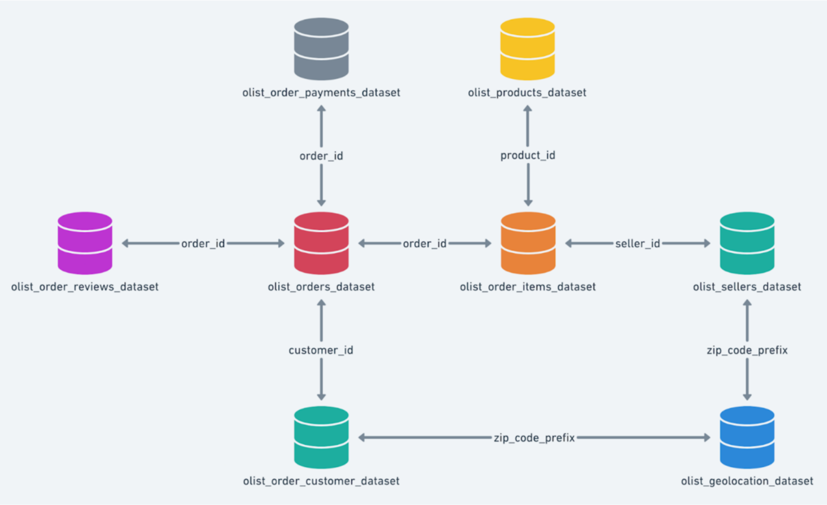
TARGET CASE STUDY

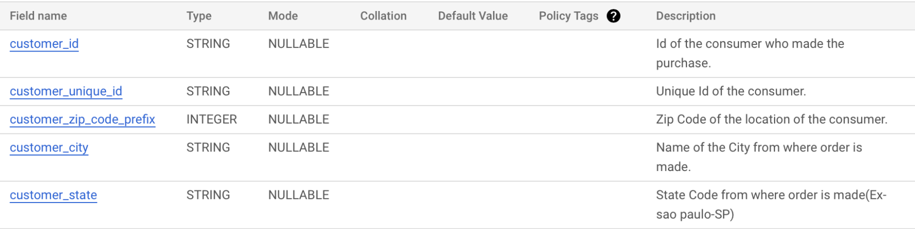
SCHEMA



1) Exploratory Analysis

1) Exploratory Analysis Data-Types and relations

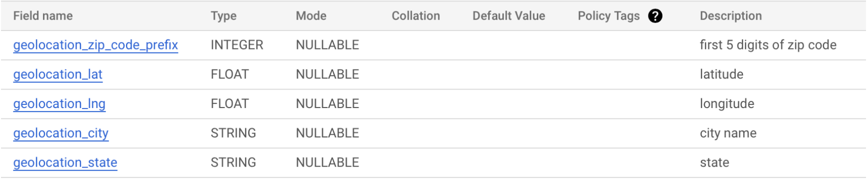
- customers Table



Field name	Type	Mode	Collation	Default Value	Policy Tags ?	Description
customer_id	STRING	NULLABLE				Id of the consumer who made the purchase.
customer_unique_id	STRING	NULLABLE				Unique Id of the consumer.
customer_zip_code_prefix	INTEGER	NULLABLE				Zip Code of the location of the consumer.
customer_city	STRING	NULLABLE				Name of the City from where order is made.
customer_state	STRING	NULLABLE				State Code from where order is made(Ex- sao paulo-SP)

Main Table	Sub Table	Dependency/Relation Columns
customers	orders	*customer_id*

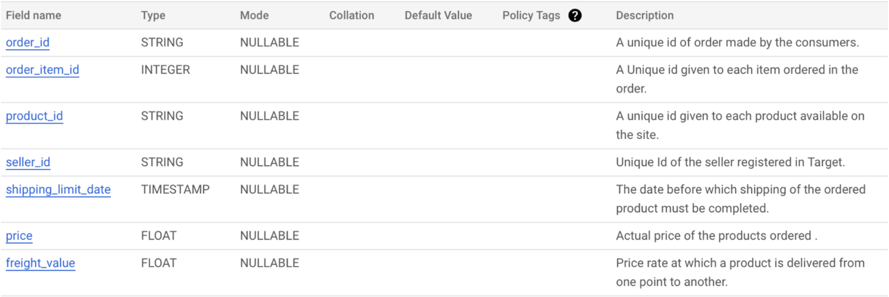
- geolocation table



Field name	Type	Mode	Collation	Default Value	Policy Tags ?	Description
geolocation_zip_code_prefix	INTEGER	NULLABLE				first 5 digits of zip code
geolocation_lat	FLOAT	NULLABLE				latitude
geolocation_lng	FLOAT	NULLABLE				longitude
geolocation_city	STRING	NULLABLE				city name
geolocation_state	STRING	NULLABLE				state

Main Table	Sub Tables	Dependency/Relation Columns
geolocation	sellers	*zipcode_prefix*
geolocation	customers	*zipcode_prefix*

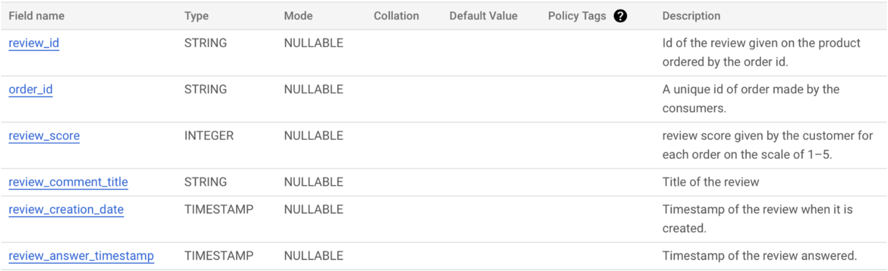
- order_items table



Field name	Type	Mode	Collation	Default Value	Policy Tags ?	Description
order_id	STRING	NULLABLE				A unique id of order made by the consumers.
order_item_id	INTEGER	NULLABLE				A Unique id given to each item ordered in the order.
product_id	STRING	NULLABLE				A unique id given to each product available on the site.
seller_id	STRING	NULLABLE				Unique Id of the seller registered in Target.
shipping_limit_date	TIMESTAMP	NULLABLE				The date before which shipping of the ordered product must be completed.
price	FLOAT	NULLABLE				Actual price of the products ordered .
freight_value	FLOAT	NULLABLE				Price rate at which a product is delivered from one point to another.

Main Table	Sub Table	Dependency/Relation Columns
order_items	orders	*Order_id*

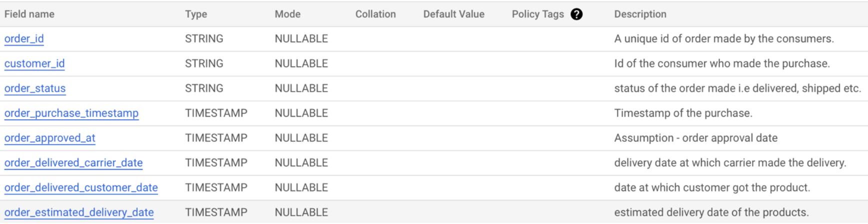
- order_reviews table



Field name	Type	Mode	Collation	Default Value	Policy Tags ?	Description
review_id	STRING	NULLABLE				Id of the review given on the product ordered by the order id.
order_id	STRING	NULLABLE				A unique id of order made by the consumers.
review_score	INTEGER	NULLABLE				review score given by the customer for each order on the scale of 1–5.
review_comment_title	STRING	NULLABLE				Title of the review
review_creation_date	TIMESTAMP	NULLABLE				Timestamp of the review when it is created.
review_answer_timestamp	TIMESTAMP	NULLABLE				Timestamp of the review answered.

Main Table	Sub Table	Dependency/Relation Columns
order_reviews	orders	*order_id*

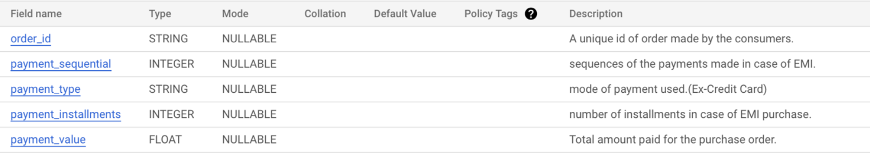
- orders table



Field name	Type	Mode	Collation	Default Value	Policy Tags ?	Description
order_id	STRING	NULLABLE				A unique id of order made by the consumers.
customer_id	STRING	NULLABLE				Id of the consumer who made the purchase.
order_status	STRING	NULLABLE				status of the order made i.e delivered, shipped etc.
order_purchase_timestamp	TIMESTAMP	NULLABLE				Timestamp of the purchase.
order_approved_at	TIMESTAMP	NULLABLE				Assumption - order approval date
order_delivered_carrier_date	TIMESTAMP	NULLABLE				delivery date at which carrier made the delivery.
order_delivered_customer_date	TIMESTAMP	NULLABLE				date at which customer got the product.
order_estimated_delivery_date	TIMESTAMP	NULLABLE				estimated delivery date of the products.

Main Table	Sub Table	Dependency/Relation Columns
orders	order_reviews	*order_id*
orders	order_items	*order_id*
orders	payments	*order_id*

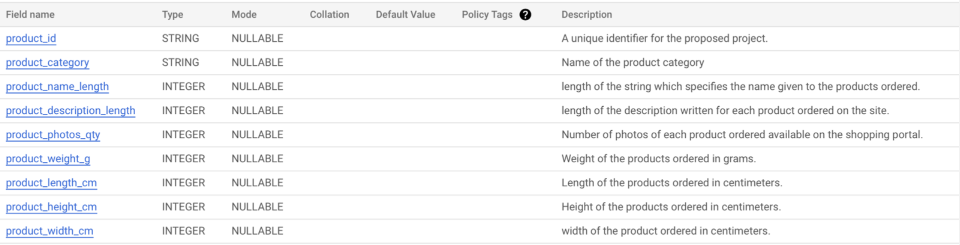
- payments table



Field name	Type	Mode	Collation	Default Value	Policy Tags ?	Description
order_id	STRING	NULLABLE				A unique id of order made by the consumers.
payment_sequential	INTEGER	NULLABLE				sequences of the payments made in case of EMI.
payment_type	STRING	NULLABLE				mode of payment used.(Ex-Credit Card)
payment_installments	INTEGER	NULLABLE				number of installments in case of EMI purchase.
payment_value	FLOAT	NULLABLE				Total amount paid for the purchase order.

Main Table	Sub Table	Dependency/Relation Columns
payments	orders	*order_id*

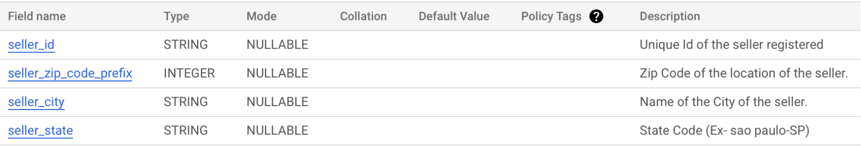
- products table



Field name	Type	Mode	Collation	Default Value	Policy Tags ?	Description
product_id	STRING	NULLABLE				A unique identifier for the proposed project.
product_category	STRING	NULLABLE				Name of the product category
product_name_length	INTEGER	NULLABLE				length of the string which specifies the name given to the products ordered.
product_description_length	INTEGER	NULLABLE				length of the description written for each product ordered on the site.
product_photos_qty	INTEGER	NULLABLE				Number of photos of each product ordered available on the shopping portal.
product_weight_g	INTEGER	NULLABLE				Weight of the products ordered in grams.
product_length_cm	INTEGER	NULLABLE				Length of the products ordered in centimeters.
product_height_cm	INTEGER	NULLABLE				Height of the products ordered in centimeters.
product_width_cm	INTEGER	NULLABLE				width of the product ordered in centimeters.

Main Table	Sub Table	Dependency/Relation Columns
products	order_items	*product_id*

- sellers table



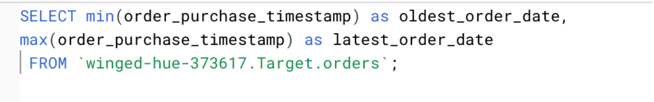
Field name	Type	Mode	Collation	Default Value	Policy Tags ?	Description
seller_id	STRING	NULLABLE				Unique Id of the seller registered
seller_zip_code_prefix	INTEGER	NULLABLE				Zip Code of the location of the seller.
seller_city	STRING	NULLABLE				Name of the City of the seller.
seller_state	STRING	NULLABLE				State Code (Ex- sao paulo-SP)

Main Table	Sub Table	Dependency/Relation Columns
sellers	order_items	*seller_id*

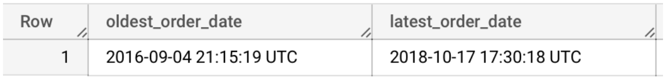
2) Time Period for the data is given:

4th Sept, 2016 to 17th Oct, 2018

ORDER TABLE **-** 4th Sept, 2016 to 17th Oct, 2018



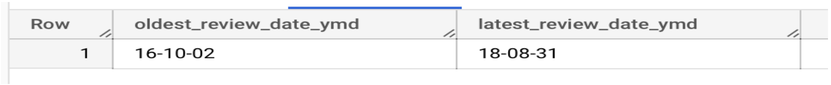
```
SELECT min(order_purchase_timestamp) as oldest_order_date,
max(order_purchase_timestamp) as latest_order_date
 FROM `winged-hue-373617.Target.orders`;
```



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

ORDER_REVIEWS **-** 2nd Oct, 2016 to 31st Aug, 2018

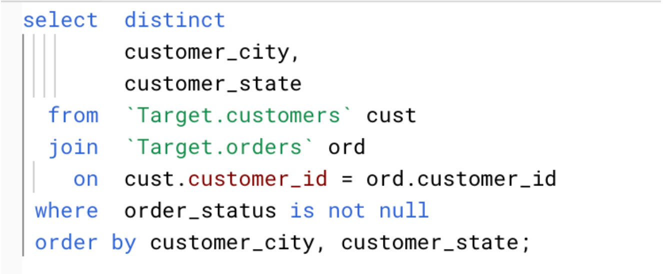
```
select min(concat(year,'-',month,'-',day)) as oldest_review_date_ymd,
max(concat(year,'-',month,'-',day)) as latest_review_date_ymd
from
(select  SUBSTR(CAST((review_creation_date) AS STRING),3,2) as day,
SUBSTR(CAST((review_creation_date) AS STRING),6,2) as month,
SUBSTR(CAST((review_creation_date) AS STRING),9,2) as year
from  `Target.order_reviews`) temp1;
```



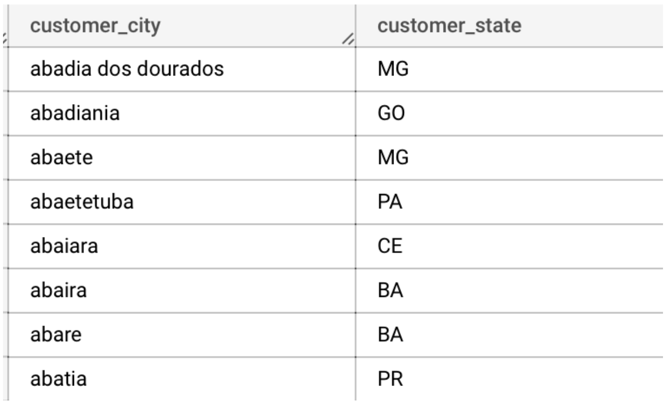
Row	oldest_review_date_ymd	latest_review_date_ymd
1	16-10-02	18-08-31

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

There are customers from 4310 different cities.



```
select  distinct
        customer_city,
        customer_state
  from  `Target.customers` cust
  join  `Target.orders` ord
    on  cust.customer_id = ord.customer_id
 where  order_status is not null
 order by customer_city, customer_state;
```



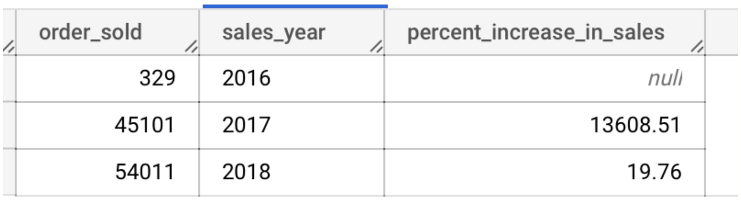
customer_city	customer_state
abadia dos dourados	MG
abadiania	GO
abaete	MG
abaetetuba	PA
abaiara	CE
abaira	BA
abare	BA
abatia	PR

2) In-depth Exploration

1) Growing trend on e-commerce in Brazil

1. Overall Sale is increasing year on year with highest sale on Black Friday 2017.

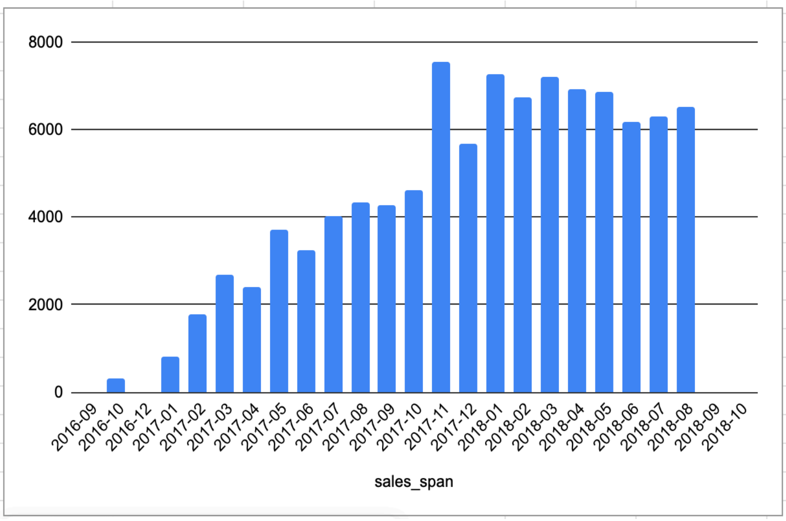
```
select *, Round(((order_sold - lag(order_sold,1) over(order by
tbl1.sales_year))/lag(order_sold,1) over(order by tbl1.sales_year)) * 100 ,2)
          as percent_increase_in_sales
from
(
select  count(order_id) as order_sold,
              SUBSTR(CAST((order_purchase_timestamp) AS STRING),01,4) as sales_year
        from `Target.orders`
        group by sales_year
        order by sales_year) tbl1
order by sales_year;
```



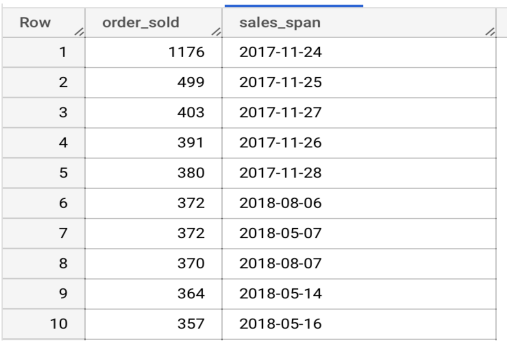
order_sold	sales_year	percent_increase_in_sales
329	2016	*null*
45101	2017	13608.51
54011	2018	19.76

2. Month by month sales - Highest sales in month of Nov, 2017

```
select  count(order_id) as order_sold,
Concat(SUBSTR(CAST((order_purchase_timestamp) AS STRING),01,4),'-',
SUBSTR(CAST((order_purchase_timestamp) AS STRING),06,2)) as sales_span
from `Target.orders`
group by sales_span
order by sales_span;
```



Top 10 Sales - **Highest sale on Black Friday 2017**



Row	order_sold	sales_span
1	1176	2017-11-24
2	499	2017-11-25
3	403	2017-11-27
4	391	2017-11-26
5	380	2017-11-28
6	372	2018-08-06
7	372	2018-05-07
8	370	2018-08-07
9	364	2018-05-14
10	357	2018-05-16

There is a steep fall in sales from the month of sept, 2018. More data/information is required to conclude the factor behind this behavior like any kind of emergencies,
Political ruling etc. applied in end of august or beginning of sept, 2018.

2) Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

I need further information to assign a specific time period for the 4 phases of a day so I can evaluate. To overcome this situation I have further drill down the 4 spans in individual category to get proper insights.

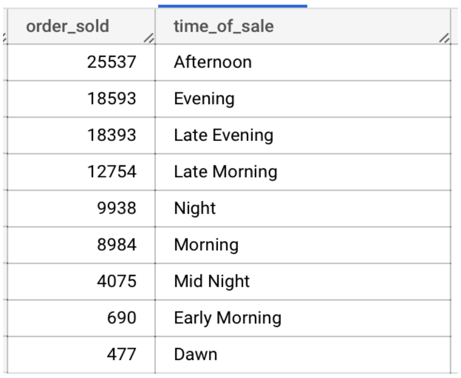
Commented [SS1]: There requires a further information to assign specific hours to the four phases mentioned in question. To overcome this I have divided these phases into further category to get proper insights.

Night	22:00:00 - 23:59:59
Mid Night	00:00:00 - 03:00:00
Dawn	03:00:01 – 05:00:00
Early Morning	05:00:01 – 07:00:00
Morning	07:00:01 – 10:00:00
Late Morning	10:00:01 – 11:59:59
Afternoon	12:00:00 – 16:00:00
Evening	16:00:01 – 19:00:00
Late Evening	19:00:01 – 22:00:00

Highest orders are placed between Afternoon followed by Evening, Late Evening and Late Morning. Least orders during the Dawn.

```
select  count(order_id) as order_sold,
case
When extract(time from order_purchase_timestamp) between "22:00:01" and "23:59:59" then
'Night'
When extract(time from order_purchase_timestamp) between "00:00:00" and "03:00:00" then
'Mid Night'
When extract(time from order_purchase_timestamp) between "03:00:01" and "05:00:00" then
'Dawn'
When extract(time from order_purchase_timestamp) between "05:00:01" and "07:00:00" then
'Early Morning'
When extract(time from order_purchase_timestamp) between "07:00:01" and "10:00:00" then
'Morning'
```

```
When extract(time from order_purchase_timestamp) between "10:00:01" and "11:59:59" then
'Late Morning'
When extract(time from order_purchase_timestamp) between "12:00:00" and "16:00:00" then
'Afternoon'
When extract(time from order_purchase_timestamp) between "16:00:01" and "19:00:00" then
'Evening'
When extract(time from order_purchase_timestamp) between "19:00:01" and "22:00:00" then
'Late Evening'
end  as time_of_sale
from `Target.orders`
group by time_of_sale
order by order_sold desc;
```

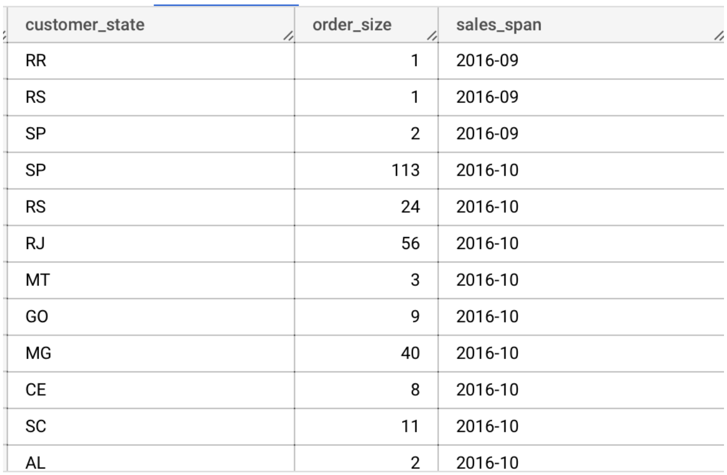


order_sold	time_of_sale
25537	Afternoon
18593	Evening
18393	Late Evening
12754	Late Morning
9938	Night
8984	Morning
4075	Mid Night
690	Early Morning
477	Dawn

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

1) Month on Month orders by states

```
select  customer_state,
count(ord.order_id) as order_size,
Concat(SUBSTR(CAST((order_purchase_timestamp) AS STRING),01,4),'-',
SUBSTR(CAST((order_purchase_timestamp) AS STRING),06,2)) as sales_span
from  `Target.customers` cust
join  `Target.orders` ord
on  cust.customer_id = ord.customer_id
where  order_status is not null
group by sales_span, customer_state
order by sales_span;
```

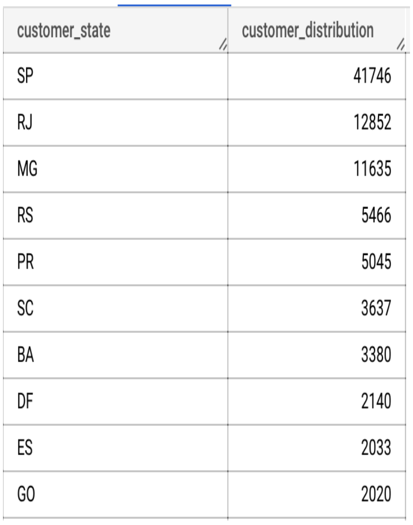


customer_state	order_size	sales_span
RR	1	2016-09
RS	1	2016-09
SP	2	2016-09
SP	113	2016-10
RS	24	2016-10
RJ	56	2016-10
MT	3	2016-10
GO	9	2016-10
MG	40	2016-10
CE	8	2016-10
SC	11	2016-10
AL	2	2016-10

2) Customer Distribution vs State

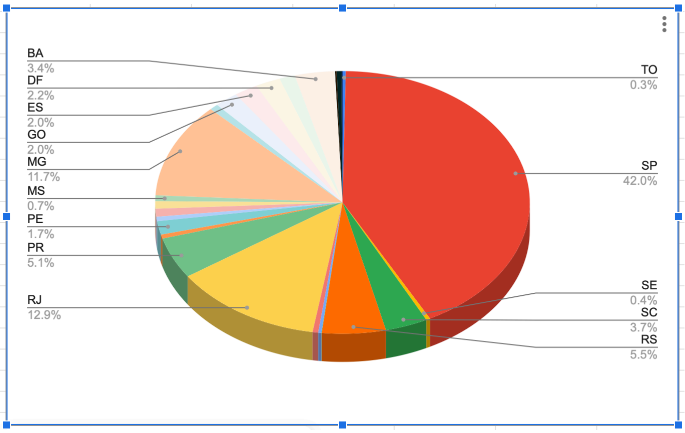
```
select  customer_state, count(customer_unique_id) as customer_distribution
from  `Target.customers`
where
group by customer_state
order by customer_distribution desc;
```

Top 10 contributing states



customer_state	customer_distribution
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380
DF	2140
ES	2033
GO	2020

Least Contributing State - TO
Max Contributing State - SP



4) Impact on Economy

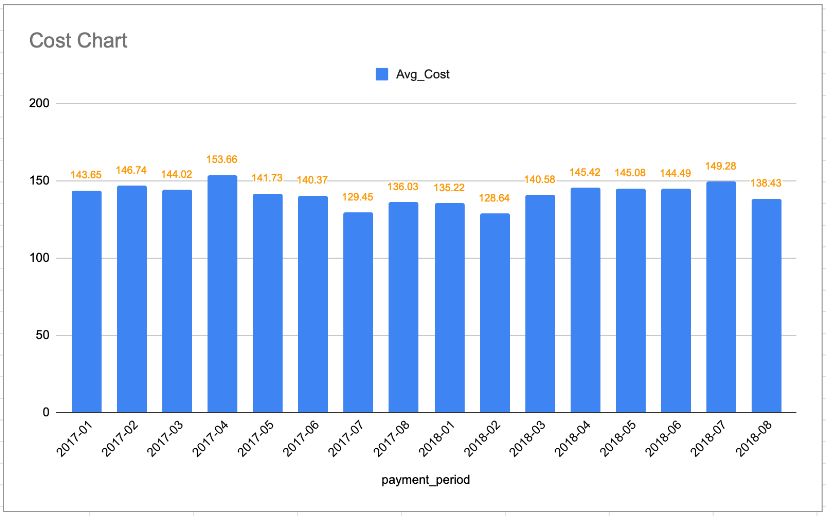
1) % increase in cost of orders from 2017 to 2018

I. Average order cost including (Product original cost + Freight charges) remains approx. same from 2017 to 2018. However look at point ii)

Commented [SS2]: **Payment_value** - This is more of a sales value rather than cost and could be influenced by the increase of clients during the 2018 so it can't reflect the cost analysis of product. Cost analysis can be done on the comparison of cost of products from last year.

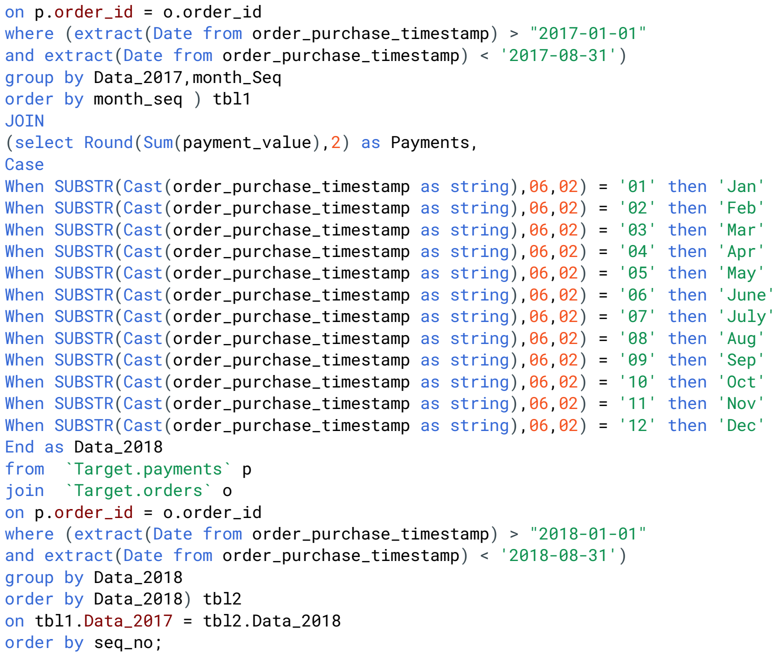
```
select distinct
Round(Avg(price + freight_value),2) as Avg_Cost,
SUBSTR(Cast(order_purchase_timestamp as string),01,07) as payment_period
from  `Target.order_items` oi
```

```
join  `Target.orders` o
on oi.order_id = o.order_id
where (extract(Date from order_purchase_timestamp) > "2018-01-01"
and  extract(Date from order_purchase_timestamp) < '2018-08-31')
or  (extract(Date from order_purchase_timestamp) > "2017-01-01"
and extract(Date from order_purchase_timestamp) < '2017-08-31')
group by payment_period
order by payment_period;
```

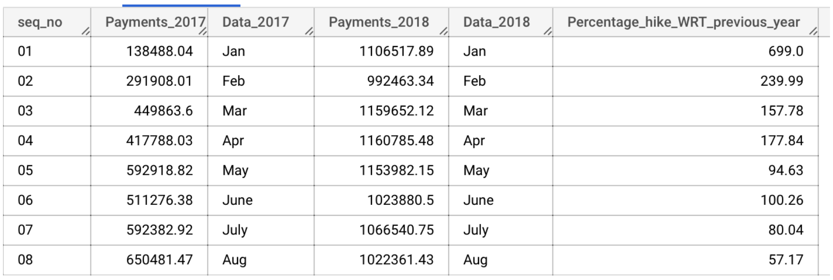


II. Percentage change in 2018 payments compare to previous year.

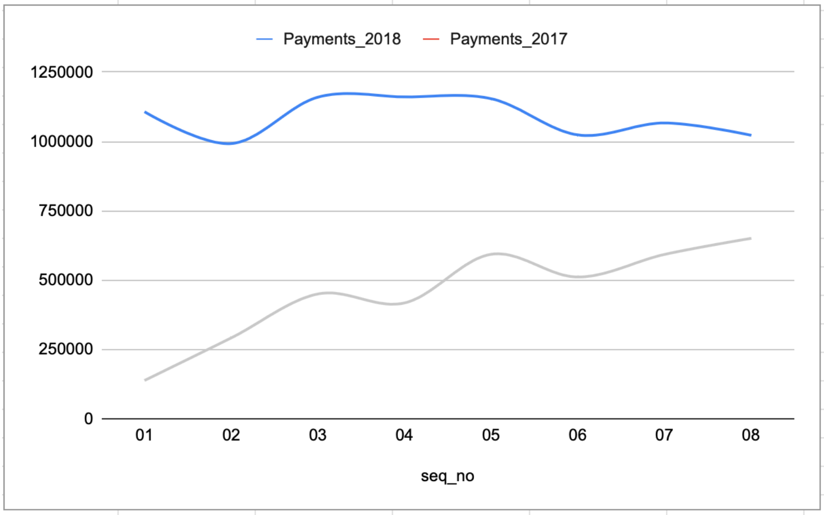
```
select  tbl1.month_Seq as seq_no,
tbl1.Payments as Payments_2017,
tbl1.Data_2017,
tbl2.Payments as Payments_2018,
tbl2.Data_2018,
Round((((tbl2.Payments - tbl1.Payments)/tbl1.Payments)*100),2) as
Percentage_hike_WRT_previous_year
from
(select Round(Sum(payment_value),2) as Payments,
SUBSTR(Cast(order_purchase_timestamp as string),06,02) as month_Seq,
Case
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '01' then 'Jan'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '02' then 'Feb'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '03' then 'Mar'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '04' then 'Apr'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '05' then 'May'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '06' then 'June'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '07' then 'July'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '08' then 'Aug'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '09' then 'Sep'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '10' then 'Oct'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '11' then 'Nov'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '12' then 'Dec'
End as Data_2017
from  `Target.payments` p
join  `Target.orders` o
```

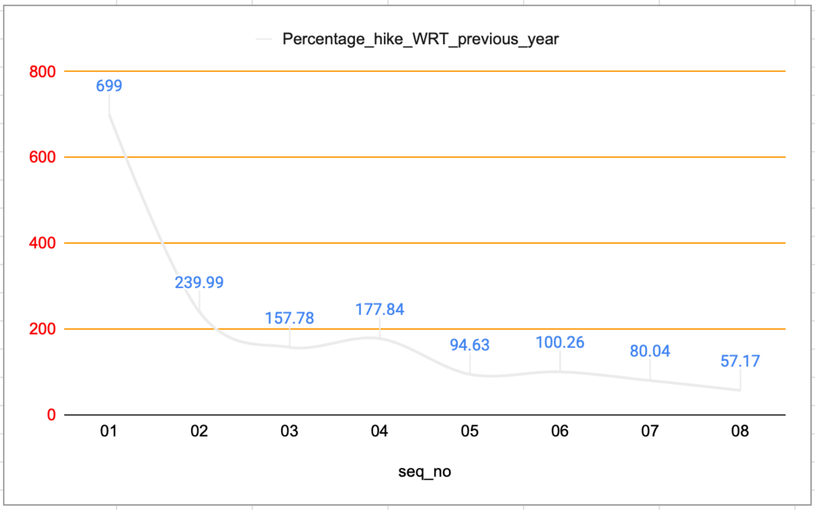


```
on p.order_id = o.order_id
where (extract(Date from order_purchase_timestamp) > "2017-01-01"
and extract(Date from order_purchase_timestamp) < '2017-08-31')
group by Data_2017,month_Seq
order by month_seq ) tbl1
JOIN
(select Round(Sum(payment_value),2) as Payments,
Case
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '01' then 'Jan'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '02' then 'Feb'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '03' then 'Mar'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '04' then 'Apr'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '05' then 'May'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '06' then 'June'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '07' then 'July'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '08' then 'Aug'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '09' then 'Sep'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '10' then 'Oct'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '11' then 'Nov'
When SUBSTR(Cast(order_purchase_timestamp as string),06,02) = '12' then 'Dec'
End as Data_2018
from  `Target.payments` p
join  `Target.orders` o
on p.order_id = o.order_id
where (extract(Date from order_purchase_timestamp) > "2018-01-01"
and extract(Date from order_purchase_timestamp) < '2018-08-31')
group by Data_2018
order by Data_2018) tbl2
on tbl1.Data_2017 = tbl2.Data_2018
order by seq_no;
```



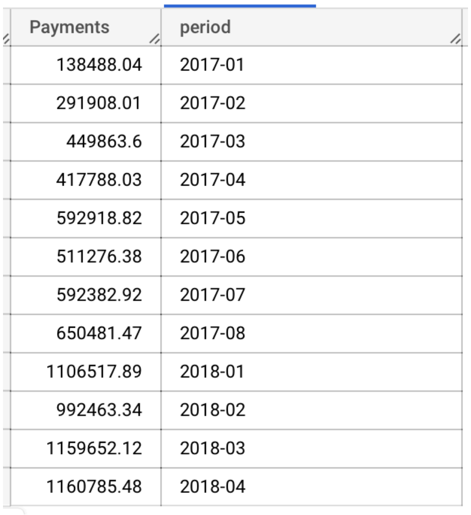
seq_no	Payments_2017	Data_2017	Payments_2018	Data_2018	Percentage_hike_WRT_previous_year
01	138488.04	Jan	1106517.89	Jan	699.0
02	291908.01	Feb	992463.34	Feb	239.99
03	449863.6	Mar	1159652.12	Mar	157.78
04	417788.03	Apr	1160785.48	Apr	177.84
05	592918.82	May	1153982.15	May	94.63
06	511276.38	June	1023880.5	June	100.26
07	592382.92	July	1066540.75	July	80.04
08	650481.47	Aug	1022361.43	Aug	57.17



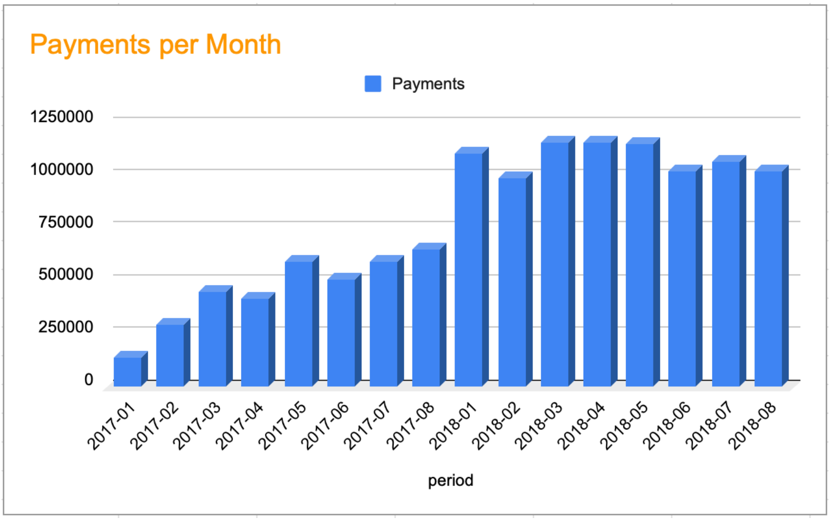


III. Total payments per month

```
select Round(Sum(payment_value),2) as Payments,
SUBSTR(Cast(order_purchase_timestamp as string),01,07) as period
from  `Target.payments` p
join  `Target.orders` o
on p.order_id = o.order_id
where (extract(Date from order_purchase_timestamp) > "2018-01-01"
and  extract(Date from order_purchase_timestamp) < '2018-08-31')
or  (extract(Date from order_purchase_timestamp) > "2017-01-01"
and extract(Date from order_purchase_timestamp) < '2017-08-31')
group by period
order by period;
```



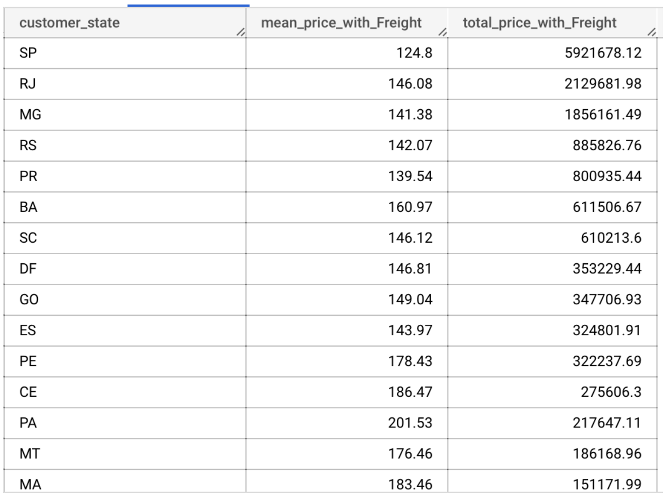
Payments	period
138488.04	2017-01
291908.01	2017-02
449863.6	2017-03
417788.03	2017-04
592918.82	2017-05
511276.38	2017-06
592382.92	2017-07
650481.47	2017-08
1106517.89	2018-01
992463.34	2018-02
1159652.12	2018-03
1160785.48	2018-04



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

```
select tbl1.customer_state,
Round(avg(tbl1.Price_with_Freight),2) as mean_price_with_Freight,
Round(Sum(tbl1.Price_with_Freight),2)  as total_price_with_Freight
```

```
from
(select   oi.order_id,
(oi.price +
oi.freight_value) as Price_with_Freight,
o.customer_id,
c.customer_state
from `Target.order_items` oi
join  `Target.orders` o
on   oi.order_id = o.order_id
join   `Target.customers` c
on  o.customer_id = c.customer_id) tbl1
group by tbl1.customer_state
order by total_price_with_Freight desc;
```



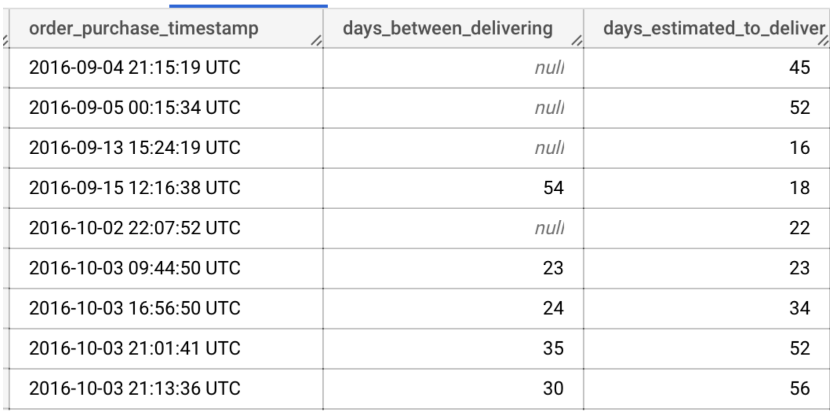
customer_state	mean_price_with_Freight	total_price_with_Freight
SP	124.8	5921678.12
RJ	146.08	2129681.98
MG	141.38	1856161.49
RS	142.07	885826.76
PR	139.54	800935.44
BA	160.97	611506.67
SC	146.12	610213.6
DF	146.81	353229.44
GO	149.04	347706.93
ES	143.97	324801.91
PE	178.43	322237.69
CE	186.47	275606.3
PA	201.53	217647.11
MT	176.46	186168.96
MA	183.46	151171.99

5) Analysis on sales, freight and delivery time

1) Calculate days between purchasing, delivering and estimated delivery

```
select   order_purchase_timestamp,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day) as
days_between_delivering,
TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, Day) as
days_estimated_to_deliver
from `Target.orders`
order by order_purchase_timestamp;
```

Null values for orders which are yet to deliver, cancelled or for which delivery date was not recorded.



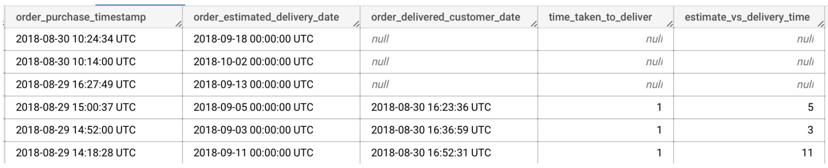
order_purchase_timestamp	days_between_delivering	days_estimated_to_deliver
2016-09-04 21:15:19 UTC	null	45
2016-09-05 00:15:34 UTC	null	52
2016-09-13 15:24:19 UTC	null	16
2016-09-15 12:16:38 UTC	54	18
2016-10-02 22:07:52 UTC	null	22
2016-10-03 09:44:50 UTC	23	23
2016-10-03 16:56:50 UTC	24	34
2016-10-03 21:01:41 UTC	35	52
2016-10-03 21:13:36 UTC	30	56

2) Find time_to_delivery & diff_estimated_delivery

Note - Time_to_delivery is mentioned as time_taken_to_deliver and diff_estimated_delivery as estimate_vs_delivery_time

Latest orders - Null value for orders which are not delivered.

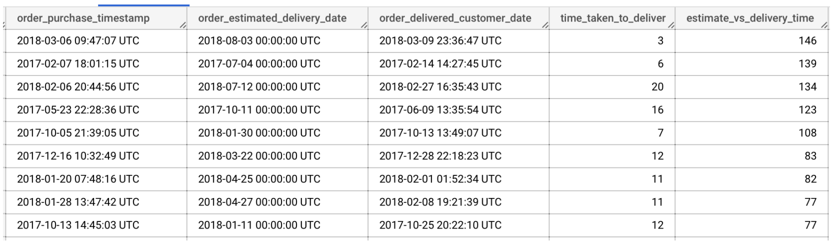
```
select   order_purchase_timestamp,
order_estimated_delivery_date,
order_delivered_customer_date,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day) as
time_taken_to_deliver,
TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, Day) as
estimate_vs_delivery_time
from `Target.orders`
order by order_purchase_timestamp desc;
```



order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	time_taken_to_deliver	estimate_vs_delivery_time
2018-08-30 10:24:34 UTC	2018-09-18 00:00:00 UTC	null	null	null
2018-08-30 10:14:00 UTC	2018-10-02 00:00:00 UTC	null	null	null
2018-08-29 16:27:49 UTC	2018-09-13 00:00:00 UTC	null	null	null
2018-08-29 15:00:37 UTC	2018-09-05 00:00:00 UTC	2018-08-30 16:23:36 UTC	1	5
2018-08-29 14:52:00 UTC	2018-09-03 00:00:00 UTC	2018-08-30 16:36:59 UTC	1	3
2018-08-29 14:18:28 UTC	2018-09-11 00:00:00 UTC	2018-08-30 16:52:31 UTC	1	11

List of orders which had maximum gap between expected delivery and actual delivery date.

```
select   order_purchase_timestamp,
order_estimated_delivery_date,
order_delivered_customer_date,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day) as
time_taken_to_deliver,
TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, Day) as
estimate_vs_delivery_time
from `Target.orders`
order by estimate_vs_delivery_time desc;
```

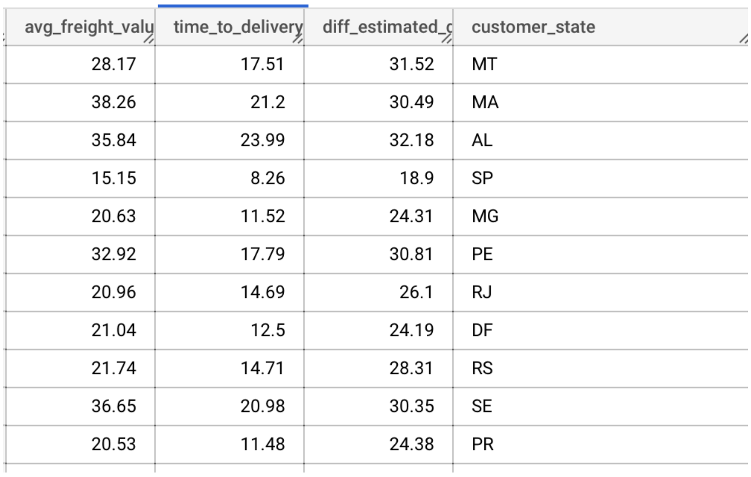


order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	time_taken_to_deliver	estimate_vs_delivery_time
2018-03-06 09:47:07 UTC	2018-08-03 00:00:00 UTC	2018-03-09 23:36:47 UTC	3	146
2017-02-07 18:01:15 UTC	2017-07-04 00:00:00 UTC	2017-02-14 14:27:45 UTC	6	139
2018-02-06 20:44:56 UTC	2018-07-12 00:00:00 UTC	2018-02-27 16:35:43 UTC	20	134
2017-05-23 22:28:36 UTC	2017-10-11 00:00:00 UTC	2017-06-09 13:35:54 UTC	16	123
2017-10-05 21:39:05 UTC	2018-01-30 00:00:00 UTC	2017-10-13 13:49:07 UTC	7	108
2017-12-16 10:32:49 UTC	2018-03-22 00:00:00 UTC	2017-12-28 22:18:23 UTC	12	83
2018-01-20 07:48:16 UTC	2018-04-25 00:00:00 UTC	2018-02-01 01:52:34 UTC	11	82
2018-01-28 13:47:42 UTC	2018-04-27 00:00:00 UTC	2018-02-08 19:21:39 UTC	11	77
2017-10-13 14:45:03 UTC	2018-01-11 00:00:00 UTC	2017-10-25 20:22:10 UTC	12	77

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

Average freight_value, time_to_delivery, diff_estimated_delivery by state

```
select   Round(avg(freight_value),2) as avg_freight_value,
Round(avg(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day)),2)
as time_to_delivery,
Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, Day)),2)
as diff_estimated_delivery,
c.customer_state
from   `Target.order_items` oi
join   `Target.orders` o
on     oi.order_id = o.order_id
join   `Target.customers` c
on     o.customer_id = c.customer_id
group by c.customer_state;
```

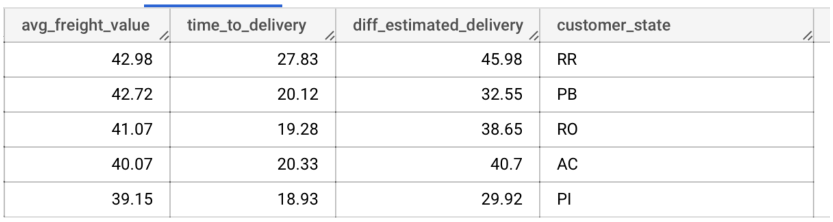


avg_freight_valu	time_to_delivery	diff_estimated_c	customer_state
28.17	17.51	31.52	MT
38.26	21.2	30.49	MA
35.84	23.99	32.18	AL
15.15	8.26	18.9	SP
20.63	11.52	24.31	MG
32.92	17.79	30.81	PE
20.96	14.69	26.1	RJ
21.04	12.5	24.19	DF
21.74	14.71	28.31	RS
36.65	20.98	30.35	SE
20.53	11.48	24.38	PR

4) Sorted data below
5) Top 5 states with highest/lowest average freight value

States with highest freight value

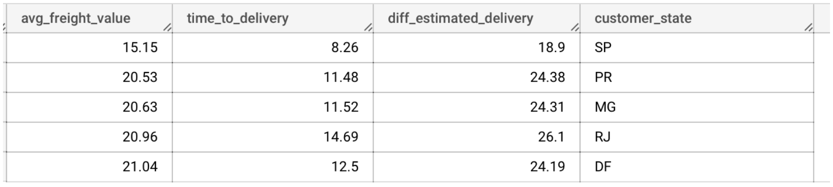
```
select   Round(avg(freight_value),2) as avg_freight_value,
Round(avg(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day)),2)
as time_to_delivery,
Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, Day)),2)
as diff_estimated_delivery,
c.customer_state
from   `Target.order_items` oi
join   `Target.orders` o
on     oi.order_id = o.order_id
join   `Target.customers` c
on     o.customer_id = c.customer_id
group by c.customer_state
order by avg_freight_value desc
limit 5;
```



avg_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
42.98	27.83	45.98	RR
42.72	20.12	32.55	PB
41.07	19.28	38.65	RO
40.07	20.33	40.7	AC
39.15	18.93	29.92	PI

States with Lowest freight value

```
select   Round(avg(freight_value),2) as avg_freight_value,
Round(avg(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day)),2)
as time_to_delivery,
Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp, Day)),2)
as diff_estimated_delivery,
c.customer_state
from   `Target.order_items` oi
join   `Target.orders` o
on     oi.order_id = o.order_id
join   `Target.customers` c
on     o.customer_id = c.customer_id
group by c.customer_state
order by avg_freight_value
limit 5;
```

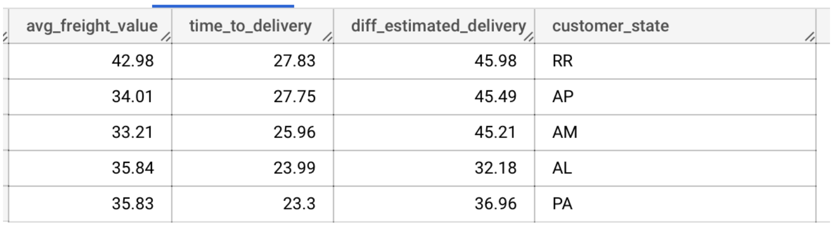


avg_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
15.15	8.26	18.9	SP
20.53	11.48	24.38	PR
20.63	11.52	24.31	MG
20.96	14.69	26.1	RJ
21.04	12.5	24.19	DF

6) Top 5 States with highest/lowest average time to delivery

- Top 5 States with highest Average time_to_delivery

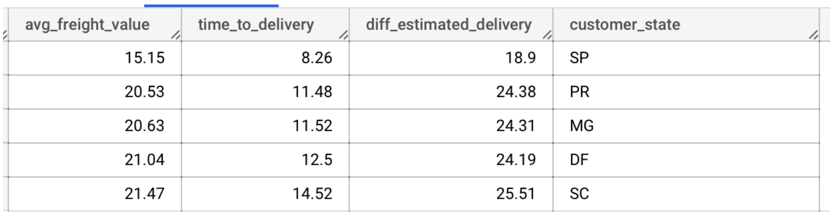
```
select   Round(avg(freight_value),2) as avg_freight_value,
Round(avg(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
Day)),2) as time_to_delivery,
Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp,
Day)),2) as diff_estimated_delivery,
c.customer_state
from   `Target.order_items` oi
join   `Target.orders` o
on     oi.order_id = o.order_id
join   `Target.customers` c
on     o.customer_id = c.customer_id
group by c.customer_state
order by time_to_delivery desc
limit 5;
```



avg_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
42.98	27.83	45.98	RR
34.01	27.75	45.49	AP
33.21	25.96	45.21	AM
35.84	23.99	32.18	AL
35.83	23.3	36.96	PA

- **Top 5 States with lowest Average time_to_delivery**

```
select   Round(avg(freight_value),2) as avg_freight_value,
Round(avg(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
Day)),2) as time_to_delivery,
Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date, order_purchase_timestamp,
Day)),2) as diff_estimated_delivery,
c.customer_state
from   `Target.order_items` oi
join   `Target.orders` o
on     oi.order_id = o.order_id
join   `Target.customers` c
on     o.customer_id = c.customer_id
group by c.customer_state
order by time_to_delivery
limit 5;
```



avg_freight_value	time_to_delivery	diff_estimated_delivery	customer_state
15.15	8.26	18.9	SP
20.53	11.48	24.38	PR
20.63	11.52	24.31	MG
21.04	12.5	24.19	DF
21.47	14.52	25.51	SC

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

I believe delivery can be consider as faster on the basis of accumulation of 2 criteria. 1st is that the gap between estimate date and delivery is higher, 2nd is the gap between purchase and delivery date is lesser.

For example: Can we say order D is fastest because it was delivered in shortest span (4 days) or Order C is fastest because it has the highest margin in estimated date vs delivery date, or order A and B are at same level in terms of delivery ?

Can we calculate time saved in terms of percentage like below. This will tell us which one is faster.
((Estimate days - delivery days)/Estimate days)*100

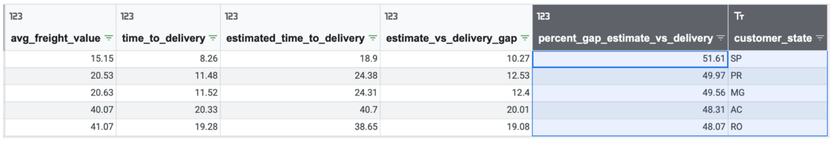
Scenario	**Estimate date(ED) - Delivery Date(DD)**	**((ED-DD)/ED)*100**	**Position**
Order A - Estimate date 35 days from purchase ; Delivery date 10 days from purchase	Gap 25 days	((35-10)/35)*100 = 71.4286	Rank 2
Order B - Estimate date 30 days from purchase ; Delivery date 05 days from purchase	Gap 25 days	((30-05)/30)*100 = 83.3333	Rank 1
Order C - Estimate date 60 days from purchase ; Delivery date 30 days from purchase	Gap 30 days	((60-30)/60)*100 = 50	Rank 4
Order D - Estimate date 10 days from purchase ; Delivery date 04 days from purchase	Gap 06 days	((10-04)/10)*100 = 60	Rank 3

Applying above concept to the data

I. Fast deliver compare to estimated date

```
select   Round(avg(freight_value),2) as avg_freight_value,
         Round(avg(TIMESTAMP_DIFF(order_delivered_customer_date,
         order_purchase_timestamp, Day)),2) as time_to_delivery,
         Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date,
         order_purchase_timestamp, Day)),2) as estimated_time_to_delivery,
         Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date,
         order_delivered_customer_date, Day)),2) as estimate_vs_delivery_gap,
         Round(Avg(case
                       when TIMESTAMP_DIFF(order_estimated_delivery_date,
                       order_delivered_customer_date, Day) = 0 then 0
                       else
                              ((TIMESTAMP_DIFF(order_estimated_delivery_date,
                              order_delivered_customer_date, Day) /
                              TIMESTAMP_DIFF(order_estimated_delivery_date,
                              order_purchase_timestamp, Day)) * 100)
                       End),2)  as percent_gap_estimate_vs_delivery,
c.customer_state
from   `Target.order_items` oi
join   `Target.orders` o
```

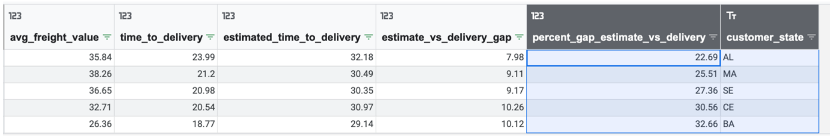
```
on     oi.order_id = o.order_id
join   `Target.customers` c
on     o.customer_id = c.customer_id
where  order_estimated_delivery_date is not null
group by c.customer_state
order by percent_gap_estimate_vs_delivery desc
limit 5;
```



avg_freight_value	time_to_delivery	estimated_time_to_delivery	estimate_vs_delivery_gap	percent_gap_estimate_vs_delivery	customer_state
15.15	8.26	18.9	10.27	51.61	SP
20.53	11.48	24.38	12.53	49.97	PR
20.63	11.52	24.31	12.4	49.56	MG
40.07	20.33	40.7	20.01	48.31	AC
41.07	19.28	38.65	19.08	48.07	RO

II. Not so fast delivery compare to estimate date

```
select   Round(avg(freight_value),2) as avg_freight_value,
         Round(avg(TIMESTAMP_DIFF(order_delivered_customer_date,
         order_purchase_timestamp, Day)),2) as time_to_delivery,
         Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date,
         order_purchase_timestamp, Day)),2) as estimated_time_to_delivery,
         Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date,
         order_delivered_customer_date, Day)),2) as estimate_vs_delivery_gap,
         Round(Avg(case
                      when TIMESTAMP_DIFF(order_estimated_delivery_date,
                      order_delivered_customer_date, Day) = 0 then 0
                      else
                            ((TIMESTAMP_DIFF(order_estimated_delivery_date,
                            order_delivered_customer_date, Day) /
                            TIMESTAMP_DIFF(order_estimated_delivery_date,
                            order_purchase_timestamp, Day)) * 100)
                      End),2)  as percent_gap_estimate_vs_delivery,
c.customer_state
from   `Target.order_items` oi
join   `Target.orders` o
on     oi.order_id = o.order_id
join   `Target.customers` c
on     o.customer_id = c.customer_id
where  order_estimated_delivery_date is not null
group by c.customer_state
order by percent_gap_estimate_vs_delivery asc
limit 5;
```

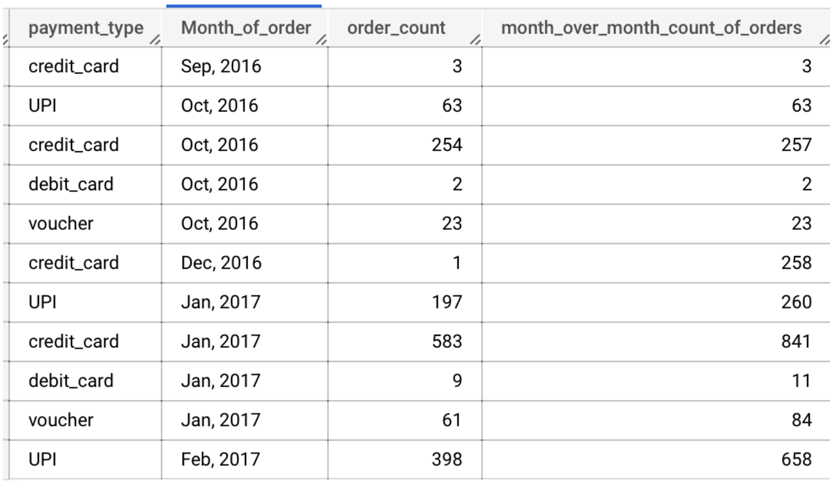


avg_freight_value	time_to_delivery	estimated_time_to_delivery	estimate_vs_delivery_gap	percent_gap_estimate_vs_delivery	customer_state
35.84	23.99	32.18	7.98	22.69	AL
38.26	21.2	30.49	9.11	25.51	MA
36.65	20.98	30.35	9.17	27.36	SE
32.71	20.54	30.97	10.26	30.56	CE
26.36	18.77	29.14	10.12	32.66	BA

6) Payment type analysis

1) Month over month count of orders for different payment types

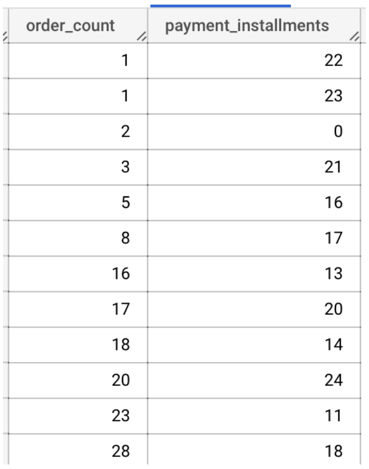
```
select   tbl1.payment_type,
Concat(Case
When tbl1.month = 01 then 'Jan'
When tbl1.month = 02 then 'Feb'
When tbl1.month = 03 then 'Mar'
When tbl1.month = 04 then 'Apr'
When tbl1.month = 05 then 'May'
When tbl1.month = 06 then 'June'
When tbl1.month = 07 then 'July'
When tbl1.month = 08 then 'Aug'
When tbl1.month = 09 then 'Sep'
When tbl1.month = 10 then 'Oct'
When tbl1.month = 11 then 'Nov'
When tbl1.month = 12 then 'Dec'
End,', ',
tbl1.year) as Month_of_order,
tbl1.No_of_orders as order_count,
Sum(tbl1.No_of_orders) over(partition by tbl1.payment_type order by tbl1.year, tbl1.month)
as month_over_month_count_of_orders
from
(select
count(ord.order_id) as No_of_orders,
pmt.payment_type,
extract(year from ord.order_purchase_timestamp) as year, extract(month from
ord.order_purchase_timestamp) as month
from `Target.orders` ord
join `Target.payments` pmt
on ord.order_id = pmt.order_id
group by year, month, pmt.payment_type
order by year, month, pmt.payment_type
) tbl1
order by tbl1.year, tbl1.month, payment_type;
```



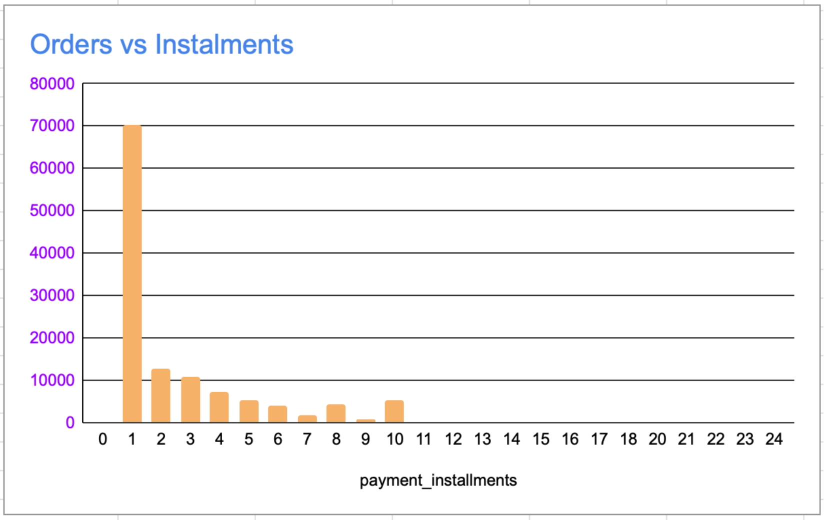
payment_type	Month_of_order	order_count	month_over_month_count_of_orders
credit_card	Sep, 2016	3	3
UPI	Oct, 2016	63	63
credit_card	Oct, 2016	254	257
debit_card	Oct, 2016	2	2
voucher	Oct, 2016	23	23
credit_card	Dec, 2016	1	258
UPI	Jan, 2017	197	260
credit_card	Jan, 2017	583	841
debit_card	Jan, 2017	9	11
voucher	Jan, 2017	61	84
UPI	Feb, 2017	398	658

2) Count of orders based on the no. of payment instalments

```
select count(pmt.order_id) as order_count,
pmt.payment_installments
from `Target.payments` ord
join  `Target.payments` pmt
on  ord.order_id = pmt.order_id
group by payment_installments
order by order_count;
```



order_count	payment_installments
1	22
1	23
2	0
3	21
5	16
8	17
16	13
17	20
18	14
20	24
23	11
28	18



Actionable Insights

i. Black Friday 2017 had the highest sales and November, 2017 with most sales.

ii. Good customer base spreaded across 4310 cities.

iii. There was approx. 20% increase in sales from 2017 to 2018 (until oct) and it doesn't include the possible November month which could have been highest contributor in 2018 sales. So there are chances of increase in sales to 20% by 2019.

iv. Highest orders are placed between Afternoon followed by Evening, Late Evening and Late Morning so these are the busiest time zone. Least orders during the Dawn.

v. Max Contributing State – SP and Least Contributing State – TO   (in terms of sales)

vi. Max sales between 2017 and 2018 (oct) is contributed by products with price approx. equals to 125 BRL (assumed as local currency of Brazil) and most overall sales by product between 124 BRL to 150 BRL.

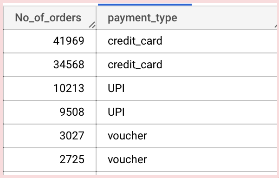
vii. States with lowest average_freight_value are also the states with max sales.

viii. Top 5 states(SP,PR,MG,AC,RO) with fast delivery have high gap between delivery and estimate-delivery date.

ix. Highest orders are made through credit card followed by UPI.

x. Maximum customers buy their orders through 1 time payment.

Commented [SS3]:



No_of_orders	payment_type
41969	credit_card
34568	credit_card
10213	UPI
9508	UPI
3027	voucher
2725	voucher

Recommendations:

i. Stock availability should be increased by x% in last quarter of the year and so does the delivery staff, customer care staff and warehouse capacity.

ii. Most sold product on Black Friday was – xxxx, so there should be addition of the products of this category. Early sales can be started before black Friday to maintain customer traffic.

iii. Busy hours are from 10:00 AM to 10:00 PM, so there should be higher percentage of employees covering these hours.

iv. Daily maintenance activities or downtime activities can be performed between 3:00 AM to 5:00 AM.

v. There can be exibitions organized in SP state to further enhance the popularity of 'Target' as their contribution to the total sales is even higher than the sum of sales of top 5 states next to them.

vi. TO – State should should get offline stores of small size with good product variety but limited stock, so People of TO could easily access the stores as well as get the sense of variety and quality of products, which could potentially lead to increase in sales in coming years.

vii. 125 BRL is sweet spot for the products pricing, so the enough products quantity/pricing should be amend to fall into this category.

viii. Freight_value should be controlled and decreased so net sales can be increased in non performing states.

ix. Estimate delivery date should be realistic and in accordance with the average delivery date. Estimate delivery date too far from date of purchase could lead to loss of potential orders and that should be reduced. (Use 5.5 & 5.6 as evidence)

x. Credit card with no cost EMI benefits and discounts should be introduced to pull more customers and increase continuous flow of income.

Extras:

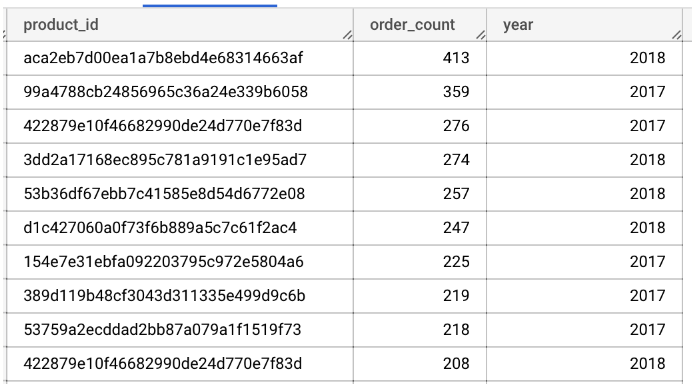
a) TOP 5 products from 2017 and 2018.

Recommendation - These product should be categorized and more products from these categories with high profit margin can be can be added to portfolio.

```
select product_id, count(product_id) as order_count,
       extract(year from o.order_purchase_timestamp) as year
  from `Target.order_items` oi
  join `Target.orders` o
    on o.order_id = oi.order_id
group by year,product_id
```

Commented [SS4]:

```
select product_id, count(product_id) as
order_count,
       extract(year from
o.order_purchase_timestamp) as year
  from `Target.order_items` oi
  join `Target.orders` o
    on o.order_id = oi.order_id
group by year,product_id
order by order_count desc,product_id;
order by order_count desc,product_id;
```

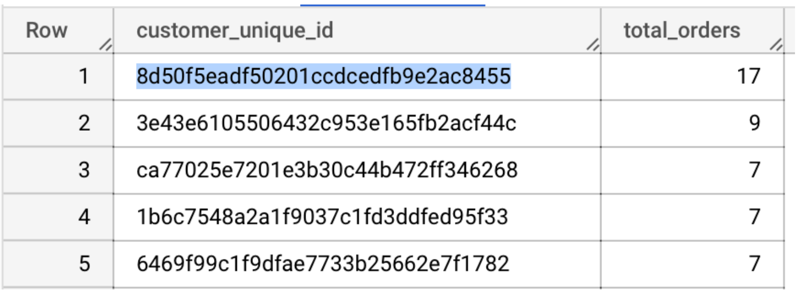


product_id	order_count	year
aca2eb7d00ea1a7b8ebd4e68314663af	413	2018
99a4788cb24856965c36a24e339b6058	359	2017
422879e10f46682990de24d770e7f83d	276	2017
3dd2a17168ec895c781a9191c1e95ad7	274	2018
53b36df67ebb7c41585e8d54d6772e08	257	2018
d1c427060a0f73f6b889a5c7c61f2ac4	247	2018
154e7e31ebfa092203795c972e5804a6	225	2017
389d119b48cf3043d311335e499d9c6b	219	2017
53759a2ecddad2bb87a079a1f1519f73	218	2017
422879e10f46682990de24d770e7f83d	208	2018

b) Top 5 Customers with **maximum number of orders** and Customer with maximum orders highlighted.

Recommendation: They can be added to loyalty programs and can be provided with benefit of referral coupons. They can also be encouraged to participate in product review program as they have high potential to visit the store/website etc.

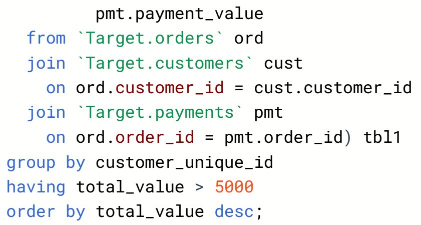
```
select   distinct customer_unique_id,
         count(order_id) total_orders
  from `Target.orders` ord
  join `Target.customers` cust
    on ord.customer_id = cust.customer_id
 group by customer_unique_id
  order by total_orders desc;
```



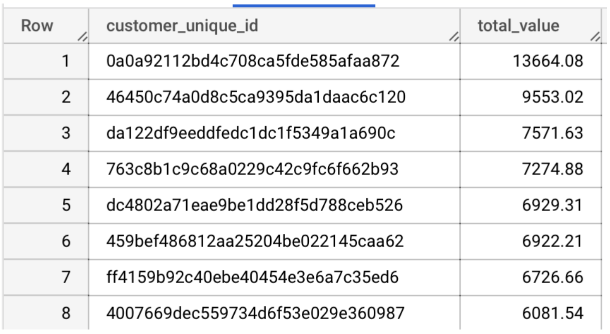
Row	customer_unique_id	total_orders
1	8d50f5eadf50201ccdcedfb9e2ac8455	17
2	3e43e6105506432c953e165fb2acf44c	9
3	ca77025e7201e3b30c44b472ff346268	7
4	1b6c7548a2a1f9037c1fd3ddfed95f33	7
5	6469f99c1f9dfae7733b25662e7f1782	7

c) **Priority Customers** i.e. with maximum buying i.e. greater than 5000 BRL (Brazilian Currency)

```
select tbl1.customer_unique_id,
       sum(tbl1.payment_value) as total_value
from(
select  customer_unique_id,
         pmt.order_id,
```



```
         pmt.payment_value
  from `Target.orders` ord
  join `Target.customers` cust
    on ord.customer_id = cust.customer_id
  join `Target.payments` pmt
    on ord.order_id = pmt.order_id) tbl1
group by customer_unique_id
having total_value > 5000
order by total_value desc;
```



Row	customer_unique_id	total_value
1	0a0a92112bd4c708ca5fde585afaa872	13664.08
2	46450c74a0d8c5ca9395da1daac6c120	9553.02
3	da122df9eeddfedc1dc1f5349a1a690c	7571.63
4	763c8b1c9c68a0229c42c9fc6f662b93	7274.88
5	dc4802a71eae9be1dd28f5d788ceb526	6929.31
6	459bef486812aa25204be022145caa62	6922.21
7	ff4159b92c40ebe40454e3e6a7c35ed6	6726.66
8	4007669dec559734d6f53e029e360987	6081.54

Recommendations – These customers can be added to the advertisement program and should be provided with recommendations of products of their preferred category, plus also the early sale access. These can be also awarded with free gifts as a gesture.