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

1. Data type of columns in a table

```
1. select column_name, data_type

from `sql_project.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

```
2. select column_name, data_type

from `sql_project.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

```
3. select column_name, data_type

from `sql_project.INFORMATION_SCHEMA.COLUMNS`

where table_name = 'order_items'
```

Row	column_name	data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64

```
4. select column_name, data_type

from `sql_project.INFORMATION_SCHEMA.COLUMNS`

where table_name = 'order_reviews'
```

Row	column_name	data_type
1	review_id	STRING
2	order_id	STRING
3	review_score	INT64
4	review_comment_title	STRING
5	review_creation_date	TIMESTAMP
6	review_answer_timestamp	TIMESTAMP

```
5. select column_name, data_type

from `sql_project.INFORMATION_SCHEMA.COLUMNS`

where table_name = 'orders'
```

Row	column_name	data_type
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP

6.
```
select column_name, data_type

from `sql_project.INFORMATION_SCHEMA.COLUMNS`

where table_name = 'payments'
```

Row	column_name	data_type
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

7.
```
select column_name, data_type

from `sql_project.INFORMATION_SCHEMA.COLUMNS`

where table_name = 'products'
```

Row	column_name	data_type
1	product_id	STRING
2	product_category	STRING
3	product_name_length	INT64
4	product_description_length	INT64
5	product_photos_qty	INT64
6	product_weight_g	INT64
7	product_length_cm	INT64
8	product_height_cm	INT64
9	product_width_cm	INT64

```
8. select column_name, data_type

from `sql_project.INFORMATION_SCHEMA.COLUMNS`

where table_name = 'sellers'
```

Row	column_name	data_type
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

2. Time period for which the data is given

```
select extract(year from min(order_purchase_timestamp)) as min, extract(year from
max(order_purchase_timestamp)) as max

from `sql_project.orders`
```

Row	min	max
1	2016	2018

3. Cities and States of customers ordered during the given period

```
select distinct customer_city, customer_state

from `sql_project.customers`

where customer_id in (

  select customer_id

  from `sql_project.orders`

)

order by customer_state, customer_city
```

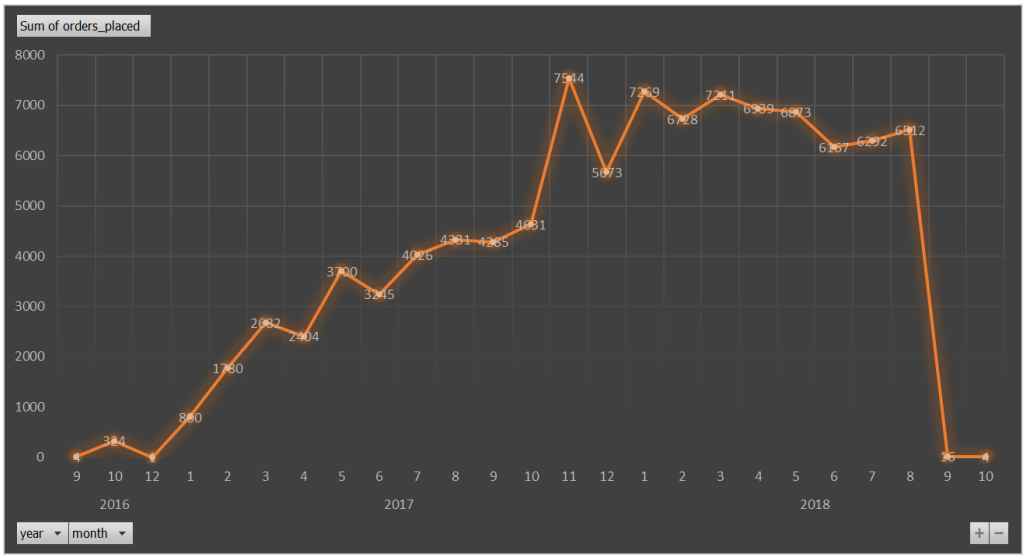
Row	customer_city	customer_state
1	brasileia	AC
2	cruzeiro do sul	AC
3	epitaciolandia	AC
4	manoel urbano	AC
5	porto acre	AC
6	rio branco	AC
7	senador guiomard	AC
8	xapuri	AC
9	agua branca	AL
10	anadia	AL

2. In-depth Exploration:

1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

Regarding seasonality, there seems to be some seasonality in e-commerce in Brazil, with peaks in orders placed occurring in specific months. The data shows a peak in Aug with 10,843 orders placed, and another peak in May with 10,573 orders placed. Additionally, there seems to be a dip in orders placed during the months of November, December, and January.

Based on this information, a possible actionable insight and recommendation could be to analyze the reasons behind the seasonality in orders placed, such as holidays or marketing campaigns, and leverage this knowledge to plan marketing and promotional activities for peak months to maximize sales. Additionally, it may be useful to investigate why there is a dip in orders during the months of September, October, and November, and take steps to improve sales during those months, such as offering special discounts or promotions.



```sql
select extract(month from order_purchase_timestamp) as month, count(order_id) as orders_placed

from `sql_project.orders`

group by month

order by orders_placed desc
```

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

2. **What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?**

Based on the provided data, it seems that Brazilian customers tend to buy more in the afternoon, with 38,135 orders placed during that time, followed by the night with 28,331 orders placed. The morning and dawn have fewer orders placed, with 27,733 and 5,242 orders placed, respectively.

One actionable insight and recommendation could be to leverage this information to optimize marketing campaigns and promotions for specific times of the day. For example, if a company finds that their sales are highest in the afternoon, they can plan to run targeted advertising campaigns during that time. This can be achieved through targeted ads on social media or through email marketing campaigns.

Additionally, companies can adjust their customer service and support schedules to align with the peak buying times. If more customers are placing orders in the afternoon, it would make sense to have more customer support representatives available during that time to provide assistance to customers and resolve any issues they may have.

```
select count(order_id) as no_of_orders_placed,

case when time_st>=0 and time_st<=6

    then "Dawn"

    when time_st>=7 and time_st<=12

    then "Morning"

    when time_st>=13 and time_st<=18

    then "Afternoon"

    else "Night"

end as time_of_the_day

from (select extract(hour from order_purchase_timestamp) as time_st, order_id from
`sql_project.orders`) s

group by time_of_the_day

order by no_of_orders_placed desc
```

Row	no_of_orders_pl	time_of_the_day
1	38135	Afternoon
2	28331	Night
3	27733	Morning
4	5242	Dawn

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

1. Get month on month orders by states

Firstly, companies can use this data to identify the states with the highest and lowest order volumes, and tailor their marketing and sales strategies accordingly. For example, companies can focus more on states with higher order volumes by running targeted advertising campaigns or offering promotions that are specific to those states. This can help to increase sales and improve customer engagement in those regions.

Secondly, by analyzing the data on a monthly basis, companies can identify trends and patterns in customer behavior. For instance, if a state experiences a significant increase or decrease in orders from one month to another, it may be useful to investigate the reasons behind this trend. Companies can then use this information to adjust their marketing and sales strategies to better cater to their customers' needs.

Thirdly, companies can also use this data to optimize their supply chain and inventory management processes. For example, if a state consistently places high orders for a particular product or category, companies can ensure that they have sufficient inventory in that region to meet demand. This can help to improve customer satisfaction and reduce order processing times.

Finally, companies can also use this data to identify opportunities for expansion. By analyzing the order volumes in different states, companies can identify regions that have high growth potential and consider expanding their operations or opening new stores in those areas.

```sql
select c.customer_state, extract(month from o.order_purchase_timestamp) as month,
count(o.order_id) as no_of_orders_placed

from `sql_project.orders` o inner join `sql_project.customers` c on c.customer_id =
o.customer_id

group by c.customer_state, month

order by c.customer_state, month
```

Row	customer_state	month	no_of_orders_pl
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
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5

2. **Distribution of customers across the states in Brazil**

Firstly, companies can use this data to identify states with the highest number of customers and prioritize their marketing efforts in those regions. For example, if a company has a large customer base in (SP), they can run targeted advertising

campaigns or offer promotions that are specific to that region. This can help to improve customer engagement and increase sales.

Secondly, companies can use this data to identify states with a low number of customers and explore opportunities to expand their operations in those regions. For example, if a company has a low customer base in (RR), they can consider opening a new store in that region or running targeted advertising campaigns to increase brand awareness and customer engagement.

Thirdly, companies can use this data to optimize their supply chain and inventory management processes. For example, if a company has a large customer base in (RJ), they can ensure that they have sufficient inventory in that region to meet demand. This can help to improve customer satisfaction and reduce order processing times.

Finally, companies can also use this data to identify opportunities for collaboration with other businesses in different states. For example, if a company has a large customer base in (SP), they can explore opportunities to partner with local businesses in that region to expand their customer reach and increase sales.

```sql
select customer_state, count(customer_unique_id) as no_of_customers

from  `sql_project.customers`

group by customer_state

order by no_of_customers desc
```

Row	customer_state	no_of_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
12	CE	1336

4. **Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.**
    1. **Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment_value” column in payments table**

There has been 136.98 percentage increase in payment_value in 2018

2017 -

No of orders - 24391

Payment_value - 3669022.11

avg value per order - 150.42

2018 -

No of orders - 55995

Payment_value - 8694733.83

avg value per order - 155.27

The significant increase in payment_value in 2018 suggests a growing e-commerce market in Brazil, which could have a positive impact on the economy. The increase in average order value could also indicate that customers are becoming more comfortable with making larger purchases online.

Based on the increase in average order value, it may be beneficial for e-commerce companies to focus on offering higher-value products or services to customers. This could also lead to an increase in revenue for the companies.

Additionally, it may be useful for e-commerce companies to analyze the factors that contributed to the increase in payment_value, such as changes in consumer behavior or market trends. This information could be used to inform marketing and sales strategies in the future.

```
select year, no_of_orders, sum_value, round(((sum_value-prev_sum)/prev_sum)*100, 2) as
percentage_increase_in_cost

from

(select year, no_of_orders, sum_value, lag(sum_value) over(order by year) as prev_sum

from

(select extract(year from o.order_purchase_timestamp) as year,  count(o.order_id) as
no_of_orders, sum(p.payment_value) as sum_value

from `sql_project.payments` p inner join `sql_project.orders` o on o.order_id =
p.order_id

where extract(month from o.order_purchase_timestamp)<9 and extract(year from
o.order_purchase_timestamp) in (2017,2018)

group by year) o) t

order by year
```

Row	year	no_of_orders	sum_value	percentage_incr
1	2017	24391	3669022.11...	*null*
2	2018	55995	8694733.83...	136.98

```sql
select year, month, no_of_orders, sum_value,
round(((sum_value-prev_sum)/prev_sum)*100, 2) as percentage_increase_in_cost

from

(select year, month, no_of_orders, sum_value, lag(sum_value) over(order by year,
month) as prev_sum

from

(select extract(year from o.order_purchase_timestamp) as year, extract(month from
o.order_purchase_timestamp) as month, count(o.order_id) as no_of_orders,
sum(p.payment_value) as sum_value

from `sql_project.payments` p inner join `sql_project.orders` o on o.order_id =
p.order_id

where extract(month from o.order_purchase_timestamp)<9 and extract(year from
o.order_purchase_timestamp) in (2017,2018)

group by year, month) o) t

order by year, month
```

Row	year	month	no_of_orders	sum_value	percentage_incr
1	2017	1	850	138488.039...	*null*
2	2017	2	1886	291908.009...	110.78
3	2017	3	2837	449863.600...	54.11
4	2017	4	2571	417788.030...	-7.13
5	2017	5	3944	592918.820...	41.92
6	2017	6	3436	511276.380...	-13.77
7	2017	7	4317	592382.920...	15.86
8	2017	8	4550	674396.320...	13.84
9	2018	1	7563	1115004.18...	65.33
10	2018	2	6952	992463.340...	-10.99
11	2018	3	7512	1159652.11...	16.85
12	2018	4	7209	1160785.47...	0.1

2. Mean & Sum of price and freight value by customer state

min avg price - SP - 109.65

max avg price - PB - 191.48

min avg_freight_val - SP - 15.15

max avg_freight_val - RR - 42.98

min sum_price - RR - 7829.43

max sum_price - SP - 5202955.05

min sum_freight_val - RR - 2235.19

max sum_freight_val - SP - 718723.07

The variation in order numbers and averages across different states in Brazil suggests that businesses should tailor their marketing and sales strategies based on the characteristics of each state. For example, in states with high average prices, businesses may want to focus on promoting higher-end products and services.

The variation in average price and freight values across different states in Brazil suggests that businesses should consider pricing strategies and logistics solutions that are specific to each state. For example, in states with high average freight values, businesses may want to explore partnerships with local logistics providers to reduce costs.

The minimum and maximum sum and average of price and freight values by state can provide businesses with valuable insights into the purchasing behaviors of customers in each state. By analyzing these data points, businesses can identify opportunities to optimize their pricing and logistics strategies to improve customer satisfaction and drive sales.

```sql
select c.customer_state, round(avg(oi.price),2) as avg_price,
round(avg(oi.freight_value),2) as avg_freight_val, round(sum(oi.price),2) as
sum_price, round(sum(oi.freight_value),2) as sum_freight_val

from `sql_project.order_items` oi inner join `sql_project.orders` o on
oi.order_id=o.order_id inner join `sql_project.customers` c on c.customer_id =
o.customer_id

group by c.customer_state

order by c.customer_state
```

Row	customer_state	avg_price	avg_freight_val	sum_price	sum_freight_val
1	AC	173.73	40.07	15982.95	3686.75
2	AL	180.89	35.84	80314.81	15914.59
3	AM	135.5	33.21	22356.84	5478.89
4	AP	164.32	34.01	13474.3	2788.5
5	BA	134.6	26.36	511349.99	100156.68
6	CE	153.76	32.71	227254.71	48351.59
7	DF	125.77	21.04	302603.94	50625.5
8	ES	121.91	22.06	275037.31	49764.6
9	GO	126.27	22.77	294591.95	53114.98
10	MA	145.2	38.26	119648.22	31523.77
11	MG	120.75	20.63	1585308.03	270853.46

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
select order_id, date_diff(order_delivered_customer_date, order_purchase_timestamp,
day) as actual_delivery,
date_diff(order_estimated_delivery_date,order_purchase_timestamp, day) as
estimated_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as
diff_estimated_and_actual_delivery

from `sql_project.orders`

where order_delivered_customer_date is not null
```

Row	order_id	actual_delivery	estimated_delive	diff_estimated_a
1	1950d777989f6a877539f5379...	30	17	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	59	28
3	65d1e226dfaeb8cdc42f66542...	35	52	16
4	635c894d068ac37e6e03dc54e...	30	32	1
5	3b97562c3aee8bdedcb5c2e45...	32	33	0
6	68f47f50f04c4cb6774570cfde...	29	31	1
7	276e9ec344d3bf029ff83a161c...	43	39	-4
8	54e1a3c2b97fb0809da548a59...	40	36	-4
9	fd04fa4105ee8045f6a0139ca5...	37	35	-1
10	302bb8109d097a9fc6e9cefc5...	33	28	-5
11	66057d37308e787052a32828...	38	32	-6

```
select count(order_id) as no_of_orders,

case when diff_estimated_and_actual_delivery<0

then "order_delayed"

when diff_estimated_and_actual_delivery>0
```

```
then "order_early"

else "order_on_time"

end as delivery_tag

from

(select order_id, date_diff( order_delivered_customer_date, order_purchase_timestamp,
day) as actual_delivery,
date_diff(order_estimated_delivery_date,order_purchase_timestamp, day) as
estimated_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as
diff_estimated_and_actual_delivery

from `sql_project.orders`

where order_delivered_customer_date is not null) t

group by delivery_tag
```

Row	no_of_orders	delivery_tag
1	87187	order_early
2	6535	order_delayed
3	2754	order_on_time

2. **Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:**
    - **time_to_delivery = order_purchase_timestamp-order_delivered_customer_date**
    - **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**

```
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 `sql_project.orders`
```

Row	order_id	time_to_delivery	diff_estimated_c
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5
11	66057d37308e787052a32828...	38	-6
12	19125e045e554eebfd7576e72...	26	2

3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_val,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,
day)),2) as time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,
day)),2) as diff_estimated_delivery

from `sql_project.order_items` oi inner join `sql_project.orders` o  on
oi.order_id=o.order_id right join `sql_project.customers` c on
c.customer_id=o.customer_id

group by c.customer_state

order by c.customer_state
```

Row	customer_state	avg_freight_val	time_to_delivery	diff_estimated_c
1	AC	40.07	20.33	20.01
2	AL	35.84	23.99	7.98
3	AM	33.21	25.96	18.98
4	AP	34.01	27.75	17.44
5	BA	26.36	18.77	10.12
6	CE	32.71	20.54	10.26
7	DF	21.04	12.5	11.27
8	ES	22.06	15.19	9.77
9	GO	22.77	14.95	11.37
10	MA	38.26	21.2	9.11
11	MG	20.63	11.52	12.4
12	MS	23.37	15.11	10.34

4. **Sort the data to get the following:**
    - **Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5**

```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_val,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,
day)),2) as time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,
day)),2) as diff_estimated_delivery

from `sql_project.order_items` oi inner join `sql_project.orders` o  on
oi.order_id=o.order_id right join `sql_project.customers` c on
c.customer_id=o.customer_id

group by c.customer_state

order by avg_freight_val asc

limit 5
```

Row	customer_state	avg_freight_val	time_to_delivery	diff_estimated_c
1	SP	15.15	8.26	10.27
2	PR	20.53	11.48	12.53
3	MG	20.63	11.52	12.4
4	RJ	20.96	14.69	11.14
5	DF	21.04	12.5	11.27

```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_val,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,
day)),2) as time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,
day)),2) as diff_estimated_delivery

from `sql_project.order_items` oi inner join `sql_project.orders` o  on
oi.order_id=o.order_id right join `sql_project.customers` c on
c.customer_id=o.customer_id

group by c.customer_state

order by avg_freight_val desc

limit 5
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECU

Row	customer_state	avg_freight_val	time_to_delivery	diff_estimated_c
1	RR	42.98	27.83	17.43
2	PB	42.72	20.12	12.15
3	RO	41.07	19.28	19.08
4	AC	40.07	20.33	20.01
5	PI	39.15	18.93	10.68

- **Top 5 states with highest/lowest average time to delivery**

```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_val,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,
day)),2) as time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,
day)),2) as diff_estimated_delivery

from `sql_project.order_items` oi inner join `sql_project.orders` o  on
oi.order_id=o.order_id right join `sql_project.customers` c on
c.customer_id=o.customer_id

group by c.customer_state

order by time_to_delivery desc

limit 5
```

Row	customer_state	avg_freight_val	time_to_delivery	diff_estimated_c
1	RR	42.98	27.83	17.43
2	AP	34.01	27.75	17.44
3	AM	33.21	25.96	18.98
4	AL	35.84	23.99	7.98
5	PA	35.83	23.3	13.37

```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_val,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,
day)),2) as time_to_delivery,
```

```
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,
day)),2) as diff_estimated_delivery

from `sql_project.order_items` oi inner join `sql_project.orders` o  on
oi.order_id=o.order_id right join `sql_project.customers` c on
c.customer_id=o.customer_id

group by c.customer_state

order by time_to_delivery asc

limit 5
```

Row	customer_state	avg_freight_val	time_to_delivery	diff_estimated_c
1	SP	15.15	8.26	10.27
2	PR	20.53	11.48	12.53
3	MG	20.63	11.52	12.4
4	DF	21.04	12.5	11.27
5	SC	21.47	14.52	10.67

- **Top 5 states where delivery is really fast/ not so fast compared to estimated date**

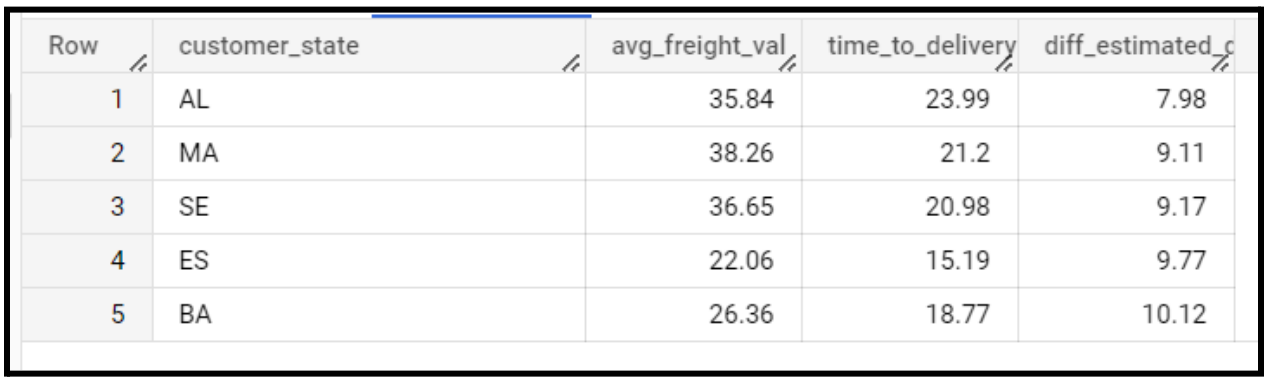
```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_val,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,
day)),2) as time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,
day)),2) as diff_estimated_delivery

from `sql_project.order_items` oi inner join `sql_project.orders` o  on
oi.order_id=o.order_id right join `sql_project.customers` c on
c.customer_id=o.customer_id

group by c.customer_state

order by diff_estimated_delivery asc
```

```
limit 5
```



Row	customer_state	avg_freight_val	time_to_delivery	diff_estimated_d
1	AL	35.84	23.99	7.98
2	MA	38.26	21.2	9.11
3	SE	36.65	20.98	9.17
4	ES	22.06	15.19	9.77
5	BA	26.36	18.77	10.12

```
select c.customer_state, round(avg(oi.freight_value),2) as avg_freight_val,
round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,
day)),2) as time_to_delivery,
round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,
day)),2) as diff_estimated_delivery

from `sql_project.order_items` oi inner join `sql_project.orders` o  on
oi.order_id=o.order_id right join `sql_project.customers` c on
c.customer_id=o.customer_id

group by c.customer_state

order by diff_estimated_delivery desc

limit 5
```

Row	customer_state	avg_freight_val	time_to_delivery	diff_estimated_d
1	AC	40.07	20.33	20.01
2	RO	41.07	19.28	19.08
3	AM	33.21	25.96	18.98
4	AP	34.01	27.75	17.44
5	RR	42.98	27.83	17.43

It seems like there is a mix of good and bad performance in terms of delivery time and freight value across different states in Brazil.

Orders being delivered before the estimated delivery time is a positive sign, and it's great to see that it happened in a majority of the orders (87187 out of 96176). However, there were also orders that were delayed (6535) and orders that were delivered on time (2754). It would be beneficial to investigate the reasons behind the delayed orders and see if there are any areas of improvement.

In terms of freight value, it's interesting to note that the highest average freight value is in RR (42.98 days) and the lowest average freight value is in SP (15.15 days). This could be because of the location of the seller or the buyer, or the type of product being shipped. It might be worth investigating these factors further to see if there is any room for improvement.

Regarding delivery time, the highest average delivery time is in RR (27.83 days) and the lowest average delivery time is in SP (8.26 days). The reason for the difference in delivery time could be due to the distance between the seller and the buyer or the logistics infrastructure in the respective states. It might be beneficial to investigate if there are any bottlenecks in the logistics process in the states with longer delivery times.

Finally, it's worth noting that delivery is faster than estimated delivery in AC (20.02 days), which is a positive sign. However, delivery is slower than estimated delivery in AL (7.68 days), which might not be a great customer experience. It would be worth investigating the reasons for this delay and see if there are any areas of improvement.

6. Payment type analysis:

1. **Month over Month count of orders for different payment types**

payment_type no_of_orders

credit_card 76795

UPI 19784

voucher 5775

debit_card 1529

Based on the payment type analysis, the majority of customers, 76,795, prefer to use credit card as the payment method. UPI is the second most popular payment type, with 19,784 orders placed using this method. Voucher is the third most popular payment type, with 5,775 orders placed using this method. Debit card is the least popular payment type, with only 1,529 orders placed using this method.

This analysis can be used to understand customers' payment preferences and to make sure that the payment methods offered by the e-commerce platform are aligned with customers' preferences. Additionally, it can be used to identify potential payment methods to add or remove from the platform based on their popularity.

```
select p.payment_type, extract(month from o.order_purchase_timestamp) as Month,
count(p.order_id) as no_of_orders

from `sql_project.payments` p inner join `sql_project.orders` o on
p.order_id=o.order_id

group by p.payment_type, Month
```

```
order by p.payment_type, Month
```

Row	payment_type	Month	no_of_orders
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807
7	UPI	7	2074
8	UPI	8	2077
9	UPI	9	903
10	UPI	10	1056
11	UPI	11	1509

2. Count of orders based on the no. of payment installments

```
select payment_installments, count(order_id) as no_of_orders

from `sql_project.payments`

group by payment_installments

order by no_of_orders desc
```

Row	payment_installr	no_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644
11	12	133
12	15	74

Based on the data, it seems that most customers opt for paying in 1 installment, with 52,546 orders. The next most popular option is paying in 2 installments, with 12,413 orders. As the number of installments increases, the number of orders decreases. It's interesting to note that there are 2 orders with no payment installments listed, which could be an error in the data or a unique case. Based on the results, we can suggest that the store should continue the option of paying in installments.