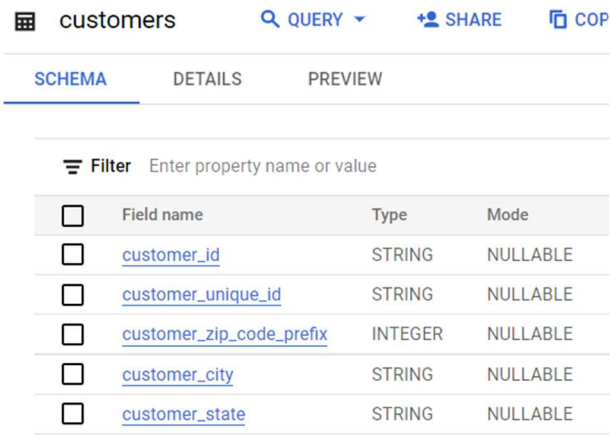
Business Case Study – Target

Q1: Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

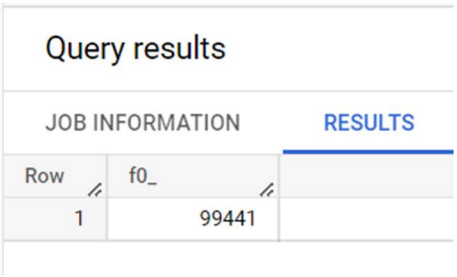
1. Data type of columns in a Datatype of Table
2. Time period for which the data is given
3. Cities and States covered in the dataset

Data type of columns in a Datatype of Table: customers



Field name	Type	Mode
customer_id	STRING	NULLABLE
customer_unique_id	STRING	NULLABLE
customer_zip_code_prefix	INTEGER	NULLABLE
customer_city	STRING	NULLABLE
customer_state	STRING	NULLABLE

```
select
count(*)
from ecommerce.customers
```



Observation: There are around 99441 rows in customers table

Cities and States covered in the dataset:

```
select
distinct
customer_city
from ecommerce.customers
```

```
select
distinct
customer_city
from ecommerce.customers
```

Press Alt+F1

Query results

SAVE RESULTS | EXPLO

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_city
1	acu
2	ico
3	ipe
4	ipu
5	ita
6	itu
7	jau
8	luz
9	poa
10	uba
11	una

Results per page: 50 | 1 – 50 of 4119

Observation: There are around 4119 distinct customer cities.

```
select
distinct
customer_state
from ecommerce.customers
```

Query results

SAVE RESULTS | EXP

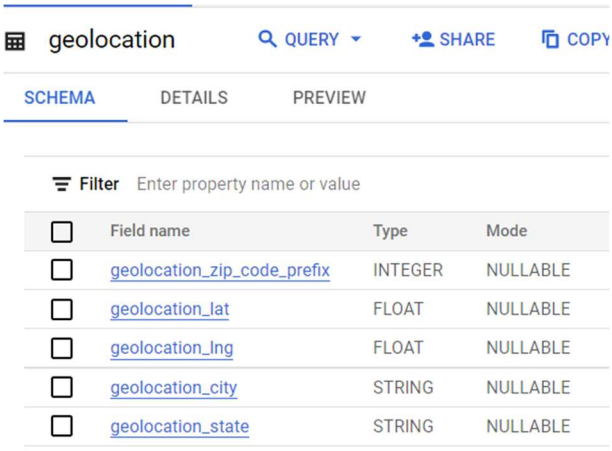
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state
1	RN
2	CE
3	RS
4	SC
5	SP
6	MG
7	BA
8	RJ
9	GO
10	MA
11	PE

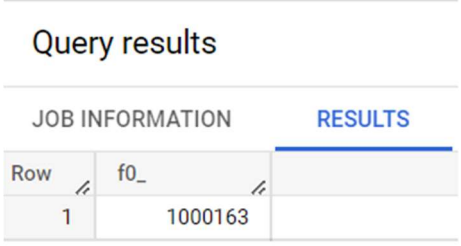
Results per page: 50 | 1 – 27 of 27

Observation: There are around 27 distinct customer state.

Data type of columns in a Datatype of Table: **geolocation**



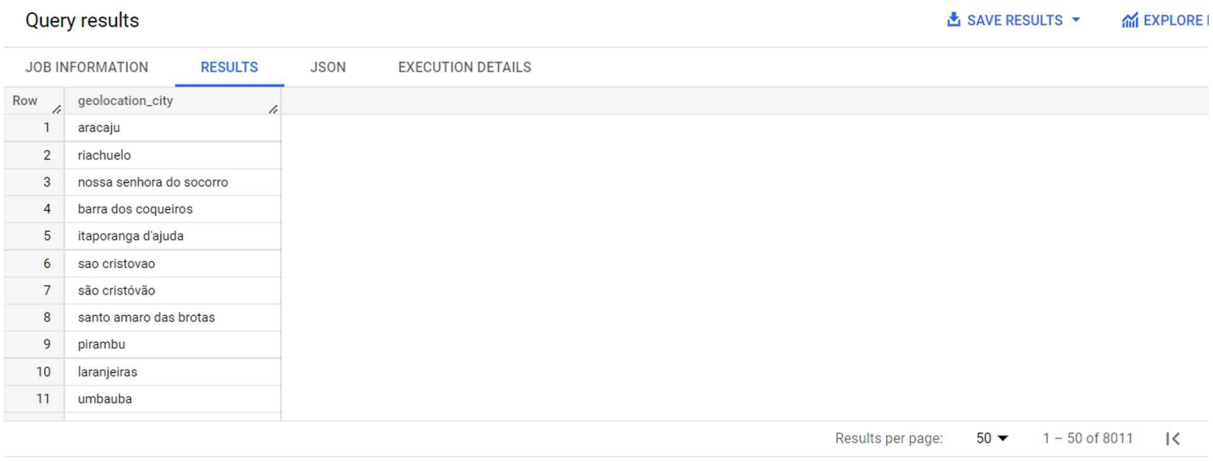

```
select
count(*)
from `ecommerce.geolocation`
```



Observation: There are around 1000163 rows in geolocation table

Cities and States covered in the dataset:

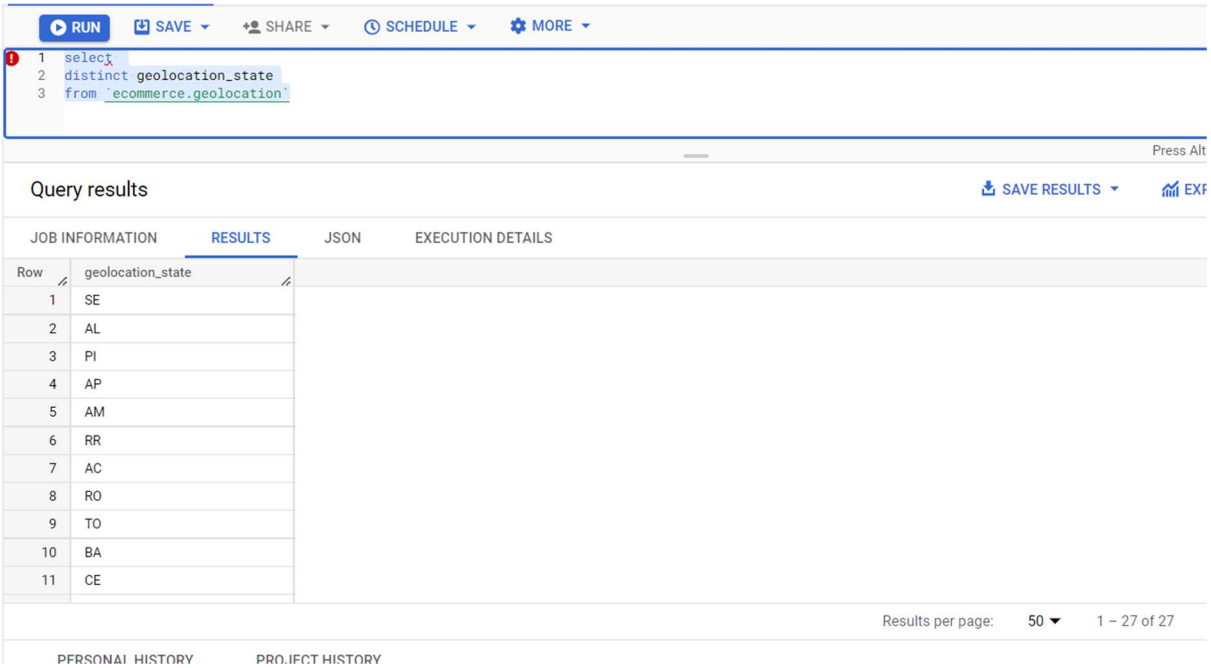
```
select
distinct geolocation_city
from `ecommerce.geolocation`
```



Observation: There are around 8011 distinct geolocation cities

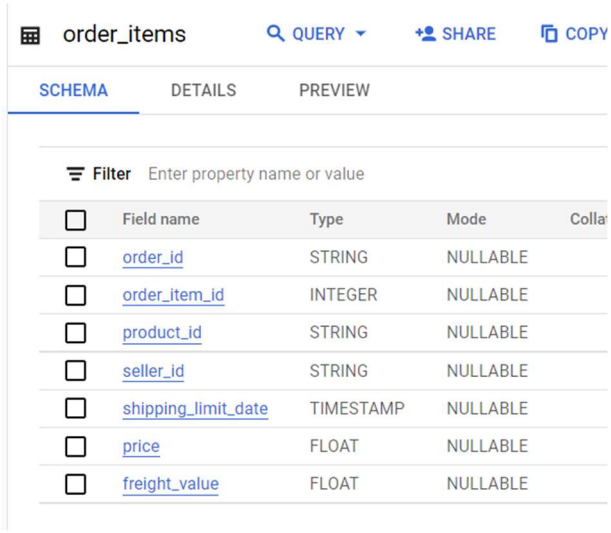
```
select
distinct geolocation_state
```

```
from `ecommerce.geolocation`
```

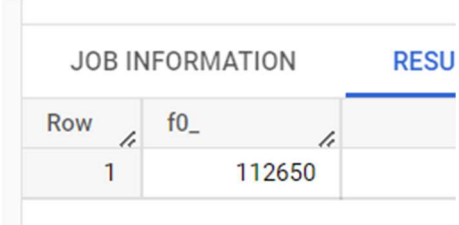


Observation: There are around 27 distinct geolocation states

Data type of columns in a Datatype of Table: order_items



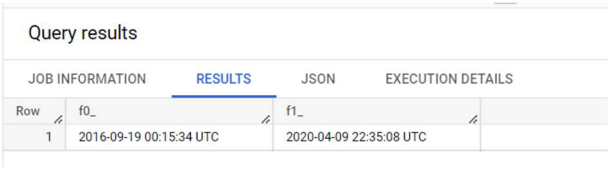

```
select
count(*)
from `ecommerce.order_items`
```



Observation: There are around 112650 rows in order_items table

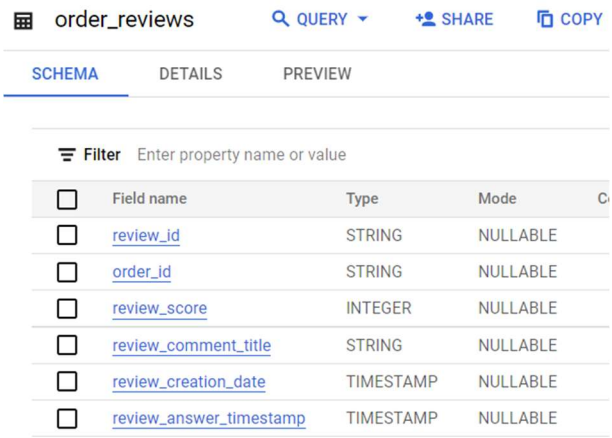
Time period for which the data is given: Shipping_limit_date

```
select min(shipping_limit_date), max(shipping_limit_date) from `ecommerce.order_items
```



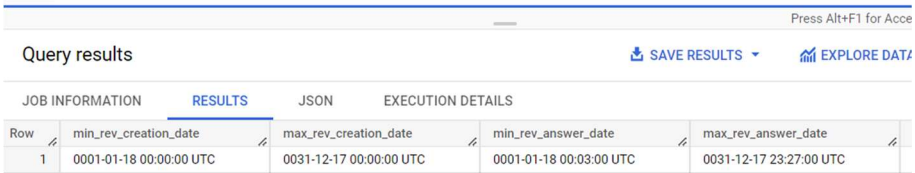
Observation: `Shipping_limit_date` range is between 19-09-2016 to 09-04-2020

Data type of columns in a Datatype of Table: order_reviews



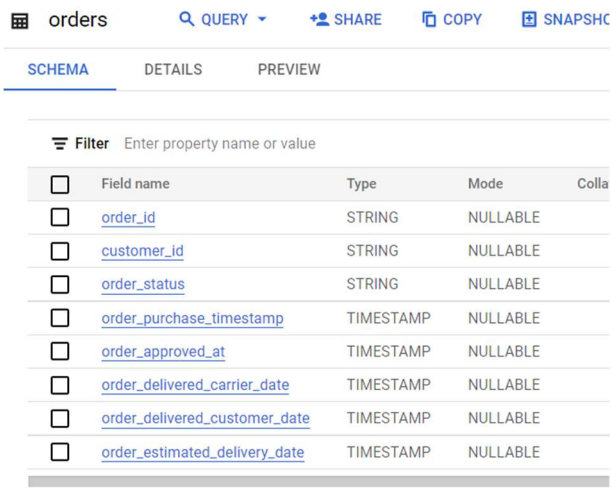
Time period for which the data is given: review_creation_date , review_answer_timestamp

```
select
min(review_creation_date) as min_rev_creation_date,
max(review_creation_date) as max_rev_creation_date,
min(review_answer_timestamp) as min_rev_answer_date,
max(review_answer_timestamp) as max_rev_answer_date
from ecommerce.order_reviews
```

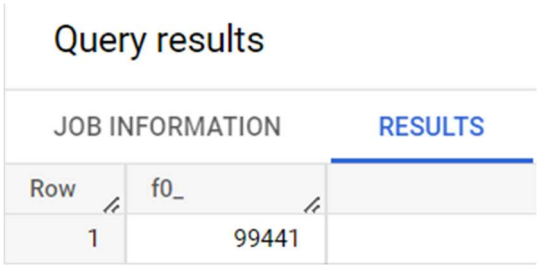


Observation: Range for review and answer is between: 0001-01-18 to 0031-12-17 but the year in the date is not correct hence we cannot consider it for range

Data type of columns in a Datatype of Table: orders



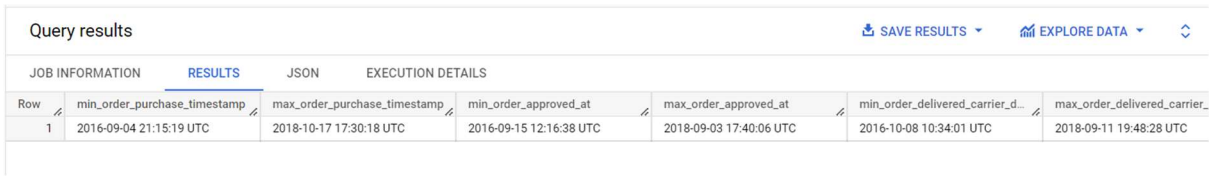

```
select
count(*)
from `ecommerce.orders`
```

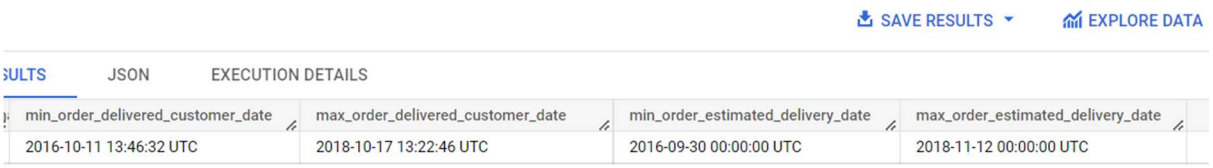


Observation: There are around 99441 rows in orders table

Time period for which the data is given:

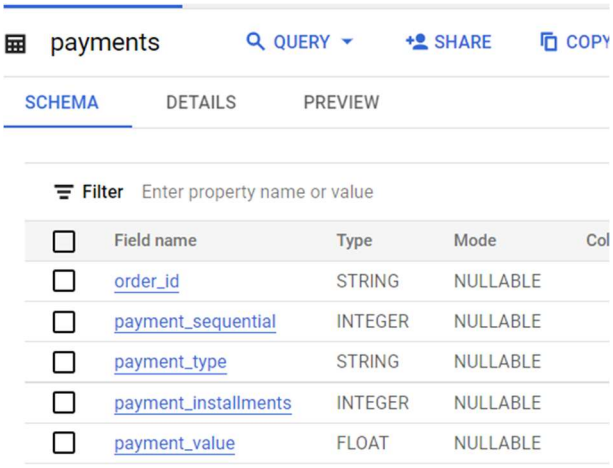
```
select
min(order_purchase_timestamp) as min_order_purchase_timestamp,
max(order_purchase_timestamp) as max_order_purchase_timestamp,
min(order_approved_at) as min_order_approved_at,
max(order_approved_at) as max_order_approved_at,
min(order_delivered_carrier_date) as min_order_delivered_carrier_date,
max(order_delivered_carrier_date) as max_order_delivered_carrier_date,
min(order_delivered_customer_date) as min_order_delivered_customer_date,
max(order_delivered_customer_date) as max_order_delivered_customer_date,
min(order_estimated_delivery_date) as min_order_estimated_delivery_date,
max(order_estimated_delivery_date) as max_order_estimated_delivery_date
from ecommerce.orders
```



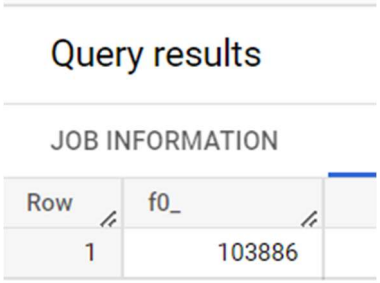


Observation: Overall date range for order purchase and delivery lies between 2016-09-04 to 2018-11-12

Data type of columns in a Datatype of Table: payments

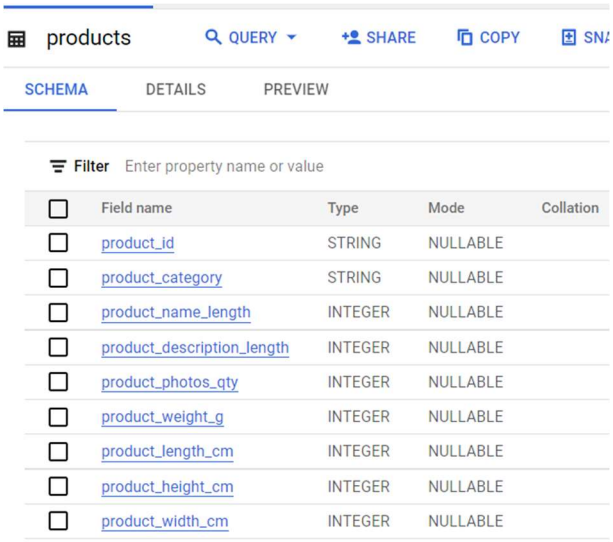



```
select
count(*)
from `ecommerce.payments`
```



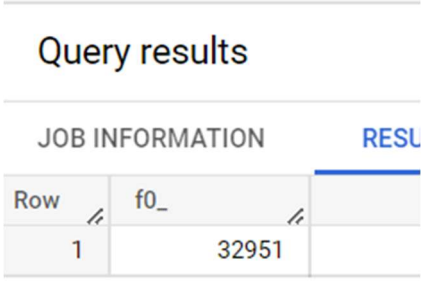
Observation: There are around 103886 rows in payments

Data type of columns in a Datatype of Table: products



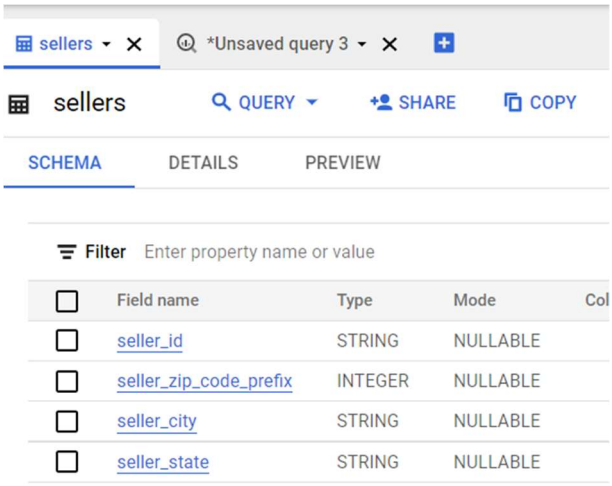
Field name	Type	Mode	Collation
product_id	STRING	NULLABLE	
product_category	STRING	NULLABLE	
product_name_length	INTEGER	NULLABLE	
product_description_length	INTEGER	NULLABLE	
product_photos_qty	INTEGER	NULLABLE	
product_weight_g	INTEGER	NULLABLE	
product_length_cm	INTEGER	NULLABLE	
product_height_cm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

```
select
count(*)
from `ecommerce.products`
```



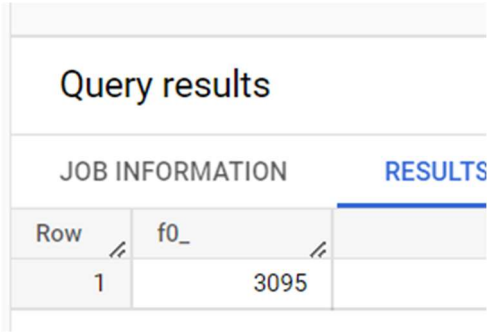
Observation: There are around 32951 rows in products table

Data type of columns in a Datatype of Table: **sellers**



Field name	Type	Mode	Col
seller_id	STRING	NULLABLE	
seller_zip_code_prefix	INTEGER	NULLABLE	
seller_city	STRING	NULLABLE	
seller_state	STRING	NULLABLE	

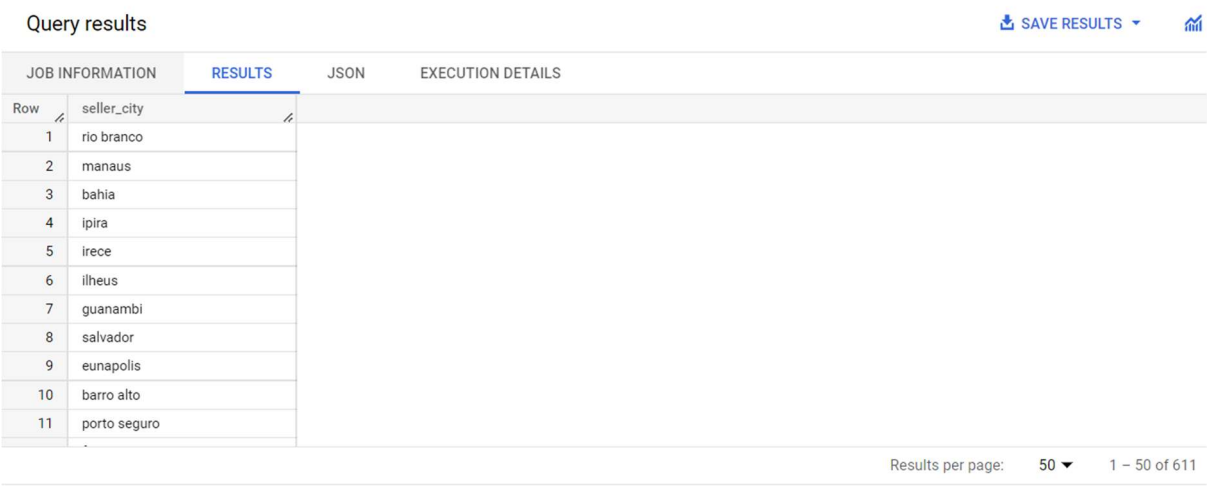
```
select
count(*)
from `ecommerce.sellers`
```



Observation: There are around 3095 rows in sellers table

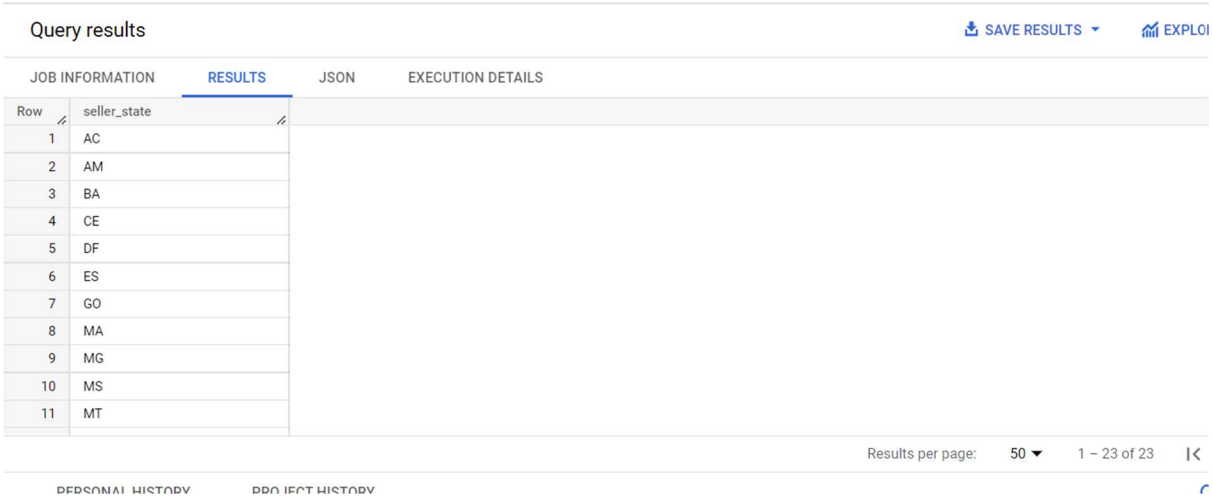
Cities and States covered in the dataset:

```
select
distinct
seller_city
from ecommerce.sellers
```



Observation: There are around 611 distinct seller cities

```
select
distinct
seller_state
from ecommerce.sellers
```



Observation: There are around 23 distinct seller states

==========

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?

```
select * from
(select
extract(month FROM ord.order_purchase_timestamp) as purcahse_month,
count(ord.order_id)
from `ecommerce.orders` as ord
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
group by extract(month FROM ord.order_purchase_timestamp)) as x
order by x.purcahse_month
```

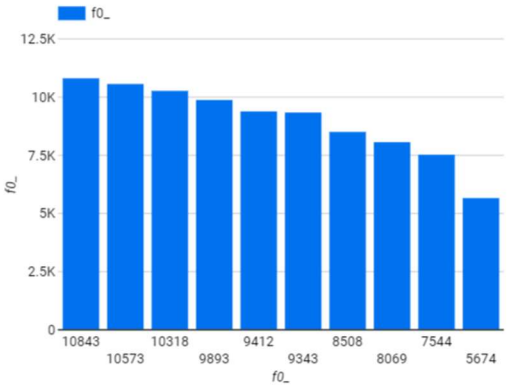
Query results

JOB INFORMATION | RESULTS | JSON

Row	purcahse_m...	f0_
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

	purcahse_month	f0_
1.	August	10,843
2.	May	10,573
3.	July	10,318
4.	March	9,893
5.	June	9,412
6.	April	9,343
7.	February	8,508
8.	January	8,069
9.	November	7,544
10.	December	5,674
11.	October	4,959

1 - 12 / 12



Observation: August, May and July month has more orders (greater than 10300 orders) in Brazil city.

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

Parts of the Day

Dawn – 4 am to 6 am. (4 to 6)

Morning - 6 am to 12 pm (noon). (6 to 12)

Afternoon - 12 pm to 5 pm. (12 to 17)

Evening - 5 pm to 9 pm. (17 to 21)

Night - 9 pm to 4 am. (21 to 4)

```sql
select
count(y.order_id),
y.time_bins from
(select
x.order_id,
case
  when x.purcahse_time between '04:00:00' and '05:59:59'
  then "Dawn"
  when x.purcahse_time between '06:00:00' and '11:59:59'
  then "Morning"
  when x.purcahse_time between '12:00:00' and '16:59:59'
  then "Afternoon"
  when x.purcahse_time between '17:00:00' and '20:59:59'
  then "Evening"
  when x.purcahse_time between '21:00:00' and '23:59:59'
  then "Night"
  when x.purcahse_time between '00:00:00' and '03:59:59'
  then "Night"
end as time_bins
from
(select
extract(time FROM ord.order_purchase_timestamp) as purcahse_time,
ord.order_id
from `ecommerce.orders` as ord
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id) as x) as y
group by y.time_bins
```

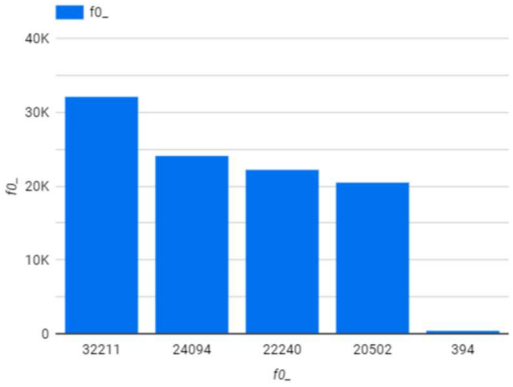
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTI

Row	f0_	time_bins
1	22240	Morning
2	20502	Night
3	24094	Evening
4	32211	Afternoon
5	394	Dawn

	time_bins	f0_
1.	Afternoon	32,211
2.	Evening	24,094
3.	Morning	22,240
4.	Night	20,502
5.	Dawn	394

1 - 5 / 5



Observation: Afternoon has more orders (around 32211 orders) in Brazil city.

==========

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

1. Get month on month orders by region, states

```
select * from
(select
extract(month FROM ord.order_purchase_timestamp) as purcahse_month,
cust.customer_city,
cust.customer_state,
count(ord.order_id) as count1
from `ecommerce.orders` as ord
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
group by extract(month FROM ord.order_purchase_timestamp), cust.customer_city, cust.custome
r_state) as x
order by x.count1 desc, x.purcahse_month
```

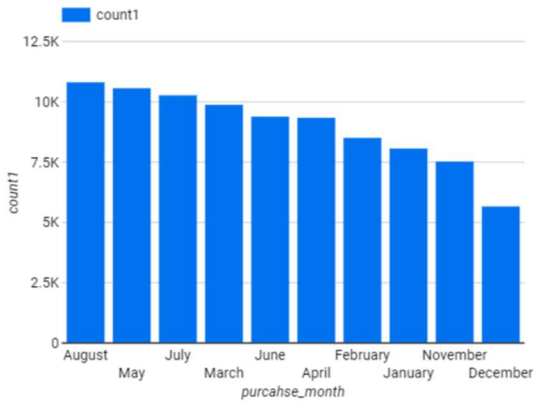
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	purcahse_m...	customer_city	customer_state	count1
1	8	sao paulo	SP	1954
2	5	sao paulo	SP	1743
3	7	sao paulo	SP	1625
4	3	sao paulo	SP	1533
5	6	sao paulo	SP	1532
6	4	sao paulo	SP	1467
7	2	sao paulo	SP	1272
8	1	sao paulo	SP	1195
9	11	sao paulo	SP	1118
10	12	sao paulo	SP	840
11	8	rio de janeiro	RJ	770
12	7	rio de janeiro	RJ	714
13	5	rio de janeiro	RJ	700

	customer_city	count1
1.	sao paulo	15,540
2.	rio de janeiro	6,882
3.	belo horizonte	2,773
4.	brasilia	2,131
5.	curitiba	1,521
6.	campinas	1,444
7.	porto alegre	1,379
8.	salvador	1,245
9.	guarulhos	1,189
10.	sao bernardo do campo	938
11.	niteroi	849
12.	santo andre	797
13.	osasco	746
14.	santos	713
15.	goiania	692
16.	sao jose dos campos	691
17.	fortaleza	654
18.	sorocaba	633
19.	recife	613
20.	florianopolis	570

1 - 50 / 4119



Observation:

Sao Paulo city in Brazil has highest orders of 15540

Rio de janerio city in Brazil has second highest order of 6882

Belo horizonte city in Brazil has third highest order of 2773

Sao Paulo and Rio de Janeiro city in Brazil has highest orders in the month of August

Belo horizonte city in Brazil has highest orders in the month of March and followed by August

Based on this it looks like August month has more orders

2. How are customers distributed in Brazil

```
select * from
(select
cust.customer_city,
count(ord.order_id) as count1
from `ecommerce.orders` as ord
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
group by cust.customer_city ) as x
order by x.count1 desc
```

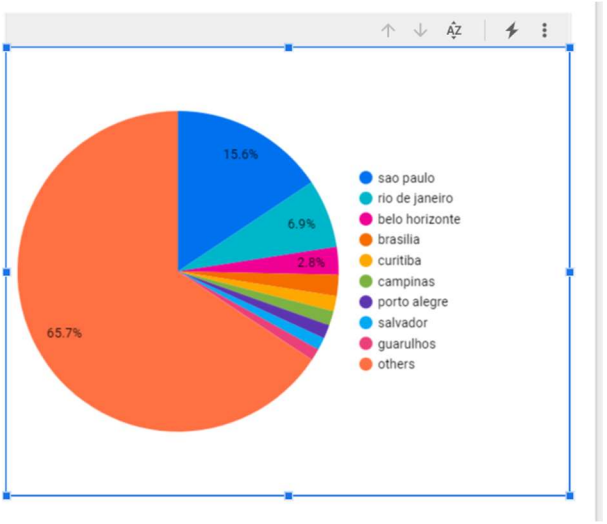
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_city	count1
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444
7	porto alegre	1379
8	salvador	1245

	customer_city	count1
1.	sao paulo	15,540
2.	rio de janeiro	6,882
3.	belo horizonte	2,773
4.	brasilia	2,131
5.	curitiba	1,521
6.	campinas	1,444
7.	porto alegre	1,379
8.	salvador	1,245
9.	guarulhos	1,189
10.	sao bernardo do campo	938
11.	niteroi	849

1 - 50 / 4119



Observation:

Sao Paulo city in Brazil has highest orders of 15540

Rio de janerio city in Brazil has second highest order of 6882

Belo horizonte city in Brazil has third highest order of 2773

==========

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 price_17 as
(select
extract(month FROM ord.order_purchase_timestamp) as purchase_month,
extract(year FROM ord.order_purchase_timestamp) as purchase_year,
round(sum(items.price), 2) as price_2017
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
where extract(month FROM ord.order_purchase_timestamp) not in (9,10,11,12)
and extract(year FROM ord.order_purchase_timestamp) in (2017)
group by extract(month FROM ord.order_purchase_timestamp), extract(year FROM ord.order_purc
hase_timestamp)),

price_18 as
(select
extract(month FROM ord.order_purchase_timestamp) as purchase_month,
extract(year FROM ord.order_purchase_timestamp) as purchase_year,
round(sum(items.price), 2) as price_2018
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
where extract(month FROM ord.order_purchase_timestamp) not in (9,10,11,12)
and extract(year FROM ord.order_purchase_timestamp) in (2018)
```

```
group by extract(month FROM ord.order_purchase_timestamp), extract(year FROM ord.order_purc
hase_timestamp))

select price_17.purchase_month, price_17.price_2017, price_18.price_2018,
(price_18.price_2018 - price_17.price_2017) as price_dif,
(round((price_18.price_2018 - price_17.price_2017)*100/(price_18.price_2018 + price_17.pric
e_2017))) as per_inc_price
from price_17 inner join price_18
on price_17.purchase_month = price_18.purchase_month
```

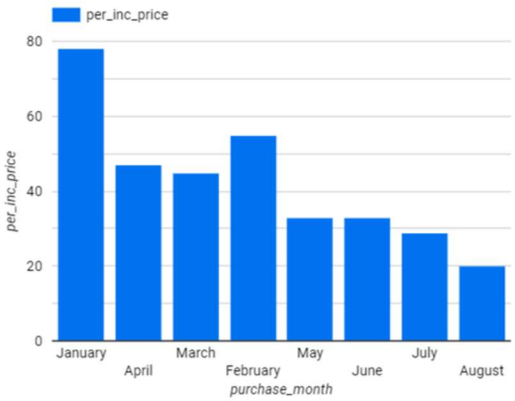
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	purchase_month	price_2017	price_2018	price_dif	per_inc_price
1	4	359927.23	996647.75	636720.52	47.0
2	7	498031.48	895507.22	397475.74	29.0
3	6	433038.6	865124.31	432085.710...	33.0
4	2	247303.02	844178.71	596875.69	55.0
5	1	120312.87	950030.36	829717.49	78.0
6	8	573971.68	854686.33	280714.649...	20.0
7	3	374344.3	983213.44	608869.139...	45.0
8	5	506071.14	996517.68	490446.540...	33.0

	purchase_month	per_inc_price
1.	January	78
2.	February	55
3.	April	47
4.	March	45
5.	June	33
6.	May	33
7.	July	29
8.	August	20

1 - 8 / 8



Observation:

Percentage increase in cost of order is high in month of January around 78% and second highest in February month around 55% for year 2018 when compared with year 2017

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

```sql
with price_avg as
(select
cust.customer_state,
round(sum(items.price + items.freight_value)) as total_price_freight,
round(sum(items.price + items.freight_value)/count(ord.order_id)) as mean_price_freight,
count(ord.order_id) as count1
from `ecommerce.orders` as ord
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
left join `ecommerce.order_items` as items
on items.order_id = ord.order_id
group by cust.customer_state )

select customer_state, total_price_freight,  mean_price_freight  from price_avg
order by total_price_freight desc
```

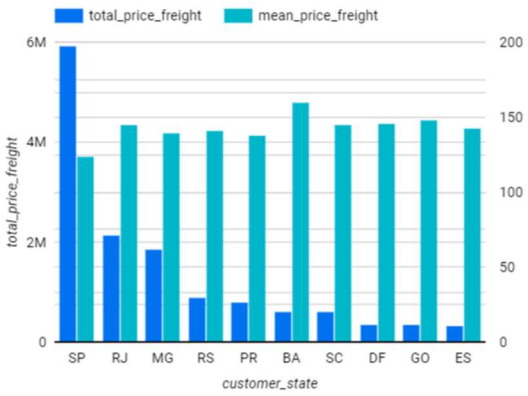
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_price_f...	mean_price...
1	SP	5921678.0	124.0
2	RJ	2129682.0	145.0
3	MG	1856161.0	140.0
4	RS	885827.0	141.0
5	PR	800935.0	138.0
6	BA	611507.0	160.0
7	SC	610214.0	145.0
8	DF	353229.0	146.0
9	GO	347707.0	148.0
10	ES	324802.0	143.0
11	PE	322238.0	178.0

	customer_state	mean_price_freight	total_price_freight...
1.	SP	124	5,921,678
2.	RJ	145	2,129,682
3.	MG	140	1,856,161
4.	RS	141	885,827
5.	PR	138	800,935
6.	BA	160	611,507
7.	SC	145	610,214
8.	DF	146	353,229
9.	GO	148	347,707
10.	ES	143	324,802
11.	PE	178	322,238

1 - 27 / 27



	customer_state	mean_price_freight	total_price_freight
1.	PB	233	140,988
2.	AL	216	96,229
3.	AC	214	19,670
4.	RO	203	57,558
5.	PA	201	217,647
6.	PI	199	108,132
7.	AP	198	16,263
8.	RR	194	10,065
9.	TO	194	61,354
10.	RN	192	101,895
11.	SE	187	73,032

1 - 27 / 27

Observation:

1) Sum of price and freight is more for SP customer state with value 5921678

And second highest for RJ state with value 2129682.0

2) Mean of price and freight is more for PB customer state with value 233

And second highest for AL State with value 216

==========

Q5: Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
select
count(*)
from `ecommerce.orders`
where order_delivered_customer_date is null
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	f0_
1	2965

There are around 2965 rows are null for `order_delivered_customer_date`

Hence analysing the number days for actual delivery and estimated delivery for non null values in field order_delivered_customer_date

```
select * from
(select
order_id,
datetime_diff(order_delivered_customer_date , order_purchase_timestamp, DAY) as time_to_del
ivery ,
datetime_diff(order_estimated_delivery_date , order_purchase_timestamp, DAY) as diff_estima
ted_delivery
from `ecommerce.orders`
where order_delivered_customer_date is not null
) as x
order by x.time_to_delivery desc, x.diff_estimated_delivery desc
```

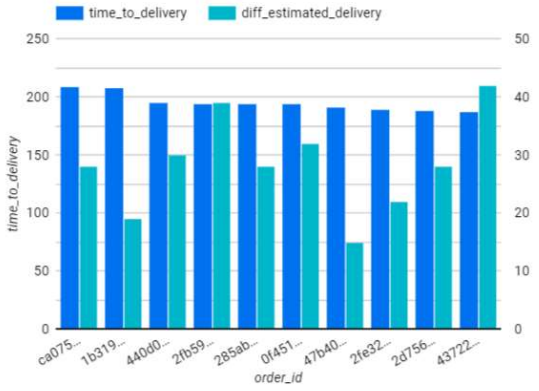
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_delivery
1	ca07593549f1816d26a572e06...	209	28
2	1b3190b2dfa9d789e1f14c05b...	208	19
3	440d0d17af552815d15a9e41a...	195	30
4	2fb597c2f772eca01b1f5c561b...	194	39
5	0f4519c5f1c541ddec9f21b3bd...	194	32
6	285ab9426d6982034523a855f...	194	28
7	47b40429ed8cce3aee9199792...	191	15
8	2fe324febf907e3ea3f2aa9650...	189	22
9	2d7561026d542c8dbd8f0daea...	188	28
10	437222e3fd1b07396f1d9ba8c...	187	42

oad more

	order_id	time_to_delivery	diff_esti...
1.	ca07593549f1816d26a572...	209	28
2.	1b3190b2dfa9d789e1f14c...	208	19
3.	440d0d17af552815d15a9e...	195	30
4.	2fb597c2f772eca01b1f5c5...	194	39
5.	285ab9426d6982034523a...	194	28
6.	0f4519c5f1c541ddec9f21b...	194	32
7.	47b40429ed8cce3aee9199...	191	15
8.	2fe324febf907e3ea3f2aa9...	189	22
9.	2d7561026d542c8dbd8f0d...	188	28
10.	437222e3fd1b07396f1d9b...	187	42
11.	c27815f7e3dd0b926b5855...	187	25

1 - 50 / 96476



2. Create columns:

- time_to_delivery = order_purchase_timestamp - order_delivered_customer_date
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
select * from
(select
order_id,
datetime_diff(order_delivered_customer_date , order_purchase_timestamp, DAY) as time_to_del
ivery ,
datetime_diff(order_estimated_delivery_date , order_purchase_timestamp, DAY) as diff_estima
ted_delivery
from `ecommerce.orders`
where order_delivered_customer_date is not null
) as x
order by x.time_to_delivery desc, x.diff_estimated_delivery desc
```

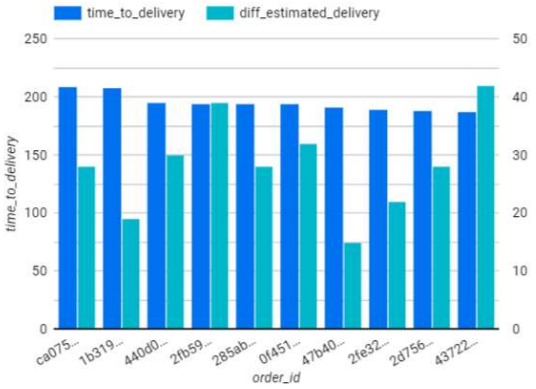
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_delivery
1	ca07593549f1816d26a572e06...	209	28
2	1b3190b2dfa9d789e1f14c05b...	208	19
3	440d0d17af552815d15a9e41a...	195	30
4	2fb597c2f772eca01b1f5c561b...	194	39
5	0f4519c5f1c541ddec9f21b3bd...	194	32
6	285ab9426d6982034523a855f...	194	28
7	47b40429ed8cce3aee9199792...	191	15
8	2fe324febf907e3ea3f2aa9650...	189	22
9	2d7561026d542c8dbd8f0daea...	188	28
10	437222e3fd1b07396f1d9ba8c...	187	42

oad more

	order_id	time_to_delivery	diff_esti...
1.	ca07593549f1816d26a572...	209	28
2.	1b3190b2dfa9d789e1f14c...	208	19
3.	440d0d17af552815d15a9e...	195	30
4.	2fb597c2f772eca01b1f5c5...	194	39
5.	285ab9426d6982034523a...	194	28
6.	0f4519c5f1c541ddec9f21b...	194	32
7.	47b40429ed8cce3aee9199...	191	15
8.	2fe324febf907e3ea3f2aa9...	189	22
9.	2d7561026d542c8dbd8f0d...	188	28
10.	437222e3fd1b07396f1d9b...	187	42
11.	c27815f7e3dd0b926b5855...	187	25

1 - 50 / 96476



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

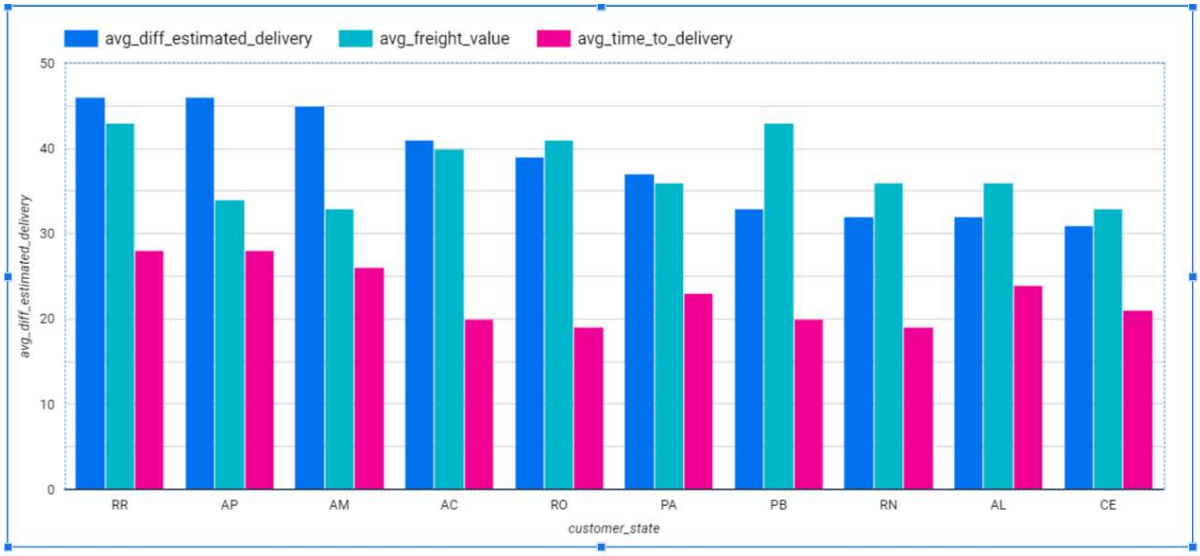
```
select
x.customer_state,
round(avg(x.freight_value)) as avg_freight_value,
round(avg(x.time_to_delivery)) as avg_time_to_delivery,
round(avg(x.diff_estimated_delivery)) as avg_diff_estimated_delivery
from
(select
ord.order_id,
items.freight_value,
cust.customer_state,
datetime_diff(ord.order_delivered_customer_date , ord.order_purchase_timestamp, DAY) as tim
e_to_delivery ,
datetime_diff(ord.order_estimated_delivery_date , ord.order_purchase_timestamp, DAY) as dif
f_estimated_delivery
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
where ord.order_delivered_customer_date is not null
) as x
group by x.customer_state
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	RJ	21.0	15.0	26.0
2	MG	21.0	12.0	24.0
3	SC	22.0	15.0	26.0
4	SP	15.0	8.0	19.0
5	GO	23.0	15.0	27.0
6	RS	22.0	15.0	28.0
7	BA	26.0	19.0	29.0
8	MT	28.0	18.0	31.0
9	SE	37.0	21.0	30.0
10	PE	33.0	18.0	31.0
11	TO	37.0	17.0	29.0
12	CE	33.0	21.0	31.0
13	PR	20.0	11.0	24.0

Results r



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

Top 5 highest average freight

```
with ord_details as (
select
x.customer_state,
round(avg(x.freight_value)) as avg_freight_value,
round(avg(x.time_to_delivery)) as avg_time_to_delivery,
round(avg(x.diff_estimated_delivery)) as avg_diff_estimated_delivery
from
(select
ord.order_id,
items.freight_value,
cust.customer_state,
```

```
datetime_diff(ord.order_delivered_customer_date , ord.order_purchase_timestamp, DAY) as tim
e_to_delivery ,
datetime_diff(ord.order_estimated_delivery_date , ord.order_purchase_timestamp, DAY) as dif
f_estimated_delivery
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
where ord.order_delivered_customer_date is not null
) as x
group by x.customer_state
)

select customer_state, avg_freight_value from ord_details
order by avg_freight_value desc limit 5
```

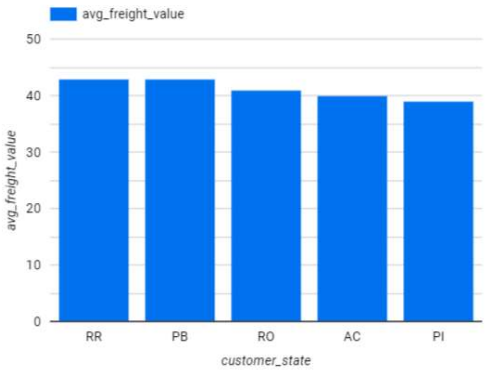
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...
1	RR	43.0
2	PB	43.0
3	RO	41.0
4	AC	40.0
5	PI	39.0

	customer_state	avg_freight_value
1.	RR	43
2.	RO	41
3.	PI	39
4.	PB	43
5.	AC	40

1 - 5 / 5



Observation: State RR has the highest average freight value followed by PB, RO, AC and PI.

Top 5 lowest average freight

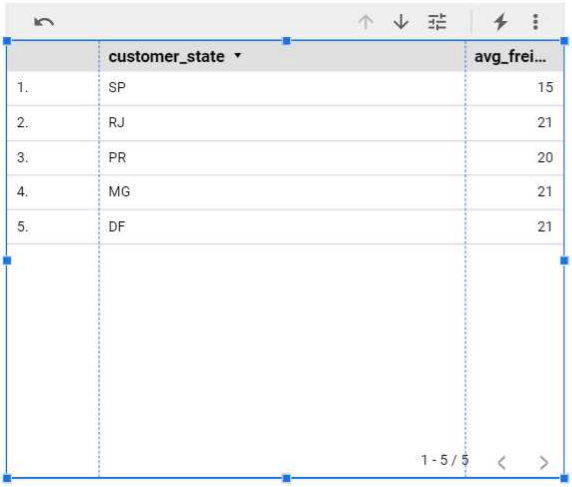
```
with ord_details as (
select
x.customer_state,
round(avg(x.freight_value)) as avg_freight_value,
round(avg(x.time_to_delivery)) as avg_time_to_delivery,
round(avg(x.diff_estimated_delivery)) as avg_diff_estimated_delivery
from
```

```sql
(select
ord.order_id,
items.freight_value,
cust.customer_state,
datetime_diff(ord.order_delivered_customer_date , ord.order_purchase_timestamp, DAY) as tim
e_to_delivery ,
datetime_diff(ord.order_estimated_delivery_date , ord.order_purchase_timestamp, DAY) as dif
f_estimated_delivery
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
where ord.order_delivered_customer_date is not null
) as x
group by x.customer_state
)
select customer_state, avg_freight_value from ord_details
order by avg_freight_value asc limit 5
```

Query results

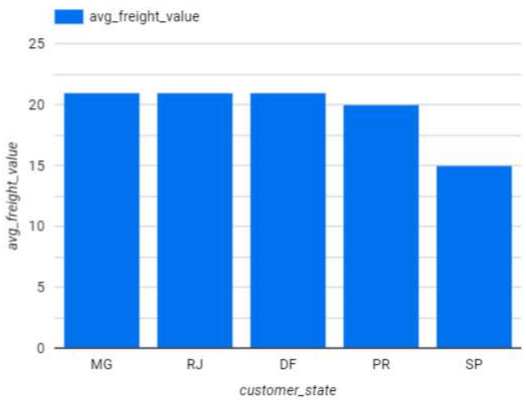
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_...
1	SP	15.0
2	PR	20.0
3	MG	21.0
4	DF	21.0
5	RJ	21.0



	customer_state	avg_frei...
1.	SP	15
2.	RJ	21
3.	PR	20
4.	MG	21
5.	DF	21

1 - 5 / 5



Observation: State SP has the lowest average freight value followed by PR, RJ, MG and DF.

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

Top 5 states with highest average time to delivery

```
with ord_details as (
select
x.customer_state,
round(avg(x.freight_value)) as avg_freight_value,
round(avg(x.time_to_delivery)) as avg_time_to_delivery,
round(avg(x.diff_estimated_delivery)) as avg_diff_estimated_delivery
from
(select
ord.order_id,
items.freight_value,
cust.customer_state,
datetime_diff(ord.order_delivered_customer_date , ord.order_purchase_timestamp, DAY) as tim
e_to_delivery ,
datetime_diff(ord.order_estimated_delivery_date , ord.order_purchase_timestamp, DAY) as dif
f_estimated_delivery
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
where ord.order_delivered_customer_date is not null
) as x
group by x.customer_state
)
select customer_state, avg_time_to_delivery from ord_details
order by avg_time_to_delivery desc limit 5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to...
1	RR	28.0
2	AP	28.0
3	AM	26.0
4	AL	24.0
5	PA	23.0

	customer_state	avg_time_to_delivery
1.	RR	28
2.	PA	23
3.	AP	28
4.	AM	26
5.	AL	24

1 - 5 / 5

Observation: State RR has the highest average time to delivery followed by PA, AP, AM and AL.

Top 5 states with lowest average time to delivery

```
with ord_details as (
select
x.customer_state,
round(avg(x.freight_value)) as avg_freight_value,
round(avg(x.time_to_delivery)) as avg_time_to_delivery,
round(avg(x.diff_estimated_delivery)) as avg_diff_estimated_delivery
from
(select
ord.order_id,
items.freight_value,
cust.customer_state,
datetime_diff(ord.order_delivered_customer_date , ord.order_purchase_timestamp, DAY) as tim
e_to_delivery ,
datetime_diff(ord.order_estimated_delivery_date , ord.order_purchase_timestamp, DAY) as dif
f_estimated_delivery
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
where ord.order_delivered_customer_date is not null
) as x
group by x.customer_state
)
select customer_state, avg_time_to_delivery from ord_details
order by avg_time_to_delivery asc limit 5
```

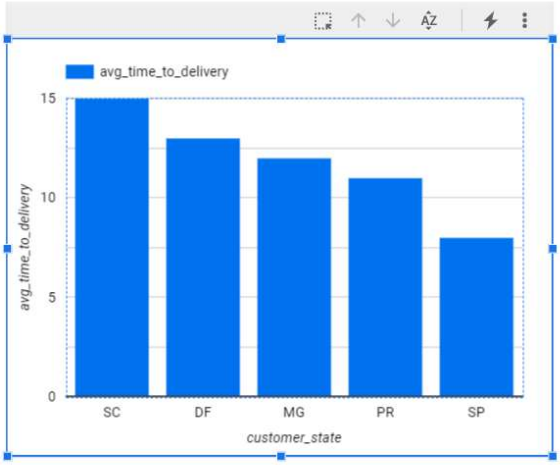
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_time_to...
1	SP	8.0
2	PR	11.0
3	MG	12.0
4	DF	13.0
5	SC	15.0

	customer_state	avg_tim...
1.	SP	8
2.	SC	15
3.	PR	11
4.	MG	12
5.	DF	13

1 - 5 / 5



Observation: State SP has the lowest average time to delivery followed by PR, MG, DF and SC.

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

Top 5 states where delivery is fast compared to estimated date

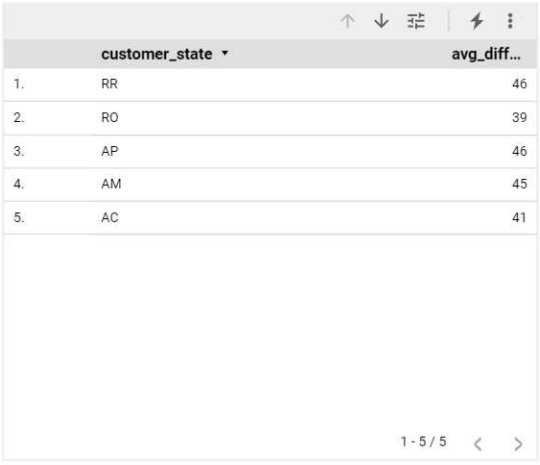
```
with ord_details as (
select
x.customer_state,
round(avg(x.freight_value)) as avg_freight_value,
round(avg(x.time_to_delivery)) as avg_time_to_delivery,
round(avg(x.diff_estimated_delivery)) as avg_diff_estimated_delivery
from
(select
ord.order_id,
items.freight_value,
```

```
cust.customer_state,
datetime_diff(ord.order_delivered_customer_date , ord.order_purchase_timestamp, DAY) as tim
e_to_delivery ,
datetime_diff(ord.order_estimated_delivery_date , ord.order_purchase_timestamp, DAY) as dif
f_estimated_delivery
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
where ord.order_delivered_customer_date is not null
) as x
group by x.customer_state
)
select customer_state, avg_diff_estimated_delivery  from ord_details
order by avg_diff_estimated_delivery desc limit 5
```

Query results

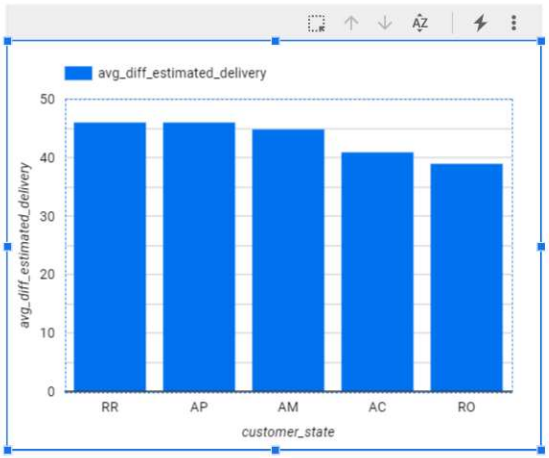
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_diff_estimated_delivery
1	RR	46.0
2	AP	46.0
3	AM	45.0
4	AC	41.0
5	RO	39.0



	customer_state	avg_diff...
1.	RR	46
2.	RO	39
3.	AP	46
4.	AM	45
5.	AC	41

1 - 5 / 5



Observation: State RR has the fastest delivery when compared with estimated delivery which is followed by AP, AM, AC and RO

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

```sql
with ord_details as (
select
x.customer_state,
round(avg(x.freight_value)) as avg_freight_value,
round(avg(x.time_to_delivery)) as avg_time_to_delivery,
round(avg(x.diff_estimated_delivery)) as avg_diff_estimated_delivery
from
(select
ord.order_id,
items.freight_value,
cust.customer_state,
datetime_diff(ord.order_delivered_customer_date , ord.order_purchase_timestamp, DAY) as tim
e_to_delivery ,
datetime_diff(ord.order_estimated_delivery_date , ord.order_purchase_timestamp, DAY) as dif
f_estimated_delivery
from `ecommerce.orders` as ord
left join `ecommerce.order_items` as items
on ord.order_id = items.order_id
left join `ecommerce.customers` as cust
on ord.customer_id = cust.customer_id
where ord.order_delivered_customer_date is not null
) as x
group by x.customer_state
)
select customer_state, avg_diff_estimated_delivery  from ord_details
order by avg_diff_estimated_delivery asc limit 5
```

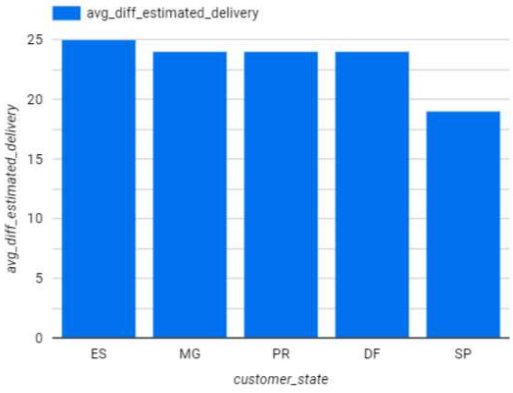
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_diff_est...
1	SP	19.0
2	DF	24.0
3	MG	24.0
4	PR	24.0
5	ES	25.0

	customer_state	avg_diff_estimated_delivery
1.	SP	19
2.	PR	24
3.	MG	24
4.	ES	25
5.	DF	24

1 - 5 / 5



Observation: State SP has slowest delivery when compared with estimated delivery which is followed by DF, MG, PR and ES

==========

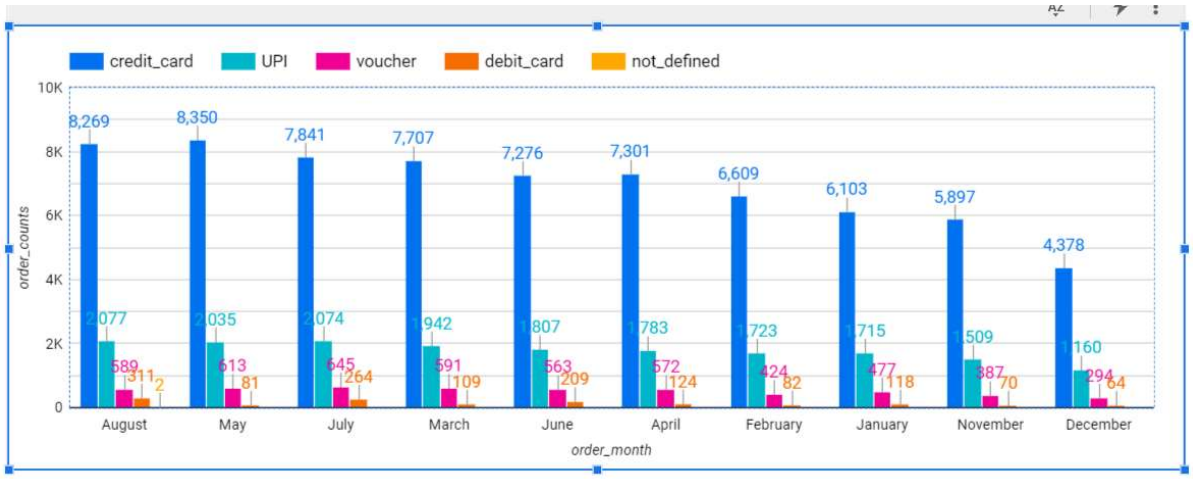
Q6: Payment type analysis:

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

```
select * from
(select
pay.payment_type,
count(ord.order_id) as order_counts,
extract(month from ord.order_purchase_timestamp) as order_month
from
`ecommerce.orders`as ord
inner join
`ecommerce.payments`as pay
on ord.order_id = pay.order_id
group by pay.payment_type, extract(month from ord.order_purchase_timestamp)) as x
order by x.order_month
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	order_counts	order_month
1	credit_card	6103	1
2	UPI	1715	1
3	voucher	477	1
4	debit_card	118	1
5	UPI	1723	2
6	credit_card	6609	2
7	voucher	424	2
8	debit_card	82	2
9	credit_card	7707	3
10	UPI	1942	3
11	debit_card	109	3
12	voucher	591	3
13	voucher	572	4



Observation:

Credit card payment is more for all the months and it is highest for August month.

UPI payment is second in all the months and it is highest for August month.

Voucher payment is highest for May month.

Debit card payment is highest for August month.

2. Distribution of payment instalments and count of orders

```
select * from (select
pay.payment_installments,
```

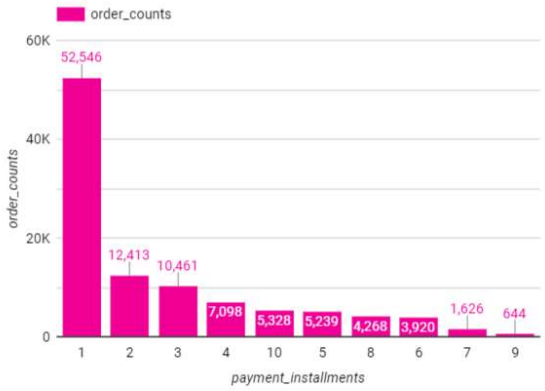
```
count(ord.order_id) as order_counts
from
`ecommerce.orders`as ord
inner join
`ecommerce.payments`as pay
on ord.order_id = pay.order_id
group by pay.payment_installments) as x
order by x.order_counts desc
```

JOB INFORMATION | RESULTS | JSON | EXECUTION D

Row	payment_installments	order_counts
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644
11	12	133
12	15	74
13	18	27

	payment_installments	order_counts
1.	1	52,546
2.	2	12,413
3.	3	10,461
4.	4	7,098
5.	10	5,328
6.	5	5,239
7.	8	4,268
8.	6	3,920
9.	7	1,626
10.	9	644
11.	12	133
12.	15	74
13.	18	27
14.	11	23
15.	24	18
16.	20	17
17.	13	16
18.	14	15

1 - 24 / 24



Observation:

Count of orders is more for single payment around 52546

And count of orders gradually decreases if the number of instalments increases.

==========

Insights / Observations :

1) 99441 rows are available in customers table.
2) There are around 4119 distinct customer cities in customer table.
3) There are around 27 distinct customer state in customer table.
4) 1000163 rows are available in geolocation table.
5) There are around 8011 distinct geolocation cities in geolocation table.
6) There are around 27 distinct geolocation states in geolocation table.
7) 112650 rows are available in order_items table.
8) Shipping_limit_date range is between 19-09-2016 to 09-04-2020 in order_items table.
9) Range for review and answer is between: 0001-01-18 to 0031-12-17 in reviews table
   But the year in the date is not correct hence we cannot consider it for range.
10) 99441 rows are available in orders table.
11) Overall date range for order purchase and delivery lies in range 2016-09-04 to 2018-11-12 in orders table.
12) 103886 rows are available in payments table.
13) 32951 rows are available in products table.
14) 3095 rows are available in sellers table.
15) There are around 611 distinct seller cities in sellers table.
16) There are around 23 distinct seller states in sellers table.
17) August, May and July month has more orders (greater than 10300 orders) in Brazil city.
18) Afternoon time has more orders (around 32211 orders) in Brazil city.
19) Sao Paulo city in Brazil has highest orders of 15540
    Rio de janerio city in Brazil has second highest order of 6882
    Belo horizonte city in Brazil has third highest order of 2773.

20) Sao Paulo and Rio de Janeiro city in Brazil has highest orders in the month of August
Belo horizonte city in Brazil has highest orders in the month of March and followed by August
Based on this it looks like August month has more orders.
21) Percentage increase in cost of order is high in month of January around 78% and second highest in February month around 55% for year 2018 when compared with year 2017.
22) Sum of price and freight is more for SP customer state with value 5921678
And second highest for RJ state with value 2129682.0.
23) Mean of price and freight is more for PB customer state with value 233
And second highest for AL State with value 216.
24) State RR has the highest average freight value followed by PB, RO, AC and PI.
25) State SP has the lowest average freight value followed by PR, RJ, MG and DF.
26) State RR has the highest average time to delivery followed by PA, AP, AM and AL.
27) State SP has the lowest average time to delivery followed by PR, MG, DF and SC.
28) State RR has the fastest delivery when compared with estimated delivery which is followed by AP, AM, AC and RO.
29) State SP has slowest delivery when compared with estimated delivery which is followed by DF, MG, PR and ES.
30) Credit card payment is more for all the months and it is highest for August month.
UPI payment is second in all the months and it is highest for August month.
Voucher payment is highest for May month.
Debit card payment is highest for August month.
31) Count of orders is more for single payment around 52546
And count of orders gradually decreases if the number of instalments increases.

Recommendations:

1) August, May and July month has more orders (greater than 10300 orders) in Brazil city. Target company can provide attractive promotions for people for months other than August, May and July so that people in Brazil start buying products over other months also. So that the order purchase is uniform across the year.

2) Afternoon time has more orders (around 32211 orders) in Brazil city. Target company can Offer discounted price during other timings so that the orders will be placed during other timings also which will increase the business.

3) Sao Paulo city in Brazil has highest orders of 15540. Rio de janerio city in Brazil has second highest order of 6882. Belo horizonte city in Brazil has third highest order of 2773. Target company can provide attractive promotions and offer discounted price to attract the customers from other cities of Brazil where the orders is less, which will increase the number of orders, number of new customers and in turn it increases the business.

4) Target company can increase the resources in Sao Paulo, Rio de janero, and Belo horizonto cities of Brazil which has the highest orders especially during the festive season.

5) Percentage increase in cost of order is high in month of January around 78% and second highest in February month around 55% for year 2018 when compared with year 2017. Target company can give promotions and offers on other months for bulk orders which will increase the business.

6) State RR has the highest average freight value followed by PB, RO, AC and PI. Hence Target company can give offer on regions with highest freight value to increase the business.

7) State SP has slowest delivery when compared with estimated delivery which is followed by DF, MG, PR and ES. So target company can increase their presence in slowest delivery regions to reduce the delivery timings.

8) Target company can give products without EMI so that customers can buy more products and pay later which will increase the business.