TARGET BUSINESS CASE STUDY

TOPIC - SQL

Business Problems & Solution Queries along with output Tables : -

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

Q.1.1) Data type of columns in a table

ANS :- Query -

```
select
        column_name,
        Data_type
from Target_Buisness_case.INFORMATION_SCHEMA.COLUMNS;
```

Query results

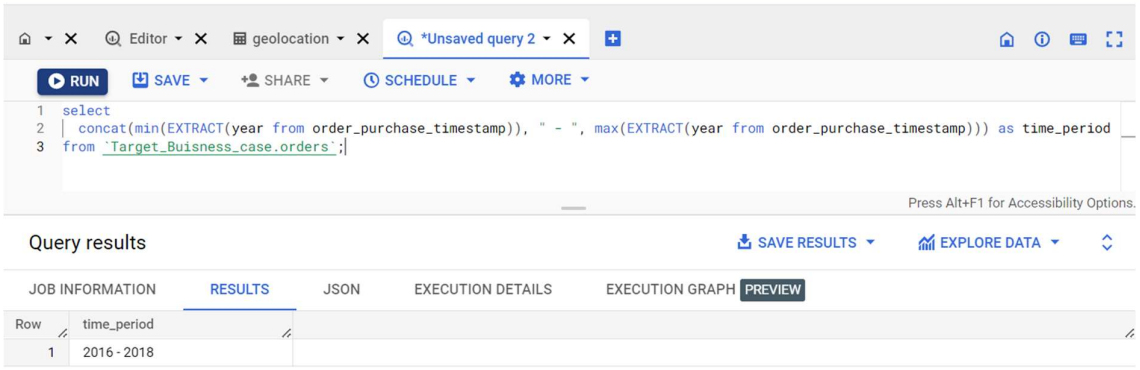
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	column_name	Data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64
8	seller_id	STRING
9	seller_zip_code_prefix	INT64
10	seller_city	STRING

Q.1.2) Time period for which the data is given

ANS :- Query -

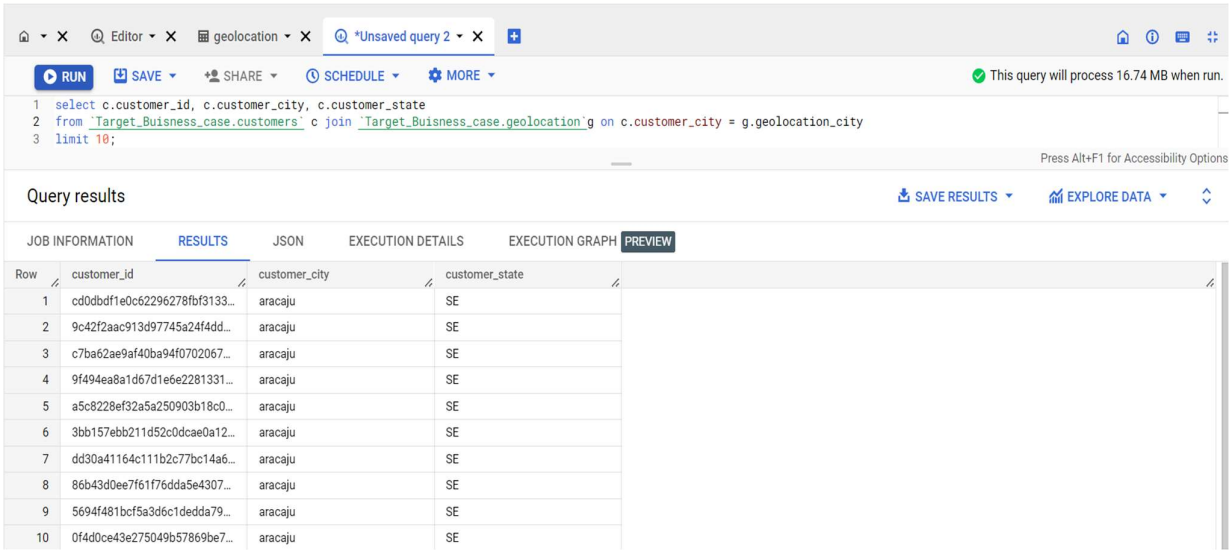
```
select
        concat(min(EXTRACT(year from order_purchase_timestamp)), " - ",
        max(EXTRACT(year from order_purchase_timestamp))) as time_period
from `Target_Buisness_case.orders`;
```



Q.1.3) Cities and States of customers ordered during the given period

ANS :- Query -

```
select
        c.customer_id,
        c.customer_city,
        c.customer_state
from `Target_Buisness_case.customers` c
join `Target_Buisness_case.geolocation`g
on c.customer_city = g.geolocation_city
limit 10;
```



Row	customer_id	customer_city	customer_state
1	cd0dbdf1e0c62296278fbf3133...	aracaju	SE
2	9c42f2aac913d97745a24f4dd...	aracaju	SE
3	c7ba62ae9af40ba94f0702067...	aracaju	SE
4	9f494ea8a1d67d1e6e2281331...	aracaju	SE
5	a5c8228ef32a5a250903b18c0...	aracaju	SE
6	3bb157ebb211d52c0dcae0a12...	aracaju	SE
7	dd30a41164c111b2c77bc14a6...	aracaju	SE
8	86b43d0ee7f61f76dda5e4307...	aracaju	SE
9	5694f481bcf5a3d6c1dedda79...	aracaju	SE
10	0f4d0ce43e275049b57869be7...	aracaju	SE

- ***2. In-depth Exploration:***

Q.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?

ANS :- Query -

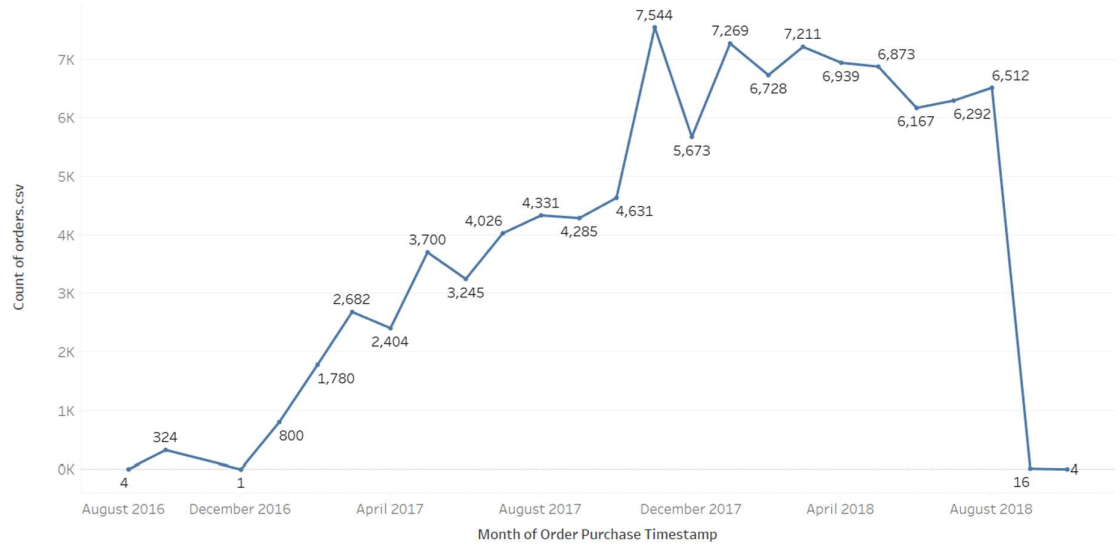
```
with t as

(
select
        EXTRACT(year from order_purchase_timestamp) as year,
        EXTRACT(month from order_purchase_timestamp) as month,
        count(order_id) as order_count
from `Target_Buisness_case.orders`
group by order_purchase_timestamp
)
select t.year, t.month, sum(t.order_count)as order_count1,
        row_number()over(order by sum(t.order_count)) as rank
from t
group by t.year, t.month
order by t.year asc, t.month asc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year	month	order_count1	rank
1	2016	9	4	2
2	2016	10	324	5
3	2016	12	1	1
4	2017	1	800	6
5	2017	2	1780	7
6	2017	3	2682	9
7	2017	4	2404	8
8	2017	5	3700	11
9	2017	6	3245	10
10	2017	7	4026	12

Trend in Sales



Q.2.2) What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

ANS :- Query -

```
with t as
(
select
case
when extract(hour from order_purchase_timestamp) between 4 and 8 then "Dawn"
when extract(hour from order_purchase_timestamp) between 8 and 12 then "Morning"
when extract(hour from order_purchase_timestamp) between 12 and 17 then "Afternoon"
when extract(hour from order_purchase_timestamp) between 17 and 21 then "Evening"
else "Night"
end as time_of_buy,
count(order_id) as order_count
from `Target_Buisness_case.orders`
group by order_purchase_timestamp
)
select
t.time_of_buy, sum(order_count) as order_count1,
from t
group by time_of_buy
order by 2 desc;
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	time_of_buy	order_count1
1	Afternoon	32366
2	Evening	24161
3	Morning	23535
4	Night	14285
5	Dawn	5094

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

Q.3.1) Get month on month orders by states

ANS :- Query -

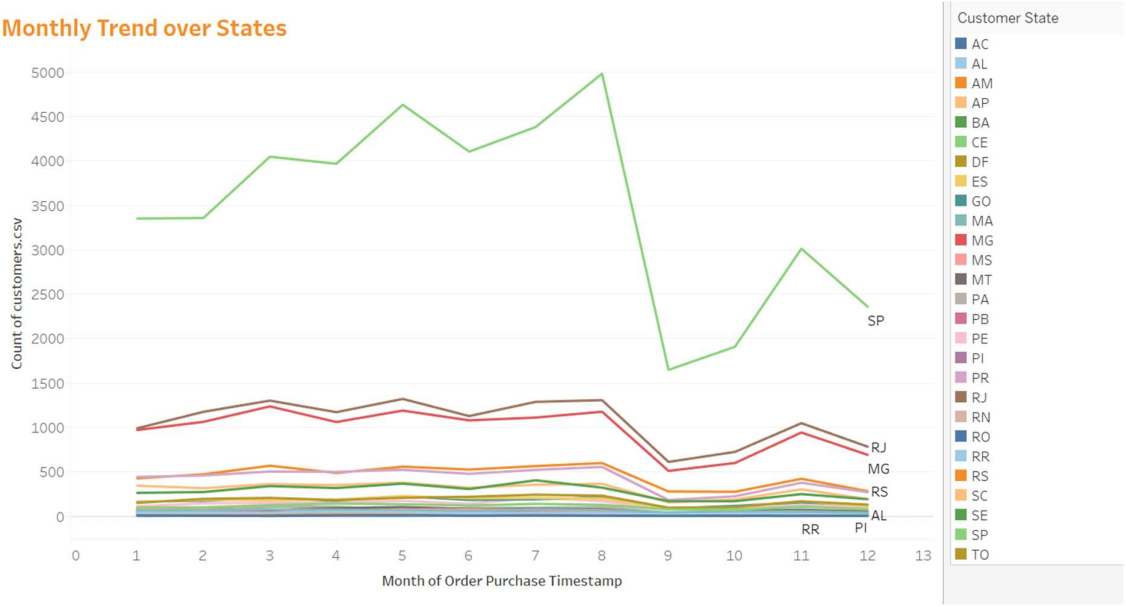
```
with t as
(select
        customer_id,
        order_id,
        EXTRACT(month from order_purchase_timestamp) as order_month,
from `Target_Buisness_case.orders`),
a as
(select
        customer_id,
        customer_state
from `Target_Buisness_case.customers`)
select
        a.customer_state,
        t.order_month,
        count(order_id) as order_count1
from t
join a on t.customer_id = a.customer_id
group by a.customer_state, t.order_month
order by t.order_month asc, order_count1 desc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	order_month	order_count1
1	SP	1	3351
2	RJ	1	990
3	MG	1	971
4	PR	1	443
5	RS	1	427
6	SC	1	345
7	BA	1	264
8	GO	1	164
9	ES	1	159
10	DF	1	151

Monthly Trend over States



Q.3.2) Distribution of customers across the states in Brazil

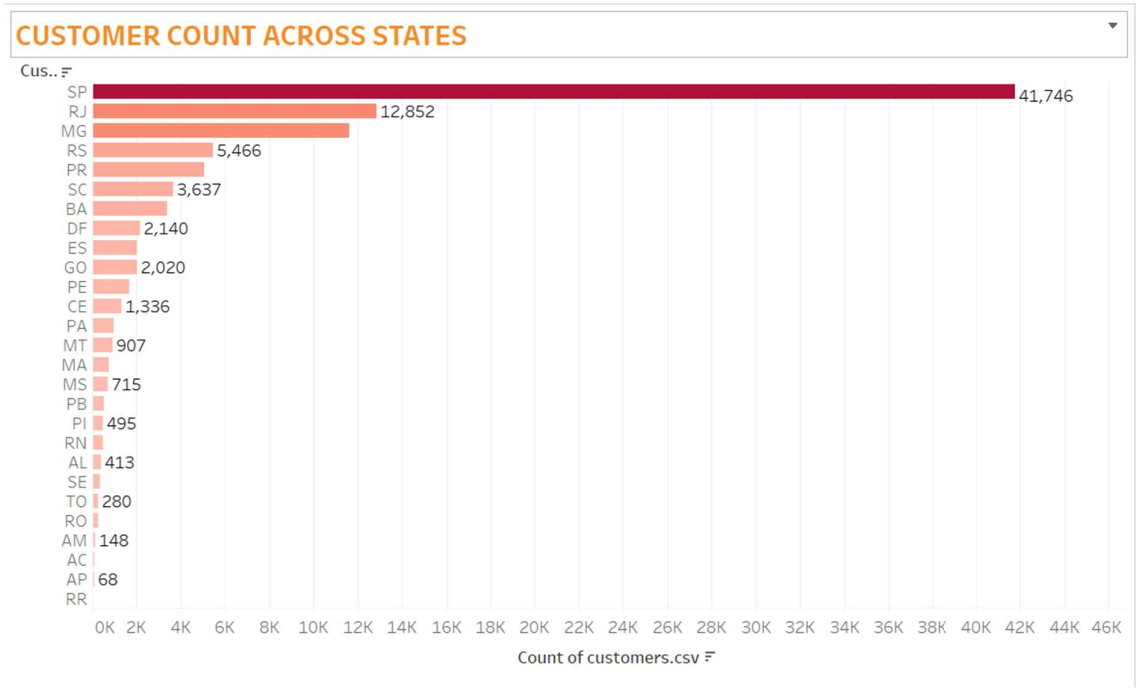
ANS :- Query -

```
select
        customer_state,
        count(customer_id) as customer_count
from `Target_Buisness_case.customers`
group by customer_state
order by count(customer_id) desc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	customer_count
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020



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

Q.4.1) Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table

ANS :- Query -

```
with t1 as
        (
        select
                EXTRACT(year from order_purchase_timestamp) as order_year_2017,
                EXTRACT(month from order_purchase_timestamp) as order_month_2017,
                round(sum(payment_value)) as cost_order1
        from `Target_Buisness_case.orders` o
        join `Target_Buisness_case.payments` p
        on o.order_id = p.order_id
        where EXTRACT(month from order_purchase_timestamp) in(1,2,3,4,5,6,7,8)
                and EXTRACT(year from order_purchase_timestamp) = 2017
        group by 1,2
        order by 2
        ),
t2 as
        (
        select
                extract(year from order_purchase_timestamp) as order_year_2018,
                extract(month from order_purchase_timestamp) as order_month_2018,
                round(sum(payment_value)) as cost_order2
        from `Target_Buisness_case.orders` o
        join `Target_Buisness_case.payments` p
        on o.order_id = p.order_id
        where extract(month from order_purchase_timestamp) in(1,2,3,4,5,6,7,8)
                and extract(year from order_purchase_timestamp) = 2018
        group by 1,2
        order by 2
        )
select
        t1.order_month_2017 as order_month,
        t1.cost_order1 as cost_order_2017,
        t2.cost_order2 as cost_order_2018,
        round((((t2.cost_order2 - t1.cost_order1) / (t1.cost_order1)) * 100))
        as percent_increase_in_cost
from t1
join t2
on t1.order_month_2017 = t2.order_month_2018
order by 1;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_month	cost_order_2017	cost_order_2018	percent_increase_in_cost
1	1	138488.0	1115004.0	705.0
2	2	291908.0	992463.0	240.0
3	3	449864.0	1159652.0	158.0
4	4	417788.0	1160785.0	178.0
5	5	592919.0	1153982.0	95.0
6	6	511276.0	1023880.0	100.0
7	7	592383.0	1066541.0	80.0
8	8	674396.0	1022425.0	52.0

Q.4.2) Mean & Sum of price and freight value by customer state

ANS :-

Query - 1 for Mean & sum of price –

```
select
        c.customer_state,
        round(avg(price)) as Mean_price,
        round(sum(price)) as sum_price
from `Target_Buisness_case.customers` c
  join `Target_Buisness_case.orders` o on c.customer_id = o.customer_id
  join `Target_Buisness_case.order_items` oi on o.order_id = oi.order_id
group by customer_state
order by 2 desc,3 desc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Mean_price	sum_price
1	PB	191.0	115268.0
2	AL	181.0	80315.0
3	AC	174.0	15983.0
4	PA	166.0	178948.0
5	RO	166.0	46141.0
6	AP	164.0	13474.0
7	PI	160.0	86914.0
8	TO	158.0	49622.0
9	RN	157.0	83035.0
10	CE	154.0	227255.0

Query -2 for Mean & sum of freight value –

```
select
        c.customer_state,
        round(avg(freight_value)) as Mean_freight_value,
        round(sum(freight_value)) as sum_freight_value
from `Target_Buisness_case.customers` c
  join `Target_Buisness_case.orders` o on c.customer_id = o.customer_id
  join `Target_Buisness_case.order_items` oi on o.order_id = oi.order_id
group by customer_state
order by 2 desc,3 desc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Mean_freight_value	sum_freight_value
1	PB	43.0	25720.0
2	RR	43.0	2235.0
3	RO	41.0	11417.0
4	AC	40.0	3687.0
5	PI	39.0	21218.0
6	MA	38.0	31524.0
7	SE	37.0	14111.0
8	TO	37.0	11733.0
9	PA	36.0	38699.0
10	RN	36.0	18860.0

- ***5. Analysis on sales, freight and delivery time***

Q.5.1) Calculate days between purchasing, delivering and estimated delivery

ANS :- Query -

```
select
        order_id,
        order_purchase_timestamp as purchase_date,
        order_delivered_customer_date as delivery_date,
        order_estimated_delivery_date as estimated_delivery_date,
        date_diff(order_delivered_customer_date, order_purchase_timestamp, day)
        as day_diff_bet_purchase_and_delivery,
        date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as day_diff_estimated_
        delivery_and_order_delivery
from `Target_Buisness_case.orders`
where order_status = "delivered"
order by day_diff_bet_purchase_and_delivery desc;
```

Query results

SAVE RESULTS EXPLORE DATA

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

Row	order_id	purchase_date	delivery_date	estimated_delivery_date	day_diff_bet_purchase_and_delivery	day_diff_estimated_delivery_and_order_delivery
1	ca07593549f1816d26a572e06dc1eab6	2017-02-21 23:31:27 ...	2017-09-19 14:36:39 ...	2017-03-22 00:00:00 U...	209	-181
2	1b3190b2dfa9d789e1f14c05b647a14a	2018-02-23 14:57:35 ...	2018-09-19 23:24:07 ...	2018-03-15 00:00:00 U...	208	-188
3	440d0d17af552815d15a9e41abe49359	2017-03-07 23:59:51 ...	2017-09-19 15:12:50 ...	2017-04-07 00:00:00 U...	195	-165
4	0f4519c5f1c541ddec9f21b3bddd533a	2017-03-09 13:26:57 ...	2017-09-19 14:38:21 ...	2017-04-11 00:00:00 U...	194	-161
5	285ab9426d6982034523a855f55a885e	2017-03-08 22:47:40 ...	2017-09-19 14:00:04 ...	2017-04-06 00:00:00 U...	194	-166
6	2fb597c2f772eca01b1f5c561bf6cc7b	2017-03-08 18:09:02 ...	2017-09-19 14:33:17 ...	2017-04-17 00:00:00 U...	194	-155
7	47b40429ed8cce3aee9199792275433f	2018-01-03 09:44:01 ...	2018-07-13 20:51:31 ...	2018-01-19 00:00:00 U...	191	-175
8	2fe324febf907e3ea3f2aa9650869fa5	2017-03-13 20:17:10 ...	2017-09-19 17:00:07 ...	2017-04-05 00:00:00 U...	189	-167
9	2d7561026d542c8dbd8f0daeadf67a43	2017-03-15 11:24:27 ...	2017-09-19 14:38:18 ...	2017-04-13 00:00:00 U...	188	-159
10	437222e3fd1b07396f1d9ba8c15fba59	2017-03-16 11:36:00 ...	2017-09-19 16:28:58 ...	2017-04-28 00:00:00 U...	187	-144

Q.5.2) Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

ANS :-

Query – 1 for time to delivery : -

```
select
        order_id,
        order_purchase_timestamp as purchase_date,
        order_delivered_customer_date as delivery_date,
        date_diff(order_delivered_customer_date, order_purchase_timestamp, hour)
        as time_to_delivery_in_hours
from `Target_Buisness_case.orders`
where order_status = "delivered"
order by time_to_delivery_in_hours desc;
```

Query results

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

Row	order_id	purchase_date	delivery_date	time_to_delivery_in_hours
1	ca07593549f1816d26a572e06dc1eab6	2017-02-21 23:31:27 UTC	2017-09-19 14:36:39 UTC	5031
2	1b3190b2dfa9d789e1f14c05b647a14a	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 UTC	5000
3	440d0d17af552815d15a9e41abe49359	2017-03-07 23:59:51 UTC	2017-09-19 15:12:50 UTC	4695
4	2fb597c2f772eca01b1f5c561bf6cc7b	2017-03-08 18:09:02 UTC	2017-09-19 14:33:17 UTC	4676
5	285ab9426d6982034523a855f55a885e	2017-03-08 22:47:40 UTC	2017-09-19 14:00:04 UTC	4671
6	0f4519c5f1c541ddec9f21b3bddd533a	2017-03-09 13:26:57 UTC	2017-09-19 14:38:21 UTC	4657
7	47b40429ed8cce3aee9199792275433f	2018-01-03 09:44:01 UTC	2018-07-13 20:51:31 UTC	4595
8	2fe324febf907e3ea3f2aa9650869fa5	2017-03-13 20:17:10 UTC	2017-09-19 17:00:07 UTC	4556
9	2d7561026d542c8dbd8f0daeadf67a43	2017-03-15 11:24:27 UTC	2017-09-19 14:38:18 UTC	4515
10	c27815f7e3dd0b926b58552628481575	2017-03-15 23:23:17 UTC	2017-09-19 17:14:25 UTC	4505

Query – 2 diff_estimated_delivery : -

```
select
        order_id,
        order_purchase_timestamp as purchase_date,
        order_delivered_customer_date as delivery_date,
        order_estimated_delivery_date as estimated_delivery_date,
        date_diff(order_estimated_delivery_date,order_delivered_customer_date, hour)
        as diff_estimated_delivery_in_hours
from `Target_Buisness_case.orders`
where order_status = "delivered"
order by order_delivered_customer_date desc, order_estimated_delivery_date desc;
```

Query results

SAVE RESULTS

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

Row	order_id	purchase_date	delivery_date	estimated_delivery_date	diff_estimated_delivery_in_hours
1	7e708aed151d6a8601ce8f2eaa712bf4	2018-06	2018-10-17 13:22:46 UTC	2018-07-13 00:00:00 UTC	-2317
2	450cb96c63e1e5b49d34f223f67976d2	2018-05-21 06:48:46 UTC	2018-10-11 16:41:14 UTC	2018-06-27 00:00:00 UTC	-2560
3	b2997e1d7061605e9285496c581d1fbd	2018-07-30 09:08:06 UTC	2018-10-02 00:18:50 UTC	2018-08-14 00:00:00 UTC	-1176
4	a2b4be96b53022618030c17ed437604d	2018-07-22 09:54:03 UTC	2018-09-27 02:24:33 UTC	2018-08-17 00:00:00 UTC	-986
5	7d09831e67caa193da82cfea3bee7aa5	2018-08-05 17:11:44 UTC	2018-09-25 00:47:25 UTC	2018-08-20 00:00:00 UTC	-864
6	f23681a0fffdb8051c674707c7e912ef	2018-07-15 02:11:15 UTC	2018-09-21 23:46:29 UTC	2018-08-06 00:00:00 UTC	-1127
7	1e7d25f611e794f9614dd3e10a8596e7	2018-08-01 19:43:06 UTC	2018-09-21 15:55:02 UTC	2018-08-23 00:00:00 UTC	-711
8	4af2fb154881f350d8696f7f7a7f80d3	2018-07-23 10:22:26 UTC	2018-09-20 16:08:33 UTC	2018-08-13 00:00:00 UTC	-928
9	1b3190b2dfa9d789e1f14c05b647a14a	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 UTC	2018-03-15 00:00:00 UTC	-4535
10	4505acb3759da6b9c7d79a80d29ab3bb	2018-08-06 14:32:27 UTC	2018-09-19 16:44:44 UTC	2018-08-17 00:00:00 UTC	-808

Q.5.3) Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

ANS :- Query -

```
select
        c.customer_state,
        round(avg(oi.freight_value),2) as mean_freight_value,
        round(avg(date_diff(o.order_delivered_customer_date,
        o.order_purchase_timestamp, hour))) as avg_time_to_delivery_in_hours,
        round(avg(date_diff(o.order_estimated_delivery_date,
        o.order_delivered_customer_date, hour)))
        as avg_diff_estimated_delivery_in_hours
from `Target_Buisness_case.order_items` oi
  join `Target_Buisness_case.orders` o
  on oi.order_id = o.order_id
  join `Target_Buisness_case.customers` c
  on o.customer_id = c.customer_id
where order_status = "delivered"
group by c.customer_state
order by 2 asc;
```

Query results

SAVE RESULTS

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

Row	customer_state	mean_freight_value	avg_time_to_delivery_in_hours	avg_diff_estimated_delivery_in_hours
1	SP	15.12	209.0	252.0
2	PR	20.47	286.0	307.0
3	MG	20.63	287.0	303.0
4	RJ	20.91	363.0	271.0
5	DF	21.07	311.0	275.0
6	SC	21.51	359.0	260.0
7	RS	21.61	364.0	322.0
8	ES	22.03	375.0	238.0
9	GO	22.56	369.0	278.0
10	MS	23.35	372.0	252.0

Q.5.4) Sort the data to get the following:

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

ANS :-

Query_1 - for Highest average freight value –

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

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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_2 - for Lowest average freight value -

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

Query results

JOB INFORMATION | RESULTS | JSON

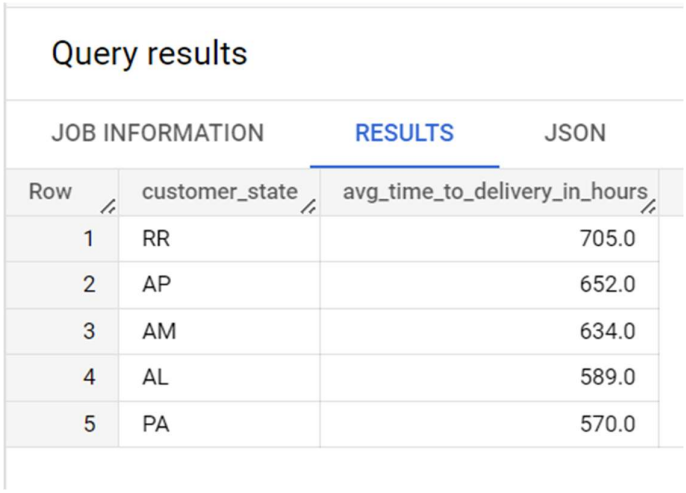
Row	customer_state	avg_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

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

ANS :-

Query_1 - for Highest average time to delivery -

```
select
        customer_state,
        round(avg(date_diff(o.order_delivered_customer_date,
        o.order_purchase_timestamp, hour))) as avg_time_to_delivery_in_hours
from `Target_Buisness_case.customers` c
join `Target_Buisness_case.orders` o
on c.customer_id = o.customer_id
where order_status = "delivered"
group by customer_state
order by avg_time_to_delivery_in_hours desc
limit 5;
```



Query results

JOB INFORMATION | RESULTS | JSON

Row	customer_state	avg_time_to_delivery_in_hours
1	RR	705.0
2	AP	652.0
3	AM	634.0
4	AL	589.0
5	PA	570.0

Query_2 - for Lowest average time to delivery -

```
select
        customer_state,
        round(avg(date_diff(o.order_delivered_customer_date,
        o.order_purchase_timestamp, hour))) as avg_time_to_delivery_in_hours
from `Target_Buisness_case.customers` c
join `Target_Buisness_case.orders` o
on c.customer_id = o.customer_id
where order_status = "delivered"
group by customer_state
order by avg_time_to_delivery_in_hours asc
limit 5;
```

Query results

JOB INFORMATION | RESULTS | JSON

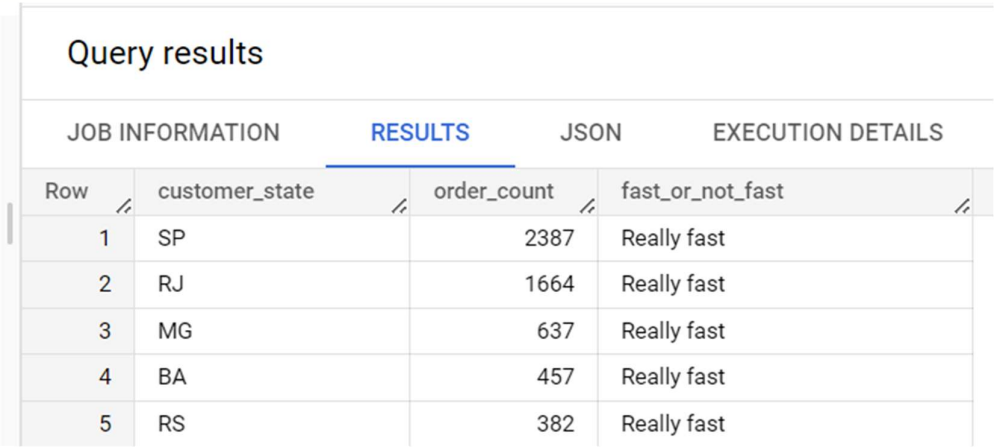
Row	customer_state	avg_time_to_delivery_in_hours
1	SP	210.0
2	PR	287.0
3	MG	288.0
4	DF	311.0
5	SC	358.0

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

ANS :-

Query_1 Top 5 states whose delivery is really fast

```
select
        customer_state,
        count(order_id) as order_count,
        if(order_delivered_customer_date > order_estimated_delivery_date ,
        "Really fast", "Not so fast") as fast_or_not_fast
from `Target_Buisness_case.customers` c
join `Target_Buisness_case.orders` o
on c.customer_id = o.customer_id
where order_status = "delivered" and
if(order_delivered_customer_date > order_estimated_delivery_date ,
"Really fast", "Not so fast") = "Really fast"
group by customer_state, 3
order by count(order_id) desc
limit 5;
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	order_count	fast_or_not_fast
1	SP	2387	Really fast
2	RJ	1664	Really fast
3	MG	637	Really fast
4	BA	457	Really fast
5	RS	382	Really fast

Query_2 Top 5 states whose delivery not so fast

```
select
        customer_state,
        count(order_id) as order_count,
        if(order_delivered_customer_date > order_estimated_delivery_date ,
        "Really fast", "Not so fast") as fast_or_not_fast
from `Target_Buisness_case.customers` c
join `Target_Buisness_case.orders` o
on c.customer_id = o.customer_id
where order_status = "delivered" and
if(order_delivered_customer_date > order_estimated_delivery_date ,
"Really fast", "Not so fast") = "Not so fast"
group by customer_state, 3
order by count(order_id) desc
limit 5;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	order_count	fast_or_not_fast
1	SP	38114	Not so fast
2	MG	10717	Not so fast
3	RJ	10686	Not so fast
4	RS	4963	Not so fast
5	PR	4677	Not so fast

- ***6. Payment type analysis:***

Q.6.1) Month over Month count of orders for different payment types

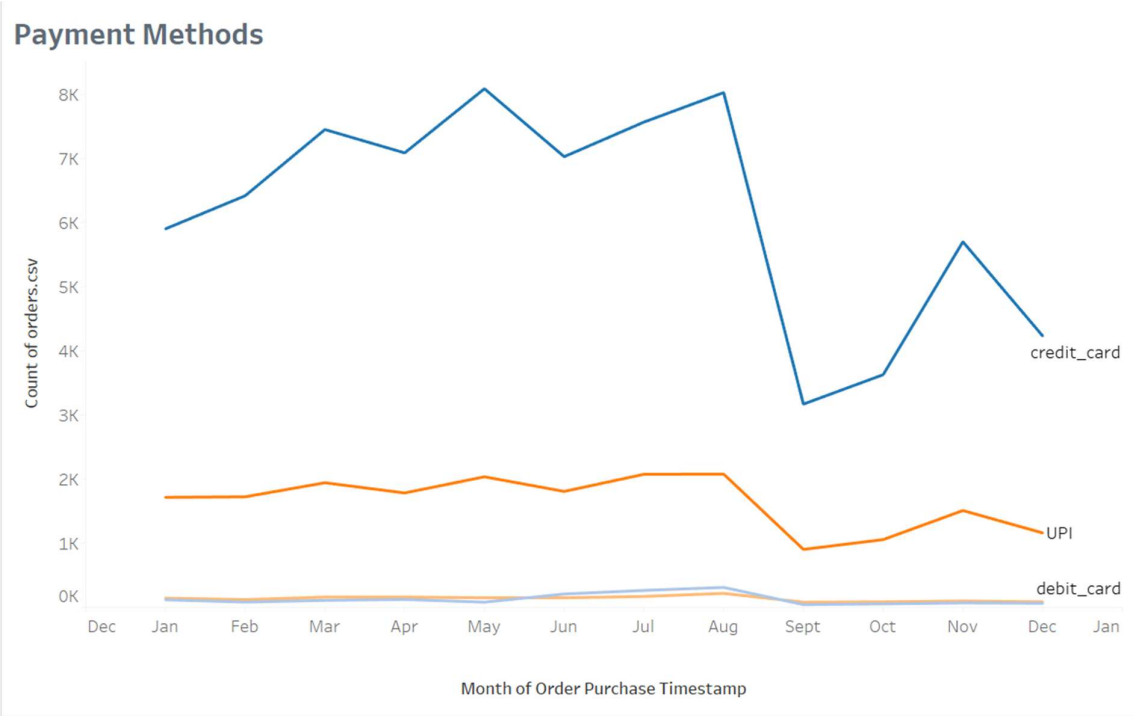
ANS :- Query -

```
select
EXTRACT(month from order_purchase_timestamp) as order_month,
payment_type,
count(o.order_id) as order_count
from `Target_Buisness_case.orders` o
join `Target_Buisness_case.payments` p
on o.order_id = p.order_id
group by order_month, payment_type
order by order_month asc, count(o.order_id) desc;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_month	payment_type	order_count
1	1	credit_card	6103
2	1	UPI	1715
3	1	voucher	477
4	1	debit_card	118
5	2	credit_card	6609
6	2	UPI	1723
7	2	voucher	424
8	2	debit_card	82
9	3	credit_card	7707
10	3	UPI	1942



Q.6.2) Count of orders based on the no. of payment installments

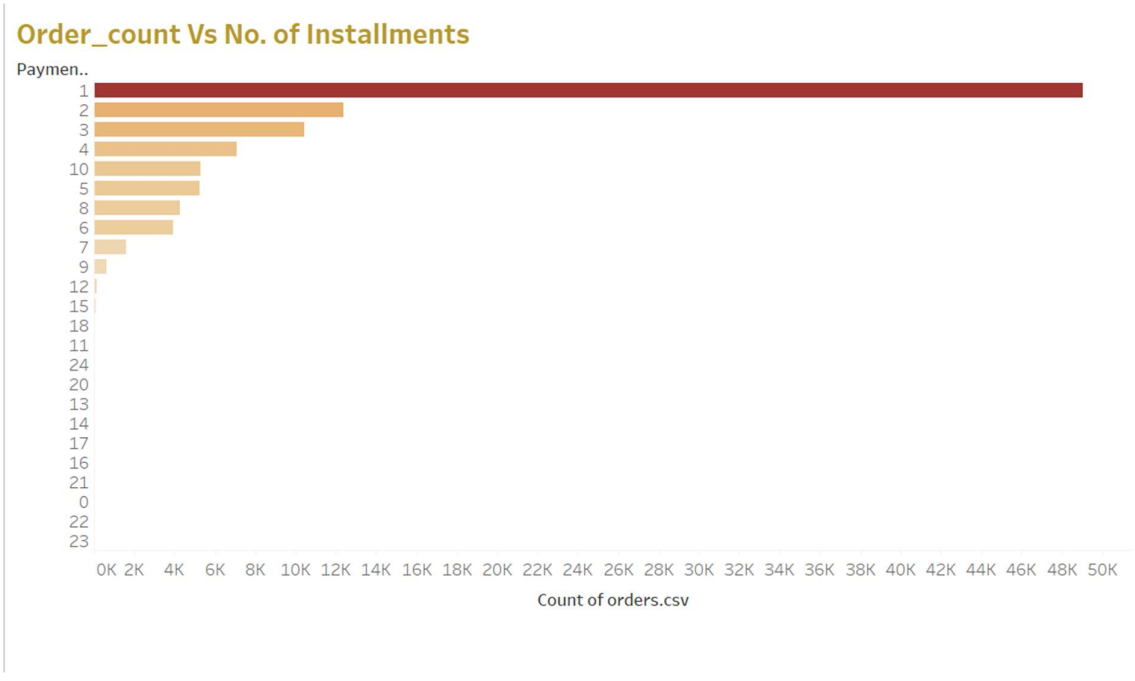
ANS :- Query -

```
select
        payment_installments,
        count(order_id) as order_count
from `Target_Buisness_case.payments`
group by payment_installments
order by count(order_id) desc;
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	payment_installments	order_count
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



****Actionable Insights:-**

- As there is growing trend of sales over the months Target should add more and more products so that sale will go on increasing in future
- There is not much evolution of Target in states other than SP.

****Recommendations:-**

- There should be less time difference between estimated and actual delivery of order so that customer will get product in time. and due to this customer will buy more products as they will have intuition that they will get product in required time period.
- As from the analysis of payment type, credit card is mostly used by customers. so, Target can provide attractive offers on credit card in terms of gift vouchers and cashback.
- Target should provide some good offers to attract crowd from the states were sale is very low.

References

1. BigQuery
2. Tableau