1.Data type of all columns in the "customers" table.

```
SELECT
column_name,
data_type
FROM `target`.INFORMATION_SCHEMA.COLUMNS
WHERE table_name='customers'
```

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

INSIGHTS : Data type of column customer_id is STRING, customer_unique_id is STRING, customer_zip_code_prefix is INTEGER, customer_city is STRING (In particular Char(3)), and customer_state is STRING (In particular Char(2))

2.Get the time range between which the orders were placed.

```
SELECT
MIN(order_purchase_timestamp) AS from_date_time,
MAX(order_purchase_timestamp) AS to_date_time
FROM `target.orders`
```

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

INSIGHTS : The orders were placed between 2016-09-04 21:15:19 UTC and 2018-10-17 17:30:18 UTC

3.Count the Cities & States of customers who ordered during the given period.

```sql
SELECT
COUNT(DISTINCT c.customer_city) AS no_of_cities,
COUNT(DISTINCT c.customer_state) AS no_of_states
FROM `target.customers` AS c
JOIN `target.orders` AS o
ON c.customer_id=o.customer_id
```

Row	no_of_cities	no_of_states
1	4119	27

INSIGHTS : customers ordered from 4119 cities and 27 states in brazil.

2.INDEPTH EXPLORATION

1. Is there a growing trend in the no. of orders placed over the past years?

```sql
WITH t AS (SELECT
FORMAT_DATE('%Y-%m',order_purchase_timestamp) AS order_year_month,
COUNT(order_id) AS no_of_orders,
FROM `target.orders`
GROUP BY order_year_month
ORDER BY order_year_month)
SELECT
t.order_year_month,
t.no_of_orders,
LAG(t.no_of_orders,1,0) OVER(ORDER BY t.order_year_month ) AS
previous_month_orders,
t.no_of_orders-LAG(t.no_of_orders,1,0) OVER(ORDER BY t.order_year_month)
AS diff_in_orders
FROM t
ORDER BY t.order_year_month
```

Row	order_year_month	no_of_orders	previous_month_orders	diff_in_orders
1	2016-09	4	0	4
2	2016-10	324	4	320
3	2016-12	1	324	-323
4	2017-01	800	1	799
5	2017-02	1780	800	980
6	2017-03	2682	1780	902
7	2017-04	2404	2682	-278
8	2017-05	3700	2404	1296
9	2017-06	3245	3700	-455
10	2017-07	4026	3245	781

INSIGHTS : There is no continuously increase in number of orders month wise in a whole year. For some months orders are increasing and then decreasing and increasing.

2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

```
WITH t AS (SELECT
COUNT(order_id) AS No_of_orders,
EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month
FROM `target.orders`
GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp),EXTRACT(MONTH FROM
order_purchase_timestamp)
)
SELECT
t.No_of_orders,
t.order_year,
t.order_month
FROM t
ORDER BY t.order_year ,t.No_of_orders DESC
```

Row	No_of_orders	order_year	order_month
1	324	2016	10
2	4	2016	9
3	1	2016	12
4	7544	2017	11
5	5673	2017	12
6	4631	2017	10
7	4331	2017	8
8	4285	2017	9
9	4026	2017	7
10	3700	2017	5

INSIGHTS : IN 2017 most orders(7544) were placed in 11th month. And in 2018 most number of orders(7269) were placed in 1st month.

3.During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

```
WITH t AS (SELECT
order_id,
order_purchase_timestamp,
EXTRACT(HOUR FROM order_purchase_timestamp) AS hour_ordered,
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 'Mornings'
 WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18
 THEN 'Afternoon'
 WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23
 THEN 'Night'
END AS order_time
FROM `target.orders`)

SELECT
t.order_time,
COUNT(t.order_id) AS no_of_orders_placed
FROM t
GROUP BY t.order_time
ORDER BY no_of_orders_placed DESC
```

Row	order_time	no_of_orders_placed
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

INSIGHTS : Brazilians mostly place orders in the Afternoon.

RECOMMENDATIONS : As Brazilians mostly place orders in the afternoon so all the products should be made available in sufficient quantity.

ASSUMPTIONS : The orders between hours 6 to 7, 12 to 13, 18 to 19 and 23 to 24 are ignored, since in the question its only mentioned 0-6, 7-12, 13-18 and 19-23. So only these hours are taken into consideration.

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

1. Get the month on month no. of orders placed in each state.

```
SELECT
COUNT(o.order_id) AS no_of_orders,
FORMAT_DATE('%Y-%m',o.order_purchase_timestamp) AS year_month,
c.customer_state
FROM `target.customers` AS c
LEFT JOIN `target.orders` AS o
ON c.customer_id=o.customer_id
GROUP BY year_month,c.customer_state
ORDER BY c.customer_state,year_month
```

Row	no_of_orders	year_month	customer_state
1	2	2017-01	AC
2	3	2017-02	AC
3	2	2017-03	AC
4	5	2017-04	AC
5	8	2017-05	AC
6	4	2017-06	AC
7	5	2017-07	AC
8	4	2017-08	AC
9	5	2017-09	AC
10	6	2017-10	AC

ASSUMPTIONS : For each state each year each month order counts are calculated.
Eg: For State AC no_of_orders = 2 in 2017-01 and no_of_orders = 6 in 2018-01

2. How are the customers distributed across all the states?

```
SELECT
COUNT(DISTINCT customer_unique_id) AS no_of_customers,
customer_state
FROM `target.customers`
GROUP BY customer_state
ORDER BY no_of_customers DESC
```

Row	no_of_customers	customer_state
1	40302	SP
2	12384	RJ
3	11259	MG
4	5277	RS
5	4882	PR
6	3534	SC
7	3277	BA
8	2075	DF
9	1964	ES
10	1952	GO

INSIGHTS : State SP has more Number of unique customers And State RR has Least number of customers.

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

1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

```
WITH t AS (SELECT
o.order_id,
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
p.payment_value
FROM `target.orders` AS o
JOIN `target.payments`AS p
ON o.order_id=p.order_id
WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017,2018) AND
EXTRACT(MONTH FROM o.order_purchase_timestamp) IN (1,2,3,4,5,6,7,8)
ORDER BY year,month
), f AS (
SELECT
t.year,
SUM(t.payment_value) AS payment_amount
FROM t
```

```
GROUP BY t.year
ORDER BY t.year DESC)
SELECT
f.year,
f.payment_amount,
LEAD(f.payment_amount,1,1) OVER(ORDER BY f.year DESC ) AS payment_of_2017,
(f.payment_amount-LEAD(f.payment_amount,1,1) OVER(ORDER BY f.year DESC
))*100/LEAD(f.payment_amount,1,1) OVER(ORDER BY f.year DESC ) AS
per_increase
FROM f
ORDER BY f.year DESC
LIMIT 1
```

Row	year	payment_amount	payment_of_2017	per_increase
1	2018	8694733.839999...	3669022.119999...	136.9768716466...

INSIGHTS : Cost of orders from 2017 to 2018 has been increased by 136.97 percentage (including months between Jan to Aug, both months included)

2. Calculate the Total & Average value of order price for each state.

```
SELECT
SUM(o_i.price) AS total_order_product_price,
AVG(o_i.price) AS average_order_product_price,
c.customer_state
FROM `target.customers`AS c
LEFT JOIN `target.orders`AS o
ON c.customer_id=o.customer_id
JOIN `target.order_items` AS o_i
ON o.order_id=o_i.order_id
GROUP BY c.customer_state
ORDER BY total_order_product_price DESC,average_order_product_price DESC
```

Row	total_order_product_	average_order_produ	customer_state
1	5202955.050001...	109.6536291597...	SP
2	1824092.669999...	125.1178180945...	RJ
3	1585308.029999...	120.7485741488...	MG
4	750304.0200000...	120.3374530874...	RS
5	683083.7600000...	119.0041393728...	PR
6	520553.3400000...	124.6535775862...	SC
7	511349.9900000...	134.6012082126...	BA
8	302603.9399999...	125.7705486284...	DF
9	294591.9499999...	126.2717316759...	GO
10	275037.3099999...	121.9137012411...	ES

INSIGHTS : State SP has maximum total and average products order price,
And State RR has minimum total and average products order price.

3. Calculate the Total & Average value of order freight for each state.

```
SELECT
SUM(o_i.freight_value) AS total_order_freight_value,
AVG(o_i.freight_value) AS average_order_freight_value,
c.customer_state
FROM `target.customers`AS c
LEFT JOIN `target.orders`AS o
ON c.customer_id=o.customer_id
JOIN `target.order_items` AS o_i
ON o.order_id=o_i.order_id
GROUP BY c.customer_state
ORDER BY total_order_freight_value DESC,average_order_freight_value DESC
```

Row	total_order_freight_value	average_order_freight_value	customer_state
1	718723.06999999518	15.147275390419265	SP
2	305589.31000000093	20.960923931682579	RJ
3	270853.46000000188	20.63016680630664	MG
4	135522.74000000229	21.735804330392845	RS
5	117851.68000000092	20.531651567944319	PR
6	100156.67999999858	26.363958936562188	BA
7	89660.260000000446	21.470368773946355	SC
8	59449.659999999873	32.917862679955654	PE
9	53114.979999999829	22.766815259322811	GO
10	50625.49999999984	21.041354945968457	DF

INSIGHTS : State SP has maximum total and average freight value , And State RR has minimum total and average freight value.

5.Analysis based on sales, freight and delivery time.

1.Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

```
SELECT
order_id,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY) AS
time_to_delivery,
DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY)
AS diff_estimated_delivery
FROM `target.orders`
WHERE order_status !='canceled'
```

Row	order_id	time_to_delivery	diff_estimated_delive
1	635c894d068ac37e6e03dc54e...	30	1
2	3b97562c3aee8bdedcb5c2e45...	32	0
3	68f47f50f04c4cb6774570cfde...	29	1
4	276e9ec344d3bf029ff83a161c...	43	-4
5	54e1a3c2b97fb0809da548a59...	40	-4
6	fd04fa4105ee8045f6a0139ca5...	37	-1
7	302bb8109d097a9fc6e9cefc5...	33	-5
8	66057d37308e787052a32828...	38	-6
9	19135c945c554eebfd7576c73...	36	-2
10	4493e45e7ca1084efcd38ddeb...	34	0

INSIGHTS : For some orders difference in estimated and actual delivery date value is negative and Positive. It means that the estimated date showing is not exact same as order delivery date.

Recommendations : The company should work on application/ML algorithm for correct prediction. so that estimated order estimated delivery date is same as order delivery customer date.
i.e. diff_estimated_delivery=0 for better customer experience.

Assumptions : Orders cancelled are assumed to be not shipped. So orders cancelled are not considered in query.

2. Find out the top 5 states with the highest & lowest average freight value.

```
SELECT * FROM (SELECT
AVG(o_i.freight_value) AS average_freight_value,
c.customer_state
FROM `target.customers`AS c
LEFT JOIN `target.orders`AS o
ON c.customer_id=o.customer_id
JOIN `target.order_items` AS o_i
ON o.order_id=o_i.order_id
GROUP BY c.customer_state
ORDER BY average_freight_value DESC
LIMIT 5)
UNION ALL
SELECT * FROM (SELECT
AVG(o_i.freight_value) AS average_freight_value,
c.customer_state
FROM `target.customers`AS c
LEFT JOIN `target.orders`AS o
```

```
ON c.customer_id=o.customer_id
JOIN `target.order_items` AS o_i
ON o.order_id=o_i.order_id
GROUP BY c.customer_state
ORDER BY average_freight_value
LIMIT 5)
```

Row	average_freight_value	customer_state
1	42.984423076923079	RR
2	42.723803986710926	PB
3	41.069712230215835	RO
4	40.0733695652174	AC
5	39.1479704797048	PI
6	15.147275390419265	SP
7	20.531651567944319	PR
8	20.63016680630664	MG
9	20.960923931682579	RJ
10	21.041354945968457	DF

INSIGHTS : Top 5 states with highest average freight value are RR,PB,RO,AC and PI. Bottom 5 states with lowest average freight value are SP,PR,MG,RJ and DF.

3. Find out the top 5 states with the highest & lowest average delivery time.

```
SELECT * FROM (SELECT
AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY)) AS
average_delivery_time,
c.customer_state
FROM `target.customers`AS c
LEFT JOIN `target.orders`AS o
ON c.customer_id=o.customer_id
JOIN `target.order_items` AS o_i
ON o.order_id=o_i.order_id
WHERE o.order_status!='cancelled'
GROUP BY c.customer_state
ORDER BY average_delivery_time DESC
LIMIT 5)
UNION ALL
SELECT * FROM(SELECT
AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY)) AS
average_delivery_time,
c.customer_state
FROM `target.customers`AS c
```

```
LEFT JOIN `target.orders`AS o
ON c.customer_id=o.customer_id
JOIN `target.order_items` AS o_i
ON o.order_id=o_i.order_id
WHERE o.order_status!='cancelled'
GROUP BY c.customer_state
ORDER BY average_delivery_time
LIMIT 5)
```

Row	average_delivery_tim	customer_state
1	27.82608695652...	RR
2	27.75308641975...	AP
3	25.96319018404...	AM
4	23.99297423887...	AL
5	23.30170777988...	PA
6	8.259608552419...	SP
7	11.48079306071...	PR
8	11.51552218007...	MG
9	12.50148619957...	DF
10	14.52098584675...	SC

INSIGHTS : RR,AP,AM,AL and PA are the 5 states with highest delivery time, And SP,PR,MG,DF and SC are the 5 states with lowest delivery time.

Assumptions : Orders cancelled are assumed to be not shipped. So orders cancelled are not considered in query.

4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

```
SELECT
c.customer_state,
AVG(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY)) AS
avg_time_to_delivery,
AVG(DATE_DIFF(o.order_estimated_delivery_date,o.order_purchase_timestamp,DAY)) AS
avg_estimated_time_delivery
FROM `target.customers` AS c
LEFT JOIN `target.orders` AS o
ON c.customer_id=o.customer_id
WHERE o.order_status ='delivered'
GROUP BY c.customer_state
```

```
HAVING avg_time_to_delivery < avg_estimated_time_delivery
ORDER BY avg_time_to_delivery
LIMIT 5
```

Row	customer_state	avg_time_to_delivery	avg_estimated_time_
1	SP	8.298093544722...	18.77682032542...
2	PR	11.52671135486...	24.25248832012...
3	MG	11.54218777523...	24.18795138277...
4	DF	12.50913461538...	23.94711538461...
5	SC	14.47518330513...	25.40637337845...

INSIGHTS : SP,PR,MG,DF and SC are the 5 states, where the order delivery is really fast as compared to the estimated date of delivery.

RECOMMENDATIONS : It is observed that there is larger difference in days between average time to delivery and average estimated time to delivery. So the company should work on the application/ML algorithm, so that average estimated time to delivery is close to average time to delivery for better customer experience.

6.Analysis based on the payments:

1. Find the month on month no. of orders placed using different payment types.

```
SELECT
COUNT(o.order_id) AS no_of_orders,
FORMAT_DATE('%Y-%m',o.order_purchase_timestamp) AS year_month,
p.payment_type
FROM `target.payments` AS p
JOIN `target.orders` AS o
ON p.order_id=o.order_id
GROUP BY p.payment_type,year_month
ORDER BY year_month,p.payment_type
```

Row	no_of_orders	year_month	payment_type
1	3	2016-09	credit_card
2	63	2016-10	UPI
3	254	2016-10	credit_card
4	2	2016-10	debit_card
5	23	2016-10	voucher
6	1	2016-12	credit_card
7	197	2017-01	UPI
8	583	2017-01	credit_card
9	9	2017-01	debit_card
10	61	2017-01	voucher

INSIGHTS : It is observed that number of orders placed for each month with credit card payment type is more compared to other payment types.

RECOMMENDATIONS : As orders with credit card are more so target should more focus on credit card customers. By providing certain offers to further increasing the number of orders. And also target should identify the reason for less number of orders by other type of payments and should work on improving the orders.

2. Find the no. of orders placed on the basis of the payment installments that have been paid.

```
SELECT
COUNT(DISTINCT order_id) as no_of_orders,
payment_installments
FROM `target.payments`
WHERE payment_installments !=0
GROUP BY payment_installments
ORDER BY payment_installments
```

Row	no_of_orders	payment_installment
1	49060	1
2	12389	2
3	10443	3
4	7088	4
5	5234	5
6	3916	6
7	1623	7
8	4253	8
9	644	9
10	5315	10

INSIGHTS : There are 49060 distinct orders for which customers have done 1st payment installment. And for 12389 distinct orders customers have done 2nd payment installment. Similarly for other number of orders can be obtained from above query.