Context

Target is one of the world's most recognized brands and one of America's leading retailers. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This business case has information of 100k orders from 2016 to 2018 made at Target in Brazil. Its features allows viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

Customers:

```
1  SELECT * FROM `targetsql-368319.targetsql.customers` LIMIT 10;
```

Query results

SAVE RESULTS

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

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

Gelocations :

RUN | SAVE | SHARE | SCHEDULE | MORE

```
1  SELECT * FROM `targetsql-368319.targetsql.geolocation` LIMIT 10;
```

Query results

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

Row	geolocation...	geolocation...	geolocation...	geolocation_city	geolocation_state
1	49010	-10.910514...	-37.052400...	aracaju	SE
2	49047	-10.9268145	-37.071063...	aracaju	SE
3	49030	-10.970164...	-37.061643...	aracaju	SE
4	49048	-10.940183...	-37.070850...	aracaju	SE
5	49050	-10.927157...	-37.063078...	aracaju	SE
6	49015	-10.923370...	-37.045169...	aracaju	SE
7	49045	-10.930406...	-37.067178...	aracaju	SE
8	49052	-10.922973...	-37.057752...	aracaju	SE
9	49044	-10.992080...	-37.103470...	aracaju	SE
10	49048	-10.940235...	-37.071043...	aracaju	SE

Order_items :

```
1  SELECT * FROM `targetsql-368319.targetsql.order_items` LIMIT 10;
```

Press Alt+F1 for accessibility options

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
1	f09e36e258656850b92657ac5...	1	44d53f1240d6332232e4393c0...	b64d51f0435e884e8de603b16...	2018-07-09 13:31:36 UTC	3.0	12.79
2	f9ccaff7267fd0cf076e795b1fa...	1	44d53f1240d6332232e4393c0...	b64d51f0435e884e8de603b16...	2018-08-14 14:04:44 UTC	3.0	15.23
3	c79bdf061e22288609201ec60...	1	5304ff3fa35856a156e1170a60...	cf6f6bc4df3999b9c6440f124f...	2017-05-12 19:05:20 UTC	3.5	8.72
4	37193e64eb9a46b7f3197762f...	1	98224bfc1eaadb3a394ec334c...	ce616e1913288884e7742faac...	2018-06-28 01:30:49 UTC	3.5	7.39
5	95d6357ffe41aa6d2998852a7...	1	98224bfc1eaadb3a394ec334c...	ce616e1913288884e7742faac...	2018-06-12 19:15:14 UTC	3.5	18.23
6	95d6357ffe41aa6d2998852a7...	2	98224bfc1eaadb3a394ec334c...	ce616e1913288884e7742faac...	2018-06-12 19:15:14 UTC	3.5	18.23
7	95d6357ffe41aa6d2998852a7...	3	98224bfc1eaadb3a394ec334c...	ce616e1913288884e7742faac...	2018-06-12 19:15:14 UTC	3.5	18.23
8	95d6357ffe41aa6d2998852a7...	4	98224bfc1eaadb3a394ec334c...	ce616e1913288884e7742faac...	2018-06-12 19:15:14 UTC	3.5	18.23
9	95d6357ffe41aa6d2998852a7...	5	98224bfc1eaadb3a394ec334c...	ce616e1913288884e7742faac...	2018-06-12 19:15:14 UTC	3.5	18.23
10	dde867f83e689b0167785b684...	1	914323edd50192310dd03435...	2c9e548be18521d1c43cde1c5...	2017-10-20 14:50:12 UTC	4.5	11.85

Order_reviews :

```
1  SELECT * FROM `targetsql-368319.targetsql.order_reviews` LIMIT 10;
```

Press Alt+F1 for accessibility optior

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	review_id	order_id	review_score	review_comment_title	review_creation_date	review_answer_timestamp
1	be7e2989673cb2a147b87ba73...	777c67eab7c0712ccde8ffbb2...	1	null	0001-04-17 00:00:00 UTC	0001-04-17 07:40:00 UTC
2	e12151267e4594d69eda14a87...	4338a4463f7f9193d2a03a83a...	1	null	0001-04-17 00:00:00 UTC	0001-04-17 09:04:00 UTC
3	41d614b133efebcd10352001b...	b8aaeda740b17cf925d3f2a13...	1	null	0001-04-17 00:00:00 UTC	0002-04-17 03:48:00 UTC
4	c950324a42c5796d06f569f77...	b159d0ce7cd881052da94fa16...	1	null	0001-04-17 00:00:00 UTC	0001-04-17 10:24:00 UTC
5	76823ada94c8861ecebcfbc7c...	2a3007ed051b02a0e0dd0709c...	1	null	0001-04-17 00:00:00 UTC	0002-04-17 13:58:00 UTC
6	fe270df00abcb5c39fc7385143...	a39d3db795a5cf4c8b6c9dd05...	1	null	0001-04-17 00:00:00 UTC	0003-04-17 12:49:00 UTC
7	1b71e0b29ec2faa0a029ded43...	0e530f6be154c9d7e7b12f341...	1	null	0001-04-17 00:00:00 UTC	0010-04-17 12:45:00 UTC
8	efe4020a945ee6fece58606694...	264c045399fb02e9f309f715c2...	1	null	0001-04-17 00:00:00 UTC	0002-04-17 01:16:00 UTC
9	23ee6bc2492808dca0e807e13...	2ba1366baecad3c3536f27546...	1	null	0001-04-17 00:00:00 UTC	0001-04-17 14:34:00 UTC
10	63c6cec8cb9a2d0990b339998...	aaae80f5b6239bd9e1b22e9aa...	1	null	0001-04-17 00:00:00 UTC	0004-04-17 12:07:00 UTC

orders :

```
1  SELECT * FROM `targetsql-368319.targetsql.orders` LIMIT 10;
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_ca
1	7a4df5d8cff4090e541401a20a...	725e9c75605414b21fd8c8d5a...	created	2017-11-25 11:10:33 UTC	null	null
2	35de4050331c6c644cddc86f4...	4ee64f4bfc542546f422da0aeb...	created	2017-12-05 01:07:58 UTC	null	null
3	b5359909123fa03c50bdb0cfe...	438449d4af8980d107bf04571...	created	2017-12-05 01:07:52 UTC	null	null
4	dba5062fbda3af4fb6c33b1e04...	964a6df3d9bdf60fe3e7b8bb69...	created	2018-02-09 17:21:04 UTC	null	null
5	90ab3e7d52544ec7bc3363c82...	7d61b9f4f216052ba664f22e9c...	created	2017-11-06 13:12:34 UTC	null	null
6	fa65dad1b0e818e3ccc5cb0e3...	9af2372a1e49340278e7c1ef8...	shipped	2017-04-20 12:45:34 UTC	2017-04-22 09:10:13 UTC	2017-04-24 11:31:1
7	1df2775799eecdf9dd8502425...	1240c2e65c4601dd860e3a367...	shipped	2017-07-13 11:03:05 UTC	2017-07-13 11:10:22 UTC	2017-07-18 18:17:3
8	6190a94657e1012983a274b8...	5fc4c97dcb63903f996714524...	shipped	2017-07-11 13:36:30 UTC	2017-07-11 13:45:15 UTC	2017-07-13 17:55:4
9	58ce513a55c740a3a81e8c8b7...	530d41b47b9dda9bc6f31d856...	shipped	2017-07-29 18:05:07 UTC	2017-07-29 18:15:17 UTC	2017-07-31 16:41:5
10	088683f795a3d30bfd61152c4f...	58d89fd1f863819ff9b040734f...	shipped	2017-07-13 10:02:47 UTC	2017-07-14 02:25:54 UTC	2017-07-20 20:02:5

payments :

```
1  SELECT * FROM `targetsql-368319.targetsql.payments` LIMIT 10;
```

Query results

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

Row	order_id	payment_se...	payment_type	payment_in...	payment_va...
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

Products :

```
1  SELECT * FROM `targetsql-368319.targetsql.products` LIMIT 10;
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	product_id	product_category	product_na...	product_des...	product_ph...	product_wei...	product_len...	product_hei...	product_wid...
1	5eb564652db742ff8f28759cd8...	null	null	null	null	null	null	null	null
2	09ff539a621711667c43eba6a...	babies	60	865	3	null	null	null	null
3	2f763ba79d9cd987b2034aac7...	electronics	45	1198	2	595	8	6	6
4	a69f15dfb803d485e8933e80b...	Watches present	53	506	6	150	11	16	6
5	e1cfc87f543782b8a78b59fc85...	Garden tools	39	524	4	369	26	7	7
6	106392145fca363410d287a81...	bed table bath	58	309	1	2083	12	2	7
7	7e33f4a1c59f89da30a335b2d...	electronics	51	381	3	1075	22	5	7
8	bc9cc914f974963c07be697fc...	HEALTH BEAUTY	55	435	1	75	14	9	7
9	5ae533eac9c0e93b3f89bc9ae...	computer accessories	58	1340	1	83	12	8	8
10	67d1a56495104e195338ec90...	pet Shop	20	2153	1	275	8	13	8

Sellers :

```
1  SELECT * FROM `targetsql-368319.targetsql.sellers` LIMIT 10;
```

Query results

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

Row	seller_id	seller_zip_c...	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

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

A. Data type of columns in a table

The below query yields the below result for the customers table. We can see the customers table has all string variables and one variable customer_zip_code_prefix is an integer

Customers:

*Unsaved query

RUN | SAVE | SHARE | SCHEDULE | MORE

```
1  SELECT
2    COLUMN_NAME, DATA_TYPE
3  FROM
4    targetsql.INFORMATION_SCHEMA.COLUMNS
5    WHERE TABLE_NAME='customers';
```

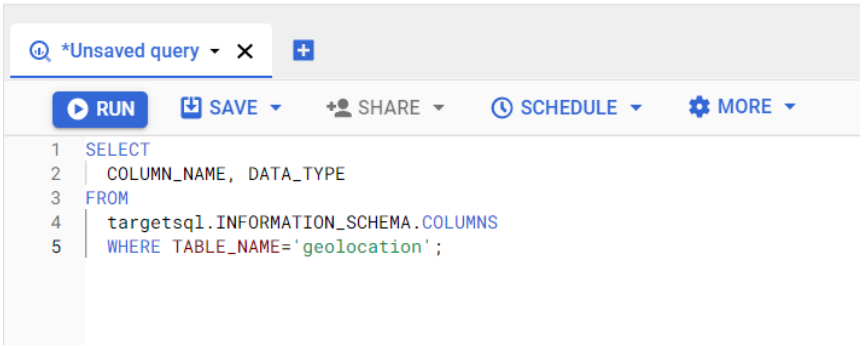
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

We can run the same query for all the other datasets/tables(replacing the table name in the above query) and see the data type.

Geolocation:



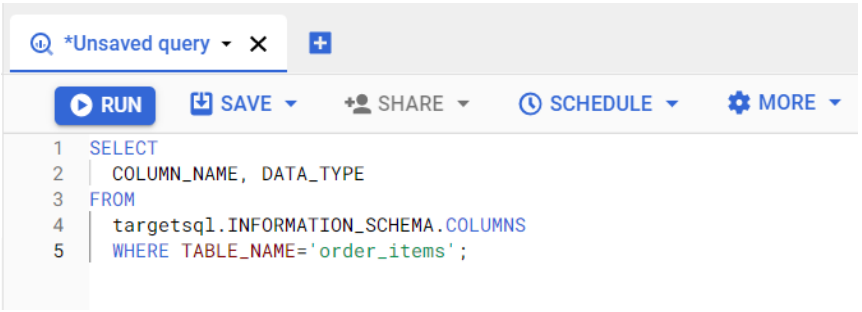
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

Since the orders table is supposed to have details of delivery details, we have a new data type as TIMESTAMP for some of the variables.

Order_items:

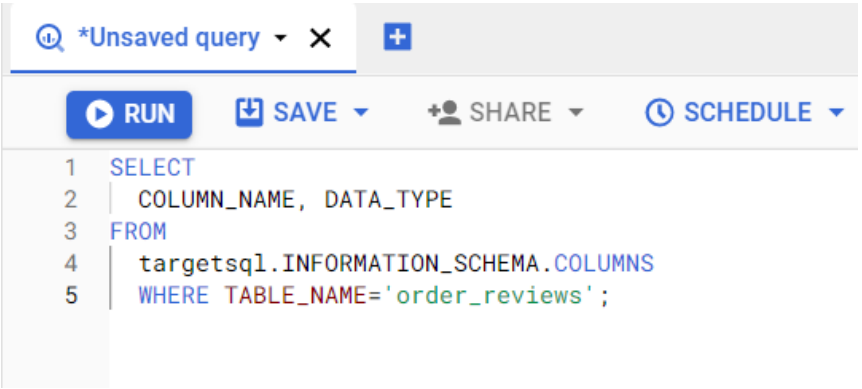


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	COLUMN_NAME	DATA_TYPE
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64

order_reviews:

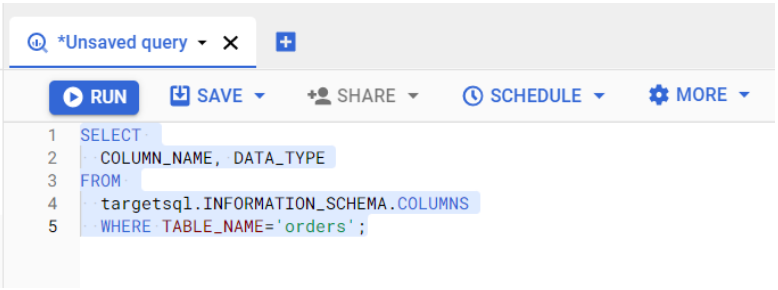


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

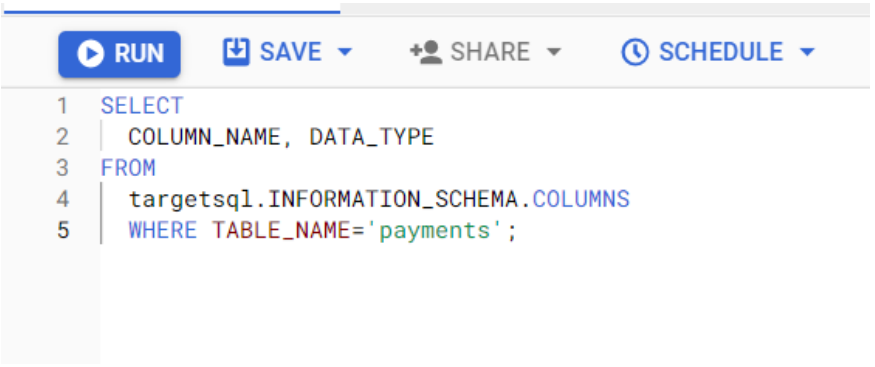
orders:



JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

payments:



JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

products:

RUN SAVE SHARE SCHEDULE

```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  targetsql.INFORMATION_SCHEMA.COLUMNS
  WHERE TABLE_NAME='products';
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	COLUMN_NAME	DATA_TYPE
1	product_id	STRING
2	product_category	STRING
3	product_name_length	INT64
4	product_description_length	INT64
5	product_photos_qty	INT64
6	product_weight_g	INT64

sellers:

RUN SAVE SHARE SCHEDULE

```
SELECT
  COLUMN_NAME, DATA_TYPE
FROM
  targetsql.INFORMATION_SCHEMA.COLUMNS
  WHERE TABLE_NAME='sellers';
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

B. Time period for which the data is given

The 1st order purchase date is of 4th September 2016 and the latest purchase date is 17th October 2018. So, the orders table contains orders for the period of 773 days starting from 4th September 2016.

RUN SAVE SHARE SCHEDULE MORE

```
SELECT min(date(a.order_purchase_timestamp))
FROM `targetsql-368319.targetsql.orders` as A;

SELECT max(date(a.order_purchase_timestamp))
FROM `targetsql-368319.targetsql.orders` as A;

SELECT (max(date(a.order_purchase_timestamp)) - min(date(a.order_purchase_timestamp))) as duration From targetsql.orders as A;
```

C. Cities and States covered in the dataset

If we take the customers and orders table, we can see there are 4310 distinct states and cities covered.

```
SELECT distinct customer_city, customer_state FROM `targetsql-368319.targetsql.customers` A , `targetsql-368319.targetsql.orders` B
WHERE A.customer_id=B.customer_id;
```

Press

Query results

SAVE RESULTS

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

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

Results per page: 50 1 – 50 of 4310

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?**
    - **A. How has the business increased/decreased over the entire duration?**

```
SELECT year, count(order_id) as orders_count from
(select* , extract(year from order_purchase_timestamp) as year from `targetsql-368319.targetsql.orders`
)
group by year
order by year;
```

Query results

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

Row	year	orders_count
1	2016	329
2	2017	45101
3	2018	54011

We can see there is a drastic increase in the number of orders in 2017 and it has increased significantly in the year 2018 as well.

```
14 select year,month, count(order_id) as orders_count from
15 (
16 select *, EXTRACT(year FROM order_purchase_timestamp) as year , EXTRACT(month FROM order_purchase_timestamp) as month from `targetsql-368319.targetsql.
   orders`
17 )
18 group by year, month
19 order by year, month;
20
```

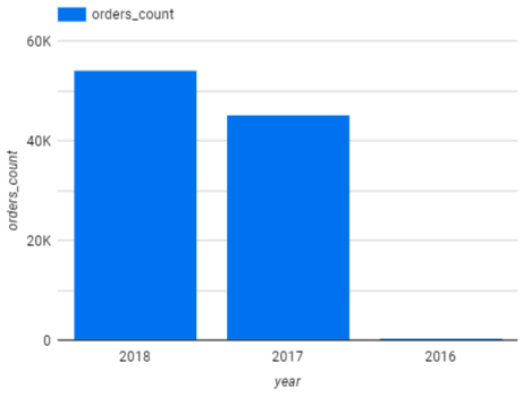
Row	year	month	orders_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331

Row	year	month	orders_count
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544
15	2017	12	5673
16	2018	1	7269
17	2018	2	6728
18	2018	3	7211
19	2018	4	6939
20	2018	5	6873
21	2018	6	6167
22	2018	7	6292

22	2018	7	6292
23	2018	8	6512
24	2018	9	16
25	2018	10	4

	year	orders_count
1.	2018	54,011
2.	2017	45,101
3.	2016	329

1 - 3 / 3



	month	orders_count
1.	August	10,843
2.	May	10,573
3.	July	10,318
4.	March	9,893
5.	June	9,412
6.	April	9,343
7.	February	8,508
8.	January	8,069
9.	November	7,544
10.	December	5,674
11.	October	4,959

1 - 12 / 12

We can clearly see a peak in sales in the months of August, May, and July. Also, in the month of November and December in 2017, we have a peak in sales which probably be due to the festive season.

2. **What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?**
    A. **Do we have any trend in buying as per the time of the day?**

```sql
SELECT
CASE
  WHEN
    EXTRACT(HOUR FROM o.order_purchase_timestamp)>=4 and
    EXTRACT(HOUR FROM o.order_purchase_timestamp)<=6 THEN 'DAWN'
  WHEN
    EXTRACT(HOUR FROM o.order_purchase_timestamp)>6 AND
    EXTRACT(HOUR FROM o.order_purchase_timestamp)<12 THEN 'MORNING'
  WHEN
    EXTRACT(HOUR FROM o.order_purchase_timestamp)>=12 AND
    EXTRACT(HOUR FROM o.order_purchase_timestamp)<=18 THEN 'AFTERNOON'
ELSE 'NIGHT'
  END AS TIME_OF_DAY, count(distinct(o.order_id)) as orders_placed
FROM `targetsql-368319.targetsql.orders` o GROUP BY TIME_OF_DAY order by orders_placed desc ;
```

Query results

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

Row	TIME_OF_DAY	orders_plac...
1	AFTERNOON	44130
2	NIGHT	32677
3	MORNING	21738
4	DAWN	896

B. Which date of the month had the highest sales:

```sql
select year , month, max(day_highest_no_of_orders) day_highest_no_of_orders, max(max_orders) orders from
  (select year, month, nth_value(day,1) over (partition by year, month order by orders_placed desc range between unbounded preceding and unbounded following )
  as day_highest_no_of_orders, max(orders_placed) over (partition by year, month) as max_orders from
(
select EXTRACT(year FROM o.order_purchase_timestamp) as year,EXTRACT(month FROM o.order_purchase_timestamp) as month,EXTRACT(day FROM o.order_purchase_timestamp) as day, count(o.order_id) as orders_placed
FROM `targetsql-368319.targetsql.orders` o GROUP BY year, month, day order by year asc, month asc, orders_placed desc
)

order by year, month)
group by year, month
order by year, month
;
```

Row	year	month	day_highest...	orders
1	2016	9	4	1
2	2016	10	4	63
3	2016	12	23	1
4	2017	1	26	86
5	2017	2	7	112
6	2017	3	20	119
7	2017	4	26	125
8	2017	5	29	160
9	2017	6	19	156
10	2017	7	18	192
11	2017	8	15	194
12	2017	9	13	207

Row	year	month	day_highest...	orders
14	2017	11	24	1176
15	2017	12	4	337
16	2018	1	22	314
17	2018	2	28	313
18	2018	3	19	303
19	2018	4	19	293
20	2018	5	7	372
21	2018	6	11	294
22	2018	7	31	322
23	2018	8	6	372
24	2018	9	3	4
25	2018	10	1	1

Evolution of E-commerce orders in the Brazil region:

A. States with the highest number of orders month on month:

```
select distinct year, month, nth_value(customer_state,1) over (partition by year, month order by cnt desc range between unbounded preceding and
unbounded following) as highest_orders_state from (

 select customer_state, EXTRACT(year FROM o.order_purchase_timestamp) year, EXTRACT(month FROM o.order_purchase_timestamp) month, count(order_id) as cnt
 from `targetsql-368319.targetsql.customers` c, `targetsql-368319.targetsql.orders` o
 where o.customer_id=c.customer_id
 group by customer_state, year , month
 order by year, cnt desc)

```

Row	year	month	highest_orders_state
1	2016	9	SP
2	2016	10	SP
3	2016	12	PR
4	2017	1	SP
5	2017	2	SP
6	2017	3	SP
7	2017	4	SP
8	2017	5	SP
9	2017	6	SP
10	2017	7	SP
11	2017	8	SP

Row	year	month	highest_orders_state
15	2017	12	SP
16	2018	1	SP
17	2018	2	SP
18	2018	3	SP
19	2018	4	SP
20	2018	5	SP
21	2018	6	SP
22	2018	7	SP
23	2018	8	SP
24	2018	9	SP
25	2018	10	SP

B. Which region/postal code has the highest number of sales month by month:

```
select distinct year, month, nth_value(customer_zip_code_prefix,1) over (partition by year, month order by cnt desc range between unbounded preceding
and unbounded following) as highest_order_region from (

select customer_zip_code_prefix, EXTRACT(year FROM o.order_purchase_timestamp) year, EXTRACT(month FROM o.order_purchase_timestamp) month, count
(order_id) as cnt
from `targetsql-368319.targetsql.customers` c, `targetsql-368319.targetsql.orders` o
where o.customer_id=c.customer_id
group by customer_zip_code_prefix, year, month
order by year, cnt desc)
;
```

Row	year	month	highest_ord...
1	2016	9	69309
2	2016	10	20511
3	2016	12	80030
4	2017	1	80030
5	2017	2	22775
6	2017	3	35500
7	2017	4	13280
8	2017	5	22790

C. The region with the highest number of orders:

```
select customer_zip_code_prefix, EXTRACT(month FROM o.order_purchase_timestamp) month, count(order_id) as cnt
from `targetsql-368319.targetsql.customers` c, `targetsql-368319.targetsql.orders` o
where o.customer_id=c.customer_id
group by customer_zip_code_prefix, month
order by month, cnt desc ;
```

Row	customer_zi...	month	cnt
1	22793	1	15
2	80030	1	15
3	22775	1	14
4	22790	1	14
5	36400	1	12
6	35500	1	11
7	24230	1	11
8	20541	1	10

D. **Region/zip code with the highest number of customers:**

```
72  select customer_zip_code_prefix, count(customer_id) as count_of_customers
73  from `targetsql-368319.targetsql.customers`
74  group by customer_zip_code_prefix
75  order by count_of_customers desc limit 10;
76
```

Row	customer_zi...	count_of_cu...
1	22790	142
2	24220	124
3	22793	121
4	24230	117
5	22775	110
6	29101	101
7	13212	95
8	35162	93
9	22631	89

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

A. **How Business expanded in terms of money for Jan to August months of the year.**

```sql
select *,((lead_order_amount - total_order_cost)/ total_order_cost)*100 as percent_inc from (
select *, lead(total_order_cost) over (order by total_order_cost ) as lead_order_amount from
(SELECT extract(year from o.order_purchase_timestamp) year,
ROUND(SUM(oi.price+oi.freight_value),2) as total_order_cost
from `targetsql-368319.targetsql.order_items` oi join `targetsql-368319.targetsql.orders` o
on oi.order_id=o.order_id

WHERE EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
group by year
order by year));
```

Query results

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

Row	year	total_order_...	lead_order_...	percent_inc
1	2017	3610270.15	8643531.14	139.415079...
2	2018	8643531.14	null	null

There is a whopping increase of 139% for total business done in terms of money from 2017 to 2018.

B. Top 5 states with the highest average freight value?

```sql
select c.customer_state,round(AVG(oi.freight_value),2) as AVG_FREIGHT_VALUE
from `targetsql-368319.targetsql.customers` c join `targetsql-368319.targetsql.orders` o
on c.customer_id=o.customer_id
join `targetsql-368319.targetsql.order_items` oi
on o.order_id=oi.order_id
GROUP BY c.customer_state
order by AVG_FREIGHT_VALUE DESC LIMIT 5;
```

Query results

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

Row	customer_state	AVG_FREIG...
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

C. Bottom 5 states with the lowest total freight value:

```sql
select c.customer_state,round(AVG(oi.freight_value),2) as AVG_FREIGHT_VALUE
from `targetsql-368319.targetsql.customers` c join `targetsql-368319.targetsql.orders` o
on c.customer_id=o.customer_id
join `targetsql-368319.targetsql.order_items` oi
on o.order_id=oi.order_id
GROUP BY c.customer_state
order by AVG_FREIGHT_VALUE ASC LIMIT 5;
```

Query results

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

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

D. Top 5 states with the highest total freight value:

```sql
select c.customer_state,round(sum(oi.freight_value),2) as total_freight_value
from `targetsql-368319.targetsql.customers` c join `targetsql-368319.targetsql.orders` o
on c.customer_id=o.customer_id
join `targetsql-368319.targetsql.order_items` oi
on o.order_id=oi.order_id
GROUP BY c.customer_state
order by total_freight_value DESC LIMIT 5;
```

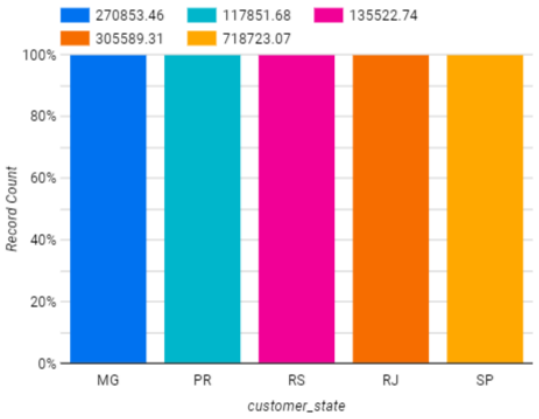
Query results

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

Row	customer_state	total_freight...
1	SP	718723.07
2	RJ	305589.31
3	MG	270853.46
4	RS	135522.74
5	PR	117851.68

	customer_state	Record Count
1.	MG	1
2.	PR	1
3.	RS	1
4.	RJ	1
5.	SP	1

1 - 5 / 5



From the above results on average and total freight values, we can see the relationship between freight value and sales. The lower freight value enhances the chance for more selling of the products.

Analysis on sales, freight, and delivery time

A. 5 states with the highest average delivery time:

```
105  Select c.customer_state,
106  round(avg(date_diff(order_delivered_customer_date,order_purchase_timestamp, day)),2)
107  as avg_delivery_time
108  from `targetsql-368319.targetsql.customers` c join `targetsql-368319.targetsql.orders` o
109  on c.customer_id=o.customer_id
110  group by c.customer_state
111  order by avg_delivery_time desc limit 5;
112
```

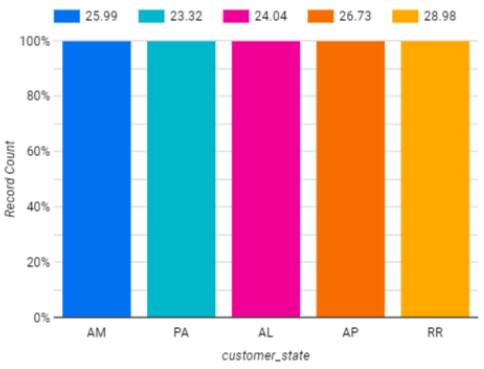
Query results

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

Row	customer_state	avg_delivery...
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

	customer_state	Record Count
1.	AM	1
2.	PA	1
3.	AL	1
4.	AP	1
5.	RR	1

1 - 5 / 5



B. 5 states where delivery is very fast:

```
113 select c.customer_state,round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)),2) as ac_est_del_diff
114 from `targetsql-368319.targetsql.customers` c join `targetsql-368319.targetsql.orders` o
115 on c.customer_id=o.customer_id
116 group by c.customer_state
117 order by ac_est_del_diff desc limit 5;
118
119
120
```

Press Al

Query results

SAVE RESULTS EXI

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

Row	customer_state	ac_est_del_...
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

C. 5 states where delivery rate is slow:

```
119 select c.customer_state,round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)),2) as ac_est_del_diff
120 from `targetsql-368319.targetsql.customers` c join `targetsql-368319.targetsql.orders` o
121 on c.customer_id=o.customer_id
122 group by c.customer_state
123 order by ac_est_del_diff asc limit 5;
124
125
126
```

Press Al

Query results

SAVE RESULTS EXI

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

Row	customer_state	ac_est_del_...
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

D. The average difference between purchase and estimated delivery time:

```sql
SELECT round(avg(date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)),2) as avg_estim_delivery_time
from `targetsql-368319.targetsql.orders`;
```

Query results

SAVE RESULTS

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

Row	avg_estim_d...
1	23.4

E. Top 10 delayed estimated delivery times:

```sql
SELECT date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as estim_delivery_time_in_days
from `targetsql-368319.targetsql.orders`
order by estim_delivery_time_in_days desc limit 10;
```

Processing location: EU

Query results

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

Row	estim_deliv...
1	155
2	149
3	146
4	144
5	144
6	142
7	140
8	116
9	109
10	106

F. Correlation between review score and estimated delivery time:

```sql
SELECT review_score, EXTRACT(year FROM order_purchase_timestamp) as year,
round(avg(date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)),2) as avg_estim_delivery_time_in_days
from `targetsql-368319.targetsql.orders` O, `targetsql-368319.targetsql.order_reviews` R
where O.order_id=R.order_id
group by year, review_score
order by year, review_score;
```

Processing location: EU

Query results

SAVE RESULTS

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

Row	review_score	year	avg_estim_d...
1	1	2016	52.25
2	2	2016	52.0
3	3	2016	54.5
4	4	2016	55.16
5	5	2016	55.72
6	1	2017	25.14
7	2	2017	25.13
8	3	2017	24.63
9	4	2017	24.53

```
SELECT review_score, EXTRACT(year FROM order_purchase_timestamp) as year,
round(avg(date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)),2) as avg_estim_delivery_time_in_days
from `targetsql-368319.targetsql.orders` O, `targetsql-368319.targetsql.order_reviews` R
where O.order_id=R.order_id
group by year, review_score
order by year, review_score;
```

Processing location: EU

Query results

SAVE RESULTS

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

Row	review_score	year	avg_estim_d...
7	2	2017	25.13
8	3	2017	24.63
9	4	2017	24.53
10	5	2017	24.07
11	1	2018	23.31
12	2	2018	23.27
13	3	2018	23.2
14	4	2018	22.73
15	5	2018	21.99

From the above table, we can see the review score does not seem to be much affected by estimated delivery time. Also, we can see that the average estimated delivery time has improved a lot with time.

G. check if the review score has any relation to early delivery than the estimated delivery.

```
SELECT review_score, EXTRACT(year FROM order_purchase_timestamp) as year,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as diff_actual_est_delivery,
count(review_score) as cnt
from `targetsql-368319.targetsql.orders` O, `targetsql-368319.targetsql.order_reviews` R
where O.order_id=R.order_id
group by year, review_score
order by year, review_score;
```

Processing location: EU

Query results

SAVE RESUI

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

Row	review_score	year	diff_actual_...	cnt
1	1	2016	27.38	89
2	2	2016	34.17	8
3	3	2016	34.62	22
4	4	2016	34.76	51
5	5	2016	38.47	156
6	1	2017	4.29	5049
7	2	2017	8.51	1466
8	3	2017	10.43	3815

```sql
SELECT review_score, EXTRACT(year FROM order_purchase_timestamp) as year,
round(avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as diff_actual_est_delivery,
count(review_score) as cnt
from `targetsql-368319.targetsql.orders` O, `targetsql-368319.targetsql.order_reviews` R
where O.order_id=R.order_id
group by year, review_score
order by year, review_score;
```

Processing location: EU

Query results

SAVE RESULTS

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

Row	review_score	year	diff_actual_...	cnt
8	3	2017	10.43	3815
9	4	2017	11.67	8889
10	5	2017	12.78	25825
11	1	2018	2.65	6286
12	2	2018	7.17	1677
13	3	2018	9.29	4342
14	4	2018	11.11	10202
15	5	2018	11.99	31347

H. Total cost of all the orders for the entire duration:

```sql
select round(sum(freight_value+price),2) as total_cost from `targetsql-368319.targetsql.order_items`;
```

Processing location: EU

Query results

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

Row	total_cost
1	15843553.24

I. Top 10 orders in terms of cost across the entire duration:

```
select order_id, round(sum(freight_value+price),2) total_cost from `targetsql-368319.targetsql.order_items`
group by order_id
order by total_cost desc limit 10;
```

Processing location: EU

Query results

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

Row	order_id	total_cost
1	03caa2c082116e1d31e67e9ae...	13664.08
2	736e1922ae60d0d6a89247b8...	7274.88
3	0812eb902a67711a1cb742b3c...	6929.31
4	fefacc66af859508bf1a7934ea...	6922.21
5	f5136e38d1a14a4dbd87dff67d...	6726.66
6	2cc9089445046817a7539d90...	6081.54
7	a96610ab360d42a2e5335a39...	4950.34
8	b4c4b76c642808cbe472a32b8...	4809.44
9	199af31afc78c699f0dbf71fb1...	4764.34
10	8dbc85d1447242f3b127dda39...	4681.78

Payment Type Analysis:

A. Month over month order count for various payment methods:

```
SELECT distinct(p.payment_type) as payment_method, extract(month from o.order_purchase_timestamp) as month,
count(distinct(p.order_id)) as order_count from `targetsql-368319.targetsql.orders` o join `targetsql-368319.targetsql.payments` p
on o.order_id=p.order_id
group by Month,payment_method
order by order_count DESC;
```

Processing location: EU

Query results

SAVE RESULTS

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

Row	payment_method	month	order_count
1	credit_card	5	8308
2	credit_card	8	8235
3	credit_card	7	7810
4	credit_card	3	7682
5	credit_card	4	7276
6	credit_card	6	7248
7	credit_card	2	6582
8	credit_card	1	6093
9	credit_card	11	5867

```sql
SELECT distinct(p.payment_type) as payment_method, extract(month from o.order_purchase_timestamp) as month,
count(distinct(p.order_id)) as order_count from `targetsql-368319.targetsql.orders` o join `targetsql-368319.targetsql.payments` p
on o.order_id=p.order_id
group by Month,payment_method
order by order_count DESC;
```

Press

Query results

SAVE RESULTS | E

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

Row	payment_method	month	order_count
10	credit_card	12	4364
11	credit_card	10	3763
12	credit_card	9	3277
13	UPI	8	2077
14	UPI	7	2074
15	UPI	5	2035
16	UPI	3	1942
17	UPI	6	1807
18	UPI	4	1783
19	UPI	2	1723


```sql
SELECT distinct(p.payment_type) as payment_method, extract(month from o.order_purchase_timestamp) as month,
count(distinct(p.order_id)) as order_count from `targetsql-368319.targetsql.orders` o join `targetsql-368319.targetsql.payments` p
on o.order_id=p.order_id
group by Month,payment_method
order by order_count DESC;
```


Press Alt+F1 fo

Query results

SAVE RESULTS | EXPLORE

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

Row	payment_method	month	order_count
20	UPI	1	1715
21	UPI	11	1509
22	UPI	12	1160
23	UPI	10	1056
24	UPI	9	903
25	voucher	8	430
26	voucher	7	417
27	voucher	3	395
28	voucher	5	374
29	voucher	6	373

Results per page: 50 | 1 – 50 of 50


```sql
SELECT distinct(p.payment_type) as payment_method, extract(month from o.order_purchase_timestamp) as month,
count(distinct(p.order_id)) as order_count from `targetsql-368319.targetsql.orders` o join `targetsql-368319.targetsql.payments` p
on o.order_id=p.order_id
group by Month,payment_method
order by order_count DESC;
```


Pres

Query results

SAVE RESULTS

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

Row	payment_method	month	order_count
30	voucher	4	353
31	voucher	1	337
32	debit_card	8	311
33	voucher	2	288
34	voucher	11	267
35	debit_card	7	264
36	voucher	10	223
37	voucher	12	220
38	debit_card	6	208
39	voucher	9	189

```sql
SELECT distinct(p.payment_type) as payment_method, extract(month from o.order_purchase_timestamp) as month,
count(distinct(p.order_id)) as order_count from `targetsql-368319.targetsql.orders` o join `targetsql-368319.targetsql.payments` p
on o.order_id=p.order_id
group by Month,payment_method
order by order_count DESC;
```

Press A

Query results

SAVE RESULTS EX

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

Row	payment_method	month	order_count
41	debit_card	1	118
42	debit_card	3	109
43	debit_card	2	82
44	debit_card	5	81
45	debit_card	11	70
46	debit_card	12	64
47	debit_card	10	54
48	debit_card	9	43
49	not_defined	8	2
50	not_defined	9	1

Results per page: 50 1 – 50 of 50

B. Order count payment method wise:

```sql
select distinct(payment_type) as payment_methods,
count(distinct(order_id)) as no_of_order from `targetsql-368319.targetsql.payments`
group by payment_methods
order by no_of_order desc;
```

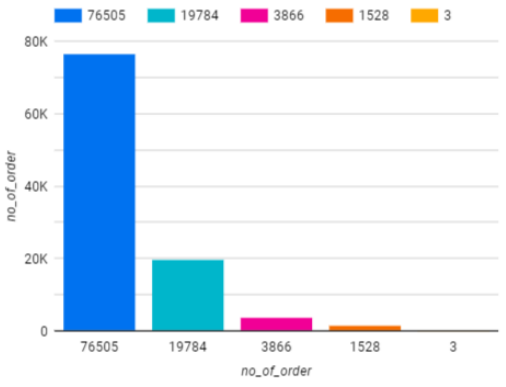
Query results

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

Row	payment_methods	no_of_order
1	credit_card	76505
2	UPI	19784
3	voucher	3866
4	debit_card	1528
5	not_defined	3

	payment_methods	no_of_order
1.	credit_card	76,505
2.	UPI	19,784
3.	voucher	3,866
4.	debit_card	1,528
5.	not_defined	3

1 - 5 / 5



Exploration on the data:

A. Top 10 and Bottom 10 cities for sellers registered with Target:

```
35  SELECT seller_city, COUNT(DISTINCT(seller_id)) as sellers_count FROM `targetsql-368319.targetsql.sellers`
36  GROUP BY seller_city
37  ORDER BY sellers_count
38  DESC LIMIT 10;
```

Query results

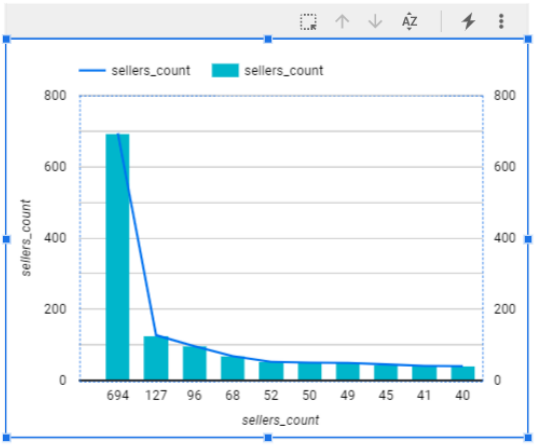
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	seller_city	sellers_count
1	sao paulo	694
2	curitiba	127
3	rio de janeiro	96
4	belo horizonte	68
5	ribeirao preto	52
6	guarulhos	50
7	ibitinga	49
8	santo andre	45
9	campinas	41
10	maringa	40

sellers

	seller_city	sellers_count
1.	sao paulo	694
2.	curitiba	127
3.	rio de janeiro	96
4.	belo horizonte	68
5.	ribeirao preto	52
6.	guarulhos	50
7.	ibitinga	49
8.	santo andre	45
9.	campinas	41
10.	maringa	40

1 - 10 / 10



B. Top 10 unique combinations of states and cities with the highest number of sellers:

```
41  select seller_state, seller_city, count(distinct seller_id) as no_of_sellers from `targetsql-368319.targetsql.sellers`
42  group by seller_state, seller_city
43  order by no_of_sellers desc limit 10;
44
```

Query results

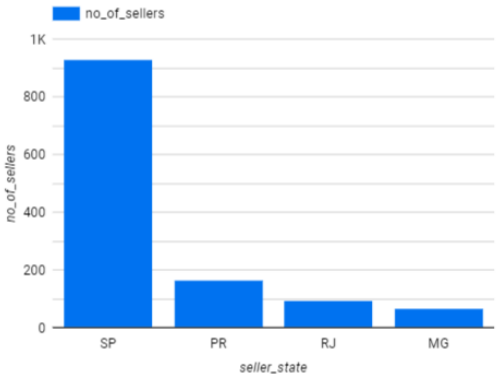
SAVE RESULTS

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

Row	seller_state	seller_city	no_of_sellers
1	SP	sao paulo	694
2	PR	curitiba	124
3	RJ	rio de janeiro	93
4	MG	belo horizonte	66
5	SP	ribeirao preto	52
6	SP	guarulhos	50
7	SP	ibitinga	49
8	SP	santo andre	45
9	SP	campinas	41
10	PR	maringa	40

	seller_state	no_of_sellers
1.	SP	931
2.	PR	164
3.	RJ	93
4.	MG	66

1 - 4 / 4



We can notice the count is similar to the count obtained from the top seller grouped by City. However, few cities here have a little less count of sellers when compared to the count obtained only on the basis of the city.

So, it seems like there are a few cities that are tagged to more than one state. There may be some inaccuracy in the demographic data in the seller's table.

Sample of the cities tagged to multiple states :

```sql
select seller_city, count(distinct seller_state) as multistate
from (select seller_state, seller_city, count(distinct seller_id) as no_of_sellers
from `targetsql-368319.targetsql.sellers`
group by seller_state, seller_city)
group by seller_city
having multistate > 1;
```

Query results

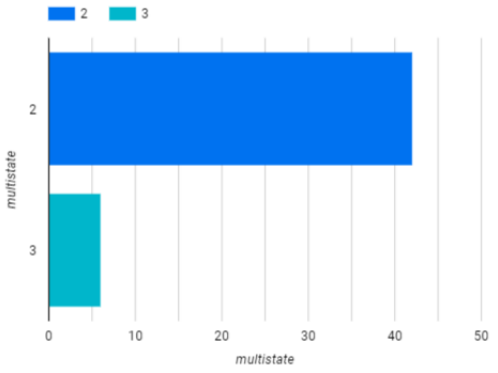
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	seller_city	multistate
1	ipira	2
2	vila velha	2
3	andradas	2
4	jacutinga	2
5	juiz de fora	2
6	belo horizonte	2
7	marechal candido rondon	3
8	guaira	2

There are 23 such cities tagged to multiple states.

	seller_city	multistate
1.	marechal candido rondon	3
2.	rio de janeiro	3
3.	andradas	2
4.	belo horizonte	2
5.	volta redonda	2
6.	juiz de fora	2
7.	curitiba	2
8.	londrina	2
9.	sao jose dos pinhais	2
10.	rio bonito	2
11.	pinhais	2

1 - 23 / 23



C. Top 5 states contributing to the highest number of sellers registration:

```sql
SELECT seller_state, COUNT(DISTINCT(seller_id)) as no_of_sellers FROM `targetsql-368319.targetsql.sellers`
GROUP BY seller_state ORDER BY no_of_sellers DESC LIMIT 5;
```

Query results SAVE

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

Row	seller_state	no_of_sellers
1	SP	1849
2	PR	349
3	MG	244
4	SC	190
5	RJ	171

D. Top 10 cities with the highest orders:

```
select c.customer_city, count(distinct(order_id)) as no_of_orders from `targetsql-368319.targetsql.customers` c, `targetsql-368319.targetsql.orders` o
where c.customer_id=o.customer_id
group by customer_city order by no_of_orders desc limit 10;
```

Press Alt+F1 for accessibility opt

Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_city	no_of_orders
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444
7	porto alegre	1379
8	salvador	1245
9	guarulhos	1189
10	sao bernardo do campo	938

E. Top 10 states with the highest orders:

```
SELECT customer_state,COUNT(DISTINCT(customer_id)) AS no_of_customers FROM `targetsql-368319.targetsql.customers`
GROUP BY customer_state ORDER BY no_of_customers DESC LIMIT 10;
```

Query results

SAVE RESUL

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

Row	customer_state	no_of_custo...
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

F. Customers with the highest number of orders:

```sql
select c.customer_unique_id, count(distinct(order_id)) as no_of_purchases
from `targetsql-368319.targetsql.customers` c, `targetsql-368319.targetsql.orders` o
where c.customer_id=o.customer_id
group by customer_unique_id
order by no_of_purchases desc limit 10;
```

Query results

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

Row	customer_unique_id	no_of_purch...
1	8d50f5eadf50201ccdcedfb9e2...	17
2	3e43e6105506432c953e165fb...	9
3	ca77025e7201e3b30c44b472f...	7
4	6469f99c1f9dfae7733b25662e...	7
5	1b6c7548a2a1f9037c1fd3ddfe...	7
6	47c1a3033b8b77b3ab6e109eb...	6
7	f0e310a6839dce9de1638e0fe...	6
8	12f5d6e1cbf93dafd9dcc19095...	6
9	de34b16117594161a6a89c50...	6
10	dc813062e0fc23409cd255f7f5...	6

G. Total numbers of orders, total numbers of orders delivered, and total numbers of orders cancelled:

```sql
SELECT COUNT(DISTINCT(order_id)) as total_orders from `targetsql-368319.targetsql.orders`;
```

Query results

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

Row	total_orders
1	99441

```sql
select count(distinct(order_id)) as orders_deliver from `targetsql-368319.targetsql.orders` where order_status='delivered';
```

Query results — SAVE RESULTS

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

Row	orders_deliv...
1	96478

```sql
select count(distinct(order_id)) as orders_cancel from `targetsql-368319.targetsql.orders` where order_status='canceled';
```

Query results — SAVE RESULTS

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

Row	orders_canc...
1	625

H. How many sellers are registered with TARGET?

```
32  SELECT COUNT(DISTINCT(seller_id)) as No_of_sellers FROM `targetsql-368319.targetsql.sellers`;
33
```

Query results

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

Row	No_of_sellers
1	3095

I. How customers have rated the services:

```
75  select review_score, count(review_score) as cnt from `targetsql-368319.targetsql.order_reviews`
76  group by review_score
77  ORDER BY review_score DESC;
```

Query results

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

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

J. check how customers' rating has improved/worsened over the entire period.

```
79  select '20'|| year as year, review_score, count(review_score) as cnt from
80  (
81  select *, EXTRACT(day FROM review_creation_date) as year from `targetsql-368319.targetsql.order_reviews`
82  )
83  group by year, review_score
84  order by year, review_score desc;
```

Query results

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

Row	year	review_score	cnt
1	2016	5	156
2	2016	4	51
3	2016	3	22
4	2016	2	8
5	2016	1	88
6	2017	5	24759
7	2017	4	8440
8	2017	3	3589
9	2017	2	1350


```
79  select '20'|| year as year, review_score, count(review_score) as cnt from
80  (
81  select *, EXTRACT(day FROM review_creation_date) as year from `targetsql-368319.targetsql.order_reviews`
82  )
83  group by year, review_score
84  order by year, review_score desc;
```

Query results

SAVE RESUL

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

Row	year	review_score	cnt
7	2017	4	8440
8	2017	3	3589
9	2017	2	1350
10	2017	1	4597
11	2018	5	32413
12	2018	4	10651
13	2018	3	4568
14	2018	2	1793
15	2018	1	6739

Results per page: 50

We can see that the rating has improved over the period of time. And, there is a significant increase in the review count in 2017 as compared to 2016.

K. Total numbers of product categories available:

```
85
86  SELECT COUNT(DISTINCT(product_category)) as no_of_category FROM `targetsql-368319.targetsql.products`;
87
```

Query results

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

Row	no_of_categ...
1	73

L. Top 10 categories with the most number of products:

```
88  SELECT product_category,COUNT(DISTINCT(product_id)) as no_of_products
89  FROM `targetsql-368319.targetsql.products`
90  GROUP BY product_category
91  ORDER BY no_of_products
92  DESC LIMIT 10;
```

Query results

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

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

M. Top 10 categories with the lowest number of products:

```sql
SELECT product_category,COUNT(DISTINCT(product_id)) as no_of_product
FROM `targetsql-368319.targetsql.products`
GROUP BY product_category
ORDER BY no_of_product
asc LIMIT 10;
```

Query results

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

Row	product_category	no_of_produ...
1	cds music dvds	1
2	insurance and services	2
3	PC Gamer	3
4	Fashion Children's Clothing	5
5	House Comfort 2	5
6	IMAGE IMPORT TABLETS	9
7	CITTE AND UPHACK FURNITU...	10
8	La Cuisine	10
9	Kitchen portable and food coach	10
10	Hygiene diapers	12

N. Top 10 highest-selling product categories in terms of no. of orders:

```sql
select product_category, count(o.order_id) as count_of_order
from `targetsql-368319.targetsql.products` p, `targetsql-368319.targetsql.orders` o, `targetsql-368319.targetsql.order_items` i
where o.order_id=i.order_id
and i.product_id=p.product_id
group by product_category
order by count_of_order
desc limit 10;
```

Query results

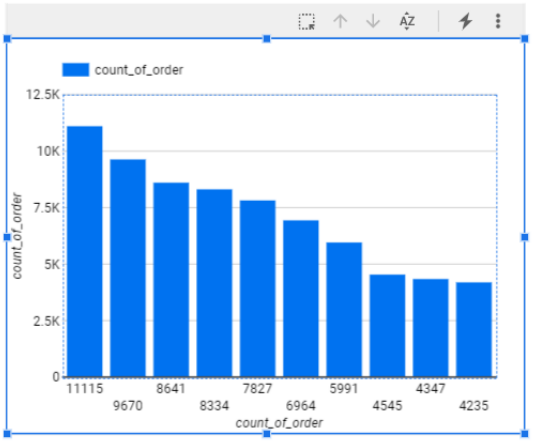
SAVE RESULTS

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

Row	product_category	count_of_or...
1	bed table bath	11115
2	HEALTH BEAUTY	9670
3	sport leisure	8641
4	Furniture Decoration	8334
5	computer accessories	7827
6	housewares	6964
7	Watches present	5991
8	telephony	4545
9	Garden tools	4347
10	automotive	4235

	product_category	count_of_order
1.	bed table bath	11,115
2.	HEALTH BEAUTY	9,670
3.	sport leisure	8,641
4.	Furniture Decoration	8,334
5.	computer accessories	7,827
6.	housewares	6,964
7.	Watches present	5,991
8.	telephony	4,545
9.	Garden tools	4,347
10.	automotive	4,235

1 - 10 / 10



0. Bottom 10 selling product categories in terms no. of orders:

```
select product_category, count(o.order_id) as count_of_order
from `targetsql-368319.targetsql.products` p, `targetsql-368319.targetsql.orders` o, `targetsql-368319.targetsql.order_items` i
where o.order_id=i.order_id
and i.product_id=p.product_id
group by product_category
order by count_of_order asc limit 10;
```

Query results

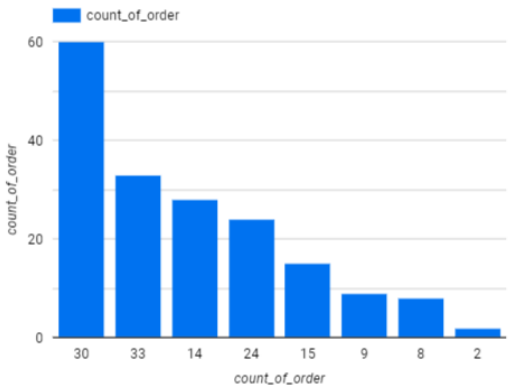
SAVE RESULTS

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

Row	product_category	count_of_or...
1	insurance and services	2
2	Fashion Children's Clothing	8
3	PC Gamer	9
4	cds music dvds	14
5	La Cuisine	14
6	Kitchen portable and food coach	15
7	Arts and Crafts	24
8	House Comfort 2	30
9	Fashion Sport	30
10	flowers	33

	product_category	count_of_order
1.	flowers	33
2.	Fashion Sport	30
3.	House Comfort 2	30
4.	Arts and Crafts	24
5.	Kitchen portable and food coach	15
6.	La Cuisine	14
7.	cds music dvds	14
8.	PC Gamer	9
9.	Fashion Children's Clothing	8
10.	insurance and services	2

1 - 10 / 10



P. Top 10 highest selling products in terms of amount:

```
115  select product_category, round(sum(price+freight_value),2) as total_cost
116  from `targetsql-368319.targetsql.products` p, `targetsql-368319.targetsql.orders` o,
117  `targetsql-368319.targetsql.order_items` i
118  where o.order_id=i.order_id
119  and i.product_id=p.product_id
120  group by product_category
121  order by total_cost desc limit 10 ;
```

Query results

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

Row	product_category	total_cost
1	HEALTH BEAUTY	1441248.07
2	Watches present	1305541.61
3	bed table bath	1241681.72
4	sport leisure	1156656.48
5	computer accessories	1059272.4
6	Furniture Decoration	902511.79
7	housewares	778397.77
8	Cool Stuff	719329.95
9	automotive	685384.32
10	Garden tools	584219.21

Q. Bottom 10 selling products in terms of amount:

```
123  select product_category, round(sum(price+freight_value),2) as total_cost
124  from `targetsql-368319.targetsql.products` p, `targetsql-368319.targetsql.orders` o,
125  `targetsql-368319.targetsql.order_items` i
126  where o.order_id=i.order_id
127  and i.product_id=p.product_id
128  group by product_category
129  order by total_cost asc limit 10 ;
```

Query results

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

Row	product_category	total_cost
1	insurance and services	324.51
2	Fashion Children's Clothing	665.36
3	cds music dvds	954.99
4	House Comfort 2	1170.58
5	flowers	1598.91
6	PC Gamer	1679.52
7	Hygiene diapers	2141.27
8	Arts and Crafts	2184.14
9	La Cuisine	2388.54
10	Fashion Sport	2697.64

From the above analysis done on product category and no. of orders and amount, we can see if there is a relation between no. of products in a category with their sales. We can observe the highest-selling product category also has the highest no. of products.

R. Reviews containing negative words like bad, dissatisfactory, worse, etc.:

```
131  SELECT count(review_id) as count
132  from `targetsql-368319.targetsql.order_reviews`
133  where review_comment_title LIKE '%bad%' or review_comment_title LIKE '%Bad%'
134  or review_comment_title LIKE'%dissatisfac' or review_comment_title LIKE "%worse%";
135
136
```

Query results

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

Row	count
1	104

S. Review comments containing positive words like good, great, recommend, satisfactory, etc.

```
137  SELECT count(review_comment_title) as count
138  from `targetsql-368319.targetsql.order_reviews`
139  where review_comment_title LIKE "%ecommend%" or review_comment_title LIKE "%great%"
140  or review_comment_title LIKE "%good%" or review_comment_title LIKE "%satisfac%";
141
```

Query results

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

Row	count
1	3079

Insights from the above analysis

1. This data is for the period of 4th august 206 to 17th august 2018.
    a. Sales are very less in 2016.
    b. Sales have picked up only after 2017.

2. Growth in sales has increased over the years. 2018 sales in terms of money are 139% higher than in 2017.

3. Highest sales are in the months of august, may and july. In 2017, the highest sales were in November and December.

4. Higher Sales are observed in the second half of the month. In many cases after 20 the sales are more.

5. Sao Paulo is the state with the highest number of sales.

6. 22790, 24220, 22793, and 24230 are some of the area zip codes with the highest sales.

7. 22790, 24220, 22793, and 24230 are the area with the highest number of customers buying from Target.

8. RR, PB, RO, AC, and PI states with high avg freight value. SP, PR, MG, RJ, DF states with the lowest avg freight value. SP, RJ, MG, RS, and PR states with the highest total freight value.

9. RR, AP, AM, AL and PA states with the highest avg estimated delivery time.

10. AC, RO, AP, AM and AR are the states where delivery is fastest.

11. AL, MA, SE, ES, and BA are the states where delivery has been slowest.

12. Estimated delivery has improved over the entire duration. Although, it did not have an impact on the review score.

13. The review score has shown better results with delivery earlier than estimated.

14. A total business of around 15 million has been done for the given time period.

15. The highest order was 13.6 k.

16. Total 3095 sellers have registered themselves with target.

17. States with more sellers have more sales.

18. Most sellers are registered from Sao Paulo.

19. Trend in buying the products is more in Afternoon or at night.

20. More than 75% of people prefer to pay via credit cards, 20% via UPI.

21. From 2017 to 2018 a massive growth in movement has been observed. Value of orders in 2017 was 3.6 million which rose up to 8.6 in 2018.

22. Total of 99k orders were made and 625 got cancelled.

23. More than 57% of the reviews have been given to the top rating.

24. There are 73 categories of products.

25. Product categories with a higher number of products tend to have more sales.

26. Health Beauty product category giving the highest sales in terms of money. And, Bed table bath is highest in terms of the number of orders.

Recommendations

1. Observe that faster delivery time has shown an increase in number of orders. So, the target can work on improving it further to achieve even better results.
2. Observe that there is a better review from the customers when the delivery is earlier than estimated. Target can work on it to speed up the delivery speed.
3. The Relative number of sales in August, May and July are larger than others. Also, in the year 2017 there are quite a few sales in november and december. We can launch more products and give more offers in these months.
4. We have noticed higher sales in the later half of the month. We use AI for personal recommendations on the basis of their interest during the second half of the month.
5. We have also seen some top customers with the highest purchase. We can put them in a special category for gifts and offers so that they can retain top customers.
6. Sao Paulo is the state with the highest number of sales. Target can invest something on studying the reason behind that and employ that technique in the other states. The seller quality, product quality, delivery time, and other factors which are making it happen in Sao Paulo can be employed in the other states to increase sales.
7. Target can reduce the freight value to increase the sales.
8. Zip codes with highest sales can be targeted further by advertising the brands.
9. Most payments are done by using credit cards so Target can tie up with more credit card partners to give more offers to increase the sales.
10. More sellers have resulted in more sales; Target can work to increase tie up with more sellers to expand their business.
11. Product categories with more number of products have shown better results. Target can also see which sellers are giving more varieties of product category and make partnerships with those sellers.
12. Health beauty has been the category with the highest number of sales in terms of amount. It can be further given importance to yield more profit.
13. Bad reviews are very less compared to good reviews, so target can make sure to maintain it. Even bad reviews can be studied and some action can be taken to improve the customer's experience.