Target SQL Case Study

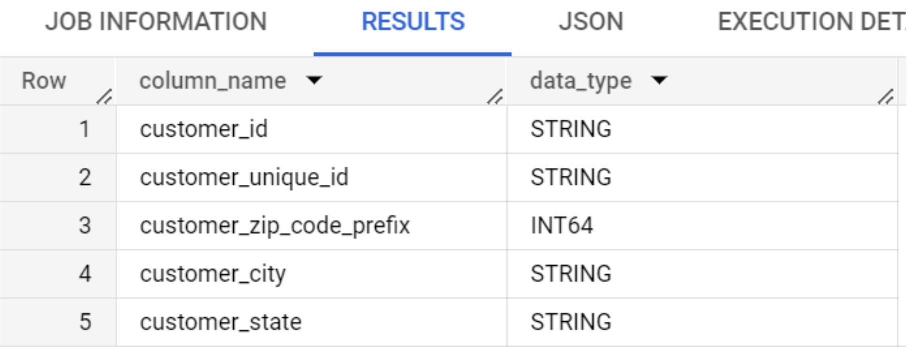
Section 1:

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

Query:

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

Output:



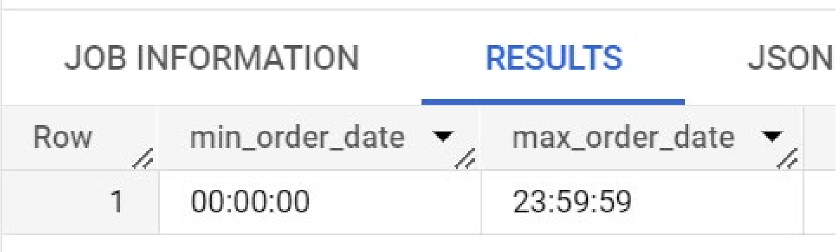
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

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

Query:

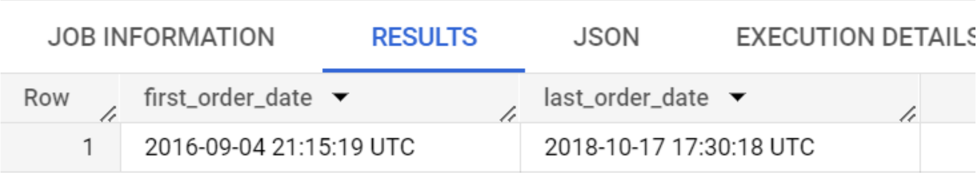
```
SELECT
  MIN(TIME(order_purchase_timestamp)) AS min_order_date,
  MAX(TIME(order_purchase_timestamp)) AS max_order_date
FROM `online_orders.orders`;
```

Output:



Row	min_order_date	max_order_date
1	00:00:00	23:59:59

Observation: Even though the orders have been placed through out the day, below are the details of first and last order placed from 2016 to 2018.



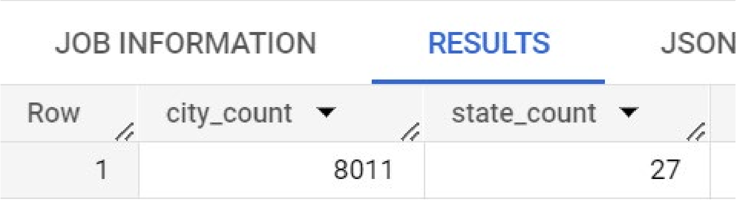
Row	first_order_date	last_order_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

Query:

```
SELECT
  COUNT(distinct geolocation_city) AS city_count,
  COUNT(distinct geolocation_state) AS state_count
FROM `online_orders.geolocation`;
```

Output:

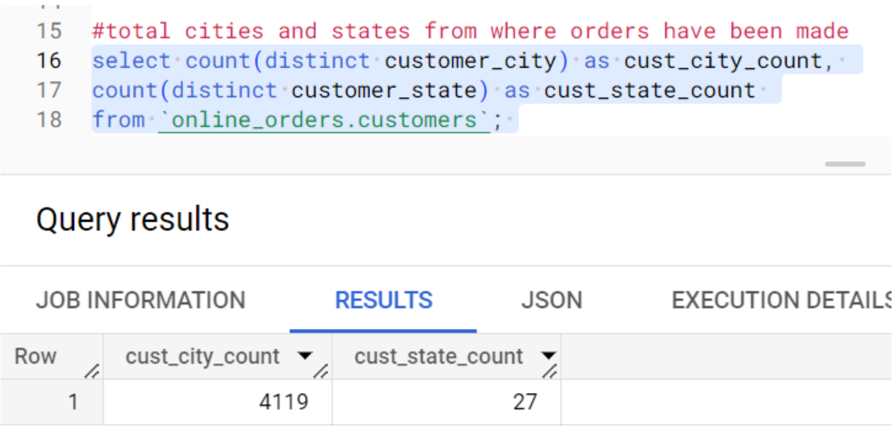


Row	city_count	state_count
1	8011	27

Observation:

The orders have not been placed from all the cities present in the data set. Below are the stats:

Total cities and states from where orders have been made:



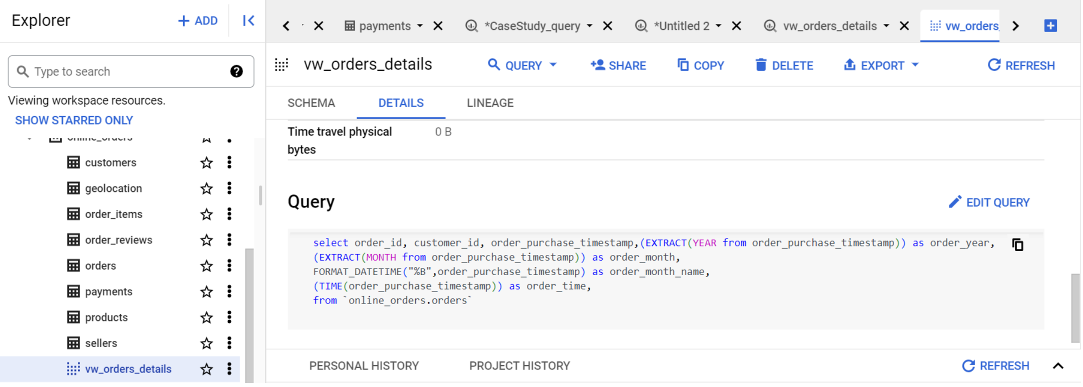
```
#total cities and states from where orders have been made
select count(distinct customer_city) as cust_city_count,
count(distinct customer_state) as cust_state_count
from `online_orders.customers`;
```

Query results

Row	cust_city_count	cust_state_count
1	4119	27

Section 2:

Below query has been created as a view and then used in this section and further:



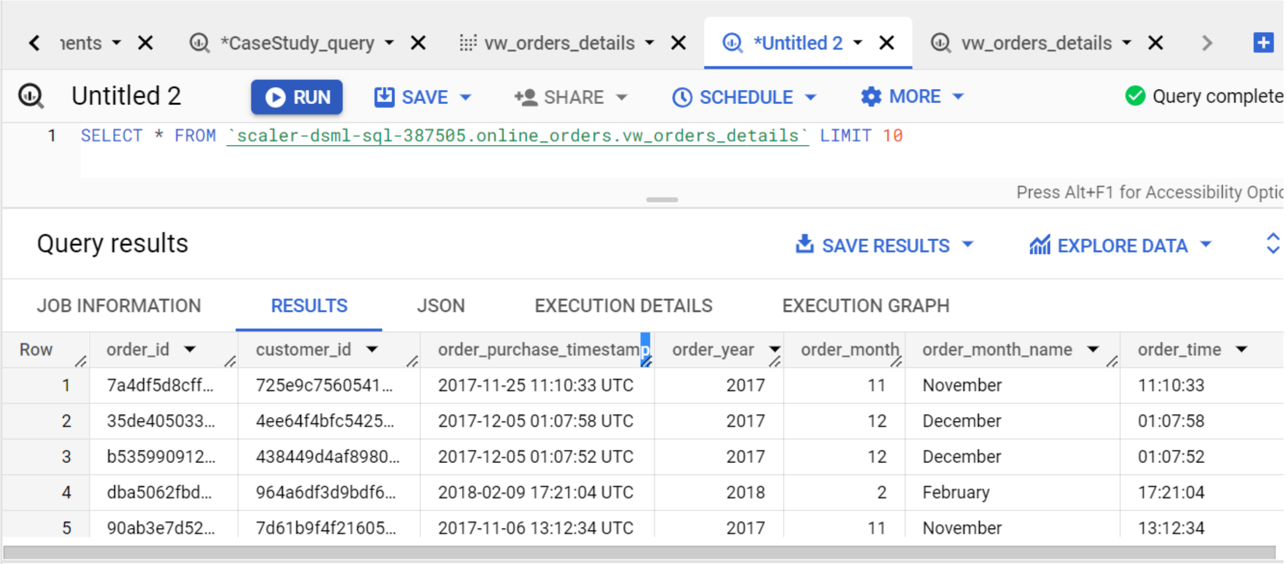
Time travel physical bytes 0 B

Query

```
select order_id, customer_id, order_purchase_timestamp,(EXTRACT(YEAR from order_purchase_timestamp)) as order_year,
(EXTRACT(MONTH from order_purchase_timestamp)) as order_month,
FORMAT_DATETIME("%B",order_purchase_timestamp) as order_month_name,
(TIME(order_purchase_timestamp)) as order_time,
from `online_orders.orders`
```

```
select order_id, customer_id, order_purchase_timestamp,
(EXTRACT(YEAR from order_purchase_timestamp)) as order_year,
(EXTRACT(MONTH from order_purchase_timestamp)) as order_month,
FORMAT_DATETIME("%B",order_purchase_timestamp) as order_month_name,
(TIME(order_purchase_timestamp)) as order_time,
from `online_orders.orders`
```

Output of running the view:



```
SELECT * FROM `scaler-dsml-sql-387505.online_orders.vw_orders_details` LIMIT 10
```

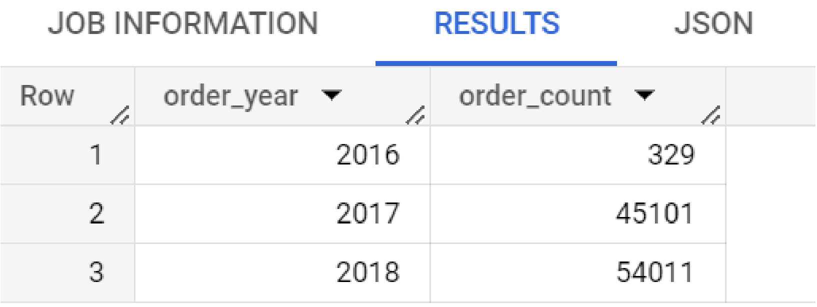
Row	order_id	customer_id	order_purchase_timestamp	order_year	order_month	order_month_name	order_time
1	7a4df5d8cff...	725e9c7560541...	2017-11-25 11:10:33 UTC	2017	11	November	11:10:33
2	35de405033...	4ee64f4bfc5425...	2017-12-05 01:07:58 UTC	2017	12	December	01:07:58
3	b535990912...	438449d4af8980...	2017-12-05 01:07:52 UTC	2017	12	December	01:07:52
4	dba5062fbd...	964a6df3d9bdf6...	2018-02-09 17:21:04 UTC	2018	2	February	17:21:04
5	90ab3e7d52...	7d61b9f4f21605...	2017-11-06 13:12:34 UTC	2017	11	November	13:12:34

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

Query:

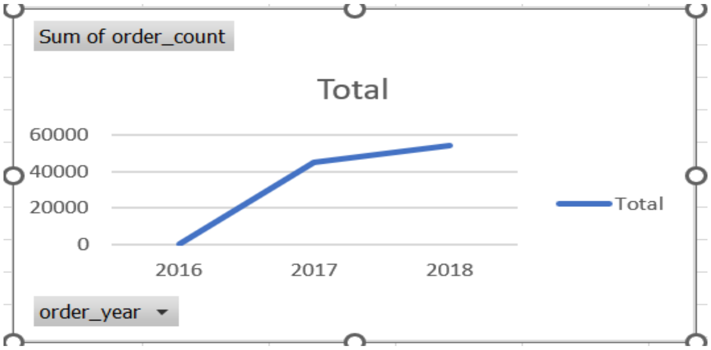
```
SELECT
  DISTINCT(order_year),
  COUNT(order_id) OVER(PARTITION BY order_year ORDER BY order_year)
AS order_count
FROM `online_orders.vw_orders_details`;
```

Output:



Row	order_year	order_count
1	2016	329
2	2017	45101
3	2018	54011

Observation: There is a growing trend each year in the number of orders being placed by the customers. There is a huge increase from 2016 to 2017 as compared to 2017 to 2018.

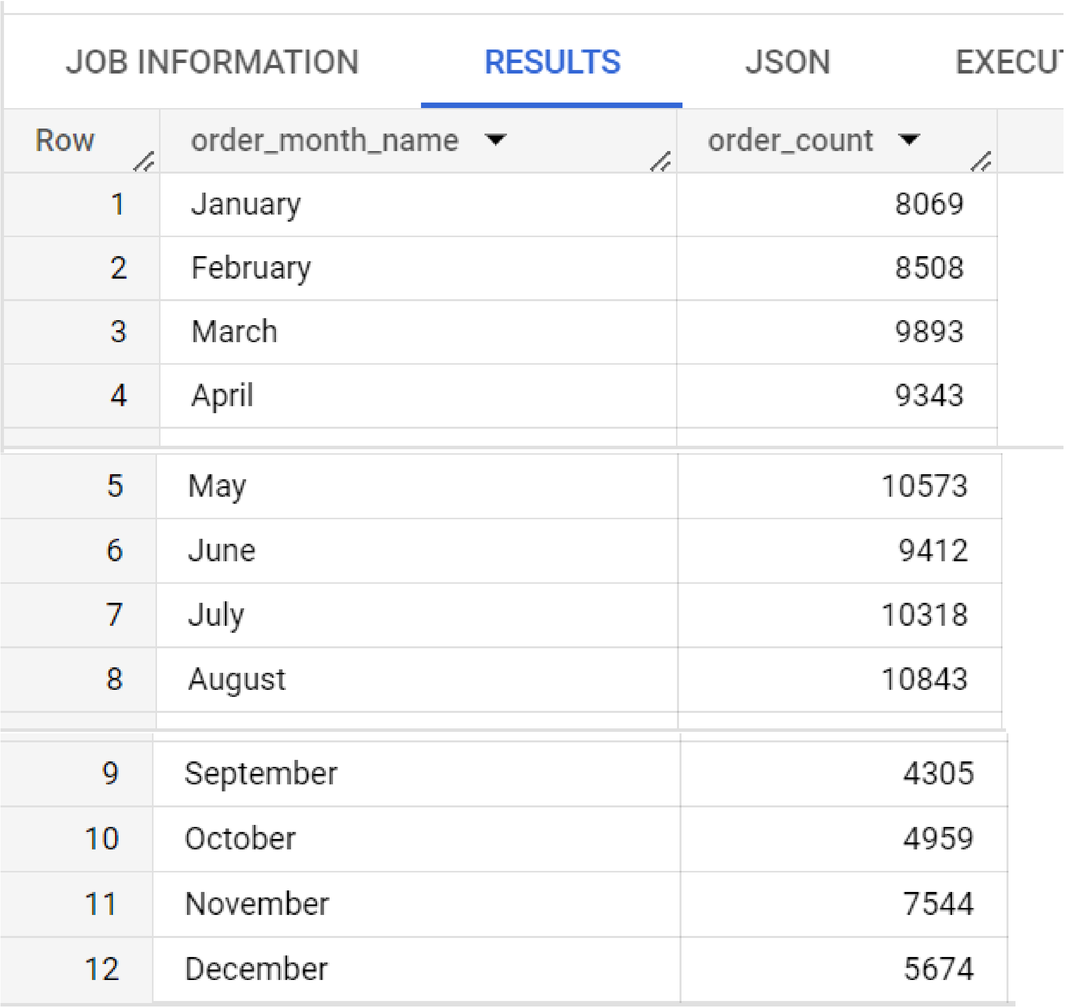


2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Query:

```
SELECT
  DISTINCT order_month_name,
  COUNT(order_id) OVER(PARTITION BY order_month ORDER BY order_month)
AS order_count
FROM `online_orders.vw_orders_details`;
```

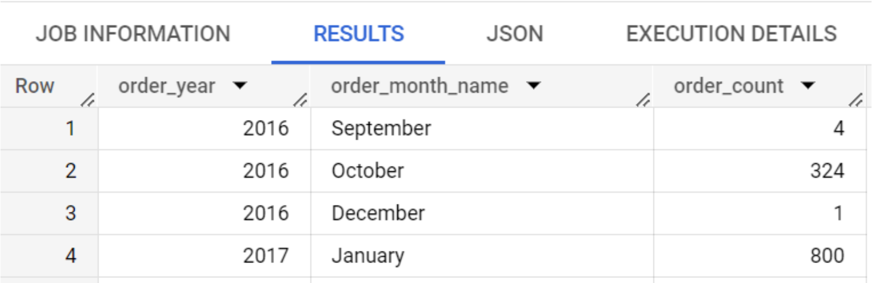
Output:



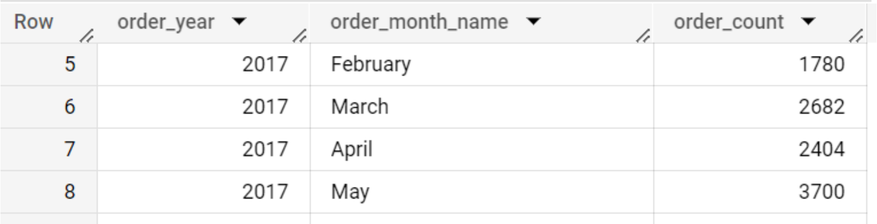
Row	order_month_name	order_count
1	January	8069
2	February	8508
3	March	9893
4	April	9343
5	May	10573
6	June	9412
7	July	10318
8	August	10843
9	September	4305
10	October	4959
11	November	7544
12	December	5674

Observation: Number of orders are more in the month of May, July and August. There is no definite trend in the increment of orders, however, we do not see any major declination in any month.
Also, since this is only on the month basis, if we include year as well, then the trend changes a bit. Since, orders are not placed in all the months in these three years, below are the sample results for the same. We see that in 2016, Orders are placed only in 3 months. However, the online trend grew from 2017.

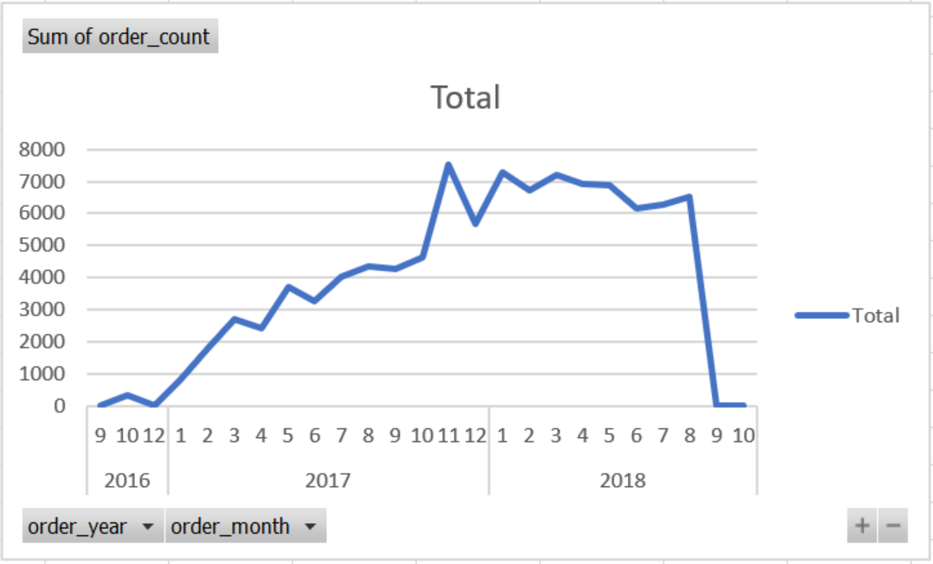
```
SELECT
  DISTINCT order_year, order_month_name,
  COUNT(order_id) OVER(PARTITION BY order_year, order_month ORDER BY order_month)
AS order_count
FROM `online_orders.vw_orders_details`
ORDER BY order_year;
```



Row	order_year	order_month_name	order_count
1	2016	September	4
2	2016	October	324
3	2016	December	1
4	2017	January	800



Row	order_year	order_month_name	order_count
5	2017	February	1780
6	2017	March	2682
7	2017	April	2404
8	2017	May	3700

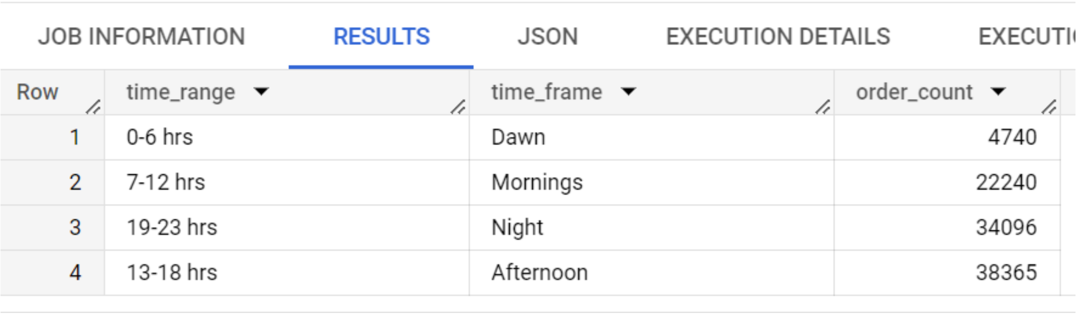


3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    1. 0-6 hrs : Dawn
    2. 7-12 hrs : Mornings
    3. 13-18 hrs : Afternoon
    4. 19-23 hrs : Night

Query:

```
SELECT
'0-6 hrs' as time_range, 'Dawn' as time_frame,
SUM(case when order_time between '00:00:00' and '06:00:00' THEN 1 ELSE 0 END) as
order_count
from `online_orders.vw_orders_details`
union all
SELECT
'7-12 hrs' as time_range, 'Mornings' as time_frame,
SUM(case when order_time between '06:00:01' and '12:00:00' THEN 1 ELSE 0 END) as
order_count
from `online_orders.vw_orders_details`
union all
SELECT
'13-18 hrs' as time_range, 'Afternoon' as time_frame,
SUM(case when order_time between '12:00:01' and '18:00:00' THEN 1 ELSE 0 END) as
order_count
from `online_orders.vw_orders_details`
union all
SELECT
'19-23 hrs' as time_range, 'Night' as time_frame,
SUM(case when order_time between '18:00:01' and '23:59:59' THEN 1 ELSE 0 END) as
order_count
from `online_orders.vw_orders_details`
ORDER BY order_count;
```

Output:



Row	time_range	time_frame	order_count
1	0-6 hrs	Dawn	4740
2	7-12 hrs	Mornings	22240
3	19-23 hrs	Night	34096
4	13-18 hrs	Afternoon	38365

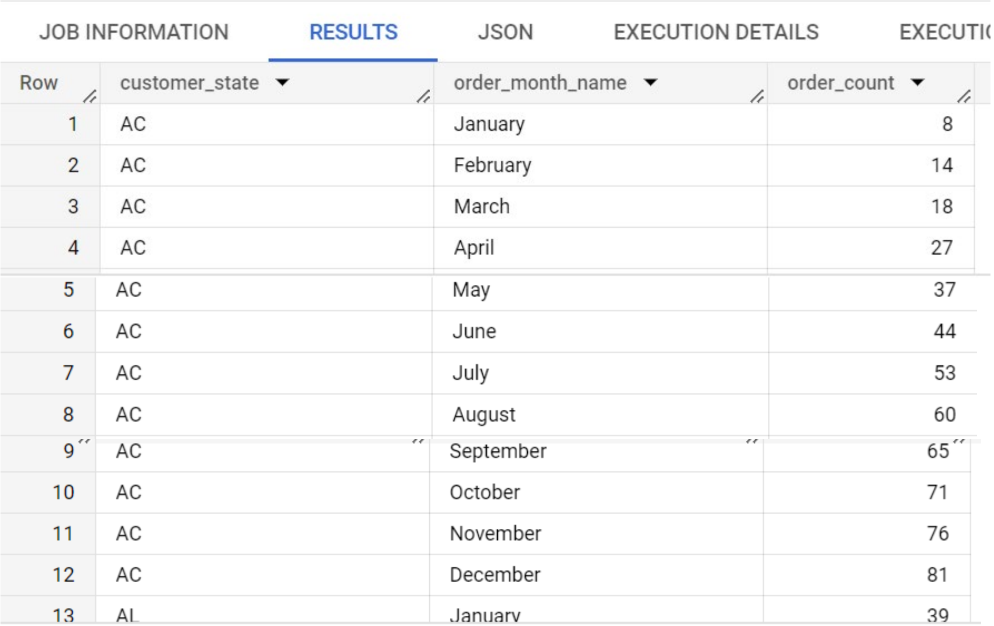
Section 3:

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

Query:

```
SELECT
  DISTINCT C.customer_state, VW.order_month_name,
  COUNT(VW.order_id) OVER(PARTITION BY C.customer_state ORDER BY VW.order_month) AS
order_count
FROM `online_orders.customers` C
INNER JOIN `online_orders.vw_orders_details` VW
  ON C.customer_id = VW.customer_id
ORDER BY C.customer_state;
```

Output:



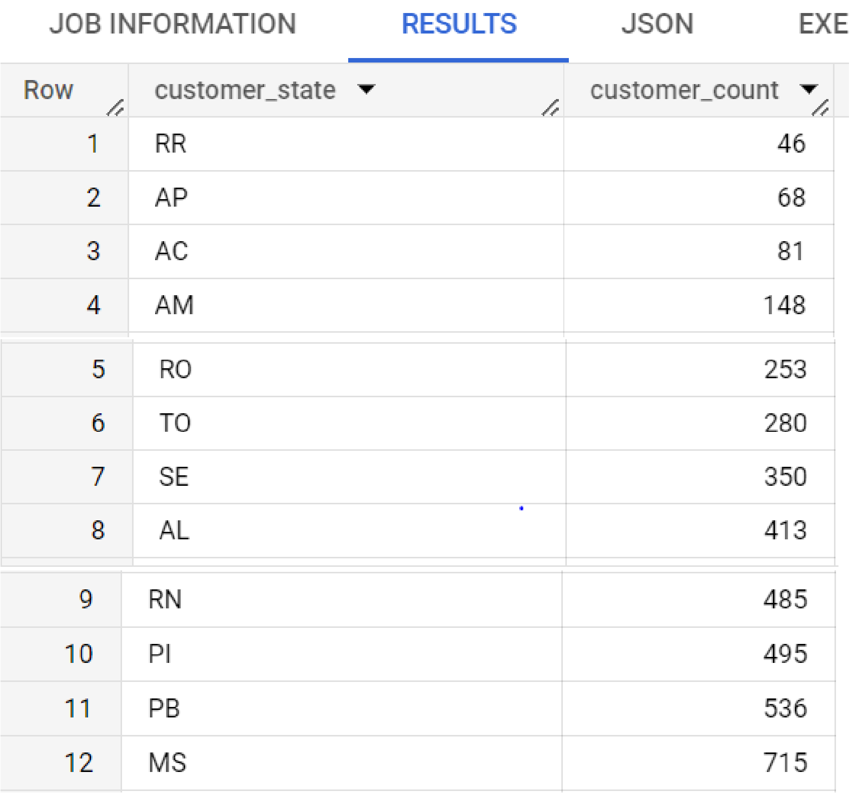
Row	customer_state	order_month_name	order_count
1	AC	January	8
2	AC	February	14
3	AC	March	18
4	AC	April	27
5	AC	May	37
6	AC	June	44
7	AC	July	53
8	AC	August	60
9	AC	September	65
10	AC	October	71
11	AC	November	76
12	AC	December	81
13	AL	January	39

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

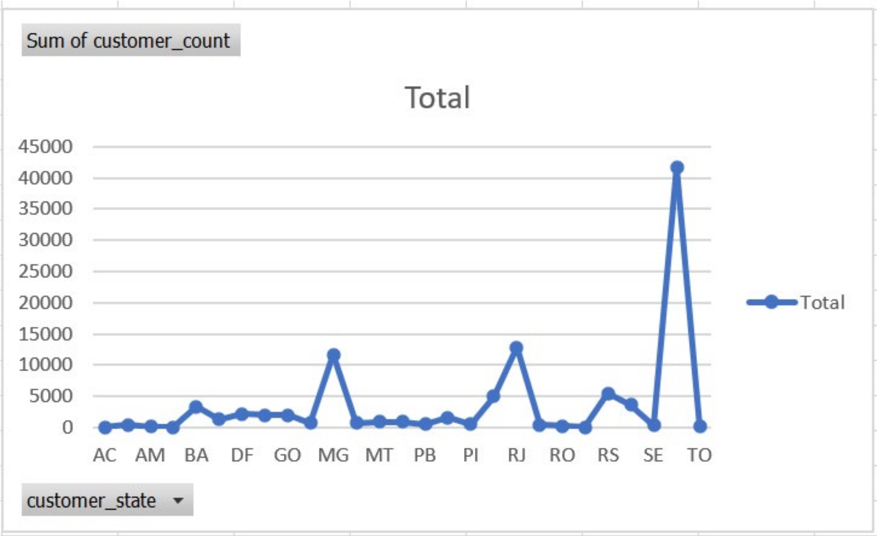
Query:

```
SELECT
  customer_state,
  COUNT(customer_id) AS customer_count
FROM `online_orders.customers`
GROUP BY customer_state
ORDER BY customer_count;
```

Output:



Row	customer_state	customer_count
1	RR	46
2	AP	68
3	AC	81
4	AM	148
5	RO	253
6	TO	280
7	SE	350
8	AL	413
9	RN	485
10	PI	495
11	PB	536
12	MS	715



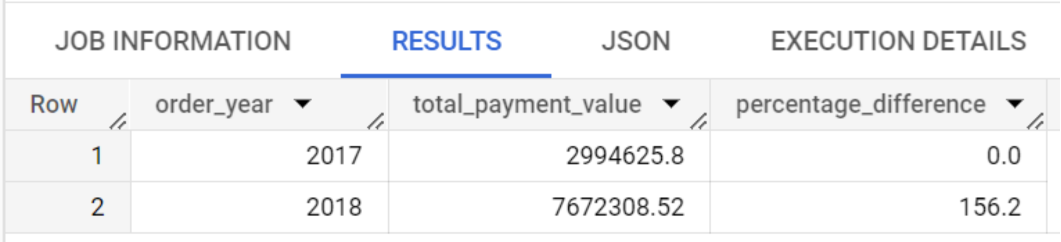
Section 4:

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.

Query:

```
WITH CTE AS
(
  SELECT VW.order_year, round(SUM(PM.payment_value),2) AS total_payment_value,
  FROM `online_orders.vw_orders_details` VW
  INNER JOIN `online_orders.payments` PM
  ON VW.order_id = PM.order_id
  WHERE VW.order_year IN(2017, 2018) AND VW.order_month BETWEEN 1 AND 7
  GROUP BY VW.order_year
  ORDER BY VW.order_year
)
SELECT
  order_year, cte.total_payment_value,
  ROUND(IFNULL(100* (total_payment_value/LAG(total_payment_value) OVER(ORDER BY
order_year) -1),0),2) AS percentage_difference
FROM CTE
ORDER BY order_year;
```

Output:



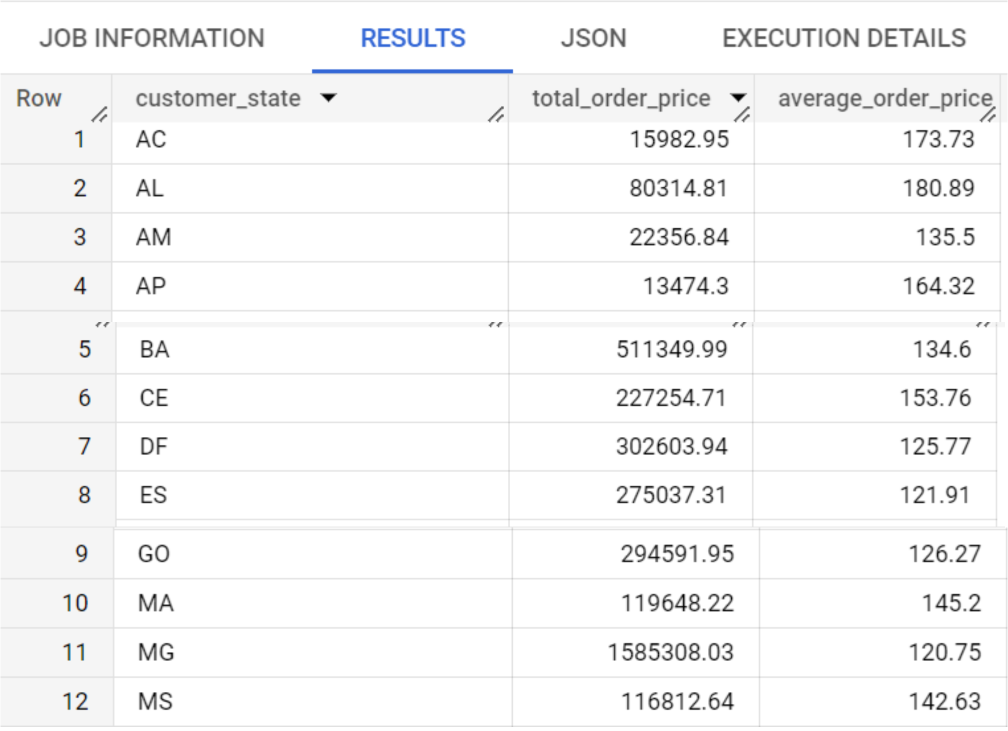
Row	order_year	total_payment_value	percentage_difference
1	2017	2994625.8	0.0
2	2018	7672308.52	156.2

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

Query:

```
SELECT
  C.customer_state,
  ROUND(SUM(OT.price),2) AS total_order_price,
  ROUND(AVG(OT.price),2) AS average_order_price
FROM `online_orders.customers` C
INNER JOIN `online_orders.orders` O
  ON C.customer_id = O.customer_id
INNER JOIN `online_orders.order_items` OT
  ON O.order_id = OT.order_id
GROUP BY C.customer_state
ORDER BY C.customer_state;
```

Output:



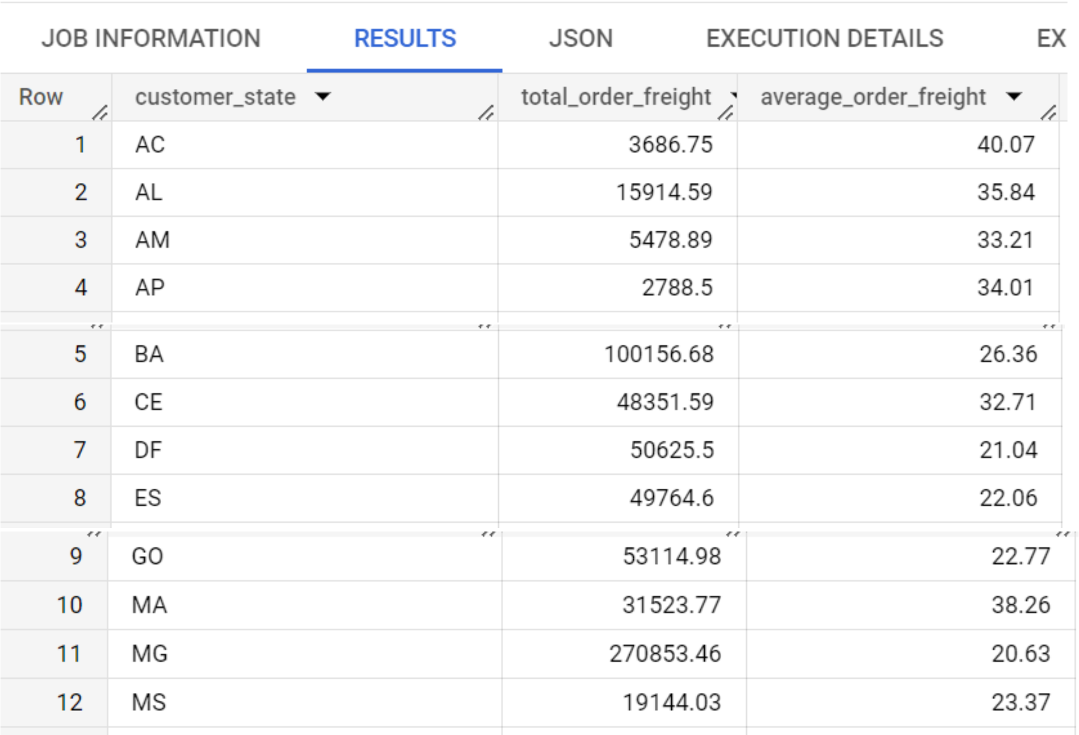
Row	customer_state	total_order_price	average_order_price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2
11	MG	1585308.03	120.75
12	MS	116812.64	142.63

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

Query:

```
SELECT
  C.customer_state,
  ROUND(SUM(OT.freight_value),2) AS total_order_freight,
  ROUND(AVG(OT.freight_value),2) AS average_order_freight
FROM `online_orders.customers` C
INNER JOIN `online_orders.orders` O
  ON C.customer_id = O.customer_id
INNER JOIN `online_orders.order_items` OT
  ON O.order_id = OT.order_id
GROUP BY C.customer_state
ORDER BY C.customer_state;
```

Output:



Row	customer_state	total_order_freight	average_order_freight
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37

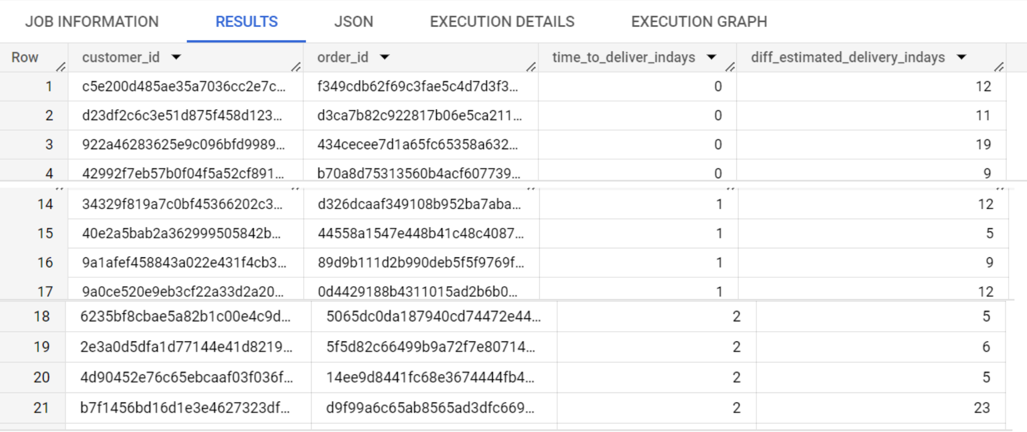
Section 5:

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.
   time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
   diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date

Query:

```
SELECT
  C.customer_id, O.order_id,
  DATETIME_DIFF(O.order_delivered_customer_date, O.order_purchase_timestamp, DAY)
AS time_to_deliver_indays,
  DATETIME_DIFF(O.order_estimated_delivery_date, O.order_delivered_customer_date,
DAY) AS diff_estimated_delivery_indays
FROM `online_orders.customers` C
INNER JOIN `online_orders.orders` O
  ON C.customer_id = O.customer_id
WHERE O.order_delivered_customer_date IS NOT NULL
ORDER BY time_to_deliver_indays;
```

Output:



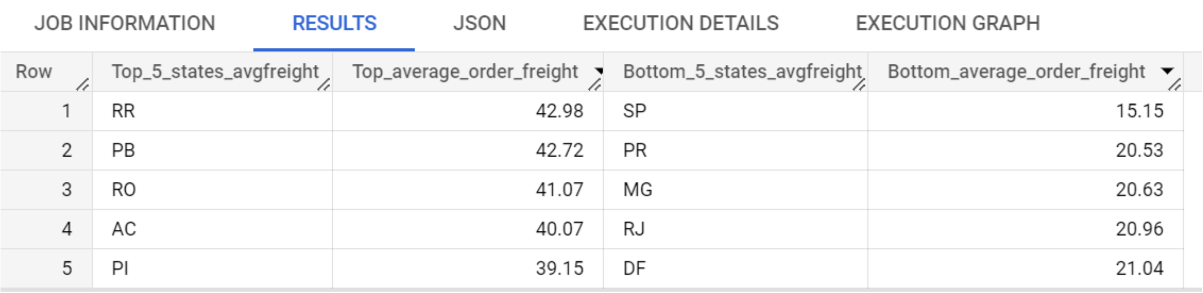
Row	customer_id	order_id	time_to_deliver_indays	diff_estimated_delivery_indays
1	c5e200d485ae35a7036cc2e7c...	f349cdb62f69c3fae5c4d7d3f3...	0	12
2	d23df2c6c3e51d875f458d123...	d3ca7b82c922817b06e5ca211...	0	11
3	922a46283625e9c096bfd9989...	434cecee7d1a65fc65358a632...	0	19
4	42992f7eb57b0f04f5a52cf891...	b70a8d75313560b4acf607739...	0	9
14	34329f819a7c0bf45366202c3...	d326dcaaf349108b952ba7aba...	1	12
15	40e2a5bab2a362999505842b...	44558a1547e448b41c48c4087...	1	5
16	9a1afef458843a022e431f4cb3...	89d9b111d2b990deb5f5f9769f...	1	9
17	9a0ce520e9eb3cf22a33d2a20...	0d4429188b4311015ad2b6b0...	1	12
18	6235bf8cbae5a82b1c00e4c9d...	5065dc0da187940cd74472e44...	2	5
19	2e3a0d5dfa1d77144e41d8219...	5f5d82c66499b9a72f7e80714...	2	6
20	4d90452e76c65ebcaaf03f036f...	14ee9d8441fc68e3674444fb4...	2	5
21	b7f1456bd16d1e3e4627323df...	d9f99a6c65ab8565ad3dfc669...	2	23

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

Query:

```
WITH CTE AS
(
  SELECT C.customer_state,
  ROUND(AVG(OT.freight_value),2) as average_order_freight,
  DENSE_RANK() OVER(ORDER BY AVG(OT.freight_value) DESC) AS top_rn,
  DENSE_RANK() OVER(ORDER BY AVG(OT.freight_value)) AS bottom_rn
  FROM `online_orders.customers` C
  INNER JOIN `online_orders.orders` O
    ON C.customer_id = O.customer_id
  INNER JOIN `online_orders.order_items` OT
    ON O.order_id = OT.order_id
  GROUP BY C.customer_state
)
SELECT T1.Top_5_states_avgfreight,T1.Top_average_order_freight,
      T2.Bottom_5_states_avgfreight, T2.Bottom_average_order_freight FROM
  (
       SELECT
       customer_state AS Top_5_states_avgfreight,
       average_order_freight AS Top_average_order_freight, top_rn
       FROM CTE
       WHERE top_rn<=5
       ORDER BY top_rn
  )T1
  INNER JOIN
  (
       SELECT
       customer_state AS Bottom_5_states_avgfreight,
       average_order_freight AS Bottom_average_order_freight, bottom_rn
       FROM CTE
       WHERE bottom_rn <=5
       ORDER BY bottom_rn
  )T2
ON T1.top_rn = T2.bottom_rn
ORDER BY T1.top_rn;
```

Output:



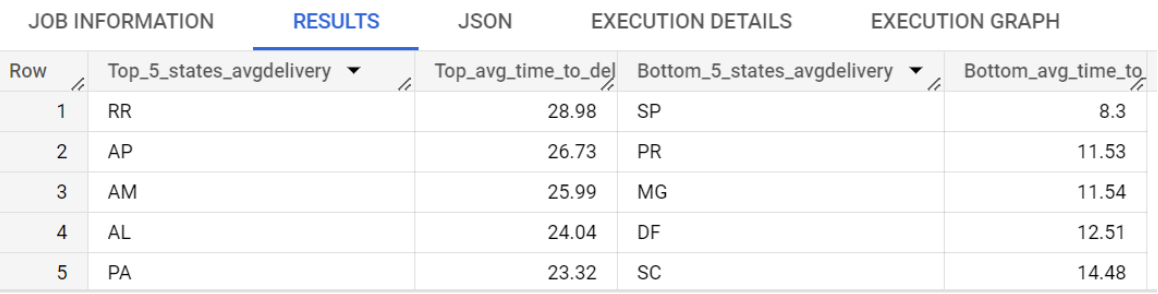
Row	Top_5_states_avgfreight	Top_average_order_freight	Bottom_5_states_avgfreight	Bottom_average_order_freight
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04

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

Query:

```
WITH CTE AS
(
  SELECT
    C.customer_state,
    AVG(DATETIME_DIFF(O.order_delivered_customer_date, O.order_purchase_timestamp,
DAY)) AS avg_time_to_deliver_indays,
    DENSE_RANK() OVER( ORDER BY AVG(DATETIME_DIFF(O.order_delivered_customer_date,
O.order_purchase_timestamp, DAY)) DESC) AS top_rn,
    DENSE_RANK() OVER( ORDER BY AVG(DATETIME_DIFF(O.order_delivered_customer_date,
O.order_purchase_timestamp, DAY)))
  AS bottom_rn
  FROM `online_orders.customers` C
  INNER JOIN `online_orders.orders` O
    ON C.customer_id = O.customer_id
  WHERE O.order_delivered_customer_date IS NOT NULL
  GROUP BY C.customer_state
  ORDER BY avg_time_to_deliver_indays
)
SELECT T1.Top_5_states_avgdelivery, T1.Top_avg_time_to_deliver_indays,
      T2.Bottom_5_states_avgdelivery, T2.Bottom_avg_time_to_deliver_indays
  FROM
  (
    SELECT customer_state AS Top_5_states_avgdelivery,
    ROUND(avg_time_to_deliver_indays,2) as Top_avg_time_to_deliver_indays, top_rn
    FROM CTE
    WHERE top_rn<=5
    ORDER BY top_rn
  )T1
  INNER JOIN
  (
    SELECT customer_state AS Bottom_5_states_avgdelivery,
    ROUND(avg_time_to_deliver_indays,2) AS Bottom_avg_time_to_deliver_indays,
bottom_rn
    FROM CTE
    WHERE bottom_rn <=5
    ORDER BY bottom_rn
  )T2
ON T1.top_rn = T2.bottom_rn
ORDER BY T1.top_rn;
```

Output:



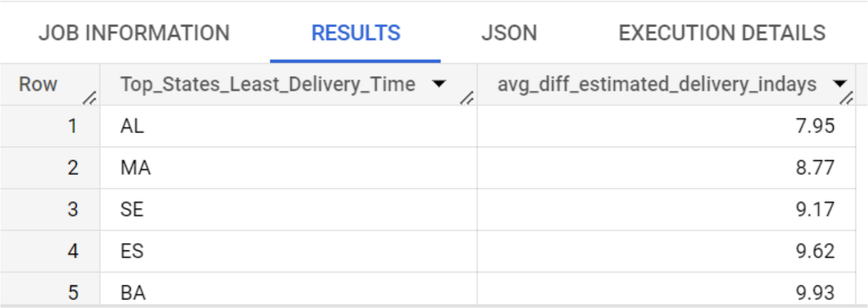
Row	Top_5_states_avgdelivery	Top_avg_time_to_del	Bottom_5_states_avgdelivery	Bottom_avg_time_to
1	RR	28.98	SP	8.3
2	AP	26.73	PR	11.53
3	AM	25.99	MG	11.54
4	AL	24.04	DF	12.51
5	PA	23.32	SC	14.48

4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
   You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Query:

```
WITH CTE AS
(
  SELECT C.customer_state,
  AVG(DATETIME_DIFF(O.order_estimated_delivery_date, O.order_delivered_customer_date,
DAY)) AS avg_diff_estimated_delivery_indays ,
  DENSE_RANK() OVER(ORDER BY AVG(DATETIME_DIFF(O.order_estimated_delivery_date,
O.order_delivered_customer_date, DAY))) AS top_rn
  FROM `online_orders.customers` C
  INNER JOIN `online_orders.orders` O
  ON C.customer_id = O.customer_id
  WHERE O.order_delivered_customer_date IS NOT NULL
  GROUP BY C.customer_state
)
SELECT
  customer_state AS Top_States_Least_Delivery_Time,
  ROUND(avg_diff_estimated_delivery_indays,2) AS avg_diff_estimated_delivery_indays
  FROM CTE
  WHERE top_rn<=5
  ORDER BY top_rn;
```

Output:



Row	Top_States_Least_Delivery_Time	avg_diff_estimated_delivery_indays
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

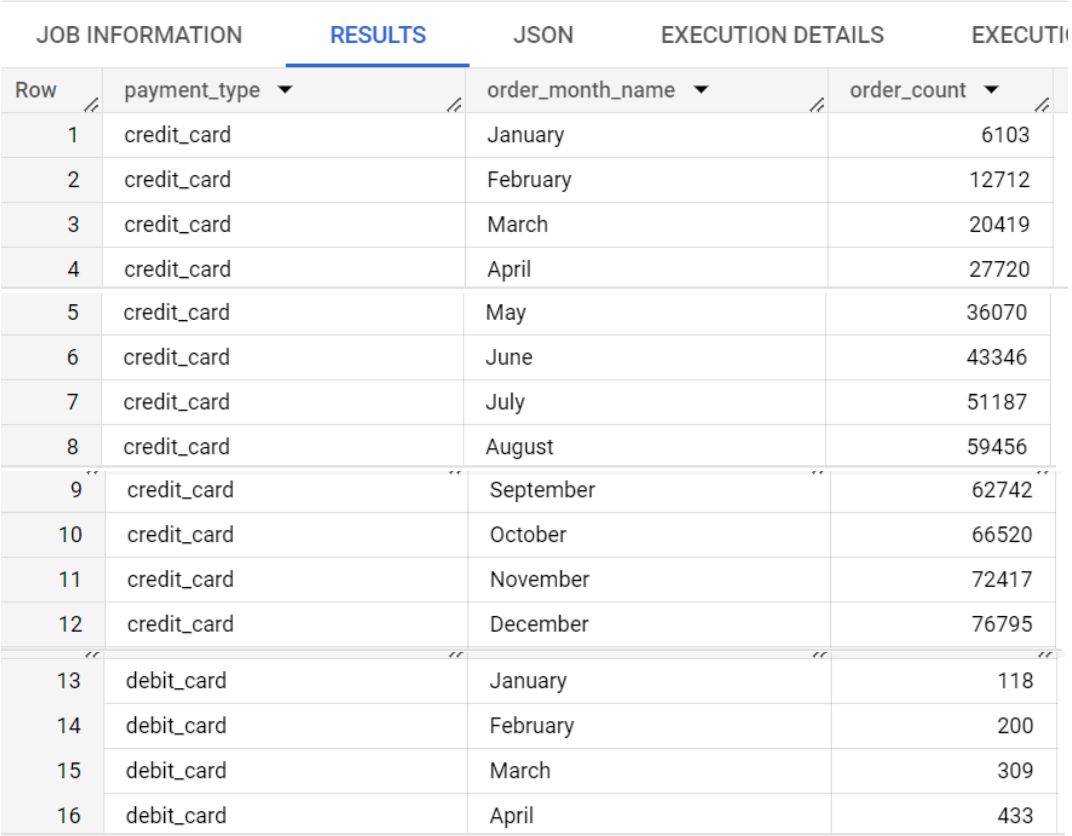
Section 6:

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

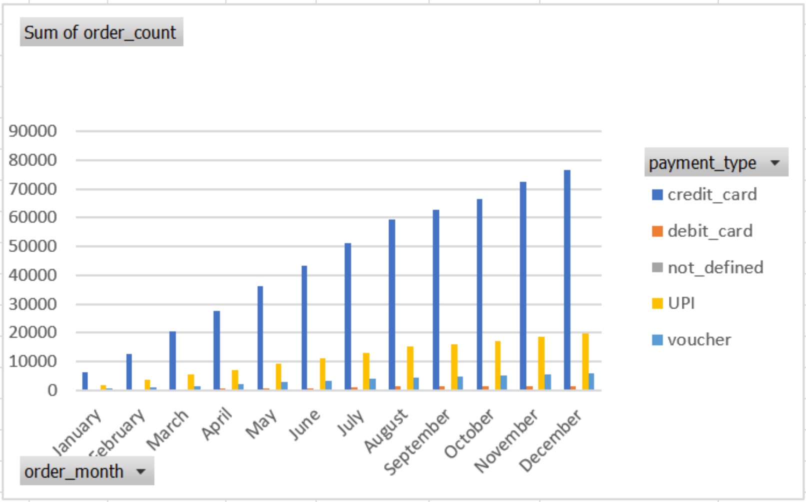
Query:

```
SELECT
  DISTINCT P.payment_type,
  VW.order_month_name,
  COUNT(VW.order_id) OVER(PARTITION BY P.payment_type ORDER BY VW.order_month) AS
order_count
FROM `online_orders.customers` C
INNER JOIN `online_orders.vw_orders_details` VW
  ON C.customer_id = VW.customer_id
INNER JOIN `online_orders.payments` P
  ON VW.order_id = P.order_id
ORDER BY UPPER(P.payment_type);
```

Output:



Row	payment_type	order_month_name	order_count
1	credit_card	January	6103
2	credit_card	February	12712
3	credit_card	March	20419
4	credit_card	April	27720
5	credit_card	May	36070
6	credit_card	June	43346
7	credit_card	July	51187
8	credit_card	August	59456
9	credit_card	September	62742
10	credit_card	October	66520
11	credit_card	November	72417
12	credit_card	December	76795
13	debit_card	January	118
14	debit_card	February	200
15	debit_card	March	309
16	debit_card	April	433

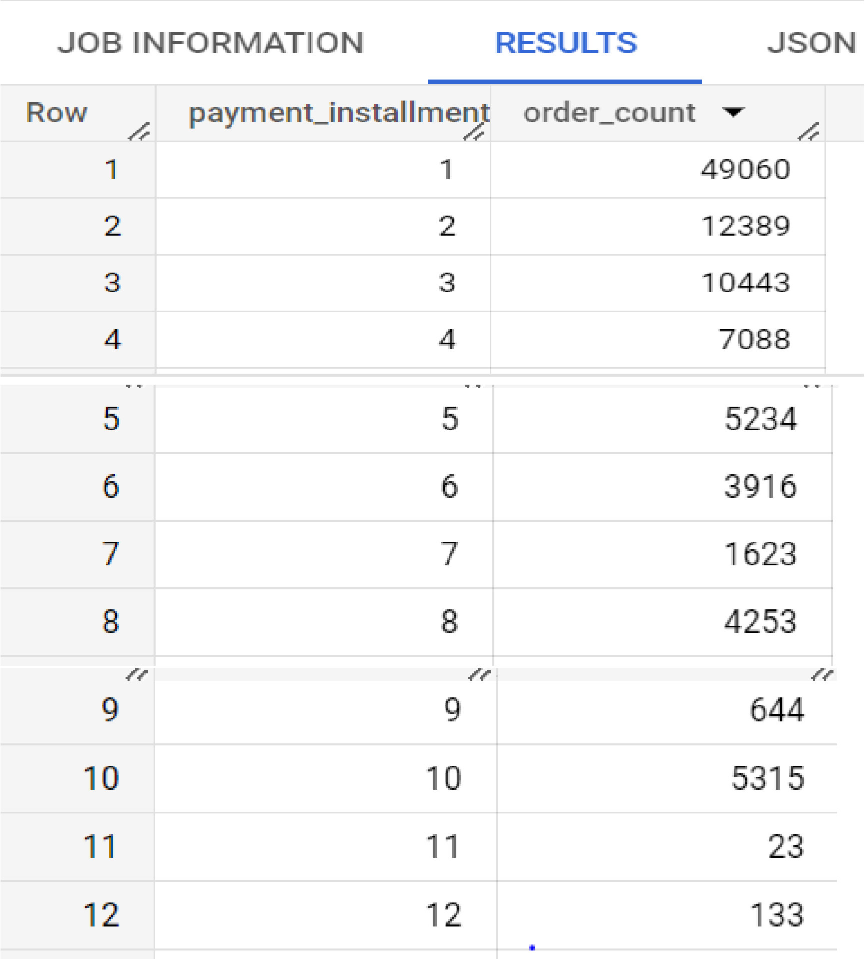


2. Find the no. of orders placed on the basis of the payment installments that have been paid.

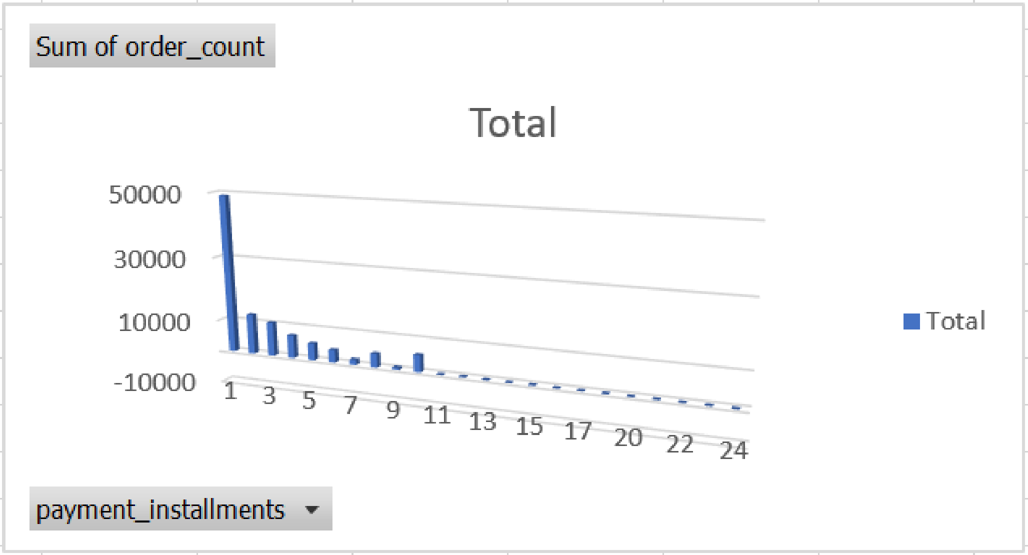
Query:

```
SELECT
  P.payment_installments,
  COUNT(DISTINCT VW.order_id) as order_count
FROM `online_orders.customers` C
INNER JOIN `online_orders.vw_orders_details` VW
  ON C.customer_id = VW.customer_id
INNER JOIN `online_orders.payments` P
  ON VW.order_id = P.order_id
WHERE P.payment_installments > 0
GROUP BY P.payment_installments
ORDER BY P.payment_installments;
```

Output:



Row	payment_installment	order_count
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315
11	11	23
12	12	133


Sum of order_count
Total
■ Total
payment_installments