SQL BUSINESS CASE: TARGET

I. 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
    `compact-cell-318317.Target.INFORMATION_SCHEMA.COLUMNS`
WHERE
    table_name = 'target.customers';
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EX

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

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

```
SELECT
  MIN (order_purchase_timestamp) AS first_order_time,
  MAX (order_purchase_timestamp) AS last_order_time
FROM
  `Target.target.orders`;
```

Query results

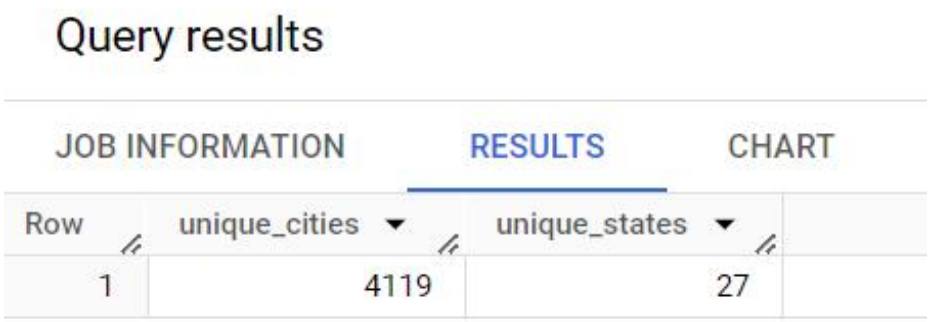
JOB INFORMATION | RESULTS | CHART | JSON

Row	first_order_time	last_order_time
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.

```
SELECT
```

```
    COUNT (DISTINCT c.customer_city) AS unique_cities,
    COUNT (DISTINCT c.customer_state) AS unique_states
FROM
    Target.target.customers AS c
JOIN
    Target.target.orders AS o
ON
    c.customer_id = o.customer_id;
```



II. In-depth Exploration:

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

```
SELECT
  EXTRACT (YEAR FROM order_purchase_timestamp) AS order_year,
  EXTRACT (MONTH FROM order_purchase_timestamp) AS order_month,
  COUNT (*) AS order_count
FROM
  Target.target.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	order_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800

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

```
SELECT
    EXTRACT (MONTH FROM order_purchase_timestamp) AS order_month,
    COUNT (*) AS total_orders
FROM
    Target.target.orders
GROUP BY
    order_month
ORDER BY
    order_month;
```

Query results

JOB INFORMATION | RESULTS | CHART

Row	order_month	total_orders
1	1	8069
2	2	8508

C. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

```
WITH brazilian_orders AS (
    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'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23
THEN 'Night'
        ELSE 'Midnight'
    END AS order_count
    FROM
        Target.target.orders
)
SELECT
    order_count,
    COUNT(*) AS order_count_total
FROM

    Target.target.orders

GROUP BY
    order_count
ORDER BY
    order_count;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	order_count	order_count_total
1	Afternoon	38135
2	Dawn	5242
3	Morning	27733
4	Night	28331

III. 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 (*) AS order_count
FROM
  Target.target.orders AS o
JOIN
  Target.target.customers AS c
ON
```

```
  o.customer_id = c.customer_id
GROUP BY
  order_year, order_month, c.customer_state
ORDER BY
  order_year, order_month, c.customer_state;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	order_year	order_month	customer_state	order_count
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2
5	2016	10	BA	4

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

```
SELECT customer_state,
  COUNT (DISTINCT customer_id) AS unique_customers_count
FROM Target.target.customers
GROUP BY customer_state
ORDER BY customer_state;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	unique_customers_c
1	AC	81
2	AL	413
3	AM	148
4	AP	68

IV. 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).

```
WITH OrderCosts AS (
    SELECT
        EXTRACT (YEAR FROM o.order_purchase_timestamp) AS year,
        SUM(p.payment_value) AS total_payment_value
    FROM
        Target.target.orders AS o
    JOIN
        Target.target.Payments_table AS p
    ON
        o.order_id = p.order_id
```

```
    WHERE
        EXTRACT (YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)
        AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
    GROUP BY
        year
)
SELECT
    (oc2018.total_payment_value - oc2017.total_payment_value) /
oc2017.total_payment_value * 100 AS percentage_increase
FROM
    (SELECT total_payment_value FROM OrderCosts WHERE year = 2017) AS
oc2017,
    (SELECT total_payment_value FROM OrderCosts WHERE year = 2018) AS
oc2018;
```

Query results

JOB INFORMATION | RESULTS

Row	percentage_increase
1	136.9768716466...

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 average_order_price
FROM
Target.target.orders AS o
JOIN Target.target.Payments_table AS p
ON
o.order_id = p.order_id
JOIN Target.target.customers AS c
ON
o.customer_id = c.customer_id
GROUP BY
c.customer_state;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	total_order_price	average_order_price
1	RJ	2144379.689999...	158.5258882235...
2	RS	890898.5399999...	157.1804057868...
3	SP	5998226.959999...	137.5046297739...
4	DF	355141.0800000...	161.1347912885...

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

```
SELECT
```

```
    c.customer_state,
    SUM (oi.freight_value) AS total_freight_value,
    AVG (oi.freight_value) AS average_freight_value
FROM
    Target.target.order_items AS oi
JOIN
    Target.target.orders AS o
ON
    oi.order_id = o.order_id
JOIN
    Target.target.customers AS c
ON
    o.customer_id = c.customer_id
GROUP BY
    c.customer_state;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION

Row	customer_state	total_freight_value	average_freight_valu
1	SP	718723.0699999...	15.14727539041...
2	RJ	305589.3100000...	20.96092393168...
3	PR	117851.6800000...	20.53165156794...
4	SC	89660.26000000...	21.47036877394...

V. 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.

```
SELECT
    order_id,
    (order_delivered_customer_date - order_purchase_timestamp) AS
Delivered_in_days,
    (order_estimated_delivery_date - order_purchase_timestamp) AS
Estimated_delivery_in_days,
    (order_estimated_delivery_date - order_delivered_customer_date)
AS      Diff_estimated_delivered_in_days
FROM
    Target.target.orders
WHERE
    order_delivered_customer_date IS NOT NULL
ORDER BY
```

```
    Delivered_in_days;
```

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

Row	order_id	Delivered_in_days	Estimated_delivery_in_days	Diff_estimated_delivered_in_days
1	1d893dd7ca5f77ebf5f59f0d20...	0-0 0 12:48:7	0-0 0 255:40:15	0-0 0 242:52:8
2	434cecee7d1a65fc65358a632...	0-0 0 18:45:10	0-0 0 490:38:14	0-0 0 471:53:4
3	f3c6775ba3d2d9fe2826f93b71...	0-0 0 20:31:39	0-0 0 300:22:13	0-0 0 279:50:34
4	8339b608be0d84fca9d8da68b...	0-0 0 20:43:20	0-0 0 675:11:27	0-0 0 654:28:7

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

```
(
  SELECT
    c.customer_state,
    AVG(oi.freight_value) AS avg_freight_value
  FROM
    Target.target.customers AS c
  JOIN
    Target.target.orders AS o
  ON
    c.customer_id = o.customer_id
  JOIN
    Target.target.order_items AS oi
  ON
    o.order_id = oi.order_id
  GROUP BY
    c.customer_state
  ORDER BY
    avg_freight_value ASC
  LIMIT 5
)
UNION ALL
(
  SELECT
    c.customer_state,
    AVG (oi.freight_value) AS avg_freight_value
  FROM
    Target.target.customers AS c
  JOIN
    Target.target.orders AS o
  ON
    c.customer_id = o.customer_id
  JOIN
    Target.target.order_items AS oi
  ON
    o.order_id = oi.order_id
  GROUP BY
    c.customer_state
  ORDER BY
    avg_freight_value DESC
  LIMIT 5
```

```
)
ORDER BY
  avg_freight_value ASC;
```

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	avg_freight_value
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...

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

```
(
    SELECT
        c.customer_state,
        ROUND (AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)), 2) AS Avg_delivery_time,
        'Highest' AS Category
    FROM
        Target.target.order_items AS oi
    JOIN
        Target.target.orders AS o
    ON
        oi.order_id = o.order_id
    JOIN
        Target.target.customers AS c
    ON
        o.customer_id = c.customer_id
    GROUP BY
        c.customer_state
    ORDER BY
        Avg_delivery_time DESC
    LIMIT 5
)
UNION ALL
(
    SELECT
        c.customer_state,
        ROUND (AVG (DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)), 2) AS Avg_delivery_time,
        'Lowest' AS Category
    FROM
        Target.target.order_items AS oi
    JOIN
        Target.target.orders AS o
    ON
        oi.order_id = o.order_id
    JOIN
       Target.target.customers AS c
```

```
    ON
        o.customer_id = c.customer_id
    GROUP BY
        c.customer_state
    ORDER BY
        Avg_delivery_time ASC
    LIMIT 5
);
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | E

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

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,
    CEIL (AVG(DATE_DIFF(o.order_estimated_delivery_date,
o.order_delivered_customer_date, DAY))) AS Deliver_speed
FROM
    Target.target.orders o
JOIN
    Target.target.customers c
ON
    o.customer_id = c.customer_id
WHERE
    o.order_status = 'delivered'      -- Include only delivered orders
GROUP BY
    c.customer_state
ORDER BY
    Deliver_speed ASC         -- States with the fastest delivery
LIMIT 5;
```

Query results

JOB INFORMATION | RESULTS | CHART | JS

Row	customer_state	Deliver_speed
1	AL	8.0
2	MA	9.0
3	SE	10.0
4	ES	10.0
5	CE	10.0

VI. Analysis based on the payments:

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

```
SELECT
    EXTRACT (MONTH FROM o.order_purchase_timestamp) AS month,
    p.payment_type,
    COUNT (DISTINCT o.order_id) AS Orders_count
FROM
    Target.target.orders AS o
JOIN
    Target.target.Payments_table AS p
ON
    o.order_id = p.order_id
GROUP BY
    month, p.payment_type
ORDER BY
    month;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION

Row	month	payment_type	Orders_count
1	1	voucher	337
2	1	credit_card	6093
3	1	debit_card	118
4	1	UPI	1715
5	2	credit_card	6582

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

```
SELECT
    p.payment_installments,
    COUNT(DISTINCT o.order_id) AS Orders_count
FROM
    Target.target.orders AS o
JOIN
    Target.target.Payments_table AS p
ON
    o.order_id = p.order_id
WHERE
    p.payment_installments >= 1
GROUP BY
    p.payment_installments
ORDER BY
    p.payment_installments;
```

Query results

JOB INFORMATION | RESULTS | CHART

Row	payment_installment	Orders_count
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234