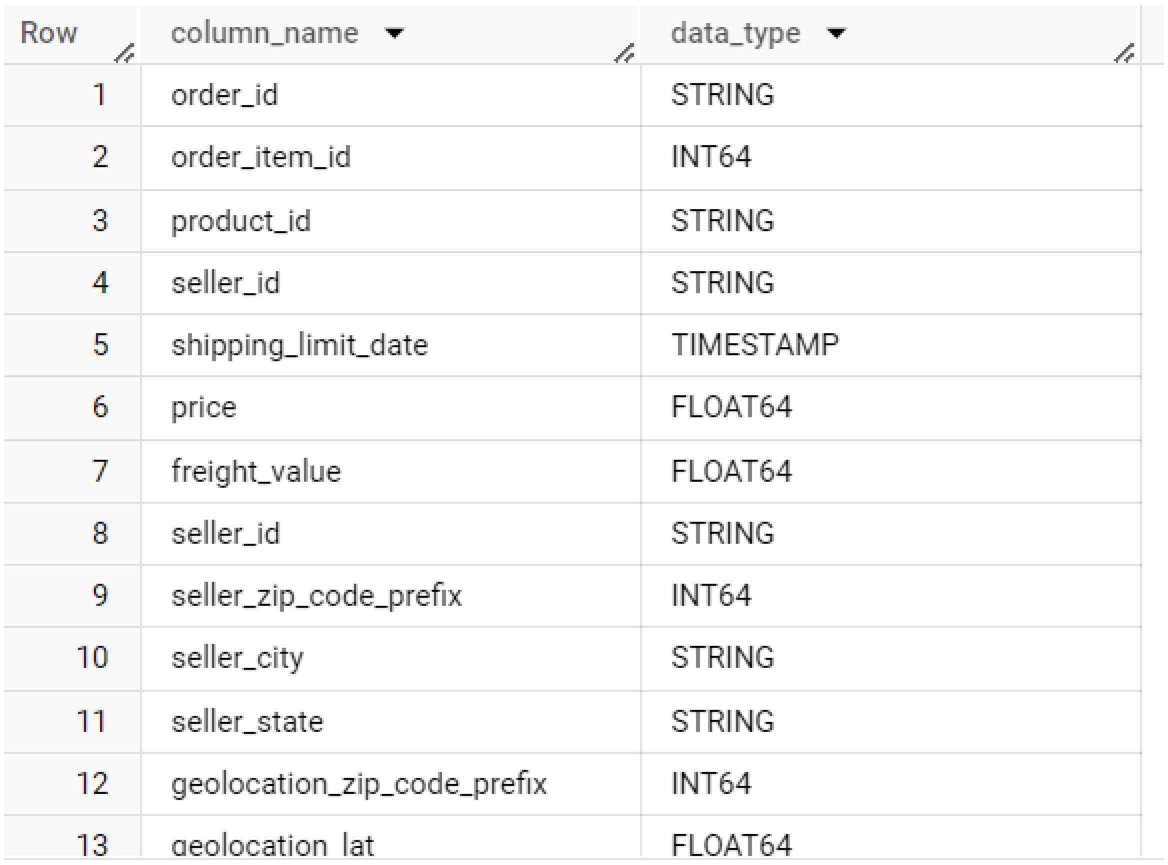
What does 'good' look like?

1. **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.

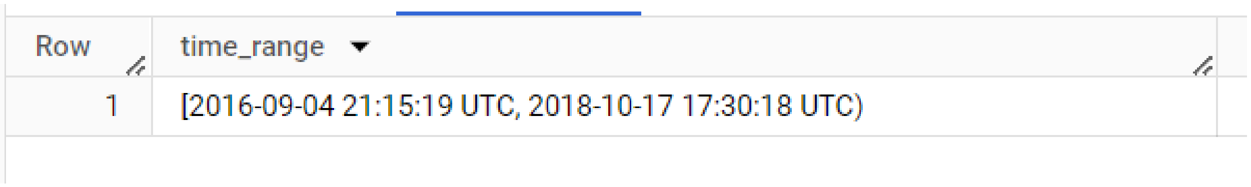
```
SELECT column_name , data_type
FROM
 `geometric-rex-415011.target.INFORMATION_SCHEMA.COLUMNS`
```



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
8	seller_id	STRING
9	seller_zip_code_prefix	INT64
10	seller_city	STRING
11	seller_state	STRING
12	geolocation_zip_code_prefix	INT64
13	geolocation_lat	FLOAT64

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

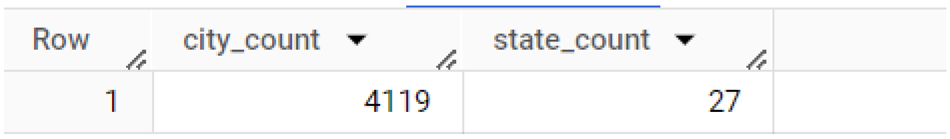
```
SELECT
       RANGE(MIN(order_purchase_timestamp),MAX(order_purchas
e_timestamp)) AS time_range
FROM `target.orders`
```



Row	time_range
1	[2016-09-04 21:15:19 UTC, 2018-10-17 17:30:18 UTC)

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

```
WITH min_max_dates AS (
  SELECT
    MIN(order_purchase_timestamp) AS min_timestamp,
    MAX(order_purchase_timestamp) AS max_timestamp
  FROM
    `target.orders`
)
SELECT
  COUNT(DISTINCT c.customer_city) AS city_count,
  COUNT(DISTINCT c.customer_state) AS state_count
FROM
  `target.orders` o
INNER JOIN
  `target.customers` c
ON
  o.customer_id = c.customer_id
CROSS JOIN
  min_max_dates
WHERE
  o.order_purchase_timestamp BETWEEN min_timestamp AND
max_timestamp;
```

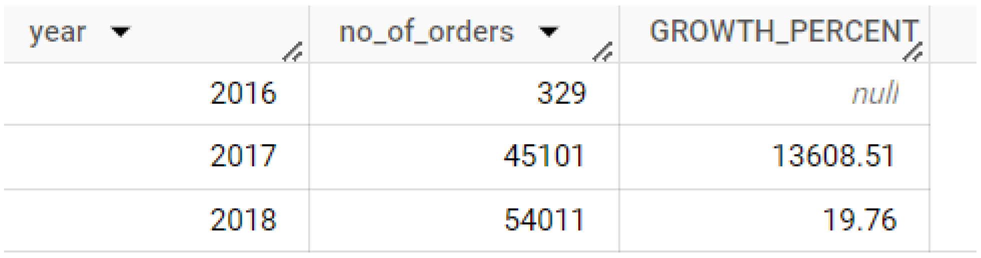


Row	city_count	state_count
1	4119	27

2. In-depth Exploration:

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

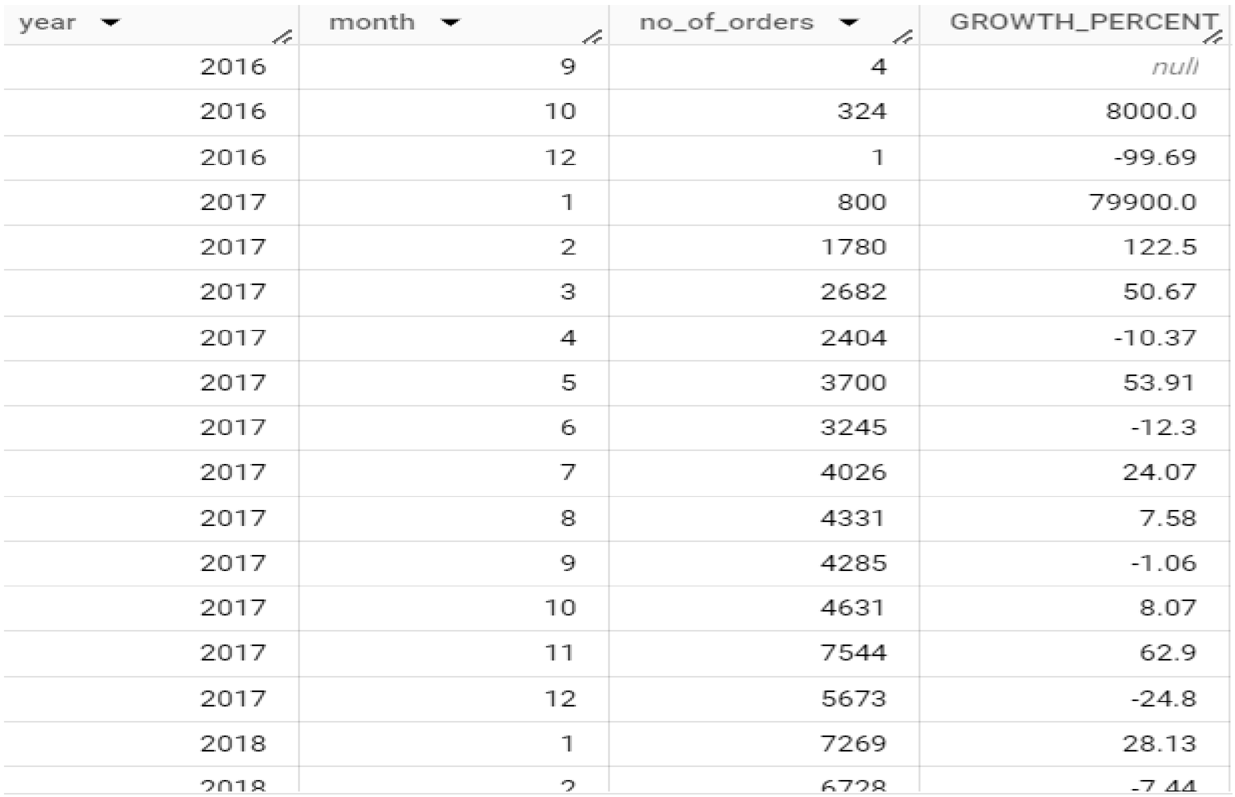
```
WITH yearly as(

SELECT EXTRACT(YEAR FROM order_purchase_timestamp) as
year,  COUNT(order_id) as no_of_orders
FROM `target.orders`
GROUP BY 1
 ORDER BY 1
)
SELECT yearly.year,yearly.no_of_orders, round((no_of_orders
- LAG(yearly.no_of_orders)OVER(order by
yearly.YEAR))/LAG(yearly.no_of_orders)OVER(order by
yearly.YEAR) *100,2) as GROWTH_PERCENT
FROM yearly
ORDER BY 1
```



year	no_of_orders	GROWTH_PERCENT
2016	329	*null*
2017	45101	13608.51
2018	54011	19.76

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

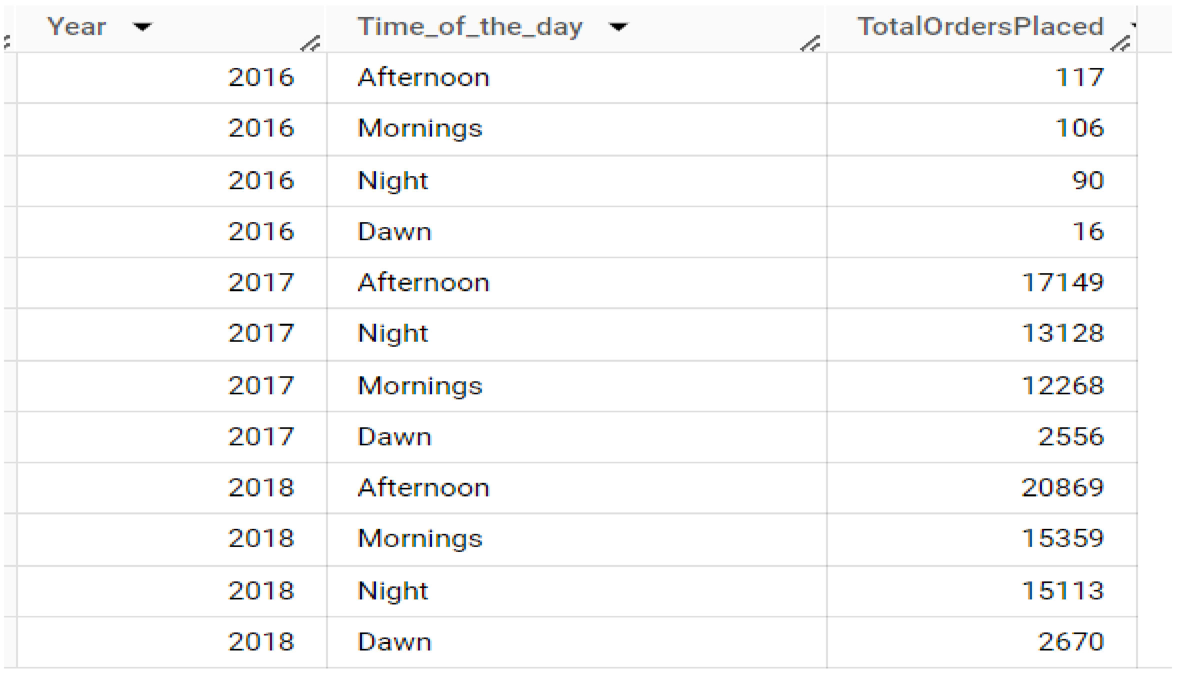
```
WITH yearly as(
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) as year,
EXTRACT(MONTH FROM order_purchase_timestamp) as month ,
COUNT(order_id) as no_of_orders
FROM `target.orders`
GROUP BY 1,2
ORDER BY 1,2
)

SELECT yearly.year,yearly.month,yearly.no_of_orders,
ROUND((no_of_orders - LAG(yearly.no_of_orders)OVER(order by
yearly.YEAR,yearly.month))/LAG(yearly.no_of_orders)OVER(orde
r by yearly.YEAR,yearly.month) *100,2) as GROWTH_PERCENT
FROM yearly
ORDER BY 1,2;
```



year	month	no_of_orders	GROWTH_PERCENT
2016	9	4	*null*
2016	10	324	8000.0
2016	12	1	-99.69
2017	1	800	79900.0
2017	2	1780	122.5
2017	3	2682	50.67
2017	4	2404	-10.37
2017	5	3700	53.91
2017	6	3245	-12.3
2017	7	4026	24.07
2017	8	4331	7.58
2017	9	4285	-1.06
2017	10	4631	8.07
2017	11	7544	62.9
2017	12	5673	-24.8
2018	1	7269	28.13
2018	2	6728	-7.44

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

```
SELECT
      EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
  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_the_day,
  COUNT(*) AS TotalOrdersPlaced
FROM
      `target.orders`
GROUP BY
      Year, Time_of_the_day
ORDER BY
      Year ASC, TotalOrdersPlaced DESC
```

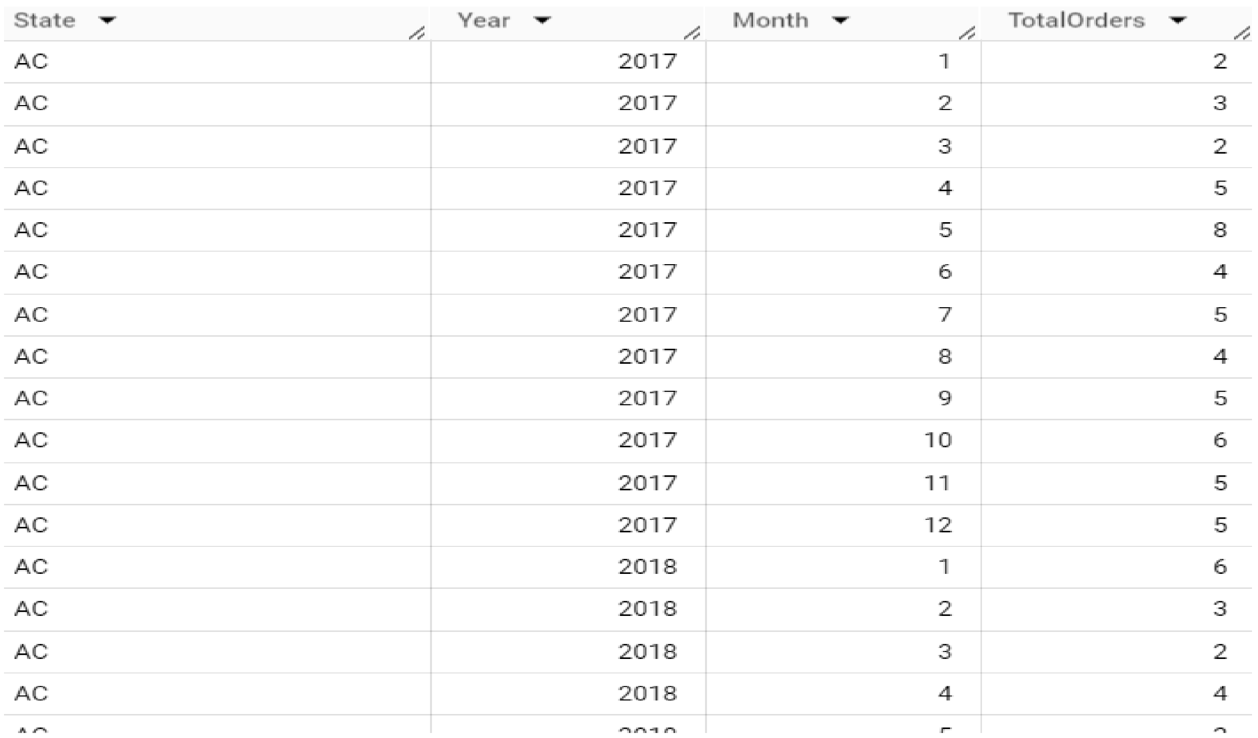


Year	Time_of_the_day	TotalOrdersPlaced
2016	Afternoon	117
2016	Mornings	106
2016	Night	90
2016	Dawn	16
2017	Afternoon	17149
2017	Night	13128
2017	Mornings	12268
2017	Dawn	2556
2018	Afternoon	20869
2018	Mornings	15359
2018	Night	15113
2018	Dawn	2670

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

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

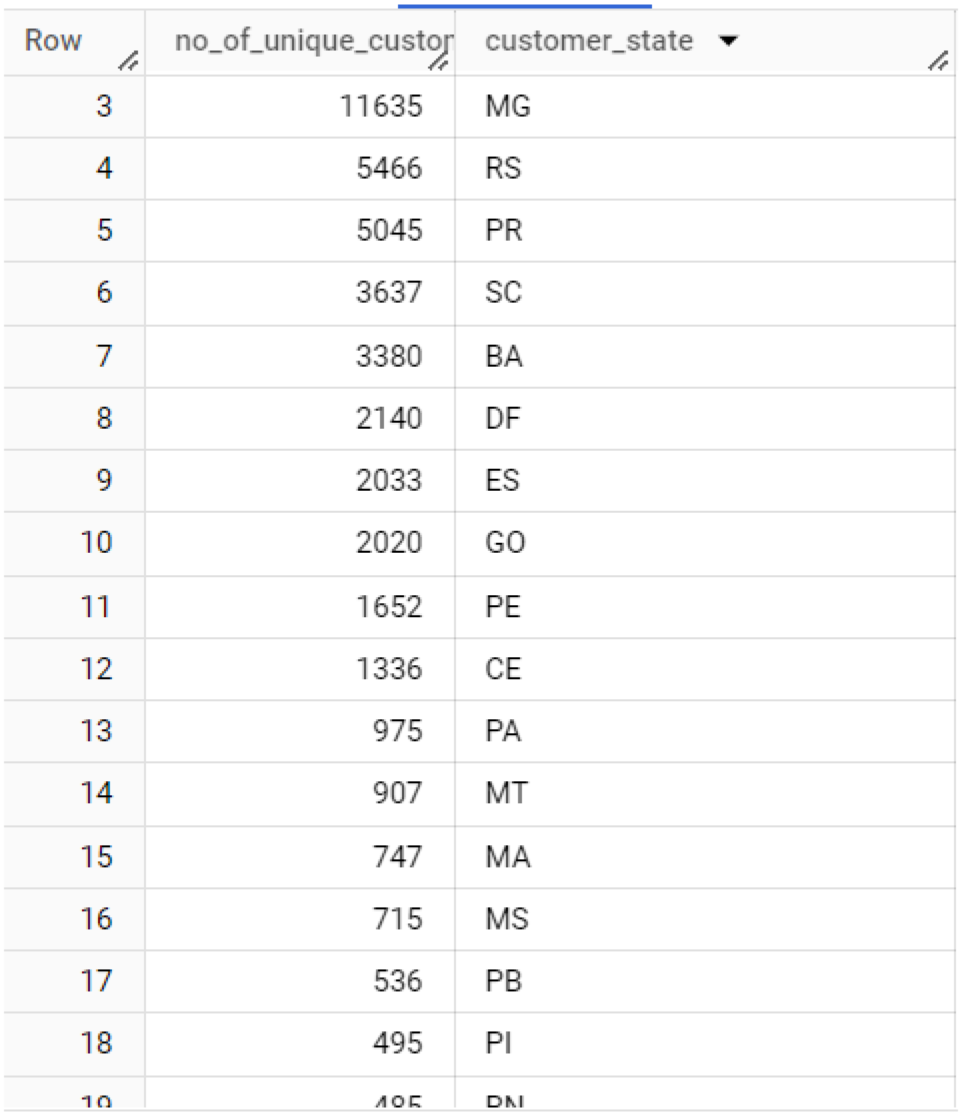
```sql
SELECT
      C.customer_state AS State,
      EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
      EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
      COUNT(*) AS TotalOrders
FROM
    target.orders AS O
INNER JOIN
   target.customers AS C
ON
    O.customer_id = C.customer_id
GROUP BY
    1, 2, 3
ORDER BY
    1, 2, 3
```



State	Year	Month	TotalOrders
AC	2017	1	2
AC	2017	2	3
AC	2017	3	2
AC	2017	4	5
AC	2017	5	8
AC	2017	6	4
AC	2017	7	5
AC	2017	8	4
AC	2017	9	5
AC	2017	10	6
AC	2017	11	5
AC	2017	12	5
AC	2018	1	6
AC	2018	2	3
AC	2018	3	2
AC	2018	4	4
AC	2018	5	[illegible]

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

```
SELECT COUNT(DISTINCT customer_id) AS no_of_unique_customers
, customer_state
FROM
 `target.customers`
GROUP BY  2
ORDER BY 1 DESC;
```

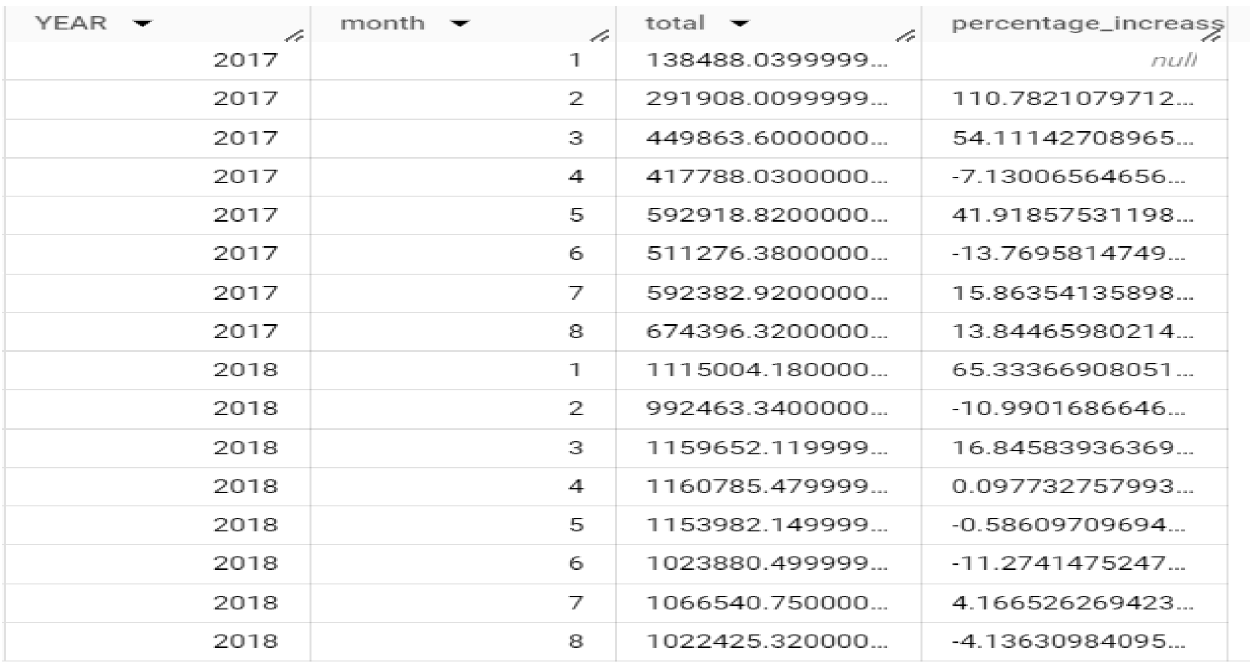


Row	no_of_unique_custor	customer_state
3	11635	MG
4	5466	RS
5	5045	PR
6	3637	SC
7	3380	BA
8	2140	DF
9	2033	ES
10	2020	GO
11	1652	PE
12	1336	CE
13	975	PA
14	907	MT
15	747	MA
16	715	MS
17	536	PB
18	495	PI
19	485	RN

4. 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).

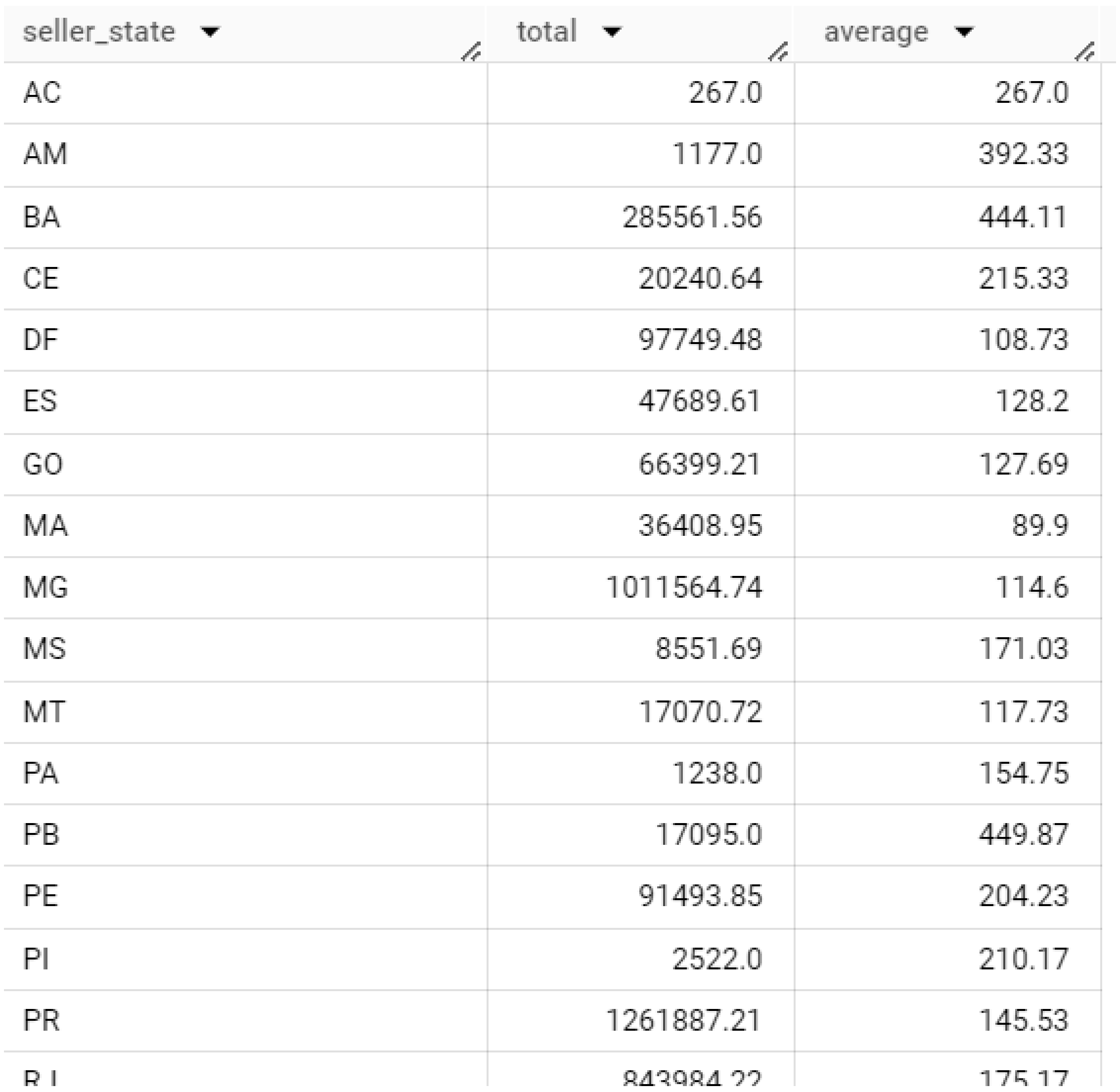
```
WITH YEARLY as(
SELECT EXTRACT(YEAR FROM o.order_purchase_timestamp) as year
, EXTRACT(MONTH FROM o.order_purchase_timestamp) as month,
SUM(p.payment_value) as total
FROM `target.orders` o
INNER JOIN `target.payments` p
ON o.order_id = p.order_id
WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) between
2017 and 2018 and
EXTRACT(MONTH FROM o.order_purchase_timestamp) between 1 and
8
GROUP BY 1,2
ORDER BY 1,2)
SELECT YEARLY.YEAR, YEARLY.month,total, (total-lag(total)
over(order by year,month))/lag(total)over(order by
year,month) *100 as percentage_increasse
FROM YEARLY
ORDER BY 1,2
```



YEAR	month	total	percentage_increass
2017	1	138488.0399999...	null
2017	2	291908.0099999...	110.7821079712...
2017	3	449863.6000000...	54.11142708965...
2017	4	417788.0300000...	-7.13006564656...
2017	5	592918.8200000...	41.91857531198...
2017	6	511276.3800000...	-13.7695814749...
2017	7	592382.9200000...	15.86354135898...
2017	8	674396.3200000...	13.84465980214...
2018	1	1115004.180000...	65.33366908051...
2018	2	992463.3400000...	-10.9901686646...
2018	3	1159652.119999...	16.84583936369...
2018	4	1160785.479999...	0.097732757993...
2018	5	1153982.149999...	-0.58609709694...
2018	6	1023880.499999...	-11.2741475247...
2018	7	1066540.750000...	4.166526269423...
2018	8	1022425.320000...	-4.13630984095...

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

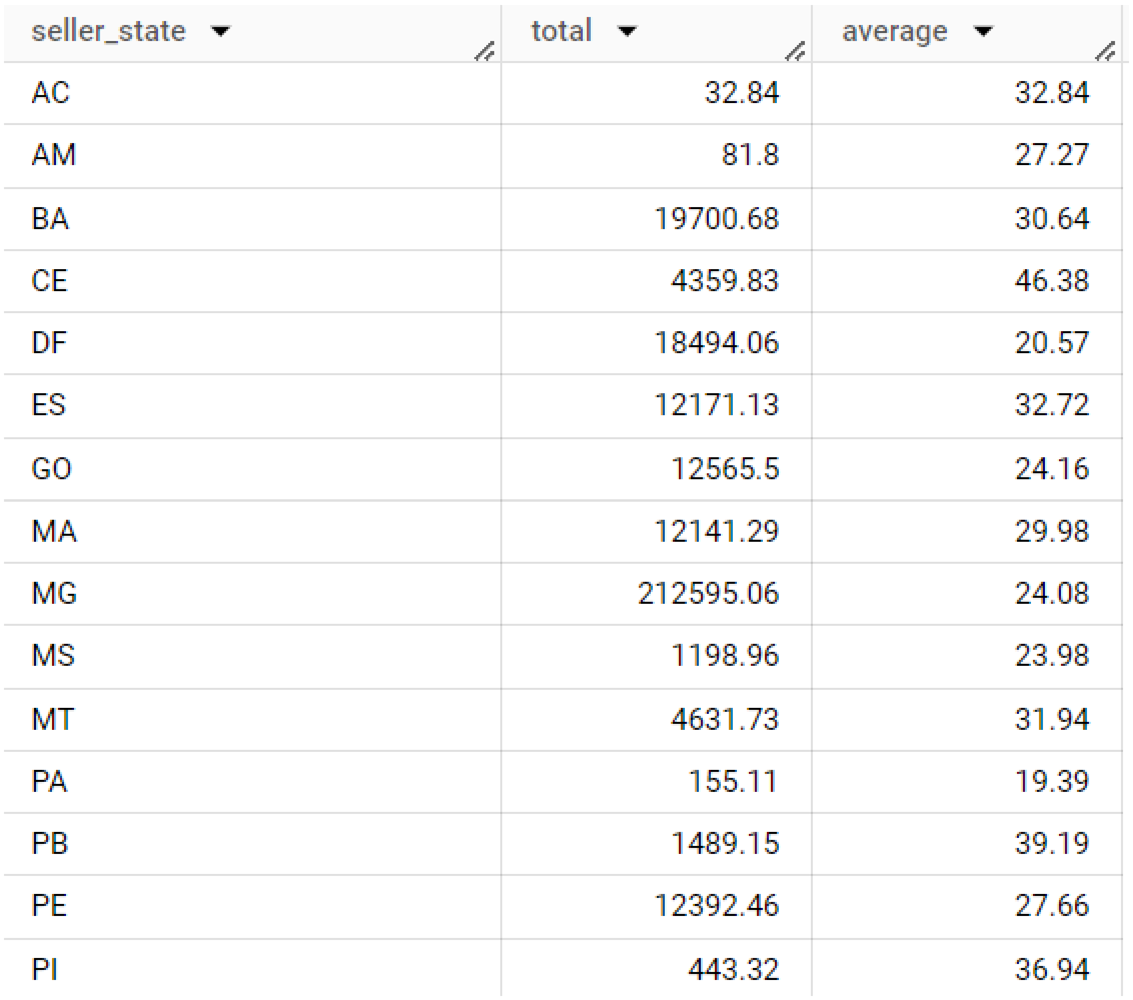
```
SELECT s.seller_state , ROUND(SUM(o.price),2) AS total ,
ROUND(AVG(o.price),2) AS average
FROM `target.order_items` o
INNER JOIN `target.sellers` s
ON o.seller_id = s.seller_id
GROUP BY 1
ORDER BY 1
```



seller_state	total	average
AC	267.0	267.0
AM	1177.0	392.33
BA	285561.56	444.11
CE	20240.64	215.33
DF	97749.48	108.73
ES	47689.61	128.2
GO	66399.21	127.69
MA	36408.95	89.9
MG	1011564.74	114.6
MS	8551.69	171.03
MT	17070.72	117.73
PA	1238.0	154.75
PB	17095.0	449.87
PE	91493.85	204.23
PI	2522.0	210.17
PR	1261887.21	145.53
RJ	843984.22	175.17

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

```
SELECT s.seller_state , ROUND(SUM(o.price),2) AS total ,
ROUND(AVG(o.price),2) AS average
FROM `target.order_items` o
INNER JOIN `target.sellers` s
ON o.seller_id = s.seller_id
GROUP BY 1
ORDER BY 1
```

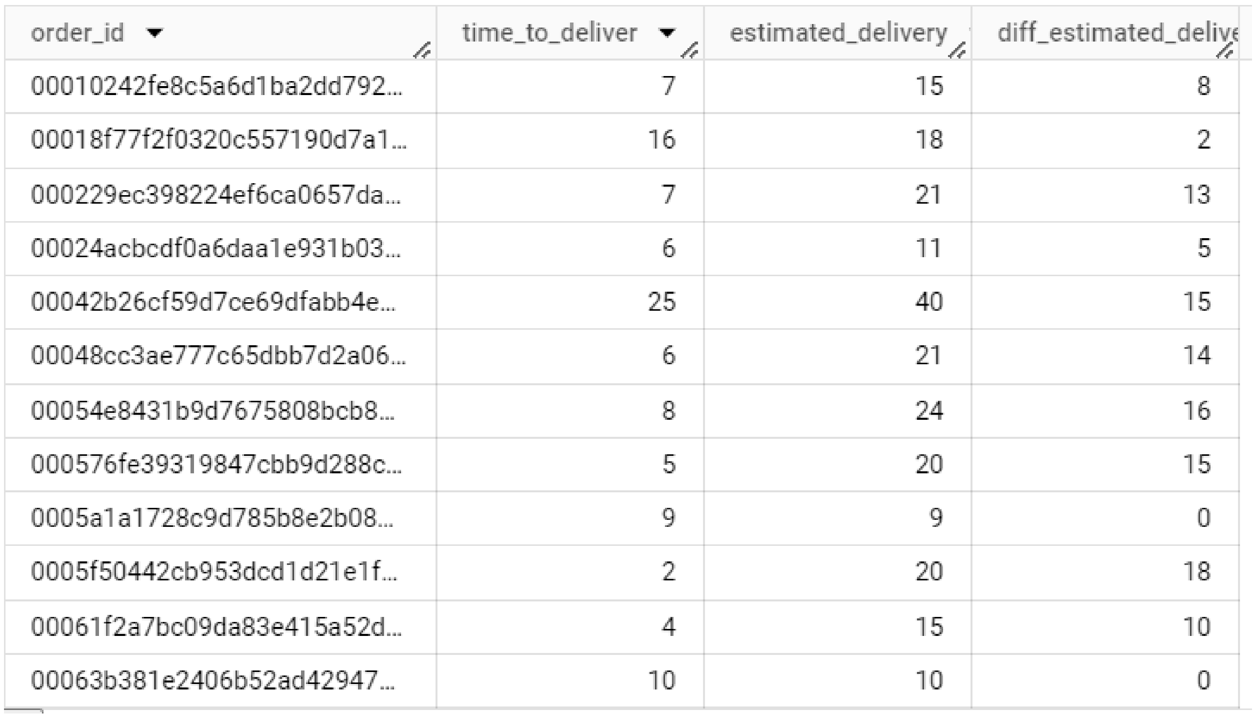


seller_state	total	average
AC	32.84	32.84
AM	81.8	27.27
BA	19700.68	30.64
CE	4359.83	46.38
DF	18494.06	20.57
ES	12171.13	32.72
GO	12565.5	24.16
MA	12141.29	29.98
MG	212595.06	24.08
MS	1198.96	23.98
MT	4631.73	31.94
PA	155.11	19.39
PB	1489.15	39.19
PE	12392.46	27.66
PI	443.32	36.94

5. 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.

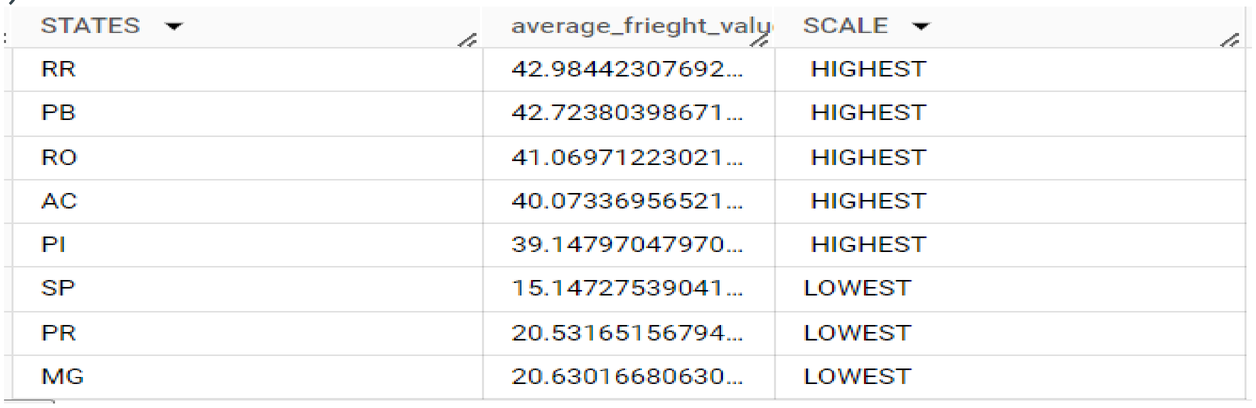
```
SELECT
order_id,DATE_DIFF(order_delivered_customer_date,order_purch
ase_timestamp,day) as
time_to_deliver,DATE_DIFF(order_estimated_delivery_date,orde
r_purchase_timestamp,day) as estimated_delivery,
DATE_DIFF(order_estimated_delivery_date,order_delivered_cust
omer_date,day) as diff_estimated_delivery
FROM `target.orders`
WHERE order_status = 'delivered'
ORDER BY 1
```



order_id	time_to_deliver	estimated_delivery	diff_estimated_delive
00010242fe8c5a6d1ba2dd792...	7	15	8
00018f77f2f0320c557190d7a1...	16	18	2
000229ec398224ef6ca0657da...	7	21	13
00024acbcdf0a6daa1e931b03...	6	11	5
00042b26cf59d7ce69dfabb4e...	25	40	15
00048cc3ae777c65dbb7d2a06...	6	21	14
00054e8431b9d7675808bcb8...	8	24	16
000576fe39319847cbb9d288c...	5	20	15
0005a1a1728c9d785b8e2b08...	9	9	0
0005f50442cb953dcd1d21e1f...	2	20	18
00061f2a7bc09da83e415a52d...	4	15	10
00063b381e2406b52ad42947...	10	10	0

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

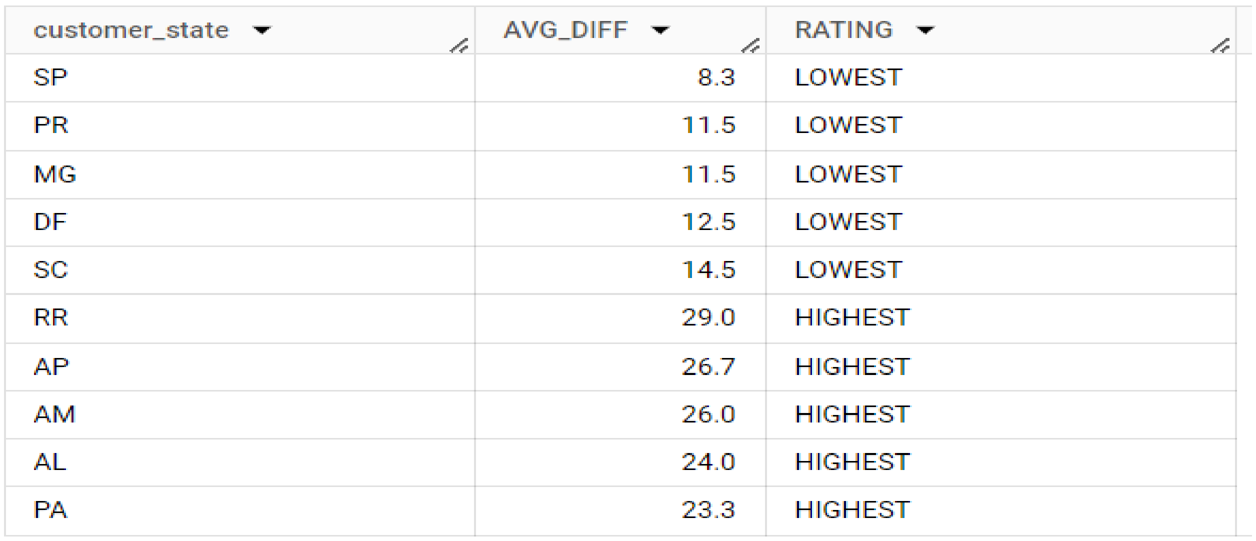
```
WITH average AS(
SELECT c.customer_state,avg(oi.freight_value) as
average_frieght_value
FROM `target.orders` o
INNER JOIN `target.customers` c
ON o.customer_id = c.customer_id
INNER JOIN `target.order_items` oi
ON oi.order_id= o.order_id
GROUP BY 1
)

(
SELECT average.customer_state AS STATES,
average.average_frieght_value, ' HIGHEST'AS SCALE
FROM average
ORDER BY average.average_frieght_value desc
LIMIT 5 )

UNION ALL
(
  SELECT average.customer_state AS STATE
,average.average_frieght_value,'LOWEST' AS SCALE
  FROM average
  ORDER BY average.average_frieght_value asc
  LIMIT 5
)
```



STATES	average_frieght_valu	SCALE
RR	42.98442307692...	HIGHEST
PB	42.72380398671...	HIGHEST
RO	41.06971223021...	HIGHEST
AC	40.07336956521...	HIGHEST
PI	39.14797047970...	HIGHEST
SP	15.14727539041...	LOWEST
PR	20.53165156794...	LOWEST
MG	20.63016680630...	LOWEST

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

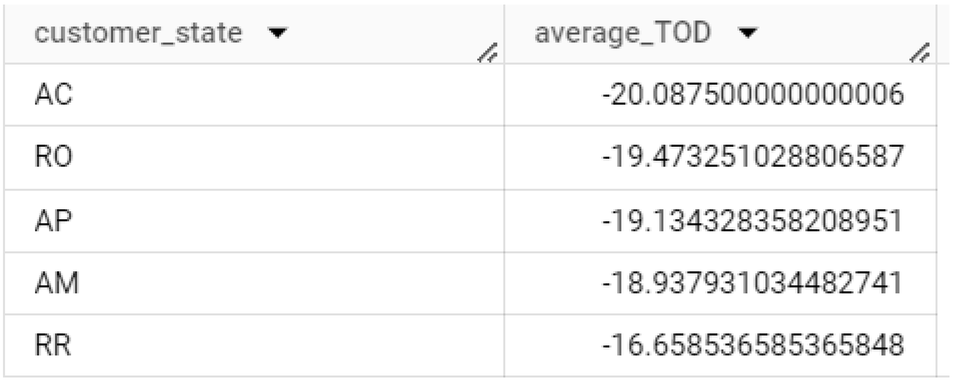
```
SELECT
c.customer_state,DATE_DIFF(o.order_delivered_customer_date,o
.order_purchase_timestamp,day) as difference
FROM `target.orders` o
INNER JOIN `target.customers` c
ON o.customer_id =c.customer_id
WHERE o.order_status = 'delivered'
)
(
  SELECT customer_state, ROUND(avg(diff.difference),1) AS
AVG_DIFF,'HIGHEST'AS RATING
  FROM diff
  GROUP BY 1
  ORDER BY 2 DESC
  LIMIT 5
)
UNION ALL
(
  SELECT customer_state, ROUND(AVG(diff.difference),1) AS
AVG_DIFF, 'LOWEST' AS RATING
  FROM diff
  GROUP BY 1
  ORDER  BY 2 ASC
  LIMIT 5
)
```



customer_state	AVG_DIFF	RATING
SP	8.3	LOWEST
PR	11.5	LOWEST
MG	11.5	LOWEST
DF	12.5	LOWEST
SC	14.5	LOWEST
RR	29.0	HIGHEST
AP	26.7	HIGHEST
AM	26.0	HIGHEST
AL	24.0	HIGHEST
PA	23.3	HIGHEST

4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

```
with dates as(
SELECT c.customer_state,
DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_t
imestamp,day) as
Actual,DATE_DIFF(o.order_estimated_delivery_date,o.order_pur
chase_timestamp,day)as estimated
FROM `target.orders` o
INNER JOIN `target.customers` c
ON o.customer_id= c.customer_id
WHERE o.order_status = 'delivered'
)
  SELECT customer_state,AVG(actual-estimated) as
average_TOD
  FROM dates
  GROUP BY 1
  ORDER BY 2 ASC
  LIMIT 5
```

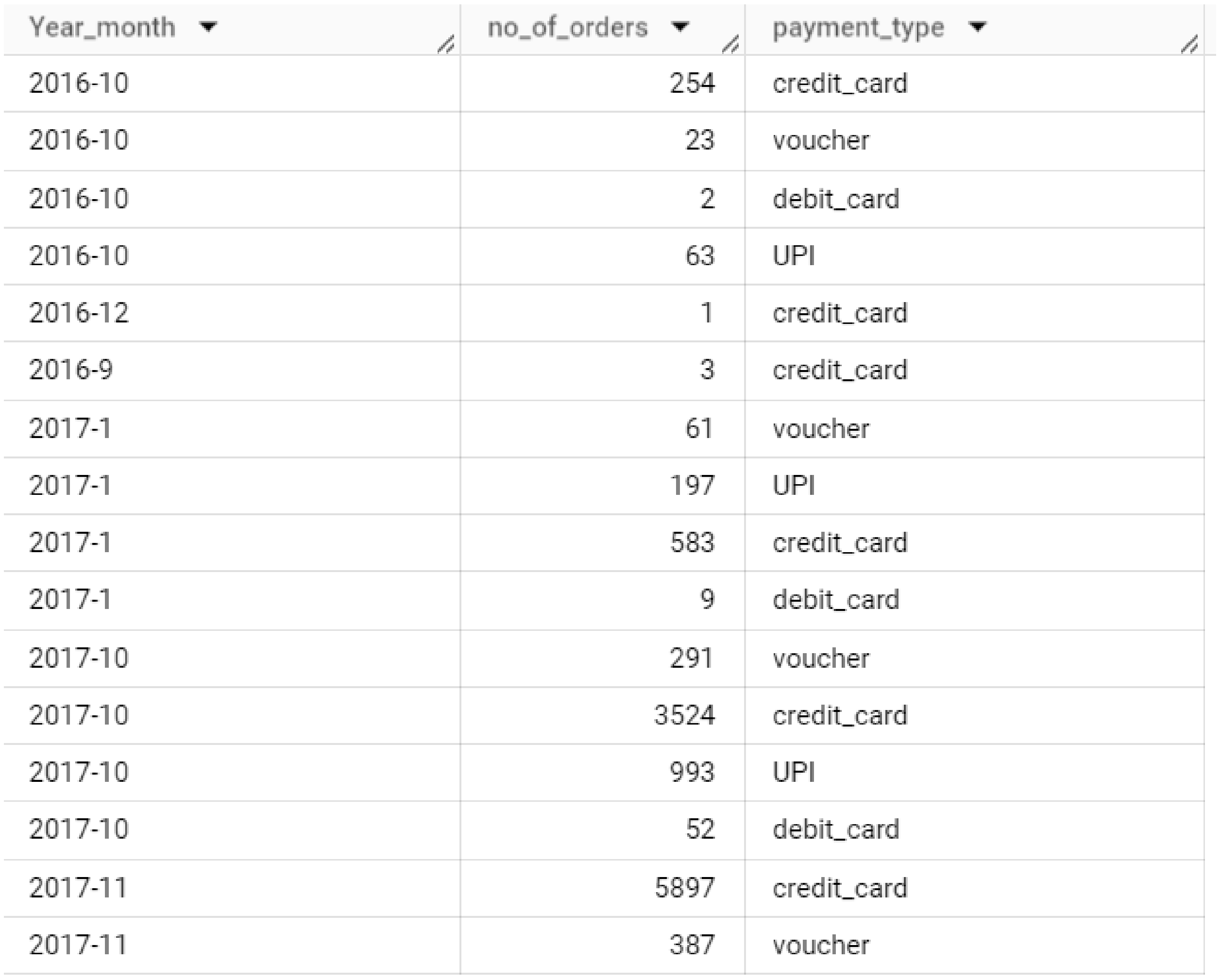


customer_state	average_TOD
AC	-20.087500000000006
RO	-19.473251028806587
AP	-19.134328358208951
AM	-18.937931034482741
RR	-16.658536585365848

6. Analysis based on the payments:

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

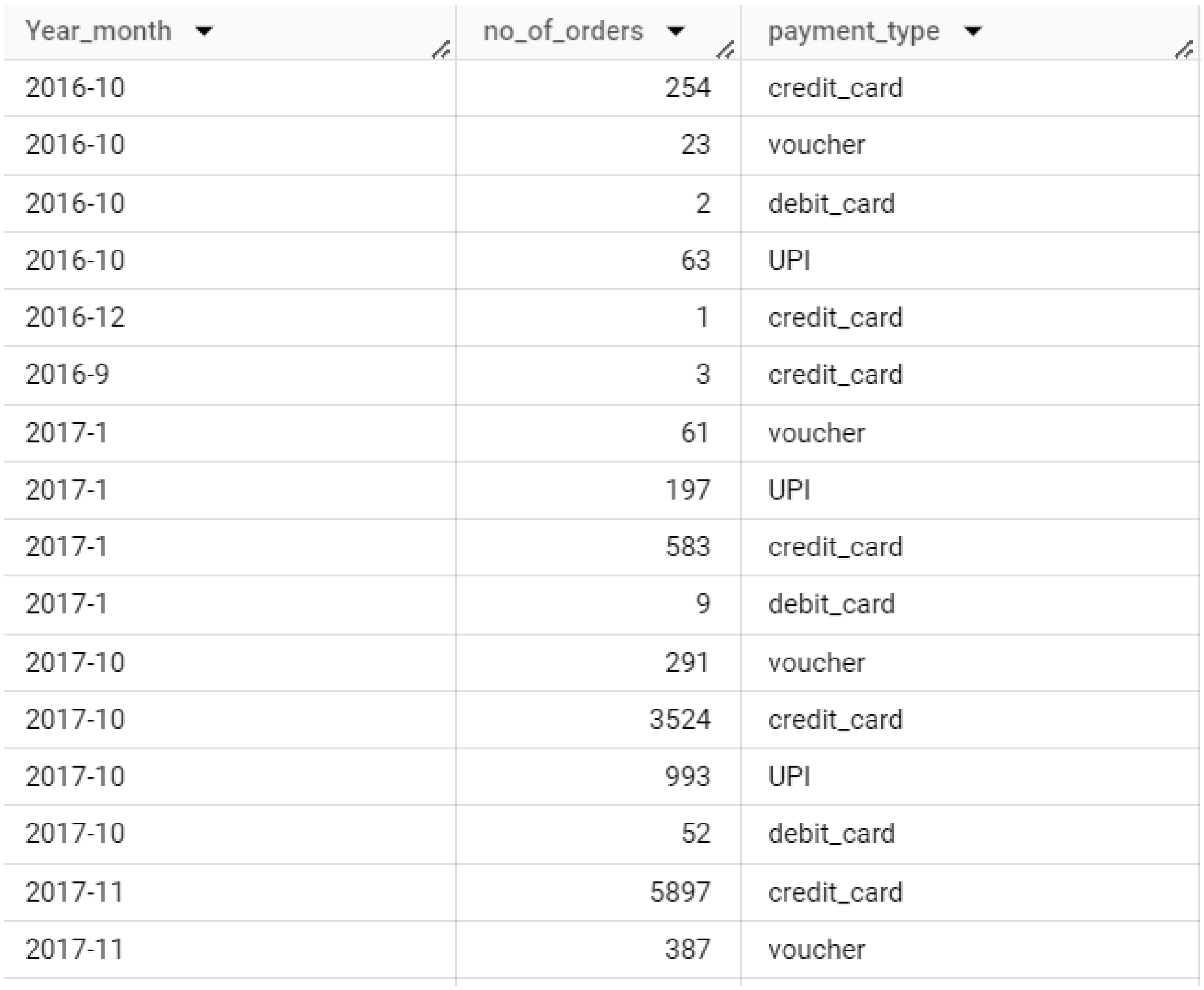
```sql
SELECT CONCAT(EXTRACT(YEAR FROM
o.order_purchase_timestamp),'-',EXTRACT(MONTH FROM
o.order_purchase_timestamp)) as Year_month ,
count(o.order_id) as no_of_orders, p.payment_type
FROM `target.payments` p
INNER JOIN `target.orders` o
ON p.order_id = o.order_id
GROUP BY 1,3
ORDER BY 1
```



Year_month	no_of_orders	payment_type
2016-10	254	credit_card
2016-10	23	voucher
2016-10	2	debit_card
2016-10	63	UPI
2016-12	1	credit_card
2016-9	3	credit_card
2017-1	61	voucher
2017-1	197	UPI
2017-1	583	credit_card
2017-1	9	debit_card
2017-10	291	voucher
2017-10	3524	credit_card
2017-10	993	UPI
2017-10	52	debit_card
2017-11	5897	credit_card
2017-11	387	voucher

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

```sql
SELECT payment_installments, count(order_id) as no_of_orders
FROM `target.payments`
WHERE payment_installments >=1
GROUP BY 1
```



Year_month	no_of_orders	payment_type
2016-10	254	credit_card
2016-10	23	voucher
2016-10	2	debit_card
2016-10	63	UPI
2016-12	1	credit_card
2016-9	3	credit_card
2017-1	61	voucher
2017-1	197	UPI
2017-1	583	credit_card
2017-1	9	debit_card
2017-10	291	voucher
2017-10	3524	credit_card
2017-10	993	UPI
2017-10	52	debit_card
2017-11	5897	credit_card
2017-11	387	voucher