Project Name: Target SQL

Business case Analysis and Report with insights and Recommendations

Author: Moncy Kurien

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

1.1. Data type of columns in a table

There are 8 tables provided for the Target Brazil database. The tables are as follows with the data types of the fields:

1. 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

- Customer_id, customer_unique_id, customer_city and customer_state are Strings and customer_zip_code_prefix is Integer.
- All fields can be nulls.
- The customer_id and the customer_unique_id could be used as primary keys.

2. 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

- Geolocation_zip_code_prefix is an Integer, geolocation_lat and geolocation_lng are Floats, geolocation_city and geolocation_state are Strings.
- All fields can be nulls.

3. 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

- order_id, product_id and seller_id are Strings, order_item_id is an Integer, shipping_limit_date is a Timestamp and price and freight_value are Floats.
- All fields can be nulls.
- The order_id, product_id, seller_id can be used as the foreign keys.

4. 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

- Order_id, payment_type are Strings, payment_sequential and payment_installments are Integer and payment_value is a Float.
- All fields can be nulls.
- The order_id can be used as the foreign keys.

5. 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

- review_id, order_id and review_comment_title are Strings, review_score is an Integer and review_creation_date and review_answer_timestamp are Timestamps.
- The review_id can be used as a primary key.
- order_id can be used as the foreign keys.

6. 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

- order_id, customer_id and order_status are Strings, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date and order_estimated_delivery_date are Timestamps.
- Order_id can be used as the primary key.
- Customer_id can be used as a foreign key.

7. 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

- product_id and product_category are Strings, product_name_length, product_description_length, product_photos_qty, product_weight_g, product_length_cm, product_height_cm and product_width_cm are Integers.
- Product_id can be used as the primary key.

8. sellers table

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

- seller_id, seller_city and seller_state are Strings and seller_zip_codes_prefix is an Integer.
- Seller_id can be used as the primary key.

1.2. Time period for which the data is given

SQL query:

```
SELECT extract(date from min(order_purchase_timestamp)) as order_min_date, e
xtract(date from max(order_purchase_timestamp)) as order_max_date FROM
`target_brazil.orders`
```

Row	order_min_date	order_max_date
1	2016-09-04	2018-10-17

- Data is given for the time period of 2016-09-04 to 2018-10-17.

SQL query:

```
SELECT DATE_DIFF(extract(date from max(order_purchase_timestamp)),
extract(date from min(order_purchase_timestamp)),day) number_of_days
FROM `target_brazil.orders`
```

- In terms of number of days, we have 773 days of data.

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

- States of customers ordered during the given period are as follows:

SQL query:

```
SELECT distinct c.customer_state
FROM `target_brazil.customers` c
join `target_brazil.orders` o
on o.customer_id = c.customer_id
order by c.customer_state
```

Row	customer_state
1	AC
2	AL
3	AM
4	AP
5	BA
6	CE
7	DF
8	ES
9	GO
10	MA
11	MG
12	MS
13	MT
14	PA
15	PB
16	PE
17	PI
18	PR
19	RJ
20	RN
21	RO
22	RR
23	RS
24	SC
25	SE
26	SP
27	TO

- There are 27 states from which customers have made online purchases from Brazil.

Cities of customers ordered during the given period are as follows:

SQL query:

```
SELECT distinct c.customer_city
FROM `target_brazil.customers` c
join `target_brazil.orders` o
on o.customer_id = c.customer_id
order by c.customer_city
```

Row	customer_city
1	abadia dos dourados
2	abadiania
3	abaete
4	abaetetuba
5	abaiara
6	abaira
7	abare
8	abatia
9	abdon batista
10	abelardo luz
11	abrantes
12	abre campo
13	abreu e lima
14	acaiaca
15	acailandia
16	acajutiba
17	acarau
18	acari
19	acegua
20	acopiara

- There are 4119 cities from which customers have placed orders from Target Brazil.

2. In-depth Exploration:

2.1. 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?

SQL query:

```
SELECT Extract(year from order_purchase_timestamp) as orders_year,
count(order_id) as Number_of_orders_placed
FROM `target_brazil.orders`
group by orders_year
order by orders_year
```

Row	orders_year	Number_of_orders_placed
1	2016	329
2	2017	45101
3	2018	54011

- We can see that over the years, the number of online orders has increased. We can see a huge jump in the online orders from 2016 to 2017. However, this is because we have data only for 4 months in 2016 (2016-09-04 onwards) and we have a whole year's data for 2017.

Year and month trend for the number of orders placed.

SQL query:

```
SELECT distinct Extract(year from order_purchase_timestamp) as orders_year,
Extract(month from order_purchase_timestamp) as orders_month,
count(order_id) as Number_of_orders_placed
FROM `target_brazil.orders`
group by orders_year, orders_month
order by orders_year, orders_month
```

Row	orders_year	orders_month	Number_of_orders_placed
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
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544
15	2017	12	5673
16	2018	1	7269
17	2018	2	6728
18	2018	3	7211
19	2018	4	6939
20	2018	5	6873
21	2018	6	6167
22	2018	7	6292
23	2018	8	6512
24	2018	9	16
25	2018	10	4

- There appears to be some seasonality in the data. We can compare the number of orders placed at a specific month in a year with the previous months and the next months. In this query, I will compare with 2 months previous and 2 months after.

Checking seasonality in months for each year:

SQL Query:

```
select *
from(select orders_year, orders_month, orders_month_name
lag(number_of_orders,2) over(partition by orders_year order by orders_mon
th) as orders_placed_2months_back,
lag(number_of_orders,1) over(partition by orders_year order by orders_mon
th) as orders_placed_prev_month,
number_of_orders as orders_placed_curr_month,
lead(number_of_orders, 1) over(partition by orders_year order by orders_m
onth) as orders_placed_next_month,
lead(number_of_orders, 2) over(partition by orders_year order by orders_m
onth) as orders_placed_2months_after

from(SELECT distinct Extract(year from order_purchase_timestamp) as order
s_year,Extract(month from order_purchase_timestamp) as orders_month,
format_date("%b", order_purchase_timestamp) as orders_month_name,
count(order_id) as Number_of_orders
FROM `target_brazil.orders`
group by orders_year, orders_month, orders_month_name
```

```
order by orders_year, orders_month) a) b

where (b.orders_placed_2months_back < b.orders_placed_prev_month and b.or
ders_placed_prev_month > b.orders_placed_curr_month)
or (b.orders_placed_prev_month < b.orders_placed_curr_month and b.orders_
placed_curr_month > b.orders_placed_next_month)
or (b.orders_placed_curr_month < b.orders_placed_next_month and b.orders_
placed_next_month > b.orders_placed_2months_after)

order by orders_year
```

Row	orders_year	orders_month	orders_month_name	orders_placed_2months_back	orders_placed_prev_month	orders_placed_curr_month	orders_placed_next_month	orders_placed_2months_
1	2016	9	Sep	*null*	*null*	4	324	1
2	2016	10	Oct	*null*	4	324	1	*null*
3	2016	12	Dec	4	324	1	*null*	*null*
4	2017	2	Feb	*null*	800	1780	2682	2404
5	2017	3	Mar	800	1780	2682	2404	3700
6	2017	4	Apr	1780	2682	2404	3700	3245
7	2017	5	May	2682	2404	3700	3245	4026
8	2017	6	Jun	2404	3700	3245	4026	4331
9	2017	7	Jul	3700	3245	4026	4331	4285
10	2017	8	Aug	3245	4026	4331	4285	4631
11	2017	9	Sep	4026	4331	4285	4631	7544
12	2017	10	Oct	4331	4285	4631	7544	5673
13	2017	11	Nov	4285	4631	7544	5673	*null*
14	2017	12	Dec	4631	7544	5673	*null*	*null*
15	2018	2	Feb	*null*	7269	6728	7211	6939
16	2018	3	Mar	7269	6728	7211	6939	6873
17	2018	4	Apr	6728	7211	6939	6873	6167
18	2018	7	Jul	6873	6167	6292	6512	16
19	2018	8	Aug	6167	6292	6512	16	4
20	2018	9	Sep	6292	6512	16	4	*null*

- From the results, we can say that there is seasonality in the data.
- In the year 2016, we have data only for Sep, Oct and Dec. Since Nov data is not available, we can skip that. The month Oct shows an increase in the number of orders than in Sep.
- In the year 2017, we can see an increase from Jan through Mar and then the trend goes down in Apr. Again, the trend goes up in May and again the trend goes up and down.
- In 2018 also we can see an up and down trend indicating a seasonality in the data.

Pulling only the peak months in each year which had the highest number of orders.

SQL query:

```
select b.orders_year,
b.orders_month,
b.orders_month_name,
b.orders_placed_curr_month

from(select orders_year,
orders_month,orders_month_name,
lag(number_of_orders,2) over(partition by orders_year order by orders_
month) as orders_placed_2months_back,
```

```
lag(number_of_orders,1) over(partition by orders_year order by orders_
month) as orders_placed_prev_month,
number_of_orders as orders_placed_curr_month,
lead(number_of_orders, 1) over(partition by orders_year order by order
s_month) as orders_placed_next_month,
lead(number_of_orders, 2) over(partition by orders_year order by order
s_month) as orders_placed_2months_after

      from(SELECT distinct Extract(year from order_purchase_timestamp
      ) as orders_year,
    Extract(month from order_purchase_timestamp) as orders_month,
    format_date("%b", order_purchase_timestamp) as orders_month_name,
    count(order_id) as Number_of_orders
    FROM `target_brazil.orders`
    group by orders_year, orders_month,orders_month_name
    order by orders_year, orders_month) a) b

where b.orders_placed_prev_month < b.orders_placed_curr_month
and b.orders_placed_curr_month > b.orders_placed_next_month

order by orders_year
```

Row	orders_year	orders_month	orders_month_name	orders_placed_curr_month
1	2016	10	Oct	324
2	2017	3	Mar	2682
3	2017	5	May	3700
4	2017	8	Aug	4331
5	2017	11	Nov	7544
6	2018	3	Mar	7211
7	2018	8	Aug	6512

- The above result shows that in 2016 there was a peak in Oct.
- In 2017, there were peaks in March, May, August and November.
- In 2018, there were peaks in March and August

Checking seasonality in overall months irrespective of the years.

SQL query:

```
select *
from(select orders_month, orders_month_name,
lag(number_of_orders,2) over(order by orders_month) as orders_placed_2months
_back,
lag(number_of_orders,1) over(order by orders_month) as orders_placed_prev_mo
nth,
number_of_orders as orders_placed_curr_month,
lead(number_of_orders, 1) over(order by orders_month) as orders_placed_next_
month,
lead(number_of_orders, 2) over(order by orders_month) as orders_placed_2mont
hs_after


from(SELECT distinct Extract(month from order_purchase_timestamp) as orders_
month,
```

```
        format_date("%b", order_purchase_timestamp) as orders_month_name,
        count(order_id) as Number_of_orders
        FROM `target_brazil.orders`
        group by orders_month, orders_month_name
        order by orders_month) a) b

where (b.orders_placed_2months_back < b.orders_placed_prev_month and b.order
s_placed_prev_month > b.orders_placed_curr_month)
or (b.orders_placed_prev_month < b.orders_placed_curr_month and b.orders_pla
ced_curr_month > b.orders_placed_next_month)
or (b.orders_placed_curr_month < b.orders_placed_next_month and b.orders_pla
ced_next_month > b.orders_placed_2months_after)

order by orders_month
```

Row	orders_month	orders_month_name	orders_placed_2months_back	orders_placed_prev_month	orders_placed_curr_month	orders_placed_next_month	orders_placed_2months_after
1	2	Feb	*null*	8069	8508	9893	9343
2	3	Mar	8069	8508	9893	9343	10573
3	4	Apr	8508	9893	9343	10573	9412
4	5	May	9893	9343	10573	9412	10318
5	6	Jun	9343	10573	9412	10318	10843
6	7	Jul	10573	9412	10318	10843	4305
7	8	Aug	9412	10318	10843	4305	4959
8	9	Sep	10318	10843	4305	4959	7544
9	10	Oct	10843	4305	4959	7544	5674
10	11	Nov	4305	4959	7544	5674	*null*
11	12	Dec	4959	7544	5674	*null*	*null*

- The result clearly shows seasonality of number of orders with respect to months irrespective of the year.
- The number of orders is increasing from Jan through Mar and then drops down in April then again to increase in May.
- Likewise there are again cyclic trends in the following months.

Pulling only the peak months which had the highest number of orders irrespective of the years.

SQL query:

```
select orders_month, orders_month_name, orders_placed_curr_month

from(select orders_month, orders_month_name,
lag(number_of_orders,2) over(order by orders_month) as orders_placed_2months
_back,
lag(number_of_orders,1) over(order by orders_month) as orders_placed_prev_mo
nth,
number_of_orders as orders_placed_curr_month,
lead(number_of_orders, 1) over(order by orders_month) as orders_placed_next_
month,
lead(number_of_orders, 2) over(order by orders_month) as orders_placed_2mont
hs_after

        from(SELECT distinct Extract(month from order_purchase_timestamp) as o
        rders_month,
        format_date("%b", order_purchase_timestamp) as orders_month_name,
        count(order_id) as Number_of_orders
```

```
        FROM `target_brazil.orders`
        group by orders_month, orders_month_name
        order by orders_month) a) b

where b.orders_placed_prev_month < b.orders_placed_curr_month
and b.orders_placed_curr_month > b.orders_placed_next_month

order by orders_month
```

Row	orders_month	orders_month_name	orders_placed_curr_month
1	3	Mar	9893
2	5	May	10573
3	8	Aug	10843
4	11	Nov	7544

- The peaks are in the months of Mar, May, Aug and Nov in the overall data.

Overall top 5 months with high number of orders:

SQL query:

```
SELECT distinct Extract(month from order_purchase_timestamp) as orders
_month,
format_date("%b", order_purchase_timestamp) as month_name,
count(order_id) as Number_of_orders_placed
FROM `target_brazil.orders`
group by orders_month,month_name
order by Number_of_orders_placed desc limit 5
```

Row	orders_month	month_name	Number_of_orders_placed
1	8	Aug	10843
2	5	May	10573
3	7	Jul	10318
4	3	Mar	9893
5	6	Jun	9412

- The top 5 months that have the highest number of orders are Aug, May, Jul which have more than 10k and Mar & Jun between 9k to 10k orders.

Overall bottom 5 months in terms of number of orders:

SQL query:

```
SELECT distinct Extract(month from order_purchase_timestamp) as orders
_month,
format_date("%b", order_purchase_timestamp) as month_name,
count(order_id) as Number_of_orders_placed
FROM `target_brazil.orders`
group by orders_month,month_name
order by Number_of_orders_placed limit 5
```

Row	orders_month	month_name	Number_of_orders_placed
1	9	Sep	4305
2	10	Oct	4959
3	12	Dec	5674
4	11	Nov	7544
5	1	Jan	8069

- The least number of orders are placed in the months of Sep, Oct, Dec, Nov and Jan

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

SQL Query:

```
SELECT extract(hour from order_purchase_timestamp) as hour, count(order_id)
as number_of_orders_placed FROM `euphoric-quanta-
381315.target_brazil.orders`
group by hour
order by hour
```

Row	hour	number_of_orders_placed
1	0	2394
2	1	1170
3	2	510
4	3	272
5	4	206
6	5	188
7	6	502
8	7	1231
9	8	2967
10	9	4785
11	10	6177
12	11	6578
13	12	5995
14	13	6518
15	14	6569
16	15	6454
17	16	6675
18	17	6150
19	18	5769
20	19	5982
21	20	6193
22	21	6217
23	22	5816
24	23	4123

- From the above result, we can see that the number of orders in the early part of the days is lesser than the later part.
- We can consider hours 1 to 6 as dawn, 7 to 12 as morning, 13 to 18 afternoon and 19 to 24 and o as night and see when the people in Brazil are placing the orders.

SQL Query: Including cancelled orders

```
select
case
  when hour between 1 and 6 then "1 to 6 - dawn"
  when hour between 7 and 12 then "7 to 12 - morning"
  when hour between 13 and 18 then "13 to 18 - afternoon"
  else "19 to 24 and 0 - night"
end as time_of_the_day,
sum(number_of_orders_placed) as number_of_orders_placed

from(SELECT extract(hour from order_purchase_timestamp) as hour,
       count(order_id) as number_of_orders_placed
     FROM `euphoric-quanta-381315.target_brazil.orders`
     group by hour
     order by hour)
group by time_of_the_day
order by time_of_the_day
```

Row	time_of_the_day	number_of_orde
1	1 to 6 - dawn	2848
2	13 to 18 - afternoon	38135
3	19 to 24 and 0 - night	30725
4	7 to 12 - morning	27733

- Most of the orders are placed in the afternoon followed by night.
- Least number of orders are placed at dawn.

SQL Query: Excluding the cancelled orders

```
select
case
  when hour between 1 and 6 then "1 to 6 - dawn"
  when hour between 7 and 12 then "7 to 12 - morning"
  when hour between 13 and 18 then "13 to 18 - afternoon"
  else "19 to 24 and 0 - night"
end as time_of_the_day,
sum(number_of_orders_placed) as number_of_orders_placed
from(
SELECT extract(hour from order_purchase_timestamp) as hour, count(order_id)
as number_of_orders_placed
              FROM `euphoric-quanta-381315.target_brazil.orders`
              where order_status != "canceled"
              group by hour
              order by hour)
group by time_of_the_day
order by time_of_the_day
```

Row	time_of_the_day	number_of_orde
1	1 to 6 - dawn	2826
2	13 to 18 - afternoon	37895
3	19 to 24 and 0 - night	30548
4	7 to 12 - morning	27547

- Based on the above result, it is similar as when we included the cancelled orders previously.

Orders placed in what time do get cancelled mostly?

SQL Query:

```
select a.time_of_the_day,
a.number_of_orders_placed as number_of_orders_cancelled,
b.number_of_orders_placed,
round(a.number_of_orders_placed / b.number_of_orders_placed * 100,3) as perc
entage_of_orders_cancelled
from (select
      case
        when hour between 1 and 6 then "1 to 6 - dawn"
        when hour between 7 and 12 then "7 to 12 - morning"
        when hour between 13 and 18 then "13 to 18 - afternoon"
        else "19 to 24 and 0 - night"
      end as time_of_the_day,
      sum(number_of_orders_placed) as number_of_orders_placed
      from(
      SELECT extract(hour from order_purchase_timestamp) as hour, count(orde
r_id) as number_of_orders_placed
                    FROM `euphoric-quanta-381315.target_brazil.orders`
                    where order_status = "canceled"
                    group by hour
                    order by hour)
      group by time_of_the_day
      order by time_of_the_day) a
join (select
        case
          when hour between 1 and 6 then "1 to 6 - dawn"
          when hour between 7 and 12 then "7 to 12 - morning"
          when hour between 13 and 18 then "13 to 18 - afternoon"
          else "19 to 24 and 0 - night"
        end as time_of_the_day,
        sum(number_of_orders_placed) as number_of_orders_placed

        from(SELECT extract(hour from order_purchase_timestamp) as hour,
        count(order_id) as number_of_orders_placed
            FROM `euphoric-quanta-381315.target_brazil.orders`
            group by hour
            order by hour)
        group by time_of_the_day order by time_of_the_day) b
on a.time_of_the_day = b.time_of_the_day
```

Row	time_of_the_day	number_of_orders_cancelled	number_of_orders_placed	percentage_of_orders_cancelled
1	7 to 12 - morning	186	27733	0.671
2	19 to 24 and 0 - night	177	30725	0.576
3	13 to 18 - afternoon	240	38135	0.629
4	1 to 6 - dawn	22	2848	0.772

- From the above result, we can see that most percentage of the cancelled orders were placed between 1 hour and 6 hour(dawn) – 0.772%
- The least percentage of orders that were cancelled were ordered at night (19 to 24 and 0 hours) - 0.576%.

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

3.1. Get month on month orders by states

SQL Query: Month on month orders by states irrespective of year

```
select customer_state, month_name,
sum(a.number_of_orders_placed) total_number_of_orders_placed
from(select extract(month from o.order_purchase_timestamp) month,
   format_datetime("%b", order_purchase_timestamp) as month_name,
   c.customer_state,
   count(order_id) as number_of_orders_placed
   from `target_brazil.orders`o
   join `target_brazil.customers`c
   on c.customer_id = o.customer_id
   group by month,month_name, c.customer_state
   order by month, c.customer_state) a
group by customer_state, month,a.month_name
order by customer_state,a.month
```

Row	customer_state	month_name	total_number_of_orders_placed
1	AC	Jan	8
2	AC	Feb	6
3	AC	Mar	4
4	AC	Apr	9
5	AC	May	10
6	AC	Jun	7
7	AC	Jul	9
8	AC	Aug	7
9	AC	Sep	5
10	AC	Oct	6
11	AC	Nov	5
12	AC	Dec	5
13	AL	Jan	39
14	AL	Feb	39
15	AL	Mar	40
16	AL	Apr	51
17	AL	May	46

SQL Query: Month on month orders by states for each year

```
select year, customer_state, month_name,
sum(a.number_of_orders_placed) total_number_of_orders_placed
from(select extract(year from o.order_purchase_timestamp) year,
       extract(month from o.order_purchase_timestamp) month,
       format_datetime("%b", order_purchase_timestamp) as month_name,
       c.customer_state,
       count(order_id) as number_of_orders_placed
       from `target_brazil.orders`o
       join `target_brazil.customers`c
       on c.customer_id = o.customer_id
       group by year,month,month_name, c.customer_state
       order by year,month, c.customer_state) a
group by customer_state,a.year, month,a.month_name
order by customer_state,a.year,a.month
```

Row	year	customer_state	month_name	total_number_of
1	2017	AC	Jan	2
2	2017	AC	Feb	3
3	2017	AC	Mar	2
4	2017	AC	Apr	5
5	2017	AC	May	8
6	2017	AC	Jun	4
7	2017	AC	Jul	5
8	2017	AC	Aug	4
9	2017	AC	Sep	5
10	2017	AC	Oct	6
11	2017	AC	Nov	5
12	2017	AC	Dec	5
13	2018	AC	Jan	6
14	2018	AC	Feb	3
15	2018	AC	Mar	2
16	2018	AC	Apr	4
17	2018	AC	May	2
18	2018	AC	Jun	3
19	2018	AC	Jul	4
20	2018	AC	Aug	3
21	2016	AL	Oct	2

Just as an analysis, looking at the number of states from which the customers are placing orders month on month for each year

SQL Query – Fetching the number of stated from where customers have ordered month by month for each year

```
select year, month_name,
count(customer_state) number_of_states_customers,
sum(a.number_of_orders_placed) total_number_of_orders_placed
from(
select extract(year from o.order_purchase_timestamp) year,
extract(month from o.order_purchase_timestamp) month,
format_datetime("%b", order_purchase_timestamp) as month_name,
c.customer_state,
count(order_id) as number_of_orders_placed
from `target_brazil.orders`o
join `target_brazil.customers`c
on c.customer_id = o.customer_id
group by year,month,month_name, c.customer_state
order by year,month, c.customer_state) a
group by a.year, month,a.month_name
order by a.year,a.month
```

Row	year	month_name	number_of_states_customers	total_number_of_orders_placed
1	2016	Sep	3	4
2	2016	Oct	21	324
3	2016	Dec	1	1
4	2017	Jan	24	800
5	2017	Feb	27	1780
6	2017	Mar	27	2682
7	2017	Apr	26	2404
8	2017	May	27	3700
9	2017	Jun	27	3245
10	2017	Jul	27	4026
11	2017	Aug	26	4331
12	2017	Sep	27	4285
13	2017	Oct	27	4631
14	2017	Nov	27	7544
15	2017	Dec	26	5673
16	2018	Jan	27	7269
17	2018	Feb	27	6728
18	2018	Mar	27	7211
19	2018	Apr	27	6939
20	2018	May	27	6873
21	2018	Jun	27	6167
22	2018	Jul	27	6292
23	2018	Aug	26	6512
24	2018	Sep	4	16
25	2018	Oct	3	4

SQL Query – Fetching the number of orders placed from each states year on year.

```
select customer_state,  year,
sum(a.number_of_orders_placed) total_number_of_orders_placed
from(
select extract(year from o.order_purchase_timestamp) year,
c.customer_state,
count(order_id) as number_of_orders_placed
from `target_brazil.orders`o
join `target_brazil.customers`c
on c.customer_id = o.customer_id
group by year, c.customer_state
order by year, c.customer_state) a
group by customer_state,a.year
order by customer_state,a.year
```

Row	customer_state	year	total_number_of_orders_placed
1	AC	2017	54
2	AC	2018	27
3	AL	2016	2
4	AL	2017	207
5	AL	2018	204
6	AM	2017	75
7	AM	2018	73
8	AP	2017	29
9	AP	2018	39
10	BA	2016	4
11	BA	2017	1592
12	BA	2018	1784
13	CE	2016	8
14	CE	2017	660
15	CE	2018	668
16	DF	2016	6
17	DF	2017	921
18	DF	2018	1213
19	ES	2016	4
20	ES	2017	968
21	ES	2018	1061
22	GO	2016	9
23	GO	2017	955
24	GO	2018	1056
25	MA	2016	4
26	MA	2017	387
27	MA	2018	356
28	MG	2016	40
29	MG	2017	5414
30	MG	2018	6181
31	MS	2017	301
32	MS	2018	414
33	MT	2016	3
34	MT	2017	420
35	MT	2018	484
36	PA	2016	4
37	PA	2017	504
38	PA	2018	467
39	PB	2016	1
40	PB	2017	260
41	PB	2018	275
42	PE	2016	7
43	PE	2017	772
44	PE	2018	873
45	PI	2016	1
46	PI	2017	226
47	PI	2018	268
48	PR	2016	20
49	PR	2017	2270
50	PR	2018	2755
51	RJ	2016	56
52	RJ	2017	6225
53	RJ	2018	6571
54	RN	2016	4
55	RN	2017	234
56	RN	2018	247
57	RO	2017	141
58	RO	2018	112

3.2. Distribution of customers across the states in Brazil

SQL query:

```
SELECT customer_state,

count(customer_id) as number_of_customers
FROM `target_brazil.customers`
group by customer_state
order by number_of_customers desc
```

Row	customer_state	number_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
16	MS	715
17	PB	536
18	PI	495
19	RN	485
20	AL	413
21	SE	350
22	TO	280
23	RO	253
24	AM	148
25	AC	81
26	AP	68
27	RR	46

- Highest number of customers are from the state ‘SP.
- The least number of customers are from the state ‘RR’.

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

4.1. 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

SQL Query:

```
select *,
round(((cost_of_orders - lag(cost_of_orders, 1) over(order by cost_of_orders
))/ lag(cost_of_orders, 1) over(order by cost_of_orders)) * 100,2) as percen
tage_increase_from_previous_year,
round(((avg_cost_of_orders - lag(avg_cost_of_orders, 1) over(order by avg_co
st_of_orders))/ lag(avg_cost_of_orders, 1) over(order by avg_cost_of_orders)
) * 100,2) as percentage_increase_from_previous_year_on_avg

from(SELECT extract(year from o.order_purchase_timestamp) as order_year,
       sum(p.payment_value) cost_of_orders,
       avg(p.payment_value) avg_cost_of_orders
       from `target_brazil.orders` o
       join `target_brazil.payments` p
       on o.order_id = p.order_id
       where extract(month from o.order_purchase_timestamp) between 01 and 08
       group by order_year
       order by order_year) x

order by order_year
```

Row	order_year	cost_of_orders	avg_cost_of_orders	percentage_increase_from_previous_year	percentage_increase_from_previous_year_on_avg
1	2017	3669022.1200000118	150.42524373744448	*null*	*null*
2	2018	8694733.83999979	155.27696830073944	136.98	3.23

- There is an increase in the total cost of orders by 136.9% from 2017 to 2018.
- There is an increase in the average cost of orders by 3.23% from 2017 to 2018.

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

SQL Query:

```
select c.customer_state,
round(avg(oi.price),2) as mean_price,
round(sum(oi.price),2) as sum_price,
round(avg(oi.freight_value),2) as mean_freight_value,
round(sum(oi.freight_value),2) as sum_freight_value
from `target_brazil.customers` c
join `target_brazil.orders` o
on o.customer_id = c.customer_id
join `target_brazil.order_items` oi
on oi.order_id = o.order_id
group by c.customer_state
order by mean_price desc
```

Row	customer_state	mean_price	sum_price	mean_freight_value	sum_freight_value
1	PB	191.48	115268.08	42.72	25719.73
2	AL	180.89	80314.81	35.84	15914.59
3	AC	173.73	15982.95	40.07	3686.75
4	RO	165.97	46140.64	41.07	11417.38
5	PA	165.69	178947.81	35.83	38699.3
6	AP	164.32	13474.3	34.01	2788.5
7	PI	160.36	86914.08	39.15	21218.2
8	TO	157.53	49621.74	37.25	11732.68
9	RN	156.97	83034.98	35.65	18860.1
10	CE	153.76	227254.71	32.71	48351.59
11	SE	153.04	58920.85	36.65	14111.47
12	RR	150.57	7829.43	42.98	2235.19
13	MT	148.3	156453.53	28.17	29715.43
14	PE	145.51	262788.03	32.92	59449.66
15	MA	145.2	119648.22	38.26	31523.77
16	MS	142.63	116812.64	23.37	19144.03
17	AM	135.5	22356.84	33.21	5478.89
18	BA	134.6	511349.99	26.36	100156.68
19	GO	126.27	294591.95	22.77	53114.98
20	DF	125.77	302603.94	21.04	50625.5
21	RJ	125.12	1824092.67	20.96	305589.31
22	SC	124.65	520553.34	21.47	89660.26
23	ES	121.91	275037.31	22.06	49764.6
24	MG	120.75	1585308.03	20.63	270853.46
25	RS	120.34	750304.02	21.74	135522.74
26	PR	119.0	683083.76	20.53	117851.68
27	SP	109.65	5202955.05	15.15	718723.07

- The state of ‘PB’ has the highest mean price and the state of SP has the lowest mean price.
- The state of ‘SP’ has the highest total price and the state ‘RR’ has the lowest total price. (Can be obtained by using sum_price in the order by clause in the query with ‘desc’).

- The state of ‘RR’ has the highest mean freight value and the state of SP has the lowest mean freight value (can be obtained by using mean_freight_value in the order by clause in the query with ‘desc’).
- The state of ‘SP’ has the highest total freight values and the state ‘RR’ has the lowest total freight value. (Can be obtained by using sum_freight_value in the order by clause in the query with ‘desc’).

5. Analysis on sales, freight and delivery time

5.1. Calculate days between purchasing, delivering and estimated delivery

Query:

```
SELECT distinct order_id, extract(date from order_purchase_timestamp)
as purchase_date,
extract(date from order_estimated_delivery_date) estimated_delivery,
extract(date from order_delivered_customer_date) customer_delivery_date,
date_diff(order_estimated_delivery_date,order_purchase_timestamp, day)
as estimated_days_to_delivery,
date_diff(order_delivered_customer_date,order_purchase_timestamp, day) as
actual_days_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)
as diff_estimated_actual_delivery

from `target_brazil.orders`

where order_delivered_customer_date is not null

order by diff_estimated_actual_delivery
```

Row	order_id	purchase_date	estimated_delive	customer_delive	estimated_days_to_delivery	actual_days_to_delivery	diff_estimated_actual_delivery
1	1b3190b2dfa9d789e1f14c05b...	2018-02-23	2018-03-15	2018-09-19	19	208	-188
2	ca07593549f1816d26a572e06...	2017-02-21	2017-03-22	2017-09-19	28	209	-181
3	47b40429ed8cce3aee9199792...	2018-01-03	2018-01-19	2018-07-13	15	191	-175
4	2fe324febf907e3ea3f2aa9650...	2017-03-13	2017-04-05	2017-09-19	22	189	-167
5	285ab9426d6982034523a855f...	2017-03-08	2017-04-06	2017-09-19	28	194	-166
6	440d0d17af552815d15a9e41a...	2017-03-07	2017-04-07	2017-09-19	30	195	-165
7	c27815f7e3dd0b926b5855262...	2017-03-15	2017-04-10	2017-09-19	25	187	-162
8	0f4519c5f1c541ddec9f21b3bd...	2017-03-09	2017-04-11	2017-09-19	32	194	-161
9	d24e8541128cea179a11a6517...	2017-06-12	2017-06-26	2017-12-04	13	175	-161
10	2d7561026d542c8dbd8f0daea...	2017-03-15	2017-04-13	2017-09-19	28	188	-159
11	6e82dcfb5eada6283dba34f16...	2017-05-17	2017-06-14	2017-11-16	27	182	-155
12	2fb597c2f772eca01b1f5c561b...	2017-03-08	2017-04-17	2017-09-19	39	194	-155
13	dfe5f68118c2576143240b8d7...	2017-03-17	2017-04-19	2017-09-19	32	186	-153
14	ed8e9faf1b75f43ee027103957...	2017-11-29	2017-12-19	2018-05-21	19	173	-153
15	2ba1366baecad3c3536f27546...	2017-02-28	2017-03-29	2017-08-28	28	181	-152

- Since there are null values in order_delivered_customer_date, fetching records that are not null.

Looking for more details with the following query.

Query:

```
SELECT distinct order_id,
extract(date from order_purchase_timestamp) purchase_date,
extract(date from order_estimated_delivery_date) estimated_delivery,
extract(date from order_delivered_customer_date) customer_delivery_date,
date_diff(order_estimated_delivery_date,order_purchase_timestamp, day) estim
ated_days_to_delivery,
date_diff(order_delivered_customer_date,order_purchase_timestamp, day) actua
l_days_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)
diff_estimated_actual_delivery
from `target_brazil.orders`
where order_delivered_customer_date is not null
and date_diff(extract(date from order_estimated_delivery_date),extract(date
from order_delivered_customer_date), day) < 0
order by diff_estimated_actual_delivery
```

Row	order_id	purchase_date	estimated_deliv	customer_delive	estimated_days_	actual_days_to_	diff_estimated_a
1	1b3190b2dfa9d789e1f14c05b...	2018-02-23	2018-03-15	2018-09-19	19	208	-188
2	ca07593549f1816d26a572e06...	2017-02-21	2017-03-22	2017-09-19	28	209	-181
3	47b40429ed8cce3aee9199792...	2018-01-03	2018-01-19	2018-07-13	15	191	-175
4	2fe324febf907e3ea3f2aa9650...	2017-03-13	2017-04-05	2017-09-19	22	189	-167
5	285ab9426d6982034523a855f...	2017-03-08	2017-04-06	2017-09-19	28	194	-166
6	440d0d17af552815d15a9e41a...	2017-03-07	2017-04-07	2017-09-19	30	195	-165
7	c27815f7e3dd0b926b5855262...	2017-03-15	2017-04-10	2017-09-19	25	187	-162
8	0f4519c5f1c541ddec9f21b3bd...	2017-03-09	2017-04-11	2017-09-19	32	194	-161
9	d24e8541128cea179a11a6517...	2017-06-12	2017-06-26	2017-12-04	13	175	-161
10	2d7561026d542c8dbd8f0daea...	2017-03-15	2017-04-13	2017-09-19	28	188	-159
11	6e82dcfb5eada6283dba34f16...	2017-05-17	2017-06-14	2017-11-16	27	182	-155
12	2fb597c2f772eca01b1f5c561b...	2017-03-08	2017-04-17	2017-09-19	39	194	-155
13	dfe5f68118c2576143240b8d7...	2017-03-17	2017-04-19	2017-09-19	32	186	-153
14	ed8e9faf1b75f43ee027103957...	2017-11-29	2017-12-19	2018-05-21	19	173	-153

- There are about 6535 order that were delayed more than the estimated delivery date. These days are shown in negatives in the diff_estimated_actual_delivery.
- About 40 deliveries have been delayed for 100 or more days, about 121 orders were delayed at least 50 days from the estimated delivery date.
- At least 2300 orders have been delayed at a double digit days(10 or more days).
- There are some null values in order_delivered_customer_date.

Checking the order_status for records with order_delivered_customer_date is null.

SQL Query

```
SELECT distinct order_status
from `target_brazil.orders`
where order_delivered_carrier_date is null
```

Row	order_status
1	created
2	shipped
3	approved
4	canceled
5	invoiced
6	delivered
7	processing
8	unavailable

- Looks like there are some orders that are in delivered status but without a delivery date.

Query to get the records that are in delivered status but without a delivery date is as follows.

SQL Query:

```
SELECT distinct * FROM `euphoric-quanta-381315.target_brazil.orders`
where order_delivered_customer_date is null
and order_status = 'delivered'
```

Row	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_da	order_delivered_customer_date	order_
1	0d3268bad9...	4f1d63d35fb7c8...	delivered	2018-07-01 21:14:02 U...	2018-07-01 21:...	2018-07-03 09:28:00 ...	null	2018-(
2	f5dd62b788...	5e89028e024b3...	delivered	2018-06-20 06:58:43 U...	2018-06-20 07:...	2018-06-25 08:05:00 ...	null	2018-(
3	20edc82cf54...	28c37425f1127...	delivered	2018-06-27 16:09:12 U...	2018-06-27 16:...	2018-07-03 19:26:00 ...	null	2018-(
4	2d1e2d5bf4...	ec05a6d8558c6...	delivered	2017-11-28 17:44:07 U...	2017-11-28 17:...	2017-11-30 18:12:23 ...	null	2017-
5	2ebdfc4f15f...	29f0540231702f...	delivered	2018-07-01 17:05:11 U...	2018-07-01 17:...	2018-07-03 13:57:00 ...	null	2018-(
6	e69f75a717...	cfda40ca8dd0a5...	delivered	2018-07-01 22:05:55 U...	2018-07-01 22:...	2018-07-03 13:57:00 ...	null	2018-(
7	2d858f4513...	e08caf668d499a...	delivered	2017-05-25 23:22:43 U...	2017-05-25 23:...	null	null	2017-(
8	ab7c89dc1bf...	dd1b84a7286eb...	delivered	2018-06-08 12:09:39 U...	2018-06-08 12:...	2018-06-12 14:10:00 ...	null	2018-(

- There are about 8 orders that show as delivered but do not have a delivery date. These need to be analysed - why these orders are marked as delivered yet do not have delivery date in the system.

5.2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

5.2.1. time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

5.2.2. diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

- For the above two questions, the following query can be used. Since there are null values in order_delivery_customer_date, the where clause is used only to fetch non null records.

SQL Query:

```
SELECT distinct order_id,
order_purchase_timestamp,
order_delivered_customer_date,
date_diff(order_purchase_timestamp, order_delivered_customer_date, day) time
_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)
diff_estimated_delivery
from `target_brazil.orders`
where order_delivered_customer_date is not null
order by time_to_delivery
limit 10
```

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	time_to_delivery	diff_estimated_delivery
1	ca07593549f1816d26a572e0...	2017-02-21 23:31:27 UTC	2017-09-19 14:36:39 UTC	-209	-181
2	1b3190b2dfa9d789e1f14c05b...	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 UTC	-208	-188
3	440d0d17af552815d15a9e41...	2017-03-07 23:59:51 UTC	2017-09-19 15:12:50 UTC	-195	-165
4	0f4519c5f1c541ddec9f21b3b...	2017-03-09 13:26:57 UTC	2017-09-19 14:38:21 UTC	-194	-161
5	285ab9426d6982034523a855...	2017-03-08 22:47:40 UTC	2017-09-19 14:00:04 UTC	-194	-166
6	2fb597c2f772eca01b1f5c561...	2017-03-08 18:09:02 UTC	2017-09-19 14:33:17 UTC	-194	-155
7	47b40429ed8cce3aee919979...	2018-01-03 09:44:01 UTC	2018-07-13 20:51:31 UTC	-191	-175
8	2fe324febf907e3ea3f2aa9650...	2017-03-13 20:17:10 UTC	2017-09-19 17:00:07 UTC	-189	-167
9	2d7561026d542c8dbd8f0dae...	2017-03-15 11:24:27 UTC	2017-09-19 14:38:18 UTC	-188	-159
10	c27815f7e3dd0b926b585526...	2017-03-15 23:23:17 UTC	2017-09-19 17:14:25 UTC	-187	-162

- The above result shows the top 10 orders that took the highest time to deliver.
- The difference in the estimated delivery date and the actual delivery date is also quite large for these orders. It suggests that these delays were not pro-actively found or assessed when the customer placed these orders.

Curious to know how the time_to_delivery and diff_estimated_delivery vary year to year.

SQL Query: year on year average time to delivery and average difference between estimated delivery and actual delivery.

```
select extract(year from order_purchase_timestamp) as order_year,
round(avg(time_to_delivery)) as avg_time_to_delivery,
round(avg(diff_estimated_delivery)) as avg_diff_est_delivery
from(SELECT distinct order_id,
       order_purchase_timestamp,
       order_delivered_customer_date,
       date_diff(order_purchase_timestamp, order_delivered_customer_date, day
       ) time_to_delivery,
       date_diff(order_estimated_delivery_date,order_delivered_customer_date,
        day) diff_estimated_delivery
       from `target_brazil.orders`
       where order_delivered_customer_date is not null
       order by time_to_delivery) a
group by order_year
order by order_year
```

Row	order_year	avg_time_to_delivery	avg_diff_est_delivery
1	2016	-19.0	35.0
2	2017	-13.0	11.0
3	2018	-12.0	10.0

- Based on the result, we can see that the average time to delivery is decreasing. Looks like there are delivery solutions in place for faster delivery.
- The average difference between estimated delivery date and the actual delivery date is also reducing. Meaning, compared to 2017, Target is able to predict the estimated delivery date better.
- Although it shows an increase, there is very less data for 2016 and for 2018 we have data only till Oct.

Looking at month on month data

SQL query:

```
select extract(month from order_purchase_timestamp) as order_month,
format_date("%b", order_purchase_timestamp) order_month_name,
round(avg(time_to_delivery)) as avg_time_to_delivery,
round(avg(diff_estimated_delivery)) as avg_diff_est_delivery
from(SELECT distinct order_id,
       order_purchase_timestamp,
       order_delivered_customer_date,
       date_diff(order_purchase_timestamp, order_delivered_customer_date, day
       ) time_to_delivery,
       date_diff(order_estimated_delivery_date,order_delivered_customer_date,
        day) diff_estimated_delivery
       from `target_brazil.orders`
```

```
        where order_delivered_customer_date is not null
        order by time_to_delivery) a
group by order_month,order_month_name
order by avg_diff_est_delivery desc
```

Row	order_month	order_month_name	avg_time_to_del	avg_diff_est_delivery
1	6	Jun	-10.0	16.0
2	1	Jan	-13.0	13.0
3	10	Oct	-12.0	12.0
4	4	Apr	-12.0	12.0
5	5	May	-11.0	12.0
6	12	Dec	-15.0	12.0
7	7	Jul	-10.0	11.0
8	2	Feb	-16.0	10.0
9	9	Sep	-11.0	10.0
10	8	Aug	-9.0	9.0
11	3	Mar	-15.0	7.0
12	11	Nov	-15.0	7.0

- Based on result we can say that the average difference between the estimated delivery date and the actual delivery date is the highest in the month of Jun overall.

5.3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery. Sort the data to get the following:

5.3.1. Top 5 states with highest/lowest average freight value. Sort in desc/asc limit 5

- **Top 5 states with lowest average freight value:**

  SQL Query:

```
SELECT distinct c.customer_state,
round(avg(oi.freight_value),2) as avg_freight_value,
round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_custome
r_date, day)),2) avg_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_cus
tomer_date, day)),2) avg_diff_estimated_delivery
from `target_brazil.customers` c
join `target_brazil.orders` o
on c.customer_id = o.customer_id
join `target_brazil.order_items` oi
on oi.order_id = o.order_id
where order_delivered_customer_date is not null
group by c.customer_state
order by avg_freight_value
limit 5
```

Row	customer_state	avg_freight_value	avg_time_to_del	avg_diff_estimat
1	SP	15.11	-8.26	10.27
2	PR	20.47	-11.48	12.53
3	MG	20.63	-11.52	12.4
4	RJ	20.91	-14.69	11.14
5	DF	21.07	-12.5	11.27

- The state ‘SP’ has the lowest average freight value.

- **Top 5 states with the highest average freight value:**

  SQL Query:

```
SELECT distinct c.customer_state,
round(avg(oi.freight_value),2) as avg_freight_value,
round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_custome
r_date, day)),2) avg_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_cus
tomer_date, day)),2) avg_diff_estimated_delivery
from `target_brazil.customers` c
join `target_brazil.orders` o
on c.customer_id = o.customer_id
join `target_brazil.order_items` oi
on oi.order_id = o.order_id
where order_delivered_customer_date is not null
group by c.customer_state
order by avg_freight_value desc
limit 5
```

Row	customer_state	avg_freight_value	avg_time_to_del	avg_diff_estimat
1	PB	43.09	-20.12	12.15
2	RR	43.09	-27.83	17.43
3	RO	41.33	-19.28	19.08
4	AC	40.05	-20.33	20.01
5	PI	39.12	-18.93	10.68

- The state ‘PB’ has the highest average freight value.

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

- **SQL Query: Top 5 states with highest average time to delivery: Since the results are in negative, the asc will give the highest average time to delivery**

```
SELECT distinct c.customer_state,
round(avg(oi.freight_value),2) as avg_freight_value,
round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_cust
omer_date, day)),2) avg_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_
customer_date, day)),2) avg_diff_estimated_delivery
from `target_brazil.customers` c
join `target_brazil.orders` o
on c.customer_id = o.customer_id
join `target_brazil.order_items` oi
on oi.order_id = o.order_id
where order_delivered_customer_date is not null
group by c.customer_state
order by avg_time_to_delivery
limit 5
```

Row	customer_state	avg_freight_valu	avg_time_to_delivery	avg_diff_estimat
1	RR	43.09	-27.83	17.43
2	AP	34.16	-27.75	17.44
3	AM	33.31	-25.96	18.98
4	AL	35.87	-23.99	7.98
5	PA	35.63	-23.3	13.37

  - The state ‘RR’ has the highest average time to deliver the orders from the purchase date.

- **SQL query: Top 5 states with lowest average time to delivery: Since the results are in negative, the desc will give the lowest average time to delivery**

```
SELECT distinct c.customer_state,
round(avg(oi.freight_value),2) as avg_freight_value,
round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_cust
omer_date, day)),2) avg_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_
customer_date, day)),2) avg_diff_estimated_delivery
from `target_brazil.customers` c
join `target_brazil.orders` o
on c.customer_id = o.customer_id
join `target_brazil.order_items` oi
on oi.order_id = o.order_id
where order_delivered_customer_date is not null
group by c.customer_state
order by avg_time_to_delivery desc limit 5
```

Row	customer_state	avg_freight_valu	avg_time_to_delivery	avg_diff_estimat
1	SP	15.11	-8.26	10.27
2	PR	20.47	-11.48	12.53
3	MG	20.63	-11.52	12.4
4	DF	21.07	-12.5	11.27
5	SC	21.51	-14.52	10.67

- The state of 'SP' has the lowest average time to deliver the orders from the purchase date.

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

- **SQL Query: Top 5 states where delivery is really fast compared to estimated date:**

```
SELECT distinct c.customer_state,
round(avg(oi.freight_value),2) as avg_freight_value,
round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_cust
omer_date, day)),2) avg_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_
customer_date, day)),2) avg_diff_estimated_delivery
from `target_brazil.customers` c
join `target_brazil.orders` o
on c.customer_id = o.customer_id
join `target_brazil.order_items` oi
on oi.order_id = o.order_id
where order_delivered_customer_date is not null
group by c.customer_state
order by avg_diff_estimated_delivery
limit 5
```

Row	customer_state	avg_freight_valu	avg_time_to_deli	avg_diff_estimated_delivery
1	AL	35.87	-23.99	7.98
2	MA	38.49	-21.2	9.11
3	SE	36.57	-20.98	9.17
4	ES	22.03	-15.19	9.77
5	BA	26.49	-18.77	10.12

- The state of 'AL' has the lowest average difference between the estimated delivery date and the actual delivery date.

- **SQL query: Top 5 states where delivery is not so fast compared to estimate date:**

```
SELECT distinct c.customer_state,
round(avg(oi.freight_value),2) as avg_freight_value,
round(avg(date_diff(o.order_purchase_timestamp, o.order_delivered_cust
omer_date, day)),2) avg_time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_
customer_date, day)),2) avg_diff_estimated_delivery
from `target_brazil.customers` c
join `target_brazil.orders` o
on c.customer_id = o.customer_id
join `target_brazil.order_items` oi
on oi.order_id = o.order_id
where order_delivered_customer_date is not null
group by c.customer_state
order by avg_diff_estimated_delivery desc
limit 5
```

Row	customer_state	avg_freight_valu	avg_time_to_deli	avg_diff_estimated_delivery
1	AC	40.05	-20.33	20.01
2	RO	41.33	-19.28	19.08
3	AM	33.31	-25.96	18.98
4	AP	34.16	-27.75	17.44
5	RR	43.09	-27.83	17.43

- The state of 'AC' has the highest average difference between the estimated delivery date and actual delivery date.

6. Payment type analysis:

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

SQL Query:

```
select payment_type,
month_name,
number_of_orders
from (SELECT distinct payment_type,
      extract(month from o.order_purchase_timestamp) as month_number,
      format_date("%b", o.order_purchase_timestamp) as month_name,
      count(o.order_id) as number_of_orders
      FROM `target_brazil.payments` p
      join `target_brazil.orders` o
      on o.order_id = p.order_id
      group by p.payment_type,month_name,month_number
      order by p.payment_type, month_number) a
```

Row	payment_type	month_name	number_of_orde
1	UPI	Jan	1715
2	UPI	Feb	1723
3	UPI	Mar	1942
4	UPI	Apr	1783
5	UPI	May	2035
6	UPI	Jun	1807
7	UPI	Jul	2074
8	UPI	Aug	2077
9	UPI	Sep	903
10	UPI	Oct	1056
11	UPI	Nov	1509
12	UPI	Dec	1160
13	credit_card	Jan	6103
14	credit_card	Feb	6609
15	credit_card	Mar	7707
16	credit_card	Apr	7301
17	credit_card	May	8350
18	credit_card	Jun	7276
19	credit_card	Jul	7841
20	credit_card	Aug	8269

Curious to see the top 3 months in each payment types that have the greatest number of orders placed.

- SQL query:

```
select payment_type, month_name, number_of_orders
from(
select
row_number() over(partition by payment_type order by number_of_orders desc)
as row_num,
payment_type,
month_name,
number_of_orders
from (SELECT distinct payment_type,
      extract(month from o.order_purchase_timestamp) as month_number,
      format_date("%b", o.order_purchase_timestamp) as month_name,
      count(o.order_id) as number_of_orders
      FROM `target_brazil.payments` p
      join `target_brazil.orders` o
      on o.order_id = p.order_id
      group by p.payment_type,month_name,month_number
      order by p.payment_type, number_of_orders desc) a) b
where row_num < 4
```

Row	payment_type	month_name	number_of_orders
1	debit_card	Aug	311
2	debit_card	Jul	264
3	debit_card	Jun	209
4	UPI	Aug	2077
5	UPI	Jul	2074
6	UPI	May	2035
7	not_defined	Aug	2
8	not_defined	Sep	1
9	credit_card	May	8350
10	credit_card	Aug	8269
11	credit_card	Jul	7841
12	voucher	Jul	645
13	voucher	May	613
14	voucher	Mar	591

- The above result shows the top 3 months in each payment type that has the highest number of orders.
- We can see that July and August are in almost all payment types as the top months.

6.2. Count of orders based on the no. of payment installments

SQL Query:

```
SELECT distinct payment_installments as number_of_payment_installments,
count(order_id) as number_of_orders
FROM `target_brazil.payments`
group by number_of_payment_installments
order by number_of_orders desc
```

Row	number_of_payment_installments	number_of_orders
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
11	12	133
12	15	74
13	18	27
14	11	23
15	24	18
16	20	17
17	13	16
18	14	15
19	17	8
20	16	5
21	21	3
22	0	2
23	22	1
24	23	1

- The result shows that most of the customers are opting for 1,2,3,4 instalments mostly.
- Very less customers are opting for 21, 0, 22, 23 instalments.

Curious to see which payment type do the customers prefer more.

SQL Query:

```
SELECT distinct payment_type, count(o.order_id) as number_of_orders
FROM `target_brazil.payments` p
join `target_brazil.orders` o
on o.order_id = p.order_id
group by p.payment_type
order by number_of_orders desc
```

Row	payment_type	number_of_orders
1	credit_card	76795
2	UPI	19784
3	voucher	5775
4	debit_card	1529
5	not_defined	3

- Most preferred payment type is credit card and followed by UPI method.
- Debits cards are not used by many customers but still it is used by enough customers.

Actionable Insights:

1. Data is given for the time period of 2016-09-04 to 2018-10-17. In terms of number of days, we have 773 days of data. More data can give better insights from the data. It would be good to acquire more data.
2. There are 27 states from which customers have made online purchases from Brazil. Although, customers from most of the states have placed at least 500 orders, some states like PI, RN, AL, SE, TO, RO and AM have less than 500 orders during the period. States like AC, AP and RR have less than 100 orders during the period. These states can be targeted for some discounted sales or special offers to increase the online orders.
3. Out of 8011 cities in the database, customers from only 4119 cities have placed orders. These cities could be analysed to find out the reason for no participation.
4. Over the years, the number of online orders has increased. This is a positive sign indicating that Target is taking a right decision investing in expanding its business online.
5. The top 5 months that have the highest number of orders are Aug, May, Jul which have more than 10k and Mar & Jun between 9k to 10k orders. The least number of orders are placed in the months of Sep, Oct, Dec, Nov and Jan. Inventory can be planned efficiently to handle the increased and reduced demands.
6. Based on the results, we can say that there is seasonality in the data. There is a peak of number of online orders in the months of Oct in 2016, March, May, August in 2017 and March and August in 2018. The overall peaks are during March, May, August and November (although Nov is among the months with least number of orders overall). These peaks look like they fall close to 'Mother's day in Brazil', 'Father's day' , 'Black Friday', 'Back to School' and 'Christmas' seasons. Better inventory stocks can be planned ahead to handle these sudden sometimes unexpected peaks in demands.
7. Based on the analysis, we can see that the number of orders in the early part of the days is lesser than the later part. Most of the orders are placed in the afternoon followed by night. Least number of orders are placed at dawn. So, any downtime can be planned during dawn to avoid any business loss due to technical unavailability of the website.

   Note: We have considered hours 1 to 6 as dawn, 7 to 12 as morning, 13 to 18 afternoon and 19 to 24 and 0 as night.
8. Based on the analysis, most percentage of the cancelled orders were placed between 1 hour and 6 hour(dawn) – 0.772%. The least percentage of orders that were cancelled were ordered at night (19 to 24 and 0 hours). This behaviour can be further analysed by requesting the customers to fill a cancellation form asking reasons for cancellation
9. Highest number of customers are from the state 'SP', 'RJ' and 'MG'. There are more than 10k customers from these states. Marketing campaigns for new products, new website functionalities and features can be experimented in these states to understand customer dynamics.
10. The least number of customers are from the state 'RR', 'AP' and 'AC'. These states have less than 100 customers. More marketing campaigns about the Target's e-commerce can be conducted in these states to increase the customer base.

11. There is an increase in the total cost of orders by 136.9% from 2017 to 2018. There is an increase in the average cost of orders by 3.23% from 2017 to 2018.
12. The states ‘PB’, ‘AL’, ‘AC’, ‘RO’ and ‘PA’ have the Top 5 mean prices and the state of SP has the lowest mean price. The states ‘SP’, ‘PR’, ‘RS’, ‘MG’ and ‘ES’ have the bottom 5 mean prices.
13. The states ‘RR’, ‘PB’, ‘RO’, ‘AC’ and ‘PI’ are the top in terms of highest mean freight values. Based on a research mentioned in this article from the following link, (https://labsnews.com/en/articles/ecommerce/brazil-ecommerce-shopping-dates/) reducing the freight values may motivate customers from Brazil to do more online purchases. We identified that ‘RR’ and ‘AC’ are among the states with least number of customers. Since they also fall in the states with highest freight values, we may be able to attract more customers by reducing freight values in these states.
14. There are about 6535 order that were delayed more than the estimated delivery date. About 40 deliveries have been delayed for 100 or more days and about 121 orders were delayed at least 50 days from the estimated delivery date. These delays will negatively impact the business and customer trust. It is necessary to take actions to make the delivery better and on time.
15. There are about 8 orders that show as delivered but do not have a delivery date. These could be data issues. These should be further analysed to confirm data issue and identify bugs in the system or process.
16. Based on the analysis, it looks the average difference between the estimated delivery date and the actual delivery date is the highest in the month of Jun followed by Jan, Oct, Apr, May and Dec overall. This can be prevented by pumping up delivery resources during these periods.
17. The states ‘RR’, ‘AP’, ‘AM’, ‘AL’ and ‘PA’ are the top states with the highest average time to deliver the orders from the purchase date. Reducing the time to delivery may increase customer satisfaction in these states.
18. The states ‘AC’, ‘RO’, ‘AM’, ‘AP’, ‘RR’ are the top 5 states with the highest average difference between the estimated delivery date and actual delivery date. Reducing this difference will increase customer trust. Better estimated delivery date prediction models should be developed or more planning is needed towards the inventory to make sure stocks are pumped up or shipping process and resources can be planned better in these states.
19. Based on the analysis, most of the customers are opting for 1, 2, 3 instalments (~10k orders) followed by 4 and 10 instalments. These are good options to keep.
20. Very less customers are opting for 21, 0, 22, 23 instalments. Approximately only 1 or 2 customers per category. These options can be removed if required.
21. Most preferred payment type is credit card and followed by UPI method and Debits cards. It is good to retain these payment methods to maintain good customer experience in the payment flow.

Recommendations:

1. Collect better and more date for deeper analysis.
2. Discount sales/special offers can be targeted for customers form States like PI, RN, AL, SE, TO, RO, AM, AC, AP and RR.
3. Plan for efficient inventory in the months of Aug, May, Jul, Mar, November and Jun to handle increased number of orders than in the months of Sep, Oct, Dec, Nov and Jan.
4. Plan any downtime if required during the dawn hours (1 am to 6 am) to avoid any business loss due to technical unavailability of the website.
5. Analyse the cancellation behaviour of the customers by requesting the customers to fill a cancellation form asking reasons for cancellation.
6. Marketing campaigns for new products, new website functionalities and features can be experimented in the states 'SP', 'RJ' and 'MG' where the customer base is high. There are more than 10k customers from these states.
7. More marketing campaigns about Target's e-commerce can be conducted to increase customer base in 'RR', 'AP', 'AC' states since these have the least number of. Incentives and rewards can be targeted at these customers in these states.
8. Reducing the freight values may attract more customers to purchase online. We identified that 'RR' and 'AC' are among the states with least number of customers. Since they also fall in the states with highest freight values, we may be able to attract more customers by reducing freight values in these states. The following Reference article: https://labsnews.com/en/articles/ecommerce/brazil-ecommerce-shopping-dates/ contains a mention about a research that says, the Free Shipping day in Brazil is one of the lucrative days for e-commerce in Brazil.
9. Take actions to make the delivery better and on time. Better prediction of estimated delivery dates should be made. Demand prediction models can help predict demand beforehand. The stocks can be checked in the inventory regularly based on the demand predictions. Delivery process and resources can be planned based on the demand predictions.
10. All necessary actions must be taken to meet the estimated delivery date communicated to the customer. Missing the estimated delivery date may impact customer trust. Refer to 'Action Insights' section for the states that are impacted by the delayed deliveries.
11. The 1, 2, 3, 4 and 10 instalments options are customer favourites. However the 21, 0, 22, 23 instalments are least favourites. These can be considered for decommission if required.
12. Credit card, UPI, Debit cards payment options are well utilized by the customers. More payment options like Wallets, Buy now Pay later options can be experimented in the form or AB testing to identify customer's interests for further enhancement.