Business_Case-Target_SQL

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**
    2. **Time period for which the data is given**
    3. **Cities and States covered in the dataset**

1. Data Type of columns in a table

1.1. Customers Table:

Field name	Type	Mode
customer_id	STRING	NULLABLE
customer_unique_id	STRING	NULLABLE
customer_zip_code_prefix	INTEGER	NULLABLE
customer_city	STRING	NULLABLE
customer_state	STRING	NULLABLE

1.2. Sellers Table:

Field name	Type	Mode
seller_id	STRING	NULLABLE
seller_zip_code_prefix	INTEGER	NULLABLE
seller_city	STRING	NULLABLE
seller_state	STRING	NULLABLE

1.3. Payments Table:

Field name	Type	Mode
order_id	STRING	NULLABLE
payment_sequential	INTEGER	NULLABLE
payment_type	STRING	NULLABLE
payment_installments	INTEGER	NULLABLE
payment_value	FLOAT	NULLABLE

1.4. Products Table:

Field name	Type	Mode
product_id	STRING	NULLABLE
product_category	STRING	NULLABLE
product_name_length	INTEGER	NULLABLE
product_description_length	INTEGER	NULLABLE
product_photos_qty	INTEGER	NULLABLE
product_weight_g	INTEGER	NULLABLE
product_length_cm	INTEGER	NULLABLE
product_height_cm	INTEGER	NULLABLE
product_width_cm	INTEGER	NULLABLE

1.5. Orders Table:

Field name	Type	Mode
order_id	STRING	NULLABLE
customer_id	STRING	NULLABLE
order_status	STRING	NULLABLE
order_purchase_timestamp	TIMESTAMP	NULLABLE
order_approved_at	TIMESTAMP	NULLABLE
order_delivered_carrier_date	TIMESTAMP	NULLABLE
order_delivered_customer_date	TIMESTAMP	NULLABLE
order_estimated_delivery_date	TIMESTAMP	NULLABLE

1.6. Order_reviews Table:

Field name	Type	Mode
review_id	STRING	NULLABLE
order_id	STRING	NULLABLE
review_score	INTEGER	NULLABLE
review_comment_title	STRING	NULLABLE
review_creation_date	TIMESTAMP	NULLABLE
review_answer_timestamp	TIMESTAMP	NULLABLE

1.7. Order_items Table:

Field name	Type	Mode
order_id	STRING	NULLABLE
order_item_id	INTEGER	NULLABLE
product_id	STRING	NULLABLE
seller_id	STRING	NULLABLE
shipping_limit_date	TIMESTAMP	NULLABLE
price	FLOAT	NULLABLE
freight_value	FLOAT	NULLABLE

1.8. Geolocation Table:

Field name	Type	Mode
geolocation_zip_code_prefix	INTEGER	NULLABLE
geolocation_lat	FLOAT	NULLABLE
geolocation_lng	FLOAT	NULLABLE
geolocation_city	STRING	NULLABLE
geolocation_state	STRING	NULLABLE

2. Time period for which data is given

```
SELECT MIN(order_purchase_timestamp) AS Period_start_date,MAX(order_purch
ase_timestamp) AS Period_end_date FROM `dsml-target-
project.target.orders`;
```

Query results

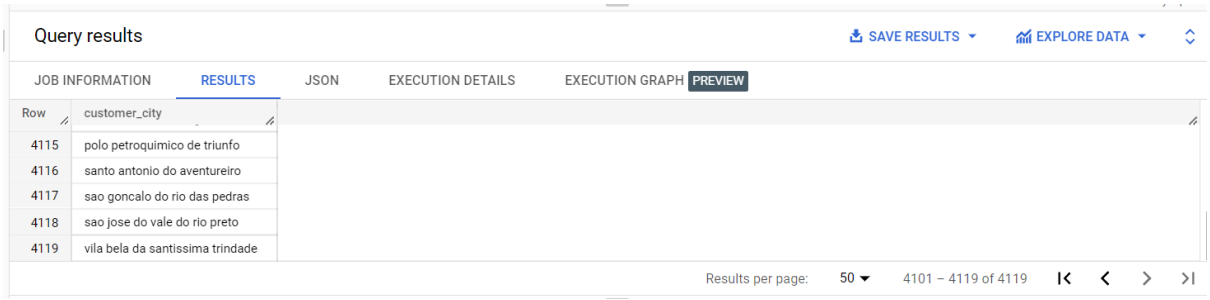
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

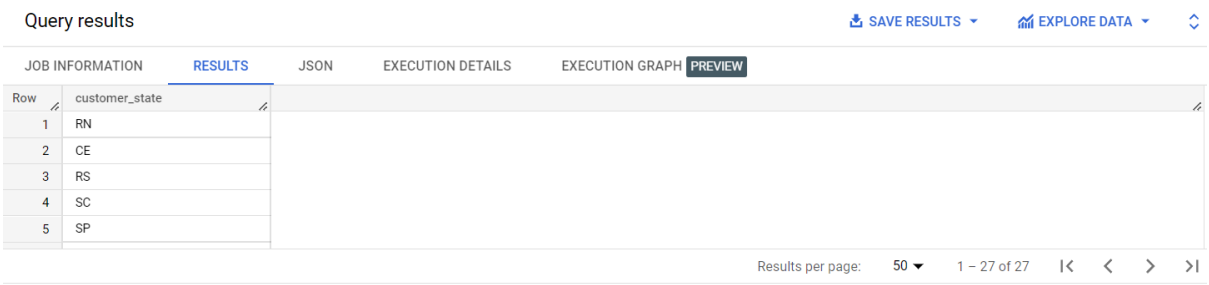
The data is present from 4th September 2016 to 17th October 2018.

3. Cities and states covered in the dataset

```
SELECT DISTINCT(customer_city) FROM `dsml-target-project.target.customers`;
```




```
SELECT DISTINCT(customer_state) FROM `dsml-target-project.target.customers`;
```



There are orders of customer from around 4119 cities in 27 states are covered in the dataset.

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

1. Trend on e-commerce in Brazil:

```
SELECT COUNT(DISTINCT order_id) AS count_of_Orders,EXTRACT(YEAR FROM order_pur
chase_timestamp) AS Year FROM `dsml-target-project.target.orders`
GROUP BY Year ORDER BY Year;
```

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

```
SELECT EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,EXTRACT(YEAR FROM
order_purchase_timestamp) AS Year, COUNT(DISTINCT order_id) AS count_of_Orders
FROM `dsml-target-project.target.orders`
GROUP BY Month,Year ORDER BY Year,Month;
```

Row	Month	Year	count_of_Orders
1	9	2016	4
2	10	2016	324
3	12	2016	1
4	1	2017	800
5	2	2017	1780
6	3	2017	2682
7	4	2017	2404
8	5	2017	3700
9	6	2017	3245
10	7	2017	4026
11	8	2017	4331
12	9	2017	4285
13	10	2017	4631
14	11	2017	7544
15	12	2017	5673
16	1	2018	7269
17	2	2018	6728
18	3	2018	7211
19	4	2018	6939
20	5	2018	6873
21	6	2018	6167
22	7	2018	6292

The first query provides the count of orders on a yearly basis and the second query provides the count of orders on a monthly basis and from the output of these queries, we can see a growing trend in the number of orders.

2. Time analysis of orders by Brazilian customers

```
SELECT COUNT(*) AS count_of_Orders,

CASE
WHEN TIME(order_purchase_timestamp) >= '04:00:00' AND TIME(order_purchase
_timestamp) < '06:00:00' THEN 'Dawn'
WHEN TIME(order_purchase_timestamp) >= '06:00:00' AND TIME(order_purchase
_timestamp) < '12:00:00' THEN 'Morning'
WHEN TIME(order_purchase_timestamp) >= '12:00:00' AND TIME(order_purchase
_timestamp) < '18:00:00' THEN 'Afternoon'
ELSE 'Night'
END AS time_period
FROM `dsml-target-project.target.orders` GROUP BY time_period;
```

Row	count_of_Orders	time_period
1	22240	Morning
2	38446	Night
3	38361	Afternoon
4	394	Dawn

4 AM – 6 AM considered as dawn, 6 AM – 12 considered as morning, 12 – 6 PM considered as afternoon, 6 PM – 3:59 AM considered as night.

From the results, it can be inferred that customers tend to buy mostly during the night followed by afternoon period.

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

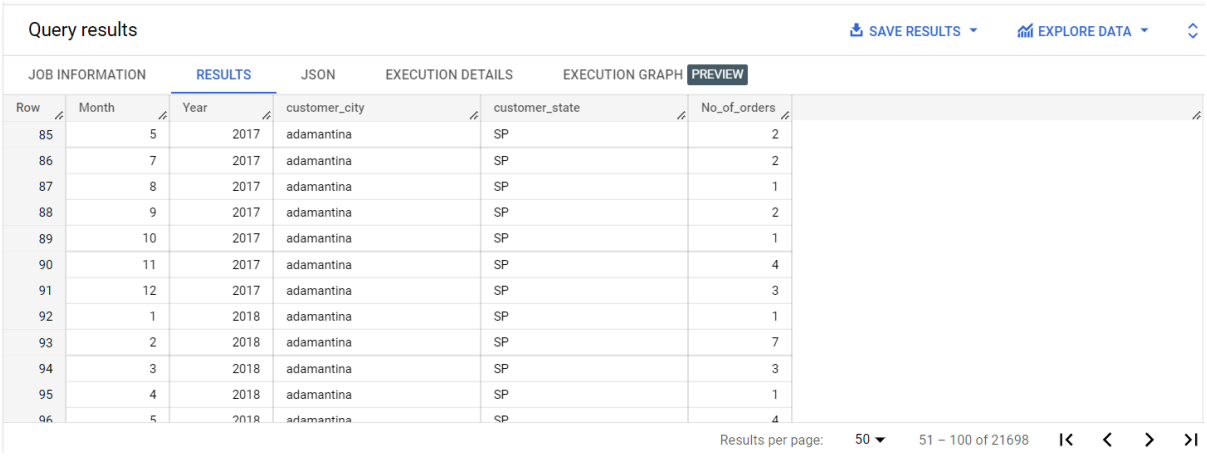
1. Get month on month orders by region, states
2. How are customers distributed in Brazil

1. Month on month orders by region and state.

Based on city (month over month)

```
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Month,EXTRACT(YEAR F
ROM o.order_purchase_timestamp) AS Year,

c.customer_city, c.customer_state, COUNT(*) AS No_of_orders
FROM `dsml-target-project.target.orders` AS o JOIN `dsml-target-
project.target.customers` AS c ON o.customer_id = c.customer_id
GROUP BY Month,Year,c.customer_city,c.customer_state ORDER BY c.customer_city
,c.customer_state,Year,Month;
```



Query results

SAVE RESULTS | EXPLORE DATA

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

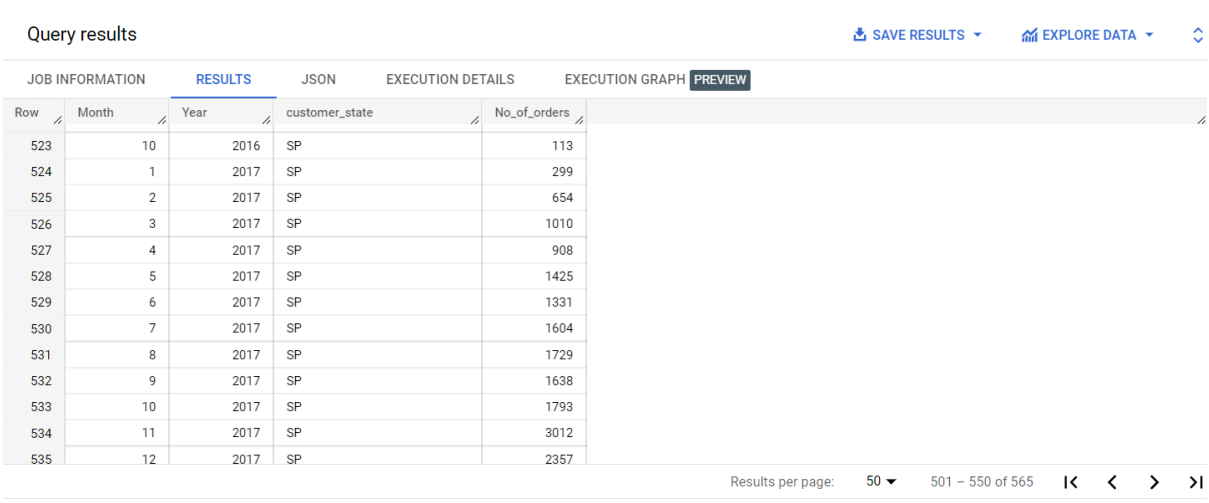
Row	Month	Year	customer_city	customer_state	No_of_orders
85	5	2017	adamantina	SP	2
86	7	2017	adamantina	SP	2
87	8	2017	adamantina	SP	1
88	9	2017	adamantina	SP	2
89	10	2017	adamantina	SP	1
90	11	2017	adamantina	SP	4
91	12	2017	adamantina	SP	3
92	1	2018	adamantina	SP	1
93	2	2018	adamantina	SP	7
94	3	2018	adamantina	SP	3
95	4	2018	adamantina	SP	1
96	5	2018	adamantina	SP	4

Results per page: 50 51 – 100 of 21698

Based on State (month over month)

```
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Month,EXTRACT(YEAR FRO
M o.order_purchase_timestamp) AS Year, c.customer_state, COUNT(*) AS No_of_orde
rs

FROM `dsml-target-project.target.orders` AS o JOIN `dsml-target-
project.target.customers` AS c ON o.customer_id = c.customer_id
GROUP BY Month,Year,c.customer_state ORDER BY c.customer_state,Year,Month;
```



Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Month	Year	customer_state	No_of_orders
523	10	2016	SP	113
524	1	2017	SP	299
525	2	2017	SP	654
526	3	2017	SP	1010
527	4	2017	SP	908
528	5	2017	SP	1425
529	6	2017	SP	1331
530	7	2017	SP	1604
531	8	2017	SP	1729
532	9	2017	SP	1638
533	10	2017	SP	1793
534	11	2017	SP	3012
535	12	2017	SP	2357

Results per page: 50 501 – 550 of 565

2. Customer Distribution

```
SELECT *,t1.count_of_customers * 100 / SUM(t1.count_of_customers) OVER (ORD
ER BY t1.count_of_customers RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED
 FOLLOWING) AS Percent_customer FROM

(
  SELECT t.customer_state,COUNT(*) AS count_of_customers FROM
  (
    SELECT DISTINCT c.customer_id, c.customer_state
    FROM `dsml-target-project.target.orders` AS o JOIN `dsml-target-
project.target.customers` AS c ON o.customer_id = c.customer_id
  ) AS t
  GROUP BY t.customer_state
) AS t1 ORDER BY t1.count of customers DESC;
```

Row	customer_state	count_of_customers	Percent_customer
1	SP	41746	41.980671956235355
2	RJ	12852	12.924246538148248
3	MG	11635	11.700405265433774
4	RS	5466	5.496726702265665
5	PR	5045	5.0733600828632053
6	SC	3637	3.6574451182107985
7	BA	3380	3.3990004123047837
8	DF	2140	2.152029846843857
9	ES	2033	2.0444283545016644
10	GO	2020	2.0313552759927997
11	PE	1652	1.661286592049557
12	CE	1336	1.3435102221417725
13	PA	975	0.98048088816484147

Results per page: 50 1 – 27 of 27

Around 42% of customers are from the state SP and the top five states have around 77% of customers.

4. **Impact on Economy: Analyze the money movemented 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)**
    2. **Mean & Sum of price and freight value by customer state**

1. Percentage increase in cost of orders from 2017 to 2018.

```
SELECT t2.MONTH, t2.Total_Cost AS Year_2018, t2.Prev_year AS Year_2017, (t2.Tota
l_Cost - t2.Prev_year) * 100 / t2.Total_Cost AS Percent_increase FROM

(
  SELECT *,lag(t1.Total_Cost) OVER(PARTITION BY t1.MONTH ORDER BY t1.year) AS Pr
ev_year FROM
  (
    SELECT *,t.Price + t.Frieght_Charge AS Total_Cost FROM
    (
      SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
      EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,SUM(oi.price) AS Pri
ce,SUM(oi.freight_value) AS Frieght_Charge FROM
      `dsml-target-project.target.orders` AS o JOIN `dsml-target-
project.target.order_items` AS oi ON o.order_id = oi.order_id
      WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) >= 2017 AND EXTRACT(MO
NTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
      GROUP BY month,year
    ) AS t
  ) AS t1
) AS t2 WHERE t2.year = 2018 ORDER BY t2.month;
```

Row	MONTH	Year_2018	Year_2017	Percent_increase
1	1	1107301.89...	137188.489...	87.610561199349632
2	2	986908.960...	286280.619...	70.992195673247835
3	3	1155126.82...	432048.590...	62.597302519563357
4	4	1159698.04...	412422.24	64.437101230248814
5	5	1149781.82...	586190.950...	49.017201367821755
6	6	1022677.11...	502963.040...	50.818979413747364
7	7	1058728.03...	584971.620...	44.7476969132481
8	8	1003308.47...	668204.600...	33.3998844841808

Percentage of increase in cost on an average is around 55-65% from 2017 to 2018 (Jan - Aug)

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

```
SELECT c.customer_state, SUM(oi.price) AS Price_sum, AVG(oi.price) AS Mean_Price,


SUM(oi.freight_value ) AS freight_value_sum, AVG(oi.freight_value ) AS Mean_freig
ht_value FROM
`dsml-target-project.target.orders` AS o JOIN `dsml-target-
project.target.customers` AS c ON o.customer_id = c.customer_id
JOIN `dsml-target-
project.target.order_items` AS oi ON o.order_id = oi.order_id GROUP BY c.customer
_state ORDER BY Price_sum DESC;
```

Row	customer_state	Price_sum	Mean_Price	freight_value_sum	Mean_freight_value
1	SP	5202955.0500014517	109.653629...	718723.0699999833	15.147275390419248
2	RJ	1824092.6699998139	125.117818...	305589.31000000035	20.96092393168248
3	MG	1585308.0299998785	120.748574...	270853.46000000357	20.630166806306541
4	RS	750304.02000002388	120.337453...	135522.74000000212	21.735804330392945
5	PR	683083.76000002027	119.004139...	117851.68000000139	20.531651567944248
6	SC	520553.340000008	124.653577...	89660.260000000431	21.470368773946436
7	BA	511349.99000000674	134.601208...	100156.67999999883	26.363958936562248
8	DF	302603.93999999797	125.770548...	50625.499999999811	21.041354945968383
9	GO	294591.94999999728	126.271731...	53114.979999999865	22.766815259322794
10	ES	275037.30999999633	121.913701...	49764.599999999889	22.058776595744682
11	PE	262788.02999999659	145.508322...	59449.6599999999	32.917862679955796
12	CE	227254.70999999763	153.758261...	48351.589999999924	32.714201623815995
13	PA	178947.80999999869	165.692416...	38699.300000000039	35.832685185185177

Results per page: 50 1 – 27 of 27

5. **Analysis on sales, freight and delivery time**
    1. **Calculate days between purchasing, delivering and estimated delivery**
    2. **Create columns:**
        - **time_to_delivery = order_purchase_timestamp-order_delivered_customer_date**
        - **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**
    3. **Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery**
    4. **Sort the data to get the following:**
        - **Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5**
        - **Top 5 states with highest/lowest average time to delivery**
        - **Top 5 states where delivery is really fast/ not so fast compared to estimated date**

```
SELECT t.customer_state, AVG(t.price) AS mean_price, AVG(t.freight_value) AS mea
n_freight_value,

AVG(t.time_to_delivery) AS mean_time_to_delivery,AVG(t.diff_estimated_delivery)
AS mean_diff_estimated_delivery FROM
(
  SELECT c.customer_state,oi.price,oi.freight_value,
  DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp, DAY) AS
time_to_delivery,
  DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date, DAY
) AS diff_estimated_delivery FROM
  `dsml-target-project.target.orders` AS o JOIN `dsml-target-
project.target.order_items` AS oi ON o.order_id = oi.order_id
  JOIN `dsml-target-
project.target.customers` AS c ON o.customer_id = c.customer_id
) AS t GROUP BY t.customer_state ORDER BY mean_freight_value DESC LIMIT 5;
```

4.1.1. Top 5 states with highest average freight value

Row	customer_state	mean_price	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	RR	150.56596153846155	42.984423076923093	27.826086956521738	17.434782608695652
2	PB	191.475215946844	42.723803986710941	20.119453924914676	12.15017064846416
3	RO	165.97352517985615	41.069712230215842	19.282051282051292	19.080586080586084
4	AC	173.72771739130434	40.073369565217405	20.329670329670336	20.010989010989018
5	PI	160.35808118081187	39.147970479704767	18.931166347992352	10.682600382409184

4.1.2. Top 5 states with lowest average freight value

Row	customer_state	mean_price	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	SP	109.65362915972931	15.147275390419248	8.25960855241909	10.26559438451439
2	PR	119.00413937282212	20.531651567944248	11.480793060718735	12.533899805275263
3	MG	120.74857414883068	20.630166806306541	11.515522180072811	12.397151041263502
4	RJ	125.11781809451955	20.96092393168248	14.689382157500321	11.14449314293797
5	DF	125.77054862842893	21.041354945968383	12.501486199575384	11.274734607218704

4.2.1. Top 5 states with highest time to delivery

Row	customer_state	mean_price	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	RR	150.56596153846155	42.984423076923093	27.826086956521738	17.434782608695652
2	AP	164.32073170731707	34.006097560975618	27.753086419753075	17.444444444444443
3	AM	135.49599999999995	33.205393939393936	25.963190184049076	18.975460122699381
4	AL	180.88921171171171	35.843671171171152	23.992974238875881	7.9765807962529349
5	PA	165.69241666666659	35.832685185185177	23.301707779886126	13.37476280834913

4.2.2. Top 5 states with lowest time to delivery

Row	customer_state	mean_price	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	SP	109.65362915972931	15.147275390419248	8.25960855241909	10.26559438451439
2	PR	119.00413937282212	20.531651567944248	11.480793060718735	12.533899805275263
3	MG	120.74857414883068	20.630166806306541	11.515522180072811	12.397151041263502
4	DF	125.77054862842893	21.041354945968383	12.501486199575384	11.274734607218704
5	SC	124.65357758620688	21.470368773946436	14.520985846754517	10.6688628599317

4.3.1. Top 5 states where delivery is faster compared to the estimated date

Row	customer_state	mean_price	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	AC	173.72771739130434	40.073369565217405	20.329670329670336	20.010989010989018
2	RO	165.97352517985615	41.069712230215842	19.282051282051292	19.080586080586084
3	AM	135.49599999999995	33.205393939393936	25.963190184049076	18.975460122699381
4	AP	164.32073170731707	34.006097560975618	27.753086419753075	17.444444444444443
5	RR	150.56596153846155	42.984423076923093	27.826086956521738	17.434782608695652

4.3.2. Top 5 states where delivery is not so fast as compared to the estimated date

Row	customer_state	mean_price	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	AL	180.88921171171171	35.843671171171152	23.992974238875881	7.9765807962529349
2	MA	145.20415048543691	38.25700242718446	21.203750000000017	9.1099999999999923
3	SE	153.04116883116873	36.653168831168855	20.978666666666651	9.1653333333333276
4	ES	121.91370124113466	22.058776595744682	15.192808988764023	9.7685393258427116
5	BA	134.6012082126874	26.363958936562248	18.774640238935675	10.119467825142538

6. Payment type analysis:

1. **Month over Month count of orders for different payment types**
2. **Distribution of payment installments and count of orders**

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

```
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,EXTRACT(YEAR FRO
M o.order_purchase_timestamp) AS year,COUNT(DISTINCT o.order_id) AS count_of_or
ders, p.payment_type FROM `dsml-target-project.target.orders` AS o JOIN `dsml-
target-project.target.payments` AS p ON o.order_id = p.order_id
GROUP BY p.payment_type,month,year ORDER BY p.payment_type,year,month;
```

Row	month	year	count_of_orders	payment_type
1	10	2016	63	UPI
2	1	2017	197	UPI
3	2	2017	398	UPI
4	3	2017	590	UPI
5	4	2017	496	UPI
6	5	2017	772	UPI
7	6	2017	707	UPI
8	7	2017	845	UPI
9	8	2017	938	UPI
10	9	2017	903	UPI
11	10	2017	993	UPI
12	11	2017	1509	UPI
13	12	2017	1160	UPI

Row	month	year	count_of_orders	payment_type
25	1	2017	582	credit_card
26	2	2017	1347	credit_card
27	3	2017	2008	credit_card
28	4	2017	1835	credit_card
29	5	2017	2833	credit_card
30	6	2017	2452	credit_card
31	7	2017	3072	credit_card
32	8	2017	3272	credit_card
33	9	2017	3274	credit_card
34	10	2017	3510	credit_card
35	11	2017	5867	credit_card
36	12	2017	4363	credit_card
37	1	2018	5511	credit_card

2. Distribution of payment installments and count of orders

```
SELECT p.payment_installments, COUNT(DISTINCT o.order_id) AS count_of_orders
FROM `dsml-target-project.target.orders` AS o JOIN `dsml-target-
project.target.payments` AS p ON o.order_id = p.order_id
GROUP BY p.payment_installments ORDER BY p.payment_installments;
```

Row	payment_installments	count_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
11	10	5315
12	11	23
13	12	133