NAME:- RHYTHM SHAH [Linkedin]

Target Business Case Study

Target is one of the world’s most recognized brands and one of America’s leading retailers. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

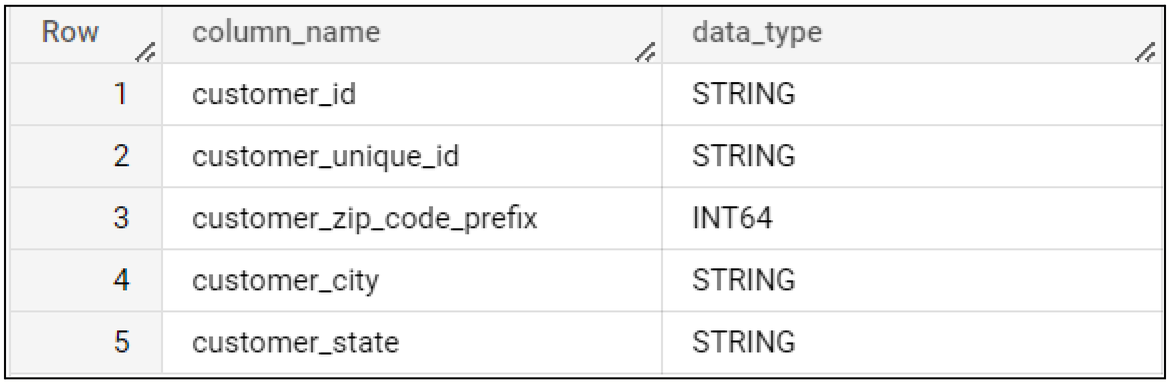
This business case has information of 100k orders from 2016 to 2018 made at Target in Brazil. Its features allows viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers

---

Q1: Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

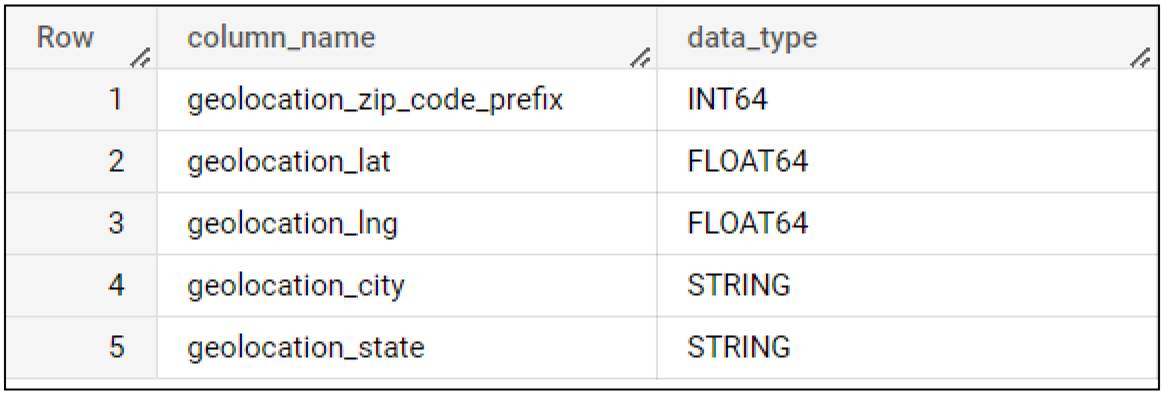
Data type of columns in a table

```
SELECT column_name,data_type
FROM   `target_dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE   table_name = 'customers';
```



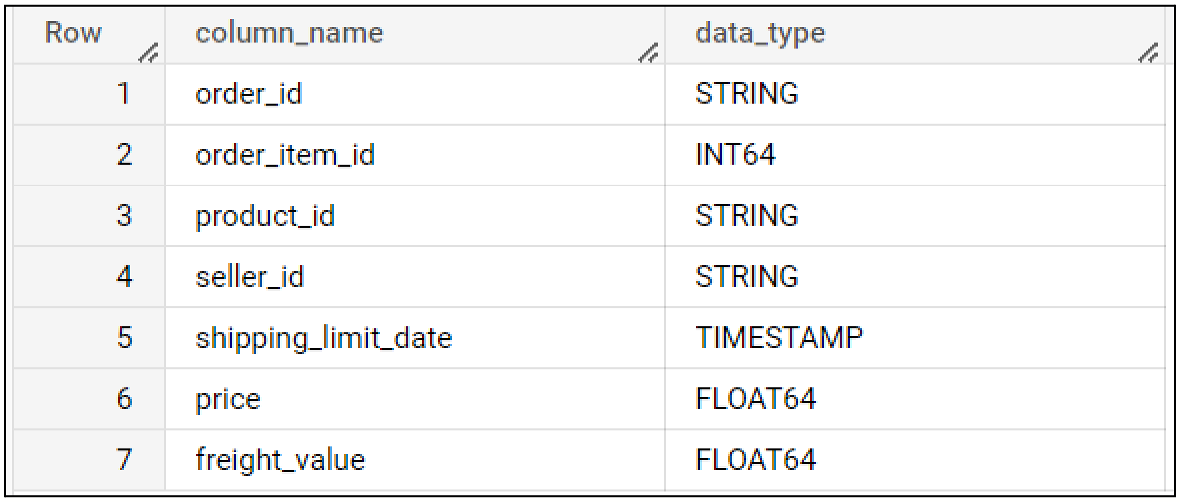
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

```
SELECT column_name,data_type
FROM   `target_dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE   table_name = 'geolocation';
```



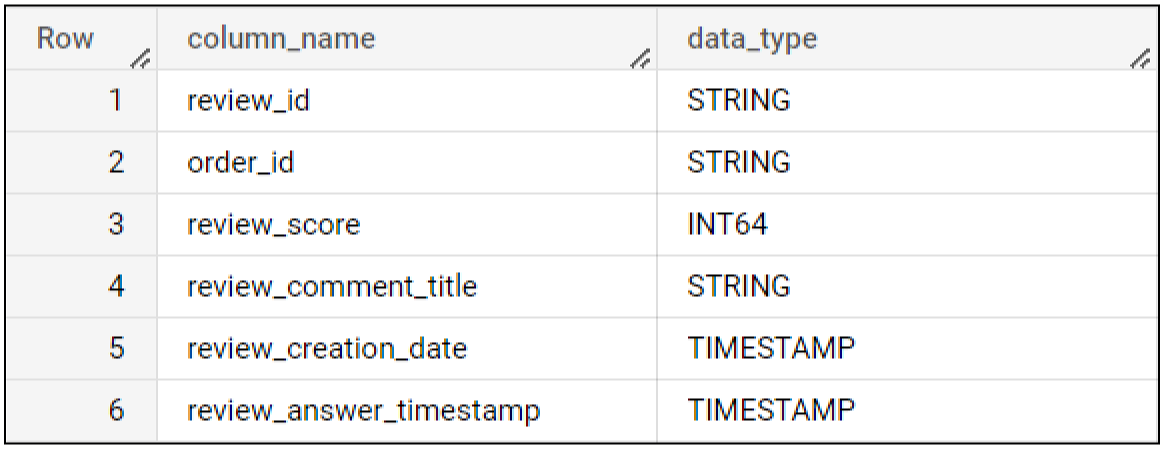
Row	column_name	data_type
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING

```
SELECT column_name,data_type
FROM   `target_dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE   table_name = 'order_items';
```



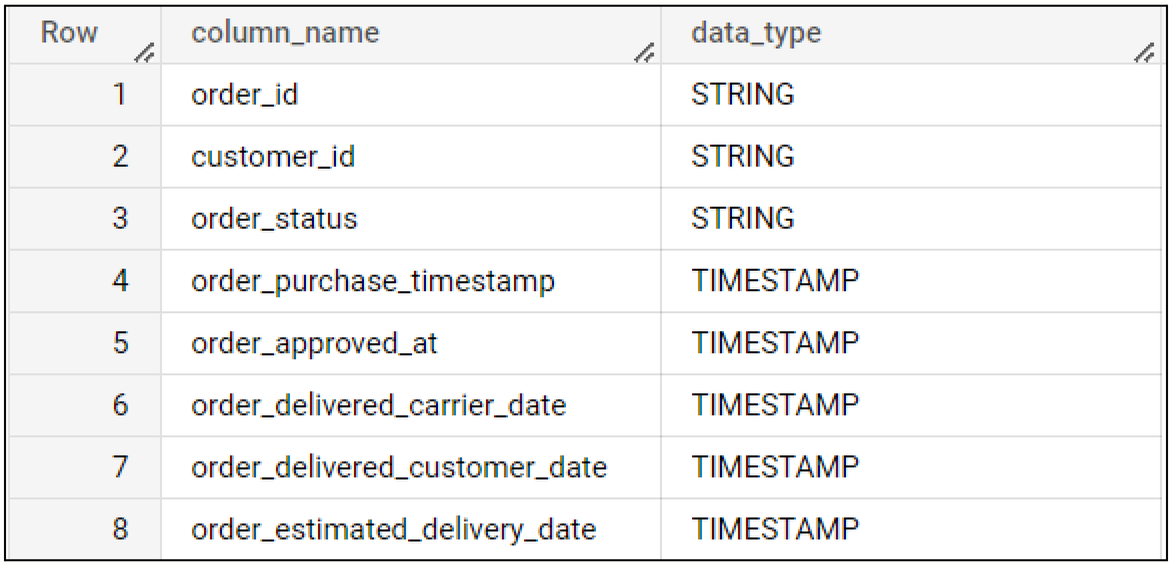
Row	column_name	data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64

```
SELECT column_name,data_type
FROM   `target_dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE   table_name = 'order_reviews';
```



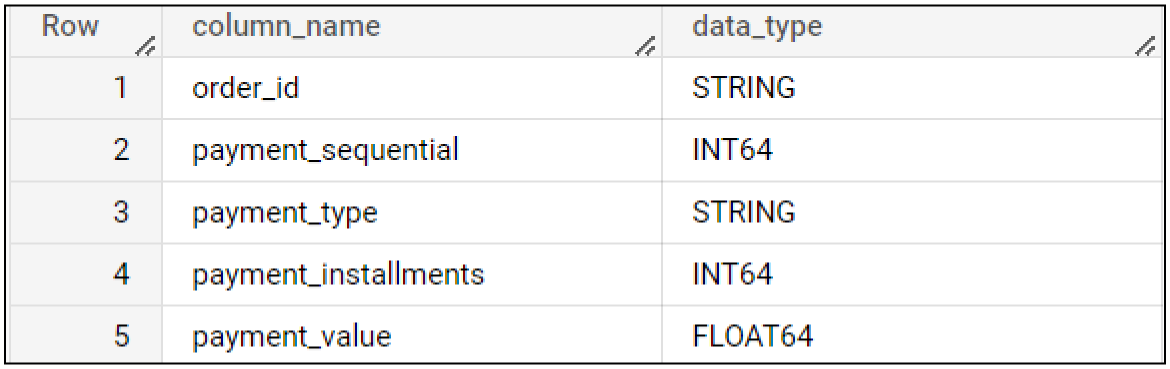
Row	column_name	data_type
1	review_id	STRING
2	order_id	STRING
3	review_score	INT64
4	review_comment_title	STRING
5	review_creation_date	TIMESTAMP
6	review_answer_timestamp	TIMESTAMP

```
SELECT column_name,data_type
FROM   `target_dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE   table_name = 'orders';
```



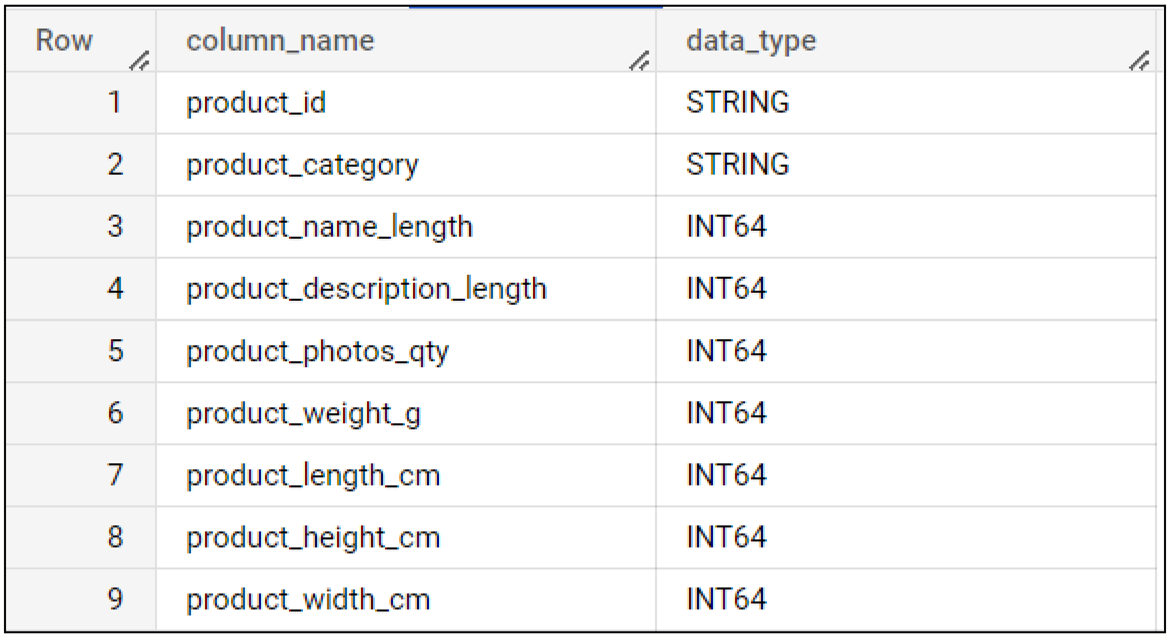
Row	column_name	data_type
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP

```
SELECT column_name,data_type
FROM   `target_dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE   table_name = ‘payments’;
```



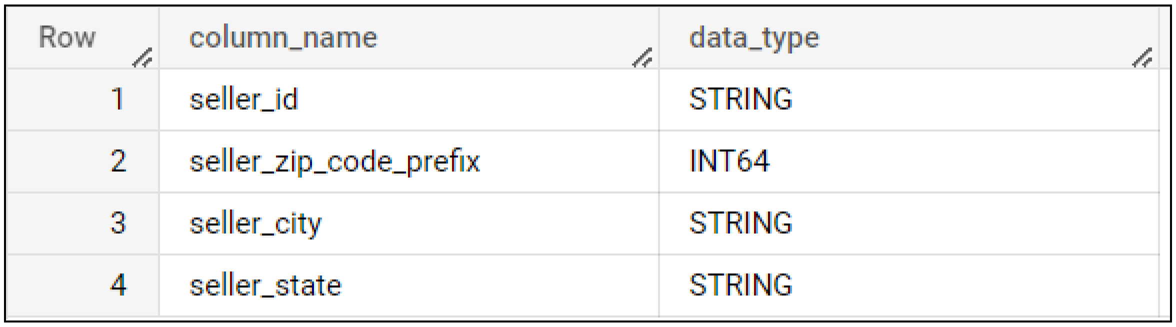
Row	column_name	data_type
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

```
SELECT column_name,data_type
FROM   `target_dataset.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

```
SELECT column_name,data_type
FROM   `target_dataset.INFORMATION_SCHEMA.COLUMNS`
WHERE   table_name = 'sellers';
```

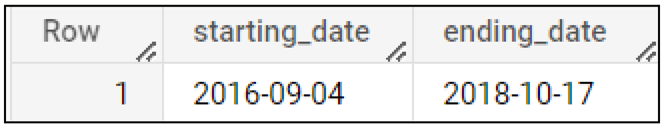


Row	column_name	data_type
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

Time period for which the data is given

The time period for the given data is from 2016 to 2018 which is mentioned in the context. Although to get the exact date we can do this by using the following query:

```
SELECT DATE(min(order_purchase_timestamp)) as starting_date,
DATE(max(order_purchase_timestamp)) ending_date
FROM `target_dataset.orders`;
```

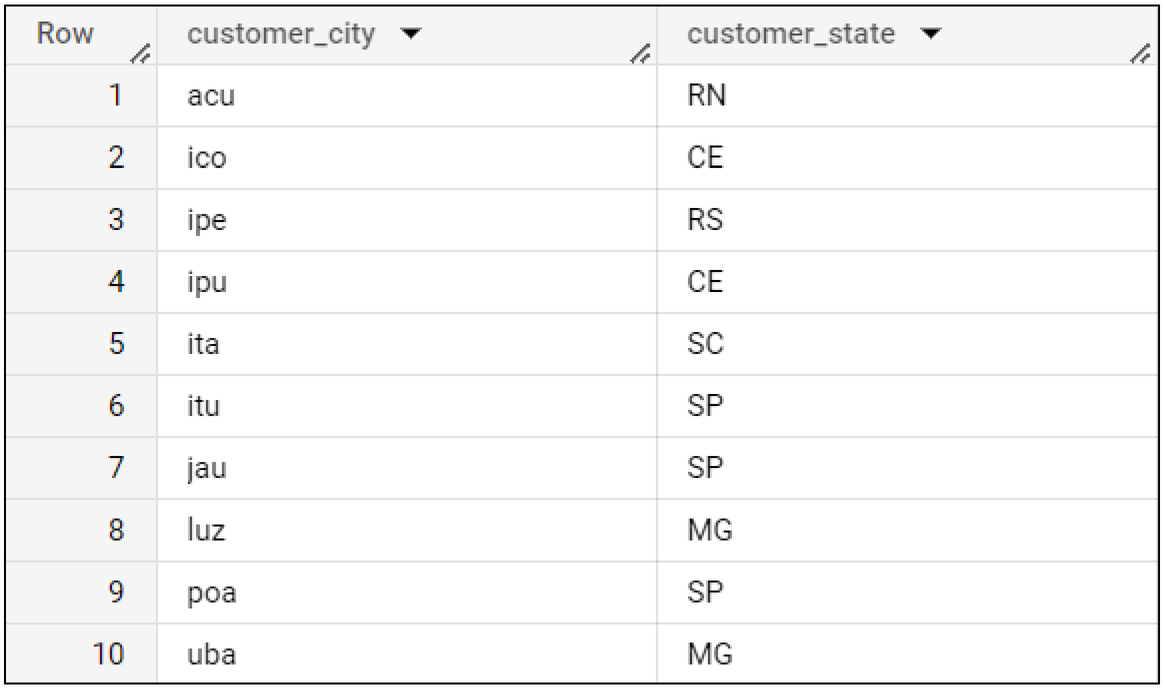


Row	starting_date	ending_date
1	2016-09-04	2018-10-17

The first order was placed on 4th September 2016 and last order was placed on 17th October 2018.

Cities and States of customers ordered during the given period

```
SELECT DISTINCT customer_city,customer_state
FROM `target_dataset.customers` c right join `target_dataset.orders` o on c.customer_id =
o.customer_id;
```



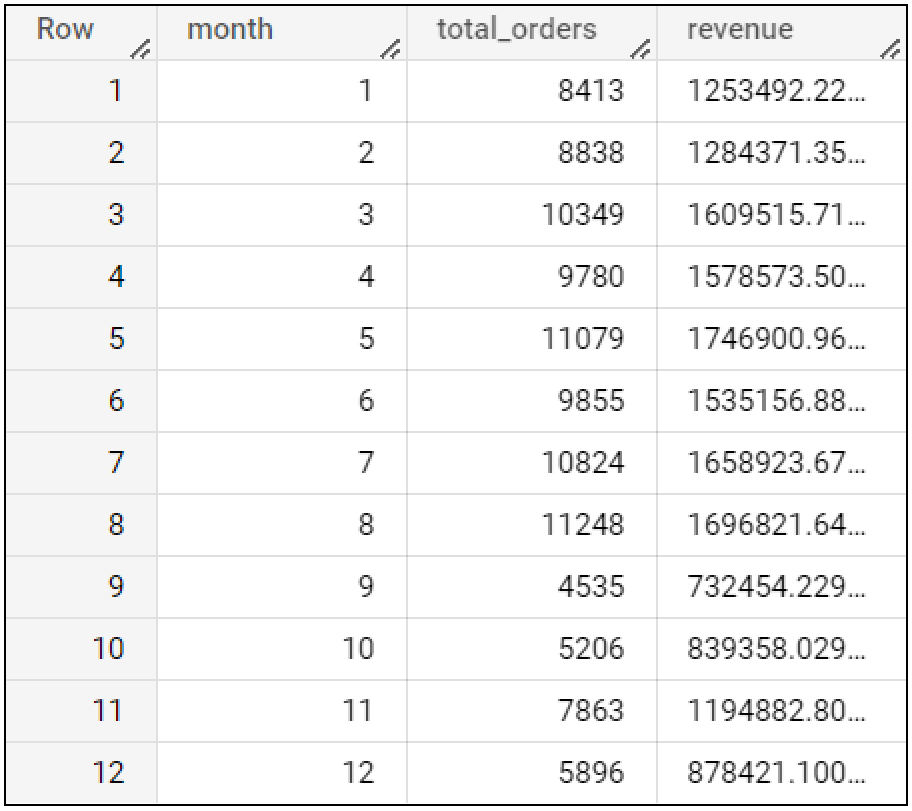
Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

```
These are some of the various different customer cities and customer states.
```

Q2 : In-depth Exploration:

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?

```
SELECT EXTRACT(month from order_purchase_timestamp) month,
COUNT(o.order_id) total_orders,
SUM(payment_value) revenue
from `target_dataset.orders` o join `target_dataset.payments` p on o.order_id = p.order_id
GROUP BY month
ORDER BY month;
```



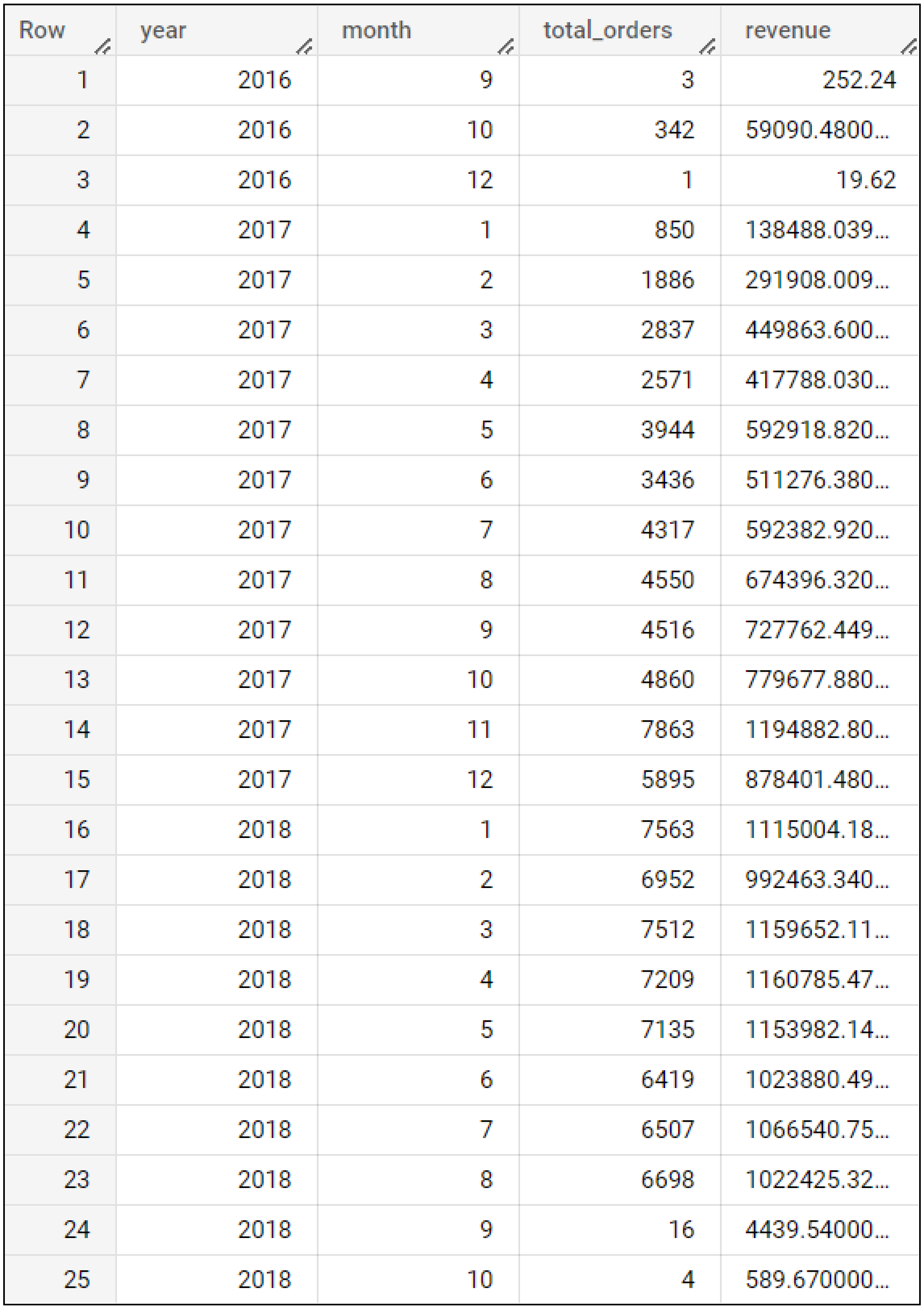
Row	month	total_orders	revenue
1	1	8413	1253492.22...
2	2	8838	1284371.35...
3	3	10349	1609515.71...
4	4	9780	1578573.50...
5	5	11079	1746900.96...
6	6	9855	1535156.88...
7	7	10824	1658923.67...
8	8	11248	1696821.64...
9	9	4535	732454.229...
10	10	5206	839358.029...
11	11	7863	1194882.80...
12	12	5896	878421.100...

On analyzing the data month wise we understand that the Number of orders and revenue is slightly increasing with slight dips till August and then we see a sharp fall to around 4500 orders in September.

To understand the data in more detail we see the number of orders year and month wise both

---

```
SELECT  EXTRACT(year from order_purchase_timestamp) year,
EXTRACT(month from order_purchase_timestamp) month,
COUNT(o.order_id) total_orders,
SUM(payment_value) revenue
from `target_dataset.orders` o join `target_dataset.payments` p on o.order_id = p.order_id
GROUP BY year, month
ORDER BY year,month;
```



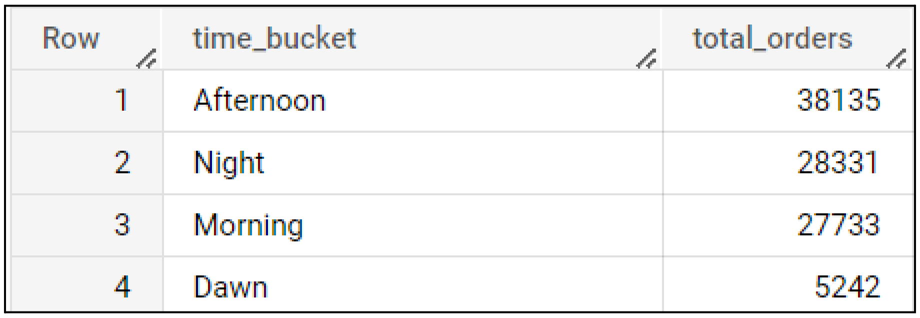
Row	year	month	total_orders	revenue
1	2016	9	3	252.24
2	2016	10	342	59090.4800...
3	2016	12	1	19.62
4	2017	1	850	138488.039...
5	2017	2	1886	291908.009...
6	2017	3	2837	449863.600...
7	2017	4	2571	417788.030...
8	2017	5	3944	592918.820...
9	2017	6	3436	511276.380...
10	2017	7	4317	592382.920...
11	2017	8	4550	674396.320...
12	2017	9	4516	727762.449...
13	2017	10	4860	779677.880...
14	2017	11	7863	1194882.80...
15	2017	12	5895	878401.480...
16	2018	1	7563	1115004.18...
17	2018	2	6952	992463.340...
18	2018	3	7512	1159652.11...
19	2018	4	7209	1160785.47...
20	2018	5	7135	1153982.14...
21	2018	6	6419	1023880.49...
22	2018	7	6507	1066540.75...
23	2018	8	6698	1022425.32...
24	2018	9	16	4439.54000...
25	2018	10	4	589.670000...

Here we can see that we see no or negligible orders in 2016 as it is the start of the company and then in 2017 we can see an increase in the number of orders and revenue. Starting 2018 we can see the order count and revenue is flat or more or less the same and after August we see that there are no more orders.

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

```
with r as(
  select order_id, EXTRACT(hour FROM order_purchase_timestamp) hour
from `target_dataset.orders`)

select case when hour between 0 and 6 then 'Dawn'
when hour between 7 and 12 then 'Morning'
when hour between 13 and 18 then 'Afternoon'
else 'Night'
end as time_bucket,
count(order_id) total_orders
from r
group by time_bucket
order by total_orders desc;
```



Row	time_bucket	total_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

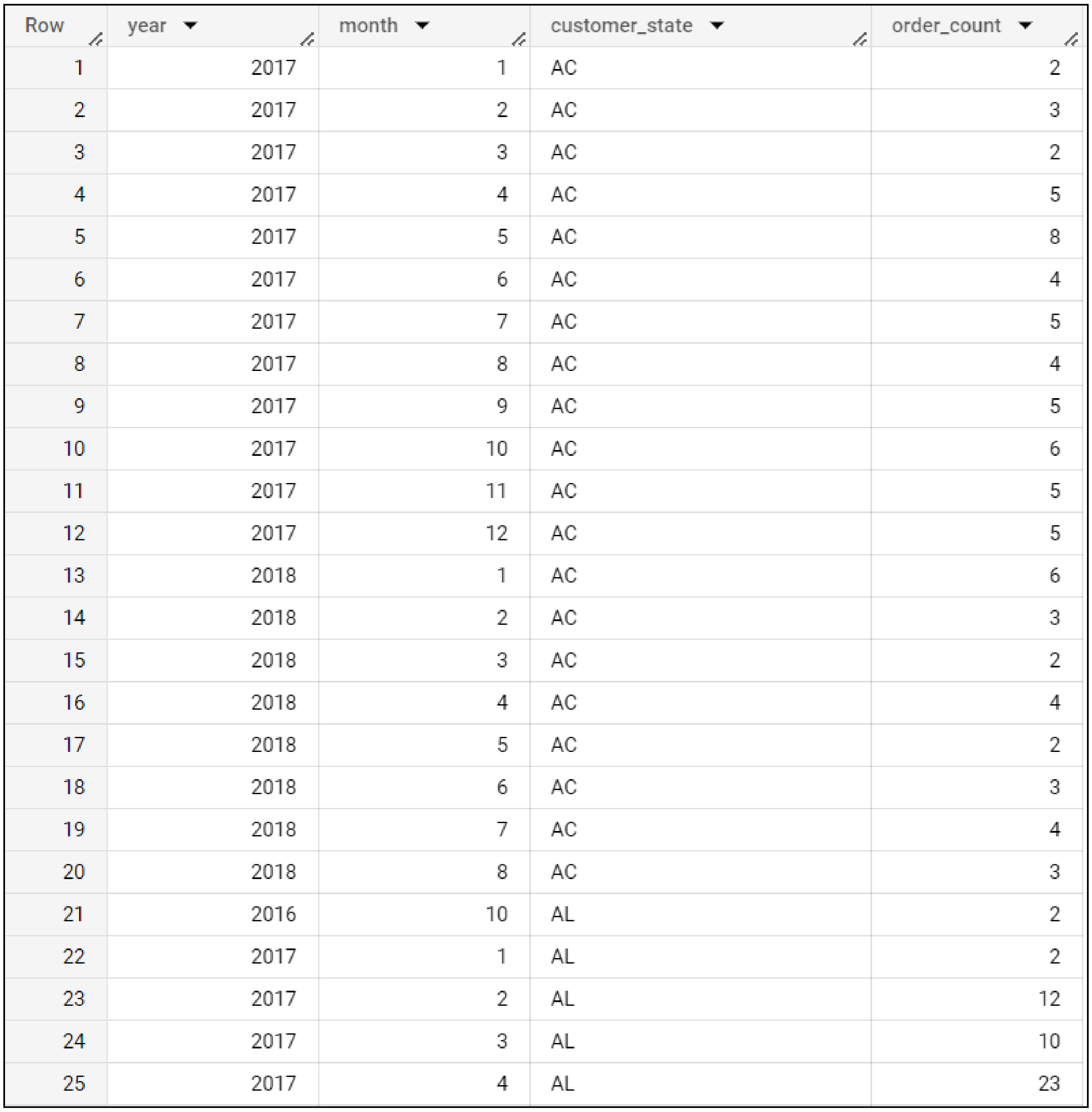
As per the data Brazilian customers tend to buy the most in the afternoon followed by night followed by morning and the least at Dawn

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

Get month on month orders by states

```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month, customer_state, COUNT(*) AS
```

```
order_count
FROM `target_dataset.orders` as o  join `target_dataset.customers` as c on  o.customer_id
= c.customer_id
GROUP BY year, month, customer_state
ORDER BY customer_state,year, month;
```

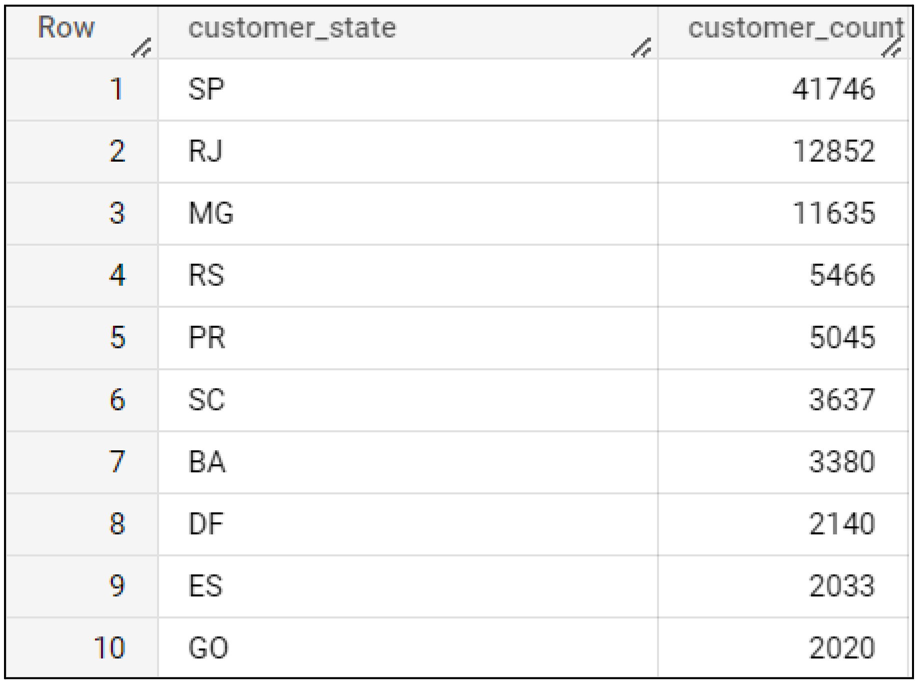


Row	year	month	customer_state	order_count
1	2017	1	AC	2
2	2017	2	AC	3
3	2017	3	AC	2
4	2017	4	AC	5
5	2017	5	AC	8
6	2017	6	AC	4
7	2017	7	AC	5
8	2017	8	AC	4
9	2017	9	AC	5
10	2017	10	AC	6
11	2017	11	AC	5
12	2017	12	AC	5
13	2018	1	AC	6
14	2018	2	AC	3
15	2018	3	AC	2
16	2018	4	AC	4
17	2018	5	AC	2
18	2018	6	AC	3
19	2018	7	AC	4
20	2018	8	AC	3
21	2016	10	AL	2
22	2017	1	AL	2
23	2017	2	AL	12
24	2017	3	AL	10
25	2017	4	AL	23

Here is list of all states along with the order count month on month by each year

Distribution of customers across the states in Brazil

```
select customer_state, count(customer_id) customer_count
from `target_dataset.customers`
group by customer_state
order by customer_count desc;
```



Row	customer_state	customer_count
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

The customer count is highest in SP followed by RJ and followed by MG

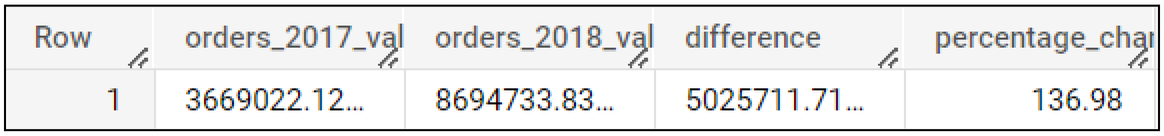
Q4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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

Considering all orders(includes canceled or returned orders)

```
with t as
(select (select sum(payment_value)
from `target_dataset.payments` p join `target_dataset.orders` o on p.order_id = o.order_id
where date(order_purchase_timestamp) between '2017-01-01' and '2017-08-31')
orders_2017_value,
(select sum(payment_value)
from `target_dataset.payments` p join `target_dataset.orders` o on p.order_id = o.order_id
where date(order_purchase_timestamp) between '2018-01-01' and '2018-08-31')
orders_2018_value)

select *, orders_2018_value-orders_2017_value difference,
round((orders_2018_value-orders_2017_value)/orders_2017_value*100,2) percentage_change
from t;
```



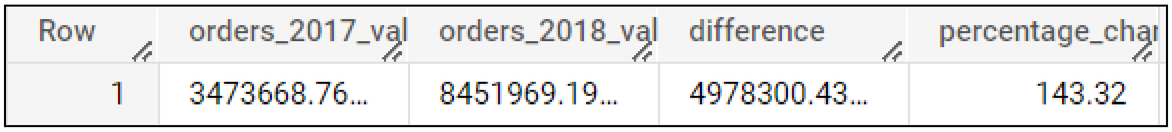
Row	orders_2017_val	orders_2018_val	difference	percentage_char
1	3669022.12...	8694733.83...	5025711.71...	136.98

The percentage change in the cost of orders from 2017 to 2018 is 136.98%

Considering only orders which were successfully delivered

```
with t as
(select (select sum(payment_value)
from `target_dataset.payments` p join `target_dataset.orders` o on p.order_id = o.order_id
where date(order_purchase_timestamp) between '2017-01-01' and '2017-08-31'
and order_delivered_customer_date is not null)  orders_2017_value,
(select sum(payment_value)
from `target_dataset.payments` p join `target_dataset.orders` o on p.order_id = o.order_id
where date(order_purchase_timestamp) between '2018-01-01' and '2018-08-31'
and order_delivered_customer_date is not null)  orders_2018_value)

select *, orders_2018_value-orders_2017_value difference,
round((orders_2018_value-orders_2017_value)/orders_2017_value*100,2) percentage_change
from t;
```



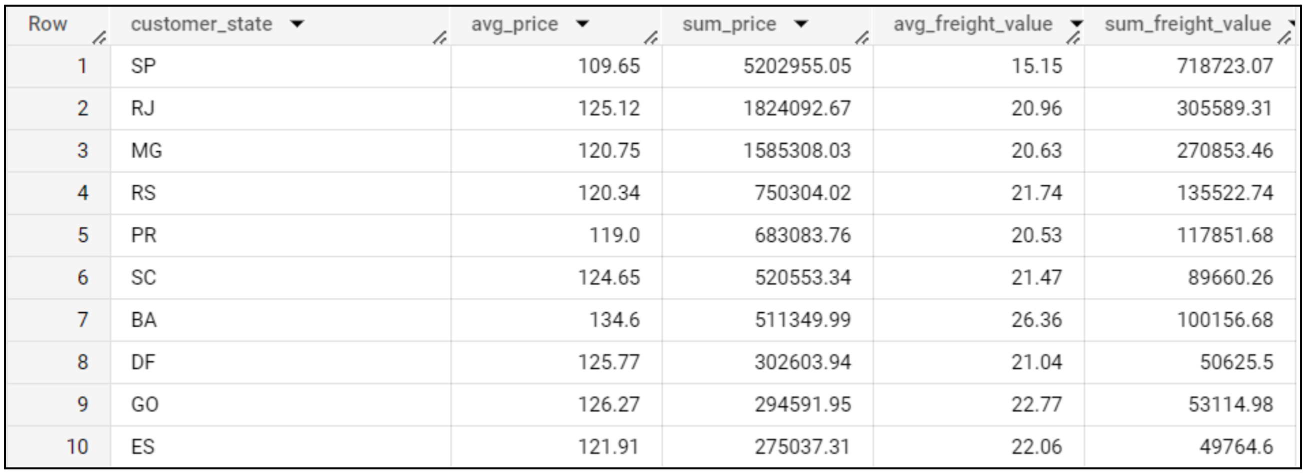
Row	orders_2017_val	orders_2018_val	difference	percentage_char
1	3473668.76...	8451969.19...	4978300.43...	143.32

The percentage change in the cost of orders which were successfully delivered from 2017 to 2018 is

143.32%

Mean & Sum of price and freight value by customer state

```
select customer_state, round(avg(price),2) avg_price, round(sum(price),2) sum_price,
round(avg(freight_value),2) avg_freight_value, round(sum(freight_value),2)
sum_freight_value
from `target_dataset.customers` c join `target_dataset.orders` o on o.customer_id =
c.customer_id
join `target_dataset.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by sum_price desc, sum_freight_value desc;
```



Row	customer_state	avg_price	sum_price	avg_freight_value	sum_freight_value
1	SP	109.65	5202955.05	15.15	718723.07
2	RJ	125.12	1824092.67	20.96	305589.31
3	MG	120.75	1585308.03	20.63	270853.46
4	RS	120.34	750304.02	21.74	135522.74
5	PR	119.0	683083.76	20.53	117851.68
6	SC	124.65	520553.34	21.47	89660.26
7	BA	134.6	511349.99	26.36	100156.68
8	DF	125.77	302603.94	21.04	50625.5
9	GO	126.27	294591.95	22.77	53114.98
10	ES	121.91	275037.31	22.06	49764.6

Here is a list of mean and sum of freight value of each state

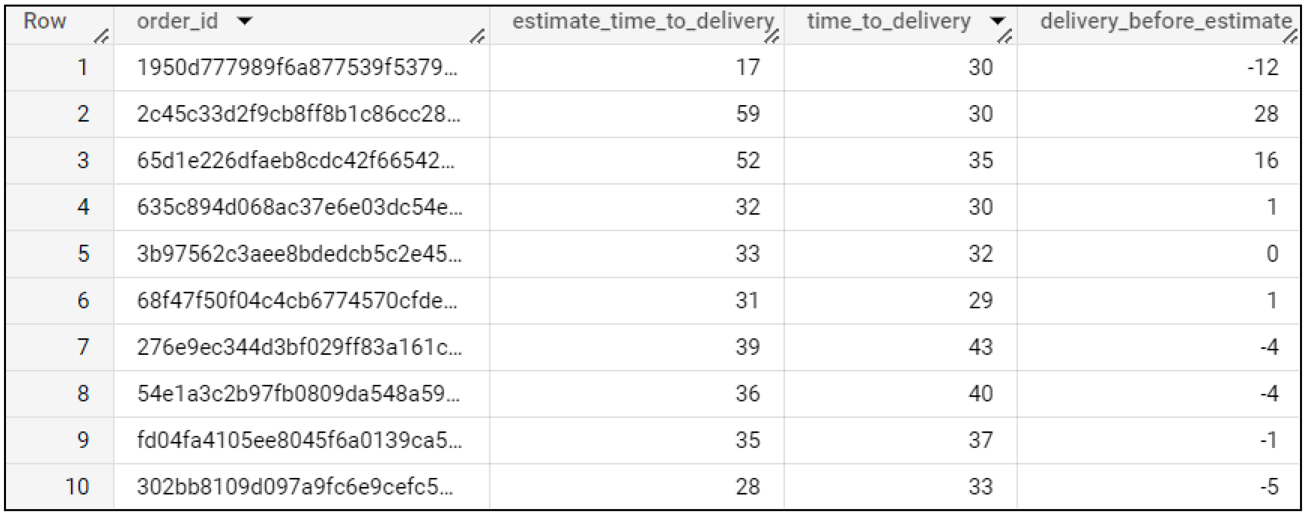
---

Q5. Analysis on sales, freight and delivery time

Calculate days between purchasing, delivering and estimated delivery

```
select order_id, date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)
estimate_time_to_delivery,
date_diff(order_delivered_customer_date,order_purchase_timestamp,day) time_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)
delivery_before_estimate
```

```
from `target_dataset.orders`
where order_delivered_customer_date is not null;
```



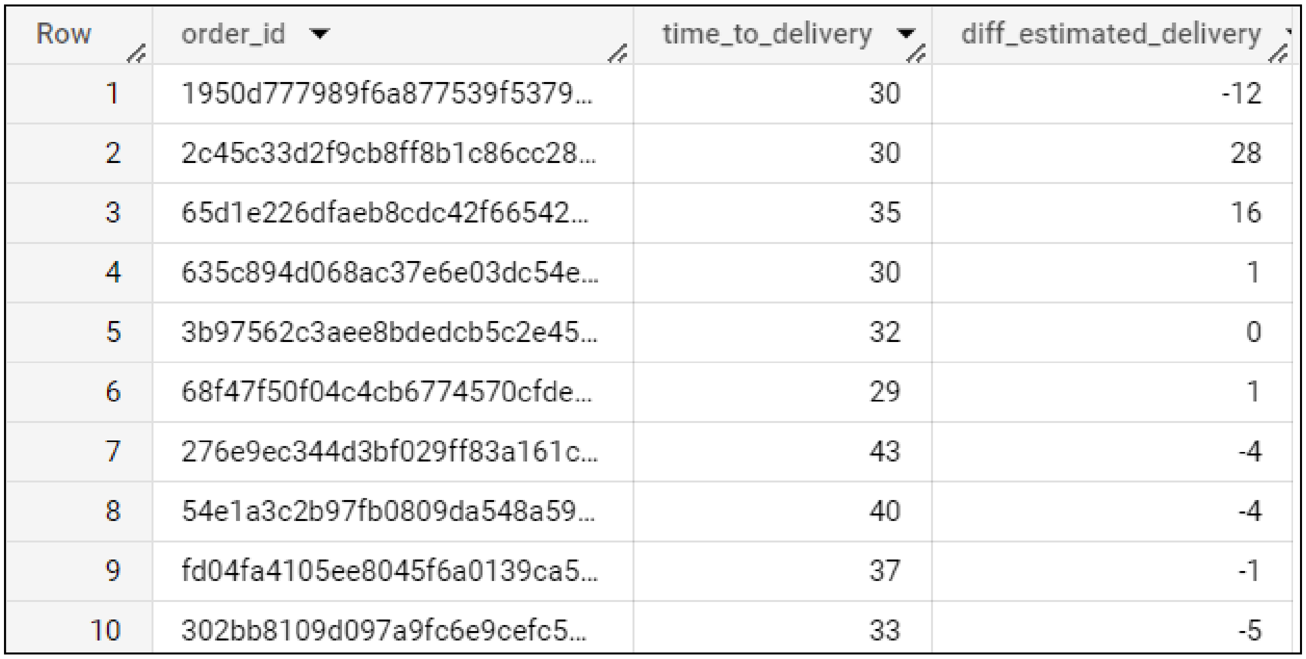
Row	order_id	estimate_time_to_delivery	time_to_delivery	delivery_before_estimate
1	1950d777989f6a877539f5379...	17	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	59	30	28
3	65d1e226dfaeb8cdc42f66542...	52	35	16
4	635c894d068ac37e6e03dc54e...	32	30	1
5	3b97562c3aee8bdedcb5c2e45...	33	32	0
6	68f47f50f04c4cb6774570cfde...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

The first column estimate_time_to_delivery shows the estimated number of days to delivery when the order was placed
The second column time_to_delivery represents the actual number of days in which the order was placed after being ordered
The third column delivery_before_estimate shows how many days before the estimated date the order was delivered. For example, 10 means the order was delivered 10 days prior than the estimated delivery date. -10 means that the order was delivered 10 days after the estimated delivery date

Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- time_to_delivery = order_delivered_customer_date-order_purchase_timestamp
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
select order_id, date_diff(order_delivered_customer_date,order_purchase_timestamp,day)
time_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)
diff_estimated_delivery
from `target_dataset.orders`
where order_delivered_customer_date is not null;
```

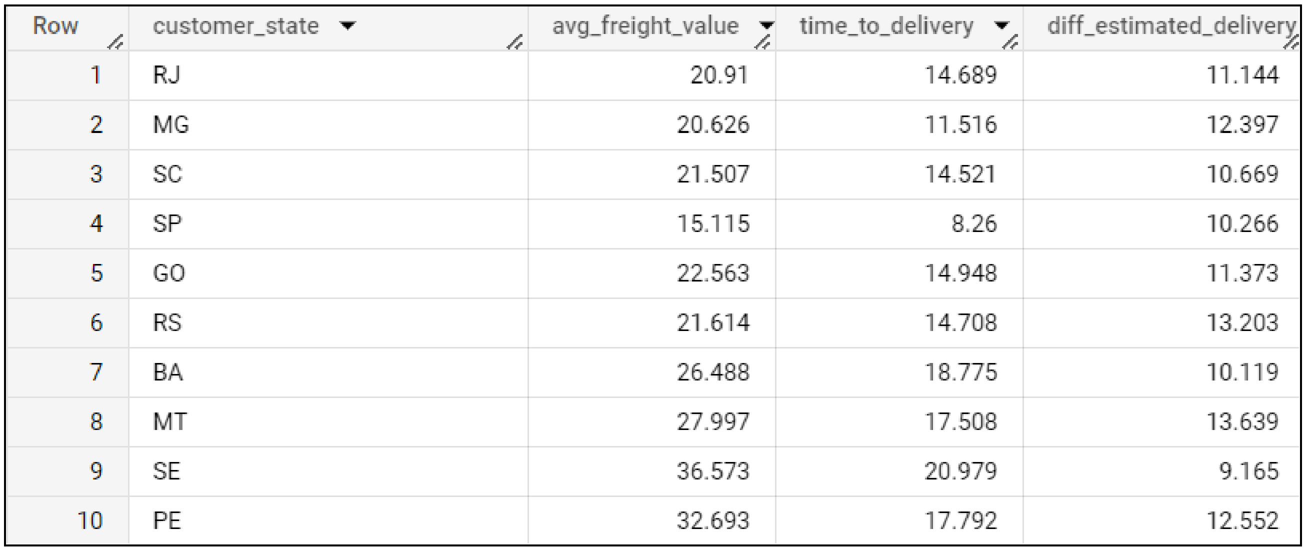


Row	order_id	time_to_delivery	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...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

```
This data is calculated only using the orders which were successfully delivered
as orders which were not delivered have null values in delivered date column
```

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

```
select customer_state, round(avg(freight_value),3) avg_freight_value,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),3)
time_to_delivery,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),3)
diff_estimated_delivery
from `target_dataset.orders` o join `target_dataset.customers` c on c.customer_id =
o.customer_id
join `target_dataset.order_items` oi on oi.order_id = o.order_id
where order_delivered_customer_date is not null
group by customer_state;
```



Row	customer_state	avg_freight_value	time_to_delivery	diff_estimated_delivery
1	RJ	20.91	14.689	11.144
2	MG	20.626	11.516	12.397
3	SC	21.507	14.521	10.669
4	SP	15.115	8.26	10.266
5	GO	22.563	14.948	11.373
6	RS	21.614	14.708	13.203
7	BA	26.488	18.775	10.119
8	MT	27.997	17.508	13.639
9	SE	36.573	20.979	9.165
10	PE	32.693	17.792	12.552

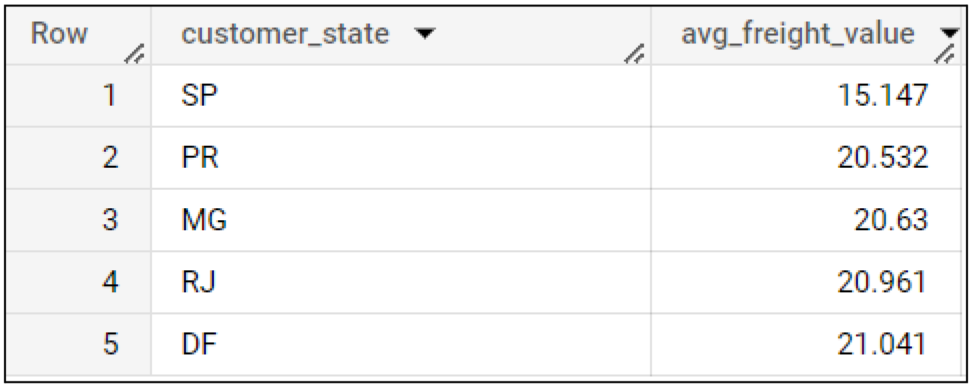
Orders only which were delivered are considered

Sort the data to get the following:

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

Ascending

```
select customer_state, round(avg(freight_value),3) avg_freight_value,
from `target_dataset.orders` o join `target_dataset.customers` c on c.customer_id =
o.customer_id
join `target_dataset.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by avg_freight_value
limit 5;
```

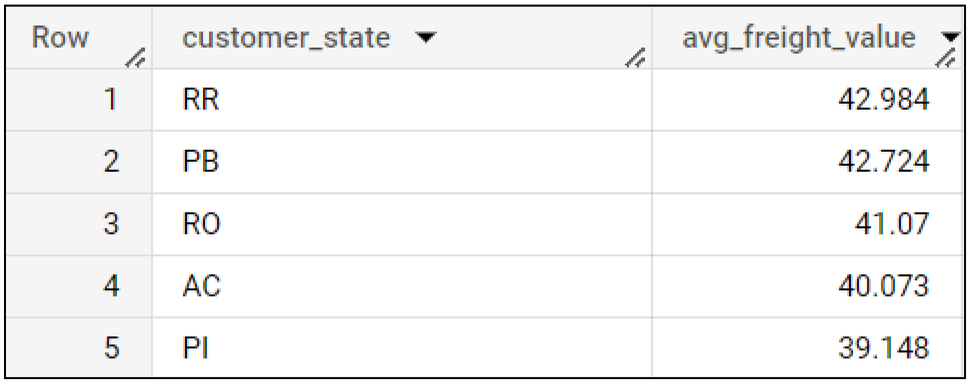


Row	customer_state	avg_freight_value
1	SP	15.147
2	PR	20.532
3	MG	20.63
4	RJ	20.961
5	DF	21.041

```
SP has the lowest mean freight value per order followed by PR and MG
```

Descending

```
select customer_state, round(avg(freight_value),3) avg_freight_value,
from `target_dataset.orders` o join `target_dataset.customers` c on c.customer_id =
o.customer_id
join `target_dataset.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by avg_freight_value desc
limit 5;
```



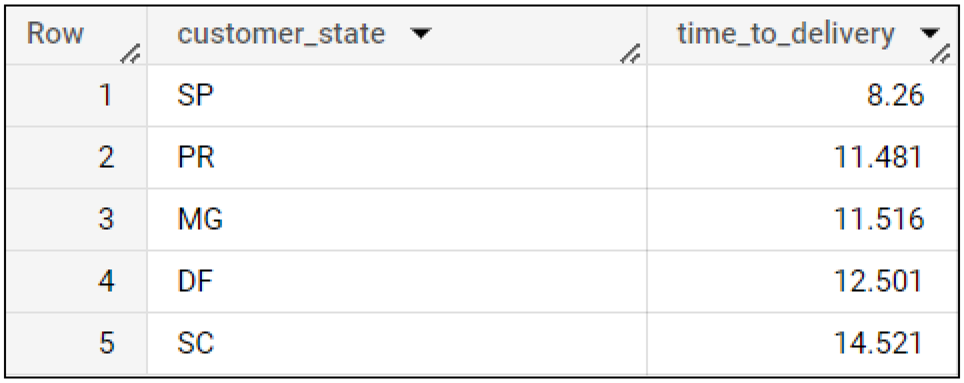
Row	customer_state	avg_freight_value
1	RR	42.984
2	PB	42.724
3	RO	41.07
4	AC	40.073
5	PI	39.148

```
RR has the highest mean freight value per order followed by PB and RO
```

Top 5 states with highest/lowest average time to delivery

Ascending

```
select customer_state,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),3)
time_to_delivery
from `target_dataset.orders` o join `target_dataset.customers` c on c.customer_id =
o.customer_id
join `target_dataset.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by time_to_delivery
limit 5;
```

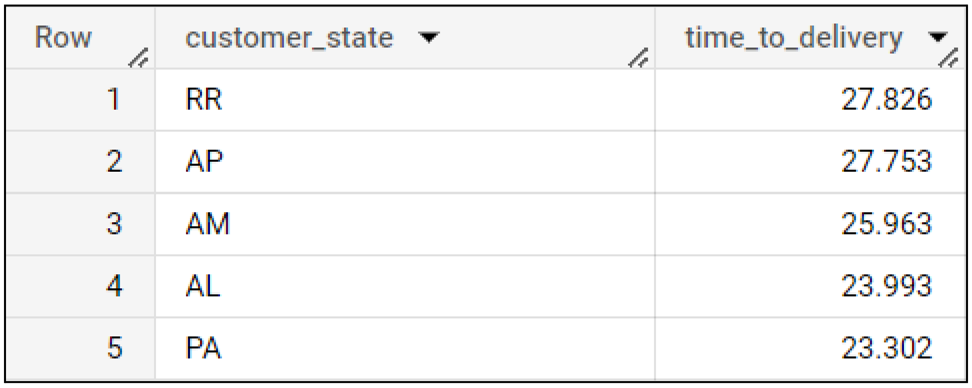


Row	customer_state	time_to_delivery
1	SP	8.26
2	PR	11.481
3	MG	11.516
4	DF	12.501
5	SC	14.521

`SP has the lowest time to delivery per order followed by PR and MG`

Descending

```
select customer_state,
round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)),3)
time_to_delivery
from `target_dataset.orders` o join `target_dataset.customers` c on c.customer_id =
o.customer_id
join `target_dataset.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by time_to_delivery desc
limit 5;
```



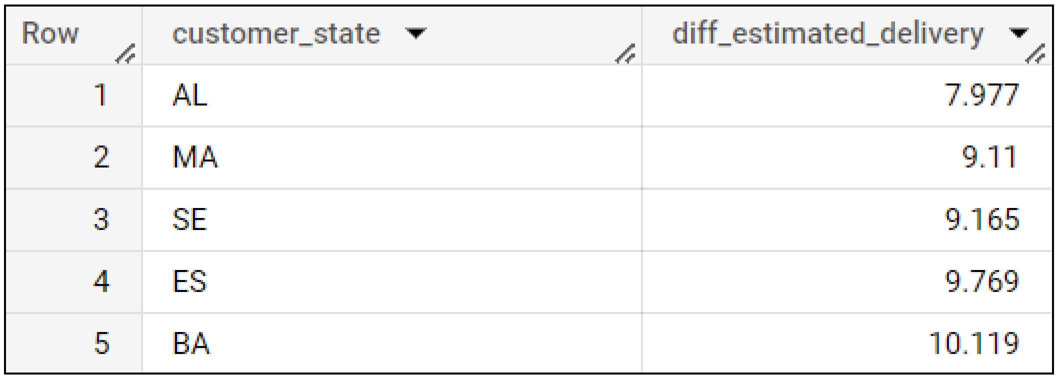
Row	customer_state	time_to_delivery
1	RR	27.826
2	AP	27.753
3	AM	25.963
4	AL	23.993
5	PA	23.302

`RR has the highest mean time to delivery per order followed by AP and AM`

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

States where delivery is not fast compared to estimated delivery date

```
select customer_state,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),3)
diff_estimated_delivery
from `target_dataset.orders` o join `target_dataset.customers` c on c.customer_id =
o.customer_id
join `target_dataset.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by diff_estimated_delivery
limit 5;
```

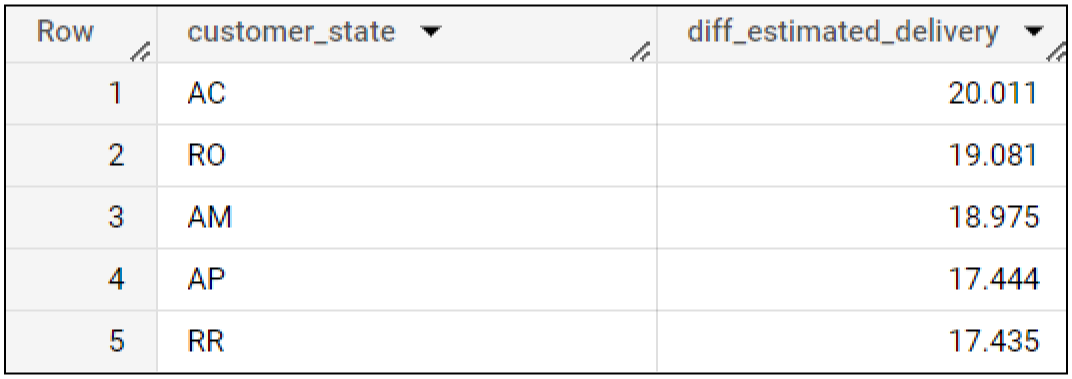


Row	customer_state	diff_estimated_delivery
1	AL	7.977
2	MA	9.11
3	SE	9.165
4	ES	9.769
5	BA	10.119

```
AL delivers the orders slowest followed by MA and SE
```

States where delivery is very fast compared to estimated delivery date

```
select customer_state,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),3)
diff_estimated_delivery
from `target_dataset.orders` o join `target_dataset.customers` c on c.customer_id =
o.customer_id
join `target_dataset.order_items` oi on oi.order_id = o.order_id
group by customer_state
order by diff_estimated_delivery desc
limit 5;
```



Row	customer_state	diff_estimated_delivery
1	AC	20.011
2	RO	19.081
3	AM	18.975
4	AP	17.444
5	RR	17.435

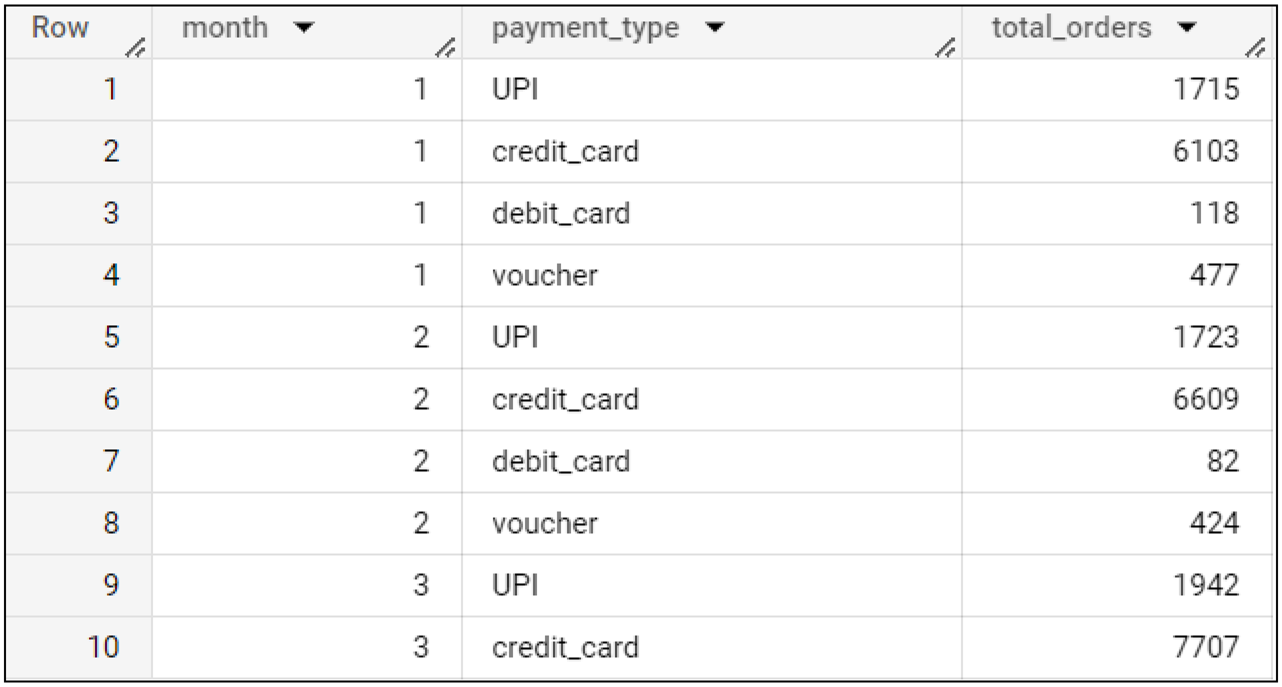
```
AC delivers the orders fastest followed by RO and AM
```

Q6. Payment type analysis

Month over Month count of orders for different payment types

Month wise

```
select
extract(month from order_purchase_timestamp) month,payment_type, count(o.order_id)
total_orders
from `target_dataset.payments` p join `target_dataset.orders` o on o.order_id = p.order_id
group by payment_type, month
order by month,payment_type;
```



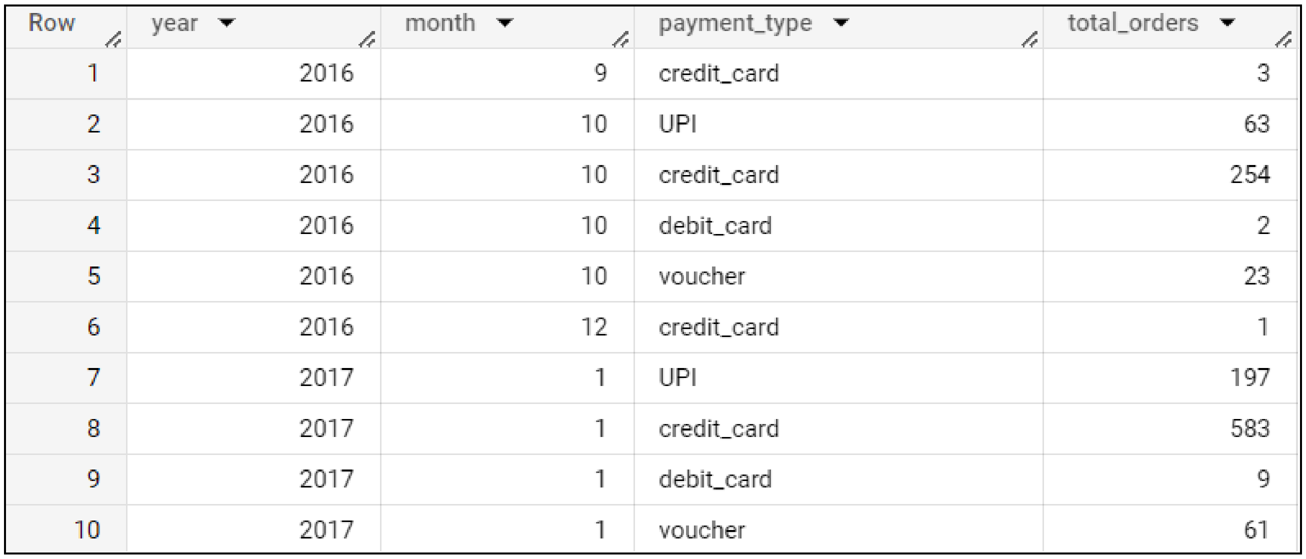
Row	month	payment_type	total_orders
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707

Every month usually the orders placed are highest with credit cards and least with debit cards. To verify it in detail we can check the order history of every month of each year

---

Year and month wise analysis

```
select extract(year from order_purchase_timestamp) year,
extract(month from order_purchase_timestamp) month,payment_type, count(o.order_id)
total_orders
from `target_dataset.payments` p join `target_dataset.orders` o on o.order_id = p.order_id
group by payment_type, year, month
order by year, month, payment_type;
```

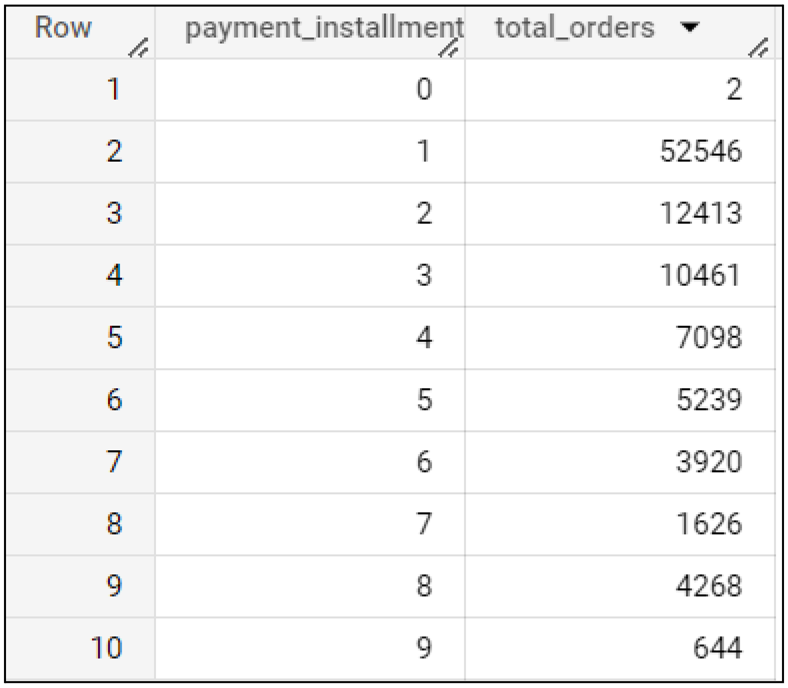


Row	year	month	payment_type	total_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61

This data also depicts that orders made with credit cards are the highest and usually lowest with debit cards

Count of orders based on the no. of payment installments

```
select payment_installments, count(order_id) total_orders
from `target_dataset.payments`
group by payment_installments;
```



Row	payment_installment	total_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

People usually tend to prefer to pay one time followed by in 2 month and 3 month emi installments

---

Q7. Actionable Insights

1. The overall data lies between 4th September 2016 and 17th October 2018.
2. Maximum orders were placed in the afternoon and the least were placed in Dawn.
3. SP has the highest number of customers followed by RJ and MG.
4. SP, PR and MG have the lowest freight charges and the time taken from ordering to delivery is also the fastest.
5. RR has the highest freight charges and time taken to delivery is also the highest.
6. Most people prefer to pay one time followed by 2 and 3 month installments.
7. Credit cards are the most preferred mode of payment for people.
8. Year 2017 had the best growth trend in both the number of orders and revenue per month.
9. The sales and number of orders in the first 8 months of 2018 was almost consistent and then we could see a sharp decline to almost no orders from the 9th month.
10. Observed 136.98% increase in cost of orders from 2017 and 2018 and 143.32% increase in cost of orders which were successfully delivered from.

Q8.Recommendations

1. In the majority of states and orders the order is delivered way before the estimated delivery date. The company should provide a better estimated delivery late which may help them to get more orders and improve their customer experience.
2. People paying in one time or fewer installments should be rewarded with cashbacks or coupon discounts of different products which may increase sales.
3. Very few people place orders at dawn. To increase this company should rollout offers and

discounts more at this time.

4. As from the data we can see that lesspeople are paying in emis. So to increase the count of emi orders, companies should give more offers and discounts for emi payments.
5. The company should give free shipment (or delivery) and add up the freight charges in the price of the item itself without the knowledge of the customer. This is a psychological proven fact that consumers tend to buy items with free delivery.
6. When the delivery is delayed than the estimated delivery date then the company should compensate the consumer at least with freight charges.
7. In states where we observe high freight charges there a cheaper delivery alternative should be used to reduce cost and increase revenue.
8. More tie ups with banks for loans and emi should be made so that a customer can get better payment options.
9. States generating negative cash flows or least profit should be focused more.
10. Rewards should be capped on spending a certain amount for the customer.