Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

By-Nikhil Mehalawat

nikhilmehalawat@gmailcom

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

- Data type of all columns in the “customers” table.

```
SELECT
  column_name,
  data_type,
  is_nullable
FROM `scaler-project-465604.Target.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```

Query completed

Using on-demand processing quota

Query results

Save results | Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

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

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

```sql
select
min(order_purchase_timestamp) as oldest,max(order_purchase_timestamp) as latest
from
scaler-project-465604.Target.orders
```

Query completed

Query results

Save results | Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

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

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

Untitled query | Run | Save | Download | Share | Schedule

```sql
select
count(distinct customer_state) as state_count,count(distinct customer_city) as city_count
from
scaler-project-465604.Target.customers as c
inner join
scaler-project-465604.Target.orders as o
on c.customer_id=o.customer_id;
```

Query completed

Query results

Save results | Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	state_count	city_count
1	27	4119

II. 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 year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
COUNT(order_id) AS total_records
FROM scaler-project-465604.Target.orders
GROUP BY year, month
ORDER BY year, month;
```

Query results

Save results | Open in

Job information | Results | Visualization | JSON | Execution details | Execution graph

Row	year	month	total_records
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

Results per page: 50 1 – 25 of 25

 # Yes, there is certainly a growing trend in the number of orders placed over the past years.

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

```
SELECT
  month,
  AVG(monthly_orders) AS avg_orders
FROM (
  SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
    EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
    COUNT(*) AS monthly_orders
  FROM `scaler-project-465604.Target.orders`
  GROUP BY year, month
)
GROUP BY month
ORDER BY month;
```

Job information | Results | Visualization | JSO

Row	month	avg_orders
1	1	4034.5
2	2	4254.0
3	3	4946.5
4	4	4671.5
5	5	5286.5
6	6	4706.0
7	7	5159.0
8	8	5421.5
9	9	1435.0
10	10	1653.0
11	11	7544.0
12	12	2837.0

 # Here, we can see monthly seasonality in terms of order peaking in November.

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

```
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'
```

```
  END AS time_of_day,
  COUNT(*) AS total_orders
FROM `scaler-project-465604.Target.orders`
GROUP BY time_of_day
ORDER BY total_orders DESC;
```

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

#Here we can see that most orders are placed at afternoon.

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

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

```
 SELECT
c.customer_state,extract(Month FROM order_purchase_timestamp ) as month
,count(o.order_id) as total_orders
from
scaler-project-465604.Target.orders as o
left join scaler-project-465604.Target.customers as c
on o.customer_id = c.customer_id
group by
c.customer_state,extract(Month FROM order_purchase_timestamp )
order by c.customer_state,extract(Month FROM order_purchase_timestamp );
```

Row	customer_state	month	total_orders
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5
12	AC	12	5
13	AL	1	39

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

```
select
customer_state as state,
count(distinct customer_unique_id) as total_customer
from
scaler-project-465604.Target.customers
group by customer_state
```

Row	state	total_customer
1	SP	40302
2	SC	3534
3	MG	11259
4	PR	4882
5	RJ	12384
6	RS	5277
7	PA	949
8	GO	1952
9	ES	1964
10	BA	3277
11	MA	726
12	MS	694

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 cte AS (

```
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    SUM(p.payment_value) AS cost_of_orders
  FROM
    scaler-project-465604.Target.orders AS o
  LEFT JOIN
    scaler-project-465604.Target.payments 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
    EXTRACT(YEAR FROM o.order_purchase_timestamp)
)

SELECT
  ROUND(
    ( (cost_2018 - cost_2017) / cost_2017 ) * 100
  , 2) AS percent_increase
FROM (
  SELECT
    MAX(CASE WHEN year = 2017 THEN cost_of_orders END) AS cost_2017,
    MAX(CASE WHEN year = 2018 THEN cost_of_orders END) AS cost_2018
  FROM cte
);
```

Row	percent_increase
1	136.98

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

```
SELECT
  c.customer_state AS state,
  SUM(p.payment_value) AS total_order_value,
  AVG(p.payment_value) AS average_order_value
FROM
  scaler-project-465604.Target.orders AS o
```

```
JOIN
  scaler-project-465604.Target.customers AS c
    ON o.customer_id = c.customer_id
JOIN
  scaler-project-465604.Target.payments AS p
    ON o.order_id = p.order_id
GROUP BY
  c.customer_state
ORDER BY
  total_order_value DESC;
```

Row	state	total_order_value	average_order_va...
4	RS	890898.5399999...	157.1804057868...
5	PR	811156.3799999...	154.1536259977...
6	SC	623086.43	165.9793367075...
7	BA	616645.8200000...	170.8160166204...
8	DF	355141.0799999...	161.1347912885...
9	GO	350092.310000001	165.7634043560...
10	ES	325967.5500000...	154.7069530137...
11	PE	324850.4399999...	187.9921527777...
12	CE	279464.03	199.9027396280...
13	PA	218295.85	215.9207220573...

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

```
SELECT
  c.customer_state AS state,
  SUM(oi.freight_value) AS total_freight_value,
  AVG(oi.freight_value) AS average_freight_value
FROM
  scaler-project-465604.Target.orders AS o
JOIN
```

```
  scaler-project-465604.Target.customers AS c
    ON o.customer_id = c.customer_id
JOIN
  scaler-project-465604.Target.orders_items AS oi
    ON o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  total_freight_value DESC;
```

Row	state	total_freight_value	average_freight_v...
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...

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
  o.order_id,
  DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)
    AS time_to_deliver,
  DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date,
DAY)
    AS diff_estimated_delivery
FROM
  scaler-project-465604.Target.orders AS o
WHERE
  o.order_delivered_customer_date IS NOT NULL
  AND o.order_purchase_timestamp IS NOT NULL
  AND o.order_estimated_delivery_date IS NOT NULL;
```

Row	order_id	time_to_deliver	diff_estimated_d...
1	ecab90c9933c58908d3d6add7...	30	16
2	8563039e855156e48fccee4d61...	30	0
3	6ea2f835b4556291ffdc53fa0b3...	33	-7
4	6a0a8bfbbe700284feb0845d95...	36	-17
5	9d531c565e28c3e0d756192f8...	56	-32
6	8fc207e94fa91a7649c5a5dab6...	54	-32
7	f31535f21d145b2345e2bf7f09...	81	-49
8	06ae7271902bbb087fc093137f...	31	1
9	4906eeadde5f70b308c20c4a8f...	32	-6
10	bca3dc20a3ec02261c5b17dc2...	30	0

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

```
WITH avg_freight AS (
  SELECT
    c.customer_state AS state,
    AVG(oi.freight_value) AS avg_freight_value
  FROM
```

```
   scaler-project-465604.Target.orders AS o
  JOIN
   scaler-project-465604.Target.customers AS c
     ON o.customer_id = c.customer_id
  JOIN
   scaler-project-465604.Target.orders_items AS oi
     ON o.order_id = oi.order_id
  GROUP BY
   c.customer_state
),

bottom_5 AS (
  SELECT *
  FROM avg_freight
  ORDER BY avg_freight_value ASC
  LIMIT 5
),

top_5 AS (
  SELECT *
  FROM avg_freight
  ORDER BY avg_freight_value DESC
  LIMIT 5
)

SELECT *
FROM (
  SELECT * FROM bottom_5
  UNION ALL
  SELECT * FROM top_5
)
ORDER BY avg_freight_value ASC;
```

Row	state	avg_freight_value
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...

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

```
WITH delivery_time AS (
  SELECT
    c.customer_state AS state,
    TIMESTAMP_DIFF(o.order_delivered_customer_date,
                   o.order_purchase_timestamp, DAY) AS delivery_days
  FROM
    scaler-project-465604.Target.orders AS o
  JOIN
    scaler-project-465604.Target.customers AS c
      ON o.customer_id = c.customer_id
  WHERE
    o.order_delivered_customer_date IS NOT NULL
),

avg_delivery AS (
  SELECT
    state,
    AVG(delivery_days) AS avg_delivery_time
  FROM delivery_time
  GROUP BY state
),

bottom_5 AS (
```

```
  SELECT *
  FROM avg_delivery
  ORDER BY avg_delivery_time ASC
  LIMIT 5
),

top_5 AS (
  SELECT *
  FROM avg_delivery
  ORDER BY avg_delivery_time DESC
  LIMIT 5
)

SELECT *
FROM (
  SELECT * FROM bottom_5
  UNION ALL
  SELECT * FROM top_5
)
ORDER BY avg_delivery_time ASC;
```

Row	state	avg_delivery_time
1	SP	8.29806148907271
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...

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

```
WITH delivered_orders AS (
  SELECT
    c.customer_state AS state,
    DATE_DIFF(
      o.order_estimated_delivery_date,
      o.order_delivered_customer_date,
      DAY
    ) AS diff_estimated_actual
  FROM
    scaler-project-465604.Target.orders AS o
  JOIN scaler-project-465604.Target.customers AS c
    ON o.customer_id = c.customer_id
  WHERE
    o.order_delivered_customer_date IS NOT NULL
    AND o.order_estimated_delivery_date IS NOT NULL
),
with_avg AS (
  SELECT
    state,
    diff_estimated_actual,
    AVG(diff_estimated_actual) OVER (PARTITION BY state) AS avg_days_early
  FROM delivered_orders
)
SELECT DISTINCT
  state,
  avg_days_early
FROM with_avg
ORDER BY avg_days_early DESC
LIMIT 5;
```

Row	state	avg_days_early
1	AC	19.7625
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

VI. 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(DISTINCT o.order_id) AS total_orders
FROM
  scaler-project-465604.Target.orders AS o
JOIN
  scaler-project-465604.Target.payments AS p
    ON o.order_id = p.order_id
GROUP BY
  order_year,
  order_month,
  p.payment_type
ORDER BY
  order_year,
  order_month,
  p.payment_type;
```

Row	order_year	order_month	payment_type	total_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	253
4	2016	10	debit_card	2
5	2016	10	voucher	11
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	582
9	2017	1	debit_card	9
10	2017	1	voucher	33
11	2017	2	UPI	398
12	2017	2	credit_card	1347
13	2017	2	debit_card	13
14	2017	2	voucher	69

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

```
SELECT
  payment_installments,
  COUNT(DISTINCT order_id) AS total_orders
FROM
  scaler-project-465604.Target.payments
WHERE
  payment_installments >= 1
  group by
  payment_installments
ORDER BY
  payment_installments;
```

Row	payment_installm...	total_orders
1	1	49060
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
11	11	23
12	12	133
13	13	16

Valuable insights

1. Delivery Time Varies Strongly by State
   - Some states consistently show longer delivery times than others.
   - The fastest states deliver much earlier than the estimated date (positive diff), while slow states consistently delay shipments.

   Logistics performance varies across regions; weak delivery partners or poor local infrastructure likely affect slower states.

2. Big Gap Between Estimated & Actual Delivery

- In several states, orders are delivered much earlier than estimated.
- In slow states, actual delivery is consistently later than estimated.

  The company's SLA estimates are inaccurate for certain regions.

3. Payment Behaviour Insights

A. Most customers prefer 1-installment payments

- The majority of orders are paid in 1 installment.
- Multi-installment payments (2–10) exist but drop sharply as installments increase.

  Consumers prefer simple, upfront payments; installment-heavy financing is not dominant.

B. Payment Types – Credit Card dominates

- Month-on-month payment trends show credit card as the most frequently used.
- Boleto is consistently popular in specific months but still second.
- Debit card and voucher usage remains very small.

Focus on credit card optimization and fraud prevention. Installment options should target high-ticket items only.

4. Freight Value Is Highly State Dependent

- States far from the fulfillment center(s) show higher freight charges.
- States near major logistics hubs show the lowest freight cost.
- High freight states also often have long delivery times, indicating inefficiency.

Shipping cost and speed correlate strongly — inefficient logistics means slow & expensive shipping.

5. Orders Peak Around Certain Months (Seasonality)

- Month-on-month analysis shows identifiable peaks (likely holidays, promotions).
- Some payment types (boleto, voucher) spike seasonally.

Marketing & logistics should align planning with these seasonal peaks.

Delivery speed and freight cost vary significantly across states — logistics optimization is the biggest opportunity.

Credit card dominates transactions; installment usage is low but important for high-value categories.

Improving delivery estimates, targeting slow states, and smart payment strategy can reduce costs and improve customer experience.