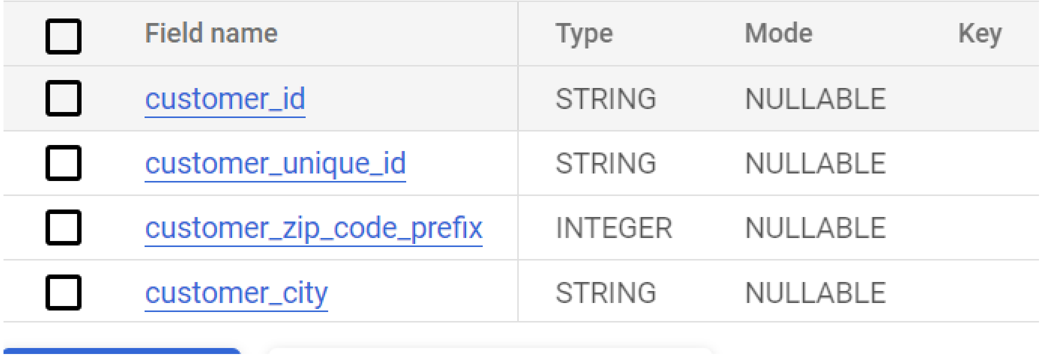
Business Case: Target SQL

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

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

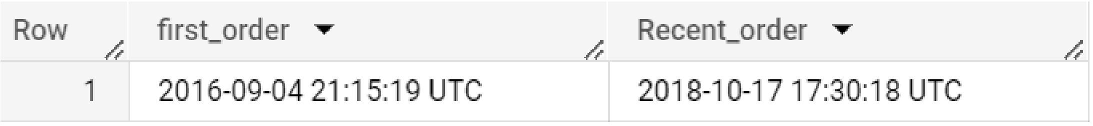


□	Field name	Type	Mode	Key
□	customer_id	STRING	NULLABLE	
□	customer_unique_id	STRING	NULLABLE	
□	customer_zip_code_prefix	INTEGER	NULLABLE	
□	customer_city	STRING	NULLABLE	

-> Data types are String and Integer in Customer Table.

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

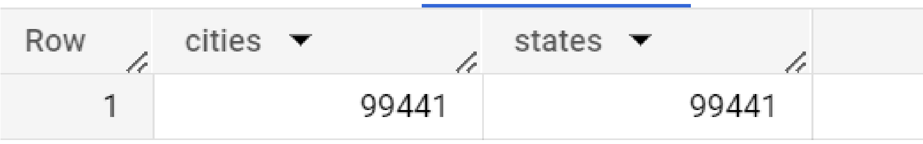
```
select min(order_purchase_timestamp) as first_order,
max(order_purchase_timestamp) as Recent_order
from   `TargetBC.orders`
```



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

- Count the number of Cities and States in our dataset.

```
select count(customer_city)as cities,count(customer_state)as states
from   `TargetBC.customer`
```

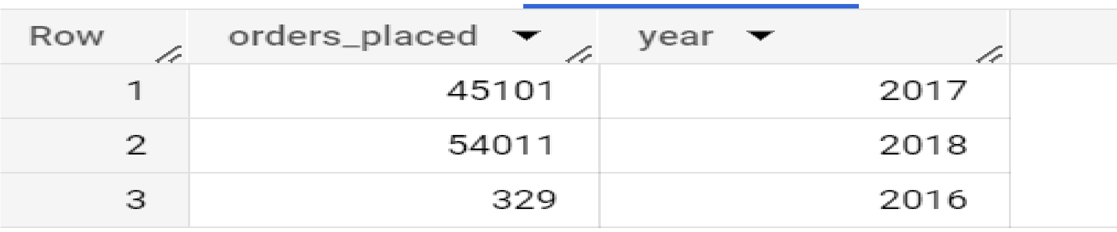


Row	cities	states
1	99441	99441

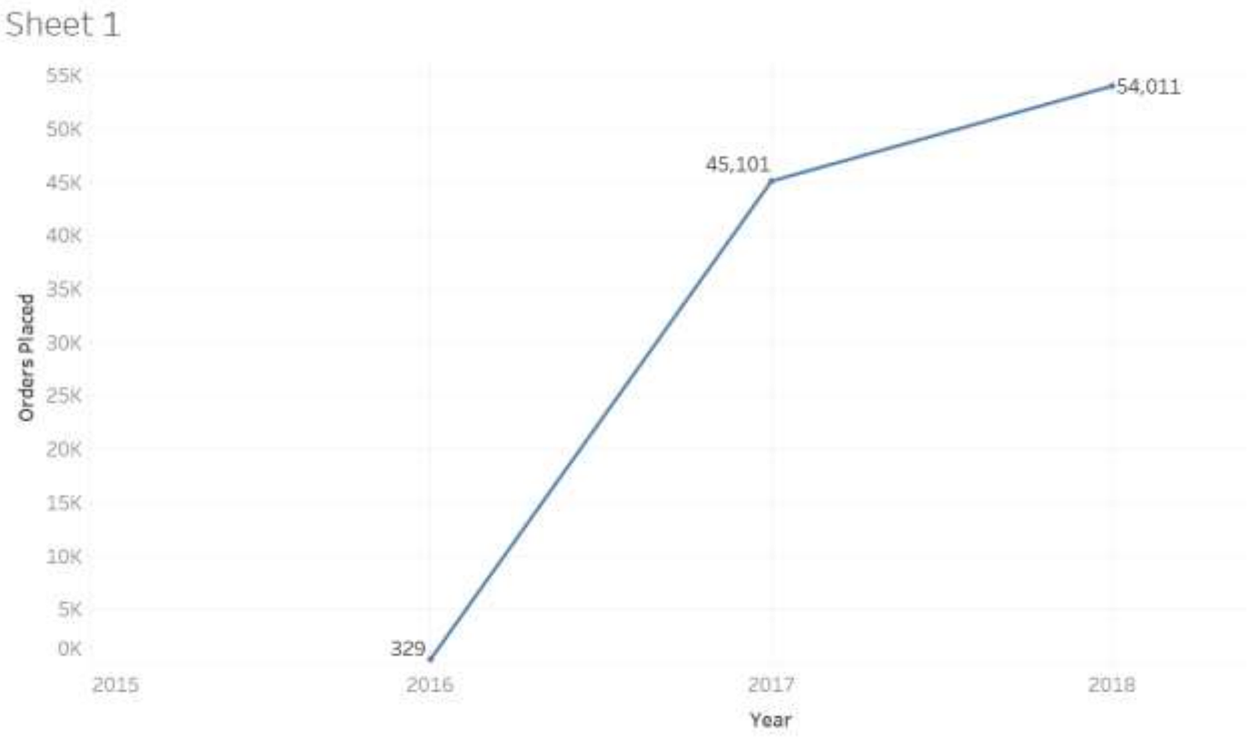
In-depth Exploration:

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

```
select count(order_id)as orders_placed, Extract(year from order_purchase_timestamp )
as year  from `TargetBC.orders`
group by Extract(year from order_purchase_timestamp )
```



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

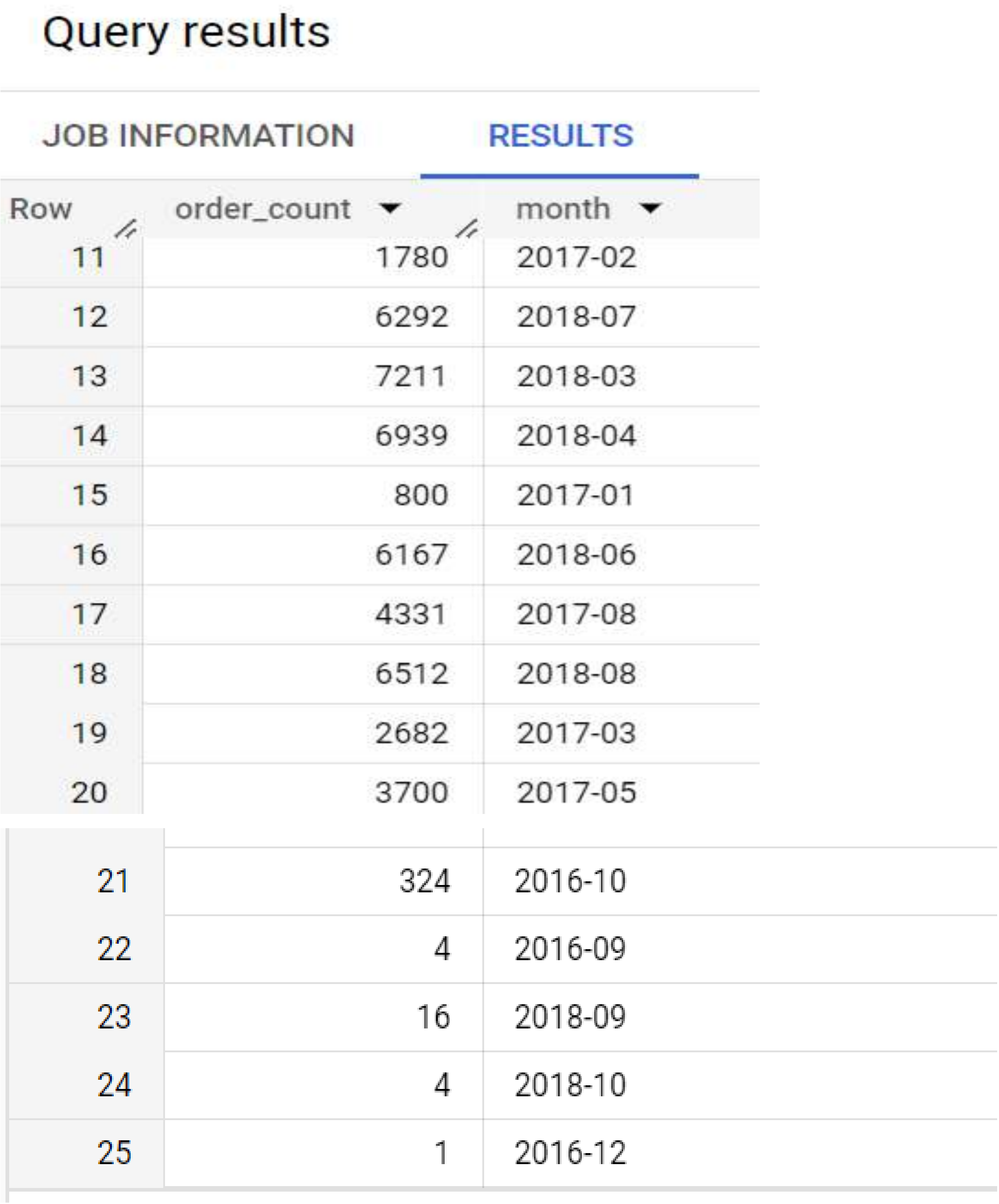


It appears that there was a significant increase in the number of orders placed between 2016 and 2017, followed by a smaller increase from 2017 to 2018. However, please note that this data is based on only three years, and it may not be representative of a larger trend. To have a more comprehensive understanding of the trend, it would be helpful to have data from additional years or a longer time period.

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

```
SELECT COUNT(order_id) AS order_count,
FORMAT_TIMESTAMP('%Y-%m', order_purchase_timestamp) AS month
FROM `TargetBC.orders`
GROUP BY month;

OR

SELECT count(order_id) as order_count,
extract(month from order_purchase_timestamp)as month,
extract(year from order_purchase_timestamp) as year

from  `TargetBC.orders`
group by month,year
order by month,year;
```

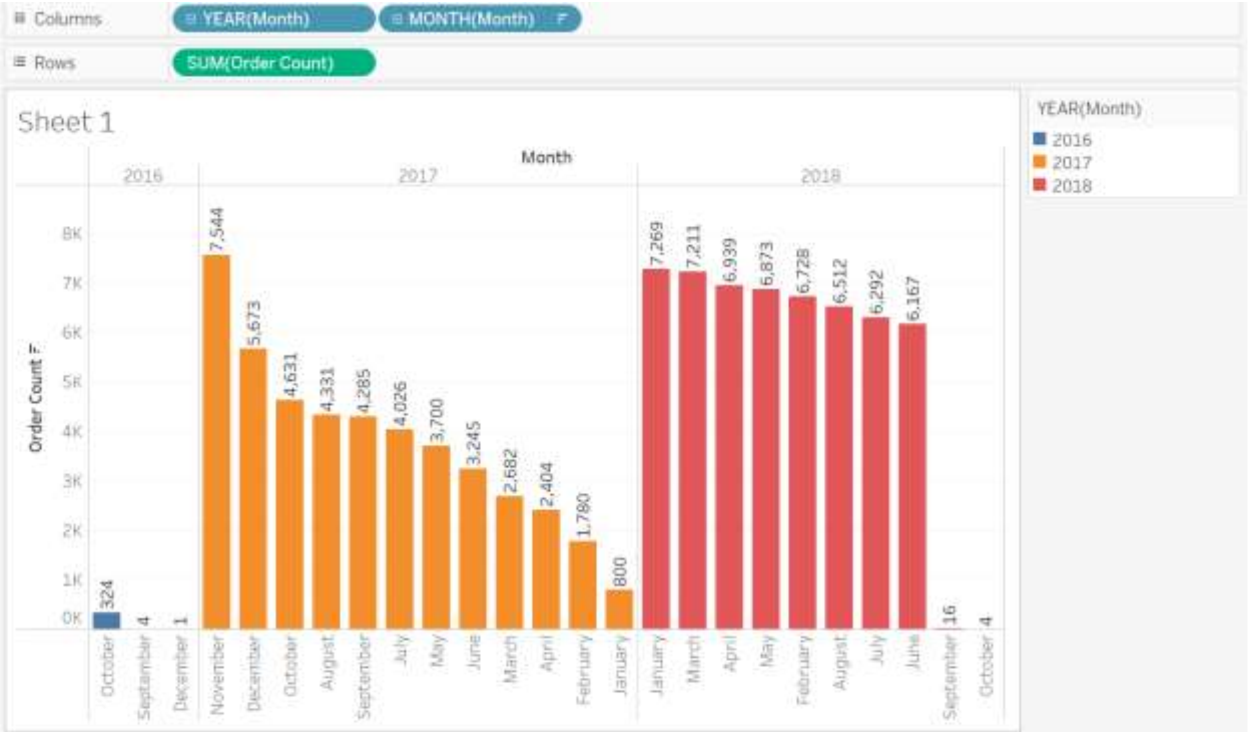
Result:



Query results

Row	order_count	month
11	1780	2017-02
12	6292	2018-07
13	7211	2018-03
14	6939	2018-04
15	800	2017-01
16	6167	2018-06
17	4331	2017-08
18	6512	2018-08
19	2682	2017-03
20	3700	2017-05
21	324	2016-10
22	4	2016-09
23	16	2018-09
24	4	2018-10
25	1	2016-12

Analysis Report:



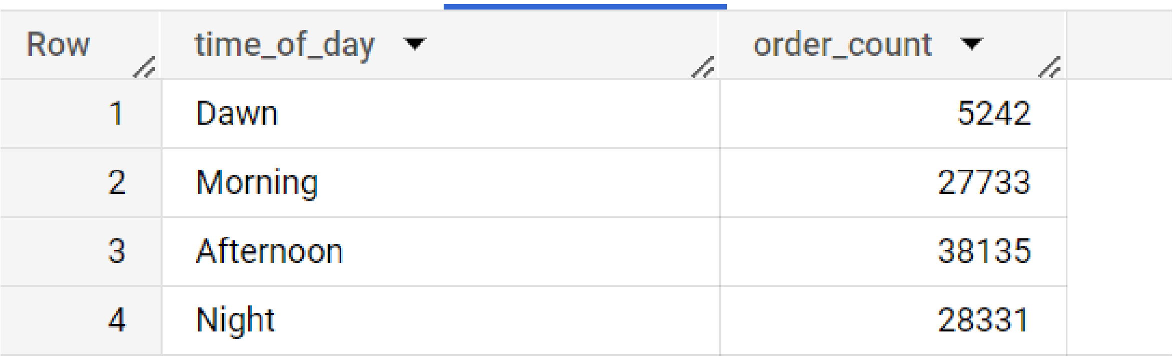
Based on the data provided, we can observe the following patterns and trends in the number of orders being placed:

- Seasonal Fluctuations: There are variations in the number of orders placed across different months. For example, in the months of November 2017 (order count: 7544), December 2017 (order count: 5673), and February 2018 (order count: 6728), there is a relatively higher number of orders compared to other months.
- Monthly Variability: The order counts fluctuate from month to month, indicating that there may be factors influencing customer behavior or demand that vary throughout the year.

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

```
SELECT
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 'Morning'
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,
COUNT(*) AS order_count
FROM
`TargetBC.orders`
GROUP BY
time_of_day
ORDER BY
MIN(EXTRACT(HOUR FROM order_purchase_timestamp));
```



Row	time_of_day	order_count
1	Dawn	5242
2	Morning	27733
3	Afternoon	38135
4	Night	28331

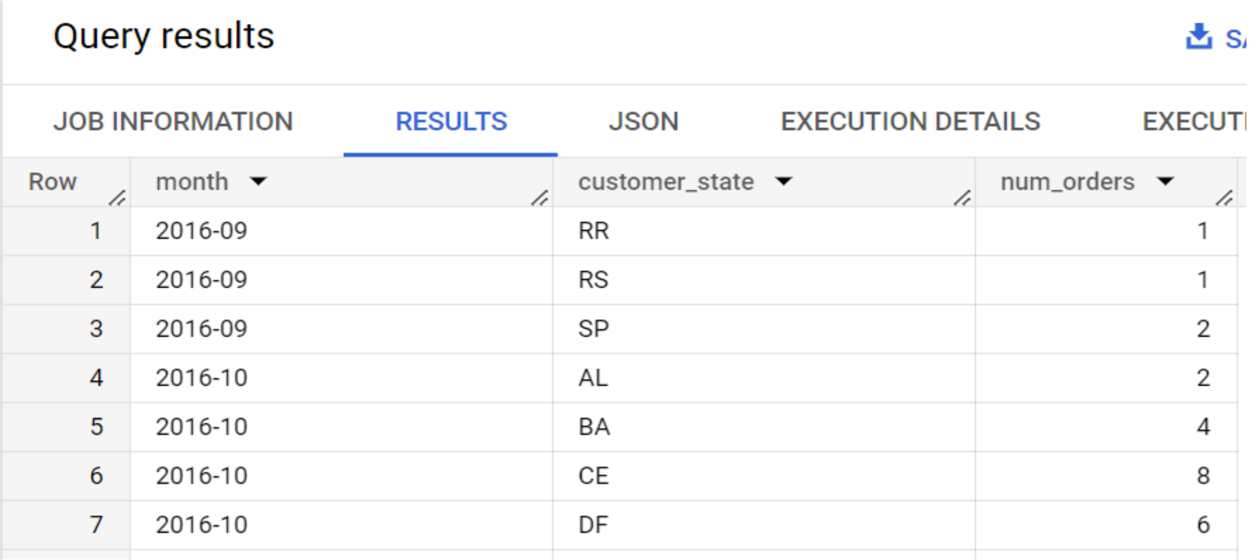
- As the highest order count is in the Afternoon category, it suggests that Brazilian customers predominantly place their orders during that time of day.

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

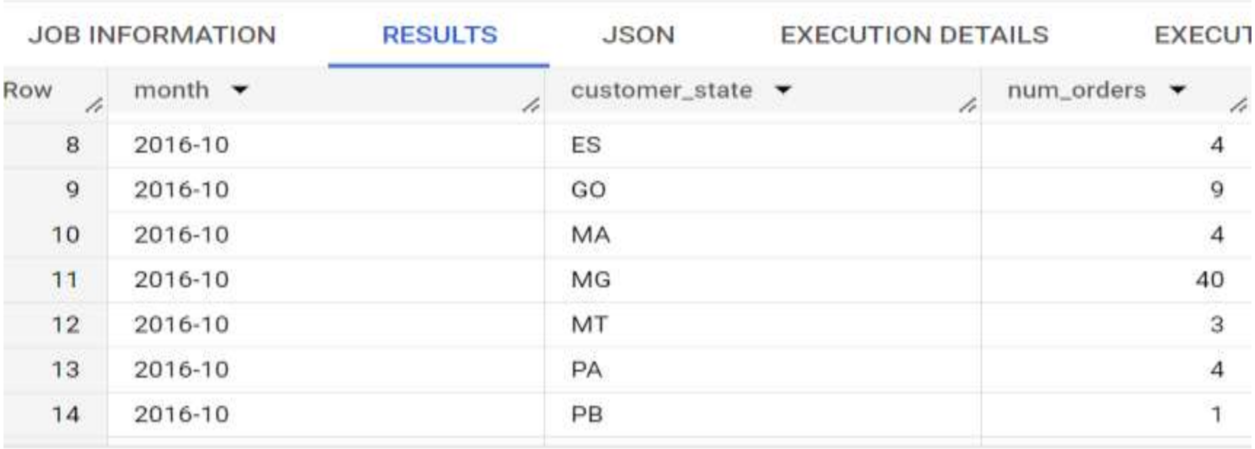
1. Get the month on month no. of orders placed in each state.

```
SELECT
FORMAT_TIMESTAMP('%Y-%m', order_purchase_timestamp)  AS month,
c.customer_state,
COUNT(DISTINCT o.order_id) AS num_orders
FROM
`TargetBC.orders` AS o
JOIN
`TargetBC.customer` AS c ON o.customer_id = c.customer_id
GROUP BY
month,
c.customer_state
ORDER BY
month,
c.customer_state;
```

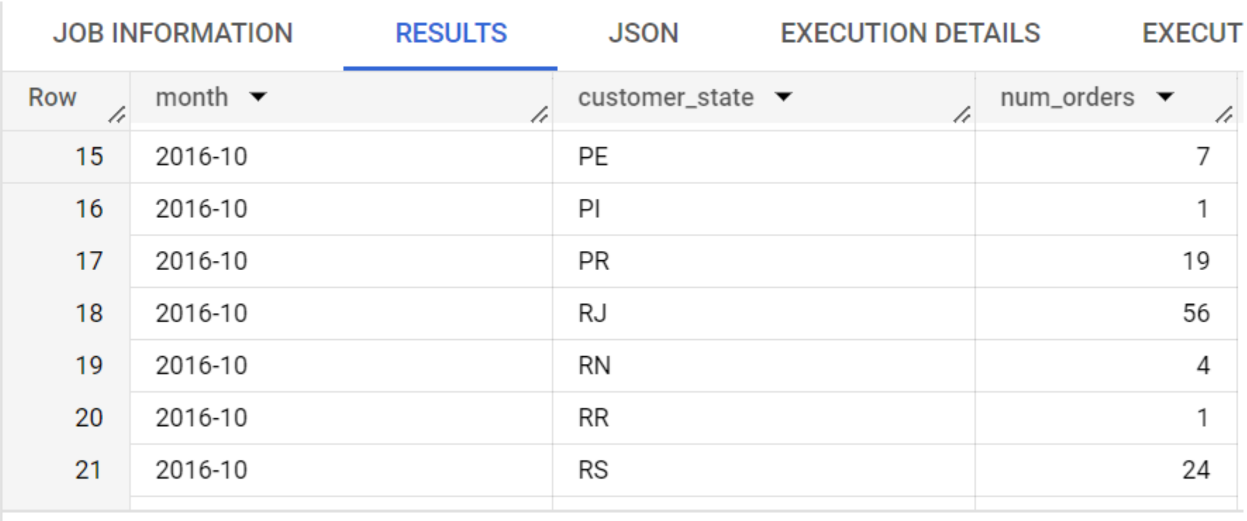
Result:



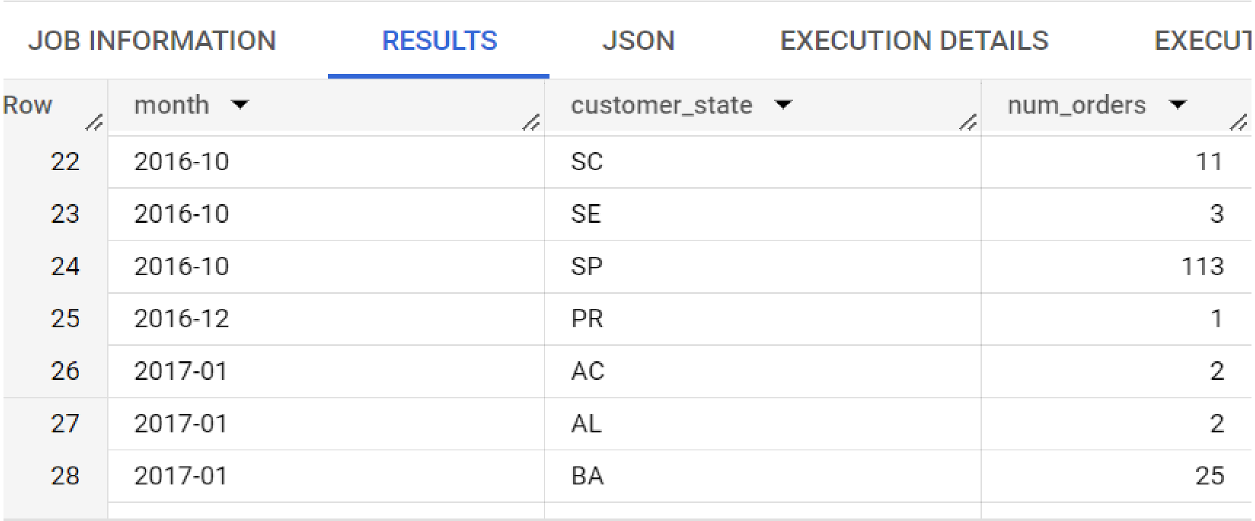
Row	month	customer_state	num_orders
1	2016-09	RR	1
2	2016-09	RS	1
3	2016-09	SP	2
4	2016-10	AL	2
5	2016-10	BA	4
6	2016-10	CE	8
7	2016-10	DF	6



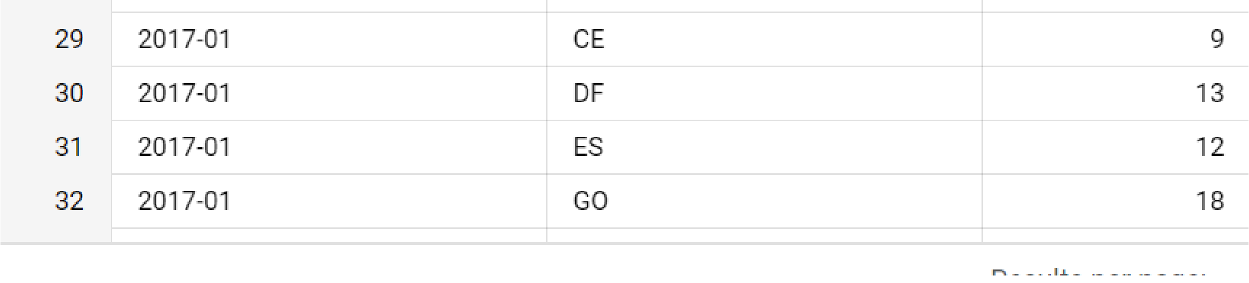
Row	month	customer_state	num_orders
8	2016-10	ES	4
9	2016-10	GO	9
10	2016-10	MA	4
11	2016-10	MG	40
12	2016-10	MT	3
13	2016-10	PA	4
14	2016-10	PB	1



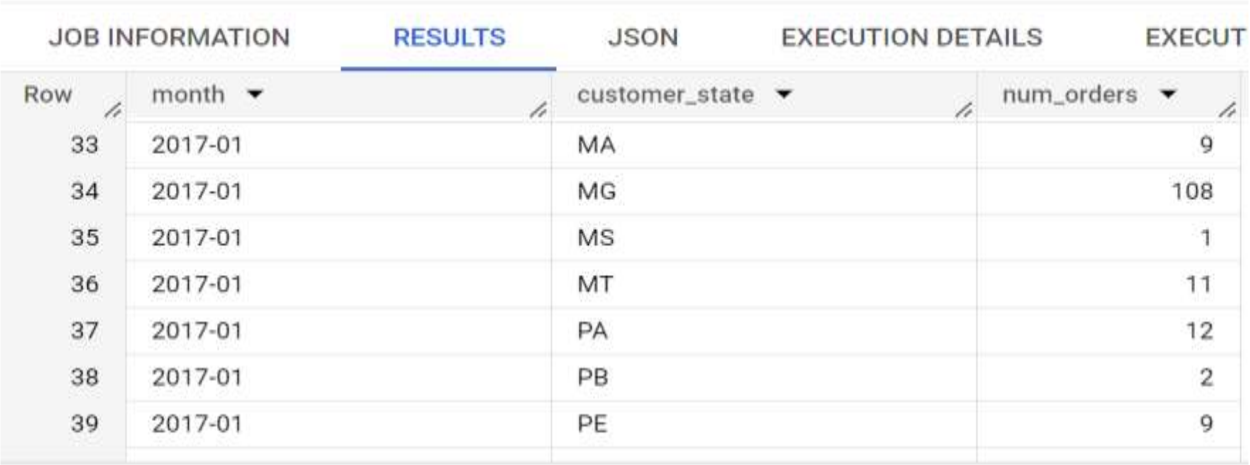
Row	month	customer_state	num_orders
15	2016-10	PE	7
16	2016-10	PI	1
17	2016-10	PR	19
18	2016-10	RJ	56
19	2016-10	RN	4
20	2016-10	RR	1
21	2016-10	RS	24



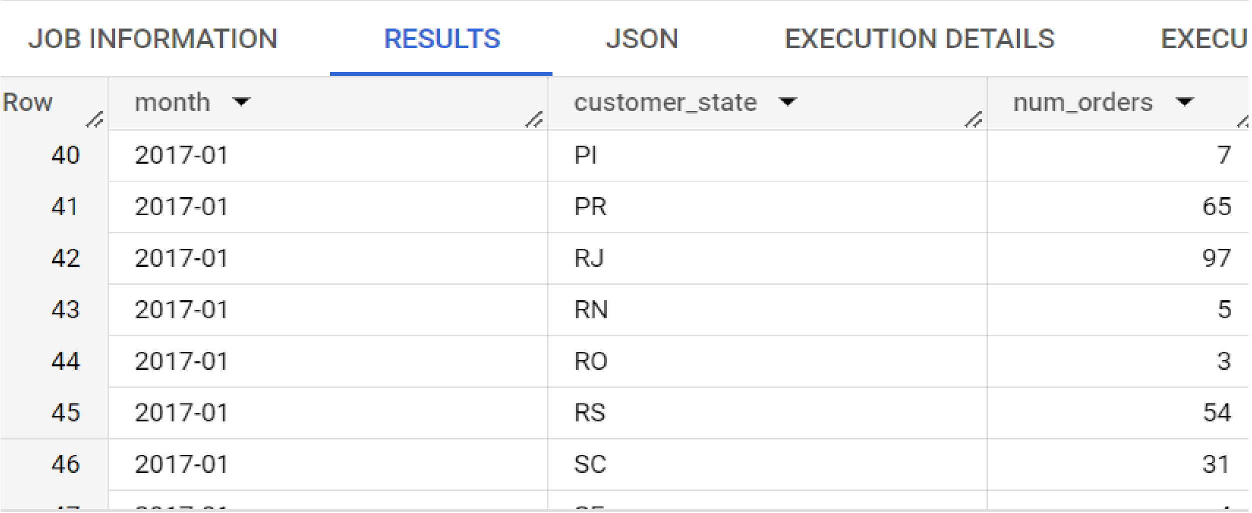
Row	month	customer_state	num_orders
22	2016-10	SC	11
23	2016-10	SE	3
24	2016-10	SP	113
25	2016-12	PR	1
26	2017-01	AC	2
27	2017-01	AL	2
28	2017-01	BA	25



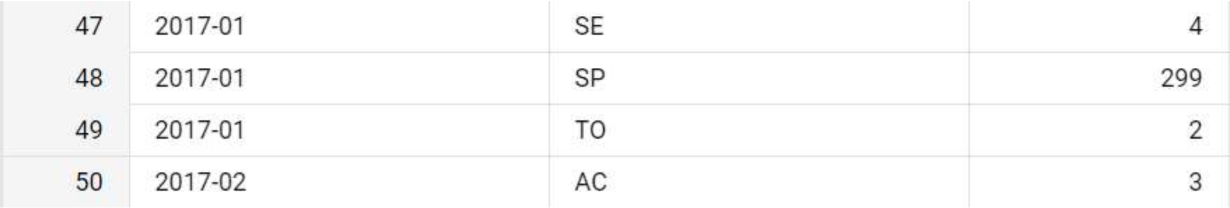
29	2017-01	CE	9
30	2017-01	DF	13
31	2017-01	ES	12
32	2017-01	GO	18



Row	month	customer_state	num_orders
33	2017-01	MA	9
34	2017-01	MG	108
35	2017-01	MS	1
36	2017-01	MT	11
37	2017-01	PA	12
38	2017-01	PB	2
39	2017-01	PE	9



Row	month	customer_state	num_orders
40	2017-01	PI	7
41	2017-01	PR	65
42	2017-01	RJ	97
43	2017-01	RN	5
44	2017-01	RO	3
45	2017-01	RS	54
46	2017-01	SC	31

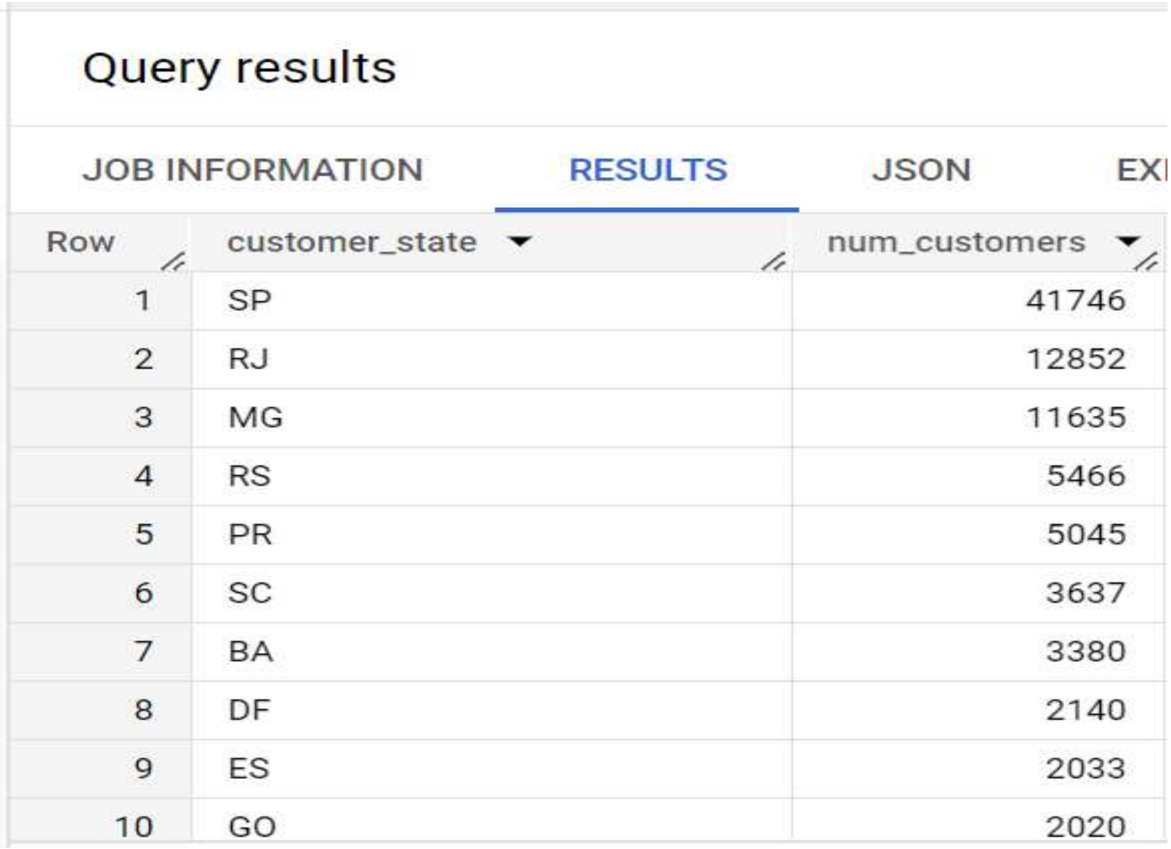


47	2017-01	SE	4
48	2017-01	SP	299
49	2017-01	TO	2
50	2017-02	AC	3

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

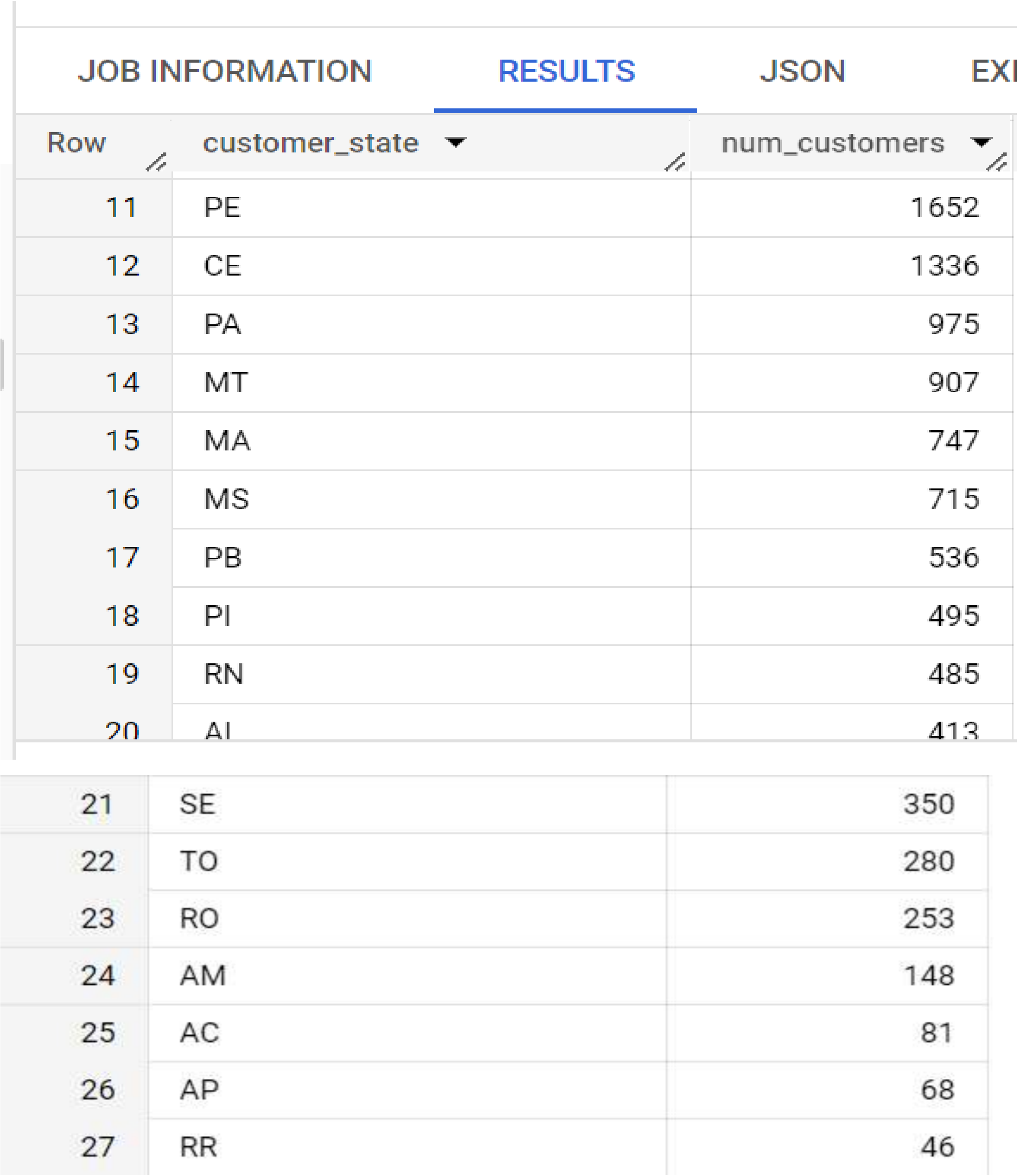
```
SELECT
customer_state,
COUNT(DISTINCT customer_id) AS num_customers
FROM
`TargetBC.customer`
GROUP BY
customer_state
ORDER BY
num_customers DESC;
```

Result:



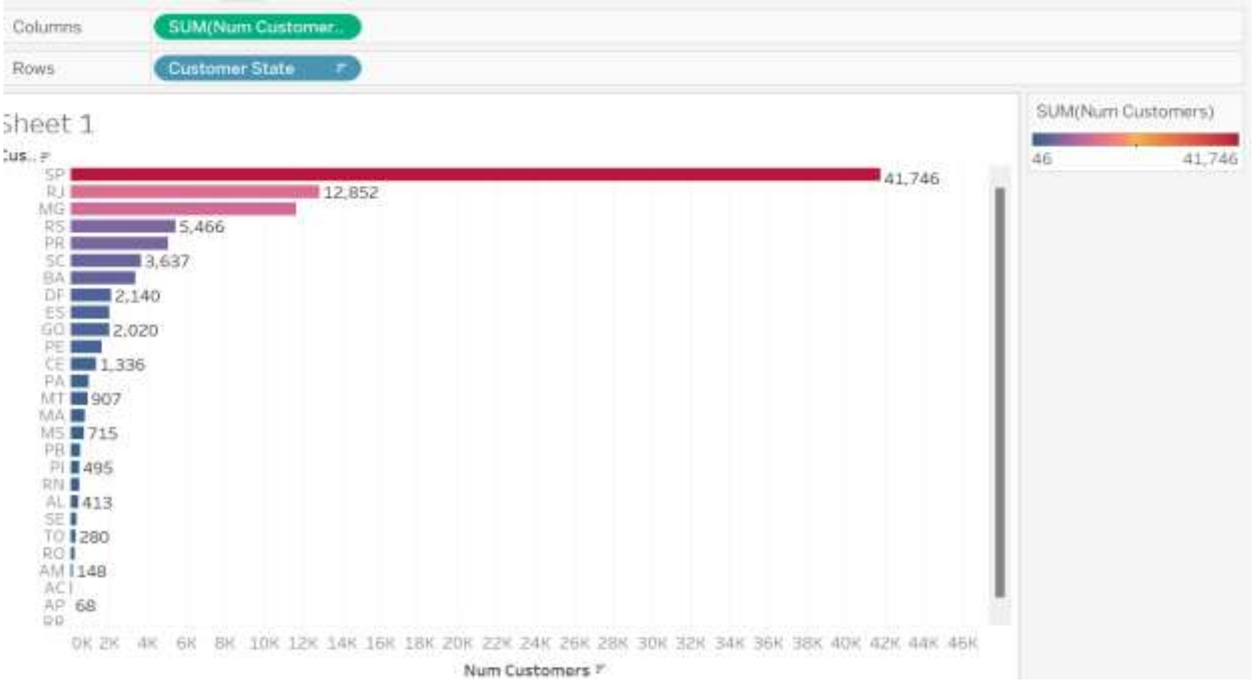
Query results

Row	customer_state	num_customers
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



Row	customer_state	num_customers
11	PE	1652
12	CE	1336
13	PA	975
14	MT	907
15	MA	747
16	MS	715
17	PB	536
18	PI	495
19	RN	485
20	AL	413
21	SE	350
22	TO	280
23	RO	253
24	AM	148
25	AC	81
26	AP	68
27	RR	46

Report Analysis:

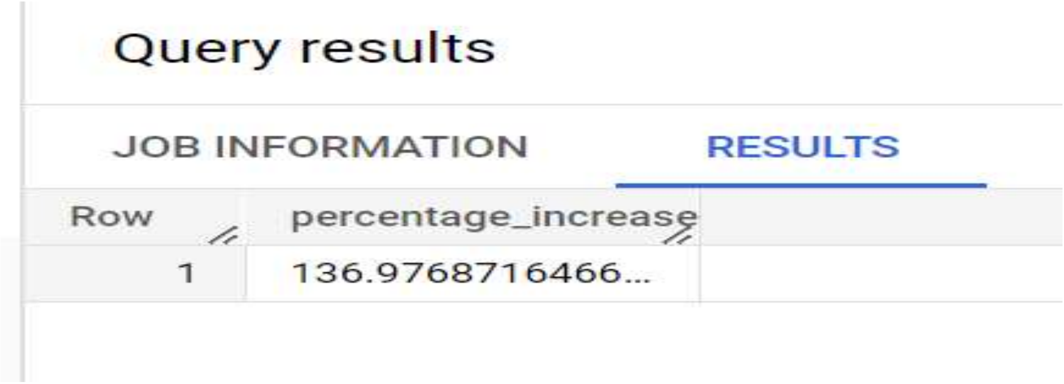


Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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.

```
SELECT
((total_payment_2018 - total_payment_2017) / total_payment_2017) * 100 AS
percentage_increase
FROM
(
SELECT
SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND EXTRACT(MONTH
FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN p.payment_value ELSE 0 END) AS
total_payment_2017,
SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 AND EXTRACT(MONTH
FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN p.payment_value ELSE 0 END) AS
total_payment_2018
FROM
`TargetBC.orders` AS o
JOIN `TargetBC.payments` AS p ON o.order_id = p.order_id
WHERE
EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)
) AS subquery;
```

Result:



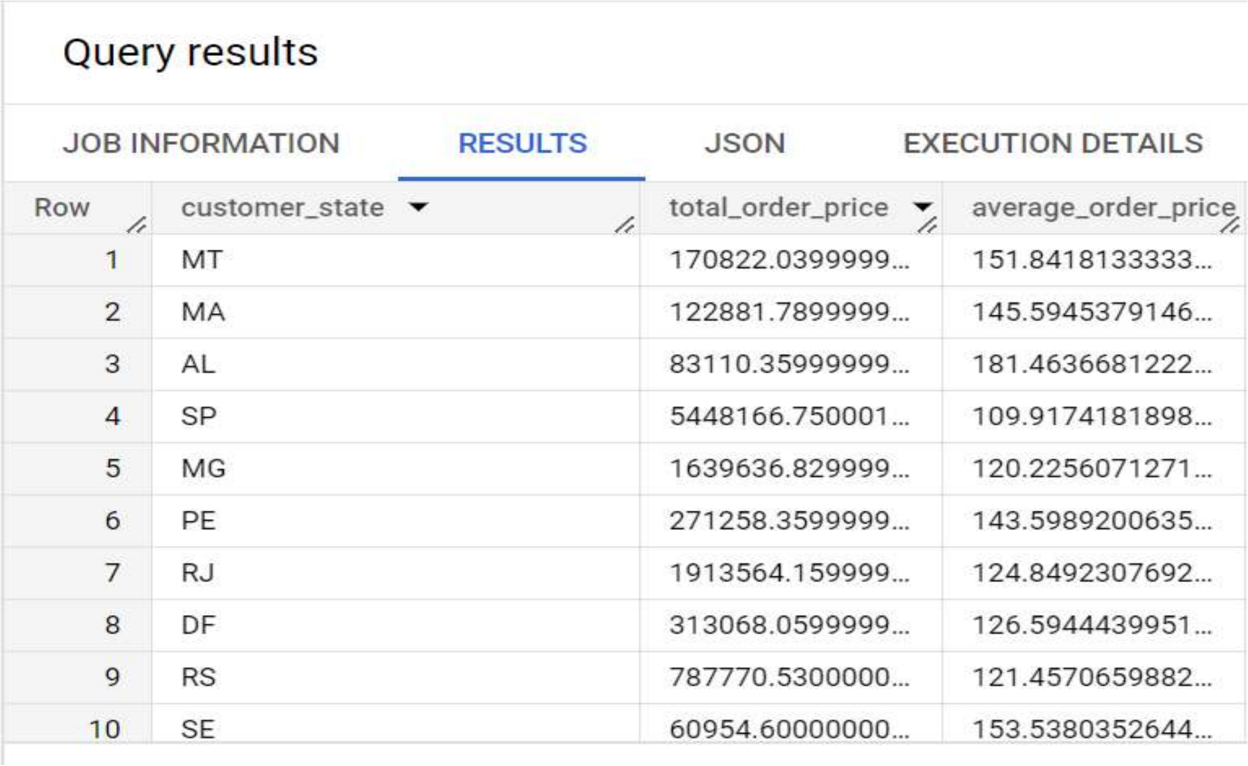
Row	percentage_increase
1	136.9768716466...

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

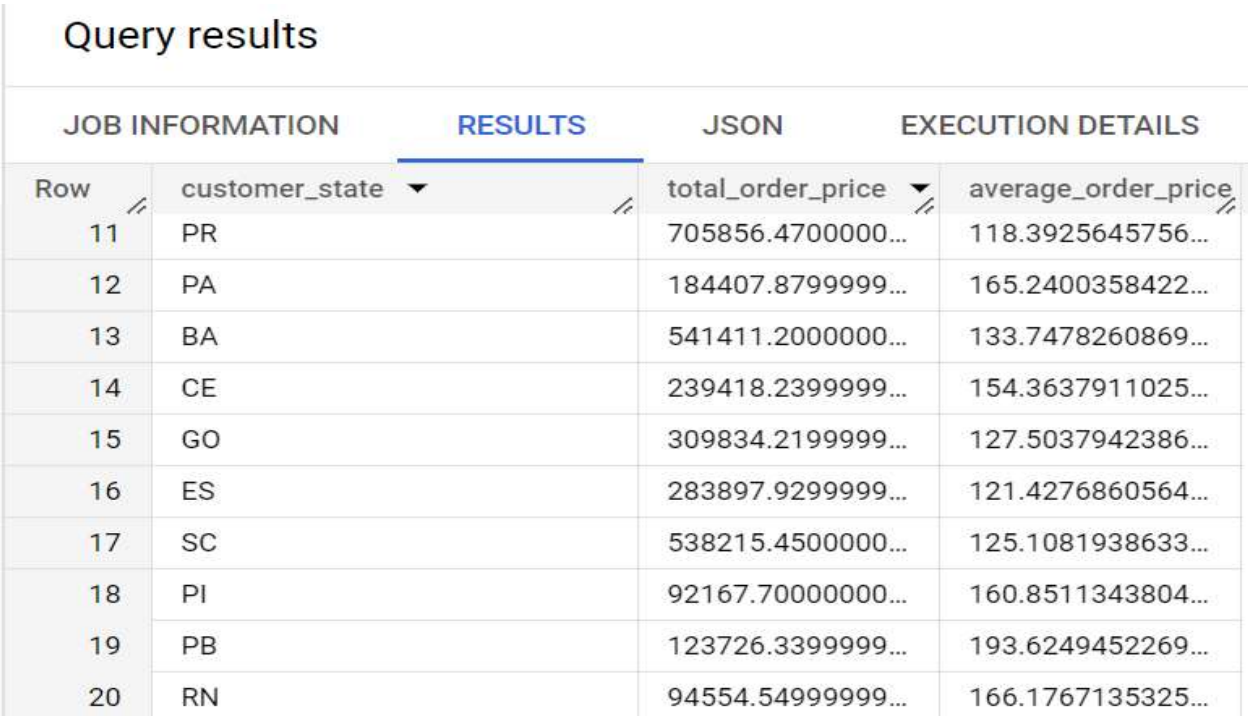
```
SELECT
c.customer_state,
SUM(oi.price) AS total_order_price,
AVG(oi.price) AS average_order_price
FROM
`TargetBC.customer` AS c
JOIN `TargetBC.orders` AS o ON c.customer_id = o.customer_id
JOIN `TargetBC.order_items` AS oi ON o.order_id = oi.order_id
JOIN `TargetBC.payments` AS p ON o.order_id = p.order_id
```

```
GROUP BY
c.customer_state;
```

Result:

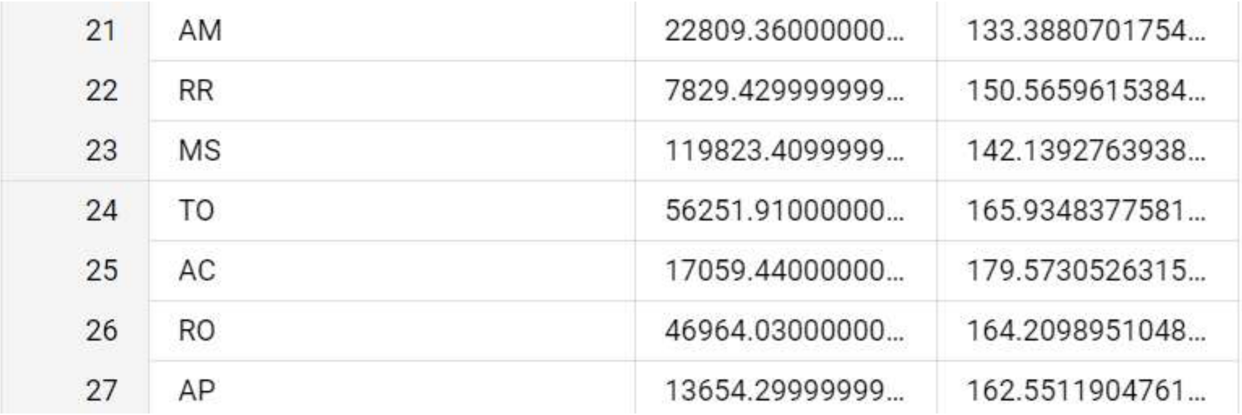


Row	customer_state	total_order_price	average_order_price
1	MT	170822.0399999...	151.8418133333...
2	MA	122881.7899999...	145.5945379146...
3	AL	83110.35999999...	181.4636681222...
4	SP	5448166.750001...	109.9174181898...
5	MG	1639636.829999...	120.2256071271...
6	PE	271258.3599999...	143.5989200635...
7	RJ	1913564.159999...	124.8492307692...
8	DF	313068.0599999...	126.5944439951...
9	RS	787770.5300000...	121.4570659882...
10	SE	60954.60000000...	153.5380352644...



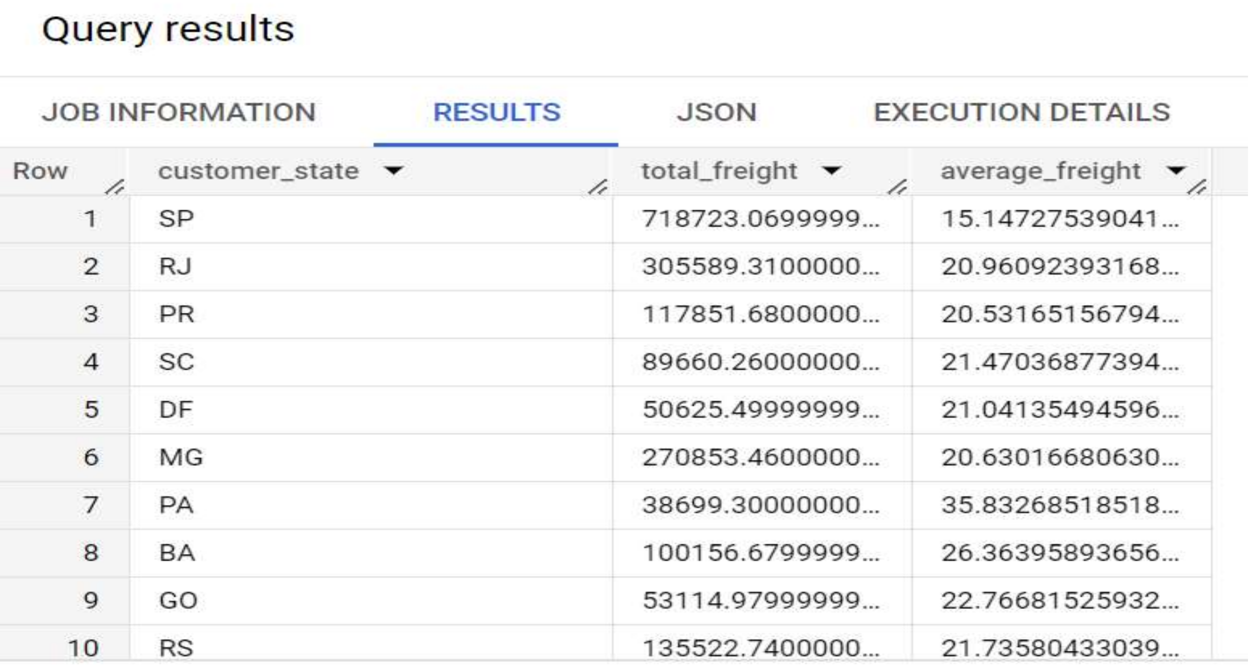
Query results

Row	customer_state	total_order_price	average_order_price
11	PR	705856.4700000...	118.3925645756...
12	PA	184407.8799999...	165.2400358422...
13	BA	541411.2000000...	133.7478260869...
14	CE	239418.2399999...	154.3637911025...
15	GO	309834.2199999...	127.5037942386...
16	ES	283897.9299999...	121.4276860564...
17	SC	538215.4500000...	125.1081938633...
18	PI	92167.70000000...	160.8511343804...
19	PB	123726.3399999...	193.6249452269...
20	RN	94554.54999999...	166.1767135325...



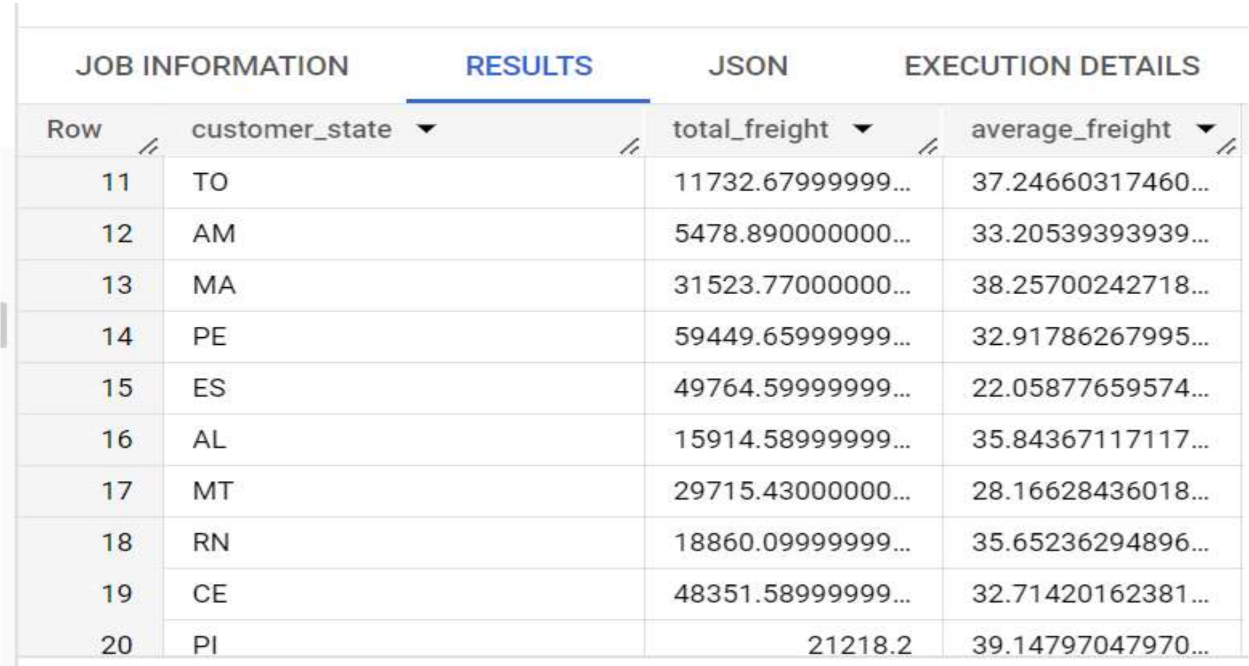
21	AM	22809.36000000...	133.3880701754...
22	RR	7829.429999999...	150.5659615384...
23	MS	119823.4099999...	142.1392763938...
24	TO	56251.91000000...	165.9348377581...
25	AC	17059.44000000...	179.5730526315...
26	RO	46964.03000000...	164.2098951048...
27	AP	13654.29999999...	162.5511904761...

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

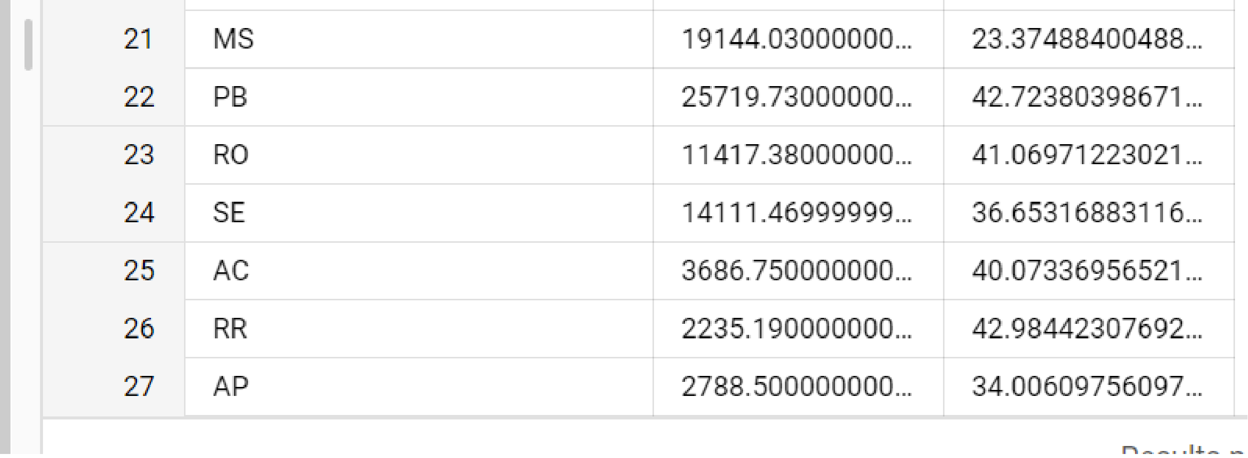


Query results

Row	customer_state	total_freight	average_freight
1	SP	718723.0699999...	15.14727539041...
2	RJ	305589.3100000...	20.96092393168...
3	PR	117851.6800000...	20.53165156794...
4	SC	89660.26000000...	21.47036877394...
5	DF	50625.49999999...	21.04135494596...
6	MG	270853.4600000...	20.63016680630...
7	PA	38699.30000000...	35.83268518518...
8	BA	100156.6799999...	26.36395893656...
9	GO	53114.97999999...	22.76681525932...
10	RS	135522.7400000...	21.73580433039...

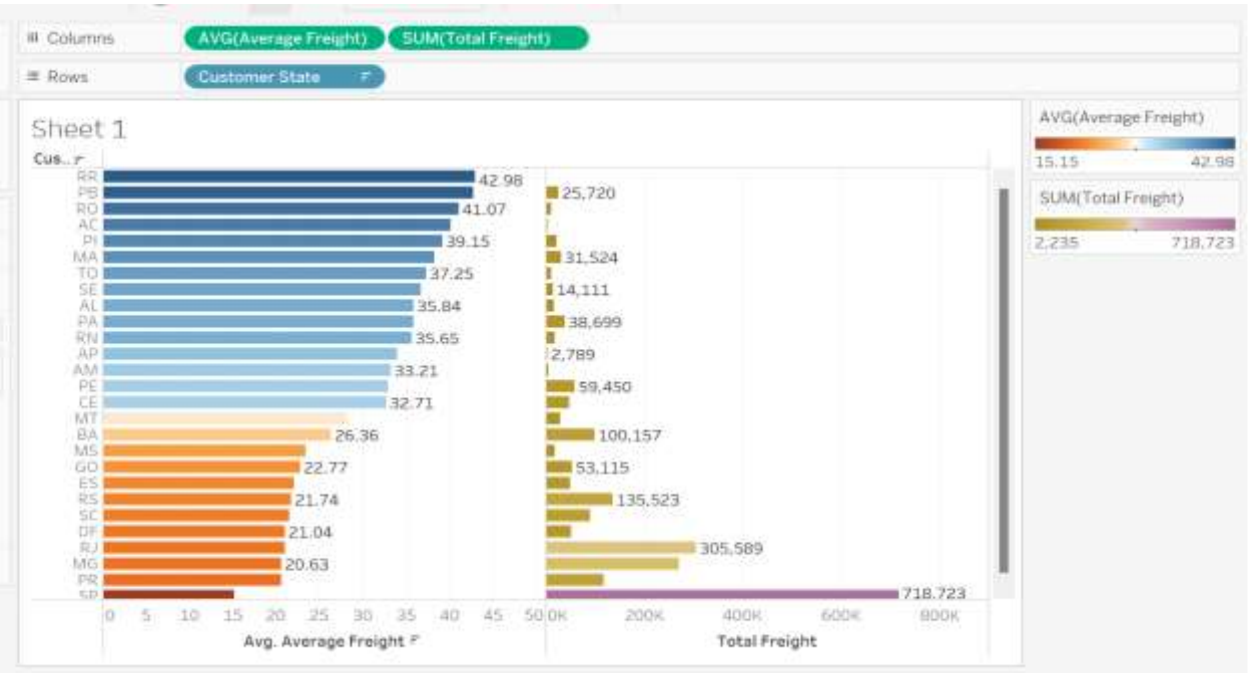


Row	customer_state	total_freight	average_freight
11	TO	11732.67999999...	37.24660317460...
12	AM	5478.890000000...	33.20539393939...
13	MA	31523.77000000...	38.25700242718...
14	PE	59449.65999999...	32.91786267995...
15	ES	49764.59999999...	22.05877659574...
16	AL	15914.58999999...	35.84367117117...
17	MT	29715.43000000...	28.16628436018...
18	RN	18860.09999999...	35.65236294896...
19	CE	48351.58999999...	32.71420162381...
20	PI	21218.2	39.14797047970...



21	MS	19144.03000000...	23.37488400488...
22	PB	25719.73000000...	42.72380398671...
23	RO	11417.38000000...	41.06971223021...
24	SE	14111.46999999...	36.65316883116...
25	AC	3686.750000000...	40.07336956521...
26	RR	2235.190000000...	42.98442307692...
27	AP	2788.500000000...	34.00609756097...

Report Analysis:



Analysis based on sales, freight and delivery time:

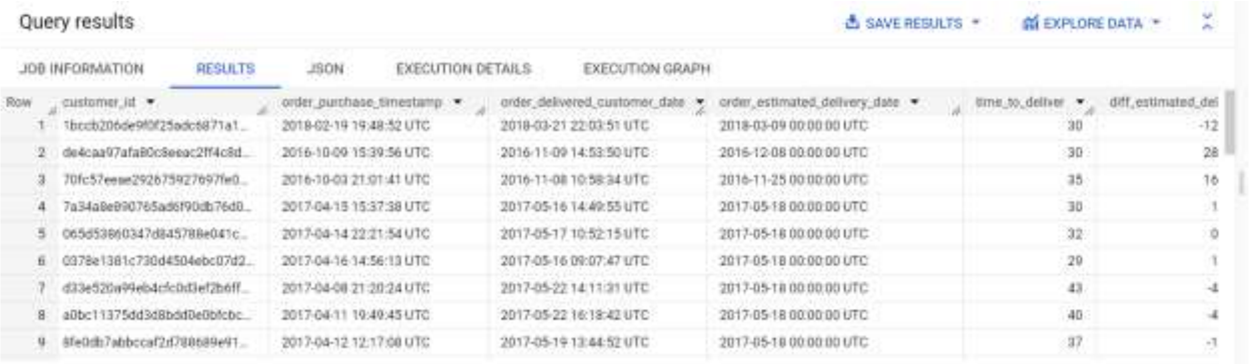
Find the no. of days taken to deliver each order from the order’s purchase date as delivery time.
Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
Do this in a single query.

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

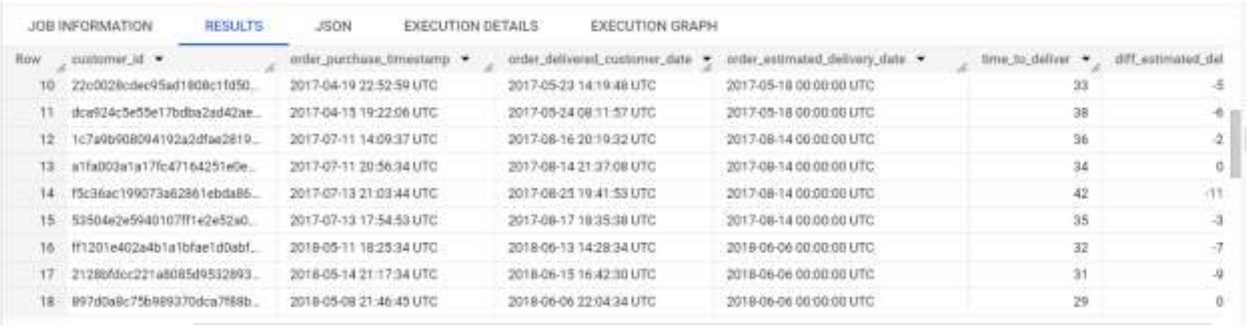
- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

```sql
SELECT
order_id,
customer_id,
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_estimated_delivery_date, order_delivered_customer_date, DAY) AS
diff_estimated_delivery
FROM
`TargetBC.orders`;
```

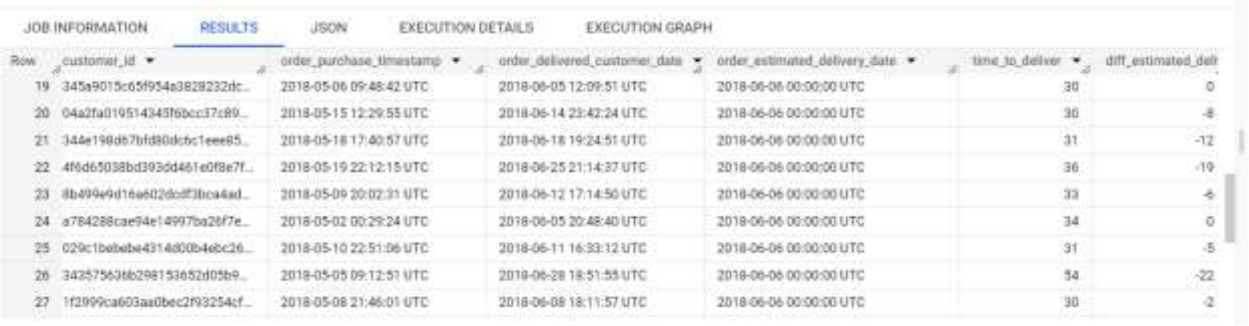
Result:



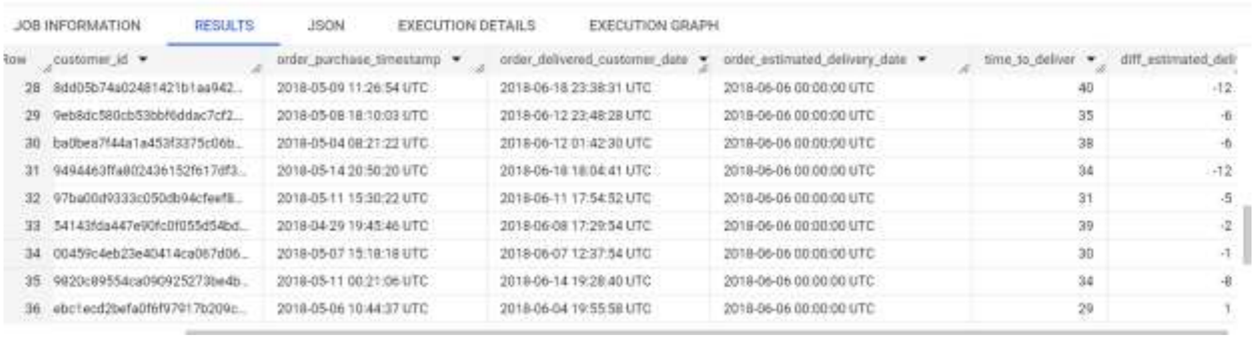
Row	customer_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_del
1	1bccb206de9f0f25adc6871a1...	2018-02-19 19:48:52 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	-12
2	de4caa97afa80c8eeac2ff4c8d...	2016-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	28
3	70fc57eeae292675927697fe0...	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	16
4	7a34a8e890765ad6f90db76d0...	2017-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	1
5	065d53860347d845788e041c...	2017-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	0378e1381c730d4504ebc07d2...	2017-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	1
7	d33e520a99eb4cfc0d3ef2b6ff...	2017-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	-4
8	a0bc11375dd3d8bdd0e0bfcbc...	2017-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	-4
9	8fe0db7abbccaf2d788689e91...	2017-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	-1



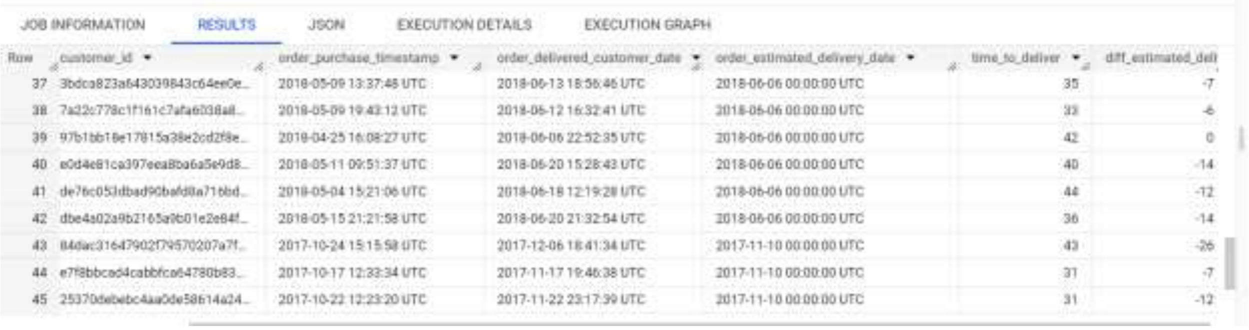
Row	customer_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_del
10	22c0028cdec95ad1808c1fd50...	2017-04-19 22:52:59 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	-5
11	dca924c5e55e17bdba2ad42ae...	2017-04-15 19:22:06 UTC	2017-05-24 08:11:57 UTC	2017-05-18 00:00:00 UTC	38	-6
12	1c7a9b908094192a2dfae2819...	2017-07-11 14:09:37 UTC	2017-08-16 20:19:32 UTC	2017-08-14 00:00:00 UTC	36	-2
13	a1fa003a1a17fc47164251e0e...	2017-07-11 20:56:34 UTC	2017-08-14 21:37:08 UTC	2017-08-14 00:00:00 UTC	34	0
14	f5c36ac199073a62861ebda86...	2017-07-13 21:03:44 UTC	2017-08-25 19:41:53 UTC	2017-08-14 00:00:00 UTC	42	-11
15	53504e2e5940107ff1e2e52a0...	2017-07-13 17:54:53 UTC	2017-08-17 18:35:38 UTC	2017-08-14 00:00:00 UTC	35	-3
16	ff1201e402a4b1a1bfae1d0abf...	2018-05-11 18:25:34 UTC	2018-06-13 14:28:34 UTC	2018-06-06 00:00:00 UTC	32	-7
17	2128bfdcc221a8085d9532893...	2018-05-14 21:17:34 UTC	2018-06-15 16:42:30 UTC	2018-06-06 00:00:00 UTC	31	-9
18	897d0a8c75b989370dca7f88b...	2018-05-08 21:46:45 UTC	2018-06-06 22:04:34 UTC	2018-06-06 00:00:00 UTC	29	0



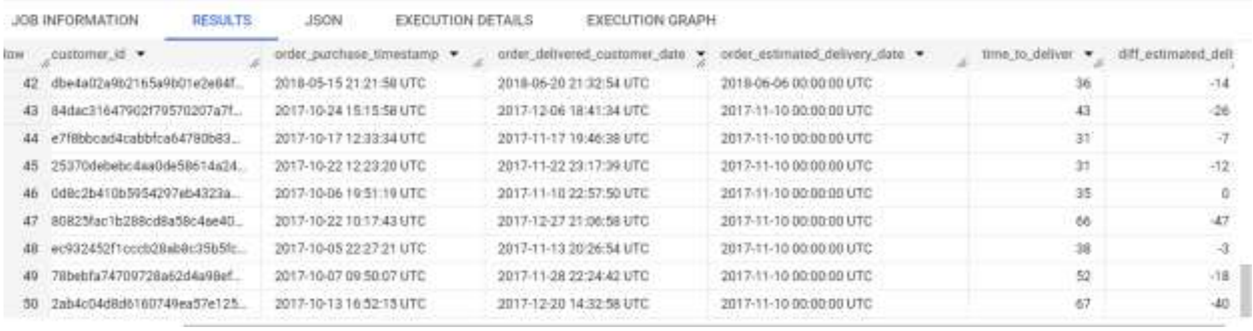
Row	customer_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_deli
19	345a9015c65f954a3828232dc...	2018-05-06 09:48:42 UTC	2018-06-05 12:09:51 UTC	2018-06-06 00:00:00 UTC	30	0
20	04a2fa019514345f6bcc37c89...	2018-05-15 12:29:55 UTC	2018-06-14 23:42:24 UTC	2018-06-06 00:00:00 UTC	30	-8
21	344e198d67bfd80dc6c1eee85...	2018-05-18 17:40:57 UTC	2018-06-18 19:24:51 UTC	2018-06-06 00:00:00 UTC	31	-12
22	4f6d65038bd393dd461e0f8e7f...	2018-05-19 22:12:15 UTC	2018-06-25 21:14:37 UTC	2018-06-06 00:00:00 UTC	36	-19
23	8b499e9d16a602dcdf3bca4ad...	2018-05-09 20:02:31 UTC	2018-06-12 17:14:50 UTC	2018-06-06 00:00:00 UTC	33	-6
24	a784288cae94e14997ba26f7e...	2018-05-02 00:29:24 UTC	2018-06-05 20:48:40 UTC	2018-06-06 00:00:00 UTC	34	0
25	029c1bebebe4314d00b4ebc26...	2018-05-10 22:51:06 UTC	2018-06-11 16:33:12 UTC	2018-06-06 00:00:00 UTC	31	-5
26	343575636b298153652d05b9...	2018-05-05 09:12:51 UTC	2018-06-28 18:51:55 UTC	2018-06-06 00:00:00 UTC	54	-22
27	1f2999ca603aa0bec2f93254cf...	2018-05-08 21:46:01 UTC	2018-06-08 18:11:57 UTC	2018-06-06 00:00:00 UTC	30	-2



Row	customer_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_deli
28	8dd05b74a02481421b1aa942...	2018-05-09 11:26:54 UTC	2018-06-18 23:38:31 UTC	2018-06-06 00:00:00 UTC	40	-12
29	9eb8dc580cb53bbf6ddac7cf2...	2018-05-08 18:10:03 UTC	2018-06-12 23:48:28 UTC	2018-06-06 00:00:00 UTC	35	-6
30	ba0bea7f44a1a453f3375c06b...	2018-05-04 08:21:22 UTC	2018-06-12 01:42:30 UTC	2018-06-06 00:00:00 UTC	38	-6
31	9494463ffa802436152f617df3...	2018-05-14 20:50:20 UTC	2018-06-18 18:04:41 UTC	2018-06-06 00:00:00 UTC	34	-12
32	97ba00d9333c050db94cfeef8...	2018-05-11 15:30:22 UTC	2018-06-11 17:54:52 UTC	2018-06-06 00:00:00 UTC	31	-5
33	54143fda447e90fc0f055d54bd...	2018-04-29 19:45:46 UTC	2018-06-08 17:29:54 UTC	2018-06-06 00:00:00 UTC	39	-2
34	00459c4eb23e40414ca067d06...	2018-05-07 15:18:18 UTC	2018-06-07 12:37:54 UTC	2018-06-06 00:00:00 UTC	30	-1
35	9820c89554ca090925273be4b...	2018-05-11 00:21:06 UTC	2018-06-14 19:28:40 UTC	2018-06-06 00:00:00 UTC	34	-8
36	ebc1ecd2befa0f6f97917b209c...	2018-05-06 10:44:37 UTC	2018-06-04 19:55:58 UTC	2018-06-06 00:00:00 UTC	29	1



Row	customer_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_deli
37	3bdca823a643039843c64ee0e...	2018-05-09 13:37:46 UTC	2018-06-13 18:56:46 UTC	2018-06-06 00:00:00 UTC	35	-7
38	7a22c778c1f161c7afa6038a8...	2018-05-09 19:43:12 UTC	2018-06-12 16:32:41 UTC	2018-06-06 00:00:00 UTC	33	-6
39	97b1bb18e17815a38e2cd2f8e...	2018-04-25 16:08:27 UTC	2018-06-06 22:52:35 UTC	2018-06-06 00:00:00 UTC	42	0
40	e0d4e81ca397eea8ba6a5e9d8...	2018-05-11 09:51:37 UTC	2018-06-20 15:28:43 UTC	2018-06-06 00:00:00 UTC	40	-14
41	de76c053dbad90bafd8a716bd...	2018-05-04 15:21:06 UTC	2018-06-18 12:19:28 UTC	2018-06-06 00:00:00 UTC	44	-12
42	dbe4a02a9b2165a9b01e2e84f...	2018-05-15 21:21:58 UTC	2018-06-20 21:32:54 UTC	2018-06-06 00:00:00 UTC	36	-14
43	84dac31647902f79570207a7f...	2017-10-24 15:15:58 UTC	2017-12-06 18:41:34 UTC	2017-11-10 00:00:00 UTC	43	-26
44	e7f8bbcad4cabbfca64780b83...	2017-10-17 12:33:34 UTC	2017-11-17 19:46:38 UTC	2017-11-10 00:00:00 UTC	31	-7
45	25370debebc4aa0de58614a24...	2017-10-22 12:23:20 UTC	2017-11-22 23:17:39 UTC	2017-11-10 00:00:00 UTC	31	-12

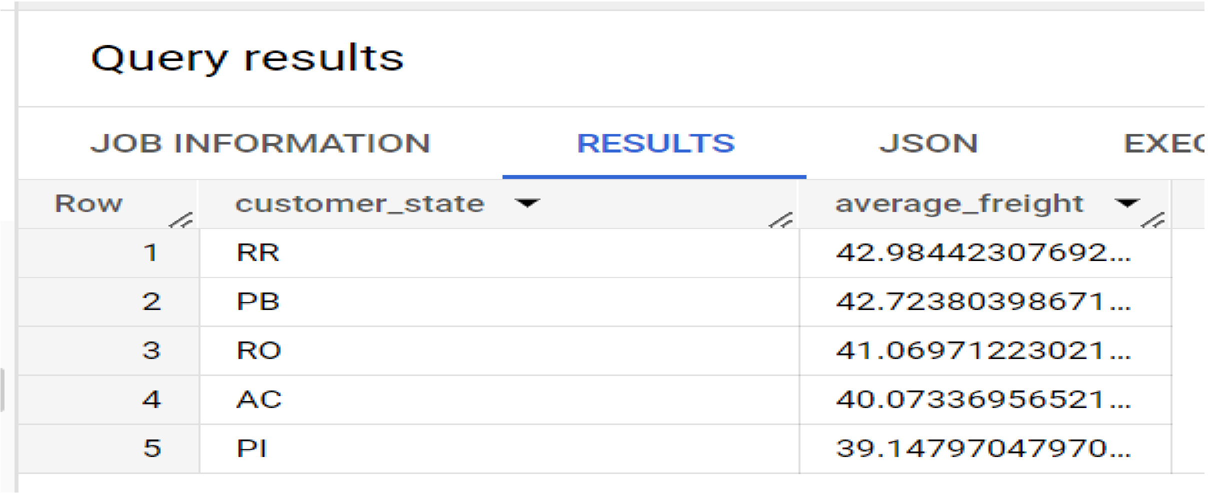


Row	customer_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_deliver	diff_estimated_deli
42	dbe4a02a9b2165a9b01e2e84f...	2018-05-15 21:21:58 UTC	2018-06-20 21:32:54 UTC	2018-06-06 00:00:00 UTC	36	-14
43	84dac31647902f79570207a7f...	2017-10-24 15:15:58 UTC	2017-12-06 18:41:34 UTC	2017-11-10 00:00:00 UTC	43	-26
44	e7f8bbcad4cabbfca64780b83...	2017-10-17 12:33:34 UTC	2017-11-17 19:46:38 UTC	2017-11-10 00:00:00 UTC	31	-7
45	25370debebc4aa0de58614a24...	2017-10-22 12:23:20 UTC	2017-11-22 23:17:39 UTC	2017-11-10 00:00:00 UTC	31	-12
46	0d8c2b410b5954297eb4323a...	2017-10-06 19:51:19 UTC	2017-11-10 22:57:50 UTC	2017-11-10 00:00:00 UTC	35	0
47	80825fac1b288cd8a58c4ae40...	2017-10-22 10:17:43 UTC	2017-12-27 21:06:58 UTC	2017-11-10 00:00:00 UTC	66	-47
48	ec932452f1cccb28ab8c35b5fc...	2017-10-05 22:27:21 UTC	2017-11-13 20:26:54 UTC	2017-11-10 00:00:00 UTC	38	-3
49	78bebfa74709728a62d4a98ef...	2017-10-07 09:50:07 UTC	2017-11-28 22:24:42 UTC	2017-11-10 00:00:00 UTC	52	-18
50	2ab4c04d8d6160749ea57e125...	2017-10-13 16:52:13 UTC	2017-12-20 14:32:58 UTC	2017-11-10 00:00:00 UTC	67	-40

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

```
SELECT
customer_state,
AVG(freight_value) AS average_freight
FROM
`TargetBC.order_items` AS oi
JOIN `TargetBC.orders` AS o ON oi.order_id = o.order_id
JOIN `TargetBC.customer` AS c ON o.customer_id = c.customer_id
GROUP BY
customer_state
ORDER BY
average_freight DESC
LIMIT 5;
```

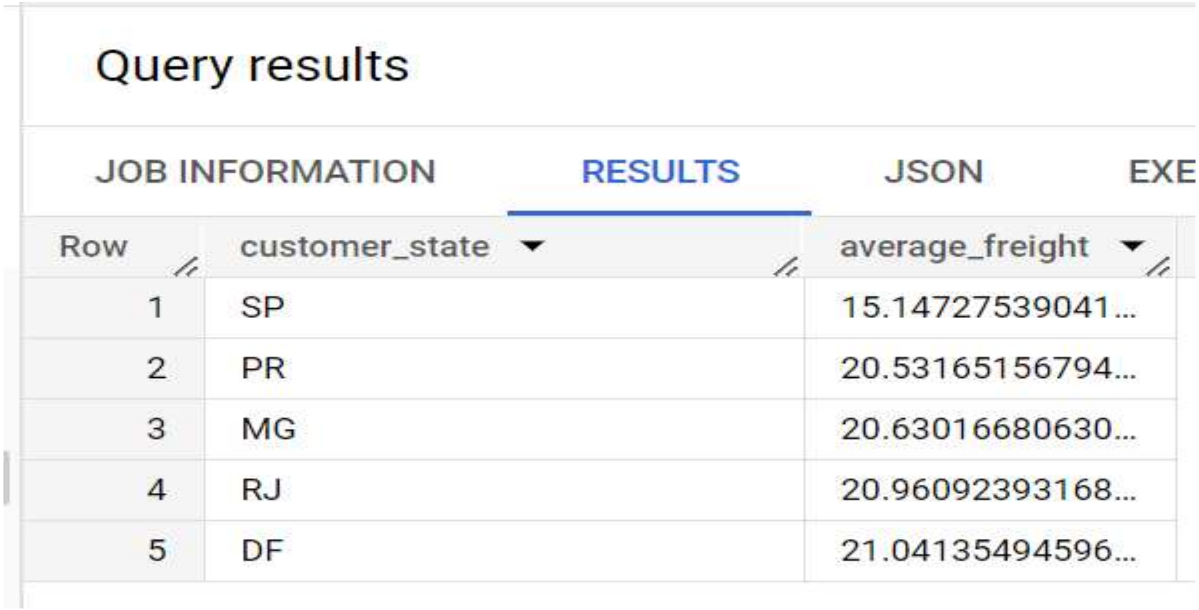
Result:



Row	customer_state	average_freight
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...

```
SELECT
customer_state,
AVG(freight_value) AS average_freight
FROM
`TargetBC.order_items` AS oi
JOIN `TargetBC.orders` AS o ON oi.order_id = o.order_id
JOIN `TargetBC.customer` AS c ON o.customer_id = c.customer_id
GROUP BY
customer_state
ORDER BY
average_freight ASC
LIMIT 5;
```

Result:

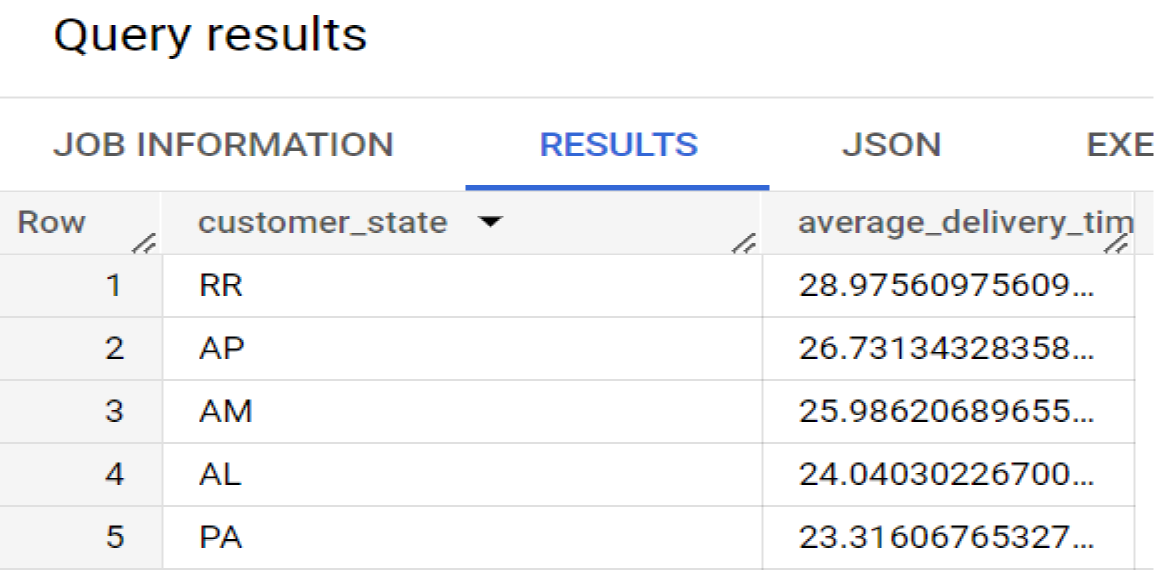


Row	customer_state	average_freight
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

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

```
SELECT
customer_state,
AVG(DATE_DIFF(TIMESTAMP(order_delivered_customer_date),
TIMESTAMP(order_purchase_timestamp), DAY)) AS average_delivery_time
FROM
`TargetBC.orders` AS o
JOIN `TargetBC.customer` AS c ON o.customer_id = c.customer_id
GROUP BY
customer_state
ORDER BY
average_delivery_time DESC
LIMIT 5;
```

Result:

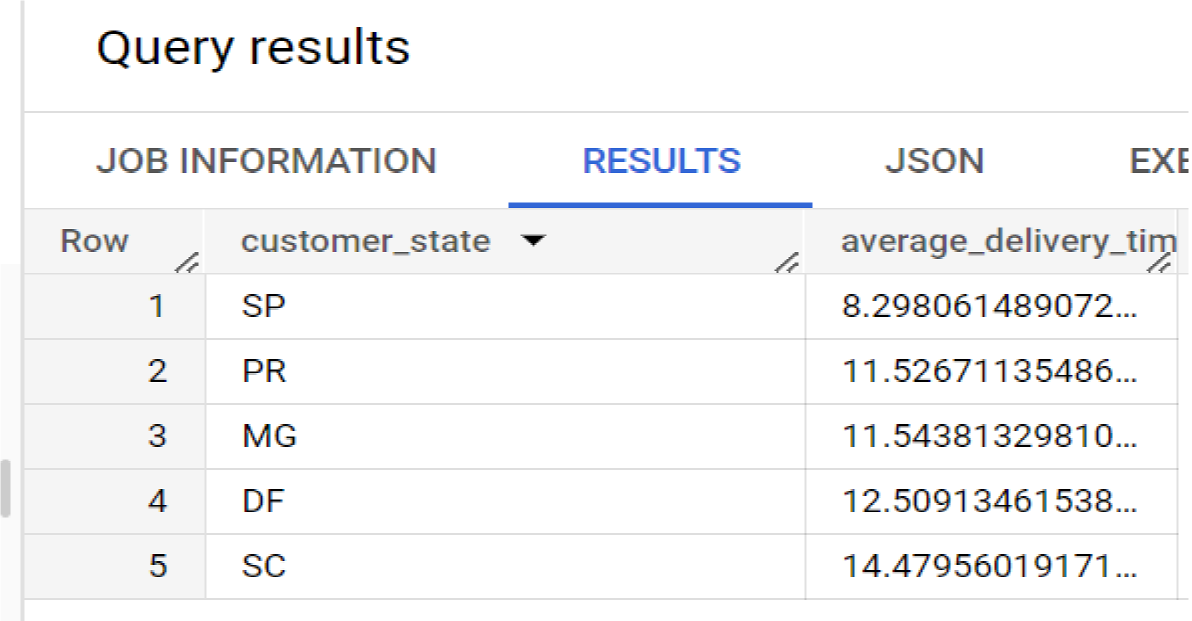


Query results

Row	customer_state	average_delivery_tim
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...

```
SELECT
customer_state,
AVG(DATE_DIFF(TIMESTAMP(order_delivered_customer_date),
TIMESTAMP(order_purchase_timestamp), DAY)) AS average_delivery_time
FROM
`TargetBC.orders` AS o
JOIN `TargetBC.customer` AS c ON o.customer_id = c.customer_id
GROUP BY
customer_state
ORDER BY
average_delivery_time asc
LIMIT 5;
```

Result:

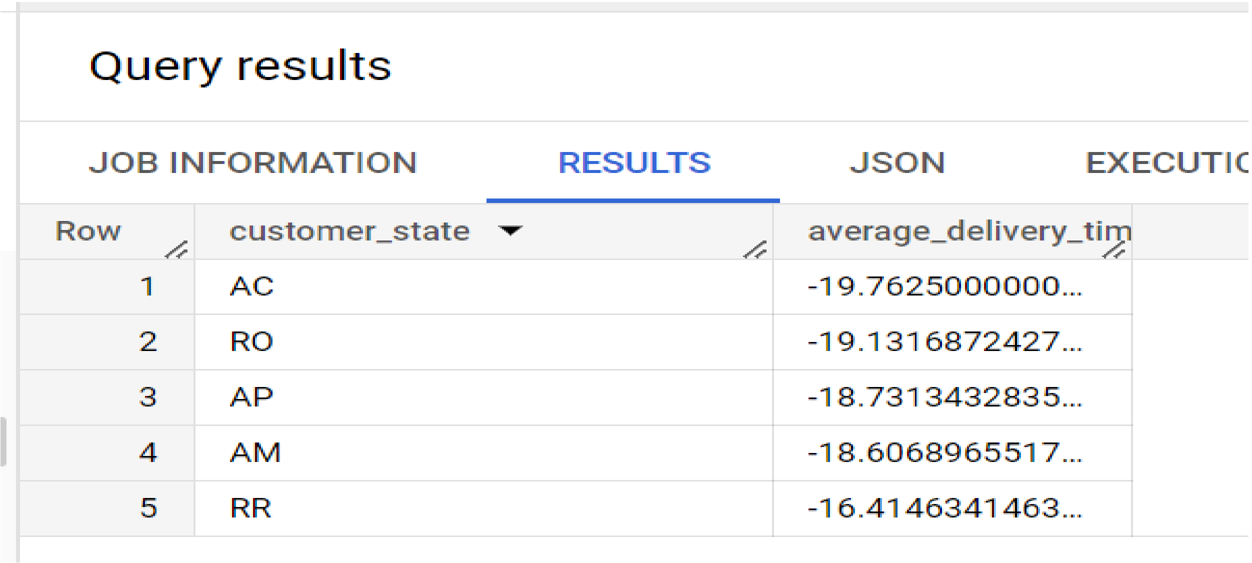


Row	customer_state	average_delivery_tim
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...

1. 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.

```
SELECT
c.customer_state,
AVG(DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, DAY)) AS
average_delivery_time_difference
FROM
`TargetBC.orders` AS o
JOIN
`TargetBC.customer` AS c ON o.customer_id = c.customer_id
GROUP BY
c.customer_state
ORDER BY
average_delivery_time_difference ASC
LIMIT
5;
```

Result:



Query results

Row	customer_state	average_delivery_tim
1	AC	-19.7625000000...
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6068965517...
5	RR	-16.4146341463...

The negative value for average_delivery_time_difference indicates that, on average, the actual delivery date is earlier than the estimated delivery date.
This suggests that the orders are being delivered faster than initially estimated.

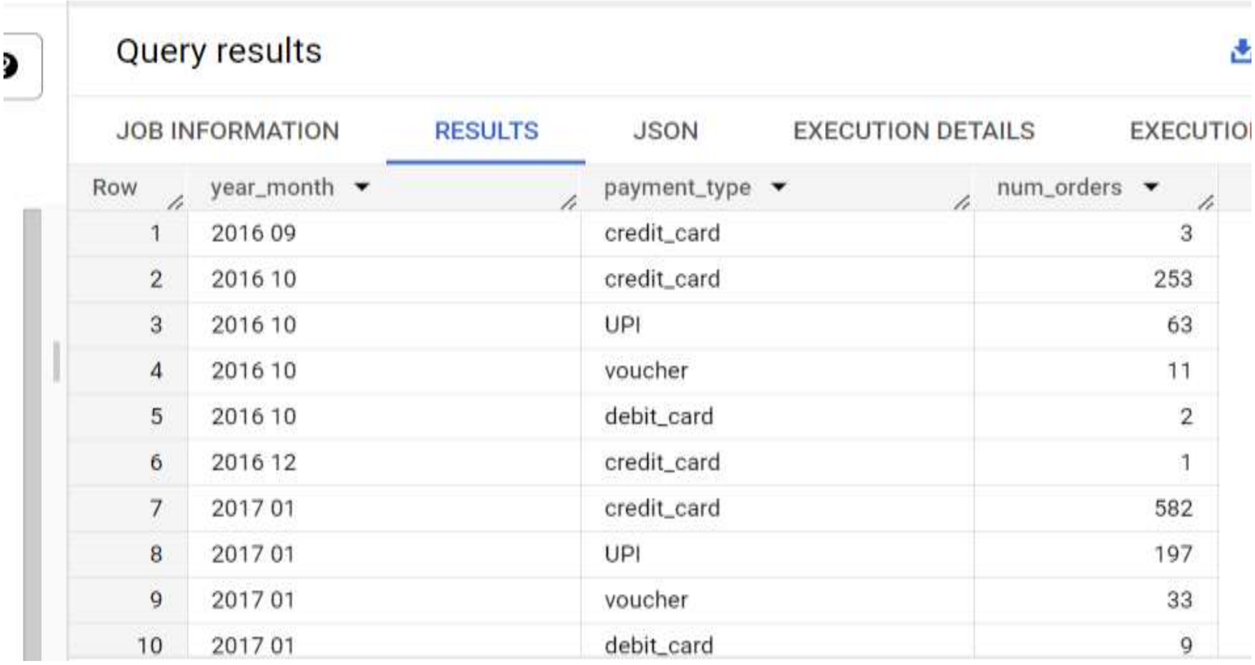
In the query, average_delivery_time_difference is calculated as the average difference between the actual delivery date (order_delivered_customer_date) and the estimated delivery date (order_estimated_delivery_date).
If the result is negative, it means that the actual delivery is occurring earlier than the estimated date.
This information can be used to identify the top 5 states where the order delivery is faster than expected.

Analysis based on the payments:

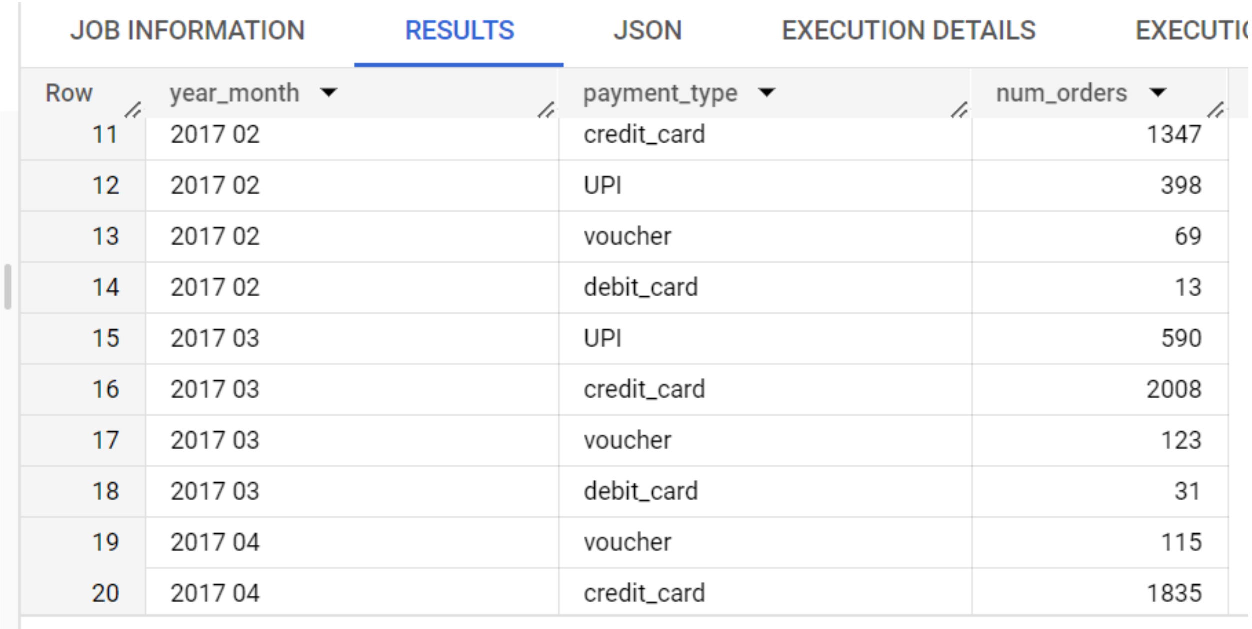
1. Find the month on month no. of orders placed using different payment types.

```
SELECT
FORMAT_TIMESTAMP('%Y-%m', TIMESTAMP_TRUNC(order_purchase_timestamp, MONTH)) AS month,
payment_type,
COUNT(DISTINCT order_id) AS num_orders
FROM
`TargetBC.orders`
JOIN
`TargetBC.payments` USING (order_id)
GROUP BY
month, payment_type
ORDER BY
month;
```

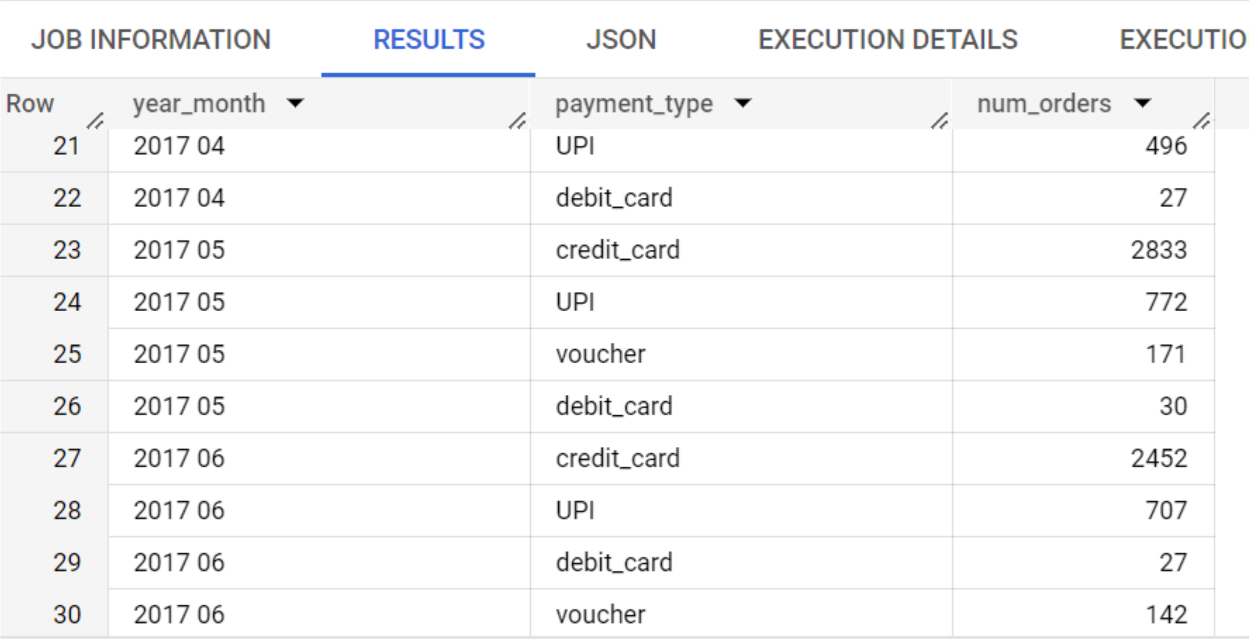
Result:



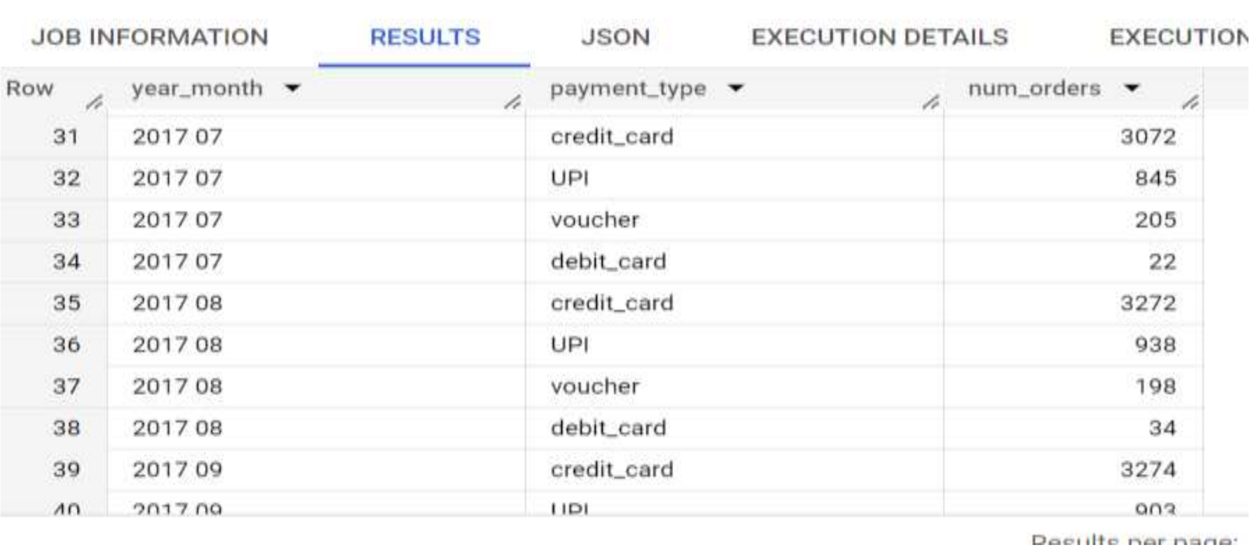
Row	year_month	payment_type	num_orders
1	2016 09	credit_card	3
2	2016 10	credit_card	253
3	2016 10	UPI	63
4	2016 10	voucher	11
5	2016 10	debit_card	2
6	2016 12	credit_card	1
7	2017 01	credit_card	582
8	2017 01	UPI	197
9	2017 01	voucher	33
10	2017 01	debit_card	9



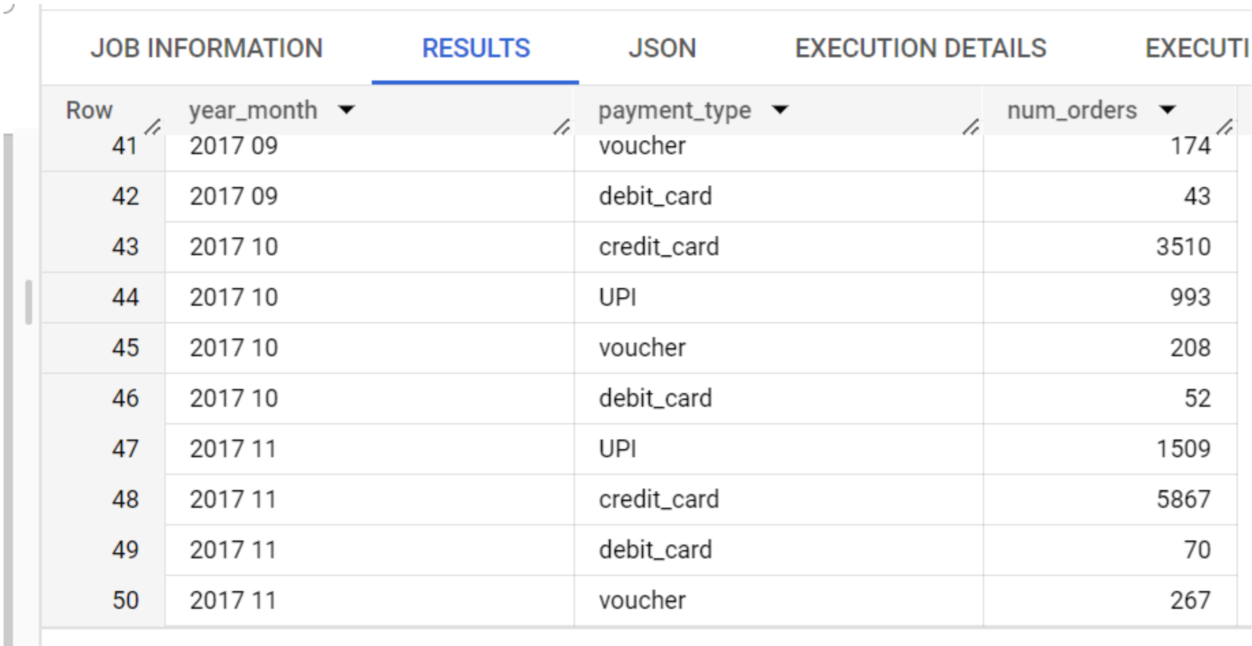
Row	year_month	payment_type	num_orders
11	2017 02	credit_card	1347
12	2017 02	UPI	398
13	2017 02	voucher	69
14	2017 02	debit_card	13
15	2017 03	UPI	590
16	2017 03	credit_card	2008
17	2017 03	voucher	123
18	2017 03	debit_card	31
19	2017 04	voucher	115
20	2017 04	credit_card	1835



Row	year_month	payment_type	num_orders
21	2017 04	UPI	496
22	2017 04	debit_card	27
23	2017 05	credit_card	2833
24	2017 05	UPI	772
25	2017 05	voucher	171
26	2017 05	debit_card	30
27	2017 06	credit_card	2452
28	2017 06	UPI	707
29	2017 06	debit_card	27
30	2017 06	voucher	142

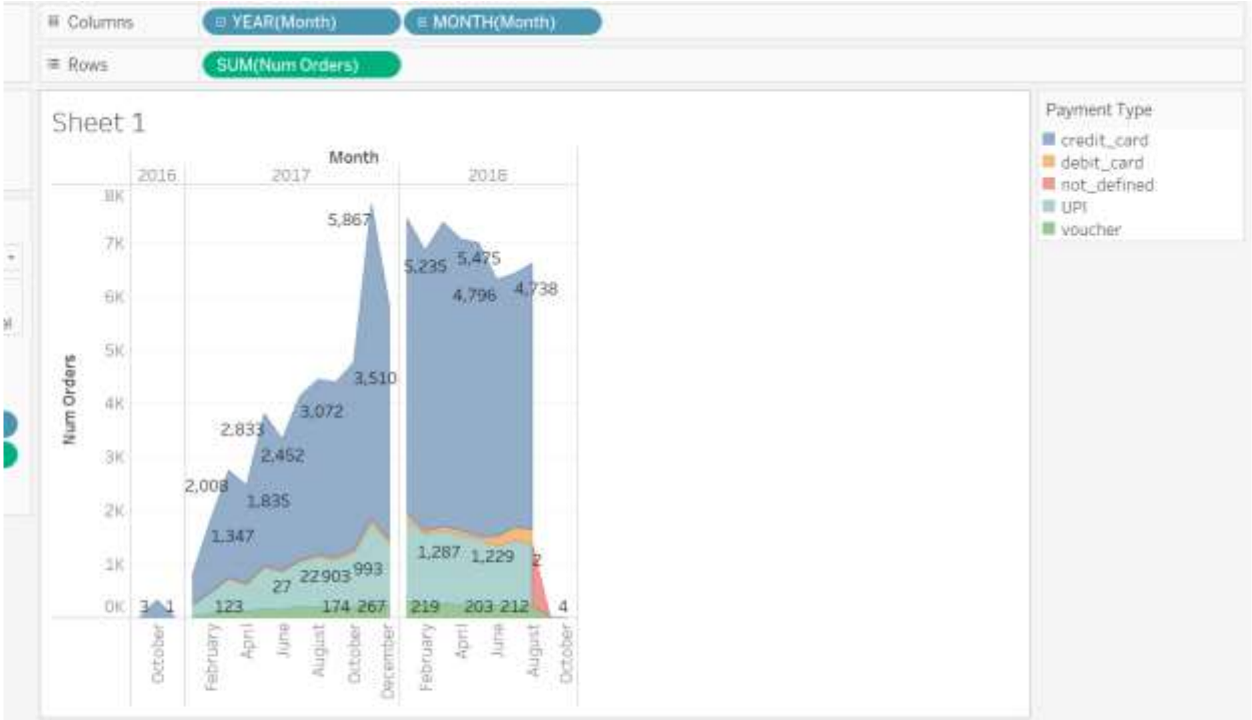


Row	year_month	payment_type	num_orders
31	2017 07	credit_card	3072
32	2017 07	UPI	845
33	2017 07	voucher	205
34	2017 07	debit_card	22
35	2017 08	credit_card	3272
36	2017 08	UPI	938
37	2017 08	voucher	198
38	2017 08	debit_card	34
39	2017 09	credit_card	3274
40	2017 09	UPI	903



Row	year_month	payment_type	num_orders
41	2017 09	voucher	174
42	2017 09	debit_card	43
43	2017 10	credit_card	3510
44	2017 10	UPI	993
45	2017 10	voucher	208
46	2017 10	debit_card	52
47	2017 11	UPI	1509
48	2017 11	credit_card	5867
49	2017 11	debit_card	70
50	2017 11	voucher	267

Report Analysis:

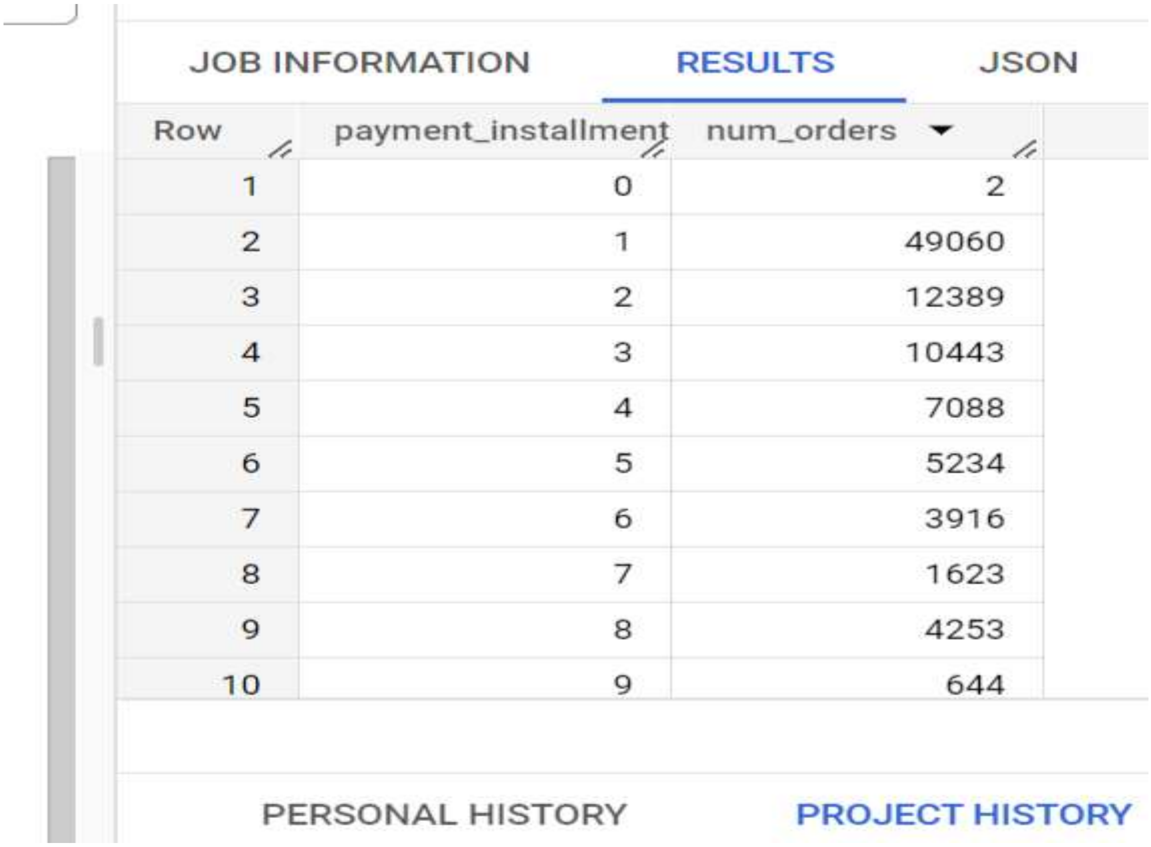


Find the no. of orders placed on the basis of the payment installments that have been paid.

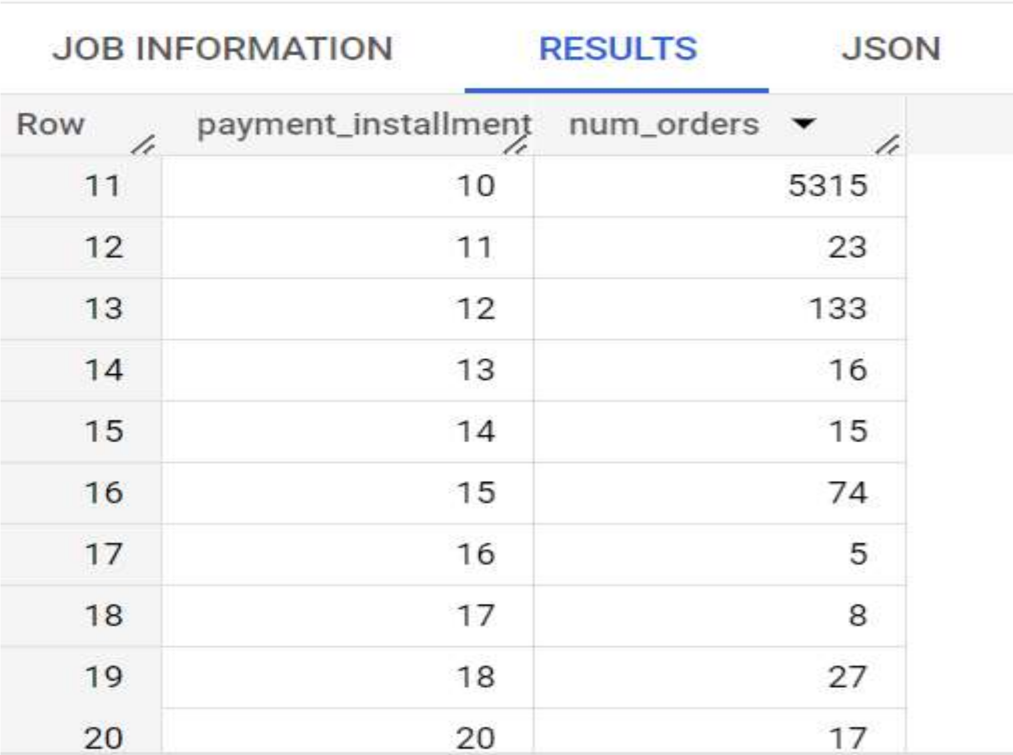
Distinct order id’s:

```
SELECT
p.payment_installments,
COUNT(DISTINCT o.order_id) AS num_orders
FROM
`TargetBC.orders` AS o
INNER JOIN
`TargetBC.payments` AS p ON o.order_id = p.order_id
GROUP BY
p.payment_installments
ORDER BY
p.payment_installments;
```

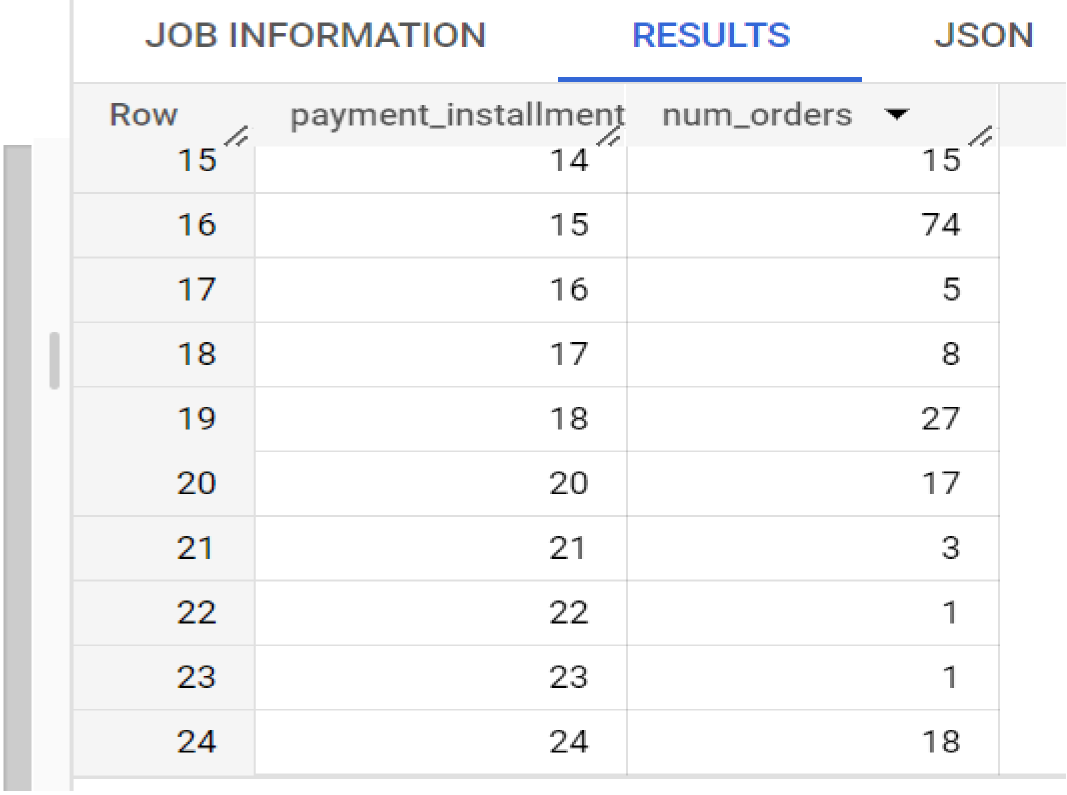
Result:



Row	payment_installment	num_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644



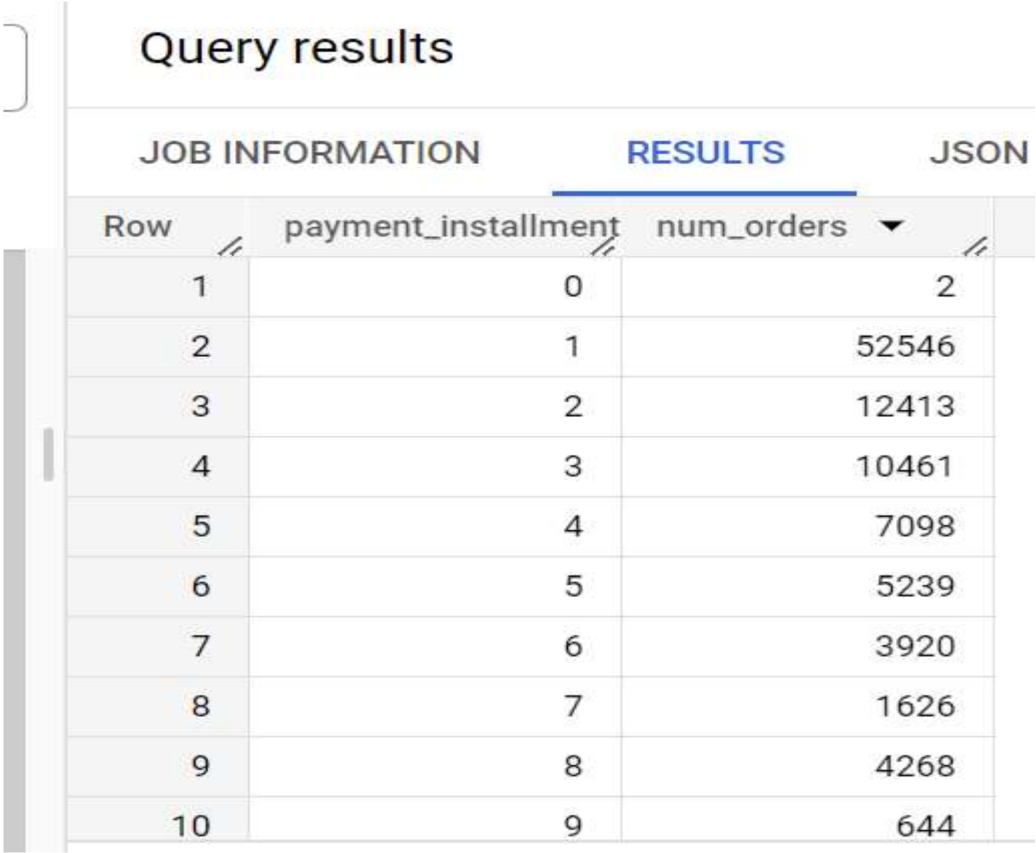
Row	payment_installment	num_orders
11	10	5315
12	11	23
13	12	133
14	13	16
15	14	15
16	15	74
17	16	5
18	17	8
19	18	27
20	20	17



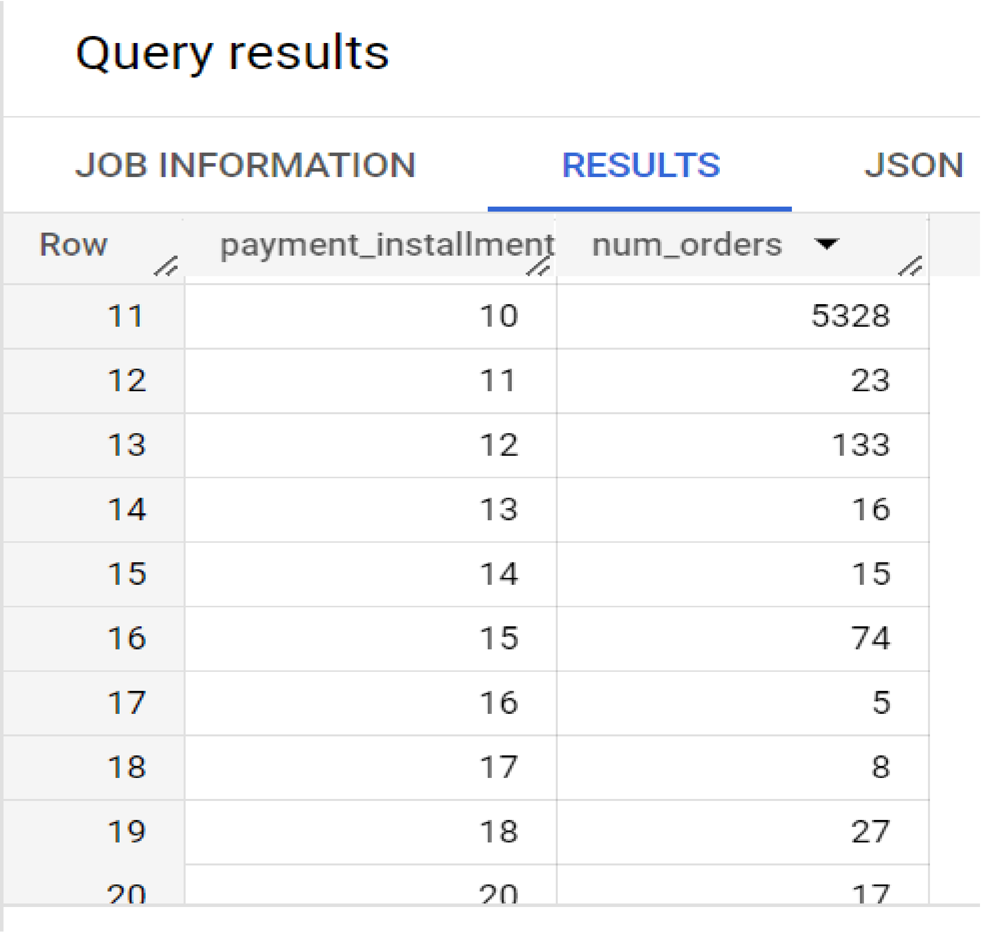
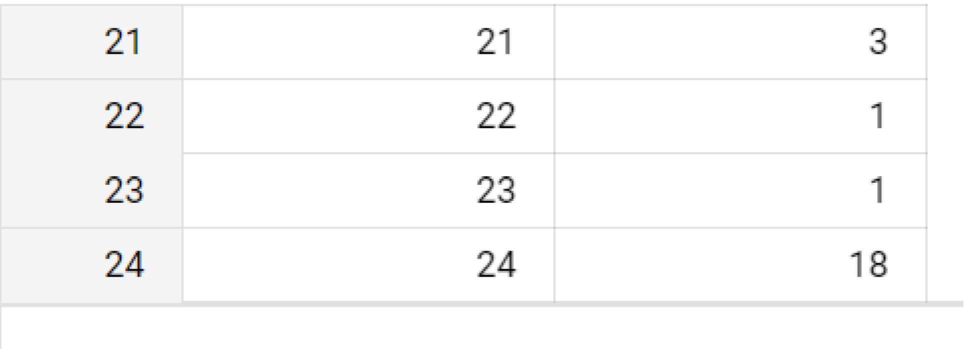
Row	payment_installment	num_orders
15	14	15
16	15	74
17	16	5
18	17	8
19	18	27
20	20	17
21	21	3
22	22	1
23	23	1
24	24	18

Without Distinct:

```
SELECT

p.payment_installments,

COUNT( o.order_id) AS num_orders

FROM

`TargetBC.orders` AS o

INNER JOIN

`TargetBC.payments` AS p ON o.order_id = p.order_id

GROUP BY

p.payment_installments

ORDER BY

p.payment_installments;
```



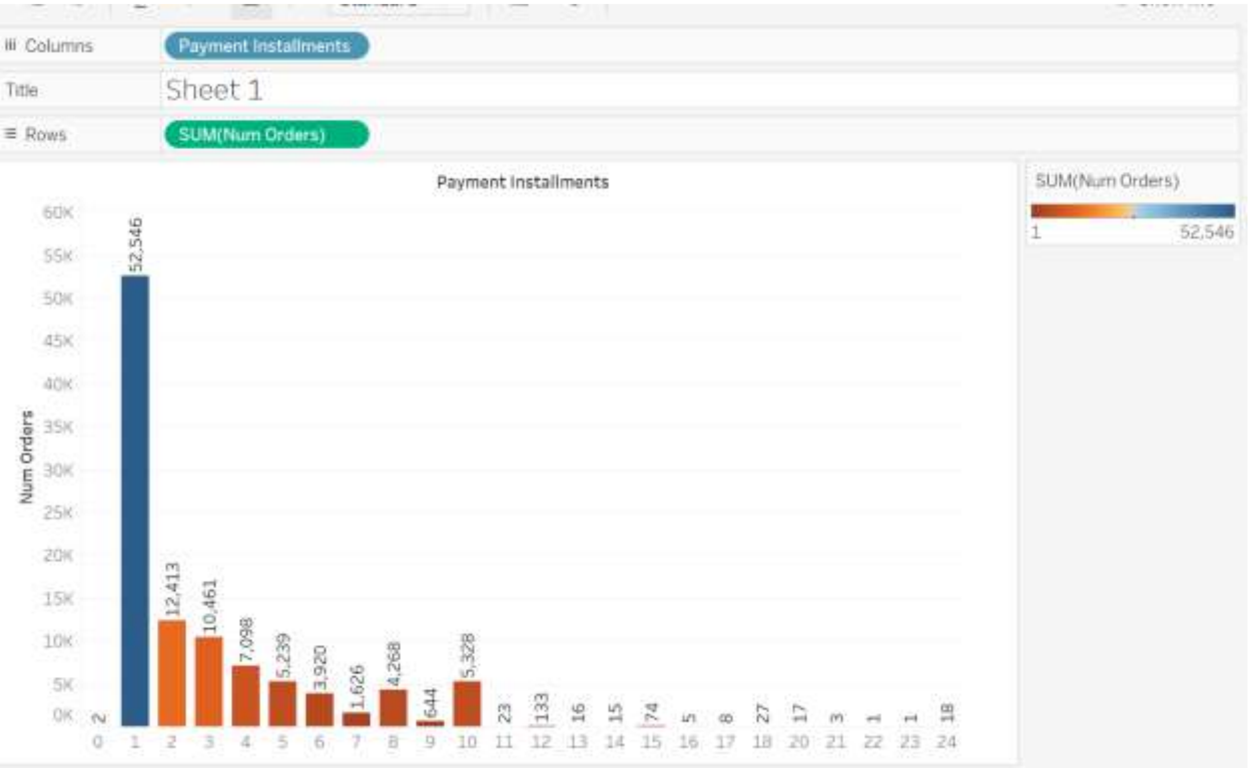
Row	payment_installment	num_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

Query results

Row	payment_installment	num_orders
11	10	5328
12	11	23
13	12	133
14	13	16
15	14	15
16	15	74
17	16	5
18	17	8
19	18	27
20	20	17
21	21	3
22	22	1
23	23	1
24	24	18

Report Analysis:



Columns: Payment Installments

Title: Sheet 1

Rows: SUM(Num Orders)