Target SQL

Overview:

Target is one of the world's most recognized brands and one of America's leading retailers. The database contains 99441 customers, 3095 sellers and also 99441 orders made by different users. Below are the query's:

```
select count(distinct customer_id) as no_of_customers
from `target_sql.customers`;
```

Row	no_of_customer
1	99441

```
select count(distinct seller_id) as no_of_sellers
from `target_sql.sellers`;
```

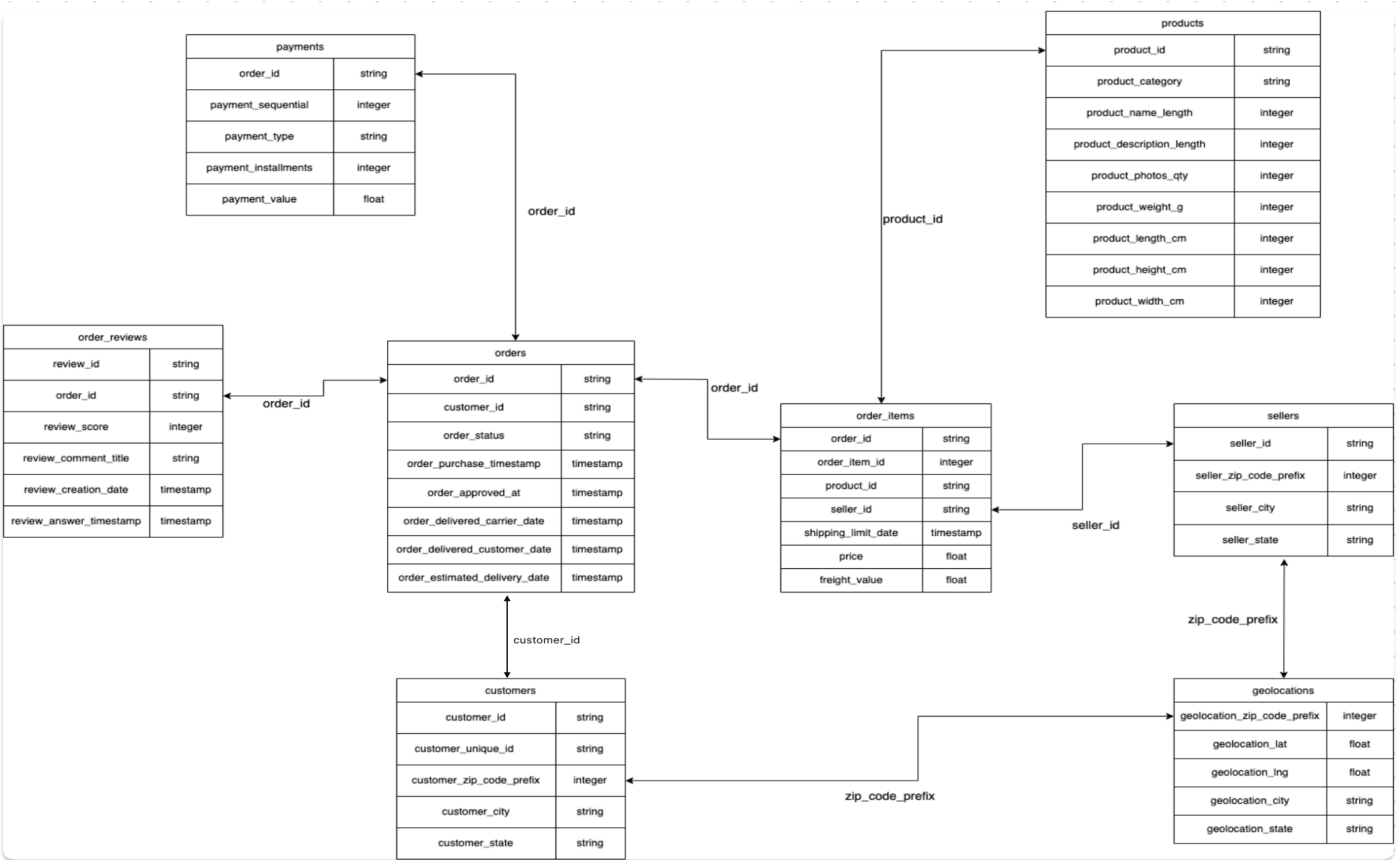
Row	no_of_sellers
1	3095

```
select count(distinct order_id) as no_of_orders
from `target_sql.orders`;
```

Row	no_of_orders
1	99441

1. Initial Exploration:

a. There are 8 tables in the target database and here are the datatypes(BigQuery) for each column in the tables.



b. The data is given for the time period of September 2016 to October 2018. Also here is the detail of the customer and the order time which is the first and last order in the dataset.

First Order:

```
select c.customer_id, c.customer_city, c.customer_state, o.order_id,
o.order_purchase_timestamp
from `target_sql.orders` o
inner join `target_sql.customers` c
on o.customer_id = c.customer_id
where o.order_purchase_timestamp = (
 select min(order_purchase_timestamp)
 from `target_sql.orders`
);
```

Row	customer_id	customer_city	customer_state	order_id	order_purchase_timestamp
1	08c5351a6aca1c1589a38f244...	boa vista	RR	2e7a8482f6fb09756ca50c10d...	2016-09-04 21:15:19 UTC

So, it can be seen that the first order is made by the customer “08c5351a6aca1c1589a38f244edeee9d..” from “boa vista” city in “RR” state on date “2016-09-04”.

Last Order:

```
select c.customer_id, c.customer_city, c.customer_state, o.order_id,
o.order_purchase_timestamp
from `target_sql.orders` o
inner join `target_sql.customers` c
on o.customer_id = c.customer_id
where o.order_purchase_timestamp = (
 select max(order_purchase_timestamp)
 from `target_sql.orders`
);
```

Row	customer_id	customer_city	customer_state	order_id	order_purchase_timestamp
1	a4b417188addbc05b26b72d5...	sorocaba	SP	10a045cdf6a5650c21e9cfeb6...	2018-10-17 17:30:18 UTC

So, it can be seen that the last order is made by the customer “a4b417188addbc05b26b72d5e44..” from “sorocaba” city in “SP” state on date “2018-10-17”.

c. List of cities and states of the customers who have placed an order during the period of 2016 to 2018. Here is the query for different cities along with the count which cities are not of order made by customers living in that city.

```
select customer_city, count(customer_city) as Count
from `target_sql.customers`
group by customer_city;
```

So there are 4119 distinct cities and output shows only the 10 cities.

Row	customer_city	Count
1	acu	3
2	ico	8
3	ipe	2
4	ipu	4
5	ita	3
6	itu	136
7	jau	74
8	luz	2
9	poa	85
10	uba	53

Now, the query for different states along with the count which states the no of order made by customers living in that state.

```
select customer_state, count(customer_state) as Count
from `target_sql.customers`
group by customer_state;
```

Row	customer_state	Count
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746
6	MG	11635
7	BA	3380
8	RJ	12852
9	GO	2020
10	MA	747

So there are 27 distinct states and output shows only the 10 states.

2. In-depth Exploration:

a. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario?
So to see the trend in e-commerce I have used the 'order' table and and grouped the data quarter wise.

```
select t2.Quarter, count(t2.order_id) as No_of_orders from (
 select
   t1.order_id,
   t1.Month_Year_Name,
   case
     when (t1.month between 9 and 12) and t1.year = 2016 then '2016_Q4'
     when (t1.month between 1 and 3) and t1.year = 2017 then '2017_Q1'
     when (t1.month between 4 and 6) and t1.year = 2017 then '2017_Q2'
     when (t1.month between 7 and 9) and t1.year = 2017 then '2017_Q3'
     when (t1.month between 10 and 12) and t1.year = 2017 then '2017_Q4'
     when (t1.month between 1 and 3) and t1.year = 2018 then '2018_Q1'
     when (t1.month between 4 and 6) and t1.year = 2018 then '2018_Q2'
     when (t1.month between 7 and 10) and t1.year = 2018 then '2018_Q3'
   end as Quarter
 from (
   select
     order_id,
     extract(year from order_purchase_timestamp) as year,
     extract(month from order_purchase_timestamp) as month,
     format_date('%B %Y', order_purchase_timestamp) as Month_Year_Name
   from `target_sql.orders`) as t1
) as t2
group by t2.Quarter
order by t2.Quarter;
```

#Note: Since there were 2 months out of range while defining the quarter so added September 2016 in 2016_Q4 and October 2018 in 2018_Q3 as I thought there was no need to create additional quarters just for 1 month.

Row	Quater	No_of_orders
1	2016_Q4	329
2	2017_Q1	5262
3	2017_Q2	9349
4	2017_Q3	12642
5	2017_Q4	17848
6	2018_Q1	21208
7	2018_Q2	19979
8	2018_Q3	12824

Here is the output of no of orders on the basis of Quarter. Now, it can be seen that during 2017_Q4, 2018_Q1 and 2018_Q2, the no of orders are at the highest so it may be due the festival season in Brazil as at that particular time they have some important festivals like Christmas, New Year, Rio Carnival, Good Friday, etc.

Also below is the query if we want to see the month which has highest no of orders in quarter wise.

```
select t1.Quarter, t1.Month_Year_Name, t1.No_of_orders from (
 select
   count(order_id) as No_of_orders,
   extract(year from order_purchase_timestamp) as year, extract(month from order_purchase_timestamp) as month,
   format_date('%B %Y', order_purchase_timestamp) as Month_Year_Name,
   case
     when (extract(month from order_purchase_timestamp) between 9 and 12) and extract(year from
order_purchase_timestamp) = 2016 then '2016_Q4'
     when (extract(month from order_purchase_timestamp) between 1 and 3) and extract(year from
order_purchase_timestamp) = 2017 then '2017_Q1'
     when (extract(month from order_purchase_timestamp) between 4 and 6) and extract(year from
order_purchase_timestamp) = 2017 then '2017_Q2'
     when (extract(month from order_purchase_timestamp) between 7 and 9) and extract(year from
order_purchase_timestamp) = 2017 then '2017_Q3'
     when (extract(month from order_purchase_timestamp) between 10 and 12) and extract(year from
order_purchase_timestamp) = 2017 then '2017_Q4'
     when (extract(month from order_purchase_timestamp) between 1 and 3) and extract(year from
order_purchase_timestamp) = 2018 then '2018_Q1'
     when (extract(month from order_purchase_timestamp) between 4 and 6) and extract(year from
order_purchase_timestamp) = 2018 then '2018_Q2'
     when (extract(month from order_purchase_timestamp) between 7 and 10) and extract(year from
order_purchase_timestamp) = 2018 then '2018_Q3'
   end as Quarter
 from `target_sql.orders`
 group by year, month, Month_Year_Name, Quarter
) as t1
order by t1.Quarter, t1.year, t1.month;
```

Row	Quarter	Month_Year_Name	No_of_orders
1	2016_Q4	September 2016	4
2	2016_Q4	October 2016	324
3	2016_Q4	December 2016	1
4	2017_Q1	January 2017	800
5	2017_Q1	February 2017	1780
6	2017_Q1	March 2017	2682
7	2017_Q2	April 2017	2404
8	2017_Q2	May 2017	3700
9	2017_Q2	June 2017	3245
10	2017_Q3	July 2017	4026
11	2017_Q3	August 2017	4331

So from the result it can be seen that in
2016_Q4 => October 2016 had maximum sales, similarly for
2017_Q1 => March 2017,
2017_Q2 => May 2017,...
And so on.

b. Here is the analysis at what time of the day Brazilian customers tend to buy.

```
select
 case
   when extract(time from order_purchase_timestamp) between '00:00:00' and '05:59:59'
     then 'Night (12 AM to 6 AM)'
   when extract(time from order_purchase_timestamp) between '06:00:00' and '11:59:59'
     then 'Morning (6 AM to 12 PM)'
   when extract(time from order_purchase_timestamp) between '12:00:00' and '17:59:59'
     then 'Afternoon (12 PM to 6 PM)'
   else 'Evening (6 PM to 12 AM)'
 end as Time_Frame,
 count(order_id) as No_of_orders
from `target_sql.orders`
group by Time_Frame
order by No_of_orders desc;
```

So from the output it can be seen that during the afternoon and evening there are maximum no of orders placed by the customers and during night there are significantly very less no of orders.

Row	Time_Frame	No_of_orders
1	Afternoon (12 PM to 6 PM)	38361
2	Evening (6 PM to 12 AM)	34100
3	Morning (6 AM to 12 PM)	22240
4	Night (12 AM to 6 AM)	4740

3. Evolution of E-commerce:

a. Month on month orders by state.

```
select
  t1.customer_state,
  t1.Month_Year_Name,
  count(t1.order_id) as No_of_orders
from (
 select
   o.order_id, c.customer_state,
   extract(month from o.order_purchase_timestamp) as Month,
   extract(year from o.order_purchase_timestamp) as Year,
   format_date('%B %Y', order_purchase_timestamp) as Month_Year_Name
 from `target_sql.customers` c
 inner join `target_sql.orders` o
 on c.customer_id = o.customer_id) as t1
group by  t1.customer_state, t1.Year, t1.Month, t1.Month_Year_Name
order by t1.customer_state, t1.Year, t1.Month;
```

Row	customer_state	Month_Year_Name	No_of_orders
1	AC	January 2017	2
2	AC	February 2017	3
3	AC	March 2017	2
4	AC	April 2017	5
5	AC	May 2017	8
6	AC	June 2017	4
7	AC	July 2017	5
8	AC	August 2017	4
9	AC	September 2017	5
10	AC	October 2017	6
11	AC	November 2017	5
12	AC	December 2017	5
13	AC	January 2018	6

Here is the output of the query which shows the month on month no of orders made by customers living in different states. So there is the data of all 27 states in Brazil and within each state we have no of orders month on month.

b. Distribution of customers across the states in Brazil

```
select
  distinct customer_state,
  count(customer_id) as No_of_customers
from `target_sql.customers`
group by customer_state
order by No_of_customers desc;
```

Here is the list of states in Brazil with the no of customers living in that state in decreasing order.

Row	customer_state	No_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

4. Impact on Economy:

a. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only). Used “payment_value” column in payments table.

To see the percent increase in cost of orders from 2017 to 2018 (summation of months from Jan to Aug) so below is the query for that using CTE.

```
with total_payment_val_2017 as (
 select round(sum(t1.payment_value), 2) as total_payment_2017 from (
   select
     p.order_id, p.payment_value
   from `target_sql.orders` o join `target_sql.payments` p
   on o.order_id = p.order_id
   where extract(year from o.order_purchase_timestamp) = 2017 and extract(month from
o.order_purchase_timestamp) between 1 and 8
 ) as t1
),
total_payment_val_2018 as (
 select round(sum(t2.payment_value), 2) as total_payment_2018 from (
   select
     p.order_id, p.payment_value
   from `target_sql.orders` o join `target_sql.payments` p
   on o.order_id = p.order_id
   where extract(year from o.order_purchase_timestamp) = 2018 and extract(month from
o.order_purchase_timestamp) between 1 and 8
 ) as t2
)
select
 total_payment_2017, total_payment_2018,
 concat(round((((total_payment_2018 - total_payment_2017) / total_payment_2017) * 100), 2), '%') as
percent_increase_2017_to_2018
from total_payment_val_2017, total_payment_val_2018;
```

So it seems like if we take total payment value year wise including month from Jan to Aug then there is a percent increase of 136.98%

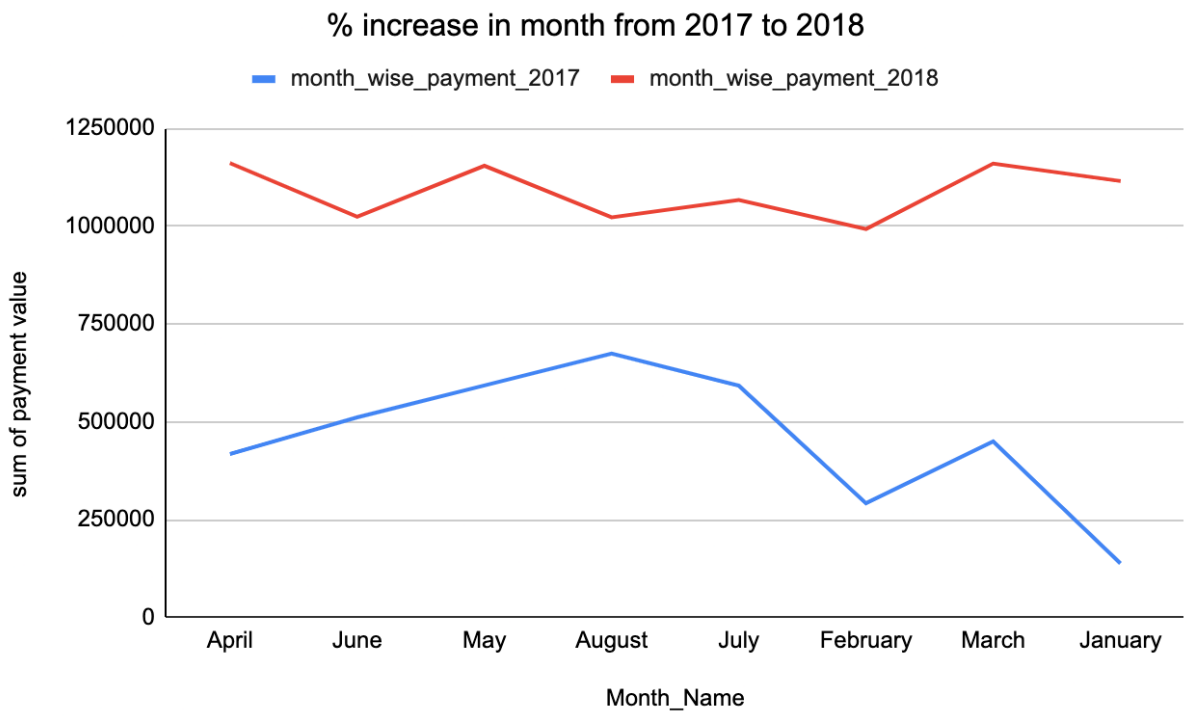
Row	total_payment_2017	total_payment_2018	percent_increase_2017_to_2018
1	3669022.12	8694733.84	136.98%

Also if we want to see the percent increase in cost of orders from 2017 to 2018 but month wise from Jan to Aug so below is the query for that using CTE.

```
with total_payment_val_2017 as (
 select t1.month, t1.Month_Name, t1.month_wise_payment_2017 from (
   select
     round(sum(p.payment_value), 2) as month_wise_payment_2017,
     format_date('%B', o.order_purchase_timestamp) as Month_Name, extract(month from
o.order_purchase_timestamp) as month
   from `target_sql.orders` as o join `target_sql.payments` as p
   on o.order_id = p.order_id
   where extract(month from o.order_purchase_timestamp) between 1 and 8 and extract(year from
o.order_purchase_timestamp) = 2017
   group by month, Month_Name
   order by month
  ) as t1
),
total_payment_val_2018 as (
 select t2.Month_Name, t2.month_wise_payment_2018 from (
   select
     round(sum(p.payment_value), 2) as month_wise_payment_2018,
     format_date('%B', o.order_purchase_timestamp) as Month_Name, extract(month from
o.order_purchase_timestamp) as month
   from `target_sql.orders` as o join `target_sql.payments` as p
   on o.order_id = p.order_id
   where extract(month from o.order_purchase_timestamp) between 1 and 8 and extract(year from
o.order_purchase_timestamp) = 2018
   group by month, Month_Name
   order by month
 ) as t2
)
select
 t_2017.Month_Name,
 t_2017.month_wise_payment_2017,
 t_2018.month_wise_payment_2018,
 concat(round((((t_2018.month_wise_payment_2018 - t_2017.month_wise_payment_2017) /
t_2017.month_wise_payment_2017) * 100), 2), '%') as month_percent_increase_2017_to_2018
from total_payment_val_2017 as t_2017
join total_payment_val_2018 as t_2018
on t_2017.Month_Name = t_2018.Month_Name
order by t_2017.month;
```

Output:

Row	Month_Name	month_wise_payment_2017	month_wise_payment_2018	month_percent_increase_2017_to_2018
1	January	138488.04	1115004.18	705.13%
2	February	291908.01	992463.34	239.99%
3	March	449863.6	1159652.12	157.78%
4	April	417788.03	1160785.48	177.84%
5	May	592918.82	1153982.15	94.63%
6	June	511276.38	1023880.5	100.26%
7	July	592382.92	1066540.75	80.04%
8	August	674396.32	1022425.32	51.61%



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

```
select
 c.customer_state, count(oi.order_id) as Count,
 round(sum(oi.price), 2) as sum_of_price,
 round(avg(oi.price), 2) as mean_of_price,
 round(sum(oi.freight_value), 2) as sum_of_freight_value,
 round(avg(oi.freight_value), 2) as mean_of_freight_value
from `target_sql.customers` c
join `target_sql.orders` o
on c.customer_id = o.customer_id
join `target_sql.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state
order by mean_of_price desc, mean_of_freight_value desc;
```

Output:

Row	customer_state	Count	sum_of_price	mean_of_price	sum_of_freight_value	mean_of_freight_value
1	PB	602	115268.08	191.48	25719.73	42.72
2	AL	444	80314.81	180.89	15914.59	35.84
3	AC	92	15982.95	173.73	3686.75	40.07
4	RO	278	46140.64	165.97	11417.38	41.07
5	PA	1080	178947.81	165.69	38699.3	35.83
6	AP	82	13474.3	164.32	2788.5	34.01
7	PI	542	86914.08	160.36	21218.2	39.15
8	TO	315	49621.74	157.53	11732.68	37.25

5. Analysis on sales, freight and delivery time:

a. Calculation of days between purchasing, delivering and estimated delivery

```
select
 order_id,
 extract(date from order_purchase_timestamp) as order_purchase,
 extract(date from order_delivered_customer_date) as order_delivered,
 extract(date from order_estimated_delivery_date) as order_estimated_delivery,
 date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as days_between_purchase_delivered,
 date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as days_between_estimated_delivered
from `target_sql.orders`
where order_delivered_customer_date is not null
order by days_between_purchase_delivered desc;
```

Here is the list of orders where I have calculated the days between the purchase and order got delivered and also days between estimated delivery and order got delivered to the customer sorted in decreasing order of days_between_purchase_delivered.

Row	order_id	order_purchase	order_delivered	order_estimated_delivery	days_between_purchase_delivered	days_between_estimated_delivered
1	ca07593549f1816d26a57...	2017-02-21	2017-09-19	2017-03-22	209	-181
2	1b3190b2dfa9d789e1f14c...	2018-02-23	2018-09-19	2018-03-15	208	-188
3	440d0d17af552815d15a9...	2017-03-07	2017-09-19	2017-04-07	195	-165
4	0f4519c5f1c541ddec9f21...	2017-03-09	2017-09-19	2017-04-11	194	-161
5	285ab9426d6982034523a...	2017-03-08	2017-09-19	2017-04-06	194	-166
6	2fb597c2f772eca01b1f5c...	2017-03-08	2017-09-19	2017-04-17	194	-155
7	47b40429ed8cce3aee919...	2018-01-03	2018-07-13	2018-01-19	191	-175
8	2fe324febf907e3ea3f2aa9...	2017-03-13	2017-09-19	2017-04-05	189	-167
9	2d7561026d542c8dbd8f0...	2017-03-15	2017-09-19	2017-04-13	188	-159
10	437222e3fd1b07396f1d9b...	2017-03-16	2017-09-19	2017-04-28	187	-144

b. Calculating time_to_delivery & diff_estimated_delivery

i. For time_to_delivery used this formula:

time_to_delivery = order_purchase_timestamp - order_delivered_customer_date

```
select
 t1.order_id, t1.order_purchase, t1.order_delivered,
 date_diff(t1.order_delivered, t1.order_purchase, day) as time_to_delivery_in_days
from (
 select
   order_id,
   extract(date from order_purchase_timestamp) as order_purchase,
   extract(date from order_delivered_customer_date) as order_delivered
 from `target_sql.orders`
 where order_delivered_customer_date is not null
)as t1
order by time_to_delivery_in_days desc;
```

Output:

Row	order_id	order_purchase	order_delivered	time_to_delivery_in_days
1	ca07593549f1816d26a57…	2017-02-21	2017-09-19	210
2	1b3190b2dfa9d789e1f14…	2018-02-23	2018-09-19	208
3	440d0d17af552815d15a9…	2017-03-07	2017-09-19	196
4	285ab9426d6982034523a…	2017-03-08	2017-09-19	195
5	2fb597c2f772eca01b1f5c…	2017-03-08	2017-09-19	195
6	0f4519c5f1c541ddec9f21…	2017-03-09	2017-09-19	194
7	47b40429ed8cce3aee919…	2018-01-03	2018-07-13	191
8	2fe324febf907e3ea3f2aa…	2017-03-13	2017-09-19	190
9	c27815f7e3dd0b926b585…	2017-03-15	2017-09-19	188
10	2d7561026d542c8dbd8f0…	2017-03-15	2017-09-19	188

ii. For diff_estimated_delivery used this formula:

diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date

```
select
 t1.order_id, t1.order_estimated_delivery, t1.order_delivered,
 date_diff(t1.order_estimated_delivery, t1.order_delivered, day) as diff_estimated_delivery_in_days
from (
 select
   order_id,
   extract(date from order_estimated_delivery_date) as order_estimated_delivery,
   extract(date from order_delivered_customer_date) as order_delivered
 from `target_sql.orders`
 where order_delivered_customer_date is not null
)as t1
order by diff_estimated_delivery_in_days desc;
```

Output:

Row	order_id	order_estimated_delivery	order_delivered	diff_estimated_delivery_in_days
1	0607f0efea4b566f1eb8f7d…	2018-08-03	2018-03-09	147
2	c72727d29cde4cf870d569…	2017-07-04	2017-02-14	140
3	eec7f369423b033e549c02f…	2018-07-12	2018-02-27	135
4	c2bb89b5c1dd978d507284…	2017-10-11	2017-06-09	124
5	40dc2ba6f322a17626aac6…	2018-01-30	2017-10-13	109
6	1a695d543b7302aa9446c8…	2018-03-22	2017-12-28	84
7	39e0115911bf404857e14b…	2018-04-25	2018-02-01	83
8	38930f76efb00b138f4d632…	2018-04-27	2018-02-08	78
9	c5132855100a12d63ed4e8…	2018-01-11	2017-10-25	78
10	559eea5a72341a4c82dbce…	2018-02-16	2017-11-30	78

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

```
select
 distinct c.customer_state,
 round(avg(oi.freight_value), 2) as mean_freight_value,
 concat(round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)), 2), ' days') as
avg_time_to_delivery,
 concat(round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)), 2), ' days')
as avg_diff_estimated_delivery
from `target_sql.customers` c
join `target_sql.orders` o
on c.customer_id = o.customer_id
join `target_sql.order_items` oi
on o.order_id = oi.order_id
where (o.order_delivered_customer_date, o.order_purchase_timestamp, o.order_estimated_delivery_date) is not null
group by c.customer_state
order by mean_freight_value;
```

Row	customer_state	mean_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	SP	15.15	8.26 days	10.27 days
2	PR	20.53	11.48 days	12.53 days
3	MG	20.63	11.52 days	12.4 days
4	RJ	20.96	14.69 days	11.14 days
5	DF	21.04	12.5 days	11.27 days
6	SC	21.47	14.52 days	10.67 days
7	RS	21.74	14.71 days	13.2 days
8	ES	22.06	15.19 days	9.77 days
9	GO	22.77	14.95 days	11.37 days
10	MS	23.37	15.11 days	10.34 days

d. Sorting the data to get top 5 states with highest/lowest average freight value.

- Top 5 highest avg freight value:

```
select
 distinct c.customer_state,
 round(avg(oi.freight_value), 2) as avg_freight_value,
from `target_sql.customers` c
join `target_sql.orders` o
on c.customer_id = o.customer_id
join `target_sql.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state
order by avg_freight_value desc
limit 5;
```

Row	customer_state	avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

- Top 5 lowest avg freight value:

```
select
 distinct c.customer_state,
 round(avg(oi.freight_value), 2) as avg_freight_value,
from `target_sql.customers` c
join `target_sql.orders` o
on c.customer_id = o.customer_id
join `target_sql.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state
order by avg_freight_value
limit 5;
```

Row	customer_state	avg_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

e. Sorting the data to get top 5 states with highest/lowest average time to delivery

- Top 5 lowest avg time to delivery:

```
select t1.customer_state, t1.avg_time_to_delivery from (
 select
   distinct c.customer_state,
   concat(round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)), 2), ' days')
as avg_time_to_delivery,
   round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)), 2) as order_by_time
 from `target_sql.customers` c
 join `target_sql.orders` o
 on c.customer_id = o.customer_id
 join `target_sql.order_items` oi
 on o.order_id = oi.order_id
 group by c.customer_state
 order by order_by_time
 limit 5
) as t1;
```

Row	customer_state	avg_time_to_delivery
1	SP	8.26 days
2	PR	11.48 days
3	MG	11.52 days
4	DF	12.5 days
5	SC	14.52 days

- Top 5 highest avg time to delivery:

```
select t1.customer_state, t1.avg_time_to_delivery from (
 select
   distinct c.customer_state,
   concat(round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)), 2), ' days')
as avg_time_to_delivery,
   round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)), 2) as order_by_time
 from `target_sql.customers` c
 join `target_sql.orders` o
 on c.customer_id = o.customer_id
 join `target_sql.order_items` oi
 on o.order_id = oi.order_id
 group by c.customer_state
 order by order_by_time desc
 limit 5
) as t1;
```

Row	customer_state	avg_time_to_delivery
1	RR	27.83 days
2	AP	27.75 days
3	AM	25.96 days
4	AL	23.99 days
5	PA	23.3 days

f. Sorting the data to find top 5 states where delivery is really fast / not so fast compared to estimated date (diff_estimated_delivery)

- Top 5 lowest avg difference in delivery and estimated delivery:

```
select t1.customer_state, t1.avg_diff_estimated_delivery from (
 select
   distinct c.customer_state,
   concat(round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)), 2), ' days')
as avg_diff_estimated_delivery,
   round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)), 2) as
order_by_diff,
 from `target_sql.customers` c
 join `target_sql.orders` o
 on c.customer_id = o.customer_id
 join `target_sql.order_items` oi
 on o.order_id = oi.order_id
 group by c.customer_state
 order by order_by_diff
 limit 5
) as t1;
```

Row	customer_state	avg_diff_estimated_delivery
1	AL	7.98 days
2	MA	9.11 days
3	SE	9.17 days
4	ES	9.77 days
5	BA	10.12 days

- Top 5 highest avg difference in delivery and estimated delivery:

```
select t1.customer_state, t1.avg_diff_estimated_delivery from (
 select
   distinct c.customer_state,
   concat(round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)), 2), ' days')
as avg_diff_estimated_delivery,
   round(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)), 2) as
order_by_diff,
 from `target_sql.customers` c
 join `target_sql.orders` o
 on c.customer_id = o.customer_id
 join `target_sql.order_items` oi
 on o.order_id = oi.order_id
 group by c.customer_state
 order by order_by_diff desc
 limit 5
) as t1;
```

Row	customer_state	avg_diff_estimated_delivery
1	AC	20.01 days
2	RO	19.08 days
3	AM	18.98 days
4	AP	17.44 days
5	RR	17.43 days

6. Payment type analysis:

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

```
select
 t1.payment_type, t1.Month_Year_Name, t1.No_of_orders,
 sum(t1.No_of_orders) over(partition by t1.payment_type order by t1.year, t1.month) as
month_over_month_count_of_orders
from (
 select
   count(o.order_id) as No_of_orders,
   p.payment_type,
   extract(year from o.order_purchase_timestamp) as year, extract(month from o.order_purchase_timestamp) as month,
   format_date('%B %Y', o.order_purchase_timestamp) as Month_Year_Name
 from `target_sql.orders` o
 join `target_sql.payments` p
 on o.order_id = p.order_id
 group by year, month, Month_Year_Name, p.payment_type
 order by p.payment_type, year, month
) as t1;
```

Row	payment_type	Month_Year_Name	No_of_orders	month_over_month_count_of_orders
1	UPI	October 2016	63	63
2	UPI	January 2017	197	260
3	UPI	February 2017	398	658
4	UPI	March 2017	590	1248
5	UPI	April 2017	496	1744
6	UPI	May 2017	772	2516
7	UPI	June 2017	707	3223
8	UPI	July 2017	845	4068
9	UPI	August 2017	938	5006
10	UPI	September 2017	903	5909

Row	payment_type	Month_Year_Name	No_of_orders	month_over_month_count_of_orders
40	credit_card	April 2018	5455	56745
41	credit_card	May 2018	5497	62242
42	credit_card	June 2018	4813	67055
43	credit_card	July 2018	4755	71810
44	credit_card	August 2018	4985	76795
45	debit_card	October 2016	2	2
46	debit_card	January 2017	9	11
47	debit_card	February 2017	13	24
48	debit_card	March 2017	31	55
49	debit_card	April 2017	27	82
50	debit_card	May 2017	30	112

So in the above result screenshots, the data is shown as month on month no of orders made by each payment type. Also there is cumulative no of orders to show month over month orders for each payment wise. The screenshot 2 depicts that for eg., for payment_type ‘credit_card’ at row 44 is the last month for credit_card and it shows the no of orders for that month as well as the cumulative no of orders month over month for credit_card.

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

```
select
 distinct payment_installments,
 count(order_id) as No_of_orders,
 round(sum(payment_value), 2) as total_payment
from `target_sql.payments`
group by payment_installments;
```

Row	payment_installments	No_of_orders	total_payment
1	0	2	188.63
2	1	52546	5907233.36
3	2	12413	1579283.03
4	3	10461	1491103.8
5	4	7098	1163907.61
6	5	5239	961174.3
7	6	3920	822611.81
8	7	1626	305157.39
9	8	4268	1313423.34
10	9	644	131015.92

7. Actionable Insights:

a. In question 5a. We have output as:

Row	order_id	order_purchase	order_delivered	order_estimated_delivery	days_between_purchase_delivered	days_between_estimated_delivered
1	ca07593549f1816d26a57...	2017-02-21	2017-09-19	2017-03-22	209	-181
2	1b3190b2dfa9d789e1f14c...	2018-02-23	2018-09-19	2018-03-15	208	-188
3	440d0d17af552815d15a9...	2017-03-07	2017-09-19	2017-04-07	195	-165
4	0f4519c5f1c541ddec9f21...	2017-03-09	2017-09-19	2017-04-11	194	-161
5	285ab9426d6982034523a...	2017-03-08	2017-09-19	2017-04-06	194	-166
6	2fb597c2f772eca01b1f5c...	2017-03-08	2017-09-19	2017-04-17	194	-155
7	47b40429ed8cce3aee919...	2018-01-03	2018-07-13	2018-01-19	191	-175
8	2fe324febf907e3ea3f2aa9...	2017-03-13	2017-09-19	2017-04-05	189	-167
9	2d7561026d542c8dbd8f0...	2017-03-15	2017-09-19	2017-04-13	188	-159
10	437222e3fd1b07396f1d9b...	2017-03-16	2017-09-19	2017-04-28	187	-144

In the above output there are 2 newly created columns as “days_between_purchase_delivery” and “days_between_estimated_delivered” which shows the no of days between the order_purchase, order_delivered and no of days between order_estiated_delivery, order_delivered respectively.

So it seems that the order_id is ordered in descending order of max no of days between the purchase and delivery. Also all the negative values in the days_between_estimated_delivered states that the order has taken more time to deliver than the expected delivery date. So we can focus more on the orders that have more negative value and find out the reasons why it is taking much more time than expected. It may be because the order type being fragile that it is tough to ship or maybe freight charges to deliver a particular type of order is very high.

Hence we can perform action on these types of order and decrease the time between expected delivery and actual delivery.

b. Now in question 1c. We have output showing no of customers state wise who have ordered.

Row	customer_state	Count
1	RN	485
2	CE	1336
3	RS	5466
4	SC	3637
5	SP	41746
6	MG	11635
7	BA	3380
8	RJ	12852
9	GO	2020
10	MA	747

So here it seems that the state SP(São Paulo), MG(Minas Gerais), RJ(Rio de Janeiro) have the maximum no of customers, it may be because these three states lie on the eastern part of Brazil and have the sea boundary so there could be a lot of trade(import/export) in these states and thus have more no of customers compared to other states. Also the transportation cost might be also low as they have good connectivity with the water transport.

8. Recommendations:

a. In question 2b, we have output as:

Row	Time_Frame	No_of_orders
1	Afternoon (12 PM to 6 PM)	38361
2	Evening (6 PM to 12 AM)	34100
3	Morning (6 AM to 12 PM)	22240
4	Night (12 AM to 6 AM)	4740

So this data shows that during different times of the day customers have made orders. It is clearly visible that during the night hours that is 12AM to 6AM these are least no of orders. So during this time period we can cut the cost on the advertisement or promotional messages. Also if some maintenance work is required on a daily basis for 10-15 minutes then we can aim during this time.

Looking at the busiest time range that is afternoon and evening we can push more advertisements and offers and also need more server efficiency to manage the load and keep the website/application running and stable.

b. Now if we see in below query and result,

```sql
select
  distinct payment_type,
  count(order_id) as no_of_orders,
  round(sum(payment_value), 2) as total_payment
from `target_sql.payments`
group by payment_type
order by no_of_orders desc;
```

Row	payment_type	no_of_orders	total_payment
1	credit_card	76795	12542084.19
2	UPI	19784	2869361.27
3	voucher	5775	379436.87
4	debit_card	1529	217989.79
5	not_defined	3	0.0

From this data it seems that max customers prefer to place orders using credit cards, so we can focus on the more offers or deals with different types of credit card to make customers more willing to buy products and hence increase the payment value.

On the other side also we need to focus on the offers on debit cards as there are significantly less no of orders made.