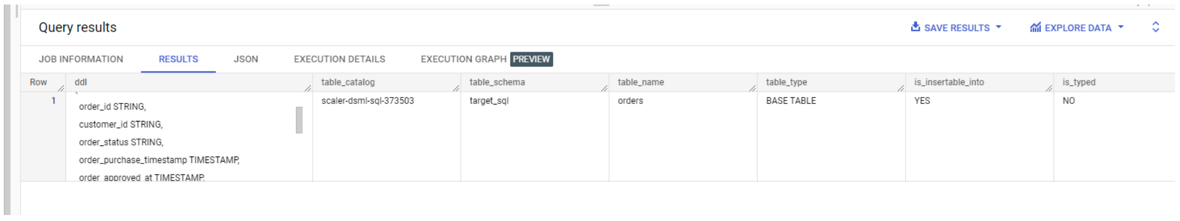
TARGET SQL

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

1. Data type of columns in a table

```
SELECT ddl,* FROM target_sql.INFORMATION_SCHEMA.TABLES
WHERE table_name = 'orders';
```

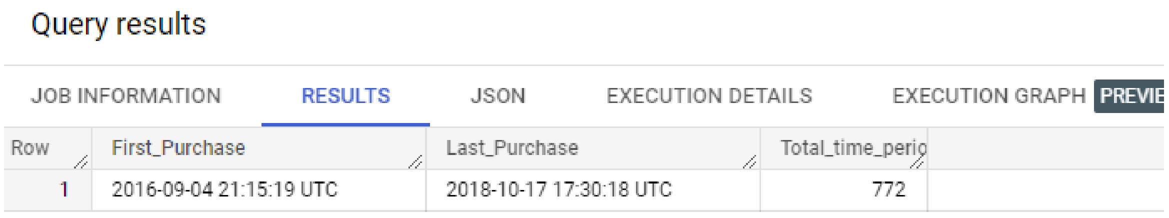

Query results

Row	ddl	table_catalog	table_schema	table_name	table_type	is_insertable_into	is_typed
1	order_id STRING, customer_id STRING, order_status STRING, order_purchase_timestamp TIMESTAMP, order_approved_at TIMESTAMP,	scaler-dsml-sql-373503	target_sql	orders	BASE TABLE	YES	NO

In ddl column, we can get the data types of the column for the given table name.

2. Time period for which the data is given.

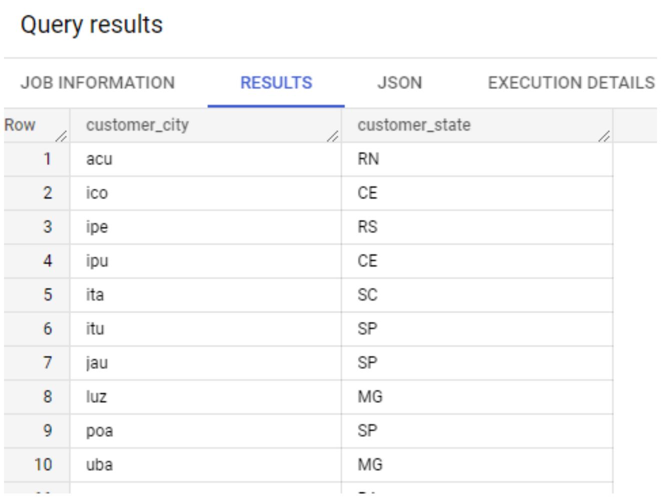
```
SELECT
  MIN(order_purchase_timestamp) AS First_Purchase,
  MAX(order_purchase_timestamp) AS Last_Purchase,
  DATE_DIFF(MAX(order_purchase_timestamp),MIN(order_purchase_timestamp),DAY) AS Total
_time_period
FROM `target_sql.orders`;
```


Query results

Row	First_Purchase	Last_Purchase	Total_time_perio
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772

3. Cities and States of customers ordered during the given period

```
SELECT DISTINCT customer_city,customer_state
FROM  `target_sql.customers` ;
```


Query results

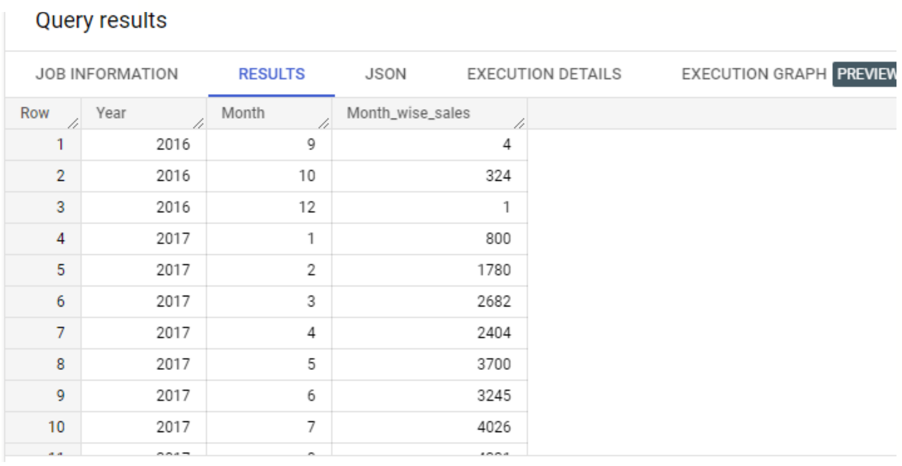
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

2. In-depth Exploration:

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?

```
SELECT * FROM
(SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
  COUNT(*) AS Month_wise_sales
FROM `target_sql.orders`
GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp), EXTRACT(MONTH FROM order_purc
hase_timestamp)) x
ORDER BY x.Year,x.Month;
```

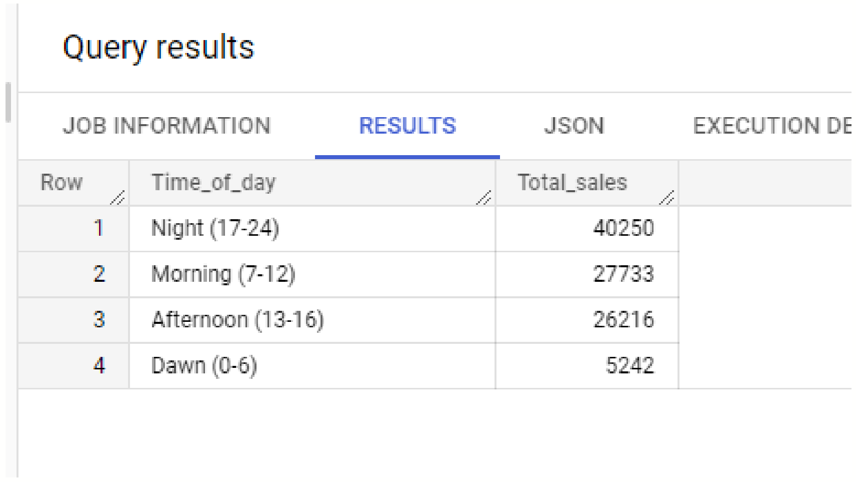

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	Year	Month	Month_wise_sales
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

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

```
SELECT * FROM
(SELECT
CASE
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn (0-6)'
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning (7-
12)'
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 16 THEN 'Afternoon (1
3-16)'
  ELSE 'Night (17-24)'
END AS Time_of_day,
COUNT(*) AS Total_sales
FROM `target_sql.orders`
GROUP BY
CASE
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn (0-6)'
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning (7-
12)'
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 16 THEN 'Afternoon (1
3-16)'
  ELSE 'Night (17-24)'
END) x
ORDER BY x.Total_sales DESC;
```

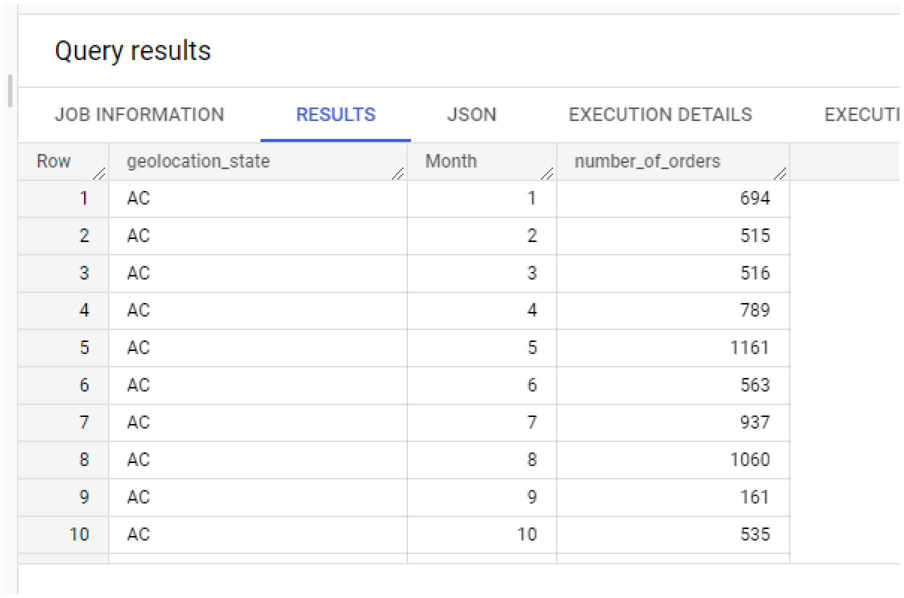

Query results

Row	Time_of_day	Total_sales
1	Night (17-24)	40250
2	Morning (7-12)	27733
3	Afternoon (13-16)	26216
4	Dawn (0-6)	5242

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

1. Get month on month orders by states

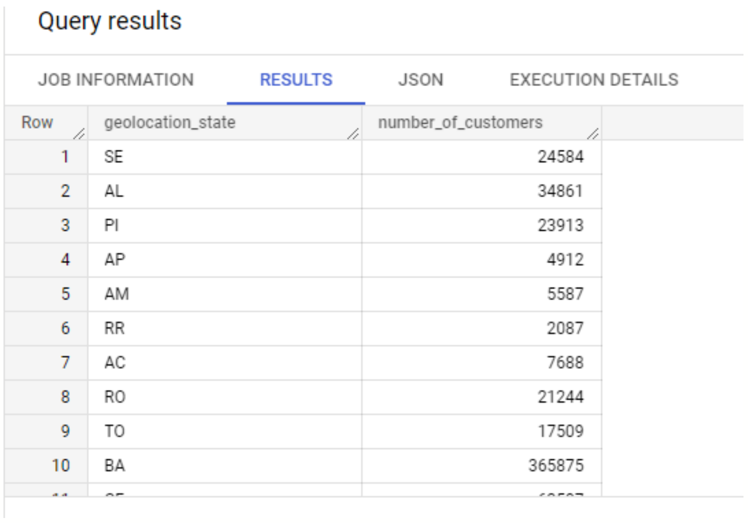
```
SELECT * FROM
(SELECT
  geolocation_state,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
  COUNT(o.order_id) AS number_of_orders
FROM `target_sql.geolocation` g LEFT JOIN `target_sql.customers` c ON geolocation_zip_c
ode_prefix=customer_zip_code_prefix
JOIN `target_sql.orders` o ON o.customer_id=c.customer_id
GROUP BY geolocation_state,EXTRACT(MONTH FROM order_purchase_timestamp)) x
ORDER BY x.geolocation_state,x.Month;
```


Query results

Row	geolocation_state	Month	number_of_orders
1	AC	1	694
2	AC	2	515
3	AC	3	516
4	AC	4	789
5	AC	5	1161
6	AC	6	563
7	AC	7	937
8	AC	8	1060
9	AC	9	161
10	AC	10	535

2. Distribution of customers across the states in Brazil

```
SELECT
  geolocation_state,
  COUNT(customer_id) AS number_of_customers
FROM `target_sql.geolocation` g JOIN `target_sql.customers` c ON geolocation_zip_code_p
refix=customer_zip_code_prefix
GROUP BY geolocation_state;
```


Query results

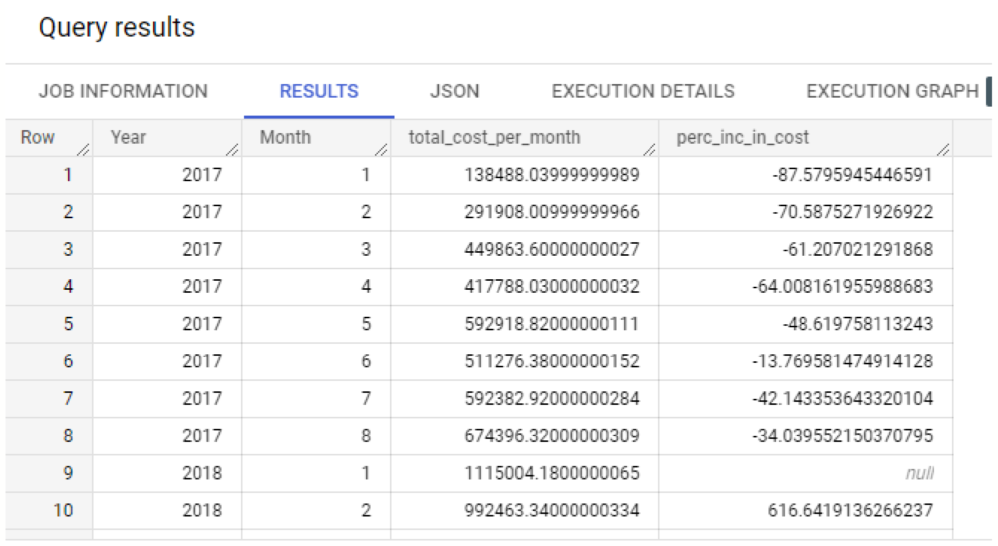
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	geolocation_state	number_of_customers
1	SE	24584
2	AL	34861
3	PI	23913
4	AP	4912
5	AM	5587
6	RR	2087
7	AC	7688
8	RO	21244
9	TO	17509
10	BA	365875

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

1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table

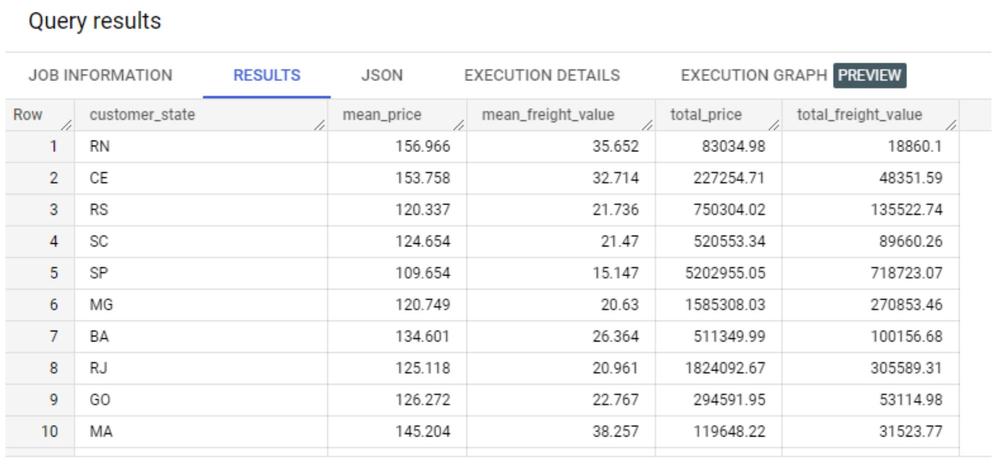
```
SELECT
  Year,
  Month,
  total_cost_per_month,
  IF(LAG(total_cost_per_month) OVER (ORDER BY Month ASC) = 0, 0 ,(total_cost_per_month -
 LAG (total_cost_per_month) OVER (ORDER BY Month ASC))/LAG (total_cost_per_month) OVER (ORDER BY M
onth ASC)*100) AS perc_inc_in_cost
FROM
(SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
  SUM(payment_value) AS total_cost_per_month,
FROM `target_sql.payments` p JOIN `target_sql.orders` o ON p.order_id=o.order_id
GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp),EXTRACT(MONTH FROM order_purchase_timestamp))
 x
WHERE x.Year BETWEEN 2017 AND 2018 AND x.Month BETWEEN 1 AND 8
order by Year,Month;
```



Row	Year	Month	total_cost_per_month	perc_inc_in_cost
1	2017	1	138488.03999999989	-87.5795945446591
2	2017	2	291908.00999999966	-70.5875271926922
3	2017	3	449863.60000000027	-61.207021291868
4	2017	4	417788.03000000032	-64.008161955988683
5	2017	5	592918.82000000111	-48.619758113243
6	2017	6	511276.38000000152	-13.769581474914128
7	2017	7	592382.92000000284	-42.143353643320104
8	2017	8	674396.32000000309	-34.039552150370795
9	2018	1	1115004.1800000065	null
10	2018	2	992463.34000000334	616.6419136266237

2. Mean & Sum of price and freight value by customer state

```
SELECT
  customer_state,
  ROUND(AVG(price),3) AS mean_price,
  ROUND(AVG(freight_value),3) AS mean_freight_value,
  ROUND(SUM(price),3) AS total_price,
  ROUND(SUM(freight_value),3) AS total_freight_value
FROM `target_sql.customers` c JOIN `target_sql.orders` o ON c.customer_id=o.customer_id
 JOIN `target_sql.order_items` i ON o.order_id=i.order_id
GROUP BY customer_state;
```



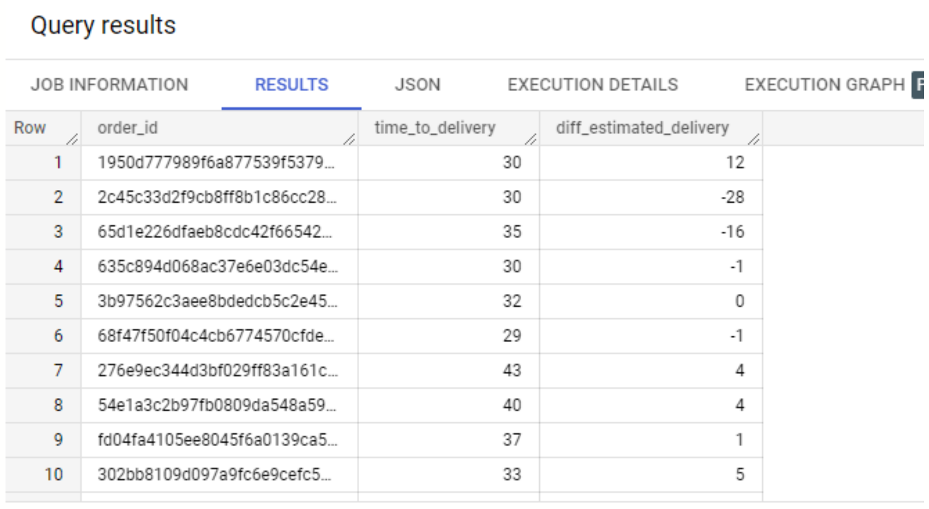
Row	customer_state	mean_price	mean_freight_value	total_price	total_freight_value
1	RN	156.966	35.652	83034.98	18860.1
2	CE	153.758	32.714	227254.71	48351.59
3	RS	120.337	21.736	750304.02	135522.74
4	SC	124.654	21.47	520553.34	89660.26
5	SP	109.654	15.147	5202955.05	718723.07
6	MG	120.749	20.63	1585308.03	270853.46
7	BA	134.601	26.364	511349.99	100156.68
8	RJ	125.118	20.961	1824092.67	305589.31
9	GO	126.272	22.767	294591.95	53114.98
10	MA	145.204	38.257	119648.22	31523.77

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery
   Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
   time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY) AS time_to_deli
very ,
  DATE_DIFF(order_delivered_customer_date,order_estimated_delivery_date,DAY) AS diff_es
timated_delivery
FROM `target_sql.orders`;
```

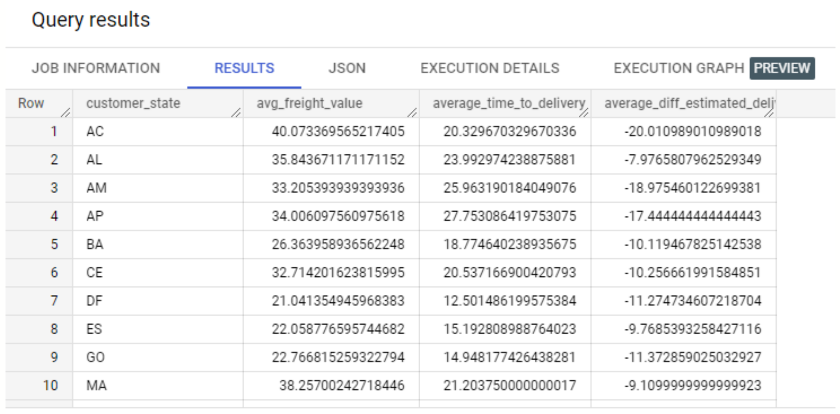

Query results

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

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

2. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

```
SELECT
  customer_state,
  AVG(freight_value) AS avg_freight_value,
  AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)) AS average
_time_to_delivery,
  AVG(DATE_DIFF(order_delivered_customer_date,order_estimated_delivery_date,DAY)) AS av
erage_diff_estimated_delivery
FROM `target_sql.order_items` i JOIN `target_sql.orders` o ON i.order_id=o.order_id JOI
N `target_sql.customers` c ON o.customer_id=c.customer_id
GROUP BY customer_state
ORDER BY customer_state;
```

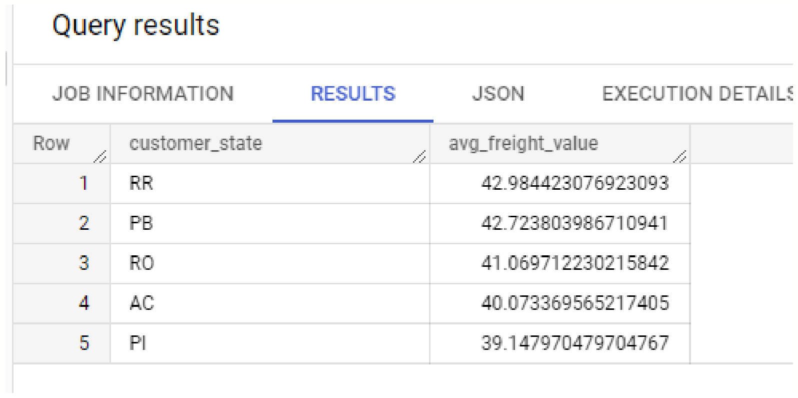

Query results

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

Row	customer_state	avg_freight_value	average_time_to_delivery	average_diff_estimated_deli
1	AC	40.073369565217405	20.329670329670336	-20.010989010989018
2	AL	35.843671171171152	23.992974238875881	-7.9765807962529349
3	AM	33.205393939393936	25.963190184049076	-18.975460122699381
4	AP	34.006097560975618	27.753086419753075	-17.444444444444443
5	BA	26.363958936562248	18.774640238935675	-10.119467825142538
6	CE	32.714201623815995	20.537166900420793	-10.256661991584851
7	DF	21.041354945968383	12.501486199575384	-11.274734607218704
8	ES	22.058776595744682	15.192808988764023	-9.7685393258427116
9	GO	22.766815259322794	14.948177426438281	-11.372859025032927
10	MA	38.25700242718446	21.203750000000017	-9.1099999999999923

3. Sort the data to get the following:
4. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

```
SELECT * FROM
(SELECT
  customer_state,
  AVG(freight_value) AS avg_freight_value
FROM `target_sql.order_items` i JOIN `target_sql.orders` o ON i.order_id=o.order_id JOI
N `target_sql.customers` c ON o.customer_id=c.customer_id
GROUP BY customer_state) x
ORDER BY x.avg_freight_value DESC LIMIT 5;
```



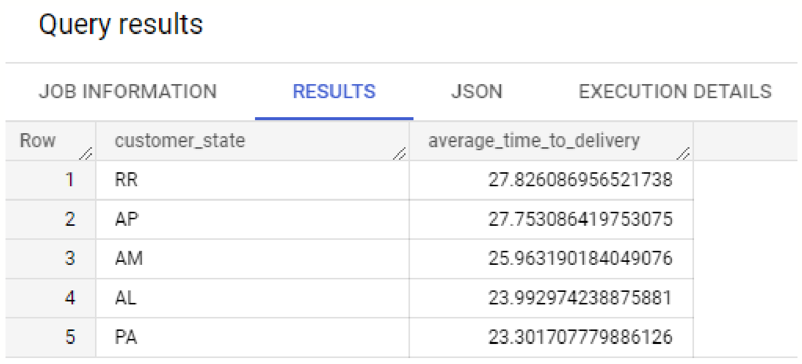
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_freight_value
1	RR	42.984423076923093
2	PB	42.723803986710941
3	RO	41.069712230215842
4	AC	40.073369565217405
5	PI	39.147970479704767

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

```
SELECT * FROM
(SELECT
  customer_state,
  AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)) AS average
_time_to_delivery
FROM `target_sql.order_items` i JOIN `target_sql.orders` o ON i.order_id=o.order_id JOI
N `target_sql.customers` c ON o.customer_id=c.customer_id
GROUP BY customer_state ) x
ORDER BY x.average_time_to_delivery DESC LIMIT 5;
```



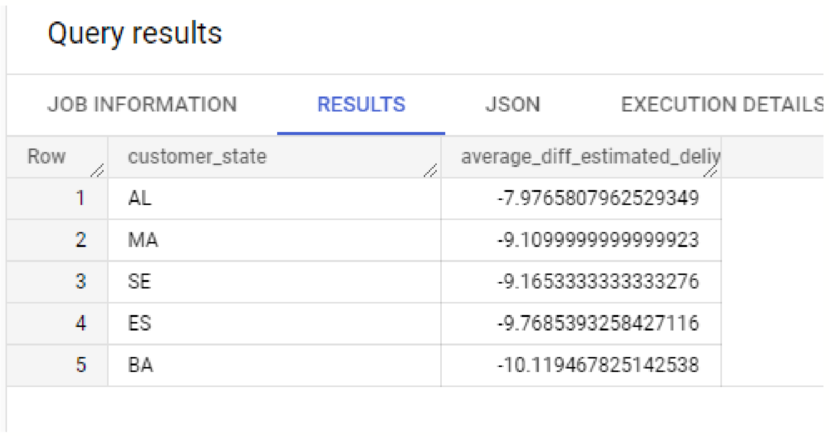
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	average_time_to_delivery
1	RR	27.826086956521738
2	AP	27.753086419753075
3	AM	25.963190184049076
4	AL	23.992974238875881
5	PA	23.301707779886126

6. Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
SELECT * FROM
(SELECT
  customer_state,
  AVG(DATE_DIFF(order_delivered_customer_date,order_estimated_delivery_date,DAY)) AS av
erage_diff_estimated_delivery
FROM `target_sql.order_items` i JOIN `target_sql.orders` o ON i.order_id=o.order_id JOI
N `target_sql.customers` c ON o.customer_id=c.customer_id
GROUP BY customer_state ) x
ORDER BY x.average_diff_estimated_delivery DESC LIMIT 5;
```



Query results

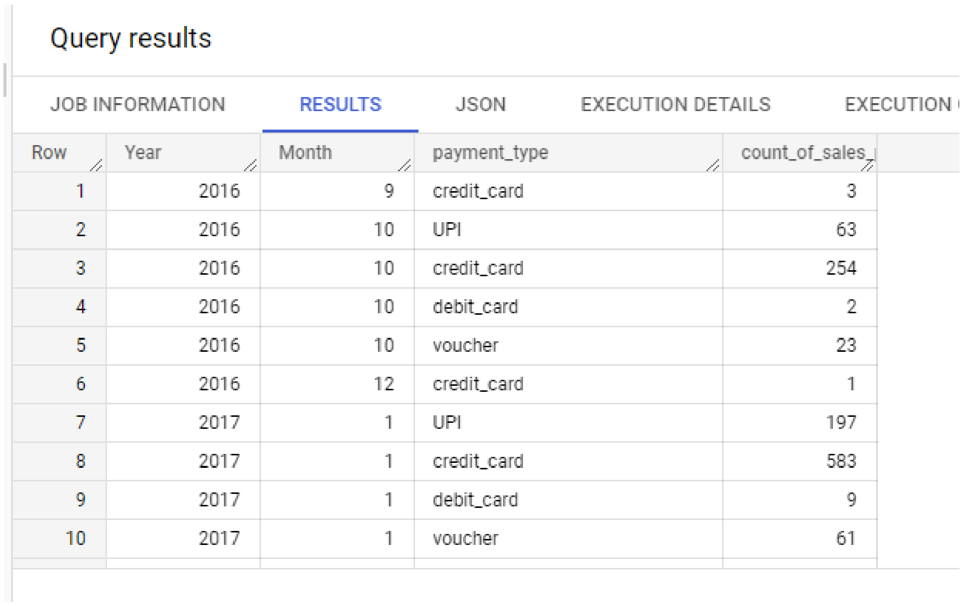
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	average_diff_estimated_deliv
1	AL	-7.9765807962529349
2	MA	-9.1099999999999923
3	SE	-9.1653333333333276
4	ES	-9.7685393258427116
5	BA	-10.119467825142538

6. Payment type analysis:

1. Month over Month count of orders for different payment types

```
SELECT * FROM
(SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
  payment_type,
  COUNT(p.order_id)  AS count_of_sales_per_payment
FROM `target_sql.payments` p JOIN  `target_sql.orders` o ON p.order_id=o.order_id
GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp), EXTRACT(MONTH FROM order_purchase
_timestamp),payment_type)
ORDER BY Year,Month,payment_type;
```

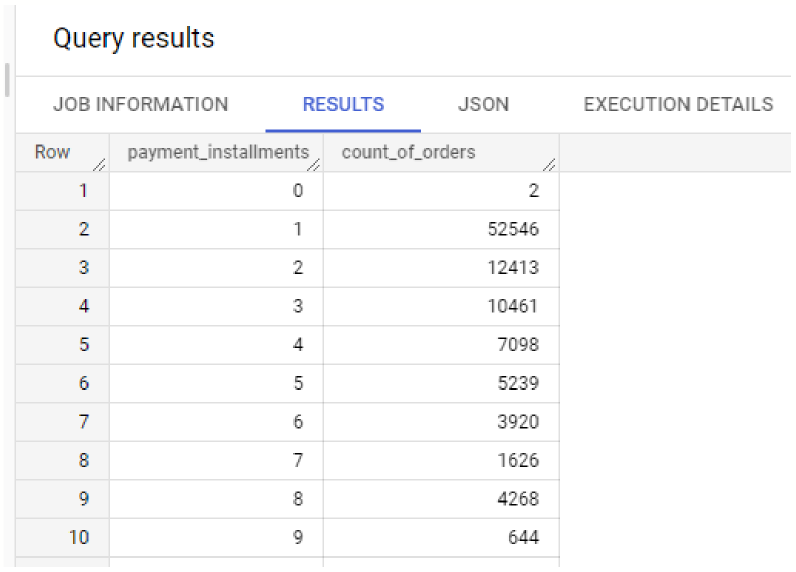

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION

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

2. Count of orders based on the no. of payment installments

```
SELECT
  payment_installments,
  COUNT(order_id) AS count_of_orders
FROM `target_sql.payments`
GROUP BY payment_installments;
```


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_installments	count_of_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