TARGET SQL

Devaraj V

CONTENTS

NOTE: Google BigQuery will be used to load datasets and run queries. Original sql queries can be accesses via the link shared here:

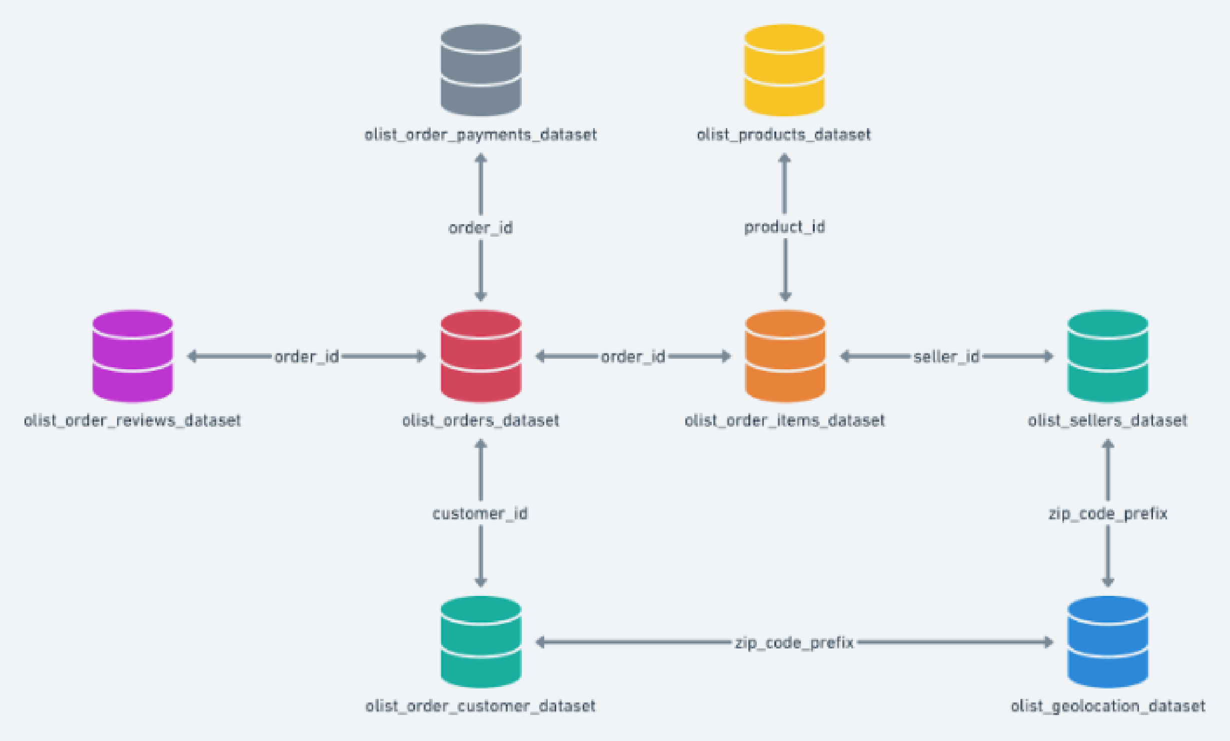
https://console.cloud.google.com/bigquery?sq=949039620844:eb159f7cba9d4a8889d209746014ab9d

1. Exploratory Data Analysis

NOTE:

- We will be using Google BigQuery to load datasets to server and run.
- Screenshots of queries, results of those queries and analysis part will be pasted in this file and insights will be provided.
- ER Diagram is shown below.

High level overview of relationship between datasets:

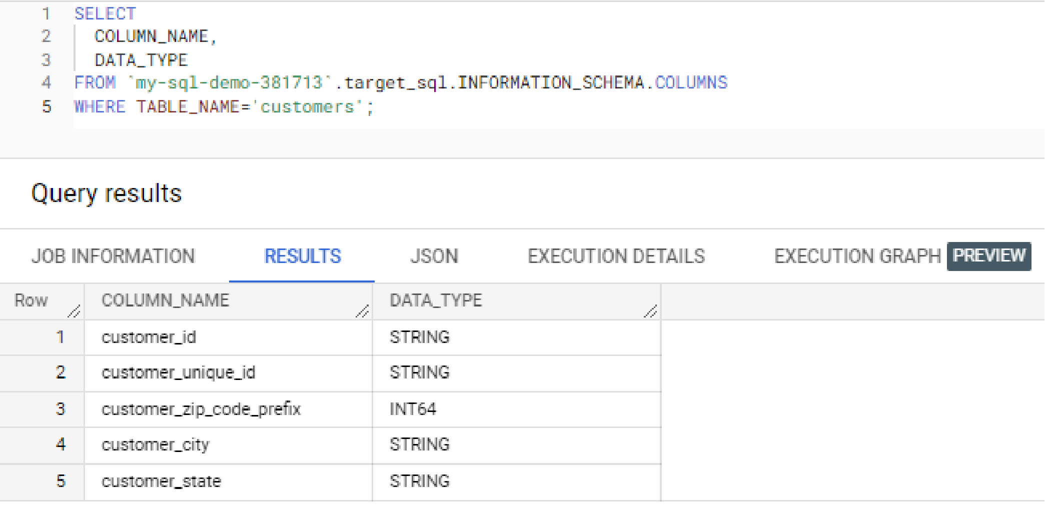


1.1 Data type of columns in the tables:

The target_sql database contains 8 tables and let us explore them one at a time.

1.1.1 Customers table:

Columns under the customers table:



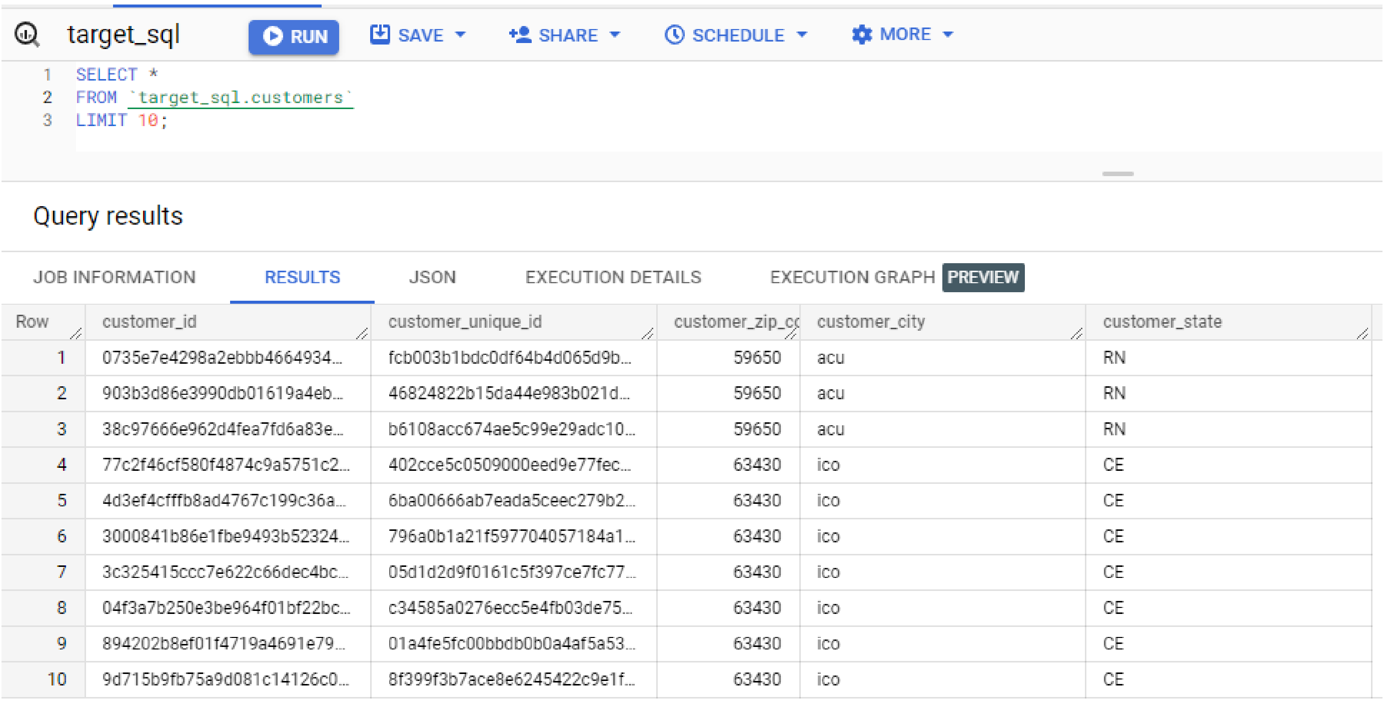
```
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM `my-sql-demo-381713`.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='customers';
```

Query results

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

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

First 10 rows of customers table: -

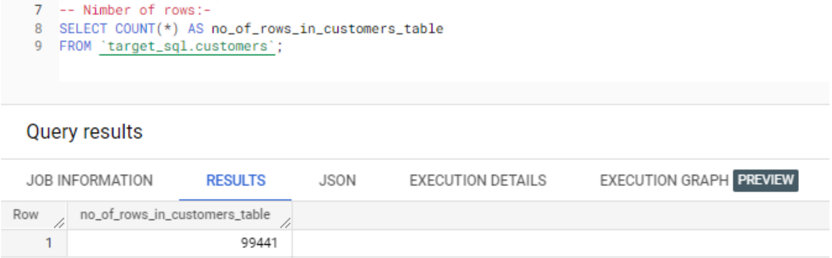


```
SELECT *
FROM `target_sql.customers`
LIMIT 10;
```

Query results

Row	customer_id	customer_unique_id	customer_zip_cc	customer_city	customer_state
1	0735e7e4298a2ebbb4664934…	fcb003b1bdc0df64b4d065d9b…	59650	acu	RN
2	903b3d86e3990db01619a4eb…	46824822b15da44e983b021d…	59650	acu	RN
3	38c97666e962d4fea7fd6a83e…	b6108acc674ae5c99e29adc10…	59650	acu	RN
4	77c2f46cf580f4874c9a5751c2…	402cce5c0509000eed9e77fec…	63430	ico	CE
5	4d3ef4cfffb8ad4767c199c36a…	6ba00666ab7eada5ceec279b2…	63430	ico	CE
6	3000841b86e1fbe9493b52324…	796a0b1a21f597704057184a1…	63430	ico	CE
7	3c325415ccc7e622c66dec4bc…	05d1d2d9f0161c5f397ce7fc77…	63430	ico	CE
8	04f3a7b250e3be964f01bf22bc…	c34585a0276ecc5e4fb03de75…	63430	ico	CE
9	894202b8ef01f4719a4691e79…	01a4fe5fc00bbdb0b0a4af5a53…	63430	ico	CE
10	9d715b9fb75a9d081c14126c0…	8f399f3b7ace8e6245422c9e1f…	63430	ico	CE

Number of rows in the customers table:-



```
-- Nimber of rows:-
SELECT COUNT(*) AS no_of_rows_in_customers_table
FROM `target_sql.customers`;
```

Query results

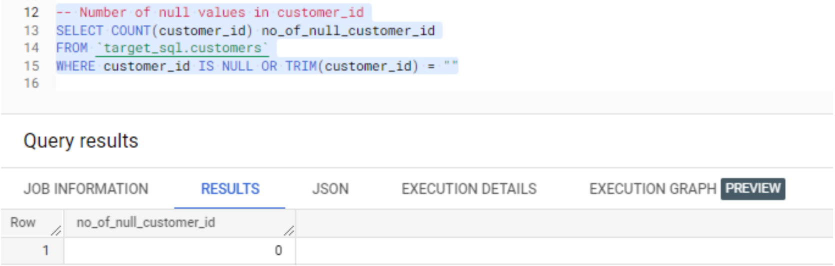
Row	no_of_rows_in_customers_table
1	99441

OBSERVATION: The customer table contains 99441 entries.

To know about the duplicates, we need to do further analysis of each column.

Nature of customer_id column: -

No. of NULLs customer id:

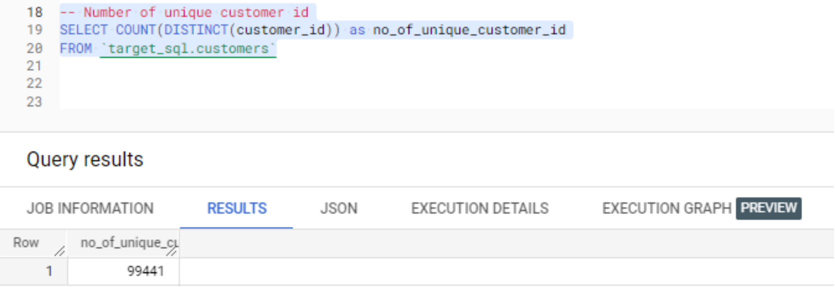


```
-- Number of null values in customer_id
SELECT COUNT(customer_id) no_of_null_customer_id
FROM `target_sql.customers`
WHERE customer_id IS NULL OR TRIM(customer_id) = ""
```

Query results

Row	no_of_null_customer_id
1	0

No. of unique customer id



```
-- Number of unique customer id
SELECT COUNT(DISTINCT(customer_id)) as no_of_unique_customer_id
FROM `target_sql.customers`
```

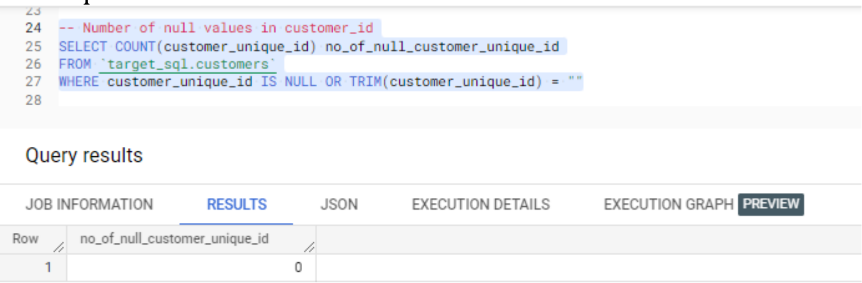
Query results

Row	no_of_unique_c
1	99441

OBSERVATIONS:

- The table contains a total of 99441 observations and the customer id column does not have any null values.
- Additionally, every entry in the customer id column is distinct. As a result, the customer_id column can serve as a unique identifier for the records stored in the customers table.

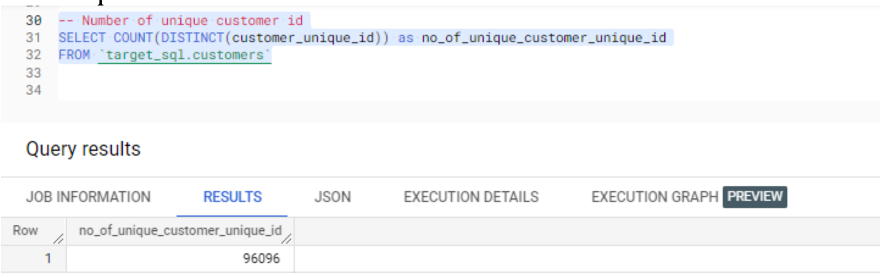
No. of NULLs customer unique id:



```
-- Number of null values in customer_id
SELECT COUNT(customer_unique_id) no_of_null_customer_unique_id
FROM `target_sql.customers`
WHERE customer_unique_id IS NULL OR TRIM(customer_unique_id) = ""
```

Row	no_of_null_customer_unique_id
1	0

No. of unique customer unique id:



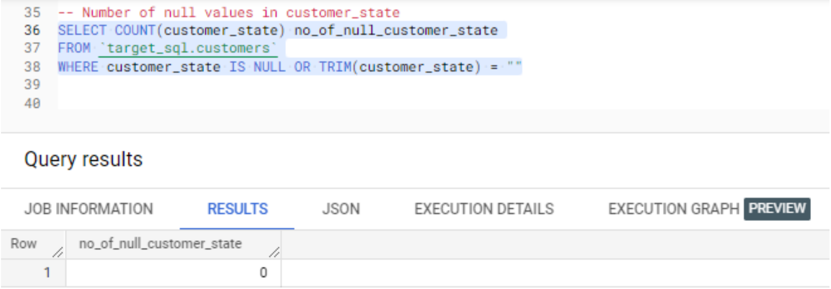
```
-- Number of unique customer id
SELECT COUNT(DISTINCT(customer_unique_id)) as no_of_unique_customer_unique_id
FROM `target_sql.customers`
```

Row	no_of_unique_customer_unique_id
1	96096

OBSERVATIONS:

- Despite the table containing 99441 observations and no null values in the customer_unique_id column, the analysis reveals that there are only 96096 unique customer_unique_ids.
- This implies that **there are 3345 instances of duplicates in the customer_unique_id column, rendering it unsuitable as a unique identifier**.

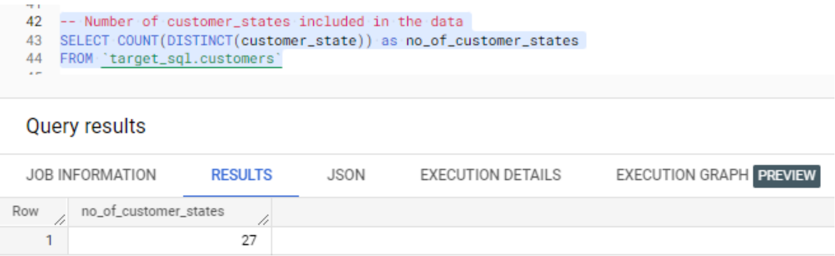
No. of NULLs in customer state column:



```
-- Number of null values in customer_state
SELECT COUNT(customer_state) no_of_null_customer_state
FROM `target_sql.customers`
WHERE customer_state IS NULL OR TRIM(customer_state) = ""
```

Row	no_of_null_customer_state
1	0

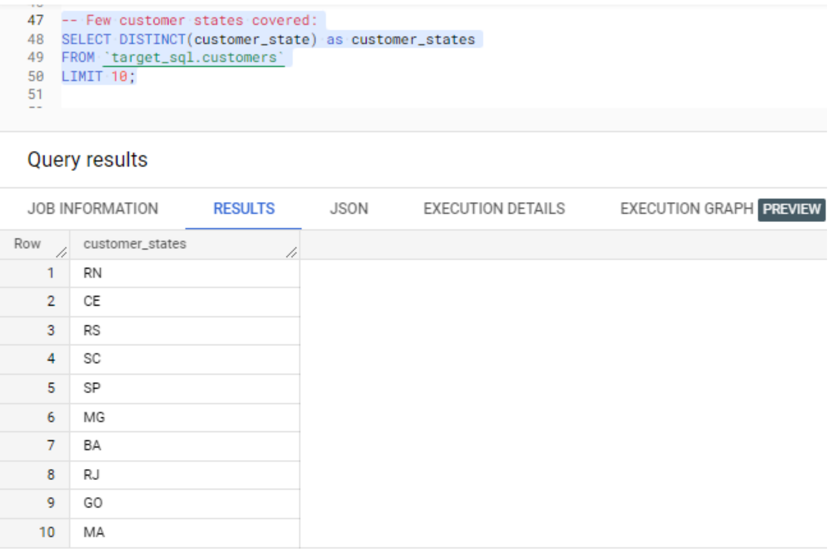
No. of customer states included in the data:



```
-- Number of customer_states included in the data
SELECT COUNT(DISTINCT(customer_state)) as no_of_customer_states
FROM `target_sql.customers`
```

Row	no_of_customer_states
1	27

Few customer states covered:



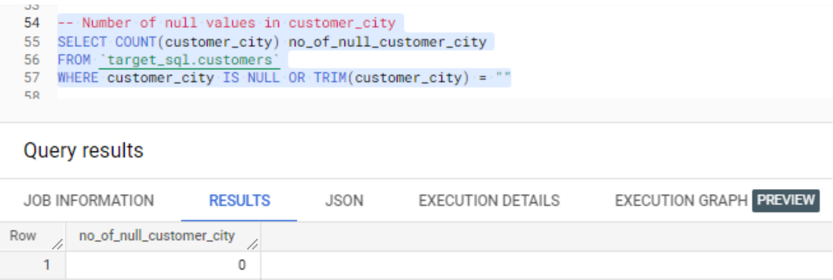
```
-- Few customer states covered:
SELECT DISTINCT(customer_state) as customer_states
FROM `target_sql.customers`
LIMIT 10;
```

Row	customer_states
1	RN
2	CE
3	RS
4	SC
5	SP
6	MG
7	BA
8	RJ
9	GO
10	MA

OBSERVATIONS:

- The customer_state column does not have any NULL values and it encompasses 27 distinct states.

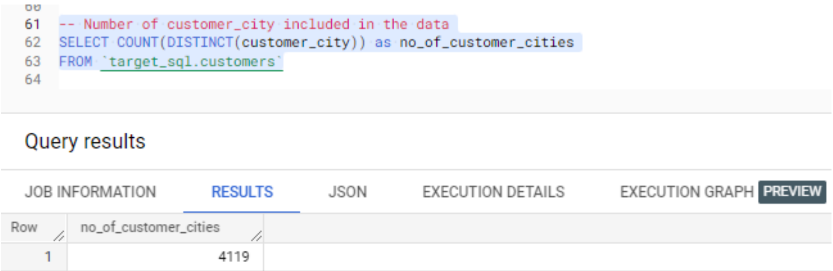
No. of NULLs in customer city column:



```
-- Number of null values in customer_city
SELECT COUNT(customer_city) no_of_null_customer_city
FROM `target_sql.customers`
WHERE customer_city IS NULL OR TRIM(customer_city) = ""
```

Row	no_of_null_customer_city
1	0

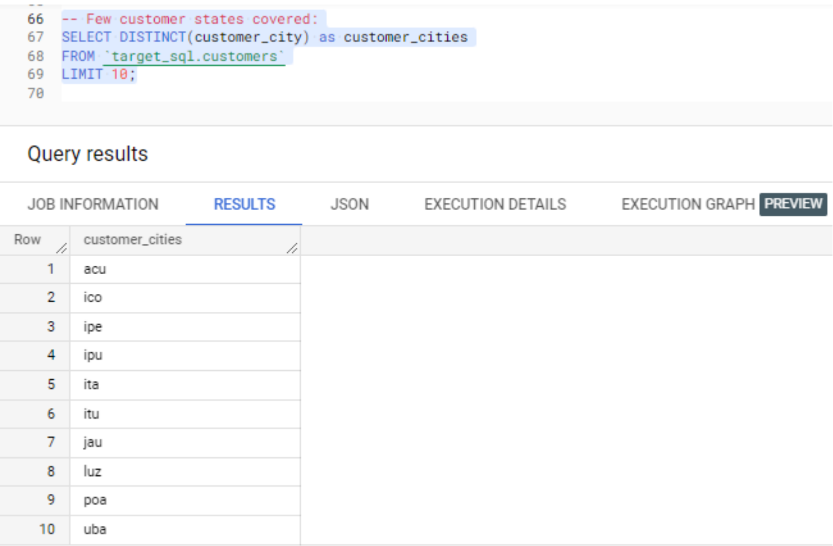
No. of customer cities included in the data:



```
-- Number of customer_city included in the data
SELECT COUNT(DISTINCT(customer_city)) as no_of_customer_cities
FROM `target_sql.customers`
```

Row	no_of_customer_cities
1	4119

Few customers city covered:



```
-- Few customer states covered:
SELECT DISTINCT(customer_city) as customer_cities
FROM `target_sql.customers`
LIMIT 10;
```

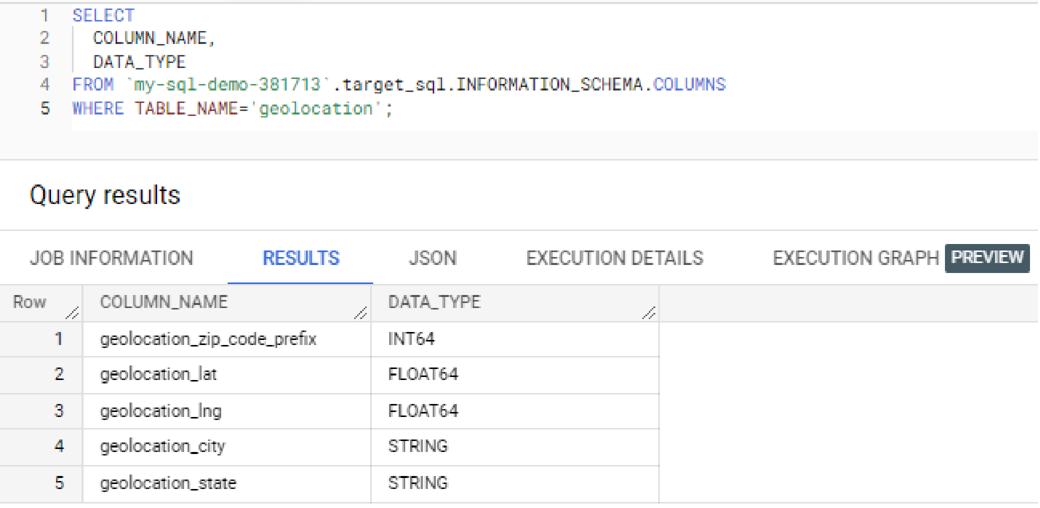
Row	customer_cities
1	acu
2	ico
3	ipe
4	ipu
5	ita
6	itu
7	jau
8	luz
9	poa
10	uba

OBSERVATIONS:

- The customer_city column does not have any NULL values and comprises 4119 distinct cities.

1.1.2 Geolocation table:

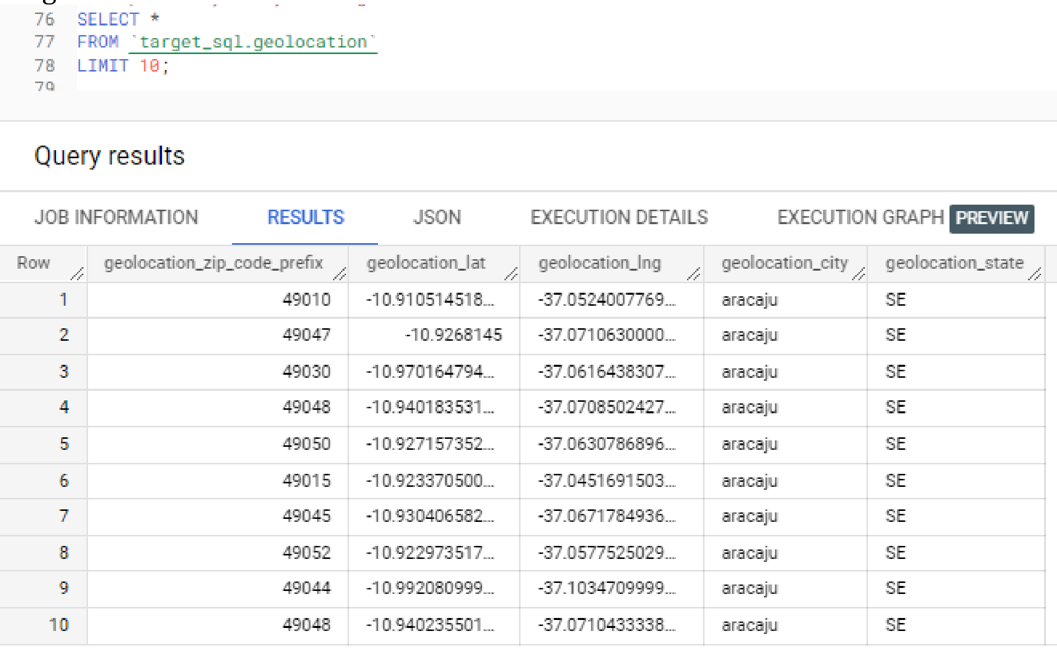
Columns under the geolocation table:



```
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM `my-sql-demo-381713`.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='geolocation';
```

Row	COLUMN_NAME	DATA_TYPE
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING

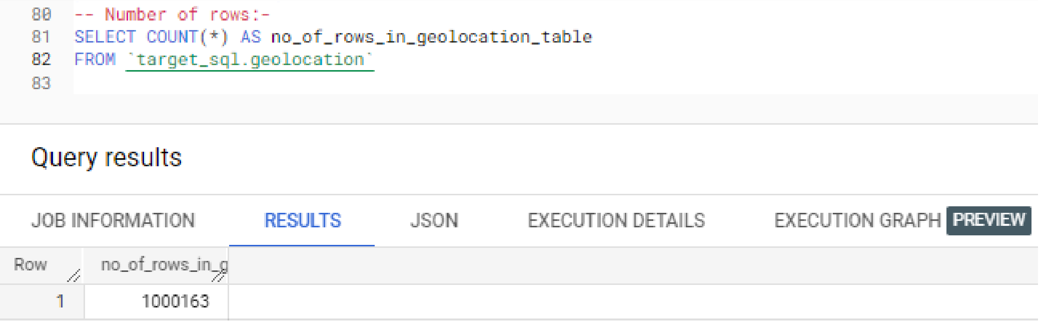
First 10 rows of geolocation table: -



```
SELECT *
FROM `target_sql.geolocation`
LIMIT 10;
```

Row	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
1	49010	-10.910514518...	-37.0524007769...	aracaju	SE
2	49047	-10.9268145	-37.0710630000...	aracaju	SE
3	49030	-10.970164794...	-37.0616438307...	aracaju	SE
4	49048	-10.940183531...	-37.0708502427...	aracaju	SE
5	49050	-10.927157352...	-37.0630786896...	aracaju	SE
6	49015	-10.923370500...	-37.0451691503...	aracaju	SE
7	49045	-10.930406582...	-37.0671784936...	aracaju	SE
8	49052	-10.922973517...	-37.0577525029...	aracaju	SE
9	49044	-10.992080999...	-37.1034709999...	aracaju	SE
10	49048	-10.940235501...	-37.0710433338...	aracaju	SE

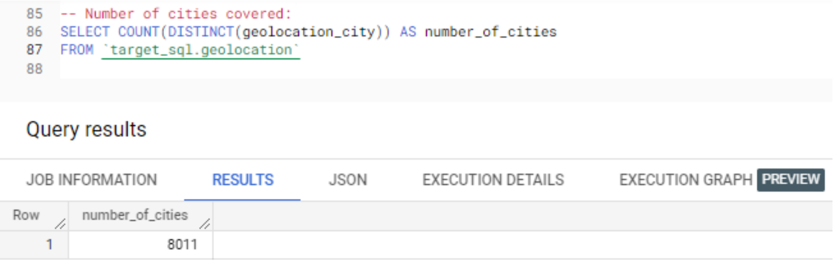
Number of rows in the geolocation table:-



```
-- Number of rows:-
SELECT COUNT(*) AS no_of_rows_in_geolocation_table
FROM `target_sql.geolocation`
```

Row	no_of_rows_in_g
1	1000163

No. of cities covered



```
-- Number of cities covered:
SELECT COUNT(DISTINCT(geolocation_city)) AS number_of_cities
FROM `target_sql.geolocation`
```

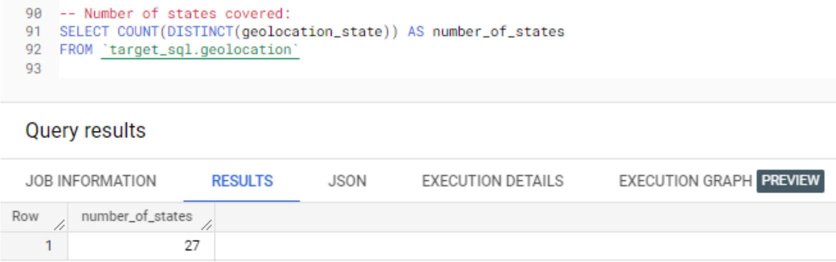
Query results

Row	number_of_cities
1	8011

OBSERVATIONS:

- According to the customers table data, there were customers from 4,119 cities out of the 8,011 cities present in the dataset.
- This suggests that no orders were placed from the remaining 3,892 cities, which accounts for almost half of the total number of cities.
- As a result, the distribution of customers is uneven, and it appears that nearly half of the potential market has not been reached.

No. of states covered



```
-- Number of states covered:
SELECT COUNT(DISTINCT(geolocation_state)) AS number_of_states
FROM `target_sql.geolocation`
```

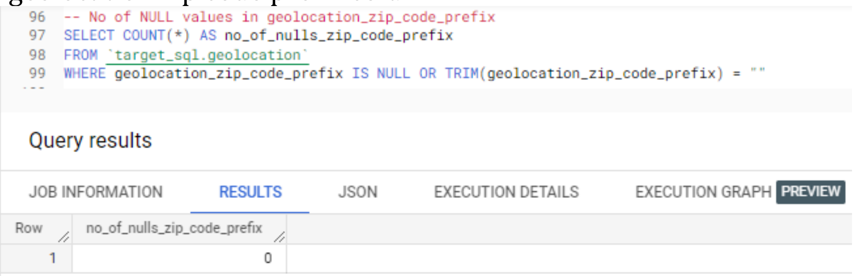
Query results

Row	number_of_states
1	27

OBSERVATIONS:

- According to data obtained from both customer records and a geolocation table, orders have been placed from all 27 states.

No. of NULL values in geolocation zip code prefix column



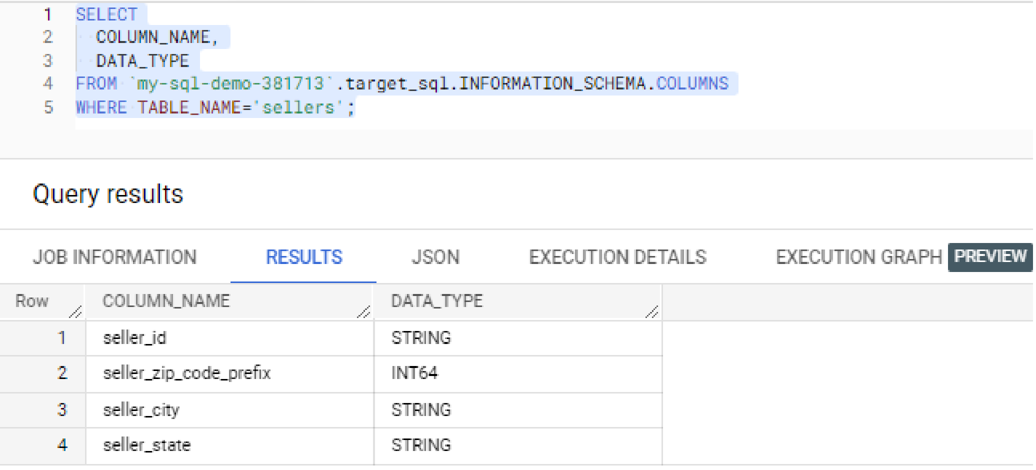
```
-- No of NULL values in geolocation_zip_code_prefix
SELECT COUNT(*) AS no_of_nulls_zip_code_prefix
FROM `target_sql.geolocation`
WHERE geolocation_zip_code_prefix IS NULL OR TRIM(geolocation_zip_code_prefix) = ""
```

Query results

Row	no_of_nulls_zip_code_prefix
1	0

1.1.3 Seller table:

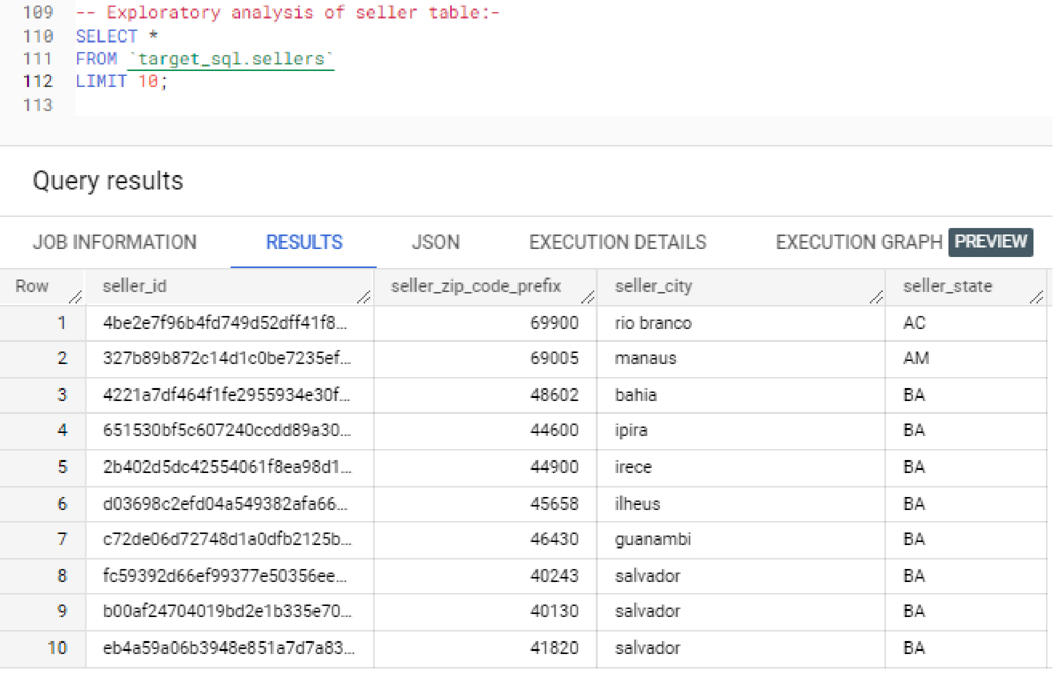
Columns under the seller table:



```
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM `my-sql-demo-381713`.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='sellers';
```

Row	COLUMN_NAME	DATA_TYPE
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

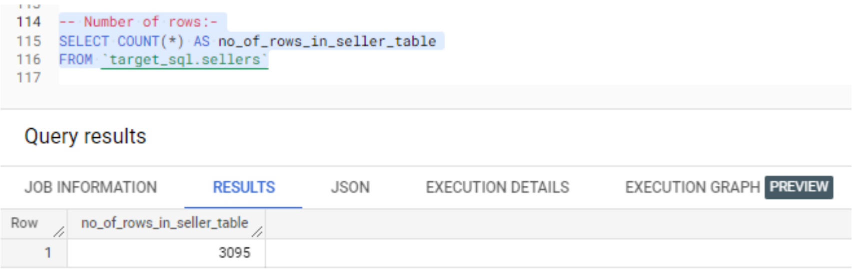
First 10 rows of seller table: -



```
-- Exploratory analysis of seller table:-
SELECT *
FROM `target_sql.sellers`
LIMIT 10;
```

Row	seller_id	seller_zip_code_prefix	seller_city	seller_state
1	4be2e7f96b4fd749d52dff41f8...	69900	rio branco	AC
2	327b89b872c14d1c0be7235ef...	69005	manaus	AM
3	4221a7df464f1fe2955934e30f...	48602	bahia	BA
4	651530bf5c607240ccdd89a30...	44600	ipira	BA
5	2b402d5dc42554061f8ea98d1...	44900	irece	BA
6	d03698c2efd04a549382afa66...	45658	ilheus	BA
7	c72de06d72748d1a0dfb2125b...	46430	guanambi	BA
8	fc59392d66ef99377e50356ee...	40243	salvador	BA
9	b00af24704019bd2e1b335e70...	40130	salvador	BA
10	eb4a59a06b3948e851a7d7a83...	41820	salvador	BA

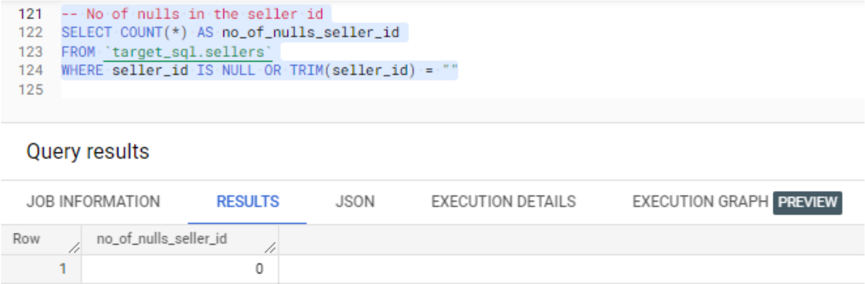
Number of rows in the seller table: -



```
-- Number of rows:-
SELECT COUNT(*) AS no_of_rows_in_seller_table
FROM `target_sql.sellers`
```

Row	no_of_rows_in_seller_table
1	3095

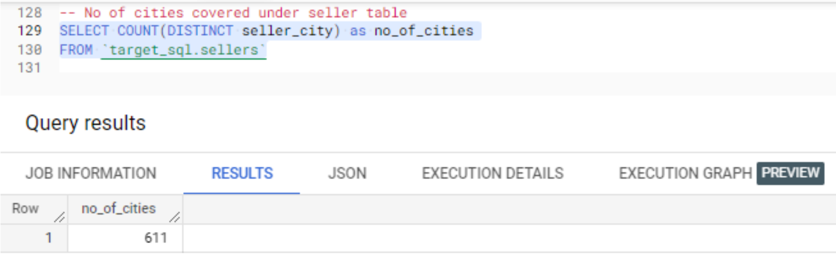
Number of NULLs in the seller id: -



```
-- No of nulls in the seller id
SELECT COUNT(*) AS no_of_nulls_seller_id
FROM `target_sql.sellers`
WHERE seller_id IS NULL OR TRIM(seller_id) = ""
```

Row	no_of_nulls_seller_id
1	0

No of cities covered under the seller table:



```
-- No of cities covered under seller table
SELECT COUNT(DISTINCT seller_city) as no_of_cities
FROM `target_sql.sellers`
```

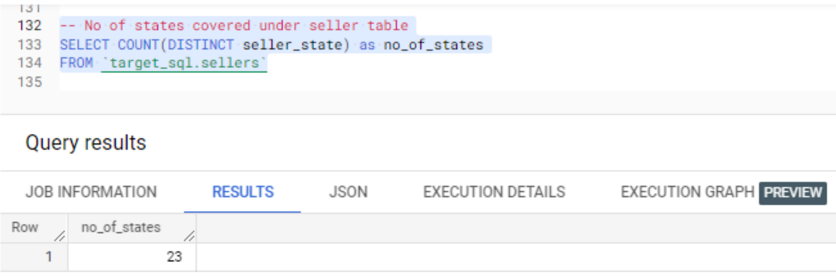
Query results

Row	no_of_cities
1	611

OBSERVATIONS: -

- Out of the total 8000 plus cities, customers belonged from just over 4000 cities.
- And sales mainly happen from only 611 cities.
- This indicates that sellers are primarily concentrated in a small number of cities.

No of states covered under the seller table:



```
-- No of states covered under seller table
SELECT COUNT(DISTINCT seller_state) as no_of_states
FROM `target_sql.sellers`
```

Query results

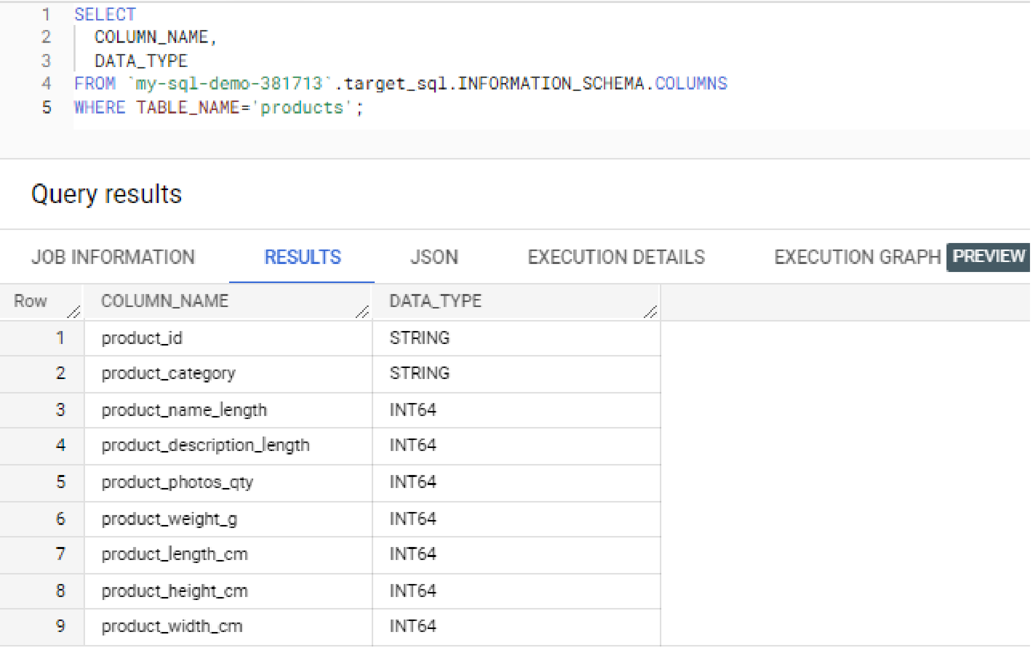
Row	no_of_states
1	23

OBSERVATIONS: -

- The customer data indicates that there are customers located in 27 states, whereas the sellers are only present in 23 states.
- This implies that there are 4 states where sellers are not present.

1.1.4 **Products table:**

Columns under the products table:

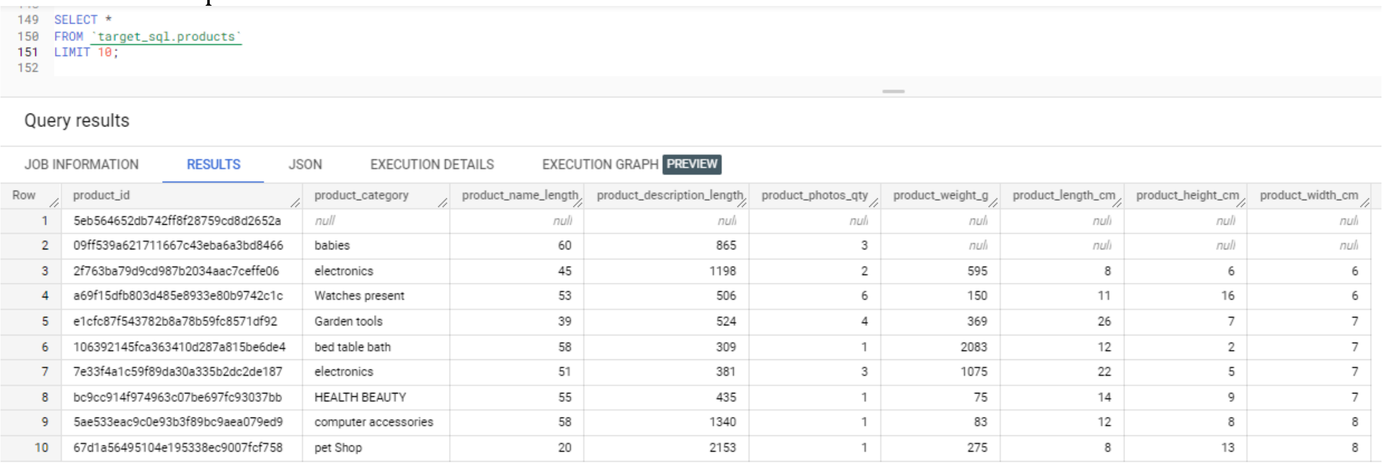


```
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM `my-sql-demo-381713`.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='products';
```

Query results

Row	COLUMN_NAME	DATA_TYPE
1	product_id	STRING
2	product_category	STRING
3	product_name_length	INT64
4	product_description_length	INT64
5	product_photos_qty	INT64
6	product_weight_g	INT64
7	product_length_cm	INT64
8	product_height_cm	INT64
9	product_width_cm	INT64

First 10 rows of products table: -

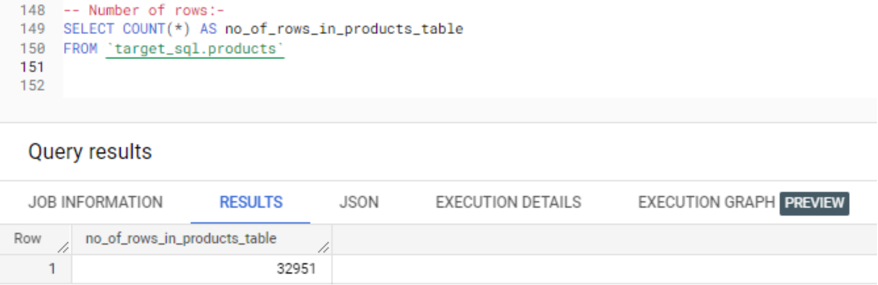


```
SELECT *
FROM `target_sql.products`
LIMIT 10;
```

Query results

Row	product_id	product_category	product_name_length	product_description_length	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm
1	5eb564652db742ff8f28759cd8d2652a	null	null	null	null	null	null	null	null
2	09ff539a621711667c43eba6a3bd8466	babies	60	865	3	null	null	null	null
3	2f763ba79d9cd987b2034aac7ceffe06	electronics	45	1198	2	595	8	6	6
4	a69f15dfb803d485e8933e80b9742c1c	Watches present	53	506	6	150	11	16	6
5	e1cfc87f543782b8a78b59fc8571df92	Garden tools	39	524	4	369	26	7	7
6	106392145fca363410d287a815be6de4	bed table bath	58	309	1	2083	12	2	7
7	7e33f4a1c59f89da30a335b2dc2de187	electronics	51	381	3	1075	22	5	7
8	bc9cc914f974963c07be697fc93037bb	HEALTH BEAUTY	55	435	1	75	14	9	7
9	5ae533eac9c0e93b3f89bc9aea079ed9	computer accessories	58	1340	1	83	12	8	8
10	67d1a56495104e195338ec9007fcf758	pet Shop	20	2153	1	275	8	13	8

Number of rows in the products table: -

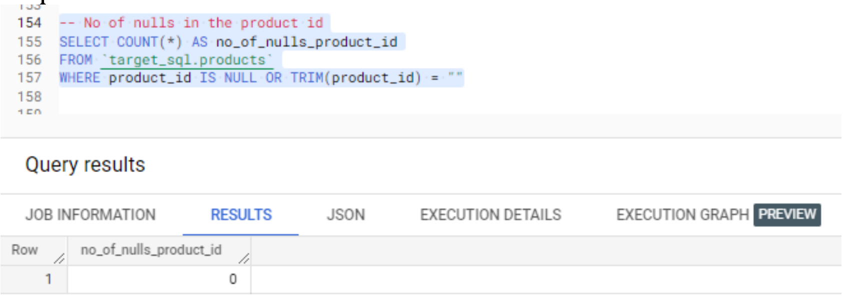


```
-- Number of rows:-
SELECT COUNT(*) AS no_of_rows_in_products_table
FROM `target_sql.products`
```

Query results

Row	no_of_rows_in_products_table
1	32951

Number of NULLs in the product id: -

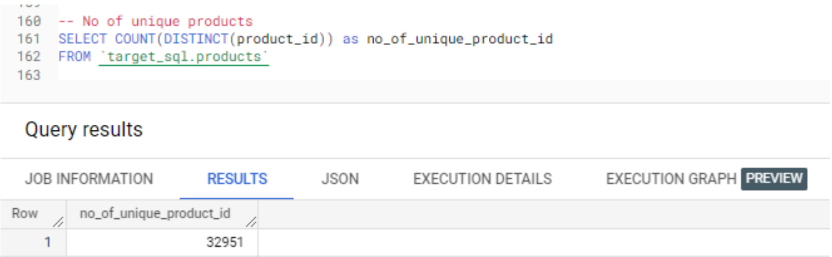


```
-- No of nulls in the product id
SELECT COUNT(*) AS no_of_nulls_product_id
FROM `target_sql.products`
WHERE product_id IS NULL OR TRIM(product_id) = ""
```

Query results

Row	no_of_nulls_product_id
1	0

No of unique products:



```
-- No of unique products
SELECT COUNT(DISTINCT(product_id)) as no_of_unique_product_id
FROM `target_sql.products`
```

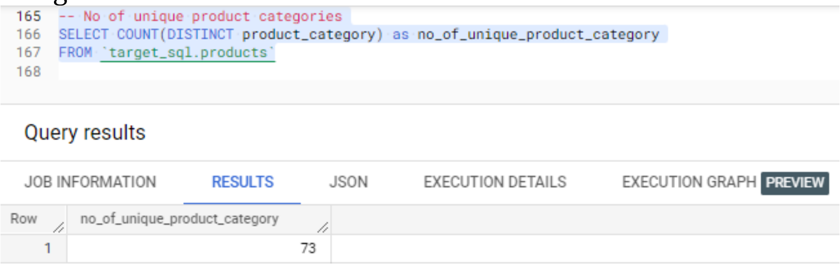
Query results

Row	no_of_unique_product_id
1	32951

OBSERVATIONS: -

- As the number of rows in the product table and the number of unique products are same, it means that **each row represents a unique product**

No of unique product categories:

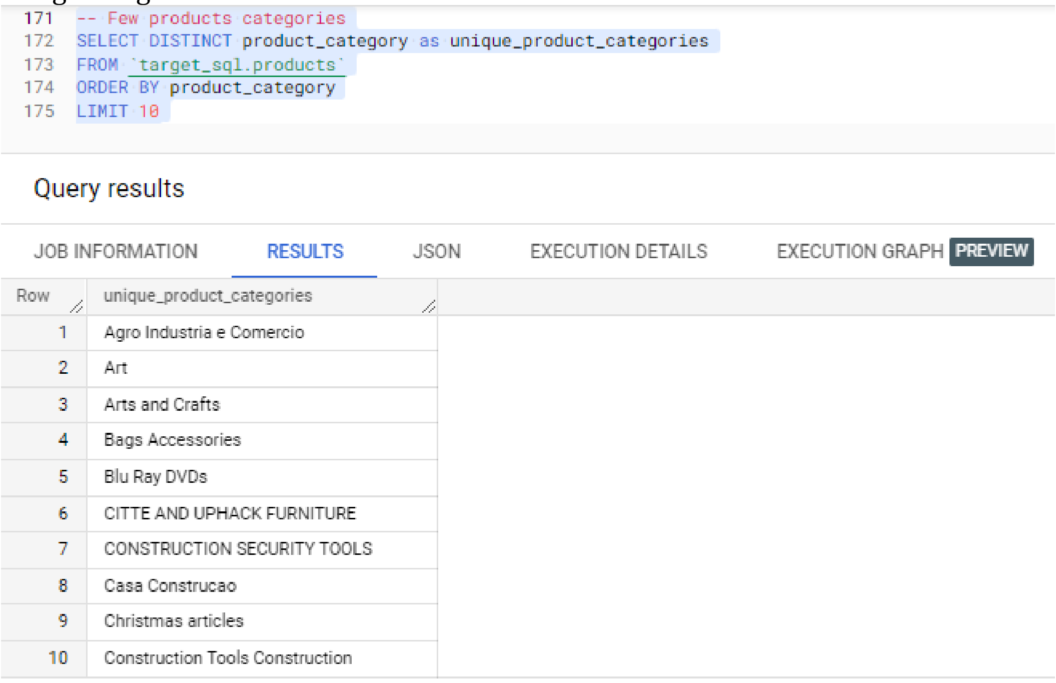


```
-- No of unique product categories
SELECT COUNT(DISTINCT product_category) as no_of_unique_product_category
FROM `target_sql.products`
```

Query results

Row	no_of_unique_product_category
1	73

Few product categories given:

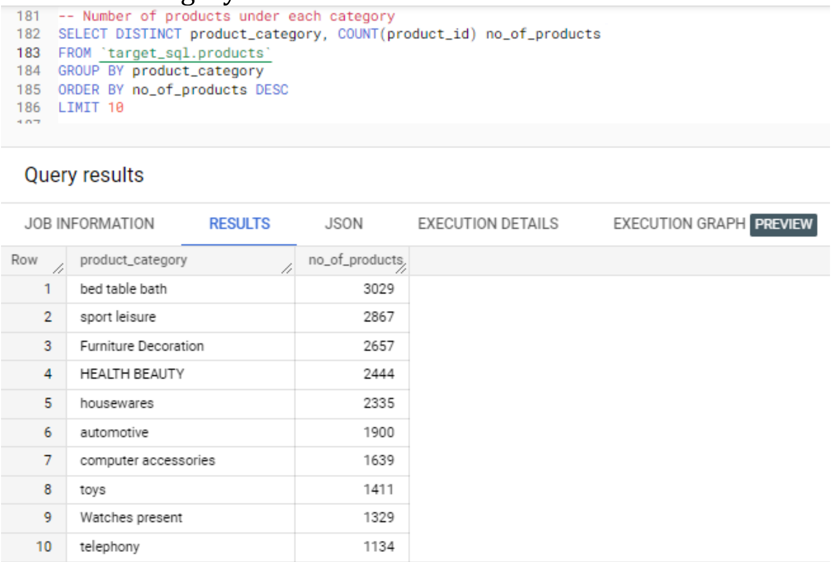


```
-- Few products categories
SELECT DISTINCT product_category as unique_product_categories
FROM `target_sql.products`
ORDER BY product_category
LIMIT 10
```

Query results

Row	unique_product_categories
1	Agro Industria e Comercio
2	Art
3	Arts and Crafts
4	Bags Accessories
5	Blu Ray DVDs
6	CITTE AND UPHACK FURNITURE
7	CONSTRUCTION SECURITY TOOLS
8	Casa Construcao
9	Christmas articles
10	Construction Tools Construction

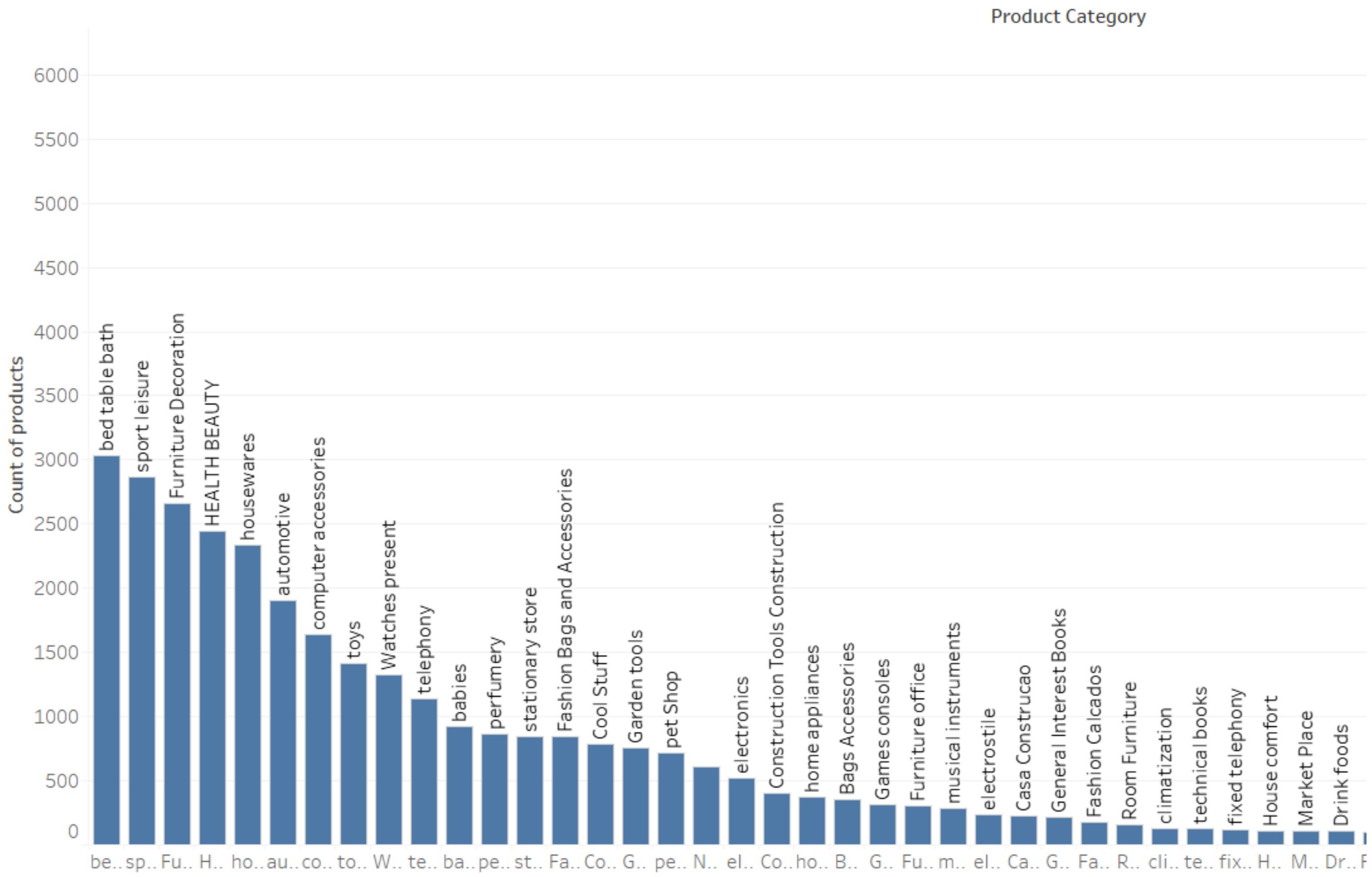
Number of products under each category:



```
-- Number of products under each category
SELECT DISTINCT product_category, COUNT(product_id) no_of_products
FROM `target_sql.products`
GROUP BY product_category
ORDER BY no_of_products DESC
LIMIT 10
```

Query results

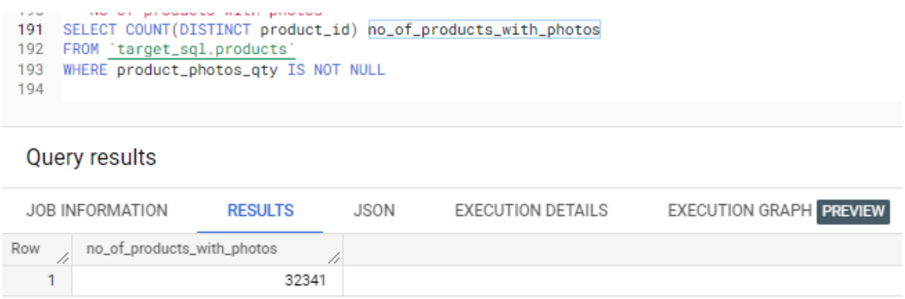
Row	product_category	no_of_products
1	bed table bath	3029
2	sport leisure	2867
3	Furniture Decoration	2657
4	HEALTH BEAUTY	2444
5	housewares	2335
6	automotive	1900
7	computer accessories	1639
8	toys	1411
9	Watches present	1329
10	telephony	1134



OBSERVATIONS:

- The 'bed table bath' category has the highest number of products (3029 items).
- Lowest number (1 no.) of products are under ‘CDs music DVDs’
- The top 5 categories contain over 2000 products each, while the bottom 5 categories have 5 products or less.

Number of products with phots: -

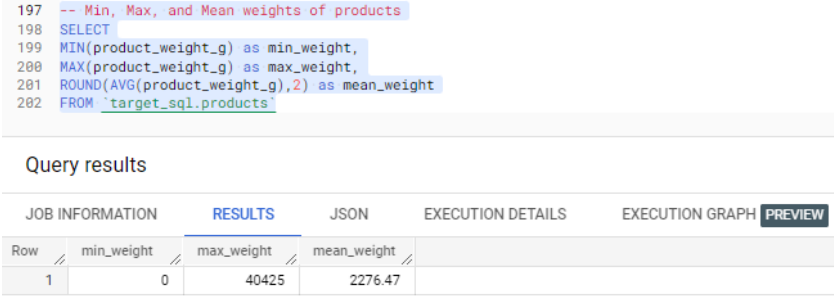


```
SELECT COUNT(DISTINCT product_id) no_of_products_with_photos
FROM `target_sql.products`
WHERE product_photos_qty IS NOT NULL
```

Query results

Row	no_of_products_with_photos
1	32341

Min, Max and Mean weight of products (in grams): -

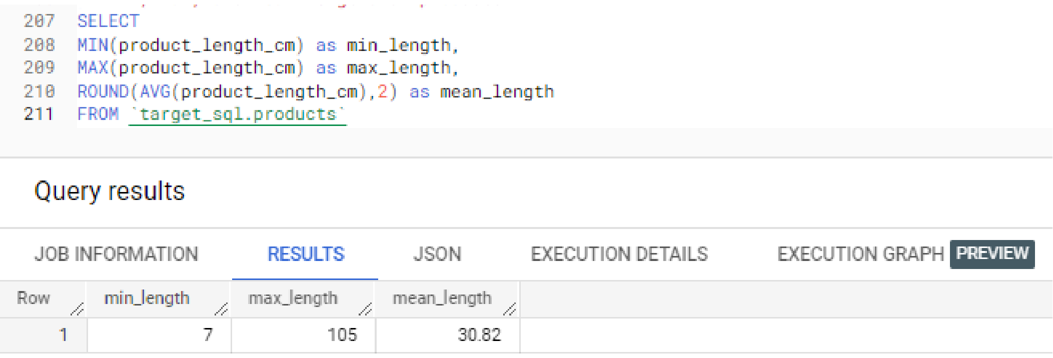


```
-- Min, Max, and Mean weights of products
SELECT
MIN(product_weight_g) as min_weight,
MAX(product_weight_g) as max_weight,
ROUND(AVG(product_weight_g),2) as mean_weight
FROM `target_sql.products`
```

Query results

Row	min_weight	max_weight	mean_weight
1	0	40425	2276.47

Min, Max and Mean length of products (in cm): -

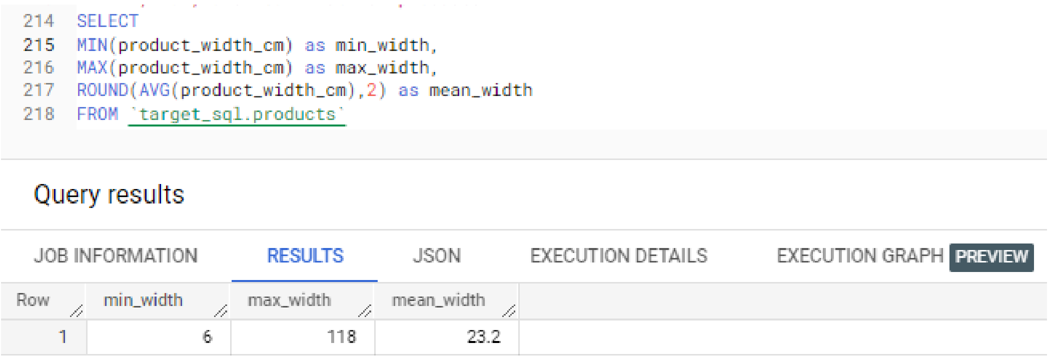


```sql
SELECT
MIN(product_length_cm) as min_length,
MAX(product_length_cm) as max_length,
ROUND(AVG(product_length_cm),2) as mean_length
FROM `target_sql.products`
```

Query results

Row	min_length	max_length	mean_length
1	7	105	30.82

Min, Max and Mean width of products (in cm): -

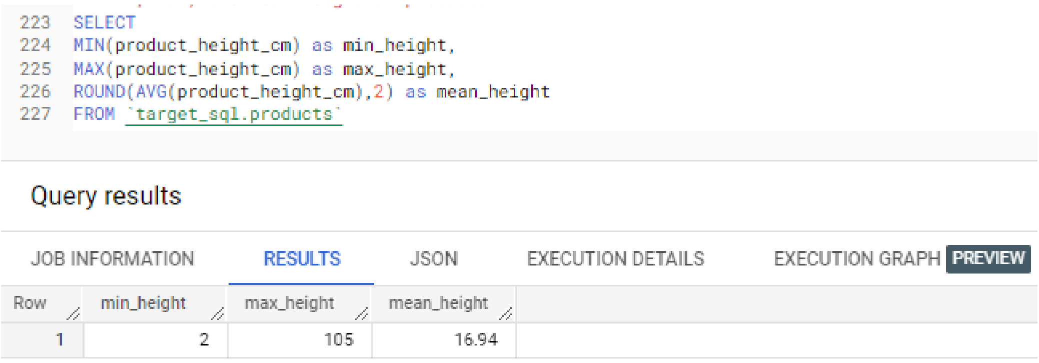


```sql
SELECT
MIN(product_width_cm) as min_width,
MAX(product_width_cm) as max_width,
ROUND(AVG(product_width_cm),2) as mean_width
FROM `target_sql.products`
```

Query results

Row	min_width	max_width	mean_width
1	6	118	23.2

Min, Max and Mean height of products (in cm)



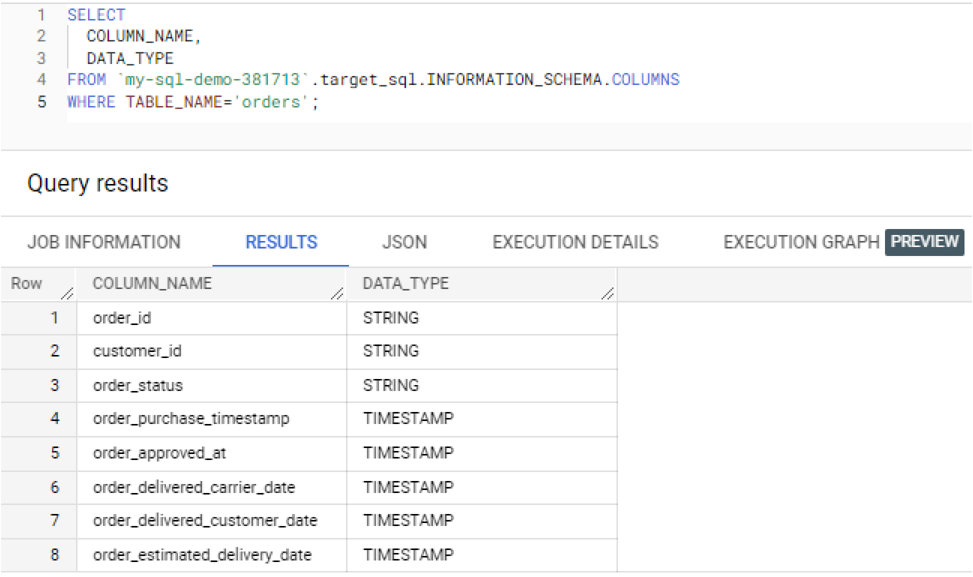
```sql
SELECT
MIN(product_height_cm) as min_height,
MAX(product_height_cm) as max_height,
ROUND(AVG(product_height_cm),2) as mean_height
FROM `target_sql.products`
```

Query results

Row	min_height	max_height	mean_height
1	2	105	16.94

1.1.5 **<u>Orders table:</u>**

Columns under the orders table:



```sql
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM `my-sql-demo-381713`.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='orders';
```

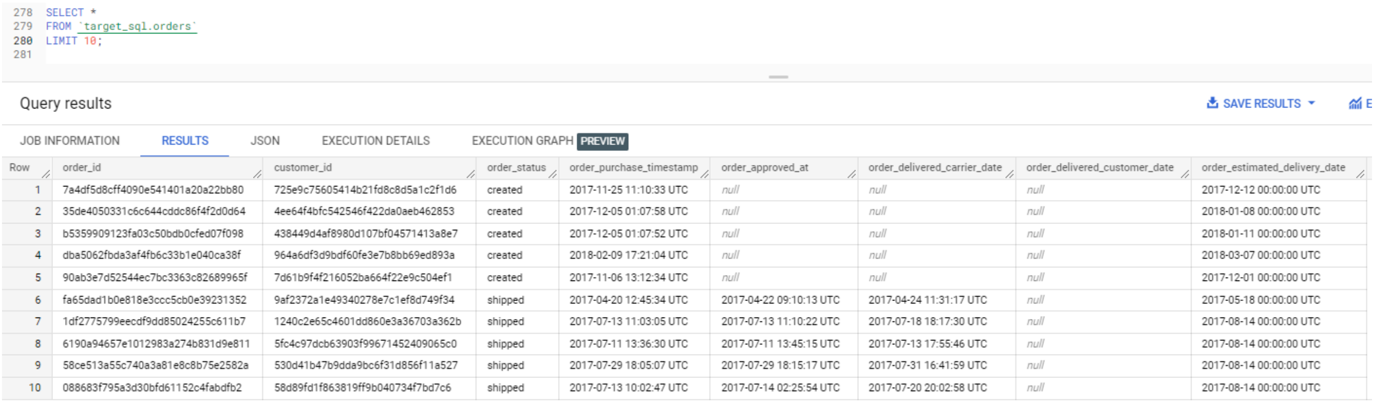
Query results

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

OBSERVATIONS:

- The columns order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, and order_estimated_delivery_date are of type TIMESTAMP in SQL
- SQL datetime functions can be utilized to obtain outputs from these columns.

First 10 rows of orders table: -

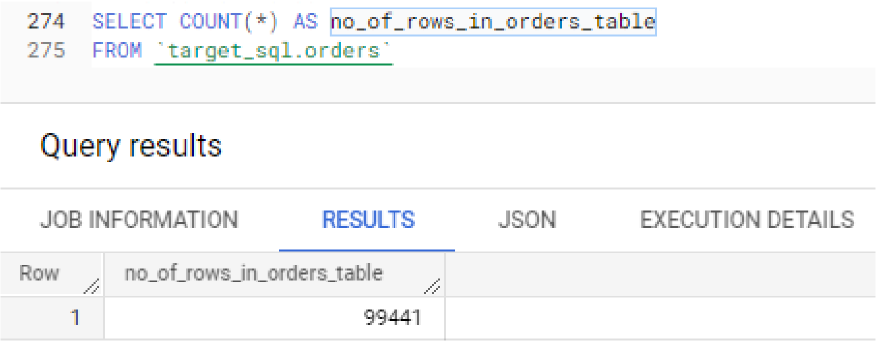


```sql
SELECT *
FROM `target_sql.orders`
LIMIT 10;
```

Query results

Row	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
1	7a4df5d8cff4090e541401a20a22bb80	725e9c75605414b21fd8c8d5a1c2f1d6	created	2017-11-25 11:10:33 UTC	null	null	null	2017-12-12 00:00:00 UTC
2	35de4050331c6c644cddc86f4f2d0d64	4ee64f4bfc542546f422da0aeb462853	created	2017-12-05 01:07:58 UTC	null	null	null	2018-01-08 00:00:00 UTC
3	b5359909123fa03c50bdb0cfed07f098	438449d4af8980d107bf04571413a8e7	created	2017-12-05 01:07:52 UTC	null	null	null	2018-01-11 00:00:00 UTC
4	dba5062fbda3af4fb6c33b1e040ca38f	964a6df3d9bdf60fe3e7b8bb69ed893a	created	2018-02-09 17:21:04 UTC	null	null	null	2018-03-07 00:00:00 UTC
5	90ab3e7d52544ec7bc3363c82689965f	7d61b9f4f216052ba664f22e9c504ef1	created	2017-11-06 13:12:34 UTC	null	null	null	2017-12-01 00:00:00 UTC
6	fa65dad1b0e818e3ccc5cb0e39231352	9af2372a1e49340278e7c1ef8d749f34	shipped	2017-04-20 12:45:34 UTC	2017-04-22 09:10:13 UTC	2017-04-24 11:31:17 UTC	null	2017-05-18 00:00:00 UTC
7	1df2775799eecdf9dd85024255c611b7	1240c2e65c4601dd860e3a36703a362b	shipped	2017-07-13 11:03:05 UTC	2017-07-13 11:10:22 UTC	2017-07-18 18:17:30 UTC	null	2017-08-14 00:00:00 UTC
8	6190a94657e1012983a274b831d9e811	5fc4c97dcb63903f99671452409065c0	shipped	2017-07-11 13:36:30 UTC	2017-07-11 13:45:15 UTC	2017-07-13 17:55:46 UTC	null	2017-08-14 00:00:00 UTC
9	58ce513a55c740a3a81e8c8b75e2582a	530d41b47b9dda9bc6f31d856f11a527	shipped	2017-07-29 18:05:07 UTC	2017-07-29 18:15:17 UTC	2017-07-31 16:41:59 UTC	null	2017-08-14 00:00:00 UTC
10	088683f795a3d30bfd61152c4fabdfb2	58d89fd1f863819ff9b040734f7bd7c6	shipped	2017-07-13 10:02:47 UTC	2017-07-14 02:25:54 UTC	2017-07-20 20:02:58 UTC	null	2017-08-14 00:00:00 UTC

Number of rows in the orders table: -

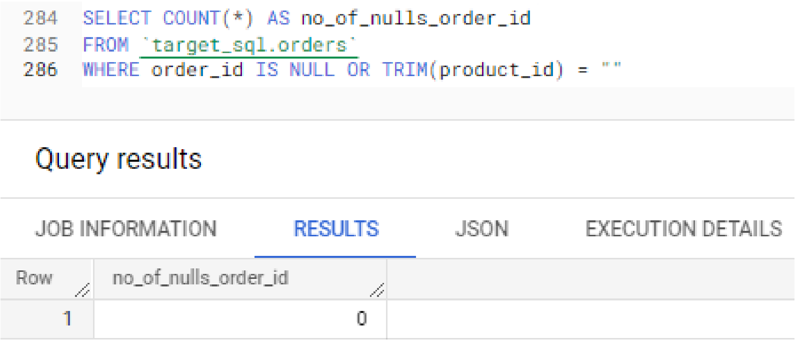


```sql
SELECT COUNT(*) AS no_of_rows_in_orders_table
FROM `target_sql.orders`
```

Query results

Row	no_of_rows_in_orders_table
1	99441

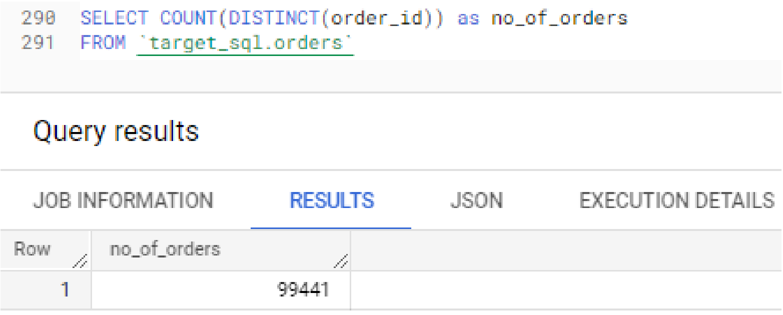
Number of NULLs in the orders id: -



```
SELECT COUNT(*) AS no_of_nulls_order_id
FROM `target_sql.orders`
WHERE order_id IS NULL OR TRIM(product_id) = ""
```

Row	no_of_nulls_order_id
1	0

No of orders under the orders table:



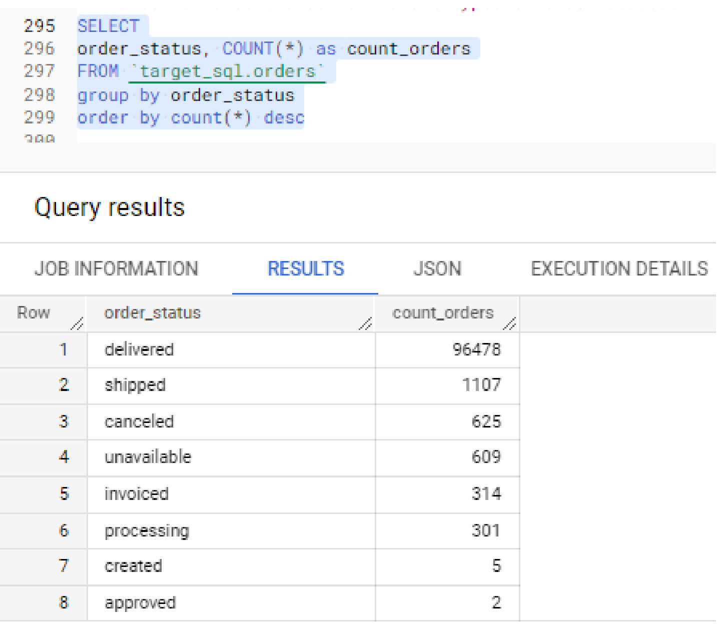
```
SELECT COUNT(DISTINCT(order_id)) as no_of_orders
FROM `target_sql.orders`
```

Row	no_of_orders
1	99441

OBSERVATIONS: -

- All the records under order table are unique.

Number of orders under different types of orders status: -



```
SELECT
order_status, COUNT(*) as count_orders
FROM `target_sql.orders`
group by order_status
order by count(*) desc
```

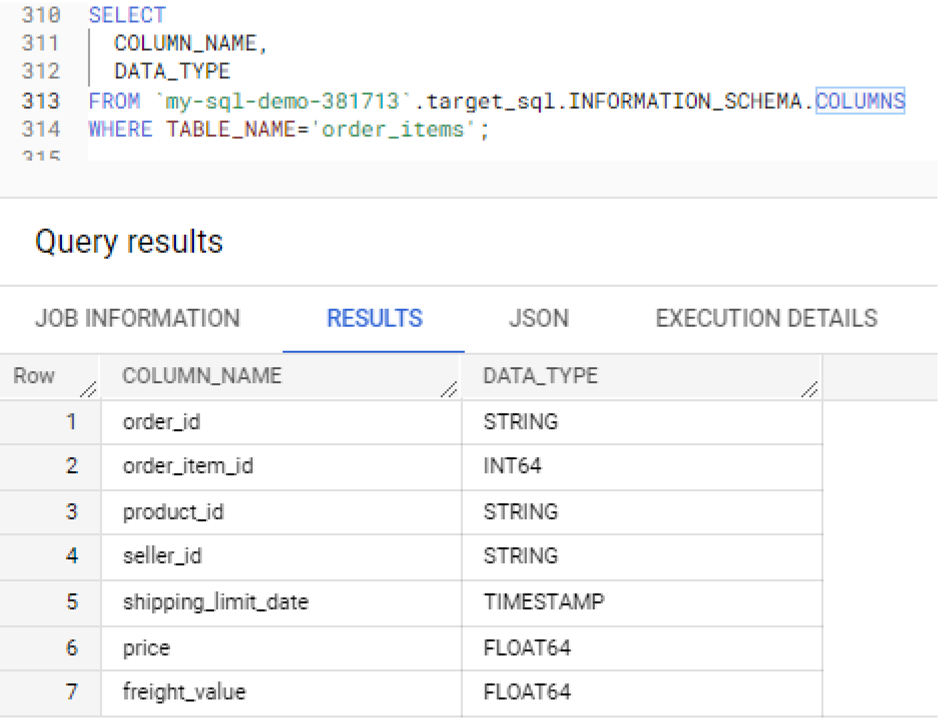
Row	order_status	count_orders
1	delivered	96478
2	shipped	1107
3	canceled	625
4	unavailable	609
5	invoiced	314
6	processing	301
7	created	5
8	approved	2

OBSERVATIONS: -

- Most of the orders have been delivered, and of the ones remaining, almost half have been shipped.
- Approximately 1200 orders have been cancelled or are unavailable.

1.1.6 Order items table:

Columns under the order items table:-



```sql
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM `my-sql-demo-381713`.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='order_items';
```

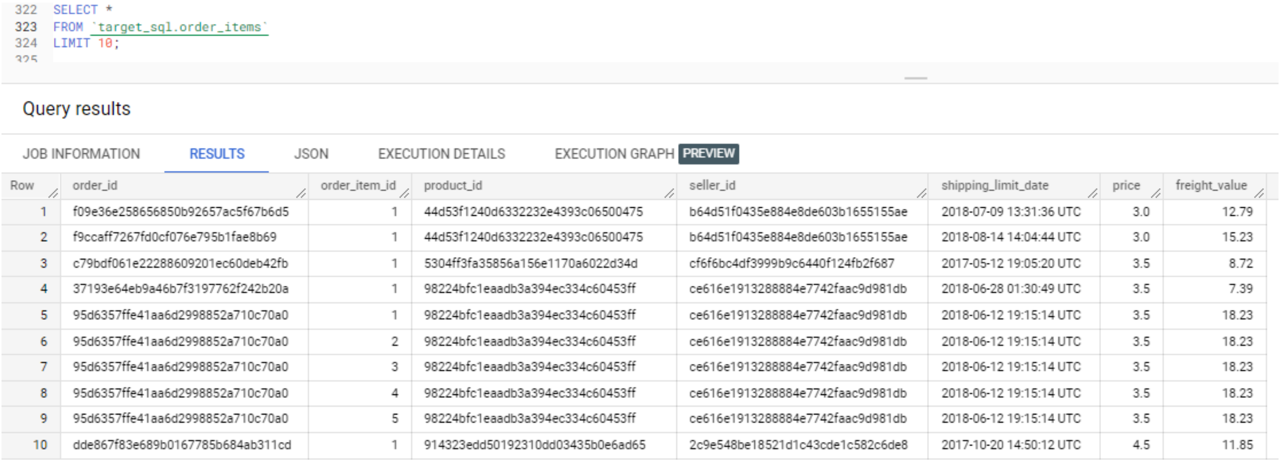
Query results

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

OBSERVATIONS:

- Shipping_limit_date is timestamp type.

First 10 rows: -

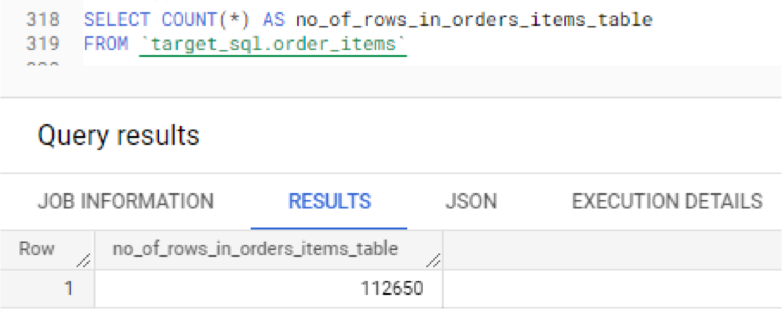


```sql
SELECT *
FROM `target_sql.order_items`
LIMIT 10;
```

Query results

Row	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
1	f09e36e258656850b92657ac5f67b6d5	1	44d53f1240d6332232e4393c06500475	b64d51f0435e884e8de603b1655155ae	2018-07-09 13:31:36 UTC	3.0	12.79
2	f9ccaff7267fd0cf076e795b1fae8b69	1	44d53f1240d6332232e4393c06500475	b64d51f0435e884e8de603b1655155ae	2018-08-14 14:04:44 UTC	3.0	15.23
3	c79bdf061e22288609201ec60deb42fb	1	5304ff3fa35856a156e1170a6022d34d	cf6f6bc4df3999b9c6440f124fb2f687	2017-05-12 19:05:20 UTC	3.5	8.72
4	37193e64eb9a46b7f3197762f242b20a	1	98224bfc1eaadb3a394ec334c60453ff	ce616e1913288884e7742faac9d981db	2018-06-28 01:30:49 UTC	3.5	7.39
5	95d6357ffe41aa6d2998852a710c70a0	1	98224bfc1eaadb3a394ec334c60453ff	ce616e1913288884e7742faac9d981db	2018-06-12 19:15:14 UTC	3.5	18.23
6	95d6357ffe41aa6d2998852a710c70a0	2	98224bfc1eaadb3a394ec334c60453ff	ce616e1913288884e7742faac9d981db	2018-06-12 19:15:14 UTC	3.5	18.23
7	95d6357ffe41aa6d2998852a710c70a0	3	98224bfc1eaadb3a394ec334c60453ff	ce616e1913288884e7742faac9d981db	2018-06-12 19:15:14 UTC	3.5	18.23
8	95d6357ffe41aa6d2998852a710c70a0	4	98224bfc1eaadb3a394ec334c60453ff	ce616e1913288884e7742faac9d981db	2018-06-12 19:15:14 UTC	3.5	18.23
9	95d6357ffe41aa6d2998852a710c70a0	5	98224bfc1eaadb3a394ec334c60453ff	ce616e1913288884e7742faac9d981db	2018-06-12 19:15:14 UTC	3.5	18.23
10	dde867f83e689b0167785b684ab311cd	1	914323edd50192310dd03435b0e6ad65	2c9e548be18521d1c43cde1c582c6de8	2017-10-20 14:50:12 UTC	4.5	11.85

Number of rows: -

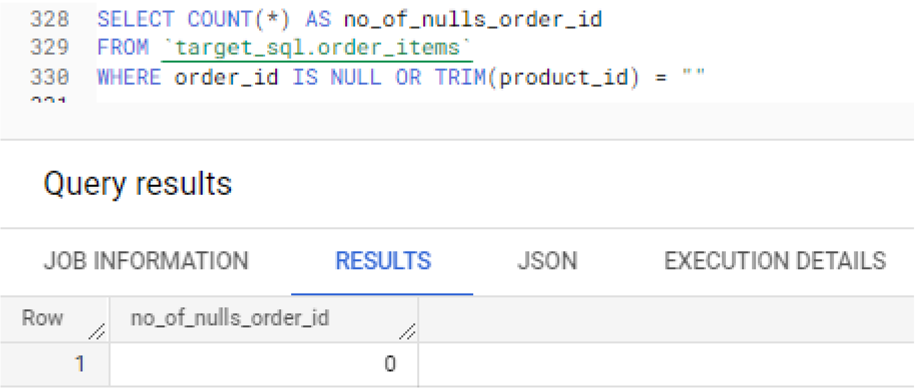


```sql
SELECT COUNT(*) AS no_of_rows_in_orders_items_table
FROM `target_sql.order_items`
```

Query results

Row	no_of_rows_in_orders_items_table
1	112650

No of NULL values: -

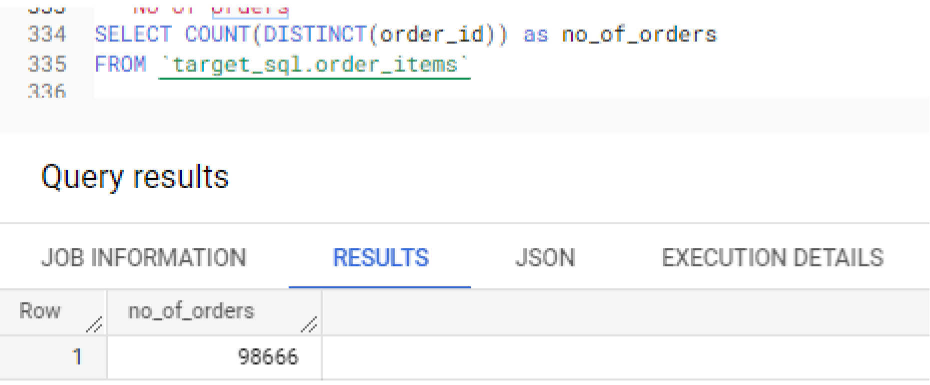


```
SELECT COUNT(*) AS no_of_nulls_order_id
FROM `target_sql.order_items`
WHERE order_id IS NULL OR TRIM(product_id) = ""
```

Query results

Row	no_of_nulls_order_id
1	0

No. of orders: -



```
SELECT COUNT(DISTINCT(order_id)) as no_of_orders
FROM `target_sql.order_items`
```

Query results

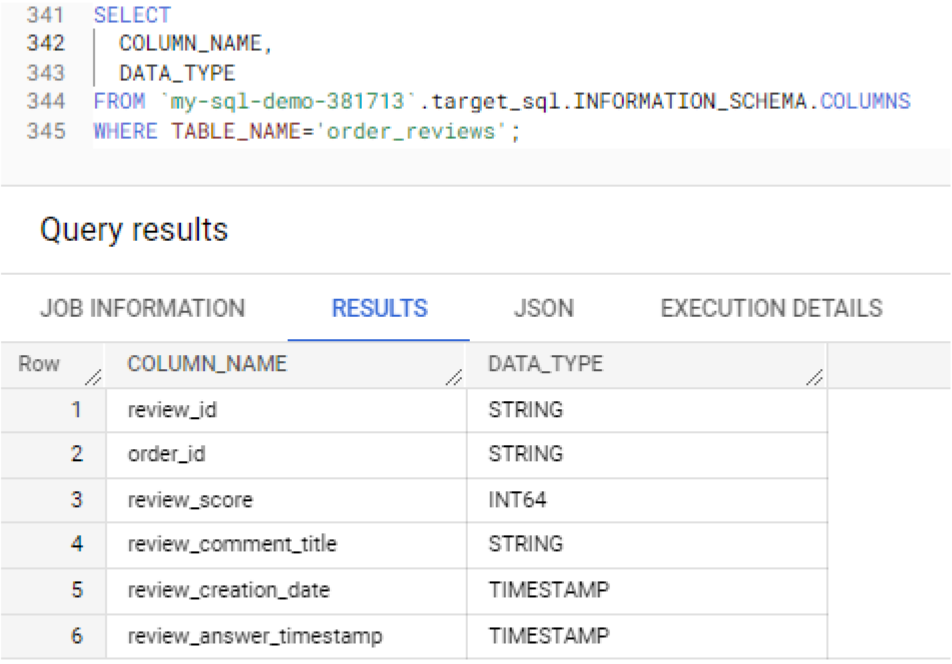
Row	no_of_orders
1	98666

OBSERVATIONS:-

- 98666 are unique orders

1.1.7 Order reviews table:

Columns under the order reviews table:-



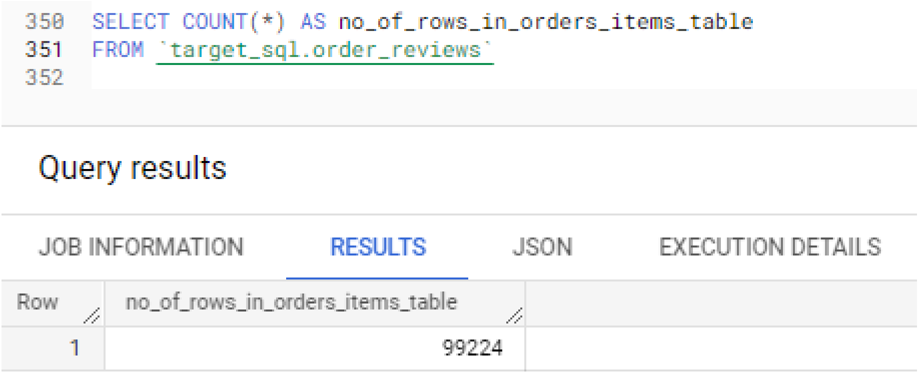
```sql
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM `my-sql-demo-381713`.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='order_reviews';
```

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

OBSERVATIONS:

- Review_creation_date and review_answer_timestamp are of timestamp type.

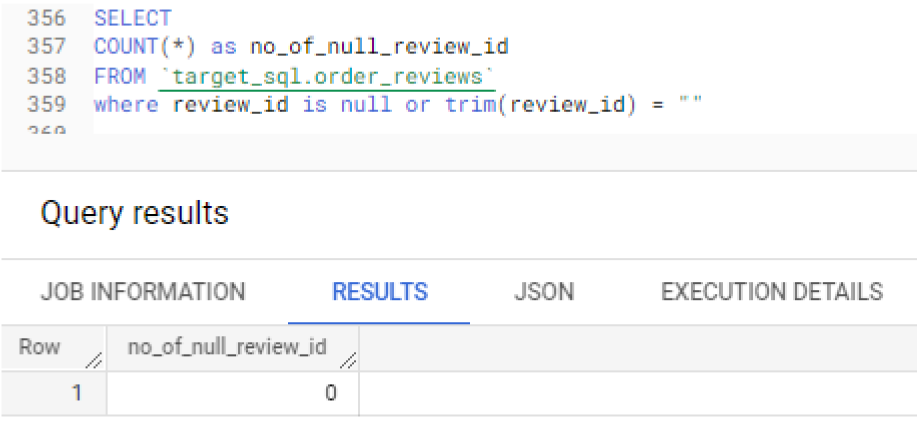
Number of rows: -



```sql
SELECT COUNT(*) AS no_of_rows_in_orders_items_table
FROM `target_sql.order_reviews`
```

Row	no_of_rows_in_orders_items_table
1	99224

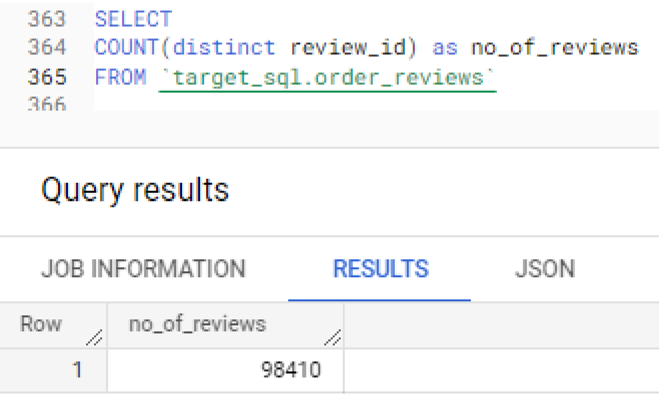
Number of NULL values in review_id: -



```sql
SELECT
COUNT(*) as no_of_null_review_id
FROM `target_sql.order_reviews`
where review_id is null or trim(review_id) = ""
```

Row	no_of_null_review_id
1	0

Number of reviews:-



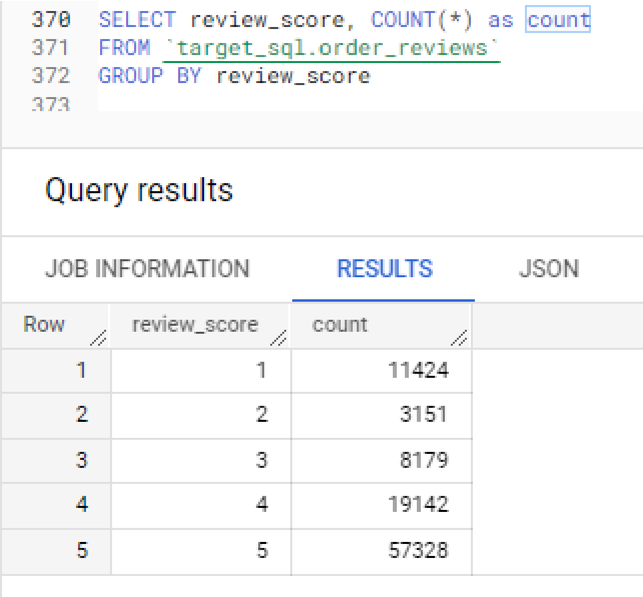
```
SELECT
COUNT(distinct review_id) as no_of_reviews
FROM `target_sql.order_reviews`
```

Row	no_of_reviews
1	98410

OBSERVATIONS: -

- There are 98410 unique reviews out of 99224 rows

Number of reviews for each review score: -



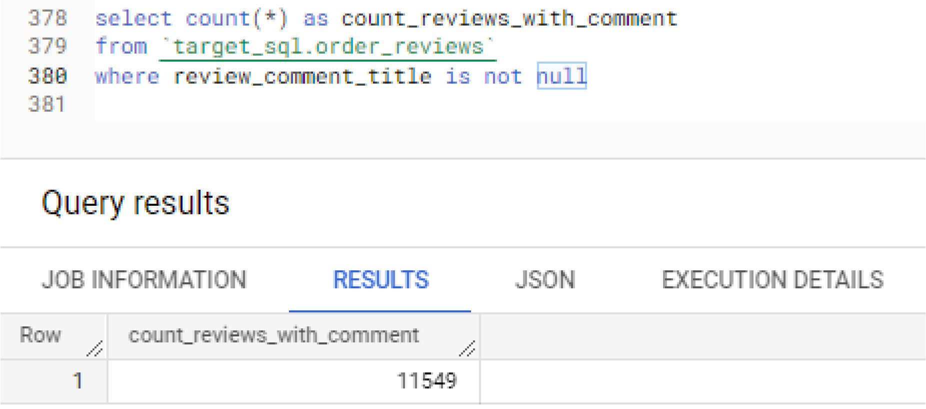
```
SELECT review_score, COUNT(*) as count
FROM `target_sql.order_reviews`
GROUP BY review_score
```

Row	review_score	count
1	1	11424
2	2	3151
3	3	8179
4	4	19142
5	5	57328

OBSERVATIONS: -

- Majority of orders received a review score of 5, followed by 4, 1, 3 and 2

Count of reviews with comments: -



```
select count(*) as count_reviews_with_comment
from `target_sql.order_reviews`
where review_comment_title is not null
```

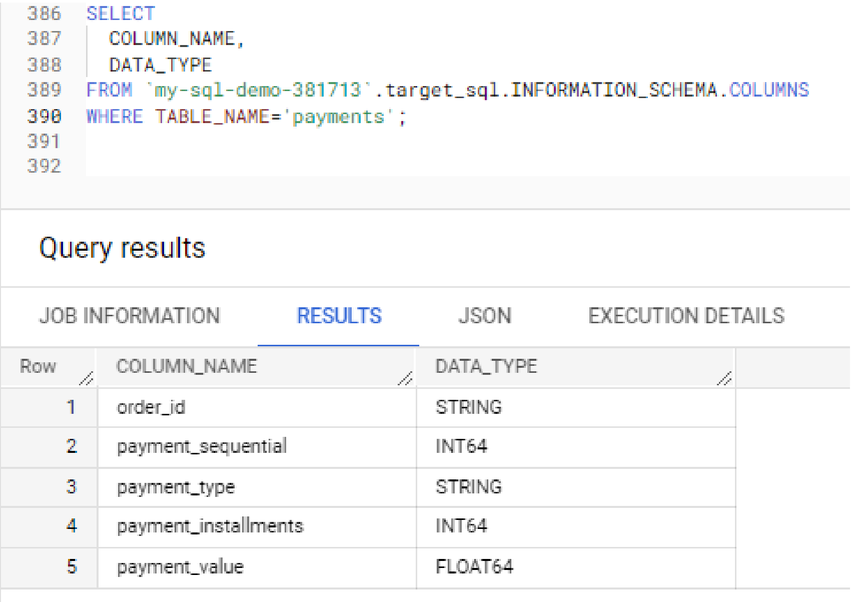
Row	count_reviews_with_comment
1	11549

OBSERVATIONS: -

- Approximately 11% of the total reviews had review_title

1.1.8 **Payments table:**

Columns under the payments table: -

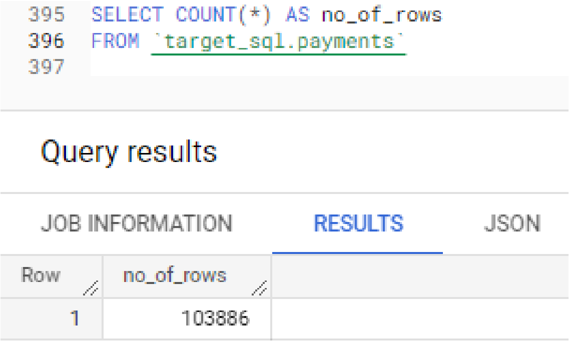


```
SELECT
  COLUMN_NAME,
  DATA_TYPE
FROM `my-sql-demo-381713`.target_sql.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='payments';
```

Query results

Row	COLUMN_NAME	DATA_TYPE
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

Number of rows: -

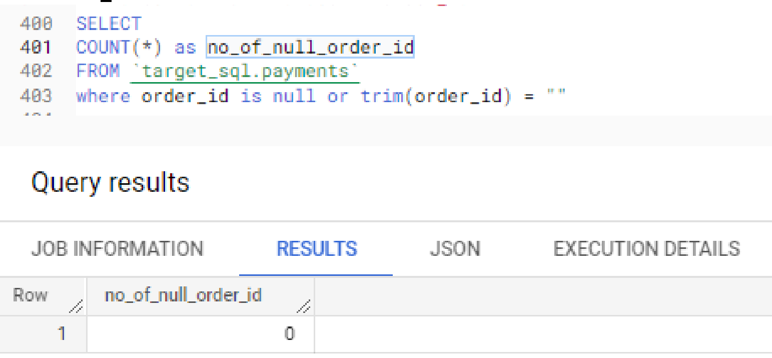


```
SELECT COUNT(*) AS no_of_rows
FROM `target_sql.payments`
```

Query results

Row	no_of_rows
1	103886

Number of NULL values in order_id:-

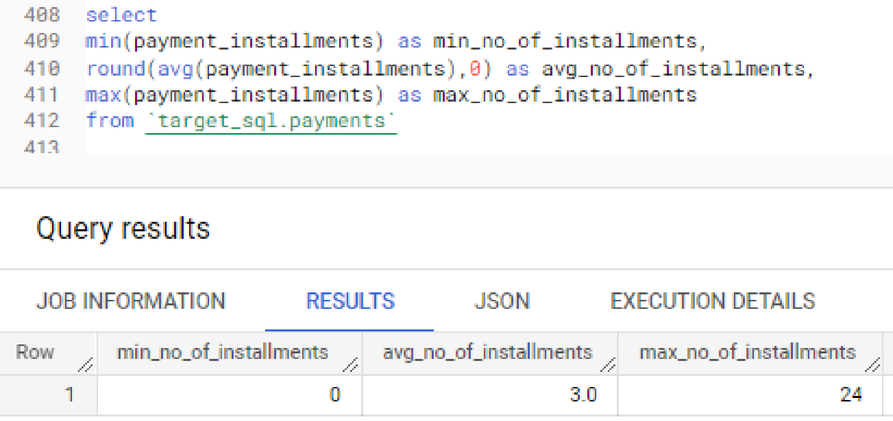


```
SELECT
COUNT(*) as no_of_null_order_id
FROM `target_sql.payments`
where order_id is null or trim(order_id) = ""
```

Query results

Row	no_of_null_order_id
1	0

Number of instalments: -

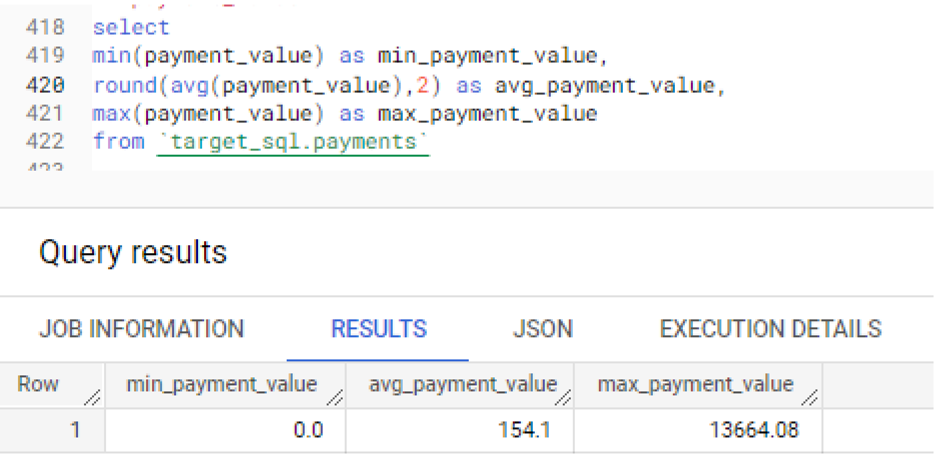


```
select
min(payment_installments) as min_no_of_installments,
round(avg(payment_installments),0) as avg_no_of_installments,
max(payment_installments) as max_no_of_installments
from `target_sql.payments`
```

Query results

Row	min_no_of_installments	avg_no_of_installments	max_no_of_installments
1	0	3.0	24

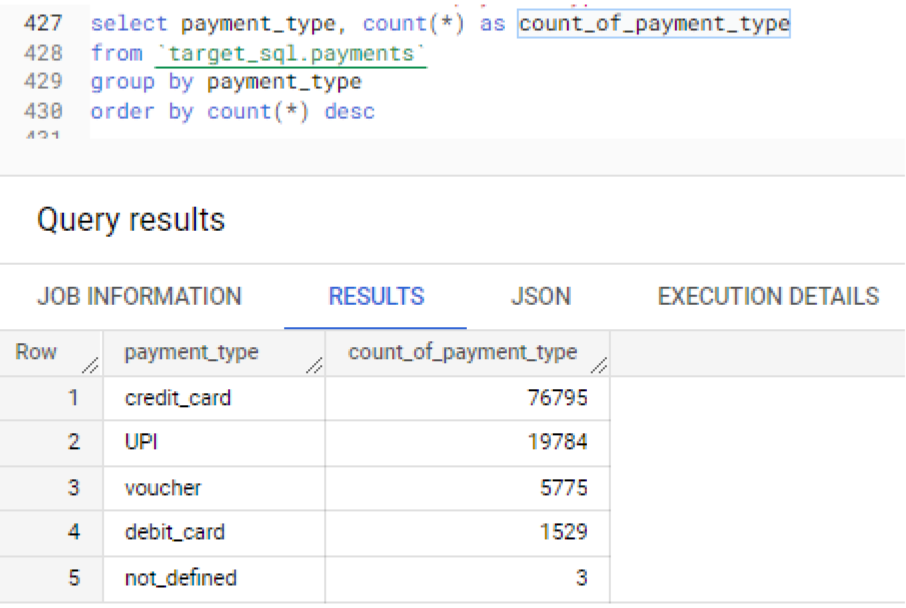
Payment value: -



```
select
min(payment_value) as min_payment_value,
round(avg(payment_value),2) as avg_payment_value,
max(payment_value) as max_payment_value
from `target_sql.payments`
```

Row	min_payment_value	avg_payment_value	max_payment_value
1	0.0	154.1	13664.08

Number of orders for each payment type:-

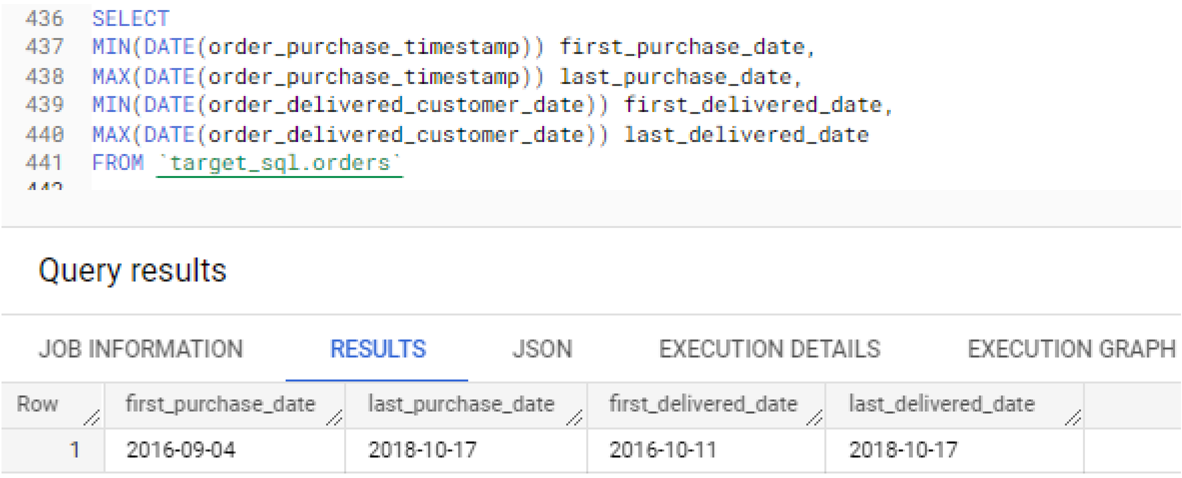


```
select payment_type, count(*) as count_of_payment_type
from `target_sql.payments`
group by payment_type
order by count(*) desc
```

Row	payment_type	count_of_payment_type
1	credit_card	76795
2	UPI	19784
3	voucher	5775
4	debit_card	1529
5	not_defined	3

1.2 Time period for which the data is given:

Time period for which the data is given: -

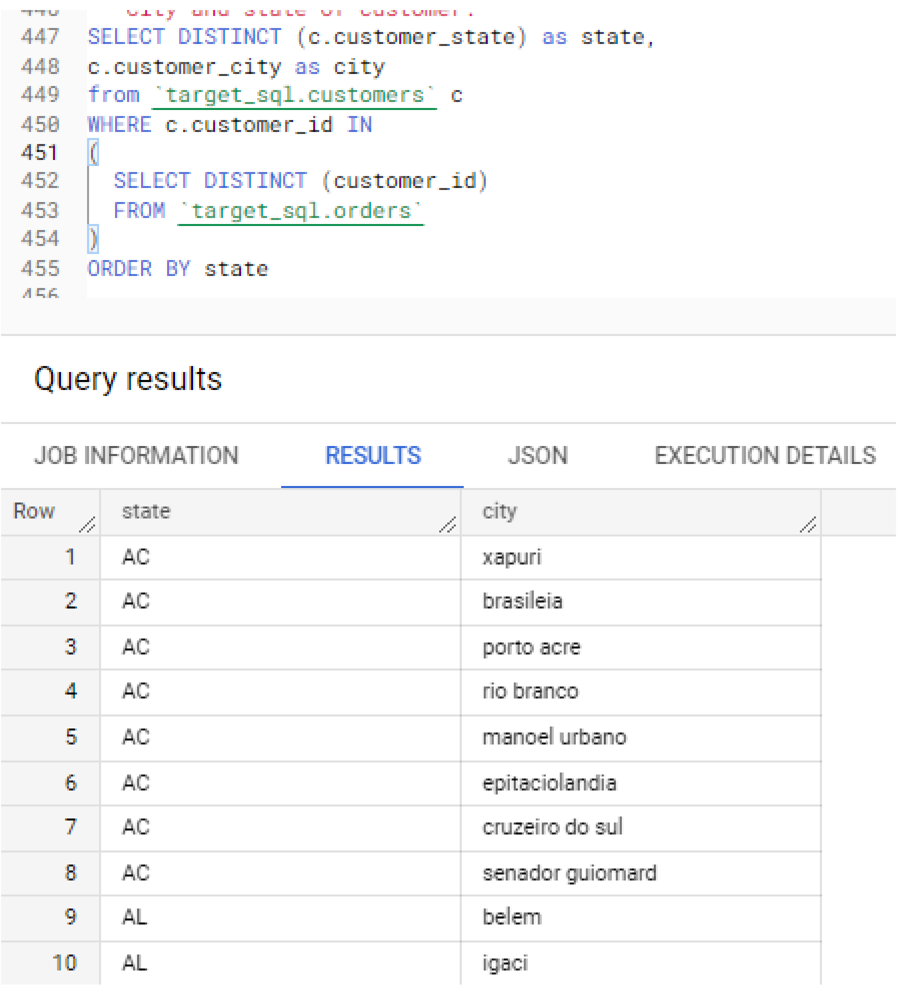


```sql
SELECT
MIN(DATE(order_purchase_timestamp)) first_purchase_date,
MAX(DATE(order_purchase_timestamp)) last_purchase_date,
MIN(DATE(order_delivered_customer_date)) first_delivered_date,
MAX(DATE(order_delivered_customer_date)) last_delivered_date
FROM `target_sql.orders`
```

Row	first_purchase_date	last_purchase_date	first_delivered_date	last_delivered_date
1	2016-09-04	2018-10-17	2016-10-11	2018-10-17

1.3 Cities and States of customers ordered during the given period:

Cities and States of customers ordered during the given period: -



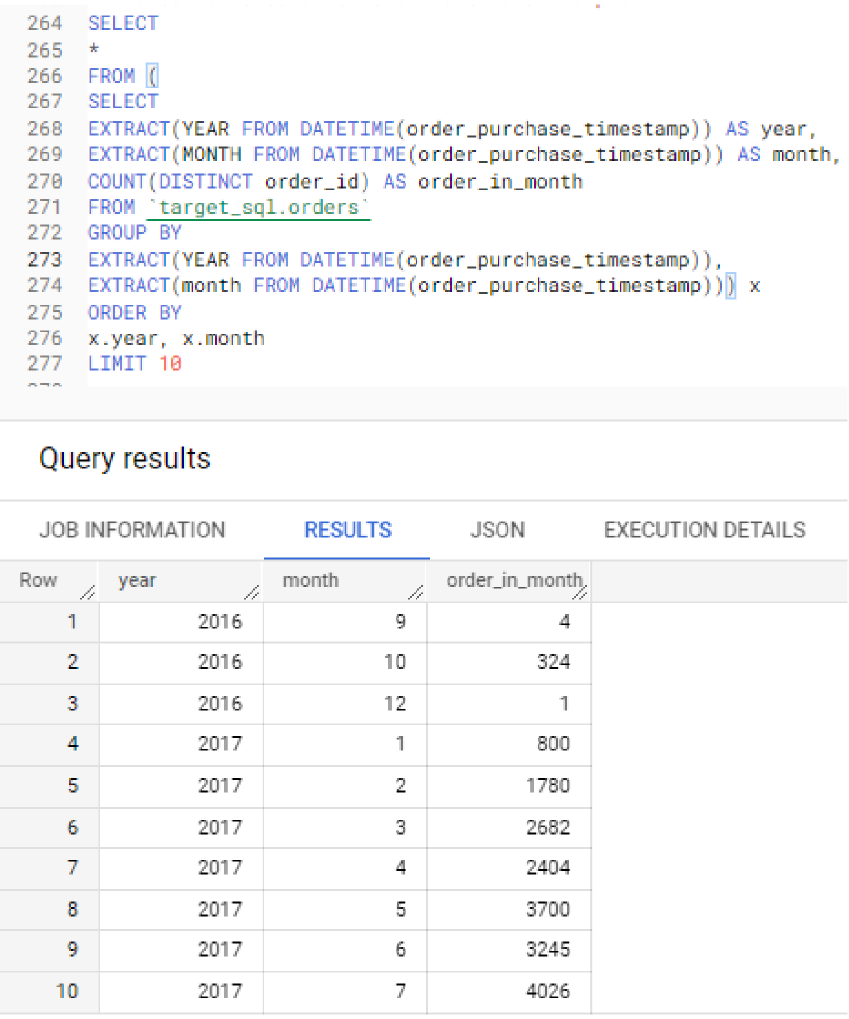
```sql
  city and state of customer:
SELECT DISTINCT (c.customer_state) as state,
c.customer_city as city
from `target_sql.customers` c
WHERE c.customer_id IN
(
  SELECT DISTINCT (customer_id)
  FROM `target_sql.orders`
)
ORDER BY state
```

Row	state	city
1	AC	xapuri
2	AC	brasileia
3	AC	porto acre
4	AC	rio branco
5	AC	manoel urbano
6	AC	epitaciolandia
7	AC	cruzeiro do sul
8	AC	senador guiomard
9	AL	belem
10	AL	igaci

2. In Depth Exploration

2.1 Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

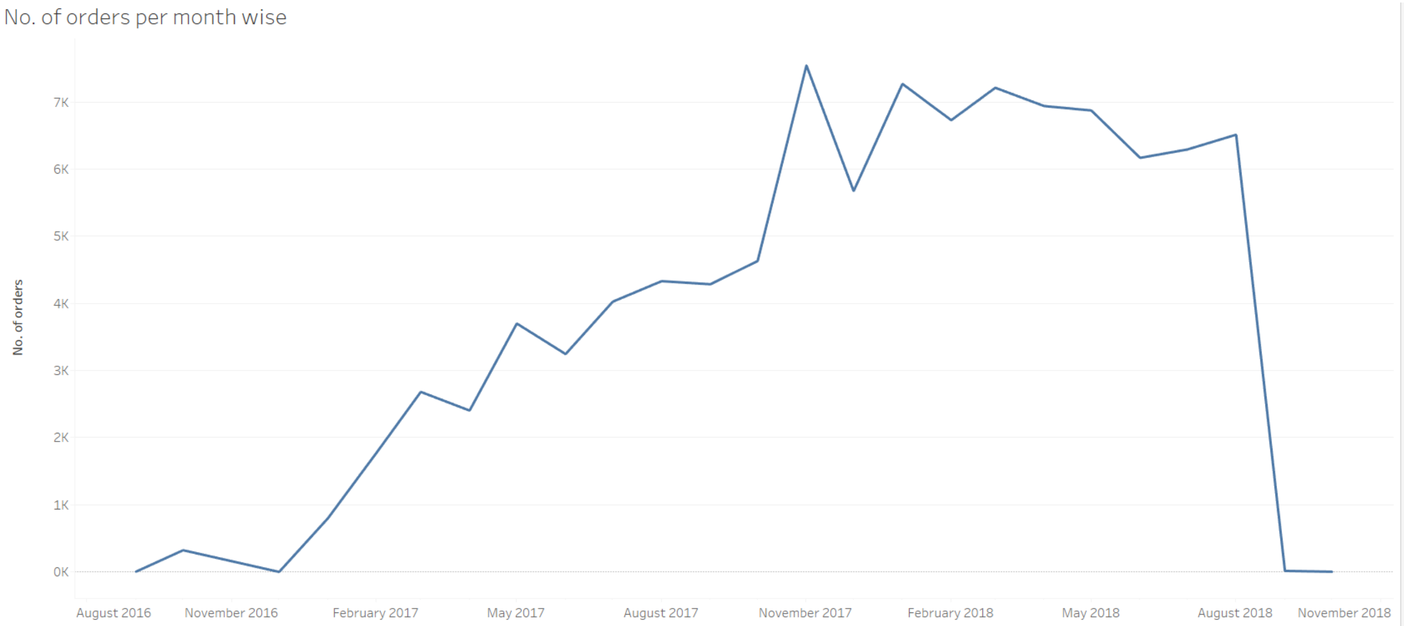
Number of orders for each month: -



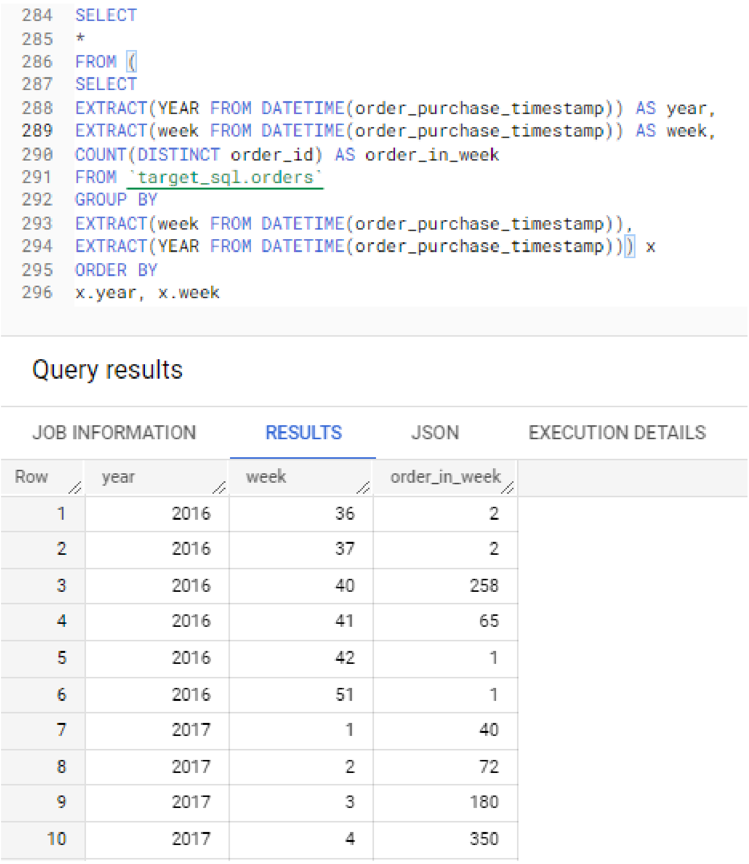
```
SELECT
*
FROM (
SELECT
EXTRACT(YEAR FROM DATETIME(order_purchase_timestamp)) AS year,
EXTRACT(MONTH FROM DATETIME(order_purchase_timestamp)) AS month,
COUNT(DISTINCT order_id) AS order_in_month
FROM `target_sql.orders`
GROUP BY
EXTRACT(YEAR FROM DATETIME(order_purchase_timestamp)),
EXTRACT(month FROM DATETIME(order_purchase_timestamp))) x
ORDER BY
x.year, x.month
LIMIT 10
```

Query results

Row	year	month	order_in_month
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026



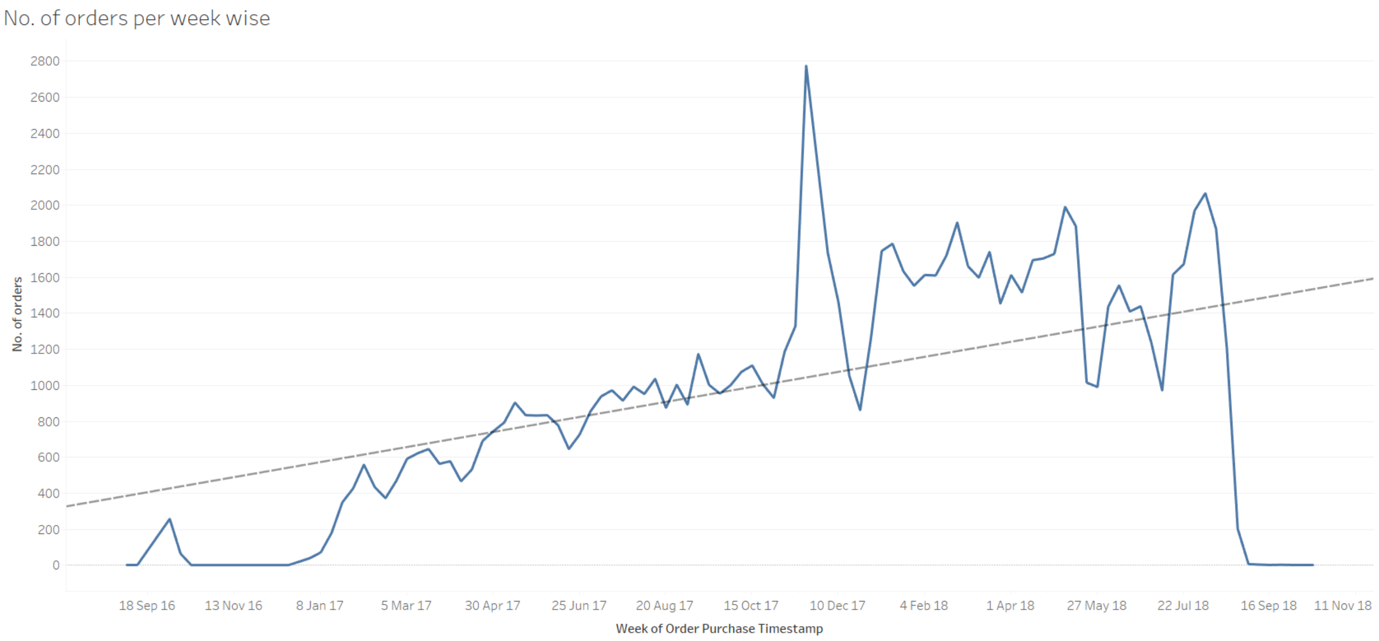
Week wise number of orders: -



```
SELECT
*
FROM (
SELECT
EXTRACT(YEAR FROM DATETIME(order_purchase_timestamp)) AS year,
EXTRACT(week FROM DATETIME(order_purchase_timestamp)) AS week,
COUNT(DISTINCT order_id) AS order_in_week
FROM `target_sql.orders`
GROUP BY
EXTRACT(week FROM DATETIME(order_purchase_timestamp)),
EXTRACT(YEAR FROM DATETIME(order_purchase_timestamp))) x
ORDER BY
x.year, x.week
```

Query results

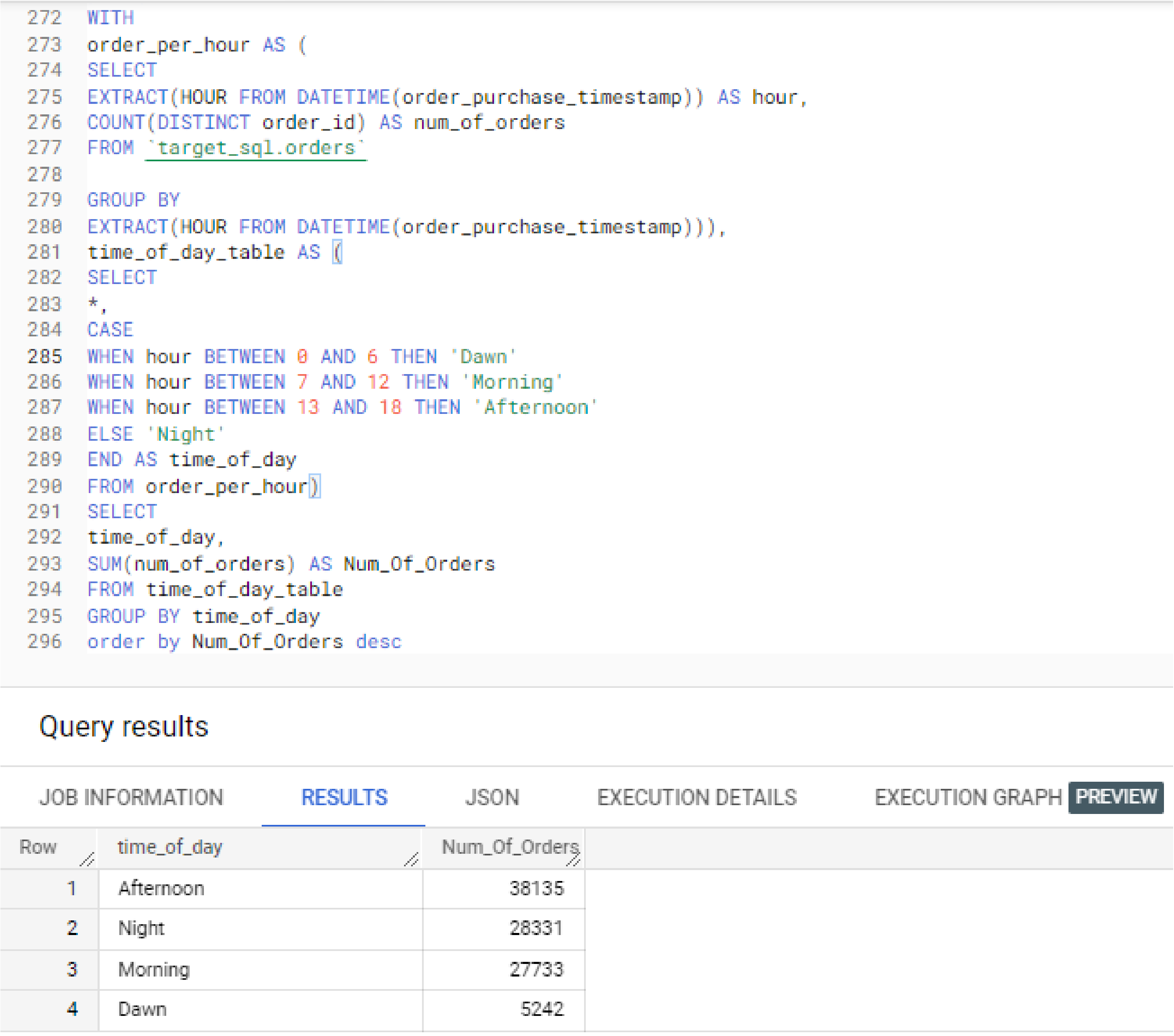
Row	year	week	order_in_week
1	2016	36	2
2	2016	37	2
3	2016	40	258
4	2016	41	65
5	2016	42	1
6	2016	51	1
7	2017	1	40
8	2017	2	72
9	2017	3	180
10	2017	4	350



OBSERVATIONS: -

- There is a rise in the number of customer orders on a weekly basis, as evidenced by the upward trend line depicted in the chart above.
- Although the chart displays some notable spikes and drops, these may be attributed to seasonal factors at certain points in time, and hence can be disregarded.
- Since there is a dearth of data available after mid-August, a sudden decline towards the end of the chart is observed, but it can be discounted.

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



```
WITH
order_per_hour AS (
SELECT
EXTRACT(HOUR FROM DATETIME(order_purchase_timestamp)) AS hour,
COUNT(DISTINCT order_id) AS num_of_orders
FROM `target_sql.orders`

GROUP BY
EXTRACT(HOUR FROM DATETIME(order_purchase_timestamp))),
time_of_day_table AS (
SELECT
*,
CASE
WHEN hour BETWEEN 0 AND 6 THEN 'Dawn'
WHEN hour BETWEEN 7 AND 12 THEN 'Morning'
WHEN hour BETWEEN 13 AND 18 THEN 'Afternoon'
ELSE 'Night'
END AS time_of_day
FROM order_per_hour)
SELECT
time_of_day,
SUM(num_of_orders) AS Num_Of_Orders
FROM time_of_day_table
GROUP BY time_of_day
order by Num_Of_Orders desc
```

Query results

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

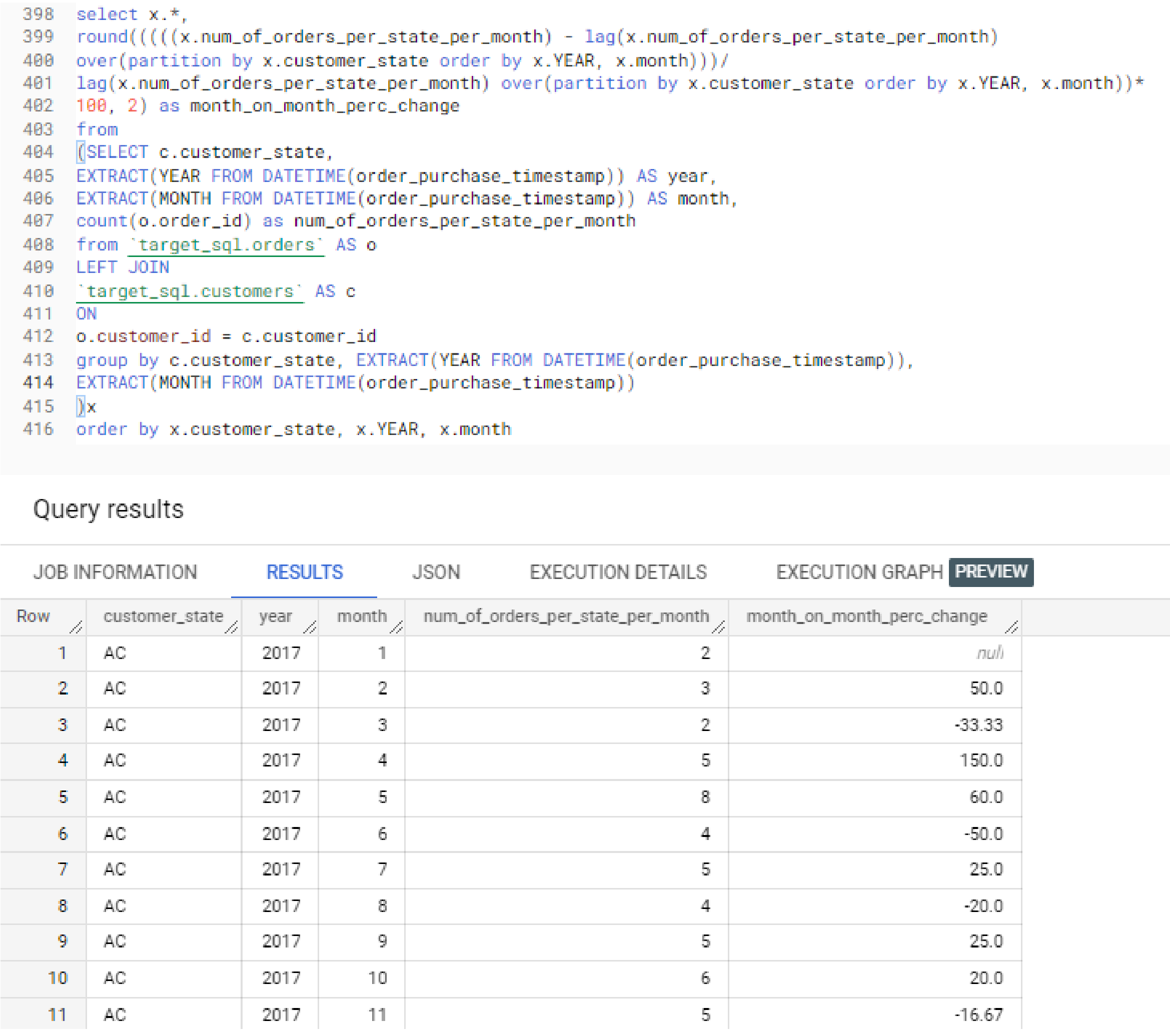
OBSERVATIONS

- The most orders (38135) are placed during the afternoon period (1 PM to 8 PM).
- The fewest orders (5242) are placed in the early morning hours (12 AM to 6 AM).
- The morning period (7 AM to 12 PM) and the night period (7 PM to 11 PM) have almost equal numbers of orders (27733 and 28331, respectively).

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

3.1 Get Month on month orders by states

Month on month orders by states: -

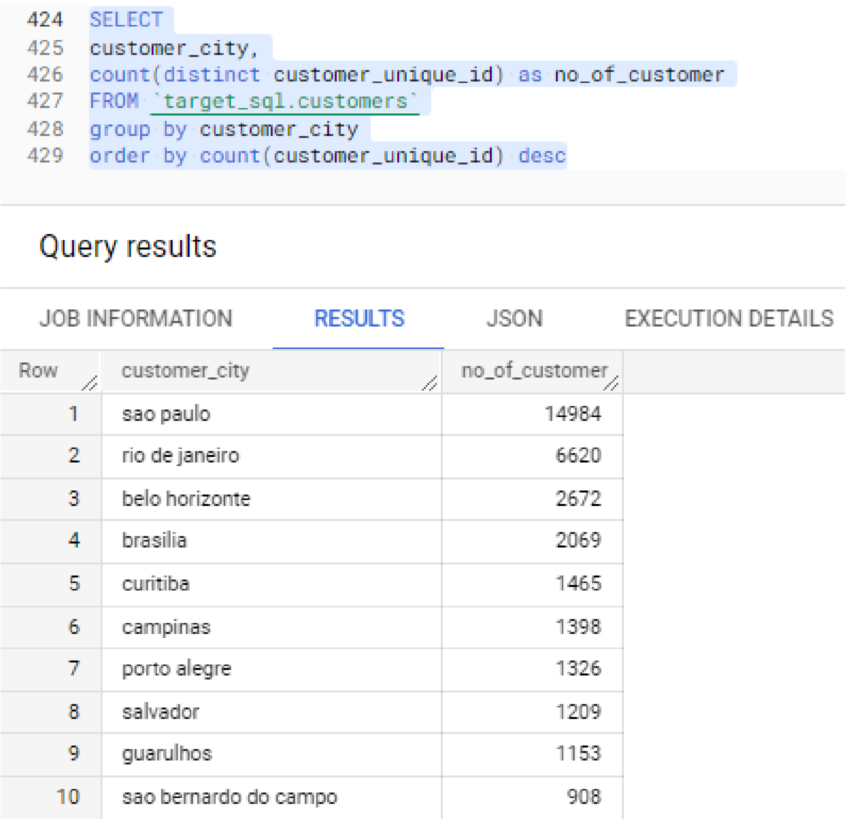


```
select x.*,
round(((((x.num_of_orders_per_state_per_month) - lag(x.num_of_orders_per_state_per_month)
over(partition by x.customer_state order by x.YEAR, x.month)))/
lag(x.num_of_orders_per_state_per_month) over(partition by x.customer_state order by x.YEAR, x.month))*
100, 2) as month_on_month_perc_change
from
(SELECT c.customer_state,
EXTRACT(YEAR FROM DATETIME(order_purchase_timestamp)) AS year,
EXTRACT(MONTH FROM DATETIME(order_purchase_timestamp)) AS month,
count(o.order_id) as num_of_orders_per_state_per_month
from `target_sql.orders` AS o
LEFT JOIN
`target_sql.customers` AS c
ON
o.customer_id = c.customer_id
group by c.customer_state, EXTRACT(YEAR FROM DATETIME(order_purchase_timestamp)),
EXTRACT(MONTH FROM DATETIME(order_purchase_timestamp))
)x
order by x.customer_state, x.YEAR, x.month
```

Row	customer_state	year	month	num_of_orders_per_state_per_month	month_on_month_perc_change
1	AC	2017	1	2	null
2	AC	2017	2	3	50.0
3	AC	2017	3	2	-33.33
4	AC	2017	4	5	150.0
5	AC	2017	5	8	60.0
6	AC	2017	6	4	-50.0
7	AC	2017	7	5	25.0
8	AC	2017	8	4	-20.0
9	AC	2017	9	5	25.0
10	AC	2017	10	6	20.0
11	AC	2017	11	5	-16.67

3.2 Distribution of customers across the states in Brazil

Distribution of customers across the states in Brazil: -

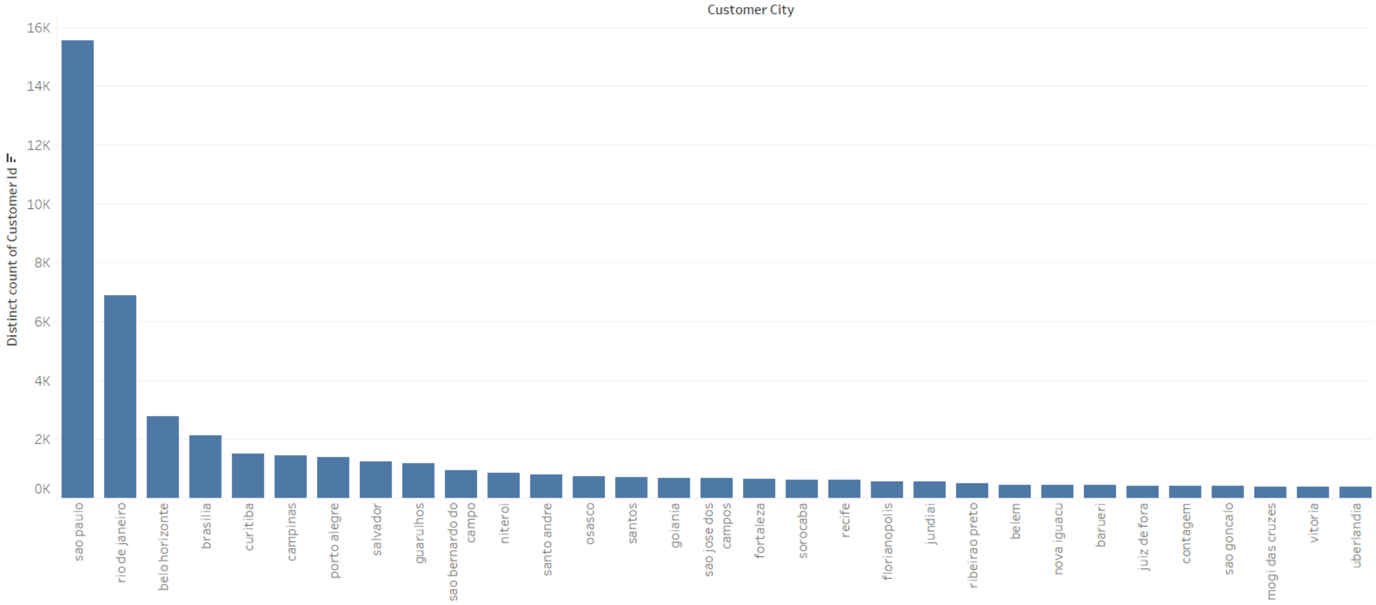


```
SELECT
customer_city,
count(distinct customer_unique_id) as no_of_customer
FROM `target_sql.customers`
group by customer_city
order by count(customer_unique_id) desc
```

Query results

Row	customer_city	no_of_customer
1	sao paulo	14984
2	rio de janeiro	6620
3	belo horizonte	2672
4	brasilia	2069
5	curitiba	1465
6	campinas	1398
7	porto alegre	1326
8	salvador	1209
9	guarulhos	1153
10	sao bernardo do campo	908

Distribution of customers in Brazil



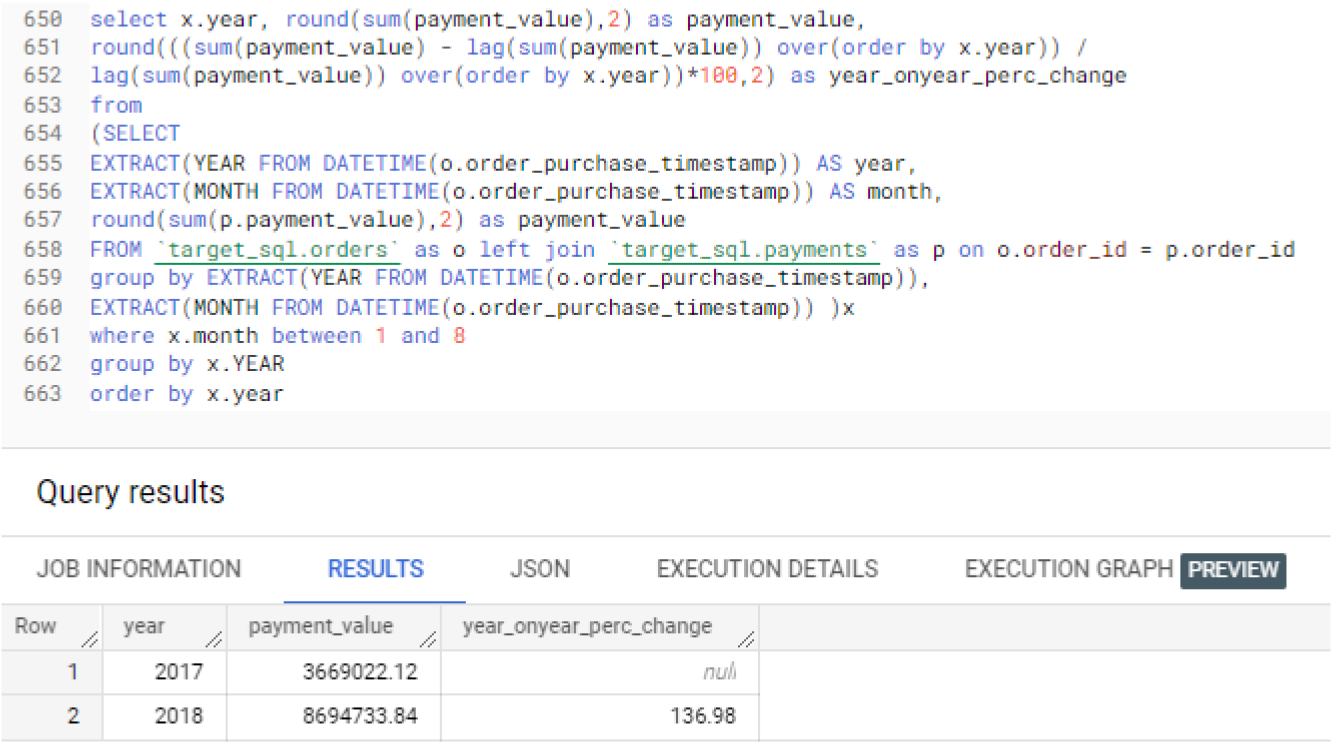
OBSERVATIONS: -

- The majority of the clientele is concentrated in densely populated cities like Sao Paulo and Rio de Janeiro.
- Sao Paulo boasts the highest number of customers (14894), while Rio de Janeiro comes in second with just over 40% of Sao Paulo's total (6620).
- There is a significant decline in the customer base as we move below the 9th state.
- States with lower rankings have minimal customer bases, sometimes as low as just one customer.

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

4.1 Get percentage increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)

Get percentage increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only): -



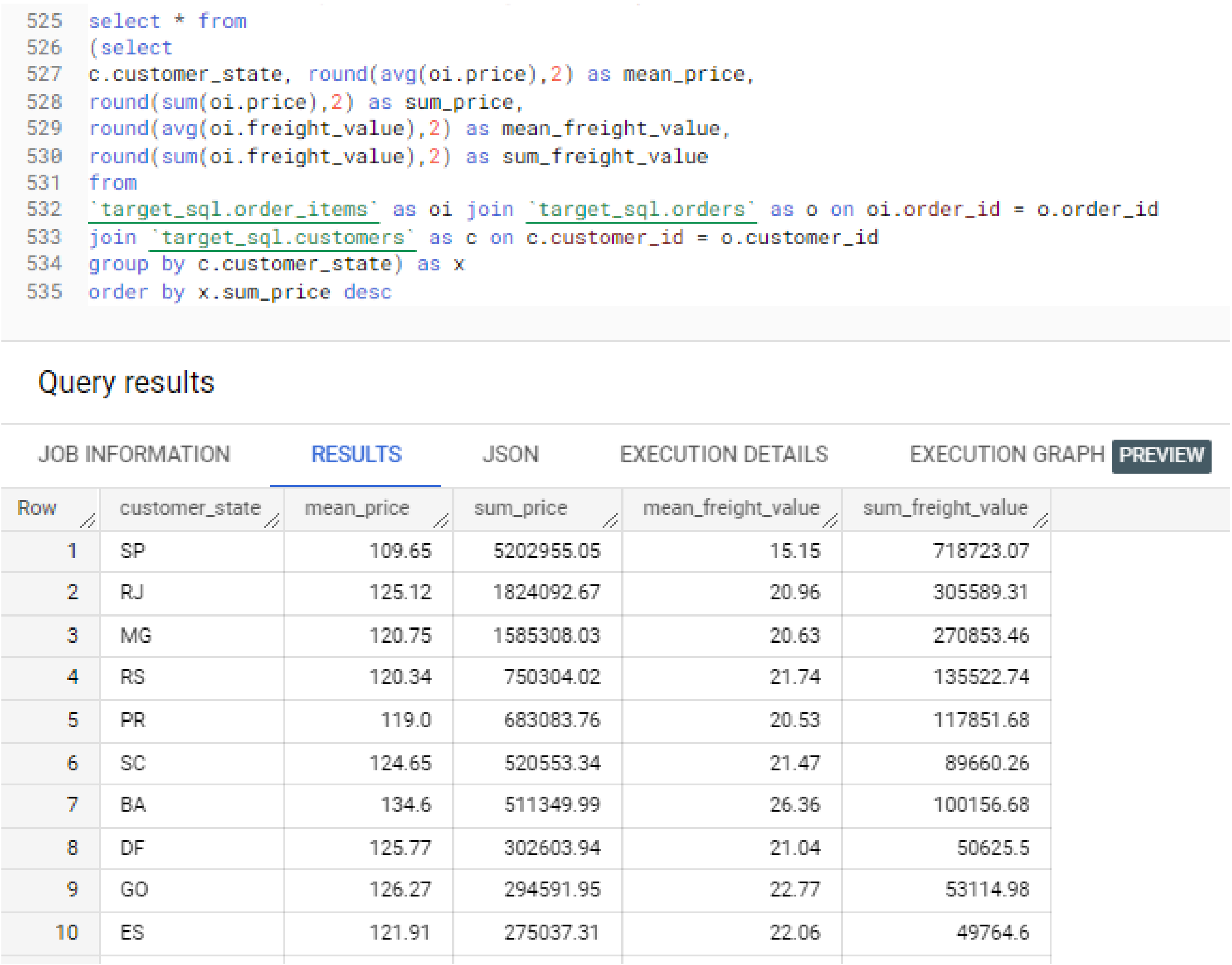
```sql
select x.year, round(sum(payment_value),2) as payment_value,
round(((sum(payment_value) - lag(sum(payment_value)) over(order by x.year)) /
lag(sum(payment_value)) over(order by x.year))*100,2) as year_onyear_perc_change
from
(SELECT
EXTRACT(YEAR FROM DATETIME(o.order_purchase_timestamp)) AS year,
EXTRACT(MONTH FROM DATETIME(o.order_purchase_timestamp)) AS month,
round(sum(p.payment_value),2) as payment_value
FROM `target_sql.orders` as o left join `target_sql.payments` as p on o.order_id = p.order_id
group by EXTRACT(YEAR FROM DATETIME(o.order_purchase_timestamp)),
EXTRACT(MONTH FROM DATETIME(o.order_purchase_timestamp)) )x
where x.month between 1 and 8
group by x.YEAR
order by x.year
```

Query results

Row	year	payment_value	year_onyear_perc_change
1	2017	3669022.12	null
2	2018	8694733.84	136.98

4.2 Mean and Sum of prices and freight value by customer state

Mean and Sum of prices and freight value by customer state: -



```
select * from
(select
c.customer_state, round(avg(oi.price),2) as mean_price,
round(sum(oi.price),2) as sum_price,
round(avg(oi.freight_value),2) as mean_freight_value,
round(sum(oi.freight_value),2) as sum_freight_value
from
`target_sql.order_items` as oi join `target_sql.orders` as o on oi.order_id = o.order_id
join `target_sql.customers` as c on c.customer_id = o.customer_id
group by c.customer_state) as x
order by x.sum_price desc
```

Query results

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

OBSERVATIONS: -

- Customers hailing from SP have placed orders for the largest number of products in terms of their total price, followed by RJ.
- The customers from PE have ordered products with the highest average price.

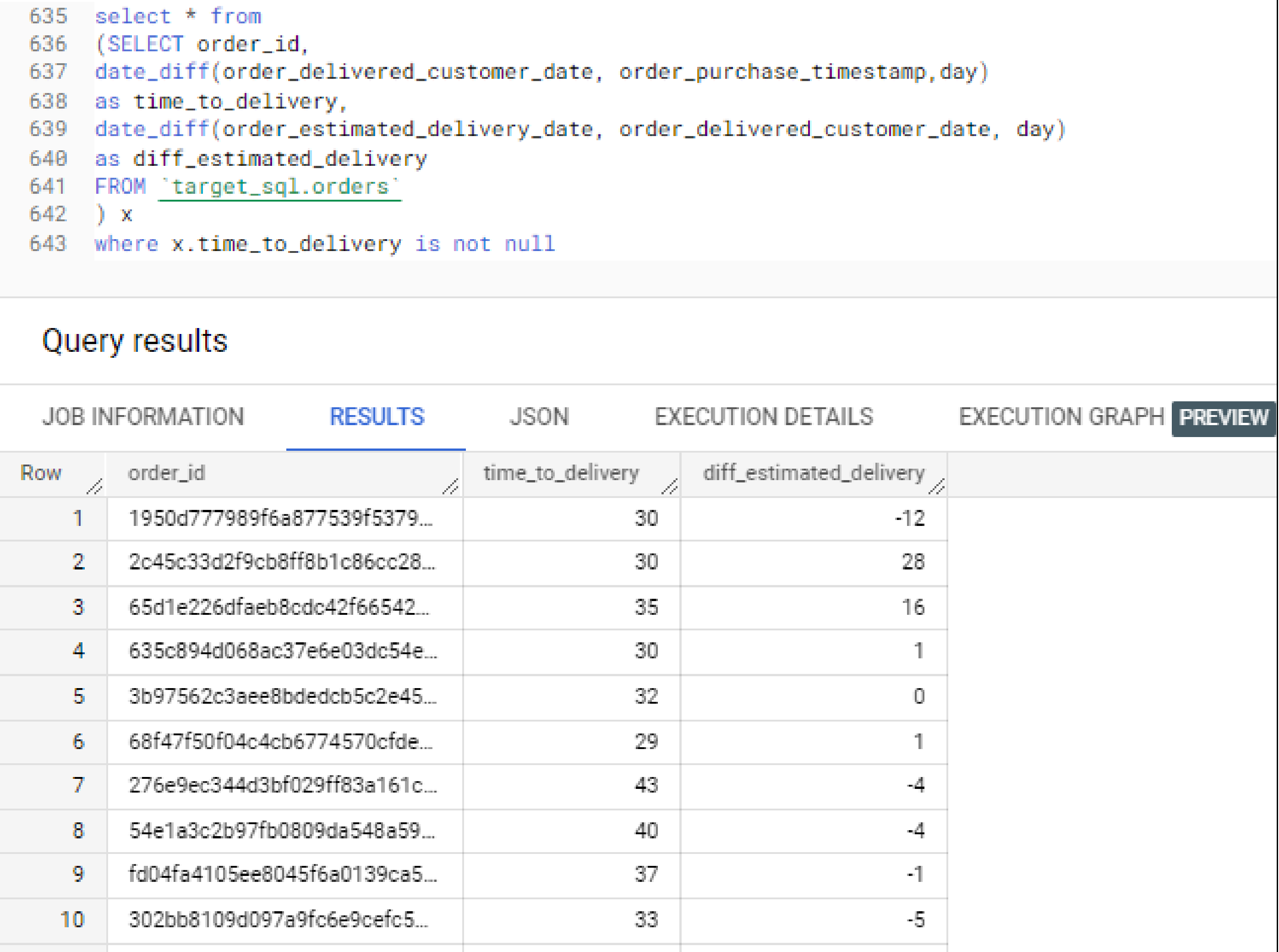
5. Analysis on Sales, Freight and Delivery Time

5.1 Calculate days between purchasing, delivering and estimated delivery and create columns time_to_delivery and diff_estimated_delivery

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

Find time_to_delivery and diff_estimated_delivery: -



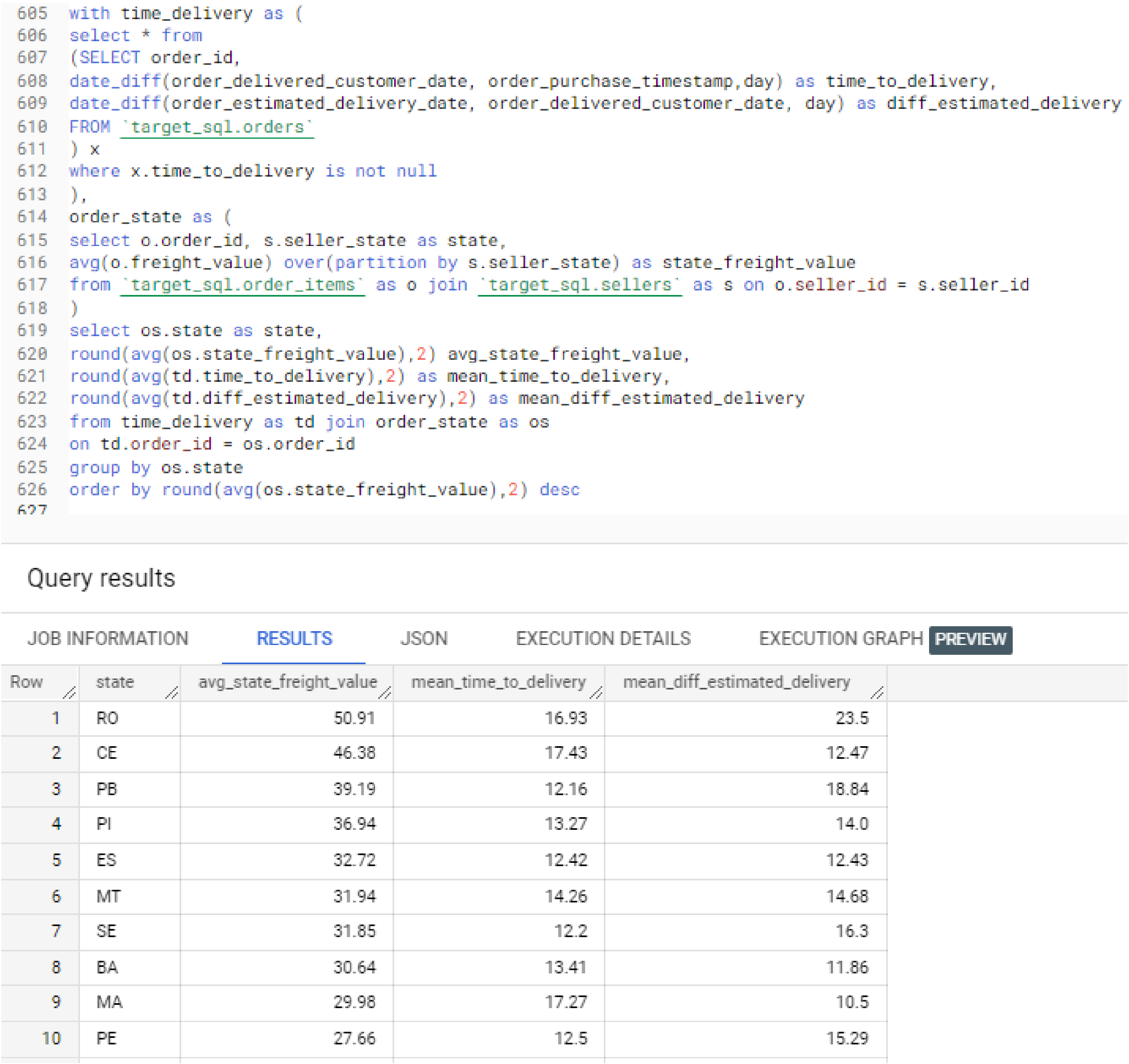
```
select * from
(SELECT order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day)
as time_to_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)
as diff_estimated_delivery
FROM `target_sql.orders`
) x
where x.time_to_delivery is not null
```

Query results

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

5.2 Group data by state, taken mean of freight_value, time_to_delivery, diff_estimated_delivery

Group data by state, taken mean of freight_value, time_to_delivery, diff_estimated_delivery: -



```sql
with time_delivery as (
select * from
(SELECT order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_to_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
FROM `target_sql.orders`
) x
where x.time_to_delivery is not null
),
order_state as (
select o.order_id, s.seller_state as state,
avg(o.freight_value) over(partition by s.seller_state) as state_freight_value
from `target_sql.order_items` as o join `target_sql.sellers` as s on o.seller_id = s.seller_id
)
select os.state as state,
round(avg(os.state_freight_value),2) avg_state_freight_value,
round(avg(td.time_to_delivery),2) as mean_time_to_delivery,
round(avg(td.diff_estimated_delivery),2) as mean_diff_estimated_delivery
from time_delivery as td join order_state as os
on td.order_id = os.order_id
group by os.state
order by round(avg(os.state_freight_value),2) desc
```

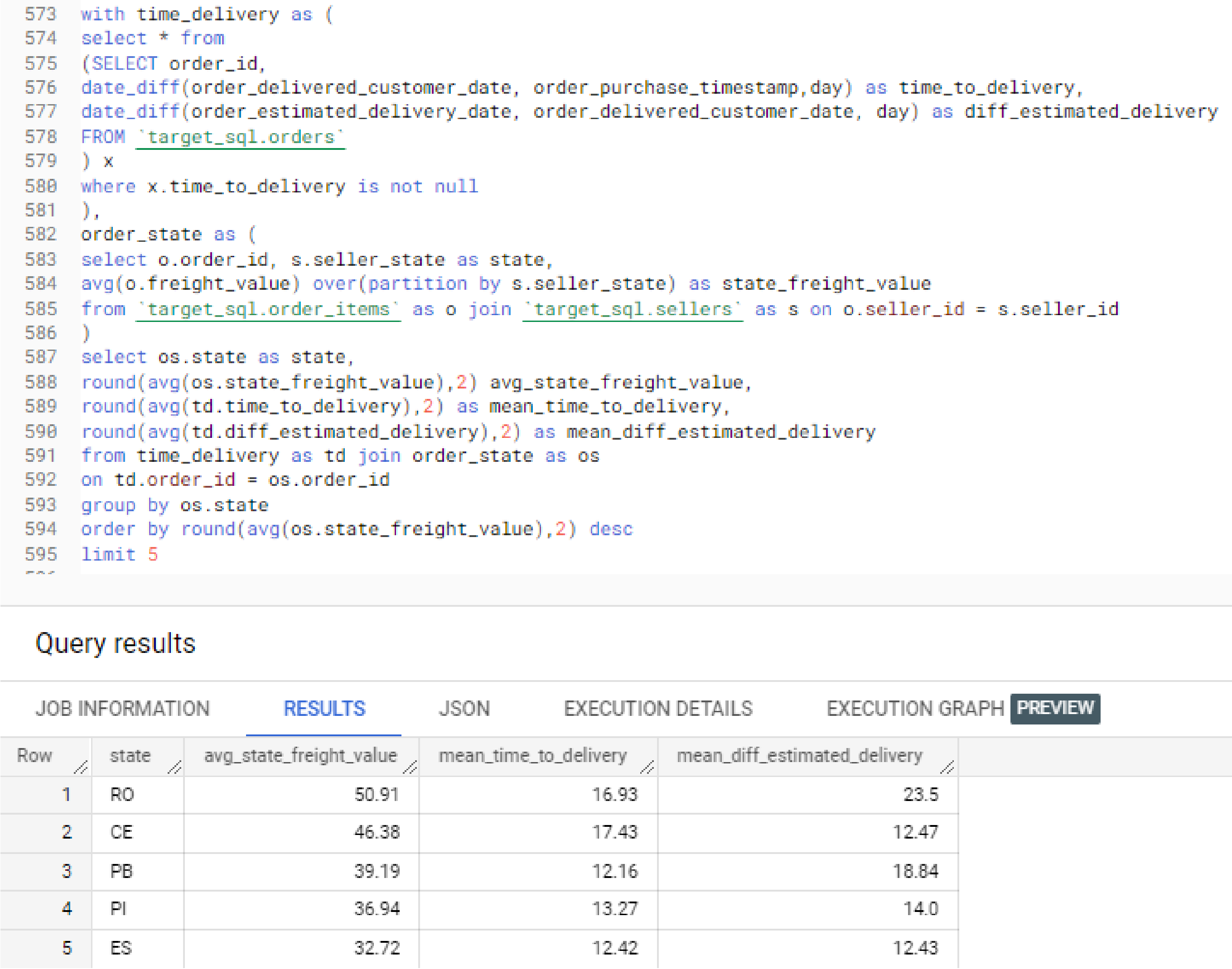
Query results

Row	state	avg_state_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	RO	50.91	16.93	23.5
2	CE	46.38	17.43	12.47
3	PB	39.19	12.16	18.84
4	PI	36.94	13.27	14.0
5	ES	32.72	12.42	12.43
6	MT	31.94	14.26	14.68
7	SE	31.85	12.2	16.3
8	BA	30.64	13.41	11.86
9	MA	29.98	17.27	10.5
10	PE	27.66	12.5	15.29

5.3 Sort the data to get the following details:

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

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

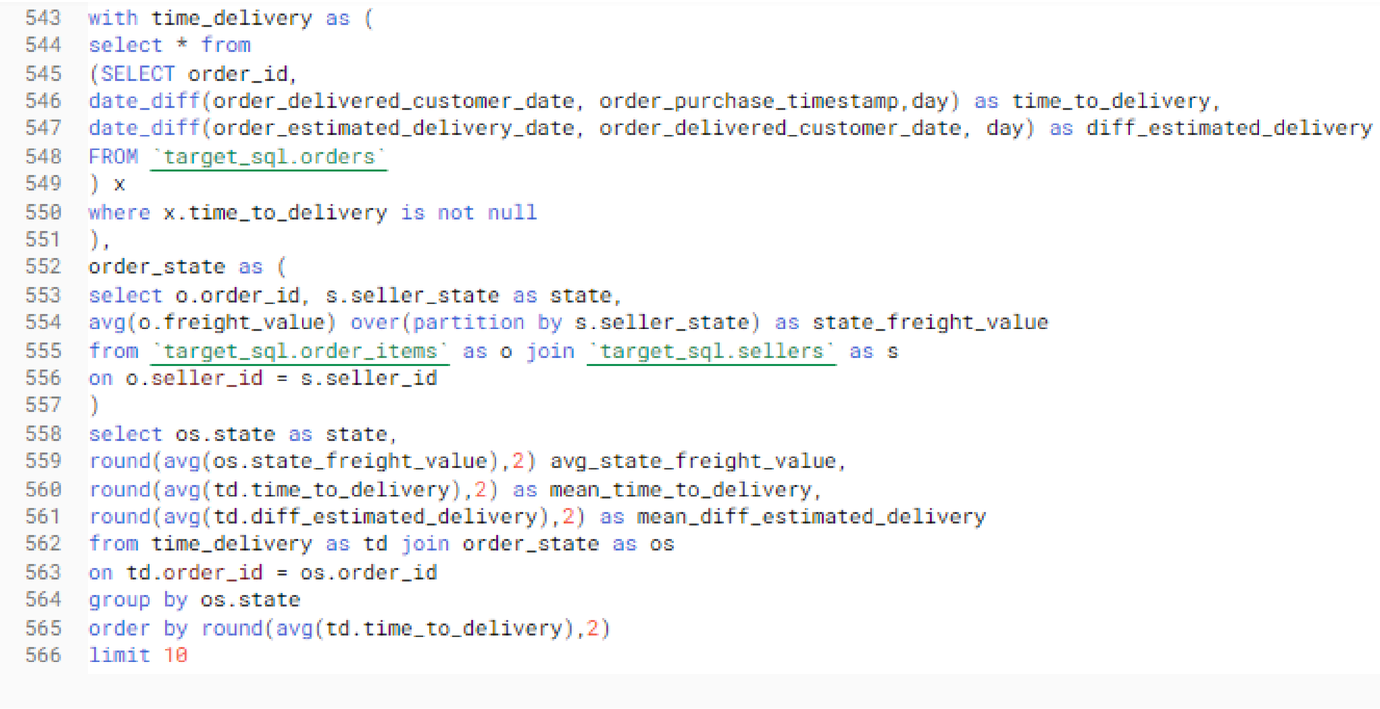


```sql
with time_delivery as (
select * from
(SELECT order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_to_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
FROM `target_sql.orders`
) x
where x.time_to_delivery is not null
),
order_state as (
select o.order_id, s.seller_state as state,
avg(o.freight_value) over(partition by s.seller_state) as state_freight_value
from `target_sql.order_items` as o join `target_sql.sellers` as s on o.seller_id = s.seller_id
)
select os.state as state,
round(avg(os.state_freight_value),2) avg_state_freight_value,
round(avg(td.time_to_delivery),2) as mean_time_to_delivery,
round(avg(td.diff_estimated_delivery),2) as mean_diff_estimated_delivery
from time_delivery as td join order_state as os
on td.order_id = os.order_id
group by os.state
order by round(avg(os.state_freight_value),2) desc
limit 5
```

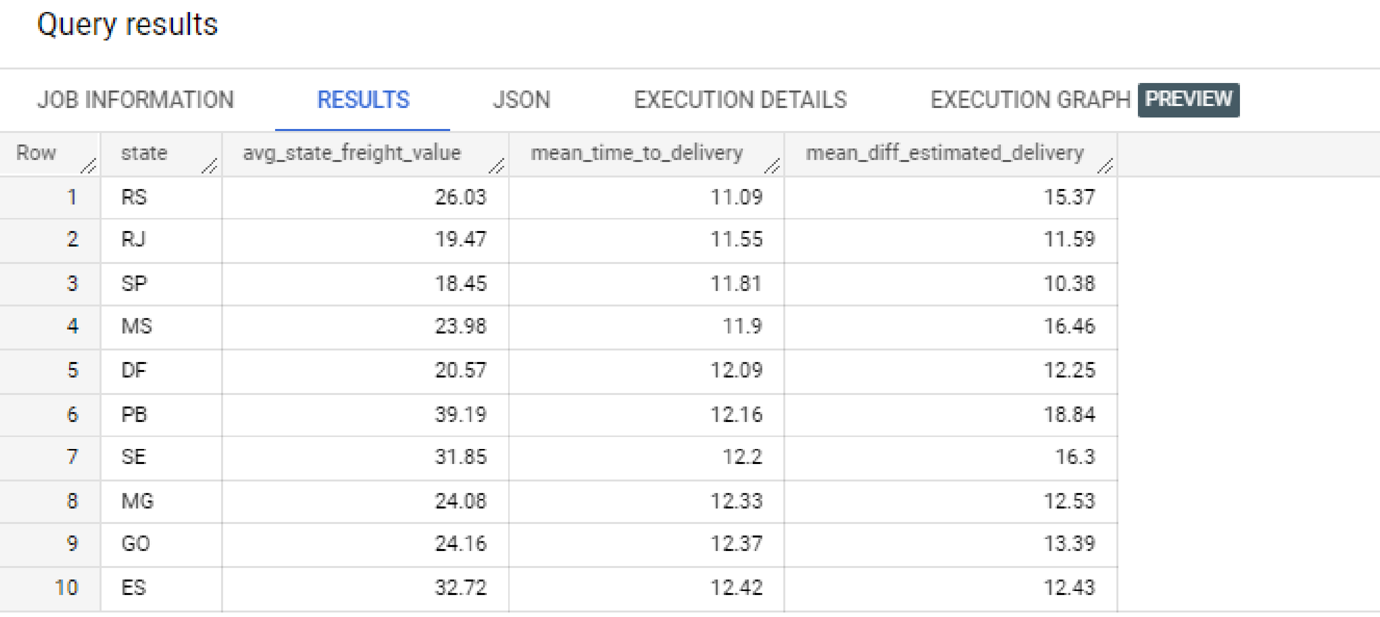
Row	state	avg_state_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	RO	50.91	16.93	23.5
2	CE	46.38	17.43	12.47
3	PB	39.19	12.16	18.84
4	PI	36.94	13.27	14.0
5	ES	32.72	12.42	12.43

5.3.2 Top 5 states with highest/ lowest average time to delivery

Top 5 states with lowest average time to delivery:-



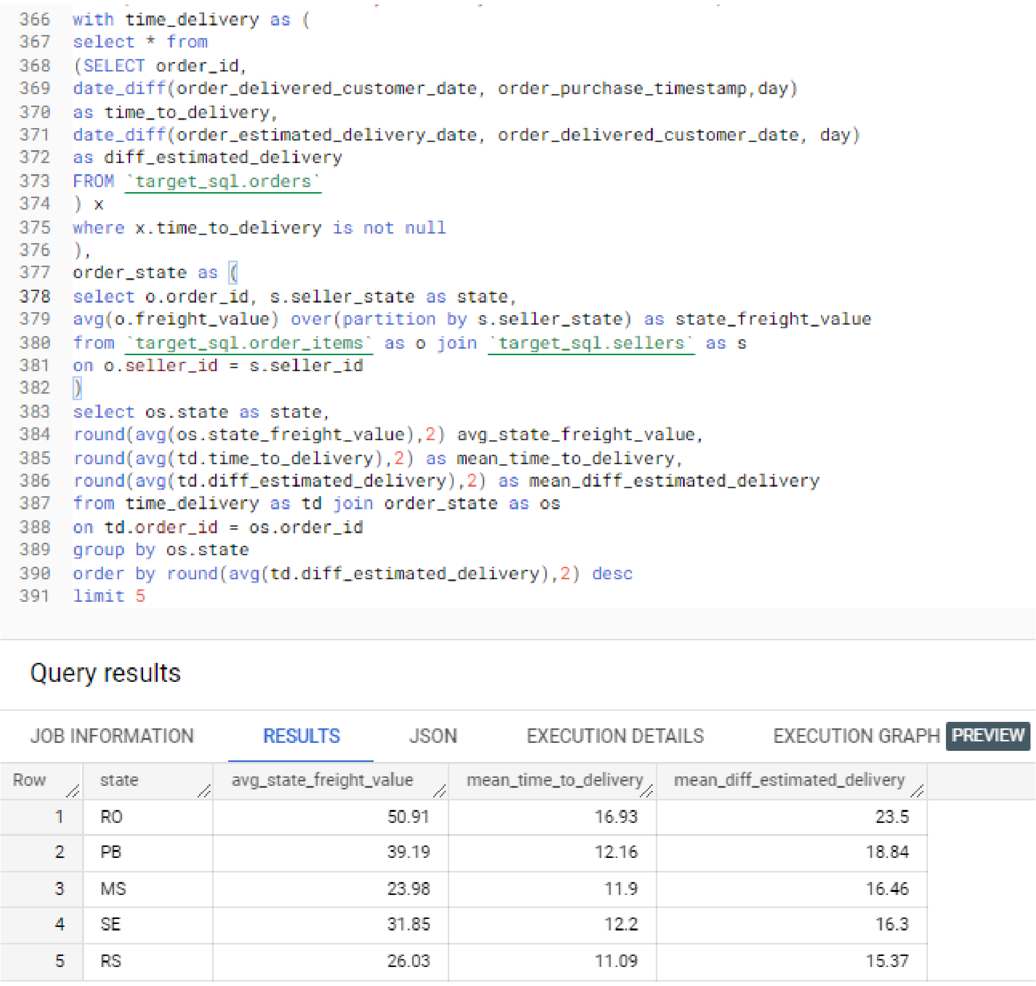
```
with time_delivery as (
select * from
(SELECT order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_to_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
FROM `target_sql.orders`
) x
where x.time_to_delivery is not null
),
order_state as (
select o.order_id, s.seller_state as state,
avg(o.freight_value) over(partition by s.seller_state) as state_freight_value
from `target_sql.order_items` as o join `target_sql.sellers` as s
on o.seller_id = s.seller_id
)
select os.state as state,
round(avg(os.state_freight_value),2) avg_state_freight_value,
round(avg(td.time_to_delivery),2) as mean_time_to_delivery,
round(avg(td.diff_estimated_delivery),2) as mean_diff_estimated_delivery
from time_delivery as td join order_state as os
on td.order_id = os.order_id
group by os.state
order by round(avg(td.time_to_delivery),2)
limit 10
```



Row	state	avg_state_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	RS	26.03	11.09	15.37
2	RJ	19.47	11.55	11.59
3	SP	18.45	11.81	10.38
4	MS	23.98	11.9	16.46
5	DF	20.57	12.09	12.25
6	PB	39.19	12.16	18.84
7	SE	31.85	12.2	16.3
8	MG	24.08	12.33	12.53
9	GO	24.16	12.37	13.39
10	ES	32.72	12.42	12.43

5.3.3 Top 5 states where delivery is really fast compared to estimated date

Top 5 states where delivery is really fast compared to estimated date: -



```
with time_delivery as (
select * from
(SELECT order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp,day)
as time_to_delivery,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)
as diff_estimated_delivery
FROM `target_sql.orders`
) x
where x.time_to_delivery is not null
),
order_state as (
select o.order_id, s.seller_state as state,
avg(o.freight_value) over(partition by s.seller_state) as state_freight_value
from `target_sql.order_items` as o join `target_sql.sellers` as s
on o.seller_id = s.seller_id
)
select os.state as state,
round(avg(os.state_freight_value),2) avg_state_freight_value,
round(avg(td.time_to_delivery),2) as mean_time_to_delivery,
round(avg(td.diff_estimated_delivery),2) as mean_diff_estimated_delivery
from time_delivery as td join order_state as os
on td.order_id = os.order_id
group by os.state
order by round(avg(td.diff_estimated_delivery),2) desc
limit 5
```

Query results

Row	state	avg_state_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	RO	50.91	16.93	23.5
2	PB	39.19	12.16	18.84
3	MS	23.98	11.9	16.46
4	SE	31.85	12.2	16.3
5	RS	26.03	11.09	15.37

OBSERVATIONS: -

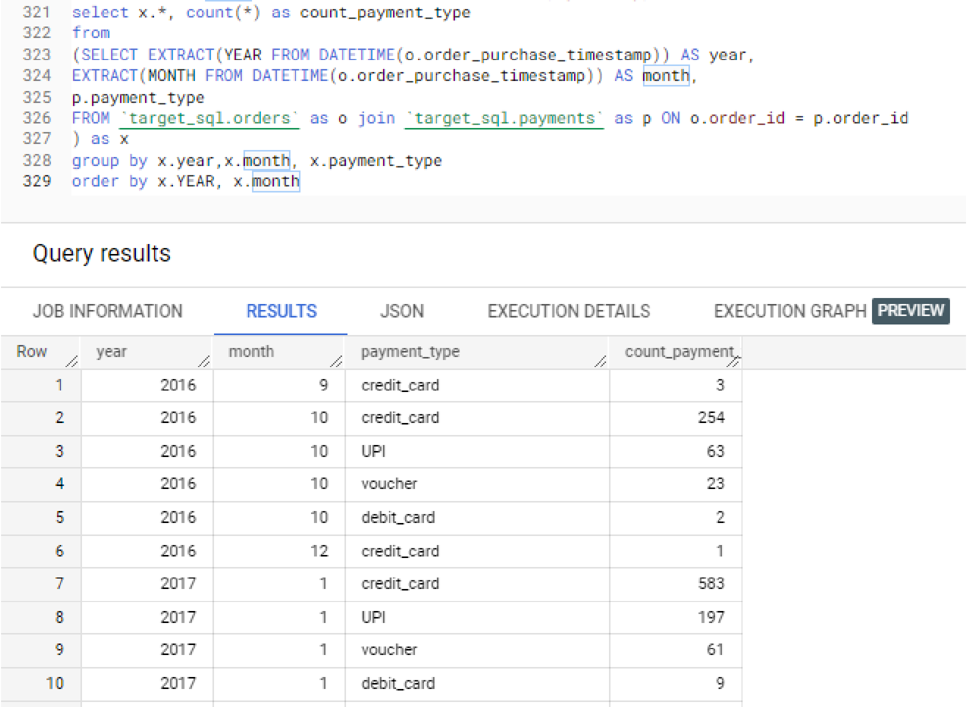
- The difference between the estimated delivery time and the actual delivery time is calculated as Mean_diff_estimated_delivery = estimated_delivery – actual_delivery.
- Mean of diff_estimated_delivery across all states is 12.4. Hence higher the 'Mean_diff_estimated_delivery', faster is the actual delivery than estimated.

- The state with the highest average freight value is RO, while SP has the lowest average freight value.
- PB State has the quickest delivery service, with a mean time-to-delivery of 12.16. In contrast, RO State has the highest number of deliveries made ahead of schedule and experiences the greatest variance between estimated and actual delivery times.
- AM has the longest time-to-delivery among all states, and it is the only state where the number of delayed deliveries exceeds the number of estimated ones, as indicated by the negative Mean_diff_estimated_delivery.
- On the other hand, RO has the highest number of deliveries made before the estimated time.

6. Payment Type Analysis

6.1 Month over month count of orders for different payment types:

Month over month count of orders for different payment modes: -



```sql
select x.*, count(*) as count_payment_type
from
(SELECT EXTRACT(YEAR FROM DATETIME(o.order_purchase_timestamp)) AS year,
EXTRACT(MONTH FROM DATETIME(o.order_purchase_timestamp)) AS month,
p.payment_type
FROM `target_sql.orders` as o join `target_sql.payments` as p ON o.order_id = p.order_id
) as x
group by x.year,x.month, x.payment_type
order by x.YEAR, x.month
```

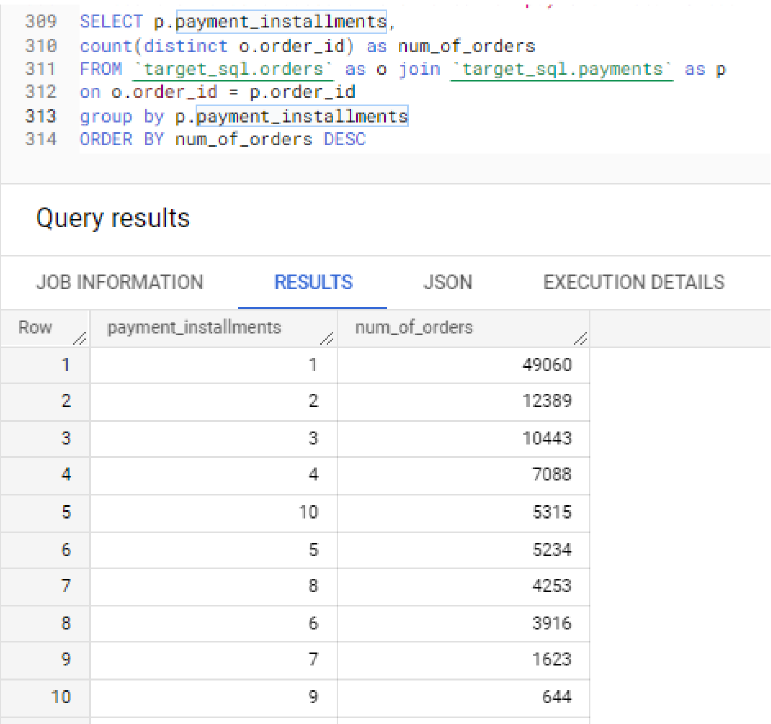
Row	year	month	payment_type	count_payment_
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9

OBSERVATIONS: -

- Limited data analysis shows that the use of credit card and UPI for the purpose of ordering is on the rise.
- Usage of credit card almost doubled from October 2016 to January 2017 and usage of UPI mode has almost tripled during the same period.

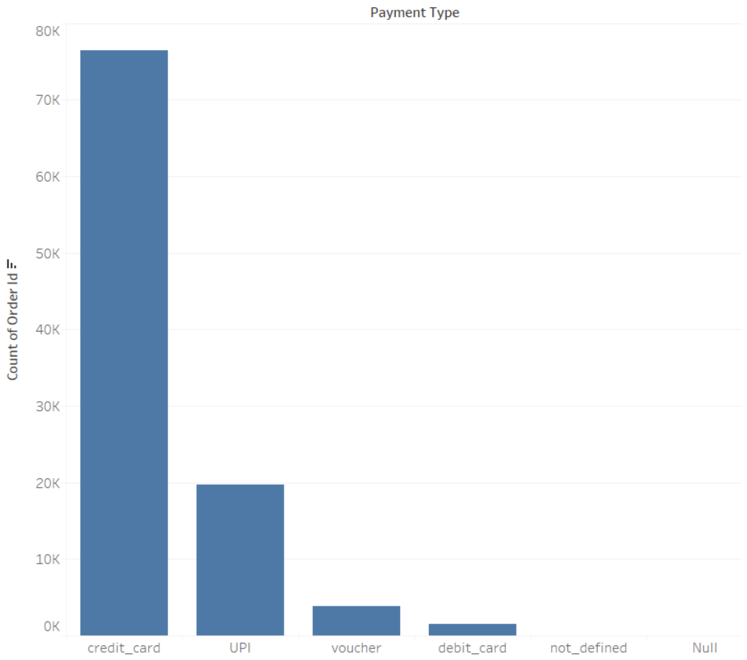
6.2 Count of orders based on the number of payment instalments:

Payment instalment wise number of orders: -



```sql
SELECT p.payment_installments,
count(distinct o.order_id) as num_of_orders
FROM `target_sql.orders` as o join `target_sql.payments` as p
on o.order_id = p.order_id
group by p.payment_installments
ORDER BY num_of_orders DESC
```

Row	payment_installments	num_of_orders
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	10	5315
6	5	5234
7	8	4253
8	6	3916
9	7	1623
10	9	644



OBSERVATIONS: -

- **Maximum number of orders (49060 no.) are completed using 1 payment instalment**. This means that for majority of order, payment is done in one go.
- Orders which used payments above 10 instalments are comparatively very low and can be neglected
- **With the increase in the number of instalments, number of orders is decreasing**
- **Most of the payments are done using credit card**, followed by upi, voucher etc.

7. Actionable Insights

Actionable insights: -

1. By providing discounts and other offers to the customers from these untapped cities during the festival seasons, we can try to increase the customer base in these regions.
2. Expand delivery network and partner with local business from the cities where sellers are not present.
    a. It is advised to develop sellers in cities, based on the number of customers present.
    b. This will help in faster delivery of orders and better customer satisfaction too.
3. The majority of the reviews lacked any written content or title, only comprising of ratings. To enhance customer experience and boost sales, incentives could be provided to encourage customers to leave written reviews.
4. To fully capitalize on the seasonal fluctuations in demand, it is essential to offer a wider range and greater quantity of products, as well as attractive promotions and discounts.
5. In order to lower the shipping costs, it is necessary to enhance the transportation infrastructure and supply chain management in states where the freight value is relatively high.
6. The majority of customers who made payments in single or double instalments belong to higher income groups, indicating lower outreach to low-income groups who prefer paying in instalments. To tap into this untapped market and foster business growth, offering no-interest loans in partnership with banks to lower income groups presents an opportunity with huge potential for growth.