By using my sql & big query

#1.a. Data type of all the column in the “customers” table.

```
select
column_name,
data_type
from bustling-art-406010.Ecommerce.INFORMATION_SCHEMA.COLUMNS
where table_name = "geolocation"
```

Row	column_name	data_type
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING

insight : Most of the columns in geolacatiom table is float and string type

#1.b. Get the time range between which the orders were placed.

```
SELECT
min(order_purchase_timestamp) as first_order,
max(order_purchase_timestamp) as last_order
 FROM ecommerce.orders;
```

	first_order	last_order
▶	2016-10-04 14:13:22	2018-09-20 13:54:16

Insight : In this query by using min and max we get the time range.

#1.c. Count the Cities & States of customer who ordered during the given period.

```
select
count(distinct c.customer_state) as state_,
count(distinct c.customer_city) as city_
From Ecommerce.orders as  o
inner join Ecommerce.customers as c
```

```
on c.customer_id = o.customer_id
```

Row	state_	city_
1	27	4119

Insight : In the query count and inner join used to get the cities and states of customer who ordered given period

#2.a. Is there a growing trend in the no. of orders placed over the past years.

```
SELECT
extract(year from order_purchase_timestamp) as year_,
count(*) as orders_count
as FROM ecommerce.orders
Group by year_
order by year_
```

year_	orders_count
2016	8
2017	1679
2018	2029

Insight : in this query use extract keyword for geting year and count aggeregation function is used.

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

```
SELECT
extract(year from order_purchase_timestamp) as year_,
extract(month from order_purchase_timestamp) as month_,
count(*) as orders_count
From ecommerce.orders
```

```
group by year_, month_

order by year_, month_;
```

year_	month_	orders_count
2016	10	8
2017	1	41
2017	2	64
2017	3	106
2017	4	90
2017	5	140
2017	6	115
2017	7	140
2017	8	165

Insight: : in this query use extract keyword for geting year from order_purchase_timestamp and count aggeregation function is used for count order.

#2.c. During what time of the day. Do the Brazilian customers mostly placed their orders?(Dawn, Morning, afternoon, or Night)

#0 - 6 hrs : Dawn

#7 - 12 hrs: Morning

#13 - 18 hrs: afternoon

#19 – 23 hrs: Night

```
select

 case

        when extract(hour from order_purchase_timestamp) Between 8 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'

  when extract(hour from order_purchase_timestamp) Between 19 and 23 Then 'Night'

  End as time_of_day,

 count(*) as No_orders
```

```
FROM ecommerce.orders
Group by time_of_day
order by No_orders desc;
```

time_of_day	No_orders
Afternoon	1437
Morning	1045
Night	1043
NULL	191

Insight: in this query used case function for (dawn, morning,afternoon,evening) and count function.

#3.a. Get the month on month no. of orders placed in each state.

```
SELECT
customer_state,
extract(month from order_purchase_timestamp) as month_,
count(*) as num_order
FROM ecommerce.orders as o join ecommerce.customers as c
 ON o.customer_id = c.customer_id
 Group by customer_state, month_
 order by num_order desc;
```

customer_state	month_	num_order
SP	5	30
SP	8	28
SP	11	27
SP	1	25
SP	3	23
SP	4	23
SP	7	20
SP	6	20
SP	2	17

Insight: in this query extract function and count function id used.

#3.b. How are the customer distributed across all the states ?

```
SELECT
customer_state,
extract(year from order_purchase_timestamp) as year_,
extract(month from order_purchase_timestamp) as month_,
count(*) as num_order
FROM ecommerce.orders as o join ecommerce.customers as c
 ON o.customer_id = c.customer_id
 Group by customer_state, month_, year_
 order by num_order desc;
```

customer_state	year_	month_	num_order
SP	2017	11	27
SP	2018	1	22
SP	2018	5	20
SP	2018	3	18
SP	2018	4	17
SP	2017	12	16
SP	2018	6	15
SP	2018	7	15
SP	2018	8	15

#4.a. Get the % increase in the cost of orders from year 2017 to 2018 (include months between jan to Aug only)

```
(
SELECT
* with base_1 as


FROM ecommerce.orders as o inner join ecommerce.payments as p using(order_id)
 where extract(year from order_purchase_timestamp) Between 2017 and 2018
 AND extract(month from order_purchase_timestamp) between 1 and 8
```

```
),
base_2 as
(
select
Extract(year from order_purchase_timestamp) as year_,
Round(sum(payment_value), 2) as cost
from base_1
Group by year_
order by 1
),
base_3 as
(
select
*, lead(cost) over(order by year_) as next_year
from base_2
)
select *, (next_year - cost) / cost * 100 as percent_increase
from base_3;
```

year_	cost	next_year	percent_increase
2017	3519.39	11039.94	213.68901997221116
2018	11039.94	NULL	NULL

Insight : In this query cte is used to get answer and lead function is used to get the cost of next year

#4.b. calculate the total & average value of order price for each state.

```
SELECT
customer_state,
Round(sum(price), 2) as total_price,
Round(avg(price), 2) as avg_price
```

```
FROM ecommerce.orders as o inner join ecommerce.order_items as i
 ON o.order_id = i.order_id
 inner join ecommerce.customers as c
  ON c.customer_id = o.customer_id
  Group by customer_state
  order by avg_price
```

customer_state	total_price	avg_price
RS	23.99	23.99
SC	89.9	29.97
SP	1286.42	116.95
RJ	351.8	117.27
RN	129.99	129.99
PE	379.8	189.9
MA	243.37	243.37
BA	249	249
MS	320	320

Insight: in this query we use round function and sum for total_price & avg price and also used join function.

#4.C. Calculate the total & average value of order freight for each state.

```
SELECT
customer_state,
Round(sum(freight_value), 2) as total_value,
Round(avg(freight_value), 2) as avg_value
FROM ecommerce.orders as o inner join ecommerce.order_items as i
 ON o.order_id = i.order_id
 inner join ecommerce.customers as c
  ON c.customer_id = o.customer_id
  Group by customer_state
  order by avg_value;
```

customer_state	total_value	avg_value
SP	129.96	11.81
SC	44.14	14.71
BA	15.22	15.22
RS	15.65	15.65
DF	21.23	21.23
RJ	69.99	23.33
PE	52.38	26.19
RN	28	28
MA	53.83	53.83

#5.a. Find the no.of days taken tu 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.

you can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula

Time_to_deliver = order_delivered_customer_date – order_purchase_timestamp

diff_estimated_delivery = order_delivered_customer_date - - order_estimated_delivered_date

```
SELECT

order_id,

timestampdiff(DAY, order_delivered_customer_date, order_purchase_timestamp) as time_to_deliver,

timestampdiff(DAY, order_delivered_customer_date, order_estimated_delivery_date) as diff_estimated_delivery

FROM ecommerce.orders

where order_status = 'delivered';
```

order_id	time_to_deliver	diff_estimated_delivery
e481f51cbdc54678b7cc49136f2d6af7	-8	7
53cdb2fc8bc7dce0b6741e2150273451	-13	5
47770eb9100c2d0c44946d9cf07ec65d	-9	17
949d5b44dbf5de918fe9c16f97b45f8a	-13	12
ad21c59c0840e6cb83a9ceb5573f8159	-2	9
a4591c265e18cb1dcee52889e2d8acc3	-16	5
6514b8ad8028c9f2cc2374ded245783f	-9	11
76c6e866289321a7c93b82b54852dc33	-9	31
e69bfb5eb88e0ed6a785585b27e16dbf	-18	6

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

Highest average freight value

```
SELECT
customer_state,
Round(avg(freight_value), 2) as avg_freigt_price
FROM ecommerce.orders as o inner join ecommerce.order_items as i
 ON o.order_id = i.order_id
inner join ecommerce.customers as c
 ON o.customer_id = c.customer_id
 Group by customer_state
 order by 2 desc
 limit 5 ;
```

customer_state	avg_freigt_price
MS	58.91
MA	53.83
RN	28
PE	26.19
RJ	23.33

Lowest average freight value

```
SELECT
```

```
customer_state,
Round(avg(freight_value), 2) as avg_freigt_price
FROM ecommerce.orders as o inner join ecommerce.order_items as i
 ON o.order_id = i.order_id
inner join ecommerce.customers as c
 ON o.customer_id = c.customer_id
 Group by customer_state
 order by 2 asc
 limit 5 ;
```

customer_state	avg_freigt_price
SP	11.81
SC	14.71
BA	15.22
RS	15.65
DF	21.23

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

Highest average freight value

```
SELECT
customer_state,
Round(avg(order_delivered_customer_date), 2) as avg_delivery_date
FROM ecommerce.orders as o
inner join ecommerce.customers as c
 ON o.customer_id = c.customer_id
 Group by customer_state
 order by avg_delivery_date desc
 limit 5 ;
```

customer_state	avg_delivery_date
PI	2018
PB	2018
MA	2018
DF	2017.7
RN	2017.67

Lowest average freight value

```
SELECT
customer_state,
Round(avg(order_delivered_customer_date), 2) as avg_delivery_date
FROM ecommerce.orders as o
 inner join ecommerce.customers as c
 ON o.customer_id = c.customer_id
 Group by customer_state
 order by avg_delivery_date asc
 limit 5 ;
```

customer_state	avg_delivery_date
SC	1780.12
GO	1815.9
RJ	1906.99
BA	1948
RS	1952.32

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

#you can use the difference between the average of actual & estimated delivery date to figure out how fast the delivery was for each state.

```
SELECT
customer_state as state,
```

```
ROUND(sum(timestampdiff(DAY,order_delivered_customer_date,
order_purchase_timestamp)) / count(order_id), 2) as average_time_for_del,

ROUND(sum(timestampdiff(DAY,order_estimated_delivery_date,
order_purchase_timestamp)) / count(order_id), 2) as average_est_dil_time

FROM ecommerce.orders as o inner join ecommerce.customers as c

 ON o.customer_id = c.customer_id

WHERE order_status = 'delivered'

GROUP BY state

order by(average_time_for_del - average_est_dil_time)
```

state	average_time_for_del	average_est_dil_time
SE	-27.67	-32.00
MA	-18.00	-25.00
AL	-24.50	-33.00
DF	-13.50	-23.00
SP	-8.16	-18.69
ES	-13.17	-23.83
PR	-11.78	-22.59
CE	-16.86	-27.71
SC	-14.07	-25.80

#6.a.Find the month on month no. of orders placed using different payment types.

```
SELECT payment_type,

extract(year from order_purchase_timestamp) as year_,

extract(month from order_purchase_timestamp) as month_,

count(o.order_id) as num_order

FROM ecommerce.payments as p

inner join ecommerce.orders as o

 ON p.order_id = o.order_id

 Group by payment_type, year_, month_

 order by  year_, month_, num_order;
```

payment_type	year_	month_	num_order
credit_card	2017	1	3
credit_card	2017	2	2
credit_card	2017	3	1
UPI	2017	3	2
credit_card	2017	4	4
credit_card	2017	5	4
UPI	2017	6	1
credit_card	2017	6	2
voucher	2017	7	1

Result 3

#6.b.Find the no. of order placed on the basis of the payment installments the have been paid.

```
SELECT
count(distinct order_id) as num_orders,
payment_installments
FROM ecommerce.payments
where payment_installments >= 1
group by payment_installments;
```

num_orders	payment_installments
2223	1
533	2
405	3
303	4
215	5
141	6
68	7
190	8

Insight: in this query used payment table by using count function count the order id.