Q1. Business Case: Target SQL

1. Usual Exploratory Analysis:

1.1 Datatype of all columns in "customers" table

Query:

```
select column_name,data_type from
`businesscase24.Target.INFORMATION_SCHEMA.COLUMNS` where table_name="customers"
```

Output:

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

Insights:

Have the personal unique id of customer besides customer consumer id.

1.2 Get the time range between which orders placed

Query:

```
SELECT min(order_purchase_timestamp) as `orders_startdate`,
     max(order_purchase_timestamp) as  `orders_enddate`
      FROM `Target.orders`
```

Output:

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

Insights:

The orders present in the data are placed between september 4,2016 and october 17,2018

1.3 Count the cities and states of customers who ordered during the given period

Query:

```
select count(distinct customer_city) as `Number of Cities`,
     count(distinct customer_state) as `Number of States`
     from `Target.customers` c join `Target.orders` o on
     c.customer_id=o.customer_id
```

Output:

Row	Number of Cities	Number of States
1	4119	27

Insights:

Orders have been placed from 27 states and 4119 different cities

2. In-depth exploration:

2.1 Is there a growing trend in the no.of orders placed over the past years

Query:

```
with cte as
(select
  extract(year from order_purchase_timestamp) as `year`,
  extract(month from order_purchase_timestamp) as `month`,
  count(distinct order_id) as `totalordersplaced`
```

```
 from `Target.orders`
 group by 1,2
order by 1,2
)


select *,
     lag(totalorderspLaced) over(order by year,month)as `prev_monthorders`,
     (
        (100*(totalorderspLaced - lag(totalorderspLaced) over(order by
 year,month)) )/ lag(totalorderspLaced) over(order by year,month)

     ) as `percentage`

     from cte

     order by 1,2
```

Output:

Row	year	month	totalorderspLaced	prev_monthorders	percentage
1	2016	9	4	null	null
2	2016	10	324	4	8000.0
3	2016	12	1	324	-99.6913580246...
4	2017	1	800	1	79900.0
5	2017	2	1780	800	122.5
6	2017	3	2682	1780	50.67415730337...
7	2017	4	2404	2682	-10.3653989560...
8	2017	5	3700	2404	53.91014975041...
9	2017	6	3245	3700	-12.2972972972...
10	2017	7	4026	3245	24.06779661016...

Insights:

Yes, it's a clear growing trend of orders when we compare between totalcountoforders related to years from 2016 to 2017 and from 2017 to 2018..while, it might be a rising curve over all but it has it's ups and downs in months and there is a huge increase jan 2017 and Nov 2017 when compared to other months. And after Nov 2017, there is a little less growth in orders comparitively and after nov 2018, the count of orders dropped drastically. The reasons can be investigated further

Recommendations:

Need to check any competitors or any potential/technical reasons as there is a major downfall of orders at the end and maintain extra stocks to be accessible during peak order months and also good time for sales and discounts

2.2 Can we see some kind of monthly seasonality in terms of the number of orders being placed

Query:

```
with cte as
 (select
    extract(year from order_purchase_timestamp) as `year`,
    extract(month from order_purchase_timestamp) as `month`,
    count(distinct order_id) as `totalorderspLaced`

 from `Target.orders`
 group by 1,2
 order by 1,2
```

```
)

select *,
     lag(totalordersplaced) over(order by year,month)as `prev_monthorders`,
     (
        (100*(totalordersplaced - lag(totalordersplaced) over(order by
 year,month)) )/ lag(totalordersplaced) over(order by year,month)

     ) as `percentage`

     from cte

     order by 5 desc
```

Output:

Row	year	month	totalordersplaced	prev_monthorders	percentage
1	2017	1	800	1	79900.0
2	2016	10	324	4	8000.0
3	2017	2	1780	800	122.5
4	2017	11	7544	4631	62.90218095443...
5	2017	5	3700	2404	53.91014975041...
6	2017	3	2682	1780	50.67415730337...
7	2018	1	7269	5673	28.13326282390...
8	2017	7	4026	3245	24.06779661016...
9	2017	10	4631	4285	8.074679113185...
10	2017	8	4331	4026	7.575757575757...

Insights:

Main peak orders are in November, January and in any one of summer month like March,May but there is a drastic fall of orders from August 2018- September 2018

Recommendations:

Need to check any competitors or any potential/technical reasons as there is a major downfall of orders at the end and maintain extra stocks to be accessible during peak order months and also good time for sales and discounts

2.3 During what time of the day, do the brazilian customers mostly place their orders?(Dawn,Morning,Afternoon,Night)

Query:

```
select
(count(order_id)) as `orders_purchased`,
(
  CASE
    WHEN  extract(hour from order_purchase_timestamp) between 0 and 6
     then   'Dawn'
    WHEN  extract(hour from order_purchase_timestamp) between 7 and 12
     then 'Morning'
    when  extract(hour from order_purchase_timestamp) between 13 and 18
     then 'Afternoon'
    else 'Night'
  END
) as `time_of_day`
from
```

```
`Target.orders`
group by 2
order by 1 desc
```

Output:

Row	orders_purchased	time_of_day
1	38135	Afternoon
2	28331	Night
3	27733	Morning
4	5242	Dawn

Insights:
As per UTC time, the most number of orders are placed in afternoon followed by night

Recommendations:
Since most orders are placed in afternoon times, good to check there will be no concurrency/technical issues while placing the order and multiple instances bandwidth ready and getting updated accordingly

3. Evolution of E-Commerce Orders in the Brazil region:

3.1 Get the month on month no.of orders placed in each state

Query:

```
select
 extract(year from order_purchase_timestamp) as `year`,
 extract(month from order_purchase_timestamp) as `month`,
 customer_state,
 count(order_id) as `count`
 from `Target.customers` c join `Target.orders` o on
c.customer_id=o.customer_id
 group by 1,2,3
 order by 4 desc
```

Output:

Row	year	month	customer_state	count
1	2018	8	SP	3253
2	2018	5	SP	3207
3	2018	4	SP	3059
4	2018	1	SP	3052
5	2018	3	SP	3037
6	2017	11	SP	3012
7	2018	7	SP	2777
8	2018	6	SP	2773
9	2018	2	SP	2703
10	2017	12	SP	2357

Insights:
The more number of orders are clearly coming from SP state most of the months.

Recommendations:

Since most of the orders are from SP state, good to have extra stock places to decrease the delivery time and increase the ease for customers and the marketing amount of that state can be decreased to some extent and can use that in states of less orders like RR,AP..

3.2 How are customers distributed across all states

Query:

```
select
 customer_state,
 count(o.customer_id) as `count`
 from `Target.customers` c join `Target.orders` o on
c.customer_id=o.customer_id
 group by 1
 order by 2 desc
```

Output:

Row	customer_state	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

Insights:

The state 'SP' has more number of active customers and `RR` has least

Recommendations:

Marketing techniques about the quality of products and ease of buying from the target retailers can be improved in RR to attract more customers so orders.

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

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

Query:

```
with cte as
(select
 extract(year from order_purchase_timestamp) as `year`,
(
```

```
   sum(payment_value)

) as `cost_of_orders`
from `Target.orders` o join `Target.payments` p on o.order_id = p.order_id
where extract(month from order_purchase_timestamp) between 1 and 8
group by 1
order by 1)

select *,
(
  round(100 *( cost_of_orders - lag(cost_of_orders) over(order by year) )/
lag(cost_of_orders) over(order by year))
) as `percentage`

from cte
```

Output:

Row	year	cost_of_orders	percentage
1	2017	3669022.119999...	null
2	2018	8694733.839999...	137.0

Insights:
There is a 137% of increase in cost of orders from year 2017 -18 when only months from january to august are included

4.2 Calculate the Total & Average value of order price for each state.

Query:

```
select
  customer_state,
   sum(price) as `totalprice`,
   avg(price) as  `avgprice`
   from  `Target.customers` c join `Target.orders` o on c.customer_id=o.customer_id
        join `Target.order_items` i on o.order_id = i.order_id
   group by customer_state
   order by 2 desc,3 desc
```

Output:

Row	customer_state	totalprice	avgprice
1	SP	5202955.050001...	109.6536291597...
2	RJ	1824092.669999...	125.1178180945...
3	MG	1585308.029999...	120.7485741488...
4	RS	750304.0200000...	120.3374530874...
5	PR	683083.7600000...	119.0041393728...
6	SC	520553.3400000...	124.6535775862...
7	BA	511349.9900000...	134.6012082126...
8	DF	302603.9399999...	125.7705486284...
9	GO	294591.9499999...	126.2717316759...
10	ES	275037.3099999...	121.9137012411...

Insights:
Highest total price of orders are recorded in `SP` and lowest total price of orders are recorded in `RR` from the data given between 2016-18

Recommendations:
The potential reasons to increase orders in the low total price states can be investigated further while extra stock can be placed to decrease delivery time for customers

4.3 Calculate the Total & Average value of order freight for each state.

Query:

```
select
  customer_state,
   sum(freight_value) as `totalfreight_value`,
   avg(freight_value) as  `avgfreight_value`
   from  `Target.customers` c join `Target.orders` o on c.customer_id=o.customer_id
        join `Target.order_items` i on o.order_id = i.order_id
   group by customer_state
   order by 2 desc,3 desc
```

Output:

Row	customer_state	totalfreight_value	avgfreight_value
1	SP	718723.0699999...	15.14727539041...
2	RJ	305589.3100000...	20.96092393168...
3	MG	270853.4600000...	20.63016680630...
4	RS	135522.7400000...	21.73580433039...
5	PR	117851.6800000...	20.53165156794...
6	BA	100156.6799999...	26.36395893656...
7	SC	89660.26000000...	21.47036877394...
8	PE	59449.65999999...	32.91786267995...
9	GO	53114.97999999...	22.76681525932...
10	DF	50625.49999999...	21.04135494596...

Insights:
The total freight value is more in the states of higher price of orders I.e., SP,RJ..

Recommendations:
In the places of getting bulk orders, the freight charges can check and can try to decrease if possible by proper planning of shipping and delivery of orders.

5. Analysis based on sales, freight and delivery time

5.1 Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

Query:

```
select
    order_id,
    o.customer_id,
    c.customer_state,
    DATE_DIFF(
        order_delivered_customer_date, order_purchase_timestamp, day
            ) as `time_to_deliver`,
    DATE_DIFF(
        order_delivered_customer_date,order_estimated_delivery_date,day
            ) as `diff_estimated_delivery`
```

```
from `Target.orders` o join `Target.customers` c on o.customer_id=c.customer_id
where order_status = "delivered"
order by 4 desc,5 desc
```

Output:

Row	order_id	customer_id	customer_state	time_to_deliver	diff_estimated_deliv
1	ca07593549f1816d26a572e06...	75683a92331068e2d281b11a...	ES	209	181
2	1b3190b2dfa9d789e1f14c05b...	d306426abe5fca15e54b645e4...	RJ	208	188
3	440d0d17af552815d15a9e41a...	7815125148cfa1e8c7fee1ff79...	PA	195	165
4	285ab9426d6982034523a855f...	9cf2c3fa2632cee748e1a59ca9...	SE	194	166
5	0f4519c5f1c541ddec9f21b3bd...	1a8a4a30dc296976717f44e78...	PI	194	161
6	2fb597c2f772eca01b1f5c561b...	217906bc11a32c1e470eb7e08...	PI	194	155
7	47b40429ed8cce3aee9199792...	cb2caaaead400c97350c37a3f...	SP	191	175
8	2fe324febf907e3ea3f2aa9650...	65b14237885b3972ebec28c0f...	SP	189	167
9	2d7561026d542c8dbd8f0daea...	8199345f57c6d1cbe9701f924...	SE	188	159
10	c27815f7e3dd0b926b5855262...	f85e9ec0719b16dc4dd0edd43...	MG	187	162

Insights/Recommendations:

The longest delivery time(209 d) is taken for orders showing above which are for states ES,RJ,PA,SE which is far more time to estimated date.. need to correct/check the estimation date and see why the estimation is that bad besides the investigation of reasons for that delay. The shortest delivery is within a day for few orders to SP,RJ… here it is delivered before estimated which sounds good but it is better to keep the estimatedate accurate or atleast close to the actual delivery for proper knowledge and availability of customer.

5.2 Find out the top 5 states with the highest & lowest average freight value.

Query:

```
(select
 customer_state,
 avg(freight_value) as avg
   from  `Target.customers` c join `Target.orders` o on c.customer_id =
 o.customer_id
   join `Target.order_items`i on o.order_id=i.order_id
  group by customer_state
 order by 2 desc
  Limit 5)

 union all

(select
  customer_state,
  avg(freight_value) as avg
   from  `Target.customers` c join `Target.orders` o on c.customer_id =
  o.customer_id
   join `Target.order_items`i on o.order_id=i.order_id
 group by customer_state
 order by 2 asc
 Limit 5)

order by avg asc
```

Output:

Row	customer_state	avg
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...
6	PI	39.14797047970...
7	AC	40.07336956521...
8	RO	41.06971223021...
9	PB	42.72380398671...
10	RR	42.98442307692...

Insights:
The lowest average freight value states are SP,PR,MG,RJ,DF
The highest average freight value states are RR,PB,RO,AC,PI

5.3 Find out the top 5 states with the highest & lowest average delivery time.

Query:

```
(select
   customer_state,
   avg(
      DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day)
      ) as `deliverytime`

     from  `Target.customers` c join `Target.orders` o on c.customer_id =
  o.customer_id
  group by 1
  order by 2 desc
  Limit 5)

union
 all

(select
   customer_state,
   avg(
      DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day)
      ) as `deliverytime`

     from  `Target.customers` c join `Target.orders` o on c.customer_id =
  o.customer_id

 group by customer_state
 order by 2 asc
 Limit 5)

order by deliverytime asc
```

Output:

Row	customer_state	deliverytime
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...
6	PA	23.31606765327...
7	AL	24.04030226700...
8	AM	25.98620689655...
9	AP	26.73134328358...
10	RR	28.97560975609...

Insights:

The lowest average delivery value(high speed of delivery) states are SP,PR,MG,DF,SC
The highest average delivery value(low speed of delivery) states are RR,AP,AM,AL,PA

5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery

Query:

```
select
customer_state,
avg (
    DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp,day)
) as `estimated_delivery`,
 avg(
   DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day)
 ) as `actual_delivery`,
(avg (
    DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp,day)
)- avg(
   DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,day)
)
)
as `deliveryspeed`

from  `Target.customers` c join `Target.orders` o on c.customer_id = o.customer_id
group by 1
order by 4 desc
Limit 5
```

Output:

Row	customer_state	estimated_delivery	actual_delivery	deliveryspeed
1	AC	40.76543209876...	20.63750000000...	20.12793209876...
2	RO	38.40711462450...	18.91358024691...	19.49353437759...
3	AP	45.70588235294...	26.73134328358...	18.97453906935...
4	AM	44.75675675675...	25.98620689655...	18.77054986020...
5	RR	46.17391304347...	28.97560975609...	17.19830328738...

Insights:
The delivery is more quicker than estimated in AC,RO,AP,AM,RR states

6.Analysis based on the payments:

6.1 Find the month on month no. of orders placed using different payment types.

Query:

```
select
payment_type,
extract(year from order_purchase_timestamp) as `year`,
extract(month from order_purchase_timestamp) as `month`,
count(distinct p.order_id) as `totalorders`
from `Target.orders` o join `Target.payments` p on o.order_id=p.order_id
group by 1,2,3
order by 2,3
```

Output:

Row	payment_type	year	month	totalorders
1	credit_card	2016	9	3
2	credit_card	2016	10	253
3	UPI	2016	10	63
4	voucher	2016	10	11
5	debit_card	2016	10	2
6	credit_card	2016	12	1
7	credit_card	2017	1	582
8	UPI	2017	1	197
9	voucher	2017	1	33
10	debit_card	2017	1	9

Insights:
From the results, it can be noted that creditcard payment type have been used by most of customers followed by UPIs, debitcard and vouchers.

6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.

Query:

```
select
payment_installments,
count(distinct p.order_id) as `totalorders`
from `Target.orders` o join `Target.payments` p on o.order_id=p.order_id
where payment_installments >=1 and payment_value>0
group by payment_installments
order by 1
```

Output:

Row	payment_installment	totalorders
1	1	49057
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315

Insights:
More payments of orders done in one installment.