Target Business case study

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

1.1Data type of columns in a table

Query:

```
SELECT *, data_type
FROM Target_E_commerce.INFORMATION_SCHEMA.COLUMNS
```

```
--Data type of columns in  table
SELECT *, data_type
FROM Target_E_commerce.INFORMATION_SCHEMA.COLUMNS
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH | PREVIEW

ow	table_catalog	table_schema	table_name	column_name
1	scaler-dsml-sql12	Target_E_commerce	order_items	order_id
2	scaler-dsml-sql12	Target_E_commerce	order_items	order_item_id
3	scaler-dsml-sql12	Target_E_commerce	order_items	product_id
4	scaler-dsml-sql12	Target_E_commerce	order_items	seller_id
5	scaler-dsml-sql12	Target_E_commerce	order_items	shipping_limit_date

Results per page: 50 1 – 49 of 49

Explanation:

To get the data type of all columns in the given data set, I have used Information_schema.columns view which returns information about all the columns from the data set.

1.2Time period for which the data is given

Query:

```
select distinct(extract(year from order_purchase_timestamp)) as year
from Target_E_commerce.orders
order by year
```

Explanation:

To know the Time period for which the data is given, I have used orders table form given data set and extracted year from the order_purchase_timestamp column and applied distinct function on top of it

Also an other way to find out the time period is extracting the date from the order_purchase_timstamp column and applying min() and max() function to get the starting and ending time.

Query Result:

Query results

JOB INFORMATION | RESULTS | JSON

Row	year
1	2016
2	2017
3	2018

From the query result it is clear that the time period for which the data is given :

2016 to 2018

1.3.cities and state of customers ordered during the given period

Query:

```
select distinct(C.customer_city),C.customer_state
from  `Target_E_commerce.customers` as C join `Target_E_commerce.orders` as O
          on C.customer_id=O.customer_id
```

```
----Cities and States of customers ordered during the given period

select distinct(C.customer_city),C.customer_state
from  `Target_E_commerce.customers` as C join `Target_E_commerce.orders` as O
          on C.customer_id=O.customer_id
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAF

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC

Explanation:

I have joined orders and customers table based on the customer_id column to know the customers who have placed the order during the given time period

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH | PREVIEW

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP

Results per page: 50 1 – 50 of 4310

We can observe the query returned the customer city and respective state name

2.In-depth Exploration

2.1.1 is there a growing trend on e-commerce in Brazil?

Query:

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

RUN SAVE SHARE SCHEDULE MORE This

```
--Is there a growing trend on e-commerce in Brazil

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

Explanation:

The growth of E-commerce business is Explained with the number of orders placed each year. To know that I have used group by clause to group the states and applied count () aggregation on the order_id column from orders table

Query Result:

Query results SA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	year	count_of_orders
1	2016	329
2	2017	45101
3	2018	54011

As we can see the number of orders are increasing in each year starting from 2016 to 2018 hence we can conclude that there is a growing trend for E-commerce business in Brazil

2.1.3can we see some seasonality with peaks at specific months?

Query:

```
select extract(month from order_purchase_timestamp) as month,
       count(order_id) as count_of_orders
from `Target_E_commerce.orders`
group by month
order by month asc;
```

```
----Can we see some seasonality with peaks at specific months?

select extract(month from order_purchase_timestamp) as month,
       count(order_id) as count_of_orders
from `Target_E_commerce.orders`
group by month
order by count_of_orders desc;
```

Explanation:

To find the seasonality with respect to months, I have extracted the month form the order_purchase column from orders table and then applied count aggregration on order_id column, group by clause is used to group all the months and to get the total count of orders for each month

Query Results:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	month	count_of_orders
1	8	10843
2	5	10573
3	7	10318
4	3	9893
5	6	9412
6	4	9343
7	2	8508
8	1	8069
9	11	7544

Results per page: 50 1 – 12 of 12

We can see that in the months August,May,July the orders were high. from my research I found that fathers day is one of the most celebrated days in a year by brazil people which is going happen in second Sunday of august that why we can see highest number of orders from August month

2.1.2 How can we describe a complete scenario?

From the query analysis it is very clear that the number of order are increasing year by year also more number of customers are purchasing the order during August, July, May months so the Business partners to look into this a peak sale month and act accordingly also in the month of September less number of orders were placed hence they can apply some strategies or discounts to increase the sale count

2.2 What time do Brazilian customers tend to buy (dawn, morning Afternoon or night)?

Query:

```
select
  case
    when extract( time from order_purchase_timestamp) between "00:00:00" and "5:59:59"
    then "DAWN"
    when extract( time from order_purchase_timestamp) between "06:00:00" and "11:59:59"
    then "Morning"
    when extract(time from order_purchase_timestamp)  between "12:00:00" and "17:59:59"
    then "Afternoon"
    when extract( time from order_purchase_timestamp) between "18:00:00" and "23:59 :59"
    then "Night"
  end time_of_the_day,
  count(order_id) as count_of_orders
from `Target_E_commerce.orders`
group by time_of_the_day
order by count_of_orders desc
```

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

select
  case
    when extract( time from order_purchase_timestamp) between "00:00:00" and "5:59:59"
    then "DAWN"
    when extract( time from order_purchase_timestamp) between "06:00:00" and "11:59:59"
    then "Morning"
    when extract(time from order_purchase_timestamp)  between "12:00:00" and "17:59:59"
    then "Afternoon"
    when extract( time from order_purchase_timestamp) between "18:00:00" and "23:59:59"
    then "Night"
  end time_of_the_day,
  count(order_id) as count_of_orders
from `Target_E_commerce.orders`
group by time_of_the_day
order by count_of_orders desc;
```

Explanation:

Using the order purchase time stamp column from the orders table time has been extracted first and then using case and when clauses conditions are applied to differentiate the time into 4 parts 1.dawn 2. Morning 3. Afternoon 4. Night. a group by clause is applied to group the data based on time of the day, order by clause is applied to sort the count of orders in descending order.

Query Results:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	time_of_the_day	count_of_orders
1	Afternoon	38361
2	Night	34100
3	Morning	22240
4	DAWN	4740

PERSONAL HISTORY PROJECT HISTORY

From the query results it is clear that during afternoon Brazilians tend to buy more and during dawn time less number of order were made

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

3.1 Get the month on month orders by state

Query:

```
select c.customer_state,extract(month from o.order_purchase_timestamp) as month,
count(o.order_id) as coount_of_orders
from `Target_E_commerce.customers` as c join `Target_E_commerce.orders` as o
      on c.customer_id=o.customer_id
group by c.customer_state,month
order by month
```

```
----month on month orders by state
select c.customer_state,extract(month from o.order_purchase_timestamp) as month ,count(o.order_id) as coount_of_orders
from `Target_E_commerce.customers` as c join `Target_E_commerce.orders` as o
      on c.customer_id=o.customer_id
group by c.customer_state,month
order by month;
```

Explanation:

I have extracted month from order_purchase_timestamp column and applied count aggregation on order_id column and group by group by clause is applied to state and month to get month on month orders state wise

Query Result:

Query results

SAVE RESULTS EXPLORE DA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	customer_state	month	coount_of_order
1	RJ	1	990
2	SP	1	3351
3	DF	1	151
4	RS	1	427
5	CE	1	99
6	PE	1	113
7	PR	1	443
8	BA	1	264
9	MG	1	971

Results per page: 50 1 – 50 of 322

3.2 Distribution of customers of customers across the states in Brazil

Query:

```
select customer_state,count(customer_id) as count_of_customers
from `Target_E_commerce.customers`
group by customer_state
order by count_of_customers desc
```

```
--Distribution of customers across the states in Brazil

select customer_state,count(customer_id) as count_of_customers
from `Target_E_commerce.customers`
group by customer_state
order by count_of_customers desc;
```

Explanation:

to find the number of customers across all the state in Brazil count aggregation is applied on the customer_id column of the customers table and group by clause is applied to state to count of customers state wise

Query Result:

Query results

SAVE RESULTS EXPLO

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	customer_state	count_of_custon
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

Results per page: 50 1 – 27 of 27

From the query result we can see that the State SP has the highest number of customers and if we sort the data in descending order of the no. of customers the query fetched below result

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	customer_state	count_of_customers
1	RR	46
2	AP	68
3	AC	81
4	AM	148
5	RO	253
6	TO	280
7	SE	350
8	AL	413
9	RN	485
10	PI	495
11	PB	536

Results per page: 50 1

PERSONAL HISTORY PROJECT HISTORY

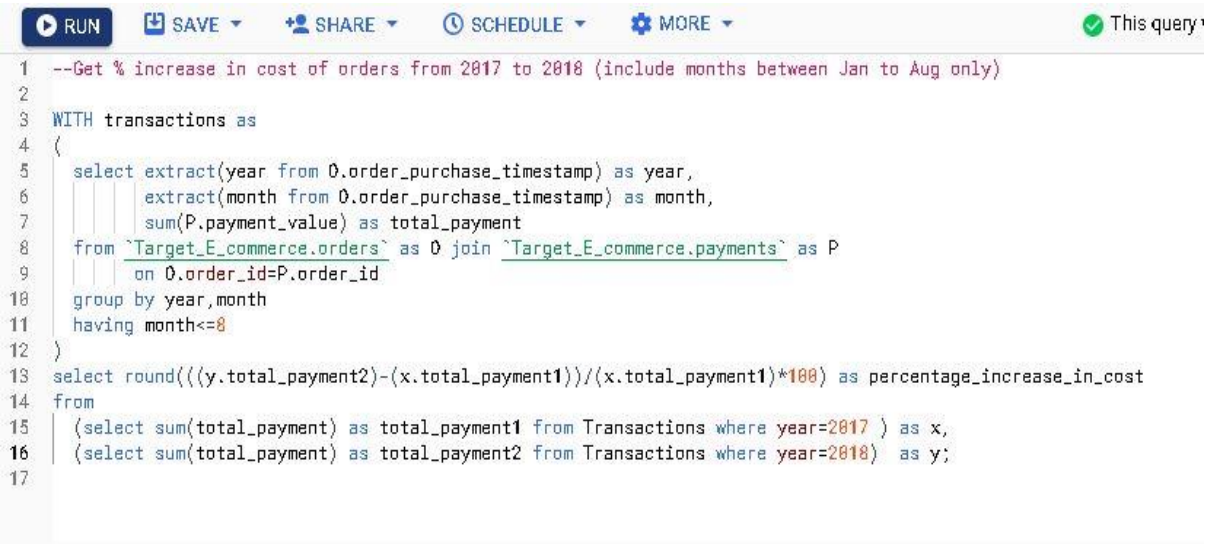
We can conclude that State RR has the least number of customers only 46.

4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices

4.1 Get % increase in cost of orders from 2017 to 2018 (include month between Jan to Aug only)

Query:

```
WITH transactions as
(
  select extract(year from O.order_purchase_timestamp) as year,
         extract(month from O.order_purchase_timestamp) as month,
         sum(P.payment_value) as total_payment
  from `Target_E_commerce.orders` as O join `Target_E_commerce.payments` as P
        on O.order_id=P.order_id
  group by year,month
  having month<=8
)
select round(((y.total_payment2)-
(x.total_payment1))/(x.total_payment1)*100) as percentage_increase_in_cost
from
  (select sum(total_payment) as total_payment1 from Transactions where year=2017 ) as x,
  (select sum(total_payment) as total_payment2 from Transactions where year=2018)  as y;
```



Explanation:

I have used with clause to write a query that will return the sum of payments grouped by month and year and in the main query where condition is applied to filter out the year wise payment at final % gain is calculated by applying the formula

Query Result:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	percentage_increase
1	137.0

From the resultant query we can see that the total increase in cost of orders from 2017 to 2018 is 137%

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

Query:

```
select customers.customer_state,
       Round(Avg(order_items.price)) as Mean_price,
       Round(Avg(order_items.freight_value)) as Mean_freight_price,
       Round(Sum(order_items.price)) as Sum_price,
       Round(Sum(order_items.freight_value)) as Sum_freight_price
from `Target_E_commerce.order_items` as order_items join `Target_E_commerce.orders`
 as orders
      on order_items.order_id=orders.order_id join `Target_E_commerce.customers` as
 customers
      on orders.customer_id=customers.customer_id
group by customers.customer_state
order by  Mean_price desc,
          Mean_freight_price desc,
          Sum_price desc,
          Sum_freight_price desc;
```

```
--Mean & Sum of price and freight value by customer state

select customers.customer_state,
       Round(Avg(order_items.price)) as Mean_price,
       Round(Avg(order_items.freight_value)) as Mean_freight_price,
       Round(Sum(order_items.price)) as Sum_price,
       Round(Sum(order_items.freight_value)) as Sum_freight_price
from `Target_E_commerce.order_items` as order_items join `Target_E_commerce.orders` as orders
      on order_items.order_id=orders.order_id join `Target_E_commerce.customers` as customers
      on orders.customer_id=customers.customer_id
group by customers.customer_state
order by  Mean_price desc,
          Mean_freight_price desc,
          Sum_price desc,
          Sum_freight_price desc;
```

Explanation:

To find the Mean & Sum of price and freight value by state I have joined 3 columns 1.order_items 2.orders 3.customers then calculated Average and sum of Price and of items and freight price of items using Avg() and Sum() aggregation functions and grouped all of them by state using group class

Query result:

Query results

SAVE RESULTS EXPI

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	customer_state	Mean_price	Mean_freight_pr	Sum_price	Sum_freight_pric
1	PB	191.0	43.0	115268.0	25720.0
2	AL	181.0	36.0	80315.0	15915.0
3	AC	174.0	40.0	15983.0	3687.0
4	RO	166.0	41.0	46141.0	11417.0
5	PA	166.0	36.0	178948.0	38699.0
6	AP	164.0	34.0	13474.0	2789.0
7	PI	160.0	39.0	86914.0	21218.0
8	TO	158.0	37.0	49622.0	11733.0
9	RN	157.0	36.0	83035.0	18860.0

Results per page: 50 1 - 27 of 27

5. Analyse on sales ,freight and delivery time

5.1 calculate the days between purchasing, delivering and estimated delivery

Query:

```
select  order_id,
        date_diff(extract(date from order_estimated_delivery_date),extract(
date from order_purchase_timestamp),day)
        as Total_days_for_Estimated_delivery,
        date_diff(extract(date from order_delivered_customer_date),extract(
date from order_purchase_timestamp),day)
        as Total_days_for_delivery

from `Target_E_commerce.orders`
order by Total_days_for_Estimated_delivery asc,Total_days_for_delivery desc
```

```sql
--Calculate days between purchasing, delivering and estimated delivery

select  order_id,
        date_diff(extract(date from order_estimated_delivery_date),extract(date from order_purchase_timestamp),day)
        as Total_days_for_Estimated_delivery,
        date_diff(extract(date from order_delivered_customer_date),extract(date from order_purchase_timestamp),day)
        as Total_days_for_delivery

from `Target_E_commerce.orders`
order by Total_days_for_Estimated_delivery asc,Total_days_for_delivery desc;
```

Explanation:

To find the days between purchasing, delivery and estimated delivery, date_diff() function is used to calculate the number of days between the Order purchase time and Order delivered time. A similar calculation is applied to find the total days for estimated delivery

Query result:

Query results

SAVE RESULTS ▾ EXP

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	order_id	Total_days_for_Estimated_delivery	Total_days_for_delivery
1	e5215415bb6f76fe3b7cb6810...	2	null
2	a217507ac495620d9a403939...	3	11
3	be604de4ce5e2e8658c03895...	3	11
4	907424cd045f2dcb775507e3...	3	11
5	149bca3daf097d7695389efb9...	3	10
6	1c709af6dc28e45be0dda17b...	3	10
7	837b75362f8a7c08c85182dfd...	3	9
8	0e0170b61d0a01970751c69f...	3	9
9	35df2a75e8f319d4fe179f529...	3	9

Results per page: 50 ▾ 1 – 50 of 99441

As we can see one of the column have null value which indicates that data is incomplete in few of fields of table

5.2 Find time_to_delivery& diff_estimated_delivery.

Query:

```
with new_table as (
select
date_diff(extract(date from order_delivered_customer_date),extract(date from order_
purchase_timestamp),day) as time_to_delivery,
date_diff(extract(date from order_estimated_delivery_date),extract(date from order_
delivered_customer_date),day) as diff_estimated_delivery
from `Target_E_commerce.orders`)
select time_to_delivery , diff_estimated_delivery
from new_table
where time_to_delivery is not null and diff_estimated_delivery is not null;
```

```
--Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
with new_table as (
select
date_diff(extract(date from order_delivered_customer_date),extract(date from order_purchase_timestamp),day) as time_to_delivery,
date_diff(extract(date from order_estimated_delivery_date),extract(date from order_delivered_customer_date),day) as
diff_estimated_delivery
from `Target_E_commerce.orders`)
select time_to_delivery , diff_estimated_delivery
from new_table
where time_to_delivery is not null and diff_estimated_delivery is not null;
```

Explanation:

Here I have used with clause to calculate the time to delivery and diff_estimated_delivery and then in the main clause the difference between them is calculated. Since in some columns of the table have incomplete values a where condition is applied to remove null values

Query results:

Query results

SAVE RESULTS EXI

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	time_to_delivery	diff_estimated_delivery
1	30	-12
2	31	29
3	36	17
4	31	2
5	33	1
6	30	2
7	44	-4
8	41	-4
9	37	-1
10	34	-5
11	39	-6
12	36	-2
13	34	0

Results per page: 50 1 – 50 of 96476

5.3 group the data by state, take mean freight value, time to delivery, Diff_estimated_delivery

Query:

```
select customers.customer_state as state,
       Round(Avg(order_items.freight_value)) as Mean_freight_price,
date_diff(extract (date from order_purchase_timestamp),extract(date from order_deli
vered_customer_date),day) as time_to_delivery,
date_diff(extract(date from order_estimated_delivery_date),extract(date from order_
delivered_customer_date),day) as diff_estimated_delivery
from `Target_E_commerce.order_items` as order_items join `Target_E_commerce.orders`
 as orders
on order_items.order_id=orders.order_id join `Target_E_commerce.customers` as custo
mers
on orders.customer_id=customers.customer_id
group by state,time_to_delivery,diff_estimated_delivery
order by time_to_delivery desc,diff_estimated_delivery  desc
```

```sql
---Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

select  customers.customer_state as state,
        Round(Avg(order_items.freight_value)) as Mean_freight_price,
        date_diff(extract (date from order_delivered_customer_date),extract(date from order_purchase_timestamp),day)
        as time_to_delivery,
        date_diff(extract(date from order_estimated_delivery_date),extract(date from order_delivered_customer_date),day)
        as diff_estimated_delivery
from `Target_E_commerce.order_items` as order_items join `Target_E_commerce.orders` as orders
      on order_items.order_id=orders.order_id join `Target_E_commerce.customers` as customers
      on orders.customer_id=customers.customer_id
group by state,time_to_delivery,diff_estimated_delivery
having time_to_delivery is not null and diff_estimated_delivery is not null
order by time_to_delivery ,diff_estimated_delivery  ;
```

Explanation:

Since the data from multiple tables is required a join condition is applied on the three tables 1. Order_items 2.orders 3.customers a group by clause is applied to group the data in state wise

Query Results:

Query results

SAVE RESULTS EX

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	state	Mean_freight_pr	time_to_delivery	diff_estimated_delivery
5	SP	8.0	1	3
6	SP	9.0	1	4
7	RJ	8.0	1	4
8	RJ	9.0	1	5
9	SP	9.0	1	5
10	PR	46.0	1	5
11	SP	11.0	1	6
12	PR	20.0	1	6
13	MG	14.0	1	6

Results per page: 50 1 – 50 of 15397

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

Query:

```
select customers.customer_state,
 Round(Avg(order_items.freight_value)) as Mean_freight_price,
from `Target_E_commerce.order_items` as order_items join `Target_E_commerce.orders`
 as orders
      on order_items.order_id=orders.order_id join `Target_E_commerce.customers` as
 customers
      on orders.customer_id=customers.customer_id
group by customers.customer_state
order by Mean_freight_price desc
limit 5;
```

Explanation:

On total 3 columns were joined to get the data from the 3 tables, average aggregation is applied on the freight value column to find the mean freight value and the group by clause is applied on customer state column to group the data based on each state, limit clause is applied to find the top 5 states with highest mean freight value

Query Results:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	customer_state	Mean_freight_pr
1	PB	43.0
2	RR	43.0
3	RO	41.0
4	AC	40.0
5	PI	39.0

We can see top 5 states with highest mean freight price we can observe that state PB has the highest Mean freight price value

5.4.2 Top 5 states with highest/lowest average time to delivery

Query:

```
select customers.customer_state as state,
round(avg(date_diff(extract(date from order_delivered_customer_date),extract(date f
rom order_purchase_timestamp),day))) as Avg_time_to_delivery,
from `Target_E_commerce.orders` as orders join `Target_E_commerce.customers` as cus
tomers
 on orders.customer_id=customers.customer_id
group by state
order by Avg_time_to_delivery asc
limit 5;
```

RUN | SAVE | SHARE | SCHEDULE | MORE

```
--Top 5 states with highest/lowest average time to delivery

select customers.customer_state as state,
        round(avg(date_diff(extract(date from order_delivered_customer_date),
                            extract(date from order_purchase_timestamp),day))) as Avg_time_to_delivery,
from `Target_E_commerce.orders` as orders join `Target_E_commerce.customers` as customers
        on orders.customer_id=customers.customer_id
group by state
order by Avg_time_to_delivery asc
limit 5;
```

Explanation:

To find the highest average time to delivery I have joined two tables 1. Orders 2. Customers. From orders table I have extracted the number of days for delivering the order and then applied average aggregation on top of it.to get state wise average time to delivery group by clause is used and there query result is sorted based on the average time to deliver the order, at final limit clause is applied to limit the number of results to 5.

Query Results:

Query results

SAVE RESU

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH | PREVIEW

Row	state	Avg_time_to_del
1	SP	9.0
2	PR	12.0
3	MG	12.0
4	DF	13.0
5	RS	15.0

we can see that the query has returned top 5 states with lowest average time to deliver the order with state SP having lowest average time to deliver the order.

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

Query:

```
with new_table as
(
select  customer_id,order_id,
        date_diff(extract(date from order_estimated_delivery_date),extract(date fro
m order_purchase_timestamp),day)
        as Total_days_for_Estimated_delivery,
        date_diff(extract(date from order_delivered_customer_date),extract(date fro
m order_purchase_timestamp),day)
        as Total_days_for_delivery
from `Target_E_commerce.orders`
)

select C.customer_state as state, (Total_days_for_Estimated_delivery-
total_days_for_delivery) as days
```

```
from new_table as NT join `Target_E_commerce.customers` as C
      on NT.customer_id=C.customer_id
group by state,days
having days is not null
order by days  desc
limit 5;
```

RUN SAVE SHARE SCHEDULE MORE This query will proc

```
--Top 5 states where delivery is really fast/ not so fast compared to estimated date
with new_table as
(
select  customer_id,order_id,
        date_diff(extract(date from order_estimated_delivery_date),extract(date from order_purchase_timestamp),day)
        as Total_days_for_Estimated_delivery,
        date_diff(extract(date from order_delivered_customer_date),extract(date from order_purchase_timestamp),day)
        as Total_days_for_delivery
from `Target_E_commerce.orders`
)

select C.customer_state as state, (Total_days_for_Estimated_delivery-total_days_for_delivery) as days
from new_table as NT join `Target_E_commerce.customers` as C
      on NT.customer_id=C.customer_id
group by state,days
having days is not null
order by days  desc
limit 5;
```

Explanation:

The I approach I followed for this is, to know the state with fastest time to deliver an order I have first wrote a calculation for time to deliver an order and estimated time to deliver an order then in the main query the difference between the number of days to deliver an order and estimated days to deliver an order is calculated. And finally if the difference is too large then we can say that it's a fastest delivery. The approach is further explained below example. Group by clause is applied to get the results state wise and limit clause to limit the top 5 states

Ex1:	Ex2:
Total days took for order delivery=2days	Total days took for order delivery=5
Estimated days for delivery=6	Estimated days for delivery=6
Difference= 6-2=4 days	Difference=6-5 =1day

We can see that in example 1 the order is delivered with in 2days but the estimated days were 6 days(difference is high→ fastest). In example 2 the order is delivered in 5 days estimated delivery time is 6 days (difference is low→ slowest)

Query Results:

Query results

SAVE RESULT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	state	days
1	SP	147
2	MA	140
3	RS	135
4	SP	124
5	RJ	109

As we can see top 5 states where most of the fastest deliveries happened hence we can conclude State SP is the one where delivery is too fast.

6.1 Month over Month count of orders for different payment types

Query:

```
select extract(month from O.order_purchase_timestamp) as month,P.payment_ty
pe,count(P.order_id) as orders
from `Target_E_commerce.orders` as O join `Target_E_commerce.payments` as P
on O.order_id=P.order_id
group by month, P.payment_type
order by month
```

RUN SAVE SHARE SCHEDULE MORE This query

```
--Month over Month count of orders for different payment types
select extract(month from O.order_purchase_timestamp) as month,
        P.payment_type,count(P.order_id) as orders
from `Target_E_commerce.orders` as O join `Target_E_commerce.payments` as P
    on O.order_id=P.order_id
group by month, P.payment_type
order by month;
```

Explanation:

To get month over month count of orders from different payment types first 2 columns were joined 1.orders(to extract the month) 2. Payments (to find the count of orders for different payment types). Group by clause is applied on the month and payment_types to get monthly orders made from different payment types.

Query Results:

Query results SAVE RESULTS EXI

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	month	payment_type	orders
1	1	credit_card	6103
2	1	UPI	1715
3	1	voucher	477
4	1	debit_card	118
5	2	UPI	1723
6	2	credit_card	6609
7	2	voucher	424
8	2	debit_card	82
9	3	credit_card	7707
10	3	UPI	1942
11	3	debit_card	109
12	3	voucher	591
13	4	voucher	572

Results per page: 50 1 – 50 of 50

We can see that the query has fetched monthly count of order for different payment types. If we sort the query based on the order count we get the below results

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	month	payment_type	orders
1	5	credit_card	8350
2	8	credit_card	8269
3	7	credit_card	7841
4	3	credit_card	7707
5	4	credit_card	7301
6	6	credit_card	7276
7	2	credit_card	6609
8	1	credit_card	6103
9	11	credit_card	5897
10	12	credit_card	4378
11	10	credit_card	3778

Results per page: 50 1

As we can see in month of May most number of orders were made through credit- card payment type

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

Query:

```
select payment_installments,count(order_id)
from `Target_E_commerce.payments`
group by payment_installments
order by payment_installments
```

RUN SAVE SHARE SCHEDULE MORE

```
--Count of orders based on the no. of payment installments
select payment_installments,count(order_id) Count_of_orders
from `Target_E_commerce.payments`
group by payment_installments
order by payment_installments;
```

Explanation:

To get the number of orders based on the no. of payment instalments simply a group by clause is applied on the payment_intallments and count aggregation is applied on the order_id column. The result is sorted based on the payment_installments

Query Results:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	payment_installments	Count_of_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328
12	11	23
13	12	133

Results per page: 50 1 – 24 of 24

From the query results we can observe that on total 24 different installments were there and the query fetched the count of orders for different installment types. If we sort the data based on the count of orders the query fetched below results

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	payment_installı	Count_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

Results per page: 50 1 – 24 of 24

We can see in payment installment 1 more number of orders(52546) were placed and in payment installment 23 least number of orders were placed (only 1 order)

7. Actionable Insights

The business people should have a look on, in which months the lowest number of orders were made and in which month highest number of orders made and act accordingly also in few states of Bazil the delivery time is too high to mitigate this a plan can be made based on subscription type like if a customer belong to Premium subscription he/she supposed to receive the order as fast as possible

8. Recommendations

Customer satisfaction is at most priority in any business. It is recommended to consider the customer feedback once order has been delivered. In the state of RR the average time to deliver an order is 29 days & the same state has least number of customers so it is recommended to minimise this time period