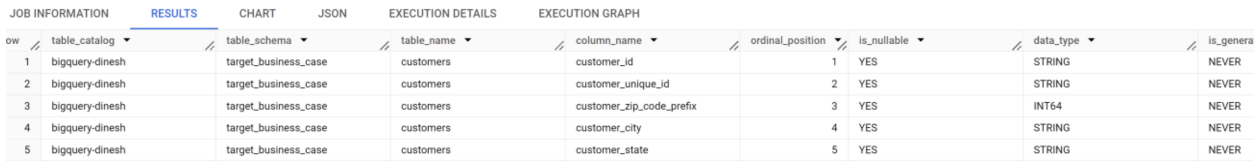
```sql
--1.1 Data type of all columns in the "customers" table.
SELECT * FROM `bigquery-dinesh.target_business_case.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
```



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

ow	table_catalog	table_schema	table_name	column_name	ordinal_position	is_nullable	data_type	is_genera
1	bigquery-dinesh	target_business_case	customers	customer_id	1	YES	STRING	NEVER
2	bigquery-dinesh	target_business_case	customers	customer_unique_id	2	YES	STRING	NEVER
3	bigquery-dinesh	target_business_case	customers	customer_zip_code_prefix	3	YES	INT64	NEVER
4	bigquery-dinesh	target_business_case	customers	customer_city	4	YES	STRING	NEVER
5	bigquery-dinesh	target_business_case	customers	customer_state	5	YES	STRING	NEVER

Insights:

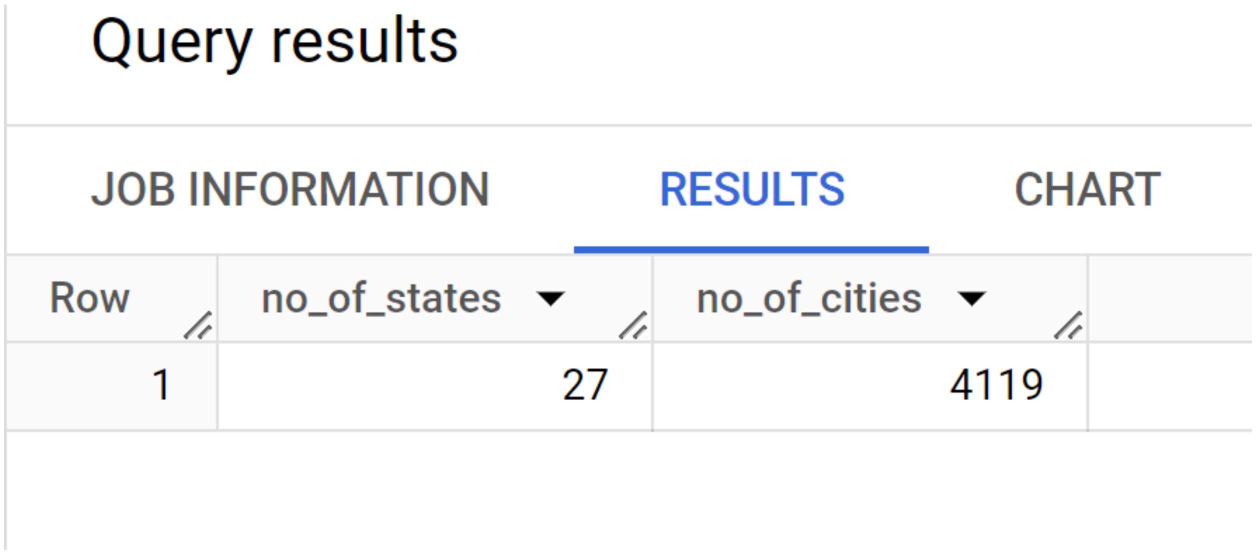
1. Gives the complete details about the data type of each column

```sql
--1.2 Get the time range between which the orders were placed.
SELECT MIN(order_purchase_timestamp) AS start_date, MAX(order_purchase_timestamp) AS
last_date
FROM bigquery-dinesh.target_business_case.orders
;
```

Insights

This gives the idea on the first order placed in the dataset and last order placed in the dataset

```sql
--1.3 Count the Cities & States of customers who ordered during the given period.
SELECT COUNT(DISTINCT customer_state) as no_of_states, COUNT(DISTINCT customer_city)
as no_of_cities
FROM bigquery-dinesh.target_business_case.customers
```



Query results

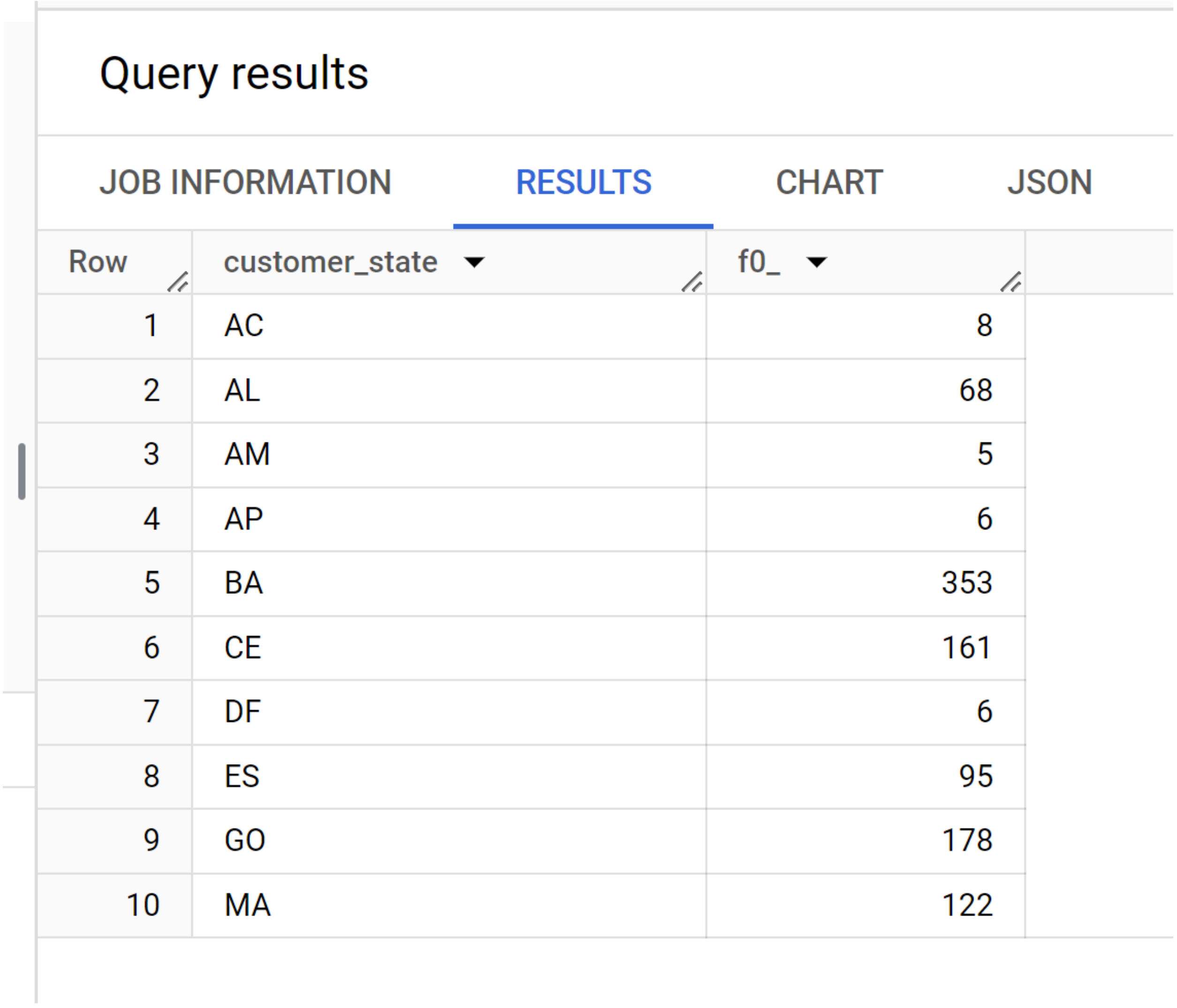
JOB INFORMATION | RESULTS | CHART

Row	no_of_states	no_of_cities
1	27	4119

Insights

1. There are total of 27 states with 4119 cities

```
-- Count the cities per state during th given period
SELECT customer_state, count(DISTINCT customer_city)
FROM bigquery-dinesh.target_business_case.customers
GROUP BY customer_state
ORDER BY customer_state
LIMIT 10
```



Query results

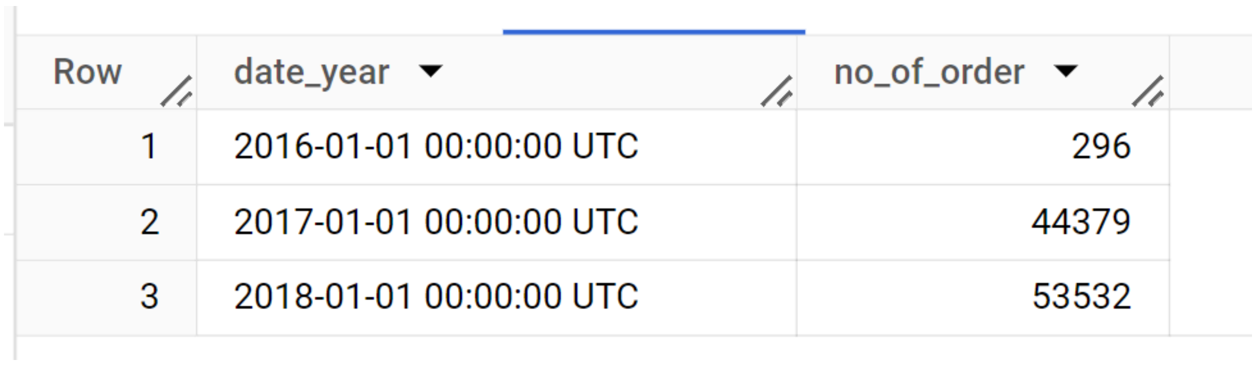
JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	f0_
1	AC	8
2	AL	68
3	AM	5
4	AP	6
5	BA	353
6	CE	161
7	DF	6
8	ES	95
9	GO	178
10	MA	122

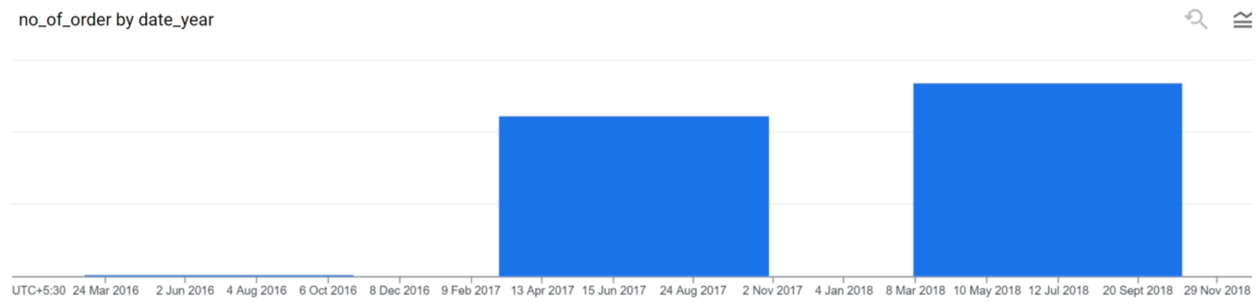
Insights

This gives rough idea on no of cities per state in the order list to identity the max sales/ revenue per state in later stages

```
-- 2.1 Is there a growing trend in the no. of orders placed over the past years?
SELECT DATE_TRUNC(order_purchase_timestamp, YEAR) as date_year, COUNT(order_id) as
no_of_order
FROM bigquery-dinesh.target_business_case.orders
where lower(order_status) not in  ('unavailable','canceled')
group by 1
order by 1
```



Row	date_year	no_of_order
1	2016-01-01 00:00:00 UTC	296
2	2017-01-01 00:00:00 UTC	44379
3	2018-01-01 00:00:00 UTC	53532

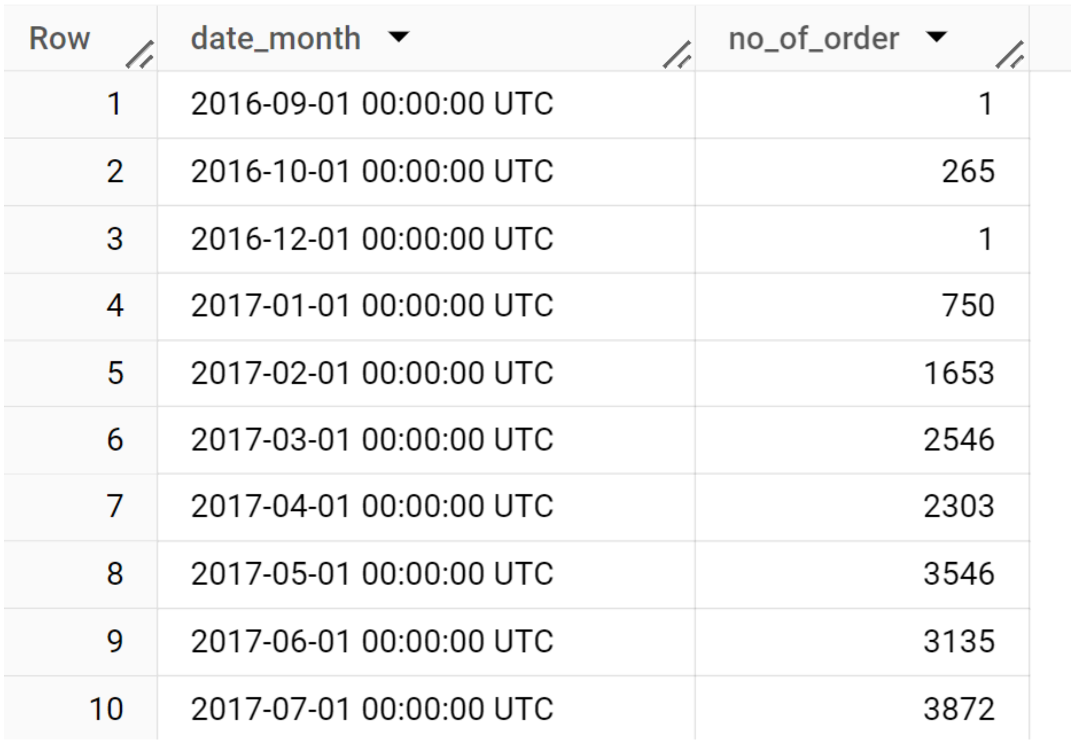


Insights

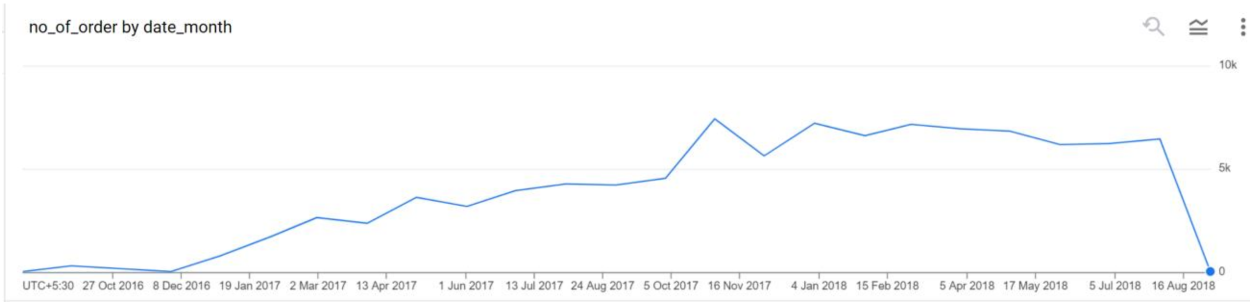
1. From the output and graph above, there is a clear indication of year-on-year growing demand

extension

```
2.1 to the year-on-year demand, Is there a growing trend in the no. of orders placed
over the month
SELECT DATE_TRUNC(order_purchase_timestamp, MONTH) as date_month, COUNT(order_id) as
no_of_order
FROM bigquery-dinesh.target_business_case.orders
where lower(order_status) = 'delivered'
group by 1
order by 1
```



Row	date_month	no_of_order
1	2016-09-01 00:00:00 UTC	1
2	2016-10-01 00:00:00 UTC	265
3	2016-12-01 00:00:00 UTC	1
4	2017-01-01 00:00:00 UTC	750
5	2017-02-01 00:00:00 UTC	1653
6	2017-03-01 00:00:00 UTC	2546
7	2017-04-01 00:00:00 UTC	2303
8	2017-05-01 00:00:00 UTC	3546
9	2017-06-01 00:00:00 UTC	3135
10	2017-07-01 00:00:00 UTC	3872

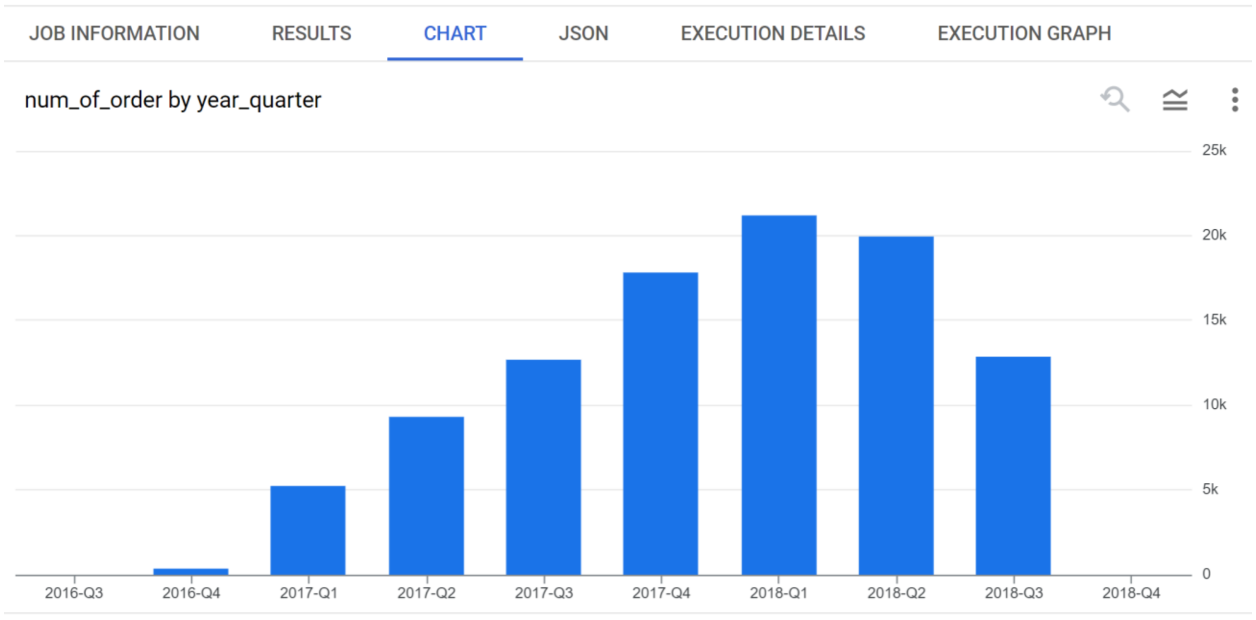


Insights

1. From the below, There is a growing demand till jan 2018 on monthly, but got stabilized till Aug 2018 and there is a sudden drop in the orders in Sep 2018 and Oct 2018.
2. Also, there is purge in the number of orders in around September, need to look for the global cause or other reasons

```
-- 2.2. Can we see some kind of monthly seasonality in terms of the no. of orders
being placed?
WITH cte AS (
  SELECT order_id, customer_id, order_purchase_timestamp,
    CASE
      WHEN EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 3 THEN 'Q1'
      WHEN EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 4 AND 6 THEN 'Q2'
      WHEN EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 7 AND 9 THEN 'Q3'
      ELSE 'Q4'
    END AS quarter,
    EXTRACT(YEAR FROM order_purchase_timestamp) AS year
  FROM bigquery-dinesh.target_business_case.orders
  where lower(order_status) not in  ('unavailable','canceled')
)

SELECT distinct year, quarter, COUNT(order_id) as num_of_order, concat(year, '-',
quarter) as year_quarter
from cte
group by year, quarter
order by year, quarter
```

Row	year	quarter	num_of_order	year_quarter
1	2016	Q3	2	2016-Q3
2	2016	Q4	294	2016-Q4
3	2017	Q1	5122	2017-Q1
4	2017	Q2	9222	2017-Q2
5	2017	Q3	12445	2017-Q3
6	2017	Q4	17590	2017-Q4
7	2018	Q1	20980	2018-Q1
8	2018	Q2	19897	2018-Q2
9	2018	Q3	12655	2018-Q3

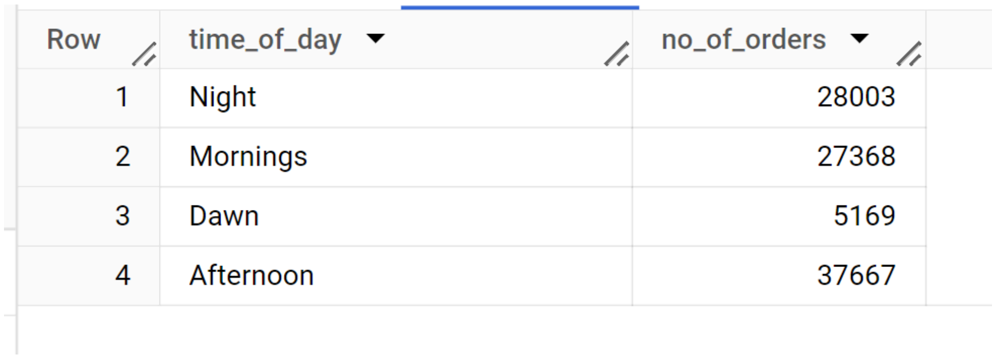


Job history

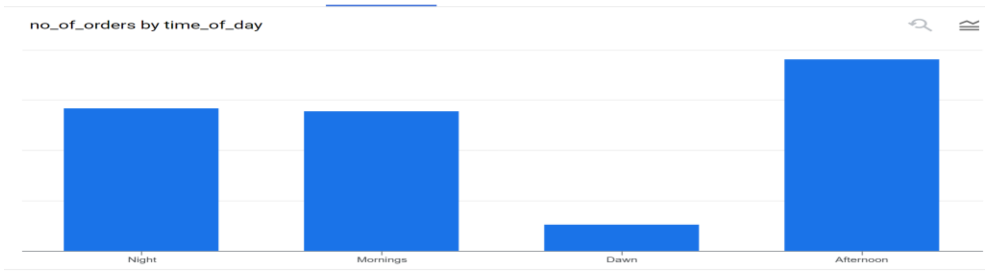
Insights:

1. There is more demand in quarter1 and quarter 2 which is Jan to Mar and April to June are preferred.
2. Having the products recommended for the user during this period would be good idea

```
--2.3 During what time of the day, do the Brazilian customers mostly place their
orders? (Dawn, Morning, Afternoon or Night)
-- 0-6 hrs : Dawn
-- 7-12 hrs : Mornings
-- 13-18 hrs : Afternoon
-- 19-23 hrs : Night
WITH tod as (
  SELECT order_id, customer_id, 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_day
  FROM bigquery-dinesh.target_business_case.orders
  where lower(order_status) not in  ('unavailable','canceled')
)
SELECT distinct time_of_day,COUNT(order_id) over(partition by time_of_day) AS
no_of_orders
FROM tod
ORDER BY time_of_day DESC
```



Row	time_of_day	no_of_orders
1	Night	28003
2	Mornings	27368
3	Dawn	5169
4	Afternoon	37667

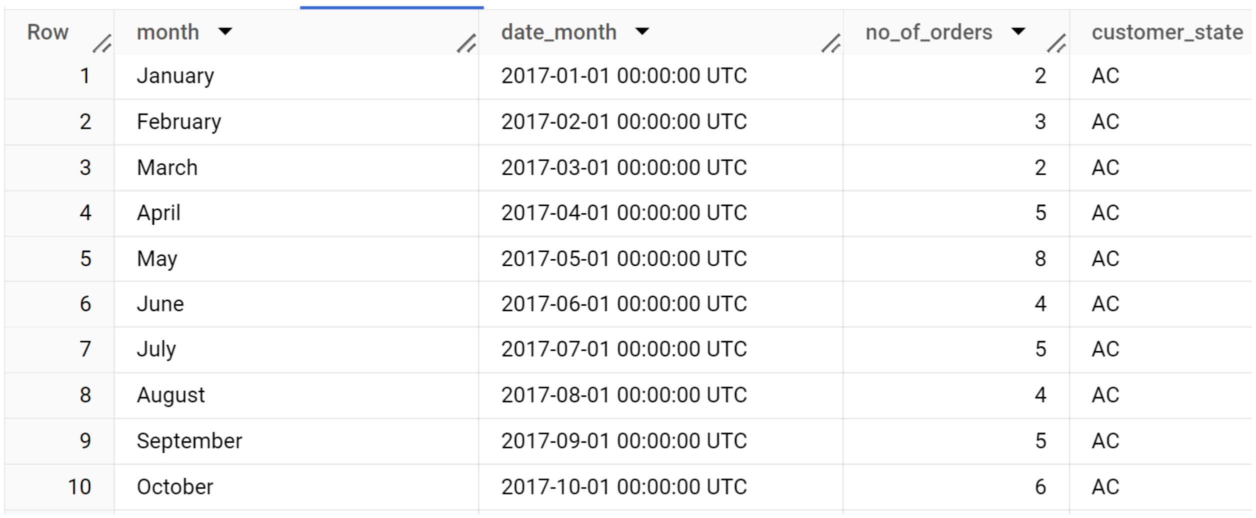


Insights

1. From graph it is evident that most of the orders are being placed in the Afternoon between 1300 to 1800 hrs and least orders are placed in the Dawn.

```
- 3Evolution of E-commerce orders in the Brazil region:
-- 3.1 Get the month on month no. of orders placed in each state.
```

```
WITH cus_order_data as (
  SELECT c.*, o.order_id, o.order_purchase_timestamp
  FROM bigquery-dinesh.target_business_case.customers c
  join bigquery-dinesh.target_business_case.orders o
  on c.customer_id = o.customer_id
  where lower(o.order_status) not in  ('unavailable','canceled')
)
  SELECT distinct FORMAT_DATE('%B', order_purchase_timestamp) as month,
DATE_TRUNC(order_purchase_timestamp, MONTH) as date_month, COUNT(order_id)
over(partition by DATE_TRUNC(order_purchase_timestamp, MONTH), customer_state) as
no_of_orders, customer_state
  FROM cus_order_data
  order by customer_state, date_month
```

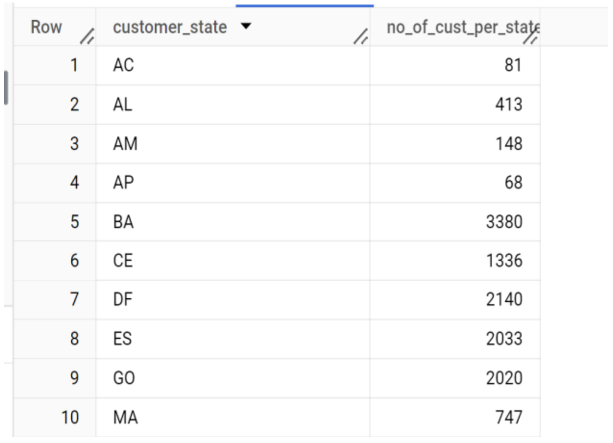


Row	month	date_month	no_of_orders	customer_state
1	January	2017-01-01 00:00:00 UTC	2	AC
2	February	2017-02-01 00:00:00 UTC	3	AC
3	March	2017-03-01 00:00:00 UTC	2	AC
4	April	2017-04-01 00:00:00 UTC	5	AC
5	May	2017-05-01 00:00:00 UTC	8	AC
6	June	2017-06-01 00:00:00 UTC	4	AC
7	July	2017-07-01 00:00:00 UTC	5	AC
8	August	2017-08-01 00:00:00 UTC	4	AC
9	September	2017-09-01 00:00:00 UTC	5	AC
10	October	2017-10-01 00:00:00 UTC	6	AC

Insights:

1. The above results give us no of orders placed by a state in a month.
2. Having the bar graph with y:axis → no of orders and x: axis as date_month for each state give us info about the no of orders being placed and in which month most of the orders are being placed for each state.

```
-- 3.2 How are the customers distributed across all the states?

SELECT customer_state, COUNT(customer_id) AS no_of_cust_per_state
FROM bigquery-dinesh.target_business_case.customers
GROUP BY customer_state
ORDER BY customer_state
```

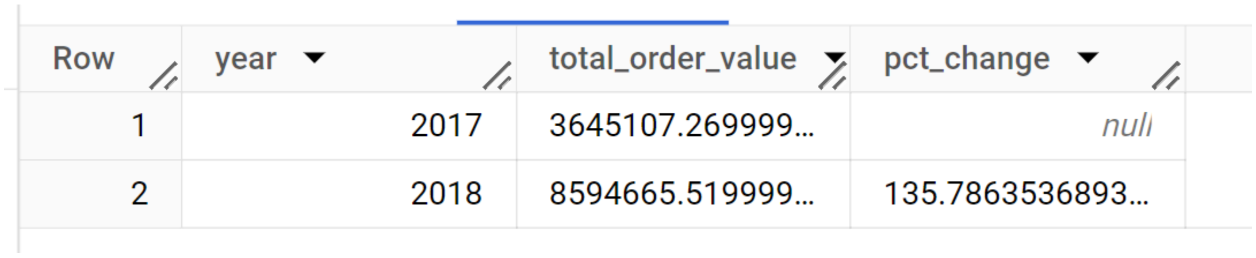


Row	customer_state	no_of_cust_per_state
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

Insights:

1. The above results indicate no of customers in each state, targeting the states with large of customers would benefit the market

```
-- 4.1 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.
WITH order_date AS(
    SELECT o.order_id,
      o.order_purchase_timestamp, p.payment_value,
      EXTRACT(YEAR FROM o.order_purchase_timestamp) as year
    FROM bigquery-dinesh.target_business_case.orders o
    JOIN bigquery-dinesh.target_business_case.payments p
    ON o.order_id = p.order_id
    WHERE (o.order_purchase_timestamp BETWEEN '2017-01-01' AND '2017-08-31') OR
(o.order_purchase_timestamp BETWEEN '2018-01-01' AND '2018-08-31') AND
(lower(order_status) not in  ('unavailable','canceled'))

)

SELECT year, total_order_value, (total_order_value - previous_year_value) * 100 /
previous_year_value as pct_change
FROM (
  SELECT year,
  SUM(payment_value) as total_order_value,
  lag(SUM(payment_value), 1) over(order by year) as previous_year_value
  FROM order_date
  group by year
)
```

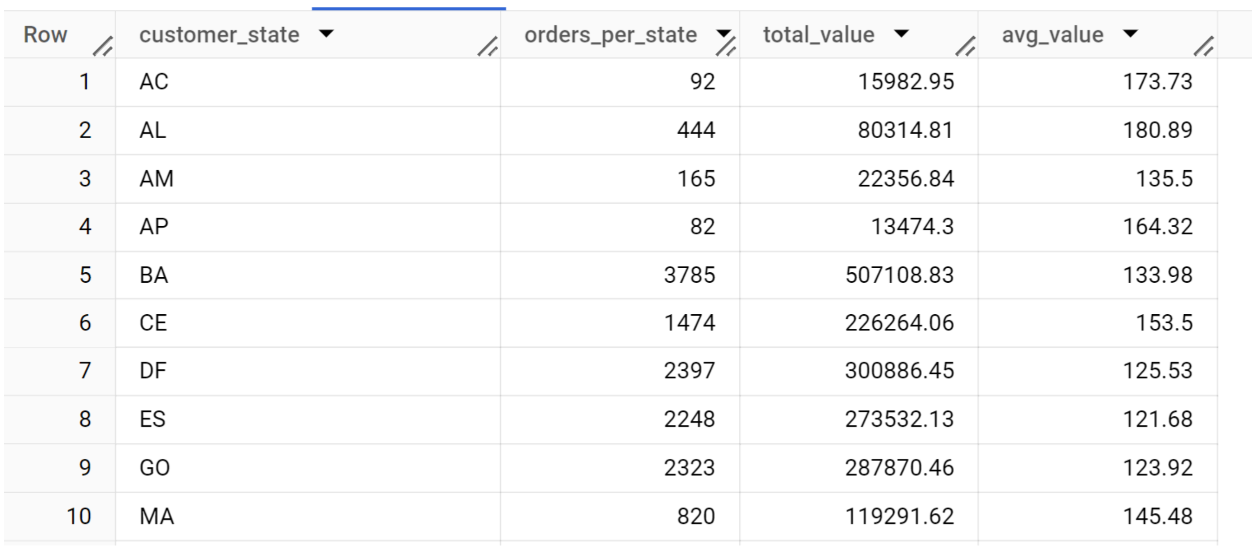


Row	year	total_order_value	pct_change
1	2017	3645107.269999...	*null*
2	2018	8594665.519999...	135.7863536893...

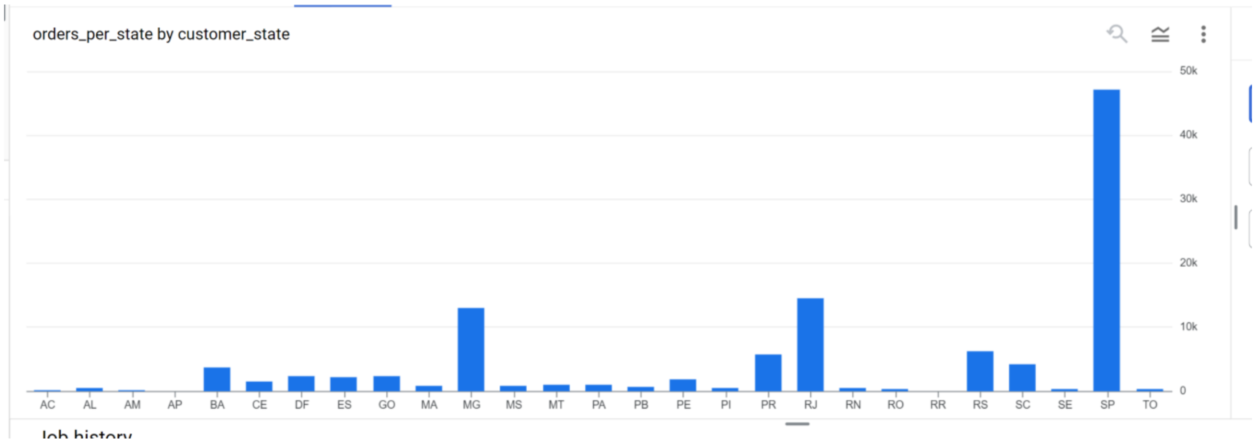
Insights:

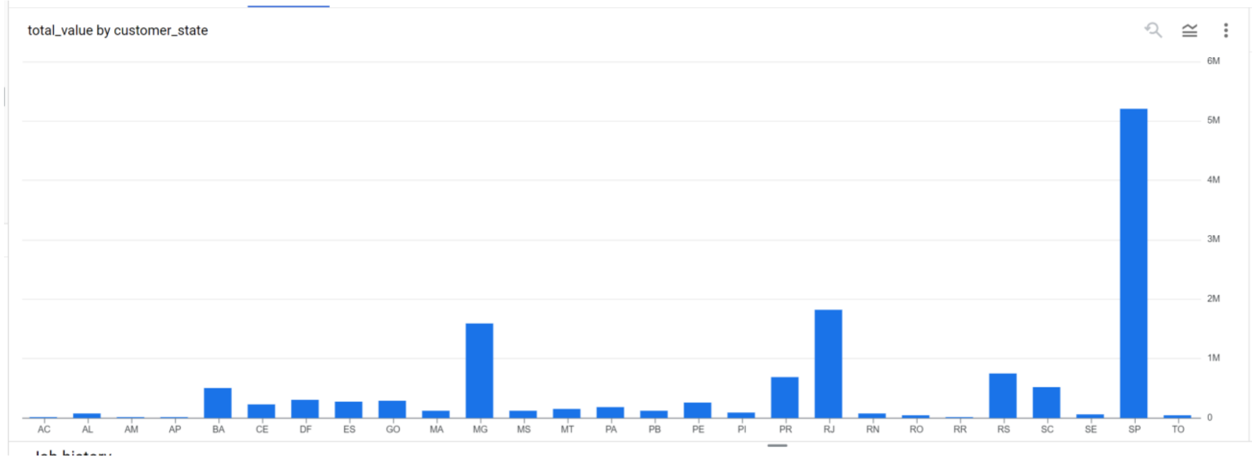
1. There is clear indication of increased revenue from previous year over 136%.

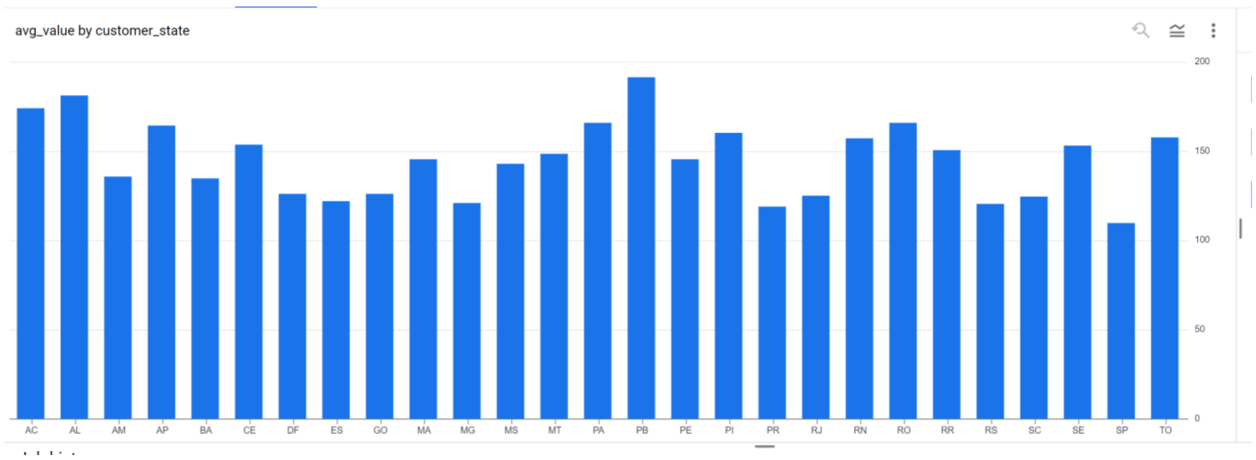
```
-- 4.2 Calculate the Total & Average value of order price for each state.
SELECT c.customer_state,
  COUNT(oi.order_id) AS orders_per_state,
  ROUND(sum(oi.price), 2) AS total_value,
  ROUND(AVG(oi.price), 2) AS avg_value
FROM bigquery-dinesh.target_business_case.customers c
JOIN bigquery-dinesh.target_business_case.orders o
ON c.customer_id = o.customer_id
JOIN bigquery-dinesh.target_business_case.order_items oi
ON o.order_id = oi.order_id
WHERE lower(order_status) not in  ('unavailable','canceled')
GROUP BY c.customer_state
ORDER BY c.customer_state
```



Row	customer_state	orders_per_state	total_value	avg_value
1	AC	92	15982.95	173.73
2	AL	444	80314.81	180.89
3	AM	165	22356.84	135.5
4	AP	82	13474.3	164.32
5	BA	3785	507108.83	133.98
6	CE	1474	226264.06	153.5
7	DF	2397	300886.45	125.53
8	ES	2248	273532.13	121.68
9	GO	2323	287870.46	123.92
10	MA	820	119291.62	145.48



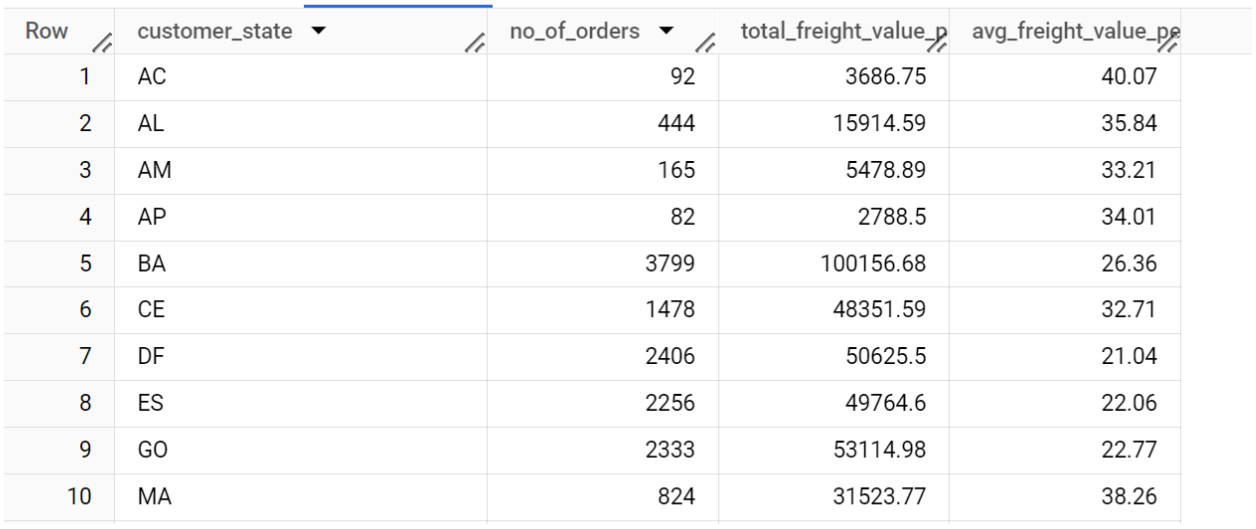




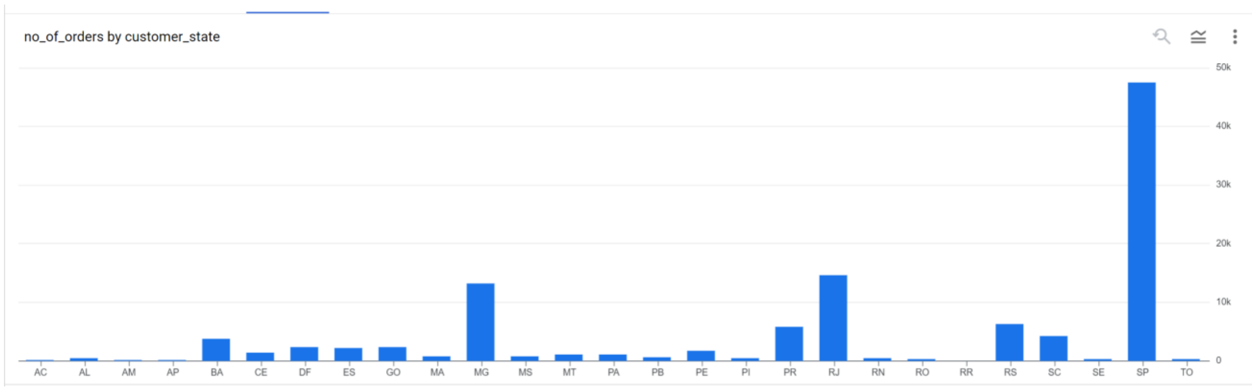
Insights:

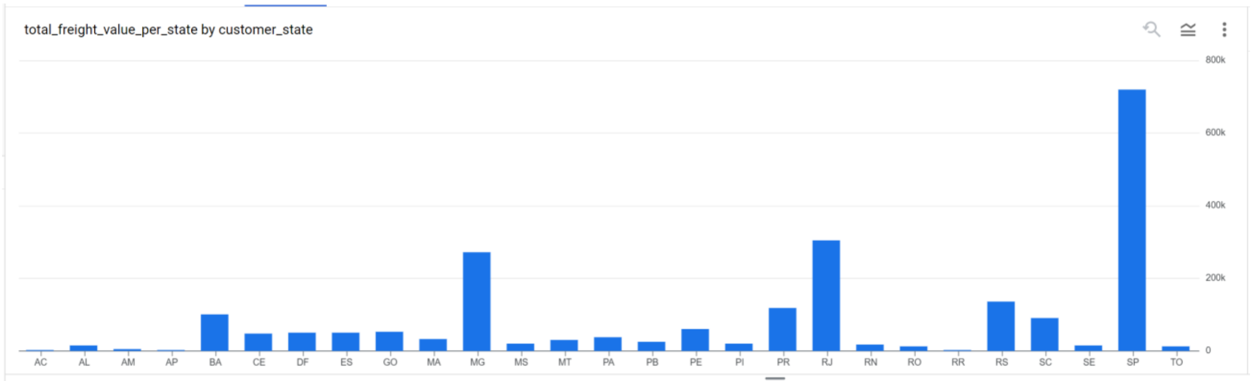
1. From the graph above it is very clear that the state **SP** has a large revenue.
2. Almost all the states have customers buying the products with almost same avg value.

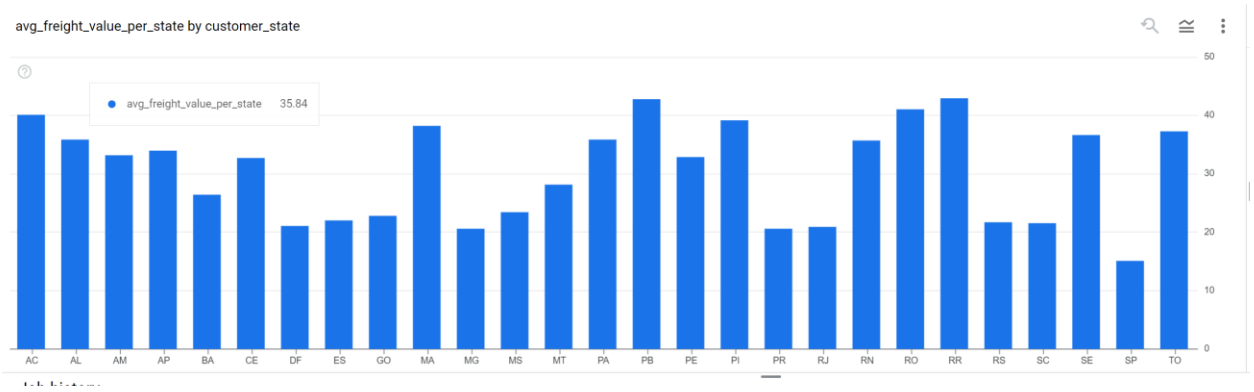
```
-- 4.3 Calculate the Total & Average value of order freight for each state.
SELECT c.customer_state,
  COUNT(oi.order_id) as no_of_orders,
  ROUND(SUM(oi.freight_value), 2) as total_freight_value_per_state,
  ROUND(AVG(oi.freight_value), 2) as avg_freight_value_per_state

FROM bigquery-dinesh.target_business_case.customers c
JOIN bigquery-dinesh.target_business_case.orders o
ON c.customer_id = o.customer_id
JOIN bigquery-dinesh.target_business_case.order_items oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
order by c.customer_state
```



Row	customer_state	no_of_orders	total_freight_value_p	avg_freight_value_pe
1	AC	92	3686.75	40.07
2	AL	444	15914.59	35.84
3	AM	165	5478.89	33.21
4	AP	82	2788.5	34.01
5	BA	3799	100156.68	26.36
6	CE	1478	48351.59	32.71
7	DF	2406	50625.5	21.04
8	ES	2256	49764.6	22.06
9	GO	2333	53114.98	22.77
10	MA	824	31523.77	38.26



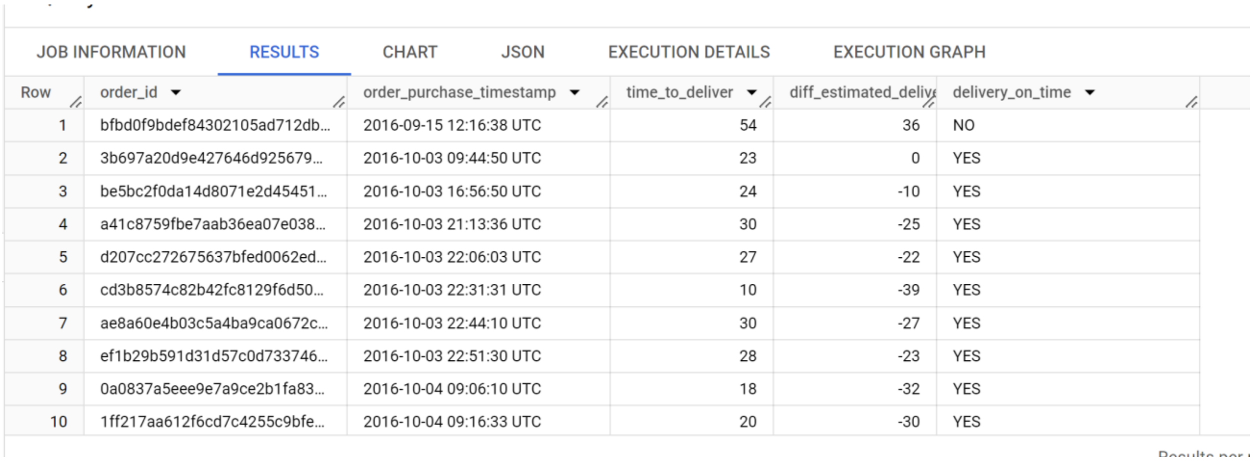




Insights

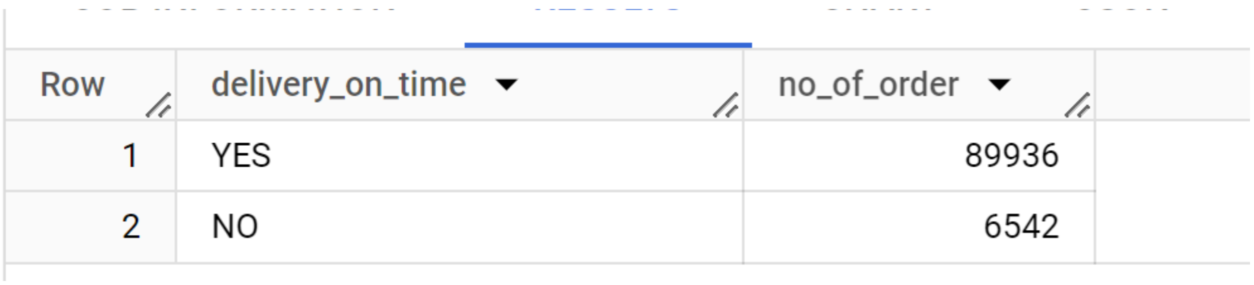
1. The total freight value of the state **SP is high but the avg freight vaue is less compared to other states.**

```
-- 5.1 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:
-- time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
-- diff_estimated_delivery = order_delivered_customer_date -
order_estimated_delivery_date
with days_to_deliver AS(
  SELECT order_id, order_purchase_timestamp,
  DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) as
time_to_deliver,
  DATETIME_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY) as
diff_estimated_delivery,
  CASE
    WHEN DATETIME_DIFF(order_delivered_customer_date, order_estimated_delivery_date,
DAY) <= 0 THEN 'YES'
    ELSE 'NO'
  END AS delivery_on_time
  FROM bigquery-dinesh.target_business_case.orders
  WHERE order_status = 'delivered'
  order by order_purchase_timestamp
)

SELECT * FROM days_to_deliver
```

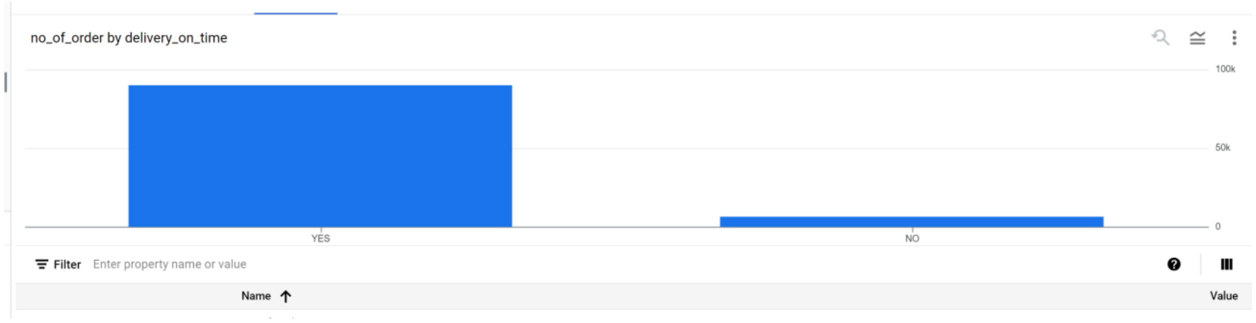


Row	order_id	order_purchase_timestamp	time_to_deliver	diff_estimated_deliv	delivery_on_time
1	bfbd0f9bdef84302105ad712db...	2016-09-15 12:16:38 UTC	54	36	NO
2	3b697a20d9e427646d925679...	2016-10-03 09:44:50 UTC	23	0	YES
3	be5bc2f0da14d8071e2d45451...	2016-10-03 16:56:50 UTC	24	-10	YES
4	a41c8759fbe7aab36ea07e038...	2016-10-03 21:13:36 UTC	30	-25	YES
5	d207cc272675637bfed0062ed...	2016-10-03 22:06:03 UTC	27	-22	YES
6	cd3b8574c82b42fc8129f6d50...	2016-10-03 22:31:31 UTC	10	-39	YES
7	ae8a60e4b03c5a4ba9ca0672c...	2016-10-03 22:44:10 UTC	30	-27	YES
8	ef1b29b591d31d57c0d733746...	2016-10-03 22:51:30 UTC	28	-23	YES
9	0a0837a5eee9e7a9ce2b1fa83...	2016-10-04 09:06:10 UTC	18	-32	YES
10	1ff217aa612f6cd7c4255c9bfe...	2016-10-04 09:16:33 UTC	20	-30	YES

```
(Continue)
-- SELECT count(*) as no_of_orders_gt_estimated_date
-- FROM days_to_deliver
-- where delivery_on_time = 'YES'

-- SELECT count(*) as no_of_orders_le_estimated_date
-- FROM days_to_deliver
-- where delivery_on_time = 'NO'

SELECT delivery_on_time, count(*) as no_of_order
FROM days_to_deliver
GROUP BY delivery_on_time
```



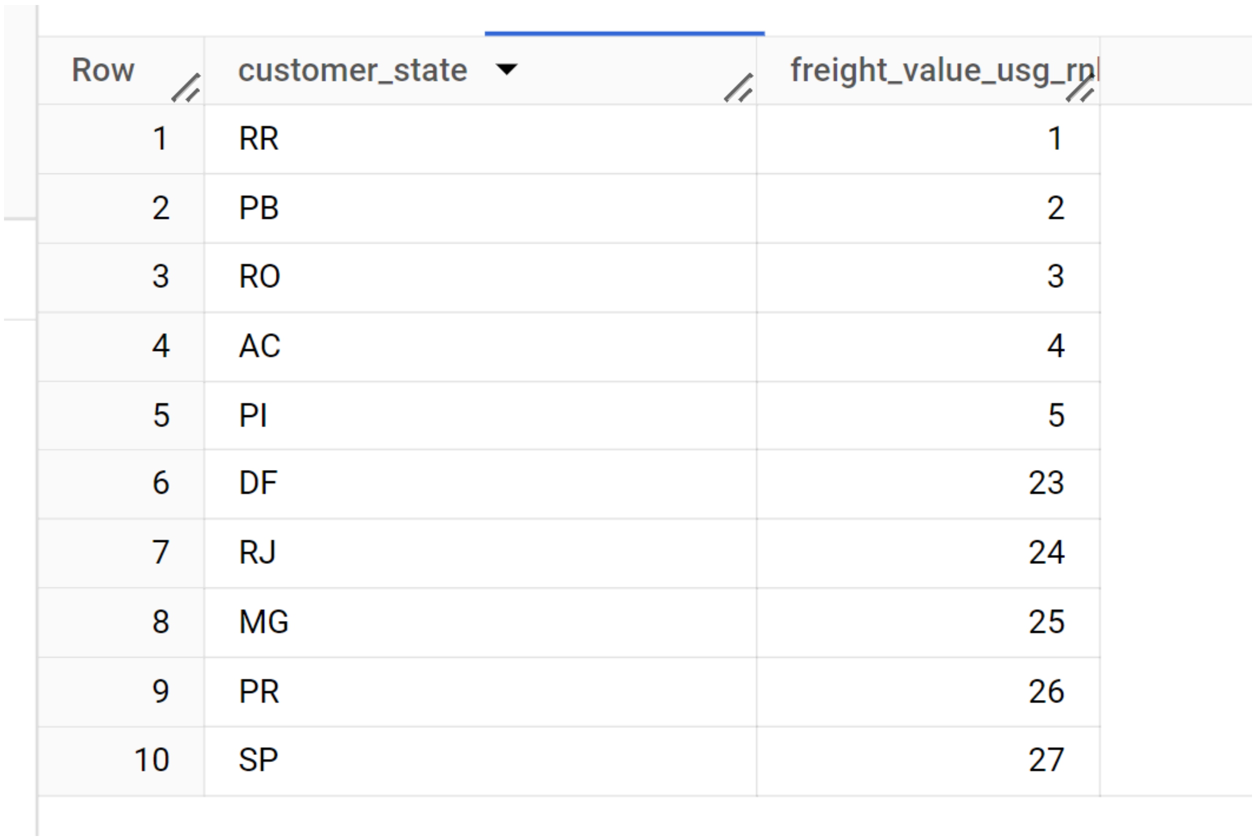
Row	delivery_on_time	no_of_order
1	YES	89936
2	NO	6542



Insights:

1. It is evident from the above graph that around 93% of the orders are delivered on time

```
-- 5.2 Find out the top 5 states with the highest & lowest average freight value.

WITH afv AS(
  SELECT c.customer_state,
    ROUND(SUM(oi.freight_value), 2) as total_freight_value_per_state,
    ROUND(AVG(oi.freight_value), 2) as avg_freight_value_per_state,
    DENSE_RANK() OVER(order by AVG(oi.freight_value) DESC) as freight_value_usg_rnk,
    DENSE_RANK() OVER(order by AVG(oi.freight_value) ASC) as freight_value_usg_rnk_asc
  FROM bigquery-dinesh.target_business_case.customers c
  JOIN bigquery-dinesh.target_business_case.orders o
  ON c.customer_id = o.customer_id
  JOIN bigquery-dinesh.target_business_case.order_items oi
  ON o.order_id = oi.order_id
  GROUP BY c.customer_state
  order by c.customer_state
)
select customer_state, freight_value_usg_rnk

from afv
where freight_value_usg_rnk <=5 or freight_value_usg_rnk_asc <=5
order by freight_value_usg_rnk
```

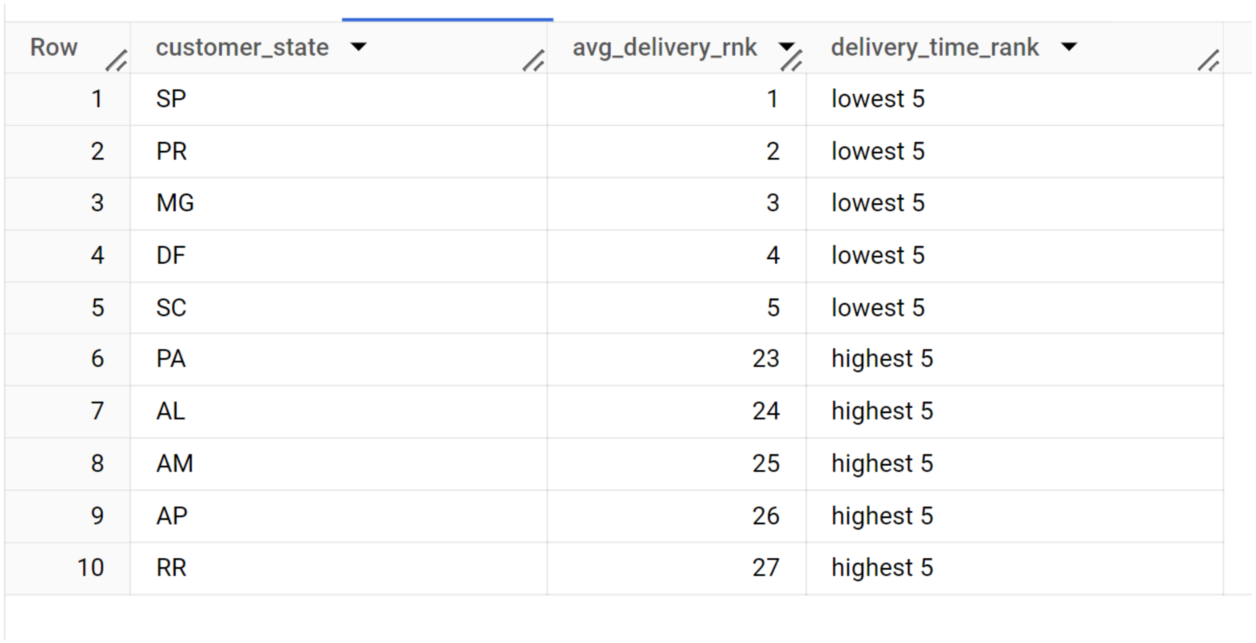


Row	customer_state	freight_value_usg_rn
1	RR	1
2	PB	2
3	RO	3
4	AC	4
5	PI	5
6	DF	23
7	RJ	24
8	MG	25
9	PR	26
10	SP	27

Insights:

1. The above data give us clear info on which states like SP, PR, MG, RJ, DF uses **less avg freight value** compared to RR,PB,RO,AC,PI

```sql
-- 5.3 Find out the top 5 states with the highest & lowest average delivery time.
WITH dr AS (
  select
    c.customer_state
    ,AVG(DATETIME_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY)) AS time_to_deliver
    ,AVG(DATETIME_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)) AS diff_estimated_delivery
    ,DENSE_RANK() OVER(ORDER BY AVG(DATETIME_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY))) AS avg_delivery_rnk
    ,DENSE_RANK() OVER(ORDER BY AVG(DATETIME_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)) DESC) AS avg_delivery_rnk_desc
  FROM bigquery-dinesh.target_business_case.customers c
  JOIN bigquery-dinesh.target_business_case.orders o
  ON c.customer_id = o.customer_id
  GROUP BY c.customer_state
  ORDER BY c.customer_state
)

SELECT customer_state,
  avg_delivery_rnk,
  CASE
    WHEN avg_delivery_rnk <= 5 THEN 'lowest 5'
    WHEN avg_delivery_rnk_desc <=5 THEN 'highest 5'
  END AS delivery_time_rank
FROM dr
WHERE  avg_delivery_rnk <= 5 OR avg_delivery_rnk_desc <=5
ORDER BY avg_delivery_rnk
```

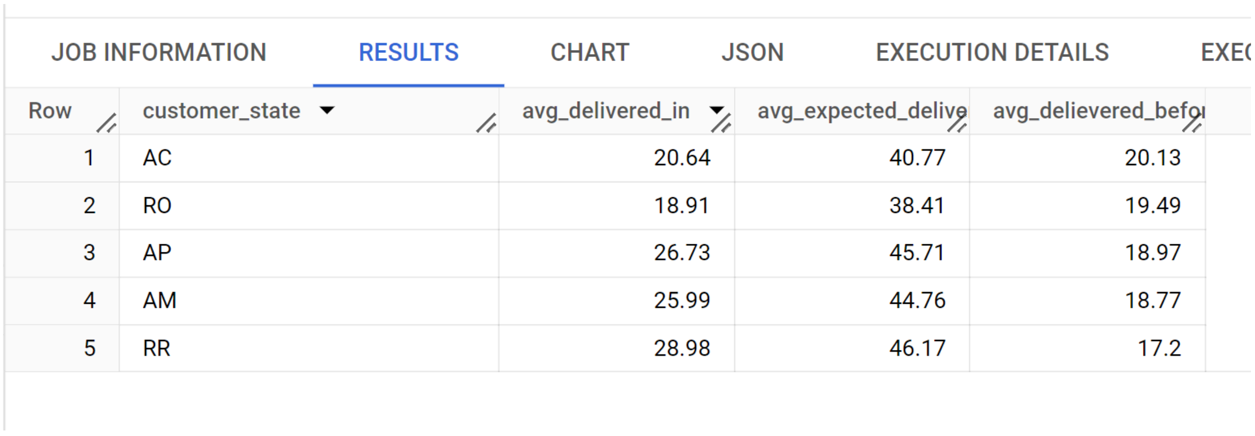


Row	customer_state	avg_delivery_rnk	delivery_time_rank
1	SP	1	lowest 5
2	PR	2	lowest 5
3	MG	3	lowest 5
4	DF	4	lowest 5
5	SC	5	lowest 5
6	PA	23	highest 5
7	AL	24	highest 5
8	AM	25	highest 5
9	AP	26	highest 5
10	RR	27	highest 5

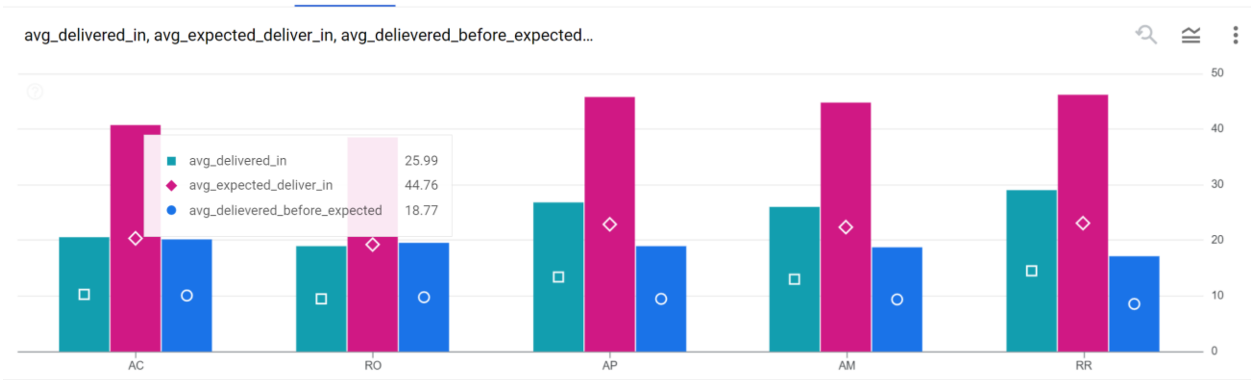
Insights:

1. The state SP, PR, MG, DF, SC uses less time to deliver the product on avg, while PA,AL.AM,AP,RR uses max avg deliver time.

```
-- 5.4 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.

WITH fast_deliver_report AS (
  SELECT c.customer_state
    ,ROUND(AVG(DATETIME_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) AS delivered_time
    ,ROUND(AVG(DATETIME_DIFF(o.order_estimated_delivery_date,
o.order_purchase_timestamp, DAY)),2) AS expected_deliver_avg
    ,ROUND(AVG(DATETIME_DIFF(o.order_estimated_delivery_date,
o.order_purchase_timestamp, DAY))-
    AVG(DATETIME_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY)),2) AS avg_date_delivered_before_expected
    ,DENSE_RANK() over(ORDER BY
ROUND(AVG(DATETIME_DIFF(o.order_estimated_delivery_date, o.order_purchase_timestamp,
DAY))-
    AVG(DATETIME_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY)),2) DESC) as fast_delivery_rank
  FROM bigquery-dinesh.target_business_case.customers c
  JOIN bigquery-dinesh.target_business_case.orders o
  ON c.customer_id = o.customer_id
  GROUP BY c.customer_state
  order BY delivered_time
)

SELECT customer_state
  ,delivered_time as avg_delivered_in
  ,expected_deliver_avg as avg_expected_deliver_in
  ,avg_date_delivered_before_expected as avg_delievered_before_expected
 FROM fast_deliver_report
 WHERE fast_delivery_rank <= 5
order by fast_delivery_rank
```


JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EXE

Row	customer_state	avg_delivered_in	avg_expected_delive	avg_delievered_befo
1	AC	20.64	40.77	20.13
2	RO	18.91	38.41	19.49
3	AP	26.73	45.71	18.97
4	AM	25.99	44.76	18.77
5	RR	28.98	46.17	17.2

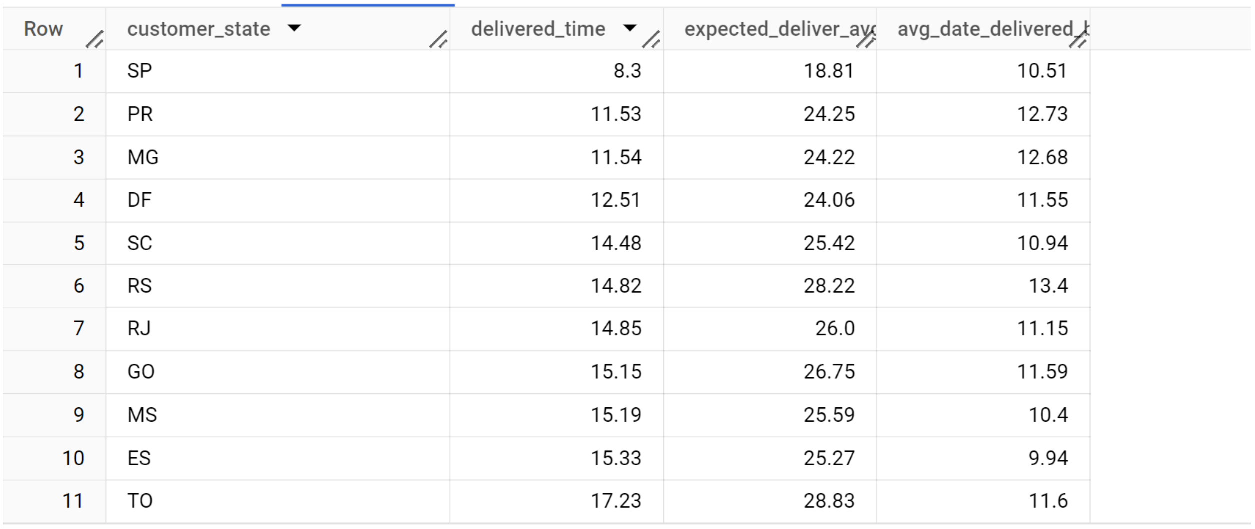


Insights:

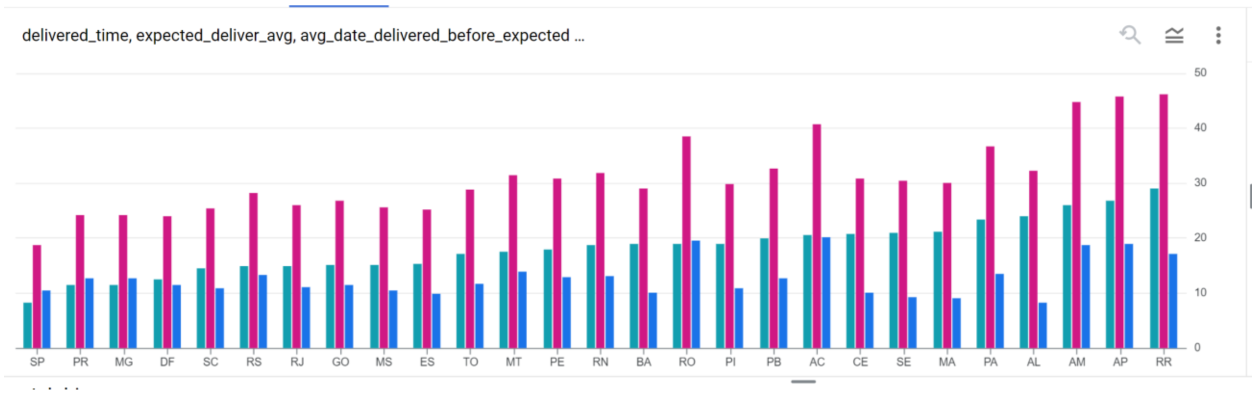
The state AC delivers faster compared to the avg no of days before the estimated delivery, followed by RO,AP,AM,RR

Additional INFO for the same Question

```
SELECT c.customer_state
  ,ROUND(AVG(DATETIME_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) AS delivered_time
  ,ROUND(AVG(DATETIME_DIFF(o.order_estimated_delivery_date,
o.order_purchase_timestamp, DAY)),2) AS expected_deliver_avg
  ,ROUND(AVG(DATETIME_DIFF(o.order_estimated_delivery_date,
o.order_purchase_timestamp, DAY))-
   AVG(DATETIME_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY)),2) AS avg_date_delivered_before_expected
FROM bigquery-dinesh.target_business_case.customers c
JOIN bigquery-dinesh.target_business_case.orders o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
order BY delivered_time
```



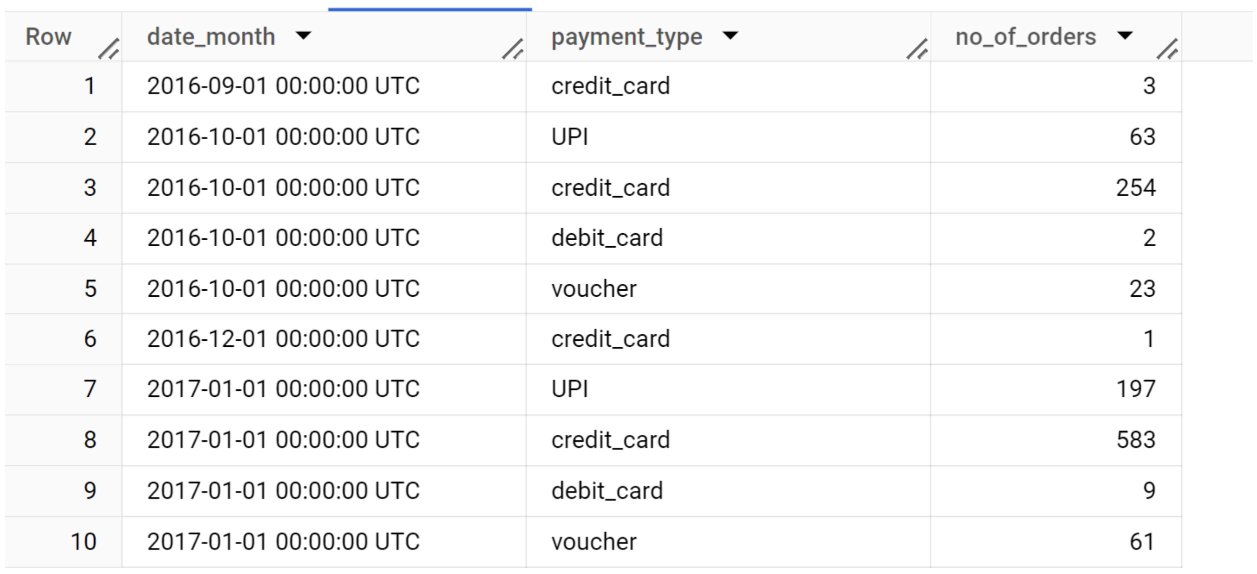
Row	customer_state	delivered_time	expected_deliver_av	avg_date_delivered_b
1	SP	8.3	18.81	10.51
2	PR	11.53	24.25	12.73
3	MG	11.54	24.22	12.68
4	DF	12.51	24.06	11.55
5	SC	14.48	25.42	10.94
6	RS	14.82	28.22	13.4
7	RJ	14.85	26.0	11.15
8	GO	15.15	26.75	11.59
9	MS	15.19	25.59	10.4
10	ES	15.33	25.27	9.94
11	TO	17.23	28.83	11.6



Insights:

1. The above graph shows the avg no. of days took to deliver the order, avg estimated no of days and avg no of days before which the order is delivered.

```
-- 6.1 Find the month on month no. of orders placed using different payment types.
SELECT
 date_trunc(o.order_purchase_timestamp, month) as date_month
 ,p.payment_type
 ,count(o.order_id) as no_of_orders

FROM bigquery-dinesh.target_business_case.orders o
JOIN bigquery-dinesh.target_business_case.payments p
ON p.order_id = o.order_id
GROUP BY date_month, p.payment_type
order by date_month, p.payment_type
```



Row	date_month	payment_type	no_of_orders
1	2016-09-01 00:00:00 UTC	credit_card	3
2	2016-10-01 00:00:00 UTC	UPI	63
3	2016-10-01 00:00:00 UTC	credit_card	254
4	2016-10-01 00:00:00 UTC	debit_card	2
5	2016-10-01 00:00:00 UTC	voucher	23
6	2016-12-01 00:00:00 UTC	credit_card	1
7	2017-01-01 00:00:00 UTC	UPI	197
8	2017-01-01 00:00:00 UTC	credit_card	583
9	2017-01-01 00:00:00 UTC	debit_card	9
10	2017-01-01 00:00:00 UTC	voucher	61

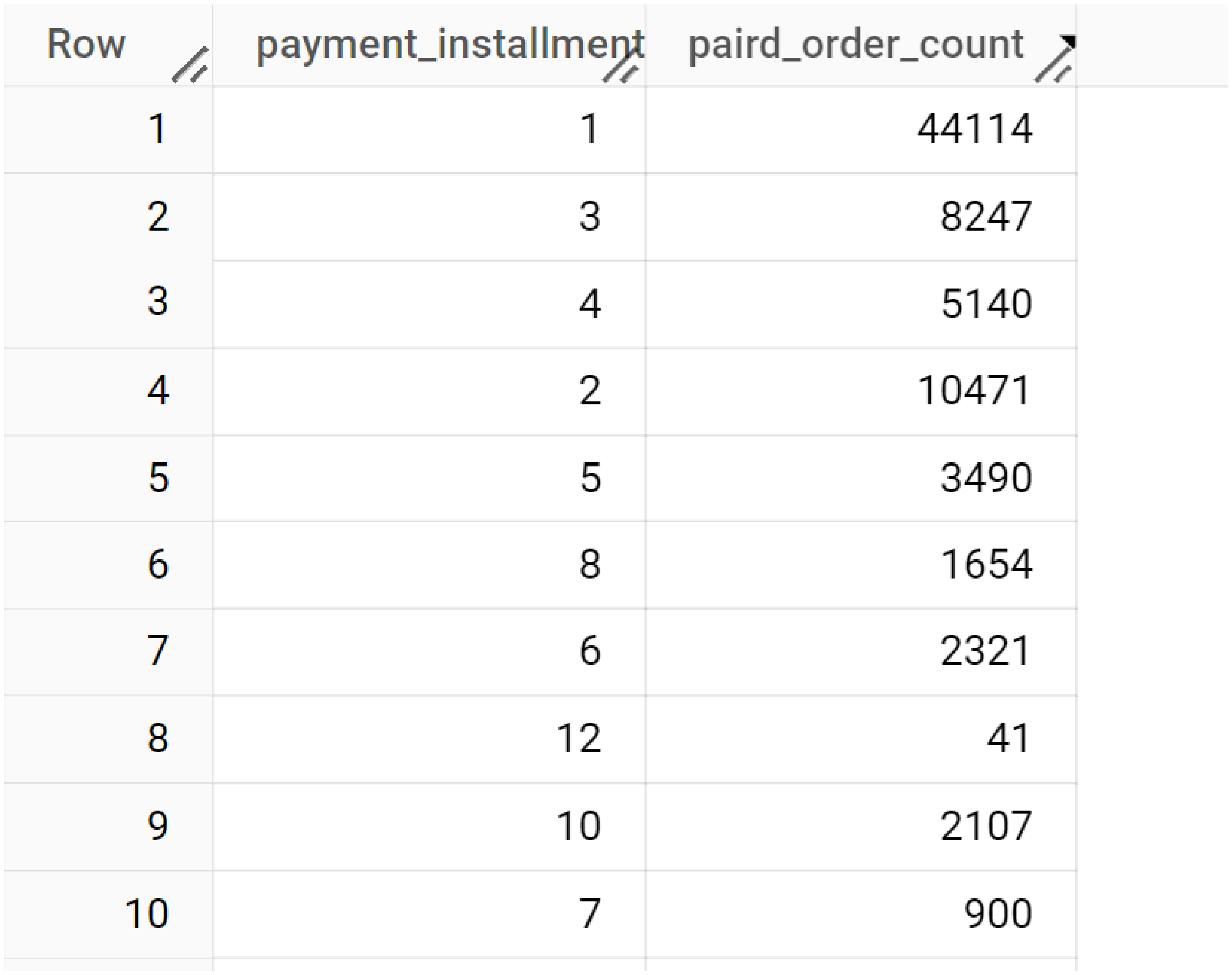
Insights:

1. The above data indicates no of payments done but different payment modes.
2. We can visual the no of payments more clearly in the bar graph or line graph with
    a. X-axis as date_month
    b. Y-axis as no_of_orders for each payment_type

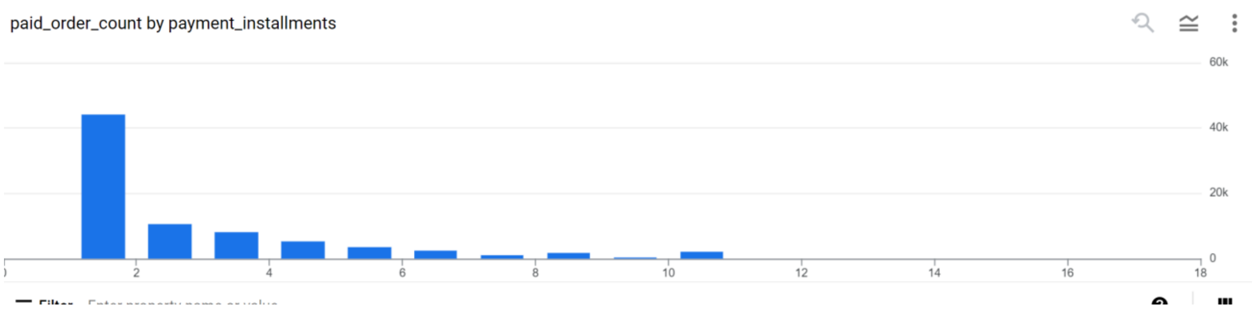
```
-- 6.2 Find the no. of orders placed on the basis of the payment installments that
have been paid.


with cte as (
  select o.*, p.payment_installments,
  max(o.order_purchase_timestamp) over() as last_recorded_date
  FROM bigquery-dinesh.target_business_case.orders o
  JOIN bigquery-dinesh.target_business_case.payments p
  ON p.order_id = o.order_id
  WHERE lower(o.order_status) = 'delivered'
),

paid_orders as (

  select order_id
    ,order_purchase_timestamp
    ,payment_installments
    ,last_recorded_date
    ,DATE_DIFF(last_recorded_date, order_purchase_timestamp, DAY) / 30 AS
months_since_last_order

  from cte
  order by order_purchase_timestamp
)
```

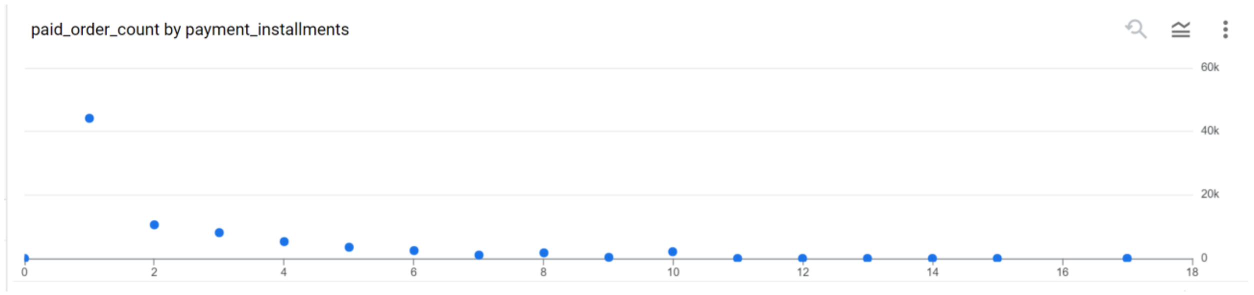
```
select payment_installments,
  count(distinct order_id) as paird_order_count
from paid_orders
where payment_installments <= months_since_last_order
group by payment_installments
```

This will get all the orders which were paid completely



Row	payment_installment	paird_order_count
1	1	44114
2	3	8247
3	4	5140
4	2	10471
5	5	3490
6	8	1654
7	6	2321
8	12	41
9	10	2107
10	7	900





Insights:

1. The above data shows the order count for fully paid,
    a. Example: lets say there is an order taken on July 2018 with 12 month payment, we are not considering since we don't have full data for the same.
2. As per the data most of the customers paid in 1 installment, which indicates targeting those customers by also checking the amount could give the more in cash instead of going for 0 emi in case.

Actionable Insights & Recommendations

1. There are a greater number of customers paying in single installments or 2 installments, gathering info about them, checking for the frequency at which these customers buy the products, the type of products these customers prefer would give the business more revenue.
2. Most of the people are using the credit card for payments, having offers on credit specific to the business will grow the business.
3. Reducing the delivery time to States like RR, AL, AM would draw us more customers.
4. We have already a fair number of customers from SP, targeting this fore more customers will reduce the freight_value drastically as no of orders increase.
5. There is a deliver delay percentage of around 7%, decreasing to 3 to4 % would increase the no of customers buying in those cities.
6. Also proving the offers in the afternoon would draw us more no of customers buying the products since from the data it is evident that a large fair number of customers are preferring to buy in the afternoon followed by mornings and night.
7. Also there is a steep fall in the orders in sep 2018, finding the root cause could give us the actual reasons.