1. Question 1:

1. Data type of columns in a table

Normally we use Describe function to know the datatype and column information for a table

"DESCRIBE TABLE_NAME"

But as I am using GCP Big query: - one simple way is to just click on table

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

2. Time period for which the data is given

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

```
1  Select min(order_purchase_timestamp) as start_date, max(order_purchase_timestamp) as end_date
2  from `Target.orders`
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS | EXPLORE DATA

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

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

Just checking the date:

```
SELECT min(date(order_purchase_timestamp)) as start_date,max(date(order_purchase_timestamp)) as end_date
FROM `Target.orders`
```

Press Alt+F1 for Accessibilit

Query results

SAVE RESULTS EXPLORE DATA

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

Row	start_date	end_date
1	2016-09-04	2018-10-17

3. Cities and States covered in the dataset

```
SELECT distinct customer_city,customer_state
FROM `Target.customers`;
```

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXE

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP

```
10
11 SELECT distinct count(customer_city) as cityCount,count(customer_state) as stateCount
12 FROM `Target.customers`;
13
```

Query results

SAVE RESULTS

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

Row	cityCount	stateCount
1	99441	99441

2. Question 2:

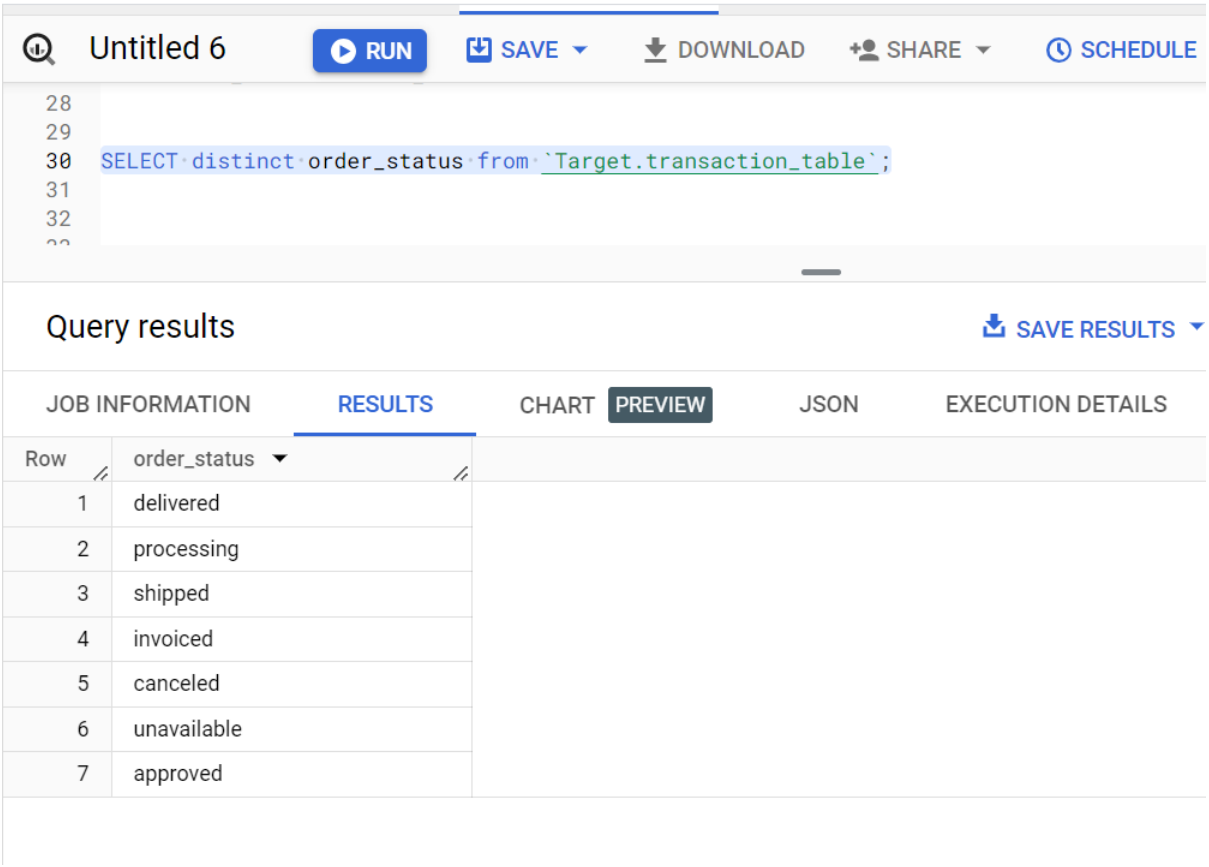
1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months

Creation of View for the overall Transaction table by join orders table with order items Table

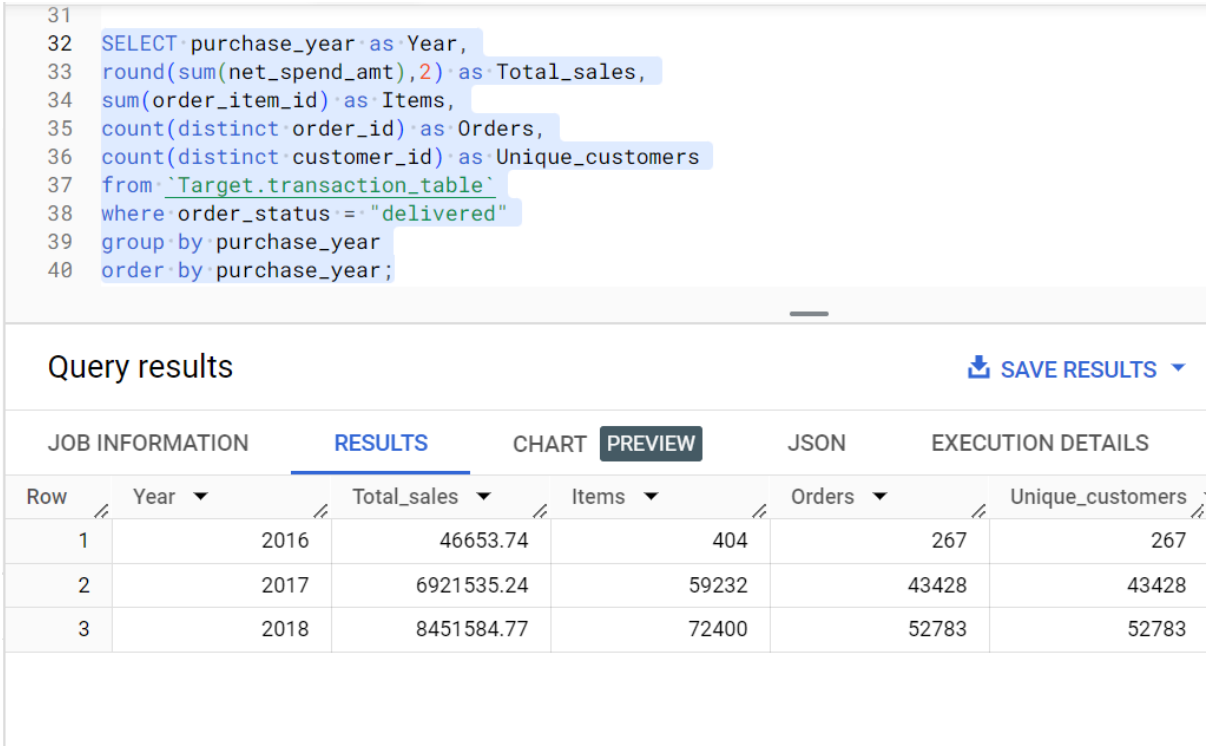
```
14
15 CREATE VIEW Target.transaction_table as
16 SELECT oi.*,
17 (oi.price + oi.freight_value) as net_spend_amt,
18 o.customer_id,
19 o.order_status,
20 o.order_purchase_timestamp,
21 o.order_delivered_customer_date,
22 o.order_estimated_delivery_date,
23 EXTRACT(month FROM order_purchase_timestamp) as purchase_month,
24 EXTRACT(year FROM order_purchase_timestamp) as purchase_year
25 from `Target.order_items` as oi
26 join `Target.orders` as o
27 on o.order_id = oi.order_id
28
```

- **Let's first explore Different type of Order status**

```
SELECT distinct order_status from `Target.transaction_table`;
```



❖ **Now checking the YOY Sales**



There is increase of sales each Year (Year on Year Increase) and also, there is increase in customer – we can ignore 2016 as we have the data from month of September,2016 Let's check the % increase in sales and % increase in customer for 2018.

```sql
with YOY_details as (
SELECT purchase_year as Year,
round(sum(net_spend_amt),2) as Total_sales,
sum(order_item_id) as Items,
count(distinct order_id) as Orders,
count(distinct customer_id) as Unique_customers
FROM `Target.transaction_table`
WHERE order_status = "delivered"
GROUP BY purchase_year
ORDER BY purchase_year
),

YOY_cal as (
SELECT *,Lag(Total_sales) OVER (ORDER BY Year) as Last_year_sales,
Lag(Unique_customers) OVER (ORDER BY Year) as Last_Unique_customers
FROM YOY_details
ORDER BY Year)
SELECT * from YOY_cal;

with YOY_details as (
SELECT purchase_year as Year,
round(sum(net_spend_amt),2) as Total_sales,
sum(order_item_id) as Items,
count(distinct order_id) as Orders,
count(distinct customer_id) as Unique_customers
FROM `Target.transaction_table`
WHERE order_status = "delivered"
GROUP BY purchase_year
ORDER BY purchase_year
),

YOY_cal as (
SELECT *,Lag(Total_sales) OVER (ORDER BY Year) as Last_year_sales,
Lag(Unique_customers) OVER (ORDER BY Year) as Last_Unique_customers
FROM YOY_details
ORDER BY Year)

SELECT *,
round((((Total_sales - Last_year_sales)*100)/Last_year_sales),0) as sales_pct_diff,
round((((Unique_customers - Last_Unique_customers)*100)/Last_Unique_customers),0) as customer_pct_diff
FROM YOY_cal
WHERE Year = 2018;
```

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Year	Total_sales	Items	Orders	Unique_customers	Last_year_sales	Last_Unique_cu
1	2018	8451584.77	72400	52783	52783	6921535.24	434

% Increase in sales from 2018 to 2017= 22%

% Increase in customers from 2018 to 2017 = 22%

❖ **Month on Month Trend Analysis: -**

➢ Using the View created in previous steps:

```
SELECT purchase_year as Year,
purchase_month as Month,
round(sum(net_spend_amt),2) as Total_sales,
sum(order_item_id) as Items,
count(distinct order_id) as Orders,
count(distinct customer_id) as Unique_customers
FROM `Target.transaction_table`
WHERE order_status = "delivered"
GROUP BY purchase_year,purchase_month
ORDER BY purchase_year ,purchase_month
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Year	Month	Total_sales	Items	Orders	Unique_customers
1	2016	9	143.46	6	1	1
2	2016	10	46490.66	397	265	265
3	2016	12	19.62	1	1	1
4	2017	1	127482.37	1273	750	750

Create the CTE grouping by Year and Month using the view which we created previously and check the top 5 month where the sales are high

```
with YOY_details as (
SELECT purchase_year as Year,
purchase_month as Month,
round(sum(net_spend_amt),2) as Total_sales,
sum(order_item_id) as Items,
count(distinct order_id) as Orders,
count(distinct customer_id) as Unique_customers
FROM `Target.transaction_table`
WHERE order_status = "delivered"
GROUP BY purchase_year,purchase_month
ORDER BY purchase_year ,purchase_month
),
YOY_cal as (
SELECT *,
DENSE_RANK() over (partition by Year order by Total_sales desc) as Ranker
FROM YOY_details
ORDER BY Year,Total_sales desc)
SELECT * FROM YOY_cal
WHERE ranker < 6;
```

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Year	Month	Total_sales	Items	Orders	Unique_customers	Ranker
1	2016	10	46490.66	397	265	265	
2	2016	9	143.46	6	1	1	
3	2016	12	19.62	1	1	1	
4	2017	11	1153364.2	10384	7289	7289	
5	2017	12	843078.29	7214	5513	5513	
6	2017	10	751117.01	6474	4478	4478	
7	2017	9	701077.49	5647	4150	4150	
8	2017	8	645832.36	5810	4193	4193	

Here we have rank the Month with maximum sales:

- **We observe that Maximum sales for the year 2017 is from the Month of November, December and than October Mostly in the Month of Christmas, Halloween and Oktoberfest**

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

Row	Year	Month	Total_sales	Items	Orders	Unique_customers
1	2017	11	1153364.2	10384	7289	7289
2	2017	12	843078.29	7214	5513	5513
3	2017	10	751117.01	6474	4478	4478
4	2017	9	701077.49	5647	4150	4150
5	2017	8	645832.36	5810	4193	4193
6	2017	5	566851.4	4637	3546	3546
7	2017	7	566299.08	5360	3872	3872
8	2017	6	490050.37	3975	3135	3135
9	2017	3	414330.95	3402	2546	2546

Load more

- **The data for 2018 is incomplete, it is just before August where the order status is “Delivered”. Here is the view for the same, With April and May month having the highest sales for 2018**

Row	Year	Month	Total_sales	Items	Orders	Unique_customers
1	2018	4	1132878.93	9563	6798	6798
2	2018	5	1128774.52	9469	6749	6749
3	2018	3	1120598.24	9621	7003	7003
4	2018	1	1077887.46	9549	7069	7069
5	2018	7	1027807.28	8229	6159	6159
6	2018	6	1011978.29	8420	6099	6099
7	2018	8	985491.64	8333	6351	6351
8	2018	2	966168.41	9216	6555	6555

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

For this we use the following Table to divide it into 5 Part

PART	TIME
Dawn	2:00:00 to 5:59:59
Morning	6:00:00 to 11:59:59
Afternoon	12:00:00 to 15:59:59
Evening	16:00:00 to 20:59:59
Night	21:01:00 to 1:59:59

```
with time_status_view as (
Select *,Time(order_purchase_timestamp) as TimePurchase,
Case
When Time(order_purchase_timestamp) between (TIME "2:00:00") and (TIME "5:59:59") then "Dawn"
When Time(order_purchase_timestamp) between (TIME '6:00:00') and (TIME '11:59:59') then "Morning"
When Time(order_purchase_timestamp) between (TIME '12:00:00') and (TIME '15:59:59') then "Afternoon"
When Time(order_purchase_timestamp) between (TIME '16:00:00') and (TIME '20:59:59') then "Evening"
When (Time(order_purchase_timestamp) between (TIME '21:00:00') and (TIME '23:59:59')) or
(Time(order_purchase_timestamp) between (TIME '00:00:00') and (TIME '1:59:59')) then "Night"
end as Timestatus
from `Target.transaction_table'
)
Select * from (Select Distinct Timestatus,Count(customer_id) over (partition by Timestatus) as CustomerCount
from time_status_view ) base
order by CustomerCount Desc;
```

Most of the customers are active in Evening and Afternoon.

Afternoon	12:00:00 to 15:59:59
Evening	16:00:00 to 20:59:59

If we just Categories them into 4 parts I.e. Dawn, Morning, Afternoon or Night

```
with time_status_view as (
Select *,Time(order_purchase_timestamp) as TimePurchase,
Case
When Time(order_purchase_timestamp) between (TIME "2:00:00") and (TIME "5:59:59") then "Dawn"
When Time(order_purchase_timestamp) between (TIME '6:00:00') and (TIME '11:59:59') then "Morning"
When Time(order_purchase_timestamp) between (TIME '12:00:00') and (TIME '15:59:59') then "Afternoon"
When (Time(order_purchase_timestamp) between (TIME '16:00:00') and (TIME '23:59:59')) or
(Time(order_purchase_timestamp) between (TIME '00:00:00') and (TIME '1:59:59')) then "Night"
end as Timestatus
from `Target.transaction_table`
)
Select * from (Select Distinct Timestatus,Count(customer_id) over (partition by Timestatus) as CustomerCount
from time_status_view ) base
order by CustomerCount Desc;
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	Timestatus	CustomerCount
1	Night	56908
2	Afternoon	29157
3	Morning	25225
4	Dawn	1360

So, in Night and Afternoon most of the customers are active.

Adding Rank function for the same :

```
with time_status_view as (
Select *,Time(order_purchase_timestamp) as TimePurchase,
Case
When Time(order_purchase_timestamp) between (TIME "2:00:00") and (TIME "5:59:59") then "Dawn"
When Time(order_purchase_timestamp) between (TIME '6:00:00') and (TIME '11:59:59') then "Morning"
When Time(order_purchase_timestamp) between (TIME '12:00:00') and (TIME '15:59:59') then "Afternoon"
When (Time(order_purchase_timestamp) between (TIME '16:00:00') and (TIME '23:59:59')) or
(Time(order_purchase_timestamp) between (TIME '00:00:00') and (TIME '1:59:59')) then "Night"
end as Timestatus
from `Target.transaction_table`
),
Timestatus_data as (
Select * from (Select Distinct Timestatus,Count(customer_id) over (partition by Timestatus) as CustomerCount
from time_status_view ) base
order by CustomerCount Desc),
FinalTimedata as (
Select *,
Rank() over (order by CustomerCount desc) as Ranker
from Timestatus_data
)
Select * from FinalTimedata
where Ranker = 1;
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	Timestatus	CustomerCount	Ranker
1	Night	56908	1

3. Question 3:

1. Get month on month orders by region, states

To get this we need orders details and the customer details for those customer We will be using Orders and Customers Table As Month-on-Month details is needed

We need to do group by based on Year, Month, State and City Query without checking order status and with order by Orders purchase year and month

```sql
with customer_orders as (
Select o.*,
EXTRACT(month FROM order_purchase_timestamp) as purchase_month,
EXTRACT(year FROM order_purchase_timestamp) as purchase_year,
c.customer_city,
c.customer_state,
c.customer_zip_code_prefix
from `Target.customers` as c
join `Target.orders` as o
on c.customer_id = o.customer_id
)
SELECT Distinct purchase_year,purchase_month,customer_state,customer_city,
count(DISTINCT order_id) as OrderCounts
from customer_orders
group by purchase_year,purchase_month,customer_state,customer_city
order by purchase_year,purchase_month,customer_state,customer_city;
```

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	purchase_year	purchase_month	customer_state	customer_city	OrderCounts
1	2016	9	RR	boa vista	1
2	2016	9	RS	passo fundo	1
3	2016	9	SP	sao joaquim da barra	1
4	2016	9	SP	sao jose dos campos	1
5	2016	10	AL	cacimbinhas	1
6	2016	10	AL	maceio	1
7	2016	10	BA	apuarema	1
8	2016	10	BA	eunapolis	1

Load more

If we just consider Order status as Delivered: -

```
224
225
226 with customer_orders as (
227 Select o.*,
228 EXTRACT(month FROM order_purchase_timestamp) as purchase_month,
229 EXTRACT(year FROM order_purchase_timestamp) as purchase_year,
230 c.customer_city,
231 c.customer_state,
232 c.customer_zip_code_prefix
233 from `Target.customers` as c
234 join `Target.orders` as o
235 on c.customer_id = o.customer_id
236 )
237 SELECT Distinct purchase_year,purchase_month,customer_state,customer_city,
238 count(DISTINCT order_id) as OrderCounts
239 from customer_orders
240 where order_status = "delivered"
241 group by purchase_year,purchase_month,customer_state,customer_city
242 order by purchase_year,purchase_month,customer_state,customer_city
243
244
```

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	purchase_year	purchase_month	customer_state	customer_city	OrderCounts
1	2016	9	SP	sao joaquim da barra	1
2	2016	10	AL	maceio	1
3	2016	10	BA	apuarema	1
4	2016	10	BA	eunapolis	1
5	2016	10	BA	iacu	1
6	2016	10	CE	bela cruz	1
7	2016	10	CE	fortaleza	4
8	2016	10	CE	sobral	1
9	2016	10	DF	brasilia	6

Most of the Sales is coming for SP – Sao Paulo city and in the Month of August in 2018 These Order counts are based on Orders with Delivered Status only.

2. How are customers distributed in Brazil

To know the Distribution of customer we just need to check the customers tables Query for the same is: -

```
245
246 Select customer_city,customer_state,count(DISTINCT customer_id) as UUID,count(DISTINCT customer_unique_id) as
247 superset_id
248 From `Target.customers`
249 group by 1,2
250 order by superset_id desc,UUID desc ;
251
252
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_city	customer_state	UUID	superset_id
1	sao paulo	SP	15540	14984
2	rio de janeiro	RJ	6882	6620
3	belo horizonte	MG	2773	2672
4	brasilia	DF	2131	2069
5	curitiba	PR	1521	1465

Please note - Here are giving alias name in order by clause but again it is DB dependent- In case of big query it is accepting so based on the DB we can change the syntax either use CTE and then use it or use it as a sub-query

4. Question 4:

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

```
252
253
254 with YearMonthCost as (
255 Select Distinct purchase_year as Year,purchase_month as Month,
256 round(sum(net_spend_amt) over (partition by purchase_year,purchase_month),2) as Cost,
257 from `Target.transaction_table`
258 where (purchase_month between 1 and 8) and order_status = "delivered"
259 order by purchase_year,purchase_month),
260
261 Year2018 as (
262 Select Month,Cost
263 from YearMonthCost
264 where Year = 2018),
265 Year2017 as (
266 Select Month,Cost
267 from YearMonthCost
268 where Year = 2017)
269
270 Select Year2018.Month, Year2017.Cost as Cost2017, Year2018.Cost as Cost2018,
271 Round((((Year2018.Cost - Year2017.Cost)/Year2017.Cost)*100),0) as PercentageChange
272 from Year2018
273 join Year2017
274 on Year2018.Month = Year2017.Month
275 order by Month;
276
```

Query results

SAVE RESULTS

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

Row	Month	Cost2017	Cost2018	PercentageChange
1	1	127482.37	1077887.46	746.0
2	2	271239.32	966168.41	256.0
3	3	414330.95	1120598.24	170.0
4	4	390812.4	1132878.93	190.0
5	5	566851.4	1128774.52	99.0
6	6	490050.37	1011978.29	107.0
7	7	566299.08	1027807.28	81.0
8	8	645832.36	985491.64	53.0

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

We will be using the View which we created a- Also first we sum up the price and Freight based on each order_id as right now it is based on Order_items so as to get avegrage based on each orders . The query for the whole is as follows:

```
281
282 with order_price as (
283 Select Distinct customer_id,order_id,Round(Sum(price) over (partition by order_id),2) as Total_Order_price,
284 Round(Sum(freight_value) over (partition by order_id),2) as Total_Order_freight_value
285 from `Target.transaction_table`
286 )
287 Select Distinct c.customer_state,
288 Round(Avg(op.Total_Order_price) over (partition by c.customer_state),2) as Mean_price,
289 Round(Sum(op.Total_Order_price) over (partition by c.customer_state),2) as Total_price,
290 Round(Avg(op.Total_Order_freight_value) over (partition by c.customer_state),2) as Mean_freight_value,
291 Round(Sum(op.Total_Order_freight_value) over (partition by c.customer_state),2) as Total_freight_value
292 from `Target.customers` as c
293 join order_price as op
294 on c.customer_id = op.customer_id
295 order by Total_price desc, Total_freight_value desc;
296
297
298
```

Query results

SAVE RESULTS EXPLO

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

Row	customer_state	Mean_price	Total_price	Mean_freight_value	Total_freight_value
1	SP	125.75	5202955.05	17.37	718723.07
2	RJ	142.93	1824092.67	23.95	305589.31
3	MG	137.33	1585308.03	23.46	270853.46
4	RS	138.13	750304.02	24.95	135522.74
5	PR	136.67	683083.76	23.58	117851.68
6	SC	144.12	520553.34	24.82	89660.26
7	BA	152.28	511349.99	29.83	100156.68
8	DF	142.4	302603.94	23.82	50625.5
9	GO	146.78	294591.95	26.46	53114.98
10	ES	135.82	275037.31	24.58	49764.6

5. Question 5:

1. Calculate days between purchasing, delivering and estimated delivery

```
298
299
300 Select Distinct order_id,order_delivered_customer_date,order_purchase_timestamp,order_estimated_delivery_date,
301 TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY) as Days_taken_to_deliver,
302 TIMESTAMP_DIFF(order_estimated_delivery_date,order_purchase_timestamp,DAY) as Days_estimated_to_deliver,
303 TIMESTAMP_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY) as Estimated_vs_Delivered,
304 from `Target.orders`
305 where order_status = "delivered";
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	order_id	order_delivered_customer_date	order_purchase_timestamp	order_estimated_delivery_date	Da
1	cec8f5f7a13e5ab934a486ec9e...	2017-04-07 13:14:56 UTC	2017-03-17 15:56:47 UTC	2017-05-18 00:00:00 UTC	
2	58527ee4726911bee84a0f42c...	2017-03-30 14:04:04 UTC	2017-03-20 11:01:17 UTC	2017-05-18 00:00:00 UTC	
3	10ed5499d1623638ee810eff1...	2017-04-18 13:52:43 UTC	2017-03-21 13:38:25 UTC	2017-05-18 00:00:00 UTC	
4	818996ea247803ddc123789f2...	2018-08-29 22:52:40 UTC	2018-08-20 15:56:23 UTC	2018-10-04 00:00:00 UTC	
5	d195cac9ccaa1394ede717d38...	2018-08-23 02:08:44 UTC	2018-08-12 18:14:29 UTC	2018-10-04 00:00:00 UTC	
6	64eeb35d3ade7fcdff9fbb1ca5...	2018-08-23 00:09:45 UTC	2018-08-16 07:55:32 UTC	2018-10-04 00:00:00 UTC	

2. Create columns:

i. time_to_delivery = order_purchase_timestamp - order_delivered_customer_date

First for this we need to Add the Column

```
307
308
309 ALTER TABLE `Target.orders`
310 ADD COLUMN time_to_delivery NUMERIC;
311
312 UPDATE `Target.orders`
313 SET time_to_delivery = TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)
314 WHERE TRUE;
315
316
317
```

Press Alt+F1 for Acc

All results

Elapsed time	Statements processed	Job status
0 sec	1	ERROR

Query error: Column already exists: time_to_delivery at [2:12]

I am not able to update the Table here because DML queries are not allowed in the free tier.

But for time being we can use this:-

```
316
317
318 Select Distinct order_id,order_delivered_customer_date,order_purchase_timestamp,order_estimated_delivery_date,
319 TIMESTAMP_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY) as time_to_delivery
320 from `Target.orders` ;
321
322
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	order_id	order_delivered_customer_date	order_purchase_timestamp	order_estimated_delivery_date	tin
1	1950d777989f6a877539f5379...	2018-03-21 22:03:51 UTC	2018-02-19 19:48:52 UTC	2018-03-09 00:00:00 UTC	
2	2c45c33d2f9cb8ff8b1c86cc28...	2016-11-09 14:53:50 UTC	2016-10-09 15:39:56 UTC	2016-12-08 00:00:00 UTC	
3	65d1e226dfaeb8cdc42f66542...	2016-11-08 10:58:34 UTC	2016-10-03 21:01:41 UTC	2016-11-25 00:00:00 UTC	
4	635c894d068ac37e6e03dc54e...	2017-05-16 14:49:55 UTC	2017-04-15 15:37:38 UTC	2017-05-18 00:00:00 UTC	
5	3b97562c3aee8bdedcb5c2e45...	2017-05-17 10:52:15 UTC	2017-04-14 22:21:54 UTC	2017-05-18 00:00:00 UTC	
6	68f47f50f04c4cb6774570cfde...	2017-05-16 09:07:47 UTC	2017-04-16 14:56:13 UTC	2017-05-18 00:00:00 UTC	

ii. diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

Adding Second Column

```
322
323
324 Select Distinct order_id,order_delivered_customer_date,order_purchase_timestamp,order_estimated_delivery_date,
325 TIMESTAMP_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY) as time_to_delivery,
326 TIMESTAMP_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY) as diff_estimated_delivery
327 from `Target.orders` ;
328
```

Press Alt+F1 for Accessibility Options

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	order_id	order_delivered_customer_date	order_purchase_timestamp	order_estimated_delivery_date	tin
1	1950d777989f6a877539f5379...	2018-03-21 22:03:51 UTC	2018-02-19 19:48:52 UTC	2018-03-09 00:00:00 UTC	
2	2c45c33d2f9cb8ff8b1c86cc28...	2016-11-09 14:53:50 UTC	2016-10-09 15:39:56 UTC	2016-12-08 00:00:00 UTC	
3	65d1e226dfaeb8cdc42f66542...	2016-11-08 10:58:34 UTC	2016-10-03 21:01:41 UTC	2016-11-25 00:00:00 UTC	
4	635c894d068ac37e6e03dc54e...	2017-05-16 14:49:55 UTC	2017-04-15 15:37:38 UTC	2017-05-18 00:00:00 UTC	
5	3b97562c3aee8bdedcb5c2e45...	2017-05-17 10:52:15 UTC	2017-04-14 22:21:54 UTC	2017-05-18 00:00:00 UTC	
6	68f47f50f04c4cb6774570cfde...	2017-05-16 09:07:47 UTC	2017-04-16 14:56:13 UTC	2017-05-18 00:00:00 UTC	
7	276e9ec344d3bf029ff83a161c...	2017-05-22 14:11:31 UTC	2017-04-08 21:20:24 UTC	2017-05-18 00:00:00 UTC	

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

```
329
330 Select c.customer_state,
331 Avg(cte.freight_value) as MeanFreight,
332 Avg(cte.time_to_delivery) as MeanDeliveryTime,
333 Avg(cte.diff_estimated_delivery) as MeanEstimatedTime
334 from `Target.customers` as c
335 join (Select Distinct customer_id as cd,order_id,price,freight_value,customer_id,
336 TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY) as time_to_delivery,
337 TIMESTAMP_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY) as diff_estimated_delivery
338 From `Target.transaction_table`
339 where order_status = "delivered") as cte
340 on cte.cd = c.customer_id
341 group by c.customer_state
342
343
344
345
346
347
```

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	MeanFreight	MeanDeliveryTime	MeanEstimatedTime
1	GO	22.93532904502...	15.05343889163...	11.32409698169...
2	SP	15.23009784923...	8.263591117917...	10.23035509188...
3	RS	21.83680595930...	14.77902962020...	13.08141013992...
4	BA	26.44315318005...	18.80919677515...	10.03523439832...
5	MG	20.77359623609...	11.50290846877...	12.36698032506...
6	MT	27.96671351351...	17.42378378378...	13.71027027027...
7	RJ	21.03448175526...	14.76171752123...	11.00212330921...
8	SC	21.75269601100...	14.42145804676...	10.71471801925...

Load more

4. Sort the data to get the following:

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

First calculate the sum of freight

Untitled 6 RUN SAVE DOWNLOAD SHARE SCHEDULE MORE

```
346
347 Select c.customer_state, Round(Avg(FreightTot),2) as MeanValue from
348 `Target.customers` as c
349 join (
350 Select Distinct customer_id,order_id,sum(freight_value) as FreightTot
351 from `Target.transaction_table`
352 group by 1,2
353 ) as view_table
354 on view_table.customer_id = c.customer_id
355 group by 1
356 order by MeanValue desc
357 limit 5;
```

Press Alt+F1 for

Query results SAVE RESULTS EXPLORE

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

Row	customer_state	MeanValue
1	RR	48.59
2	PB	48.35
3	RO	46.22
4	AC	45.52
5	PI	43.04

We also need to check the Freight Value for Order Status – Delivered:

```
361
362 Select c.customer_state, Round(Avg(FreightTot),2) as MeanValue from
363 `Target.customers` as c
364 join (
365 Select Distinct customer_id,order_id,sum(freight_value) as FreightTot
366 from `Target.transaction_table`
367 where order_status = "delivered"
368 group by 1,2
369 ) as view_table
370 on view_table.customer_id = c.customer_id
371 group by 1
372 order by MeanValue desc
373 limit 5;
```

Query results SAVE RESULTS

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

Row	customer_state	MeanValue
1	PB	48.84
2	RR	48.34
3	RO	46.43
4	AC	45.55
5	PI	42.98

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

Highest Avg

```
375
376 Select c.customer_state,
377 Round(Avg(cte.time_to_delivery),2) as MeanDeliveryTime,
378 from `Target.customers` as c
379 join (Select Distinct customer_id,order_id,
380 TIMESTAMP_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY) as time_to_delivery
381 From `Target.transaction_table`
382 where order_status = "delivered") as cte
383 on cte.customer_id = c.customer_id
384 group by c.customer_state
385 order by MeanDeliveryTime desc
386 limit 5 ;
```

Press Alt+F1 for Acces

Query results

SAVE RESULTS | EXPLORE DATA

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

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

Lowest Avg

```
389
390 Select c.customer_state,
391 Round(Avg(cte.time_to_delivery),2) as MeanDeliveryTime,
392 from `Target.customers` as c
393 join (Select Distinct customer_id,order_id,
394 TIMESTAMP_DIFF(order_purchase_timestamp,order_delivered_customer_date,DAY) as time_to_delivery
395 From `Target.transaction_table`
396 where order_status = "delivered") as cte
397 on cte.customer_id = c.customer_id
398 group by c.customer_state
399 order by MeanDeliveryTime ASC
400 limit 5 ;
```

Press Alt+F1 for Acc

Query results

SAVE RESULTS | EXPLORE DAT

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

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

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

Fastest Delivery :-

```
404
405 Select c.customer_state,
406 Round(avg(cte.diff_estimated_delivery),2) as Meandiff_estimated_delivery,
407 from `Target.customers` as c
408 join (Select Distinct customer_id,order_id,
409 TIMESTAMP_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY) as diff_estimated_delivery
410 From `Target.transaction_table`
411 where order_status = "delivered") as cte
412 on cte.customer_id = c.customer_id
413 group by c.customer_state
414 order by Meandiff_estimated_delivery DESC
415 limit 5 ;
416
```

Press Alt+F1 for Accessibil

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	Meandiff_estimated
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

Slowest Delivery :-

```
418
419 Select c.customer_state,
420 Round(avg(cte.diff_estimated_delivery),2) as Meandiff_estimated_delivery,
421 from `Target.customers` as c
422 join (Select Distinct customer_id,order_id,
423 TIMESTAMP_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY) as diff_estimated_delivery
424 From `Target.transaction_table`
425 where order_status = "delivered") as cte
426 on cte.customer_id = c.customer_id
427 group by c.customer_state
428 order by Meandiff_estimated_delivery ASC
429 limit 5 ;
```

Press Alt+F1 for Accessibility Opt

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	Meandiff_estimated
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

6. Question 6:

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

```
Select Year,Month,payment_type,count(order_id) as OrdersCount from
(Select Distinct o.order_id,payment_type,
EXTRACT(Month from o.order_purchase_timestamp) as Month,
EXTRACT(Year from o.order_purchase_timestamp) as Year
from `Target.orders` as o
join `Target.payments` as p
on o.order_id = p.order_id) abc
group by 1,2,3
order by 1,2,3;
```

Press Alt+F1 for Accessibility O

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Year	Month	payment_type	OrdersCount
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	253
4	2016	10	debit_card	2
5	2016	10	voucher	11

If only MoM: -

```
Select Month,payment_type,count(order_id) as OrdersCount from
(Select Distinct o.order_id,payment_type,
EXTRACT(Month from o.order_purchase_timestamp) as Month
from `Target.orders` as o
join `Target.payments` as p
on o.order_id = p.order_id) abc
group by 1,2
order by 1,2;
```

Press Alt+F1 for Accessibility O

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Month	payment_type	OrdersCount
1	1	UPI	1715
2	1	credit_card	6093
3	1	debit_card	118
4	1	voucher	337
5	2	UPI	1723

2. Distribution of payment installments and count of orders

```
456
457 Select Month,payment_type,count(order_id)as OrdersCount ,
458 round(Avg(payment_installments),2) as MeanInstallment from
459 (Select Distinct o.order_id,payment_type,p.payment_installments,
460 EXTRACT(Month from o.order_purchase_timestamp) as Month
461 from `Target.orders` as o
462 join `Target.payments` as p
463 on o.order_id = p.order_id) abc
464 group by 1,2
465 order by OrdersCount desc, MeanInstallment desc;
```

Press Alt+F1 for Accessibility

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Month	payment_type	OrdersCount	MeanInstallment
1	5	credit_card	8329	3.61
2	8	credit_card	8253	3.53
3	7	credit_card	7828	3.59
4	3	credit_card	7698	3.47
5	4	credit_card	7287	3.38

7. Actionable Insights:

1. Checking all the order_id that has been cancelled and do some analysis on the product category

```
468
469
470 Select Distinct Order_id from `Target.transaction_table`
471 where order_status = 'canceled'
472
```

Press Alt+F1 for Accessibility Opti

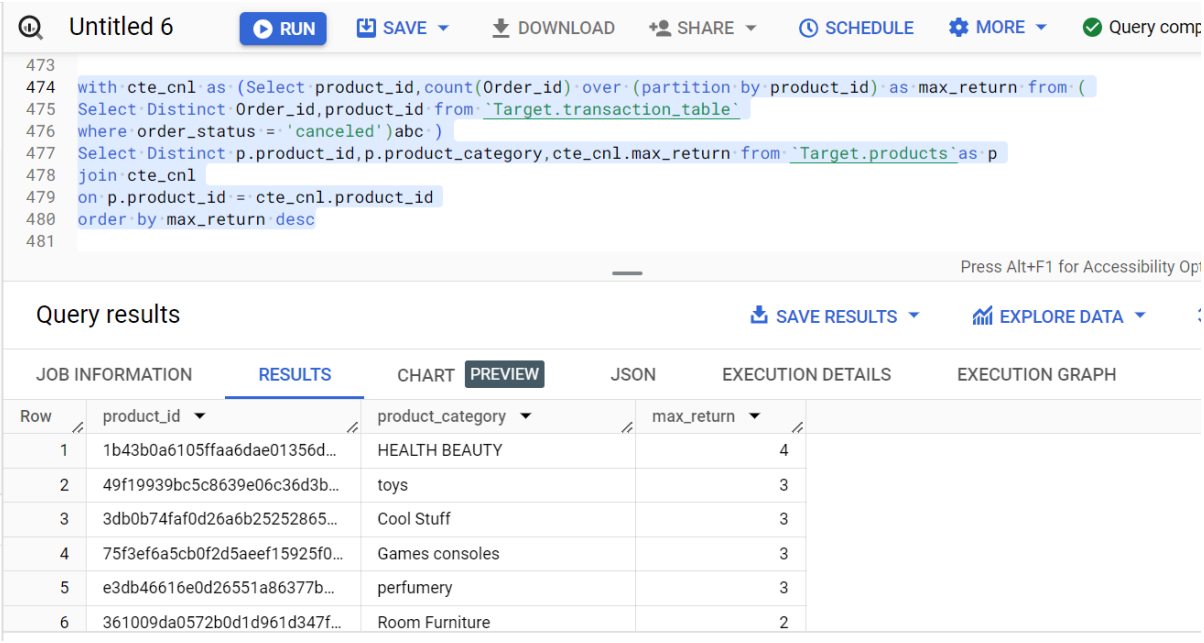
Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Order_id
1	6ef587afa4703fd874edb7ade8...
2	697705c05da5ef945de721dd3...
3	2db84a082bda455eb1da1072...
4	5ad5a1ef522da1732ab963ea1...
5	9aeaadde646cd9e4c51638f24...
6	c7da51b752a41bf0fc419eb6b...
7	e95c2973b9be01bfe69bd4e1f...
8	d688c0804e20281bdd7c7c864...

check the top category where we see the returns



Max Return is from HEALTH BEAUTY, Games consoles and toys, need to be cautious about these product categories

Take extra shipping charges if returned So only the customer who are very sure will buy the item.

2. We can try to reduce the time to deliver mean time I.e., Purchase time – Delivered time

8. Recommendations

- During the Night customers tend to buy more, we can give some promotion/discount offers to attract more customers.
- We can also check the customers who haven't bought in the recent 6 months but were buying before that and those Gone away customer, we can target them again as our Opportunity customer.
- See the Top buying product categories, we can create some product-based offer to attract more customers

Top Ten product Categories: -

```
485
486 with cte_cnl as (Select product_id,count(Order_id) over (partition by product_id) as max_return from (
487 Select Distinct Order_id,product_id from `Target.transaction_table`
488 where order_status = 'delivered')abc )
489
490 Select Distinct p.product_id,p.product_category,cte_cnl.max_return from `Target.products`as p
491 join cte_cnl
492 on p.product_id = cte_cnl.product_id
493 order by max_return desc
494
```

Press Alt+F1 for Accessibility Option

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	product_id	product_category	max_return
1	99a4788cb24856965c36a24e3...	bed table bath	456
2	aca2eb7d00ea1a7b8ebd4e683...	Furniture Decoration	425
3	422879e10f46682990de24d77...	Garden tools	352
4	d1c427060a0f73f6b889a5c7c...	computer accessories	313
5	389d119b48cf3043d311335e4...	Garden tools	309

Results per page: 50 1 – 50 of 32216

- During the festive period we observe more sales, so we can create a new buy list zone for the Christmas events.
- We can also share the promotion with the consent of the customer and send them direct/email promotional offers.
- There is a high sales using credit cards – We can try to give more offers discount for the same to attract more customer.