Nipun Roy
nipunroy02@gmail.com

Target_SQL_Project
(E-commerce)

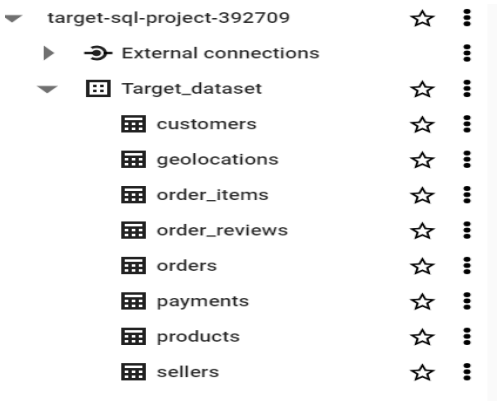
Target is a globally renowned brand and a prominent retailer in the United States.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018.

By analysing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as

- Order processing
- Pricing strategies
- Payment and shipping efficiency
- Customer demographics
- Product characteristics
- Customer satisfaction levels.

Usual exploratory analysis steps like checking the structure & characteristics of the dataset.



1. Data type of all columns in the "customers" table.

SQL QUERY:-

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

Nipun Roy
nipunroy02@gmail.com

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

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

SQL Query:-

```
SELECT MIN(order_purchase_timestamp) AS start_time,
MAX(order_purchase_timestamp) AS end_time
FROM `Target_dataset.orders`;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

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

SQL Query-

```
SELECT COUNT(DISTINCT c.customer_city) as city, COUNT(DISTINCT c.customer_state) as state
FROM `Target_dataset.orders` o
JOIN `Target_dataset.customers` c
ON o.customer_id = c.customer_id;
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	city	state
1	4119	27

Since we are looking for total count of cities & states of customers who ordered, so we need to join orders table to get the details of all the customers who placed orders.

In-depth Exploration:

1. **Is there a growing trend in the no. of orders placed over the past years?**
2. **Can we see some kind of monthly seasonality in terms of the no. of orders being placed?**

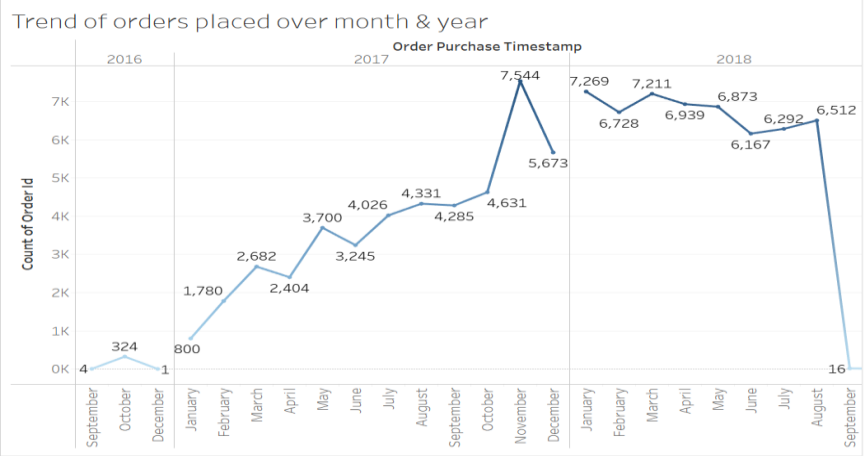
SQL Query:-

```
SELECT EXTRACT(year FROM order_purchase_timestamp) AS year,
EXTRACT (month FROM order_purchase_timestamp) AS month, COUNT(order_id) AS total_orders
FROM `Target_dataset.orders`
GROUP BY 1,2
ORDER BY year ASC, month ASC;
```

output:-

Nipun Roy
nipunroy02@gmail.com

Row	year	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



Trends & seasonality:-

1. The number of orders are increasing quarter over quarter in 2017.
   Count of orders
   Q1 (jan,feb,mar) < Q2(apr,may,jun) and so on..
2. November 2017 has the highest orders followed by decrease in orders in Dec 2017.
3. Year 2018 has more orders in 1st half of the year as compared to 2017.
4. The last 2 months of 2018 (sep,oct) can be ignored since there is an abnormal decline in values and we should not infer any conclusion without knowing the proper info.

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

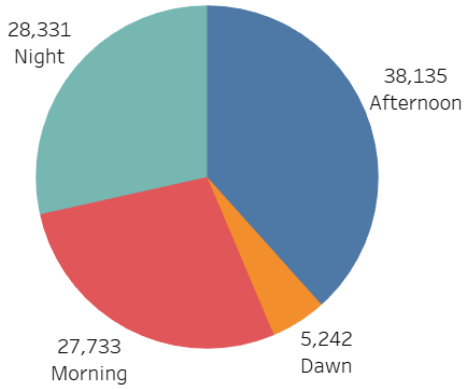
i. 0-6 hrs : Dawn
ii. 7-12 hrs : Mornings
iii. 13-18 hrs : Afternoon
iv. 19-23 hrs : Night

SQL Query:-

```
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_dataset.orders`
GROUP BY time_of_day
ORDER BY time_of_day;
```

Nipun Roy
nipunroy02@gmail.com

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



This pie chart shows that the Brazilian customers mostly place their orders in Afternoon followed by night and morning time, whereas in dawn time lowest number of orders are placed.

Evolution of E-commerce orders in the Brazil region:

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

SQL Query:-

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
customer_state,
COUNT(order_id) AS total_orders
FROM `Target_dataset.orders` o
JOIN `Target_dataset.customers` c
ON o.customer_id = c.customer_id
GROUP BY year, month, customer_state
ORDER BY year, month, customer_state;
```

Row	year	month	customer_state	total_orders
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
6	2016	10	CE	8
7	2016	10	DF	6
8	2016	10	ES	4
9	2016	10	GO	9
10	2016	10	MA	4

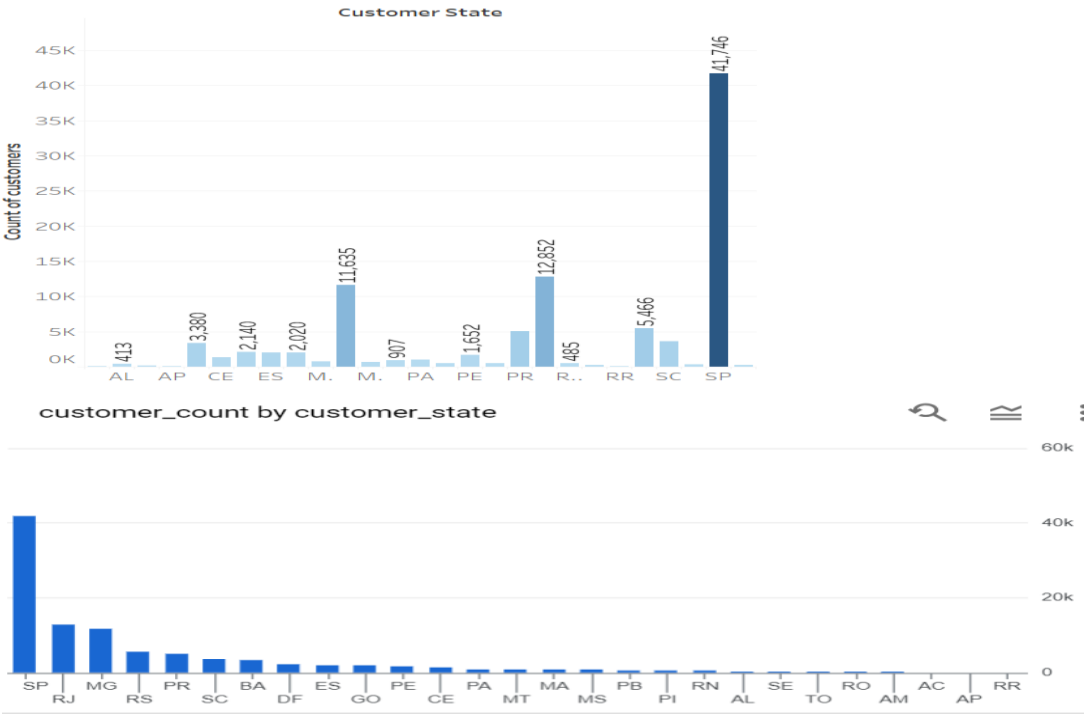
2. How are the customers distributed across all the states?

SQL Query:-

```
SELECT customer_state, COUNT(*) AS customer_count
FROM `Target_dataset.customers`
GROUP BY customer_state
ORDER BY customer_count DESC;
```

Nipun Roy
nipunroy02@gmail.com

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



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

1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

SQL Query:-

```
WITH cte1 AS (
SELECT *
FROM `Target_dataset.orders` o
JOIN `Target_dataset.payments` p
ON o.order_id = p.order_id
WHERE EXTRACT(year FROM order_purchase_timestamp) BETWEEN 2017 AND 2018
AND EXTRACT(month FROM order_purchase_timestamp) BETWEEN 1 AND 8),

cte2 AS(
  SELECT EXTRACT(year FROM order_purchase_timestamp) AS year,
  SUM(payment_value) AS cost
  FROM cte1
  GROUP BY year)

SELECT *,
  ROUND((cost - LAG(cost, 1) OVER (ORDER BY year))*100/LAG(cost, 1) OVER (ORDER BY year),2)
AS percent_inc
FROM cte2
```

Nipun Roy
nipunroy02@gmail.com

Row	year	cost	percent_inc
1	2017	3669022.119999...	null
2	2018	8694733.839999...	136.98

In this I have joined the orders table to payments table to get the payment_value on conditions like year 2017 & 2018, months Jan to aug.
Once I have the details within the condition, next step is to get the aggregated cost per year
And then final expression i.e... (Cost –previous year cost)*100 divided by previous year cost to get the percentage increment (rounded by 2 decimal places).

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

SQL Query:-

```
SELECT c.customer_state,
  COUNT(DISTINCT o.order_id) AS unique_ord,
  SUM(ot.price) AS total_amount,
  SUM(ot.price)/COUNT(DISTINCT o.order_id) AS avg_price
FROM `Target_dataset.orders` o
JOIN `Target_dataset.order_items` ot
ON o.order_id = ot.order_id
JOIN `Target_dataset.customers` c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY unique_ord DESC;
```

Row	customer_state	unique_ord	total_amount	avg_price
1	SP	41375	5202955.050001...	125.7511794562...
2	RJ	12762	1824092.669999...	142.9315679360...
3	MG	11544	1585308.029999...	137.3274454261...
4	RS	5432	750304.0200000...	138.1266605301...
5	PR	4998	683083.7600000...	136.6714205682...
6	SC	3612	520553.3400000...	144.1177574750...
7	BA	3358	511349.9900000...	152.2781387730...
8	DF	2125	302603.9399999...	142.4018541176...
9	ES	2025	275037.3099999...	135.8208938271...
10	GO	2007	294591.9499999...	146.7822371699...

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

```
SELECT c.customer_state,
  COUNT(DISTINCT o.order_id) AS unique_ord,
  SUM(ot.freight_value) AS total_amount,
  SUM(ot.freight_value)/COUNT(DISTINCT o.order_id) AS avg_order_fright
FROM `Target_dataset.orders` o
JOIN `Target_dataset.order_items` ot
ON o.order_id = ot.order_id
JOIN `Target_dataset.customers` c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY unique_ord DESC;
```

Nipun Roy
nipunroy02@gmail.com

Row	customer_state	unique_ord	total_amount	avg_order_fright
1	SP	41375	718723.0699999...	17.37095033232...
2	RJ	12762	305589.3100000...	23.94525231155...
3	MG	11544	270853.4600000...	23.46270443520...
4	RS	5432	135522.7400000...	24.94895802650...
5	PR	4998	117851.6800000...	23.57976790716...
6	SC	3612	89660.26000000...	24.82288482834...
7	BA	3358	100156.6799999...	29.82628945801...
8	DF	2125	50625.49999999...	23.82376470588...
9	ES	2025	49764.59999999...	24.57511111111...
10	GO	2007	53114.97999999...	26.46486297957...

Analysis based on sales and delivery time.

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.
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:

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

SQL Query:-

```
SELECT
    order_id,
    order_purchase_timestamp,
    order_delivered_customer_date,
    order_estimated_delivery_date,
    DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date,DAY) AS
time_to_deliver,
    DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date,DAY) AS
diff_estimated_delivery
FROM
    `Target_dataset.orders`;
```

Row	order_id	order_purchase	order_delivered_	order_estimated	time_to_deliver	diff_estimated_deliv
1	1950d77798...	2018-02-19 ...	2018-03-21 ...	2018-03-09 ...	-30	12
2	2c45c33d2f...	2016-10-09 ...	2016-11-09 ...	2016-12-08 ...	-30	-28
3	65d1e226df...	2016-10-03 ...	2016-11-08 ...	2016-11-25 ...	-35	-16
4	635c894d06...	2017-04-15 ...	2017-05-16 ...	2017-05-18 ...	-30	-1
5	3b97562c3a...	2017-04-14 ...	2017-05-17 ...	2017-05-18 ...	-32	0
6	68f47f50f04...	2017-04-16 ...	2017-05-16 ...	2017-05-18 ...	-29	-1
7	276e9ec344...	2017-04-08 ...	2017-05-22 ...	2017-05-18 ...	-43	4
8	54e1a3c2b9...	2017-04-11 ...	2017-05-22 ...	2017-05-18 ...	-40	4
9	fd04fa4105e...	2017-04-12 ...	2017-05-19 ...	2017-05-18 ...	-37	1
10	302bb8109d...	2017-04-19 ...	2017-05-23 ...	2017-05-18 ...	-33	5

2. 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.

SQL Query:-

```
SELECT
    customer_state,
    AVG(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date,DAY)) AS
avg_delivery_speed
```

Nipun Roy
nipunroy02@gmail.com

```
FROM `Target_dataset.orders` o
JOIN `Target_dataset.customers` c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY avg_delivery_speed ASC
LIMIT 5;
```

Row	customer_state	avg_delivery_speed
1	AC	-19.7625000000...
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6068965517...
5	RR	-16.4146341463...

Analysis based on the payments:

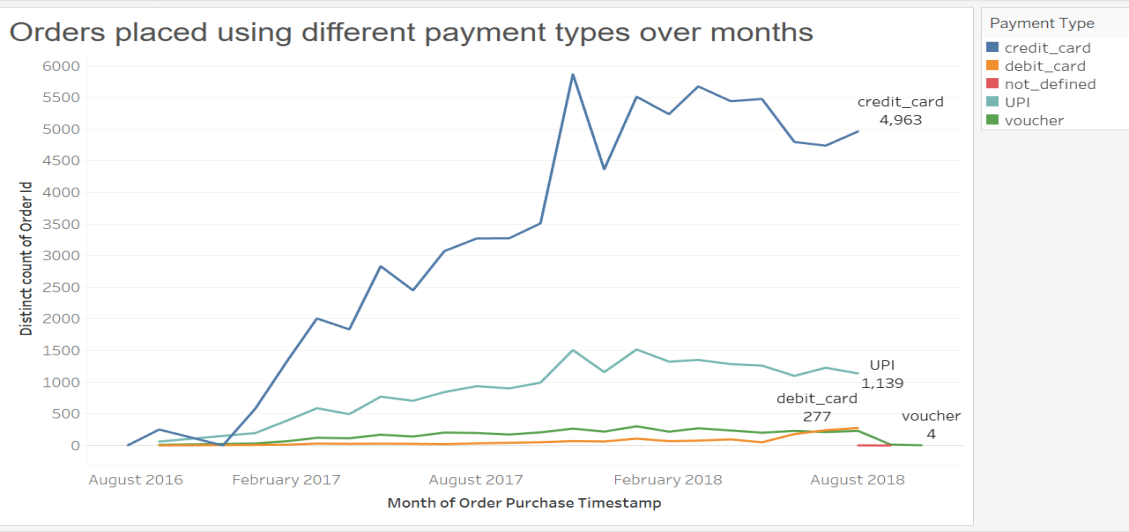
1. Find the month on month no. of orders placed using different payment types.

SQL Query:-

```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  payment_type,
  COUNT(o.order_id) AS total_orders
  FROM `Target_dataset.orders` o
  JOIN `Target_dataset.payments` p
  ON o.order_id = p.order_id
  GROUP BY year, month, payment_type
  ORDER BY year, month, payment_type;
```

Row	year	month	payment_type	total_orders
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

Nipun Roy
nipunroy02@gmail.com

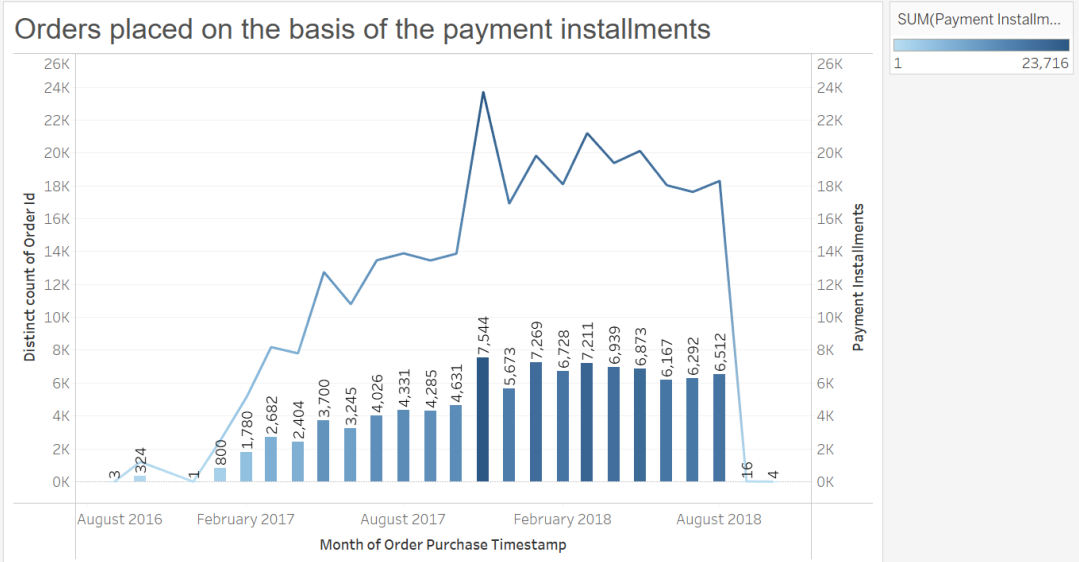


2. Find the no. of orders placed on the basis of the payment instalments that have been paid.

```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  payment_installments,
  COUNT(o.order_id) AS total_orders
  FROM `Target_dataset.orders` o
  JOIN `Target_dataset.payments` p
  ON o.order_id = p.order_id
  WHERE payment_installments >= 1
  GROUP BY year, month, payment_installments
  ORDER BY year, month, payment_installments;
```

Row	year	month	payment_installment	total_orders
1	2016	9	1	1
2	2016	9	2	1
3	2016	9	3	1
4	2016	10	1	144
5	2016	10	2	30
6	2016	10	3	43
7	2016	10	4	26
8	2016	10	5	20
9	2016	10	6	18
10	2016	10	7	13

Nipun Roy
nipunroy02@gmail.com



Line chart Represents:-Payments Installments

Bar Represents: - Distinct Count of Order Id.