Target Business Case - Analysis Report

By : Chirag Mahendra Hire

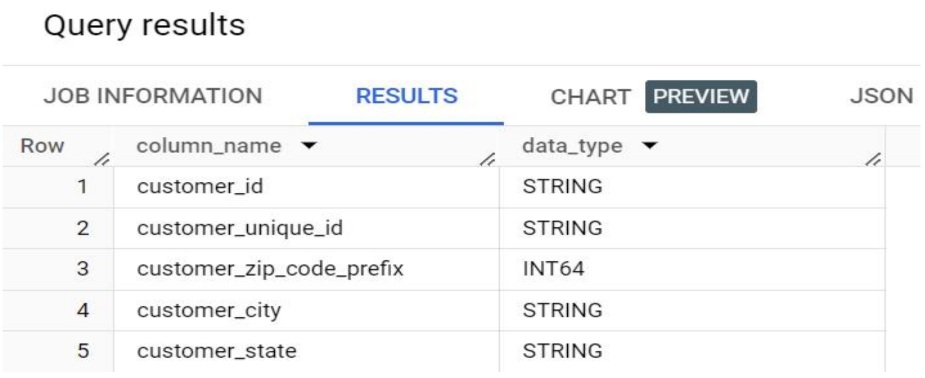
A. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset :

1. Data type of all columns in the "customers" table.

Query:

```
select column_name, data_type
from `sql-intro-business-case.target_company.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers';
```

Output:



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

Insights:

- ➢ The customers table has 5 columns out of which 4 are of string data type and 1 is integer.

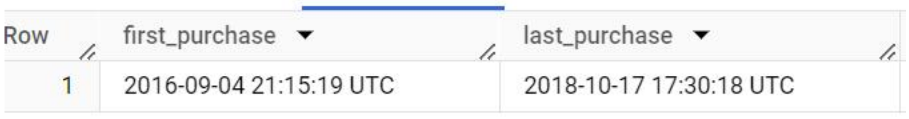
Recommendation: NA

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

Query:

```
select
  min(order_purchase_timestamp) as first_purchase,
  max(order_purchase_timestamp) as last_purchase
from
  `target_company.orders`
```

Output:



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

Insights:

I. The time range between which the orders were placed was 4 Sept. 2016 to 17 Oct. 2018.

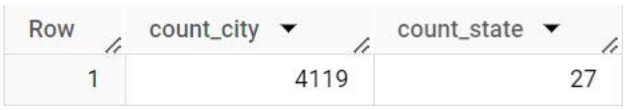
Recommendation: NA

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

Query:

```
select
  count(distinct customer_city) as count_city,
  count(distinct customer_state) as count_state
from
  `target_company.customers`;
```

Output:



Row	count_city	count_state
1	4119	27

Insights:

➢ Total Cities of are 4119 and states are 27 from where the customers order the products.

Recommendation: NA

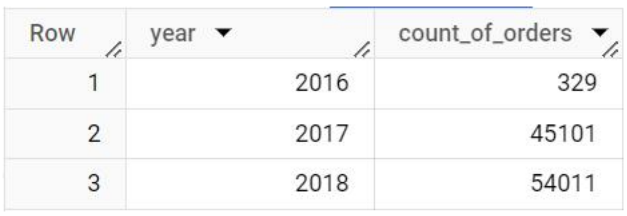
B. In-depth Exploration:

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

Query:

```
select
  extract(year from order_purchase_timestamp) as `year`,
  count(order_id) as `count_of_orders`
from
  `target_company.orders`
group by
  year
order by
  year asc;
```

Output:



Row	year	count_of_orders
1	2016	329
2	2017	45101
3	2018	54011

Insights: As we can see from the data, there is the growing trend in orders placed by the customer over the past years. The orders are incresing year after year.

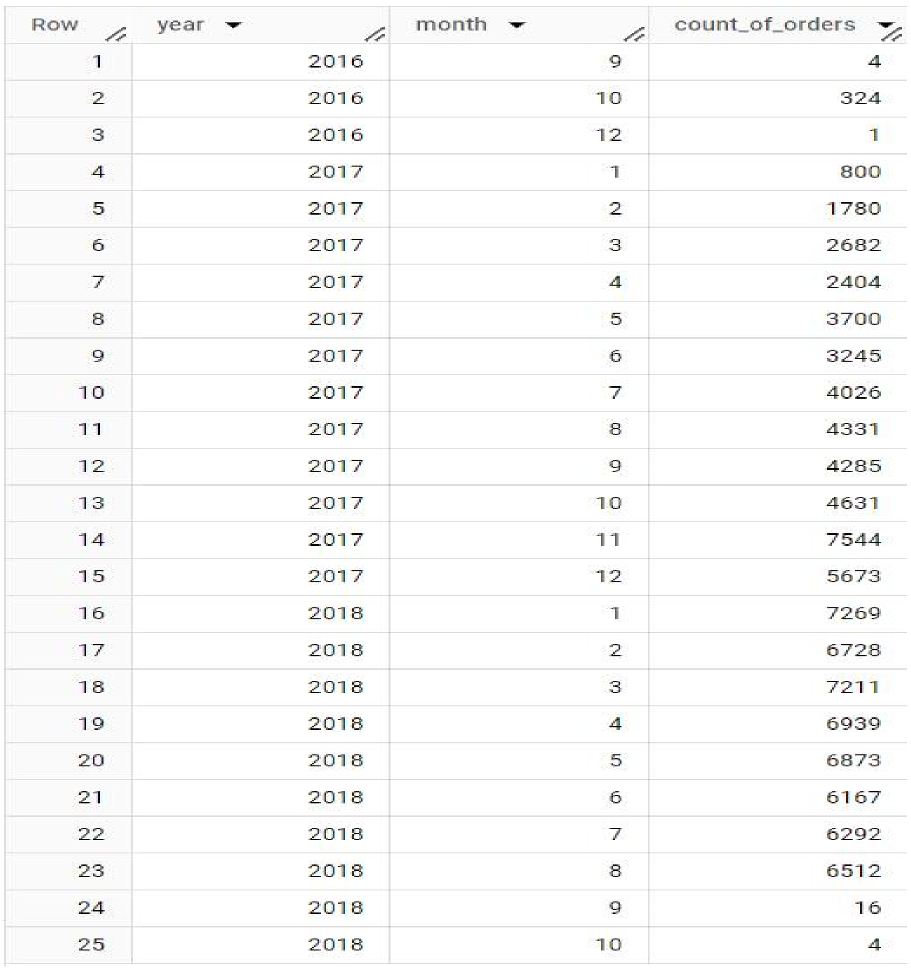
Recommendation: NA

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

Query:

```
select
  extract(year from order_purchase_timestamp) as `year`,
  extract(month from order_purchase_timestamp) as `month`,
  count(order_id) as `count_of_orders`
from
  `target_company.orders`
group by
  year, month
order by
  year asc, month asc;
```

Output:



Row	year	month	count_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544
15	2017	12	5673
16	2018	1	7269
17	2018	2	6728
18	2018	3	7211
19	2018	4	6939
20	2018	5	6873
21	2018	6	6167
22	2018	7	6292
23	2018	8	6512
24	2018	9	16
25	2018	10	4

Insights:

- From the above data it is observed the monthly orders from the customers are increasing.
- As we can observed the highest number of orders are placed in the month of November 2017 which are 7544 orders.
- Also in 2018 all months except September and October, the sales of products is good.
- In the last 2 months of 2018 the order is drastically decreased.

Recommendation:

- By seeing the above data company should evaluate what is the reason products sold more in November 2017, whether it is heavy discount on products or due to any festival this should be evaluate.
- Finally the big concern for Tiger is the last two months of 2018. The orders purchased by the customers in these months are very less.

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

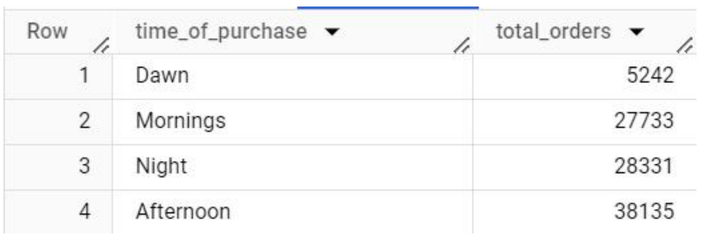
1. 0-6 hrs : Dawn
2. 7-12 hrs : Mornings
3. 13-18 hrs : Afternoon
4. 19-23 hrs : Night

Query:

```
select
  t.time_of_purchase,
  count(customer_id) as `total_orders`
from(select
  customer_id,
  extract(hour from order_purchase_timestamp) as `ordering_hour`,
  order_purchase_timestamp,
  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_purchase`

from
  `target_company.orders`) as `t`
group by
  time_of_purchase
order by
  total_orders
```

Output:



Row	time_of_purchase	total_orders
1	Dawn	5242
2	Mornings	27733
3	Night	28331
4	Afternoon	38135

Insights:

- From the above output we see that the customers order more in the afternoon.
- And approximately same in the morning and night time.
- At the dawn time we can see the number of orders are very less.

Recommendation: NA

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

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

I. I have selected one state from the total states, as there are total 27 states so it is not feasible to give explanation about the whole data in this word file.
II. Therefore it is easy to give the insights and recommendations based on one state data.
III. But for reference purpose I am also going to write the query for the month on month no. Of orders placed in each state.

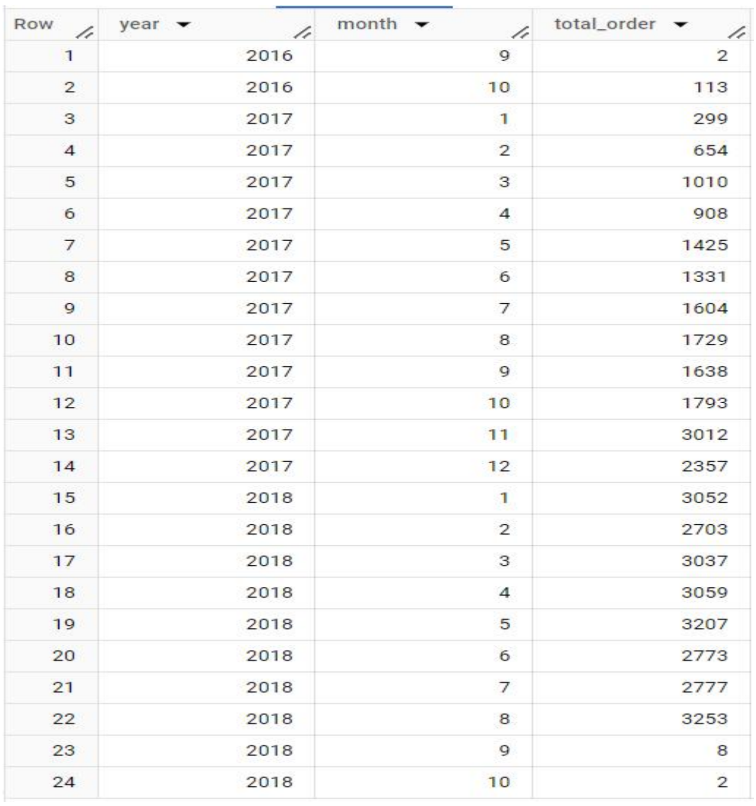
Query for All States:

```
select
  c.customer_state,
  extract(year from o.order_purchase_timestamp) as `years`,
  extract(month from o.order_purchase_timestamp) as `months`,
  count(o.order_id)  `total_orders`
from
  `target_company.customers` as `c` inner join `target_company.orders` as
`o`
  on c.customer_id = o.customer_id
group by
  customer_state,
  years, months
order by
  customer_state,
  years, months;
```

Query for Single State:

```
select
  extract(year from o.order_purchase_timestamp) as `year`,
  extract(month from o.order_purchase_timestamp) as `month`,
  count(o.order_id) as total_order
from
  `target_company.customers` as `c` inner join `target_company.orders` as
`o`
  on c.customer_id = o.customer_id
where
  c.customer_state = 'SP'
GROUP BY
  year, month
order by
  year, month;
```

Output:



Row	year	month	total_order
1	2016	9	2
2	2016	10	113
3	2017	1	299
4	2017	2	654
5	2017	3	1010
6	2017	4	908
7	2017	5	1425
8	2017	6	1331
9	2017	7	1604
10	2017	8	1729
11	2017	9	1638
12	2017	10	1793
13	2017	11	3012
14	2017	12	2357
15	2018	1	3052
16	2018	2	2703
17	2018	3	3037
18	2018	4	3059
19	2018	5	3207
20	2018	6	2773
21	2018	7	2777
22	2018	8	3253
23	2018	9	8
24	2018	10	2

Insights:

- ➢ The above data is for SP state which has the highest no. Of orders within all states.
- ➢ The data shows that the orders are increasing month by month but it is fluctuating.
- ➢ We can see the rapid increase in no. Of orders in November 2017.
- ➢ Also in September 2018 the orders are rapidly decreased, despite of the last months total orders are very high.

Recommendation:

- ➢ We need to analyse the month where total number of orders are very high, whether it is seasonality or discount offers on the products.
- ➢ Also check what is the reason behind very low number of orders after September 2018.

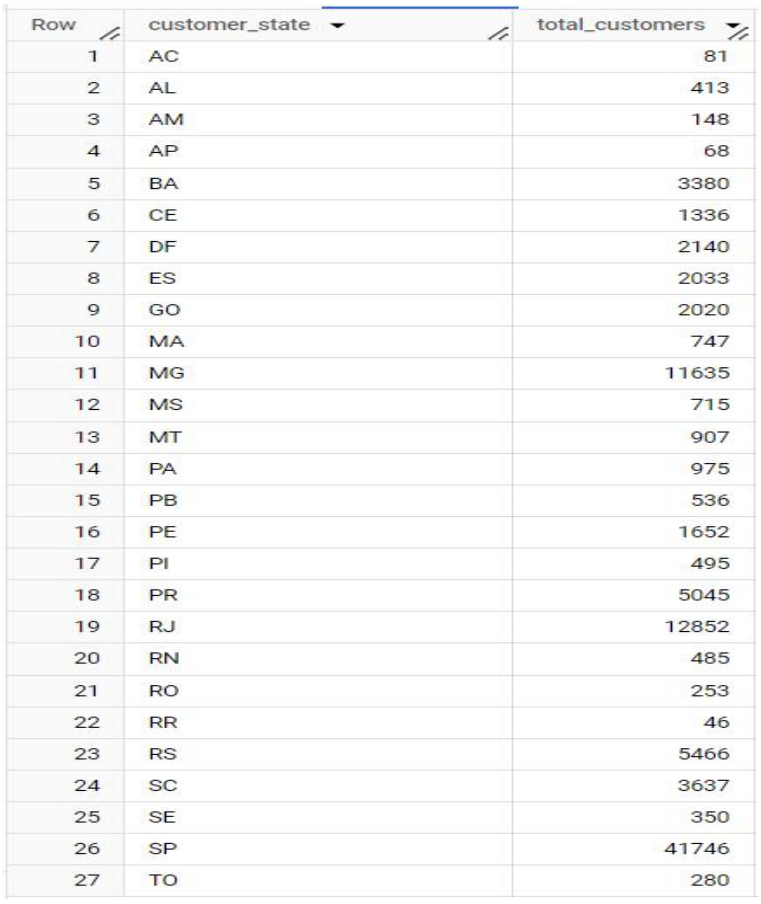
2. How are the customers distributed across all the states?

The query to showcase the customer distribution across all the states in Brazil is as follow:

Query:

```
select
  customer_state,
  count(customer_id) as total_customers
from
  `target_company.customers`
group by
  customer_state
order by
  customer_state;
```

Output:



Row	customer_state	total_customers
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635
12	MS	715
13	MT	907
14	PA	975
15	PB	536
16	PE	1652
17	PI	495
18	PR	5045
19	RJ	12852
20	RN	485
21	RO	253
22	RR	46
23	RS	5466
24	SC	3637
25	SE	350
26	SP	41746
27	TO	280

Insights:

- The above data is sorted according to the states in Brazil and the total number of customers in that states.
- We can see the huge difference between the orders placed by the customers from different states.
- The customers in state ‘SP’ placed more number of orders compare to all the states.
- And the customers in state ‘RR’ placed very less number of orders compare to other states.

Recommendation:

- The huge difference between number of orders in the states is due to some reasons, that can be the population of that state
- The Other reason can be the the development of that state, in comparison with people living in the normal cities to the people living in metro cities orders more.
- To improve the sells in the states which have less orders, the company can give discounts on the products initially to attract customers.

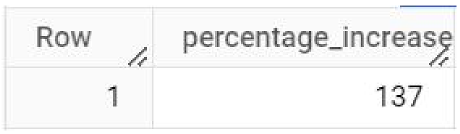
D. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

1. 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.

Query:

```
select
  cast((t.cost_of_orders - t.lag)/t.lag*100 as int) as
`percentage_increase`
from(select
  extract(year from ord.order_purchase_timestamp) as `year`,
  cast(sum(pmt.payment_value)as int) as `cost_of_orders`,
  lag(cast(sum(pmt.payment_value)as int), 1)
  over(order by cast(sum(pmt.payment_value)as int))  as `lag`
from
  `target_company.orders` as `ord` inner join `target_company.payments` as
`pmt`
  on ord.order_id = pmt.order_id
where
  (extract(year from ord.order_purchase_timestamp) = 2017 or
  extract(year from ord.order_purchase_timestamp) = 2018) and
  extract(month from ord.order_purchase_timestamp) between 1 and 8
group by
  extract(year from ord.order_purchase_timestamp)) as t
order by
  percentage_increase
limit 1 offset 1;
```

Output:



Row	percentage_increase
1	137

Insights:

- The percentage increase in the cost of orders in the year 2018 compare to 2017 within the months January to August is 137%.
- The cost of orders in 2017 is 3669022 and in 2018 is 8694734.
- 137% growth in the orders cost are very good figures.

Recommendation: NA

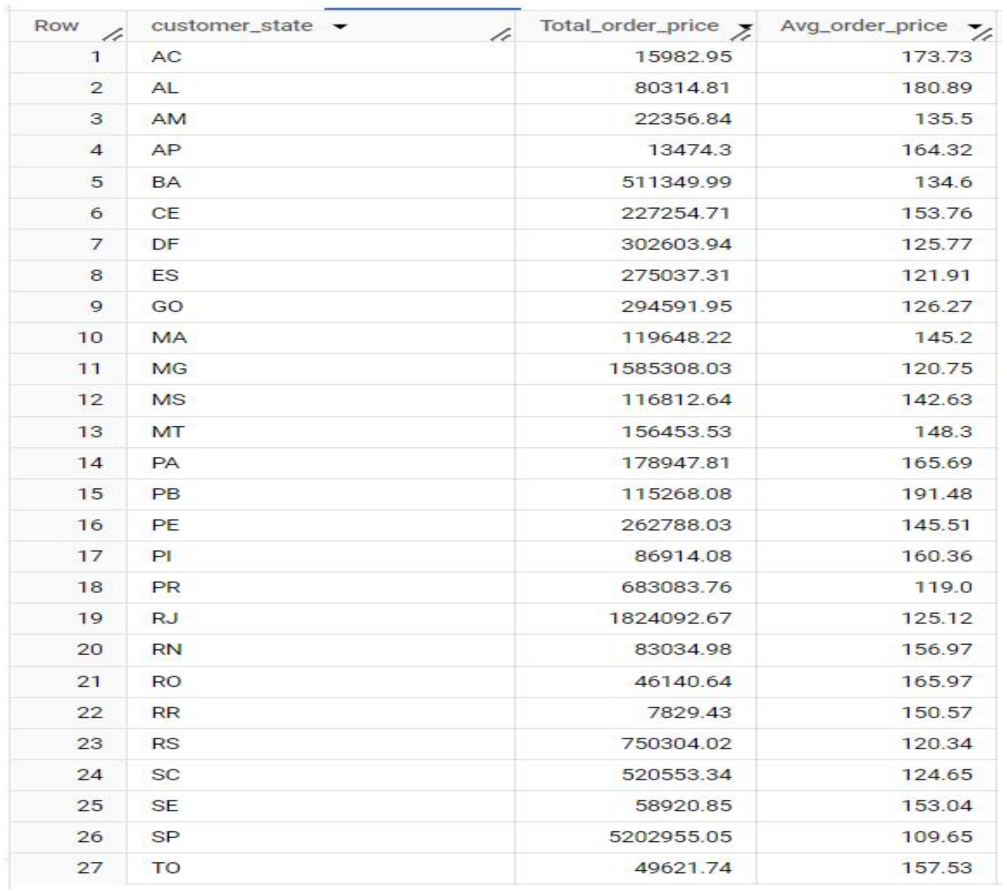
2. Calculate the Total & Average value of order price for each state.

Query:

```
select
  c.customer_state,
  round(sum(oi.price),2) as `Total_order_price`,
  round(avg(oi.price),2) as `Avg_order_price`

from
  `target_company.order_items` as `oi` inner join `target_company.orders`
as `o`
  on oi.order_id = o.order_id
  inner Join `target_company.customers` as `c`
  on o.customer_id = c.customer_id
group by
  c.customer_state
order by
  c.customer_state;
```

Output:



Row	customer_state	Total_order_price	Avg_order_price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2
11	MG	1585308.03	120.75
12	MS	116812.64	142.63
13	MT	156453.53	148.3
14	PA	178947.81	165.69
15	PB	115268.08	191.48
16	PE	262788.03	145.51
17	PI	86914.08	160.36
18	PR	683083.76	119.0
19	RJ	1824092.67	125.12
20	RN	83034.98	156.97
21	RO	46140.64	165.97
22	RR	7829.43	150.57
23	RS	750304.02	120.34
24	SC	520553.34	124.65
25	SE	58920.85	153.04
26	SP	5202955.05	109.65
27	TO	49621.74	157.53

Insights:

- The above data represents three columns in the first column all the 27 states are mentioned.
- In second column the total order price for each state is given.
- And in the last column average order price is calculated for each state.

➢ In the output table we can see the state ‘SP’ has highest total order price but lowest average price. So this is only possible when there is more population than other states and more cities than other states.

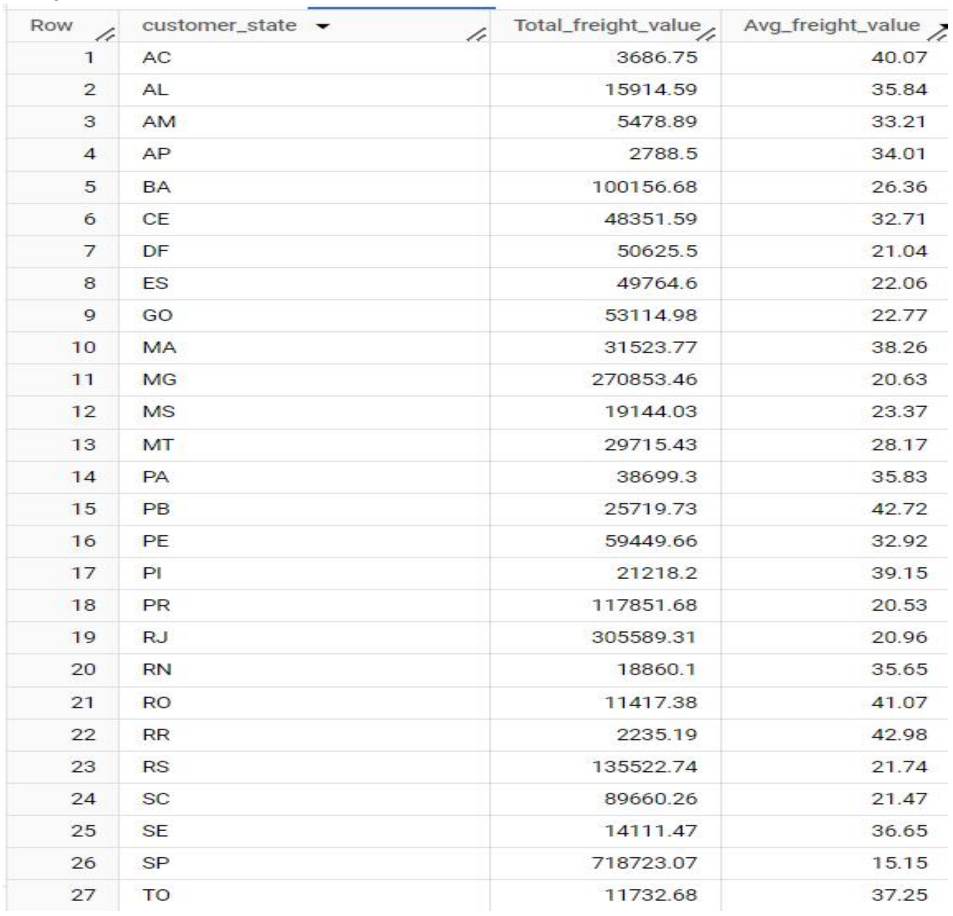
Recommendation: NA

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

Query:

```
select
  c.customer_state,
  round(sum(oi.freight_value),2) as `Total_freight_value`,
  round(avg(oi.freight_value),2) as `Avg_freight_value`
from
  `target_company.order_items` as `oi` inner join `target_company.orders`
as `o`
  on oi.order_id = o.order_id
  inner Join `target_company.customers` as `c`
  on o.customer_id = c.customer_id
group by
  c.customer_state
order by
  c.customer_state;
```

Output:



Row	customer_state	Total_freight_value	Avg_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37
13	MT	29715.43	28.17
14	PA	38699.3	35.83
15	PB	25719.73	42.72
16	PE	59449.66	32.92
17	PI	21218.2	39.15
18	PR	117851.68	20.53
19	RJ	305589.31	20.96
20	RN	18860.1	35.65
21	RO	11417.38	41.07
22	RR	2235.19	42.98
23	RS	135522.74	21.74
24	SC	89660.26	21.47
25	SE	14111.47	36.65
26	SP	718723.07	15.15
27	TO	11732.68	37.25

Insights:

- In above data total freight value and average freight values are calculated for every state.
- State ‘SP’ has highest total freight value, and state ‘RR’ has lowest total freight value.
- State ‘RR’ has highest average freight value, and state ‘SP’ has the lowest average freight value.
- If we consider state ‘SP’ which have highest freight value but lowest average freight value denotes that it has more number of orders.
- So from that we can conclude that more number of orders is one factor to reduce freight value.

Recommendation:

- Try to improve sales in the states which have more average freight value.

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

1. 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.

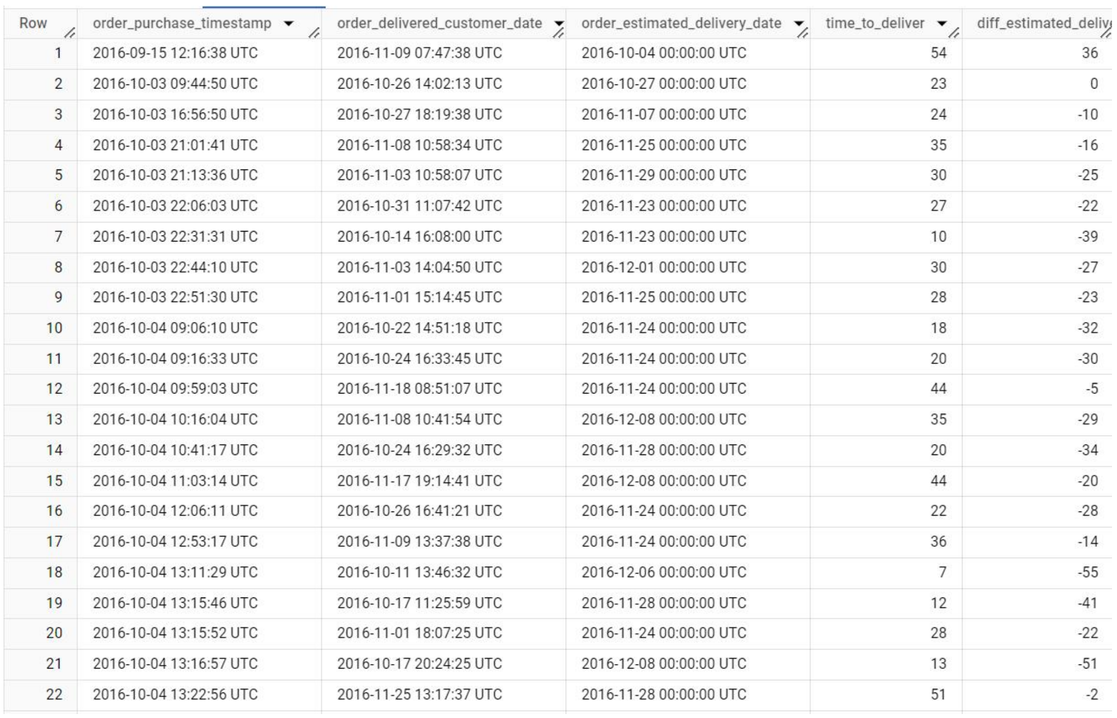
You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

1. **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
2. **diff_estimated_delivery** = order_delivered_customer_date - order_estimated_delivery_date

Query:

```
select
  order_purchase_timestamp,
  order_delivered_customer_date,
  order_estimated_delivery_date,
  date_diff(order_delivered_customer_date, order_purchase_timestamp, day)
as `time_to_deliver`,
  date_diff(order_delivered_customer_date, order_estimated_delivery_date,
day) as `diff_estimated_delivery`
from
  `target_company.orders`
where
  order_delivered_customer_date is not null
order by
  order_purchase_timestamp;
```

Output:



Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_deliv
1	2016-09-15 12:16:38 UTC	2016-11-09 07:47:38 UTC	2016-10-04 00:00:00 UTC	54	36
2	2016-10-03 09:44:50 UTC	2016-10-26 14:02:13 UTC	2016-10-27 00:00:00 UTC	23	0
3	2016-10-03 16:56:50 UTC	2016-10-27 18:19:38 UTC	2016-11-07 00:00:00 UTC	24	-10
4	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	-16
5	2016-10-03 21:13:36 UTC	2016-11-03 10:58:07 UTC	2016-11-29 00:00:00 UTC	30	-25
6	2016-10-03 22:06:03 UTC	2016-10-31 11:07:42 UTC	2016-11-23 00:00:00 UTC	27	-22
7	2016-10-03 22:31:31 UTC	2016-10-14 16:08:00 UTC	2016-11-23 00:00:00 UTC	10	-39
8	2016-10-03 22:44:10 UTC	2016-11-03 14:04:50 UTC	2016-12-01 00:00:00 UTC	30	-27
9	2016-10-03 22:51:30 UTC	2016-11-01 15:14:45 UTC	2016-11-25 00:00:00 UTC	28	-23
10	2016-10-04 09:06:10 UTC	2016-10-22 14:51:18 UTC	2016-11-24 00:00:00 UTC	18	-32
11	2016-10-04 09:16:33 UTC	2016-10-24 16:33:45 UTC	2016-11-24 00:00:00 UTC	20	-30
12	2016-10-04 09:59:03 UTC	2016-11-18 08:51:07 UTC	2016-11-24 00:00:00 UTC	44	-5
13	2016-10-04 10:16:04 UTC	2016-11-08 10:41:54 UTC	2016-12-08 00:00:00 UTC	35	-29
14	2016-10-04 10:41:17 UTC	2016-10-24 16:29:32 UTC	2016-11-28 00:00:00 UTC	20	-34
15	2016-10-04 11:03:14 UTC	2016-11-17 19:14:41 UTC	2016-12-08 00:00:00 UTC	44	-20
16	2016-10-04 12:06:11 UTC	2016-10-26 16:41:21 UTC	2016-11-24 00:00:00 UTC	22	-28
17	2016-10-04 12:53:17 UTC	2016-11-09 13:37:38 UTC	2016-11-24 00:00:00 UTC	36	-14
18	2016-10-04 13:11:29 UTC	2016-10-11 13:46:32 UTC	2016-12-06 00:00:00 UTC	7	-55
19	2016-10-04 13:15:46 UTC	2016-10-17 11:25:59 UTC	2016-11-28 00:00:00 UTC	12	-41
20	2016-10-04 13:15:52 UTC	2016-11-01 18:07:25 UTC	2016-11-24 00:00:00 UTC	28	-22
21	2016-10-04 13:16:57 UTC	2016-10-17 20:24:25 UTC	2016-12-08 00:00:00 UTC	13	-51
22	2016-10-04 13:22:56 UTC	2016-11-25 13:17:37 UTC	2016-11-28 00:00:00 UTC	51	-2

Insights:

- In the above data we fetched the information of no. Of days required to delivered the order and difference between order delivered date and order delivery estimated date.
- As we can see in the first row 54 days required to delivered an order. And the difference between delivered date of order and estimated date of order is 36.
- It means that order is delivered 36 days late than the estimated date.
- In row no. 3 difference between delivered date and estimated delivery date is -10, which means that order is delivered before estimated date.
- From the data we can conclude that most of the times order is delivered before estimated date.

Recommendation:

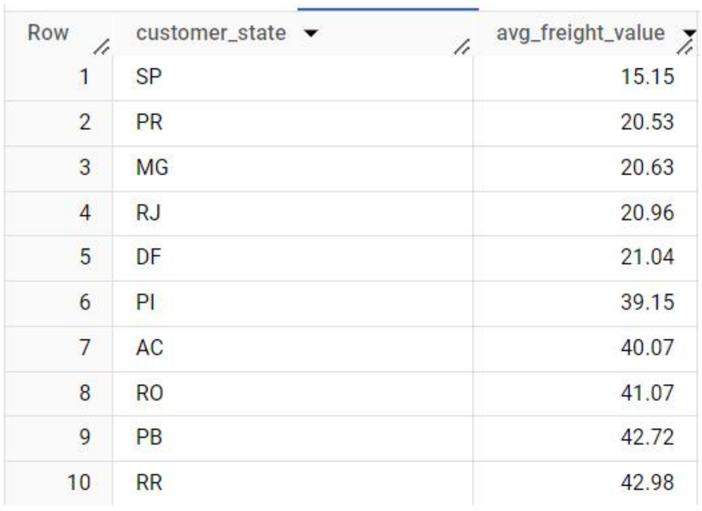
- The difference in the days of delivery date and estimated date is mostly in 2 digits, like 40, 30, 35, 25 days, etc.
- Even the company is delivering most of the time before estimated date though the difference between estimated and delivery date is more many times. The company need to work on estimated date accuracy.

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

Query:

```
select
  t.customer_state,
  t.avg_freight_value
from(select
  c.customer_state,
  round(avg(oi.freight_value),2) as `Avg_freight_value`,
  dense_rank() over(order by avg(oi.freight_value) desc) as `rank_desc`,
  dense_rank() over(order by avg(oi.freight_value)) as `rank_asc`
from
  `target_company.order_items` as `oi` inner join `target_company.orders`
as `o`
  on oi.order_id = o.order_id
  inner Join `target_company.customers` as `c`
  on o.customer_id = c.customer_id
group by
  c.customer_state
order by
  avg_freight_value) as t
where
  t.rank_desc <= 5 or
  t.rank_asc <= 5
order by
  avg_freight_value;
```

Output:



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

Insights:

➢ The 10 rows displayed in the table out of which first five are lowest average freight value on the basis of states, and 6-10 are highest average freight value of the states.

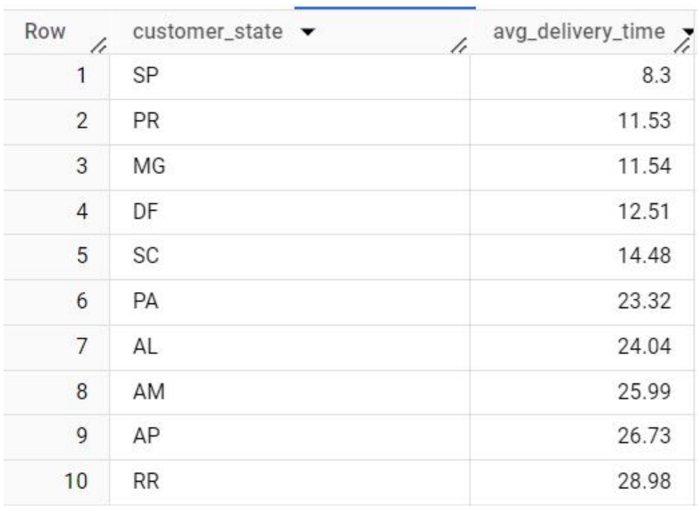
Recommendation: NA

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

Query:

```
select
  t.customer_state,
  t.avg_delivery_time
from(select
  c.customer_state,
  round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)),2) as `avg_delivery_time`,
  dense_rank() over(order by avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)) desc) as `rank_desc`,
  dense_rank() over(order by avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day))) as `rank_asc`
from
  `target_company.orders` as `o` inner Join `target_company.customers` as
`c`
  on o.customer_id = c.customer_id
group by
  c.customer_state
order by
  avg_delivery_time)as t
where
  t.rank_desc <= 5 or
  t.rank_asc <= 5
order by
  avg_delivery_time;
```

Output:



Row	customer_state	avg_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	PA	23.32
7	AL	24.04
8	AM	25.99
9	AP	26.73
10	RR	28.98

Insights:

- ➢ The 10 rows displayed in the table out of which first five are lowest average delivery time in terms of no. Of days of states, and 6-10 are highest average delivery time of the states.

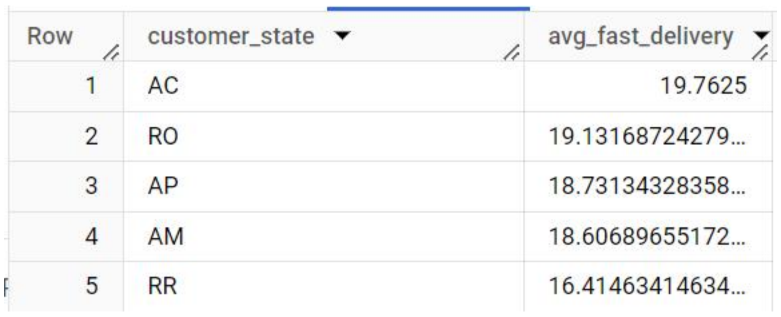
Recommendation: NA

4. 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.

Query:

```
select
  c.customer_state,
  avg(date_diff(order_estimated_delivery_date,
order_delivered_customer_date, day)) as `avg_fast_delivery`
from
  `target_company.orders` as `o` inner Join `target_company.customers` as
`c`
  on o.customer_id = c.customer_id
where
  order_delivered_customer_date is not null
group by
  customer_state
order by
  avg_fast_delivery desc
limit 5;
```

Output:



Row	customer_state	avg_fast_delivery
1	AC	19.7625
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

Insights:

- The data fetched for the top 5 states according to average fastest delivery compared to estimated delivery date.
- The state 'AC' have the average fastest delivery, which averagely delivered the product 19.76 days before the estimated date.

Recommendation: NA

F. Analysis based on the payments:

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

Query (For month on month orders placed using different payment types):

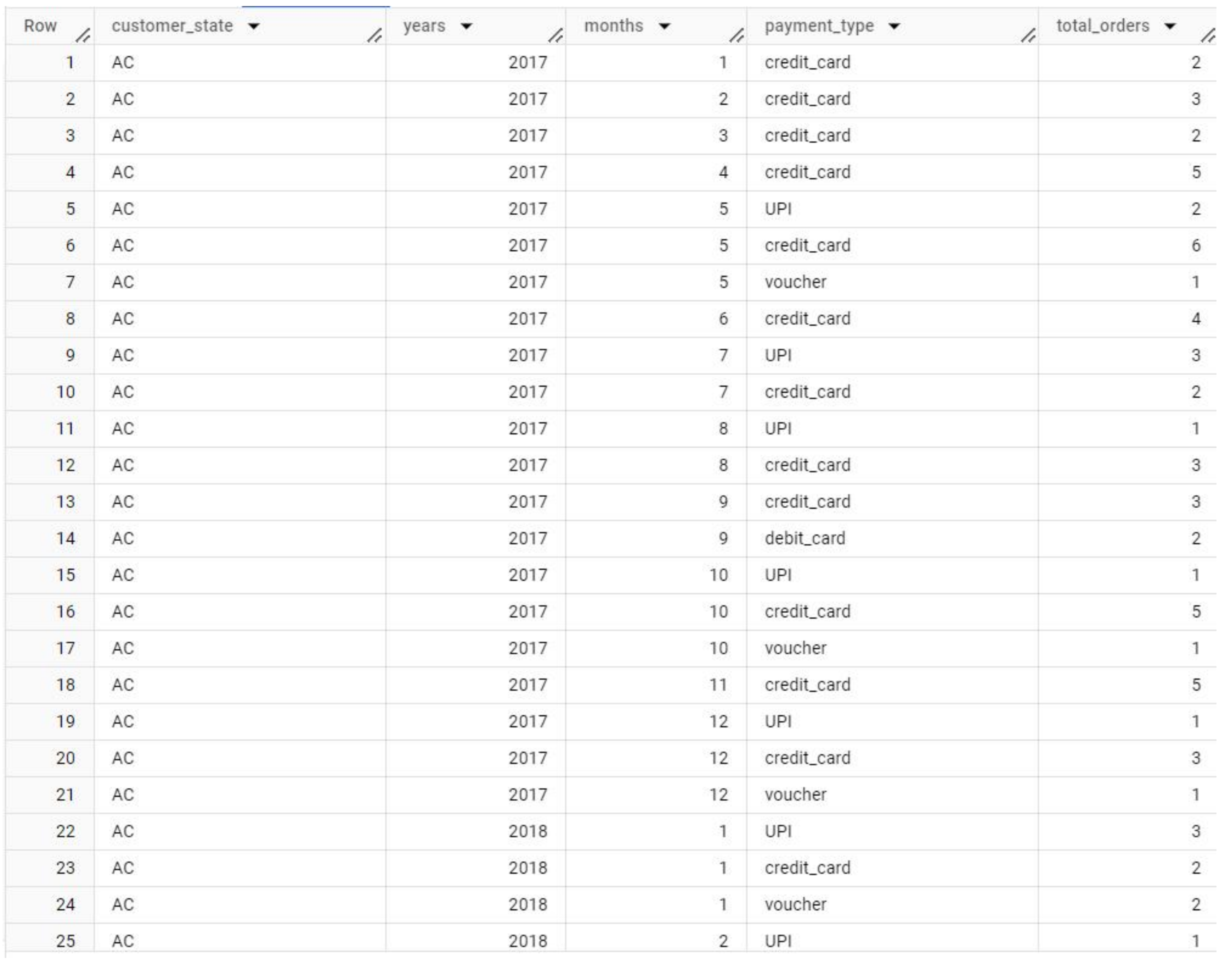
```
select
 c.customer_state,
 extract(year from o.order_purchase_timestamp) as `years`,
 extract(month from o.order_purchase_timestamp) as `months`,
  p.payment_type,
 count(o.order_id) `total_orders`
from
 `target_company.customers` as `c` inner join `target_company.orders` as
`o`
 on c.customer_id = o.customer_id
  inner join
  `target_company.payments` as `p`
  on o.order_id = p.order_id
group by
 customer_state,
 years, months, payment_type
order by
 customer_state,
 years, months,
 payment_type;
```

Query(Total orders grouped by payment_type):

```
select
  payment_type,
  count(order_id) as total_orders
from
  `target_company.payments`
group by
  payment_type
order by
  total_orders;
```

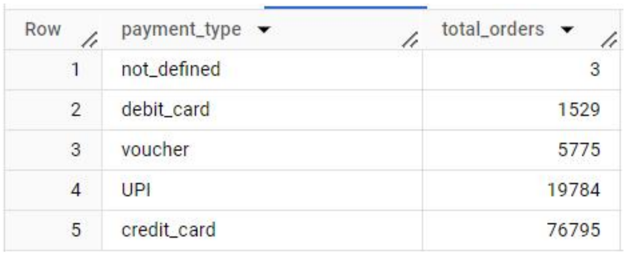
Output:

A. For month on month orders placed using different payment types:



Row	customer_state	years	months	payment_type	total_orders
1	AC	2017	1	credit_card	2
2	AC	2017	2	credit_card	3
3	AC	2017	3	credit_card	2
4	AC	2017	4	credit_card	5
5	AC	2017	5	UPI	2
6	AC	2017	5	credit_card	6
7	AC	2017	5	voucher	1
8	AC	2017	6	credit_card	4
9	AC	2017	7	UPI	3
10	AC	2017	7	credit_card	2
11	AC	2017	8	UPI	1
12	AC	2017	8	credit_card	3
13	AC	2017	9	credit_card	3
14	AC	2017	9	debit_card	2
15	AC	2017	10	UPI	1
16	AC	2017	10	credit_card	5
17	AC	2017	10	voucher	1
18	AC	2017	11	credit_card	5
19	AC	2017	12	UPI	1
20	AC	2017	12	credit_card	3
21	AC	2017	12	voucher	1
22	AC	2018	1	UPI	3
23	AC	2018	1	credit_card	2
24	AC	2018	1	voucher	2
25	AC	2018	2	UPI	1

B. Total orders grouped by payment_type:



Row	payment_type	total_orders
1	not_defined	3
2	debit_card	1529
3	voucher	5775
4	UPI	19784
5	credit_card	76795

Insights:

- ➢ The first output table is sorted based on the states, years, months and payment type to count how many orders are placed in a month using same payment type.
- ➢ And from the second table we can say that most customers used credit cards as payment type to order.
- ➢ The reason for more credit card usage will be more offers on credit cards or its pay later facility.

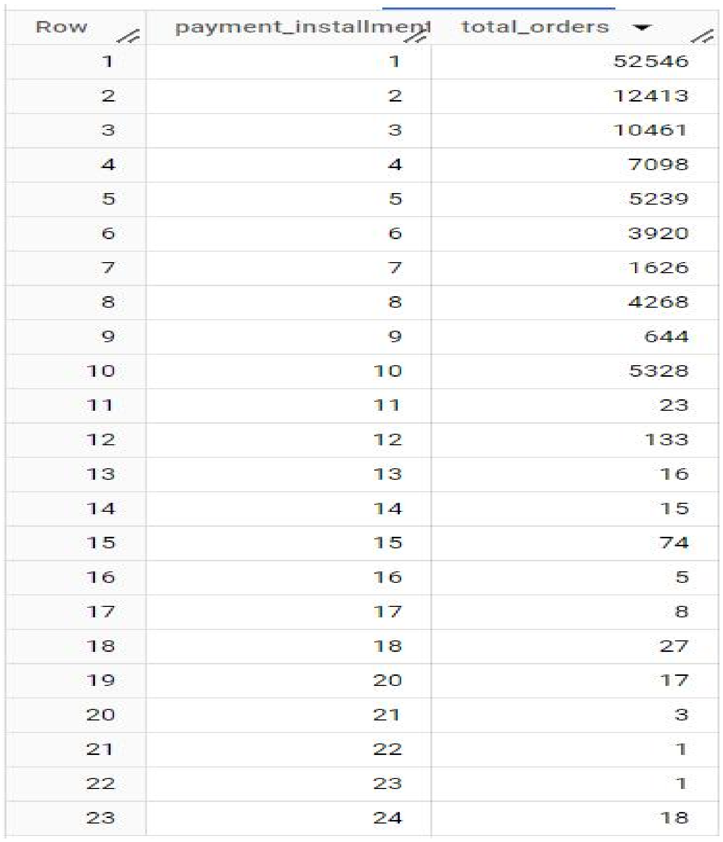
Recommendation: NA

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

Query:

```
select
  payment_installments,
  count(order_id) as total_orders
from
  `target_company.payments`
where
  payment_installments >= 1
group by
  payment_installments;
```

Output:



Row	payment_installments	total_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328
11	11	23
12	12	133
13	13	16
14	14	15
15	15	74
16	16	5
17	17	8
18	18	27
19	20	17
20	21	3
21	22	1
22	23	1
23	24	18

Insights:

- In the above table number of orders placed on the basis of the payment installments are shown.
- Most of the customers have selected for installments between 1-10.
- Most of the orders are purchase only with one installment. Total 52546 orders are placed using 1 installment.

Recommendation: NA