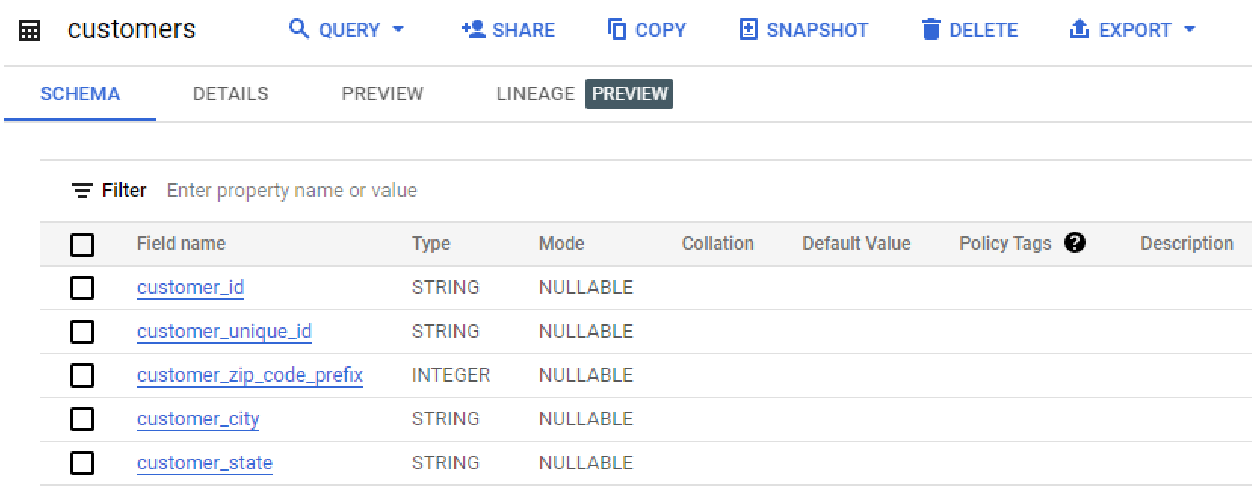
Case Study

Part 1: Target DB details

Total Number of tables: 8

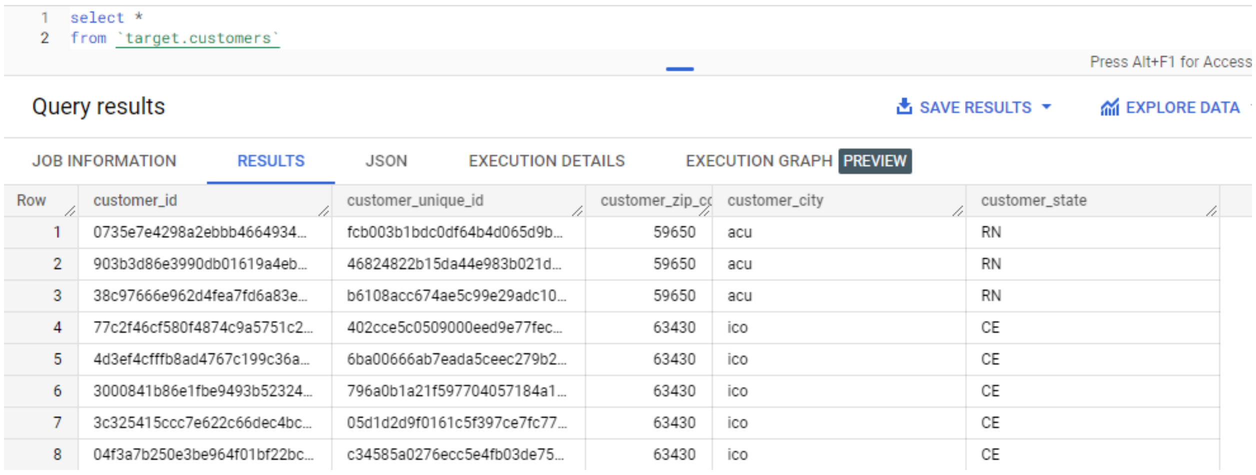
1. **Customers**:

   Customer master table, holding the demographic info of a customer



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
customer_id	STRING	NULLABLE				
customer_unique_id	STRING	NULLABLE				
customer_zip_code_prefix	INTEGER	NULLABLE				
customer_city	STRING	NULLABLE				
customer_state	STRING	NULLABLE				

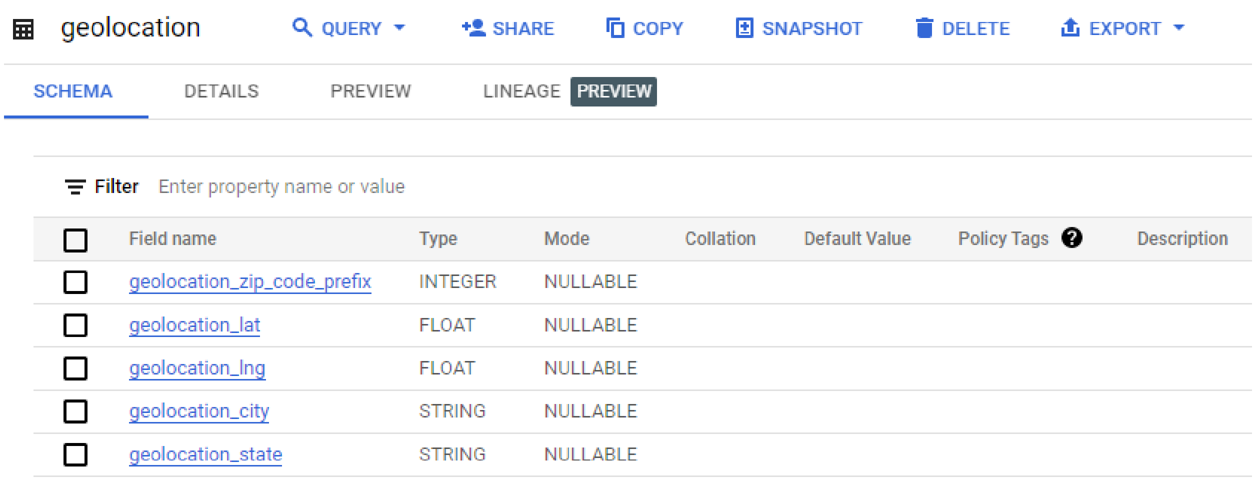
Features	**Description**
customer_id	Id of the consumer who made the purchase.
customer_unique_id	Unique Id of the consumer.
customer_zip_code_prefix	Zip Code of the location of the consumer.
customer_city	Name of the City from where order is made.
customer_state	State Code from where order is made(Ex- sao paulo-SP).



```sql
select *
from `target.customers`
```

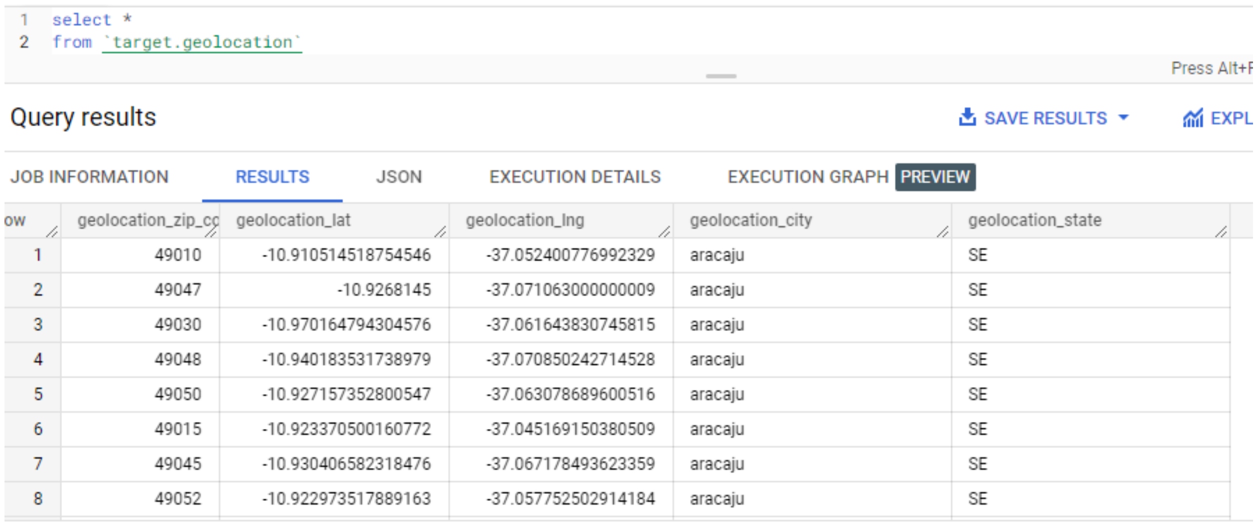
Row	customer_id	customer_unique_id	customer_zip_cc	customer_city	customer_state
1	0735e7e4298a2ebbb4664934...	fcb003b1bdc0df64b4d065d9b...	59650	acu	RN
2	903b3d86e3990db01619a4eb...	46824822b15da44e983b021d...	59650	acu	RN
3	38c97666e962d4fea7fd6a83e...	b6108acc674ae5c99e29adc10...	59650	acu	RN
4	77c2f46cf580f4874c9a5751c2...	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

2. **Geolocation**:
   Geo-location master table containing geo coding information for every zip code



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
geolocation_zip_code_prefix	INTEGER	NULLABLE				
geolocation_lat	FLOAT	NULLABLE				
geolocation_lng	FLOAT	NULLABLE				
geolocation_city	STRING	NULLABLE				
geolocation_state	STRING	NULLABLE				

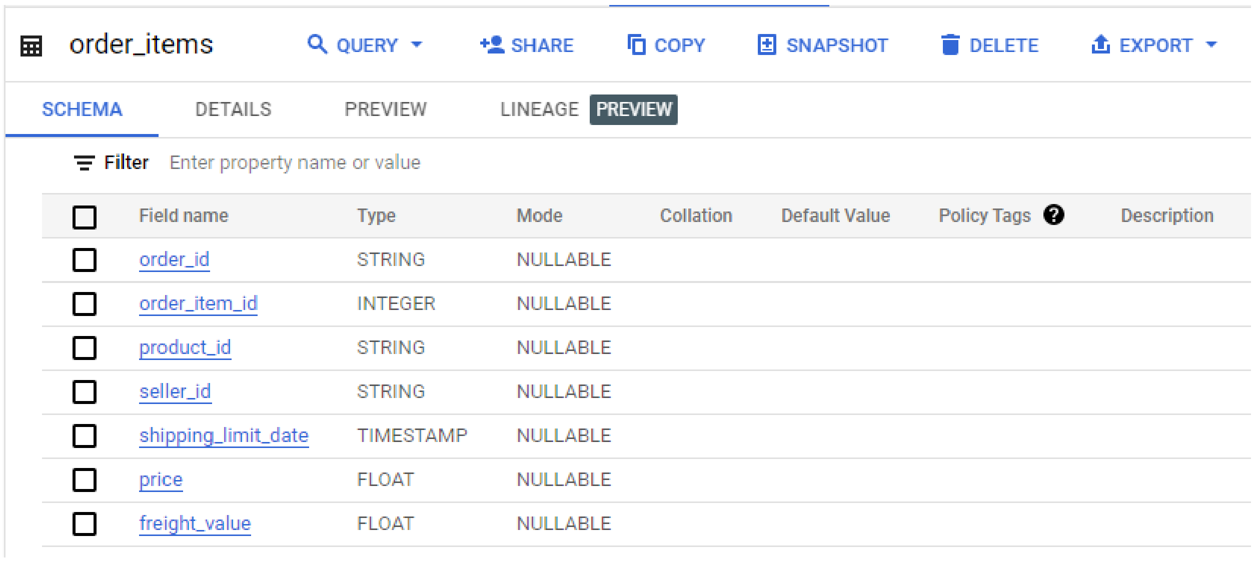
Features	Description
geolocation_zip_code_prefix	first 5 digits of zip code
geolocation_lat	latitude
geolocation_lng	longitude
geolocation_city	city name
geolocation_state	state



```
select *
from `target.geolocation`
```

ow	geolocation_zip_cc	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
1	49010	-10.910514518754546	-37.052400776992329	aracaju	SE
2	49047	-10.9268145	-37.071063000000009	aracaju	SE
3	49030	-10.970164794304576	-37.061643830745815	aracaju	SE
4	49048	-10.940183531738979	-37.070850242714528	aracaju	SE
5	49050	-10.927157352800547	-37.063078689600516	aracaju	SE
6	49015	-10.923370500160772	-37.045169150380509	aracaju	SE
7	49045	-10.930406582318476	-37.067178493623359	aracaju	SE
8	49052	-10.922973517889163	-37.057752502914184	aracaju	SE

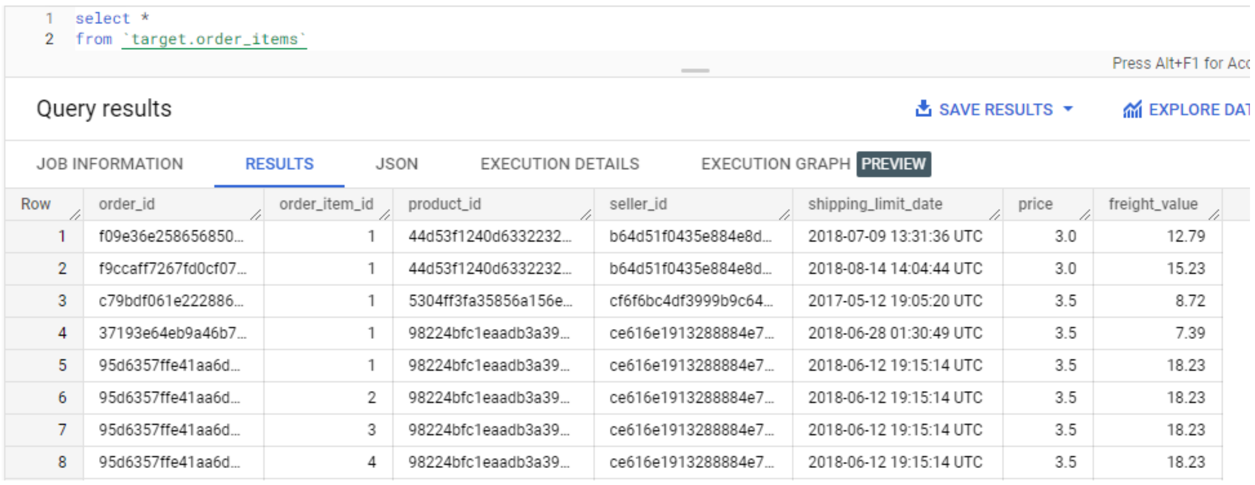
3. **order_items**:
   Order details containing order information



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
order_id	STRING	NULLABLE				
order_item_id	INTEGER	NULLABLE				
product_id	STRING	NULLABLE				
seller_id	STRING	NULLABLE				
shipping_limit_date	TIMESTAMP	NULLABLE				
price	FLOAT	NULLABLE				
freight_value	FLOAT	NULLABLE				

Features	Description
order_id	A unique id of order made by the consumers.
order_item_id	A Unique id given to each item ordered in the order.
product_id	A unique id given to each product available on the site.
seller_id	Unique Id of the seller registered in Target.

shipping_limit_date	The date before which shipping of the ordered product must be completed.
price	Actual price of the products ordered .
freight_value	Price rate at which a product is delivered from one point to another.



```
select *
from `target.order_items`
```

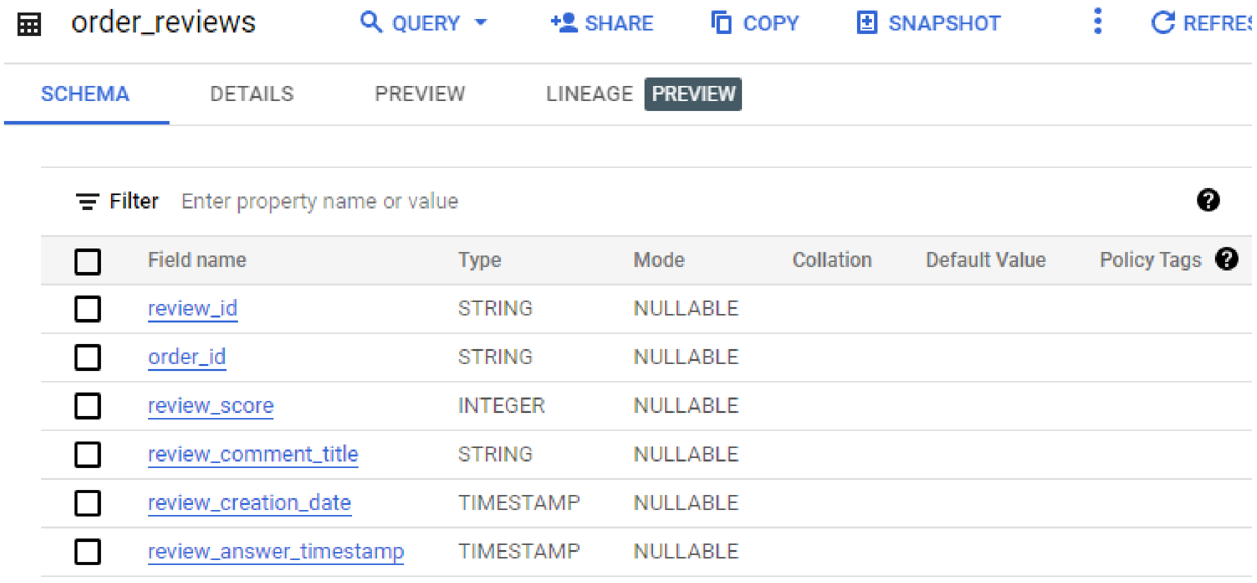
Press Alt+F1 for Acc

Query results

Row	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
1	f09e36e258656850…	1	44d53f1240d6332232…	b64d51f0435e884e8d…	2018-07-09 13:31:36 UTC	3.0	12.79
2	f9ccaff7267fd0cf07…	1	44d53f1240d6332232…	b64d51f0435e884e8d…	2018-08-14 14:04:44 UTC	3.0	15.23
3	c79bdf061e222886…	1	5304ff3fa35856a156e…	cf6f6bc4df3999b9c64…	2017-05-12 19:05:20 UTC	3.5	8.72
4	37193e64eb9a46b7…	1	98224bfc1eaadb3a39…	ce616e1913288884e7…	2018-06-28 01:30:49 UTC	3.5	7.39
5	95d6357ffe41aa6d…	1	98224bfc1eaadb3a39…	ce616e1913288884e7…	2018-06-12 19:15:14 UTC	3.5	18.23
6	95d6357ffe41aa6d…	2	98224bfc1eaadb3a39…	ce616e1913288884e7…	2018-06-12 19:15:14 UTC	3.5	18.23
7	95d6357ffe41aa6d…	3	98224bfc1eaadb3a39…	ce616e1913288884e7…	2018-06-12 19:15:14 UTC	3.5	18.23
8	95d6357ffe41aa6d…	4	98224bfc1eaadb3a39…	ce616e1913288884e7…	2018-06-12 19:15:14 UTC	3.5	18.23

4. **order_reviews**:

dataset containing the reviews on the ordered product

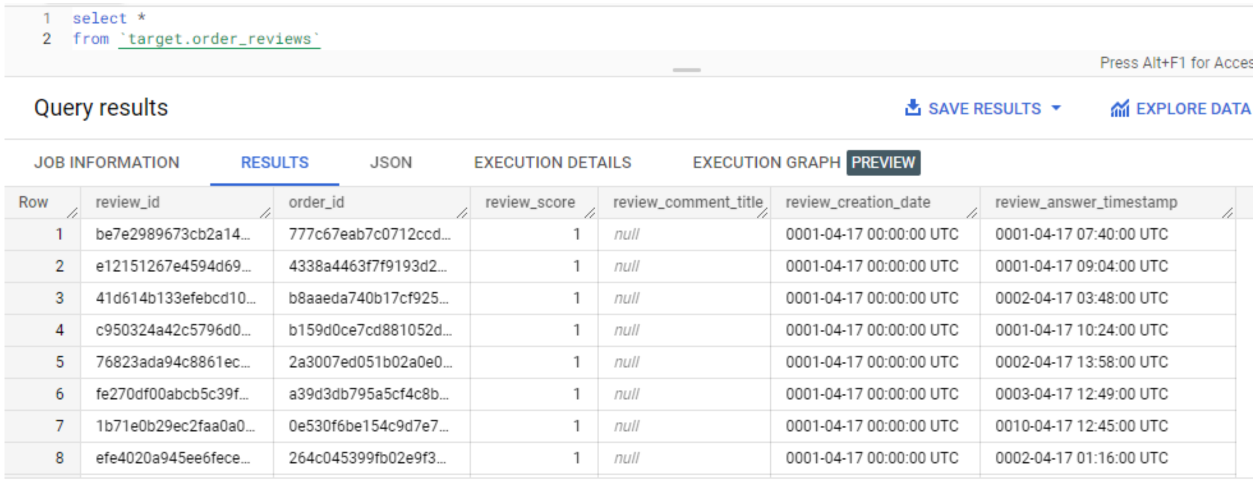


order_reviews

Field name	Type	Mode	Collation	Default Value	Policy Tags
review_id	STRING	NULLABLE			
order_id	STRING	NULLABLE			
review_score	INTEGER	NULLABLE			
review_comment_title	STRING	NULLABLE			
review_creation_date	TIMESTAMP	NULLABLE			
review_answer_timestamp	TIMESTAMP	NULLABLE			

Features	**Description**
review_id	Id of the review given on the product ordered by the order id.
order_id	A unique id of order made by the consumers.

review_score	review score given by the customer for each order on the scale of 1–5.
review_comment_title	Title of the review
review_comment_message	Review comments posted by the consumer for each order.
review_creation_date	Timestamp of the review when it is created.
review_answer_timestamp	Timestamp of the review answered.

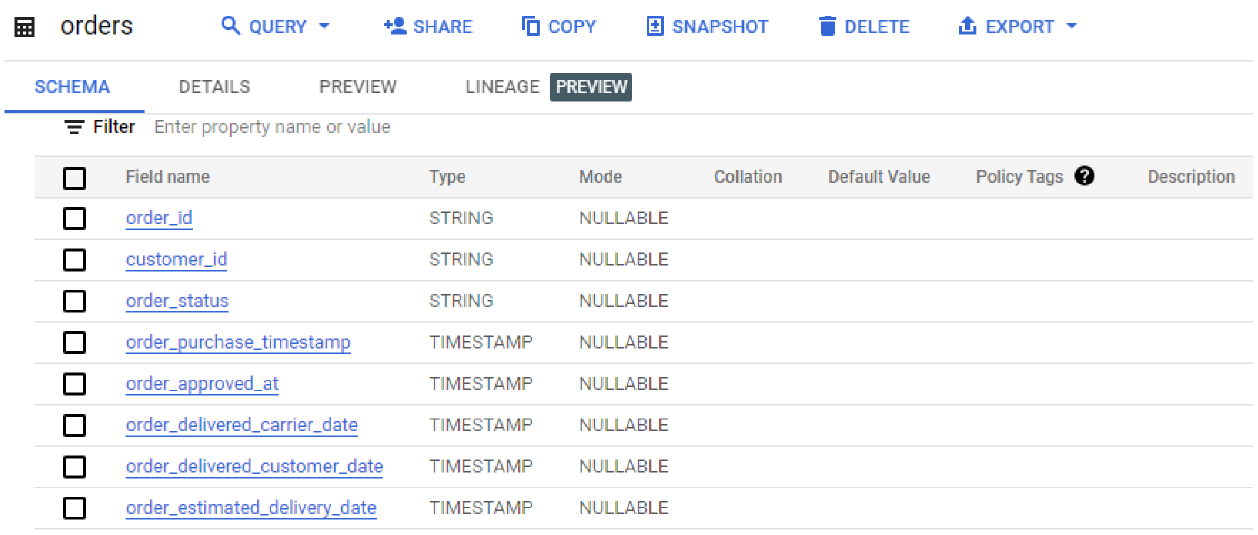


```
select *
from `target.order_reviews`
```

Query results

Row	review_id	order_id	review_score	review_comment_title	review_creation_date	review_answer_timestamp
1	be7e2989673cb2a14...	777c67eab7c0712ccd...	1	null	0001-04-17 00:00:00 UTC	0001-04-17 07:40:00 UTC
2	e12151267e4594d69...	4338a4463f7f9193d2...	1	null	0001-04-17 00:00:00 UTC	0001-04-17 09:04:00 UTC
3	41d614b133efebcd10...	b8aaeda740b17cf925...	1	null	0001-04-17 00:00:00 UTC	0002-04-17 03:48:00 UTC
4	c950324a42c5796d0...	b159d0ce7cd881052d...	1	null	0001-04-17 00:00:00 UTC	0001-04-17 10:24:00 UTC
5	76823ada94c8861ec...	2a3007ed051b02a0e0...	1	null	0001-04-17 00:00:00 UTC	0002-04-17 13:58:00 UTC
6	fe270df00abcb5c39f...	a39d3db795a5cf4c8b...	1	null	0001-04-17 00:00:00 UTC	0003-04-17 12:49:00 UTC
7	1b71e0b29ec2faa0a0...	0e530f6be154c9d7e7...	1	null	0001-04-17 00:00:00 UTC	0010-04-17 12:45:00 UTC
8	efe4020a945ee6fece...	264c045399fb02e9f3...	1	null	0001-04-17 00:00:00 UTC	0002-04-17 01:16:00 UTC

5. **orders**:
This table contains, details of the Order placed in terms of customer tracking

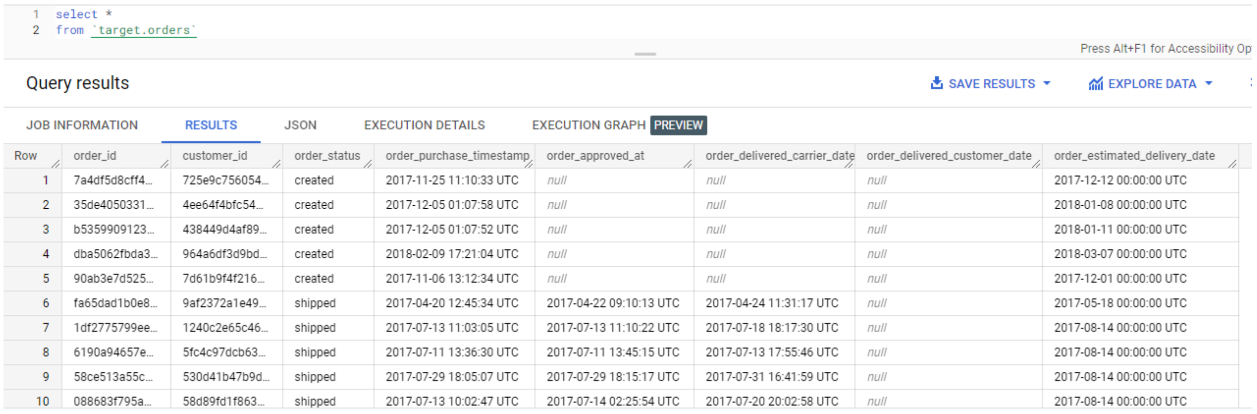


orders

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
order_id	STRING	NULLABLE				
customer_id	STRING	NULLABLE				
order_status	STRING	NULLABLE				
order_purchase_timestamp	TIMESTAMP	NULLABLE				
order_approved_at	TIMESTAMP	NULLABLE				
order_delivered_carrier_date	TIMESTAMP	NULLABLE				
order_delivered_customer_date	TIMESTAMP	NULLABLE				
order_estimated_delivery_date	TIMESTAMP	NULLABLE				

Features	**Description**
order_id	A unique id of order made by the consumers.
customer_id	Id of the consumer who made the purchase.
order_status	status of the order made i.e delivered, shipped etc.

order_purchase_timestamp	Timestamp of the purchase.
order_delivered_carrier_date	delivery date at which carrier made the delivery.
order_delivered_customer_date	date at which customer got the product.
order_estimated_delivery_date	estimated delivery date of the products.



```sql
select *
from `target.orders`
```

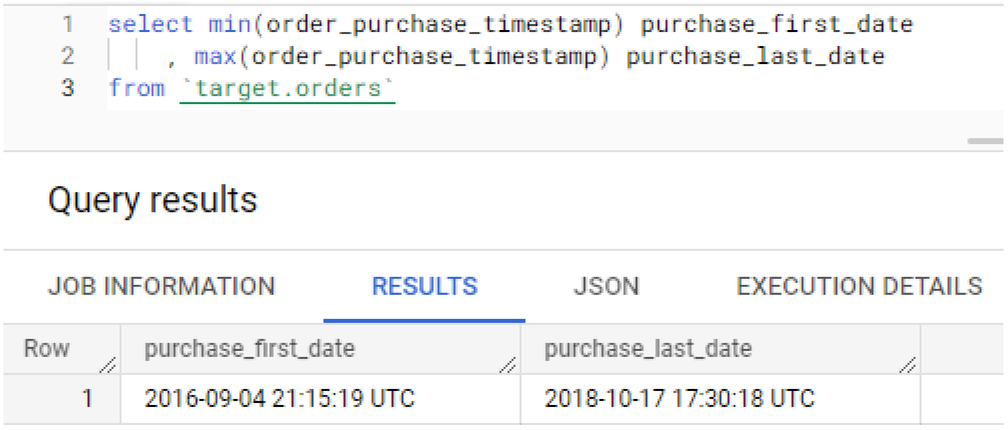
Press Alt+F1 for Accessibility Op

Query results

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

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	7a4df5d8cff4...	725e9c756054...	created	2017-11-25 11:10:33 UTC	null	null	null	2017-12-12 00:00:00 UTC
2	35de4050331...	4ee64f4bfc54...	created	2017-12-05 01:07:58 UTC	null	null	null	2018-01-08 00:00:00 UTC
3	b5359909123...	438449d4af89...	created	2017-12-05 01:07:52 UTC	null	null	null	2018-01-11 00:00:00 UTC
4	dba5062fbda3...	964a6df3d9bd...	created	2018-02-09 17:21:04 UTC	null	null	null	2018-03-07 00:00:00 UTC
5	90ab3e7d525...	7d61b9f4f216...	created	2017-11-06 13:12:34 UTC	null	null	null	2017-12-01 00:00:00 UTC
6	fa65dad1b0e8...	9af2372a1e49...	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	1df2775799ee...	1240c2e65c46...	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	6190a94657e...	5fc4c97dcb63...	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	58ce513a55c...	530d41b47b9d...	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	088683f795a...	58d89fd1f863...	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

This table contains order **purchase details over 2 years** from 4th Sep 2016 till last order purchase date 17th Oct 2018



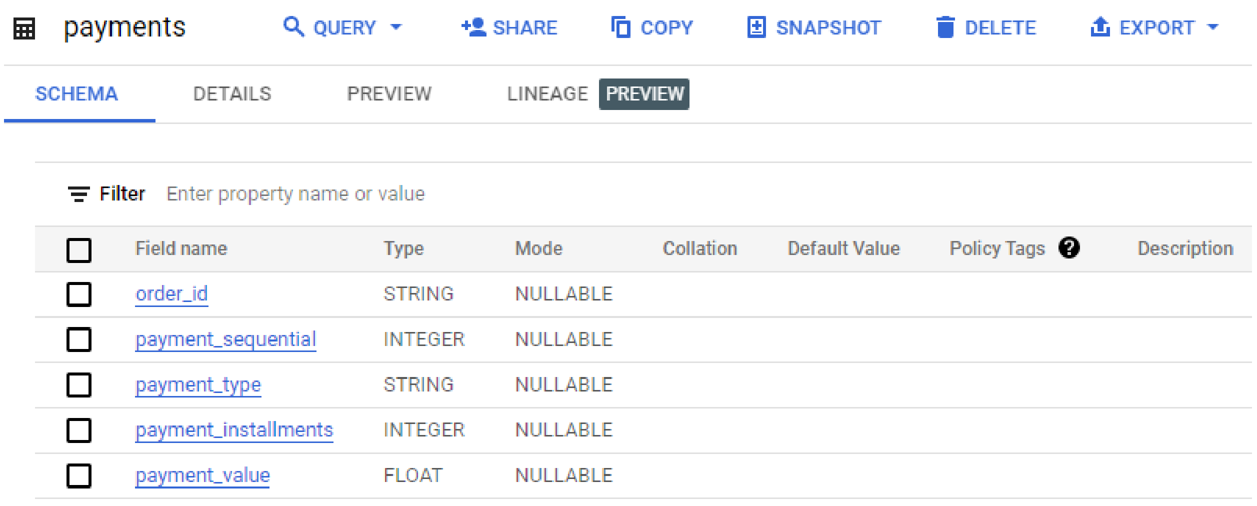
```sql
select min(order_purchase_timestamp) purchase_first_date
      , max(order_purchase_timestamp) purchase_last_date
from `target.orders`
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

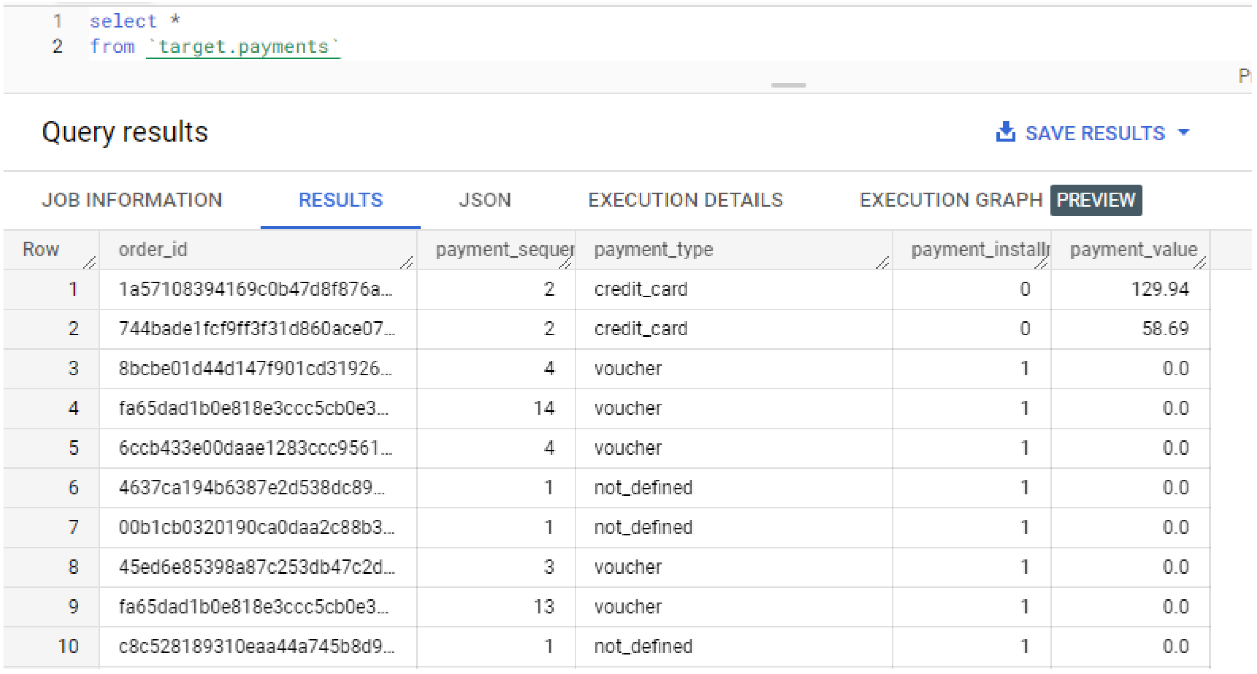
Row	purchase_first_date	purchase_last_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

6. **payments**:
   this table contains Payment details of every Order



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
order_id	STRING	NULLABLE				
payment_sequential	INTEGER	NULLABLE				
payment_type	STRING	NULLABLE				
payment_installments	INTEGER	NULLABLE				
payment_value	FLOAT	NULLABLE				

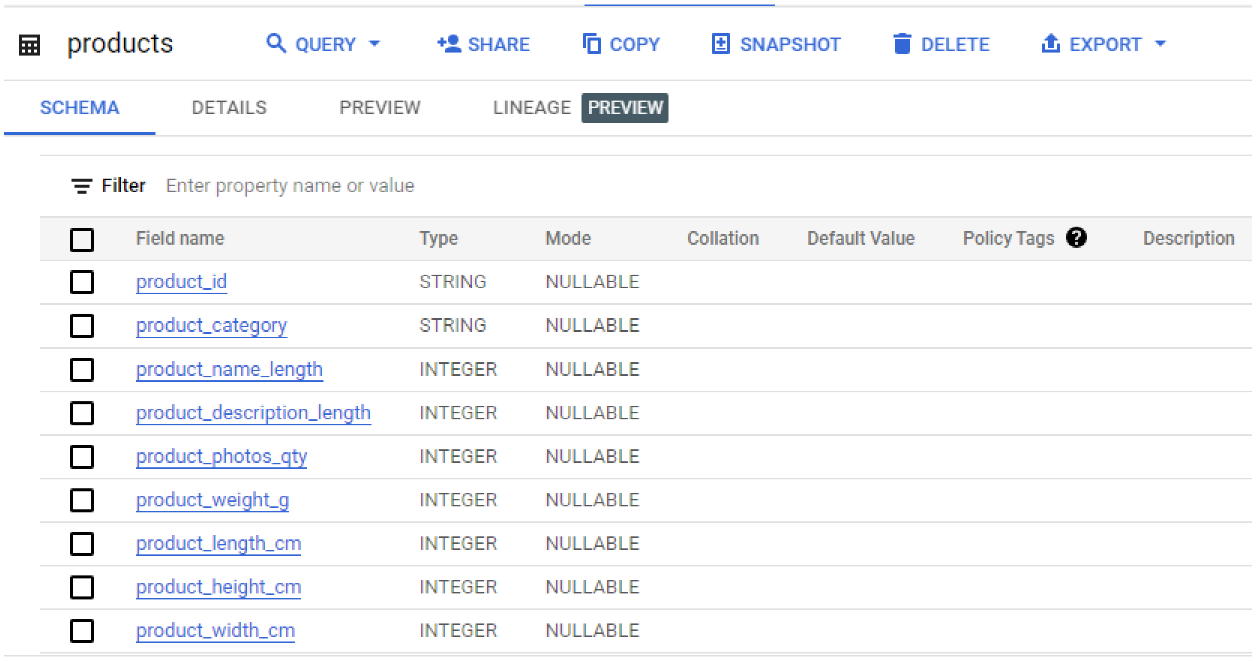
Features	Description
order_id	A unique id of order made by the consumers.
payment_sequential	sequences of the payments made in case of EMI.
payment_type	mode of payment used.(Ex-Credit Card)
payment_installments	number of installments in case of EMI purchase.
payment_value	Total amount paid for the purchase order.



```
select *
from `target.payments`
```

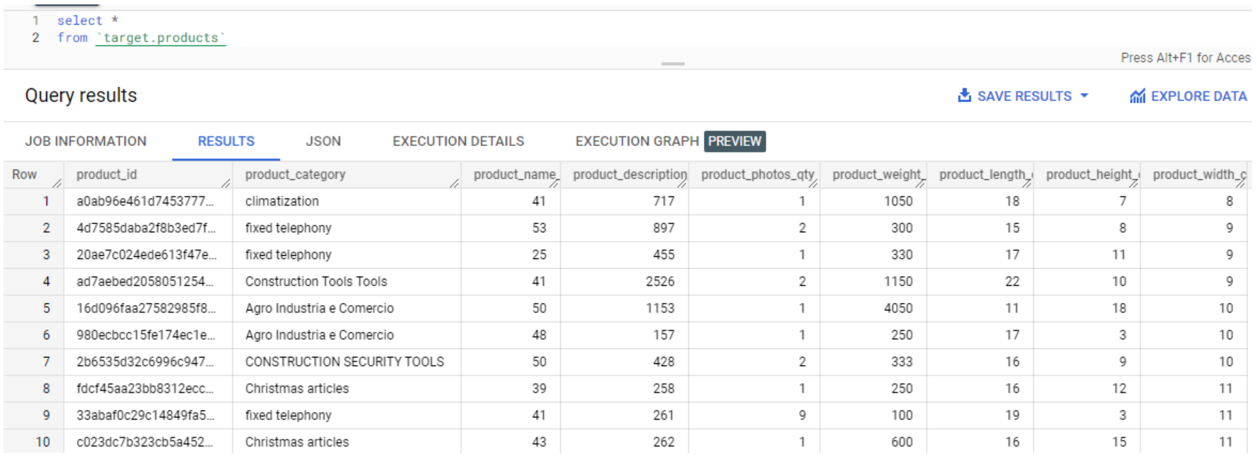
Row	order_id	payment_sequer	payment_type	payment_install	payment_value
1	1a57108394169c0b47d8f876a...	2	credit_card	0	129.94
2	744bade1fcf9ff3f31d860ace07...	2	credit_card	0	58.69
3	8bcbe01d44d147f901cd31926...	4	voucher	1	0.0
4	fa65dad1b0e818e3ccc5cb0e3...	14	voucher	1	0.0
5	6ccb433e00daae1283ccc9561...	4	voucher	1	0.0
6	4637ca194b6387e2d538dc89...	1	not_defined	1	0.0
7	00b1cb0320190ca0daa2c88b3...	1	not_defined	1	0.0
8	45ed6e85398a87c253db47c2d...	3	voucher	1	0.0
9	fa65dad1b0e818e3ccc5cb0e3...	13	voucher	1	0.0
10	c8c528189310eaa44a745b8d9...	1	not_defined	1	0.0

7. **products**:
   Master table containing product Information



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
product_id	STRING	NULLABLE				
product_category	STRING	NULLABLE				
product_name_length	INTEGER	NULLABLE				
product_description_length	INTEGER	NULLABLE				
product_photos_qty	INTEGER	NULLABLE				
product_weight_g	INTEGER	NULLABLE				
product_length_cm	INTEGER	NULLABLE				
product_height_cm	INTEGER	NULLABLE				
product_width_cm	INTEGER	NULLABLE				

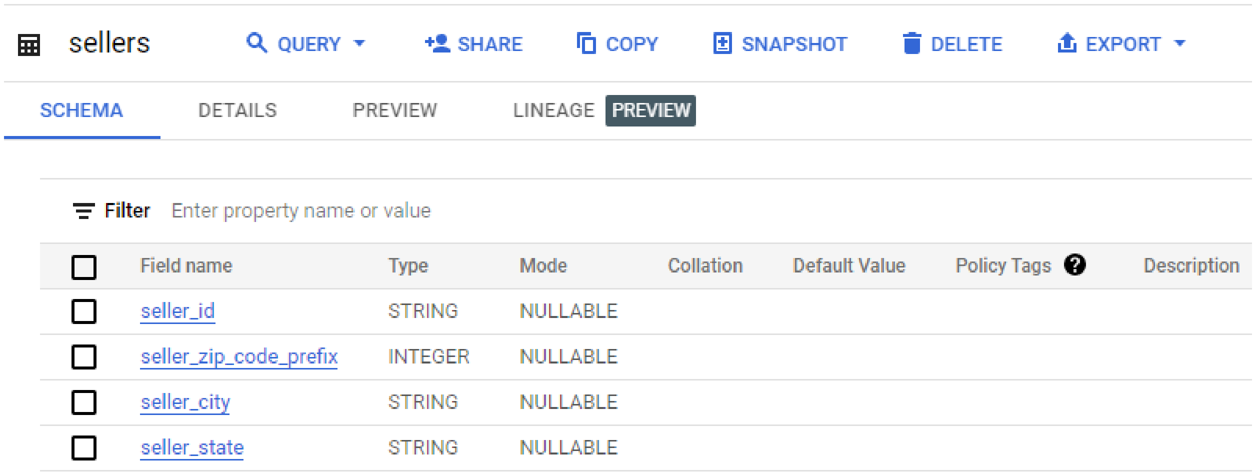
Features	Description
product_id	A unique identifier for the proposed project.
product_category_name	Name of the product category
product_name_lenght	length of the string which specifies the name given to the products ordered.
product_description_lenght	length of the description written for each product ordered on the site.
product_photos_qty	Number of photos of each product ordered available on the shopping portal.
product_weight_g	Weight of the products ordered in grams.
product_length_cm	Length of the products ordered in centimeters.
product_height_cm	Height of the products ordered in centimeters.
product_width_cm	width of the product ordered in centimeters.



```
select *
from `target.products`
```

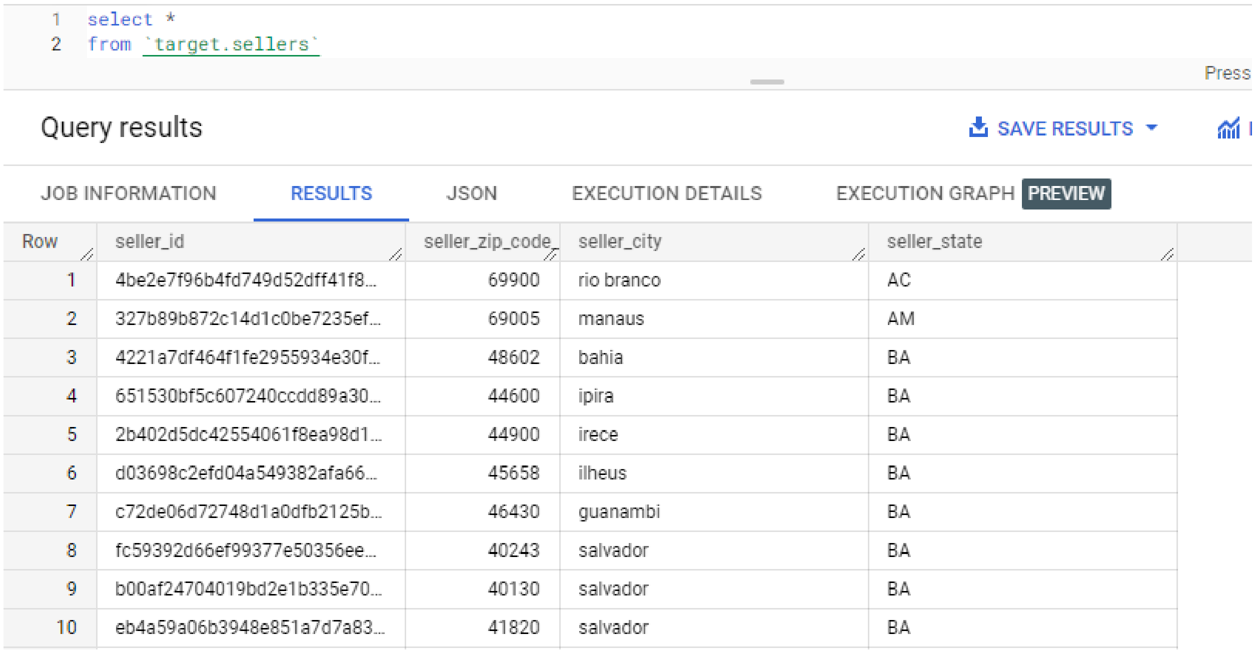
Row	product_id	product_category	product_name_	product_description	product_photos_qty	product_weight_	product_length_	product_height_	product_width_c
1	a0ab96e461d7453777...	climatization	41	717	1	1050	18	7	8
2	4d7585daba2f8b3ed7f...	fixed telephony	53	897	2	300	15	8	9
3	20ae7c024ede613f47e...	fixed telephony	25	455	1	330	17	11	9
4	ad7aebed2058051254...	Construction Tools Tools	41	2526	2	1150	22	10	9
5	16d096faa27582985f8...	Agro Industria e Comercio	50	1153	1	4050	11	18	10
6	980ecbcc15fe174ec1e...	Agro Industria e Comercio	48	157	1	250	17	3	10
7	2b6535d32c6996c947...	CONSTRUCTION SECURITY TOOLS	50	428	2	333	16	9	10
8	fdcf45aa23bb8312ecc...	Christmas articles	39	258	1	250	16	12	11
9	33abaf0c29c14849fa5...	fixed telephony	41	261	9	100	19	3	11
10	c023dc7b323cb5a452...	Christmas articles	43	262	1	600	16	15	11

8. **sellers**:
Master dataset containing Seller demographic information



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
seller_id	STRING	NULLABLE				
seller_zip_code_prefix	INTEGER	NULLABLE				
seller_city	STRING	NULLABLE				
seller_state	STRING	NULLABLE				

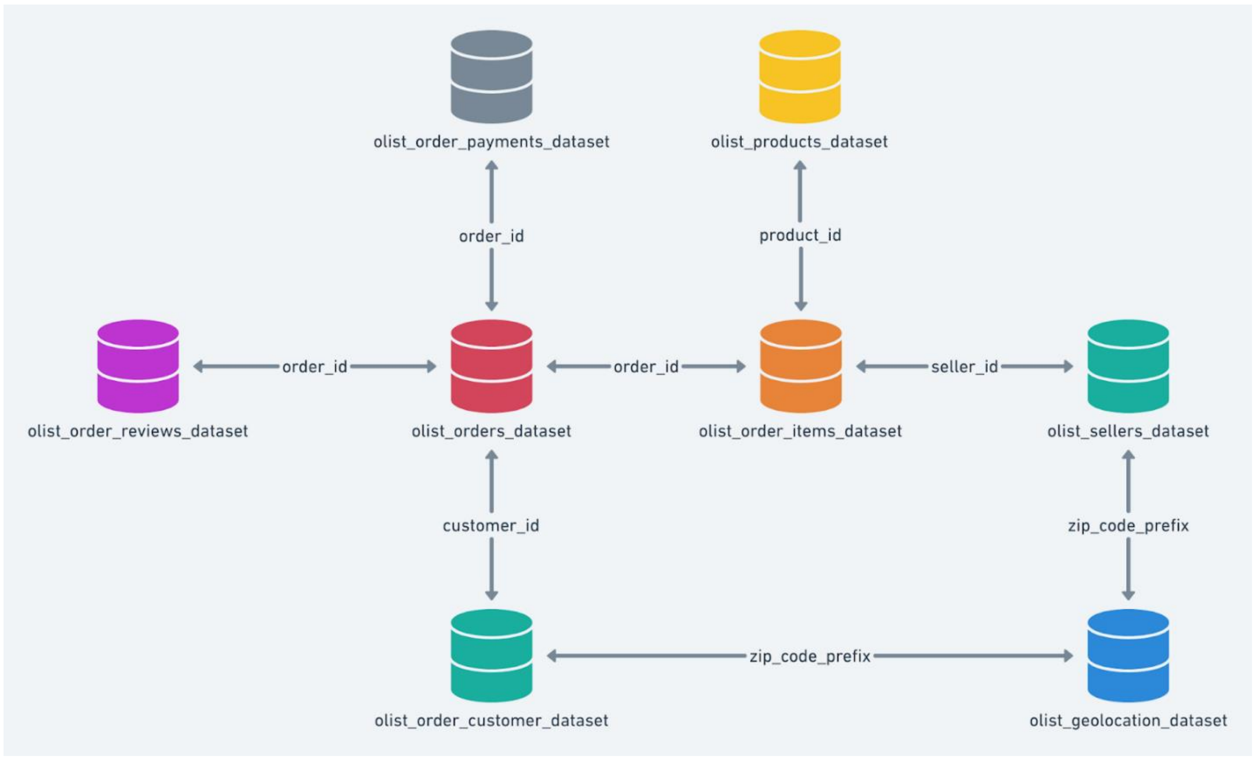
Features	Description
seller_id	Unique Id of the seller registered
seller_zip_code_prefix	Zip Code of the location of the seller.
seller_city	Name of the City of the seller.
seller_state	State Code (Ex- sao paulo-SP)



```
select *
from `target.sellers`
```

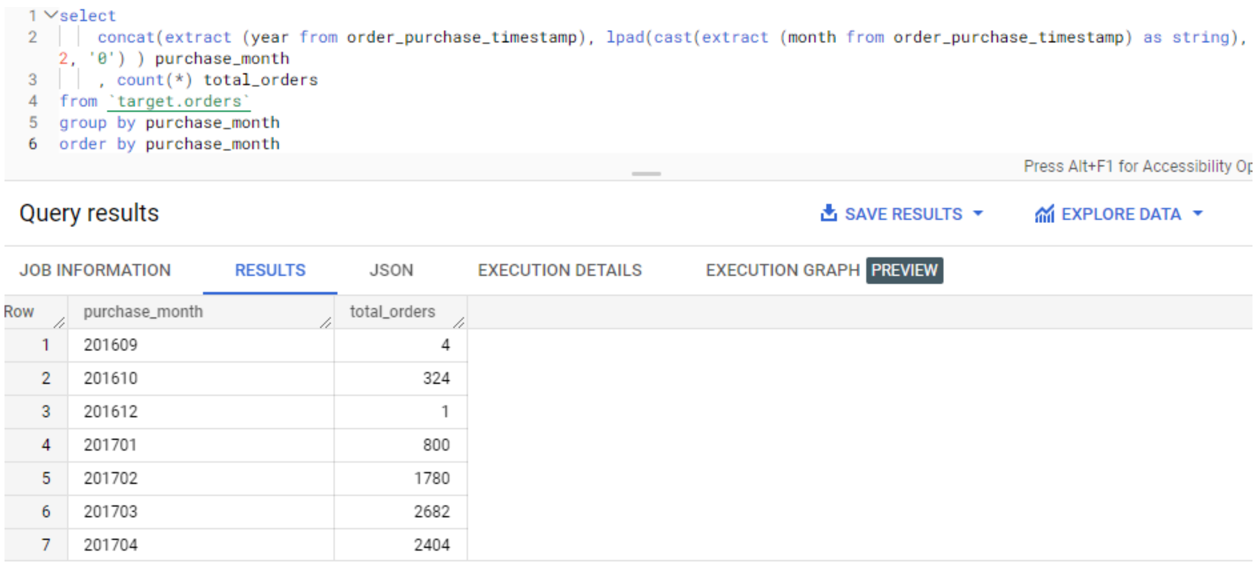
Row	seller_id	seller_zip_code_prefix	seller_city	seller_state
1	4be2e7f96b4fd749d52dff41f8...	69900	rio branco	AC
2	327b89b872c14d1c0be7235ef...	69005	manaus	AM
3	4221a7df464f1fe2955934e30f...	48602	bahia	BA
4	651530bf5c607240ccdd89a30...	44600	ipira	BA
5	2b402d5dc42554061f8ea98d1...	44900	irece	BA
6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

Relationship Diagram:



Part 2: Target DB In-depth Exploration

1. Number of Order are increasing
   It is evident from the data that count of orders are Increasing MoM up until November, 2017 where is attains its peak at 7544 orders. However, post this month count is moving in steady rate.



```
select
    concat(extract (year from order_purchase_timestamp), lpad(cast(extract (month from order_purchase_timestamp) as string),
2, '0') ) purchase_month
    , count(*) total_orders
from `target.orders`
group by purchase_month
order by purchase_month
```

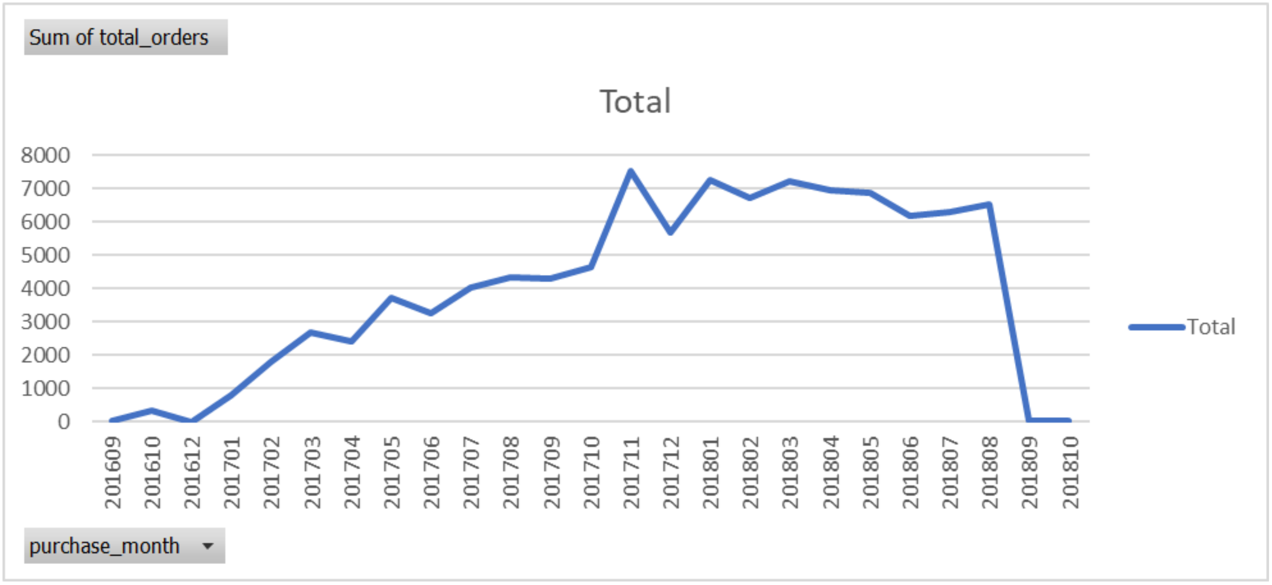
Press Alt+F1 for Accessibility Op

Query results

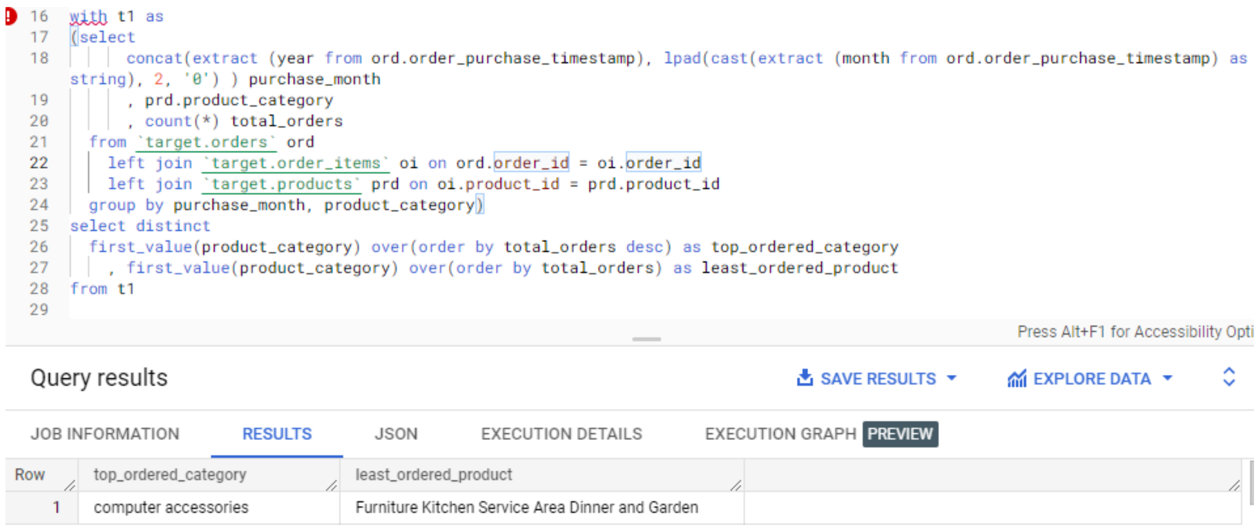
SAVE RESULTS   EXPLORE DATA

JOB INFORMATION   RESULTS   JSON   EXECUTION DETAILS   EXECUTION GRAPH PREVIEW

Row	purchase_month	total_orders
1	201609	4
2	201610	324
3	201612	1
4	201701	800
5	201702	1780
6	201703	2682
7	201704	2404



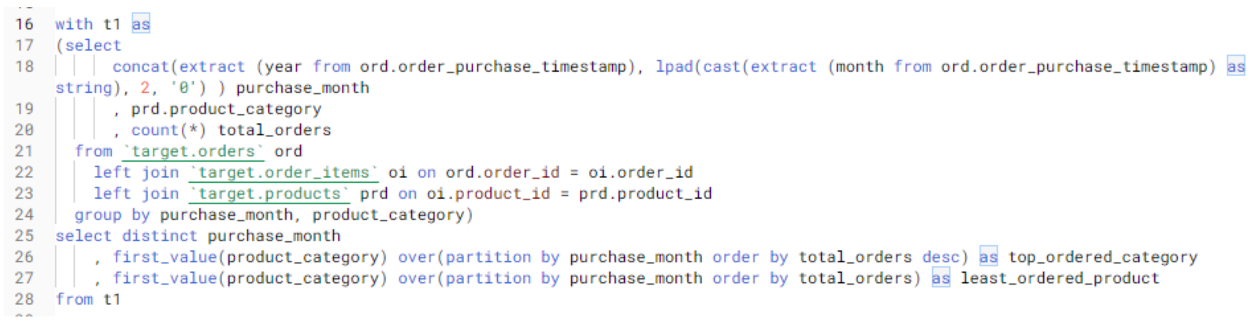
Overall, top Purchase category is '*computer accessories*' and lowest purchase category is 'Furniture Kitchen Service Area Dinner and Garden'



```
with t1 as
(select
      concat(extract (year from ord.order_purchase_timestamp), lpad(cast(extract (month from ord.order_purchase_timestamp) as
string), 2, '0') ) purchase_month
      , prd.product_category
      , count(*) total_orders
  from `target.orders` ord
    left join `target.order_items` oi on ord.order_id = oi.order_id
    left join `target.products` prd on oi.product_id = prd.product_id
  group by purchase_month, product_category)
select distinct
  first_value(product_category) over(order by total_orders desc) as top_ordered_category
    , first_value(product_category) over(order by total_orders) as least_ordered_product
from t1
```

Row	top_ordered_category	least_ordered_product
1	computer accessories	Furniture Kitchen Service Area Dinner and Garden

MoM top and least purchased categories are:



```
with t1 as
(select
      concat(extract (year from ord.order_purchase_timestamp), lpad(cast(extract (month from ord.order_purchase_timestamp) as
string), 2, '0') ) purchase_month
      , prd.product_category
      , count(*) total_orders
  from `target.orders` ord
    left join `target.order_items` oi on ord.order_id = oi.order_id
    left join `target.products` prd on oi.product_id = prd.product_id
  group by purchase_month, product_category)
select distinct purchase_month
    , first_value(product_category) over(partition by purchase_month order by total_orders desc) as top_ordered_category
    , first_value(product_category) over(partition by purchase_month order by total_orders) as least_ordered_product
from t1
```

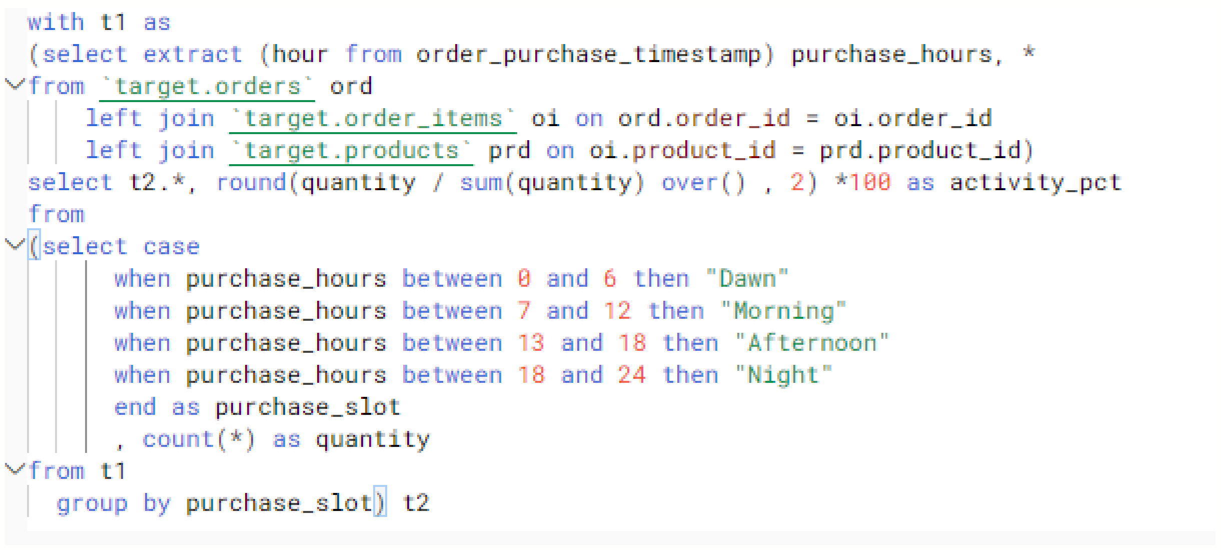
purchase_month	top_ordered_category	least_ordered_product
201609	HEALTH BEAUTY	telephony
201610	Furniture Decoration	General Interest Books
201612	Fashion Bags and Accessories	Fashion Bags and Accessories
201701	Furniture Decoration	Construction Tools Construction
201702	Furniture Decoration	Industry Commerce and Business
201703	Furniture Decoration	Fashion Women's Clothing
201704	bed table bath	technical books
201705	bed table bath	Construction Tools Construction
201706	bed table bath	House Comfort 2
201707	bed table bath	CITTE AND UPHACK FURNITURE
201708	bed table bath	CONSTRUCTION SECURITY TOOLS
201709	bed table bath	cine photo
201710	bed table bath	cds music dvds
201711	bed table bath	Art
201712	bed table bath	Furniture
201801	bed table bath	HOUSE PASTALS OVEN AND CAFE

201802	computer accessories	PC Gamer
201803	bed table bath	Fashion Sport
201804	bed table bath	Fashion Sport
201805	HEALTH BEAUTY	IMAGE IMPORT TABLETS
201806	HEALTH BEAUTY	IMAGE IMPORT TABLETS
201807	HEALTH BEAUTY	Blu Ray DVDs
201808	HEALTH BEAUTY	PC Gamer
201809		Furniture Kitchen Service Area Dinner and Garden
201810		

2. Considering, when purchase_hours between 0 and 6 then "Dawn"

   when purchase_hours between 7 and 12 then "Morning"
   when purchase_hours between 13 and 18 then "Afternoon"
   when purchase_hours between 18 and 24 then "Night"

Brazilians found to be more active in their purchases during 'AfterNoon' time

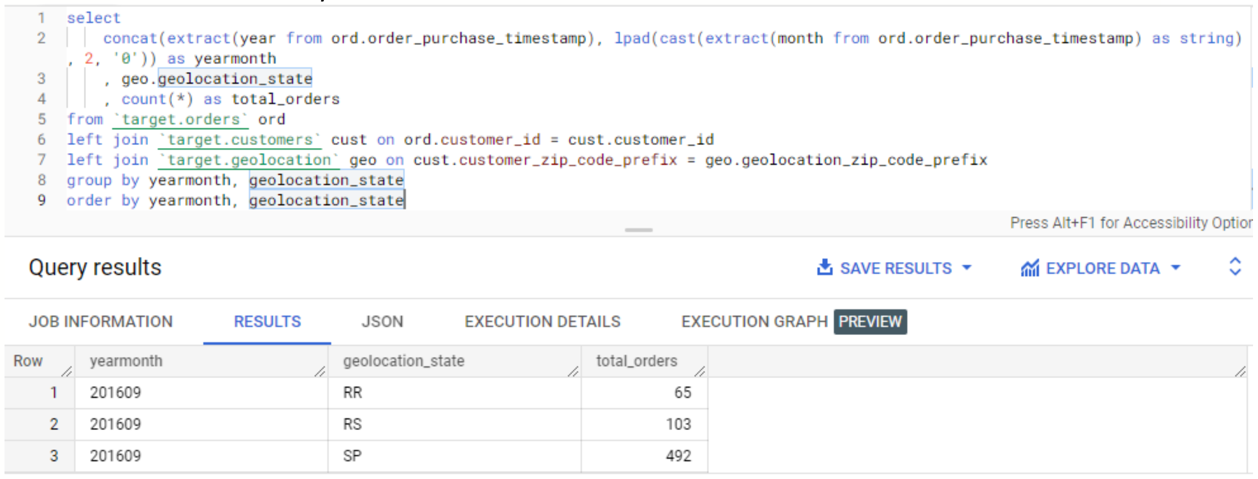


```
with t1 as
(select extract (hour from order_purchase_timestamp) purchase_hours, *
from `target.orders` ord
    left join `target.order_items` oi on ord.order_id = oi.order_id
    left join `target.products` prd on oi.product_id = prd.product_id)
select t2.*, round(quantity / sum(quantity) over() , 2) *100 as activity_pct
from
(select case
      when purchase_hours between 0 and 6 then "Dawn"
      when purchase_hours between 7 and 12 then "Morning"
      when purchase_hours between 13 and 18 then "Afternoon"
      when purchase_hours between 18 and 24 then "Night"
      end as purchase_slot
      , count(*) as quantity
from t1
  group by purchase_slot) t2
```

purchase_slot	quantity	activity_pct
Morning	31731	28
Afternoon	43843	39
Night	31887	28
Dawn	5964	5

Part 3: Evolution of E-Commerce Order in Brazil region

1. Month on Month orders by state



```
select
    concat(extract(year from ord.order_purchase_timestamp), lpad(cast(extract(month from ord.order_purchase_timestamp) as string), 2, '0')) as yearmonth
    , geo.geolocation_state
    , count(*) as total_orders
from `target.orders` ord
left join `target.customers` cust on ord.customer_id = cust.customer_id
left join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix
group by yearmonth, geolocation_state
order by yearmonth, geolocation_state
```

Press Alt+F1 for Accessibility Optior

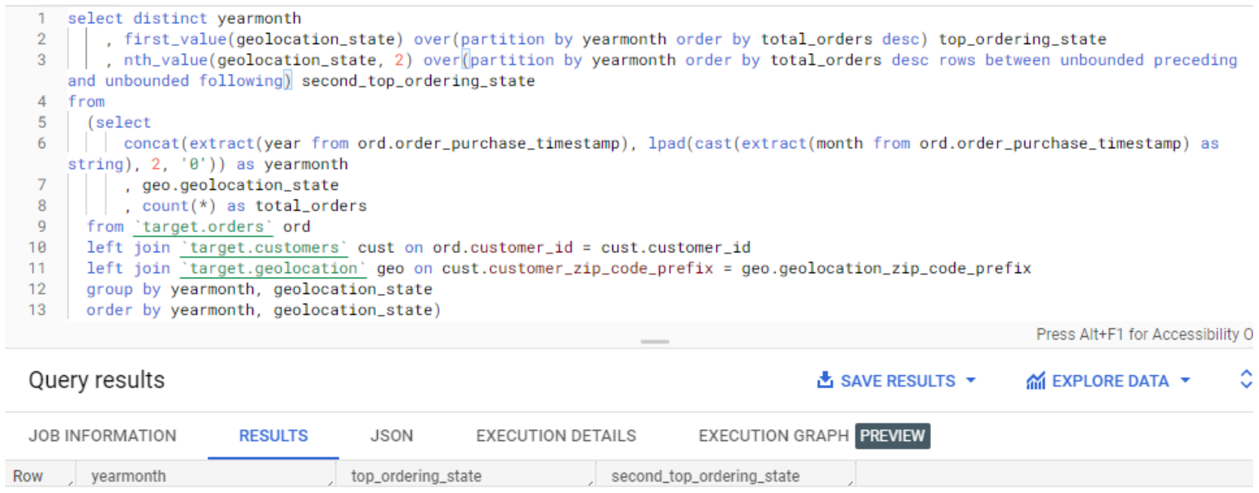
Query results — SAVE RESULTS — EXPLORE DATA

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

Row	yearmonth	geolocation_state	total_orders
1	201609	RR	65
2	201609	RS	103
3	201609	SP	492

Target - MoM orders by State.csv

<u>From the data it is evident that, 'SP' has maximum purchase order in almost every month followed by 'RS', 'RG' and 'MG'</u>



```
select distinct yearmonth
    , first_value(geolocation_state) over(partition by yearmonth order by total_orders desc) top_ordering_state
    , nth_value(geolocation_state, 2) over(partition by yearmonth order by total_orders desc rows between unbounded preceding and unbounded following) second_top_ordering_state
from
  (select
      concat(extract(year from ord.order_purchase_timestamp), lpad(cast(extract(month from ord.order_purchase_timestamp) as string), 2, '0')) as yearmonth
      , geo.geolocation_state
      , count(*) as total_orders
  from `target.orders` ord
  left join `target.customers` cust on ord.customer_id = cust.customer_id
  left join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix
  group by yearmonth, geolocation_state
  order by yearmonth, geolocation_state)
```

Press Alt+F1 for Accessibility O

Query results — SAVE RESULTS — EXPLORE DATA

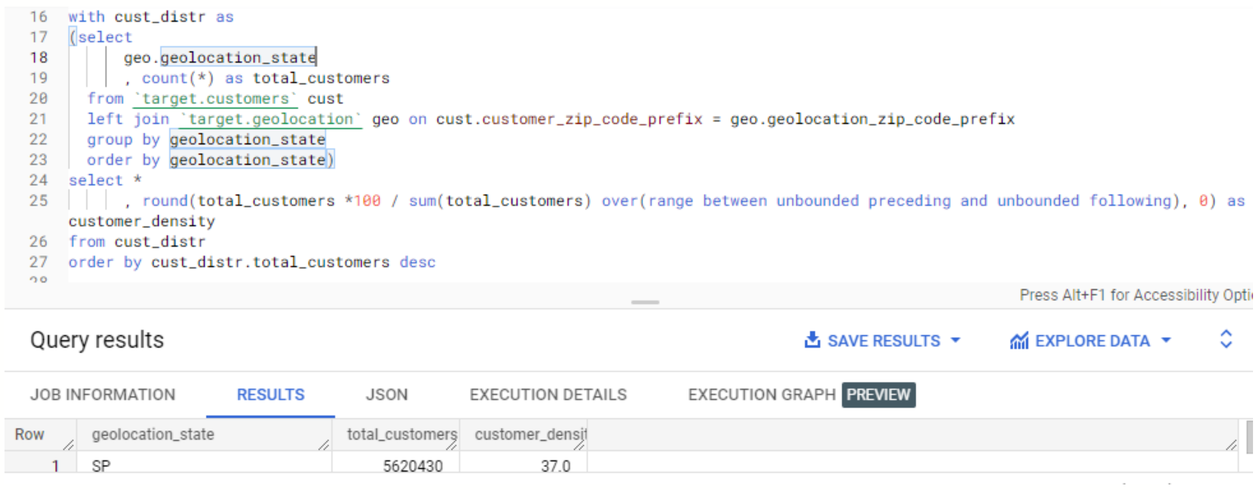
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row | yearmonth | top_ordering_state | second_top_ordering_state

yearmonth	top_ordering_state	second_top_ordering_state
201609	SP	RS
201610	SP	RJ
201612	PR	
201701	SP	MG
201702	SP	MG

201703	SP	RJ
201704	SP	RJ
201705	SP	RJ
201706	SP	RJ
201707	SP	RJ
201708	SP	RJ
201709	SP	RJ
201710	SP	RJ
201711	SP	RJ
201712	SP	RJ
201801	SP	MG
201802	SP	RJ
201803	SP	MG
201804	SP	RJ
201805	SP	RJ
201806	SP	MG
201807	SP	RJ
201808	SP	RJ
201809	SP	MG
201810	SP	RJ

2. Distribution of customers Across the states in Brazil



```
with cust_distr as
(select
      geo.geolocation_state
      , count(*) as total_customers
  from `target.customers` cust
  left join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix
  group by geolocation_state
  order by geolocation_state)
select *
      , round(total_customers *100 / sum(total_customers) over(range between unbounded preceding and unbounded following), 0) as
customer_density
from cust_distr
order by cust_distr.total_customers desc
```

Press Alt+F1 for Accessibility Opti

Query results

```
Row  geolocation_state  total_customers  customer_densit
1    SP                 5620430          37.0
```

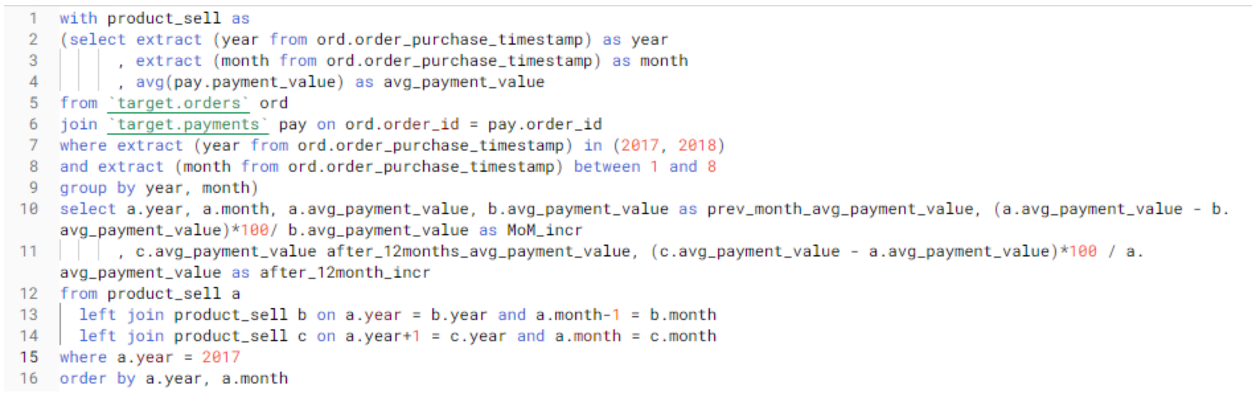
geolocation_state	total_customers	customer_density
SP	5620430	37

RJ	3015690	20
MG	2878728	19
RS	805370	5
PR	626021	4
SC	538638	4
BA	365875	2
ES	316654	2
GO	133146	1
MT	122395	1
PE	114588	1
DF	93309	1
PA	83554	1
CE	63507	0
MS	61473	0
MA	53383	0
AL	34861	0
PB	27714	0
SE	24584	0
PI	23913	0
RO	21244	0
RN	20595	0
TO	17509	0
AC	7688	0
AM	5587	0
AP	4912	0
RR	2087	0
	278	0

Conclusion: Result shows that, Target has its maximum customer base from SP(39%) , RJ(20%) and MG(19%) while lowest base at AM, AP and RR.
Also, it is observed that SP has the maximum purchase order and major business contributor for Target.

Part 4: Impact on Economy

1. **% Increase in cost of orders from 2017 to 2018**

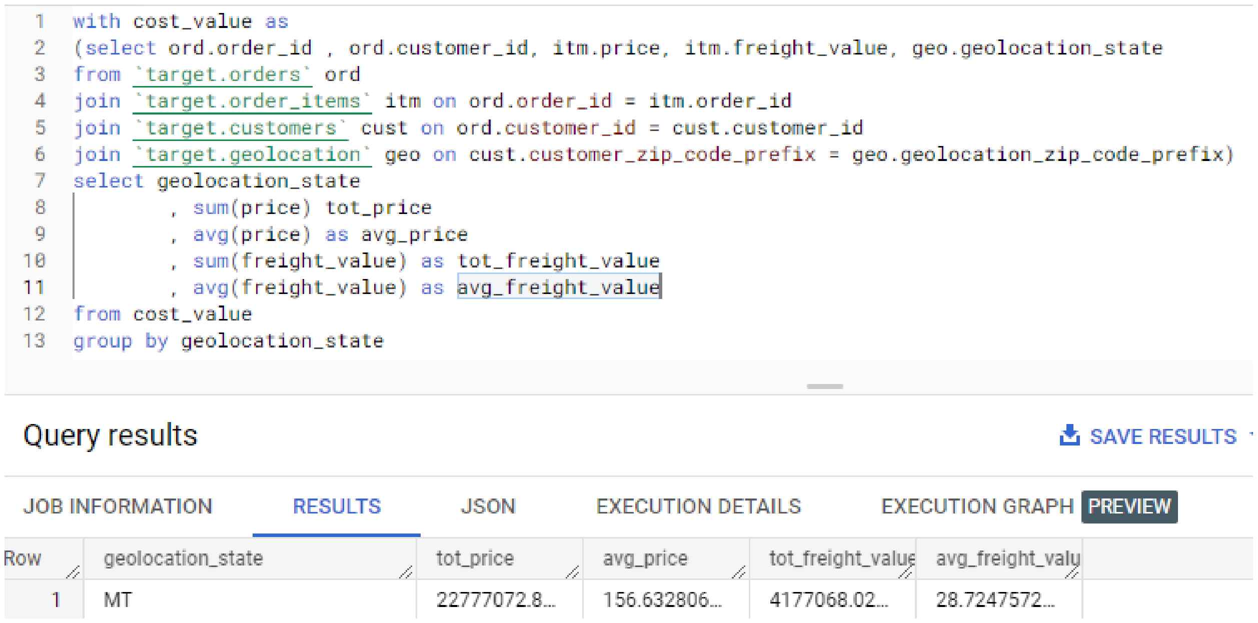


```
with product_sell as
(select extract (year from ord.order_purchase_timestamp) as year
      , extract (month from ord.order_purchase_timestamp) as month
      , avg(pay.payment_value) as avg_payment_value
from `target.orders` ord
join `target.payments` pay on ord.order_id = pay.order_id
where extract (year from ord.order_purchase_timestamp) in (2017, 2018)
and extract (month from ord.order_purchase_timestamp) between 1 and 8
group by year, month)
select a.year, a.month, a.avg_payment_value, b.avg_payment_value as prev_month_avg_payment_value, (a.avg_payment_value - b.avg_payment_value)*100/ b.avg_payment_value as MoM_incr
      , c.avg_payment_value after_12months_avg_payment_value, (c.avg_payment_value - a.avg_payment_value)*100 / a.avg_payment_value as after_12month_incr
from product_sell a
  left join product_sell b on a.year = b.year and a.month-1 = b.month
  left join product_sell c on a.year+1 = c.year and a.month = c.month
where a.year = 2017
order by a.year, a.month
```

year	month	avg_payment_value	prev_month_avg_payment_value	MoM_incr	after_12months_avg_payment_value	after_12month_incr
2017	1	162.93			147.43	-9.51
2017	2	154.78	162.93	-5.00	142.76	-7.76
2017	3	158.57	154.78	2.45	154.37	-2.65
2017	4	162.50	158.57	2.48	161.02	-0.91
2017	5	150.33	162.50	-7.49	161.74	7.58
2017	6	148.80	150.33	-1.02	159.51	7.20
2017	7	137.22	148.80	-7.78	163.91	19.45
2017	8	148.22	137.22	8.01	152.65	2.99

From the result, it is evident that even though MoM % increase in total sales is not consistant, YoY % increase is positive and increases as company progresses from Jan to Aug.

2. value by customer state



```
with cost_value as
(select ord.order_id , ord.customer_id, itm.price, itm.freight_value, geo.geolocation_state
from `target.orders` ord
join `target.order_items` itm on ord.order_id = itm.order_id
join `target.customers` cust on ord.customer_id = cust.customer_id
join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix)
select geolocation_state
        , sum(price) tot_price
        , avg(price) as avg_price
        , sum(freight_value) as tot_freight_value
        , avg(freight_value) as avg_freight_value
from cost_value
group by geolocation_state
```

Query results

Row	geolocation_state	tot_price	avg_price	tot_freight_value	avg_freight_valu
1	MT	22777072.8...	156.632806...	4177068.02...	28.7247572...

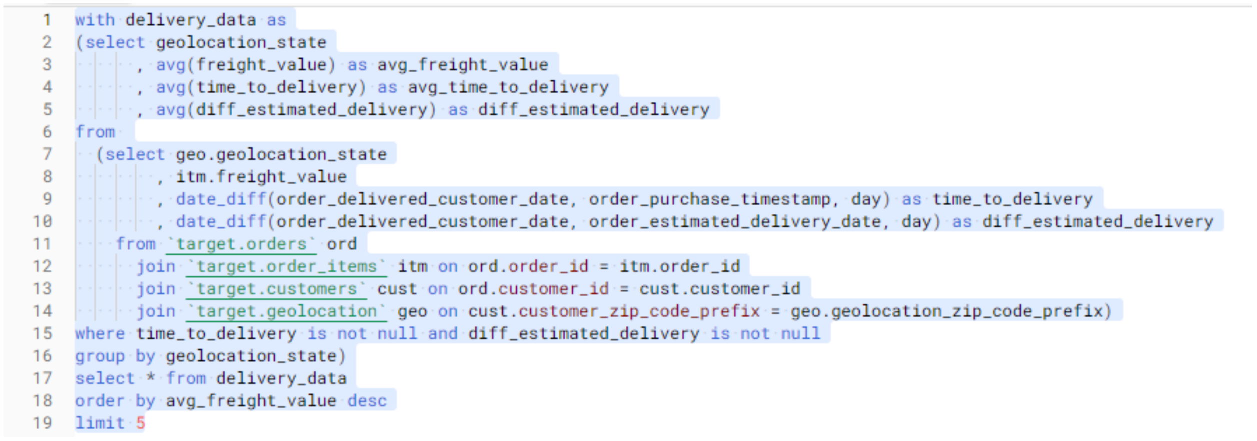
geolocation_state	tot_price	avg_price	tot_freight_value	avg_freight_value
MT	22777072.82	156.6328065	4177068.03	28.72475728
MA	9020091.01	150.9512344	2275191.86	38.07533863
AL	7191886.1	196.644686	1237356.22	33.8325054
SP	711838740.5	111.2803217	98574572.43	15.40996507
MG	397190155.9	121.179718	67058347.09	20.45899545
PE	17545068.94	137.4240739	4195977.72	32.86555067
RJ	440142503.5	127.8128982	71966793.75	20.8984236
DF	13141649.62	124.6611106	2214955.55	21.01097098
RS	111183139.6	120.1812283	19910834.35	21.52222485
SE	3976184.44	146.1080488	943582.83	34.67269898
PR	85392469.28	119.2119441	14432159.77	20.14798072
PA	15586180.17	166.9792823	3409472.09	36.52666635
BA	62377311.67	149.6397065	11345094	27.21625045
CE	10819201.81	151.3238571	2306600.06	32.26149433
GO	20860945.92	134.6188827	3590268.56	23.16855353
ES	43634878.56	123.3648243	7799979.09	22.05215374
SC	79666423.29	127.4134771	13472314.62	21.54677441
PI	4581195.05	172.9405455	1045754.34	39.47732503
PB	6278650.25	198.8613768	1350462.24	42.77269312
RN	3721308.97	160.3183254	790793.15	34.06829011
AM	825147.21	131.6654236	216974.1	34.62168502
RR	360027.85	149.3271879	102394.21	42.46960182
MS	9891112.52	139.1018116	1698977.52	23.8932527
TO	3350329.32	168.4598411	743027.53	37.36059584
AC	1494037.73	179.3132177	325767.64	39.09837254

RO	3577073.58	150.5058939	889573.06	37.42891656
AP	988578.63	177.1011519	199028.01	35.65532247

It appears that cost is mainly driven by freight value as avg cost is higher in the state where freight cost is also the highest. This indicates that in order to maximize profit, Target must reduce the freight cost by setting up more chains/ warehouse in high-cost states.

Part 5: Analysis on sales, freight and delivery time

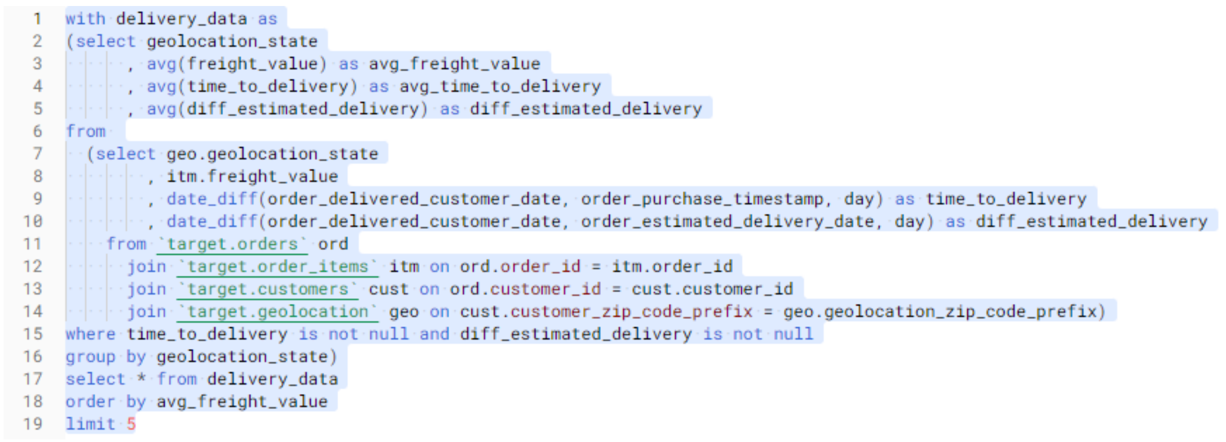
1. States with Max freight value



```sql
with delivery_data as
(select geolocation_state
      , avg(freight_value) as avg_freight_value
      , avg(time_to_delivery) as avg_time_to_delivery
      , avg(diff_estimated_delivery) as diff_estimated_delivery
from
  (select geo.geolocation_state
        , itm.freight_value
        , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery
        , date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
    from `target.orders` ord
      join `target.order_items` itm on ord.order_id = itm.order_id
      join `target.customers` cust on ord.customer_id = cust.customer_id
      join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix)
where time_to_delivery is not null and diff_estimated_delivery is not null
group by geolocation_state)
select * from delivery_data
order by avg_freight_value desc
limit 5
```

geolocation_state	avg_freight_value	avg_time_to_delivery	diff_estimated_delivery
PB	42.99199104	19.75987626	-12.44857253
RR	42.61937226	23.98150852	-20.88175182
PI	39.25947523	18.01911453	-11.62964126
AC	39.03228208	20.10112635	-18.5637855
MA	38.33372724	20.90479714	-9.017447262

2. States with Min Freight Value

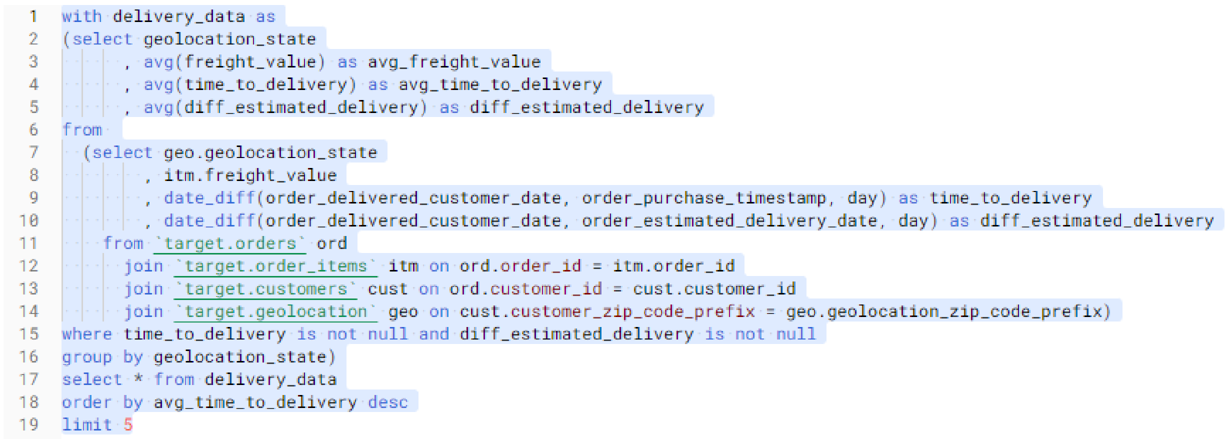


```sql
with delivery_data as
(select geolocation_state
      , avg(freight_value) as avg_freight_value
      , avg(time_to_delivery) as avg_time_to_delivery
      , avg(diff_estimated_delivery) as diff_estimated_delivery
from
  (select geo.geolocation_state
        , itm.freight_value
        , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery
        , date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
    from `target.orders` ord
      join `target.order_items` itm on ord.order_id = itm.order_id
      join `target.customers` cust on ord.customer_id = cust.customer_id
      join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix)
where time_to_delivery is not null and diff_estimated_delivery is not null
group by geolocation_state)
select * from delivery_data
order by avg_freight_value
limit 5
```

geolocation_state	avg_freight_value	avg_time_to_delivery	diff_estimated_delivery
SP	15.39066382	8.442028327	-10.36532505

PR	20.10363143	10.99765013	-12.70182477
MG	20.44687046	11.35698216	-12.47260452
RJ	20.8521473	14.39477827	-11.50210383
DF	21.02212044	12.43733215	-11.50673315

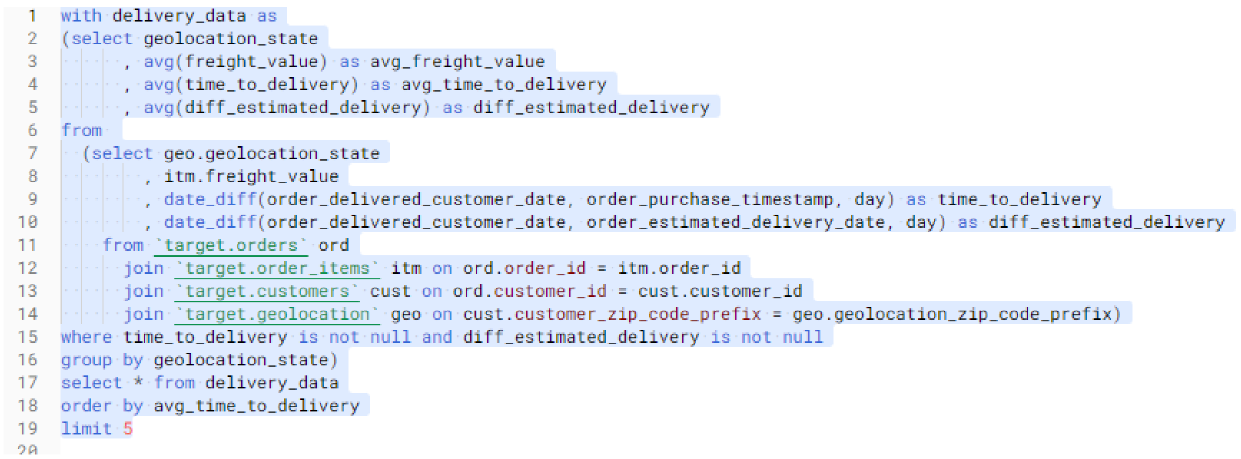
3. State with highest average time to delivery



```
with delivery_data as
(select geolocation_state
        , avg(freight_value) as avg_freight_value
        , avg(time_to_delivery) as avg_time_to_delivery
        , avg(diff_estimated_delivery) as diff_estimated_delivery
from
  (select geo.geolocation_state
          , itm.freight_value
          , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery
          , date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
    from `target.orders` ord
      join `target.order_items` itm on ord.order_id = itm.order_id
      join `target.customers` cust on ord.customer_id = cust.customer_id
      join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix)
where time_to_delivery is not null and diff_estimated_delivery is not null
group by geolocation_state)
select * from delivery_data
order by avg_time_to_delivery desc
limit 5
```

geolocation_state	avg_freight_value	avg_time_to_delivery	diff_estimated_delivery
AP	35.97956936	30.40461792	-15.64760858
AM	34.66841624	24.37997433	-20.56097561
RR	42.61937226	23.98150852	-20.88175182
AL	33.81782657	22.87006928	-8.456950497
PA	36.3714774	22.7334021	-13.58793319

4. State with lowest average time to delivery

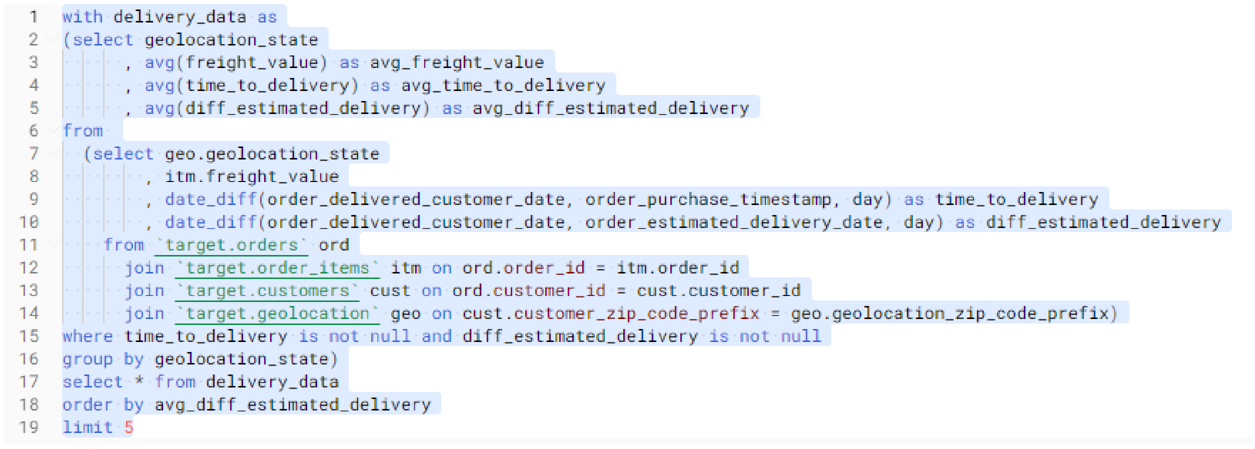


```
with delivery_data as
(select geolocation_state
        , avg(freight_value) as avg_freight_value
        , avg(time_to_delivery) as avg_time_to_delivery
        , avg(diff_estimated_delivery) as diff_estimated_delivery
from
  (select geo.geolocation_state
          , itm.freight_value
          , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery
          , date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
    from `target.orders` ord
      join `target.order_items` itm on ord.order_id = itm.order_id
      join `target.customers` cust on ord.customer_id = cust.customer_id
      join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix)
where time_to_delivery is not null and diff_estimated_delivery is not null
group by geolocation_state)
select * from delivery_data
order by avg_time_to_delivery
limit 5
```

geolocation_state	avg_freight_value	avg_time_to_delivery	diff_estimated_delivery

SP	15.39066382	8.442028327	-10.36532505
PR	20.10363143	10.99765013	-12.70182477
MG	20.44687046	11.35698216	-12.47260452
DF	21.02212044	12.43733215	-11.50673315
RJ	20.8521473	14.39477827	-11.50210383

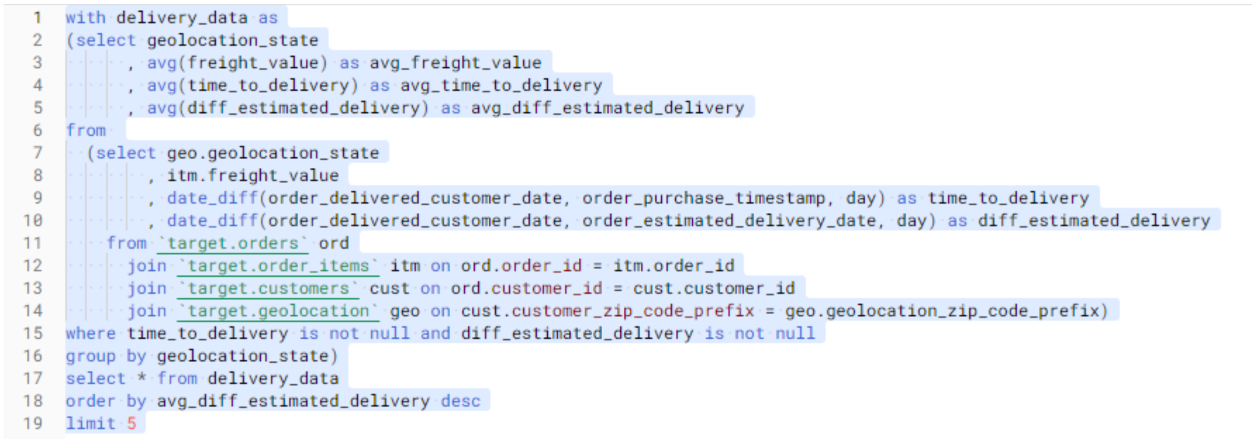
5. State with delivery is fast compared to estimated delivery dates



```
with delivery_data as
(select geolocation_state
      , avg(freight_value) as avg_freight_value
      , avg(time_to_delivery) as avg_time_to_delivery
      , avg(diff_estimated_delivery) as avg_diff_estimated_delivery
from
  (select geo.geolocation_state
        , itm.freight_value
        , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery
        , date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
    from `target.orders` ord
      join `target.order_items` itm on ord.order_id = itm.order_id
      join `target.customers` cust on ord.customer_id = cust.customer_id
      join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix)
where time_to_delivery is not null and diff_estimated_delivery is not null
group by geolocation_state)
select * from delivery_data
order by avg_diff_estimated_delivery
limit 5
```

geolocation_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
RR	42.61937226	23.98150852	-20.88175182
AM	34.66841624	24.37997433	-20.56097561
RO	37.77502655	18.64485215	-19.10356141
AC	39.03228208	20.10112635	-18.5637855
AP	35.97956936	30.40461792	-15.64760858

6. State with delivery is slower compared to estimated delivery dates



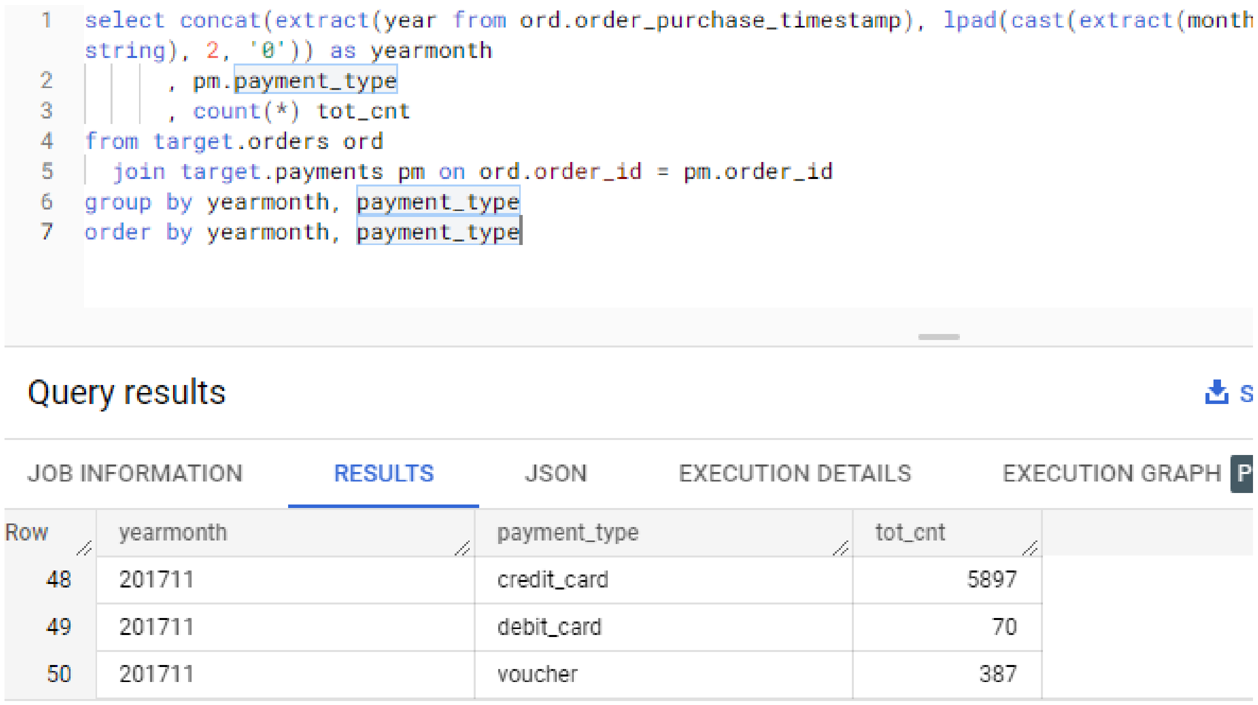
```
with delivery_data as
(select geolocation_state
      , avg(freight_value) as avg_freight_value
      , avg(time_to_delivery) as avg_time_to_delivery
      , avg(diff_estimated_delivery) as avg_diff_estimated_delivery
from
  (select geo.geolocation_state
        , itm.freight_value
        , date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery
        , date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
    from `target.orders` ord
      join `target.order_items` itm on ord.order_id = itm.order_id
      join `target.customers` cust on ord.customer_id = cust.customer_id
      join `target.geolocation` geo on cust.customer_zip_code_prefix = geo.geolocation_zip_code_prefix)
where time_to_delivery is not null and diff_estimated_delivery is not null
group by geolocation_state)
select * from delivery_data
order by avg_diff_estimated_delivery desc
limit 5
```

geolocation_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
AL	33.81782657	22.87006928	-8.456950497
SE	34.63162663	21.17164768	-8.743129958
MA	38.33372724	20.90479714	-9.017447262
CE	32.12015869	20.80449347	-10.01594086
ES	22.03246976	14.70498584	-10.08855785

From the above analysis, it appears that 'Target' has good delivery network (-ve average diff_estimated_delivery indicates, delivery is made before estimated date)
It can also be seen that products with high freight value are generally delivered much before that the expected delivery date.

Part 6: Payment Type Analysis

1. Month on Month Payment type wise count



```
select concat(extract(year from ord.order_purchase_timestamp), lpad(cast(extract(month
string), 2, '0')) as yearmonth
        , pm.payment_type
        , count(*) tot_cnt
from target.orders ord
  join target.payments pm on ord.order_id = pm.order_id
group by yearmonth, payment_type
order by yearmonth, payment_type
```

Query results

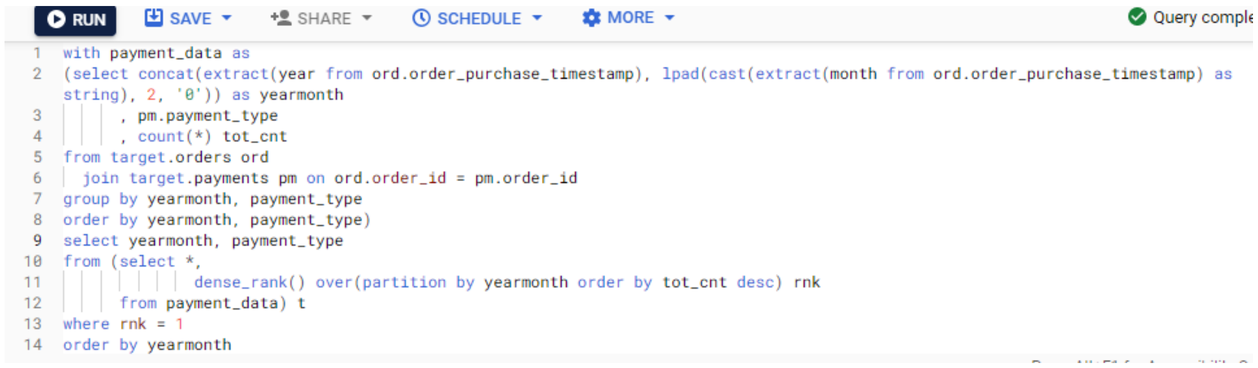
Row	yearmonth	payment_type	tot_cnt
48	201711	credit_card	5897
49	201711	debit_card	70
50	201711	voucher	387

yearmonth	payment_type	tot_cnt
201609	credit_card	3
201610	UPI	63
201610	credit_card	254
201610	debit_card	2
201610	voucher	23
201612	credit_card	1
201701	UPI	197
201701	credit_card	583
201701	debit_card	9
201701	voucher	61
201702	UPI	398
201702	credit_card	1356
201702	debit_card	13
201702	voucher	119

201703	UPI	590
201703	credit_card	2016
201703	debit_card	31
201703	voucher	200
201704	UPI	496
201704	credit_card	1846
201704	debit_card	27
201704	voucher	202
201705	UPI	772
201705	credit_card	2853
201705	debit_card	30
201705	voucher	289
201706	UPI	707
201706	credit_card	2463
201706	debit_card	27
201706	voucher	239
201707	UPI	845
201707	credit_card	3086
201707	debit_card	22
201707	voucher	364
201708	UPI	938
201708	credit_card	3284
201708	debit_card	34
201708	voucher	294
201709	UPI	903
201709	credit_card	3283
201709	debit_card	43
201709	voucher	287
201710	UPI	993
201710	credit_card	3524
201710	debit_card	52
201710	voucher	291
201711	UPI	1509
201711	credit_card	5897
201711	debit_card	70
201711	voucher	387
201712	UPI	1160
201712	credit_card	4377
201712	debit_card	64
201712	voucher	294
201801	UPI	1518

201801	credit_card	5520
201801	debit_card	109
201801	voucher	416
201802	UPI	1325
201802	credit_card	5253
201802	debit_card	69
201802	voucher	305
201803	UPI	1352
201803	credit_card	5691
201803	debit_card	78
201803	voucher	391
201804	UPI	1287
201804	credit_card	5455
201804	debit_card	97
201804	voucher	370
201805	UPI	1263
201805	credit_card	5497
201805	debit_card	51
201805	voucher	324
201806	UPI	1100
201806	credit_card	4813
201806	debit_card	182
201806	voucher	324
201807	UPI	1229
201807	credit_card	4755
201807	debit_card	242
201807	voucher	281
201808	UPI	1139
201808	credit_card	4985
201808	debit_card	277
201808	not_defined	2
201808	voucher	295
201809	not_defined	1
201809	voucher	15
201810	voucher	4

Monthwise top payment type

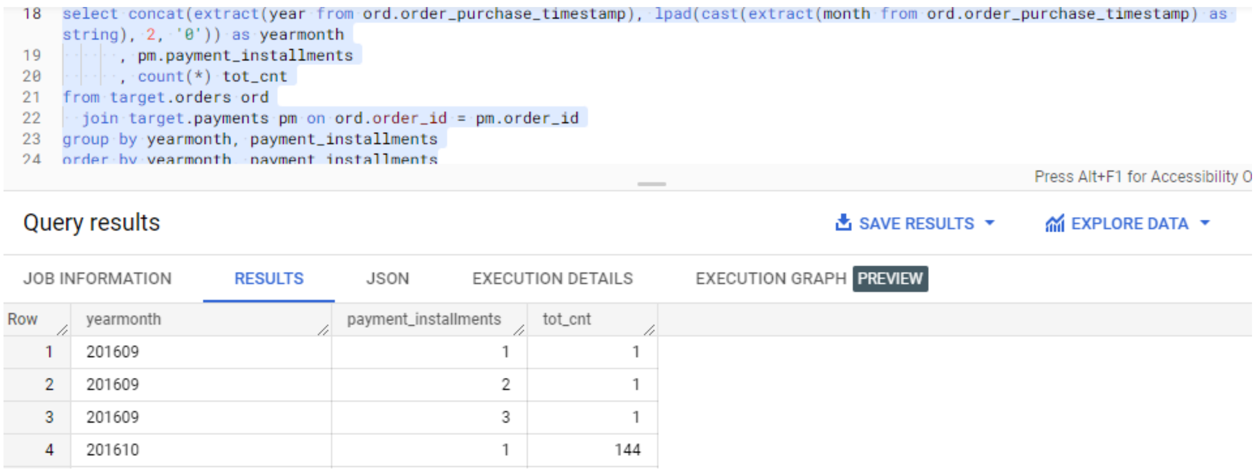


```
with payment_data as
(select concat(extract(year from ord.order_purchase_timestamp), lpad(cast(extract(month from ord.order_purchase_timestamp) as
string), 2, '0')) as yearmonth
        , pm.payment_type
        , count(*) tot_cnt
from target.orders ord
  join target.payments pm on ord.order_id = pm.order_id
group by yearmonth, payment_type
order by yearmonth, payment_type)
select yearmonth, payment_type
from (select *,
              dense_rank() over(partition by yearmonth order by tot_cnt desc) rnk
        from payment_data) t
where rnk = 1
order by yearmonth
```

yearmonth	payment_type
201609	credit_card
201610	credit_card
201612	credit_card
201701	credit_card
201702	credit_card
201703	credit_card
201704	credit_card
201705	credit_card
201706	credit_card
201707	credit_card
201708	credit_card
201709	credit_card
201710	credit_card
201711	credit_card
201712	credit_card
201801	credit_card
201802	credit_card
201803	credit_card
201804	credit_card
201805	credit_card
201806	credit_card
201807	credit_card
201808	credit_card
201809	voucher
201810	voucher

From the above result it is clear that 'Credit Card' payment is the most favored payment type in almost all the months

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



```
select concat(extract(year from ord.order_purchase_timestamp), lpad(cast(extract(month from ord.order_purchase_timestamp) as
string), 2, '0')) as yearmonth
        , pm.payment_installments
        , count(*) tot_cnt
from target.orders ord
  join target.payments pm on ord.order_id = pm.order_id
group by yearmonth, payment_installments
order by yearmonth, payment_installments
```

Row	yearmonth	payment_installments	tot_cnt
1	201609	1	1
2	201609	2	1
3	201609	3	1
4	201610	1	144

yearmonth	payment_installments	tot_cnt
201609	1	1
201609	2	1
201609	3	1
201610	1	144
201610	2	30
201610	3	43
201610	4	26
201610	5	20
201610	6	18
201610	7	13
201610	8	3
201610	9	3
201610	10	42
201612	1	1
201701	1	469
201701	2	72
201701	3	65
201701	4	52
201701	5	39
201701	6	32
201701	7	16
201701	8	42
201701	9	5
201701	10	56
201701	12	2

201702	1	1044
201702	2	194
201702	3	170
201702	4	107
201702	5	90
201702	6	49
201702	7	31
201702	8	82
201702	9	11
201702	10	102
201702	11	1
201702	12	4
201702	17	1
201703	1	1490
201703	2	286
201703	3	269
201703	4	183
201703	5	129
201703	6	120
201703	7	45
201703	8	130
201703	9	24
201703	10	146
201703	12	10
201703	14	1
201703	15	4
201704	1	1268
201704	2	279
201704	3	256
201704	4	164
201704	5	139
201704	6	103
201704	7	46
201704	8	122
201704	9	18
201704	10	165
201704	12	10
201704	15	1
201705	1	1797
201705	2	428
201705	3	416

201705	4	285
201705	5	239
201705	6	198
201705	7	82
201705	8	109
201705	9	47
201705	10	332
201705	11	2
201705	12	5
201705	15	4
201706	1	1580
201706	2	382
201706	3	368
201706	4	267
201706	5	196
201706	6	167
201706	7	78
201706	8	105
201706	9	29
201706	10	259
201706	12	3
201706	13	1
201706	18	1
201707	1	1980
201707	2	506
201707	3	468
201707	4	332
201707	5	292
201707	6	165
201707	7	68
201707	8	109
201707	9	52
201707	10	336
201707	11	2
201707	12	3
201707	14	1
201707	15	1
201707	18	2
201708	1	2153
201708	2	494
201708	3	508

201708	4	370
201708	5	246
201708	6	180
201708	7	81
201708	8	173
201708	9	38
201708	10	292
201708	11	1
201708	12	5
201708	14	2
201708	15	1
201708	18	4
201708	20	2
201709	1	2209
201709	2	508
201709	3	476
201709	4	320
201709	5	235
201709	6	184
201709	7	90
201709	8	202
201709	9	34
201709	10	234
201709	12	10
201709	13	1
201709	15	4
201709	16	1
201709	18	6
201709	20	2
201710	1	2405
201710	2	604
201710	3	525
201710	4	330
201710	5	244
201710	6	187
201710	7	64
201710	8	222
201710	9	26
201710	10	244
201710	11	1
201710	12	4

201710	15	1
201710	16	1
201710	18	1
201710	20	1
201711	1	3863
201711	2	919
201711	3	800
201711	4	549
201711	5	436
201711	6	288
201711	7	140
201711	8	239
201711	9	61
201711	10	514
201711	11	4
201711	12	12
201711	13	5
201711	14	3
201711	15	5
201711	17	1
201711	18	1
201711	20	4
201711	21	3
201711	22	1
201711	24	15
201712	1	3004
201712	2	690
201712	3	589
201712	4	416
201712	5	280
201712	6	209
201712	7	107
201712	8	185
201712	9	34
201712	10	358
201712	11	3
201712	12	7
201712	14	2
201712	15	5
201712	17	1
201712	18	1

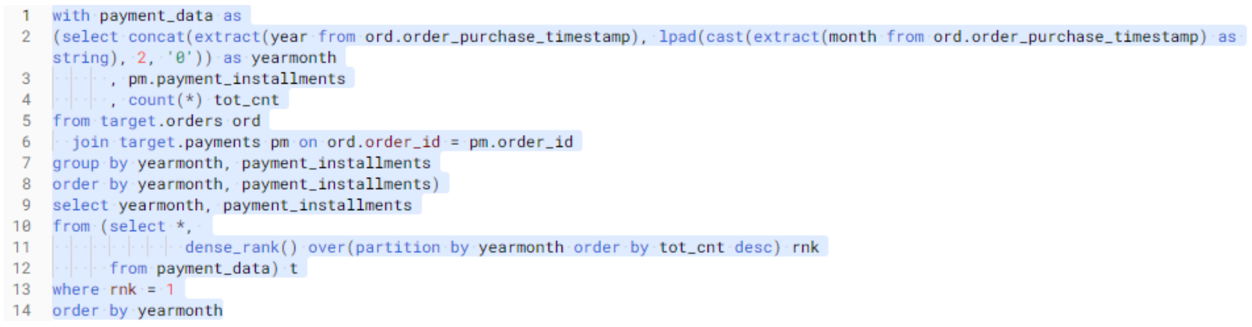
201712	20	3
201712	24	1
201801	1	4076
201801	2	892
201801	3	769
201801	4	500
201801	5	355
201801	6	257
201801	7	97
201801	8	278
201801	9	29
201801	10	290
201801	11	1
201801	12	7
201801	14	2
201801	15	8
201801	18	1
201801	24	1
201802	1	3697
201802	2	923
201802	3	700
201802	4	434
201802	5	296
201802	6	233
201802	7	84
201802	8	300
201802	9	33
201802	10	241
201802	12	5
201802	13	1
201802	15	5
201803	1	3790
201803	2	959
201803	3	755
201803	4	496
201803	5	370
201803	6	290
201803	7	106
201803	8	296
201803	9	46
201803	10	376

201803	11	2
201803	12	12
201803	13	1
201803	14	1
201803	15	11
201803	20	1
201804	0	1
201804	1	3760
201804	2	961
201804	3	678
201804	4	485
201804	5	326
201804	6	245
201804	7	94
201804	8	318
201804	9	31
201804	10	291
201804	11	1
201804	12	12
201804	13	1
201804	15	4
201804	17	1
201805	0	1
201805	1	3609
201805	2	905
201805	3	680
201805	4	465
201805	5	344
201805	6	278
201805	7	106
201805	8	414
201805	9	31
201805	10	286
201805	11	1
201805	12	6
201805	15	6
201805	17	1
201805	18	2
201806	1	3283
201806	2	774
201806	3	625

201806	4	408
201806	5	322
201806	6	289
201806	7	82
201806	8	340
201806	9	26
201806	10	245
201806	11	2
201806	12	7
201806	13	2
201806	15	7
201806	18	4
201806	20	1
201806	23	1
201806	24	1
201807	1	3435
201807	2	795
201807	3	585
201807	4	443
201807	5	334
201807	6	212
201807	7	97
201807	8	321
201807	9	29
201807	10	243
201807	12	2
201807	13	1
201807	14	1
201807	15	3
201807	16	1
201807	17	2
201807	18	3
201808	1	3468
201808	2	811
201808	3	715
201808	4	466
201808	5	307
201808	6	216
201808	7	99
201808	8	278
201808	9	37

201808	10	276
201808	11	2
201808	12	7
201808	13	3
201808	14	2
201808	15	4
201808	16	2
201808	17	1
201808	18	1
201808	20	3
201809	1	16
201810	1	4

MoM highest installment number



```sql
with payment_data as
(select concat(extract(year from ord.order_purchase_timestamp), lpad(cast(extract(month from ord.order_purchase_timestamp) as 
string), 2, '0')) as yearmonth
        , pm.payment_installments
        , count(*) tot_cnt
from target.orders ord
  join target.payments pm on ord.order_id = pm.order_id
group by yearmonth, payment_installments
order by yearmonth, payment_installments)
select yearmonth, payment_installments
from (select *, 
              dense_rank() over(partition by yearmonth order by tot_cnt desc) rnk
      from payment_data) t
where rnk = 1
order by yearmonth
```

yearmonth	payment_installments
201609	1
201609	3
201609	2
201610	1
201612	1
201701	1
201702	1
201703	1
201704	1
201705	1
201706	1
201707	1
201708	1
201709	1
201710	1

201711	1
201712	1
201801	1
201802	1
201803	1
201804	1
201805	1
201806	1
201807	1
201808	1
201809	1
201810	1

Month over Month, It appears that mostly customer prefers to do full payment over creating multiple installments.

Part 7: Insights

It appears that count of orders is Increasing MoM up until November, 2017 where is attains its peak at 7544 orders. However, post this month count is moving in steady rate. Out of all category of purchase, Top category is '*computer accessories*' and lowest is 'Furniture Kitchen Service Area Dinner and Garden'.

Customer count wise, Target has its maximum customer base from SP(39%) , RJ(20%) and MG(19%) while lowest base at AM, AP and RR.
Also, it is observed that SP has the maximum purchase order and major business contributor for Target, followed by 'RS', 'RG' and 'MG'

On the tendency of Ordering time, assuming
when purchase_hours between 0 and 6 then "Dawn"
when purchase_hours between 7 and 12 then "Morning"
when purchase_hours between 13 and 18 then "Afternoon"
when purchase_hours between 18 and 24 then "Night"

Brazilians found to be more active in their purchases during 'AfterNoon' time.

On the financial side, Even though MoM % increase in total sales is not consistent, YoY % increase is positive and increases as company progresses from Jan to Aug.
It is seen that cost is mainly driven by freight value as avg cost is higher in the state where freight cost is also the highest. This indicates that in order to maximize profit, Target must reduce the freight cost by setting up more chains/ warehouse in high-cost states.

From the data it is also seen that, 'Target' has good delivery network (-ve average diff_estimated_delivery indicates, delivery is made before estimated date)
Products with high freight value are generally delivered much before that the expected delivery date.

Among all payment methods, 'Credit Card' is the most favored payment type in almost all the months, customers are also seen to go for full payment over creating multiple installments.

Part 8: Recommendations

1. ‘Target’ has excellent delivery Network and that could be the strength in driving the successful business, However they should also focus on making goods available at major hubs as high freight cost is mainly driving factor for overall products cost, thus limiting the profit margin
2. Afternoon being the crucial timing for the business, ‘Target’ needs to align work timing and expert call center agents accordingly. They can also focus on cross-selling with the help of efficient strategies during this time.
3. Credit card being the most favored payment mode, ‘Target’ can enhance the spend tendency by making tie-ups with bank to give an extra perks(rewards, cashbacks, gift vouchers) with their shopping. But should also encourage customer to use other means of payment.