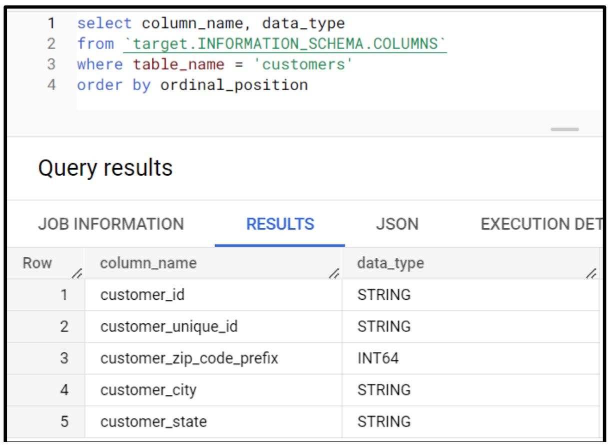
Business case study : Target

1. Structure & characteristics of the dataset:

1.Data type of columns in a table:

- Customers:



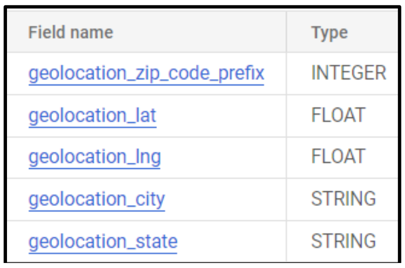
```
select column_name, data_type
from `target.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
order by ordinal_position
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DET

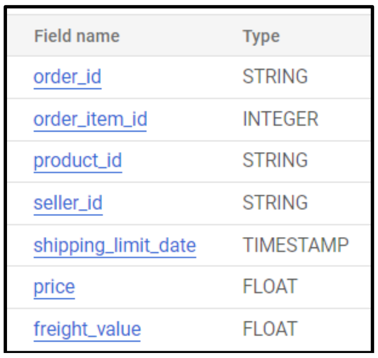
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

- Geolocations:



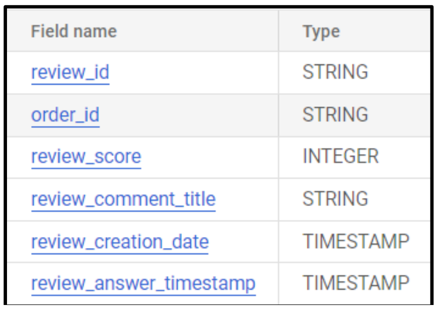
Field name	Type
geolocation_zip_code_prefix	INTEGER
geolocation_lat	FLOAT
geolocation_lng	FLOAT
geolocation_city	STRING
geolocation_state	STRING

- Orders_items:



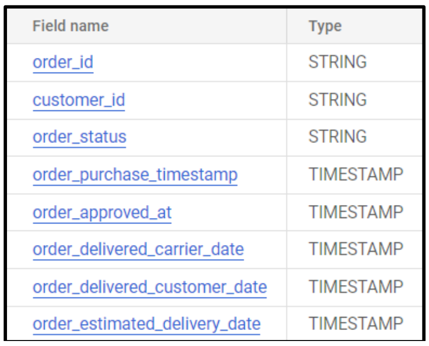
Field name	Type
order_id	STRING
order_item_id	INTEGER
product_id	STRING
seller_id	STRING
shipping_limit_date	TIMESTAMP
price	FLOAT
freight_value	FLOAT

- Order_reviews:



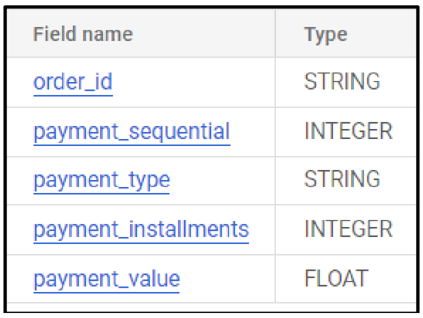
Field name	Type
review_id	STRING
order_id	STRING
review_score	INTEGER
review_comment_title	STRING
review_creation_date	TIMESTAMP
review_answer_timestamp	TIMESTAMP

- Orders:



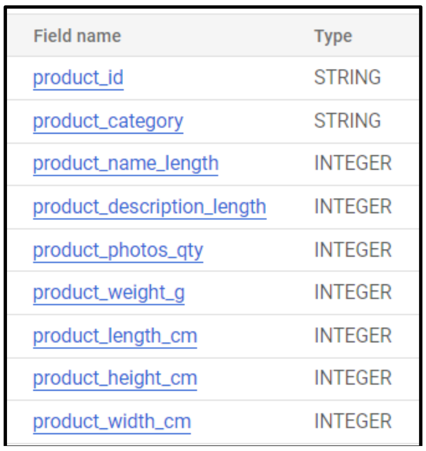
Field name	Type
order_id	STRING
customer_id	STRING
order_status	STRING
order_purchase_timestamp	TIMESTAMP
order_approved_at	TIMESTAMP
order_delivered_carrier_date	TIMESTAMP
order_delivered_customer_date	TIMESTAMP
order_estimated_delivery_date	TIMESTAMP

- Payments:



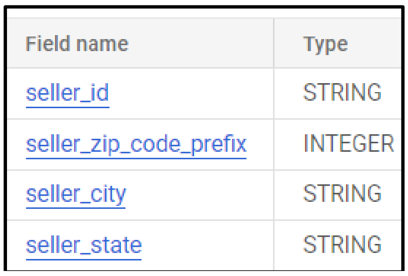
Field name	Type
order_id	STRING
payment_sequential	INTEGER
payment_type	STRING
payment_installments	INTEGER
payment_value	FLOAT

- Products:



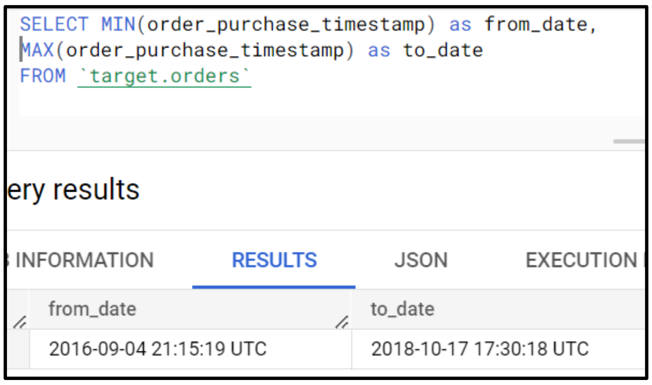
Field name	Type
product_id	STRING
product_category	STRING
product_name_length	INTEGER
product_description_length	INTEGER
product_photos_qty	INTEGER
product_weight_g	INTEGER
product_length_cm	INTEGER
product_height_cm	INTEGER
product_width_cm	INTEGER

- Sellers:



Field name	Type
seller_id	STRING
seller_zip_code_prefix	INTEGER
seller_city	STRING
seller_state	STRING

2.time period for which the data is given:



```
SELECT MIN(order_purchase_timestamp) as from_date,
MAX(order_purchase_timestamp) as to_date
FROM `target.orders`
```

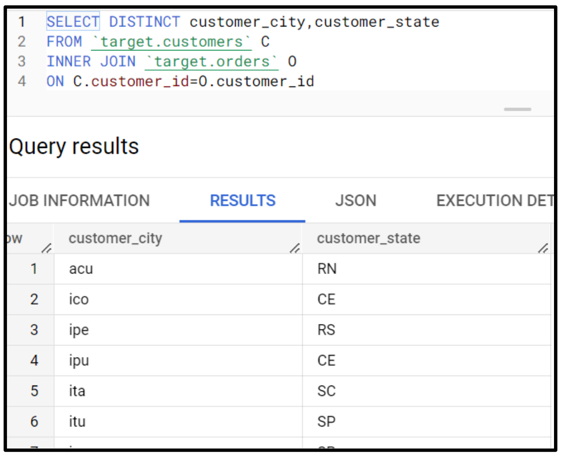
ery results

INFORMATION RESULTS JSON EXECUTION

```
from_date                  to_date
2016-09-04 21:15:19 UTC    2018-10-17 17:30:18 UTC
```

- Data is present for range : sept 2016 -oct 2018

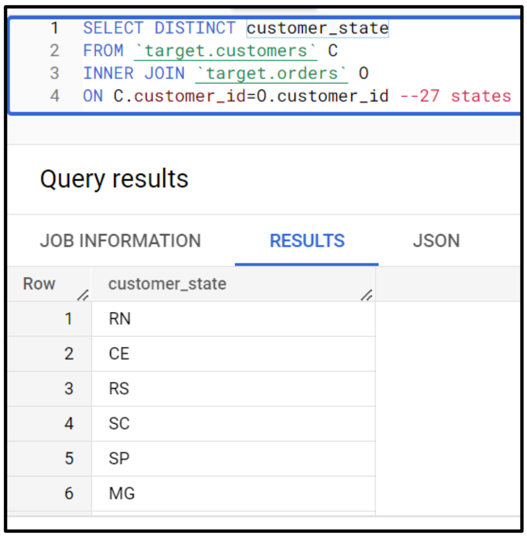
2. Cities and states of customers ordered during the given period:



```sql
SELECT DISTINCT customer_city,customer_state
FROM `target.customers` C
INNER JOIN `target.orders` O
ON C.customer_id=O.customer_id
```

Query results

ow	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP



```sql
SELECT DISTINCT customer_state
FROM `target.customers` C
INNER JOIN `target.orders` O
ON C.customer_id=O.customer_id --27 states
```

Query results

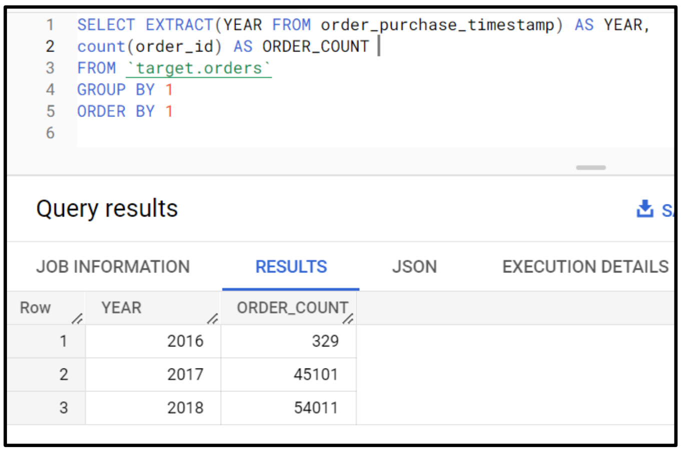
Row	customer_state
1	RN
2	CE
3	RS
4	SC
5	SP
6	MG

- Total 27 states from where orders have been placed.

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?**

   - To check growing trend lets first find out count of orders per year:



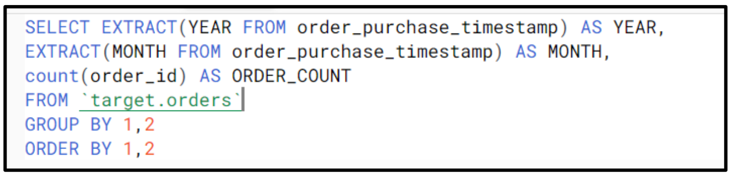
```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS YEAR,
count(order_id) AS ORDER_COUNT
FROM `target.orders`
GROUP BY 1
ORDER BY 1
```

Query results

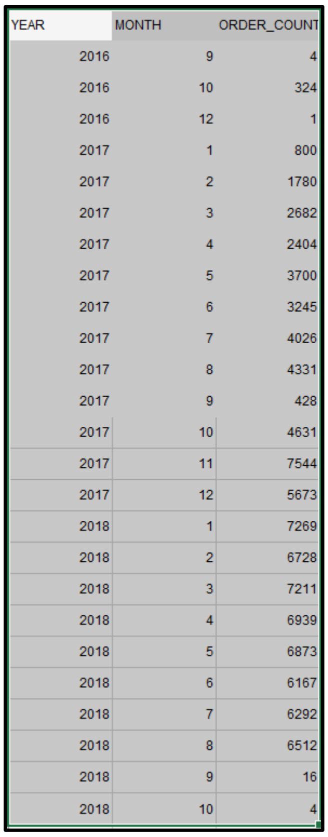
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	YEAR	ORDER_COUNT
1	2016	329
2	2017	45101
3	2018	54011

- ➢ And count of orders per month -year:

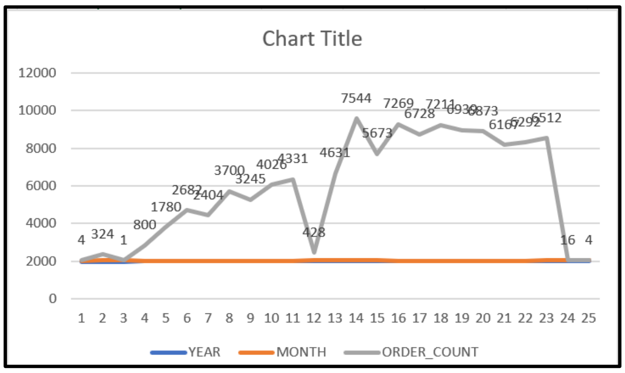


```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS YEAR,
EXTRACT(MONTH FROM order_purchase_timestamp) AS MONTH,
count(order_id) AS ORDER_COUNT
FROM `target.orders`
GROUP BY 1,2
ORDER BY 1,2
```



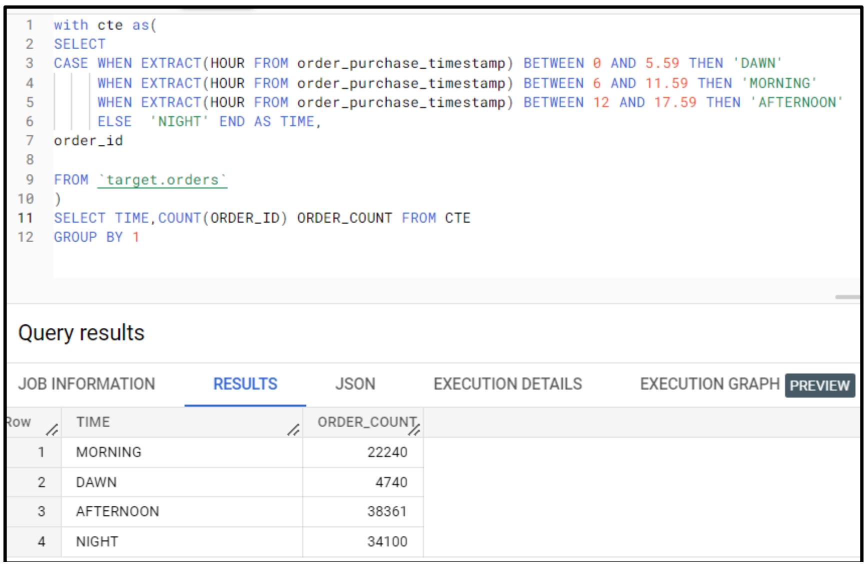
YEAR	MONTH	ORDER_COUNT
2016	9	4
2016	10	324
2016	12	1
2017	1	800
2017	2	1780
2017	3	2682
2017	4	2404
2017	5	3700
2017	6	3245
2017	7	4026
2017	8	4331
2017	9	428
2017	10	4631
2017	11	7544
2017	12	5673
2018	1	7269
2018	2	6728
2018	3	7211
2018	4	6939
2018	5	6873
2018	6	6167
2018	7	6292
2018	8	6512
2018	9	16
2018	10	4

- ➢ It can be observed that time period for which the data is given is 25 months.
- ➢ There is a increasing trend in orders , trend sustains during 2018. There a slight fall we can observe during october 2017 following with a great hike in november month and again a fall in end of december 2017 and january 2018.



2. What time do brazilian customers tend to buy (dawn, morning, afternoon or night):

➢ As no time mentioned , i am considering :
   i. 0-6 as dawn time
   ii. 6-12 as morning time
   iii. 12-18 as afternoon
   iv. And 18 to 00 as night



```
with cte as(
SELECT
CASE WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 5.59 THEN 'DAWN'
     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 6 AND 11.59 THEN 'MORNING'
     WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 12 AND 17.59 THEN 'AFTERNOON'
     ELSE  'NIGHT' END AS TIME,
order_id

FROM `target.orders`
)
SELECT TIME,COUNT(ORDER_ID) ORDER_COUNT FROM CTE
GROUP BY 1
```

Query results

JOB INFORMATION   RESULTS   JSON   EXECUTION DETAILS   EXECUTION GRAPH PREVIEW

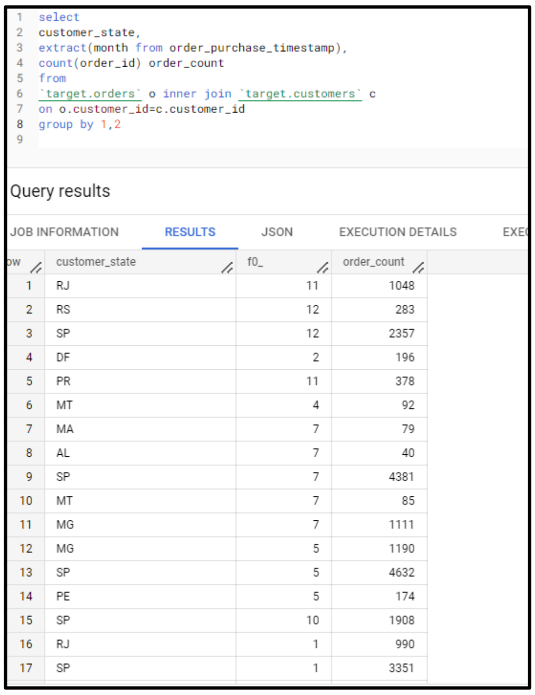
Row	TIME	ORDER_COUNT
1	MORNING	22240
2	DAWN	4740
3	AFTERNOON	38361
4	NIGHT	34100

➢ There are very a smaller number of orders in dawn time.

3. Evolution of e-commerce orders in the brazil region

1. Get month on month orders by states:

➢ To get month on month data for orders , orders table can be used and to fetch state data customer table can be used.



```sql
select
customer_state,
extract(month from order_purchase_timestamp),
count(order_id) order_count
from
`target.orders` o inner join `target.customers` c
on o.customer_id=c.customer_id
group by 1,2
```

Query results

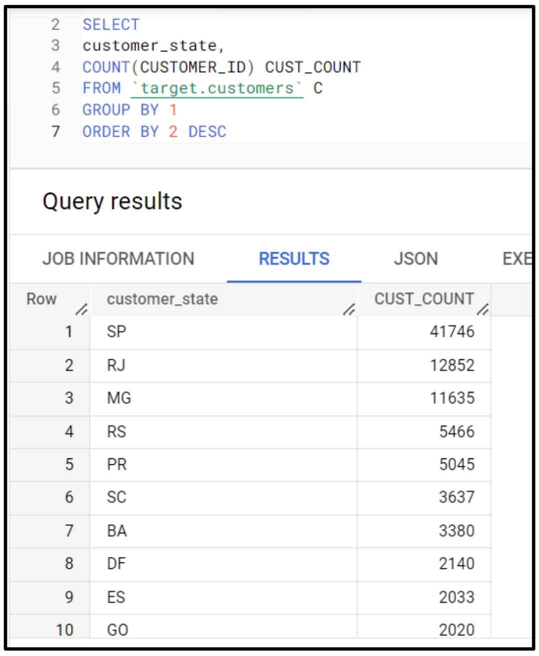
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

ow	customer_state	f0_	order_count
1	RJ	11	1048
2	RS	12	283
3	SP	12	2357
4	DF	2	196
5	PR	11	378
6	MT	4	92
7	MA	7	79
8	AL	7	40
9	SP	7	4381
10	MT	7	85
11	MG	7	1111
12	MG	5	1190
13	SP	5	4632
14	PE	5	174
15	SP	10	1908
16	RJ	1	990
17	SP	1	3351

➢ We can observe that the state 'SP' has maximum number of orders placed.

3.Distribution of customers across the states in Brazil:

➢ State wise customer number count can be fetched from customer table directly:



```sql
SELECT
customer_state,
COUNT(CUSTOMER_ID) CUST_COUNT
FROM `target.customers` C
GROUP BY 1
ORDER BY 2 DESC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXE

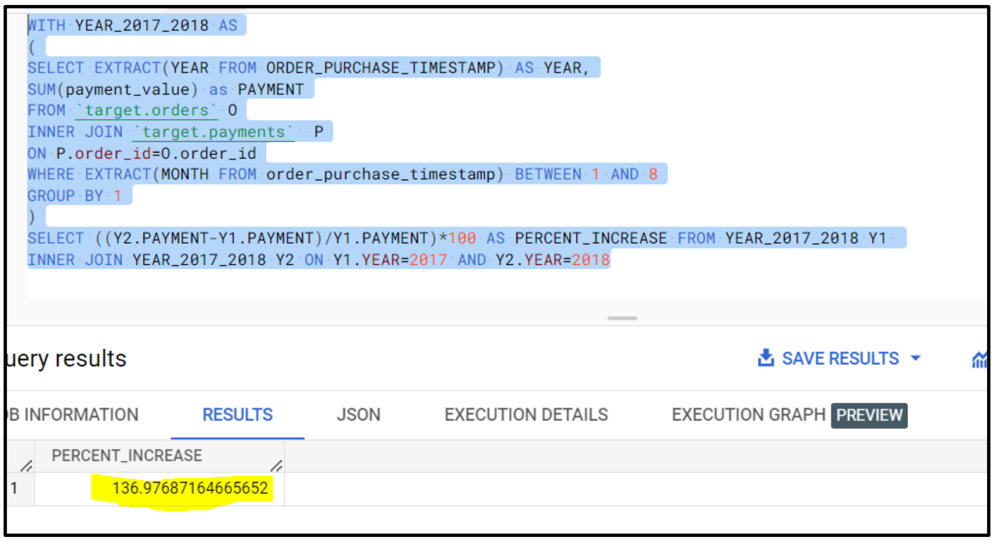
Row	customer_state	CUST_COUNT
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

➢ The state 'SP' has maximum number of customer i.e. 41746 customers

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):

- Used below query to check percent increase from 2017 to 2018 , i have not used filter on year as we have to take 1st month to 8th month and these month's data for 2016 is not available in dataset so those will not be included in result:

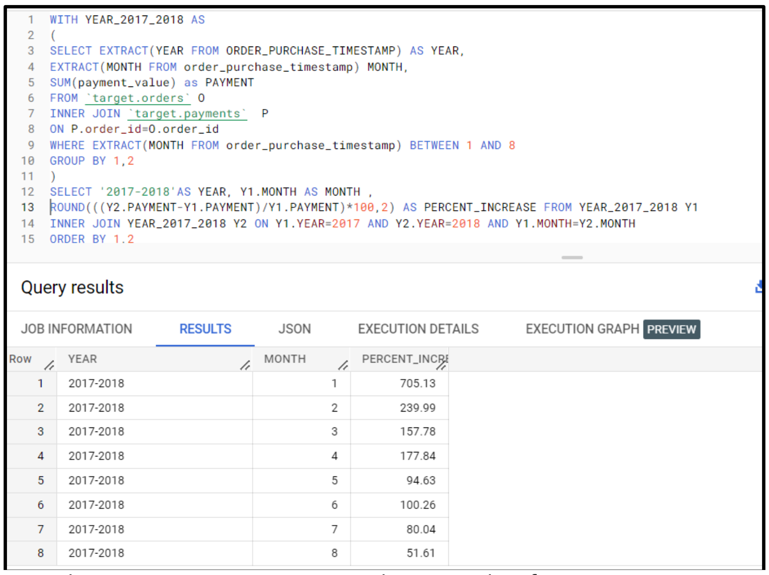


```sql
WITH YEAR_2017_2018 AS
(
SELECT EXTRACT(YEAR FROM ORDER_PURCHASE_TIMESTAMP) AS YEAR,
SUM(payment_value) as PAYMENT
FROM `target.orders` O
INNER JOIN `target.payments`  P
ON P.order_id=O.order_id
WHERE EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY 1
)
SELECT ((Y2.PAYMENT-Y1.PAYMENT)/Y1.PAYMENT)*100 AS PERCENT_INCREASE FROM YEAR_2017_2018 Y1
INNER JOIN YEAR_2017_2018 Y2 ON Y1.YEAR=2017 AND Y2.YEAR=2018
```

uery results

	PERCENT_INCREASE
1	136.97687164665652

- There is 136.97 % increase from 2017 to 2018
- Lets check month-wise % increase:



```sql
WITH YEAR_2017_2018 AS
(
SELECT EXTRACT(YEAR FROM ORDER_PURCHASE_TIMESTAMP) AS YEAR,
EXTRACT(MONTH FROM order_purchase_timestamp) MONTH,
SUM(payment_value) as PAYMENT
FROM `target.orders` O
INNER JOIN `target.payments`  P
ON P.order_id=O.order_id
WHERE EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY 1,2
)
SELECT '2017-2018'AS YEAR, Y1.MONTH AS MONTH ,
ROUND(((Y2.PAYMENT-Y1.PAYMENT)/Y1.PAYMENT)*100,2) AS PERCENT_INCREASE FROM YEAR_2017_2018 Y1
INNER JOIN YEAR_2017_2018 Y2 ON Y1.YEAR=2017 AND Y2.YEAR=2018 AND Y1.MONTH=Y2.MONTH
ORDER BY 1.2
```

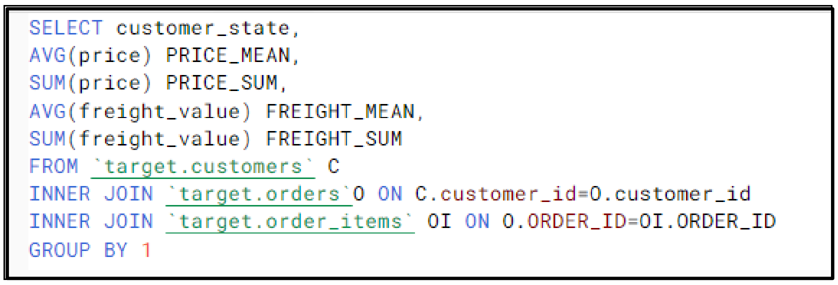
Query results

Row	YEAR	MONTH	PERCENT_INCRE
1	2017-2018	1	705.13
2	2017-2018	2	239.99
3	2017-2018	3	157.78
4	2017-2018	4	177.84
5	2017-2018	5	94.63
6	2017-2018	6	100.26
7	2017-2018	7	80.04
8	2017-2018	8	51.61

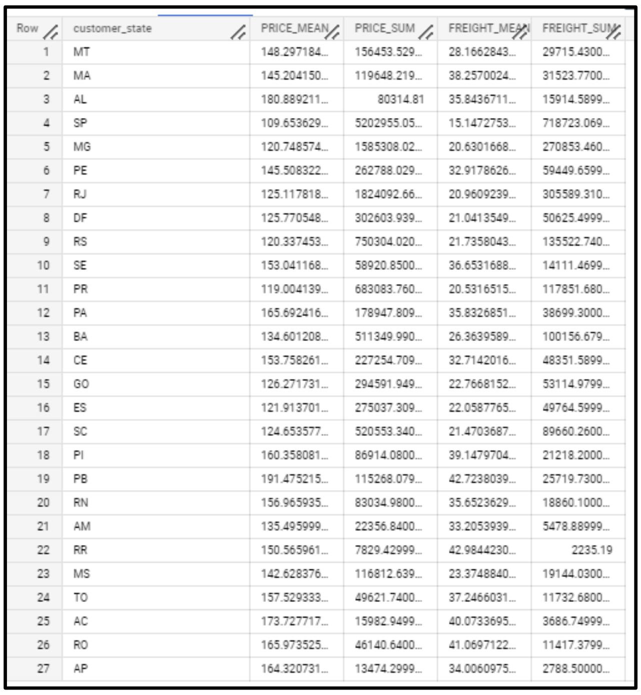
- There is huge change in payment % in the month of january.
- And lowest in month of august.
- It can be observed than percentage is almost decreasing from january to august.

3. Mean & sum of price and freight value by customer state:

- Price and freight value are available in order_items table, we can join with order and customer table to find customer state to get state data:

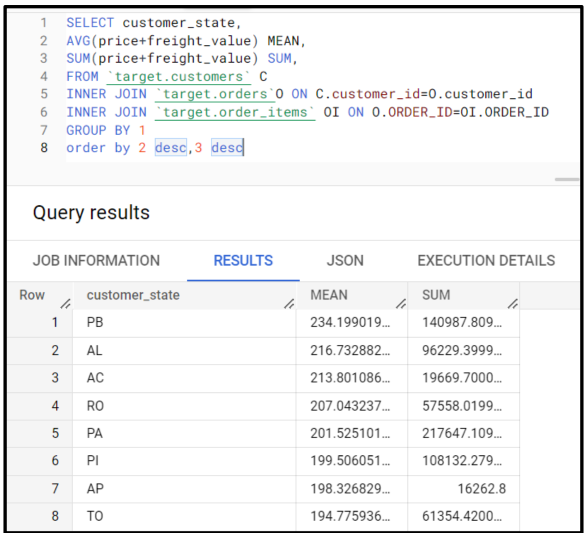


```sql
SELECT customer_state,
AVG(price) PRICE_MEAN,
SUM(price) PRICE_SUM,
AVG(freight_value) FREIGHT_MEAN,
SUM(freight_value) FREIGHT_SUM
FROM `target.customers` C
INNER JOIN `target.orders`O ON C.customer_id=O.customer_id
INNER JOIN `target.order_items` OI ON O.ORDER_ID=OI.ORDER_ID
GROUP BY 1
```



Row	customer_state	PRICE_MEAN	PRICE_SUM	FREIGHT_MEAN	FREIGHT_SUM
1	MT	148.297184...	156453.529...	28.1662843...	29715.4300...
2	MA	145.204150...	119648.219...	38.2570024...	31523.7700...
3	AL	180.889211...	80314.81	35.8436711...	15914.5899...
4	SP	109.653629...	5202955.05...	15.1472753...	718723.069...
5	MG	120.748574...	1585308.02...	20.6301668...	270853.460...
6	PE	145.508322...	262788.029...	32.9178626...	59449.6599...
7	RJ	125.117818...	1824092.66...	20.9609239...	305589.310...
8	DF	125.770548...	302603.939...	21.0413549...	50625.4999...
9	RS	120.337453...	750304.020...	21.7358043...	135522.740...
10	SE	153.041168...	58920.8500...	36.6531688...	14111.4699...
11	PR	119.004139...	683083.760...	20.5316515...	117851.680...
12	PA	165.692416...	178947.809...	35.8326851...	38699.3000...
13	BA	134.601208...	511349.990...	26.3639589...	100156.679...
14	CE	153.758261...	227254.709...	32.7142016...	48351.5899...
15	GO	126.271731...	294591.949...	22.7668152...	53114.9799...
16	ES	121.913701...	275037.309...	22.0587765...	49764.5999...
17	SC	124.653577...	520553.340...	21.4703687...	89660.2600...
18	PI	160.358081...	86914.0800...	39.1479704...	21218.2000...
19	PB	191.475215...	115268.079...	42.7238039...	25719.7300...
20	RN	156.965935...	83034.9800...	35.6523629...	18860.1000...
21	AM	135.495999...	22356.8400...	33.2053939...	5478.88999...
22	RR	150.565961...	7829.42999...	42.9844230...	2235.19
23	MS	142.628376...	116812.639...	23.3748840...	19144.0300...
24	TO	157.529333...	49621.7400...	37.2466031...	11732.6800...
25	AC	173.727717...	15982.9499...	40.0733695...	3686.74999...
26	RO	165.973525...	46140.6400...	41.0697122...	11417.3799...
27	AP	164.320731...	13474.2999...	34.0060975...	2788.50000...

➢ If we combine price and freight value we can observe that pb has highest mean (234.20) and pb has highest sum (5921678.1) :



```sql
SELECT customer_state,
AVG(price+freight_value) MEAN,
SUM(price+freight_value) SUM,
FROM `target.customers` C
INNER JOIN `target.orders`O ON C.customer_id=O.customer_id
INNER JOIN `target.order_items` OI ON O.ORDER_ID=OI.ORDER_ID
GROUP BY 1
order by 2 desc,3 desc
```

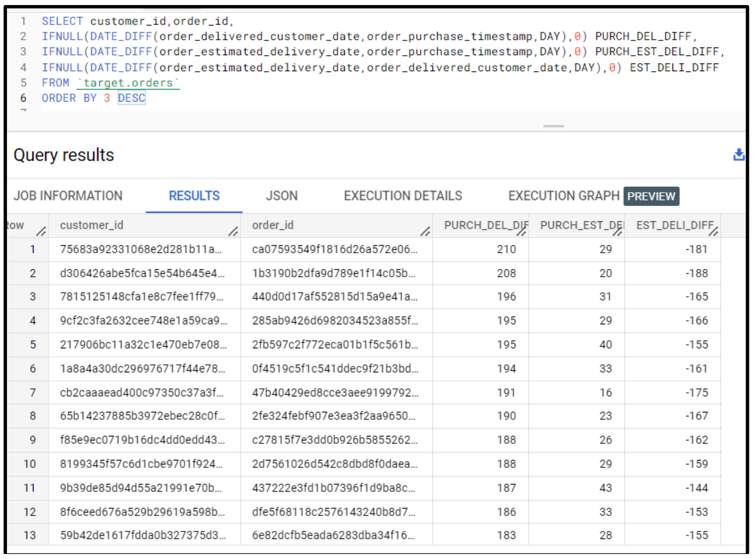
Query results

Row	customer_state	MEAN	SUM
1	PB	234.199019...	140987.809...
2	AL	216.732882...	96229.3999...
3	AC	213.801086...	19669.7000...
4	RO	207.043237...	57558.0199...
5	PA	201.525101...	217647.109...
6	PI	199.506051...	108132.279...
7	AP	198.326829...	16262.8
8	TO	194.775936...	61354.4200...

5. Analysis on sales, freight and delivery time:

1. Calculate days between purchasing, delivering and estimated delivery:

- ➢ Purchase date, delivery date and estimated delivery dates are available in orders table and they can be directly accessed to find out the differences:

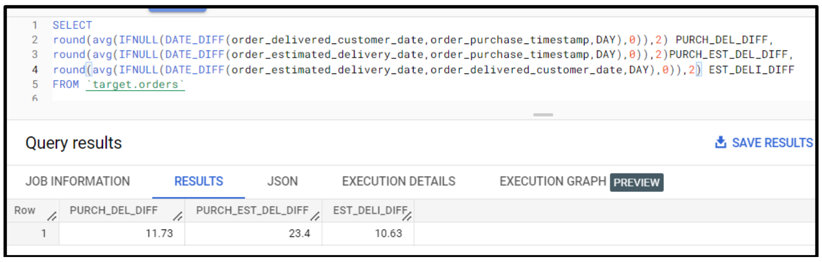


```sql
SELECT customer_id,order_id,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0) PURCH_DEL_DIFF,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp,DAY),0) PURCH_EST_DEL_DIFF,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0) EST_DELI_DIFF
FROM `target.orders`
ORDER BY 3 DESC
```

Query results

Row	customer_id	order_id	PURCH_DEL_DIF	PURCH_EST_DE	EST_DELI_DIFF
1	75683a92331068e2d281b11a…	ca07593549f1816d26a572e06…	210	29	-181
2	d306426abe5fca15e54b645e4…	1b3190b2dfa9d789e1f14c05b…	208	20	-188
3	7815125148cfa1e8c7fee1ff79…	440d0d17af552815d15a9e41a…	196	31	-165
4	9cf2c3fa2632cee748e1a59ca9…	285ab9426d6982034523a855f…	195	29	-166
5	217906bc11a32c1e470eb7e08…	2fb597c2f772eca01b1f5c561b…	195	40	-155
6	1a8a4a30dc296976717f44e78…	0f4519c5f1c541ddec9f21b3bd…	194	33	-161
7	cb2caaaead400c97350c37a3f…	47b40429ed8cce3aee9199792…	191	16	-175
8	65b14237885b3972ebec28c0f…	2fe324febf907e3ea3f2aa9650…	190	23	-167
9	f85e9ec0719b16dc4dd0edd43…	c27815f7e3dd0b926b5855262…	188	26	-162
10	8199345f57c6d1cbe9701f924…	2d7561026d542c8dbd8f0daea…	188	29	-159
11	9b39de85d94d55a21991e70b…	437222e3fd1b07396f1d9ba8c…	187	43	-144
12	8f6ceed676a529b29619a598b…	dfe5f68118c2576143240b8d7…	186	33	-153
13	59b42de1617fdda0b327375d3…	6e82dcfb5eada6283dba34f16…	183	28	-155

- ➢ Average difference among dates are:



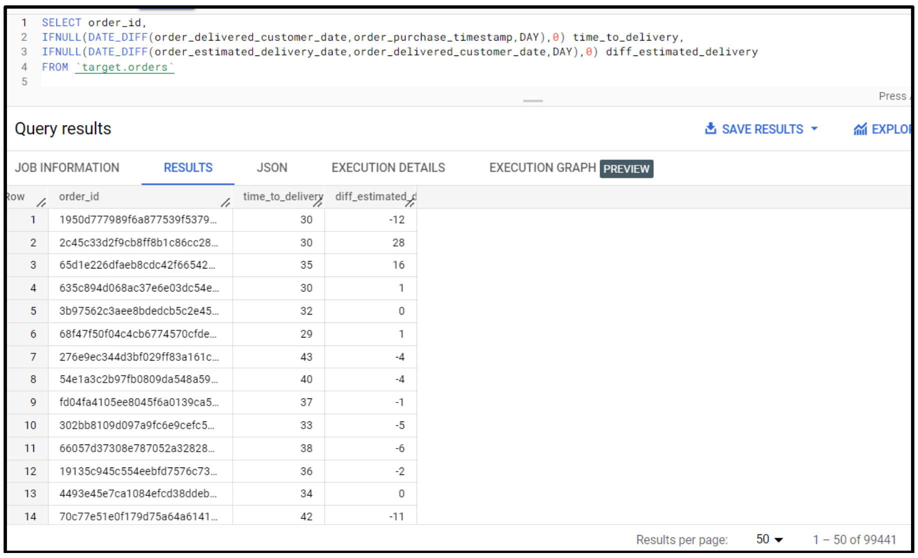
```sql
SELECT
round(avg(IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0)),2) PURCH_DEL_DIFF,
round(avg(IFNULL(DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp,DAY),0)),2)PURCH_EST_DEL_DIFF,
round(avg(IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0)),2) EST_DELI_DIFF
FROM `target.orders`
```

Query results

Row	PURCH_DEL_DIFF	PURCH_EST_DEL_DIFF	EST_DELI_DIFF
1	11.73	23.4	10.63

2. Find time_to_delivery & diff_estimated_delivery:

- ➢ Purchase date, delivery date and estimated delivery dates from orders table can be used to find out time_to_delivery and diff_estimated_delivery for each orders:



```sql
SELECT order_id,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0) time_to_delivery,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0) diff_estimated_delivery
FROM `target.orders`
```

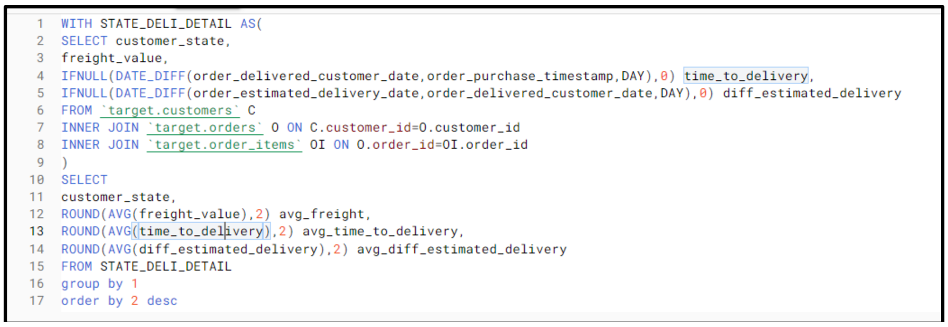
Query results

Row	order_id	time_to_delivery	diff_estimated_d
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
11	66057d37308e787052a32828...	38	-6
12	19135c945c554eebfd7576c73...	36	-2
13	4493e45e7ca1084efcd38ddeb...	34	0
14	70c77e51e0f179d75a64a6141...	42	-11

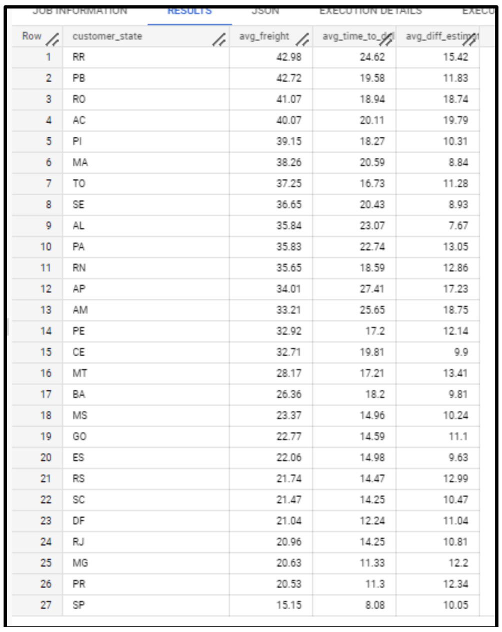
Results per page: 50 1 – 50 of 99441

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

- ➢ As we have already calculated time to delivery and diff_estimated_delivery , we can join this data with order_items table to get freight value.
- ➢ And then this data can be joined with customers table to fetch customers states as following:



```sql
WITH STATE_DELI_DETAIL AS(
SELECT customer_state,
freight_value,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0) time_to_delivery,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0) diff_estimated_delivery
FROM `target.customers` C
INNER JOIN `target.orders` O ON C.customer_id=O.customer_id
INNER JOIN `target.order_items` OI ON O.order_id=OI.order_id
)
SELECT
customer_state,
ROUND(AVG(freight_value),2) avg_freight,
ROUND(AVG(time_to_delivery),2) avg_time_to_delivery,
ROUND(AVG(diff_estimated_delivery),2) avg_diff_estimated_delivery
FROM STATE_DELI_DETAIL
group by 1
order by 2 desc
```



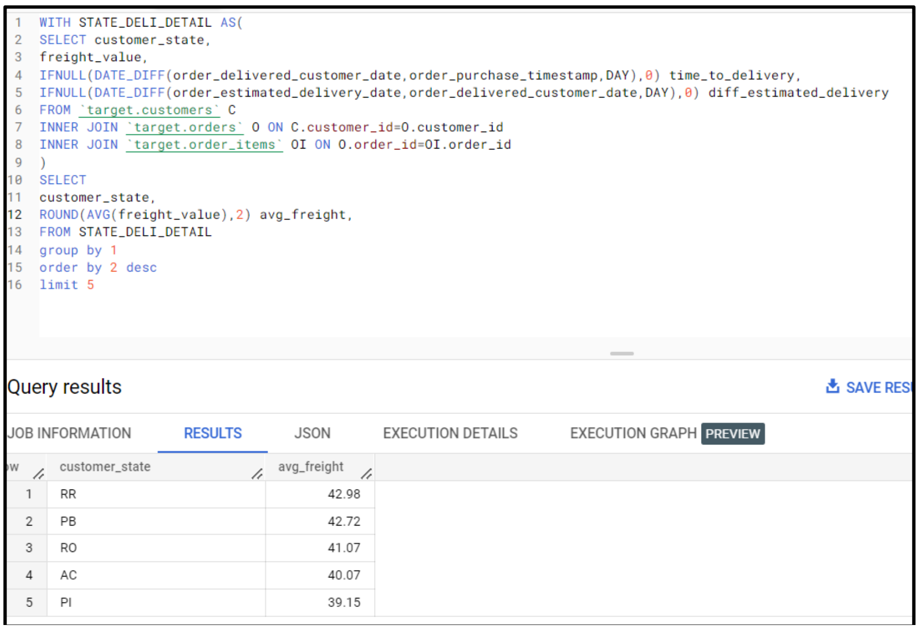
Row	customer_state	avg_freight	avg_time_to_del	avg_diff_estimat
1	RR	42.98	24.62	15.42
2	PB	42.72	19.58	11.83
3	RO	41.07	18.94	18.74
4	AC	40.07	20.11	19.79
5	PI	39.15	18.27	10.31
6	MA	38.26	20.59	8.84
7	TO	37.25	16.73	11.28
8	SE	36.65	20.43	8.93
9	AL	35.84	23.07	7.67
10	PA	35.83	22.74	13.05
11	RN	35.65	18.59	12.86
12	AP	34.01	27.41	17.23
13	AM	33.21	25.65	18.75
14	PE	32.92	17.2	12.14
15	CE	32.71	19.81	9.9
16	MT	28.17	17.21	13.41
17	BA	26.36	18.2	9.81
18	MS	23.37	14.96	10.24
19	GO	22.77	14.59	11.1
20	ES	22.06	14.98	9.63
21	RS	21.74	14.47	12.99
22	SC	21.47	14.25	10.47
23	DF	21.04	12.24	11.04
24	RJ	20.96	14.25	10.81
25	MG	20.63	11.33	12.2
26	PR	20.53	11.3	12.34
27	SP	15.15	8.08	10.05

4. Sort the data to get the following:

- ➢ Data can be sorted based on various columns to get below insights (5th and 6th points)

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

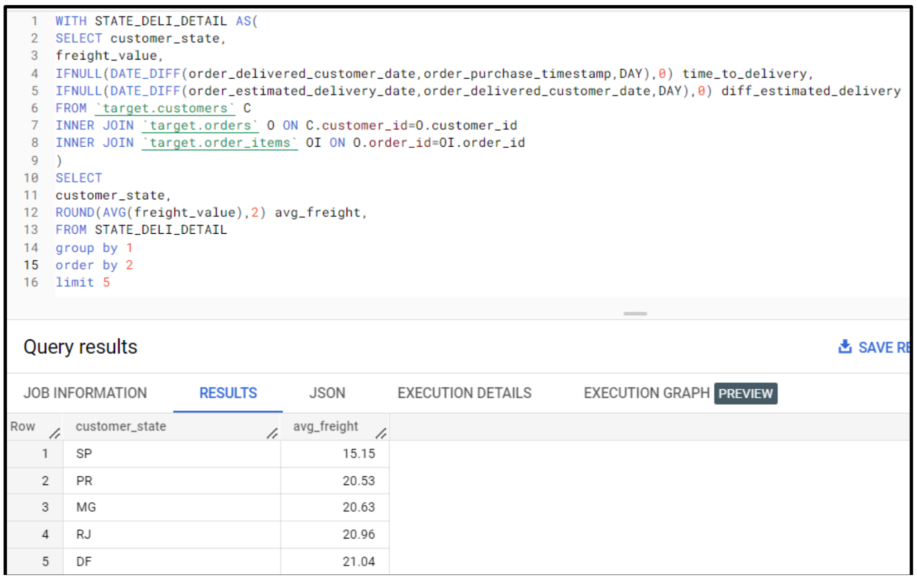
- ➢ Top 5 states with highest average freight value:



```
WITH STATE_DELI_DETAIL AS(
SELECT customer_state,
freight_value,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0) time_to_delivery,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0) diff_estimated_delivery
FROM `target.customers` C
INNER JOIN `target.orders` O ON C.customer_id=O.customer_id
INNER JOIN `target.order_items` OI ON O.order_id=OI.order_id
)
SELECT
customer_state,
ROUND(AVG(freight_value),2) avg_freight,
FROM STATE_DELI_DETAIL
group by 1
order by 2 desc
limit 5
```

w	customer_state	avg_freight
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

- ➢ Top 5 states with lowest average freight value:

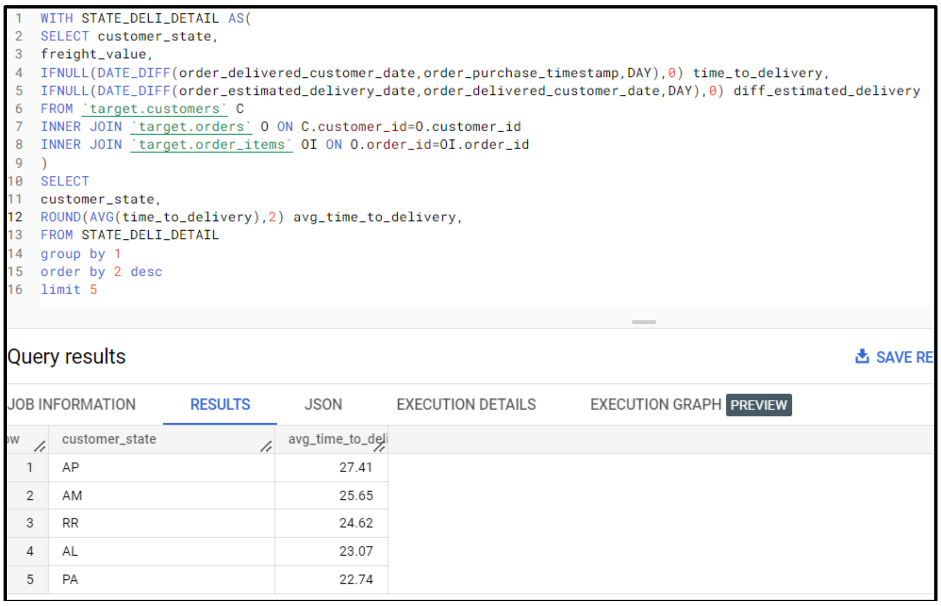


```
WITH STATE_DELI_DETAIL AS(
SELECT customer_state,
freight_value,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0) time_to_delivery,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0) diff_estimated_delivery
FROM `target.customers` C
INNER JOIN `target.orders` O ON C.customer_id=O.customer_id
INNER JOIN `target.order_items` OI ON O.order_id=OI.order_id
)
SELECT
customer_state,
ROUND(AVG(freight_value),2) avg_freight,
FROM STATE_DELI_DETAIL
group by 1
order by 2
limit 5
```

Row	customer_state	avg_freight
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

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

➢ Top 5 states with highest average delivery time:

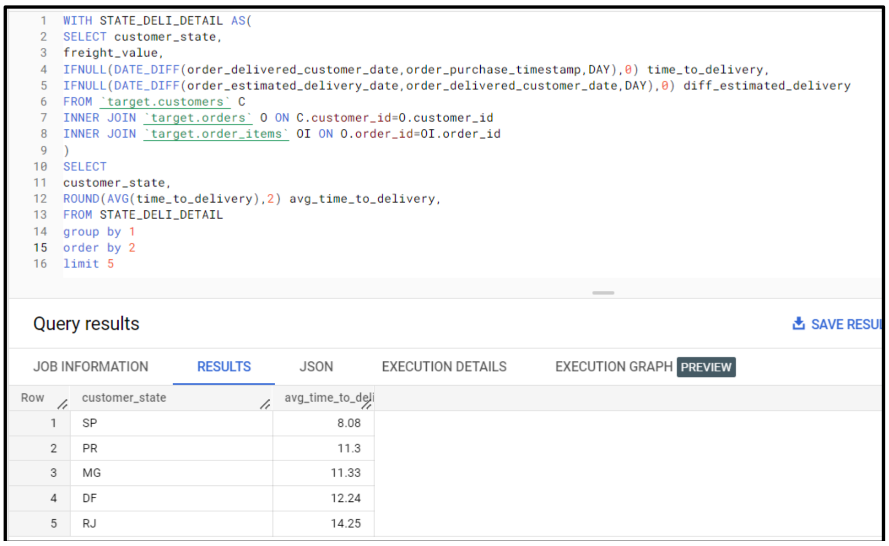


```sql
WITH STATE_DELI_DETAIL AS(
SELECT customer_state,
freight_value,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0) time_to_delivery,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0) diff_estimated_delivery
FROM `target.customers` C
INNER JOIN `target.orders` O ON C.customer_id=O.customer_id
INNER JOIN `target.order_items` OI ON O.order_id=OI.order_id
)
SELECT
customer_state,
ROUND(AVG(time_to_delivery),2) avg_time_to_delivery,
FROM STATE_DELI_DETAIL
group by 1
order by 2 desc
limit 5
```

Query results

ow	customer_state	avg_time_to_deli
1	AP	27.41
2	AM	25.65
3	RR	24.62
4	AL	23.07
5	PA	22.74

➢ Top 5 states with lowest average delivery time:



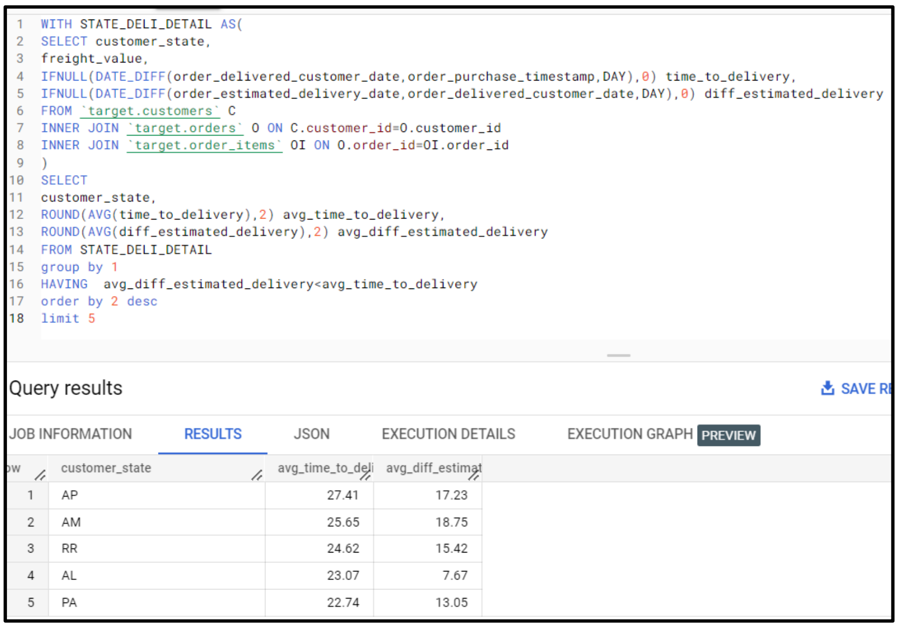
```sql
WITH STATE_DELI_DETAIL AS(
SELECT customer_state,
freight_value,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0) time_to_delivery,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0) diff_estimated_delivery
FROM `target.customers` C
INNER JOIN `target.orders` O ON C.customer_id=O.customer_id
INNER JOIN `target.order_items` OI ON O.order_id=OI.order_id
)
SELECT
customer_state,
ROUND(AVG(time_to_delivery),2) avg_time_to_delivery,
FROM STATE_DELI_DETAIL
group by 1
order by 2
limit 5
```

Query results

Row	customer_state	avg_time_to_deli
1	SP	8.08
2	PR	11.3
3	MG	11.33
4	DF	12.24
5	RJ	14.25

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

➢ We can filter our data having average delivery time is more then expected delivery time difference and then we will select top 5 among this data set:

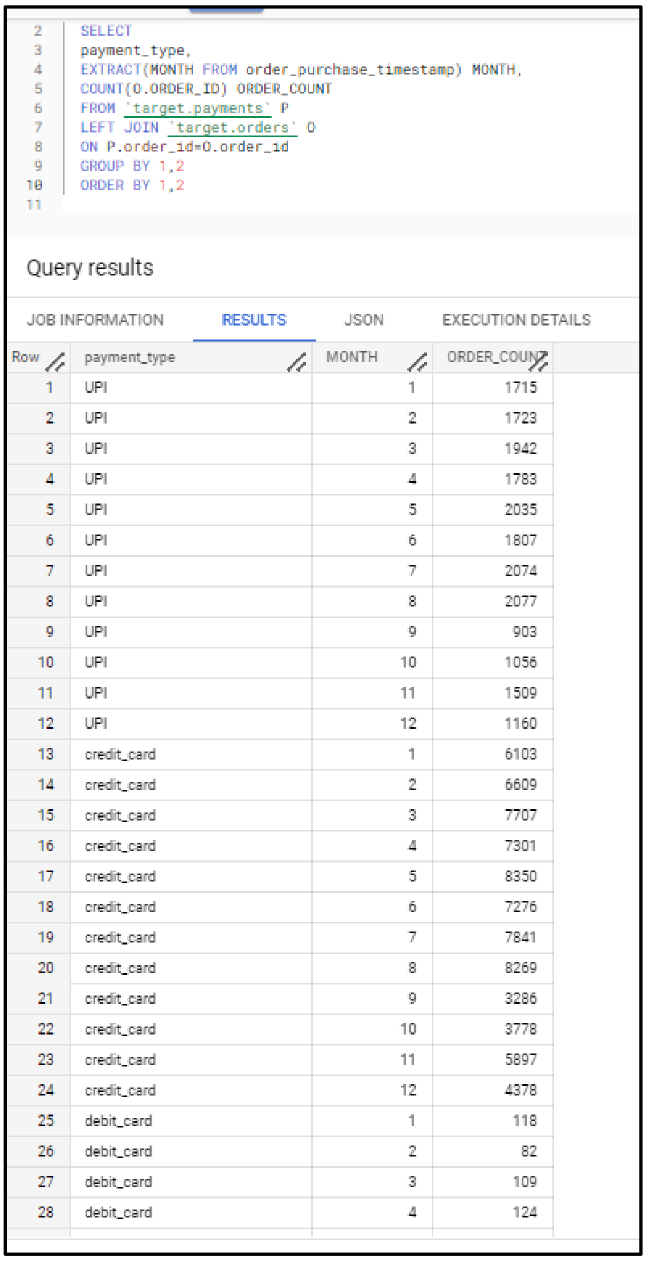


```
WITH STATE_DELI_DETAIL AS(
SELECT customer_state,
freight_value,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY),0) time_to_delivery,
IFNULL(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY),0) diff_estimated_delivery
FROM `target.customers` C
INNER JOIN `target.orders` O ON C.customer_id=O.customer_id
INNER JOIN `target.order_items` OI ON O.order_id=OI.order_id
)
SELECT
customer_state,
ROUND(AVG(time_to_delivery),2) avg_time_to_delivery,
ROUND(AVG(diff_estimated_delivery),2) avg_diff_estimated_delivery
FROM STATE_DELI_DETAIL
group by 1
HAVING  avg_diff_estimated_delivery<avg_time_to_delivery
order by 2 desc
limit 5
```

ow	customer_state	avg_time_to_deli	avg_diff_estimat
1	AP	27.41	17.23
2	AM	25.65	18.75
3	RR	24.62	15.42
4	AL	23.07	7.67
5	PA	22.74	13.05

6. Payment type analysis:

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



```
SELECT
payment_type,
EXTRACT(MONTH FROM order_purchase_timestamp) MONTH,
COUNT(O.ORDER_ID) ORDER_COUNT
FROM `target.payments` P
LEFT JOIN `target.orders` O
ON P.order_id=O.order_id
GROUP BY 1,2
ORDER BY 1,2
```

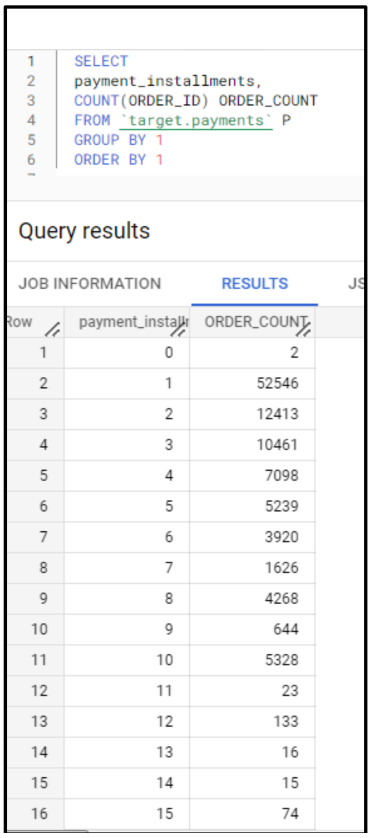
Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	payment_type	MONTH	ORDER_COUNT
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807
7	UPI	7	2074
8	UPI	8	2077
9	UPI	9	903
10	UPI	10	1056
11	UPI	11	1509
12	UPI	12	1160
13	credit_card	1	6103
14	credit_card	2	6609
15	credit_card	3	7707
16	credit_card	4	7301
17	credit_card	5	8350
18	credit_card	6	7276
19	credit_card	7	7841
20	credit_card	8	8269
21	credit_card	9	3286
22	credit_card	10	3778
23	credit_card	11	5897
24	credit_card	12	4378
25	debit_card	1	118
26	debit_card	2	82
27	debit_card	3	109
28	debit_card	4	124

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

- ➢ This can be achieved directly using payments table, i have used below query to get the insight:



```
SELECT
payment_installments,
COUNT(ORDER_ID) ORDER_COUNT
FROM `target.payments` P
GROUP BY 1
ORDER BY 1
```

Query results

JOB INFORMATION RESULTS JS

Row	payment_install	ORDER_COUNT
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
11	10	5328
12	11	23
13	12	133
14	13	16
15	14	15
16	15	74

7. Actionable insights

1. Overview of target data:

- We have 99,441 customers of data available .
- 14994 different locations of customers
- Customers are from different 4119 cities and 27 states from brazil.
- From total 99441 orders , 1107 are shipped ,625 were canceled, 96478 are delivered.
- 68% customers are from southeast brazil , 14% are from south brazil and rest are other other regions of brazil .
- Total 3095 sellers who are from 611 different cities and 23 states in brazil and from 2246 different areas as per zip-code data.
- São paulo state has the highest numbers of sellers in country.
- Time period for which the data is given is 25 months, from sept 2016 -oct 2018

2. Performance of the company:

- Compare to 2017 , no of orders has increased in 2018.
- Average number of order are higher during november month , september and october month average orders are comparatively low , in may and july and august have higher average orders compare to other months.
- Tuesday, monday and wednesdays have relatively higher number of orders.

- There is an increasing trend in orders, trend sustains during 2018. There is a slight fall we can observe during october 2017 following with a great hike in november month and again a fall in end of december 2017 and January 2018.
- We can observe there is 815% growth increased in terms of orders and 707% growth increment in terms of revenue in January from 2017 to 2018.
- Growth rate for July and august in 2017 to 2018 is relatively extremely low!
- 2017-february, 2017-march,2017-november were the highest growing sale month compared to the previous month.
- In products data, total 32951 different products available in target with 73 different product category. Health and beauty, watches present, bed table bath, sport leisure, computer accessories, furniture decoration, housewares, automotive are some of the top selling product categories.
- Health and beauty products are top selling having highest orders.
- Pcs and musical instruments category have relatively less number of products, but contributes in a high revenue.

3. Delivery time analysis:

- After purchasing, the average time for approving the order by seller is 0.26 days and median time is 0, means within a day.
- Average time taken for a carrier to start the delivery is 2 and a half day, average time taken to complete delivery is 12 days, average estimated time delivery is 23 days.
- There is a positive correlation between freight value and delivery time.
- States são paulo, paraná, minas gerais, distrito federal ,santa catarina and rio de janeiro are some of the states having faster delivery time relatively.

4. Freight value and delivery time relation:

- Average estimated delivery time and delivery time have a positive correlation with avg freight value. States, where orders are taking more delivery time have more freight value. We can say delivery time is proportional to freight value or vice versa.

5. Region and state vise analysis:

- São paulo ,rio de janeiro , minas gerais ,rio grande do sul and paraná are top 5 highest orders states and also generating highest revenue. More than 80% of orders are coming from south, southeast and northeast brazil. 90% of the revenue is coming from south, southeast and northeast brazil.

6. Peak time of the day:

- There are very less number of orders in dawn time.
- Customers tends to buy mostly in night.

7. Payment analysis:

- Most common mode of payment is using credit card for the payment.
- Number of installments are mostly in the range 0-3.
- Less no of orders can be seen with higher no of installments.

8. Recommendations :

1. Lesser Delivery time: From the distribution and statistical analysis we can observe the average time to complete the delivery is 12 days. Which should be reduced to at least half, as due to high competition in e-commerce market, it is vital to do so.

2. Region to be focused: If we look at top states where delivery is really slow compared to estimated date, they are all from north Brazil region. Delivering faster in the north states may create and increase new customers and revenue from north.

3. Increasing network in north Brazil: having small towns can help increase the customer base. As north Brazil has the worlds largest river and most extensive rain forest, must be a good travel destination, introducing necessary survival/ camping/adventure products can help increase revenue and order from northern region . Top selling items are between 10-100 dollars, introducing new different more products from top selling categories can increase revenue more.

4. Providing Discounts: It was observed an increasing trend in revenue and orders over time , yet during October and January sales are decreasing probably after festival sales. Introducing possible discount on not so running product can help sell more products during those low going months.

5. Introducing Offers: Looking at the Analysis we can see that mostly customers tend to shop in Night and reason behind this can be their free time during night, introducing offers during that time can attract more number of customers increasing overall profit of the company.

6. Adding more Sellers: In order to provide diverse range of products more sellers should be added to platform.

7. Providing credit card benefits: As this can be observed that credit card is the most used payment type, providing benefits on different cards can be used to attract customers.

8. Increasing Area of service: There are total 8011 cities available in given data but customers from only 4119 are placing order, which is almost half of total number of cities, finding out reason for same can help to improve area of service.

9. Figuring out losses: Despite having maximum number of customer in SP state. It has low average revenue. Finding out reason and eliminating same can help in Company's performance.