Business Case: Target SQ

Task:

- Finiding the date type, time period of data, cites and state covered in data set
- Monthly and timely sale analysis to understand the behaviour of customer
- Trend of sales region wise and customer distribution
- Price increase analysis
- payment type/method analysis

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
customers=pd.read_csv(r'C:\Users\sudhanshu tomar\Desktop\datasets\project 6
```

In [17]:
```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 5 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   customer_id               99441 non-null  object
 1   customer_unique_id        99441 non-null  object
 2   customer_zip_code_prefix  99441 non-null  int64 
 3   customer_city             99441 non-null  object
 4   customer_state            99441 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
```

In [3]:
```
geolocation=pd.read_csv(r'C:\Users\sudhanshu tomar\Desktop\datasets\project
```

In [22]:
```
geolocation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000163 entries, 0 to 1000162
Data columns (total 5 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   geolocation_zip_code_prefix  1000163 non-null  int64  
 1   geolocation_lat              1000163 non-null  float64
 2   geolocation_lng              1000163 non-null  float64
 3   geolocation_city             1000163 non-null  object 
 4   geolocation_state            1000163 non-null  object 
dtypes: float64(2), int64(1), object(2)
memory usage: 38.2+ MB
```

In [4]:
```
order_items=pd.read_csv(r'C:\Users\sudhanshu tomar\Desktop\datasets\project
```

In [23]:
```
order_items.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   order_id             112650 non-null  object 
 1   order_item_id        112650 non-null  int64  
 2   product_id           112650 non-null  object 
 3   seller_id            112650 non-null  object 
 4   shipping_limit_date  112650 non-null  object 
 5   price                112650 non-null  float64
 6   freight_value        112650 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.0+ MB
```

In [15]:
```
order_reviews=pd.read_csv(r'C:\Users\sudhanshu tomar\Desktop\datasets\projec
```

In [24]:
```
order_reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99224 entries, 0 to 99223
Data columns (total 6 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   review_id                99224 non-null  object
 1   order_id                 99224 non-null  object
 2   review_score             99224 non-null  int64 
 3   review_comment_title     11549 non-null  object
 4   review_creation_date     99224 non-null  object
 5   review_answer_timestamp  99224 non-null  object
dtypes: int64(1), object(5)
memory usage: 4.5+ MB
```

In [6]:

```python
orders=pd.read_csv(r'C:\Users\sudhanshu tomar\Desktop\datasets\project 6 mys
```

In [25]:

```python
orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  object
 4   order_approved_at              99281 non-null  object
 5   order_delivered_carrier_date   97658 non-null  object
 6   order_delivered_customer_date  96476 non-null  object
 7   order_estimated_delivery_date  99441 non-null  object
dtypes: object(8)
memory usage: 6.1+ MB
```

In [7]:

```python
payments=pd.read_csv(r'C:\Users\sudhanshu tomar\Desktop\datasets\project 6 m
```

In [26]:

```python
payments.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103886 entries, 0 to 103885
Data columns (total 5 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   order_id              103886 non-null  object 
 1   payment_sequential    103886 non-null  int64  
 2   payment_type          103886 non-null  object 
 3   payment_installments  103886 non-null  int64  
 4   payment_value         103886 non-null  float64
dtypes: float64(1), int64(2), object(2)
memory usage: 4.0+ MB
```

In [8]:

```python
products=pd.read_csv(r'C:\Users\sudhanshu tomar\Desktop\datasets\project 6 m
```

In [27]:
```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   product_id                  32951 non-null  object 
 1   product category            32341 non-null  object 
 2   product_name_length         32341 non-null  float64
 3   product_description_length  32341 non-null  float64
 4   product_photos_qty          32341 non-null  float64
 5   product_weight_g            32949 non-null  float64
 6   product_length_cm           32949 non-null  float64
 7   product_height_cm           32949 non-null  float64
 8   product_width_cm            32949 non-null  float64
dtypes: float64(7), object(2)
memory usage: 2.3+ MB
```

In [9]:
```
sellers=pd.read_csv(r'C:\Users\sudhanshu tomar\Desktop\datasets\project 6 my
```

In [28]:
```
sellers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3095 entries, 0 to 3094
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   seller_id               3095 non-null   object
 1   seller_zip_code_prefix  3095 non-null   int64 
 2   seller_city             3095 non-null   object
 3   seller_state            3095 non-null   object
dtypes: int64(1), object(3)
memory usage: 96.8+ KB
```

In [43]:
```
orders.head()
```

Out[43]:

	order_id	customer_id	order_status	order_purcl
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	201
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	201
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	201
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	201
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	201

In [49]:
```
orders['order_purchase_timestamp']=pd.to_datetime(orders['order_purchase_tim
```

In [64]:
```
print(orders['order_purchase_timestamp'].max())
print(orders['order_purchase_timestamp'].min())
print(orders['order_purchase_timestamp'].max()-orders['order_purchase_timest
```

```
2018-10-17 17:30:18
2016-09-04 21:15:19
772 days 20:14:59
```

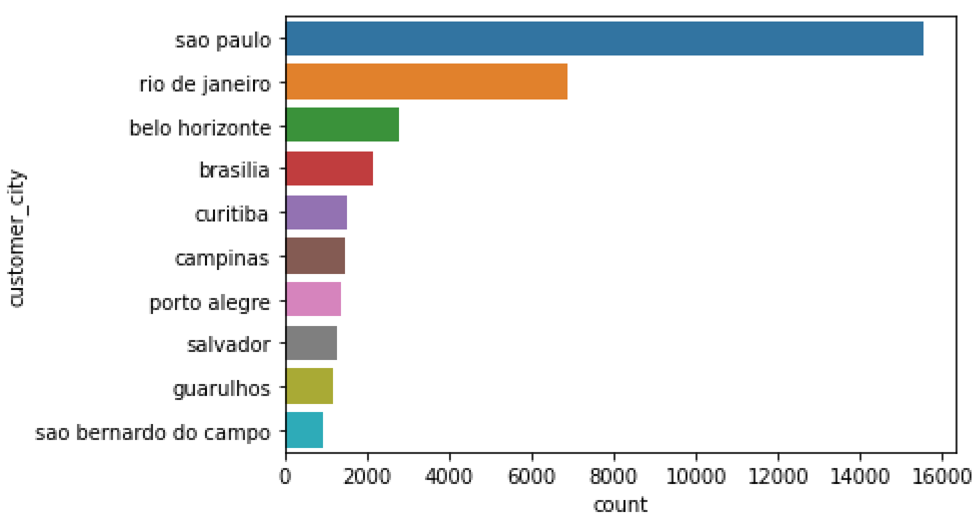
This data covers a total time of 772 days, between 2016-09-04 to 2018-10-17.

In [66]:
```
customers['customer_city'].value_counts()
```

Out[66]:
```
sao paulo             15540
rio de janeiro         6882
belo horizonte         2773
brasilia               2131
curitiba               1521
                      ...
bequimao                  1
andarai                   1
vargem grande             1
curvelandia               1
eugenio de castro         1
Name: customer_city, Length: 4119, dtype: int64
```

In [74]:
```
sns.countplot(y=customers['customer_city'],order=customers['customer_city'].
```

Out[74]: <AxesSubplot:xlabel='count', ylabel='customer_city'>



Most of the customer are based in sao paulo city.

In [75]:

```
customers['customer_state'].value_counts()
```

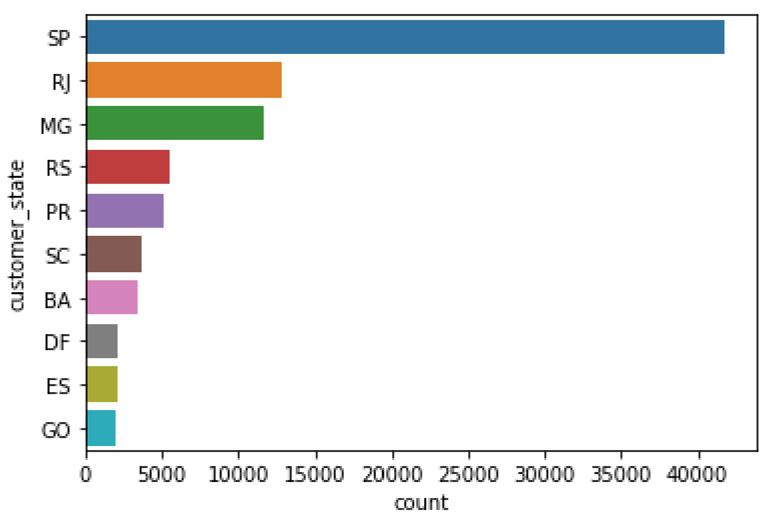
Out[75]:

```
SP    41746
RJ    12852
MG    11635
RS     5466
PR     5045
SC     3637
BA     3380
DF     2140
ES     2033
GO     2020
PE     1652
CE     1336
PA      975
MT      907
MA      747
MS      715
PB      536
PI      495
RN      485
AL      413
SE      350
TO      280
RO      253
AM      148
AC       81
AP       68
RR       46
Name: customer_state, dtype: int64
```

In [73]:

```
sns.countplot(y=customers['customer_state'],order=customers['customer_state'
```

Out[73]: <AxesSubplot:xlabel='count', ylabel='customer_state'>

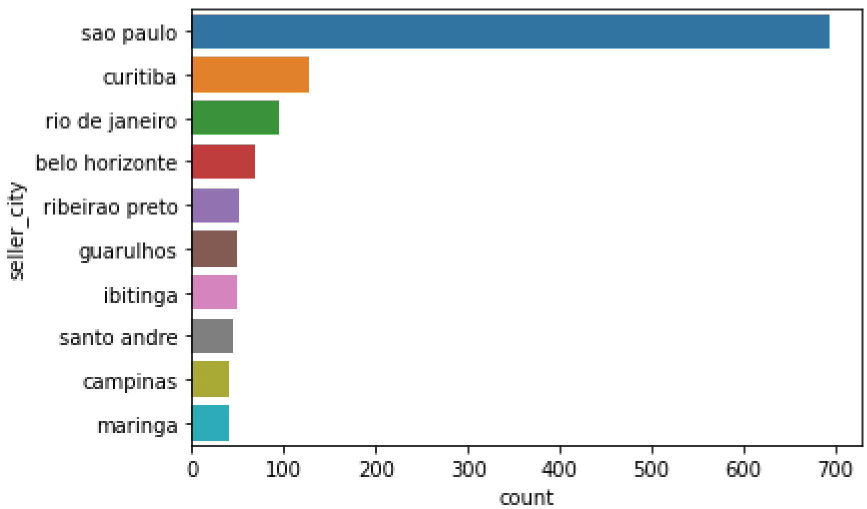


In [77]:
```
sellers['seller_city'].value_counts()
```

Out[77]:
```
sao paulo                694
curitiba                 127
rio de janeiro            96
belo horizonte            68
ribeirao preto            52
                        ...
taruma                     1
s jose do rio preto        1
domingos martins           1
messias targino            1
leme                       1
Name: seller_city, Length: 611, dtype: int64
```

In [79]:
```
sns.countplot(y=sellers['seller_city'],order=sellers['seller_city'].value_co
```

Out[79]: <AxesSubplot:xlabel='count', ylabel='seller_city'>

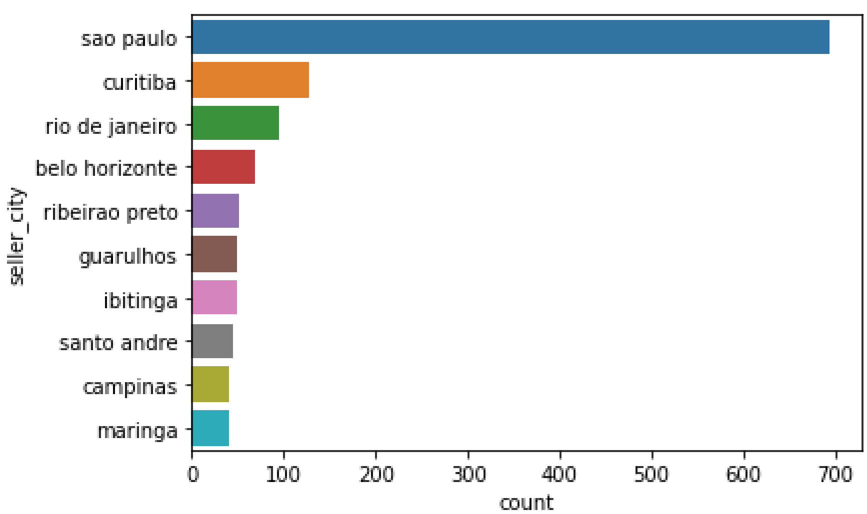


In [80]:
```
sellers['seller_state'].value_counts()
```

Out[80]:
```
SP    1849
PR     349
MG     244
SC     190
RJ     171
RS     129
GO      40
DF      30
ES      23
BA      19
CE      13
PE       9
PB       6
RN       5
MS       5
MT       4
RO       2
SE       2
PI       1
AC       1
MA       1
AM       1
PA       1
Name: seller_state, dtype: int64
```

In [81]:
```
sns.countplot(y=sellers['seller_city'],order=sellers['seller_city'].value_co
```

Out[81]: <AxesSubplot:xlabel='count', ylabel='seller_city'>



Maximum sellers are also from sao paulo.

In [ ]:
```

```

RUN SAVE SHARE SCHEDULE MORE

```sql
SELECT
format_date("%Y_%m", order_purchase_timestamp) as month_of_order,
count(*)
FROM
    `dsml-scaler-sql-365405.Target.orders`
group by month_of_order
order by month_of_order
limit 1000
```

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	month_of_order	f0_
1	2016_09	4

RUN SAVE SHARE SCHEDULE MORE

```sql
SELECT
case
when
extract(hour from order_purchase_timestamp) >= 5 and
          extract(hour from order_purchase_timestamp) < 12
      then 'morning'
      when extract(hour from order_purchase_timestamp) >= 12 and
          extract(hour from order_purchase_timestamp) < 17
      then 'afternoon'
      else 'evening'
 end as day_time,
count(*)
FROM
   `dsml-scaler-sql-365405.Target.orders`
group by day_time
order by day_time
limit 1000
```

RUN SAVE SHARE SCHEDULE MORE

```sql
SELECT customer_city,
format_date("%Y_%m", order_purchase_timestamp) as month_of_order,

count(*)
 FROM `dsml-scaler-sql-365405.Target.customers`
left join `dsml-scaler-sql-365405.Target.orders` using (customer_id)
group by month_of_order, customer_city
order by month_of_order
LIMIT 1000
```

RUN SAVE SHARE SCHEDULE MORE

```
SELECT
  FORMAT_DATE("%Y_%m",order_purchase_timestamp) AS year_month,
  payment_type,
  COUNT(*)
FROM
  `dsml-scaler-sql-365405.Target.payments`
LEFT JOIN
  `dsml-scaler-sql-365405.Target.orders`
USING
  (order_id)
GROUP BY
  year_month,
  payment_type
ORDER BY
  year_month
```

RUN SAVE SHARE SCHEDU

```sql
SELECT
    payment_installments,
  COUNT(*)
FROM
  `dsml-scaler-sql-365405.Target.payments`
group by payment_installments
LIMIT
  1000
```

Query results

JOB INFORMATION RESULTS JSON EXE

Row	payment_in...	f0_
1	0	2
2	1	52546
3	2	12413

RUN SAVE SHARE SCHEDULE MORE

```
SELECT
  customer_state,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, day)),2) AS diff_estimated_delivery,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)),2) AS time_to_delivery
FROM
  `dsml-scaler-sql-365405.Target.orders`
LEFT JOIN
  `dsml-scaler-sql-365405.Target.customers`
USING
  (customer_id)
WHERE
  DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, day) IS NOT NULL
  AND DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) IS NOT NULL
GROUP BY
  customer_state
LIMIT
```

RUN SAVE SHARE SCHEDULE MORE

```sql
SELECT
  customer_state,
  avg(freight_value) as avg_freight_value
FROM
  `dsml-scaler-sql-365405.Target.order-items`
LEFT JOIN
  `dsml-scaler-sql-365405.Target.orders`
USING
  (order_id)
LEFT JOIN
  `dsml-scaler-sql-365405.Target.customers`
USING
  (customer_id)
  group by (customer_state)
order by avg_freight_value
LIMIT
  1000
```

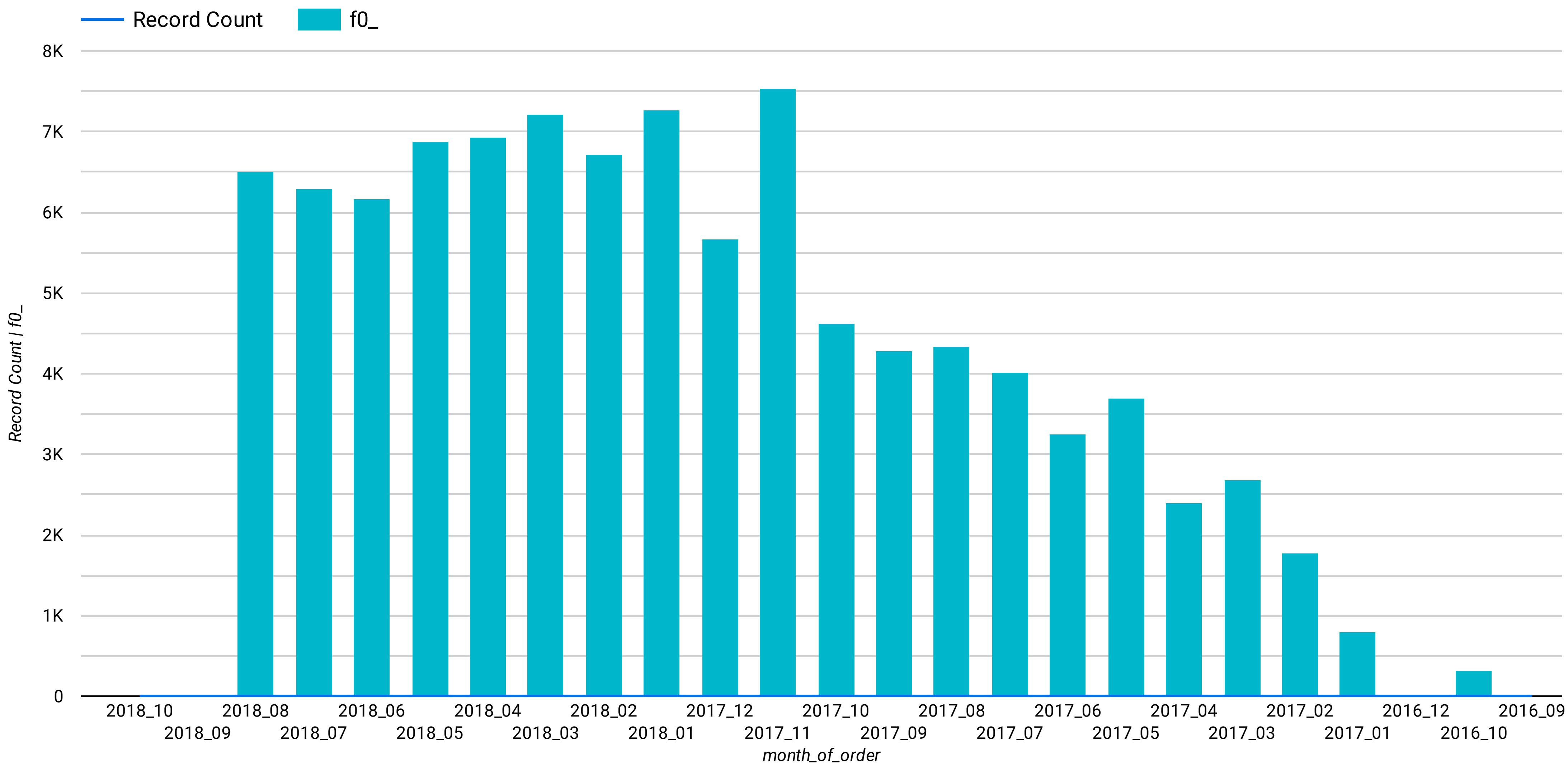
RUN SAVE SHARE SCHEDULE MORE

```
SELECT
format_date("%Y_%m", order_purchase_timestamp) as month_of_order,
count(*)
FROM
    `dsml-scaler-sql-365405.Target.orders`
group by month_of_order
order by month_of_order
limit 1000
```

Month wise sale analysis

	month_of_order ▾	f0_
1.	2018_10	4
2.	2018_09	16
3.	2018_08	6,512
4.	2018_07	6,292
5.	2018_06	6,167
6.	2018_05	6,873
7.	2018_04	6,939
8.	2018_03	7,211
9.	2018_02	6,728
10.	2018_01	7,269
11.	2017_12	5,673
12.	2017_11	7,544
13.	2017_10	4,631
14.	2017_09	4,285
15.	2017_08	4,331
16.	2017_07	4,026
17.	2017_06	3,245
18.	2017_05	3,700
19.	2017_04	2,404
20.	2017_03	2,682
21.	2017_02	1,780
22.	2017_01	800
23.	2016_12	1
24.	2016_10	324
25.	2016 09	4

1 - 25 / 25 ‹ ›

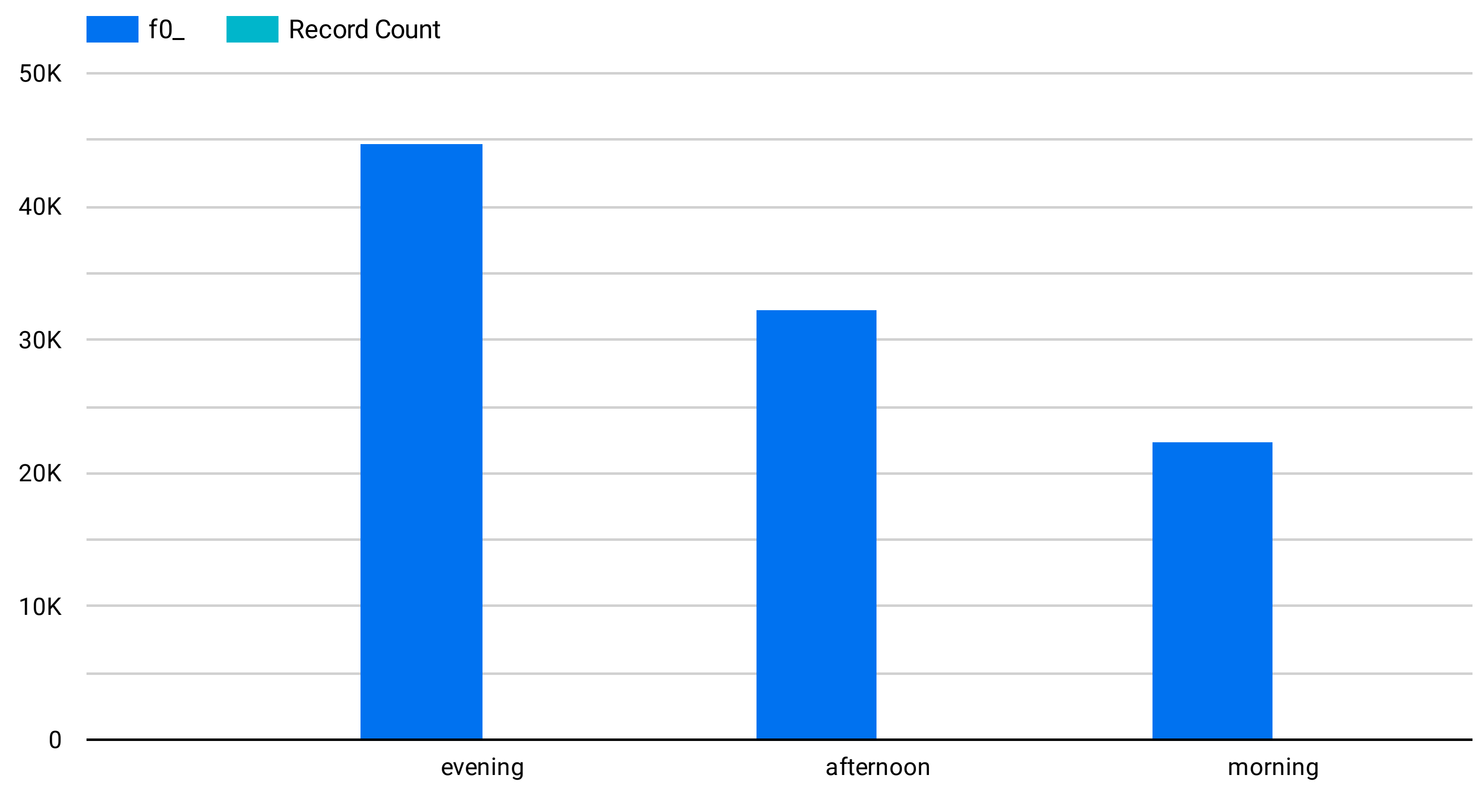


It is clear from above graph that sales increased after November 2017, which is also the peak sale month.

Time wise comparison of sale

	day_time	f0_ ▾
1.	evening	44,802
2.	afternoon	32,211
3.	morning	22,428

1 - 3 / 3

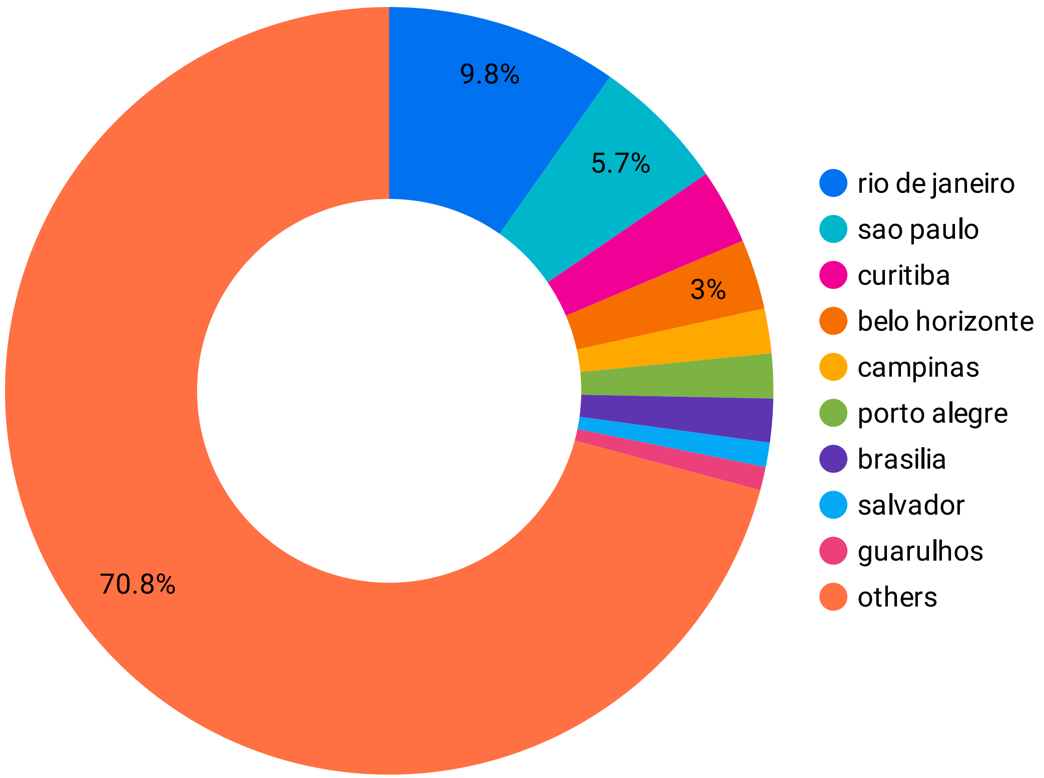


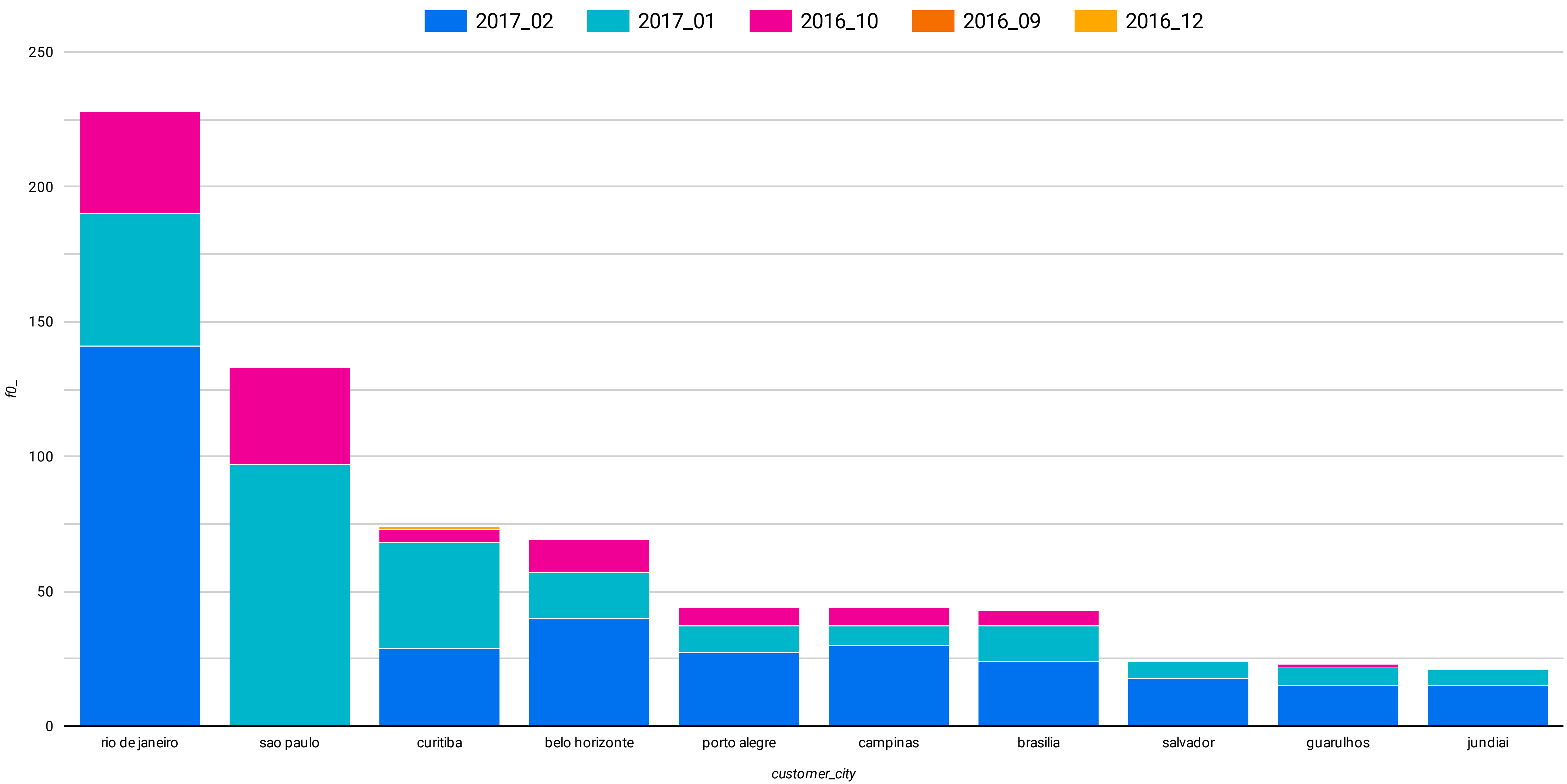
Most of the people prefer to buy in evening time.

Customer distribution based on cities

	customer_city	f0_ ▾
1.	rio de janeiro	228
2.	sao paulo	133
3.	curitiba	74
4.	belo horizonte	69
5.	campinas	44
6.	porto alegre	44
7.	brasilia	43
8.	salvador	24
9.	guarulhos	23
10.	ribeirao preto	21
11.	jundiai	21
12.	contagem	20
13.	juiz de fora	19
14.	goiania	17
15.	florianopolis	16
16.	fortaleza	15
17.	santos	14
18.	cuiaba	14
19.	sao carlos	14
20.	santo andre	14
21.	belem	13
22.	sao goncalo	13

1 - 50 / 725



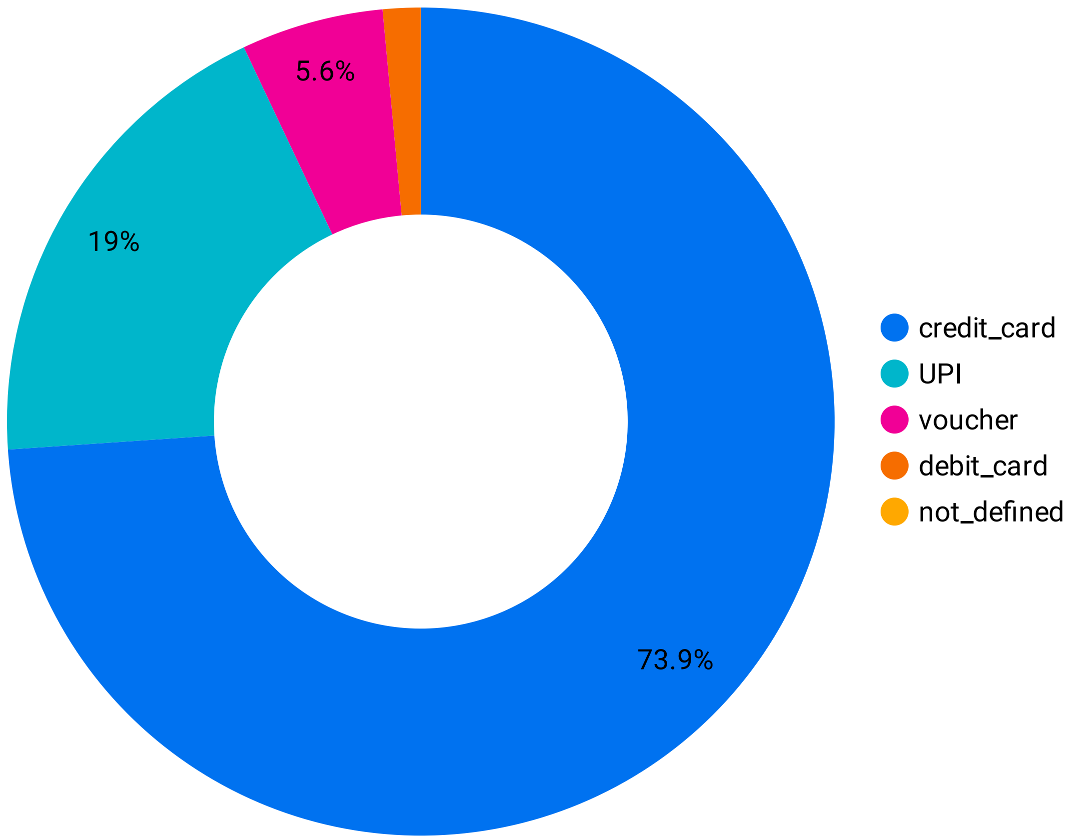


Maximum orders are from Rio de Janerio followed by Sao Paulo.

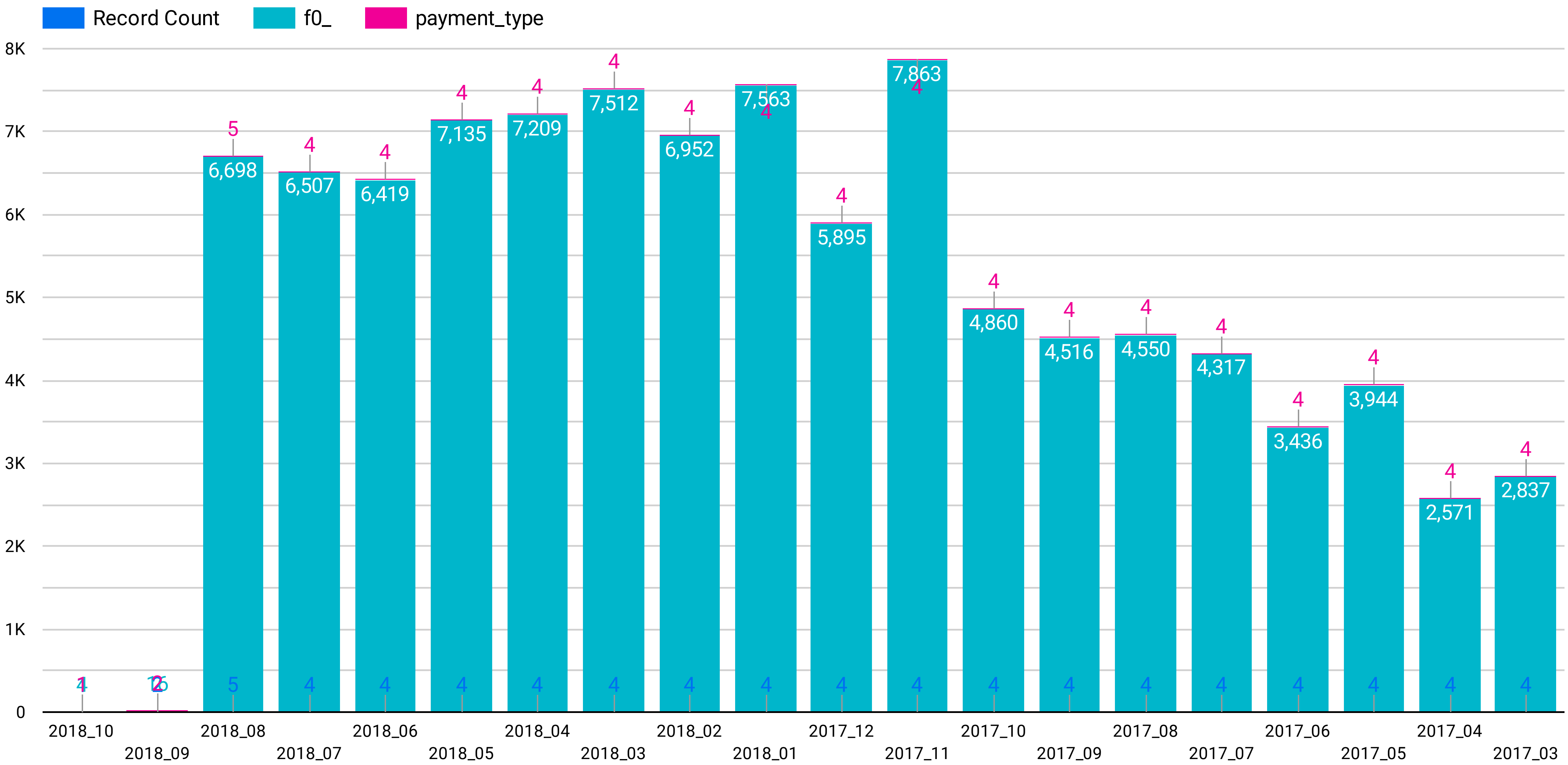
Method of payment used by customers

	year_month	payment_type	f0_ ▾
1.	2017_11	credit_card	5,897
2.	2018_03	credit_card	5,691
3.	2018_01	credit_card	5,520
4.	2018_05	credit_card	5,497
5.	2018_04	credit_card	5,455
6.	2018_02	credit_card	5,253
7.	2018_08	credit_card	4,985
8.	2018_06	credit_card	4,813
9.	2018_07	credit_card	4,755
10.	2017_12	credit_card	4,377
11.	2017_10	credit_card	3,524
12.	2017_08	credit_card	3,284
13.	2017_09	credit_card	3,283
14.	2017_07	credit_card	3,086
15.	2017_05	credit_card	2,853
16.	2017_06	credit_card	2,463
17.	2017_03	credit_card	2,016
18.	2017_04	credit_card	1,846
19.	2018_01	UPI	1,518
20.	2017_11	UPI	1,509

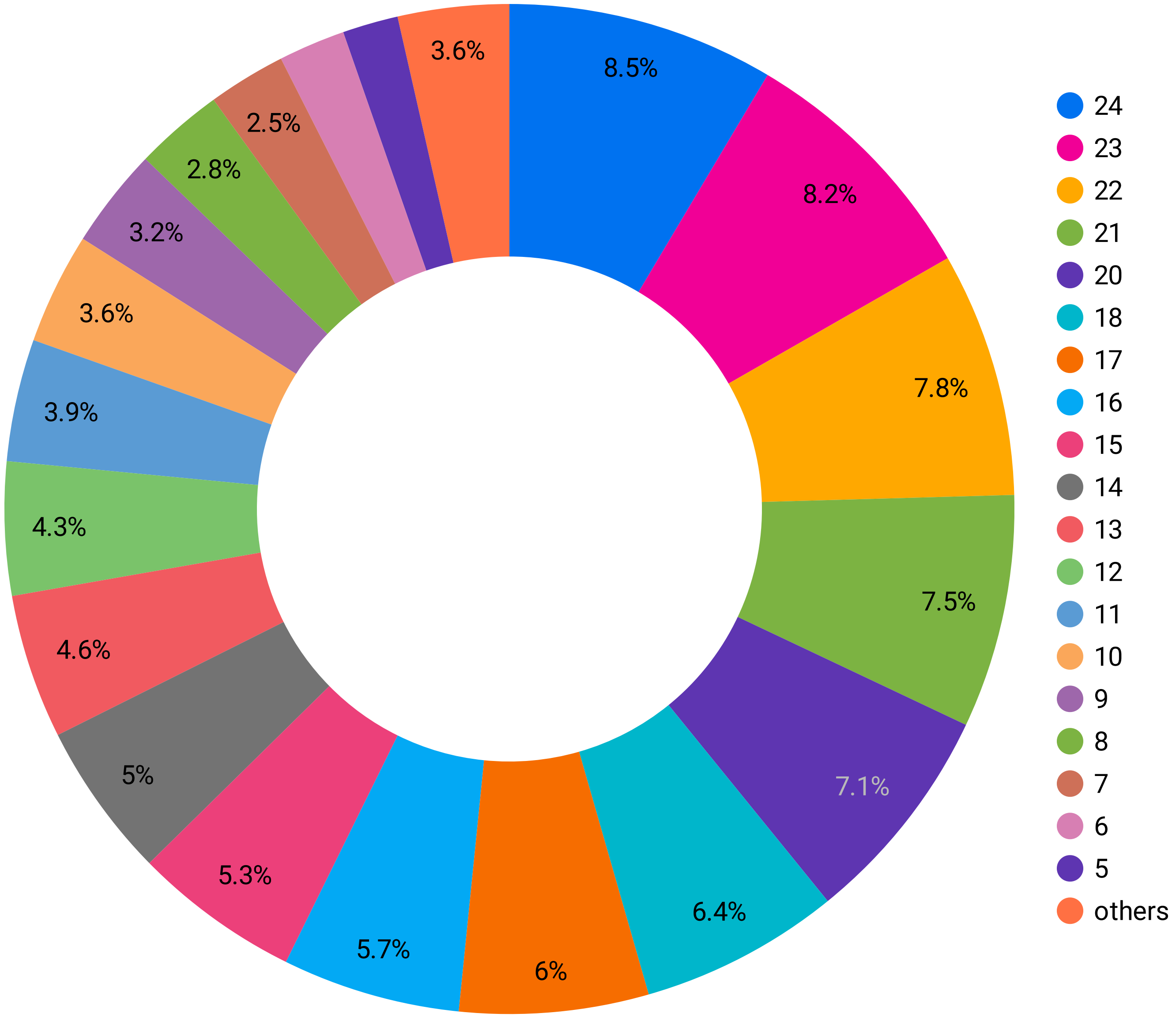
1 - 50 / 90



Majority of payments are done by credit cards followed by upi and vouchers


Record Count
f0_
payment_type
8K
7K
6K
5K
4K
3K
2K
1K
0
6,698
6,507
6,419
7,135
7,209
7,512
6,952
7,563
5,895
7,863
4,860
4,516
4,550
4,317
3,436
3,944
2,571
2,837
2018_10
2018_09
2018_08
2018_07
2018_06
2018_05
2018_04
2018_03
2018_02
2018_01
2017_12
2017_11
2017_10
2017_09
2017_08
2017_07
2017_06
2017_05
2017_04
2017_03

Distribution of Installment duration



Most of the people opt for 21 to 24 month installment payment method

Estimated and actual time delivery time comparison

Highest to lowest delivery time

	customer_state	diff_estimated_delivery	time_to_delivery
1.	AP	45.87	26.73
2.	RR	45.63	28.98
3.	AM	44.92	25.99
4.	AC	40.72	20.64
5.	RO	38.39	18.91
6.	PA	36.79	23.32
7.	PB	32.65	19.95
8.	AL	32.21	24.04
9.	RN	31.87	18.82
10.	MT	31.37	17.59
11.	CE	31	20.82
12.	PE	30.69	17.97
13.	SE	30.48	21.03
14.	MA	30.08	21.12
15.	PI	29.7	18.99
16.	BA	29.07	18.87
17.	TO	28.73	17.23
18.	RS	28.16	14.82
19.	GO	26.72	15.15
20.	RJ	26	14.85

1 - 27 / 27

Lowest to highest delivery time

	customer_state	diff_estimated_delivery ▲	time_to_delivery
1.	SP	18.78	8.3
2.	DF	23.95	12.51
3.	MG	24.19	11.54
4.	PR	24.25	11.53
5.	ES	25.22	15.33
6.	SC	25.42	14.48
7.	MS	25.6	15.19
8.	RJ	26	14.85
9.	GO	26.72	15.15
10.	RS	28.16	14.82
11.	TO	28.73	17.23
12.	BA	29.07	18.87
13.	PI	29.7	18.99
14.	MA	30.08	21.12
15.	SE	30.48	21.03
16.	PE	30.69	17.97
17.	CE	31	20.82
18.	MT	31.37	17.59
19.	RN	31.87	18.82
20.	AL	32.21	24.04

1 - 27 / 27

Comparison of estimated and actual delivery time difference

Top state where delivery is fast, Actual time is very less than estimated time

	customer_st...	diff_estimated_del...	time_to_delivery	Time diff...
1.	AC	40.72	20.64	20.08
2.	RO	38.39	18.91	19.48
3.	AP	45.87	26.73	19.14
4.	AM	44.92	25.99	18.93
5.	RR	45.63	28.98	16.65
6.	MT	31.37	17.59	13.78
7.	PA	36.79	23.32	13.47
8.	RS	28.16	14.82	13.34
9.	RN	31.87	18.82	13.05
10.	PE	30.69	17.97	12.72
11.	PR	24.25	11.53	12.72
12.	PB	32.65	19.95	12.7
13.	MG	24.19	11.54	12.65
14.	GO	26.72	15.15	11.57
15.	TO	28.73	17.23	11.5
16.	DF	23.95	12.51	11.44
17.	RJ	26	14.85	11.15
18.	SC	25.42	14.48	10.94
19.	PI	29.7	18.99	10.71
20.	SP	18.78	8.3	10.48

1 - 27 / 27 < >

Top state where delivery is slow, Actual time is not very less than estimated time

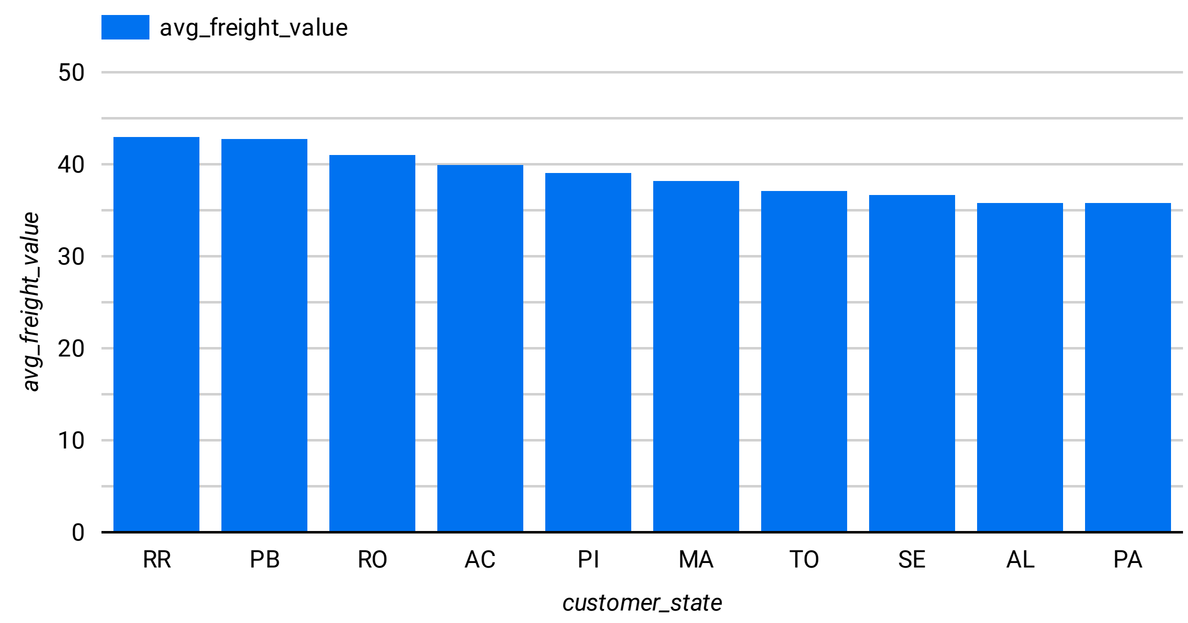
	customer_st...	diff_estimated_del...	time_to_delivery	Time diff...
1.	AL	32.21	24.04	8.17
2.	MA	30.08	21.12	8.96
3.	SE	30.48	21.03	9.45
4.	ES	25.22	15.33	9.89
5.	CE	31	20.82	10.18
6.	BA	29.07	18.87	10.2
7.	MS	25.6	15.19	10.41
8.	SP	18.78	8.3	10.48
9.	PI	29.7	18.99	10.71
10.	SC	25.42	14.48	10.94
11.	RJ	26	14.85	11.15
12.	DF	23.95	12.51	11.44
13.	TO	28.73	17.23	11.5
14.	GO	26.72	15.15	11.57
15.	MG	24.19	11.54	12.65
16.	PB	32.65	19.95	12.7
17.	PR	24.25	11.53	12.72
18.	PE	30.69	17.97	12.72
19.	RN	31.87	18.82	13.05
20.	RS	28.16	14.82	13.34

1 - 27 / 27 < >

State with highest to lowest freight value

	customer_state	avg_freight_value ▼
1.	RR	42.98
2.	PB	42.72
3.	RO	41.07
4.	AC	40.07
5.	PI	39.15
6.	MA	38.26
7.	TO	37.25
8.	SE	36.65
9.	AL	35.84
10.	PA	35.83

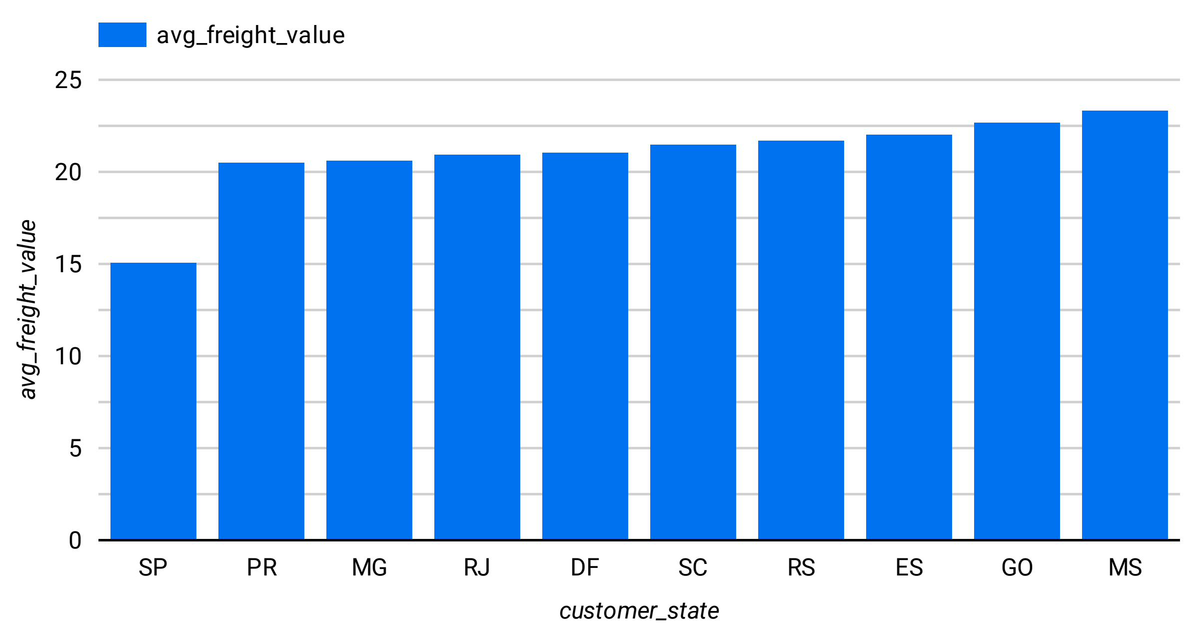
1 - 27 / 27



State with lowest to highest freight value

	customer_state	avg_freight_value ▲
1.	SP	15.15
2.	PR	20.53
3.	MG	20.63
4.	RJ	20.96
5.	DF	21.04
6.	SC	21.47
7.	RS	21.74
8.	ES	22.06
9.	GO	22.77
10.	MS	23.37

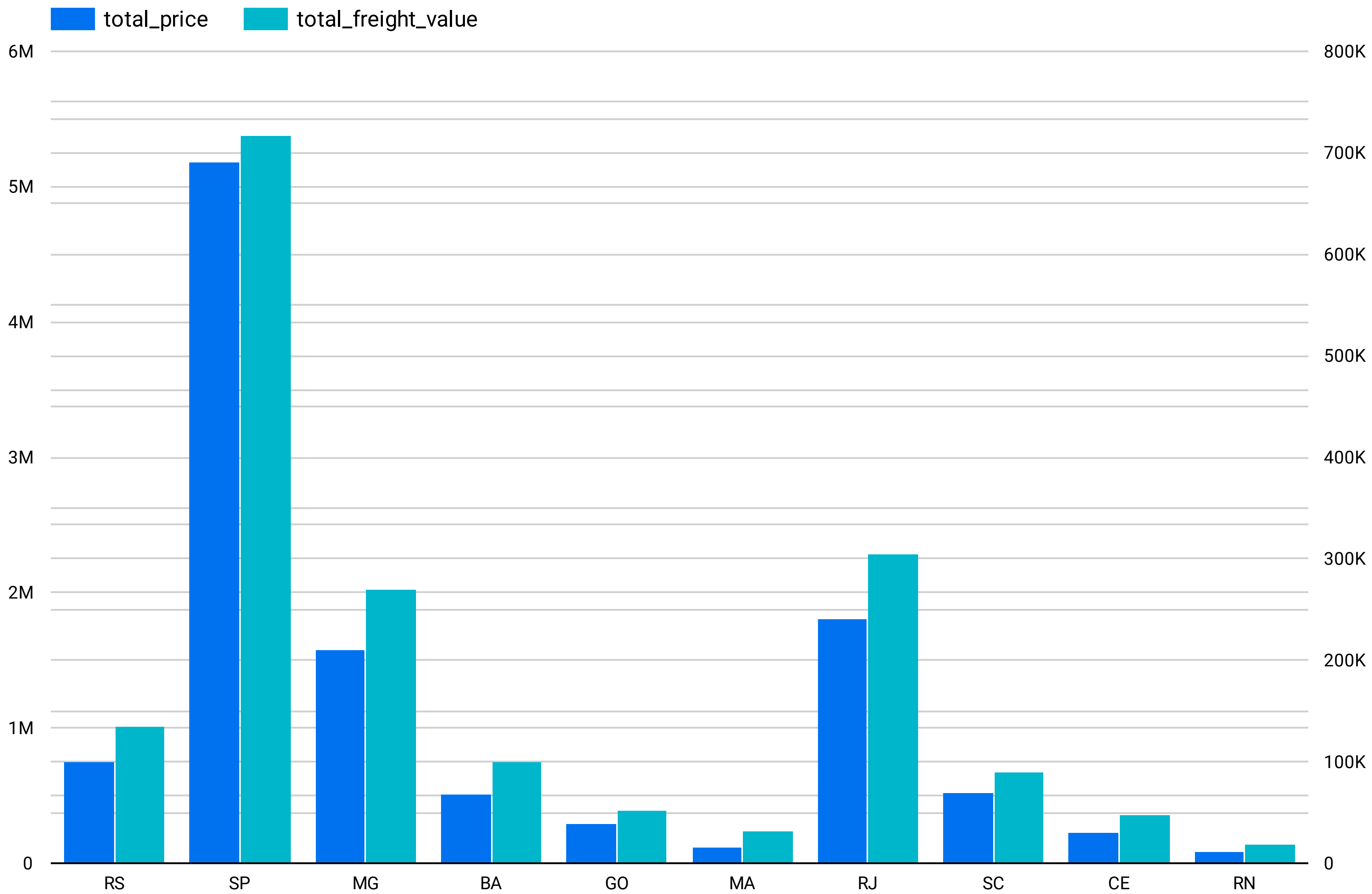
1 - 27 / 27



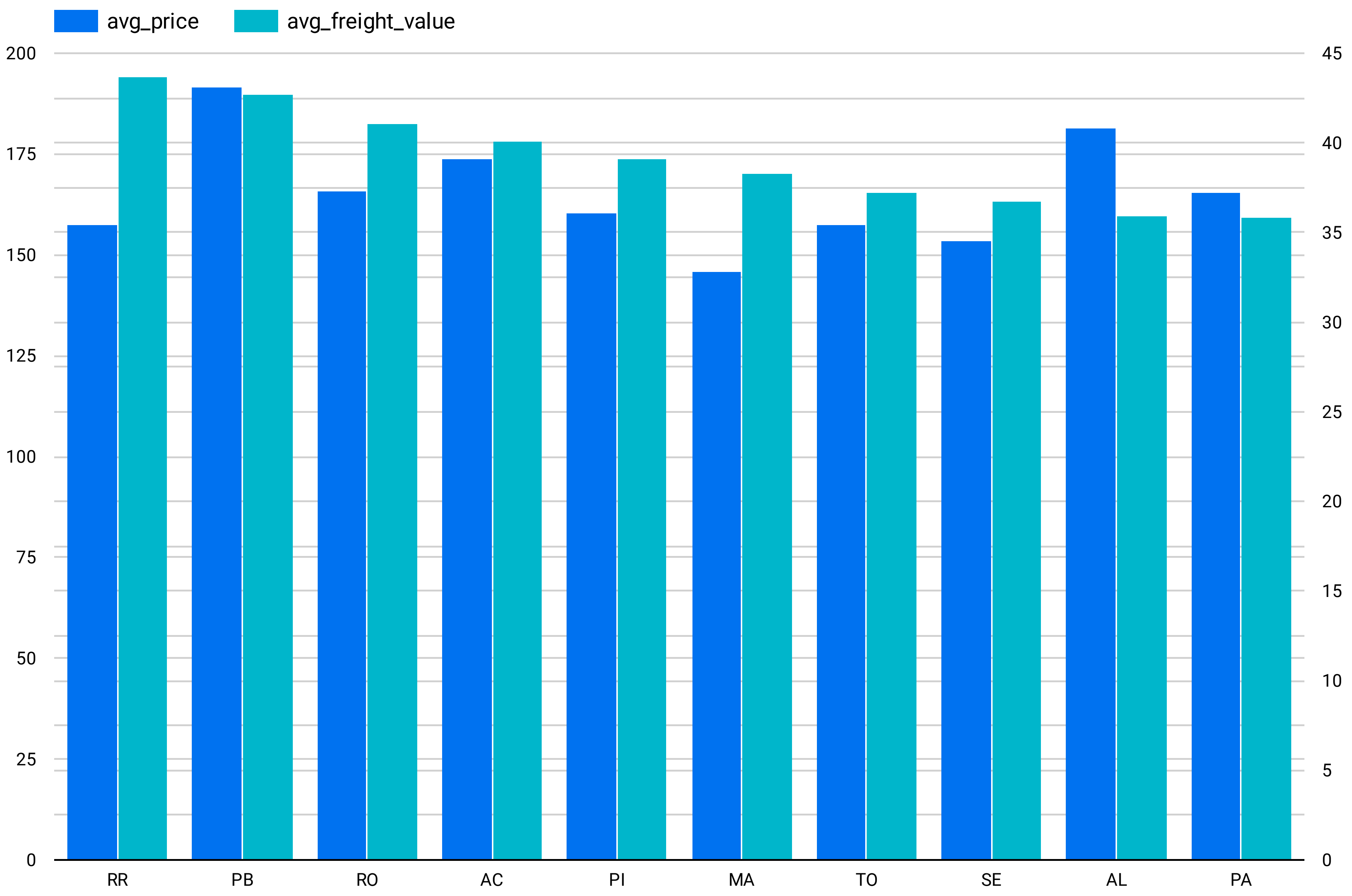
Mean and sum of price and freight value by customer state

```
SELECT
  customer_state,
  ROUND(SUM(price),2) AS total_price,
  ROUND(AVG(price),2) AS avg_price,
  ROUND(SUM(freight_value),2) AS total_freight_value,
  ROUND(AVG(freight_value),2) AS avg_freight_value
FROM
  `dsml-scaler-sql-365405.Target.customers`
LEFT JOIN
  `dsml-scaler-sql-365405.Target.orders`
USING
  (customer_id)
LEFT JOIN
  `dsml-scaler-sql-365405.Target.order-items`
USING
  (order_id)
WHERE
  FORMAT_DATE("%Y_%m", order_purchase_timestamp) BETWEEN '2017_01'
  AND '2018_08'
GROUP BY
  customer_state
```

	customer_state	total_price	avg_price	total_freight_value	avg_freight_value
1.	RS	746,162.4	120.17	134,951.39	21.73
2.	SP	5,188,099.23	109.63	716,782.63	15.15
3.	MG	1,580,496.82	120.82	270,044.12	20.64
4.	BA	510,455.94	134.51	100,055.39	26.37
5.	GO	293,607.56	126.28	52,944.7	22.77
6.	MA	118,943.96	145.94	31,229.18	38.32
7.	PE	261,418.93	145.31	59,130.27	32.87
8.	PB	115,218.18	191.71	25,694.89	42.75
9.	ES	274,119.52	121.72	49,615.25	22.03
10.	PR	681,068.25	119.28	117,372.47	20.56
11.	RO	46,140.64	165.97	11,417.38	41.07
12.	PA	177,860.21	165.61	38,503.81	35.85
13.	TO	49,621.74	157.53	11,732.68	37.25
14.	MT	156,125.74	148.41	29,540.59	28.08
15.	PI	86,704.08	160.27	21,182.11	39.15
16.	AL	80,232.32	181.52	15,867.18	35.9
17.	AM	22,356.84	135.5	5,478.89	33.21
18.	DF	301,560.17	125.75	50,469.16	21.05
19.	SE	58,635.4	153.5	14,051	36.78
20.	RR	7,716.84	157.49	2,142.53	43.73
21.	AP	13,474.3	164.32	2,788.5	34.01
22.	AC	15,982.95	173.73	3,686.75	40.07
23.	MS	116,812.64	142.63	19,144.03	23.37
24.	RJ	1,812,846.22	124.82	304,488.16	20.96
25.	SC	518,180.28	124.5	89,379.09	21.48



State Sao Paulo out numbered other state in terms of total price value and in terms of total freight value



Average value of total price and total freight value is almost same in all state across whole country

Percentage increase in total sales and average sale price from January 2017 to August 2018

```
SELECT
  *,
  ROUND((x.total_revenue - (LAG(x.total_revenue) OVER(ORDER BY x.year_month))),2) AS
revenue_growth,
  ROUND((x.total_revenue - (LAG(x.total_revenue) OVER(ORDER BY
x.year_month)))/((LAG(x.total_revenue) OVER(ORDER BY x.year_month))),2) AS
percent_revenue_growth,
  ROUND((x.avg_revenue - (LAG(x.avg_revenue) OVER(ORDER BY x.year_month))),2) AS
avg_revenue_growth,
  ROUND((x.avg_revenue - (LAG(x.avg_revenue) OVER(ORDER BY
x.year_month)))/((LAG(x.avg_revenue) OVER(ORDER BY x.year_month))),2) AS
percent_avg_revenue_growth
FROM (
  SELECT
    FORMAT_DATE("%Y_%m", order_purchase_timestamp) AS year_month,
    ROUND(SUM(payment_value),2) AS total_revenue,
    ROUND(AVG(payment_value),2) AS avg_revenue
  FROM
    `dsml-scaler-sql-365405.Target.orders`
  LEFT JOIN
    `dsml-scaler-sql-365405.Target.payments`
  USING
    (order_id)
  WHERE
    FORMAT_DATE("%Y_%m", order_purchase_timestamp) BETWEEN '2017_01'
    AND '2018_08'
  GROUP BY
    year_month
  ORDER BY
    year_month ) x
ORDER BY
  year_month
```

	year_month	total_revenue	revenue_growth	percent_revenue_grow...	avg_revenue	avg_revenue_growth	percent_avg_revenue_growth
1.	2017_01	138,488.04	null	null	162.93	null	null
2.	2017_02	291,908.01	153,419.97	1.11	154.78	-8.15	-0.05
3.	2017_03	449,863.6	157,955.59	0.54	158.57	3.79	0.02
4.	2017_04	417,788.03	-32,075.57	-0.07	162.5	3.93	0.02
5.	2017_05	592,918.82	175,130.79	0.42	150.33	-12.17	-0.07
6.	2017_06	511,276.38	-81,642.44	-0.14	148.8	-1.53	-0.01
7.	2017_07	592,382.92	81,106.54	0.16	137.22	-11.58	-0.08
8.	2017_08	674,396.32	82,013.4	0.14	148.22	11	0.08
9.	2017_09	727,762.45	53,366.13	0.08	161.15	12.93	0.09
10.	2017_10	779,677.88	51,915.43	0.07	160.43	-0.72	0
11.	2017_11	1,194,882.8	415,204.92	0.53	151.96	-8.47	-0.05
12.	2017_12	878,401.48	-316,481.32	-0.26	149.01	-2.95	-0.02
13.	2018_01	1,115,004.18	236,602.7	0.27	147.43	-1.58	-0.01
14.	2018_02	992,463.34	-122,540.84	-0.11	142.76	-4.67	-0.03
15.	2018_03	1,159,652.12	167,188.78	0.17	154.37	11.61	0.08
16.	2018_04	1,160,785.48	1,133.36	0	161.02	6.65	0.04
17.	2018_05	1,153,982.15	-6,803.33	-0.01	161.74	0.72	0
18.	2018_06	1,023,880.5	-130,101.65	-0.11	159.51	-2.23	-0.01
19.	2018_07	1,066,540.75	42,660.25	0.04	163.91	4.4	0.03
20.	2018_08	1,022,425.32	-44,115.43	-0.04	152.65	-11.26	-0.07

1 - 20 / 20