















SQL Project- Target

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

Snapshot:

	geolocation	 QUERY ▾	 SHARE	 COPY	 SNAPSHOT	 DELETE	 EXPORT
SCHEMA DETAILS PREVIEW LINEAGE							
 Filter Enter property name or value							
<input type="checkbox"/>	Field name	Type	Mode	Key	Collation	Default value	Policy tags
<input type="checkbox"/>	geolocation_zip_code_prefix	INTEGER	NULLABLE				
<input type="checkbox"/>	geolocation_lat	FLOAT	NULLABLE				
<input type="checkbox"/>	geolocation_lng	FLOAT	NULLABLE				
<input type="checkbox"/>	geolocation_city	STRING	NULLABLE				
<input type="checkbox"/>	geolocation_state	STRING	NULLABLE				

	order_items	 QUERY ▾	 SHARE	 COPY	 SNAPSHOT	
SCHEMA DETAILS PREVIEW LINEAGE						
 Filter Enter property name or value						
<input type="checkbox"/>	Field name	Type	Mode	Key	Collation	Default
<input type="checkbox"/>	order_id	STRING	NULLABLE			
<input type="checkbox"/>	order_item_id	INTEGER	NULLABLE			
<input type="checkbox"/>	product_id	STRING	NULLABLE			
<input type="checkbox"/>	seller_id	STRING	NULLABLE			
<input type="checkbox"/>	shipping_limit_date	TIMESTAMP	NULLABLE			
<input type="checkbox"/>	price	FLOAT	NULLABLE			
<input type="checkbox"/>	freight_value	FLOAT	NULLABLE			

Insights:

The tables show to have multiple data types depending on the value stored in respective column. For e.g.: Id column has type as Integer, Date column has type as DateTime/TimeStamp, Columns like Price, freight value has data type as float to store values after decimal.

2. Time period for which the data is given


Query:

```
SELECT min(order_purchase_timestamp) as first_placed_order,max(order_purchase_timestamp) as last_placed_order,
min(order_delivered_customer_date) as first_delivered_order,max(order_delivered_customer_date) as last_delivered_order
FROM `dsml-sql-beginning-day1.Target.orders`
```

Snapshot:

```
1 SELECT min(order_purchase_timestamp) as first_placed_order,max(order_purchase_timestamp) as last_placed_order,
2 min(order_delivered_customer_date) as first_delivered_order,max(order_delivered_customer_date)as last_delivered_order
3 FROM `dsml-sql-beginning-day1.Target.orders`
```

Query results

 SAVE RESULTS

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	first_placed_order	last_placed_order	first_delivered_order	last_delivered_order		
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	2016-10-11 13:46:32 UTC	2018-10-17 13:22:46 UTC		

Insights:

The selected data shows that the first order was placed on 04 Sept,2016 and last order placed was on 17 Oct, 2018. Also, the first order delivered was on 11 Oct, 2016 and last order was delivered on 17 Oct, 2018.

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

Query:

1.

```
select distinct customer_city,customer_state,customer_zip_code_prefix
from `Target.customers` c
```
2.

```
select distinct customer_city,customer_state
from `Target.customers` c
inner join `Target.orders` o
on c.customer_id=o.customer_id
order by customer_city
```

Snapshot:

```

5
6 select distinct customer_city, customer_state
7 from `Target.customers` c order by customer_city
8
9
10

```

Press Alt+F1 for accessibility options

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_city	customer_state			
1	abadia dos dourados	MG			
2	abadiania	GO			
3	abaete	MG			
4	abaetetuba	PA			
5	abaiaira	CE			
6	abaira	BA			
7	abare	BA			
8	abatia	PR			
9	abdon batista	SC			
10	abelardo luz	SC			
11	abranes	BA			
12	abre campo	MG			
13	abreu e lima	PE			
14	acaiaca	MG			

Results per page: 200 1 – 200 of 4310

2. In-depth Exploration:

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

Query for YEAR by YEAR and month by month analysis side by side (With respect to Order Counts)

```

with Year2016 as(
select extract(month from order_purchase_timestamp) as Month,
count(order_id) as Order_count_2016
from `Target.orders`
where extract(year from order_purchase_timestamp)=2016
group by Month
order by Month),
Year2017 as(select extract(month from order_purchase_timestamp) as Month,
count(order_id) as Order_count_2017
from `Target.orders`
where extract(year from order_purchase_timestamp)=2017
group by Month
order by Month),
Year_2018 as(select extract(month from order_purchase_timestamp) as Month,
count(order_id) as Order_count_2018
from `Target.orders`
where extract(year from order_purchase_timestamp)=2018
group by Month
order by Month)

select y7.month,
y6.Order_count_2016 as OC_2016,
y7.Order_count_2017 as OC_2017,
y8.Order_count_2018 as OC_2018
from Year2016 y6 right join Year2017 y7
on y6.month=y7.month
left join Year_2018 y8

```

on y7.month=y8.month
order by y7.Month

Snapshot:

Row	month	OC_2016	OC_2017	OC_2018
1	1	null	800	7269
2	2	null	1780	6728
3	3	null	2682	7211
4	4	null	2404	6939
5	5	null	3700	6873
6	6	null	3245	6167
7	7	null	4026	6292
8	8	null	4331	6512
9	9	4	4285	16
10	10	324	4631	4
11	11	null	7544	null
12	12	1	5673	null

Insights:

Having a glance at the above data we can see that in start of 2017 the number of orders place were less but gradually by end of the year we see a subsequent growth in order count. The growth in order count was maintained till Aug 2018. The **highest order count** was observed in month of **Nov 2017**.

Query for YEAR by YEAR and month by month analysis side by side (With respect to Total Payments made)

```
with Year2016 as(
select extract(month from order_purchase_timestamp) as Month,
round(sum(p.payment_value),2) as Total_Payment_2016
from `Target.orders` o
inner join `Target.payments` p
on o.order_id=p.order_id
where extract(year from order_purchase_timestamp)=2016
group by Month
order by Month),
Year2017 as(select extract(month from order_purchase_timestamp) as Month,
round(sum(p.payment_value),2) as Total_Payment_2017
from `Target.orders` o
inner join `Target.payments` p
on o.order_id=p.order_id
where extract(year from order_purchase_timestamp)=2017
group by Month
order by Month),
Year_2018 as(select extract(month from order_purchase_timestamp) as Month,
round(sum(p.payment_value),2) as Total_Payment_2018
from `Target.orders` o
inner join `Target.payments` p
on o.order_id=p.order_id
where extract(year from order_purchase_timestamp)=2018
group by Month
```

order by Month)

```
select y7.month,  
y6.Total_Payment_2016 as Total_2016,  
y7.Total_Payment_2017 as Total_2017,  
y8.Total_Payment_2018 as Total_2018  
from Year2016 y6 right join Year2017 y7  
on y6.month=y7.month  
left join Year_2018 y8  
on y7.month=y8.month  
order by y7.Month
```

Snapshot:

Row	month	Total_2016	Total_2017	Total_2018
1	1	null	138488.04	1115004.18
2	2	null	291908.01	992463.34
3	3	null	449863.6	1159652.12
4	4	null	417788.03	1160785.48
5	5	null	592918.82	1153982.15
6	6	null	511276.38	1023880.5
7	7	null	592382.92	1066540.75
8	8	null	674396.32	1022425.32
9	9	252.24	727762.45	4439.54
10	10	59090.48	779677.88	589.67
11	11	null	1194882.8	null
12	12	19.62	878401.48	null

Insights:

Above data shows that, revenue for each month of 2017 has been increasing from start to end of the year. The **highest peak** in sales/revenue is observed in month of **Nov 2017**. In year 2018, the revenue has been mostly stable till Aug 2018.

Query for Only Year by Year analysis (With Respect to Order Count):

```
select  
FORMAT_DATETIME("%Y", order_purchase_timestamp) as Year,  
count(distinct order_id) as Order_count  
from `Target.orders`  
group by Year  
order by Year;
```

Snapshot:

Row	Year	Order_count
1	2016	329
2	2017	45101
3	2018	54011

Insights:

We simply see a **growing trend of orders coming** in for Target year by year. The growth between 2016 and 2017 is subsequently more than what we see between 2017 and 2018.

Query for Only Year by Year analysis (With Respect to Total Payment):

```
select
FORMAT_DATETIME("%Y", order_purchase_timestamp) as Year,
sum(payment_value) as Total_Payment
from `Target.orders` o
inner join `Target.payments` p
on o.order_id=p.order_id
group by Year
order by Year;
```

Snapshot:

Row	Year	Total_Payment
1	2016	59362.34
2	2017	7249746.73
3	2018	8699763.05

Insights:

We simply see a **growing trend of revenue coming** in for Target year by year. The growth between 2016 and 2017 is subsequently more than what we see between 2017 and 2018.

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

Query:

```
with Time_data as(
select order_purchase_timestamp ,
TIME(order_purchase_timestamp) as formatted_time,
case when TIME(order_purchase_timestamp) between TIME(04,00,00) and TIME(05,59,59) then
'Dawn'
when TIME(order_purchase_timestamp) between TIME(06,00,00) and TIME(11,59,59) then 'Morning'
when TIME(order_purchase_timestamp) between TIME(12,00,00) and TIME(17,59,59) then
'Afternoon'
when TIME(order_purchase_timestamp) between TIME(18,00,00) and TIME(20,59,59) then 'Evening'
when TIME(order_purchase_timestamp) between TIME(21,00,00) and TIME(23,59,59) then 'Night'
when TIME(order_purchase_timestamp) between TIME(00,00,00) and TIME(03,59,59) then 'Midnight'
end as Time_of_the_Day
from `Target.orders`
```

order by Time_of_the_Day)

```
select Time_of_the_Day,  
count(*) as Order_Count  
from Time_data  
group by Time_of_the_Day  
order by Order_Count
```

Snapshot:

```
1 with Time_data as(  
2 | select order_purchase_timestamp ,  
3 TIME(order_purchase_timestamp) as formatted_time,  
4 case when TIME(order_purchase_timestamp) between TIME(04,00,00) and TIME(05,59,59) then 'Dawn'  
5 when TIME(order_purchase_timestamp) between TIME(06,00,00) and TIME(11,59,59) then 'Morning'  
6 when TIME(order_purchase_timestamp) between TIME(12,00,00) and TIME(17,59,59) then 'Afternoon'  
7 when TIME(order_purchase_timestamp) between TIME(18,00,00) and TIME(20,59,59) then 'Evening'  
8 when TIME(order_purchase_timestamp) between TIME(21,00,00) and TIME(23,59,59) then 'Night'  
9 when TIME(order_purchase_timestamp) between TIME(00,00,00) and TIME(03,59,59) then 'Midnight'  
10 end as Time_of_the_Day  
11 from `Target.orders`  
12 order by Time_of_the_Day)  
13  
14 select Time_of_the_Day,  
15 count(*) as Order_Count  
16 from Time_data  
17 group by Time_of_the_Day  
18 order by Order_Count
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	Time_of_the_Day	Order_Count				
1	Dawn	394				
2	Midnight	4346				
3	Night	16156				
4	Evening	17944				
5	Morning	22240				
6	Afternoon	38361				

Insights:

The max number of orders are placed by customers during afternoon span of time, followed by morning span, evening span to night span.

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

1. Get month on month orders by states

Query:

```
select distinct customer_state,  
sum(if(extract(month from order_purchase_timestamp)=1,1,0)) as Jan_count,  
sum(if(extract(month from order_purchase_timestamp)=2,1,0)) as Feb_count,  
sum(if(extract(month from order_purchase_timestamp)=3,1,0)) as Mar_count,  
sum(if(extract(month from order_purchase_timestamp)=4,1,0)) as Apr_count,  
sum(if(extract(month from order_purchase_timestamp)=5,1,0)) as May_count,  
sum(if(extract(month from order_purchase_timestamp)=6,1,0)) as Jun_count,
```

```

sum(if(extract(month from order_purchase_timestamp)=7,1,0)) as Jul_count,
sum(if(extract(month from order_purchase_timestamp)=8,1,0)) as Aug_count,
sum(if(extract(month from order_purchase_timestamp)=9,1,0)) as Sep_count,
sum(if(extract(month from order_purchase_timestamp)=10,1,0)) as Oct_count,
sum(if(extract(month from order_purchase_timestamp)=11,1,0)) as Nov_count,
sum(if(extract(month from order_purchase_timestamp)=12,1,0)) as Dec_count
from `Target.customers` c
inner join `Target.orders` o
on c.customer_id=o.customer_id
group by customer_state
order by customer_state

```

Snapshot:

Row	customer_state	Jan_count	Feb_count	Mar_count	Apr_count	May_count	Jun_count	Jul_count	Aug_count	Sep_count	Oct_count	Nov_count	Dec_count
1	AC	8	6	4	9	10	7	9	7	5	6	5	5
2	AL	39	39	40	51	46	34	40	34	20	30	26	14
3	AM	12	16	14	19	19	8	23	9	9	3	10	6
4	AP	11	4	8	5	11	4	7	5	2	3	4	4
5	BA	264	273	340	318	368	307	405	323	170	170	250	192
6	CE	99	101	126	143	136	121	140	130	77	74	108	81
7	DF	151	196	207	183	208	220	243	232	97	104	168	131
8	ES	159	186	182	188	228	204	206	200	93	104	170	113
9	GO	164	176	199	177	226	184	192	213	88	117	157	127
10	MA	66	67	77	73	65	59	79	70	42	52	56	41
11	MG	971	1063	1237	1061	1190	1080	1111	1177	511	600	943	691
12	MS	71	75	79	58	74	76	74	59	33	34	46	36
13	MT	96	84	71	92	104	83	85	78	35	55	74	50

Results per page: 50 1 - 27 of 27

Insights:

Going through data is observed that max number of orders are placed by customers from state named SP, RJ and MG for each month.

And, the minimum number of order placed by customers from states are : RR, AC, AP for all months individually.

2. Distribution of customers across the states in Brazil

Query:

```

select distinct customer_state,
count(*) as Number_of_Customers
from `Target.customers` c
group by customer_state
order by customer_state

```

Snapshot:

1	select distinct customer_state,
2	count(*) as Number_of_Customers
3	from `Target.customers` c
4	group by customer_state
5	order by customer_state
6	

Query results			
JOB INFORMATION		RESULTS	JSON
Row	customer_state	Number_of_Cust	
1	AC	81	
2	AL	413	
3	AM	148	
4	AP	68	
5	BA	3380	
6	CE	1336	
7	DF	2140	
8	ES	2033	
9	GO	2020	
10	MA	747	
11	MG	11635	
12	MS	715	
13	MT	907	
14	PA	975	

Insights:

The states having highest number of customers with Target are SP, RJ and MG in the mentioned order, while the states with lowest number of customers are RR, AP and AC.

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

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

Query:

```
select Year,
Yearly_Sale,
((Yearly_Sale - LAG(Yearly_Sale,1) over (order by Year)) / (LAG(Yearly_Sale,1) over (order by Year)))
* 100 as Growth_percent
from
(select extract(year from order_purchase_timestamp) as Year,
round(sum(p.payment_value),2) as Yearly_Sale
from `Target.orders` o
join `Target.payments` p
on o.order_id=p.order_id
where extract(month from o.order_purchase_timestamp) between 1 and 8
group by Year)
order by Year
```

Snapshot:

```

1 select Year,
2 Yearly_Sale,
3 ((Yearly_Sale-LAG(Yearly_Sale,1) over (order by Year))/(LAG(Yearly_Sale,1) over (order by Year)))*100 as Growth_percent
4 from
5 (select extract(year from order_purchase_timestamp) as Year,
6 round(sum(p.payment_value),2) as Yearly_Sale
7 from `Target.orders` o
8 join `Target.payments` p
9 on o.order_id=p.order_id
10 where extract(month from o.order_purchase_timestamp) between 1 and 8
11 group by Year)
12 order by Year
13

```

Press A

Query results

[SAVE RESULTS](#) [EX](#)

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH [PREVIEW](#)

Row	Year	Yearly_Sale	Growth_percent
1	2017	3669022.12	null
2	2018	8694733.84	136.976871...

Insights:

The data above shows the revenue taking place in 2017 & 2018 for first 8 months in each year. We can observe that there is approx. 136% of growth in the sale during the mentioned duration.

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

Query:

```

select customer_state,
round(sum(price),2) as Sum_Price,
round(avg(price),2) as Mean_Price,
round(sum(freight_value),2) as SUM_Freight,
round(avg(freight_value),2) as Mean_Freight
from `Target.customers` c
join `Target.orders` o
on c.customer_id=o.customer_id
join `Target.order_items` oi
on o.order_id=oi.order_id
group by customer_state

```

Snapshot:

```

1  select customer_state,
2  round(sum(price),2) as Sum_Price,
3  round(avg(price),2) as Mean_Price,
4  round(sum(freight_value),2) as SUM_Freight,
5  round(avg(freight_value),2) as Mean_Freight
6  from `Target.customers` c
7  join `Target.orders` o
8  on c.customer_id=o.customer_id
9  join `Target.order_items` oi
10 on o.order_id=oi.order_id
11 group by customer_state

```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	Sum_Price	Mean_Price	SUM_Freight	Mean_Freight	
1	RN	83034.98	156.97	18860.1	35.65	
2	CE	227254.71	153.76	48351.59	32.71	
3	RS	750304.02	120.34	135522.74	21.74	
4	SC	520553.34	124.65	89660.26	21.47	
5	SP	5202955.05	109.65	718723.07	15.15	
6	MG	1585308.03	120.75	270853.46	20.63	
7	BA	511349.99	134.6	100156.68	26.36	
8	RJ	1824092.67	125.12	305589.31	20.96	
9	GO	294591.95	126.27	53114.98	22.77	
10	MA	119648.22	145.2	31523.77	38.26	
11	PE	262788.03	145.51	59449.66	32.92	

Insights:

In above dataset, we can observe sum and average of order price and freight value individually for each state.

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

Query:

```

select order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY) as
Days_btwn_delivered_purchase,
DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp,DAY) as
Days_btwn_est_delivered_purchase,
DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,DAY) as
Days_btwn_est_delivered_delivered
from `Target.orders`
order by order_purchase_timestamp

```

Snapshot:

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	Days_btwn_deliv	Days_btwn_est	Days_btwn_est
1	2016-09-04 21:15:19 UTC	null	2016-10-20 00:00:00 UTC	null	45	null
2	2016-09-05 00:15:34 UTC	null	2016-10-28 00:00:00 UTC	null	52	null
3	2016-09-13 15:24:19 UTC	null	2016-09-30 00:00:00 UTC	null	16	null
4	2016-09-15 12:16:38 UTC	2016-11-09 07:47:38 UTC	2016-10-04 00:00:00 UTC	54	18	-36
5	2016-10-02 22:07:52 UTC	null	2016-10-25 00:00:00 UTC	null	22	null
6	2016-10-03 09:44:50 UTC	2016-10-26 14:02:13 UTC	2016-10-27 00:00:00 UTC	23	23	0
7	2016-10-03 16:56:50 UTC	2016-10-27 18:19:38 UTC	2016-11-07 00:00:00 UTC	24	34	10
8	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	52	16
9	2016-10-03 21:13:36 UTC	2016-11-03 10:58:07 UTC	2016-11-29 00:00:00 UTC	30	56	25
10	2016-10-03 22:06:03 UTC	2016-10-31 11:07:42 UTC	2016-11-23 00:00:00 UTC	27	50	22
11	2016-10-03 22:31:31 UTC	2016-10-14 16:08:00 UTC	2016-11-23 00:00:00 UTC	10	50	39

Results per page: 50 1 - 50 of 99441

Insights:

The above result set shows the days taken for a purchased order to deliver with comparison to estimated delivery for each order present in the database.

- Find time to delivery & diff estimated delivery. Formula for the same given below:
 - time to delivery = order delivered customer date - order purchase timestamp
 - diff estimated delivery = order estimated delivery date - order delivered customer date

Query:

```
select order_purchase_timestamp,
order_delivered_customer_date,
order_estimated_delivery_date,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) as time_to_delivery,
DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) as
diff_estimated_delivery
from `Target.orders`
order by order_purchase_timestamp
```

Snapshot:

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_c
1	2016-09-04 21:15:19 UTC	null	2016-10-20 00:00:00 UTC	null	null
2	2016-09-05 00:15:34 UTC	null	2016-10-28 00:00:00 UTC	null	null
3	2016-09-13 15:24:19 UTC	null	2016-09-30 00:00:00 UTC	null	null
4	2016-09-15 12:16:38 UTC	2016-11-09 07:47:38 UTC	2016-10-04 00:00:00 UTC	54	-36
5	2016-10-02 22:07:52 UTC	null	2016-10-25 00:00:00 UTC	null	null
6	2016-10-03 09:44:50 UTC	2016-10-26 14:02:13 UTC	2016-10-27 00:00:00 UTC	23	0
7	2016-10-03 16:56:50 UTC	2016-10-27 18:19:38 UTC	2016-11-07 00:00:00 UTC	24	10
8	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	16
9	2016-10-03 21:13:36 UTC	2016-11-03 10:58:07 UTC	2016-11-29 00:00:00 UTC	30	25
10	2016-10-03 22:06:03 UTC	2016-10-31 11:07:42 UTC	2016-11-23 00:00:00 UTC	27	22
11	2016-10-03 22:31:31 UTC	2016-10-14 16:08:00 UTC	2016-11-23 00:00:00 UTC	10	39

Insights:

The above result set shows the days taken for a purchased order to deliver with comparison to estimated delivery for each order present in the database.

3. Group data by state, take mean of freight value, time to delivery, diff estimated delivery

Query:

```
select customer_state,
Round(AVG(freight_value),2) as AVG_Freight_Value,
Round(AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)),2) as Avg_Time_to_delivery,
Round(Avg(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY)),2) as Avg_Diff_estimated_delivery
from `Target.orders` o
join `Target.customers` c
on o.customer_id=c.customer_id
join `Target.order_items` oi
on o.order_id=oi.order_id
group by customer_state
order by customer_state
```

Snapshot:

Row	customer_state	AVG_Freight_Value	Avg_Time_to_delivery	Avg_Diff_estimated_delivery
1	AC	40.07	20.33	20.01
2	AL	35.84	23.99	7.98
3	AM	33.21	25.96	18.98
4	AP	34.01	27.75	17.44
5	BA	26.36	18.77	10.12
6	CE	32.71	20.54	10.26
7	DF	21.04	12.5	11.27
8	ES	22.06	15.19	9.77
9	GO	22.77	14.95	11.37
10	MA	38.26	21.2	9.11
11	MG	20.63	11.52	12.4

Insights:

The above result set shows the average freight values and days taken for a purchased order to deliver with comparison to estimated delivery for each state present in the database.

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

Query for 5 Highest Avg Freight Value:

```
select customer_state,
Round(AVG(freight_value),2) as AVG_Freight_Value,
from `Target.orders` o
join `Target.customers` c
on o.customer_id=c.customer_id
join `Target.order_items` oi
on o.order_id=oi.order_id
group by customer_state
order by AVG_Freight_Value desc
```

limit 5

Snapshot:

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

Query for 5 Lowest Avg Freight Value:

```
select customer_state,
Round(AVG(freight_value),2) as AVG_Freight_Value,
from `Target.orders` o
join `Target.customers` c
on o.customer_id=c.customer_id
join `Target.order_items` oi
on o.order_id=oi.order_id
group by customer_state
order by AVG_Freight_Value
limit 5
```

Snapshot:

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

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

Query for 5 Highest Avg Time To Delivery:

```
select customer_state,
Round(AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)),2) as
Avg_Time_to_delivery,
from `Target.orders` o
join `Target.customers` c
on o.customer_id=c.customer_id
join `Target.order_items` oi
on o.order_id=oi.order_id
group by customer_state
order by Avg_Time_to_delivery desc
limit 5
```

Snapshot:

Row	customer_state	Avg_Time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

Insights:

The result set the states where the average time to delivery is highest.

Query for 5 Lowest Avg Time To Delivery:

```
select customer_state,  
Round(AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)),2) as  
Avg_Time_to_delivery,  
from `Target.orders` o  
join `Target.customers` c  
on o.customer_id=c.customer_id  
join `Target.order_items` oi  
on o.order_id=oi.order_id  
group by customer_state  
order by Avg_Time_to_delivery  
limit 5
```

Snapshot:

Row	customer_state	Avg_Time_to_delivery
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

Insights:

The result set the states where the average time to delivery is lowest (fast delivery).

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

Query for Fast Delivery:

```
select customer_state,  
Round(Avg(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY)),2) as  
Avg_Diff_estimated_delivery  
from `Target.orders` o  
join `Target.customers` c  
on o.customer_id=c.customer_id
```

```

join `Target.order_items` oi
on o.order_id=oi.order_id
group by customer_state
order by Avg_Diff_estimated_delivery
limit 5

```

Snapshot:

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

Query for Slow Delivery (Delayed Delivery):

```

select customer_state,
Round(Avg(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY)),2) as
Avg_Diff_estimated_delivery
from `Target.orders` o
join `Target.customers` c
on o.customer_id=c.customer_id
join `Target.order_items` oi
on o.order_id=oi.order_id
group by customer_state
order by Avg_Diff_estimated_delivery desc
limit 5

```

Snapshot:

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

6. Payment type analysis:

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

Query

```

select EXTRACT(Month from order_purchase_timestamp) as Month,
EXTRACT(Year from order_purchase_timestamp) as Year,
sum(if(payment_type='credit_card',1,0)) as Credit_card_count,
sum(if(payment_type='voucher',1,0)) as Voucher_count,
sum(if(payment_type='debit_card',1,0)) as Debit_card_count,
sum(if(payment_type='UPI',1,0)) as UPI_count,
sum(if(payment_type='not_defined',1,0)) as No_Type_count,
from

```



```
(select order_purchase_timestamp,
payment_type
from `Target.orders` o
join `Target.payments` p
on o.order_id=p.order_id)
group by Month, Year
order by Year,Month
```

Snapshot:

Row	Month	Year	Credit_card_coun	Voucher_count	Debit_card_coun	UPI_count	No_Type_count
1	9	2016	3	0	0	0	0
2	10	2016	254	23	2	63	0
3	12	2016	1	0	0	0	0
4	1	2017	583	61	9	197	0
5	2	2017	1356	119	13	398	0
6	3	2017	2016	200	31	590	0
7	4	2017	1846	202	27	496	0
8	5	2017	2853	289	30	772	0
9	6	2017	2463	239	27	707	0
10	7	2017	3086	364	22	845	0
11	8	2017	3284	294	34	938	0

Results per page: 50 ▼ 1 – 25 of 25

Insights:

The result set depicts that the greatest number of payments across the specified period has been made by Credit Cards, which is followed by UPI and Voucher payments.

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

Query:

```
select payment_installments,
count(order_id) as count_of_orders
from `Target.payments`
group by payment_installments
order by payment_installments
```

Snapshot:

Row	payment_install	count_of_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328
12	11	23
13	12	133

Insights:

The result set depicts that majority of customers tend to pay the amount of order in a single instalment.