

1. 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.
  - b. Get the time range between which the orders were placed.
  - c. Count the Cities & States of customers who ordered during the given period.

a ans) the columns in the customer table are mentioned below and the data types are correct with respective to the each field

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

B ans) The data has a time range from 2016-09 to 2018-10 where the orders are placed

```
SELECT min(order_purchase_timestamp) as min_range,max(order_purchase_timestamp)
as max_range
FROM `ecommerce-408205.ecommerce.orders`
LIMIT 1000
```

C ans) there were 4119 cities and 27 state

```
SELECT count(distinct customer_city) as city_count, count(distinct customer_state) as
state_count
FROM `ecommerce-408205.ecommerce.customers`
```

## 2.In-depth Exploration:

- a. Is there a growing trend in the no. of orders placed over the past years?
- b. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
- c. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
  - i. 0-6 hrs : Dawn
  - ii. 7-12 hrs : Mornings
  - iii. 13-18 hrs : Afternoon
  - iv. 19-23 hrs : Night

2a ans) since the 2016 orders data is for a period of 4months , we can neglect that. We can see a growth trend from 2017 to 2018.

```
SELECT extract(year from order_purchase_timestamp) as order_year,count(distinct
order_id) as order_count
FROM `ecommerce-408205.ecommerce.orders`
group by extract(year from order_purchase_timestamp )
```

The screenshot shows a data query interface. On the left, there is a sidebar with a search bar and a list of resources under the 'ecommerce' category, including customers, geolocation, order\_items, order\_reviews, orders, payments, products, and sellers. The main area displays a SQL query in a text editor with a 'RUN' button. Below the query editor, the 'Query results' section is active, showing a table with two columns: 'order\_year' and 'order\_count'. The results are grouped by year, showing data for 2018, 2017, and 2016.

Row	order_year	order_count
1	2018	54011
2	2017	45101
3	2016	329

2b ans) there is a spike up in numbers during the month of may,july and August.near flat trend in the month of jan to april.flat declined numbers in the month of sep to dec.

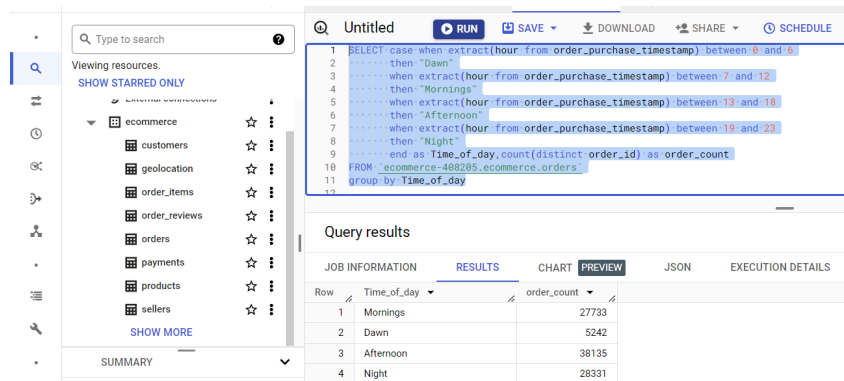
```
SELECT extract(month from order_purchase_timestamp) as order_month,count(distinct
order_id) as order_count
FROM `ecommerce-408205.ecommerce.orders`
group by extract(month from order_purchase_timestamp )
```

The screenshot shows a data query interface. On the left, there is a sidebar with a search bar and a list of resources under the 'ecommerce' category, including customers, geolocation, order\_items, order\_reviews, orders, payments, products, and sellers. The main area displays a SQL query in a text editor with a 'RUN' button. Below the query editor, the 'Query results' section is active, showing a table with two columns: 'order\_month' and 'order\_count'. The results are grouped by month, showing data for months 1 through 12.

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

2c ans) the orders during dawn is the lowest, orders in the afternoon is the highest

```
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 Time_of_day, count(distinct order_id) as order_count
FROM `ecommerce-408205.ecommerce.orders`
group by Time_of_day
```



The screenshot shows a SQL query editor with a sidebar on the left containing a search bar and a list of resources (ecommerce, customers, geolocation, order\_items, order\_reviews, orders, payments, products, sellers). The main area displays the SQL query, which is the same as the one provided in the previous block. Below the query, the 'Query results' section is visible, showing a table with 4 rows and 2 columns: 'Time\_of\_day' and 'order\_count'.

Row	Time_of_day	order_count
1	Mornings	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

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

- Get the month on month no. of orders placed in each state.
- How are the customers distributed across all the states?

3a ans)

```
SELECT c.customer_state,extract(month from o.order_purchase_timestamp) as  
order_month,count(distinct order_id) as count_orders  
FROM `ecommerce-408205.ecommerce.orders` as o  
join `ecommerce-408205.ecommerce.customers` as c  
on o.customer_id=c.customer_id  
group by c.customer_state,order_month  
order by customer_state,order_month
```

The screenshot shows a data platform interface. On the left is a sidebar with a search bar and a list of resources under 'ecommerce-408205'. The main area displays a SQL query in a text editor. Below the query, the 'Query results' section is active, showing a table with 7 rows and 4 columns: 'customer\_state', 'order\_month', and 'count\_orders'. The table data is as follows:

Row	customer_state	order_month	count_orders
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9

3b ans)

```
SELECT c.customer_state,count(distinct c.customer_id) as count_customer  
FROM `ecommerce-408205.ecommerce.orders` as o  
join `ecommerce-408205.ecommerce.customers` as c  
on o.customer_id=c.customer_id  
group by c.customer_state  
order by count_customer desc
```

The screenshot shows a data platform interface. On the left is a sidebar with a search bar and a list of resources under 'ecommerce-408205'. The main area displays a SQL query in a text editor. Below the query, the 'Query results' section is active, showing a table with 11 rows and 2 columns: 'customer\_state' and 'count\_customer'. The table data is as follows:

Row	customer_state	count_customer
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
11	PE	1652

4.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).

You can use the "payment\_value" column in the payments table to get the cost of orders.

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

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

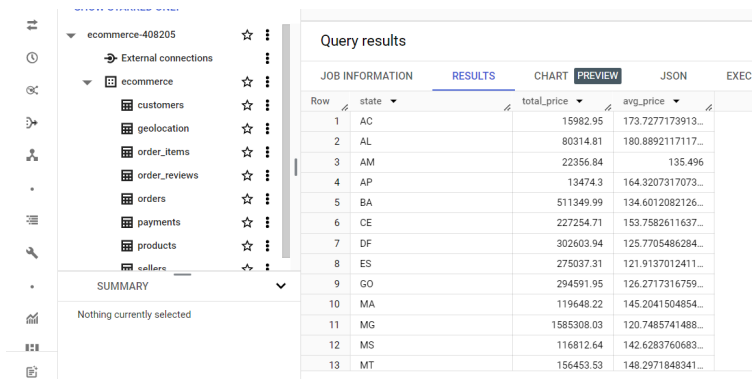
4a ans) `select order_year,order_month,cost,round((((cost-prev_cost)/prev_cost)*100,2) as percent_change`  
`from`  
`(`  
`select *,lag(cost,1,null) over(order by order_year,order_month) as prev_cost`  
`from`  
`(`  
`SELECT extract(Year from order_purchase_timestamp) as order_year,`  
`extract(Month from order_purchase_timestamp) as order_month,`  
`sum(p.payment_value) as cost`  
`FROM `ecommerce-408205.ecommerce.orders` as o`  
`join `ecommerce-408205.ecommerce.payments` as p`  
`on o.order_id=p.order_id`  
`group by order_year,order_month`  
`)`  
`where order_month between 1 AND 8`  
`order by order_year,order_month asc`  
`)`

Query results

Row	order_year	order_month	cost	percent_change
1	2017	1	138488.0399999...	null
2	2017	2	291908.0099999...	110.78
3	2017	3	449863.6000000...	54.11
4	2017	4	417788.0300000...	-7.13
5	2017	5	592918.8200000...	41.92
6	2017	6	511276.3800000...	-13.77
7	2017	7	592382.9200000...	15.86
8	2017	8	674396.3200000...	13.84
9	2018	1	1115004.180000...	65.33
10	2018	2	992463.3400000...	-10.99
11	2018	3	1159652.119999...	16.85
12	2018	4	1160785.479999...	0.1
13	2018	5	1153982.149999...	-0.59

4b ans)

```
select distinct state,total_price,avg_price
from
(
select c.customer_state as state,sum(oi.price) over(partition by c.customer_state) as
total_price,
      avg(oi.price) over(partition by c.customer_state) as avg_price
from `ecommerce-408205.ecommerce.orders` as o
join `ecommerce-408205.ecommerce.customers` as c
on o.customer_id=c.customer_id
left join `ecommerce-408205.ecommerce.order_items` as oi
on o.order_id=oi.order_id
order by c.customer_state asc
)
order by state asc
```



The screenshot shows a data analytics interface. On the left is a sidebar with a tree view of data sources: 'ecommerce-408205' (expanded) containing 'External connections', 'ecommerce' (expanded) containing 'customers', 'geolocation', 'order\_items', 'order\_reviews', 'orders', 'payments', 'products', and 'callers'. Below this is a 'SUMMARY' section with the text 'Nothing currently selected'. On the right is a 'Query results' panel with tabs for 'JOB INFORMATION', 'RESULTS', 'CHART', 'PREVIEW', 'JSON', and 'EXECUTION'. The 'RESULTS' tab is active, displaying a table with 13 rows and 4 columns: 'Row', 'state', 'total\_price', and 'avg\_price'. The data is sorted by 'state' in ascending order.

Row	state	total_price	avg_price
1	AC	15982.95	173.7277173913...
2	AL	80314.81	180.8892117117...
3	AM	22356.84	135.496
4	AP	13474.3	164.9207317073...
5	BA	511349.99	134.6012082126...
6	CE	227254.71	153.7582611637...
7	DF	302603.94	125.7705486284...
8	ES	275037.31	121.9137012411...
9	GO	294591.95	126.2717316759...
10	MA	119648.22	145.2041504854...
11	MG	1585308.03	120.7485741488...
12	MS	116812.64	142.6283760683...
13	MT	156453.53	148.2971848341...

4c ans)

```
select distinct state,total_fprice,avg_fprice
from
(
select c.customer_state as state,sum(oi.freight_value) over(partition by c.customer_state) as
total_fprice,
      avg(oi.freight_value) over(partition by c.customer_state) as avg_fprice
from `ecommerce-408205.ecommerce.orders` as o
join `ecommerce-408205.ecommerce.customers` as c
on o.customer_id=c.customer_id
left join `ecommerce-408205.ecommerce.order_items` as oi
on o.order_id=oi.order_id
order by c.customer_state asc
)
order by state asc
```

Query results

Row	state	total_fprice	avg_fprice
1	AC	3686.75	40.07336956521...
2	AL	15914.59	35.84367117117...
3	AM	5478.89	33.20539393939...
4	AP	2788.5	34.00609756097...
5	BA	100156.68	26.36395893656...
6	CE	48351.59	32.71420162381...
7	DF	50625.5	21.04135494596...
8	ES	49764.6	22.05877659574...
9	GO	53114.98	22.76681525932...
10	MA	31523.77	38.25700242718...
11	MG	270853.46	20.63016680630...
12	MS	19144.03	23.37488400488...
13	MT	29715.43	28.16628436018...

## 5. 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.

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

- $\text{time\_to\_deliver} = \text{order\_delivered\_customer\_date} - \text{order\_purchase\_timestamp}$
- $\text{diff\_estimated\_delivery} = \text{order\_estimated\_delivery\_date} - \text{order\_delivered\_customer\_date}$

- B. Find out the top 5 states with the highest & lowest average freight value.  
 C. Find out the top 5 states with the highest & lowest average delivery time.  
 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 averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

5a ans)

```
select order_id,datetime_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_deliver,
       datetime_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery
from `ecommerce-408205.ecommerce.orders`
order by order_id asc
```

The screenshot shows a data management interface. On the left is a sidebar with 'External connections' and a list of tables: customers, geolocation, order\_items, order\_reviews, orders, payments, products, and sellers. The 'Query results' table on the right displays data for 12 rows, with columns for Row, order\_id, time\_to\_deliver, and diff\_estimated\_delivery.

Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	8
2	0001877f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcd0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18
11	00061f2a7bc09da83e415a52d...	4	10
12	00063b381e2406b52ad42947...	10	0

5b ans)

select distinct state,avg\_fprice

from

(

select c.customer\_state as state,avg(oi.freight\_value) over(partition by c.customer\_state) as avg\_fprice

from `ecommerce-408205.ecommerce.orders` as o

join `ecommerce-408205.ecommerce.customers` as c

on o.customer\_id=c.customer\_id

left join `ecommerce-408205.ecommerce.order\_items` as oi

on o.order\_id=oi.order\_id

order by c.customer\_state asc

)

order by avg\_fprice asc

limit 5

state avg\_fprice

SP 15.147275390419187

PR 20.531651567944252

MG 20.630166806306651

RJ 20.960923931682558

DF 21.041354945968411



```
select distinct state,avg_fprice
from
(
select c.customer_state as state,avg(oi.freight_value) over(partition by c.customer_state) as
avg_fprice
from `ecommerce-408205.ecommerce.orders` as o
join `ecommerce-408205.ecommerce.customers` as c
on o.customer_id=c.customer_id
left join `ecommerce-408205.ecommerce.order_items` as oi
on o.order_id=oi.order_id
order by c.customer_state asc
)
order by avg_fprice desc
limit 5
```

state	avg_fprice
RR	42.984423076923079
PB	42.723803986710962
RO	41.069712230215828
AC	40.073369565217391
PI	39.147970479704796

5c ans)

```
select c.customer_state as state,  
avg(datetime_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as  
avg_delivery_time,  
from `ecommerce-408205.ecommerce.orders` o  
left join `ecommerce-408205.ecommerce.customers` c  
on o.customer_id=c.customer_id  
group by c.customer_state  
order by avg_delivery_time asc  
limit 5
```

```
select c.customer_state as state,  
avg(datetime_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as  
avg_delivery_time,  
from `ecommerce-408205.ecommerce.orders` o  
left join `ecommerce-408205.ecommerce.customers` c  
on o.customer_id=c.customer_id  
group by c.customer_state  
order by avg_delivery_time desc  
limit 5
```

state	avg_delivery_time
SP	8.2980614890725874
PR	11.526711354864908
MG	11.543813298106569
DF	12.509134615384616
SC	14.479560191711331

state	avg_delivery_time
RR	28.975609756097562
AP	26.731343283582085
AM	25.986206896551728
AL	24.040302267002513
PA	23.316067653276981

5d ans)

```
select c.customer_state as  
state,avg(datetime_diff(order_estimated_delivery_date,order_delivered_customer_date,day)  
) as avg_delivery_rate  
from `ecommerce-408205.ecommerce.orders` o  
left join `ecommerce-408205.ecommerce.customers` c  
on o.customer_id=c.customer_id  
group by c.customer_state  
order by avg_delivery_rate asc  
limit 5
```

state      avg\_delivery\_rate  
AL 7.9471032745591943  
MA      8.76847977684797  
SE 9.1731343283582127  
ES 9.6185463659147885  
BA 9.93488943488941

6. Analysis based on the payments:

- Find the month on month no. of orders placed using different payment types.
- Find the no. of orders placed on the basis of the payment installments that have been paid.

6a ans) 

```
select extract(year from o.order_purchase_timestamp) as year,extract(Month from  
o.order_purchase_timestamp) as month,p.payment_type as payment_type,count(o.order_id) as  
order_count  
from `ecommerce-408205.ecommerce.orders` o  
join `ecommerce-408205.ecommerce.payments` p  
on o.order_id=p.order_id  
group by year,month,payment_type  
order by year,month,payment_type asc
```

Row	year	month	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61
11	2017	2	UPI	398
12	2017	2	credit_card	1164

6b ans)

```
select p.payment_installments as payment_installments, count(o.order_id) as order_count
from `ecommerce-408205.ecommerce.orders` o
join `ecommerce-408205.ecommerce.payments` p
on o.order_id=p.order_id
group by payment_installments
order by payment_installments asc
```

The screenshot displays the BigQuery web interface. On the left, a sidebar shows a project named 'ecommerce' with various tables listed: customers, geolocation, order\_items, order\_reviews, orders, payments, products, and sellers. The main panel is titled 'Untitled' and contains a SQL query. Below the query editor, the 'Query results' section is active, showing a table with two columns: 'payment\_installments' and 'order\_count'. The results are ordered by 'payment\_installments' in ascending order.

Row	payment_installments	order_count
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