Q1.)

1.

```
SELECT
  column_name,
  data_type
FROM
  scaler-dsml-427107.Target.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'customers'
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUT

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

Inference:

By using the above query, we have got the datatypes of all the columns of the customers table.

2.

```
SELECT
  MIN(order_purchase_timestamp) AS First_order,
  MAX(order_purchase_timestamp) AS Last_order
FROM
  Target.orders
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECU

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

Inference:

If we take order_purchase_timestamp as the time of order being placed then the first order was placed on 4th September, 2016 and last order was placed on 17th October, 2018. Hence this is the time range between which orders were placed.

3.

```
SELECT
```

```
  COUNT(DISTINCT geolocation_city) AS Total_cities,
  COUNT(DISTINCT geolocation_state) AS Total_states
FROM
  Target.geolocation
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTI

Row	Total_cities	Total_states
1	8011	27

Inference:

Using the count function, we calculated the total number of cities and states from where orders were placed during the given range of time.

Q2.)

1.

```
Select
    count(order_purchase_timestamp) as No_of_orders,
    extract(year from order_purchase_timestamp) as Year
from `Target.orders`
group by Year
order by Year
```

Query results

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

Row	No_of_orders	Year
1	329	2016
2	45101	2017
3	54011	2018

Inference:

By using Extract function, we extracted the year from order_purchase_timestamp to group the data according to year and then counted the orders year wise.
Here we observe that there is a growing trend in number of orders year after year.

2.

```
SELECT
  COUNT(order_purchase_timestamp) as NO_of_orders,
  EXTRACT(month from order_purchase_timestamp) AS month
FROM
  Target.orders
GROUP BY
```

```
  month
ORDER BY
  month
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	No_of_orders	month
1	8069	1
2	8508	2
3	9893	3
4	9343	4
5	10573	5
6	9412	6
7	10318	7
8	10843	8
9	4305	9
10	4959	10
11	7544	11
12	5674	12

Inference:

According to the data, we can clearly see there is a hike in number of orders in the month of May, July and August in which August has the highest number of orders.

3.

```
select
    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,
    count(order_purchase_timestamp) as No_of_orders
from Target.orders
group by order_time
order by No_of_orders desc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

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

Inference:

According the results we can conclude that the Brazilian customers mostly place their orders in the afternoon time.

Q3.)

1.

```
select

    EXTRACT(month from o.order_purchase_timestamp) AS month,
    count(o.order_purchase_timestamp) as No_of_orders,
    c.customer_state
from `Target.orders` as o
join
    `Target.customers` as c
on
  c.customer_id = o.customer_id
group by
      c.customer_state,
      month
order by month
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION

Row	month	No_of_orders	customer_state
1	1	990	RJ
2	1	3351	SP
3	1	151	DF
4	1	427	RS
5	1	99	CE
6	1	113	PE
7	1	443	PR
8	1	264	BA
9	1	971	MG
10	1	51	RN

Inference:

Here is the month on month number of order of each state where the complete data is grouped by customer's state and the month in which order has been placed.

2.

```
SELECT
      customer_state,
      COUNT(customer_id) AS No_of_customers
FROM
      `Target.customers`
GROUP BY
      customer_state
ORDER BY
      customer_state asc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	No_of_customers
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

Inference:

Here is the distribution of customers across all the states which shows the number of customers in every state.

Q4.)

1.

```
select

        round(total_2017,2) as total_cost_2017,
        round(total_2018,2) as total_cost_2018,
        round(((year_2018.total_2018 - year_2017.total_2017) /
year_2017.total_2017) * 100,2) as Percentage_increase
from
```

```
    (select
          sum(p.payment_value) as total_2017
      from
          `Target.payments` as p
      join
          `Target.orders` as o
      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 year_2017,
    (select
          sum(p.payment_value) as total_2018
      from
          `Target.payments` as p
      join
          `Target.orders` as o
      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 year_2018
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTIC

Row	total_cost_2017	total_cost_2018	Percentage_increase
1	3669022.12	8694733.84	136.98

Inference:

The result of this query shows the total order value in 2017 and total order value in 2018 then the percentage increase in the total cost of order in 2018 compared to 2017 considering the months of January to August only which comes to be 136.98%.

2.

```sql
SELECT
  c.customer_state AS State,
  ROUND(SUM(oi.price),2) AS Total_Price,
  ROUND(AVG(oi.price),2) AS Average_Price
FROM
  `Target.orders` AS o
JOIN
  `Target.customers` AS c
ON
  o.customer_id = c.customer_id
JOIN
  `Target.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAIL

Row	State	Total_Price	Average_Price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

Inference:

Here we have obtained the state wise total price and average price of the orders by using the sum and avg functions for which we have joined the 3 tables namely orders, order_items and customers then we grouped the data with respect to the customers state.

3.

```
SELECT
  c.customer_state AS State,
  ROUND(SUM(oi.freight_value),2) AS Total_freight_value,
  ROUND(AVG(oi.freight_value),2) AS Average_freight_value
FROM
  `Target.orders` AS o
JOIN
  `Target.customers` AS c
ON
  o.customer_id = c.customer_id
JOIN
  `Target.order_items` AS oi
ON
  o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTIC

Row	State	Total_freight_value	Average_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

Inference:

Here we have obtained the total freight value and average freight value of orders for each state i.e price rate at which a product is delivered from one place to another. Here we joined three tables and grouped the data with respect to customers state and used the sum and avg functions for calculations.

Q5.)

1.

```
SELECT
  order_purchase_timestamp as order_date,
  order_delivered_customer_date as Delivery_date,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
day) AS time_to_deliver,
  DATE_DIFF(order_delivered_customer_date,
order_estimated_delivery_date, day) AS diff_estimated_delivery
FROM
  Target.orders
ORDER BY
  time_to_deliver DESC,
  diff_estimated_delivery desc
```

Query results

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

Row	order_date	Delivery_date	time_to_deliver	diff_estimated_delivery
1	2017-02-21 23:31:27 UTC	2017-09-19 14:36:39 UTC	209	181
2	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 UTC	208	188
3	2017-03-07 23:59:51 UTC	2017-09-19 15:12:50 UTC	195	165
4	2017-03-08 22:47:40 UTC	2017-09-19 14:00:04 UTC	194	166
5	2017-03-09 13:26:57 UTC	2017-09-19 14:38:21 UTC	194	161
6	2017-03-08 18:09:02 UTC	2017-09-19 14:33:17 UTC	194	155
7	2018-01-03 09:44:01 UTC	2018-07-13 20:51:31 UTC	191	175
8	2017-03-13 20:17:10 UTC	2017-09-19 17:00:07 UTC	189	167
9	2017-03-15 11:24:27 UTC	2017-09-19 14:38:18 UTC	188	159
10	2017-03-15 23:23:17 UTC	2017-09-19 17:14:25 UTC	187	162

Inference:

Here we have calculated the number of days taken to an order to get delivered to a customer by taking the difference of order date and delivery date. Also we have calculated the difference of estimated delivery date to the actual delivery date where it shows that in many cases the order got delivered after the estimated delivery date.

2.

```
SELECT
  state,
  ROUND(Avg_freight_value,2) AS Avg_freight_value
FROM (
  SELECT
    c.customer_state AS state,
```

```
    AVG(oi.freight_value) AS Avg_freight_value,
    DENSE_RANK() OVER (ORDER BY AVG(oi.freight_value) DESC) AS rank_high,
    DENSE_RANK() OVER (ORDER BY AVG(oi.freight_value) ASC) AS rank_low
  FROM
    `Target.orders` AS o
  JOIN
    `Target.customers` AS c
  ON
    o.customer_id = c.customer_id
  JOIN
    `Target.order_items` AS oi
  ON
    o.order_id = oi.order_id
  GROUP BY
    c.customer_state )
WHERE
  rank_high <= 5
  OR rank_low <=5
ORDER BY
  Avg_freight_value desc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	state	Avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15
6	DF	21.04
7	RJ	20.96
8	MG	20.63
9	PR	20.53
10	SP	15.15

Inference:

Here first we have calculated the average freight value for each state by grouping the data state wise and ranked the states according to average freight value where first 5 rows show the top 5 states with highest average freight value and last 5 rows shows the lowest average freight value.

3.

```
SELECT
  state,
  ROUND(avg_del_time,2) AS Avg_del_time
FROM (
  SELECT
    c.customer_state AS state,
    AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)) AS Avg_del_time,
    DENSE_RANK() OVER (ORDER BY
AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)) DESC) AS rank_high,
    DENSE_RANK() OVER (ORDER BY
AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)) ASC) AS rank_low
  FROM
    `Target.orders` AS o
  JOIN
    `Target.customers` AS c
  ON
    o.customer_id = c.customer_id
  GROUP BY
    c.customer_state )
WHERE
  rank_high <=5
  OR rank_low <=5
ORDER BY
  avg_del_time desc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	state	Avg_del_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32
6	SC	14.48
7	DF	12.51
8	MG	11.54
9	PR	11.53
10	SP	8.3

Inference:

Here first we have calculated the average delivery time for each state by grouping the data state wise and ranked the states according to average delivery time where first 5 rows show the top 5 states with highest delivery time and last 5 rows shows the lowest average delivery time.

4.

```
SELECT
  state,
  (avg_del_time - avg_est_time) AS fast_del
FROM (
  SELECT
    c.customer_state AS state,
    AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, day)) AS avg_del_time,
    AVG(DATE_DIFF(order_delivered_customer_date,
order_estimated_delivery_date, day)) AS avg_est_time
  FROM
    `Target.orders` AS o
  JOIN
    `Target.customers` AS c
  ON
    o.customer_id = c.customer_id
  GROUP BY
    c.customer_state )
ORDER BY
  fast_del asc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	state	fast_del
1	SP	18.43338683788...
2	DF	23.62788461538...
3	MG	23.84077498899...
4	PR	23.89092017062...
5	ES	24.95037593984...

Inference:

The above results show the top 5 states where delivery was really fast. Here we have taken the difference of the average of the actual delivery time taken to deliver a product and the average of the estimated delivery time of each state.

Q6.)

1.

```
select
    extract(month from order_purchase_timestamp) as Month,
    p.payment_type as Payment_type,
    count(o.order_id) as No_of_orders
from
    `Target.orders` as o
join
    `Target.payments` as p
on
    o.order_id = p.order_id
group by
    p.payment_type,
    month
order by month
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DET

Row	Month	Payment_type	No_of_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

Inference:

Here we can see that the complete data has been grouped with respect to the payment type of the customers for the orders and is in order of month on month
By analysing the results, we can conclude that in every month a very high number of the customers use credit card for the payment of their orders.

2.

```
SELECT
  payment_installments AS No_of_installments,
  COUNT(DISTINCT order_id) AS No_of_orders
FROM
  `Target.payments`
GROUP BY
  payment_installments
ORDER BY
  payment_installments
```

Query results

JOB INFORMATION | RESULTS | CHART

Row	No_of_installments	No_of_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644

Inference:

Here we have calculated the number of orders for which the customers has done the payments in the installments. Here the data have been grouped with respect to the different number of installments with the distinct number of orders.