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

a. Data type of columns in a table

Solution –

- **The Data Type of ORDERS TABLE – STRING, TIMESTAMP**.

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

- The **Data Type of CUSTOMERS TABLE – STRING, INTEGER.**

Filter Enter property name or value

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

- The **Data Type of GEOLOCATION TABLE – INTEGER,FLOAT,STRING.**

Filter Enter property name or value

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

- The **Data Type of ORDER_ITEM TABLE – INTEGER,STRING,FLOAT,TIMESTAMP.**

Filter Enter property name or value

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

- The **Data Type of ORDER_REVIEWS TABLE – STRING,INTEGER,TIMESTAMP.**

Filter Enter property name or value

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

- The **Data Type of PAYMENTS TABLE – STRING,INTERGER,FLOAT.**

Filter Enter property name or value

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

- The **Data Type of PRODUCTS TABLE – STRING,INTEGER.**

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

- The **Data Type of SELLERS TABLE – STRING, INTEGER**

Filter Enter property name or value

Field name	Type
seller_id	STRING
seller_zip_code_prefix	INTEGER
seller_city	STRING
seller_state	STRING

b. Time period for which the data is given

Solution -

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

OUTPUT

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	f0_	f1_
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

Solution:

```
select DISTINCT c.customer_city, c.customer_state
from `Target_SQL.customers` as c
join  `Target_SQL.orders` as o on c.customer_id = o.customer_id
```

OUTPUT

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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
10	uba	MG

2. In-depth Exploration:

a. 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?

Solution –

```
select
extract (year from order_purchase_timestamp) as YEAR,
extract (month  from order_purchase_timestamp) as MONTH,
count (order_id) as count_of_orders
from `Target_SQL.orders
group by YEAR, MONTH
order by YEAR, MONTH, count_of_orders;
```

OUTPUT

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	YEAR	MONTH	count_of_orders
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

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

Solution -

```
select
count(order_id) Orders,
CASE
when extract(hour from order_purchase_timestamp) between 0 and 6
then "Dawn"
when extract(hour from order_purchase_timestamp) between 7 and 12
then 'Morning'
when extract(hour from order_purchase_timestamp) between 13 and 18 then 'Afternoon'
else 'Night'
 end as Order_Time
from `Target_SQL.orders`
```

```
group by Order_Time
order by Orders desc;
```

OUTPUT

JOB INFORMATION | RESULTS

Row	Orders	Order_Time
1	38135	Afternoon
2	28331	Night

INSIGHTS

- E-commerce on Brazil really has a growing trend along the time. We can see some seasonality with peaks at specific months, but in general we can see some clear view that customers are more to buy things online than before.

RECOMMENDATION

- "As there were increased orders in in afternoon at Brazil region I would recommend the company to show more ads during the afternoon, which will help company to generate more sales and I also recommend the company to lower the number of adds to the cost at dawn period as there are less sales compared to morning, afternoon and night, and same cost can be utilized to run more ads in afternoon."

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

a. Get month on month orders by states

Solution –

```
select c.customer_state,
extract (year from order_purchase_timestamp) as YEAR,
extract (month  from order_purchase_timestamp) as MONTH,
count(o.order_id) as Orders
```

```
from `Target_SQL.orders` as o
join `Target_SQL.customers` as c on o.customer_id = c.customer_id
group by YEAR, MONTH, c.customer_state
order by Orders desc
```

Query results

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

Row	customer_state	YEAR	MONTH	Orders
1	SP	2018	8	3253
2	SP	2018	5	3207
3	SP	2018	4	3059
4	SP	2018	1	3052
5	SP	2018	3	3037
6	SP	2017	11	3012
7	SP	2018	7	2777
8	SP	2018	6	2773
9	SP	2018	2	2703
10	SP	2017	12	2357
11	SP	2017	10	1793

3.b. Distribution of customers across the states in Brazil

Solution:

```
select count(customer_id)as count, customer_state
from `Target_SQL.customers`
group by customer_state
```

OUTPUT

Query results

JOB INFORMATION | RESULTS | JSON

Row	count	customer_state
1	485	RN
2	1336	CE
3	5466	RS
4	3637	SC
5	41746	SP
6	11635	MG
7	3380	BA
8	12852	RJ
9	2020	GO
10	747	MA
11	1652	PE
12	536	PB
13	2033	ES

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

a. 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

Solution

```
with CTE as( select  p.payment_value as cost_of_orders, o.order_purchase_timestamp as order_date
from `Target_SQL.orders` o
join `Target_SQL.payments` p  on o.order_id= p.order_id)
select a.cost_of_orders_2017 as _2017,
       b.cost_of_orders_2018 as _2018,
 round(((b.cost_of_orders_2018-a.cost_of_orders_2017)/a.cost_of_orders_2017)*100, 2) as

 Increase_perc_from_2017_to_2018  from(
   (select  round(sum(cost_of_orders),2) as cost_of_orders_2017
   from CTE
   where order_date between "2017-01-01 00:00:00" and "2017-08-31 23:59:59")a
   cross join (select  round(sum(cost_of_orders),2) as cost_of_orders_2018
```

```
    from    CTE
    where order_date between "2018-01-01 00:00:00" and "2018-08-31 23:59:59")b
);
```

OUTPUT

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	_2017	_2018	Increase_perc_from_
1	3669022.12	8694733.84	136.98

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

Solution

```
select c.customer_state, avg(price) as mean_price, sum(price) as price_sum,
 avg(freight_value)as mean_freight, sum(freight_value) as price_freight
from `Target_SQL.order_item` oi
 left join `Target_SQL.orders` o on oi.order_id = o.order_id
left join `Target_SQL.customers` c on o.customer_id = c.customer_id
group by c.customer_state
order by 2;
```

OUTPUT

Query results

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

Row	customer_state	mean_price	price_sum	mean_freight	price_freight
1	SP	109.6536291597...	5202955.050002...	15.14727539041...	718723.0699999...
2	PR	119.0041393728...	683083.7600000...	20.53165156794...	117851.6800000...
3	RS	120.3374530874...	750304.0200000...	21.73580433039...	135522.7400000...
4	MG	120.7485741488...	1585308.029999...	20.63016680630...	270853.4600000...
5	ES	121.9137012411...	275037.3099999...	22.05877659574...	49764.59999999...
6	SC	124.6535775862...	520553.3400000...	21.47036877394...	89660.26000000...
7	RJ	125.1178180945...	1824092.669999...	20.96092393168...	305589.3100000...
8	DF	125.7705486284...	302603.9399999...	21.04135494596...	50625.49999999...

INSIGHTS

- Increase in sales with a c good amount is good for TARGET. So Brazil is the profitable region and TARGET should focus more on generating the revenue in Brazil.

RECOMMENDATION

- Overall Brazil region has provided good business to company. Target should plan to improve and maintain its brand in this region.

5. Analysis on sales, freight and delivery time

a. Calculate days between purchasing, delivering and estimated delivery

Solution

```
select distinct order_id, date_diff(order_delivered_customer_date,
order_purchase_timestamp,day) as diff_cust_purc_date,
            date_diff(order_delivered_customer_date,
order_estimated_delivery_date,day) as diff_cust_est_date,
            date_diff(order_purchase_timestamp, order_estimated_delivery_date,day) as
diff_puc_est_date, order_status

from `Target_SQL.orders`
--where order_delivered_customer_date is not null and order_purchase_timestamp is not
null and order_estimated_delivery_date is not null
order by order_id
```

OUTPUT

Query results

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

Row	order_id	diff_cust_purc_date	diff_cust_est_date	diff_puc_est_date	order_status
1	00010242fe8c5a6d1ba2dd792...	7	-8	-15	delivered
2	00018f77f2f0320c557190d7a1...	16	-2	-18	delivered
3	000229ec398224ef6ca0657da...	7	-13	-21	delivered
4	00024acbcdf0a6daa1e931b03...	6	-5	-11	delivered
5	00042b26cf59d7ce69dfabb4e...	25	-15	-40	delivered
6	00048cc3ae777c65dbb7d2a06...	6	-14	-21	delivered
7	00054e8431b9d7675808bcb8...	8	-16	-24	delivered
8	000576fe39319847cbb9d288c...	5	-15	-20	delivered

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

- time_to_delivery = order_delivered_customer_date-order_purchase_timestamp
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

Solution

```
select order_id,
extract(day from Date_of_delivering - Date_of_puchasing)as time_to_delivery,
extract(day from Date_of_delivering - Date_estimated_delivery)as
diff_estimated_delivery
from(
  select order_id,
  extract(date from o.order_purchase_timestamp)as Date_of_puchasing,
  extract(date from o.order_delivered_customer_date)as Date_of_delivering,
  extract(date from o.order_estimated_delivery_date)as Date_estimated_delivery,
  from `Target_SQL.orders` o)
```

OUTPUT

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_delive
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	31	-29
3	65d1e226dfaeb8cdc42f66542...	36	-17
4	635c894d068ac37e6e03dc54e...	31	-2
5	3b97562c3aee8bdedcb5c2e45...	33	-1
6	68f47f50f04c4cb6774570cfde...	30	-2
7	276e9ec344d3bf029ff83a161c...	44	4
8	54e1a3c2b97fb0809da548a59...	41	4

c. Sort the data to get the following:

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

Solution IN ASC

```
select c.customer_state,avg(date_diff( o.order_delivered_customer_date,
o.order_purchase_timestamp,day)) as time_to_delivery
,avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day))
as diff_estimated_delivery, avg(oi.freight_value) as Avg_freight_value from
`Target_SQL.orders` o join `Target_SQL.order_item` oi
on o.order_id=oi.order_id
join `Target_SQL.customers` c on c.customer_id=o.customer_id
GROUP BY c.customer_state
order by Avg_freight_value
limit 5
```

OUTPUT in ASC

Query results

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

Row	customer_state	time_to_delivery	diff_estimated_delive	Avg_freight_value
1	SP	8.259608552419...	10.26559438451...	15.14727539041...
2	PR	11.48079306071...	12.53389980527...	20.53165156794...
3	MG	11.51552218007...	12.39715104126...	20.63016680630...
4	RJ	14.68938215750...	11.14449314293...	20.96092393168...
5	DF	12.50148619957...	11.27473460721...	21.04135494596...

Solution in DESC

```
select c.customer_state,avg(date_diff( o.order_delivered_customer_date,
o.order_purchase_timestamp,day)) as time_to_delivery
,avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day))
as diff_estimated_delivery, avg(oi.freight_value) as Avg_freight_value from
`Target_SQL.orders` o join `Target_SQL.order_item` oi
on o.order_id=oi.order_id
join `Target_SQL.customers` c on c.customer_id=o.customer_id
GROUP BY c.customer_state
order by Avg_freight_value desc
limit 5
```

OUTPUT in DESC

Query results

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

Row	customer_state	time_to_delivery	diff_estimated_delive	Avg_freight_value
1	RR	27.82608695652...	17.43478260869...	42.98442307692...
2	PB	20.11945392491...	12.15017064846...	42.72380398671...
3	RO	19.28205128205...	19.08058608058...	41.06971223021...
4	AC	20.32967032967...	20.01098901098...	40.07336956521...
5	PI	18.93116634799...	10.68260038240...	39.14797047970...

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

Solution in ASC

```
select c.customer_state,avg(date_diff( o.order_delivered_customer_date,
o.order_purchase_timestamp,day)) as time_to_delivery,
avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day))
as diff_estimated_delivery, avg(oi.freight_value) as Avg_freight_value
from `Target_SQL.orders` o join `Target_SQL.order_item` oi
on o.order_id=oi.order_id
join `Target_SQL.customers` c on c.customer_id=o.customer_id
GROUP BY c.customer_state
order by time_to_delivery
limit 5;
```

OUTPUT IN ASC

Query results

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

Row	customer_state	time_to_delivery	diff_estimated_delive	Avg_freight_value
1	SP	8.259608552419...	10.26559438451...	15.14727539041...
2	PR	11.48079306071...	12.53389980527...	20.53165156794...
3	MG	11.51552218007...	12.39715104126...	20.63016680630...
4	DF	12.50148619957...	11.27473460721...	21.04135494596...
5	SC	14.52098584675...	10.66886285993...	21.47036877394...

Solution in DESC

```
select c.customer_state,avg(date_diff( o.order_delivered_customer_date,
o.order_purchase_timestamp,day)) as time_to_delivery,
avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day))
as diff_estimated_delivery, avg(oi.freight_value) as Avg_freight_value
```

```
from `Target_SQL.orders` o join `Target_SQL.order_item` oi
on o.order_id=oi.order_id
join `Target_SQL.customers` c on c.customer_id=o.customer_id
GROUP BY c.customer_state
order by time_to_delivery desc
limit 5;
```

OUTPUT IN DESC

Query results

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

Row	customer_state	time_to_delivery	diff_estimated_delive	Avg_freight_value
1	RR	27.82608695652...	17.43478260869...	42.98442307692...
2	AP	27.75308641975...	17.44444444444...	34.00609756097...
3	AM	25.96319018404...	18.97546012269...	33.20539393939...
4	AL	23.99297423887...	7.976580796252...	35.84367117117...
5	PA	23.30170777988...	13.37476280834...	35.83268518518...

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

Solution in ASC

```
select c.customer_state,avg(date_diff( o.order_delivered_customer_date,
o.order_purchase_timestamp,day)) as time_to_delivery
,avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day))
as diff_estimated_delivery, avg(oi.freight_value) as Avg_freight_value
from `Target_SQL.orders` o
join `Target_SQL.order_item` oi on o.order_id=oi.order_id
join `Target_SQL.customers` c on c.customer_id=o.customer_id
GROUP BY c.customer_state
order by diff_estimated_delivery
limit 5;
```

OUTPUT IN ASC

Query results

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

Row	customer_state	time_to_delivery	diff_estimated_delive	Avg_freight_value
1	AL	23.99297423887...	7.976580796252...	35.84367117117...
2	MA	21.20375000000...	9.109999999999...	38.25700242718...
3	SE	20.97866666666...	9.165333333333...	36.65316883116...
4	ES	15.19280898876...	9.768539325842...	22.05877659574...
5	BA	18.77464023893...	10.11946782514...	26.36395893656...

Solution in DESC

```
select c.customer_state,avg(date_diff( o.order_delivered_customer_date,
o.order_purchase_timestamp,day)) as time_to_delivery
,avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day))
as diff_estimated_delivery, avg(oi.freight_value) as Avg_freight_value
from `Target_SQL.orders` o
join `Target_SQL.order_item` oi on o.order_id=oi.order_id
join `Target_SQL.customers` c on c.customer_id=o.customer_id
GROUP BY c.customer_state
order by diff_estimated_delivery desc
limit 5;
```

OUTPUT IN DESC

Query results

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

Row	customer_state	time_to_delivery	diff_estimated_delive	Avg_freight_value
1	AC	20.32967032967...	20.01098901098...	40.07336956521...
2	RO	19.28205128205...	19.08058608058...	41.06971223021...
3	AM	25.96319018404...	18.97546012269...	33.20539393939...
4	AP	27.75308641975...	17.44444444444...	34.00609756097...
5	RR	27.82608695652...	17.43478260869...	42.98442307692...

INSIGHTS

- There are some cities with high fright values and very estimated delivery time.

RECOMMENDATION

TARGET should do the following

- Improvise of your supply chain

- Ensure quality project management
- Optimizing transportation with technology
- Consolidate purchases
- Streamline warehouse processes, Work with trusted suppliers.

6. Payment type analysis:

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

Solution

```
select Year, Month,payment_type, No_of_order
from(
  select payment_type,
  extract (Year from o.order_purchase_timestamp) as Year,
  extract (month from o.order_purchase_timestamp) as month,
  count(o.order_id) as No_of_order
  from `Target_SQL.payments` p
  join `Target_SQL.orders` o on p.order_id = o.order_id
  join `Target_SQL.customers` c on c.customer_id = o.customer_id
  group by Year, Month,payment_type)
  order by Year, Month;
```

OUTPUT

Query results

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

Row	Year	Month	payment_type	No_of_order
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197

8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9
11	2017	2	credit_card	1356
12	2017	2	UPI	398

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

Solution

```
select count(order_id) as Volume, payment_installments
from `Target_SQL.payments`
group by payment_installments
```

OUTPUT

Query results

JOB INFORMATION | RESULTS | JSON

Row	Volume	payment_installment
1	2	0
2	52546	1
3	12413	2
4	10461	3
5	7098	4
6	5239	5
7	3920	6
8	1626	7

INSIGHTS

- Maximum number of payments are made using credit cards and next highest used payment mode is UPI. Maximum number instalments type used is type 1.

RECOMMENDATION

- By showing adds and by providing information to these category customers regarding sales and offers, and also sometimes by providing loyalty bonuses more repeated orders can be achieved and hence increase in sales can be achieved.