Business Case: Target SQL

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

A.Data type of columns in a table

select column_name,data_type from information_schema.columns

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

Data_type of columns in table

B.Time period for which the data is given

SELECT min(order_purchase_timestamp),max(order_purchase_timestamp) FROM `dsmlsql.target.orders`

Query results

SAVE RESULTS EXPLORE DATA

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

Row	f0_	f1_
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

SELECT c.customer_id,c.customer_state,c.customer_city

FROM `dsmlsql.target.customers` c JOIN `target.orders` o ON c.customer_id = o.customer_id

WHERE o.order_id IS NOT NULL

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?***

```
WITH month_year as

(SELECT EXTRACT(year FROM order_purchase_timestamp) as year ,EXTRACT(month FROM order_purchase_timestamp) as month

FROM `dsmlsql.target.orders`)

SELECT year,month,count(month) as order_count

FROM month_year GROUP BY year,month

ORDER BY year,month,count(month)
```

JOB INFORMATION | RESULTS | JSON | EXECUTIO

Row	year	month	order_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544
15	2017	12	5673
16	2018	1	7269
17	2018	2	6728
18	2018	3	7211
19	2018	4	6939

```
WITH month_year as
(SELECT EXTRACT(year FROM order_purchase_timestamp) as year ,EXTRACT(month FROM order_purchase_timestamp) as month
FROM `dsmlsql.target.orders`),
purchase_trend as
(SELECT year,month,count(month) as order_count
FROM month_year GROUP BY year,month
ORDER BY year,month,count(month)),
rank_by_order as
(SELECT *,DENSE_RANK() OVER (partition by year ORDER BY year,order_count DESC ) as order_rank
FROM purchase_trend)
SELECT * FROM rank_by_order WHERE order_rank = 1
```

Row	year	month	order_count	order_rank
1	2017	11	7544	1
2	2016	10	324	1
3	2018	1	7269	1

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

```
WITH tend_to_buy as
(SELECT EXTRACT(hour FROM order_purchase_timestamp) as peak_hours
FROM `dsmlsql.target.orders`),
```

```
buy_time as

(SELECT peak_hours,

CASE WHEN peak_hours BETWEEN 2 AND 6 THEN 'DAWN'

WHEN peak_hours BETWEEN 6 AND 12 THEN 'MORNING'

WHEN peak_hours BETWEEN 12 AND 17 THEN 'AFTERNOON'

WHEN peak_hours BETWEEN 17 AND 21 THEN 'EVENING'

ELSE'NIGHT' END as trending_time,count(peak_hours) as order_count

FROM tend_to_buy

GROUP BY peak_hours

ORDER BY count(peak_hours) DESC )

SELECT trending_time,sum(order_count) as total_order FROM buy_time GROUP BY
trending_time ORDER BY total_order DESC
```

Row	trending_time	total_order
1	AFTERNOON	32366
2	MORNING	27733
3	EVENING	24161
4	NIGHT	13503
5	DAWN	1678

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

A.Get month on month orders by states

```
WITH month_year as

(SELECT customer_id,EXTRACT(year FROM order_purchase_timestamp) as
year ,EXTRACT(month FROM order_purchase_timestamp) as month

FROM `dsmlsql.target.orders`)

SELECT c.customer_state,my.year,my.month,count(c.customer_state) as customer_count
```

FROM `dsmlsql.target.customers` c

JOIN month_year my ON c.customer_id = my.customer_id

GROUP BY c.customer_state,my.year,my.month ORDER BY c.customer_state,my.year,my.month

Row	customer_state	year	month	custoemr_count
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6
11	AC	2017	11	5
12	AC	2017	12	5

B.Distribution of customers across the states in Brazil

WITH month_year as

(SELECT customer_id,EXTRACT(year FROM order_purchase_timestamp) as year ,EXTRACT(month FROM order_purchase_timestamp) as month

FROM `dsmlsql.target.orders`)

SELECT c.customer_state,count(c.customer_state) as customer_count

FROM `dsmlsql.target.customers` c

JOIN month_year my ON c.customer_id = my.customer_id

GROUP BY c.customer_state ORDER BY count(c.customer_state) DESC

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
11	PE	1652
12	CE	1336

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

A.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

```
WITH month_year as

(SELECT customer_id,EXTRACT(year FROM order_purchase_timestamp) as
year ,EXTRACT(month FROM order_purchase_timestamp) as month

FROM `dsmlsql.target.orders`),

customer_order as

(SELECT * FROM `target.orders` o

JOIN month_year my ON o.customer_id = my.customer_id  ),

revenue_growth as

(SELECT co.year,co.month,count(co.order_id) order_count,round(sum(payment_value)) as
cost FROM customer_order co JOIN `target.payments` p ON co.order_id = p.order_id

GROUP BY co.year,co.month ORDER BY co.year,co.month)

SELECT year,month,cost,cost - LAG(cost,1) OVER (ORDER BY year,month) as revenue

,((cost - LAG(cost,1) OVER (ORDER BY year,month) )/LAG(cost,1) OVER (ORDER BY
year,month) ) * 100 as revene_percent
```

Row	year	month	cost	revenue	revene_percent
1	2017	1	138488.0	*null*	*null*
2	2017	2	291908.0	153420.0	110.782161...
3	2017	3	449864.0	157956.0	54.1115693...
4	2017	4	417788.0	-32076.0	-7.1301548...
5	2017	5	592919.0	175131.0	41.9186285...
6	2017	6	511276.0	-81643.0	-13.769671...
7	2017	7	592383.0	81107.0	15.8636431...
8	2017	8	674396.0	82013.0	13.8445904...
9	2018	1	1115004.0	440608.0	65.3337208...
10	2018	2	992463.0	-122541.0	-10.990184...
11	2018	3	1159652.0	167189.0	16.8458673...
12	2018	4	1160785.0	1133.0	0.09770172...

FROM revenue_growth WHERE year BETWEEN 2017 AND 2018 AND month BETWEEN 1 AND 8 ORDER BY year,month

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

```
WITH customer_orderid as

(SELECT * FROM `target.customers` c JOIN `target.orders` o ON c.customer_id = o.customer_id),

customer_orderid_orderitems as

(SELECT * FROM customer_orderid co JOIN `target.order_items` oi ON co.order_id = oi.order_id)

SELECT customer_state,round(sum(price)) as total_price,avg(price) as mean_price,

round(sum(freight_value)) as total_delicharge, avg(freight_value) as mean_delivery

FROM customer_orderid_orderitems

GROUP BY customer_state
```

Row	customer_state	total_price	mean_price	total_delicharge	mean_delivery
1	MT	156454.0	148.297184...	29715.0	28.1662843...
2	MA	119648.0	145.204150...	31524.0	38.2570024...
3	AL	80315.0	180.889211...	15915.0	35.8436711...
4	SP	5202955.0	109.653629...	718723.0	15.1472753...
5	MG	1585308.0	120.748574...	270853.0	20.6301668...
6	PE	262788.0	145.508322...	59450.0	32.9178626...
7	RJ	1824093.0	125.117818...	305589.0	20.9609239...
8	DF	302604.0	125.770548...	50625.0	21.0413549...
9	RS	750304.0	120.337453...	135523.0	21.7358043...
10	SE	58921.0	153.041168...	14111.0	36.6531688...
11	PR	683084.0	119.004139...	117852.0	20.5316515...
12	PA	178948.0	165.692416...	38699.0	35.8326851...

5. Analysis on sales, freight and delivery time

1. ***Calculate days between purchasing, delivering and estimated delivery***
2. ***Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:***

```
SELECT order_id,order_purchase_timestamp order_date,

order_estimated_delivery_date estimated_delivery,

order_delivered_customer_date delivery_date,

abs(date_diff(order_purchase_timestamp,order_estimated_delivery_date,day)) as actual_est_delivery,

abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as diff_estimated_delivery,

abs(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) as time_to_delivery

FROM `dsmlsql.target.orders`

WHERE order_status = 'delivered'
```

Row	order_id	order_date	estimat	delivery_d	actual_est_delivery	diff_estimated_delivery	time_to_delivery
1	1a0b31f08d...	2017-04...	20...	2017-...	36	29	6
2	cec8f5f7a1...	2017-03...	20...	2017-...	61	40	20
3	58527ee47...	2017-03...	20...	2017-...	58	48	10
4	10ed5499d...	2017-03...	20...	2017-...	57	29	28
5	818996ea2...	2018-08...	20...	2018-...	44	35	9
6	d195cac9c...	2018-08...	20...	2018-...	52	41	10
7	64eeb35d3...	2018-08...	20...	2018-...	48	41	6
8	2691ae869f...	2018-08...	20...	2018-...	42	35	6
9	1cd147d1c...	2018-08...	20...	2018-...	44	35	8
10	b36d2e6b1...	2018-08...	20...	2018-...	55	42	12
11	88ab6b0ed...	2018-08...	20...	2018-...	51	35	16
12	c15790c44...	2018-08...	20...	2018-...	44	34	10

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

```
WITH order_stat as

(SELECT order_id,customer_id,order_purchase_timestamp order_date,

order_estimated_delivery_date estimated_delivery,

order_delivered_customer_date delivery_date,

abs(date_diff(order_purchase_timestamp,order_estimated_delivery_date,day)) as actual_est_delivery,

abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as diff_estimated_delivery,

abs(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) as time_to_delivery

FROM `dsmlsql.target.orders`

WHERE order_status = 'delivered'),

orderitems_stat as

(SELECT * FROM `target.order_items` oi JOIN order_stat o ON oi.order_id = o.order_id)

SELECT c.customer_state,count(c.customer_state) as customer_count,

round(avg(diff_estimated_delivery)) avg_est_del_days,

round(avg(time_to_delivery)) avg_time_del_days,
```

round(avg(freight_value)) as freight_amount

FROM `target.customers` c

JOIN orderitems_stat os ON c.customer_id = os.customer_id

GROUP BY c.customer_state

Row	customer_state	customer_count	avg_est_del_day	avg_time_del_da	freight_amount
1	GO	2277	13.0	15.0	23.0
2	SP	46448	11.0	8.0	15.0
3	RS	6134	14.0	15.0	22.0
4	BA	3683	13.0	19.0	26.0
5	MG	12916	13.0	12.0	21.0
6	MT	1037	15.0	18.0	28.0
7	RJ	14143	14.0	15.0	21.0
8	SC	4097	12.0	15.0	22.0
9	SE	375	14.0	21.0	37.0
10	PE	1746	15.0	18.0	33.0
11	TO	310	13.0	17.0	37.0
12	CE	1426	14.0	21.0	33.0

4.Sort the data to get the following:

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

order_estimated_delivery_date estimated_delivery,

order_delivered_customer_date delivery_date,

abs(date_diff(order_purchase_timestamp,order_estimated_delivery_date,day)) as actual_est_delivery,

abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as diff_estimated_delivery,

abs(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) as time_to_delivery

FROM `dsmlsql.target.orders`

WHERE order_status = 'delivered'),

```
orderitems_stat as
(SELECT * FROM `target.order_items` oi JOIN order_stat o ON oi.order_id = o.order_id),
cust_ord_items as
(SELECT c.customer_state,count(c.customer_state) as customer_count,
round(avg(diff_estimated_delivery)) avg_est_del_days,
round(avg(time_to_delivery)) avg_time_del_days,
round(avg(freight_value)) as freight_amount
FROM `target.customers` c
JOIN orderitems_stat os ON c.customer_id = os.customer_id
GROUP BY c.customer_state)
SELECT * FROM cust_ord_items ORDER BY freight_amount DESC LIMIT 5
```

Row	customer_state	customer_count	avg_est_del_day	avg_time_del_da	freight_amount
1	RR	46	25.0	28.0	43.0
2	PB	586	14.0	20.0	43.0
3	RO	273	20.0	19.0	41.0
4	AC	91	21.0	20.0	40.0
5	PI	523	14.0	19.0	39.0

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

```
order_estimated_delivery_date estimated_delivery,
order_delivered_customer_date delivery_date,
abs(date_diff(order_purchase_timestamp,order_estimated_delivery_date,day)) as actual_est_delivery,
abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as diff_estimated_delivery,
abs(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) as time_to_delivery
```

```
FROM `dsmlsql.target.orders`
WHERE order_status = 'delivered'),
orderitems_stat as
(SELECT * FROM `target.order_items` oi JOIN order_stat o ON oi.order_id = o.order_id),
cust_ord_items as
(SELECT c.customer_state,count(c.customer_state) as customer_count,
round(avg(diff_estimated_delivery)) avg_est_del_days,
round(avg(time_to_delivery)) avg_time_del_days,
round(avg(freight_value)) as freight_amount
FROM `target.customers` c
JOIN orderitems_stat os ON c.customer_id = os.customer_id
GROUP BY c.customer_state)
SELECT * FROM cust_ord_items ORDER BY freight_amount DESC LIMIT 5
```

Row	customer_state	customer_count	avg_est_del_day	avg_time_del_da	freight_amount
1	SP	46448	11.0	8.0	15.0
2	PR	5649	13.0	11.0	20.0
3	RJ	14143	14.0	15.0	21.0
4	DF	2355	12.0	13.0	21.0
5	MG	12916	13.0	12.0	21.0

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

```
WITH order_stat as
(SELECT order_id,customer_id,order_purchase_timestamp order_date,
order_estimated_delivery_date estimated_delivery,
order_delivered_customer_date delivery_date,
```

```
abs(date_diff(order_purchase_timestamp,order_estimated_delivery_date,day)) as
actual_est_delivery,

abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as
diff_estimated_delivery,

abs(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) as
time_to_delivery

FROM `dsmlsql.target.orders`

WHERE order_status = 'delivered'),


orderitems_stat as

(SELECT * FROM `target.order_items` oi JOIN order_stat o ON oi.order_id = o.order_id),


cust_ord_items as

(SELECT c.customer_state,count(c.customer_state) as customer_count,

round(avg(diff_estimated_delivery)) avg_est_del_days,

round(avg(time_to_delivery)) avg_time_del_days,

round(avg(freight_value)) as freight_amount

FROM `target.customers` c

JOIN orderitems_stat os ON c.customer_id = os.customer_id

GROUP BY c.customer_state)


SELECT customer_state,avg_time_del_days FROM cust_ord_items ORDER BY
avg_time_del_days DESC LIMIT 5
```

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

```
WITH order_stat as

(SELECT order_id,customer_id,order_purchase_timestamp order_date,

order_estimated_delivery_date estimated_delivery,

order_delivered_customer_date delivery_date,

abs(date_diff(order_purchase_timestamp,order_estimated_delivery_date,day)) as
actual_est_delivery,

abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as
diff_estimated_delivery,

abs(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) as
time_to_delivery

FROM `dsmlsql.target.orders`

WHERE order_status = 'delivered'),


orderitems_stat as

(SELECT * FROM `target.order_items` oi JOIN order_stat o ON oi.order_id = o.order_id),


cust_ord_items as

(SELECT c.customer_state,count(c.customer_state) as customer_count,

round(avg(diff_estimated_delivery)) avg_est_del_days,

round(avg(time_to_delivery)) avg_time_del_days,

round(avg(freight_value)) as freight_amount
```

```
FROM `target.customers` c
JOIN orderitems_stat os ON c.customer_id = os.customer_id
GROUP BY c.customer_state)

SELECT customer_state,avg_time_del_days FROM cust_ord_items ORDER BY
avg_time_del_days LIMIT 5
```

Row	customer_state	avg_time_del_da
1	SP	8.0
2	PR	11.0
3	MG	12.0
4	DF	13.0
5	RS	15.0

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

```
WITH order_stat as
(SELECT order_id,customer_id,order_purchase_timestamp order_date,
order_estimated_delivery_date estimated_delivery,
order_delivered_customer_date delivery_date,
abs(date_diff(order_purchase_timestamp,order_estimated_delivery_date,day)) as
actual_est_delivery,
abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as
diff_estimated_delivery,
abs(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) as
time_to_delivery
FROM `dsmlsql.target.orders`
WHERE order_status = 'delivered'),
```

```
orderitems_stat as
(SELECT * FROM `target.order_items` oi JOIN order_stat o ON oi.order_id = o.order_id),

cust_ord_items as
(SELECT c.customer_state,count(c.customer_state) as customer_count,
round(avg(diff_estimated_delivery)) avg_est_del_days,
round(avg(time_to_delivery)) avg_time_del_days,
round(avg(freight_value)) as freight_amount
FROM `target.customers` c
JOIN orderitems_stat os ON c.customer_id = os.customer_id
GROUP BY c.customer_state)

SELECT customer_state,avg_est_del_days FROM cust_ord_items ORDER BY
avg_est_del_days DESC LIMIT 5
```

Row	customer_state	avg_est_del_day
1	AP	25.0
2	RR	25.0
3	AC	21.0
4	AM	20.0
5	RO	20.0

```
WITH order_stat as
(SELECT order_id,customer_id,order_purchase_timestamp order_date,
order_estimated_delivery_date estimated_delivery,
order_delivered_customer_date delivery_date,
abs(date_diff(order_purchase_timestamp,order_estimated_delivery_date,day)) as
actual_est_delivery,
```

```
abs(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as
diff_estimated_delivery,

abs(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) as
time_to_delivery

FROM `dsmlsql.target.orders`

WHERE order_status = 'delivered'),

orderitems_stat as

(SELECT * FROM `target.order_items` oi JOIN order_stat o ON oi.order_id = o.order_id),

cust_ord_items as

(SELECT c.customer_state,count(c.customer_state) as customer_count,

round(avg(diff_estimated_delivery)) avg_est_del_days,

round(avg(time_to_delivery)) avg_time_del_days,

round(avg(freight_value)) as freight_amount

FROM `target.customers` c

JOIN orderitems_stat os ON c.customer_id = os.customer_id

GROUP BY c.customer_state)

SELECT customer_state,avg_est_del_days FROM cust_ord_items ORDER BY
avg_est_del_days LIMIT 5
```

Row	customer_state	avg_est_del_day
1	SP	11.0
2	AL	12.0
3	SC	12.0
4	ES	12.0
5	MS	12.0

6. Payment type analysis:

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

```
WITH month_year as

(SELECT order_id,customer_id,EXTRACT(year FROM order_purchase_timestamp) as year ,EXTRACT(month FROM order_purchase_timestamp) as month

FROM `dsmlsql.target.orders`)


SELECT my.year,my.month, p.payment_type,count(p.order_id) as order_count

FROM `target.payments` p JOIN month_year my ON p.order_id = my.order_id

GROUP BY p.payment_type,my.year,my.month ORDER BY my.year,my.month,p.payment_type
```

Row	year	month	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61
11	2017	2	UPI	398
12	2017	2	credit_card	1356
13	2017	2	debit_card	13

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

WITH month_year as

(SELECT order_id,customer_id,EXTRACT(year FROM order_purchase_timestamp) as year ,EXTRACT(month FROM order_purchase_timestamp) as month

FROM `dsmlsql.target.orders`)

SELECT p.payment_installments,count(p.order_id) as order_count

FROM `target.payments` p JOIN month_year my ON p.order_id = my.order_id

GROUP BY p.payment_installments

Row	payment_install	order_count
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

7.Actionable Insights :

From the provided Data , below are some of the insights .

1.orders :

a.Most of the orders has been placed on the winter /festive season .

So we can leverage the product availability as per the requirements.

b.During non-seasonal month , we can plan for flash sale which will improve the sale of products.

Payments:

Most of the payment is happening using credit card so could provide more credit cards providers with offers , so it will attract more customer and repeated orders.

8.Recommendations :

Reduce the time taken to deliver the product.

Since the gap between estimated and actual delivery time is more , can provide relatively lesser estimated time.

Reach the customers in states (where purchase is less) through marketing the target retail through social media and other modes .

Improve the product description content to be more precise and simple