TARGET_SQL BUSINESS CASE STUDY :-

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:
    a. Data type of all columns in the "customers" table.

```
SELECT column_name, data_type
FROM `target_sql.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

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

    b. Get the time range between which the orders were placed.

```
SELECT
    MIN(order_purchase_timestamp) AS earliest_order,
    MAX(order_purchase_timestamp) AS latest_order
FROM `target_sql.orders`;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | E

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

    c. Count the Cities & States of customers who ordered during the given period.

```sql
SELECT
    COUNT(DISTINCT customer_city) AS unique_cities,
    COUNT(DISTINCT customer_state) AS unique_states
FROM `target_sql.customers`;
```

Query results

JOB INFORMATION | RESULTS | CHA

Row	unique_cities	unique_states
1	4119	27

2. In-depth Exploration:
    a. Is there a growing trend in the no. of orders placed over the past years?

```sql
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS
order_year,
    COUNT(order_id) AS total_orders
FROM `target_sql.orders`
GROUP BY order_year
ORDER BY order_year;
```

Query results

JOB INFORMATION | RESULTS | CHA

Row	order_year	total_orders
1	2016	329
2	2017	45101
3	2018	54011

    b. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

```sql
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS
order_year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS
order_month,
    COUNT(order_id) AS total_orders
FROM `target_sql.orders`
GROUP BY order_year, order_month
ORDER BY order_year, order_month;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	order_year	order_month	total_orders
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

c. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
   i. 0-6 hrs : Dawn
   ii. 7-12 hrs : Mornings
   iii. 13-18 hrs : Afternoon
   iv. 19-23 hrs : Night

```sql
SELECT
    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,
    COUNT(order_id) AS total_orders
FROM `target_sql.orders`
GROUP BY time_of_day
ORDER BY total_orders DESC;
```

Query results

JOB INFORMATION | RESULTS | CHART | J

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

3. Evolution of E-commerce orders in the Brazil region:
    a. Get the month on month no. of orders placed in each state.

```
SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS
order_year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp)
AS order_month,
    c.customer_state,
    COUNT(o.order_id) AS total_orders
FROM `target_sql.orders` o
JOIN `target_sql.customers` c ON o.customer_id =
c.customer_id
```

```
GROUP BY order_year, order_month, c.customer_state
ORDER BY order_year, order_month, total_orders DESC;
```

Query results

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

Row	order_year	order_month	customer_state	total_orders
1	2016	9	SP	2
2	2016	9	RR	1
3	2016	9	RS	1
4	2016	10	SP	113
5	2016	10	RJ	56
6	2016	10	MG	40
7	2016	10	RS	24
8	2016	10	PR	19
9	2016	10	SC	11
10	2016	10	GO	9

b. How are the customers distributed across all the states?

```
SELECT
    customer_state,
    COUNT(DISTINCT customer_id) AS total_customers
FROM `target_sql.customers`
GROUP BY customer_state
ORDER BY total_customers DESC;
```

Query results

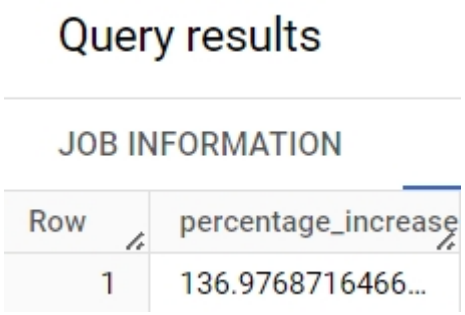
JOB INFORMATION | RESULTS | CHART | J

Row	customer_state	total_customers
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.
    a. 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.

```
SELECT
     (SUM(CASE
             WHEN EXTRACT(YEAR FROM
o.order_purchase_timestamp) = 2018
              AND EXTRACT(MONTH FROM
o.order_purchase_timestamp) BETWEEN 1 AND 8
             THEN p.payment_value
             ELSE 0
         END)
      -
      SUM(CASE
             WHEN EXTRACT(YEAR FROM
o.order_purchase_timestamp) = 2017
```

```
            AND EXTRACT(MONTH FROM
o.order_purchase_timestamp) BETWEEN 1 AND 8
            THEN p.payment_value
            ELSE 0
        END))
    / SUM(CASE
            WHEN EXTRACT(YEAR FROM
o.order_purchase_timestamp) = 2017
            AND EXTRACT(MONTH FROM
o.order_purchase_timestamp) BETWEEN 1 AND 8
            THEN p.payment_value
            ELSE 0
        END) * 100 AS percentage_increase
FROM `target_sql.orders` o
JOIN `target_sql.payments` p ON o.order_id =
p.order_id;
```



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

```
SELECT
    c.customer_state,
    SUM(p.payment_value) AS total_order_price,
    AVG(p.payment_value) AS avg_order_price
FROM `target_sql.orders` o
JOIN `target_sql.customers` c ON o.customer_id =
c.customer_id
```

```
JOIN `target_sql.payments` p ON o.order_id =
p.order_id
GROUP BY c.customer_state
ORDER BY total_order_price DESC;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUT

Row	customer_state	total_order_price	avg_order_price
1	SP	5998226.959999...	137.5046297739...
2	RJ	2144379.689999...	158.5258882235...
3	MG	1872257.260000...	154.7064336473...
4	RS	890898.5399999...	157.1804057868...
5	PR	811156.3799999...	154.1536259977...
6	SC	623086.4299999...	165.9793367075...
7	BA	616645.8200000...	170.8160166204...
8	DF	355141.0800000...	161.1347912885...
9	GO	350092.3100000...	165.7634043560...
10	ES	325967.55	154.7069530137...

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

```
SELECT
    c.customer_state,
    SUM(oi.freight_value) AS total_freight_cost,
    AVG(oi.freight_value) AS avg_freight_cost
FROM `target_sql.orders` o
JOIN `target_sql.customers` c ON o.customer_id =
c.customer_id
JOIN `target_sql.order_items` oi ON o.order_id =
oi.order_id
GROUP BY c.customer_state
ORDER BY total_freight_cost DESC;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTI

Row	customer_state	total_freight_cost	avg_freight_cost
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...

5. Analysis based on sales, freight and delivery time.
    a. 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.
    Do this in a single query.

    You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:
        i. time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
        ii. diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

```
SELECT
    order_id,

DATE_DIFF(DATE(order_delivered_customer_date),
DATE(order_purchase_timestamp), DAY) AS
time_to_deliver,

DATE_DIFF(DATE(order_delivered_customer_date),
```

```
DATE(order_estimated_delivery_date), DAY) AS
diff_estimated_delivery
FROM `target_sql.orders`
WHERE order_delivered_customer_date IS NOT
NULL;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTI

Row	order_id	time_to_deliver	diff_estimated_delive
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	31	-29
3	65d1e226dfaeb8cdc42f66542...	36	-17
4	635c894d068ac37e6e03dc54e...	31	-2
5	3b97562c3aee8bdedcb5c2e45...	33	-1
6	68f47f50f04c4cb6774570cfde...	30	-2
7	276e9ec344d3bf029ff83a161c...	44	4
8	54e1a3c2b97fb0809da548a59...	41	4
9	fd04fa4105ee8045f6a0139ca5...	37	1
10	302bb8109d097a9fc6e9cefc5...	34	5

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

```
SELECT customer_state, AVG(freight_value) AS
avg_freight
FROM `target_sql.orders` o
JOIN `target_sql.customers` c ON o.customer_id =
c.customer_id
JOIN `target_sql.order_items` oi ON o.order_id =
oi.order_id
GROUP BY customer_state
ORDER BY avg_freight DESC
LIMIT 10;
```

Query results

JOB INFORMATION | RESULTS | CHART | JS

Row	customer_state	avg_freight
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...
6	MA	38.25700242718...
7	TO	37.24660317460...
8	SE	36.65316883116...
9	AL	35.84367117117...
10	PA	35.83268518518...

```
SELECT customer_state, AVG(freight_value) AS
avg_freight
FROM `target_sql.orders` o
JOIN `target_sql.customers` c ON o.customer_id =
c.customer_id
JOIN `target_sql.order_items` oi ON o.order_id =
oi.order_id
GROUP BY customer_state
ORDER BY avg_freight ASC
LIMIT 10;
```

Query results

JOB INFORMATION | RESULTS | CHART | J

Row	customer_state	avg_freight
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...
6	SC	21.47036877394...
7	RS	21.73580433039...
8	ES	22.05877659574...
9	GO	22.76681525932...
10	MS	23.37488400488...

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

```
SELECT
    c.customer_state,

AVG(DATE_DIFF(DATE(o.order_delivered_customer_date),
DATE(o.order_purchase_timestamp), DAY)) AS
avg_delivery_time
FROM `target_sql.orders` o
JOIN `target_sql.customers` c ON o.customer_id =
c.customer_id
WHERE o.order_delivered_customer_date IS NOT NULL
GROUP BY c.customer_state
ORDER BY avg_delivery_time DESC
LIMIT 5;
```

Query results

JOB INFORMATION | RESULTS | CHART | JS

Row	customer_state	avg_delivery_time
1	RR	29.34146341463...
2	AP	27.17910447761...
3	AM	26.35862068965...
4	AL	24.50125944584...
5	PA	23.72515856236...

```
SELECT
    c.customer_state,

AVG(DATE_DIFF(DATE(o.order_delivered_customer_date),
DATE(o.order_purchase_timestamp), DAY)) AS
avg_delivery_time
FROM `target_sql.orders` o
```

```
JOIN `target_sql.customers` c ON o.customer_id =
c.customer_id
WHERE o.order_delivered_customer_date IS NOT NULL
GROUP BY c.customer_state
ORDER BY avg_delivery_time ASC
LIMIT 5;
```

Query results

JOB INFORMATION | RESULTS | CHART | JS

Row	customer_state	avg_delivery_time
1	SP	8.700530929744...
2	PR	11.93804590696...
3	MG	11.94654337296...
4	DF	12.89903846153...
5	SC	14.90752748801...

d. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

```
SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_estimated_delivery_date,
o.order_delivered_customer_date, DAY)) AS
avg_fast_delivery
FROM `target_sql.orders` o
JOIN `target_sql.customers` c ON o.customer_id =
c.customer_id
WHERE o.order_delivered_customer_date IS NOT NULL
GROUP BY c.customer_state
ORDER BY avg_fast_delivery DESC
LIMIT 5;
```

Query results

JOB INFORMATION | RESULTS | CHART | JS

Row	customer_state	avg_fast_delivery
1	AC	19.76250000000...
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

6. Analysis based on the payments:

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

```
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS
order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS
order_month,
p.payment_type,
COUNT(o.order_id) AS total_orders
FROM `target_sql.orders` o
JOIN `target_sql.payments` p
ON o.order_id = p.order_id
GROUP BY order_year, order_month, p.payment_type
ORDER BY order_year, order_month, total_orders DESC;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	order_year	order_month	payment_type	total_orders
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9

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

```
SELECT payment_installments,
COUNT(order_id) AS total_orders
FROM `target_sql.payments`
GROUP BY payment_installments
```

Query results

JOB INFORMATION | RESULTS | CHART

Row	payment_installment	total_orders
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