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

A. Data type of all columns in the “customers” table.

Hint: We want you to display the data type of each column present in the “customers” table.

Field name	Type	Mode	Description	Key	Collation	Default Value	Policy Tags	Data Policies
customer_id	STRING	NULLABLE	-	-	-	-	-	-
customer_unique_id	STRING	NULLABLE	-	-	-	-	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-	-	-
customer_city	STRING	NULLABLE	-	-	-	-	-	-
customer_state	STRING	NULLABLE	-	-	-	-	-	-

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

Hint: We want you to get the date & time when the first and last orders in our dataset were placed.

Query:

```
select max(order_purchase_timestamp) as last_order, min(order_purchase_timestamp) as first_order
from `brazil_db.orders`;
```

Results:

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

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

Hint: We want you to count the number of unique cities & states where orders were placed by the customers during the given time period.

Query:

```
select count(distinct c.customer_city) as unique_city,count(distinct c.customer_state) as unique_state
from `brazil_db.customers` c
join
`brazil_db.orders` o on
c.customer_id = o.customer_id;
```

Output:

Row	unique_city	unique_state
1	4119	27

II. In-depth Exploration:

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

Hint: We want you to find out if no. of orders placed has increased gradually in each month, over the past years.

Query:

```
select
extract(MONTH from order_purchase_timestamp) as order_month,
extract(YEAR from order_purchase_timestamp)as order_year,
count(order_id) as tot_orders
from `brazil_db.orders`
group by order_month, order_year
order by order_month asc, order_year asc;
```

Output:

Row	order_month	order_year	tot_orders
1	1	2017	800
2	1	2018	7269
3	2	2017	1780
4	2	2018	6728
5	3	2017	2682
6	3	2018	7211
7	4	2017	2404
8	4	2018	6939
9	5	2017	3700

Row	order_month	order_year	tot_orders
10	5	2018	6873
11	6	2017	3245
12	6	2018	6167
13	7	2017	4026
14	7	2018	6292
15	8	2017	4331
16	8	2018	6512
17	9	2016	4
18	9	2017	4285

Row	order_month	order_year	tot_orders
19	9	2018	16
20	10	2016	324
21	10	2017	4631
22	10	2018	4
23	11	2017	7544
24	12	2016	1
25	12	2017	5673

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

Hint: We want you to find out if the no. of orders placed are at peak during certain months.

Query:

```
select
extract(MONTH from order_purchase_timestamp) as month_,
count(order_id) as tot_orders
from `brazil_db.orders`
group by month_
order by month_ asc;
```

Output:

Row	month_	tot_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412

Row	month_	tot_orders
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

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

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon

● 19-23 hrs : Night

Hint: We want you to categorize the hours of a day into the given time brackets/ intervals and find out during which intervals the Brazilian customers usually order the most.

Query:

```
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 '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 most_orders_placed_in,
  count(order_id) as tot_orders
from `brazil_db.orders`
group by most_orders_placed_in
order by tot_orders desc
limit 1;
```

Output:

Row	most_orders_placed_in	tot_orders
1	Afternoon	38135

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

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

Hint: We want you to get the no. of orders placed in each state, in each month by our customers.

Query:

```
Select c.customer_state as state,
format_date('%Y-%m' , date(o.order_purchase_timestamp)) as year_month,
count(o.order_id) as tot_orders
from `brazil_db.orders` o join `brazil_db.customers` c on
o.customer_id = c.customer_id
```

group by state, year_month

order by year_month, state;

Output:

Row	state	year_month	tot_orders
1	RR	2016-09	1
2	RS	2016-09	1
3	SP	2016-09	2
4	AL	2016-10	2
5	BA	2016-10	4
6	CE	2016-10	8
7	DF	2016-10	6
8	ES	2016-10	4
9	GO	2016-10	9
10	MA	2016-10	4

Row	state	year_month	tot_orders
11	MG	2016-10	40
12	MT	2016-10	3
13	PA	2016-10	4
14	PB	2016-10	1
15	PE	2016-10	7
16	PI	2016-10	1
17	PR	2016-10	19
18	RJ	2016-10	56
19	RN	2016-10	4
20	RR	2016-10	1

Row	state	year_month	tot_orders
21	RS	2016-10	24
22	SC	2016-10	11
23	SE	2016-10	3
24	SP	2016-10	113
25	PR	2016-12	1
26	AC	2017-01	2
27	AL	2017-01	2
28	BA	2017-01	25
29	CE	2017-01	9
30	DF	2017-01	13

Row	state	year_month	tot_orders
31	ES	2017-01	12
32	GO	2017-01	18
33	MA	2017-01	9
34	MG	2017-01	108
35	MS	2017-01	1
36	MT	2017-01	11
37	PA	2017-01	12
38	PB	2017-01	2
39	PE	2017-01	9
40	PI	2017-01	7

Row	state	year_month	tot_orders
41	PR	2017-01	65
42	RJ	2017-01	97
43	RN	2017-01	5
44	RO	2017-01	3
45	RS	2017-01	54
46	SC	2017-01	31
47	SE	2017-01	4
48	SP	2017-01	299
49	TO	2017-01	2
50	AC	2017-02	3

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

Hint: We want you to get the no. of unique customers present in each state.

Query:

```
select
customer_state,
count(distinct customer_id) as Unique_customers
from `brazil_db.customers`
group by customer_state;
```

Output:

Row	customer_state	Unique_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

Row	customer_state	Unique_customers
10	MA	747
11	MG	11635
12	MS	715
13	MT	907
14	PA	975
15	PB	536
16	PE	1652
17	PI	495
18	PR	5045

Row	customer_state	Unique_customers
19	RJ	12852
20	RN	485
21	RO	253
22	RR	46
23	RS	5466
24	SC	3637
25	SE	350
26	SP	41746
27	TO	280

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

A. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
   Hint: You can use the payment_value column in the payments table to get the cost of orders.

Query:

```
with yearly_cost as (
  Select
  extract(YEAR from o.order_purchase_timestamp) as order_year,
  sum(p.payment_value) as total_cost
  from `brazil_db.orders` o
  join `brazil_db.payments` p
  on o.order_id = p.order_id
  where extract(MONTH from o.order_purchase_timestamp) between 1 AND 8
    AND extract(YEAR from o.order_purchase_timestamp) IN (2017, 2018)
  group by order_year
)
Select
  ((max(CASE when order_year = 2018 then total_cost END) -
    max(CASE when order_year = 2017 then total_cost END)) /
    max(CASE when order_year = 2017 then total_cost END)) * 100 as percentage_increase
from yearly_cost;
#using max to prevent rows merging... it'll separate 2018 and 2017 rows
```

Output:

Row	percentage_incre...
1	136.9768716466...

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

   Hint: We want you to fetch the total price and the average price of orders for each state.

Query:

```
select c.customer_state, sum(p.payment_value) as tot_cost,
avg(p.payment_value) as avg_state_amount
from `brazil_db.payments` p
join
`brazil_db.orders` o on
p.order_id=o.order_id
join
`brazil_db.customers` c on
o.customer_id=c.customer_id
group by c.customer_state;
```

Output:

Row	customer_state	tot_cost	avg_state_amount
1	SP	5998226.959999...	137.5046297739...
2	BA	616645.8200000...	170.8160166204...
3	RJ	2144379.689999...	158.5258882235...
4	MT	187029.2900000...	195.2289039665...
5	GO	350092.3100000...	165.7634043560...
6	ES	325967.5499999...	154.7069530137...
7	MG	1872257.260000...	154.7064336473...
8	PR	811156.380000001	154.1536259977...
9	RS	890898.5400000...	157.1804057868...

Row	customer_state	tot_cost	avg_state_amount
19	MS	137534.8399999...	186.8679891304...
20	TO	61485.33000000...	204.2701993355...
21	PA	218295.8500000...	215.9207220573...
22	RO	60866.19999999...	233.2038314176...
23	PI	108523.9699999...	207.1068129770...
24	RN	102718.1299999...	196.7780268199...
25	AC	19680.62	234.2930952380...
26	AP	16262.80000000...	232.3257142857...
27	RR	10064.61999999...	218.7960869565...

Row	customer_state	tot_cost	avg_state_amount
10	SC	623086.4300000...	165.9793367075...
11	SE	75246.25000000...	208.4383656509...
12	CE	279464.0299999...	199.9027396280...
13	PE	324850.4399999...	187.9921527777...
14	AM	27966.93	181.6034415584...
15	DF	355141.0800000...	161.1347912885...
16	PB	141545.7199999...	248.3258245614...
17	AL	96962.05999999...	227.0774238875...
18	MA	152523.0200000...	198.8566101694...

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

Hint: We want you to fetch the total freight value and the average freight value of orders for each state.

Query:

```
select c.customer_state, sum(oi.freight_value) as tot_freight_value,
avg(oi.freight_value) as avg_state_freight_value
from `brazil_db.order_items` oi
join
`brazil_db.orders` o on
oi.order_id=o.order_id
join
`brazil_db.customers` c on
o.customer_id=c.customer_id
group by c.customer_state;
```

Output:

Row	customer_state	tot_freight_value	avg_state_freight...
1	SP	718723.0700000...	15.14727539041...
2	RJ	305589.3100000...	20.96092393168...
3	PR	117851.6799999...	20.53165156794...
4	RS	135522.7399999...	21.73580433039...
5	MG	270853.4600000...	20.63016680630...
6	GO	53114.98000000...	22.76681525932...
7	SC	89660.25999999...	21.47036877394...
8	PB	25719.73	42.72380398671...
9	DF	50625.50000000...	21.04135494596...

Row	customer_state	tot_freight_value	avg_state_freight...
10	PA	38699.30000000...	35.83268518518...
11	BA	100156.6800000...	26.36395893656...
12	ES	49764.59999999...	22.05877659574...
13	PE	59449.66000000...	32.91786267995...
14	MS	19144.03000000...	23.37488400488...
15	MT	29715.42999999...	28.16628436018...
16	MA	31523.76999999...	38.25700242718...
17	CE	48351.59000000...	32.71420162381...
18	TO	11732.68	37.24660317460...

Row	customer_state	tot_freight_value	avg_state_freight...
19	AM	5478.890000000...	33.20539393939...
20	AL	15914.59000000...	35.84367117117...
21	RO	11417.38000000...	41.06971223021...
22	RN	18860.10000000...	35.65236294896...
23	PI	21218.20000000...	39.14797047970...
24	AC	3686.749999999...	40.07336956521...
25	SE	14111.46999999...	36.65316883116...
26	RR	2235.189999999...	42.98442307692...
27	AP	2788.499999999...	34.00609756097...

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

A. 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. Do this in a single query.

Hint: 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_estimated_delivery_date - order_delivered_customer_date

Query:

```
select order_id,

date_diff(order_delivered_customer_date,order_purchase_timestamp,DAY)as
no_of_days_to_deliver,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,DAY) as
diff_estimated_delivery

from `brazil_db.orders`

where order_delivered_customer_date is not null
```

Output:

Row	order_id	no_of_days_to_deliver	diff_estimated_delivery
1	65d1e226dfaeb8cdc42f665422522d14	35	16
2	2c45c33d2f9cb8ff8b1c86cc28c11c30	30	28
3	1950d777989f6a877539f53795b4c3c3	30	-12
4	bfbd0f9bdef84302105ad712db648a6c	54	-36
5	98974b076b01553d49ee6467905675a7	43	6
6	c4b41c36dd589e901f6879f25a74ec1d	36	14
7	d2292ff2201e74c5db154d1b7ae68cbb	29	20
8	95e01270fcbae9863423400103359279	30	19
9	ed8c7b1b3eb256c70ce0c74231e1da88	44	5
10	5cc475c7c03290048eb2e742cd64cb5e	68	-18

Row	order_id	no_of_days_to_deliver	diff_estimated_delivery
11	6b3ee7697a02619a0ace2b3f0aa46bde	47	2
12	3b2ca3293a7ce539ea2379d704fa37ce	43	7
13	b2f92b2f7047cd8b35580d629d7b3bfb	43	7
14	e2eaf909eb6ba881117aa407992a5ffb	40	10
15	90aea7c4e52538a18cb9bbfd16f09185	43	10
16	dd11631d8b02780b78bd97ec44a1ca8c	30	23
17	f781dacd75aa73166785a35929546894	33	19
18	b0cfee6ce5503781be6004e0e31e4309	55	-1
19	893dc49276696129a9bf6f2a9fe23e57	29	24
20	bb9fd560a08ca7403789dedaca85da14	31	22

Row	order_id	no_of_days_to_deliver	diff_estimated_delivery
21	63638a6806d67773f3adba8534553fff	51	2
22	5bda3e4d7080c484cd646a53165a7892	40	12
23	5da2004d85f2828349d887c989da2adc	51	2
24	9315757eaf0e782619bed131508dbddb	30	21
25	2d9e3c3c7f0f3ba8a8fa3db2f1211ceb	32	21
26	a41c8759fbe7aab36ea07e038b2d4465	30	25
27	8d47519f5c9c42540b37bd3db0492878	33	22
28	262118ce178bb3e4590a3adcf6d62e6b	42	12
29	7cb5aaf8b7b7b439174c2e4efdf4c7a3	32	22
30	05bfcf9e287cad216cadb8a850f8db5c	32	22

Row	order_id	no_of_days_to_deliver	diff_estimated_delivery
31	6e0d2539e48e84296bae6290de41058c	43	12
32	ae8a60e4b03c5a4ba9ca0672c164b181	30	27
33	11837ac14fcdfb163c0b78d91be89498	40	15
34	c8258090bca7cc6a62c1e15d7fd995d4	30	23
35	6460e331ee8f69c01f1423cca48d9991	45	7
36	f9b03f7a6e5788ffb6cb6b4cd0d01892	37	15
37	7e2dddf174cc4d2768bf39c43ab2bd7a	31	21
38	8b388d845a3e8700444b971fa2bef902	34	27
39	5f4c8250b9e6ee47971e0b19da4aa0f2	41	17
40	b7026ba7432929d99880463305b8ce98	34	25

Row	order_id	no_of_days_to_deliver	diff_estimated_delivery
41	cac247cece621e1bc64748d44331d22d	30	32
42	80606b26965c5ed21e85a085e0667b63	44	20
43	c3d9e402b6a0fbe2a5f7fc5b41117c38	35	29
44	4b038a7f8dfc36755eca741381e379e3	45	13
45	d01f0aba52a4d7b2cca8d7623538a434	31	31
46	10d1e90a86d2be981a52b6a927c97dc5	32	34
47	39d0be719247e3b3a38846ba810197ea	34	34
48	09dd83f72b2516179cd46b81121fad23	48	13
49	676157c2d1075c88ea0009f8d7d2d389	33	33
50	4fb7f38042d214ada4d81cc23d228cc8	37	27

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

Hint: We want you to find the top 5 & the bottom 5 states arranged in increasing order of the average freight value.

Query:

```
with state_avg_frieght_values as
(select avg(oi.freight_value) as avg_frieght,
c.customer_state
from `brazil_db.order_items` oi
join
`brazil_db.orders` o on
oi.order_id=o.order_id
join
`brazil_db.customers` c on
o.customer_id=c.customer_id
group by c.customer_state)
select * from
( select 'lowest_frieght' as category, customer_state, avg_frieght
  from state_avg_frieght_values
  order by avg_frieght asc limit 5)
UNION ALL
( select 'highest_frieght' as category,customer_state, avg_frieght
  from state_avg_frieght_values
  order by avg_frieght desc limit 5)
```

Output:

Row	category	customer_state	avg_frieght
1	lowest_frieght	SP	15.147275390419187
2	lowest_frieght	PR	20.531651567944177
3	lowest_frieght	MG	20.6301668063067
4	lowest_frieght	RJ	20.960923931682533
5	lowest_frieght	DF	21.041354945968447
6	highest_frieght	RR	42.984423076923079
7	highest_frieght	PB	42.723803986710969
8	highest_frieght	RO	41.069712230215814
9	highest_frieght	AC	40.073369565217391
10	highest_frieght	PI	39.147970479704853

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

Hint: We want you to find the top 5 & the bottom 5 states arranged in increasing order of the average delivery time.

Query:

```
with delivery_time as
(select
date_diff(order_delivered_customer_date,order_purchase_timestamp,DAY)as
no_of_days_to_deliver,
c.customer_state
from `brazil_db.orders` o
join
`brazil_db.customers` c on
o.customer_id=c.customer_id
where order_delivered_customer_date is not null
),
state_avg_delivery_day as(
  select customer_state,
  avg(no_of_days_to_deliver) as deliver_days
  from delivery_time
  group by customer_state
)
```

```
select * from
( select 'lowest_delivery' as category, customer_state, deliver_days
  from state_avg_delivery_day
  order by deliver_days asc limit 5)
UNION ALL
( select 'highest_delivery' as category,customer_state, deliver_days
  from state_avg_delivery_day
  order by deliver_days desc limit 5)
```

Output:

Row	category	customer_state	deliver_days
1	lowest_delivery	SP	8.2980614890727011
2	lowest_delivery	PR	11.526711354864918
3	lowest_delivery	MG	11.543813298106569
4	lowest_delivery	DF	12.509134615384614
5	lowest_delivery	SC	14.479560191711343
6	highest_delivery	RR	28.975609756097565
7	highest_delivery	AP	26.731343283582088
8	highest_delivery	AM	25.986206896551714
9	highest_delivery	AL	24.040302267002534
10	highest_delivery	PA	23.31606765327696

E. 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 averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Hint: Include only the orders that are already delivered.

Query:

```
with delivery_diff as(
  select
  date_diff(order_estimated_delivery_date,order_delivered_customer_date,DAY) as
early_delivery_days,
  c.customer_state
  from `brazil_db.orders` o
  join
  `brazil_db.customers` c on
```

```
  o.customer_id=c.customer_id
  where order_delivered_customer_date is not null
),
avg_fast_delivery as (
  select
  customer_state,
  avg(early_delivery_days) as avg_early_delivery
  from delivery_diff
  group by customer_state
)
select
customer_state,
avg_early_delivery
from avg_fast_delivery
order by avg_early_delivery desc
limit 5;
```

Output:

Row	customer_state	avg_early_delivery
1	AC	19.762499999999996
2	RO	19.131687242798357
3	AP	18.731343283582092
4	AM	18.606896551724137
5	RR	16.414634146341459

VI. Analysis based on the payments:

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

Hint: We want you to count the no. of orders placed using different payment methods in each month over the past years.

Query:

```
select
format_date('%Y-%m',DATE(o.order_purchase_timestamp)) as year_month,
p.payment_type,
count(distinct o.order_id) AS total_orders
from `brazil_db.orders` o
join
`brazil_db.payments` p
on o.order_id = p.order_id
group by year_month, p.payment_type
order by year_month, p.payment_type;
```

Output:

Row	year_month	payment_type	total_orders
1	2016-09	credit_card	3
2	2016-10	UPI	63
3	2016-10	credit_card	253
4	2016-10	debit_card	2
5	2016-10	voucher	11
6	2016-12	credit_card	1
7	2017-01	UPI	197
8	2017-01	credit_card	582
9	2017-01	debit_card	9
10	2017-01	voucher	33

Row	year_month	payment_type	total_orders
11	2017-02	UPI	398
12	2017-02	credit_card	1347
13	2017-02	debit_card	13
14	2017-02	voucher	69
15	2017-03	UPI	590
16	2017-03	credit_card	2008
17	2017-03	debit_card	31
18	2017-03	voucher	123
19	2017-04	UPI	496
20	2017-04	credit_card	1835

Row	year_month	payment_type	total_orders
21	2017-04	debit_card	27
22	2017-04	voucher	115
23	2017-05	UPI	772
24	2017-05	credit_card	2833
25	2017-05	debit_card	30
26	2017-05	voucher	171
27	2017-06	UPI	707
28	2017-06	credit_card	2452
29	2017-06	debit_card	27
30	2017-06	voucher	142

Row	year_month	payment_type	total_orders
31	2017-07	UPI	845
32	2017-07	credit_card	3072
33	2017-07	debit_card	22
34	2017-07	voucher	205
35	2017-08	UPI	938
36	2017-08	credit_card	3272
37	2017-08	debit_card	34
38	2017-08	voucher	198
39	2017-09	UPI	903
40	2017-09	credit_card	3274

Row	year_month	payment_type	total_orders
41	2017-09	debit_card	43
42	2017-09	voucher	174
43	2017-10	UPI	993
44	2017-10	credit_card	3510
45	2017-10	debit_card	52
46	2017-10	voucher	208
47	2017-11	UPI	1509
48	2017-11	credit_card	5867
49	2017-11	debit_card	70
50	2017-11	voucher	267

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

Hint: We want you to count the no. of orders placed based on the no. of payment installments where at least one installment has been successfully paid.

Query:

```
select

payment_installments,

count(distinct order_id) as total_orders
```

```
from `brazil_db.payments`
where payment_installments >= 1
group by payment_installments
order by payment_installments;
```

Output:

Row	payment_installm...	total_orders
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644

Row	payment_installm...	total_orders
10	10	5315
11	11	23
12	12	133
13	13	16
14	14	15
15	15	74
16	16	5
17	17	8
18	18	27

Row	payment_installm...	total_orders
19	20	17
20	21	3
21	22	1
22	23	1
23	24	18