

## BUSINESS CASE STUDY - TARGET SQL

### Q1. Initial exploration of dataset like checking the characteristics of data

#### ---1) Data type of columns in a table

Query:

```
SELECT column_name, data_type
FROM `dsm1-sql-16-2-23.Target_SQL`.
INFORMATION_SCHEMA.COLUMNS
```

RUN SAVE SHARE SCHEDULE MORE				
1	Q1.1			
2	SELECT column_name, data_type			
3	FROM `dsm1-sql-16-2-23.Target_SQL`.			
4	INFORMATION_SCHEMA.COLUMNS			
5				

Query results		SAVE RESULTS
JOB INFORMATION		RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW
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	

#### ---2) Time period for which the data is given

Query:

```
SELECT
  MIN(order_purchase_timestamp) AS start_date,
  MAX(order_estimated_delivery_date) AS end_date,
  DATE_DIFF(MAX(order_estimated_delivery_date),MIN(order_purchase_timestamp),day) as Total_time_period_for_given_data
FROM `Target_SQL.orders`
```

RUN SAVE SHARE SCHEDULE MORE				
6	---			
7	SELECT			
8	MIN(order_purchase_timestamp) AS start_date,			
9	MAX(order_estimated_delivery_date) AS end_date,			
10	DATE_DIFF(MAX(order_estimated_delivery_date),MIN(order_purchase_timestamp),day) as Total_time_period_for_given_data			
11	FROM `Target_SQL.orders`			
12				

Query results		SAVE RESULTS	EXPLORE DATA
JOB INFORMATION		RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW	
Row	start_date end_date Total_time_perio		
1	2016-09-04 21:15:19 UTC 2018-11-12 00:00:00 UTC 798		

### --3) Cities and States of customers ordered during the given period

Query:

```
SELECT
  o.order_id,
  c.customer_city,
  c.customer_state
FROM `Target_SQL.orders` as o
LEFT JOIN `Target_SQL.customers` as c
ON o.customer_id = c.customer_id
```

```
12
13 ----3
14 SELECT
15   o.order_id,
16   c.customer_city,
17   c.customer_state
18 FROM `Target_SQL.orders` as o
19 LEFT JOIN `Target_SQL.customers` as c
20 ON o.customer_id = c.customer_id
21
22 ----Q2.
```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	order_id	customer_city	customer_state			
1	6190a94657e1012983a274b8...	maceio	AL			

## Q2. 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?

QUERY :

```
SELECT
  x.active_years as year,
  sum(CASE WHEN x.active_months = 1 then 1 else 0 END) AS JAN,
  sum(CASE WHEN x.active_months = 2 then 1 else 0 END) AS FEB,
  sum(CASE WHEN x.active_months = 3 then 1 else 0 END) AS MAR,
  sum(CASE WHEN x.active_months = 4 then 1 else 0 END) AS APR,
  sum(CASE WHEN x.active_months = 5 then 1 else 0 END) AS MAY,
  sum(CASE WHEN x.active_months = 6 then 1 else 0 END) AS JUN,
  sum(CASE WHEN x.active_months = 7 then 1 else 0 END) AS JUL,
  sum(CASE WHEN x.active_months = 8 then 1 else 0 END) AS AUG,
  sum(CASE WHEN x.active_months = 9 then 1 else 0 END) AS SEPT,
  sum(CASE WHEN x.active_months = 10 then 1 else 0 END) AS OCT,
  sum(CASE WHEN x.active_months = 11 then 1 else 0 END) AS NOV,
  sum(CASE WHEN x.active_months = 12 then 1 else 0 END) AS DEC,
  COUNT(DISTINCT x.order_id) as no_of_orders
FROM
  (SELECT
    order_id,
    order_purchase_timestamp,
    EXTRACT(YEAR from order_purchase_timestamp) AS active_years,
    EXTRACT(MONTH from order_purchase_timestamp) AS active_months
    FROM `Target_SQL.orders`
    ORDER BY order_purchase_timestamp) as x
GROUP BY year
order by year
```

orders

\*Unsaved query 2

\*Unsaved query 3

geolocation

SQL Queries 2023-03-09

▶ RUN

🔒 SAVE

👤 SHARE

🕒 SCHEDULE

⚙️ MORE

23 a)

24 SELECT

25 x.active\_years as year,

26 sum(CASE WHEN x.active\_months = 1 then 1 else 0 END) AS JAN,

27 sum(CASE WHEN x.active\_months = 2 then 1 else 0 END) AS FEB,

28 sum(CASE WHEN x.active\_months = 3 then 1 else 0 END) AS MAR,

29 sum(CASE WHEN x.active\_months = 4 then 1 else 0 END) AS APR,

30 sum(CASE WHEN x.active\_months = 5 then 1 else 0 END) AS MAY,

31 sum(CASE WHEN x.active\_months = 6 then 1 else 0 END) AS JUN,

32 sum(CASE WHEN x.active\_months = 7 then 1 else 0 END) AS JUL,

33 sum(CASE WHEN x.active\_months = 8 then 1 else 0 END) AS AUG,

34 sum(CASE WHEN x.active\_months = 9 then 1 else 0 END) AS SEPT,

Press Alt+F1 for Accessibility Options.

Query results

📌 SAVE RESULTS

📊 EXPLORE DATA

↕

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	MAY	JUN	JUL	AUG	SEPT	OCT	NOV	DEC	no_of_orders	
1	0	0	0	0	4	324	0	1	329	
2	2404	3700	3245	4026	4331	4285	4631	7544	5673	45101
3	6939	6873	6167	6292	6512	16	4	0	0	54011

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

### QUERY

```

Select
SUM(CASE WHEN x.hours between 0 and 6 THEN 1 ELSE 0 END) AS DAWN,
SUM(CASE WHEN x.hours between 7 and 12 THEN 1 ELSE 0 END) AS MORNING,
SUM(CASE WHEN x.hours between 13 and 18 THEN 1 ELSE 0 END) AS AFTERNOON,
SUM(CASE WHEN x.hours between 19 and 23 THEN 1 ELSE 0 END) AS NIGHT,
COUNT (*) as Total_orders
FROM (select
    order_purchase_timestamp,
    EXTRACT(HOUR from order_purchase_timestamp) as hours
FROM `Target_SQL.orders`) as x

```

RUN SAVE SHARE SCHEDULE MORE

```

50 ---B |
51 Select
52     SUM(CASE WHEN x.hours between 0 and 6 THEN 1 ELSE 0 END) AS DAWN,
53     SUM(CASE WHEN x.hours between 7 and 12 THEN 1 ELSE 0 END) AS MORNING,
54     SUM(CASE WHEN x.hours between 13 and 18 THEN 1 ELSE 0 END) AS AFTERNOON,
55     SUM(CASE WHEN x.hours between 19 and 23 THEN 1 ELSE 0 END) AS NIGHT,
56     COUNT (*) as Total_orders
57     FROM (select
58         order_purchase_timestamp,
59         EXTRACT(HOUR from order_purchase_timestamp) as hours
60     FROM `Target_SQL.orders`) as x
61
62

```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row		DAWN	MORNING	AFTERNOON	NIGHT	Total_orders
1		5242	27733	38135	28331	99441

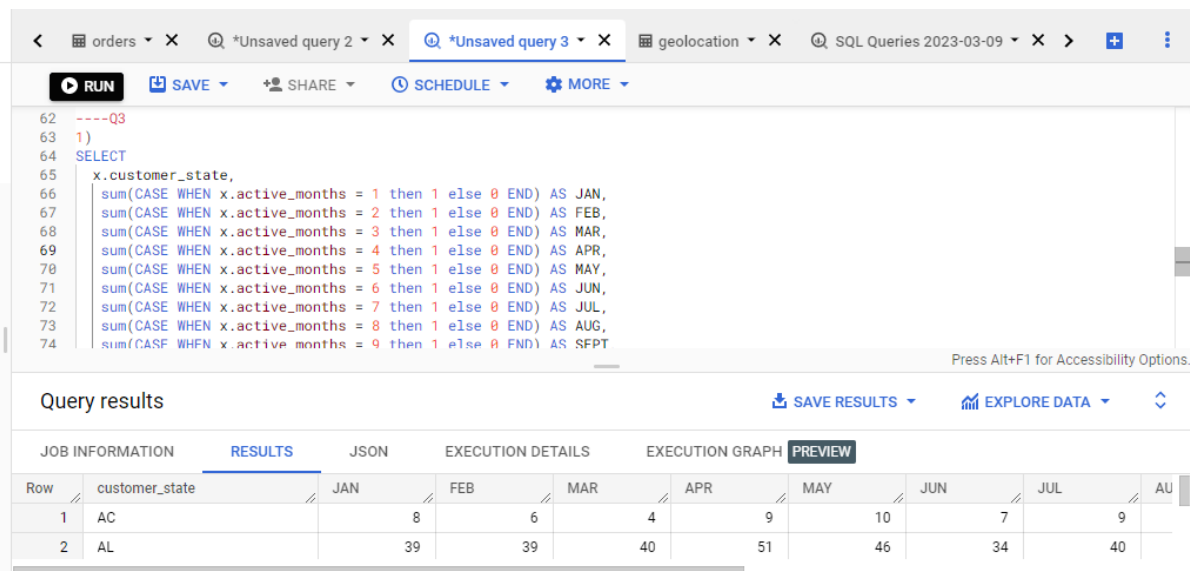
**Inference** –Brazilian people preferred to purchase more in **Afternoon**

### Q3. Evolution of E-commerce orders in the Brazil region

#### 1) Get month on month orders by states

QUERY:

```
SELECT
  x.customer_state,
  sum(CASE WHEN x.active_months = 1 then 1 else 0 END) AS JAN,
  sum(CASE WHEN x.active_months = 2 then 1 else 0 END) AS FEB,
  sum(CASE WHEN x.active_months = 3 then 1 else 0 END) AS MAR,
  sum(CASE WHEN x.active_months = 4 then 1 else 0 END) AS APR,
  sum(CASE WHEN x.active_months = 5 then 1 else 0 END) AS MAY,
  sum(CASE WHEN x.active_months = 6 then 1 else 0 END) AS JUN,
  sum(CASE WHEN x.active_months = 7 then 1 else 0 END) AS JUL,
  sum(CASE WHEN x.active_months = 8 then 1 else 0 END) AS AUG,
  sum(CASE WHEN x.active_months = 9 then 1 else 0 END) AS SEPT,
  sum(CASE WHEN x.active_months = 10 then 1 else 0 END) AS OCT,
  sum(CASE WHEN x.active_months = 11 then 1 else 0 END) AS NOV,
  sum(CASE WHEN x.active_months = 12 then 1 else 0 END) AS DEC,
  COUNT(DISTINCT x.order_id) as no_of_orders
FROM
  (SELECT
    o.order_id,
    c.customer_state,
    o.order_purchase_timestamp,
    EXTRACT(YEAR from order_purchase_timestamp) AS active_years,
    EXTRACT(MONTH from order_purchase_timestamp) AS active_months
  FROM `Target_SQL.orders` as o
  LEFT JOIN `Target_SQL.customers` as c
  ON o.customer_id = c.customer_id
  ORDER BY order_purchase_timestamp) as x
GROUP BY x.customer_state
order by x.customer_state
```



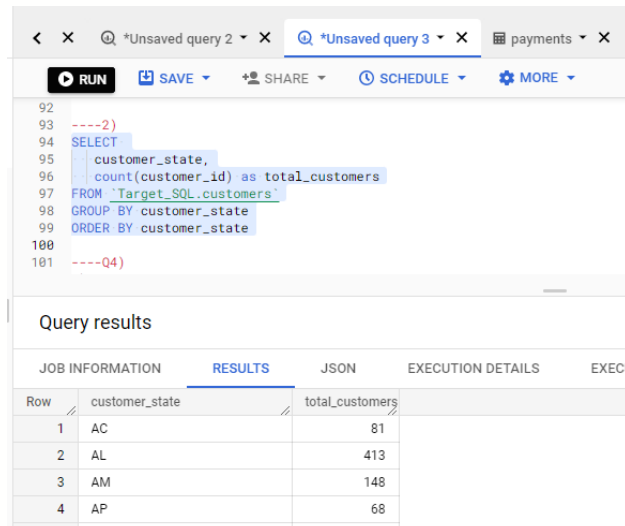
The screenshot shows a SQL query editor with a toolbar at the top containing buttons for RUN, SAVE, SHARE, SCHEDULE, and MORE. The query is displayed in the editor area, and the results are shown in a table below. The table has columns for customer\_state, JAN, FEB, MAR, APR, MAY, JUN, JUL, and AUG. The results are grouped by customer\_state, with AC and AL being the only states shown. The data shows the number of orders for each month for each state.

Row	customer_state	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
1	AC	8	6	4	9	10	7	9	
2	AL	39	39	40	51	46	34	40	

## 2) Distribution of customers across the states in Brazil

QUERY:

```
SELECT
    customer_state,
    count(customer_id) as total_customers
FROM `Target_SQL.customers`
GROUP BY customer_state
ORDER BY customer_state
```



The screenshot shows a SQL query editor with a toolbar at the top containing buttons for RUN, SAVE, SHARE, SCHEDULE, and MORE. The query is as follows:

```
92
93 ----2)
94 SELECT
95     customer_state,
96     count(customer_id) as total_customers
97 FROM `Target_SQL.customers`
98 GROUP BY customer_state
99 ORDER BY customer_state
100
101 ----Q4)
```

Below the query editor, the 'Query results' section is displayed with a tabbed interface showing 'JOB INFORMATION', 'RESULTS', 'JSON', 'EXECUTION DETAILS', and 'EXECU'. The 'RESULTS' tab is active, showing a table with 4 rows and 2 columns: 'customer\_state' and 'total\_customers'.

Row	customer_state	total_customers
1	AC	81
2	AL	413
3	AM	148
4	AP	68

**Q4. 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)**

QUERY:

```
WITH total_amount as
(SELECT
    x.year as year,
    SUM(CASE WHEN x.year= 2017 THEN x.payment_value END) as total_amount_2017,
    SUM(CASE WHEN x.year= 2018 THEN x.payment_value END) as total_amount_2018
FROM
    (SELECT
        p.payment_value,
        o.order_purchase_timestamp,
        EXTRACT(YEAR from order_purchase_timestamp) as year,
        EXTRACT(MONTH FROM order_purchase_timestamp) as month
    FROM `Target_SQL.payments` as p
    LEFT JOIN `Target_SQL.orders` as o
    on p.order_id = o.order_id
    WHERE EXTRACT(YEAR from order_purchase_timestamp) IN(2017,2018) AND EXTRACT
    (MONTH FROM order_purchase_timestamp) between 1 and 8 ) as x
    GROUP BY x.year)
SELECT (MAX(total_amount_2018)-MAX(total_amount_2017))/
    MAX(total_amount_2017)*100 as rise_in_cost
FROM total_amount
```

Q \*BUSINESS\_CASE\_QUERYE... 18 X Q \*Unsaved query 3 X geolocation X Q \*Unsaved query X payment > +

RUN SAVE SHARE SCHEDULE MORE

```

104 WITH total_amount as
105 (SELECT
106   x.year as year,
107   SUM(CASE WHEN x.year= 2017 THEN x.payment_value END) as total_amount_2017,
108   SUM(CASE WHEN x.year= 2018 THEN x.payment_value END) as total_amount_2018
109 FROM
110 (SELECT
111   p.payment_value,
112   o.order_purchase_timestamp,
113   EXTRACT(YEAR from order_purchase_timestamp) as year,
114   EXTRACT(MONTH FROM order_purchase_timestamp) as month
115 FROM `Target_SQL.payments` as p
116 LEFT JOIN `Target_SQL.orders` as o

```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	rise_in_cost
1	136.976871...

## ---2) Mean & Sum of price and freight value by customer state

QUERY:

```

SELECT
  c.customer_state,
  ROUND(SUM(price)/COUNT(price),2) as mean_price,
  ROUND(SUM(price),2) as total_price,
  ROUND(SUM(freight_value)/COUNT(freight_value),2) as mean_freight_value,
  ROUND(SUM(freight_value),2) as total_freight_price
from `Target_SQL.order_items` as oi
LEFT JOIN `Target_SQL.orders` as o
ON oi.order_id = o.order_id
LEFT JOIN `Target_SQL.customers` as c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state

```

RUN SAVE SHARE SCHEDULE MORE

```

1 SELECT
2   c.customer_state,
3   ROUND(SUM(price)/COUNT(price),2) as mean_price,
4   ROUND(SUM(price),2) as total_price,
5   ROUND(SUM(freight_value)/COUNT(freight_value),2) as mean_freight_value,
6   ROUND(SUM(freight_value),2) as total_freight_price
7 from `Target_SQL.order_items` as oi
8 LEFT JOIN `Target_SQL.orders` as o
9 ON oi.order_id = o.order_id
10 LEFT JOIN `Target_SQL.customers` as c
11 ON o.customer_id = c.customer_id
12 GROUP BY c.customer_state
13 ORDER BY c.customer_state

```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

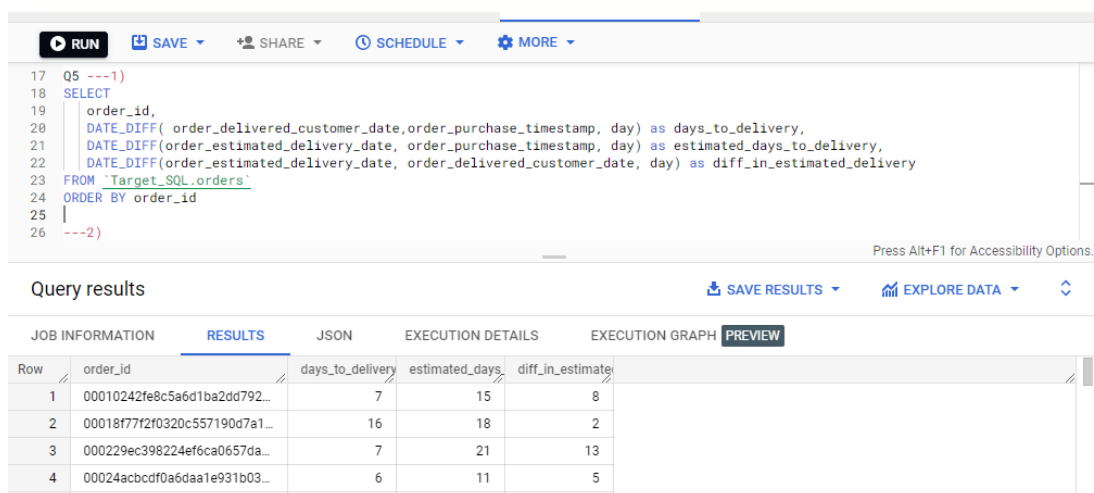
Row	customer_state	mean_price	total_price	mean_freight_va	total_freight_pric
1	AC	173.73	15982.95	40.07	3686.75
2	AL	180.89	80314.81	35.84	15914.59

## 5. Analysis on sales, freight and delivery time

### --1). Calculate days between purchasing, delivering and estimated delivery

QUERY

```
SELECT
    order_id,
    DATE_DIFF( order_delivered_customer_date,order_purchase_timestamp, day)
as days_to_delivery,
    DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day)
as estimated_days_to_delivery,
    DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
day) as diff_in_estimated_delivery
FROM `Target_SQL.orders`
ORDER BY order_id
```



The screenshot shows a SQL query editor with a query that calculates delivery metrics. Below the editor, the 'Query results' section displays a table with 4 rows of data. The table has columns for 'order\_id', 'days\_to\_delivery', 'estimated\_days', and 'diff\_in\_estimated\_delivery'. The 'order\_id' column is truncated in the first three rows. The 'diff\_in\_estimated\_delivery' column is also truncated in the first three rows.

Row	order_id	days_to_delivery	estimated_days	diff_in_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	15	8
2	00018f77f2f0320c557190d7a1...	16	18	2
3	000229ec398224ef6ca0657da...	7	21	13
4	00024acbcd0a6daa1e931b03...	6	11	5

### --2) Find time\_to\_delivery & diff\_estimated\_delivery. Formula for the same given below:

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

QUERY:

```
SELECT
    order_id,
    DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date, day)
as time_to_delivery,
    DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
day) as diff_estimated_delivery
FROM `Target_SQL.orders`
ORDER BY order_id
```

Query editor interface showing a SQL query and its results.

```

26 ---2)
27 SELECT
28     order_id,
29     DATE_DIFF(order_purchase_timestamp, order_delivered_customer_date, day) as time_to_delivery,
30     DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
31 FROM `Target_SQL.orders`
32 ORDER BY order_id
33
34

```

Query results table:

Row	order_id	time_to_delivery	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	-7	8
2	00018f77f2f0320c557190d7a1...	-16	2
3	000229ec398224ef6ca0657da...	-7	13
4	00024acbcd0a6daa1e931b03...	-6	5
5	00042b26cf59d7ce69dfabb4e...	-25	15

### --3) Group data by state, take mean of freight\_value, time\_to\_delivery, diff\_estimated\_delivery

QUERY:

```

SELECT
    g.geolocation_state,
    ROUND(SUM(oi.freight_value)/COUNT(oi.freight_value),2) as mean_freight_value,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) as time_to_delivery,
    AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)) as diff_estimated_delivery
FROM `Target_SQL.orders` as o
LEFT JOIN `Target_SQL.order_items` as oi
on o.order_id = oi.order_id
LEFT JOIN `Target_SQL.customers` as c
on o.customer_id = c.customer_id
JOIN `Target_SQL.geolocation` as g
ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY g.geolocation_state
ORDER BY g.geolocation_state

```

Query editor interface showing a SQL query and its results.

```

34 ---3)
35 SELECT
36     g.geolocation_state,
37     ROUND(SUM(oi.freight_value)/COUNT(oi.freight_value),2) as mean_freight_value,
38     AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day)) as time_to_delivery,
39     AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)) as diff_estimated_delivery
40 FROM `Target_SQL.orders` as o
41 LEFT JOIN `Target_SQL.order_items` as oi
42 on o.order_id = oi.order_id
43 LEFT JOIN `Target_SQL.customers` as c
44 on o.customer_id = c.customer_id
45 JOIN `Target_SQL.geolocation` as g
46 ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix

```

Query results table:

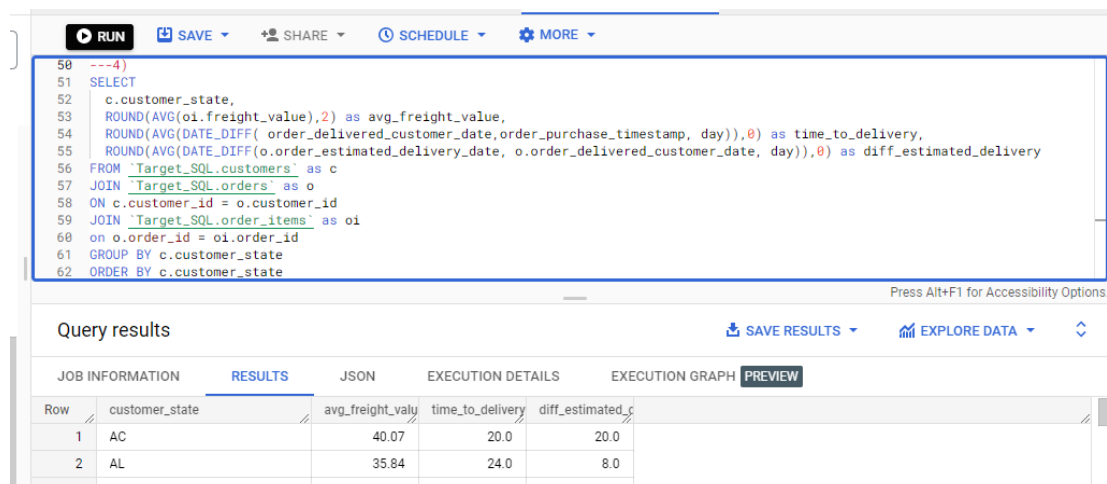
Row	geolocation_state	mean_freight_value	time_to_delivery	diff_estimated_delivery
1	AC	39.1	20.1011263...	18.5637855...
2	AL	33.83	22.8700692...	8.45695049...



#### --- 4) Sort the data to get the following:

QUERY:

```
SELECT
    c.customer_state,
    ROUND(AVG(oi.freight_value),2) as avg_freight_value,
    ROUND(AVG(DATE_DIFF( order_delivered_customer_date,order_purchase_timestamp, day
)),0) as time_to_delivery,
    ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_
date, day)),0) as diff_estimated_delivery
FROM `Target_SQL.customers` as c
JOIN `Target_SQL.orders` as o
ON c.customer_id = o.customer_id
JOIN `Target_SQL.order_items` as oi
on o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state
```



The screenshot shows a SQL query editor with a toolbar (RUN, SAVE, SHARE, SCHEDULE, MORE) and a query window containing the same SQL code as above. Below the editor is a 'Query results' section with a table showing the output. The table has columns for 'customer\_state', 'avg\_freight\_valu', 'time\_to\_delivery', and 'diff\_estimated\_c'. Two rows are visible: one for 'AC' with values 40.07, 20.0, and 20.0, and another for 'AL' with values 35.84, 24.0, and 8.0.

Row	customer_state	avg_freight_valu	time_to_delivery	diff_estimated_c
1	AC	40.07	20.0	20.0
2	AL	35.84	24.0	8.0

#### --- (5) - Top 5 states with highest/lowest average freight value - sort in desc/asc limit

QUERY:

```
(SELECT
    c.customer_state,
    ROUND(AVG(oi.freight_value),2) as mean_freight_value
FROM `Target_SQL.customers` as c
JOIN `Target_SQL.orders` as o
ON c.customer_id = o.customer_id
JOIN `Target_SQL.order_items` as oi
on o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY ROUND(AVG(oi.freight_value),2) DESC
LIMIT 5)
UNION ALL
(SELECT
    c.customer_state,
    ROUND(AVG(oi.freight_value),2) as mean_freight_value
FROM `Target_SQL.customers` as c
JOIN `Target_SQL.orders` as o
ON c.customer_id = o.customer_id
```

```

JOIN `Target_SQL.order_items` as oi
on o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY ROUND(AVG(oi.freight_value),2) ASC
LIMIT 5)

```

The screenshot shows a SQL query editor with a toolbar at the top containing buttons for RUN, SAVE, SHARE, SCHEDULE, and MORE. The query text is as follows:

```

65 (SELECT
66   c.customer_state,
67   ROUND(AVG(oi.freight_value),2) as mean_freight_value
68 FROM `Target_SQL.customers` as c
69 JOIN `Target_SQL.orders` as o
70 ON c.customer_id = o.customer_id
71 JOIN `Target_SQL.order_items` as oi
72 on o.order_id = oi.order_id
73 GROUP BY c.customer_state
74 ORDER BY ROUND(AVG(oi.freight_value),2) DESC
75 LIMIT 5)
76 UNION ALL
77 DISJUNCT

```

### ---6) Top 5 states with highest/lowest average time to delivery

QUERY

```

(SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF( order_delivered_customer_date,order_purchase_timesta
mp, day )),0) as time_to_delivery
FROM `Target_SQL.customers` as c
JOIN `Target_SQL.orders` as o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY ROUND(AVG(DATE_DIFF( order_delivered_customer_date,order_purchase_
timestamp, day)),0) DESC
LIMIT 5)
UNION ALL
(SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF( order_delivered_customer_date,order_purchase_timesta
mp, day)),0) as time_to_delivery
FROM `Target_SQL.customers` as c
JOIN `Target_SQL.orders` as o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY ROUND(AVG(DATE_DIFF( order_delivered_customer_date,order_purchase_
timestamp, day)),0) asc
LIMIT 5)

```

```

89  ----6)|
90  (SELECT
91  | c.customer_state,
92  | ROUND(AVG(DATE_DIFF( order_delivered_customer_date,order_purchase_timestamp, day)),0) as time_to_delivery
93  FROM `Target_SQL.customers` as c
94  JOIN `Target_SQL.orders` as o
95  ON c.customer_id = o.customer_id
96  GROUP BY c.customer_state
97  ORDER BY ROUND(AVG(DATE_DIFF( order_delivered_customer_date,order_purchase_timestamp, day)),0) DESC
98  LIMIT 5)
99  UNION ALL
100 (SELECT
101 | c.customer_state,

```

**QUERY:**

[RUN](#)
[SAVE](#)
[SHARE](#)
[SCHEDULE](#)
[MORE](#)

```

110 ----7)
111 (SELECT
112   c.customer_state,
113   ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)),0) as diff_estimated_delivery
114 FROM `Target_SQL.customers` as c
115 JOIN `Target_SQL.orders` as o
116 ON c.customer_id = o.customer_id
117 GROUP BY c.customer_state
118 ORDER BY ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)),0) ASC
119 LIMIT 5)
120 UNION ALL
121 (SELECT
122   c.customer_state,
123   ROUND(AVG(DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, day)),0) as diff_estimated_delivery
124 FROM `Target_SQL.customers` as c

```

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH [PREVIEW](#)

Row	customer_state	diff_estimated_c
6	AL	8.0
7	MA	9.0

## 6) Payment type analysis:

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

QUERY:

SELECT

```

x.payment_type,
sum(CASE WHEN x.active_months = 1 then 1 else 0 END) AS JAN,
sum(CASE WHEN x.active_months = 2 then 1 else 0 END) AS FEB,
sum(CASE WHEN x.active_months = 3 then 1 else 0 END) AS MAR,
sum(CASE WHEN x.active_months = 4 then 1 else 0 END) AS APR,
sum(CASE WHEN x.active_months = 5 then 1 else 0 END) AS MAY,
sum(CASE WHEN x.active_months = 6 then 1 else 0 END) AS JUN,
sum(CASE WHEN x.active_months = 7 then 1 else 0 END) AS JUL,
sum(CASE WHEN x.active_months = 8 then 1 else 0 END) AS AUG,
sum(CASE WHEN x.active_months = 9 then 1 else 0 END) AS SEPT,
sum(CASE WHEN x.active_months = 10 then 1 else 0 END) AS OCT,
sum(CASE WHEN x.active_months = 11 then 1 else 0 END) AS NOV,
sum(CASE WHEN x.active_months = 12 then 1 else 0 END) AS DEC,
COUNT(DISTINCT x.order_id) as no_of_orders
FROM
(SELECT
  p.payment_type,
  o.order_id,
  o.order_purchase_timestamp,
  EXTRACT(MONTH from order_purchase_timestamp) AS active_months
FROM `Target_SQL.orders` as o
LEFT JOIN `Target_SQL.payments` as p
ON o.order_id = p.order_id
ORDER BY order_purchase_timestamp) as x
GROUP BY x.payment_type
order by x.payment_type

```

<div> <div>*Unsaved query 3</div> <div>geolocation</div> <div>*Unsaved query</div> <div>payments</div> </div> <div> <div>RUN</div> <div>SAVE</div> <div>SHARE</div> <div>SCHEDULE</div> <div>MORE</div> </div>									
<pre> 3 SELECT 4   x.payment_type, 5   sum(CASE WHEN x.active_months = 1 then 1 else 0 END) AS JAN, 6   sum(CASE WHEN x.active_months = 2 then 1 else 0 END) AS FEB, 7   sum(CASE WHEN x.active_months = 3 then 1 else 0 END) AS MAR, 8   sum(CASE WHEN x.active_months = 4 then 1 else 0 END) AS APR, 9   sum(CASE WHEN x.active_months = 5 then 1 else 0 END) AS MAY, 10  sum(CASE WHEN x.active_months = 6 then 1 else 0 END) AS JUN, 11  sum(CASE WHEN x.active_months = 7 then 1 else 0 END) AS JUL, 12  sum(CASE WHEN x.active_months = 8 then 1 else 0 END) AS AUG, 13  sum(CASE WHEN x.active_months = 9 then 1 else 0 END) AS SEPT, 14  sum(CASE WHEN x.active_months = 10 then 1 else 0 END) AS OCT, </pre>									
<div>Query results</div> <div>SAVE RESULTS EXPLORE DATA</div>									
<div> <div>JOB INFORMATION</div> <div>RESULTS</div> <div>JSON</div> <div>EXECUTION DETAILS</div> <div>EXECUTION GRAPH</div> <div>PREVIEW</div> </div>									
Row	payment_type	JAN	FEB	MAR	APR	MAY	JUN	JUL	
1	null	0	0	0	0	0	0	0	
2	UPI	1715	1723	1942	1783	2035	1807	2074	
3	credit_card	6103	6609	7707	7301	8350	7276	7841	
4	debit_card	110	60	100	124	61	200	264	

## ----2) Count of orders based on the no. of payment instalments

QUERY:

```

SELECT
  p.payment_installments,
  COUNT(DISTINCT o.order_id) as total_no_of_orders
FROM `Target_SQL.orders` as o
LEFT JOIN `Target_SQL.payments` as p
on o.order_id = p.order_id
GROUP BY p.payment_installments
ORDER BY p.payment_installments

```

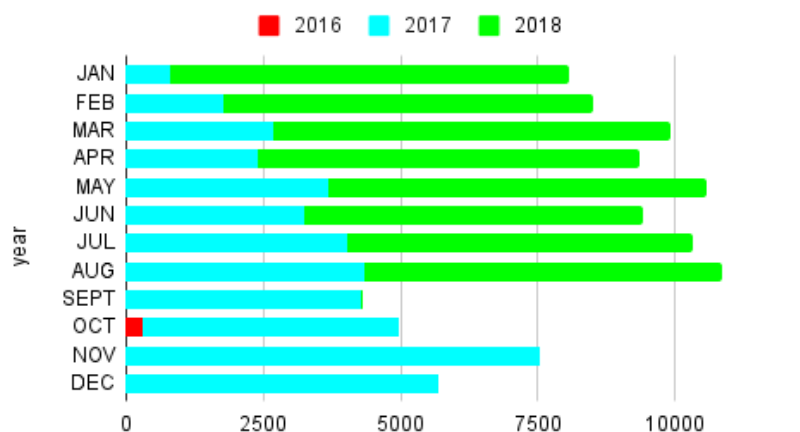
<div> <div>SINESS_CASE_QUERIE... 18</div> <div>*Unsaved query 3</div> <div>geolocation</div> <div>*Unsaved query</div> <div>payments</div> </div> <div> <div>RUN</div> <div>SAVE</div> <div>SHARE</div> <div>SCHEDULE</div> <div>MORE</div> </div>									
<pre> 31 ---2) 32 SELECT 33   p.payment_installments, 34   COUNT(DISTINCT o.order_id) as total_no_of_orders 35 FROM `Target_SQL.orders` as o 36 LEFT JOIN `Target_SQL.payments` as p 37 on o.order_id = p.order_id 38 GROUP BY p.payment_installments 39 ORDER BY p.payment_installments </pre>									
<div>Query results</div> <div>SAVE RESULTS EXPLORE DATA</div>									
<div> <div>JOB INFORMATION</div> <div>RESULTS</div> <div>JSON</div> <div>EXECUTION DETAILS</div> <div>EXECUTION GRAPH</div> <div>PREVIEW</div> </div>									
Row	payment_installments	total_no_of_orders							
3	1	49060							
4	2	12389							
5	3	10443							
6	4	7088							
7	5	5234							

## 7) ACTIONABLE INSIGHTS

The given target orders data (oct,2016 to sept,2018) for the region Brazil, has some interesting insights coming away.

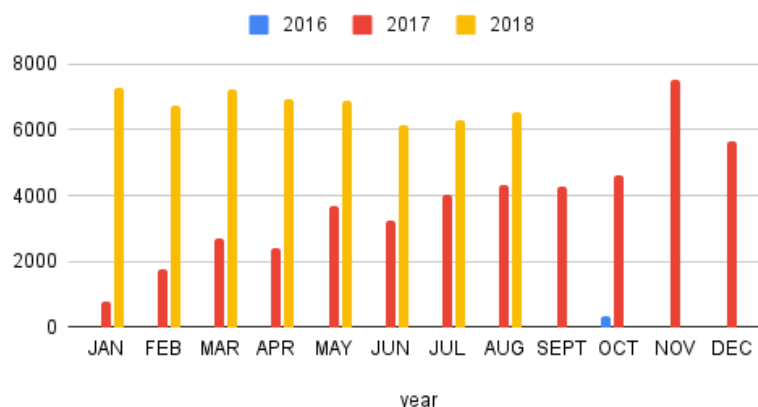
year	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEPT	OCT	NOV	DEC	no_of_orders
2016	0	0	0	0	0	0	0	0	4	324	0	1	329
2017	800	1780	2682	2404	3700	3245	4026	4331	4285	4631	7544	5673	45101
2018	7269	6728	7211	6939	6873	6167	6292	6512	16	4	0	0	54011

Total order placement over months in 3 years



- In the above stacked bar chart, it can be seen that year 2018 shows proper growth in order counts when compared to the other two years (data derived out of Q2.1)

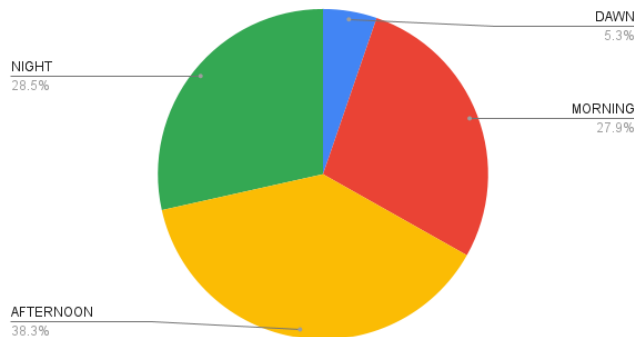
Order placed over months in 3 years



- The above given column chart briefly says that the highest ever order placement was observed during Nov, 2017 followed by Jan, 2018.
- The order counts for months Jan to Jun, 2017 are significantly dull compared to the year 2018 in the respective months (data derived out of Q2.1)

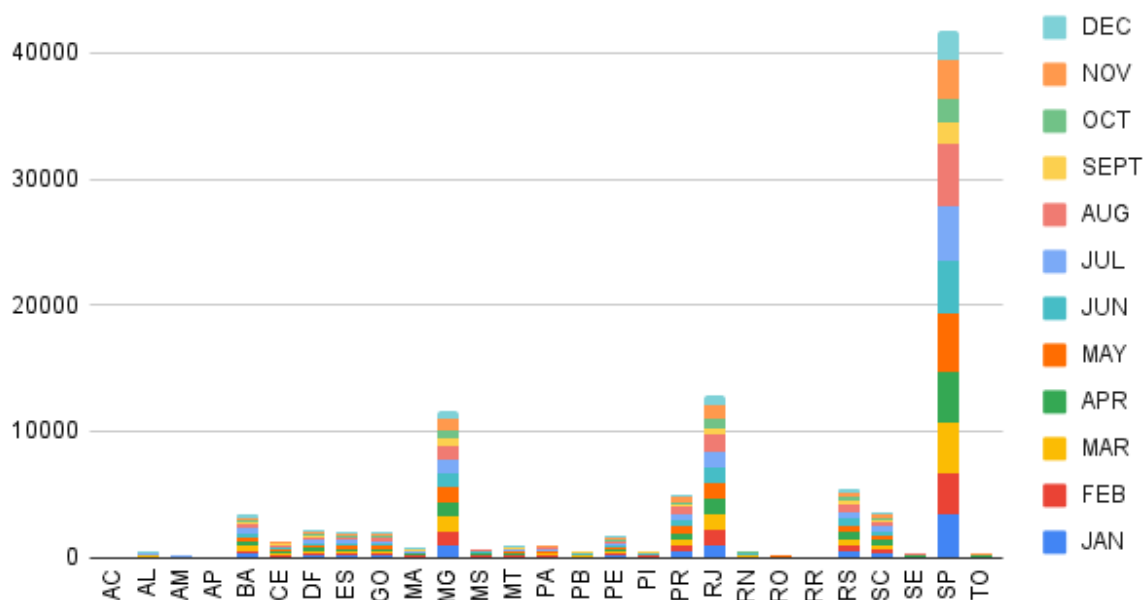
DAWN	MORNING	AFTERNOON	NIGHT	Total_orders
5242	27733	38135	28331	99441

Sales pattern over different hours in a day



From this pie-chart, it can be observed that out of total orders, people of Brazil preferred afternoon (1 – 5 PM) hours for purchase and least at dawn (data derived from Q2.2)

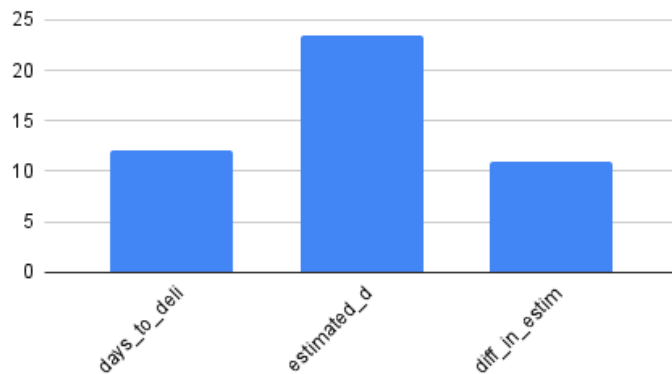
Order placed over months from different states of Brazil



- In this stacked bar chart it can be observed that most orders were placed by state – “SP” which is almost half the number of total orders placed, especially in the month of July which signifies summer in Brazil.
- And when explored more into the customer counts(Q3.2), most number of customers were also found in SP(41,746), RJ(12,852) and MG(11,635)

In regard to insights from Q4, total sum of payment, 2018 (8694733.839) for the period of Jan to Aug is **136%** higher than the total sum of payment, 2017(Jan – Aug) is 3669022.11. So based on this and earlier observations sales and price both spiked in the year 2018 compared to 2017.

AVG time between purchasing, delivering and estimated delivery



From this graph you can observe the time taken to actually deliver a product averages around 12 days, whereas the estimated time to deliver from the date of order averages around 23 days and finally the difference between actual delivered date and estimated date is around 11 days (data derived from Q5)

avg\_freight\_value, time\_to\_delivery and diff\_estimated\_delivery



The above given combo graph signifies that average freight value(shipment cost) lies below 40 for most of the states, time taken to delivery is high in states – “AP” and “RR”. And lowest in state – “SP”.

Top 5 states with lowest/highest average freight value

	customer_state	mean_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04
6	RR	42.98
7	PB	42.72
8	RO	41.07
9	AC	40.07
10	PI	39.15

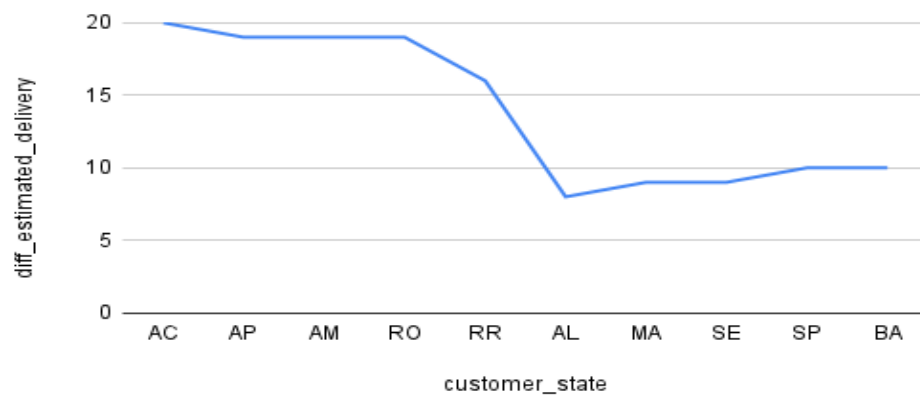
States – SP, PR, MG, RJ, DF charges less freight values of which “SP” is the least of 15.15 amongst all the listed states.

States – RR, PB, RO, AC, PI charges the highest freight values of which “RR” is charges the most of 42.9

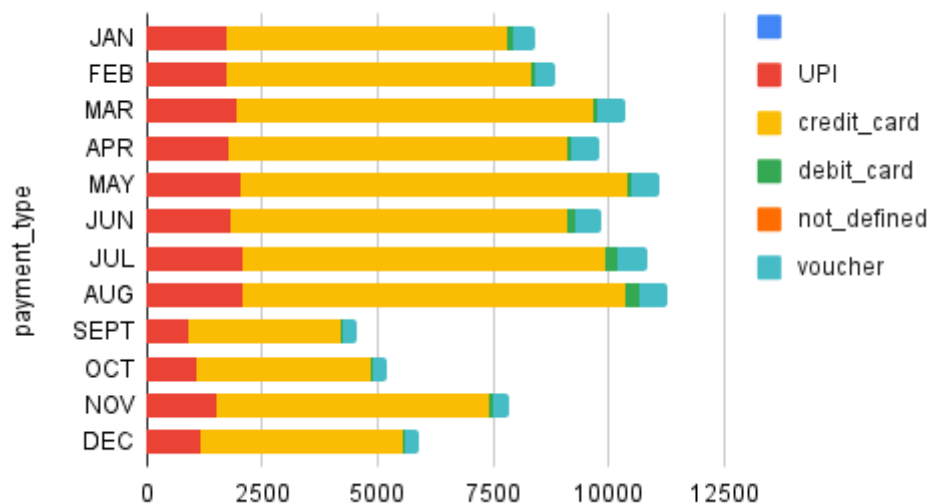


Top 5 states where delivery is really fast/ not so fast compared to estimated date

diff\_estimated\_delivery vs. TOP 5 not so fastest and fastest customer\_states

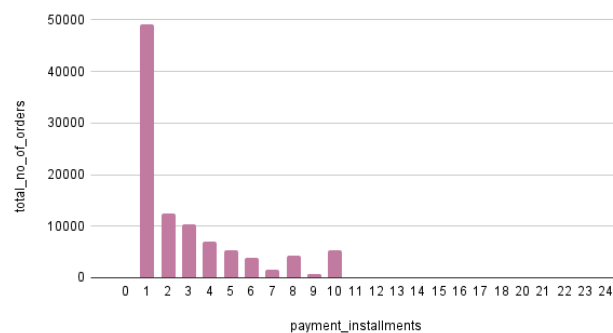


Payment types followed for order counts across months



From this above given stacked bar chart, it is observed that credit-card payment is the most practised payment type followed by UPI payments. And maximum purchase is observed over months of May, July and August. (data derived from Q6.1)

total\_no\_of\_orders vs. payment\_installments



From the column chart on the left, it can be observed that most of the payment instalments for the given orders last for 1 month followed by 2 to 4 months. And sometimes for 8 or 10 months and rarely others (data derived from Q6.2)

## Q8. Recommendations

From the data analysed, it seems that order counts are in increasing trends over the year. However a few things can be improvised from my point of view, which are listed below:

- Expansion of market across areas with more population will see a good impact in sales.
- From the given data alone, the state “SP” seems to have absurdly high number of orders, so similar strategy which is followed in the state “SP” can be followed in other places to boost the sale numbers in other areas.
- **Time taken to deliver** orders are is high in states – “AP” and “RR” and lowest in state – “SP” and “SP” is also the state with most orders, which can also be a reason why order counts are high in “SP”. So, efforts should be taken to improve the delivery time where time taken to deliver is high, which might help in showing better counts in the future.
- Reason why **Freight value** is also less in state SP, an area with most no. of orders and lowest delivery time can also be because the warehouses are more in that area. So expanding warehouses across areas can be helpful for increasing order counts in other areas.
- In the given data, **credit card and UPI** were the most used payment types, so introducing more lucrative credit card offers in partnership with different banks can add more customers.
- From the data analyzation of **payment instalments**, it was understood that most order payments were completed in 1-5 months, so the products for which such kind of payment is done can be sorted out and increased in numbers across states and create more demand.