Q1 Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

Part (1):Data type of columns in a table:

Ans: For example, data type from orders table:

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
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				

Part (2):Time period for which the data is given:
Query Code:

```
select
min(order_purchase_timestamp) as min_time_stamp,
max(order_purchase_timestamp) as max_time_stamp
from `scaler-dsml-sql-373605.target_sql.orders`
```

Query Result:

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

Part (3): Cities and States of customers ordered during the given period:
Query Code:

```
select
distinct
customer_state,
customer_city
from `scaler-dsml-sql-373605.target_sql.customers`
order by customer_state,customer_city
```

Query Result:

Row	customer_state	customer_city
1	AC	brasileia
2	AC	cruzeiro do sul
3	AC	epitaciolandia
4	AC	manoel urbano
5	AC	porto acre
6	AC	rio branco
7	AC	senador guiomard
8	AC	xapuri
9	AL	agua branca
10	AL	anadia

Q:2 In-depth Exploration:

Part (1) (i) : Is there a growing trend on e-commerce in Brazil?
Query Code:

```
select
extract (year from timestamp (o.order_purchase_timestamp )) as year,
extract (month from timestamp (o.order_purchase_timestamp )) as month,
count(o.order_id) as cnt_ord
from `scaler-dsml-sql-373605.target_sql.orders` as o
inner join `scaler-dsml-sql-373605.target_sql.customers`as c
on o.customer_id=c.customer_id
group by year,month
order by year,month
```

Query Result:

Row	year	month	cnt_ord
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

Part(1) (ii) How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?
Query Code:

```
with cte_1 as(
select
extract (year from timestamp (o.order_purchase_timestamp )) as year,
extract (month from timestamp (o.order_purchase_timestamp )) as month,
count(o.order_id) as cnt_ord
from `scaler-dsml-sql-373605.target_sql.orders` as o
inner join `scaler-dsml-sql-373605.target_sql.customers`as c
on o.customer_id=c.customer_id
group by year,month
order by year,month
),
cte_2 as (
select
year,
month,
cnt_ord
from cte_1
where year=2016
),
cte_3 as(
select
year,
month,
cnt_ord
from cte_1
where year=2017
),
cte_4 as (
select
year,
month,
cnt_ord
from cte_1
where year=2018
)
select
ct3.month,
ifnull(ct2.cnt_ord,0) as cnt_ord_2016,
ifnull(ct3.cnt_ord,0) as cnt_ord_2017,
ifnull(ct4.cnt_ord,0) as cnt_ord_2018,
(ifnull(ct2.cnt_ord,0)+ifnull(ct3.cnt_ord,0)+ifnull(ct4.cnt_ord,0)) as total_cnt
_ord
from cte_3 as ct3
left join cte_2 as ct2
on ct3.month=ct2.month
left join cte_4 as ct4
on ct3.month=ct4.month
order by ct3.month
```

Query Result:

Row	month	cnt_ord_2016	cnt_ord_2017	cnt_ord_2018	total_cnt_ord
1	1	0	800	7269	8069
2	2	0	1780	6728	8508
3	3	0	2682	7211	9893
4	4	0	2404	6939	9343
5	5	0	3700	6873	10573
6	6	0	3245	6167	9412
7	7	0	4026	6292	10318
8	8	0	4331	6512	10843
9	9	4	4285	16	4305
10	10	324	4631	4	4959
11	11	0	7544	0	7544
12	12	1	5673	0	5674

Part (2): What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

Here, I took assumptions like

- Dawn period is 00:00:00 to 05:59:59
- Morning period is 06:00:00 to 11:59:59
- Afternoon period is 12:00:00 to 17:59:59
- Night period is 18:00:00 to 23:59:59

Query Code:

```
select distinct
Hour_stamp,
count(order_id) over(partition by hour_stamp) as cnt_ord
from (select
customer_id,
order_id,
case
when extract (hour from timestamp (order_purchase_timestamp) )>=1 and extract (h
our from timestamp (order_purchase_timestamp) )<6 then "dawn"
when extract (hour from timestamp (order_purchase_timestamp) )>=6 and extract (h
our from timestamp (order_purchase_timestamp) )<12 then "Morning"
when extract (hour from timestamp (order_purchase_timestamp) )>=12 and extract (
hour from timestamp (order_purchase_timestamp) )<18 then "Afternoon"
else "Night"
End as Hour_stamp from `scaler-dsml-sql-373605.target_sql.orders`) t
inner join `scaler-dsml-sql-373605.target_sql.customers` c
on t.customer_id=c.customer_id
order by cnt_ord
```

Query Result:

Row	Hour_stamp	cnt_ord
1	dawn	2346
2	Morning	22240
3	Night	36494
4	Afternoon	38361

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

Part (1): Get month on month orders by states

Query Code:

```
select
c.customer_state,
extract (year from timestamp (o.order_purchase_timestamp)) as year,
extract (month from timestamp (o.order_purchase_timestamp)) as month,
count(o.order_id) as ord_cnt,
from `scaler-dsml-sql-373605.target_sql.orders` o
inner join `scaler-dsml-sql-373605.target_sql.customers` c
on o.customer_id=c.customer_id
group by c.customer_state,year,month
order by year,month,ord_cnt desc
```

Query Result:

Row	customer_state	year	month	ord_cnt
1	SP	2016	9	2
2	RR	2016	9	1
3	RS	2016	9	1
4	SP	2016	10	113
5	RJ	2016	10	56
6	MG	2016	10	40
7	RS	2016	10	24
8	PR	2016	10	19
9	SC	2016	10	11
10	GO	2016	10	9

Part (2): Distribution of customers across the states in Brazil

Query Code:

```
select
customer_state,
count(customer_id) as cnt_cid
from `scaler-dsml-sql-373605.target_sql.customers`
group by customer_state
order by cnt_cid desc
```

Query Result:

Row	customer_state	cnt_cid
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

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

Part (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

Query Code:

```
with cte_1 as (
select
extract (year from timestamp(order_purchase_timestamp)) as year,
order_id,
order_purchase_timestamp
from `scaler-dsml-sql-373605.target_sql.orders`
),
cte_2 as (
select
order_id,
order_purchase_timestamp,
extract (month from timestamp(order_purchase_timestamp)) as month
from cte_1
```

```sql
where extract (month from timestamp(order_purchase_timestamp)) between 1 and
 8
),
cte_3 as (
select distinct
ct1.year,
ct2.month,
count(distinct py.order_id) over(partition by ct2.month) as cnt_ord,
round(sum(py.payment_value) over(partition by ct2.month),2) as sum_month_pay
ment_value,
round(avg(py.payment_value) over(partition by ct2.month),2) as avg_month_pay
ment_value
from cte_2 as ct2
inner join `scaler-dsml-sql-373605.target_sql.payments` as py
on ct2.order_id=py.order_id
inner join cte_1 as ct1
on ct1.order_id=py.order_id
where ct1.year=2017
order by ct2.month
),
cte_4 as (
select distinct
ct1.year,
ct2.month,
count(distinct py.order_id) over(partition by ct2.month) as cnt_ord,
round(sum(py.payment_value) over(partition by ct2.month),2) as sum_month_pay
ment_value,
round(avg(py.payment_value) over(partition by ct2.month),2) as avg_month_pay
ment_value
from cte_2 as ct2
inner join `scaler-dsml-sql-373605.target_sql.payments` as py
on ct2.order_id=py.order_id
inner join cte_1 as ct1
on ct1.order_id=py.order_id
where ct1.year=2018
order by ct2.month
)
select
concat(ct3.year,"-",ct4.year) as years,
ct3.month,
ct4.cnt_ord as cnt_ord_2018,
ct3.cnt_ord as cnt_ord_2017,
round(((ct4.cnt_ord-ct3.cnt_ord)/ct3.cnt_ord)*100,2) as percent_ord_change,
ct4.sum_month_payment_value as sum_month_payment_value_2018,
ct3.sum_month_payment_value as sum_month_payment_value_2017,
round(((ct4.sum_month_payment_value-
ct3.sum_month_payment_value)/ct3.sum_month_payment_value)*100,2) as percent_
sum_PV_change,
ct4.avg_month_payment_value as avg_month_payment_value_2018,
ct3.avg_month_payment_value as avg_month_payment_value_2017,
round(((ct4.avg_month_payment_value-
ct3.avg_month_payment_value)/ct3.avg_month_payment_value)*100,2) as percent_
avg_PV_change
from cte_3 as ct3
inner join cte_4 as ct4
on ct3.month=ct4.month
order by ct3.month
```

Query Result:

Row	years	month	cnt_ord_2018	cnt_ord_2017	percent_ord_change	sum_month_payr	sum_month_payn	percent_sum_PY	avg_month_payn	avg_month_payn	percent_avg_PV
1	2017-2018	1	7269	800	808.63	1115004.18	138488.04	705.13	147.43	162.93	-9.51
2	2017-2018	2	6728	1780	277.98	992463.34	291908.01	239.99	142.76	154.78	-7.77
3	2017-2018	3	7211	2682	168.87	1159652.12	449863.6	157.78	154.37	158.57	-2.65
4	2017-2018	4	6939	2404	188.64	1160785.48	417788.03	177.84	161.02	162.5	-0.91
5	2017-2018	5	6873	3700	85.76	1153982.15	592918.82	94.63	161.74	150.33	7.59
6	2017-2018	6	6167	3245	90.05	1023880.5	511276.38	100.26	159.51	148.8	7.2
7	2017-2018	7	6292	4026	56.28	1066540.75	592382.92	80.04	163.91	137.22	19.45
8	2017-2018	8	6512	4331	50.36	1022425.32	674396.32	51.61	152.65	148.22	2.99

Part (2): Mean & Sum of price and freight value by customer state

Query Code:

```
with cte_1 as (
select
order_id,
customer_id
from `scaler-dsml-sql-373605.target_sql.orders`
),
cte_2 as (
select
c.customer_state,
c.customer_id
from `scaler-dsml-sql-373605.target_sql.customers` c
inner join `scaler-dsml-sql-373605.target_sql.orders` o
on c.customer_id= o.customer_id
),
cte_3 as (
select
o.order_id,
o.customer_id,
oe.price,
oe.freight_value,
from `scaler-dsml-sql-373605.target_sql.orders`o
inner join `scaler-dsml-sql-373605.target_sql.order_items` oe
on oe.order_id=o.order_id
)
select
distinct
ct2.customer_state,
round(avg(ct3.price) over(partition by ct2.customer_state ),2) as avg_price,
round(sum(ct3.price) over(partition by ct2.customer_state ),2) as sum_price,
round(avg(ct3.freight_value) over(partition by ct2.customer_state ),2) as av
g_freight,
round(sum(ct3.freight_value) over(partition by ct2.customer_state ),2) as su
m_freight,
from cte_2 as ct2
left join cte_3 as ct3
on ct2.customer_id=ct3.customer_id
order by avg_price,avg_freight,sum_price,sum_freight
```

Query Result:

Row	customer_state	avg_price	sum_price	avg_freight	sum_freight
1	SP	109.65	5202955.05	15.15	718723.07
2	PR	119.0	683083.76	20.53	117851.68
3	RS	120.34	750304.02	21.74	135522.74
4	MG	120.75	1585308.03	20.63	270853.46
5	ES	121.91	275037.31	22.06	49764.6
6	SC	124.65	520553.34	21.47	89660.26
7	RJ	125.12	1824092.67	20.96	305589.31
8	DF	125.77	302603.94	21.04	50625.5
9	GO	126.27	294591.95	22.77	53114.98
10	BA	134.6	511349.99	26.36	100156.68

Q:5 Analysis on sales, freight, and delivery time

Part (1): Calculate days between purchasing, delivering and estimated delivery

Query Code:

```
select
order_id,
DATE_DIFF(DATE_4,DATE_1,day) as expected_days,
DATE_DIFF(DATE_3,DATE_1,day) as actual_days,
DATE_DIFF(DATE_2,DATE_1,day) as dispatch_days
from (select
order_id,
extract(date from timestamp(order_purchase_timestamp)) as DATE_1,
extract(date from timestamp(order_delivered_carrier_date)) as DATE_2,
extract(date from timestamp(order_delivered_customer_date)) as DATE_3,
extract(date from timestamp(order_estimated_delivery_date)) as DATE_4,
from `scaler-dsml-sql-373605.target_sql.orders`
) as t
where DATE_1 is not null and DATE_2 is not null and DATE_3 is not null and D
ATE_4 is not null
```

Query Result:

Row	order_id	expected_days	actual_days	dispatch_days
1	770d331c84e5b214bd9dc70a1...	53	7	4
2	2c45c33d2f9cb8ff8b1c86cc28...	60	31	5
3	dabf2b0e35b423f94618bf965f...	52	7	4
4	8beb59392e21af5eb9547ae1a...	53	11	6
5	65d1e226dfaeb8cdc42f66542...	53	36	22
6	cec8f5f7a13e5ab934a486ec9e...	62	21	18
7	58527ee4726911bee84a0f42c...	59	10	2
8	10ed5499d1623638ee810eff1...	58	28	14
9	818996ea247803ddc123789f2...	45	9	2
10	d195cac9ccaa1394ede717d38...	53	11	2

Part (2): Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

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

Query code:

```
select
order_id,
DATETIME_DIFF(DATETIME_2,DATETIME_1,day) as time_to_delivery,
DATETIME_DIFF(DATETIME_3,DATETIME_2,day) as diff_estimated_delivery
from
(select
order_id,
extract(DATETIME  from timestamp(order_purchase_timestamp)) as DATETIME_1
,
extract(DATETIME  from timestamp(order_delivered_customer_date)) as DATET
IME_2,
extract(DATETIME  from timestamp(order_estimated_delivery_date)) as DATET
IME_3,
from `scaler-dsml-sql-373605.target_sql.orders`
) as t
where DATETIME_1 is not null and DATETIME_2 is not null and DATETIME_3 is
 not null
```

Query Result:

Row	order_id	time_to_delivery	diff_estimated_d
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
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	34	-5

Part (3) Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Query code:

```
with cte_1 as (
select
order_id,
customer_id,
DATETIME_DIFF(DATETIME_2,DATETIME_1,day) as time_to_delivery,
DATETIME_DIFF(DATETIME_3,DATETIME_2,day) as diff_estimated_delivery
from (select
order_id,
customer_id,
extract(DATETIME  from timestamp(order_purchase_timestamp)) as DATETIME_1,
extract(DATETIME  from timestamp(order_delivered_customer_date)) as DATETIME
_2,
extract(DATETIME  from timestamp(order_estimated_delivery_date)) as DATETIME
_3,
from `scaler-dsml-sql-373605.target_sql.orders`
) as t
where DATETIME_1 is not null and DATETIME_2 is not null and DATETIME_3 is no
t null
)
select
distinct
c.customer_state,
round(avg(oe.freight_value) over(partition by c.customer_state),2) as avg_fr
eight,
round(avg(ct1.time_to_delivery) over(partition by c.customer_state),2) as av
g_time_to_delivery,
round(avg(ct1.diff_estimated_delivery) over(partition by c.customer_state),2
) as avg_diff_estimated_delivery
from `scaler-dsml-sql-373605.target_sql.customers` as c
```

```
inner join cte_1 as ct1
on ct1.customer_id=c.customer_id
inner join `scaler-dsml-sql-373605.target_sql.order_items` as oe
on oe.order_id=ct1.order_id
order by c.customer_state,avg_freight,avg_time_to_delivery,avg_diff_estimate
d_delivery desc
```

Query Result:

Row	customer_state	avg_freight	avg_time_to_deli	avg_diff_estimat
1	AC	40.05	20.68	20.98
2	AL	35.87	24.45	8.74
3	AM	33.31	26.34	19.93
4	AP	34.16	28.22	18.4
5	BA	26.49	19.19	10.98
6	CE	32.73	20.92	11.1
7	DF	21.07	12.89	12.2
8	ES	22.03	15.59	10.65
9	GO	22.56	15.34	12.29
10	MA	38.49	21.59	9.91

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

Query Code:

```
select
distinct
c.customer_state,
round(avg(oe.freight_value) over(partition by c.customer_state),2) as avg_fr
eight,
from `scaler-dsml-sql-373605.target_sql.orders` as o
inner join `scaler-dsml-sql-373605.target_sql.customers` as c
on c.customer_id=o.customer_id
inner join `scaler-dsml-sql-373605.target_sql.order_items` as oe
on oe.order_id=o.order_id
order by avg_freight desc
limit 5
```

Query Result:

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

Part (6) Top 5 states with highest/lowest average time to delivery

Query Code:

```
with cte_1 as (
select
order_id,
customer_id,
DATETIME_DIFF(DATETIME_2,DATETIME_1,day) as time_to_delivery,
from (select
order_id,
customer_id,
extract(DATETIME  from timestamp(order_purchase_timestamp)) as DATETIME_1,
extract(DATETIME  from timestamp(order_delivered_customer_date)) as DATETIME
_2
from `scaler-dsml-sql-373605.target_sql.orders`
) as t
where DATETIME_1 is not null and DATETIME_2 is not null
)
select
distinct
c.customer_state,
round(avg(ct1.time_to_delivery) over(partition by c.customer_state),2) as av
g_time_to_delivery
from `scaler-dsml-sql-373605.target_sql.customers` as c
inner join cte_1 as ct1
on ct1.customer_id=c.customer_id
inner join `scaler-dsml-sql-373605.target_sql.order_items` as oe
on oe.order_id=ct1.order_id
order by avg_time_to_delivery desc
limit 5
```

Query Result:

Row	customer_state	avg_time_to_del
1	AP	28.22
2	RR	28.17
3	AM	26.34
4	AL	24.45
5	PA	23.7

Part (7) Top 5 states where delivery is really fast/ not so fast compared to estimated date

Query Code:

```
with cte_1 as (
select
order_id,
customer_id,
DATETIME_DIFF(DATETIME_2,DATETIME_1,day) as time_to_delivery,
DATETIME_DIFF(DATETIME_3,DATETIME_2,day) as diff_estimated_delivery
from (select
order_id,
customer_id,
extract(DATETIME  from timestamp(order_purchase_timestamp)) as DATETIME_1,
extract(DATETIME  from timestamp(order_delivered_customer_date)) as DATETIME
_2,
extract(DATETIME  from timestamp(order_estimated_delivery_date)) as DATETIME
_3,
from `scaler-dsml-sql-373605.target_sql.orders`
) as t
where DATETIME_1 is not null and DATETIME_2 is not null and DATETIME_3 is no
t null
)
select*,
from (
select
distinct
c.customer_state,
round(avg(ct1.time_to_delivery) over(partition by c.customer_state),2) as av
g_time_to_delivery,
round(avg(ct1.diff_estimated_delivery) over(partition by c.customer_state),2
) as avg_diff_estimated_delivery
from `scaler-dsml-sql-373605.target_sql.customers` as c
inner join cte_1 as ct1
on ct1.customer_id=c.customer_id
inner join `scaler-dsml-sql-373605.target_sql.order_items` as oe
on oe.order_id=ct1.order_id ) as t
order by avg_diff_estimated_delivery desc
limit 5
```

Query Result:

Row	customer_state	avg_time_to_del	avg_diff_estima
1	AC	20.68	20.98
2	RO	19.66	20.04
3	AM	26.34	19.93
4	AP	28.22	18.4
5	RR	28.17	18.33

Q:6 Payment type analysis:

Part (1) Month over Month count of orders for different payment types

Query Code:

```
select
extract (year from timestamp (o.order_purchase_timestamp )) as year,
extract (month from timestamp (o.order_purchase_timestamp )) as month,
p.payment_type,
count(distinct o.order_id) as cnt_ord
from `scaler-dsml-sql-373605.target_sql.orders` as o
inner join `scaler-dsml-sql-373605.target_sql.payments` p
on o.order_id=p.order_id
group by year,month,p.payment_type
order by p.payment_type,year,month
```

Query Result:

Row	year	month	payment_type	cnt_ord
1	2016	10	UPI	63
2	2017	1	UPI	197
3	2017	2	UPI	398
4	2017	3	UPI	590
5	2017	4	UPI	496
6	2017	5	UPI	772
7	2017	6	UPI	707
8	2017	7	UPI	845
9	2017	8	UPI	938
10	2017	9	UPI	903

Part (2) Count of orders based on the no. of payment installments

Query Code:

```
select
distinct
payment_installments,
count(order_id) over(partition by payment_installments) as cnt_order
from `scaler-dsml-sql-373605.target_sql.payments`
order by payment_installments,cnt_order desc
```

Query Result:

Row	payment_installr	cnt_order
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

Q7) Actionable Insights:

We have been asked to perform data analytics for the given data set, which has information of 100k orders from 2016 to 2018 made at Target in Brazil.

1. In our analysis, we have come across total unique customer States i.e., 27 and 4119 customer cities, The State MG having highest number of cities i.e., 745 and State RR having lowest number of cities i.e., 2.
2. E-commerce growing trend in the number of orders placed by customers has been observed in the first eight months of the year 2017 and 2018 and generally orders peak in August month.
3. More than 75% of the total orders preferred by customers in afternoon to night timestamp zone.
4. It has been observed that the approximate 42% of total orders placed by customers who belong to State SP and State RR having lowest order count i.e.,0.07% of total orders.
5. Top 5 States of Brazil, which have the highest customer potential i.e., SP,RJ,MG,RS and PR which represents 77% of total customers.
6. Highest percentage increase in average cost per order (payment value) from 2017 to 2018 has been observed in month of July i.e., 19.45% and Highest percentage increase in number of orders from 2017 to 2018 is observed in the March month i.e., 808.63%
7. The State SP having lowest mean freight charges value of 15.15 while State RR has the highest mean freight value of 42.98.
8. The minimum average time of delivery of product is approximately 9 days for states like SP and maximum average time of delivery of product is approximately 29 days for states like RR and AP.
9. It has been observed that month over month and year over year customers' most preferred first payment type is Credit Card and second most preferred payment type is UPI.
10. Based on analysis, more than 50% of customers preferred payment done through one instalment and less than 10% of customers wanted to go with payment having more than 6 instalments.

Q8) Recommendations:

As per the findings, the organisation has a major business presence in 55% of total cities in Brazil and even though State MG has the highest number of cities but in terms of customer count it is at third position. So, to gain more customers from other States we can adopt following recommendations:

- As the standard deviation of actual time of delivery of product in Brazil is approximately 10% which is very high and it is directly related to the freight cost and overall cost of product it can easily improve through by adopting proper Network planning like optimal number of locations, size of warehouses/distribution centres, sourcing strategies and best distribution channels.
- Build strong relationship between Value Proposition and operation Strategy like; Everyday low pricing vs Cost efficiency, Customer Experience vs Responsiveness through configure to order, Product innovation vs Efficiency through outsourcing manufacturing & logistics for retainability of customer and expansion of presence in new areas.
- Just to create a unified view of demand, organisations may adopt following steps like; Generate forecast of Consumer demand; Forecast the retailer order; optimise to generate supply plan; aggregate week supply plan to generate financial plan; compare financial plan and adjust trade plan.