TARGET

SQL Business Case

Target, a leading U.S. retailer, is known for its exceptional value, innovation, and customer experience. This business case focuses on **Target's operations in Brazil**, analyzing **100,000 orders from 2016 to 2018**. The dataset provides insights into key aspects such as **order status, pricing strategies, payment and shipping efficiency, customer demographics, product attributes, and customer reviews**. By examining these factors, we can uncover trends in operational performance, optimize logistics, and enhance customer satisfaction to drive business growth.

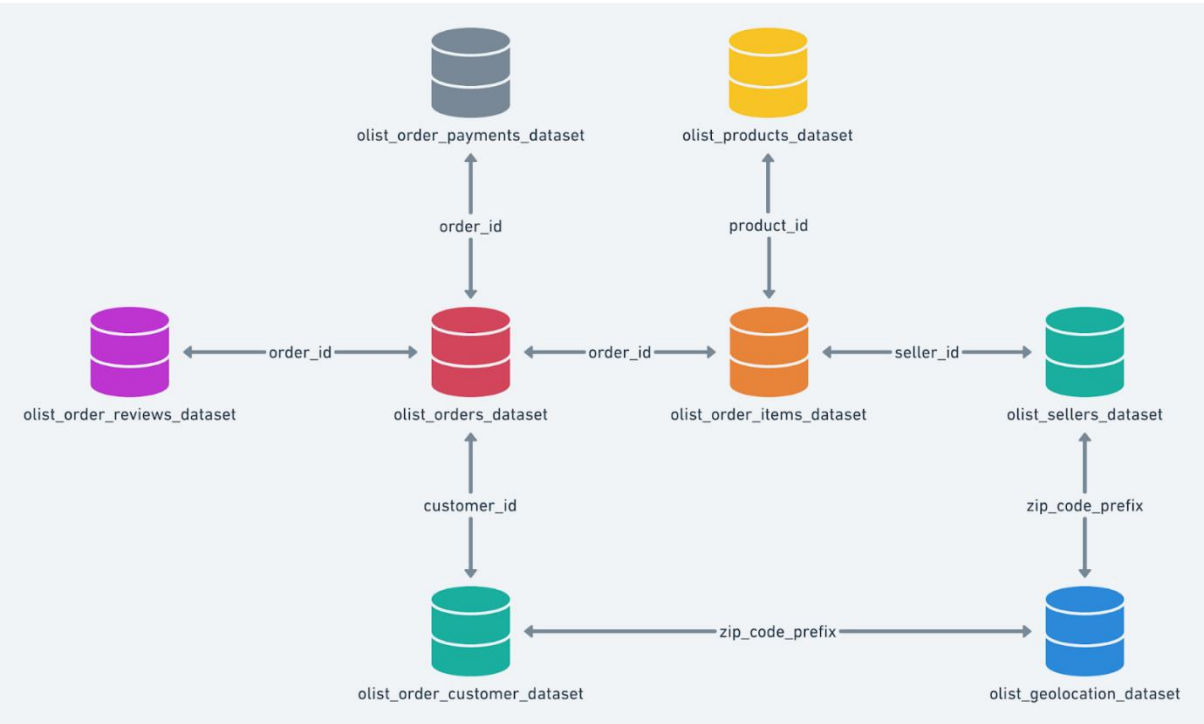
Dataset:

https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb

The data is available in 8 csv files:

1. customers.csv
2. sellers.csv
3. order_items.csv
4. geolocation.csv
5. payments.csv
6. reviews.csv
7. orders.csv
8. products.csv

Dataset Schema:



Problem Statement:

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.

Q1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

1.Data type of columns in a table

a. Data type of columns in 'customers' table

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

b. Data type of columns in 'products' table

Field name	Type	Mode
product_id	STRING	NULLABLE
product_category	STRING	NULLABLE
product_name_length	INTEGER	NULLABLE
product_description_length	INTEGER	NULLABLE
product_photos_qty	INTEGER	NULLABLE
product_weight_g	INTEGER	NULLABLE
product_length_cm	INTEGER	NULLABLE
product_height_cm	INTEGER	NULLABLE
product_width_cm	INTEGER	NULLABLE

c. Data type of columns in ‘orders’ table

Field name	Type	Mode
order_id	STRING	NULLABLE
customer_id	STRING	NULLABLE
order_status	STRING	NULLABLE
order_purchase_timestamp	TIMESTAMP	NULLABLE
order_approved_at	TIMESTAMP	NULLABLE
order_delivered_carrier_date	TIMESTAMP	NULLABLE
order_delivered_customer_date	TIMESTAMP	NULLABLE
order_estimated_delivery_date	TIMESTAMP	NULLABLE

d. Data type of columns in ‘order_items’ table

Field name	Type	Mode
order_id	STRING	NULLABLE
order_item_id	INTEGER	NULLABLE
product_id	STRING	NULLABLE
seller_id	STRING	NULLABLE
shipping_limit_date	TIMESTAMP	NULLABLE
price	FLOAT	NULLABLE
freight_value	FLOAT	NULLABLE

e. Data type of columns in ‘order_reviews’ table

Field name	Type	Mode
review_id	STRING	NULLABLE
order_id	STRING	NULLABLE
review_score	INTEGER	NULLABLE
review_comment_title	STRING	NULLABLE
review_creation_date	TIMESTAMP	NULLABLE
review_answer_timestamp	TIMESTAMP	NULLABLE

f. Data type of columns in ‘payments’ table

Field name	Type	Mode
order_id	STRING	NULLABLE
payment_sequential	INTEGER	NULLABLE
payment_type	STRING	NULLABLE
payment_installments	INTEGER	NULLABLE
payment_value	FLOAT	NULLABLE

g. Data type of columns in ‘sellers’ table

Field name	Type	Mode
seller_id	STRING	NULLABLE
seller_zip_code_prefix	INTEGER	NULLABLE
seller_city	STRING	NULLABLE
seller_state	STRING	NULLABLE

h. Data type of columns in ‘geolocation’ table

Field name	Type	Mode
geolocation_zip_code_prefix	INTEGER	NULLABLE
geolocation_lat	FLOAT	NULLABLE
geolocation_lng	FLOAT	NULLABLE
geolocation_city	STRING	NULLABLE
geolocation_state	STRING	NULLABLE

2.Time period for which the data is given

Query:

```
select
min(order_purchase_timestamp),
max(order_purchase_timestamp)
from `SQL_Project.orders`;
```

Output:

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

Insight:

The first purchase was made on 04 September 2016 at 21:15:19 UTC. And the last purchase was made on 17 October 2018 at 17:30:18 UTC. So roughly, **the time period for which we have data is about 2 years, 1 month and 13 days.**

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

Query:

```
select distinct o.order_purchase_timestamp,
c.customer_city,
c.customer_state
from `SQL_Project.orders` o
join `SQL_Project.customers` c
on o.customer_id = c.customer_id
where order_purchase_timestamp between
  (select min(order_purchase_timestamp)
   from `SQL_Project.orders` )
   and (select max(order_purchase_timestamp)
   from `SQL_Project.orders`)
order by o.order_purchase_timestamp;
```

Output:

Row	order_purchase_timestamp	customer_city	customer_state
1	2016-09-04 21:15:19 UTC	boa vista	RR
2	2016-09-05 00:15:34 UTC	passo fundo	RS
3	2016-09-13 15:24:19 UTC	sao jose dos campos	SP
4	2016-09-15 12:16:38 UTC	sao joaquim da barra	SP
5	2016-10-02 22:07:52 UTC	sao paulo	SP
6	2016-10-03 09:44:50 UTC	sao paulo	SP
7	2016-10-03 16:56:50 UTC	panambi	RS
8	2016-10-03 21:01:41 UTC	rio de janeiro	RJ
9	2016-10-03 21:13:36 UTC	porto alegre	RS
10	2016-10-03 22:06:03 UTC	hortolandia	SP

Q2. 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?

Query:

```
select
extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year,
count(order_id) as total_purchase
from
`SQL_Project.orders`
group by  year, month
order by year, month;
```

Output:

Row	month	year	total_purchase
1	9	2016	4
2	10	2016	324
3	12	2016	1
4	1	2017	800
5	2	2017	1780
6	3	2017	2682
7	4	2017	2404
8	5	2017	3700
9	6	2017	3245
10	7	2017	4026

Row	month	year	total_purchase
11	8	2017	4331
12	9	2017	4285
13	10	2017	4631
14	11	2017	7544
15	12	2017	5673
16	1	2018	7269
17	2	2018	6728
18	3	2018	7211
19	4	2018	6939
20	5	2018	6873

Row	month	year	total_purchase
21	6	2018	6167
22	7	2018	6292
23	8	2018	6512
24	9	2018	16
25	10	2018	4

Insights:

We can see a linear increase in the purchases made by the Brazilian customers from 2016 to 2018, along with the peak sales made roughly in the middle of the year. In 2017, the highest sales are made in the latter half of the year. While in 2018, we have maximum purchases made in the beginning of the year, remains almost constant, but decreases dramatically as the year begins to end.

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

Query:

```
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 7 AND
EXTRACT(HOUR FROM order_purchase_timestamp) < 12 THEN
'morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 12 AND
EXTRACT(HOUR FROM order_purchase_timestamp) < 17 THEN
'afternoon'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 17 AND
EXTRACT(HOUR FROM order_purchase_timestamp) < 20  THEN 'dusk'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 4 AND
EXTRACT(HOUR FROM order_purchase_timestamp) < 7 THEN 'dawn'
    ELSE 'night'
  END AS time_of_day,
  SUM(payment_value) AS total_purchases,
  count(o.order_id) AS total_orders,
FROM `SQL_Project.orders' o join `SQL_Project.payments` p
on o.order_id = p.order_id
GROUP BY time_of_day
ORDER BY total_orders desc, total_purchases desc
```

Output:

Row	time_of_day	total_purchases	total_orders
1	afternoon	5299071.830000...	33603
2	night	4194711.929999...	28092
3	morning	3473626.600000...	22618
4	dusk	2918977.950000...	18649
5	dawn	122483.8099999...	924

Insights:

The maximum orders are placed in the afternoon , followed by night , morning and dusk. Very less customers tend to order during the dawn.

Q3. Evolution of E-Commerce orders in Brazil regions:

1. Get month on month orders by state.

Query:

```
SELECT customer_state,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
COUNT(*) AS order_count
FROM `SQL_Project.orders` o
Join `SQL_Project.customers` c
on o.customer_id = c.customer_id
GROUP BY customer_state, month
ORDER BY order_count desc, customer_state;
```

Output:

Row	customer_state	month	order_count
1	SP	8	4982
2	SP	5	4632
3	SP	7	4381
4	SP	6	4104
5	SP	3	4047
6	SP	4	3967
7	SP	2	3357
8	SP	1	3351
9	SP	11	3012
10	SP	12	2357
11	SP	10	1908
12	SP	9	1648
13	RJ	5	1321
14	RJ	8	1307
15	RJ	3	1302

Insights:

The maximum number of orders made are from the state of Sao Paulo (SP) during the middle of the year, followed by Rio de Janeiro(RJ) and Minas Gerais(MG) consecutively.

2. Distribution of customers across the states in Brazil.

Query:

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

Output:

Row	customer_state	num_of_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
11	PE	1652
12	CE	1336
13	PA	975
14	MT	907
15	MA	747

Insights:

The maximum number of customers are from the state of **Sao Paulo(SP)** followed by **Rio de Janeiro(RJ)** and **Minas Gerais(MG).**

Q4. Impact on Economy: Analyse the money movement by e-commerce by looking at order prices, freight and others.

1. 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.

Query:

```
SELECT cost_2017.total_cost as order_cost_2017,
cost_2018.total_cost as order_cost_2018,
((cost_2018.total_cost - cost_2017.total_cost) /
cost_2017.total_cost) * 100 AS percentage_increase

FROM (SELECT SUM(payment_value) AS total_cost
      FROM `SQL_Project.orders` as o
      JOIN`SQL_Project.payments` as p
      ON o.order_id = p.order_id
      WHERE EXTRACT(YEAR FROM order_purchase_timestamp) = 2017
      AND EXTRACT(MONTH FROM order_purchase_timestamp)
          BETWEEN 1 AND 8) as cost_2017,

     (SELECT SUM(payment_value) AS total_cost
     FROM `SQL_Project.orders` as o
     JOIN `SQL_Project.payments` as p
     ON o.order_id = p.order_id
     WHERE EXTRACT(YEAR FROM order_purchase_timestamp) = 2018
     AND EXTRACT(MONTH FROM order_purchase_timestamp)
         BETWEEN 1 AND 8) as cost_2018
```

Output:

Row	order_cost_2017	order_cost_2018	percentage_increase
1	3669022.120000...	8694733.839999...	136.9768716466...

Insights:

The total order cost of the year 2017 is **3669022.12** and that of year 2018 is **8694733.839**. Thus, the percent increase in 2017 to 2018 is approximately **136.976871%.**

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

Query:

```
SELECT customer_state,
SUM(price) AS sum_price,
SUM(freight_value) AS sum_freight,
AVG(price) AS mean_price,
AVG(freight_value) AS mean_freight

FROM `SQL_Project.order_items` as oi
JOIN `SQL_Project.orders` as o
on oi.order_id = o.order_id
JOIN `SQL_Project.customers` as c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY sum_price DESC, sum_freight DESC;
```

Output:

Row	customer_state	sum_price	sum_freight	mean_price	mean_freight
1	SP	5202955.050001...	718723.0699999...	109.6536291597...	15.14727539041...
2	RJ	1824092.669999...	305589.3100000...	125.1178180945...	20.96092393168...
3	MG	1585308.029999...	270853.4600000...	120.7485741488...	20.63016680630...
4	RS	750304.0200000...	135522.7400000...	120.3374530874...	21.73580433039...
5	PR	683083.7600000...	117851.6800000...	119.0041393728...	20.53165156794...
6	SC	520553.3400000...	89660.26000000...	124.6535775862...	21.47036877394...
7	BA	511349.9900000...	100156.6799999...	134.6012082126...	26.36395893656...
8	DF	302603.9399999...	50625.49999999...	125.7705486284...	21.04135494596...
9	GO	294591.9499999...	53114.97999999...	126.2717316759...	22.76681525932...
10	ES	275037.3099999...	49764.59999999...	121.9137012411...	22.05877659574...
11	PE	262788.0299999...	59449.65999999...	145.5083222591...	32.91786267995...
12	CE	227254.7099999...	48351.58999999...	153.7582611637...	32.71420162381...
13	PA	178947.8099999...	38699.30000000...	165.6924166666...	35.83268518518...

Insights:

The products with highest price and freight are ordered by the customers of **Sao Paulo(SP)** followed by **Rio de Janeiro(RJ)** and **Minas Gerais(MG).**

Q5. Analysis on sales, freight and delivery time

1.Calculate days between purchasing, delivering and estimated delivery

Query:

```
select
 order_id,
 order_purchase_timestamp as purchase_date,
 order_delivered_customer_date as delivery_date,
 order_estimated_delivery_date as estimated_delivery_date,
TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) AS diff_purchase_delivery,
TIMESTAMP_DIFF(order_estimated_delivery_date,
order_purchase_timestamp, DAY) AS
diff_purchase_estimated_delivery
FROM `SQL_Project.orders`
order by  diff_purchase_delivery,
diff_purchase_estimated_delivery;
```

Output:

Row	order_id	purchase_date	delivery_date	estimated_delivery_date	diff_purchase_delivery	diff_purchase_estimated_delivery
1	e5215415bb...	2016-10-22 08:25:27 UTC	null	2016-10-24 00:00:00 UTC	null	1
2	3213c825fd...	2018-08-07 21:40:46 UTC	null	2018-08-10 00:00:00 UTC	null	2
3	54282e97f6...	2018-09-03 09:06:57 UTC	null	2018-09-06 00:00:00 UTC	null	2
4	2a06568281...	2018-08-21 11:00:37 UTC	null	2018-08-24 00:00:00 UTC	null	2
5	749309f96d...	2018-08-14 17:57:15 UTC	null	2018-08-17 00:00:00 UTC	null	2
6	2ad4df0af7...	2018-07-30 12:04:19 UTC	null	2018-08-02 00:00:00 UTC	null	2
7	0b16257b1b...	2018-08-07 18:31:25 UTC	null	2018-08-10 00:00:00 UTC	null	2
8	e49e7ce147...	2018-08-07 11:16:28 UTC	null	2018-08-10 00:00:00 UTC	null	2
9	a0f2f8b052...	2018-07-31 10:48:42 UTC	null	2018-08-03 00:00:00 UTC	null	2
10	1e0f40bbfe7...	2018-08-06 20:17:25 UTC	null	2018-08-09 00:00:00 UTC	null	2

2.Find time_to_delivery & diff_estimated_delivery.

Query:

```
select
 order_id,
 order_purchase_timestamp as purchase_date,
 order_delivered_customer_date as delivery_date,
 order_estimated_delivery_date as estimated_delivery_date,
TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) AS time_to_delivery,
TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY) AS diff_estimated_delivery
FROM `SQL_Project.orders`
order by  time_to_delivery, diff_estimated_delivery;
```

Output:

Row	order_id	purchase_date	delivery_date	estimated_delivery_date	time_to_delivery	diff_estimated_delivery
1	7a4df5d8cff4...	2017-11-25 11:10:33 UTC	null	2017-12-12 00:00:00 UTC	null	null
2	35de4050331...	2017-12-05 01:07:58 UTC	null	2018-01-08 00:00:00 UTC	null	null
3	b5359909123...	2017-12-05 01:07:52 UTC	null	2018-01-11 00:00:00 UTC	null	null
4	dba5062fbda...	2018-02-09 17:21:04 UTC	null	2018-03-07 00:00:00 UTC	null	null
5	90ab3e7d525...	2017-11-06 13:12:34 UTC	null	2017-12-01 00:00:00 UTC	null	null
6	fa65dad1b0e...	2017-04-20 12:45:34 UTC	null	2017-05-18 00:00:00 UTC	null	null
7	1df2775799e...	2017-07-13 11:03:05 UTC	null	2017-08-14 00:00:00 UTC	null	null
8	6190a94657e...	2017-07-11 13:36:30 UTC	null	2017-08-14 00:00:00 UTC	null	null
9	58ce513a55c...	2017-07-29 18:05:07 UTC	null	2017-08-14 00:00:00 UTC	null	null
10	088683f795a...	2017-07-13 10:02:47 UTC	null	2017-08-14 00:00:00 UTC	null	null

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

Query:

```
SELECT customer_state,
AVG(freight_value) as avg_freight_value,
AVG(EXTRACT( DAY FROM order_delivered_customer_date))
AS avg_delivery_date,
AVG(TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY))
AS diff_estimated_delivery
FROM `SQL_Project.order_items` oi
JOIN `SQL_Project.orders` o
ON oi.order_id = o.order_id
JOIN `SQL_Project.customers` c
ON c.customer_id = o.customer_id
GROUP BY customer_state
ORDER BY avg_freight_value, avg_delivery_date,
diff_estimated_delivery;
```

Output:

Row	customer_state	avg_freight_value	avg_delivery_date	diff_estimated_delivery
1	SP	15.147275390419132	16.136726740305242	10.265594384514429
2	PR	20.531651567944269	15.888829881394958	12.533899805275258
3	MG	20.630166806306651	15.756599829681832	12.39715104126349
4	RJ	20.960923931682483	16.069277534285341	11.144493142937899
5	DF	21.041354945968422	16.137154989384253	11.274734607218669
6	SC	21.470368773946323	15.869692532942894	10.668862859931693
7	RS	21.735804330392952	15.753138757541203	13.203000163052348
8	ES	22.058776595744732	15.915505617977546	9.7685393258427116
9	GO	22.766815259322772	16.299077733860344	11.372859025032941
10	MS	23.374884004884006	16.020961775585697	10.337854500616515

Insights:

The state with lowest average freight value , average delivery date and lowest difference between purchase and estimated delivery is **Sao Paulo(SP)** followed by **Parana(PR)**, **Minas Gerais(MG) and Rio de Janeiro(RJ).**

4. Sort the data to get top 5 states with highest/lowest average freight value - sort in desc/asc limit 5.

a. Top 5 states with highest freight value

Query:

```
SELECT customer_state,
AVG(freight_value) AS avg_freight_value,
FROM `SQL_Project.customers` as c
JOIN `SQL_Project.orders` as o
ON c.customer_id = o.customer_id
JOIN `SQL_Project.order_items` as oi
ON o.order_id = oi.order_id
GROUP BY customer_state
ORDER BY avg_freight_value DESC
LIMIT 5;
```

Output:

Row	customer_state	avg_freight_value
1	RR	42.9844230769230...
2	PB	42.7238039867109...
3	RO	41.0697122302158...
4	AC	40.0733695652174...
5	PI	39.1479704797047...

Insights:

The state with highest average freight value is **Roraima(RR)**, followed by **Paraiba(PB), Rondonia(RO)** and **Acre(AC)**.

b. Top 5 states with lowest freight value.

Query:

```
SELECT customer_state,
AVG(freight_value) AS avg_freight_value,
FROM `SQL_Project.customers` as c
JOIN `SQL_Project.orders` as o
ON c.customer_id = o.customer_id
JOIN `SQL_Project.order_items` as oi
ON o.order_id = oi.order_id
GROUP BY customer_state
ORDER BY avg_freight_value ASC
LIMIT 5;
```

Output:

Row	customer_state	avg_freight_value
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

Insights:

The lowest freight rate is from the state of **Sao Paulo(SP),** followed by **Parana(PR)** , **Minas Gerais (MG)** and **Rio de Janeiro(RJ).**

5. Get top 5 states with highest/lowest average time to delivery

a. Top 5 states with lowest average time to delivery.

Query:

```
SELECT customer_state,
AVG(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) ) AS avg_delivery_time,
FROM `SQL_Project.customers` as c
JOIN `SQL_Project.orders` as o
ON c.customer_id = o.customer_id
JOIN `SQL_Project.order_items` as oi
ON o.order_id = oi.order_id
GROUP BY customer_state
ORDER BY avg_delivery_time ASC
LIMIT 5;
```

Output:

Row	customer_state	avg_delivery_time
1	SP	8.25960855241909
2	PR	11.480793060718735
3	MG	11.515522180072811
4	DF	12.501486199575384
5	SC	14.520985846754517

Insights:

The lowest average delivery time is in the states of **Sao Paulo(SP),** followed by **Parana(PR)** , **Minas Gerais (MG)** and **Distrito Federal (DF).**

b. Top 5 states with highest average time to delivery.

Query:

```
SELECT customer_state,
AVG(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY) ) AS avg_delivery_time,
FROM `SQL_Project.customers` as c
JOIN `SQL_Project.orders` as o
ON c.customer_id = o.customer_id
JOIN `SQL_Project.order_items` as oi
ON o.order_id = oi.order_id
GROUP BY customer_state
ORDER BY avg_delivery_time DESC
LIMIT 5;
```

Output:

Row	customer_state	avg_delivery_time
1	RR	27.82608695652...
2	AP	27.75308641975...
3	AM	25.96319018404...
4	AL	23.99297423887...
5	PA	23.30170777988...

Insights:

The highest average delivery time is in the states of **Roraima (RR)** followed by **Amapa(AP)** , **Amazonas(AM)** and **Alagoas(AL).**

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

Query:

```
SELECT
customer_state,
SUM(CASE WHEN order_delivered_customer_date <=
order_estimated_delivery_date
THEN ELSE 0 END) AS fast_deliveries,
SUM(CASE WHEN order_delivered_customer_date >
order_estimated_delivery_date
THEN  ELSE 0 END) AS not_fast_deliveries
FROM
`SQL_Project.orders` as o
JOIN `SQL_Project.customers` as c
ON c.customer_id = o.customer_id
GROUP BY
customer_state
ORDER BY
fast_deliveries DESC
LIMIT 5;
```

Output:

Row	customer_state	fast_deliveries	not_fast_deliveries
1	SP	38108	2387
2	MG	10717	638
3	RJ	10689	1664
4	RS	4962	382
5	PR	4677	246

Insights:

The fastest deliveries are made in in the states **Sao Paulo(SP),** followed by **Minas Gerais (MG) and Rio de Janeiro(RR)**.

Q6. Payment type analysis:

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

Query:

```
SELECT
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  payment_type,
  COUNT(*) AS order_count
FROM
  `SQL_Project.orders` as o
  JOIN `SQL_Project.payments` as p
  ON o.order_id = p.order_id
GROUP BY
  month, payment_type
ORDER BY
  month, payment_type;
```

Output:

Row	month	payment_type	order_count
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707
11	3	debit_card	109

Insights:

The highest number of orders are made by the **Credit Card** payment mode followed by **UPI**, **Voucher** and **Debit Card**.

2. Count of orders based on the no. of payment instalments.

Query:

```
SELECT payment_installments,
COUNT(o.order_id) as order_count
FROM `SQL_Project.orders` as o
JOIN `SQL_Project.payments`  as p
ON o.order_id = p.order_id
GROUP BY payment_installments
ORDER BY order_count DESC;
```

Output:

Row	payment_installment	order_count
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

Insights:

The maximum number of orders are made in 1 instalment, followed by 2, 3, 4 and 10 instalments.

RECOMMENDATIONS:

No seasonality was observed in 2018. The possible reasons could be the **non-availability of shopping offers** during the festive season or the **lack of attractiveness of the offers**. This area can be improved to boost sales.

Nearly two-thirds of the customers come from just three states. **Target** can focus on expanding its reach in other states to attract more customers and increase sales.

Very few people shop late at night. This presents an **opportunity** for **Target** to implement strategies to **increase sales during off-peak hours**.

The **average difference between the estimated and actual delivery date** ranges from **8 to 20 days**. Reducing this variance can enhance the customer experience by ensuring smoother and more predictable deliveries.

Some states, such as **RR and PB**, have **high freight costs**. Optimizing logistics in these regions can help **reduce operational costs** related to freight.

The **longest average delivery time** is **28 days**, which is considerably high. **Reducing delivery times** can significantly **improve customer satisfaction**.