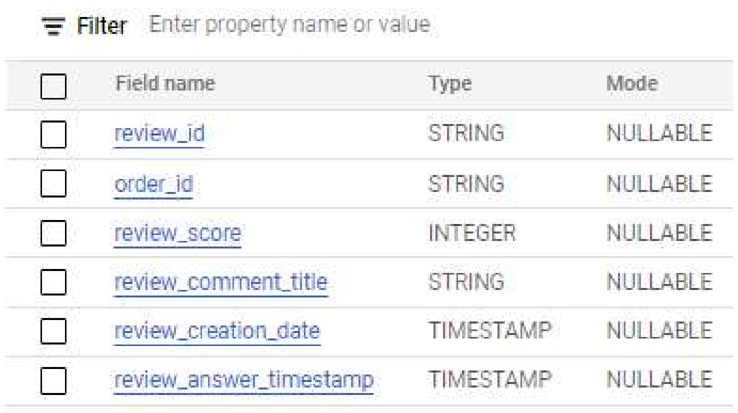
Target SQL

Q-1) Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

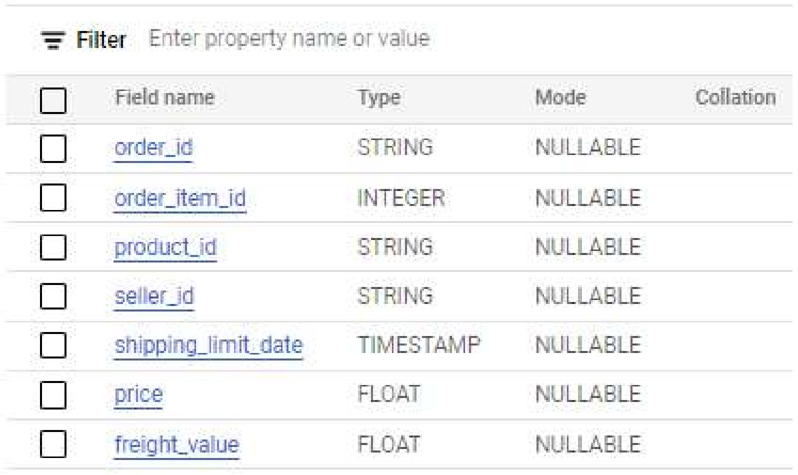
1) Data type of columns in a table



Filter Enter property name or value

Field name	Type	Mode
review_id	STRING	NULLABLE
order_id	STRING	NULLABLE
review_score	INTEGER	NULLABLE
review_comment_title	STRING	NULLABLE
review_creation_date	TIMESTAMP	NULLABLE
review_answer_timestamp	TIMESTAMP	NULLABLE

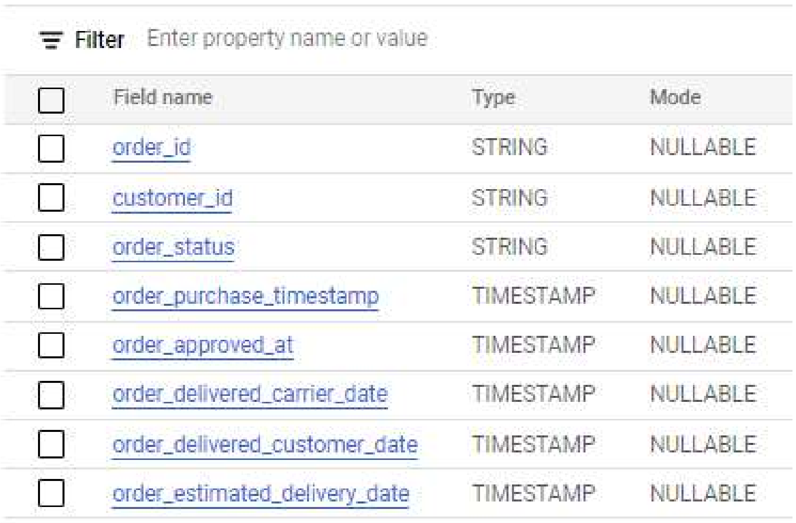
Customer



Filter Enter property name or value

Field name	Type	Mode	Collation
order_id	STRING	NULLABLE	
order_item_id	INTEGER	NULLABLE	
product_id	STRING	NULLABLE	
seller_id	STRING	NULLABLE	
shipping_limit_date	TIMESTAMP	NULLABLE	
price	FLOAT	NULLABLE	
freight_value	FLOAT	NULLABLE	

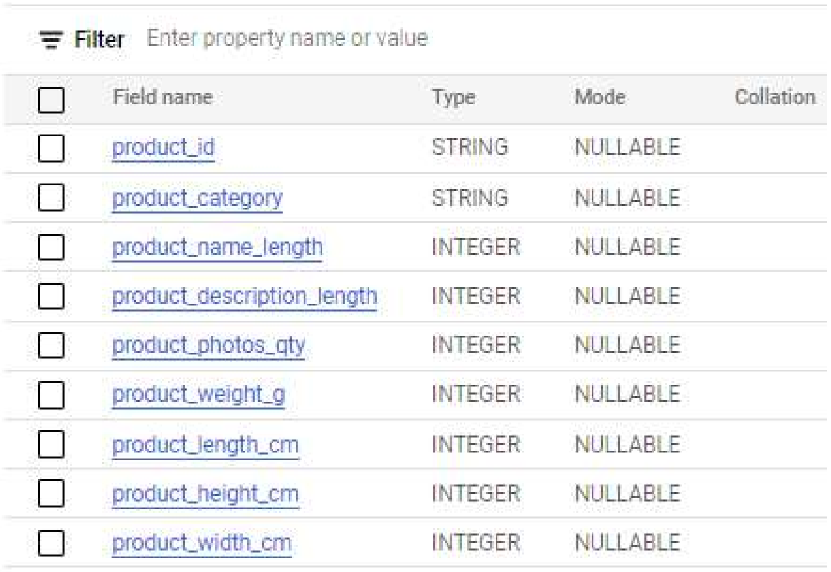
Order_items



Filter Enter property name or value

Field name	Type	Mode
order_id	STRING	NULLABLE
customer_id	STRING	NULLABLE
order_status	STRING	NULLABLE
order_purchase_timestamp	TIMESTAMP	NULLABLE
order_approved_at	TIMESTAMP	NULLABLE
order_delivered_carrier_date	TIMESTAMP	NULLABLE
order_delivered_customer_date	TIMESTAMP	NULLABLE
order_estimated_delivery_date	TIMESTAMP	NULLABLE

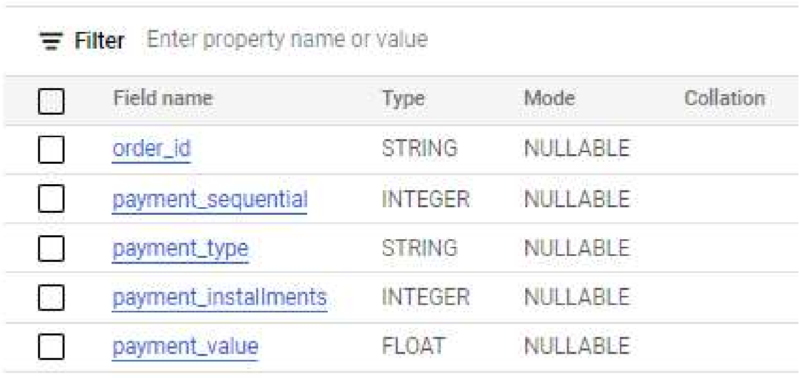
Orders



Filter Enter property name or value

Field name	Type	Mode	Collation
product_id	STRING	NULLABLE	
product_category	STRING	NULLABLE	
product_name_length	INTEGER	NULLABLE	
product_description_length	INTEGER	NULLABLE	
product_photos_qty	INTEGER	NULLABLE	
product_weight_g	INTEGER	NULLABLE	
product_length_cm	INTEGER	NULLABLE	
product_height_cm	INTEGER	NULLABLE	
product_width_cm	INTEGER	NULLABLE	

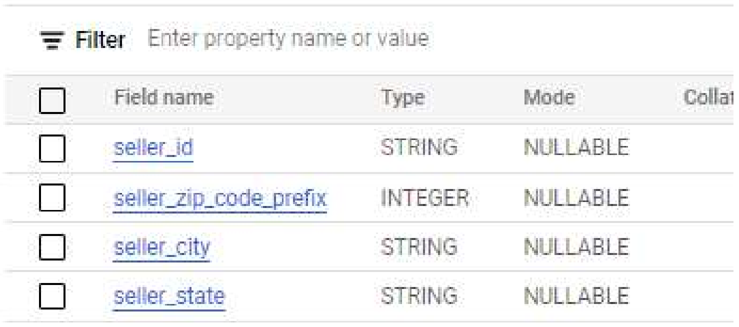
Product



Filter Enter property name or value

	Field name	Type	Mode	Collation
☐	order_id	STRING	NULLABLE	
☐	payment_sequential	INTEGER	NULLABLE	
☐	payment_type	STRING	NULLABLE	
☐	payment_installments	INTEGER	NULLABLE	
☐	payment_value	FLOAT	NULLABLE	

Payments



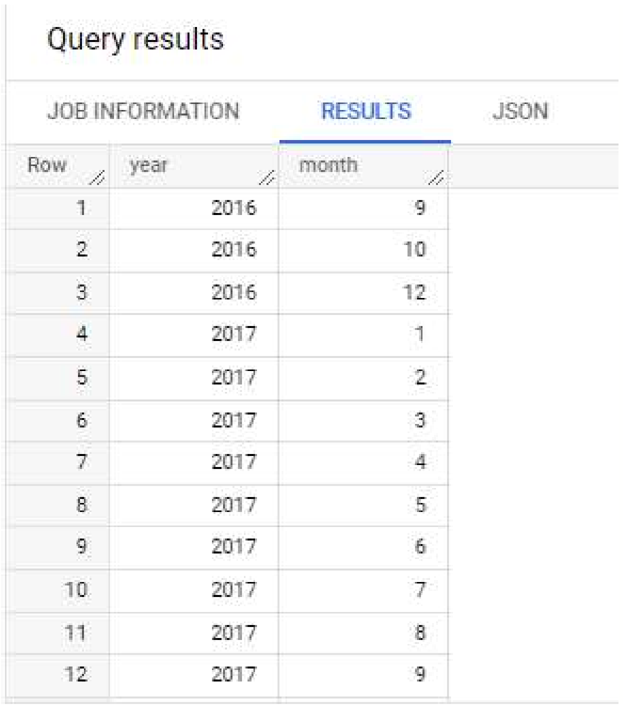
Filter Enter property name or value

	Field name	Type	Mode	Collat
☐	seller_id	STRING	NULLABLE	
☐	seller_zip_code_prefix	INTEGER	NULLABLE	
☐	seller_city	STRING	NULLABLE	
☐	seller_state	STRING	NULLABLE	

Seller

2) Time period for which the data is given

```
select
distinct
extract (year from order_purchase_timestamp) as year,
extract (month from order_purchase_timestamp) as month
from `ecommerce.orders`
order by year, month
```


Query results

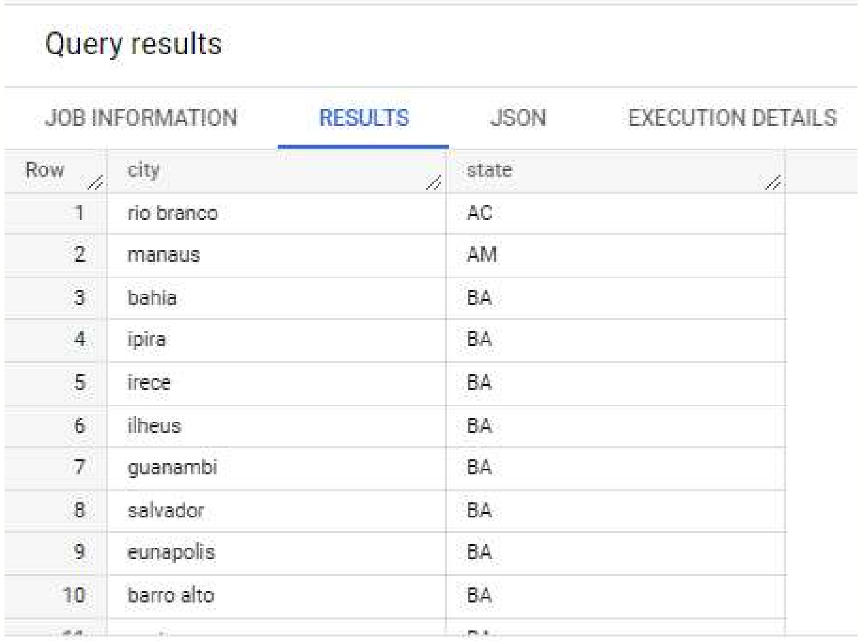
JOB INFORMATION | RESULTS | JSON

Row	year	month
1	2016	9
2	2016	10
3	2016	12
4	2017	1
5	2017	2
6	2017	3
7	2017	4
8	2017	5
9	2017	6
10	2017	7
11	2017	8
12	2017	9

3) Cities and States covered in the dataset:

All cities and states:

```
select  customer_city as city, customer_state as state from `ecommerce.customers`
union distinct
select  seller_city as city, seller_state as state from `ecommerce.sellers`
```

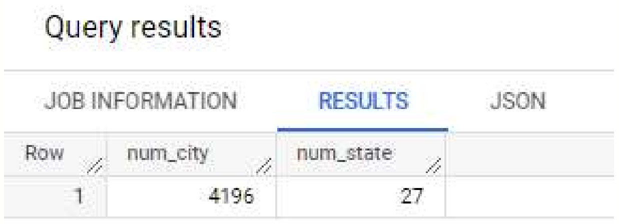

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	city	state
1	rio branco	AC
2	manaus	AM
3	bahia	BA
4	ipira	BA
5	irece	BA
6	ilheus	BA
7	guanambi	BA
8	salvador	BA
9	eunapolis	BA
10	barro alto	BA

Count of cities and states:

```
select
count(distinct city) as num_city,
count(distinct state) as num_state
from
(select  customer_city as city, customer_state as state from `ecommerce.customers`
union distinct
select  seller_city as city, seller_state as state from `ecommerce.sellers`)
```



Query results

JOB INFORMATION RESULTS JSON

Row	num_city	num_state
1	4196	27

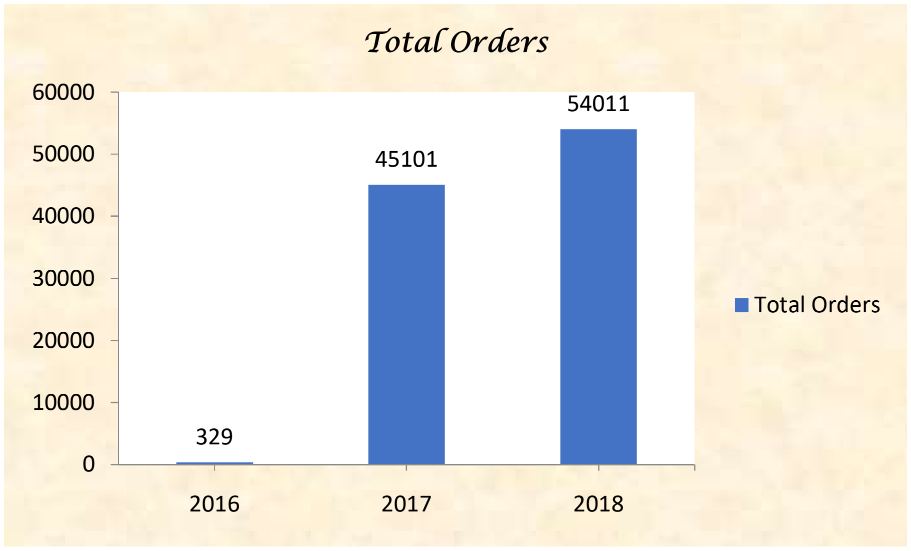
Q2) In-depth Exploration:

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?

Yearly Analysis:

```
with raw as (select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
extract (day from order_purchase_timestamp) as day,
order_id
from `ecommerce.orders`),
trend as (select
year,
month,
day,
count(order_id) as num_order
from raw
group by year,month,day
order by year,month,day)
select
```

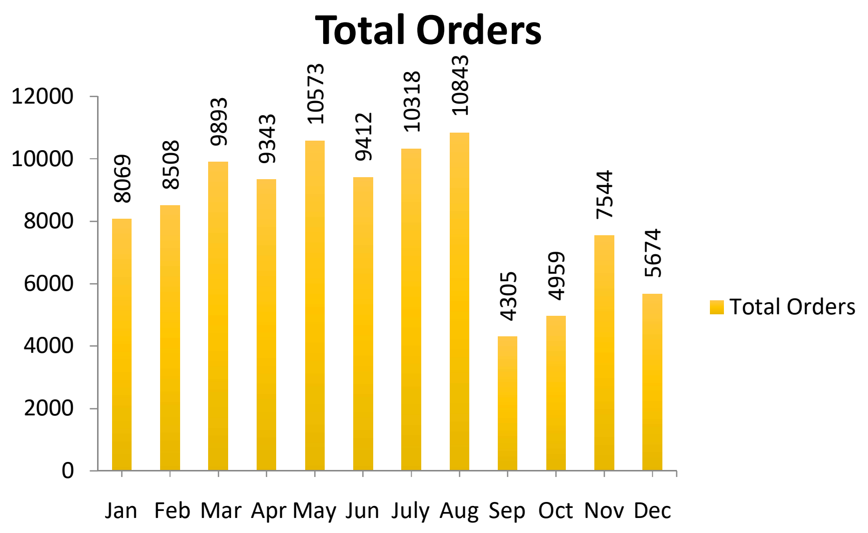
```
trend.year,
sum(trend.num_order) as total_orders
from trend
group by trend.year
order by trend.year
```



Monthly Analysis

```
with raw as (select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
extract (day from order_purchase_timestamp) as day,
order_id
from `ecommerce.orders`),
trend as (select
year,
month,
day,
count(order_id) as num_order
from raw
group by year,month,day
order by year,month,day)
select
```

```
trend.month,
sum(trend.num_order) as total_orders
from trend
group by trend.month
order by trend.month
```



Monthly Analysis Year by Year

For Year 2016:

```
with raw as (select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
extract (day from order_purchase_timestamp) as day,
order_id
from `ecommerce.orders`),
trend as (select year,month,day,
count(order_id) as num_order
from raw
group by year,month,day
order by year,month,day)
select
trend.month,
sum(trend.num_order) as total_orders
from trend
where trend.year = 2016
group by trend.month
```

```
order by trend.month
```

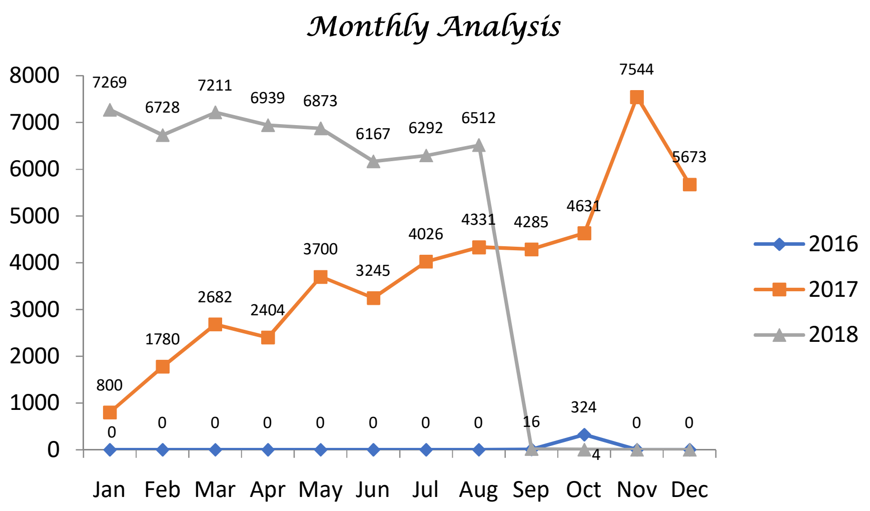
For Year 2017:

```
with raw as (select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
extract (day from order_purchase_timestamp) as day,
order_id
from `ecommerce.orders`),
trend as (select year,month,day,
count(order_id) as num_order
from raw
group by year,month,day
order by year,month,day)
select
trend.month,
sum(trend.num_order) as total_orders
from trend
where trend.year = 2017
group by trend.month
order by trend.month
```

For Year 2018:

```
with raw as (select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
extract (day from order_purchase_timestamp) as day,
order_id
from `ecommerce.orders`),
trend as (select year,month,day,
count(order_id) as num_order
from raw
group by year,month,day
order by year,month,day)
select
trend.month,
sum(trend.num_order) as total_orders
from trend
where trend.year = 2018
group by trend.month
order by trend.month
```

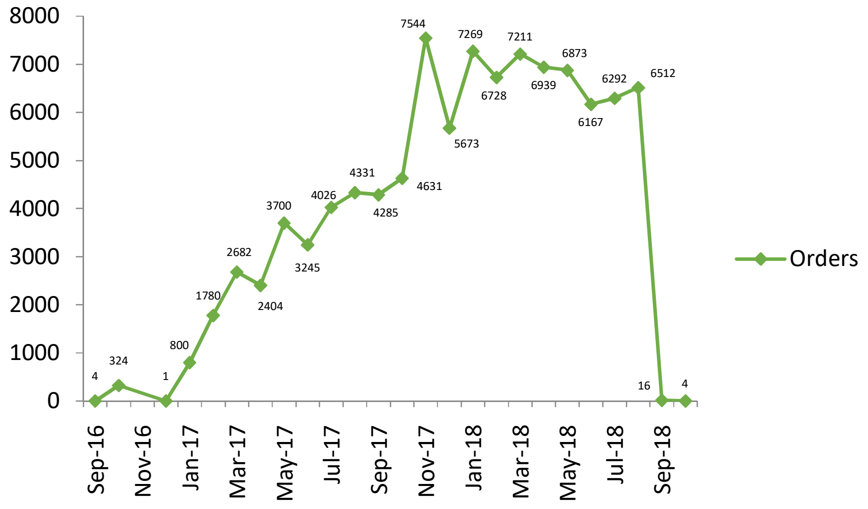
After Getting Table for all three years, compiling them in one Excel sheet and plotting the chart:



Since Inception:

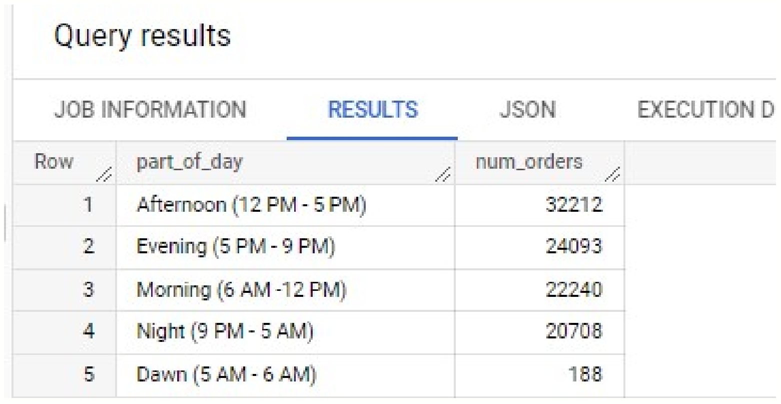
```
with raw as (select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
format_date('%b', order_purchase_timestamp) as month_name,
extract (day from order_purchase_timestamp) as day,
order_id
from `ecommerce.orders`),

trend as (
select year,month,month_name,day,
count(order_id) as num_order
from raw
group by year,month,month_name,day
order by year,month,day)
```

```
select
concat(trend.month_name,' ', trend.year) as time,
sum(trend.num_order) as total_orders
from trend
group by trend.year, trend.month, trend.month_name
order by trend.year, trend.month
```



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

```
with part as (select
order_id,
time(order_purchase_timestamp) as time,
case
when time(order_purchase_timestamp) between '05:00:01' and '06:00:00'
then 'Dawn (5 AM - 6 AM)'
when time(order_purchase_timestamp) between '06:00:01' and '12:00:00'
then 'Morning (6 AM -12 PM)'
when time(order_purchase_timestamp) between '12:00:01' and '17:00:00'
then 'Afternoon (12 PM - 5 PM)'
when time(order_purchase_timestamp) between '17:00:01' and '21:00:00'
then 'Evening (5 PM - 9 PM)'
else 'Night (9 PM - 5 AM)'
end as part_of_day
from `ecommerce.orders`
order by time)
select
part.part_of_day,
count(part.order_id) as num_orders
from part
group by part.part_of_day
```



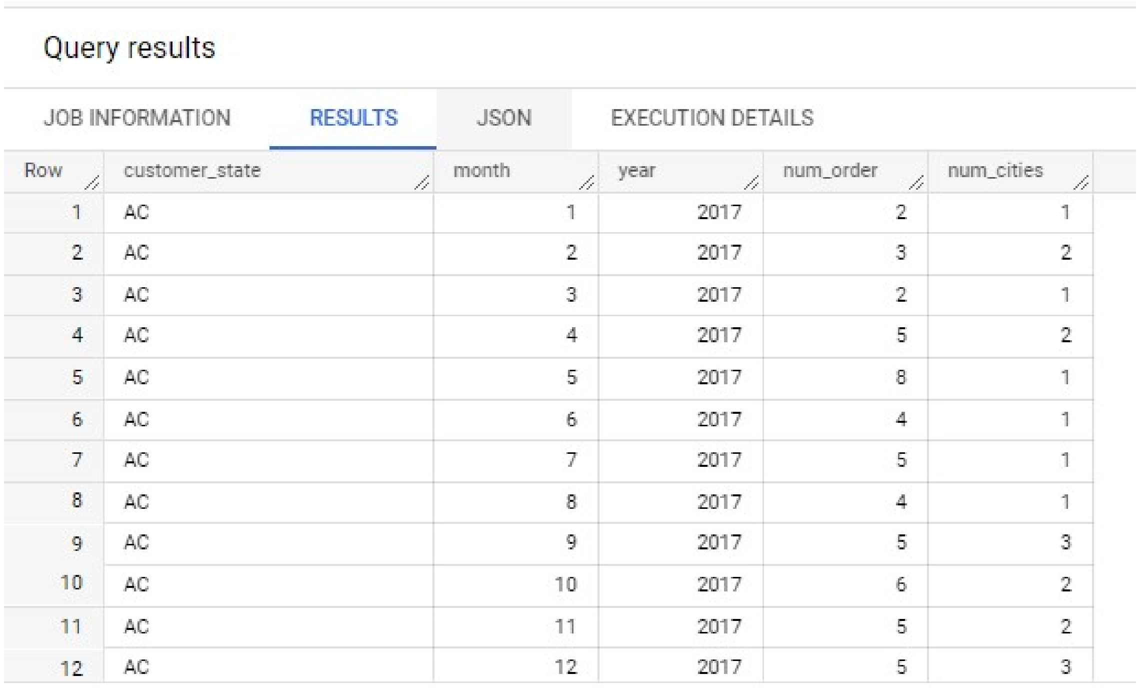
Row	part_of_day	num_orders
1	Afternoon (12 PM - 5 PM)	32212
2	Evening (5 PM - 9 PM)	24093
3	Morning (6 AM -12 PM)	22240
4	Night (9 PM - 5 AM)	20708
5	Dawn (5 AM - 6 AM)	188

Q3) Evolution of E-commerce orders in the Brazil region:

1) Get month on month orders by region, states:

State-wise:

```
with state_month as (select
order_id,
extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year,
customer_city,
customer_state
from `ecommerce.orders` as e
left join `ecommerce.customers` as c on e.customer_id = c.customer_id)

select
customer_state,
month,
year,
count(order_id) as num_order,
count(distinct customer_city) as num_cities
from state_month
group by customer_state, month, year
order by customer_state, year ,month
```



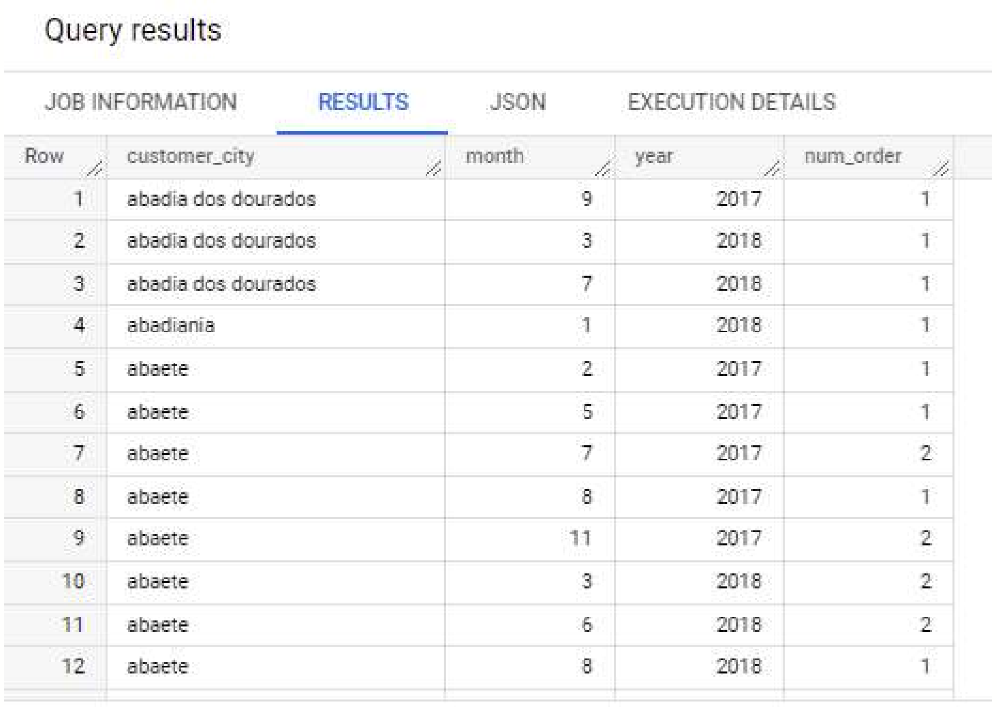
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	month	year	num_order	num_cities
1	AC	1	2017	2	1
2	AC	2	2017	3	2
3	AC	3	2017	2	1
4	AC	4	2017	5	2
5	AC	5	2017	8	1
6	AC	6	2017	4	1
7	AC	7	2017	5	1
8	AC	8	2017	4	1
9	AC	9	2017	5	3
10	AC	10	2017	6	2
11	AC	11	2017	5	2
12	AC	12	2017	5	3

City-wise:

```
with state_month as (select
order_id,
extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year,
customer_city,
customer_state
from `ecommerce.orders` as e
left join `ecommerce.customers` as c on e.customer_id = c.customer_id)

select
customer_city, month, year, count(order_id) as num_order
from state_month
group by customer_city, month, year
order by customer_city, year,  month
```

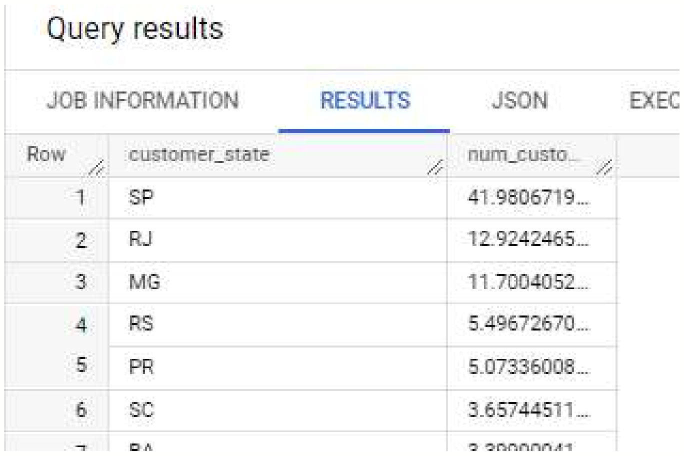

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_city	month	year	num_order
1	abadia dos dourados	9	2017	1
2	abadia dos dourados	3	2018	1
3	abadia dos dourados	7	2018	1
4	abadiania	1	2018	1
5	abaete	2	2017	1
6	abaete	5	2017	1
7	abaete	7	2017	2
8	abaete	8	2017	1
9	abaete	11	2017	2
10	abaete	3	2018	2
11	abaete	6	2018	2
12	abaete	8	2018	1

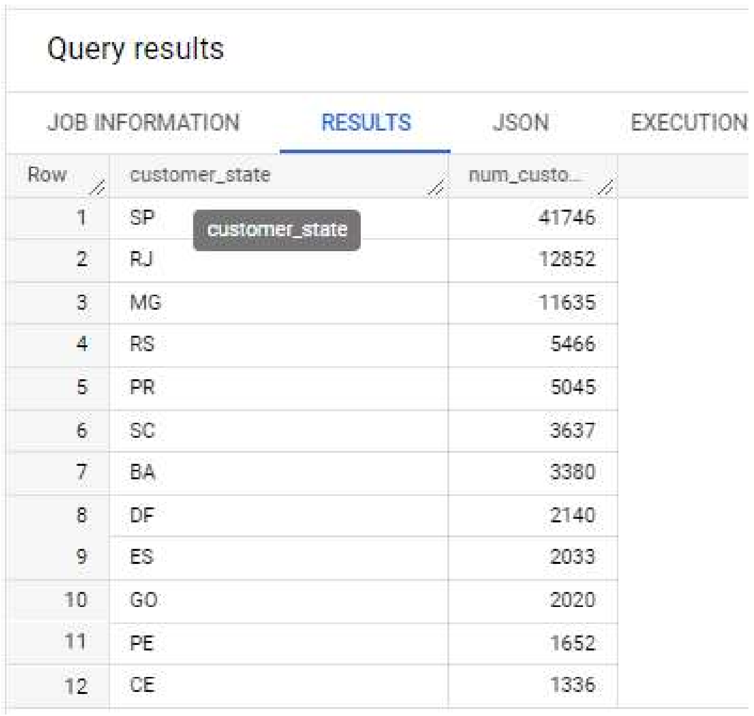
2) How are customers distributed in Brazil

```
select
customer_state,
count(customer_id) as num_customer
from `ecommerce.customers`
group by customer_state
order by num_customer desc
```


Query results

JOB INFORMATION RESULTS JSON EXEC

Row	customer_state	num_custo...
1	SP	41.9806719...
2	RJ	12.9242465...
3	MG	11.7004052...
4	RS	5.49672670...
5	PR	5.07336008...
6	SC	3.65744511...


Query results

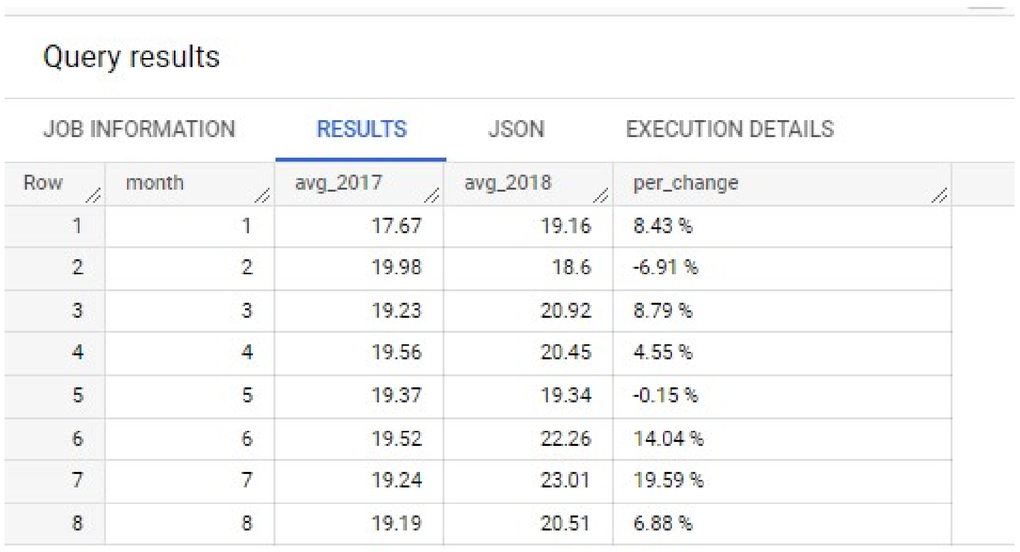
JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	customer_state	num_custo...
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020
11	PE	1652
12	CE	1336

Q4) Impact on Economy: Analyze the money movemented 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)

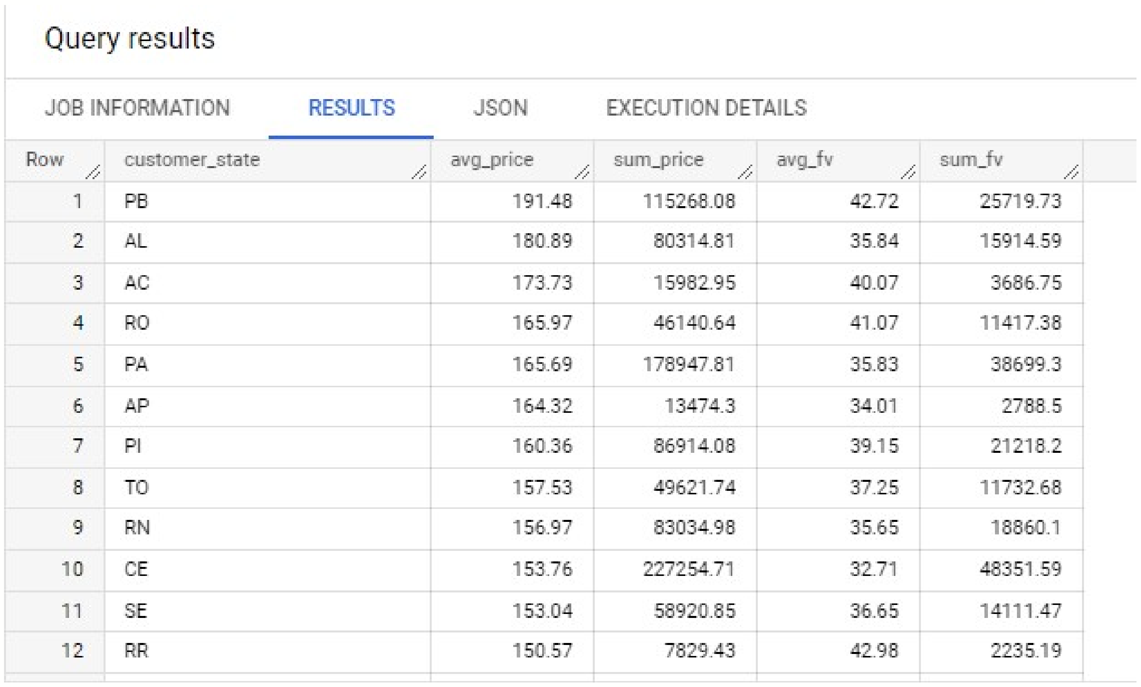
```
with cost as (select
extract (month from order_purchase_timestamp) as month,
extract (year from order_purchase_timestamp) as year,
freight_value
from `ecommerce.order_items` as oi
left join `ecommerce.orders` as o on oi.order_id = o.order_id
where extract (month from order_purchase_timestamp) between 1 and 8),

cost_2017 as (select
month, year, round(avg(freight_value),2) as avg_cost
from cost
group by year, month
having year = 2017
order by  month),

cost_2018 as (select
month, year, round(avg(freight_value),2) as avg_cost
from cost
group by year,month
having year = 2018
order by  month)
select c17.month,
c17.avg_cost as avg_2017,
c18.avg_cost as avg_2018,
concat(round((((c18.avg_cost -
c17.avg_cost)/c17.avg_cost)*100),2),' %') as per_change
from cost_2017 as c17
join cost_2018 as c18 on c17.month = c18.month
order by c17.month
```



Row	month	avg_2017	avg_2018	per_change
1	1	17.67	19.16	8.43 %
2	2	19.98	18.6	-6.91 %
3	3	19.23	20.92	8.79 %
4	4	19.56	20.45	4.55 %
5	5	19.37	19.34	-0.15 %
6	6	19.52	22.26	14.04 %
7	7	19.24	23.01	19.59 %
8	8	19.19	20.51	6.88 %

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

```
with cost as (select
customer_state,
price,
freight_value
from `ecommerce.order_items` as oi
left join `ecommerce.orders` as o on oi.order_id = o.order_id
left join `ecommerce.customers` as c on o.customer_id = c.customer_id)

select
customer_state,
round(avg(price),2) as avg_price,
round(sum(price),2) as sum_price,
round(avg(freight_value),2) as avg_fv,
round(sum(freight_value),2) as sum_fv
from cost
group by customer_state
order by avg_price desc
```



Query results

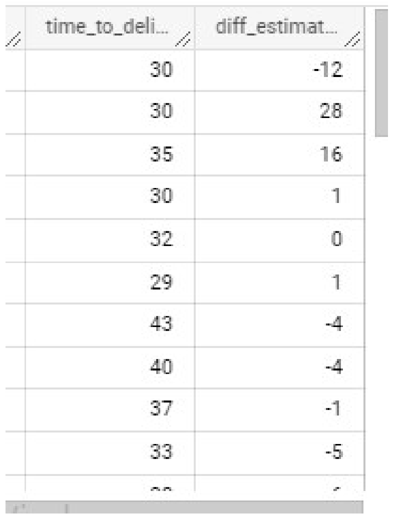
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	avg_price	sum_price	avg_fv	sum_fv
1	PB	191.48	115268.08	42.72	25719.73
2	AL	180.89	80314.81	35.84	15914.59
3	AC	173.73	15982.95	40.07	3686.75
4	RO	165.97	46140.64	41.07	11417.38
5	PA	165.69	178947.81	35.83	38699.3
6	AP	164.32	13474.3	34.01	2788.5
7	PI	160.36	86914.08	39.15	21218.2
8	TO	157.53	49621.74	37.25	11732.68
9	RN	156.97	83034.98	35.65	18860.1
10	CE	153.76	227254.71	32.71	48351.59
11	SE	153.04	58920.85	36.65	14111.47
12	RR	150.57	7829.43	42.98	2235.19

Q 5) Analysis on sales, freight and delivery time:

1) Calculate days between purchasing, delivering and estimated delivery
2) Create columns:
   a. time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
   b. diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

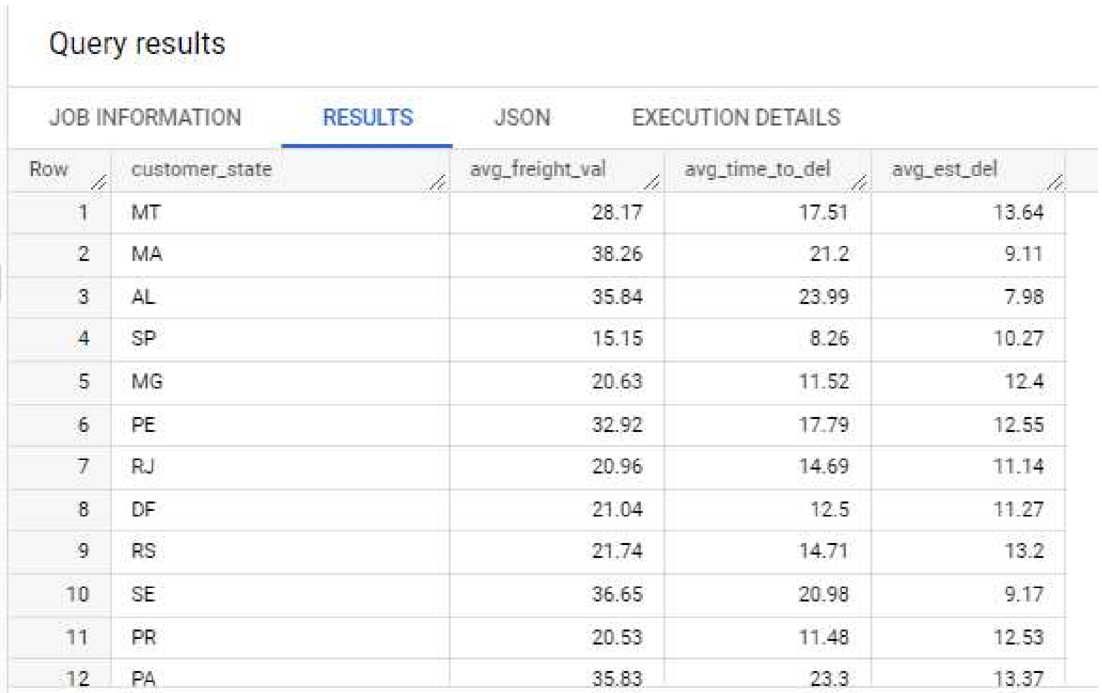
```
select
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_to_d
elivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as diff_es
timated_delivery
from `ecommerce.orders`
```



time_to_deli...	diff_estimat...
30	-12
30	28
35	16
30	1
32	0
29	1
43	-4
40	-4
37	-1
33	-5

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

```
with start as (select order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_to_eli
very,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as diff_es
timated_delivery, customer_state
from `ecommerce.orders` as o
join `ecommerce.customers` as c on o.customer_id = c.customer_id)

select
customer_state,
round(avg(freight_value),2) as avg_freight_val,
round(avg(time_to_delivery),2) as avg_time_to_del,
round(avg(diff_estimated_delivery),2) as avg_est_del
from `ecommerce.order_items` as oi
join start as s on oi.order_id = s.order_id
group by customer_state
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_val	avg_time_to_del	avg_est_del
1	MT	28.17	17.51	13.64
2	MA	38.26	21.2	9.11
3	AL	35.84	23.99	7.98
4	SP	15.15	8.26	10.27
5	MG	20.63	11.52	12.4
6	PE	32.92	17.79	12.55
7	RJ	20.96	14.69	11.14
8	DF	21.04	12.5	11.27
9	RS	21.74	14.71	13.2
10	SE	36.65	20.98	9.17
11	PR	20.53	11.48	12.53
12	PA	35.83	23.3	13.37

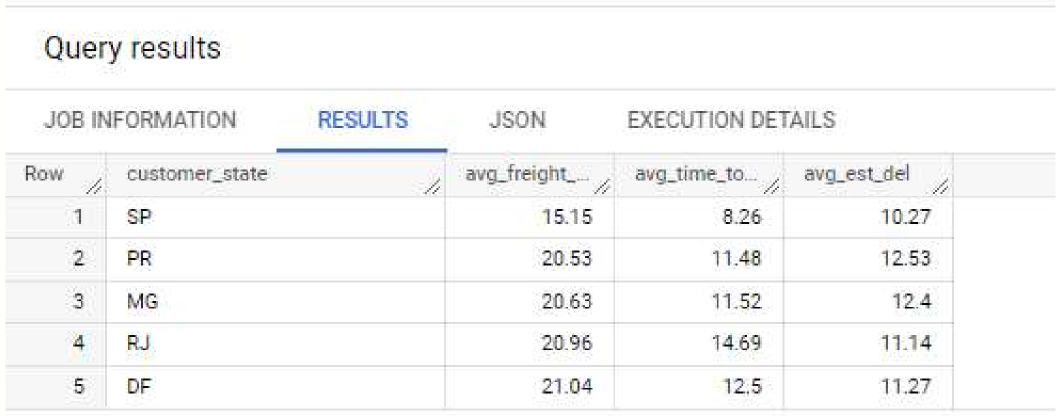
4) Sort the data to get the following:

1) Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Ascending:

```
with start as (select order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_t
o_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as diff
_estimated_delivery,
customer_state
from `ecommerce.orders` as o
join `ecommerce.customers` as c on o.customer_id = c.customer_id)


select
customer_state,
round(avg(freight_value),2) as avg_freight_val,
round(avg(time_to_delivery),2) as avg_time_to_del,
round(avg(diff_estimated_delivery),2) as avg_est_del
from `ecommerce.order_items` as oi
join start as s on oi.order_id = s.order_id
```

```
group by customer_state
order by avg_freight_val
limit 5
```



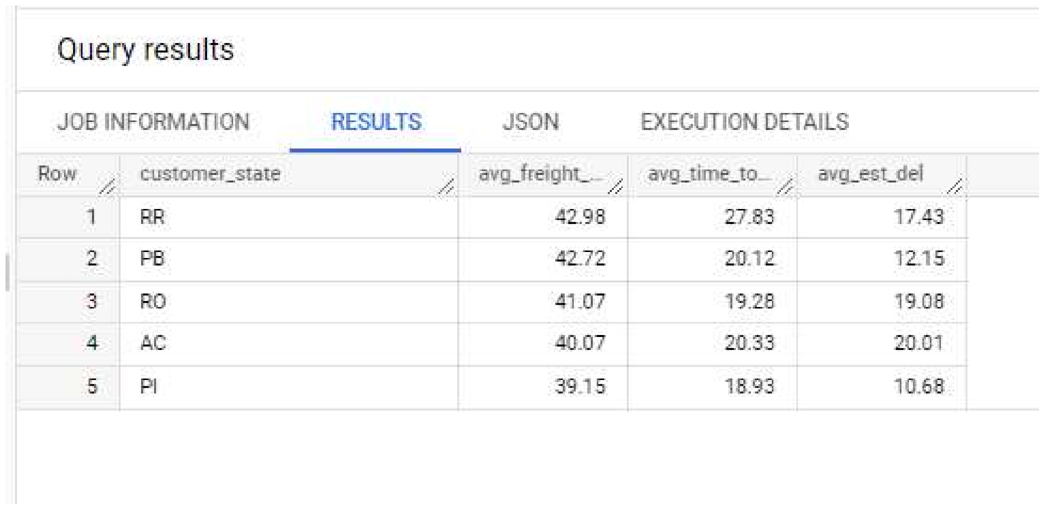
Query results

JOB INFORMATION   RESULTS   JSON   EXECUTION DETAILS

Row	customer_state	avg_freight_...	avg_time_to...	avg_est_del
1	SP	15.15	8.26	10.27
2	PR	20.53	11.48	12.53
3	MG	20.63	11.52	12.4
4	RJ	20.96	14.69	11.14
5	DF	21.04	12.5	11.27

Descending:

```
with start as (select order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_t
o_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as diff
_estimated_delivery,
customer_state
from `ecommerce.orders` as o
join `ecommerce.customers` as c on o.customer_id = c.customer_id)


select
customer_state,
round(avg(freight_value),2) as avg_freight_val,
round(avg(time_to_delivery),2) as avg_time_to_del,
round(avg(diff_estimated_delivery),2) as avg_est_del
from `ecommerce.order_items` as oi
join start as s on oi.order_id = s.order_id
group by customer_state
order by avg_freight_val desc
limit 5
```


Query results

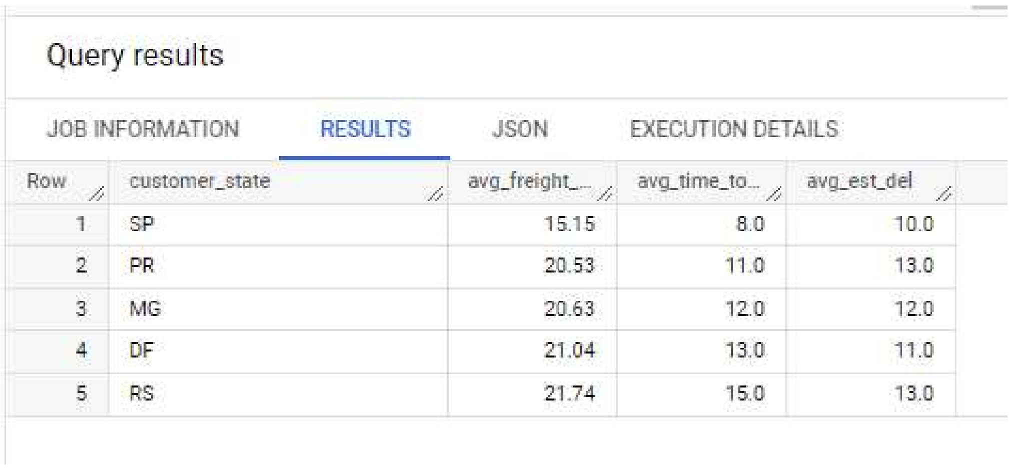
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...	avg_time_to...	avg_est_del
1	RR	42.98	27.83	17.43
2	PB	42.72	20.12	12.15
3	RO	41.07	19.28	19.08
4	AC	40.07	20.33	20.01
5	PI	39.15	18.93	10.68

2) Top 5 states with highest/lowest average time to delivery

Ascending:

```
with start as (select order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_t
o_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as diff
_estimated_delivery,
customer_state
from `ecommerce.orders` as o
join `ecommerce.customers` as c on o.customer_id = c.customer_id)


select
customer_state,
round(avg(freight_value),2) as avg_freight_val,
round(avg(time_to_delivery),0) as avg_time_to_del,
round(avg(diff_estimated_delivery),0) as avg_est_del
from `ecommerce.order_items` as oi
join start as s on oi.order_id = s.order_id
group by customer_state
order by avg_time_to_del
limit 5
```



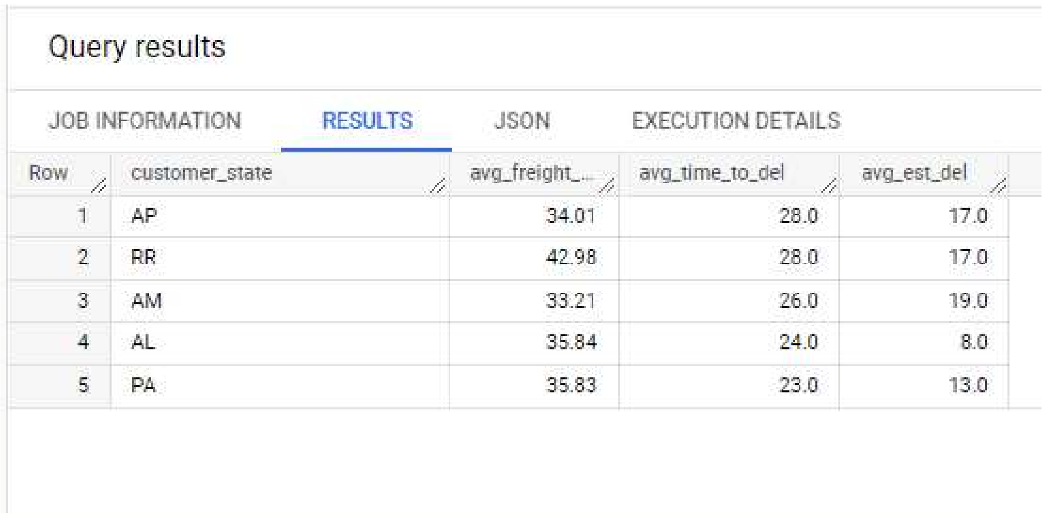
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...	avg_time_to...	avg_est_del
1	SP	15.15	8.0	10.0
2	PR	20.53	11.0	13.0
3	MG	20.63	12.0	12.0
4	DF	21.04	13.0	11.0
5	RS	21.74	15.0	13.0

Descending:

```
with start as (select order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_t
o_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as diff
_estimated_delivery,
customer_state
from `ecommerce.orders` as o
join `ecommerce.customers` as c on o.customer_id = c.customer_id)


select
customer_state,
round(avg(freight_value),2) as avg_freight_val,
round(avg(time_to_delivery),0) as avg_time_to_del,
round(avg(diff_estimated_delivery),0) as avg_est_del
from `ecommerce.order_items` as oi
join start as s on oi.order_id = s.order_id
group by customer_state
order by avg_time_to_del desc
limit 5
```


Query results

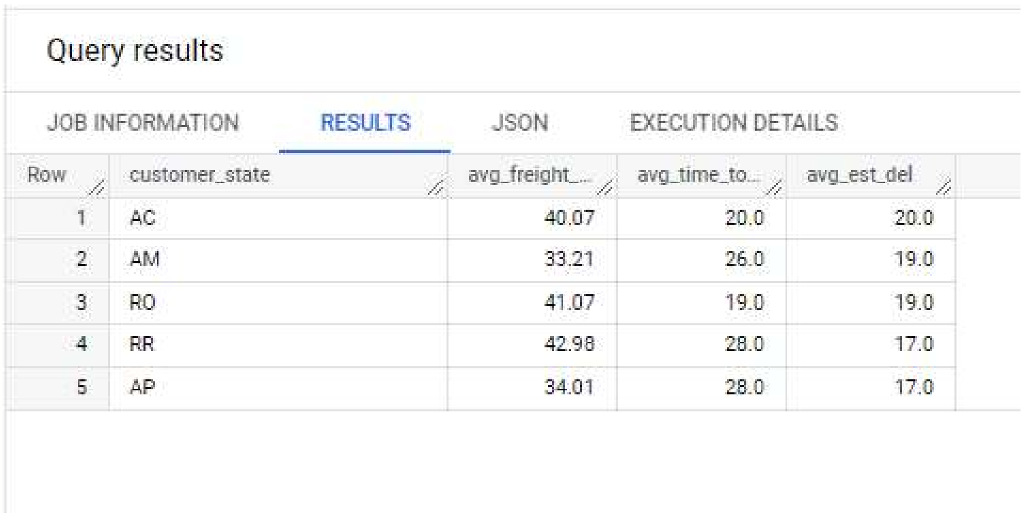
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...	avg_time_to_del	avg_est_del
1	AP	34.01	28.0	17.0
2	RR	42.98	28.0	17.0
3	AM	33.21	26.0	19.0
4	AL	35.84	24.0	8.0
5	PA	35.83	23.0	13.0

3) Top 5 states where delivery is really fast/ not so fast compared to estimated date

Fast:

```
with start as (select order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_t
o_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as diff
_estimated_delivery,
customer_state
from `ecommerce.orders` as o
join `ecommerce.customers` as c on o.customer_id = c.customer_id)


select
customer_state,
round(avg(freight_value),2) as avg_freight_val,
round(avg(time_to_delivery),0) as avg_time_to_del,
round(avg(diff_estimated_delivery),0) as avg_est_del
from `ecommerce.order_items` as oi
join start as s on oi.order_id = s.order_id
group by customer_state
order by avg_est_del desc
limit 5
```

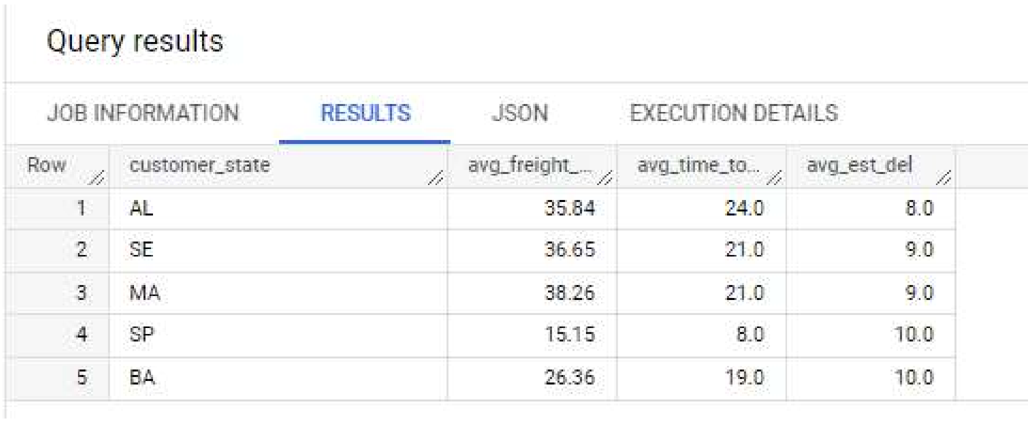

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...	avg_time_to...	avg_est_del
1	AC	40.07	20.0	20.0
2	AM	33.21	26.0	19.0
3	RO	41.07	19.0	19.0
4	RR	42.98	28.0	17.0
5	AP	34.01	28.0	17.0

slow:

```
with start as (select order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_t
o_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date,day) as diff
_estimated_delivery,
customer_state
from `ecommerce.orders` as o
join `ecommerce.customers` as c on o.customer_id = c.customer_id)

select
customer_state,
round(avg(freight_value),2) as avg_freight_val,
round(avg(time_to_delivery),0) as avg_time_to_del,
round(avg(diff_estimated_delivery),0) as avg_est_del
from `ecommerce.order_items` as oi
join start as s on oi.order_id = s.order_id
group by customer_state
order by avg_est_del
limit 5
```


Query results

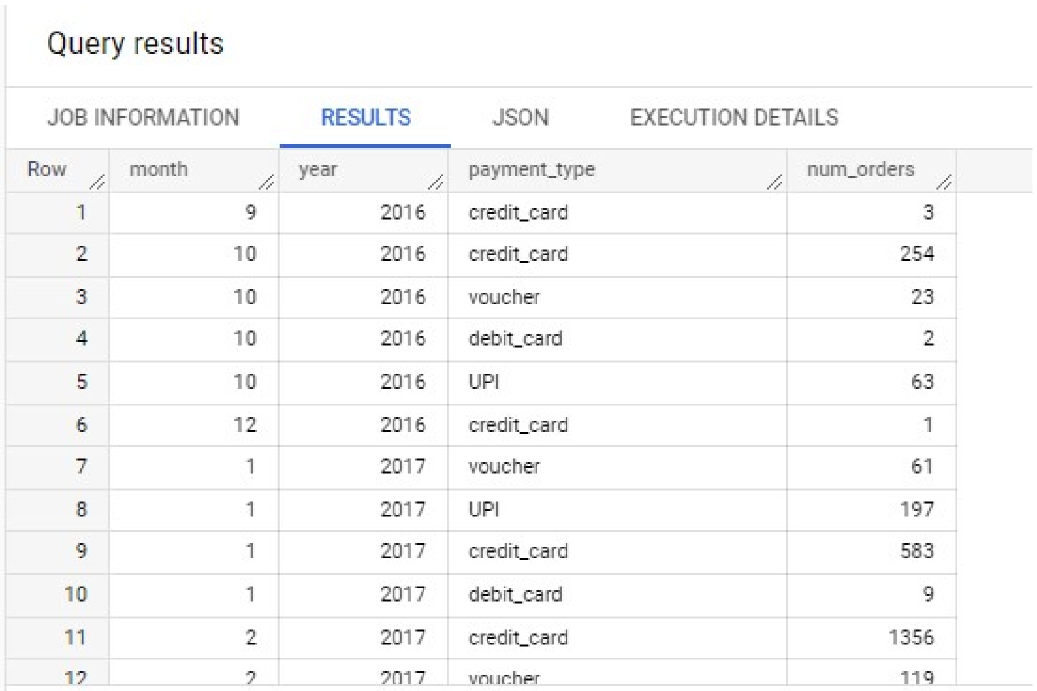
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...	avg_time_to...	avg_est_del
1	AL	35.84	24.0	8.0
2	SE	36.65	21.0	9.0
3	MA	38.26	21.0	9.0
4	SP	15.15	8.0	10.0
5	BA	26.36	19.0	10.0

Q 6) Payment type analysis

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

```
with raw as (select p.order_id,
payment_sequential,
payment_type,
extract (month from order_purchase_timestamp) as month,
extract (year from order_purchase_timestamp) as year
from `ecommerce.payments` as p
left join `ecommerce.orders` as o on p.order_id = o.order_id)

select
month, year, payment_type,
 count(order_id) as num_orders
from raw
group by payment_type, month ,year
order by year, month
```



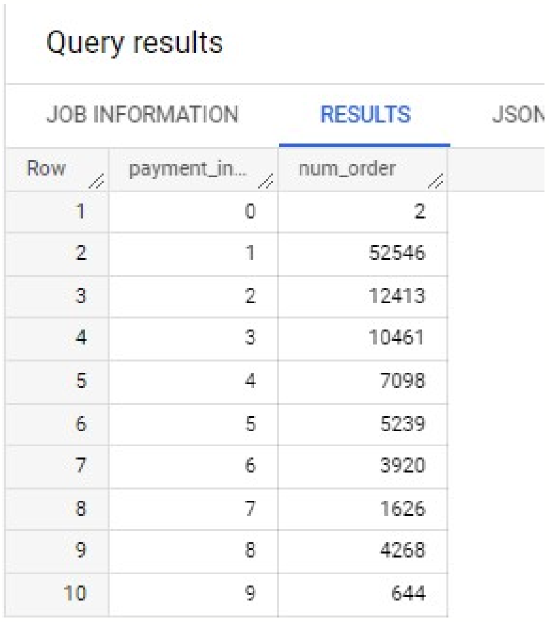
Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	month	year	payment_type	num_orders
1	9	2016	credit_card	3
2	10	2016	credit_card	254
3	10	2016	voucher	23
4	10	2016	debit_card	2
5	10	2016	UPI	63
6	12	2016	credit_card	1
7	1	2017	voucher	61
8	1	2017	UPI	197
9	1	2017	credit_card	583
10	1	2017	debit_card	9
11	2	2017	credit_card	1356
12	2	2017	voucher	119

2) Distribution of payment installments and count of orders

```
select
payment_installments,
count(order_id) as num_order
from `ecommerce.payments`
group by payment_installments
```



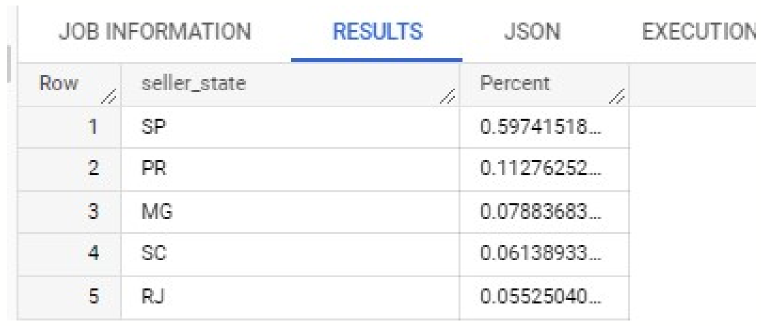
Query results

JOB INFORMATION RESULTS JSON

Row	payment_in...	num_order
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

Seller distribution:

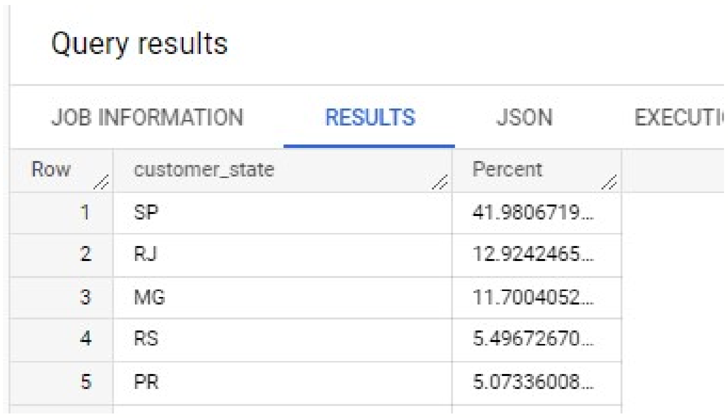
```
select seller_state,
count(seller_id)/(select count(seller_id) from `ecommerce.sellers`) as Percent
from `ecommerce.sellers`
group by seller_state
order by Percent desc
```



Row	seller_state	Percent
1	SP	0.59741518...
2	PR	0.11276252...
3	MG	0.07883683...
4	SC	0.06138933...
5	RJ	0.05525040...

Customer distribution:

```
select
customer_state,
(count(customer_id)/ ( select count(distinct customer_id)
 from `ecommerce.customers`
)) * 100 as Percent
from `ecommerce.customers`
group by customer_state
order by Percent desc
```



Row	customer_state	Percent
1	SP	41.9806719...
2	RJ	12.9242465...
3	MG	11.7004052...
4	RS	5.49672670...
5	PR	5.07336008...

Insights:

- *Data is available for year 2016,2017, and 2018*
- *Num of states = 27 and Num of Cities = 4196*
- *On year to year basis, There is increase in number of orders*
- *On month to month basis, highest orders in August and Least orders in Sept from overall available data.*
- *Assuming that Ecommerce began in end of 2016, there are very less orders there. Following year, 2017 has positive growth and was having uptrend with peak in November. 2018 is Downtrend, where orders dropped hugely in September.*
- *From available data, there is uptrend from Sept 2016 to Sept 2018 and then there is drop observed in Sept 2018. We assume that there can be cycle repeating from Sept to Sept every two years.*
- *Citizens in Brazil purchase more in following time zones:*
  - *Afternoon > Evening > Morning > Night >Dawn*
- *Majority of Customers are from State of São Paulo (42%), State of Rio de Janeiro (13%) and State of Minas Gerais (12%).*
- *83% Sellers are from SP, PR, MG and SC states*
- *From 2017 to 2018, Except Feb and May, else all months are facing increase in average order cost*
- *Value of state, having Lowest average freight value, is $15*
- *Value of state, having Highest average freight value, is $43*
- *Value of state, having Lowest average time to delivery, is 8 days*
- *Value of state, having Highest average time to delivery, is 28 days*
- *Most transactions are of Credit card and UPI type*
- *Most orders are of one installment EMI type.*

Recommendations:

- Customers are from 27 states but 67% orders so far are from customers of three states (RJ, SP, and MG) only. There is dire need of Proper Marketing in all rest of 24 states in order to increase the number of orders for our Ecommerce.
- There is need to merge more sellers in the network of Target Ecommerce so that we can reach too far away places in less time.
- Due to Agglomeration of Sellers there is increase in freight value for an order for other states. This will get reduce by joining different sellers in different states.
- By increasing networking, warehousing, infrastructure and logistics we can reduce the time to delivery to all states in Brazil.
- Because average delivery time is more (starting from 8 days!), less customers are willing to purchase from us online. By working on above points we hope to serve our customers more effectively.