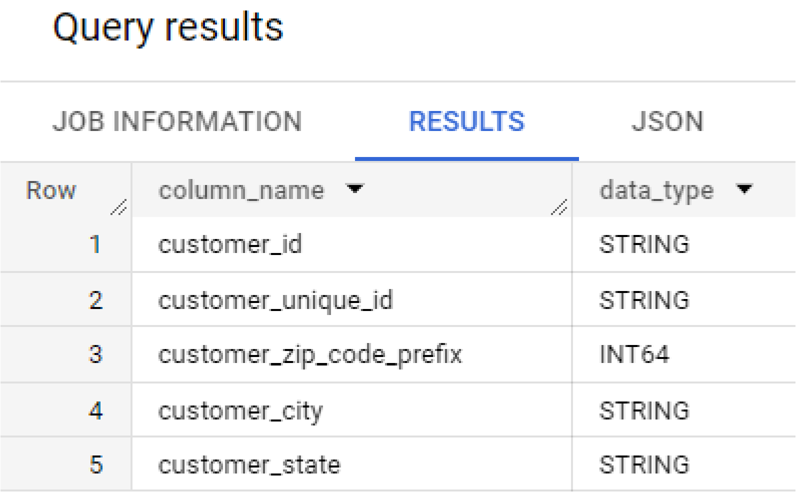
1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:**

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

   Query 1:

```
SELECT column_name, data_type
from target_brazil_market.INFORMATION_SCHEMA.COLUMNS
where table_name = "customers";
```



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

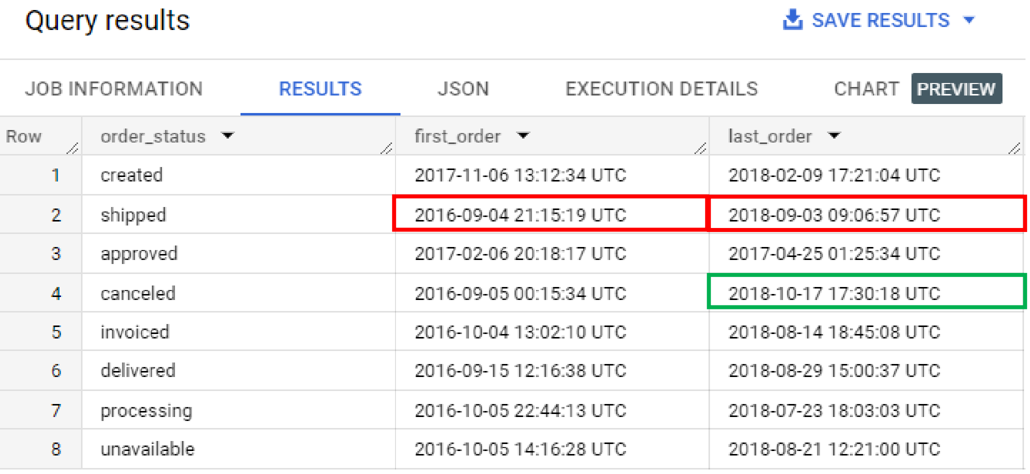
Figure 1.1.1: Data type of all columns in the customer table

- As per the requirement, we need the column names and their respective datatypes.
- In order to do that we can use the **INFORMATION_SCHEMA** as it provides information such as the name of a database or table, the data type of a column, or access privileges.

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

Query 1:

```
select order_status, min(order_purchase_timestamp) as first_order,
max(order_purchase_timestamp) as last_order
from `target_brazil_market.orders`
group by order_status
```



Row	order_status	first_order	last_order
1	created	2017-11-06 13:12:34 UTC	2018-02-09 17:21:04 UTC
2	shipped	2016-09-04 21:15:19 UTC	2018-09-03 09:06:57 UTC
3	approved	2017-02-06 20:18:17 UTC	2017-04-25 01:25:34 UTC
4	canceled	2016-09-05 00:15:34 UTC	2018-10-17 17:30:18 UTC
5	invoiced	2016-10-04 13:02:10 UTC	2018-08-14 18:45:08 UTC
6	delivered	2016-09-15 12:16:38 UTC	2018-08-29 15:00:37 UTC
7	processing	2016-10-05 22:44:13 UTC	2018-07-23 18:03:03 UTC
8	unavailable	2016-10-05 14:16:28 UTC	2018-08-21 12:21:00 UTC

Figure 1.2.1: shows the fist and last order in the given dataset.

- To meet the specified criteria, we utilize the min and max aggregate functions to identify the initial and ultimate orders recorded in the provided database.
- However, it's important to note that an order's lifecycle commences when it is "created" and concludes upon being "shipped".
- Additionally, considering that products cannot be cancelled post-delivery, I have highlighted the relevant dates with a red outline for reference.

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

Query 1:

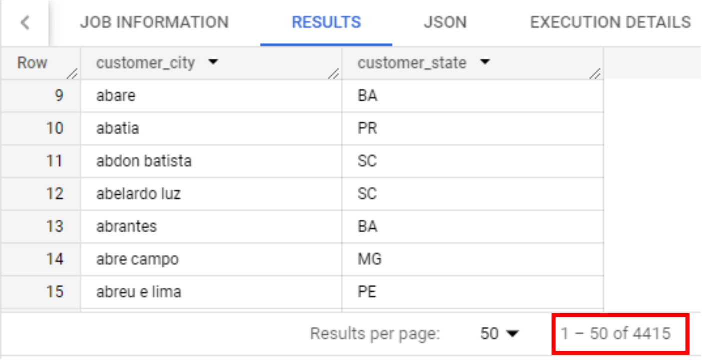
```
select distinct customer_city, customer_state
from `target_brazil_market.customers`
-- where customer_city is null or customer_state is null

UNION distinct

-- select distinct geolocation_city, geolocation_state
-- from `target_brazil_market.geolocation`
-- -- where geolocation_city is null or geolocation_state is null

-- Union distinct

select distinct seller_city, seller_state
from `target_brazil_market.sellers`
-- where seller_city is null or seller_state is null
order by customer_city asc
```



Row	customer_city	customer_state
9	abare	BA
10	abatia	PR
11	abdon batista	SC
12	abelardo luz	SC
13	abrantes	BA
14	abre campo	MG
15	abreu e lima	PE

Results per page: 50 1 – 50 of 4415

Figure 1.3.1: Record showing the No. of cities and countries in our record.

- As we are considering just the cities and countries who ordered during the given dataset, we'll be excluding the *"geolocation table"* since it's a library with id's of cities and countries.
- Total number of cities are 4415 and states are 27.

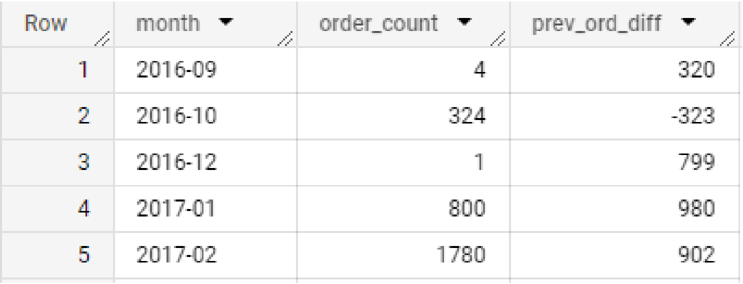
2. In-depth Exploration:

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

Query 1:

```
with x as
(select format_date("%Y-%m",date_trunc(order_purchase_timestamp,month)) as
month, count(order_id) as order_count
from `target_brazil_market.orders`
-- where order_status = 'shipped'
-- and order_delivered_customer_date is not null
group by 1
order by 1)

select month, order_count, (lead(order_count,1) over(order by month) -
x.order_count) as prev_ord_diff
from x
order by 1
```



Row	month	order_count	prev_ord_diff
1	2016-09	4	320
2	2016-10	324	-323
3	2016-12	1	799
4	2017-01	800	980
5	2017-02	1780	902

Figure 2.1.1: Purchasing Trends over years.

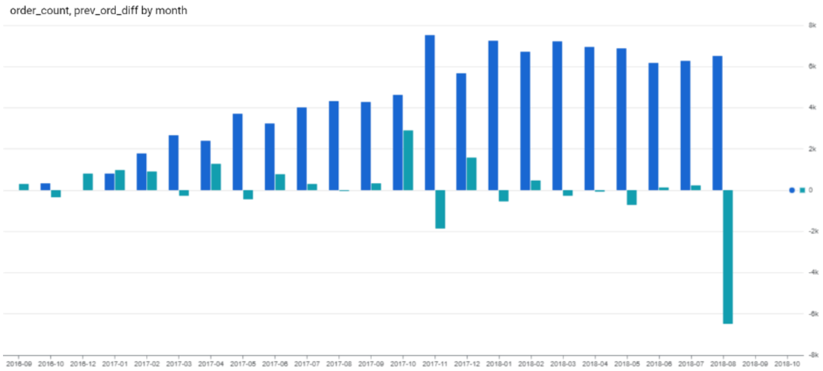


Figure 2.1.2: Schematic representation of the order records

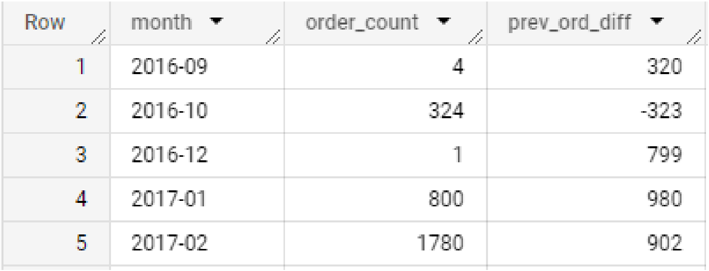
- The number of orders gradually increases until OCT 2017 and after that there a rapid rise on NOV 2017.
- From that point the number of orders keeps fluctuating until OCT 2018 and the sales drops almost to zero.

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

<u>Query 1:</u>

```
with x as
(select format_date("%Y-%m",date_trunc(order_purchase_timestamp,month)) as
month, count(order_id) as order_count
from `target_brazil_market.orders`
-- where order_status = 'shipped'
-- and order_delivered_customer_date is not null
group by 1
order by 1)

select month, order_count, (lead(order_count,1) over(order by month) -
x.order_count) as prev_ord_diff
from x
order by 1
```



Row	month	order_count	prev_ord_diff
1	2016-09	4	320
2	2016-10	324	-323
3	2016-12	1	799
4	2017-01	800	980
5	2017-02	1780	902

Figure 2.2.1: Purchasing Trends over years.

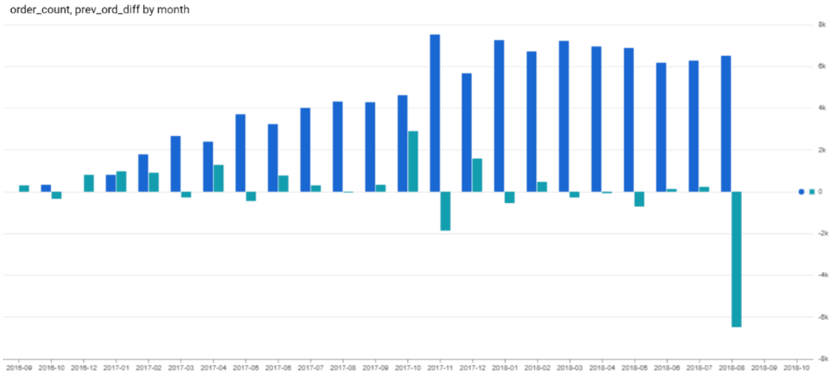


Figure 2.2.2: Schematic representation of the order records

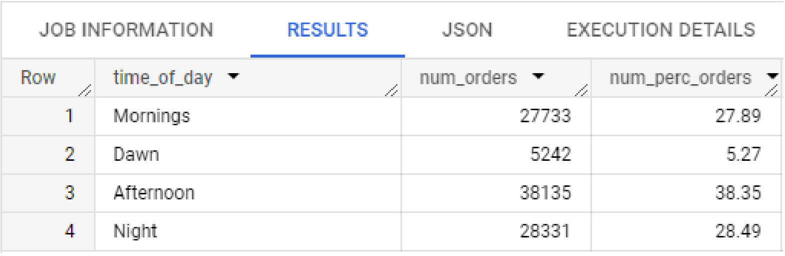
- The orders increased abruptly on the NOV 2017 apart from there are not any significant events which can be noticed in the chart.

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 1:

```
select case
when extract(hour from order_purchase_timestamp) < 7 then 'Dawn'
when extract(hour from order_purchase_timestamp) < 13 then 'Mornings'
when extract(hour from order_purchase_timestamp) < 19 then 'Afternoon'
else 'Night'
end as time_of_day, count(customer_id) as num_orders,
round((count(customer_id)/99441)*100,2) as num_perc_orders
from `target_brazil_market.orders`
group by 1
```



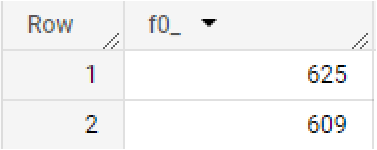
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	time_of_day	num_orders	num_perc_orders
1	Mornings	27733	27.89
2	Dawn	5242	5.27
3	Afternoon	38135	38.35
4	Night	28331	28.49

Figure 2.3.1: Orders ratio in terms of time of day.

Query 2:

```
select count(*)
from `target_brazil_market.orders`
where order_status = 'unavailable' and order_delivered_carrier_date is null
and order_delivered_customer_date is null
union all
select count(*)
from `target_brazil_market.orders`
where order_status = 'canceled'
```

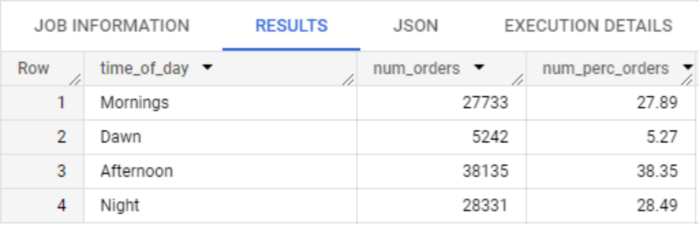


Row	f0_
1	625
2	609

Figure 2.3.2: Number of obsolete data to be excluded.

Query 3:

```
select case
when extract(hour from order_purchase_timestamp) < 7 then 'Dawn'
when extract(hour from order_purchase_timestamp) < 13 then 'Mornings'
when extract(hour from order_purchase_timestamp) < 19 then 'Afternoon'
else 'Night'
end as time_of_day, count(customer_id) as num_orders,
round((count(customer_id)/98207)*100,2) as num_perc_orders
from `target_brazil_market.orders`
where order_status not in ('unavailable','canceled')
group by 1
```



Row	time_of_day	num_orders	num_perc_orders
1	Mornings	27733	27.89
2	Dawn	5242	5.27
3	Afternoon	38135	38.35
4	Night	28331	28.49

Figure 2.3.3: Orders ratio in terms of time of day excluding the obsolete data.

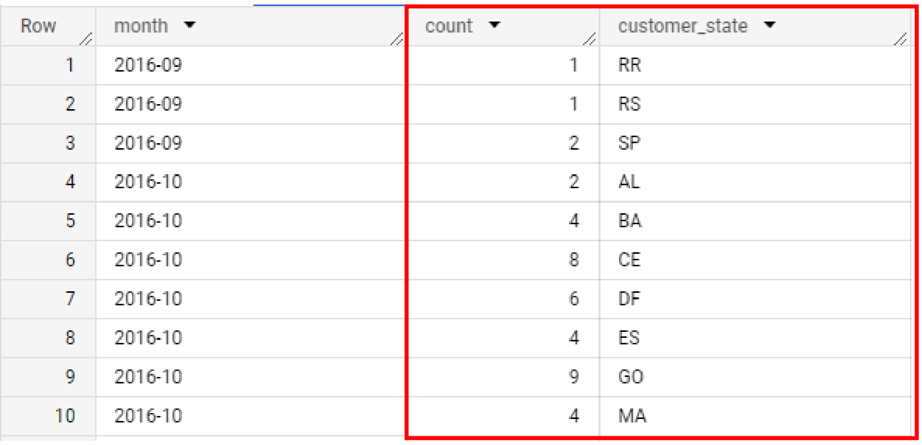
- Taking all the data into account, 38% percent of sales happens during the afternoon.
- Excluding the amount of data with order status as *“cancelled”* and *“unavailable”* which is 1234 rows. Still, most of the sales happens during afternoon which constitutes 38% of the overall sales.

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

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

Query 1:

```
select format_date('%Y-%m',o.order_purchase_timestamp)as month,
count(o.customer_id) as count , c.customer_state
from `target_brazil_market.orders` as o
inner join `target_brazil_market.customers` as c
on o.customer_id = c.customer_id
group by 1, 3
order by 1, 3
```



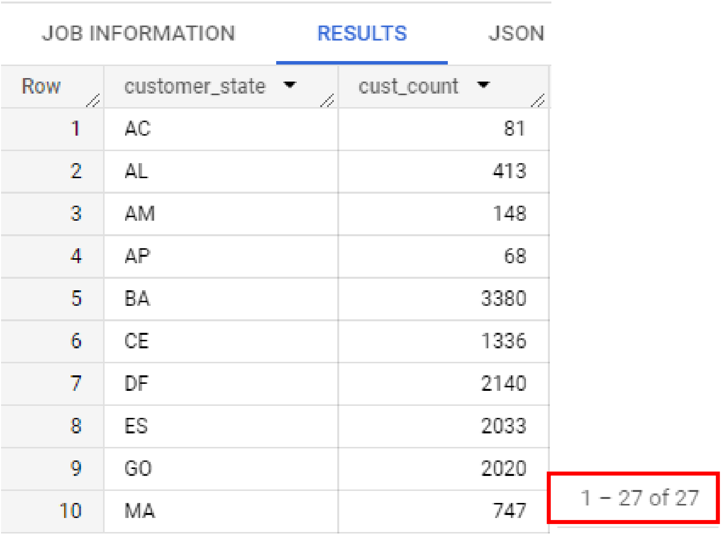
Row	month	count	customer_state
1	2016-09	1	RR
2	2016-09	1	RS
3	2016-09	2	SP
4	2016-10	2	AL
5	2016-10	4	BA
6	2016-10	8	CE
7	2016-10	6	DF
8	2016-10	4	ES
9	2016-10	9	GO
10	2016-10	4	MA

Figure 3.1.1: month-on-month order per each state

3.2 How are the customers distributed across all the states?

Query 1:

```
select customer_state, count(customer_id) as cust_count
from `target_brazil_market.customers`
group by 1
order by 1
```



Row	customer_state	cust_count
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

Figure 3.2.1: Customers distribution across each state

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

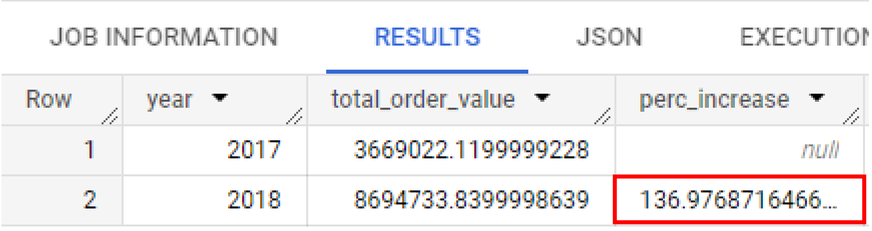
4.1 Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).You can use the "payment_value" column in the payments table to get the cost of orders.

Query 1:

- There was 136.97 % increase from the year 2017 t0 2018 with considering the given parameters.

```
with y as
(select extract(year from o.order_purchase_timestamp) as year,
sum(p.payment_value) as total_order_value
from `target_brazil_market.payments` as p
inner join `target_brazil_market.orders` as o
on  p.order_id = o.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 1
order by 1)

select year, total_order_value, ((total_order_value - lag(total_order_value)
over(order by year))/ lag(total_order_value) over(order by year))*100 as
perc_increase
from y
order by 1
```



Row	year	total_order_value	perc_increase
1	2017	3669022.1199999228	null
2	2018	8694733.8399998639	136.9768716466...

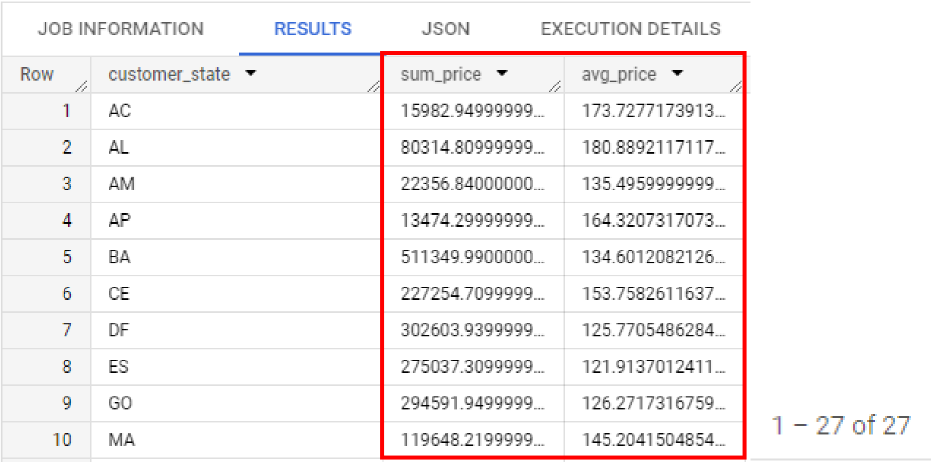
Figure 4.1.1: Increase of orders in percentage.

- There was 136.97 % increase from the year 2017 t0 2018 with considering the given parameters.

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

Query 1:

```
select c.customer_state, sum(oi.price) as sum_price, avg(oi.price) as
avg_price
FROM `target_brazil_market.order_items` oi
left join `target_brazil_market.orders` o
on o.order_id = oi.order_id
left join `target_brazil_market.customers` c
on o.customer_id = c.customer_id
group by c.customer_state
order by 1
```



Row	customer_state	sum_price	avg_price
1	AC	15982.94999999...	173.7277173913...
2	AL	80314.80999999...	180.8892117117...
3	AM	22356.84000000...	135.4959999999...
4	AP	13474.29999999...	164.3207317073...
5	BA	511349.9900000...	134.6012082126...
6	CE	227254.7099999...	153.7582611637...
7	DF	302603.9399999...	125.7705486284...
8	ES	275037.3099999...	121.9137012411...
9	GO	294591.9499999...	126.2717316759...
10	MA	119648.2199999...	145.2041504854...

1 – 27 of 27

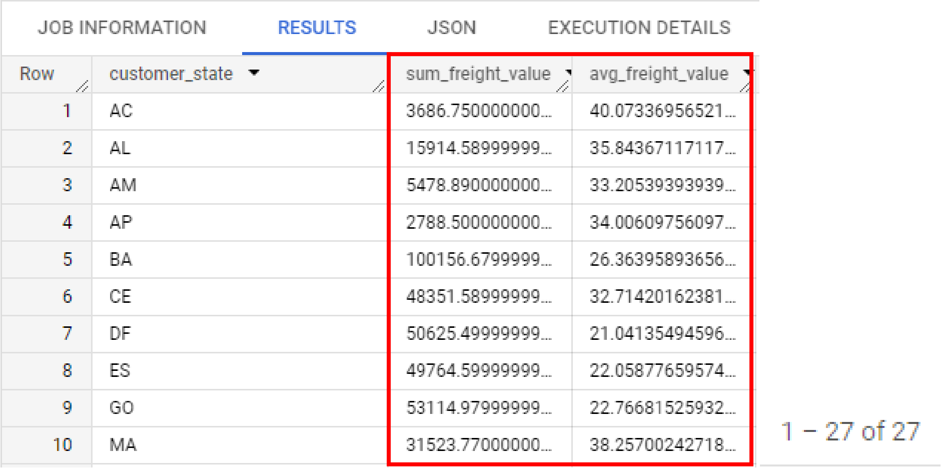
Figure 4.2.1: Sum and avg of order_price across each state

- I have included a sample of 10 rows but the total number of records includes all 27 states.

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

<u>Query 1:</u>

```
select c.customer_state,
sum(oi.freight_value) as sum_freight_value,
avg(oi.freight_value) as avg_freight_value
from `target_brazil_market.order_items` oi
left join `target_brazil_market.orders` o
on o.order_id = oi.order_id
left join `target_brazil_market.customers` c
on o.customer_id = c.customer_id
group by c.customer_state
order by 1
```



Row	customer_state	sum_freight_value	avg_freight_value
1	AC	3686.750000000...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.890000000...	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...

1 – 27 of 27

Figure 4.3.1: Sum and avg of freight_value across each state

- I have included a sample of 10 rows but the actual records includes all 27 states.

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

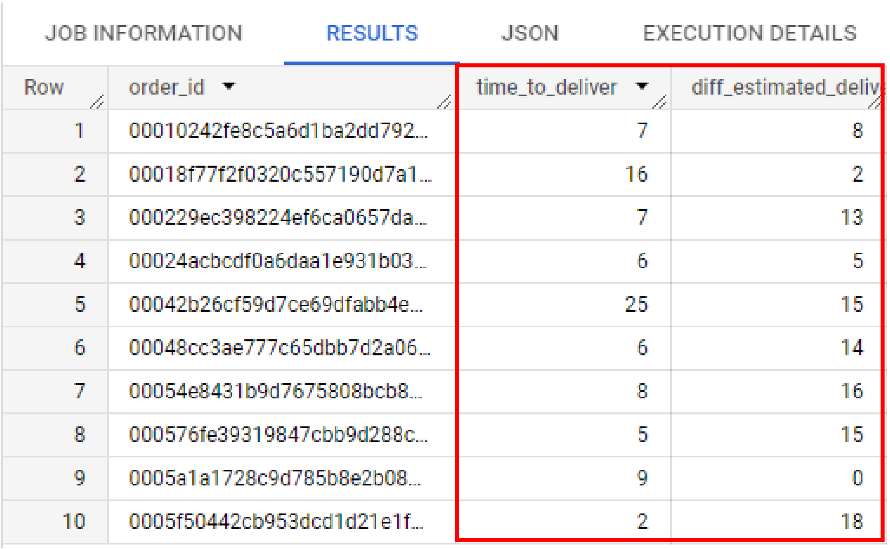
   5.1 Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
   Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
   Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

Query 1:

```
select order_id, datetime_diff(order_delivered_customer_date,
order_purchase_timestamp, day) as time_to_deliver
, datetime_diff(order_estimated_delivery_date,
order_delivered_customer_date, day) as diff_estimated_delivery,
from `target_brazil_market.orders`
where order_delivered_customer_date is not null
order by 1
limit 10
```



Row	order_id	time_to_deliver	diff_estimated_deliv
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

Figure 5.1.1: Order efficiency report

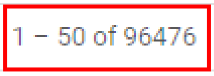

1 – 50 of 96476

Figure 5.1.2: Data type of all columns in the customer table

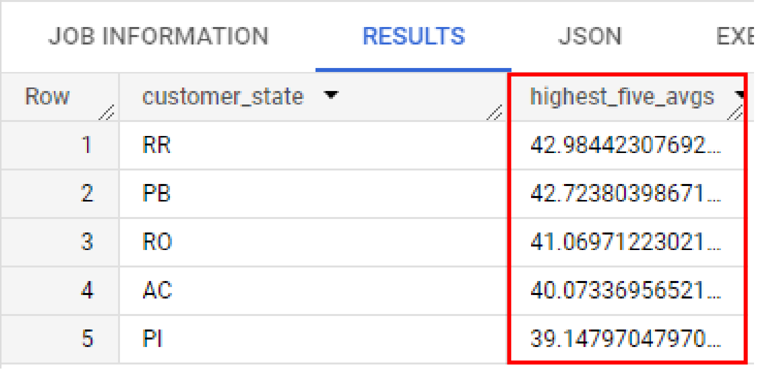
- The *null value of "order_delivered_customer_date"* implies the order is either cancelled or other technical issues.
- There excluding those data from 99441 records we get 96476 records.

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

Query 1:

```
with CTE as
(select c.customer_state, avg(oi.freight_value) as avg_values
from `target_brazil_market.order_items` as oi
left join `target_brazil_market.orders` as o
on oi.order_id = o.order_id
inner join `target_brazil_market.customers` as c
on o.customer_id = c.customer_id
group by 1)
```

```
select CTE.customer_state, CTE.avg_values as highest_five_avgs
from CTE
order by 2 desc
limit 5
```



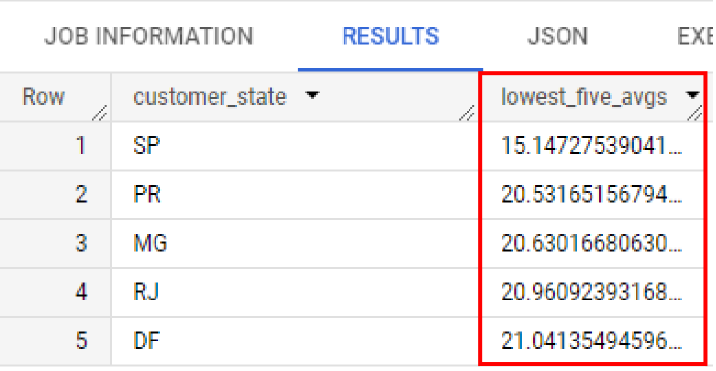
Row	customer_state	highest_five_avgs
1	RR	42.98442307692…
2	PB	42.72380398671…
3	RO	41.06971223021…
4	AC	40.07336956521…
5	PI	39.14797047970…

Figure 5.2.1: states with highest avg's of freight value.

Query 2:

```
with CTE as
(select c.customer_state, avg(oi.freight_value) as avg_values
from `target_brazil_market.order_items` as oi
left join `target_brazil_market.orders` as o
on oi.order_id = o.order_id
inner join `target_brazil_market.customers` as c
on o.customer_id = c.customer_id
group by 1)

select CTE.customer_state, CTE.avg_values as lowest_five_avgs
from CTE
order by 2
limit 5
```



Row	customer_state	lowest_five_avgs
1	SP	15.14727539041…
2	PR	20.53165156794…
3	MG	20.63016680630…
4	RJ	20.96092393168…
5	DF	21.04135494596…

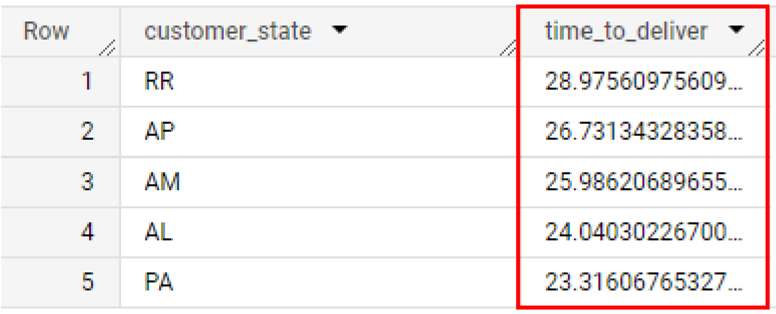
Figure 5.2.2: states with lowest avg's of freight value.

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

Query 1:

```
with CTE as
(select c.customer_state, avg(datetime_diff(order_delivered_customer_date,
order_purchase_timestamp, day)) as time_to_deliver
from `target_brazil_market.orders` as o
inner join `target_brazil_market.customers` as c
on o.customer_id = c.customer_id
where order_delivered_customer_date is not null
group by 1)
```

```
select CTE.customer_state, CTE.time_to_deliver
from CTE
order by 2 desc
limit 5
```



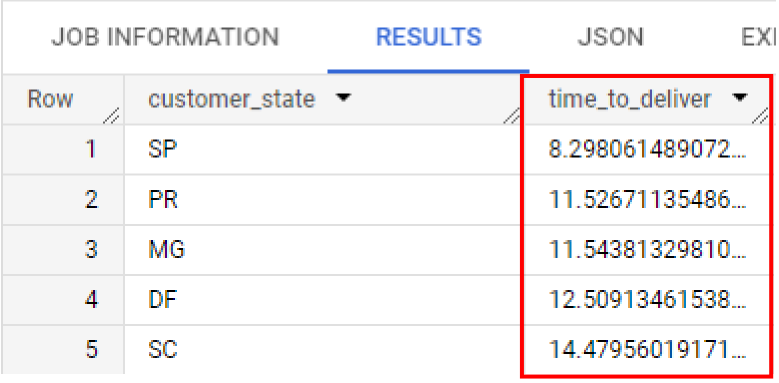
Row	customer_state	time_to_deliver
1	RR	28.97560975609…
2	AP	26.73134328358…
3	AM	25.98620689655…
4	AL	24.04030226700…
5	PA	23.31606765327…

Figure 5.3.1: states with highest avg's of delivery time.

Query 2:

```
with CTE as
(select c.customer_state, avg(datetime_diff(order_delivered_customer_date,
order_purchase_timestamp, day)) as time_to_deliver
from `target_brazil_market.orders` as o
inner join `target_brazil_market.customers` as c
on o.customer_id = c.customer_id
where order_delivered_customer_date is not null
group by 1)

select CTE.customer_state, CTE.time_to_deliver
from CTE
order by 2
limit 5
```



Row	customer_state	time_to_deliver
1	SP	8.298061489072…
2	PR	11.52671135486…
3	MG	11.54381329810…
4	DF	12.50913461538…
5	SC	14.47956019171…

Figure 5.3.2: states with lowest avg's of delivery time.

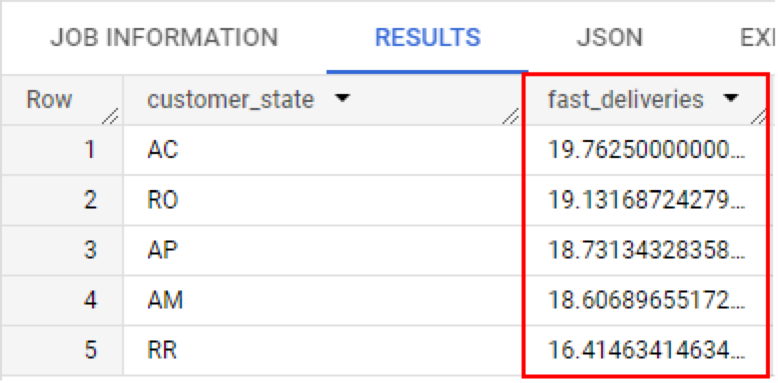
5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

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 1:

```
select c.customer_state
, avg(datetime_diff(order_estimated_delivery_date,
order_delivered_customer_date, day)) as fast_deliveries
from `target_brazil_market.orders` as o
inner join `target_brazil_market.customers` as c
on o.customer_id = c.customer_id
```

```
where order_delivered_customer_date is not null
group by 1
order by 2 desc
limit 5
```



Row	customer_state	fast_deliveries
1	AC	19.76250000000...
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

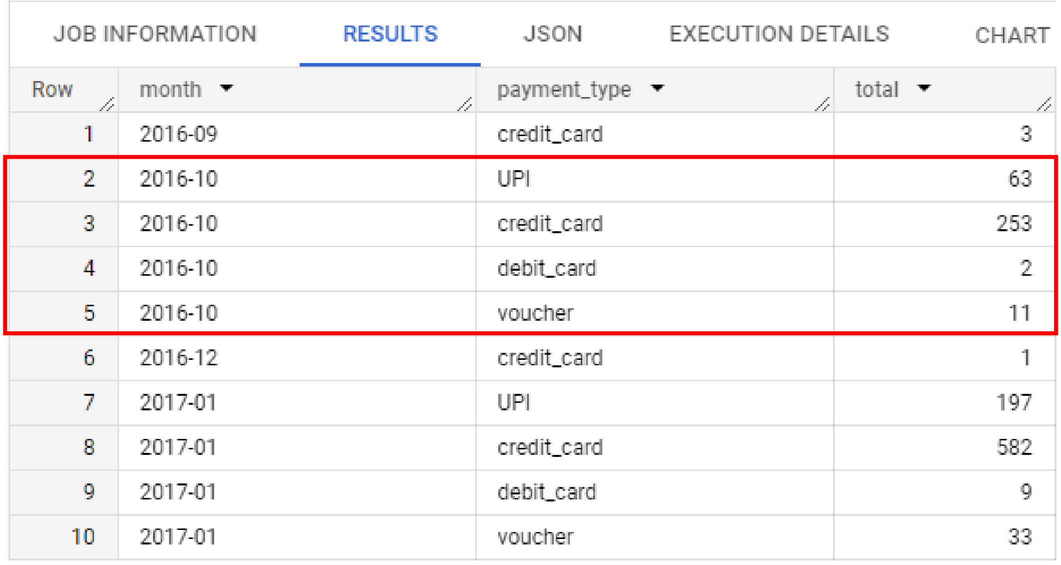
Figure 5.4.1: States which has the best service in terms of delivery.

6. Analysis based on the payments:

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

Query 1:

```
select format_date('%Y-%m',o.order_purchase_timestamp) as month,
p.payment_type, count(distinct p.order_id) as total
from `target_brazil_market.payments` as p
left join `target_brazil_market.orders` as o
on p.order_id = o.order_id
group by 1,2
order by 1, 2
```



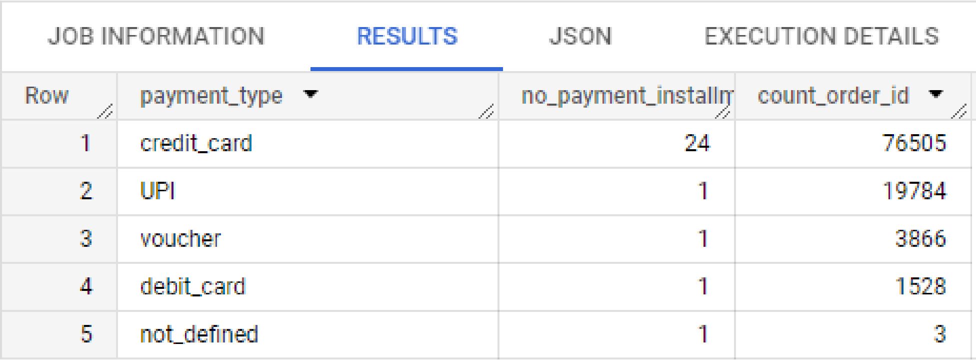
Row	month	payment_type	total
1	2016-09	credit_card	3
2	2016-10	UPI	63
3	2016-10	credit_card	253
4	2016-10	debit_card	2
5	2016-10	voucher	11
6	2016-12	credit_card	1
7	2017-01	UPI	197
8	2017-01	credit_card	582
9	2017-01	debit_card	9
10	2017-01	voucher	33

Figure 6.1.1: no. of orders across payment_type for each month.

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

Query 1:

```
select payment_type,
count (distinct payment_installments) as no_payment_installments,
count(distinct order_id) as count_order_id,
from `target_brazil_market.payments`
group by payment_type
order by count_order_id desc;
```



Row	payment_type	no_payment_installm	count_order_id
1	credit_card	24	76505
2	UPI	1	19784
3	voucher	1	3866
4	debit_card	1	1528
5	not_defined	1	3

Figure 6.2.1: no. of orders across completed payment installments.

Actionable Insights:

- There are totally 99441 customer records available.
- We have the data for 25 months.
- We have customers from 27 states and 4415 cities in Brazil.
- I felt the records are incomplete, for example there are around 625 records with no order status.
- There are 609 records where the orders are cancelled which means 96476 orders are delivered.

Recommendation:

- Average days taken to deliver are 12 days. From a consumer standpoint this is a very big inconvenience. This could be a major contributing factor for target to not have a successful run in the Brazil market.
- Therefore, I would recommend target to improve their delivery network.
- This is a suggestion based on how the data is saved. For example, the order_items table got orders with multiple items inside each order. These individual items have their own id which can cause more confusion instead they can be marked with their own product id which will make more sense.