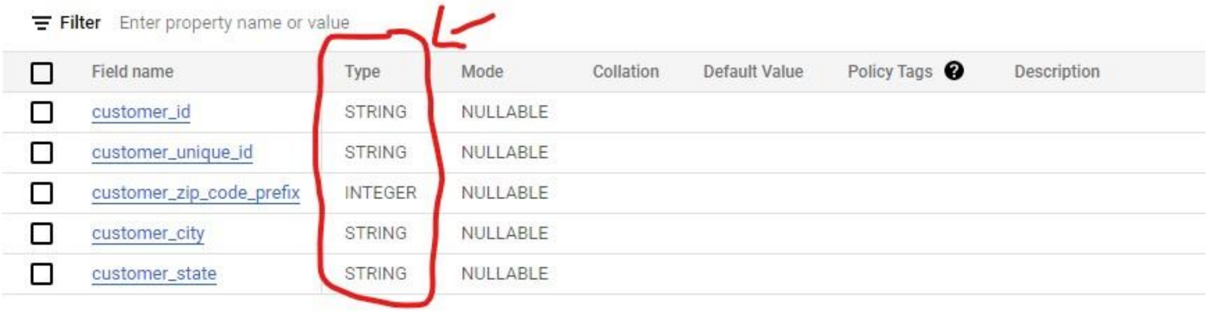
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table



☐	Field name	Type	Mode	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				

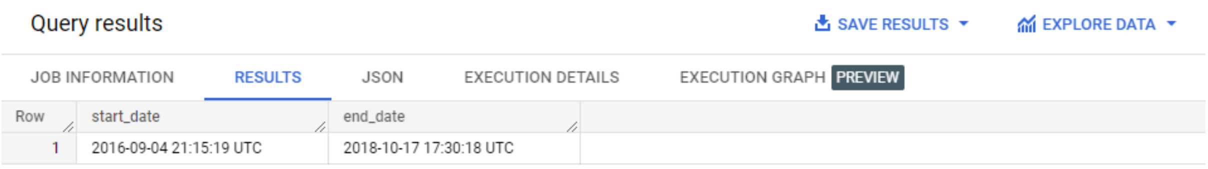
In order to view the data type of each columns in a table, simply double click the table name under the project name in your BigQuery and the data types of each columns will be displayed as depicted in the above image.

2. Time period for which the data is given

Ans:

```
select

min(order_purchase_timestamp) as start_date,
max(order_purchase_timestamp) as end_date
from `target.orders`;
```



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

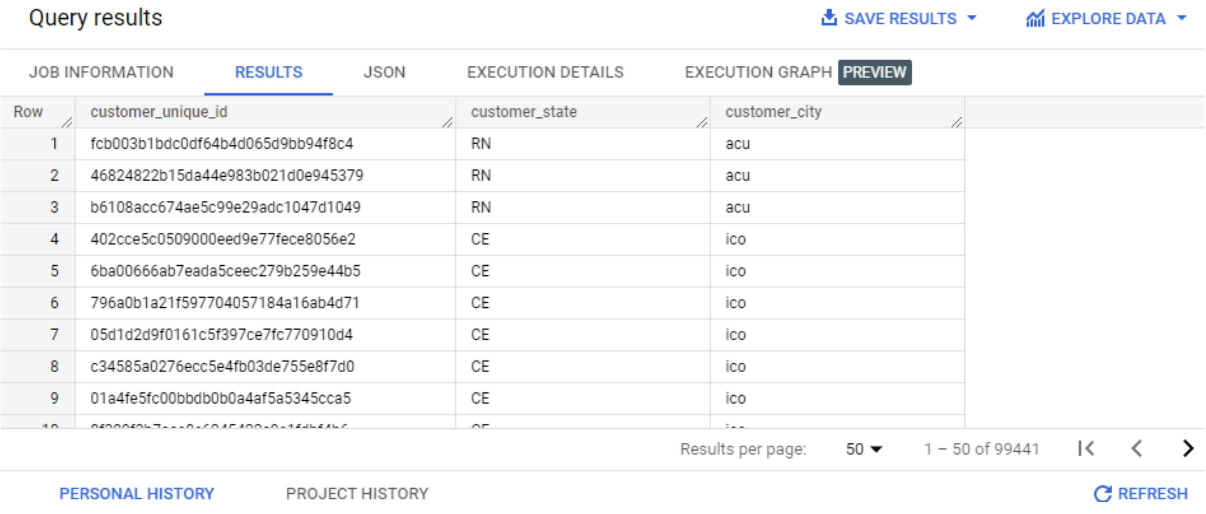
The above query depicts the time-period for which the Target data is given. It shows that the provided data is from 2016-09-04 to 2018-10-17.

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

Ans:

```
select

customer_unique_id,
customer_state,
customer_city
from `target.customers`;
```



Row	customer_unique_id	customer_state	customer_city
1	fcb003b1bdc0df64b4d065d9bb94f8c4	RN	acu
2	46824822b15da44e983b021d0e945379	RN	acu
3	b6108acc674ae5c99e29adc1047d1049	RN	acu
4	402cce5c0509000eed9e77fece8056e2	CE	ico
5	6ba00666ab7eada5ceec279b259e44b5	CE	ico
6	796a0b1a21f597704057184a16ab4d71	CE	ico
7	05d1d2d9f0161c5f397ce7fc770910d4	CE	ico
8	c34585a0276ecc5e4fb03de755e8f7d0	CE	ico
9	01a4fe5fc00bbdb0b0a4af5a5345cca5	CE	ico

In the above query, we have displayed state and city of each customer along with their unique customer Ids.

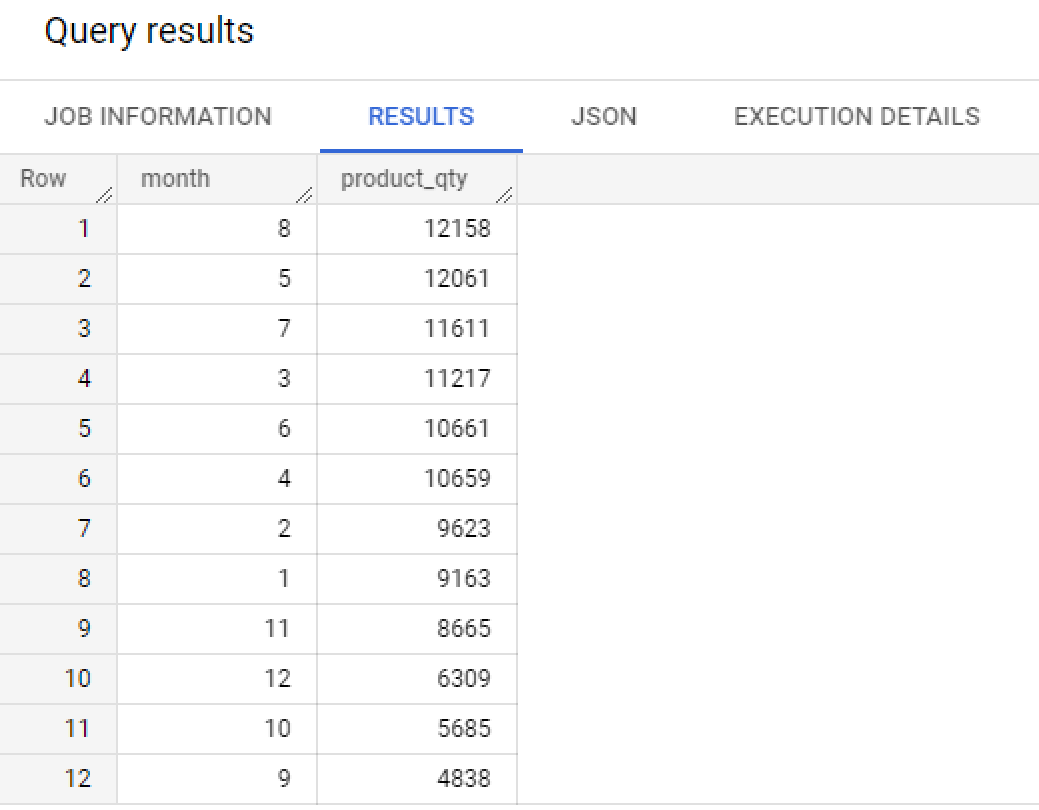
2. In-depth Exploration:

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

Ans: In order to answer this question, I have written 2 queries to get the insights. Following are the queries with their findings.

First Query:

```
select distinct
extract(month from o.order_purchase_timestamp) as month,
count(oi.product_id) over(partition by (extract(month from o.order_purchase_timestamp))) as product_qty,
from `target.orders` o join `target.order_items` oi on o.order_id = oi.order_id
order by product_qty desc;
```

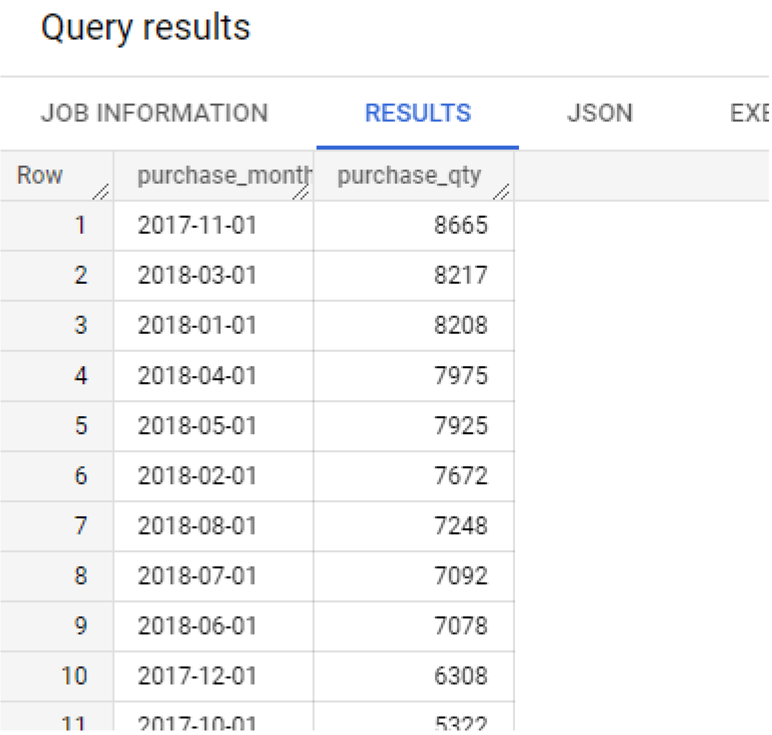


Row	month	product_qty
1	8	12158
2	5	12061
3	7	11611
4	3	11217
5	6	10661
6	4	10659
7	2	9623
8	1	9163
9	11	8665
10	12	6309
11	10	5685
12	9	4838

From the above query it is clear that overall August was the month in which the cumulative sales peaked followed by may and July.

Second Query:

```
select
date(date_trunc(order_purchase_timestamp, month)) as purchase_month,
count(i.product_id) as purchase_qty
from `target.orders` o left join `target.order_items` i
on o.order_id = i.order_id
group by purchase_month
order by 2 desc;
```



Row	purchase_month	purchase_qty
1	2017-11-01	8665
2	2018-03-01	8217
3	2018-01-01	8208
4	2018-04-01	7975
5	2018-05-01	7925
6	2018-02-01	7672
7	2018-08-01	7248
8	2018-07-01	7092
9	2018-06-01	7078
10	2017-12-01	6308
11	2017-10-01	5322

The above query shows total products purchased in every month.

By analysing the above result, we can get that:

A. It is clear that there is a clear growing trend in e-commerce in Brazil as 8 out of 10 months with highest purchases are of 2018.

B. Nov,2017 not only had the highest sales in 2017 by it also had the highest sales in the total given time-period.

C. March, 2018 witnessed the highest sales in 2018 and second highest sales in the given time-period.

D. 2018 had the major contribution in sales with 8 months in the top 10 month with the highest sales between 2017 and 2018.

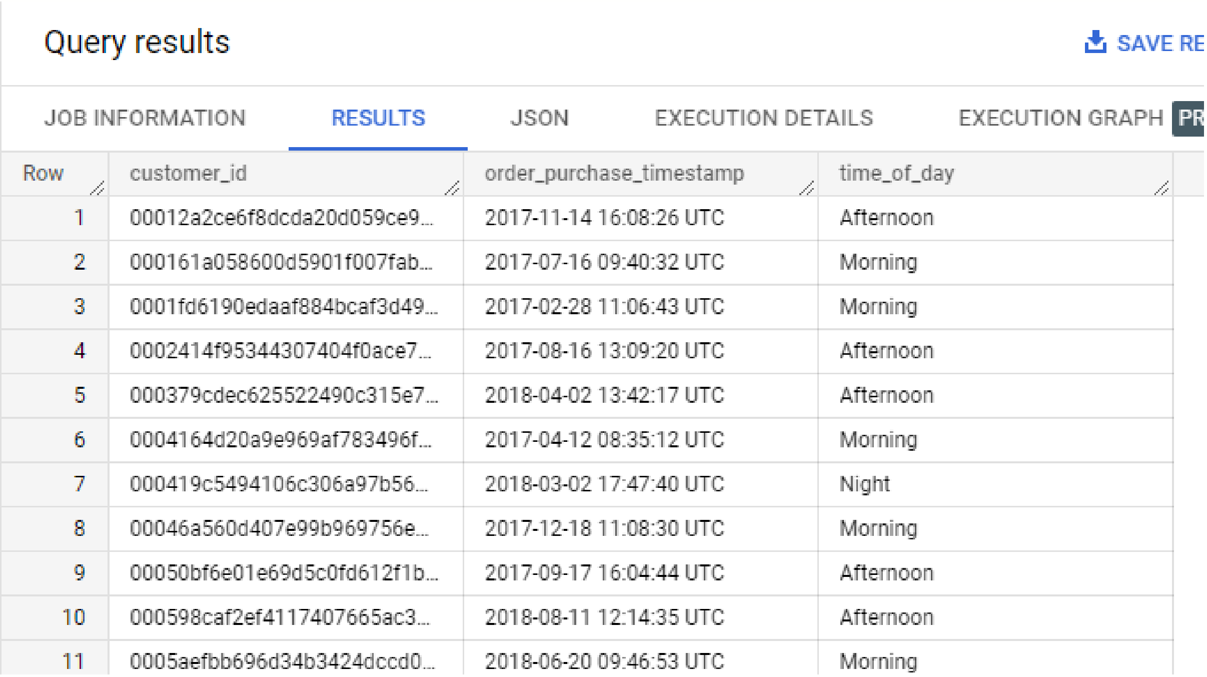
2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

Ans:

```
select

customer_id,
order_purchase_timestamp,
(case
   when time(order_purchase_timestamp) between '00:00:01' and '06:59:59' then 'Dawn'
   when time(order_purchase_timestamp) between '07:00:00' and '11:59:59' then 'Morning'
   when time(order_purchase_timestamp) between '12:00:00' and '16:59:59' then 'Afternoon'
   else 'Night'

end) as time_of_day
from `target.orders`
order by 1;
```



Row	customer_id	order_purchase_timestamp	time_of_day
1	00012a2ce6f8dcda20d059ce9…	2017-11-14 16:08:26 UTC	Afternoon
2	000161a058600d5901f007fab…	2017-07-16 09:40:32 UTC	Morning
3	0001fd6190edaaf884bcaf3d49…	2017-02-28 11:06:43 UTC	Morning
4	0002414f95344307404f0ace7…	2017-08-16 13:09:20 UTC	Afternoon
5	000379cdec625522490c315e7…	2018-04-02 13:42:17 UTC	Afternoon
6	0004164d20a9e969af783496f…	2017-04-12 08:35:12 UTC	Morning
7	000419c5494106c306a97b56…	2018-03-02 17:47:40 UTC	Night
8	00046a560d407e99b969756e…	2017-12-18 11:08:30 UTC	Morning
9	00050bf6e01e69d5c0fd612f1b…	2017-09-17 16:04:44 UTC	Afternoon
10	000598caf2ef4117407665ac3…	2018-08-11 12:14:35 UTC	Afternoon
11	0005aefbb696d34b3424dccd0…	2018-06-20 09:46:53 UTC	Morning

In the above query we have used case in order to divide the time of the day between dawn(00:00 to 06:59), morning(07:00 to 11:59), evening(12:00 to 16:59) and night(17:00 to 23:59).

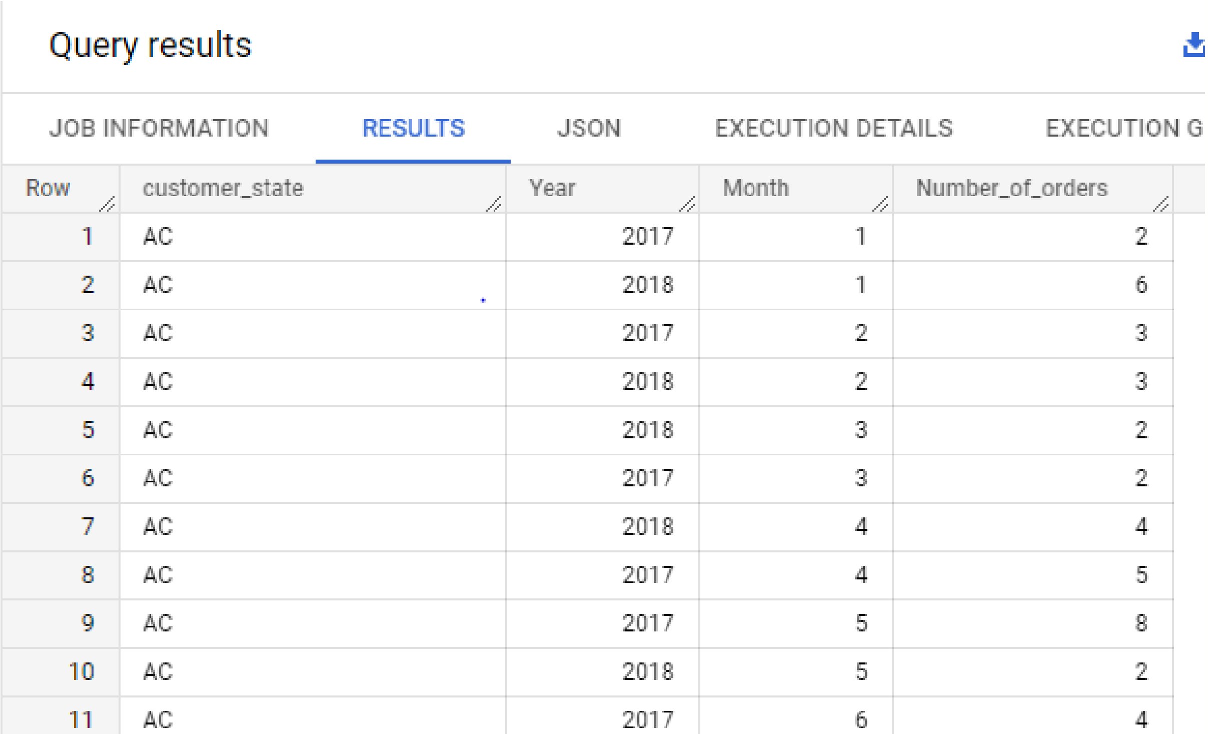
3. Evolution of E-commerce orders in the Brazil region:

1. Get month on month orders by states

Ans:

```
select

c.customer_state,
extract(year from o.order_purchase_timestamp) Year,
extract(month from o.order_purchase_timestamp) Month,
count(o.order_id) Number_of_orders
from `target.orders` o join `target.customers` c on o.customer_id = c.customer_id
group by c.customer_state, Year, Month
order by 1,3;
```



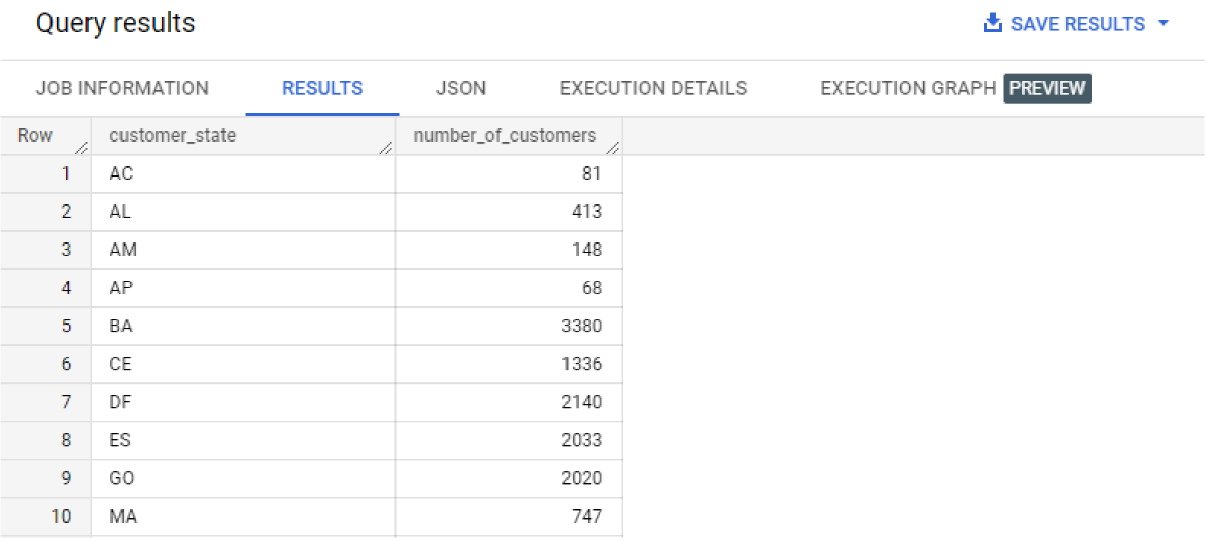
Row	customer_state	Year	Month	Number_of_orders
1	AC	2017	1	2
2	AC	2018	1	6
3	AC	2017	2	3
4	AC	2018	2	3
5	AC	2018	3	2
6	AC	2017	3	2
7	AC	2018	4	4
8	AC	2017	4	5
9	AC	2017	5	8
10	AC	2018	5	2
11	AC	2017	6	4

Here I have grouped the data by year, month and customer state. Here we can compare month on month sales(number of orders in a month in 2017 vs number of orders in 2018).

2. Distribution of customers across the states in Brazil.

Ans:

```
select
customer_state,
count(customer_id) as number_of_customers
from `target.customers`
group by customer_state
order by customer_state;
```



Row	customer_state	number_of_customers
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

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

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

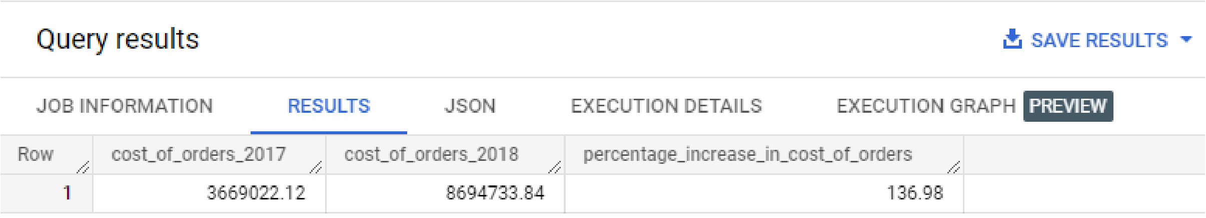
Ans:

```
select

round(t1.cost_2017,2) as cost_of_orders_2017,

round(t2.cost_2018,2) as cost_of_orders_2018,

round(((t2.cost_2018-t1.cost_2017)/t1.cost_2017)*100,2) as percentage_increase_in_cost_of_orders
from (select sum(p1.payment_value) cost_2017
        from `target.payments` p1 join `target.orders` o1
        on p1.order_id = o1.order_id
        where (extract(month from o1.order_purchase_timestamp) between 1 and 8) and (extract(year
        from o1.order_purchase_timestamp) = 2017)) t1
cross join
        (select sum(p2.payment_value) cost_2018
        from `target.payments` p2 join `target.orders` o2
        on p2.order_id = o2.order_id
        where (extract(month from o2.order_purchase_timestamp) between 1 and 8) and (extract(year
        from o2.order_purchase_timestamp) = 2018)) t2
```



Row	cost_of_orders_2017	cost_of_orders_2018	percentage_increase_in_cost_of_orders
1	3669022.12	8694733.84	136.98

Here, I have filtered the data between two tables with data between January and august and the year 2017 and 2018 respectively. After that I have used a cross join to merge these two tables and used the following formula to get the percentage increase in cost of purchase:

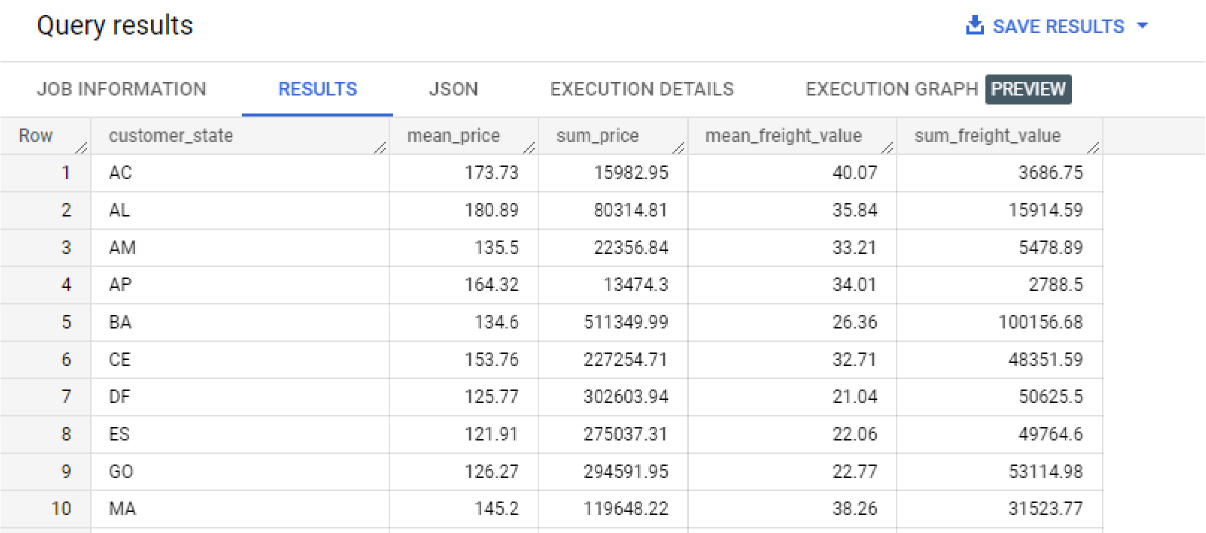
((Cost_of_orders_2018 – Cost_of_orders_2017)/ Cost_of_orders_2017) * 100

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

Ans:

```
select distinct

c.customer_state,
round(avg(price) over(partition by c.customer_state),2) as mean_price,
round(sum(price) over(partition by c.customer_state),2) as sum_price,
round(avg(freight_value) over(partition by c.customer_state),2) as mean_freight_value,
round(sum(freight_value) over(partition by c.customer_state),2) as sum_freight_value
from `target.order_items` oi join `target.orders` o on o.order_id = oi.order_id
join `target.customers` c on o.customer_id = c.customer_id
order by 1;
```



Row	customer_state	mean_price	sum_price	mean_freight_value	sum_freight_value
1	AC	173.73	15982.95	40.07	3686.75
2	AL	180.89	80314.81	35.84	15914.59
3	AM	135.5	22356.84	33.21	5478.89
4	AP	164.32	13474.3	34.01	2788.5
5	BA	134.6	511349.99	26.36	100156.68
6	CE	153.76	227254.71	32.71	48351.59
7	DF	125.77	302603.94	21.04	50625.5
8	ES	121.91	275037.31	22.06	49764.6
9	GO	126.27	294591.95	22.77	53114.98
10	MA	145.2	119648.22	38.26	31523.77

Here we have join 3 tables – orders, order_items and customers in order to get the customer state, price and freight value. Then we have used the window function to partition the resulting table by customer_state and performed the desired aggregations.

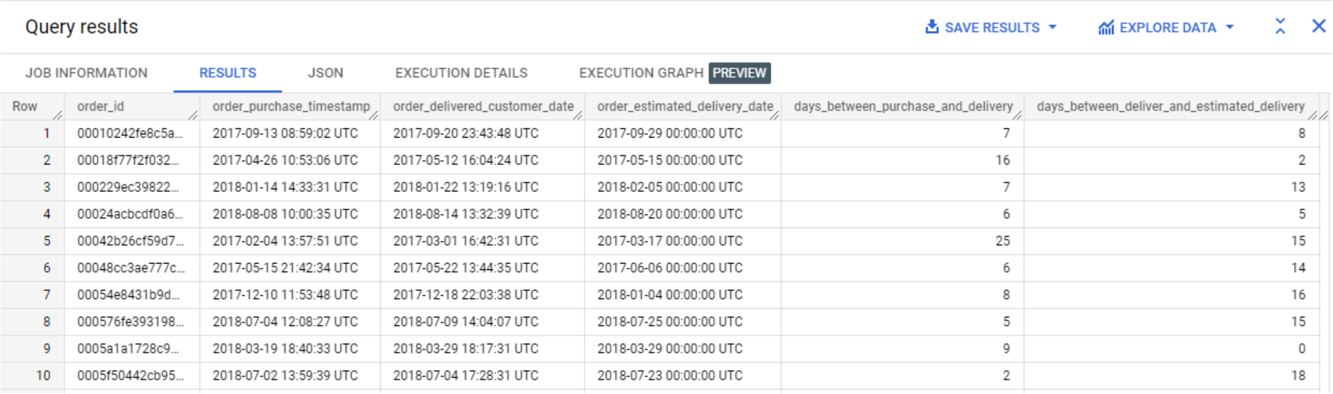
5. Analysis on sales, freight and delivery time

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

Ans:

```
select

order_id,
order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
(date_diff(order_delivered_customer_date, order_purchase_timestamp, day)) as days_between_purchase_and_delivery,
(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)) as days_between_deliver_and_estimated_delivery
from `target.orders`
order by 1;
```



Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	days_between_purchase_and_delivery	days_between_deliver_and_estimated_delivery
1	00010242fe8c5a...	2017-09-13 08:59:02 UTC	2017-09-20 23:43:48 UTC	2017-09-29 00:00:00 UTC	7	8
2	00018f77f2f032...	2017-04-26 10:53:06 UTC	2017-05-12 16:04:24 UTC	2017-05-15 00:00:00 UTC	16	2
3	000229ec39822...	2018-01-14 14:33:31 UTC	2018-01-22 13:19:16 UTC	2018-02-05 00:00:00 UTC	7	13
4	00024acbcdf0a6...	2018-08-08 10:00:35 UTC	2018-08-14 13:32:39 UTC	2018-08-20 00:00:00 UTC	6	5
5	00042b26cf59d7...	2017-02-04 13:57:51 UTC	2017-03-01 16:42:31 UTC	2017-03-17 00:00:00 UTC	25	15
6	00048cc3ae777c...	2017-05-15 21:42:34 UTC	2017-05-22 13:44:35 UTC	2017-06-06 00:00:00 UTC	6	14
7	00054e8431b9d...	2017-12-10 11:53:48 UTC	2017-12-18 22:03:38 UTC	2018-01-04 00:00:00 UTC	8	16
8	000576fe393198...	2018-07-04 12:08:27 UTC	2018-07-09 14:04:07 UTC	2018-07-25 00:00:00 UTC	5	15
9	0005a1a1728c9...	2018-03-19 18:40:33 UTC	2018-03-29 18:17:31 UTC	2018-03-29 00:00:00 UTC	9	0
10	0005f50442cb95...	2018-07-02 13:59:39 UTC	2018-07-04 17:28:31 UTC	2018-07-23 00:00:00 UTC	2	18

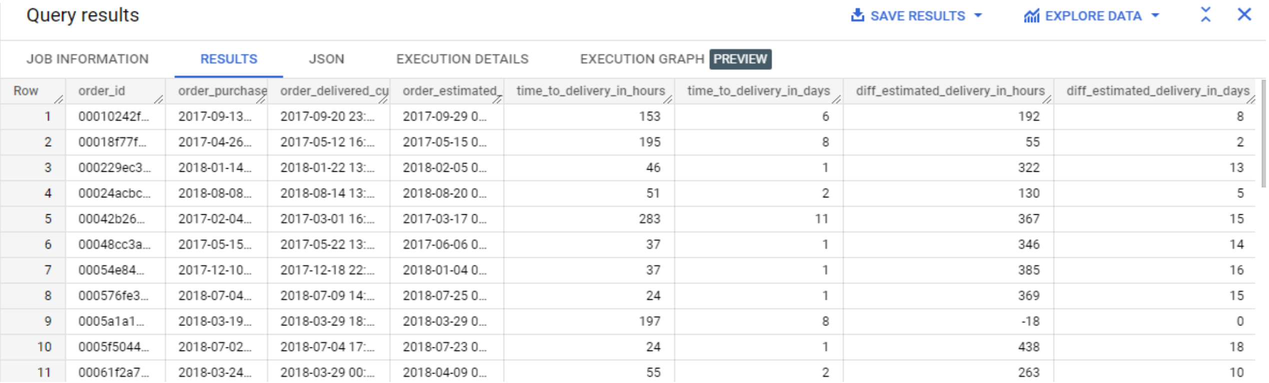
Here, we have calculated the difference between the number of days between 'order purchase date and order delivery date' and 'order delivery date and estimated order delivery date' using the date_diff dunction.

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
    1. time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

2. diff_estimated_delivery = order_estimated_delivery_date- order_delivered_customer_date

Ans:

```
select
order_id,
order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
abs(datetime_diff(order_purchase_timestamp, order_delivered_carrier_date, hour)) as time_to_delivery_in_hours,
abs(date_diff(order_purchase_timestamp, order_delivered_carrier_date, day)) as time_to_delivery_in_days,
datetime_diff(order_estimated_delivery_date, order_delivered_customer_date, hour) as diff_estimated_delivery_in_hours,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery_in_days
from `target.orders`
order by 1;
```



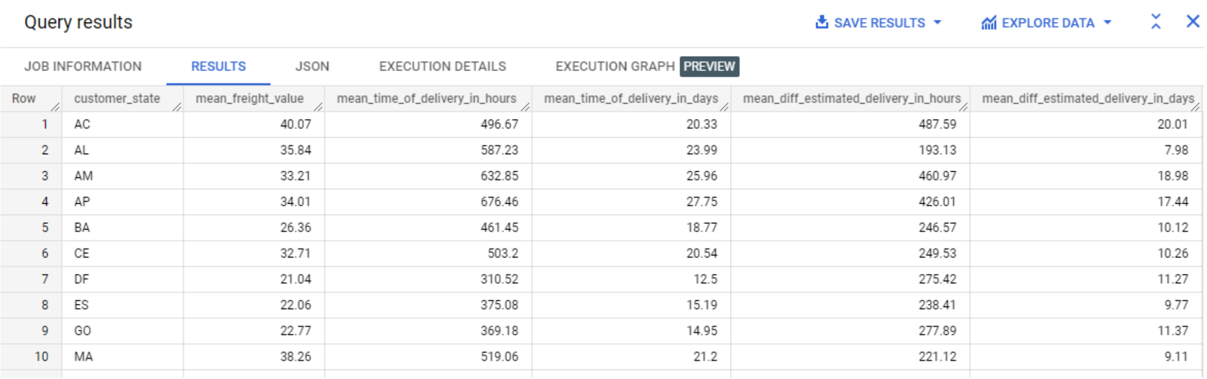
Row	order_id	order_purchase	order_delivered_cu	order_estimated_	time_to_delivery_in_hours	time_to_delivery_in_days	diff_estimated_delivery_in_hours	diff_estimated_delivery_in_days
1	00010242f...	2017-09-13...	2017-09-20 23:...	2017-09-29 0...	153	6	192	8
2	00018f77f...	2017-04-26...	2017-05-12 16:...	2017-05-15 0...	195	8	55	2
3	000229ec3...	2018-01-14...	2018-01-22 13:...	2018-02-05 0...	46	1	322	13
4	00024acbc...	2018-08-08...	2018-08-14 13:...	2018-08-20 0...	51	2	130	5
5	00042b26...	2017-02-04...	2017-03-01 16:...	2017-03-17 0...	283	11	367	15
6	00048cc3a...	2017-05-15...	2017-05-22 13:...	2017-06-06 0...	37	1	346	14
7	00054e84...	2017-12-10...	2017-12-18 22:...	2018-01-04 0...	37	1	385	16
8	000576fe3...	2018-07-04...	2018-07-09 14:...	2018-07-25 0...	24	1	369	15
9	0005a1a1...	2018-03-19...	2018-03-29 18:...	2018-03-29 0...	197	8	-18	0
10	0005f5044...	2018-07-02...	2018-07-04 17:...	2018-07-23 0...	24	1	438	18
11	00061f2a7...	2018-03-24...	2018-03-29 00:...	2018-04-09 0...	55	2	263	10

Here I have used the given formula to calculate the both the **time_to_deliver and diff_estimated_delivery in hours as well as in days** in order to get much deeper insight depending on the use case.

3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Ans:

```
select
c.customer_state,
round(avg(oi.freight_value),2) mean_freight_value,
round(avg(datetime_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,hour)),2) mean_time_of_delivery_in_hours,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,day)),2) mean_time_of_delivery_in_days,
round(avg(datetime_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, hour)),2) mean_diff_estimated_delivery_in_hours,
round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)),2) mean_diff_estimated_delivery_in_days
from `target.orders` o join `target.order_items` oi on o.order_id = oi.order_id
join `target.customers` c on o.customer_id = c.customer_id
group by c.customer_state
order by 1;
```



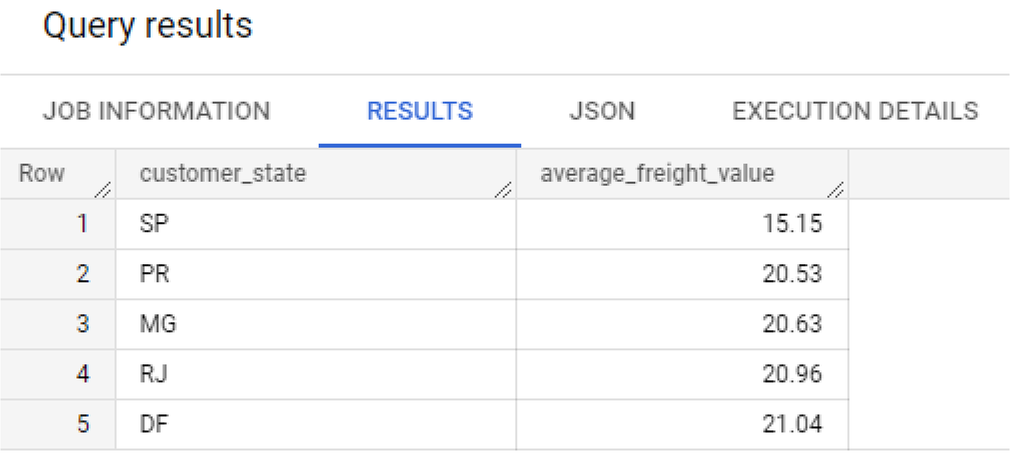
Row	customer_state	mean_freight_value	mean_time_of_delivery_in_hours	mean_time_of_delivery_in_days	mean_diff_estimated_delivery_in_hours	mean_diff_estimated_delivery_in_days
1	AC	40.07	496.67	20.33	487.59	20.01
2	AL	35.84	587.23	23.99	193.13	7.98
3	AM	33.21	632.85	25.96	460.97	18.98
4	AP	34.01	676.46	27.75	426.01	17.44
5	BA	26.36	461.45	18.77	246.57	10.12
6	CE	32.71	503.2	20.54	249.53	10.26
7	DF	21.04	310.52	12.5	275.42	11.27
8	ES	22.06	375.08	15.19	238.41	9.77
9	GO	22.77	369.18	14.95	277.89	11.37
10	MA	38.26	519.06	21.2	221.12	9.11

Here also I have used the given formula to find the mean of time_to_delivery and mean of diff_estimated_delivery in **hours as well as days** based on customer states in order to get a deeper insight on the data based on the requirement.

4. Sort the data to get the following:
5. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Ans: **Top 5 states with the lowest average freight value:**

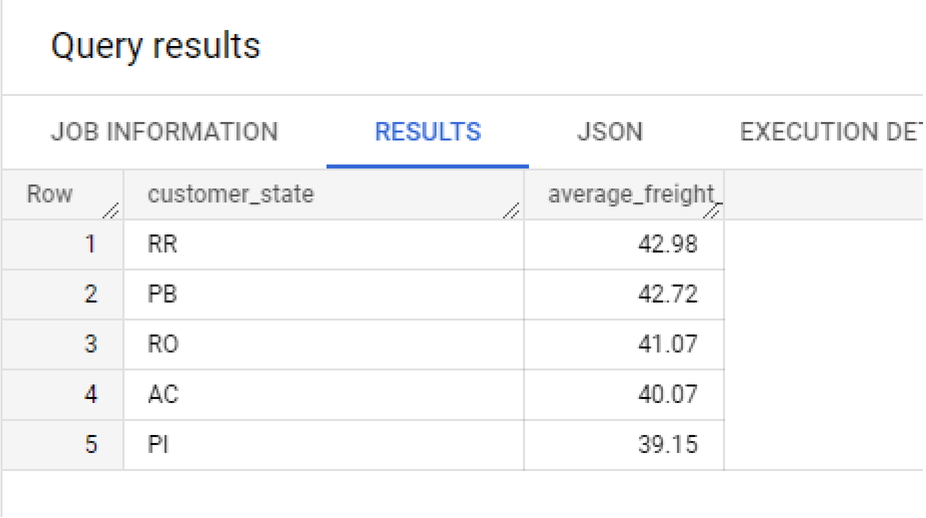
```
select
c.customer_state,
round(avg(oi.freight_value),2) average_freight_value
from `target.customers` c join `target.orders` o on c.customer_id = o.customer_id
join `target.order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by 2
limit 5;
```



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

Top 5 states with the highest average freight value:

```
select
c.customer_state,
round(avg(oi.freight_value),2) average_freight_value
from `target.customers` c join `target.orders` o on c.customer_id = o.customer_id
join `target.order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by 2 desc
limit 5;
```

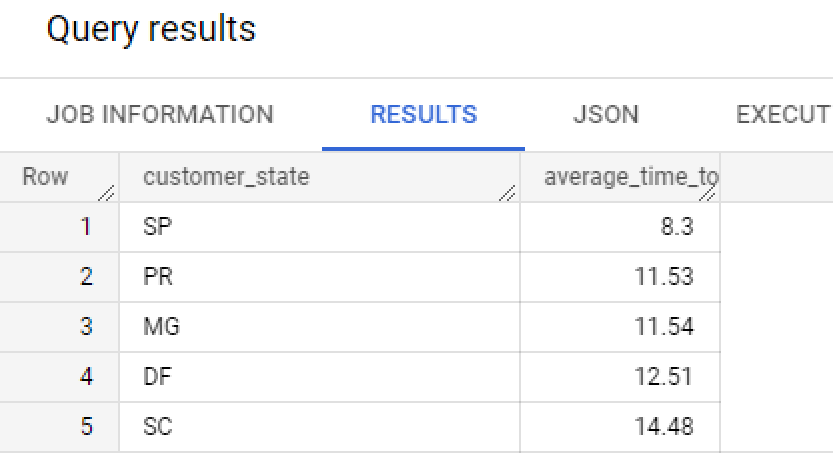


Row	customer_state	average_freight_
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

6. Top 5 states with highest/lowest average time to delivery

Ans: **Top 5 states with the lowest average time to delivery:**

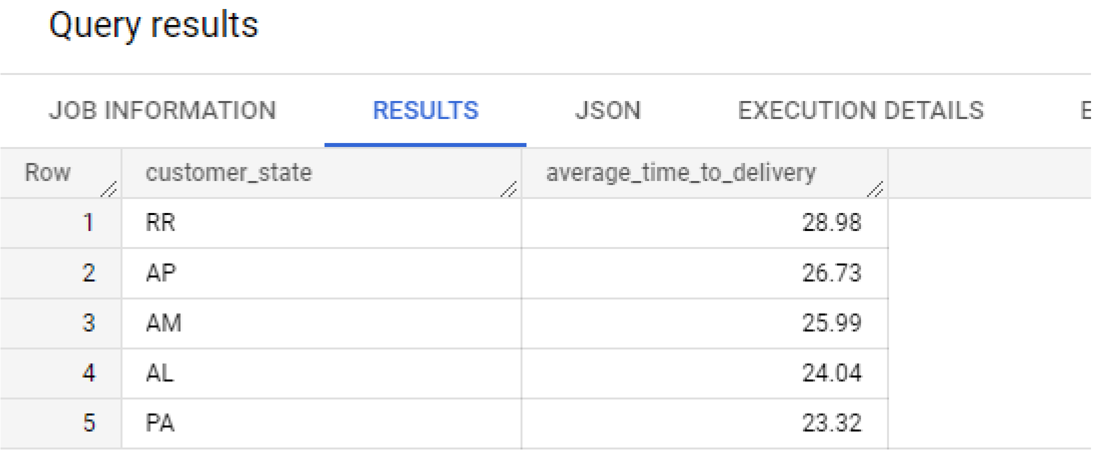
```
select
c.customer_state,
round(avg(abs(date_diff(o.order_purchase_timestamp, o.order_delivered_customer_date, day))),2) average_time_to_delivery
from `target.orders` o join `target.customers` c on o.customer_id = c.customer_id
group by c.customer_state
order by 2
limit 5;
```



Row	customer_state	average_time_to
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Top 5 states with the highest average time to delivery:

```
select
c.customer_state,
round(avg(abs(date_diff(o.order_purchase_timestamp, o.order_delivered_customer_date, day))),2) average_time_to_delivery
from `target.orders` o join `target.customers` c on o.customer_id = c.customer_id
group by c.customer_state
order by 2 desc
limit 5;
```



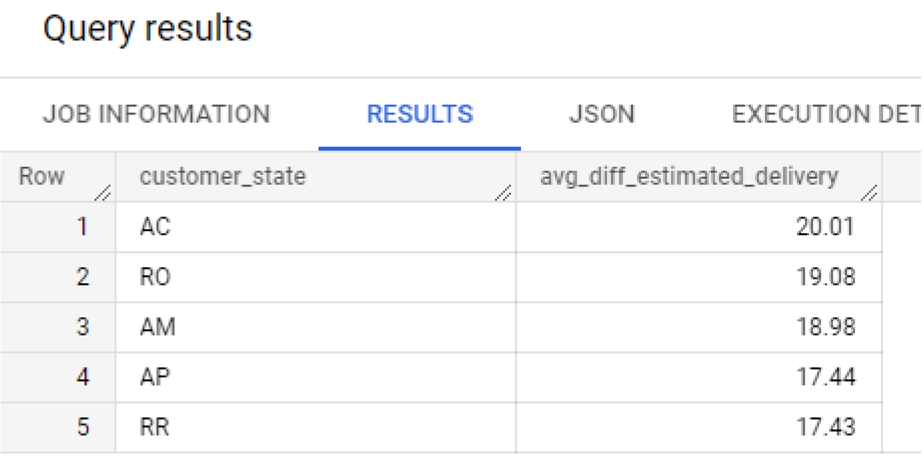
Row	customer_state	average_time_to_delivery
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

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

Ans:

Top 5 states where delivery is really fast as compared to the estimated date:

```
select
c.customer_state,
round(avg(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)),2) avg_diff_estimated
_delivery
from `target.orders` o join `target.order_items` oi on o.order_id = oi.order_id
join `target.customers` c on o.customer_id = c.customer_id
group by c.customer_state
order by 2 desc
limit 5;
```

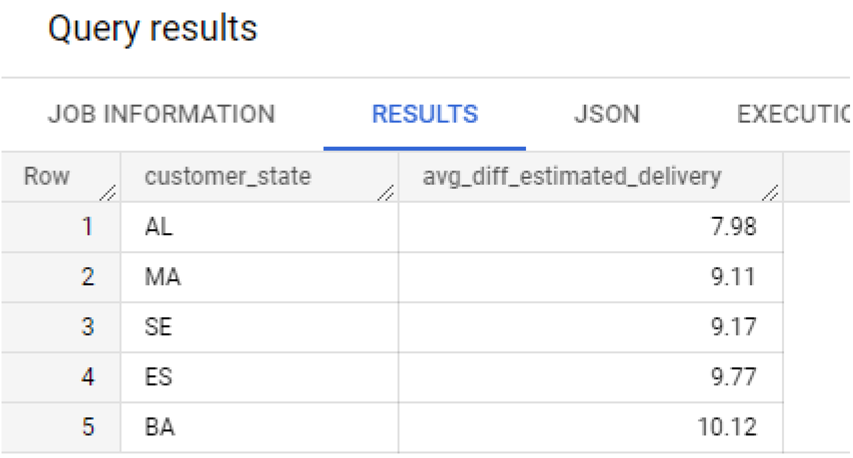


Row	customer_state	avg_diff_estimated_delivery
1	AC	20.01
2	RO	19.08
3	AM	18.98
4	AP	17.44
5	RR	17.43

In order to get the top 5 cities with **very fast delivery** compared to estimated delivery date, we have to sort the **average_diff_estimated_delivery in descending order**. Hence, we will have the data of top states with highest difference in estimated delivery date and order delivery date.

Top 5 states where delivery is not so fast as compared to the estimated date:

```
select
c.customer_state,
round(avg(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)),2) avg_diff_estimated_delivery
from `target.orders` o join `target.order_items` oi on o.order_id = oi.order_id
join `target.customers` c on o.customer_id = c.customer_id
group by c.customer_state
order by 2
limit 5;
```



Row	customer_state	avg_diff_estimated_delivery
1	AL	7.98
2	MA	9.11
3	SE	9.17
4	ES	9.77
5	BA	10.12

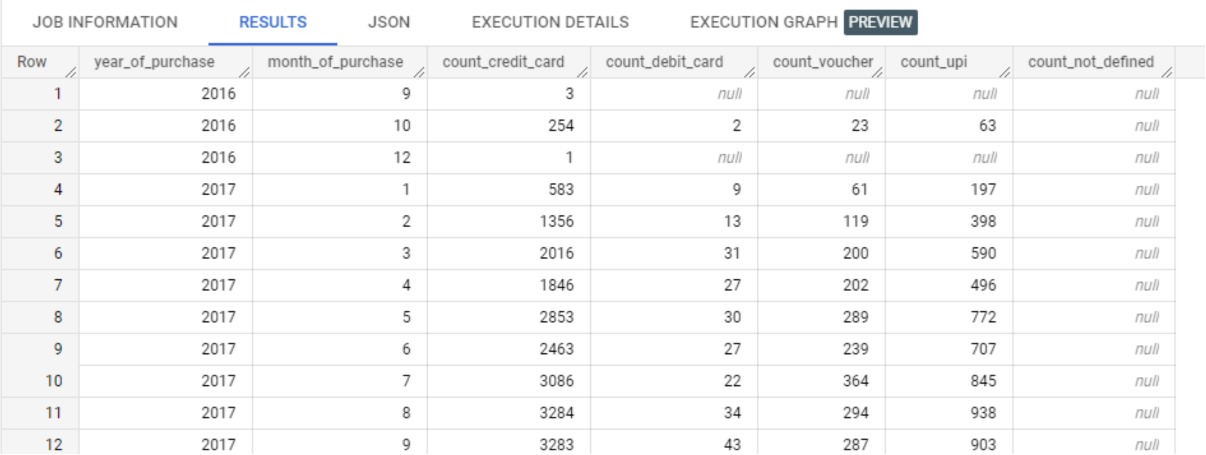
In order to get the top 5 cities with **not so fast delivery** compared to estimated delivery date, we have to sort the **average_diff_estimated_delivery in ascending order**. Hence, we will have the data of top states with least difference in estimated delivery date and order delivery date.

6. Payment type analysis:

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

Ans:

```
with credit_card as (
select
extract(year from o.order_purchase_timestamp) year_of_purchase,
extract(month from o.order_purchase_timestamp) month_of_purchase,
count(p.payment_type) count_credit_card
from `target.orders` o join `target.payments` p on o.order_id = p.order_id
group by year_of_purchase,month_of_purchase, p.payment_type
having p.payment_type = 'credit_card'
),
debit_card as (
select
extract(year from o.order_purchase_timestamp) year_of_purchase,
extract(month from o.order_purchase_timestamp) month_of_purchase,
count(p.payment_type) count_debit_card
from `target.orders` o join `target.payments` p on o.order_id = p.order_id
group by year_of_purchase,month_of_purchase, p.payment_type
having p.payment_type = 'debit_card'
),
voucher as (
   select
extract(year from o.order_purchase_timestamp) year_of_purchase,
extract(month from o.order_purchase_timestamp) month_of_purchase,
count(p.payment_type) count_voucher
from `target.orders` o join `target.payments` p on o.order_id = p.order_id
group by year_of_purchase,month_of_purchase, p.payment_type
having p.payment_type = 'voucher'
),
UPI as (
    select
extract(year from o.order_purchase_timestamp) year_of_purchase,
extract(month from o.order_purchase_timestamp) month_of_purchase,
count(p.payment_type) count_upi
from `target.orders` o join `target.payments` p on o.order_id = p.order_id
group by year_of_purchase,month_of_purchase, p.payment_type
having p.payment_type = 'UPI'
),
not_defined as (
    select
extract(year from o.order_purchase_timestamp) year_of_purchase,
extract(month from o.order_purchase_timestamp) month_of_purchase,
count(p.payment_type) count_not_defined
from `target.orders` o join `target.payments` p on o.order_id = p.order_id
group by year_of_purchase,month_of_purchase, p.payment_type
having p.payment_type = 'not_defined'
)
select distinct
cc.year_of_purchase,
cc.month_of_purchase,
cc.count_credit_card,
dc.count_debit_card,
v.count_voucher,
upi.count_upi,
nd.count_not_defined
from credit_card cc left join debit_card dc on cc.month_of_purchase = dc.month_of_purchase and cc.year_of_purchase=dc.year_of_purchase
left join voucher v on cc.month_of_purchase = v.month_of_purchase and cc.year_of_purchase = v.year_of_purchase
left join upi on upi.month_of_purchase = cc.month_of_purchase and cc.year_of_purchase = upi.year_of_purchase
left join not_defined nd on cc.month_of_purchase = nd.month_of_purchase and cc.year_of_purchase = nd.year_of_purchase
order by 1,2;
```



Row	year_of_purchase	month_of_purchase	count_credit_card	count_debit_card	count_voucher	count_upi	count_not_defined
1	2016	9	3	null	null	null	null
2	2016	10	254	2	23	63	null
3	2016	12	1	null	null	null	null
4	2017	1	583	9	61	197	null
5	2017	2	1356	13	119	398	null
6	2017	3	2016	31	200	590	null
7	2017	4	1846	27	202	496	null
8	2017	5	2853	30	289	772	null
9	2017	6	2463	27	239	707	null
10	2017	7	3086	22	364	845	null
11	2017	8	3284	34	294	938	null
12	2017	9	3283	43	287	903	null

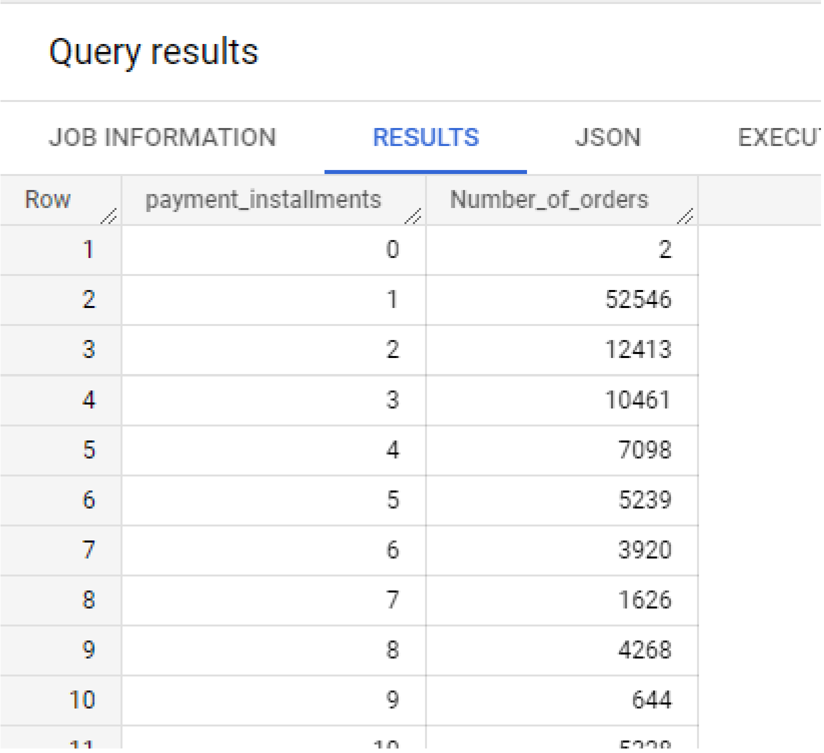
Here I have used CTE to in order to get the count of orders with each payment type for each month of the year.

We can clearly see that maximum people preferred payment through their credit card as compared to any other payment type.

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

Ans:

```
select
payment_installments,
count(order_id) Number_of_orders,
from `target.payments`
group by payment_installments
order by 1;
```



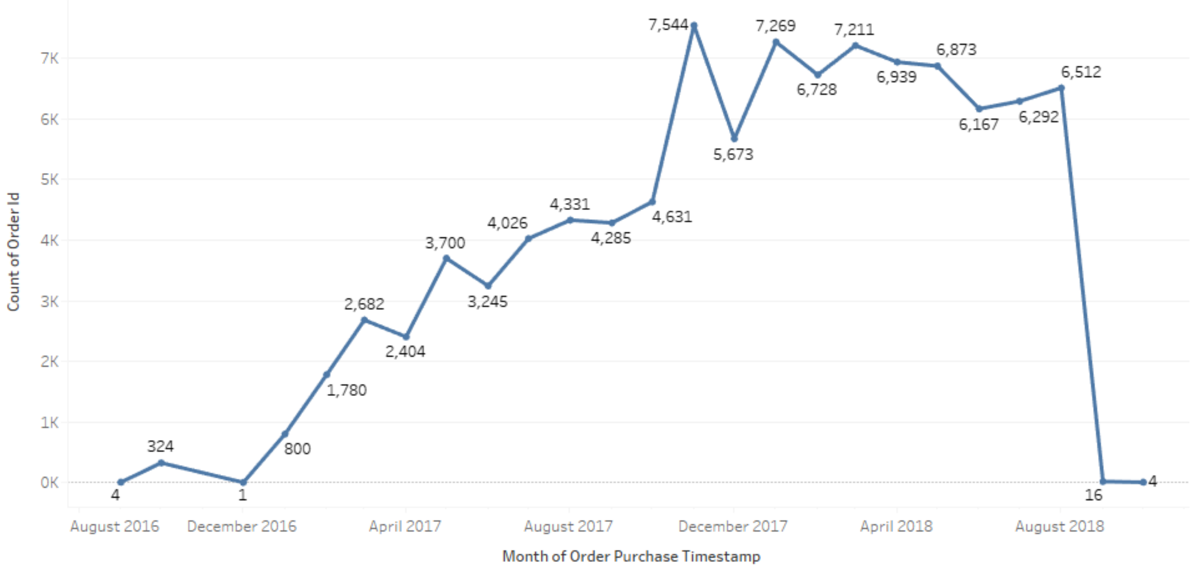
Row	payment_installments	Number_of_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328

Here I have grouped the data on the number of payment instalments.

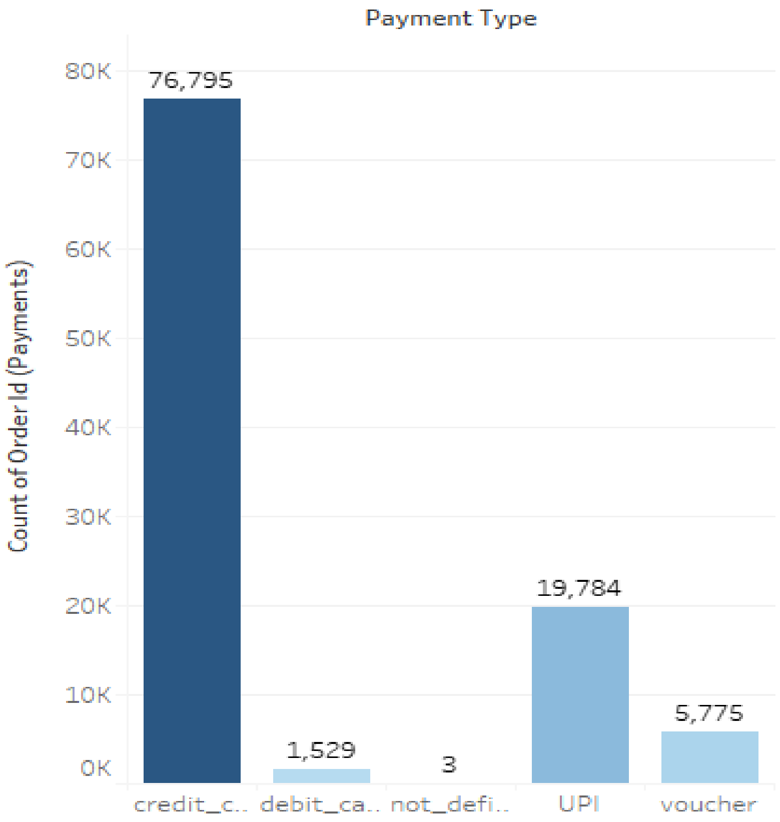
We can see that maximum people opted to pay in one instalment followed by people who paid in 2 instalments.

Additional Graphs for Additional Insight

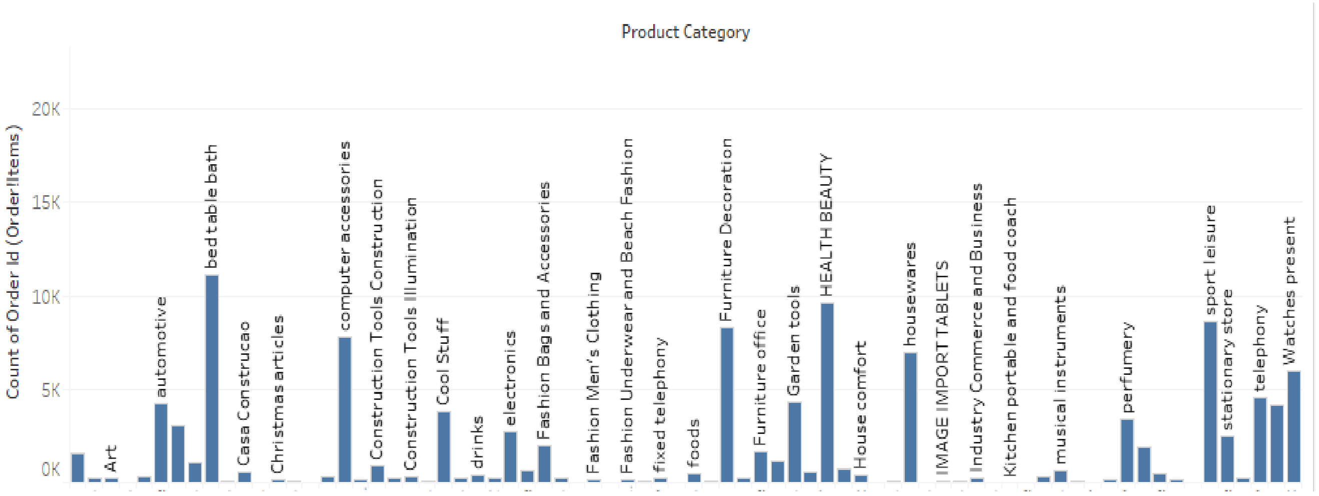
1. Number of orders VS Month of order purchase:



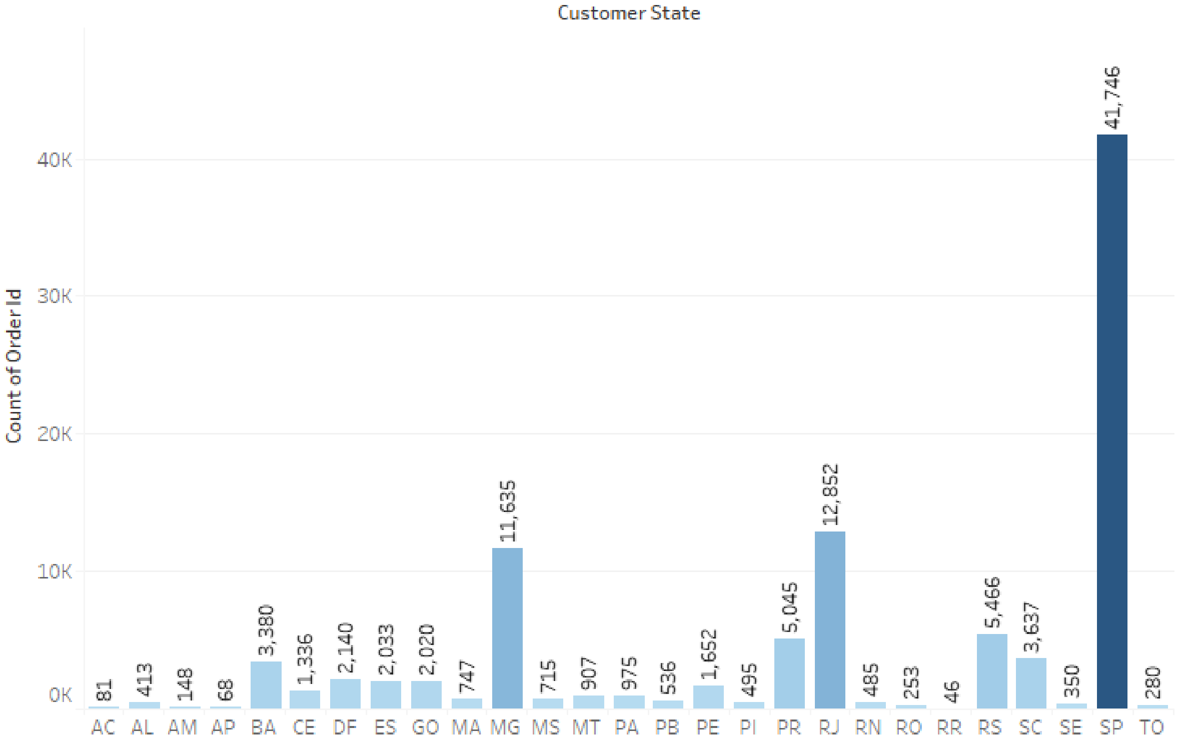
2. Number of orders VS Payment Type:



3. Number of orders VS Product Category:



4. Number of orders VS Customer State



RECOMMENDATIONS

1. A drastic increase on sales was seen from Aug, 2016 to Dec, 2017 after which it remained somewhat constant with some ups and downs. In order to increase the sales, it is recommended that more products must be included in the on demand product categories like Bed table bath, computer accessories, furniture decoration, health beauty and sports and leisure. Further some lucrative offers must be given to the customers so that their purchasing decision becomes more easy.

2. Credit card is the most favoured means of payment followed by UPI and vouchers. It is recommended to give more offers on other means of payment such as UPI, vouchers and debit cards in order to increase their usage also.