Business Case: Target SQL

1. **Import the dataset and do the usual exploratory analysis steps like checking the structure & characteristics of the dataset:**
   a. The data type of columns in a table.

Query 1:

```
SELECT
    column_name, data_type
FROM
    INFORMATION_SCHEMA.COLUMNS
WHERE
    table_schema = “target_retail" AND table_name = "customers";
```

Output 1:

COLUMN_NAME	DATA_TYPE
customer_id	text
customer_unique_id	text
customer_zip_code_prefix	int
customer_city	text
customer_state	text

Query 2:

```
SELECT
    column_name, data_type
FROM
    INFORMATION_SCHEMA.COLUMNS
WHERE
    table_schema = “target_retail" AND table_name = “geolocation";
```

Output 2:

COLUMN_NAME	DATA_TYPE
geolocation_zip_code_prefix	int
geolocation_lat	double
geolocation_lng	double
geolocation_city	text
geolocation_state	text

Query 3:

```
SELECT
    column_name, data_type
FROM
    INFORMATION_SCHEMA.COLUMNS
WHERE
    table_schema = “target_retail" AND table_name = “order_items";
```

Output 3:

COLUMN_NAME	DATA_TYPE
order_id	text
order_item_id	int
product_id	text
seller_id	text
shipping_limit_date	text
price	double
freight_value	double

Query 4:

```
SELECT
    column_name, data_type
FROM
    INFORMATION_SCHEMA.COLUMNS
WHERE
    table_schema = “target_retail" AND table_name = “order_reviews";
```

Output 4:

COLUMN_NAME	DATA_TYPE
review_id	text
order_id	text
review_score	int
review_comment_title	text
review_creation_date	text
review_answer_timestamp	text

Query 5:

```
SELECT
    column_name, data_type
FROM
    INFORMATION_SCHEMA.COLUMNS
WHERE
    table_schema = “target_retail" AND table_name = "orders";
```

Output 5:

	COLUMN_NAME	DATA_TYPE
▶	order_id	text
	customer_id	text
	order_status	text
	order_purchase_timestamp	text
	order_approved_at	text
	order_delivered_carrier_date	text
	order_delivered_customer_date	text
	order_estimated_delivery_date	text

Query 6:

```
SELECT
    column_name, data_type
FROM
    INFORMATION_SCHEMA.COLUMNS
WHERE
    table_schema = “target_retail" AND table_name = "payments";
```

Output 6:

	COLUMN_NAME	DATA_TYPE
▶	order_id	text
	payment_sequential	int
	payment_type	text
	payment_installments	int
	payment_value	double

Query 7:

```
SELECT
    column_name, data_type
FROM
    INFORMATION_SCHEMA.COLUMNS
WHERE
    table_schema = “target_retail" AND table_name = "products";
```

Output 7:

COLUMN_NAME	DATA_TYPE
product_id	text
product category	text
product_name_length	double
product_description_length	double
product_photos_qty	double
product_weight_g	double
product_length_cm	double
product_height_cm	double
product_width_cm	double

Query 8:

```
SELECT
    column_name, data_type
FROM
    INFORMATION_SCHEMA.COLUMNS
WHERE
    table_schema = “target_retail" AND table_name = “sellers”;
```

Output 8:

COLUMN_NAME	DATA_TYPE
seller_id	text
seller_zip_code_prefix	int
seller_city	text
seller_state	text

b. The time period for which the data is given.

Explanation:

Perusing the ER diagram and the question, we can grasp that the data revolves around orders received at Target in Brazil. For this reason, we should refer ***order_purchase_timestamp*** *column in the* ***orders*** *table.*

Query:

```
SELECT
    MIN(order_purchase_timestamp) AS first_order_received_timestamp,
    MAX(order_purchase_timestamp) AS last_order_received_timestamp
FROM
    orders;
```

Output:

first_order_received_timestamp	last_order_received_timestamp
2016-09-04 21:15:19	2018-10-17 17:30:18

c. Cities and States covered in the dataset.

Explanation:

*This question can be answered by referring **customer_city** and **customer_state** columns in the **customers** table. The reason is simple - since our data is order-centric, and orders are placed by customers, the locations covered in our dataset will be those present in the **customer_city** and **customer_state** columns.*

Query 1:

```
SELECT
    DISTINCT(customer_city) AS cities_covered_in_dataset
FROM
    customers
ORDER BY
    cities_covered_in_dataset;
```

Output 1:

cities_covered_in_dataset
abadia dos dourados
abadiania
abaete
abaetetuba
abaiara
abaira
abare
abatia
abdon batista
abelardo luz

Query 2:

```
SELECT
    DISTINCT(customer_state) AS states_covered_in_dataset
FROM
    customers
ORDER BY
    states_covered_in_dataset;
```

Output 2:

states_covered_in_dataset
AC
AL
AM
AP
BA
CE
DF
ES
GO
MA

2. In-depth Exploration:

a. Is there a growing trend in e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

Explanation:

*We define a trend in e-commerce in three ways - One, the number of customer interactions v/s time. Two, number of orders v/s time. Three, total sales v/s time. To accomplish this, we need to access and join two tables - **orders** and **payments**.*

Note:

1. *Not filtering based on **order_status** in the **orders** table because we want to analyze trends in e-commerce and all types of orders, even the **canceled** ones, contribute towards the popularity of e-commerce over time.*
2. *Considering only the DISTINCT number of **customer_id** and **order_id** because no duplicates are present in the **orders** table but are there in the **payments** table.*
3. *Not considering **customer_id** because the count (with and without DISTINCT) of the column is equal to that of **order_id**. Hence, the analysis of **order_id** will be valid for **customer_id** as well.*

Query 1 (monthly level analysis):

```
SELECT
    CONCAT(YEAR(o.order_purchase_timestamp), "-", MONTH(o.order_purchase_timestamp)) AS purchase_year_month,
    COUNT(o.order_id) AS num_distinct_orders,
    ROUND(SUM(p.payment_value)) AS purchase_amount
FROM
    orders AS o
INNER JOIN
    payments AS p
ON
    o.order_id = p.order_id
GROUP BY
    YEAR(o.order_purchase_timestamp), MONTH(o.order_purchase_timestamp),
    CONCAT(YEAR(o.order_purchase_timestamp), "-", MONTH(o.order_purchase_timestamp))
ORDER BY
    YEAR(o.order_purchase_timestamp), MONTH(o.order_purchase_timestamp);
```

Output 1:

purchase_year_month	num_distinct_orders	purchase_amount
2016-9	3	252
2016-10	342	59090
2016-12	1	20
2017-1	850	138488
2017-2	1886	291908
2017-3	2837	449864
2017-4	2571	417788
2017-5	3944	592919
2017-6	3436	511276
2017-7	4317	592383
2017-8	4550	674396
2017-9	4516	727762
2017-10	4860	779678
2017-11	7863	1194883
2017-12	5895	878401
2018-1	7563	1115004
2018-2	6952	992463
2018-3	7512	1159652
2018-4	7209	1160785
2018-5	7135	1153982
2018-6	6419	1023880
2018-7	6507	1066541
2018-8	6698	1022425
2018-9	16	4440
2018-10	4	590

Query 2 (yearly level analysis):

```
SELECT
    YEAR(o.order_purchase_timestamp) AS purchase_year,
    COUNT(DISTINCT o.order_id) AS num_distinct_orders,
    ROUND(SUM(p.payment_value)) AS purchase_amount
FROM
    orders AS o
INNER JOIN
    payments AS p
ON
    o.order_id = p.order_id
GROUP BY
    YEAR(o.order_purchase_timestamp)
ORDER BY
    purchase_year;
```

Output 2:

purchase_year	num_distinct_orders	purchase_amount
2016	328	59362
2017	45101	7249747
2018	54011	8699763

Inference:

- *Growth in the popularity of e-commerce (**Target**) over time in Brazil is reflected by growth in the number of orders and total purchase amount in **Output 2** - aggregated on the yearly level.*
  - *Even though we only have 4 months of data from 2016, extrapolating the order and sales numbers for previous months of this year will not exceed those in 2017 and 2018.*
  - *We can see a significant increase in orders and sales numbers in 2018 as compared to 2017 even when we only have 10 months of data for this year.*
- *We can see 2 definitive effects of seasonality in the months of March (significant increase in orders and sales numbers) and December (significant drop in orders and sales numbers) - from **Output 1** - aggregated on the monthly level.*

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

Explanation:

The data reflecting the buying habits of customers per the time of day can be very beneficial to increase revenue and thus profit. This data can also help in improving the customers' experience - offers, facilities, etc.

Note:

1. *I have added evening to the above list of time-span provided because there was a bit of stretch between Afternoon and Night. The following list depicts the segregation of the time of day into different time-span.*
   - *Morning - 6 AM to 12 Noon*
   - *Afternoon - 12 Noon to 4 PM*
   - *Evening - 4 PM to 7 PM*
   - *Night - 7 PM to 3 AM*
   - *Dawn - 3 AM to 6 AM*

Query 1 (yearly level analysis):

```
WITH t1 AS (
    SELECT
          YEAR(order_purchase_timestamp) AS purchase_year,
       CASE
          WHEN TIME(order_purchase_timestamp) BETWEEN "00:00:00" AND "02:60:00" THEN "Night"
          WHEN TIME(order_purchase_timestamp) BETWEEN "03:00:00" AND "05:60:00" THEN "Dawn"
          WHEN TIME(order_purchase_timestamp) BETWEEN "06:00:00" AND "11:60:00" THEN "Morning"
          WHEN TIME(order_purchase_timestamp) BETWEEN "12:00:00" AND "15:60:00" THEN "Afternoon"
          WHEN TIME(order_purchase_timestamp) BETWEEN "16:00:00" AND "18:60:00" THEN "Evening"
          WHEN TIME(order_purchase_timestamp) BETWEEN "19:00:00" AND "23:60:00" THEN "Night"
       END AS time_band,
       order_id
    FROM
          orders
)

SELECT
    purchase_year, time_band, COUNT(order_id) AS total_orders
FROM
    t1
GROUP BY
    purchase_year, time_band
ORDER BY
    purchase_year;
```

Output 1:

purchase_year	time_band	total_orders
2016	Afternoon	91
2016	Dawn	3
2016	Evening	48
2016	Morning	84
2016	Night	103
2017	Afternoon	11580
2017	Dawn	327
2017	Evening	8279
2017	Morning	9774
2017	Night	15141
2018	Afternoon	13865
2018	Dawn	336
2018	Evening	10267
2018	Morning	12382
2018	Night	17161

Query 2 (cumulative analysis):

```
WITH t1 AS (
     SELECT
          CASE
               WHEN TIME(order_purchase_timestamp) BETWEEN "00:00:00" AND "02:60:00" THEN "Night"
               WHEN TIME(order_purchase_timestamp) BETWEEN "03:00:00" AND "05:60:00" THEN "Dawn"
               WHEN TIME(order_purchase_timestamp) BETWEEN "06:00:00" AND "11:60:00" THEN "Morning"
               WHEN TIME(order_purchase_timestamp) BETWEEN "12:00:00" AND "15:60:00" THEN "Afternoon"
               WHEN TIME(order_purchase_timestamp) BETWEEN "16:00:00" AND "18:60:00" THEN "Evening"
               WHEN TIME(order_purchase_timestamp) BETWEEN "19:00:00" AND "23:60:00" THEN "Night"
          END AS time_band,
          order_id
     FROM
               orders
)

SELECT
     time_band, COUNT(order_id) AS total_orders
FROM
     t1
GROUP BY
     time_band;
```

Output 2:

time_band	total_orders
Morning	22240
Night	32405
Afternoon	25536
Evening	18594
Dawn	666

Inference:

- ***Output 1*** *and* ***Output 2*** *are both in-sync with each other. From* ***Output 2****, we can conclude Brazilians shop the most during Night hours (7 PM to 3 AM) followed by Afternoon hours (12 Noon to 4 PM).* ***Output 1*** *confirms this trend on a yearly basis with most orders received during Night hours followed by Afternoon hours.*

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

a. Get month-on-month orders by region and state

Explanation:

The evolution of e-commerce in Brazilian regions can be associated with cumulative order amount and volume of orders. The locations for which these numbers are greater are those with high e-commerce penetration than those with lesser numbers. We can examine this penetration on two basis - one by state and the other by city.

Query 1 (by state):

```
WITH t1 AS (
    SELECT
          c.customer_state, o.order_id, p.payment_value
    FROM
          orders AS o
    INNER JOIN
          customers AS c
    ON
          o.customer_id = c.customer_id
    INNER JOIN
          payments AS p
    ON
          o.order_id = p.order_id
)
SELECT
    customer_state AS state, COUNT(order_id) AS total_orders, ROUND(SUM(payment_value)) AS total_order_amount
FROM
    t1
GROUP BY
    customer_state
ORDER BY
    total_order_amount DESC, total_orders DESC;
```

Output 1:

state	total_orders	total_order_amount
SP	43622	5998227
RJ	13527	2144380
MG	12102	1872257
RS	5668	890899
PR	5262	811156
SC	3754	623086
BA	3610	616646
DF	2204	355141
GO	2112	350092
ES	2107	325968
PE	1728	324850
CE	1398	279464
PA	1011	218296
MT	958	187029

Query 2 (by city):

```
WITH t1 AS (
    SELECT
            c.customer_city, o.order_id, p.payment_value
    FROM
            orders AS o
    INNER JOIN
            customers AS c
    ON
            o.customer_id = c.customer_id
    INNER JOIN
            payments AS p
    ON
            o.order_id = p.order_id
)
SELECT
    customer_city AS city, COUNT(order_id) AS total_orders, ROUND(SUM(payment_value)) AS total_order_amount
FROM
    t1
GROUP BY
    customer_city
ORDER BY
    total_order_amount DESC, total_orders DESC;
```

Output 2:

city	total_orders	total_order_amount
sao paulo	16221	2203373
rio de janeiro	7207	1161927
belo horizonte	2872	421765
brasilia	2193	354217
curitiba	1576	247392
porto alegre	1418	224731
salvador	1347	218072
campinas	1515	216248
guarulhos	1250	165122
niteroi	915	139997
goiania	741	125495
sao bernard...	979	120435
fortaleza	683	119863
santos	733	112343
recife	639	110745
florianopolis	586	106512
santo andre	820	105627

Inference:

- ***State - SP*** *has the largest contribution (approximately three times more when compared to the state at second place) towards the Number of orders and Revenue generated for e-commerce followed by* ***RJ****.*
- ***City - Sau Paulo*** *has the largest contribution (approximately two times more when compared to the city at second place) towards the Number of orders and Revenue generated for e-commerce followed by* ***Rio De Janerio****.*

b. How are customers distributed in Brazil?

Explanation:

We have to refer to the ***customers*** *table (having no redundant details) to fetch the distribution of customers across Brazil.*

Query 1 (by city):

```
SELECT
    DISTINCT customer_city AS city,
    COUNT(customer_id) OVER(PARTITION BY customer_city) AS num_customers_by_city,
    COUNT(customer_id) OVER(PARTITION BY customer_city) * 100 / COUNT(customer_id) OVER() AS percent_of_total
FROM
    customers
ORDER BY
    num_customers_by_city DESC;
```

Output 1:

city	num_customers_by_city	percent_of_total
sao paulo	15540	15.6274
rio de janeiro	6882	6.9207
belo horizonte	2773	2.7886
brasilia	2131	2.1430
curitiba	1521	1.5296
campinas	1444	1.4521
porto alegre	1379	1.3868
salvador	1245	1.2520
guarulhos	1189	1.1957
sao bernardo do campo	938	0.9433
niteroi	849	0.8538
santo andre	797	0.8015
osasco	746	0.7502
santos	713	0.7170
goiania	692	0.6959
sao jose dos campos	691	0.6949
fortaleza	654	0.6577
sorocaba	633	0.6366
recife	613	0.6164
florianopolis	570	0.5732
jundiai	565	0.5682
ribeirao preto	510	0.5129
belem	447	0.4495

Query 2 (by state):

```
SELECT
    DISTINCT customer_state AS state,
    COUNT(customer_id) OVER(PARTITION BY customer_state) AS num_customers_by_state,
    100 * COUNT(customer_id) OVER(PARTITION BY customer_state) / COUNT(customer_id) OVER() AS percent_of_total
FROM
    customers
ORDER BY
    num_customers_by_state DESC;
```

Output 2:

state	num_customers_by_state	percent_of_total
SP	41746	41.9807
RJ	12852	12.9242
MG	11635	11.7004
RS	5466	5.4967
PR	5045	5.0734
SC	3637	3.6574
BA	3380	3.3990
DF	2140	2.1520
ES	2033	2.0444
GO	2020	2.0314
PE	1652	1.6613
CE	1336	1.3435
PA	975	0.9805
MT	907	0.9121
MA	747	0.7512
MS	715	0.7190

Inference:

- ***City - Sau Paulo*** *has the largest customer base (more than twice as compared to the city at second place) followed by* ***Rio De Janerio****.*
- ***State - SP*** *has the largest customer base (more than thrice as compared to the state at second place) followed by* ***RJ****.*
- *Both of these inferences are in sync with inferences drawn from Q3.a, i.e., a location with a larger customer base will have more orders and will generate more revenue.*

4. Impact on the Economy: Analyze the money movemented by e-commerce by looking at order prices, freight, and others:

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

Explanation:

We want to analyze the percent change in the cost of all orders (sum of ***price*** *and* ***freight_value*** *columns from* ***order_items*** *table) between Jan and Aug on a year-on-year basis. Since we have data between Sept'16 and Oct'18, we will have only one row in the output corresponding to the percent change in 2018 from 2017.*

Query:

```
WITH t1 AS (
    SELECT
          YEAR(o.order_purchase_timestamp) AS order_year,
       ROUND(SUM(oi.price) + SUM(oi.freight_value)) AS cost_of_orders
    FROM
          orders AS o
    INNER JOIN
          order_items AS oi
    ON
          o.order_id = oi.order_id
    WHERE
          MONTH(order_purchase_timestamp) < 9
    GROUP BY
          YEAR(o.order_purchase_timestamp)
), t2 AS (
    SELECT
          *,
       LEAD(order_year) OVER (ORDER BY order_year ASC) AS next_year,
       LEAD(cost_of_orders) OVER (ORDER BY order_year ASC) AS next_year_cost_of_orders
    FROM
          t1
), t3 AS (
    SELECT
          *,
       CONCAT(ROUND((next_year_cost_of_orders - cost_of_orders) * 100 / cost_of_orders), "%") AS percent_change_in_order_cost
    FROM
          t2
    WHERE next_year IS NOT NULL
) SELECT * FROM t3;
```

Output:

order_year	cost_of_orders	next_year	next_year_cost_of_ord...	percent_change_in_order_cost
2017	3610270	2018	8643531	139%

Inference:

- *We saw a whopping* ***139%*** *increase in the cost of orders from 2017 to 2018.*

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

Explanation:

*The Sum and Average of **price** and **freight_value** columns in the **order_items** table will help us in identify states that contribute most towards the order value.*

Query:

```
SELECT
    c.customer_state AS state,
    ROUND(SUM(oi.price)) AS total_order_price, ROUND(AVG(oi.price)) AS avg_order_price,
    ROUND(SUM(oi.freight_value)) AS total__order_freight_value, ROUND(AVG(oi.freight_value)) AS avg_order_freight_value
FROM
    customers AS c
INNER JOIN
    orders AS o
ON
    c.customer_id = o.customer_id
INNER JOIN
    order_items AS oi
ON
    o.order_id = oi.order_id
GROUP BY
    c.customer_state
ORDER BY
    SUM(oi.price) DESC, SUM(oi.freight_value) DESC;
```

Output:

state	total_order_price	avg_order_price	total__order_freight_value	avg_order_freight_value
SP	5202955	110	718723	15
RJ	1824093	125	305589	21
MG	1585308	121	270853	21
RS	750304	120	135523	22
PR	683084	119	117852	21
SC	520553	125	89660	21
BA	511350	135	100157	26
DF	302604	126	50625	21
GO	294592	126	53115	23
ES	275037	122	49765	22
PE	262788	146	59450	33
CE	227255	154	48352	33
PA	178948	166	38699	36
MT	156454	148	29715	28
MA	119648	145	31524	38
MS	116813	143	19144	23

Inference:

- ***SP** contributes most towards the **price** and **freight_value** even when the average values are significantly less than other states which mean that a huge number of orders are received from **SP**.*

5. Analysis of sales, freight, and delivery time:

a. Calculate days between purchasing, delivering, and estimated delivery.

Explanation:

The analysis can help identify a bottleneck in the delivery mechanism by reflecting the time elapsed in different stages of the delivery mechanism and thus giving the organization insights on where to focus its energy to improve logistics.

Query 1 (order level analysis):

```
WITH t1 AS (
    SELECT
           o.order_id,
        DATE(o.order_purchase_timestamp) AS purchase_date,
        DATE(o.order_estimated_delivery_date) AS estimated_delivery_date,
        DATE(o.order_delivered_carrier_date) AS delivery_date_carrier,
        DATE(o.order_delivered_customer_date) AS delivery_date_customer,
        DATE(oi.shipping_limit_date) AS max_shipping_date
FROM
    orders AS o
INNER JOIN
    order_items AS oi
ON
    o.order_id = oi.order_id
)
SELECT
    order_id,
    delivery_date_customer - purchase_date AS total_delivery_time,
    estimated_delivery_date - delivery_date_customer AS estimation_error,
    max_shipping_date - purchase_date AS max_shipping_delay,
    delivery_date_customer - delivery_date_carrier AS carrier_delivery_delay
FROM
    t1;
```

Output 1:

order_id	total_delivery_time	estimation_error	max_shipping_delay	carrier_delivery_delay
00125cb692d04887809806618a2a145f	84	13	6	80
00571ded73b3c061925584feab0db425	12	82	7	8
00571ded73b3c061925584feab0db425	12	82	7	8
006dd93155bc2abd844cc5eed3a0fe7f	5	8896	5	1
00946f674d880be1f188abc10ad7cf46	8	8888	5	6
00946f674d880be1f188abc10ad7cf46	8	8888	5	6
00a0116ff15ff973ea16bee881208ae7	85	12	4	84
00bdcdda88e6b02977fc6ce3d412c600	6	106	2	6
011b142c9e082a5c1d10e0a88cd9c8e8	91	2	6	87

Query 2 (state-level aggregated analysis):

```
WITH t1 AS (
    SELECT
          o.order_id, c.customer_state AS state, DATE(o.order_purchase_timestamp) AS purchase_date,
          DATE(o.order_estimated_delivery_date) AS estimated_delivery_date,
          DATE(o.order_delivered_carrier_date) AS delivery_date_carrier,
          DATE(o.order_delivered_customer_date) AS delivery_date_customer,
          DATE(oi.shipping_limit_date) AS max_shipping_date
    FROM
          orders AS o
    INNER JOIN
          customers AS c
    ON
          o.customer_id = c.customer_id
    INNER JOIN
          order_items AS oi
    ON
          o.order_id = oi.order_id
), t2 AS (
    SELECT
          state, delivery_date_customer - purchase_date AS total_delivery_time,
          estimated_delivery_date - delivery_date_customer AS estimation_error,
          max_shipping_date - purchase_date AS max_shipping_delay,
          delivery_date_customer - delivery_date_carrier AS carrier_delivery_delay
    FROM
          t1
) SELECT
    state, ROUND(AVG(total_delivery_time)) AS avg_total_delivery_time, ROUND(AVG(estimation_error)) AS avg_estimation_error,
    ROUND(AVG(max_shipping_delay)) AS avg_max_shipping_delay, ROUND(AVG(carrier_delivery_delay)) AS avg_carrier_delivery_delay
FROM
    t2
GROUP BY
    state
ORDER BY
    avg_total_delivery_time DESC;
```

Output 2:

state	avg_total_delivery_time	avg_estimation_error	avg_max_shipping_delay	avg_carrier_delivery_delay
AP	736	278	126	725
SE	702	59	137	573
AC	651	354	18	642
MA	643	-33	131	585
AM	630	278	79	567
PB	575	195	196	488
PA	535	239	79	483
CE	492	160	139	415
PE	455	283	118	359

Inference:

- *We need improvements in our delivery estimation tool and delivery mechanism in general because of the time delay between purchase and delivery date.*
- *Detailed state-level analysis shed light on which states, relatively in particular, we need betterment of the delivery mechanism.*

b. Create columns:
   i. time_to_delivery = order_purchase_timestamp - order_delivered_customer_date

Query:

```
ALTER TABLE orders
ADD time_to_delivery INT AS (DATE(order_purchase_timestamp) - DATE(order_delivered_customer_date));
```

   ii. diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date

Query:

```
ALTER TABLE orders
ADD diff_estimated_delivery INT AS (DATE(order_estimated_delivery_date) - DATE(order_delivered_customer_date));
```

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

Query:

```
SELECT
    c.customer_state AS state,
    ROUND(AVG(oi.freight_value)) AS avg_order_freight_value,
    ROUND(AVG(o.time_to_delivery)) AS avg_time_to_delivery,
    ROUND(AVG(o.diff_estimated_delivery)) AS avg_diff_estimated_delivery
FROM
    customers AS c
INNER JOIN
    orders AS o
ON
    c.customer_id = o.customer_id
INNER JOIN
    order_items AS oi
ON
    o.order_id = oi.order_id
GROUP BY
    c.customer_state;
```

Output:

state	avg_order_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
SP	15	-140	270
RS	22	-352	219
SC	21	-317	207
BA	26	-440	202
MS	23	-235	213
RJ	21	-412	105
MG	21	-223	325
RO	41	-323	389
PR	21	-240	264
MT	28	-356	252

d. Sort the data to get the following:

i. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5.

Query 1 (highest):

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

Output 1:

state	avg_order_freight_value
RR	43
PB	43
RO	41
AC	40
PI	39

Query 2 (lowest):

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

Output 2:

state	avg_order_freight_value
SP	15
DF	21
MG	21
SC	21
PR	21

ii. Top 5 states with highest/lowest average time to delivery.

Query 1 (highest):

```
SELECT
    c.customer_state AS state, ROUND(AVG(o.time_to_delivery)) AS avg_time_to_delivery
FROM
    customers AS c
INNER JOIN
    orders AS o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    avg_time_to_delivery DESC LIMIT 5;
```

Output 1:

state	avg_time_to_delivery
SP	-144
MG	-225
PR	-253
MS	-263
DF	-270

Query 2 (lowest):

```
SELECT
    c.customer_state AS state, ROUND(AVG(o.time_to_delivery)) AS avg_time_to_delivery
FROM
    customers AS c
INNER JOIN
    orders AS o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    avg_time_to_delivery ASC LIMIT 5;
```

Output 2:

state	avg_time_to_delivery
AP	-736
AM	-698
MA	-647
AC	-619
PB	-610

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

Query 1 (fast w.r.t. estimated date):

```
SELECT
    c.customer_state AS state, ROUND(AVG(o.diff_estimated_delivery)) AS avg_diff_estimated_delivery
FROM
    customers AS c
INNER JOIN
    orders AS o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    avg_diff_estimated_delivery ASC LIMIT 5;
```

Output 1:

state	avg_diff_estimated_delivery
MA	-29
SE	61
AL	73
RJ	107
PI	132

Query 2 (not so fast w.r.t. estimated date):

```
SELECT
    c.customer_state AS state, ROUND(AVG(o.diff_estimated_delivery)) AS avg_diff_estimated_delivery
FROM
    customers AS c
INNER JOIN
    orders AS o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    avg_diff_estimated_delivery DESC LIMIT 5;
```

Output 2:

state	avg_diff_estimated_delivery
RO	429
AC	397
AP	327
MG	320
AM	303

6. Payment type analysis:

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

Explanation:

The analysis will help us understand the evolution of customers' preferred payment methods over time for the retailer and can help us in incentivizing the trend and generate more revenue and thus profit.

Query:

```
SELECT
    YEAR(o.order_purchase_timestamp) AS purchase_year,
    MONTH(o.order_purchase_timestamp) AS purchase_month,
    p.payment_type, COUNT(o.order_id)
FROM
    orders AS o
INNER JOIN
    payments AS p
ON
    o.order_id = p.order_id
GROUP BY
    YEAR(o.order_purchase_timestamp), MONTH(o.order_purchase_timestamp), p.payment_type
ORDER BY
    YEAR(o.order_purchase_timestamp), MONTH(o.order_purchase_timestamp), p.payment_type;
```

Output:

purchase_year	purchase_month	payment_type	num_orders
2016	9	credit_card	3
2016	10	credit_card	254
2016	10	debit_card	2
2016	10	UPI	63
2016	10	voucher	23
2016	12	credit_card	1
2017	1	credit_card	583
2017	1	debit_card	9
2017	1	UPI	197
2017	1	voucher	61
2017	2	credit_card	1356
2017	2	debit_card	13
2017	2	UPI	398

Inference:

Usage of Credit Cards by customers increased 20 folds over the period of 2 years at the retailer.

b. Distribution of payment installments and count of orders.

Explanation:

The analysis will help understand the payment cycle, and other things Finance - Accounts Receivable, Accounts Payable, Assets, Liabilities, etc.

Query:

```
SELECT
    payment_installments, COUNT(order_id) AS num_orders
FROM
    payments
GROUP BY
    payment_installments
ORDER BY
    payment_installments;

SELECT
    DISTINCT payment_installments,
    COUNT(order_id) OVER (PARTITION BY payment_installments) AS num_orders,
    COUNT(order_id) OVER (PARTITION BY payment_installments) * 100 / COUNT(order_id) OVER() AS percent_of_total_orders
FROM
    payments
ORDER BY
    payment_installments;
```

Output:

payment_installments	num_orders
0	2
1	52546
2	12413
3	10461
4	7098
5	5239
6	3920
7	1626

Inference:

More than 70% of total orders are paid off by customers within 3 installments.

Actionable Insights:

- *The popularity of **Target** grew multifold over time in Brazil with consistent growth in Order Volume and Revenue each year from 2016 to 2018.*
- *Two definitive seasonality trends exist in the dataset - one in March (significant increase in orders and revenue) and the other in December (significant drop in orders and revenue).*
- *Brazilians shop the most during Night hours (7 PM to 3 AM) followed by Afternoon hours (12 Noon to 4 PM) and this pattern is consistent over the years.*
- ***State - SP** leads all the states in terms of Order Volume, Revenue, Customer Base, and Order Cost followed by **RJ**.*
- ***City - Sau Paulo** leads all the cities in terms of Order Volume, Revenue, and Customer Base followed by **Rio De Janerio**.*
- *Order Cost (price + freight value) increased by a whopping 139% in 2018 from the 2017 level.*
- *Scope of improvement in delivery estimation tool due to highly incorrect estimations.*
- *Scope of improvement in delivery mechanism due to the time delay between purchase and delivery dates.*
- *Usage of Credit Cards by Brazilian Customers increased 20 folds over the period of 2 years.*
- *More than 70% of total orders are paid off by customers within 3 installments.*

Recommendations:

- *During the low sales seasonality trend in December, customers can be incentivized to make purchases at the store which will generate revenue for Target.*
- *Ease of checkout, smooth customer experience, and staff availability should be the prime objective of Target during the Night and Afternoon to maintain the influx of customers during this time. While promotional offers and extended services could be offered during other parts of the day to ramp up revenue.*
- *Customer surveys could be launched in those states and cities which are loss-making or contribute little towards Revenue and Sales to understand the customers' needs and act accordingly.*
- *Data Science consultants could be hired to improve the delivery estimation tool and use other data available at their disposal to put that into use to generate customer insights and drive business forward basis the data.*
- *Delivery mechanisms should be paid a fair amount of scrutiny to reduce the time taken to deliver orders. Reduction in delivery time can increase the Order Volume multi-folds this increasing Revenue and Profit.*
- *Tie-ups can be formed with different financial institutions to provide promotional offers on a wide range of credit cards functional in the country as an incentive, i.e., No Cost EMI, discounts on particular credit cards, etc.*