About Target

- 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.

Data is available in 8 csv files:

1. customers.csv
2. geolocation.csv
3. order_items.csv
4. payments.csv
5. reviews.csv
6. orders.csv
7. products.csv
8. sellers.csv

Each feature or columns of different CSV files are described below:

The customers.csv contain following features:

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

The sellers.csv contains following features:

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

The order_items.csv contain following features:

- order_id: A unique id of order made by the consumers.
- order_item_id: A Unique id given to each item ordered in the order.
- product_id: A unique id given to each product available on the site.
- seller_id: Unique Id of the seller registered in Target.
- shipping_limit_date: The date before which shipping of the ordered product must be completed.
- price: Actual price of the products ordered .
- freight_value: Price rate at which a product is delivered from one point to another.

The geolocations.csv contain following features:

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

The payments.csv contain following features:

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

The orders.csv contain following features:

- order_id: A unique id of order made by the consumers.
- customer_id :Id of the consumer who made the purchase.

- order_status: status of the order made i.e delivered, shipped etc.
- order_purchase_timestamp: Timestamp of the purchase.
- order_delivered_carrier_date :delivery date at which carrier made the delivery.
- order_delivered_customer_date: date at which customer got the product.
- order_estimated_delivery_date: estimated delivery date of the products.

The reviews.csv contain following features:

- review_id: Id of the review given on the product ordered by the order id.
- order_id: A unique id of order made by the consumers.
- review_score :review score given by the customer for each order on the scale of 1–5.
- review_comment_title: Title of the review
- review_comment_message: Review comments posted by the consumer for each order.
- review_creation_date :Timestamp of the review when it is created.
- review_answer_timestamp: Timestamp of the review answered.

The products.csv contain following features:

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

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import figure
import warnings
warnings.filterwarnings('ignore')
import statsmodels.api as sm
from scipy.stats import norm
from scipy.stats import t
```

In [2]:
```
# !pip install google-cloud
# !pip install google-cloud-bigquery[pandas]
# !pip install google-cloud-storage
```

In [3]:
```
# pip install --upgrade google-cloud-bigquery[bqstorage,pandas]
```

In [4]:
```
pd.set_option('display.max_columns', 500)
```

In [5]:
```
pd.set_option('display.max_rows', 500)
```

In [6]:
```
#Set environment variables for your notebook
import os
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "C:\\Users\\sunny\\Data Science Studies\\DSML projects\\Target - S
```

In [144]:
```
#Imports google cloud client library and initiates BQ service
from google.cloud import bigquery
bigquery_client = bigquery.Client()
```

In [8]:
```
%load_ext google.cloud.bigquery
```

In [9]:
```
# fetching  informations like state full name and region
# about the brazil per  region and states :
```

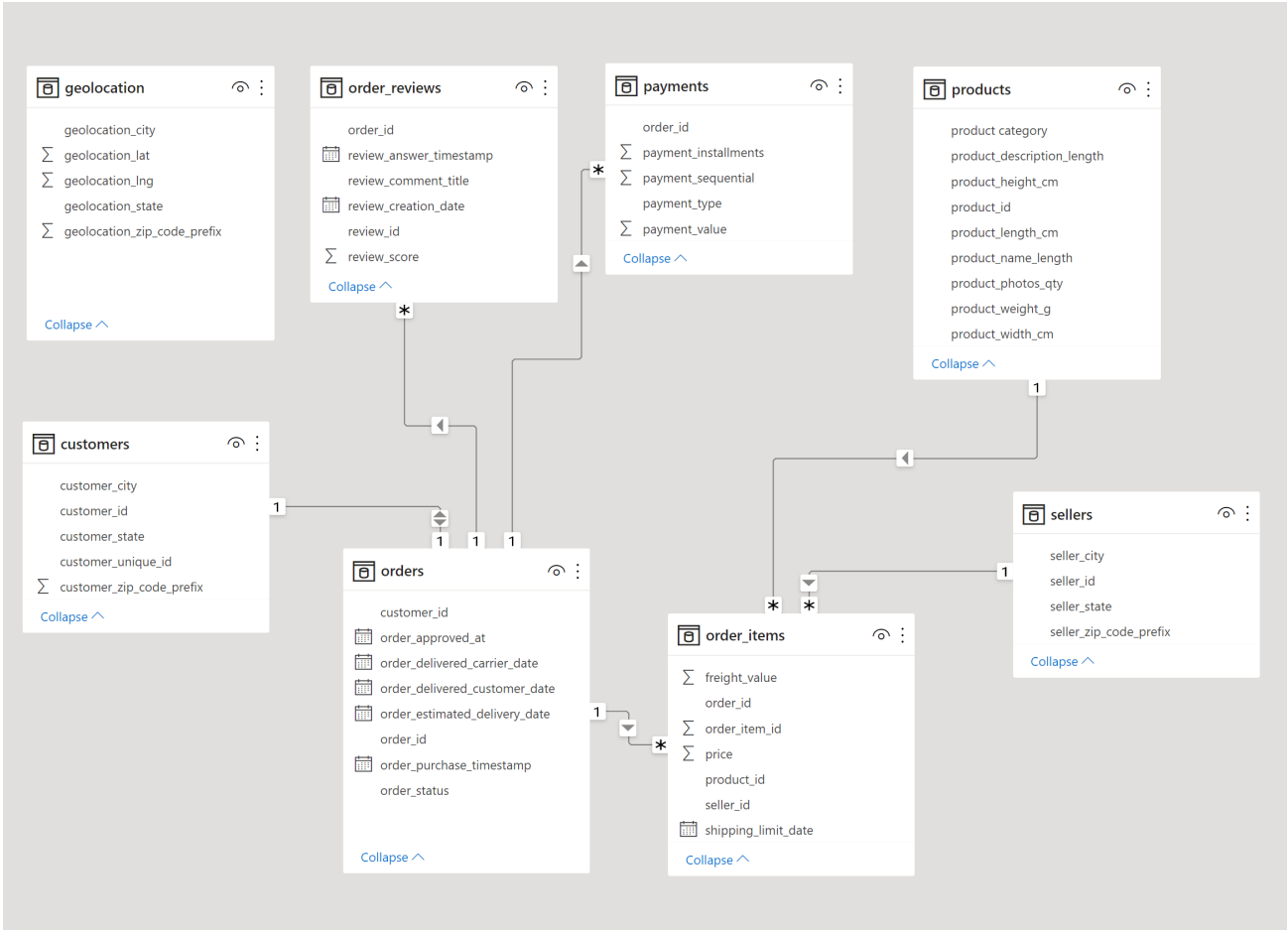
In [10]:
```
brazil = pd.read_html("https://brazil-help.com/brazilian_states.htm")[2]
new_header = brazil.iloc[1] #grab the first row for the header
brazil = brazil[1:] #take the data less the header row
brazil.columns = new_header #set the header row as the df header
brazil.drop(1,axis = 0,inplace=True)
brazil.reset_index(inplace=True)
brazil.drop(["index"],axis = 1,inplace=True)
brazil.columns = ['customer_state', 'State', 'Capitol City', 'Region',
       'Size (in km²)', 'Population (2007 estimate)', '% Pop. Urban/Rural',
       'Number of Municipal Districts', 'Per Capita GNP in Reais (R$)',
       'Life Expectancy (2007 projection)']
brazil.drop(["Size (in km²)","Life Expectancy (2007 projection)"
             ,"Number of Municipal Districts",
            "Population (2007 estimate)",
            "% Pop. Urban/Rural",
            "Per Capita GNP in Reais (R$)",
            "Capitol City"],axis = 1,inplace=True)
```

Problem Statement :

- **given this data to analyze and provide some insights and recommendations from it.**

In [ ]:
```
```

Schema / structure of data :


geolocation
geolocation_city
geolocation_lat
geolocation_lng
geolocation_state
geolocation_zip_code_prefix
Collapse
order_reviews
order_id
review_answer_timestamp
review_comment_title
review_creation_date
review_id
review_score
Collapse
payments
order_id
payment_installments
payment_sequential
payment_type
payment_value
Collapse
products
product category
product_description_length
product_height_cm
product_id
product_length_cm
product_name_length
product_photos_qty
product_weight_g
product_width_cm
Collapse
customers
customer_city
customer_id
customer_state
customer_unique_id
customer_zip_code_prefix
Collapse
orders
customer_id
order_approved_at
order_delivered_carrier_date
order_delivered_customer_date
order_estimated_delivery_date
order_id
order_purchase_timestamp
order_status
Collapse
order_items
freight_value
order_id
order_item_id
price
product_id
seller_id
shipping_limit_date
Collapse
sellers
seller_city
seller_id
seller_state
seller_zip_code_prefix
Collapse
1
*

In [ ]:

Starting with exploring all the tables and doing exploratory data analysis :

Analyzing Customers Table :

In [ ]:

In [ ]:

In [11]:
```
bigquery_client.query("""

SELECT
    * EXCEPT(is_generated, generation_expression, is_stored, is_updatable,is_hidden
    ,is_system_defined,is_partitioning_column,
    clustering_ordinal_position,collation_name,column_default,ordinal_position,table_catalog)
  FROM
    target-360705.target.INFORMATION_SCHEMA.COLUMNS
  WHERE
    table_name = 'customers';

""").to_dataframe()
```

Out[11]:

	table_schema	table_name	column_name	is_nullable	data_type
0	target	customers	customer_id	YES	STRING
1	target	customers	customer_unique_id	YES	STRING
2	target	customers	customer_zip_code_prefix	YES	INT64
3	target	customers	customer_city	YES	STRING
4	target	customers	customer_state	YES	STRING

Total number of Customer data we have :

In [12]:
```
bigquery_client.query(
"""

select count(distinct(customer_id)) from `target.customers` ;


""").to_dataframe()["f0_"][0]
```

Out[12]: 99441

In [13]:
```
# We have 99,441 customers of data available.
```

In [14]:
```
bigquery_client.query(
"""

select count(distinct(customer_unique_id)) from `target.customers` ;


""").to_dataframe()["f0_"][0]
```

Out[14]: 96096

In [15]:
```
# We have 96096 number of Unique Customers ids.
```

14994 unique zip-code-prefix

In [16]:
```
bigquery_client.query(
"""

select count(distinct(customer_zip_code_prefix)) from `target.customers` ;


""").to_dataframe()["f0_"][0]
```

Out[16]: 14994

In [17]:
```
bigquery_client.query(
"""

select count(distinct(customer_city)) from `target.customers` ;


""").to_dataframe()["f0_"][0]
```

Out[17]: 4119

In [18]:
```
bigquery_client.query(
"""

select count(distinct(customer_state)) from `target.customers` ;


""").to_dataframe()["f0_"][0]
```

Out[18]: 27

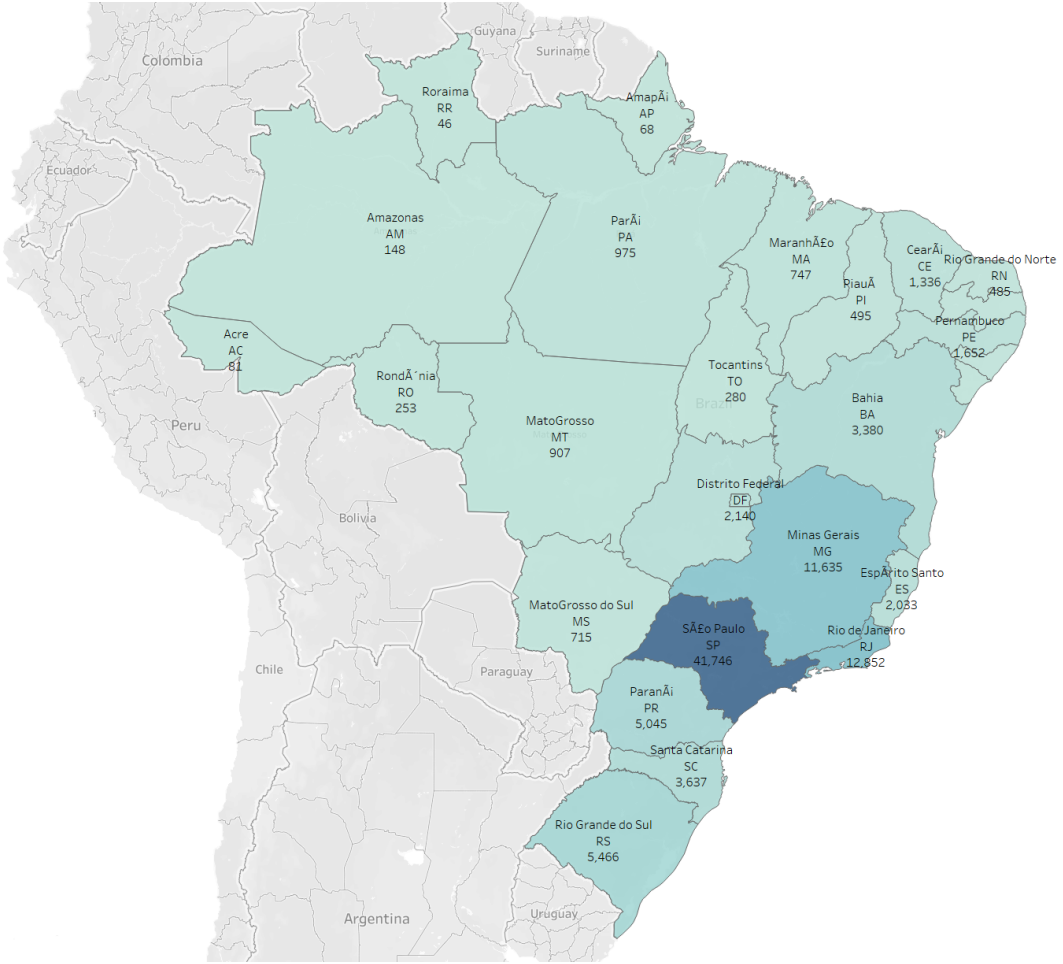
Customers are from different 4119 cities and 27 states from Brazil.

In [19]:
```
df = bigquery_client.query(
"""

select
customer_state,
COUNT(customer_id) as Number_of_customers

from
    `target.customers`
GROUP BY
    customer_state
ORDER BY
    Number_of_customers
;


""").to_dataframe()
df.merge(brazil,on="customer_state")
```

Out[19]:

	customer_state	Number_of_customers	State	Region
0	RR	46	Roraima	North
1	AP	68	Amapá	North
2	AC	81	Acre	North
3	AM	148	Amazonas	North
4	RO	253	Rondônia	North
5	TO	280	Tocantins	North
6	SE	350	Sergipe	Northeast
7	AL	413	Alagoas	Northeast
8	RN	485	Rio Grande do Norte	Northeast
9	PI	495	Piauí	Northeast
10	PB	536	Paraíba	Northeast

	customer_state	Number_of_customers	State	Region
11	MS	715	MatoGrosso do Sul	Center West
12	MA	747	Maranhão	Northeast
13	MT	907	MatoGrosso	Center West
14	PA	975	Pará	North
15	CE	1336	Ceará	Northeast
16	PE	1652	Pernambuco	Northeast
17	GO	2020	Goiás	Center West
18	ES	2033	Espírito Santo	Southeast
19	DF	2140	Distrito Federal	Center West
20	BA	3380	Bahia	Northeast
21	SC	3637	Santa Catarina	South
22	PR	5045	Paraná	South
23	RS	5466	Rio Grande do Sul	South
24	MG	11635	Minas Gerais	Southeast
25	RJ	12852	Rio de Janeiro	Southeast
26	SP	41746	São Paulo	Southeast


Guyana
Suriname
Colombia
Roraima
RR
46
AmapÃi
AP
68
Ecuador
Amazonas
AM
148
ParÃi
PA
975
MaranhÃ£o
MA
747
CearÃi
CE
1,336
Rio Grande do Norte
RN
485
PiauÃ
PI
495
Pernambuco
PE
1,652
Acre
AC
81
RondÂ´nia
RO
253
Tocantins
TO
280
Brazil
Bahia
BA
3,380
Peru
MatoGrosso
MT
907
Distrito Federal
DF
2,140
Minas Gerais
MG
11,635
Bolivia
EspÃrito Santo
ES
2,033
MatoGrosso do Sul
MS
715
SÃ£o Paulo
SP
41,746
Rio de Janeiro
RJ
12,852
Chile
Paraguay
ParanÃi
PR
5,045
Santa Catarina
SC
3,637
Rio Grande do Sul
RS
5,466
Argentina
Uruguay

In [20]:
```
((df.merge(brazil,on="customer_state").groupby("Region")["Number_of_customers"].sum())/99441)*100
```

Out[20]:
```
Region
Center West     5.814503
North           1.861405
Northeast       9.446808
South          14.227532
Southeast      68.649752
Name: Number_of_customers, dtype: Float64
```

total 99441 customers are there in given data !

68% customers are from southeast Brazil .

14% are from south Brazil.

rest are other other regions of Brazil .

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Analyzing Sellers Data :

In [21]:
```
bigquery_client.query("""

SELECT
    * EXCEPT(is_generated, generation_expression, is_stored, is_updatable,is_hidden
    ,is_system_defined,is_partitioning_column,
    clustering_ordinal_position,collation_name,column_default,ordinal_position,table_catalog)
  FROM
    target-360705.target.INFORMATION_SCHEMA.COLUMNS
  WHERE
    table_name = 'sellers';

""").to_dataframe()
```

Out[21]:

	table_schema	table_name	column_name	is_nullable	data_type
0	target	sellers	seller_id	YES	STRING
1	target	sellers	seller_zip_code_prefix	YES	INT64
2	target	sellers	seller_city	YES	STRING
3	target	sellers	seller_state	YES	STRING

In [ ]:

In [22]:
```
bigquery_client.query(
"""

SELECT
    COUNT(DISTINCT(seller_id))
FROM
    `target.sellers` ;


""").to_dataframe()["f0_"][0]
```

Out[22]: 3095

In [23]:
```
bigquery_client.query(
"""

SELECT
    COUNT(DISTINCT(seller_zip_code_prefix))
FROM
    `target.sellers` ;


""").to_dataframe()["f0_"][0]
```

Out[23]: 2246

In [24]:
```
bigquery_client.query(
"""

SELECT
    COUNT(DISTINCT(seller_city))
FROM
    `target.sellers` ;


""").to_dataframe()["f0_"][0]
```

Out[24]: 611

In [25]:
```
bigquery_client.query(
"""

SELECT
    COUNT(DISTINCT(seller_state))
FROM
    `target.sellers` ;


""").to_dataframe()["f0_"][0]
```

Out[25]: 23

In [26]:
```
# 3095 sellers data
# salers are from 611 cities and 23 states in Brazil .
# from 2246 different areas as per zip-code data .
```

In [ ]:
```

```

number of sellers per state

In [27]:
```
bigquery_client.query(
"""

select
    seller_state,
    string_field_2 as seller_state_,
    count(seller_id) as  number_of_sellers

from `target-360705.target.Brazil_data` as bd
join `target-360705.target.sellers` as s
on bd.string_field_1 = s.seller_state

group by seller_state,string_field_2
order by number_of_sellers desc
;

""").to_dataframe()
```

Out[27]:

	seller_state	seller_state_	number_of_sellers
0	SP	São Paulo	1849
1	PR	Paraná	349
2	MG	Minas Gerais	244
3	SC	Santa Catarina	190
4	RJ	Rio de Janeiro	171
5	RS	Rio Grande do Sul	129
6	GO	Goiás	40
7	DF	Distrito Federal	30
8	ES	Espírito Santo	23
9	BA	Bahia	19
10	CE	Ceará	13
11	PE	Pernambuco	9

	seller_state	seller_state_	number_of_sellers
12	PB	Paraíba	6
13	RN	Rio Grande do Norte	5
14	MS	MatoGrosso do Sul	5
15	MT	MatoGrosso	4
16	RO	Rondônia	2
17	SE	Sergipe	2
18	AC	Acre	1
19	AM	Amazonas	1
20	PA	Pará	1
21	MA	Maranhão	1
22	PI	Piauí	1

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Analysing products Tables :

In [28]:
```
bigquery_client.query("""

SELECT
    * EXCEPT(is_generated, generation_expression, is_stored, is_updatable,is_hidden
    ,is_system_defined,is_partitioning_column,
    clustering_ordinal_position,collation_name,column_default,ordinal_position,table_catalog)
  FROM
    target-360705.target.INFORMATION_SCHEMA.COLUMNS
  WHERE
    table_name = 'products';


""").to_dataframe()
```

Out[28]:

	table_schema	table_name	column_name	is_nullable	data_type
0	target	products	product_id	YES	STRING
1	target	products	product_category	YES	STRING
2	target	products	product_name_length	YES	INT64
3	target	products	product_description_length	YES	INT64
4	target	products	product_photos_qty	YES	INT64
5	target	products	product_weight_g	YES	INT64
6	target	products	product_length_cm	YES	INT64
7	target	products	product_height_cm	YES	INT64
8	target	products	product_width_cm	YES	INT64

In [29]:
```
# unique products avaliable in Target :
```

In [30]:
```
bigquery_client.query(
"""

SELECT
    COUNT((product_id))
FROM
    `target.products` ;


""").to_dataframe()["f0_"][0]
```

Out[30]: 32951

In [31]:
```
# product_category_name
bigquery_client.query(
"""

SELECT
    COUNT(DISTINCT(product_category))
FROM
    `target.products` ;


""").to_dataframe()["f0_"][0]
```

Out[31]: 73

In [32]:
```
# In products Data ,
# total 32951 different products available in Target
# in 73 different product_category .
```

In [33]:
```
# TOP 20 HIGHEST NUMBER OF PRODUCTS PER PRODUCT CATEGORIES
```

In [34]:
```
# product_category_name
bigquery_client.query(
"""

SELECT
    product_category,

    COUNT(DISTINCT(product_id)) AS Number_of_product_per_category
FROM
    `target.products`
GROUP BY product_category
ORDER BY
    COUNT(DISTINCT(product_id)) desc
;


""").to_dataframe()
```

Out[34]:

	product_category	Number_of_product_per_category
0	bed table bath	3029
1	sport leisure	2867
2	Furniture Decoration	2657
3	HEALTH BEAUTY	2444
4	housewares	2335
5	automotive	1900
6	computer accessories	1639
7	toys	1411
8	Watches present	1329
9	telephony	1134
10	babies	919

	product_category	Number_of_product_per_category
11	perfumery	868
12	stationary store	849
13	Fashion Bags and Accessories	849
14	Cool Stuff	789
15	Garden tools	753
16	pet Shop	719
17	None	610
18	electronics	517
19	Construction Tools Construction	400
20	home appliances	370
21	Bags Accessories	349
22	Games consoles	317
23	Furniture office	309
24	musical instruments	289
25	electrostile	231
26	Casa Construcao	225
27	General Interest Books	216
28	Fashion Calcados	173
29	Room Furniture	156
30	climatization	124
31	technical books	123
32	fixed telephony	116
33	House comfort	111
34	Drink foods	104
35	Market Place	104

	product_category	Number_of_product_per_category
36	Fashion Men's Clothing	95
37	Furniture Kitchen Service Area Dinner and Garden	94
38	SIGNALIZATION AND SAFETY	93
39	CONSTRUCTION SECURITY TOOLS	91
40	ELECTRICES 2	90
41	Construction Tools Garden	88
42	foods	82
43	drinks	81
44	Construction Tools Illumination	78
45	Agro Industria e Comercio	74
46	Industry Commerce and Business	68
47	Christmas articles	65
48	audio	58
49	Art	55
50	Fashion Underwear and Beach Fashion	53
51	Blu Ray DVDs	48
52	Furniture	45
53	Construction Tools Tools	39
54	HOUSE PASTALS OVEN AND CAFE	31
55	Imported books	31
56	PCs	30
57	cine photo	28
58	song	27
59	Fashion Women's Clothing	27
60	party articles	26

	product_category	Number_of_product_per_category
61	Fashion Sport	19
62	Arts and Crafts	19
63	flowers	14
64	Hygiene diapers	12
65	Kitchen portable and food coach	10
66	La Cuisine	10
67	CITTE AND UPHACK FURNITURE	10
68	IMAGE IMPORT TABLETS	9
69	Fashion Children's Clothing	5
70	House Comfort 2	5
71	PC Gamer	3
72	insurance and services	2
73	cds music dvds	1

In [ ]:

order_items

In [35]:
```
# structure of order_items table :
```

In [36]:
```
bigquery_client.query("""

SELECT
    * EXCEPT(is_generated, generation_expression, is_stored, is_updatable,is_hidden
    ,is_system_defined,is_partitioning_column,
    clustering_ordinal_position,collation_name,column_default,ordinal_position,table_catalog)
  FROM
    target-360705.target.INFORMATION_SCHEMA.COLUMNS
  WHERE
    table_name = 'order_items';

""").to_dataframe()
```

Out[36]:

	table_schema	table_name	column_name	is_nullable	data_type
0	target	order_items	order_id	YES	STRING
1	target	order_items	order_item_id	YES	INT64
2	target	order_items	product_id	YES	STRING
3	target	order_items	seller_id	YES	STRING
4	target	order_items	shipping_limit_date	YES	TIMESTAMP
5	target	order_items	price	YES	FLOAT64
6	target	order_items	freight_value	YES	FLOAT64

In [37]:
```
# restructuring order_items_tables
# and save it as a VIEW called order_items_SALES.
```

CREATED A VIEW HAVING ORDER QUANTITY AS PER ORDER_ITEM_ID FROM ORIGINAL ORDER_ITEM TABLEM

ADDED A COLUMN CALLED TOTAL_AMT_TO_PAY

WHICH IS

total_amt_to_pay = (price * quantity) + freight_value

```
CREATE VIEW `target.order_items_SALES`
AS


SELECT
*,
ROUND(((x.price + x.freight_value) * x.quantity),2) as Total_Amt_to_Pay
FROM
(
SELECT
    DISTINCT
    order_id,
    product_id,
    seller_id,
    shipping_limit_date,
    price,
    freight_value,
    COUNT(product_id) OVER (PARTITION BY order_id, product_id) AS quantity
FROM
    `target.order_items`

) as x
;
```

In [38]:
```
print(bigquery_client.query("""

SELECT
  table_name, view_definition, use_standard_sql
FROM
  target.INFORMATION_SCHEMA.VIEWS
WHERE
  table_name = 'order_items_SALES'

  ;

""").to_dataframe()["view_definition"][0])
```

```
SELECT
    *,
    ROUND(((x.price + x.freight_value) * x.quantity),2) as Total_Amt_to_Pay
    FROM
    (
    SELECT
        DISTINCT
        order_id,
        product_id,
        seller_id,
        shipping_limit_date,
        price,
        freight_value,
        COUNT(product_id) OVER (PARTITION BY order_id, product_id) AS quantity
    FROM
        `target.order_items`

    ) as x
```

In [39]:
```
# after restructuring the above table ,
# getting all infomation about tables presented in dataset :
```

In [40]:
```
bigquery_client.query("""

SELECT
    table_schema,table_name,table_type
  FROM
    target.INFORMATION_SCHEMA.TABLES
""").to_dataframe()
```

Out[40]:

	table_schema	table_name	table_type
0	target	order_items	BASE TABLE
1	target	Brazil_data	BASE TABLE
2	target	sellers	BASE TABLE
3	target	geolocation	BASE TABLE
4	target	products	BASE TABLE
5	target	orders	BASE TABLE
6	target	payments	BASE TABLE
7	target	order_items_SALES	VIEW
8	target	customers	BASE TABLE
9	target	order_reviews	BASE TABLE

In [ ]:

In [ ]:

Analyzing Orders table:

In [41]:
```
bigquery_client.query("""

SELECT
    * EXCEPT(is_generated, generation_expression, is_stored, is_updatable,is_hidden
    ,is_system_defined,is_partitioning_column,
    clustering_ordinal_position,collation_name,column_default,ordinal_position,table_catalog)
  FROM
    target-360705.target.INFORMATION_SCHEMA.COLUMNS
  WHERE
    table_name = 'orders';

""").to_dataframe()
```

Out[41]:

	table_schema	table_name	column_name	is_nullable	data_type
0	target	orders	order_id	YES	STRING
1	target	orders	customer_id	YES	STRING
2	target	orders	order_status	YES	STRING
3	target	orders	order_purchase_timestamp	YES	TIMESTAMP
4	target	orders	order_approved_at	YES	TIMESTAMP
5	target	orders	order_delivered_carrier_date	YES	TIMESTAMP
6	target	orders	order_delivered_customer_date	YES	TIMESTAMP
7	target	orders	order_estimated_delivery_date	YES	TIMESTAMP

Time period for which the data is given

In [42]:
```
bigquery_client.query(
"""

SELECT

DATE_DIFF(MAX(DATE(order_delivered_customer_date)),MIN(DATE(order_purchase_timestamp)),month) AS time_period_in_m

FROM
target.orders;
""").to_dataframe()
```

Out[42]:

	time_period_in_months
0	25

number of orders as per order_status presented in data:

In [43]:
```
bigquery_client.query(
"""

SELECT

    order_status,
    count(1) as Number_of_orders


FROM
    `target.orders`
GROUP BY
    order_status
;

""").to_dataframe()
```

Out[43]:

	order_status	Number_of_orders
0	created	5
1	shipped	1107
2	approved	2
3	canceled	625
4	invoiced	314
5	delivered	96478
6	processing	301
7	unavailable	609

In [ ]:
```

```

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?

revenue per year

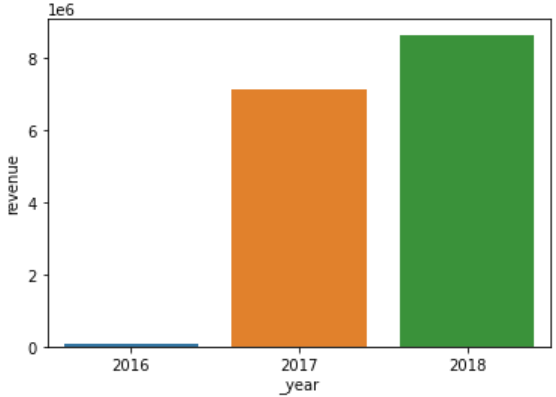
In [44]:
```
df = bigquery_client.query("""

SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) as _year,

    ROUND(SUM(Total_Amt_to_Pay),2) AS revenue ,
    count(o.order_id) as number_of_orders
from
    `target-360705.target.orders` as o
join
`target-360705.target.order_items_SALES` as ois

on o.order_id = ois.order_id

group by _year
ORDER BY _year
;

""").to_dataframe()
```

In [45]:
```
df
```

Out[45]:

	_year	revenue	number_of_orders
0	2016	57183.21	332
1	2017	7142672.43	46314
2	2018	8643697.60	55779

In [46]:
```
sns.barplot(x = df["_year"],y = df["revenue"])
plt.show()
```



In [47]:
```
((8643697.60 - 7142672.43)/7142672.43)*100
```

Out[47]:
```
21.014895821002952
```

In [48]:
```
## compare to 2017 , revenue has increased in 2018 by 21%.
```

In [ ]:
```

```

In [49]:

```
df =  bigquery_client.query("""

SELECT
x.month,
AVG(x.num_of_orders) AS avgerage_orders_per_month
from
(

SELECT
    extract(year from order_purchase_timestamp) as year,
    extract(month from order_purchase_timestamp) as month,
    count(order_id) as num_of_orders
from
    `target-360705.target.orders`
group by
    year,month
order by
    year,month
) as x
group by x.month
order by x.month
;
""").to_dataframe()
```

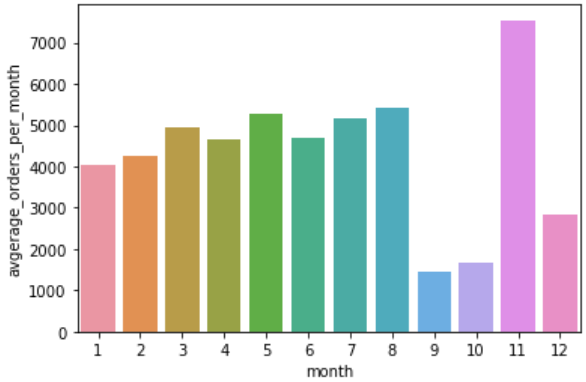
In [50]:
```
df
```

Out[50]:

	month	avgerage_orders_per_month
0	1	4034.5
1	2	4254.0
2	3	4946.5
3	4	4671.5
4	5	5286.5
5	6	4706.0
6	7	5159.0
7	8	5421.5
8	9	1435.0
9	10	1653.0
10	11	7544.0
11	12	2837.0

In [51]:
```
sns.barplot(x = df["month"],
            y = df["avgerage_orders_per_month"])
```

Out[51]:
```
<AxesSubplot:xlabel='month', ylabel='avgerage_orders_per_month'>
```



Average number of order are higher during November month.

september and october month average orders are comparatively low.

in may and july and august have higher average orders compare to other months.

In [ ]:

Number of orders per week trend :

In [52]:
```
df = bigquery_client.query("""

SELECT
    extract(year from order_purchase_timestamp) as year,
    extract(month from order_purchase_timestamp) as month,
    extract(week from order_purchase_timestamp) as week,
    count(order_id) as num_of_orders
from
    `target-360705.target.orders`
group by
    year,month,week
order by
    year,month,week

;
""").to_dataframe()
```

In [53]:
```
df
```

Out[53]:

	year	month	week	num_of_orders
0	2016	9	36	2
1	2016	9	37	2
2	2016	10	40	258
3	2016	10	41	65
4	2016	10	42	1
5	2016	12	51	1
6	2017	1	1	40
7	2017	1	2	72
8	2017	1	3	180
9	2017	1	4	350
10	2017	1	5	158
11	2017	2	5	269
12	2017	2	6	559
13	2017	2	7	435
14	2017	2	8	373
15	2017	2	9	144
16	2017	3	9	325
17	2017	3	10	592
18	2017	3	11	623
19	2017	3	12	646
20	2017	3	13	496
21	2017	4	13	68
22	2017	4	14	578

	year	month	week	num_of_orders
23	2017	4	15	467
24	2017	4	16	532
25	2017	4	17	691
26	2017	4	18	68
27	2017	5	18	677
28	2017	5	19	793
29	2017	5	20	904
30	2017	5	21	834
31	2017	5	22	492
32	2017	6	22	340
33	2017	6	23	834
34	2017	6	24	778
35	2017	6	25	646
36	2017	6	26	647
37	2017	7	26	80
38	2017	7	27	855
39	2017	7	28	939
40	2017	7	29	972
41	2017	7	30	915
42	2017	7	31	265
43	2017	8	31	727
44	2017	8	32	952
45	2017	8	33	1036
46	2017	8	34	875
47	2017	8	35	741

	year	month	week	num_of_orders
48	2017	9	35	262
49	2017	9	36	893
50	2017	9	37	1174
51	2017	9	38	1002
52	2017	9	39	954
53	2017	10	40	1001
54	2017	10	41	1075
55	2017	10	42	1110
56	2017	10	43	1004
57	2017	10	44	441
58	2017	11	44	489
59	2017	11	45	1187
60	2017	11	46	1329
61	2017	11	47	2775
62	2017	11	48	1764
63	2017	12	48	491
64	2017	12	49	1736
65	2017	12	50	1458
66	2017	12	51	1052
67	2017	12	52	862
68	2017	12	53	74
69	2018	1	0	1187
70	2018	1	1	1746
71	2018	1	2	1786
72	2018	1	3	1634

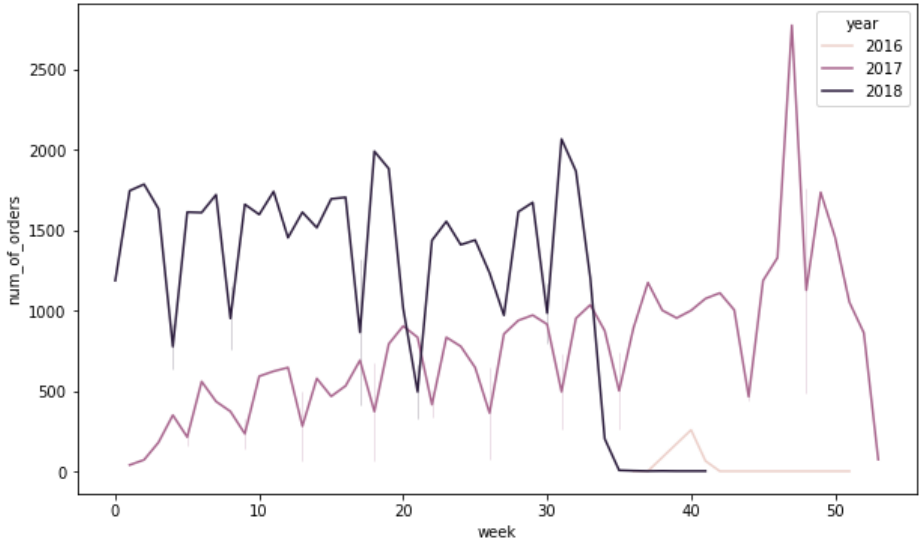
	year	month	week	num_of_orders
73	2018	1	4	916
74	2018	2	4	637
75	2018	2	5	1613
76	2018	2	6	1610
77	2018	2	7	1721
78	2018	2	8	1147
79	2018	3	8	757
80	2018	3	9	1661
81	2018	3	10	1598
82	2018	3	11	1741
83	2018	3	12	1454
84	2018	4	13	1612
85	2018	4	14	1516
86	2018	4	15	1695
87	2018	4	16	1705
88	2018	4	17	411
89	2018	5	17	1319
90	2018	5	18	1991
91	2018	5	19	1883
92	2018	5	20	1015
93	2018	5	21	665
94	2018	6	21	326
95	2018	6	22	1437
96	2018	6	23	1555
97	2018	6	24	1410

	year	month	week	num_of_orders
98	2018	6	25	1439
99	2018	7	26	1235
100	2018	7	27	971
101	2018	7	28	1615
102	2018	7	29	1673
103	2018	7	30	798
104	2018	8	30	1172
105	2018	8	31	2067
106	2018	8	32	1868
107	2018	8	33	1202
108	2018	8	34	203
109	2018	9	35	7
110	2018	9	36	4
111	2018	9	37	2
112	2018	9	38	3
113	2018	10	39	2
114	2018	10	41	2

In [54]:
```
df["year"] = df["year"].astype("object")
df["month"] = df["month"].astype("object")
```

In [55]:
```
plt.figure(figsize = (10,6))
sns.lineplot(data  = df,
             x = "week",
             y = "num_of_orders", hue = "year"
             )
```

Out[55]: <AxesSubplot:xlabel='week', ylabel='num_of_orders'>

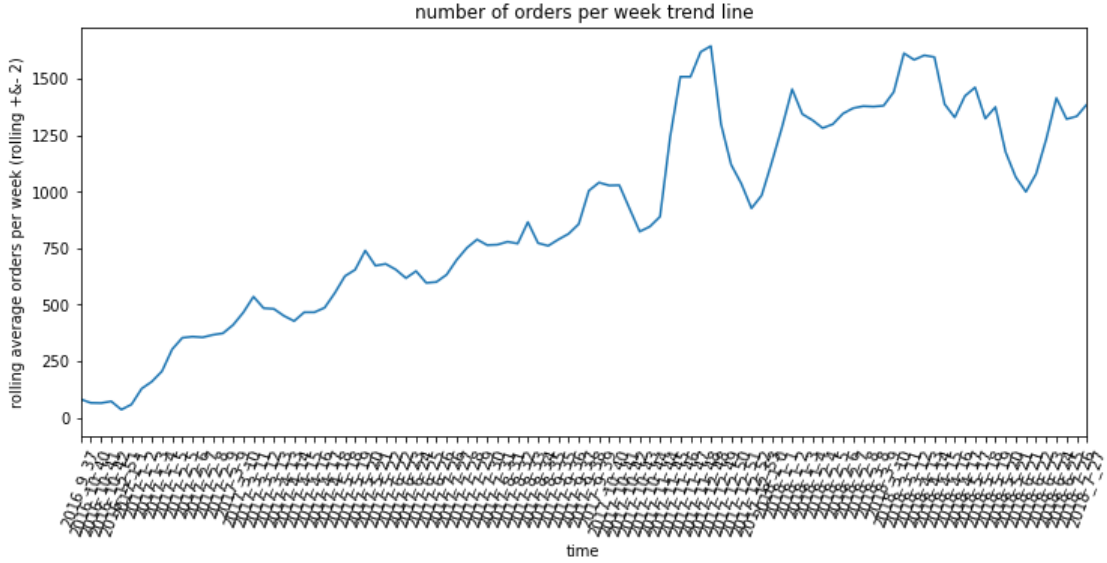


growing number of orders trend as per numbers of orders :

In [56]:
```
df = bigquery_client.query("""

select
    *,
    round(avg(num_of_orders) over (order by year,month,week
              rows between 2 preceding and 2 following),0) as rolling_avg_order
from
    (
    SELECT
        extract(year from order_purchase_timestamp) as year,
        extract(month from order_purchase_timestamp) as month,
        extract(week from order_purchase_timestamp) as week,
        count(order_id) as num_of_orders
    from
        `target-360705.target.orders`
    group by year,month,week
    order by year,month,week
    ) as x
;

""").to_dataframe()
```

In [57]:
```
df["time"] = df["year"].astype("str")+"_"+df["month"].astype("str")+"_"+df["week"].astype("str")
df.reset_index(inplace=True)
```

In [58]:
```
plt.figure(figsize=(12,5))
sns.lineplot(x = df["time"],
             y = df["rolling_avg_order"])
plt.title("number of orders per week trend line")
plt.xlabel("time")
plt.ylabel("rolling average orders per week (rolling +&- 2)")
plt.xlim(1,100)
plt.xticks(rotation = 70)
plt.show()
```



there is a increasing trend in orders , trend sustains during 2018.

there a slight fall we can observe during october 2017 following with a great hike in november month and again a fall in end of december 2017 and january 2018.

In [ ]:

In [ ]:

Revenue trend for given time duration :

In [59]:

```
df = bigquery_client.query("""

select
year,
month,
week,
revenue,
avg(revenue) over (order by year,month,week
                        rows between 4 preceding and 2 following) rolling_avg_revenue,
avg(Number_of_orders) over (order by year,month,week
                        rows between 4 preceding and 2 following) rolling_avg_orders

FROM
(
SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) as year,
    EXTRACT(MONTH FROM order_purchase_timestamp) as month,
    EXTRACT(week from order_purchase_timestamp) as week,
    ROUND(SUM(Total_Amt_to_Pay),2) AS revenue,
    count(o.order_id) as Number_of_orders
from
    `target-360705.target.orders` as o
join
`target-360705.target.order_items_SALES` as ois

on o.order_id = ois.order_id

group by year,month,week
ORDER BY year,month,week
)
;


""").to_dataframe()
```

In [60]:
```
df
```

Out[60]:

	year	month	week	revenue	rolling_avg_revenue	rolling_avg_orders
0	2016	9	36	211.29	16217.073333	88.333333
1	2016	9	37	143.46	14290.897500	82.750000
2	2016	10	40	48296.47	11436.642000	66.400000
3	2016	10	41	8512.37	10054.458333	62.000000
4	2016	12	51	19.62	10475.338571	63.428571
5	2017	1	1	3143.54	14263.398571	90.285714
6	2017	1	2	13000.62	23181.517143	141.571429
7	2017	1	3	26727.71	20817.211429	127.285714
8	2017	1	4	62570.29	26342.442857	155.285714
9	2017	1	5	31746.33	38273.141429	236.000000
10	2017	2	5	47188.99	48662.971429	293.142857
11	2017	2	6	83534.51	54488.782857	337.428571
12	2017	2	7	75872.35	54371.034286	330.571429
13	2017	2	8	53781.30	53439.197143	325.571429
14	2017	2	9	25903.47	62046.814286	388.571429
15	2017	3	9	56047.43	70893.768571	443.428571
16	2017	3	10	91999.65	73030.680000	455.142857
17	2017	3	11	109117.67	73104.765714	465.142857
18	2017	3	12	98492.89	68143.710000	420.142857
19	2017	3	13	76390.95	79678.978571	484.714286
20	2017	4	13	19053.91	82033.614286	506.428571
21	2017	4	14	106650.35	82512.700000	498.571429
22	2017	4	15	72529.88	82552.725714	507.714286

	year	month	week	revenue	rolling_avg_revenue	rolling_avg_orders
23	2017	4	16	95353.25	69830.455714	425.142857
24	2017	4	17	109397.85	72916.061429	450.714286
25	2017	4	18	9437.00	89170.545714	556.714286
26	2017	5	18	97990.19	93856.435714	606.571429
27	2017	5	19	132835.30	103898.867143	661.428571
28	2017	5	20	139451.58	100717.971429	654.714286
29	2017	5	21	142826.90	93037.098571	603.285714
30	2017	5	22	73086.98	109706.064286	715.000000
31	2017	6	22	55631.74	112320.878571	733.000000
32	2017	6	23	126119.76	108105.791429	712.000000
33	2017	6	24	116293.89	102696.702857	670.142857
34	2017	6	25	103329.69	84312.717143	559.428571
35	2017	6	26	101587.96	92275.995714	611.571429
36	2017	7	26	14139.00	102862.251429	698.714286
37	2017	7	27	128829.93	105528.102857	720.428571
38	2017	7	28	129735.53	107729.845714	738.857143
39	2017	7	29	144780.72	98079.940000	683.571429
40	2017	7	30	131706.09	98545.958571	697.714286
41	2017	7	31	35780.35	118237.118571	823.571429
42	2017	8	31	104850.09	121110.125714	851.285714
43	2017	8	32	151977.12	120399.197143	846.571429
44	2017	8	33	148940.98	119384.434286	811.857143
45	2017	8	34	124759.03	108130.701429	715.857143
46	2017	8	35	137677.38	126445.327143	807.571429
47	2017	9	35	52929.96	137755.424286	872.714286

	year	month	week	revenue	rolling_avg_revenue	rolling_avg_orders
48	2017	9	36	163982.73	138722.408571	881.857143
49	2017	9	37	184020.77	140405.045714	868.571429
50	2017	9	38	158746.01	146745.362857	882.857143
51	2017	9	39	160719.44	152375.398571	931.428571
52	2017	10	40	169141.25	171769.638571	1057.428571
53	2017	10	41	177087.63	171762.511429	1074.000000
54	2017	10	42	188689.64	155539.688571	967.000000
55	2017	10	43	163932.84	146358.957143	891.714286
56	2017	10	44	70461.01	149270.652857	927.285714
57	2017	11	44	94480.89	154976.518571	975.142857
58	2017	11	45	181101.31	193155.580000	1229.142857
59	2017	11	46	209082.31	201933.945714	1324.571429
60	2017	11	47	444341.06	190184.540000	1249.714286
61	2017	11	48	250138.20	217895.938571	1437.571429
62	2017	12	48	81687.00	236528.994286	1579.857143
63	2017	12	49	264440.80	234392.757143	1560.285714
64	2017	12	50	224912.28	221120.350000	1492.714286
65	2017	12	51	166147.65	159097.918571	1092.142857
66	2017	12	52	116175.46	148974.362857	1010.142857
67	2017	12	53	10184.04	176626.837143	1192.142857
68	2018	1	0	179273.31	177217.952857	1201.285714
69	2018	1	1	275254.32	180449.738571	1229.857143
70	2018	1	2	268578.61	176237.341429	1211.428571
71	2018	1	3	247534.78	173330.112857	1178.000000
72	2018	1	4	136660.87	206066.111429	1402.571429

	year	month	week	revenue	rolling_avg_revenue	rolling_avg_orders
73	2018	2	4	95824.86	212462.302857	1458.285714
74	2018	2	5	239336.03	207900.388571	1454.000000
75	2018	2	6	224046.65	195872.087143	1360.285714
76	2018	2	7	243320.92	176928.534286	1226.714286
77	2018	2	8	184380.50	195910.930000	1333.571429
78	2018	3	8	114929.91	218972.184286	1474.000000
79	2018	3	9	269537.64	224398.377143	1496.571429
80	2018	3	10	257253.64	226118.320000	1478.571429
81	2018	3	11	277319.38	230117.427143	1466.285714
82	2018	3	12	236086.25	241232.214286	1522.714286
83	2018	4	13	271314.67	264503.771429	1665.714286
84	2018	4	14	262184.01	266608.535714	1679.285714
85	2018	4	15	277830.81	239014.810000	1506.857143
86	2018	4	16	284270.99	229524.037143	1443.428571
87	2018	4	17	64097.56	244079.921429	1520.428571
88	2018	5	17	210883.97	250091.305714	1562.000000
89	2018	5	18	337977.44	235838.045714	1488.285714
90	2018	5	19	313394.36	214021.481429	1334.285714
91	2018	5	20	162411.19	182871.720000	1126.142857
92	2018	5	21	125114.86	208569.547143	1278.714286
93	2018	6	21	66222.66	216894.214286	1315.714286
94	2018	6	22	243982.35	199948.158571	1231.142857
95	2018	6	23	269156.64	187171.880000	1164.428571
96	2018	6	24	219355.05	192377.841429	1195.142857
97	2018	6	25	223960.41	199731.494286	1240.000000

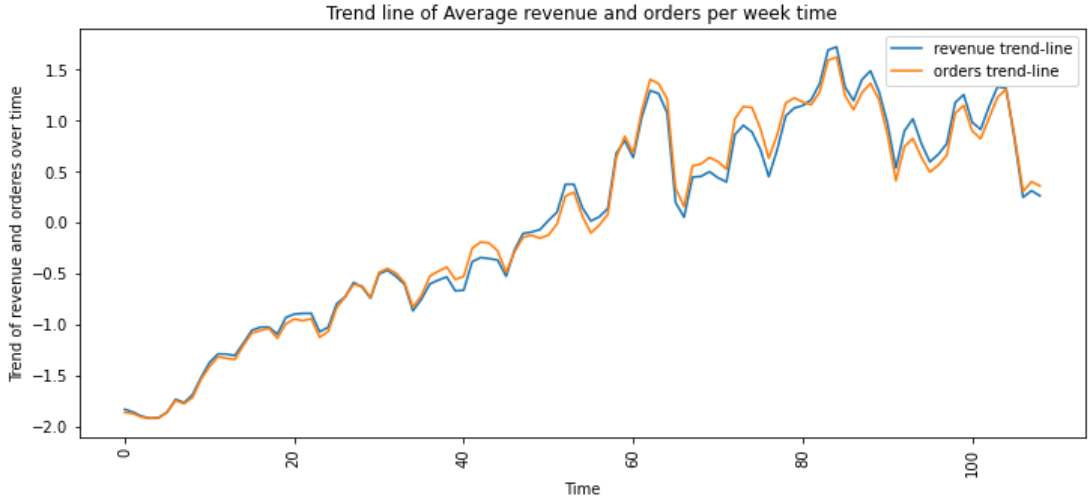
	year	month	week	revenue	rolling_avg_revenue	rolling_avg_orders
98	2018	7	26	198852.92	228197.580000	1428.000000
99	2018	7	27	176590.43	233645.084286	1463.571429
100	2018	7	28	265485.26	214577.641429	1348.857143
101	2018	7	29	282114.88	209641.501429	1314.000000
102	2018	7	30	135684.54	225457.621429	1407.000000
103	2018	8	30	184802.07	239193.072857	1500.428571
104	2018	8	31	334673.25	237894.922857	1535.428571
105	2018	8	32	295001.08	203015.412857	1327.571429
106	2018	8	33	167503.38	162737.067143	1078.000000
107	2018	8	34	21328.69	167245.821667	1121.500000
108	2018	9	35	166.46	163734.572000	1102.800000

In [61]:
```
df["time"] = df["year"].astype("str")+"_"+df["month"].astype("str")+"_"+df["week"].astype("str")
df.reset_index(inplace=True)
```

In [62]:
```
df["standardized_rolling_avg_revenue"]=(df["rolling_avg_revenue"]-df["rolling_avg_revenue"].mean())/df["rolling_a
```

In [63]:
```
df["standardized_rolling_avg_orders"]=(df["rolling_avg_orders"]-df["rolling_avg_orders"].mean())/df["rolling_avg_
```

In [64]:
```
plt.figure(figsize=(12,5))
sns.lineplot(x = df["index"],
             y = df["standardized_rolling_avg_revenue"])
sns.lineplot(x = df["index"],
             y = df["standardized_rolling_avg_orders"])
plt.xticks(rotation = 90)
plt.xlabel("Time")
plt.ylabel("Trend of revenue and orderes over time")
plt.title("Trend line of Average revenue and orders per week time")
plt.legend(["revenue trend-line","orders trend-line"])

plt.show()
```



from above we can observe the trend of increasing orders with time and also for revenue.

In [ ]:

In [ ]:

Month wise growth rate from 2017 to 2018.

In [65]:
```
bigquery_client.query("""

select
*,
((revenue_2018 - revenue_2017)/revenue_2017)*100 as revenue_Growth_rate,
((orders_2018 - orders_2017)/orders_2017)*100 as orders_Growth_rate

from
(
WITH
sale_2017 AS
    (
    SELECT

        extract(year from o.order_purchase_timestamp) as year,
        extract(month from o.order_purchase_timestamp) as month,
        count(distinct(ois.order_id)) as number_of_orders,
        sum(ois.Total_Amt_to_Pay) as revenue

    FROM
        target.orders as o
        join target.order_items_SALES as ois
        on ois.order_id = o.order_id
        WHERE extract(year from o.order_purchase_timestamp) = 2017 and
        extract(month from o.order_purchase_timestamp) between 1 and 8
        group by year,month
        order by number_of_orders,revenue
    ),
sale_2018 AS
    (
    SELECT

        extract(year from o.order_purchase_timestamp) as year,
        extract(month from o.order_purchase_timestamp) as month,
        count(distinct(ois.order_id)) as number_of_orders,
        sum(ois.Total_Amt_to_Pay) as revenue

    FROM
        target.orders as o
```

```
        join target.order_items_SALES as ois
        on ois.order_id = o.order_id
        WHERE extract(year from o.order_purchase_timestamp) = 2018
        and
        extract(month from o.order_purchase_timestamp) between 1 and 8
        group by year,month
        order by number_of_orders,revenue
    )


SELECT
s17.month,
s17.number_of_orders as orders_2017,
s18.number_of_orders as orders_2018,
s17.revenue as revenue_2017,
s18.revenue as revenue_2018



FROM
sale_2017 AS s17
left join
sale_2018 AS s18
on
s17.month = s18.month
order by
s17.month
)

;

""").to_dataframe()
```

Out[65]:

	month	orders_2017	orders_2018	revenue_2017	revenue_2018	revenue_Growth_rate	orders_Growth_rate
0	1	789	7220	137188.49	1107301.89	707.139061	815.082383
1	2	1733	6694	286280.62	986908.96	244.734813	286.266590

	month	orders_2017	orders_2018	revenue_2017	revenue_2018	revenue_Growth_rate	orders_Growth_rate
2	3	2641	7188	432048.59	1155126.82	167.360396	172.169633
3	4	2391	6934	412422.24	1159698.04	181.191926	190.004182
4	5	3660	6853	586190.95	1149781.82	96.144587	87.240437
5	6	3217	6160	502963.04	1022677.11	103.330469	91.482748
6	7	3969	6273	584971.62	1058728.03	80.987931	58.049887
7	8	4293	6452	668204.60	1003308.47	50.149890	50.291172

from above query we can observe there's 815% growth increased in terms of orders and 707% growth increment in terms of revenue in January from 2017 to 2018.

growth rate for july and august in 2017 to 2018 is relatively very low!

In [ ]:

In [ ]:

In [ ]:

% increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)

In [66]:
```
bigquery_client.query("""

WITH

sale_2017 AS
    (
    SELECT

        EXTRACT(MONTH FROM o.order_approved_at) as month,
        ROUND(SUM(ois.Total_Amt_to_Pay),2) AS revenue
    from
        `target-360705.target.orders` as o
    join
    `target-360705.target.order_items_SALES` as ois

    on o.order_id = ois.order_id
    WHERE EXTRACT(YEAR FROM o.order_approved_at) = 2017
    and
     (EXTRACT (MONTH FROM o.order_approved_at) BETWEEN 1 and 8)
    group by month
    ORDER BY month
    )

,

sale_2018 AS
    (
    SELECT

        EXTRACT(MONTH FROM o.order_approved_at) as month,
        ROUND(SUM(ois.Total_Amt_to_Pay),2) AS revenue
    from
        `target-360705.target.orders` as o
    join
    `target-360705.target.order_items_SALES` as ois

    on o.order_id = ois.order_id
    WHERE EXTRACT(YEAR FROM o.order_approved_at) = 2018
    and
```

```
  (EXTRACT (MONTH FROM o.order_approved_at) BETWEEN 1 and 8)
    group by month
    ORDER BY month
    )

SELECT
  s18.month,
  round(((s18.revenue - s17.revenue )/(s17.revenue))*100,2) as _difference

FROM
sale_2017 as s17
JOIN sale_2018 as s18 ON
    s17.month = s18.month

order by s18.month




;


""").to_dataframe()
```

Out[66]:

	month	_difference
0	1	739.65
1	2	242.21
2	3	172.18
3	4	177.49
4	5	100.29
5	6	102.27
6	7	79.25

	month	_difference
7	8	54.62

In [ ]:

Month on month growth :

In [ ]:

In [67]:
```
df = bigquery_client.query("""

select
*,
(( revenue -lag_rev )/lag_rev)*100 as growth_increase_compare_to_prev_month
from
(
    select
    *,
    lag(revenue,1) over (order by year,month ) as lag_rev
    from
    (
        SELECT

            extract(year from o.order_purchase_timestamp) as year,
            extract(month from o.order_purchase_timestamp) as month,
            count(distinct(ois.order_id)) as number_of_orders,
            sum(ois.Total_Amt_to_Pay) as revenue

        FROM
            target.orders as o
            join target.order_items_SALES as ois
            on ois.order_id = o.order_id

        group by year,month
        order by number_of_orders,revenue
    )
    order by year,month
)

;

""").to_dataframe()
```

In [68]:
```
df = df[(df["year"]==2017) | (df["year"]==2018)]
```

In [69]:
```
df = df[1:]
df["mont_year"] = df["month"].astype("str")+"_"+df["year"].astype("str")
```

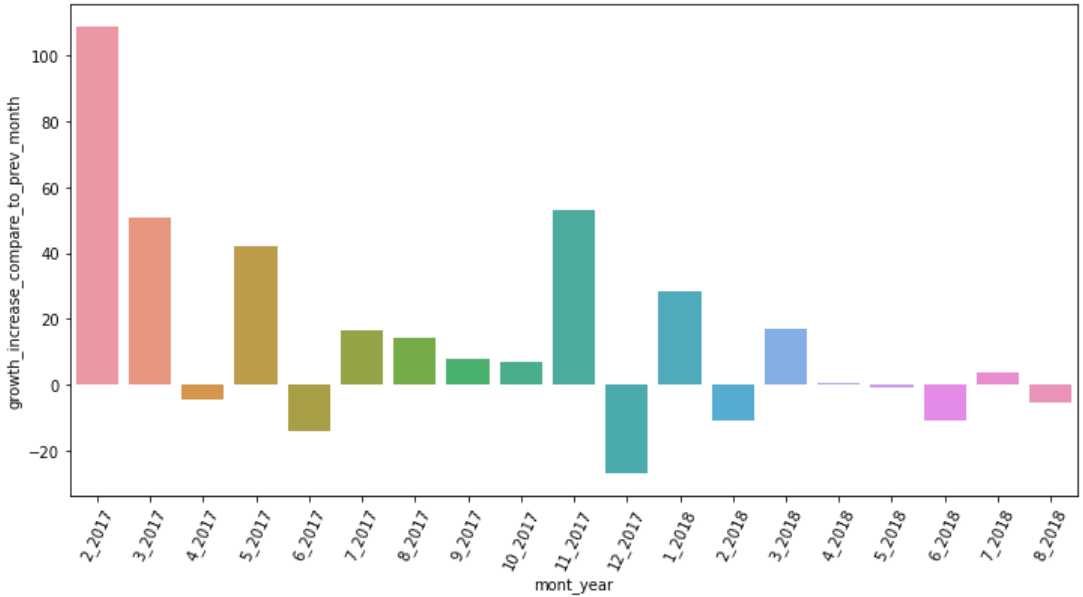
In [70]:
```
df = df[:-1]
```

In [71]:
```
df
```

Out[71]:

	year	month	number_of_orders	revenue	lag_rev	growth_increase_compare_to_prev_month	mont_year
4	2017	2	1733	286280.62	137188.49	108.676850	2_2017
5	2017	3	2641	432048.59	286280.62	50.917862	3_2017
6	2017	4	2391	412422.24	432048.59	-4.542626	4_2017
7	2017	5	3660	586190.95	412422.24	42.133690	5_2017
8	2017	6	3217	502963.04	586190.95	-14.198089	6_2017
9	2017	7	3969	584971.62	502963.04	16.305091	7_2017
10	2017	8	4293	668204.60	584971.62	14.228550	8_2017
11	2017	9	4243	720398.91	668204.60	7.811127	9_2017
12	2017	10	4568	769312.37	720398.91	6.789774	10_2017
13	2017	11	7451	1179143.77	769312.37	53.272431	11_2017
14	2017	12	5624	863547.23	1179143.77	-26.764891	12_2017
15	2018	1	7220	1107301.89	863547.23	28.227137	1_2018
16	2018	2	6694	986908.96	1107301.89	-10.872638	2_2018
17	2018	3	7188	1155126.82	986908.96	17.044922	3_2018
18	2018	4	6934	1159698.04	1155126.82	0.395733	4_2018
19	2018	5	6853	1149781.82	1159698.04	-0.855069	5_2018
20	2018	6	6160	1022677.11	1149781.82	-11.054681	6_2018
21	2018	7	6273	1058728.03	1022677.11	3.525152	7_2018
22	2018	8	6452	1003308.47	1058728.03	-5.234542	8_2018

In [72]:
```
plt.figure(figsize=(12,6))
sns.barplot(x = df["mont_year"],y = df["growth_increase_compare_to_prev_month"])
plt.xticks(rotation = 65)
plt.show()
```



2017-february, 2017-march,2017-november were the highest growing sale month compare to its previous month.

In [ ]:

In [73]:
```
df = bigquery_client.query("""

select

FORMAT_TIMESTAMP("%a", order_purchase_timestamp) as day,
count(order_id) as number_or_orders
from
target.orders
group by day


;

""").to_dataframe()
```

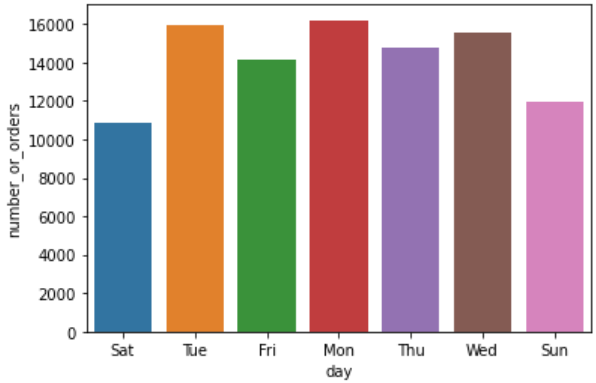
In [74]:
```
df
```

Out[74]:

	day	number_or_orders
0	Sat	10887
1	Tue	15963
2	Fri	14122
3	Mon	16196
4	Thu	14761
5	Wed	15552
6	Sun	11960

In [75]:
```
sns.barplot(x = df["day"],
            y = df["number_or_orders"])
```

Out[75]:
```
<AxesSubplot:xlabel='day', ylabel='number_or_orders'>
```



Tuesday, monday and wednesdays have relatively higher number of orders.

In [ ]:

In [ ]:

What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

In [76]:

```
orders_per_hour = bigquery_client.query("""

SELECT
    X.HOUR,
    COUNT(X.order_id) as number_of_orders
FROM
            (
            SELECT
            order_id,
            EXTRACT (HOUR FROM order_purchase_timestamp) AS HOUR
            FROM
            `target.orders`
            )
            AS X
GROUP BY
    X.HOUR

;

""").to_dataframe()
```

In [77]:
```
orders_per_hour
```

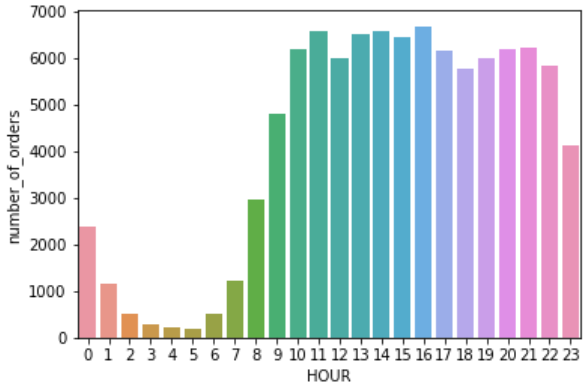
Out[77]:

	HOUR	number_of_orders
0	11	6578
1	1	1170
2	17	6150
3	13	6518
4	12	5995
5	18	5769
6	10	6177
7	21	6217
8	15	6454
9	22	5816
10	16	6675
11	20	6193
12	0	2394
13	23	4123
14	3	272
15	14	6569
16	9	4785
17	7	1231
18	19	5982
19	4	206
20	8	2967
21	2	510
22	6	502

	HOUR	number_of_orders
23	5	188

In [78]:
```
sns.barplot(x = orders_per_hour["HOUR"],
            y = orders_per_hour["number_of_orders"])
```

Out[78]: <AxesSubplot:xlabel='HOUR', ylabel='number_of_orders'>



In [79]:
```
bigquery_client.query("""

SELECT
    X.HOUR,
    COUNT(X.order_id) as number_of_orders,
    CASE
        WHEN X.HOUR BETWEEN 5 AND 7 THEN 'Dawn'
        WHEN X.HOUR BETWEEN 8 AND 11 THEN 'Morning'
        WHEN X.hour BETWEEN 12 AND 17 THEN 'Afternoon'
        WHEN X.hour BETWEEN 18 and 21 THEN 'Evening'
        ELSE 'night'
        END AS time
FROM
            (
            SELECT
            order_id,
            EXTRACT (HOUR FROM order_purchase_timestamp) AS HOUR
            FROM
            `target.orders`
            )
            AS X
GROUP BY
    X.HOUR

;

""").to_dataframe()
```

Out[79]:

	HOUR	number_of_orders	time
0	11	6578	Morning
1	1	1170	night
2	17	6150	Afternoon
3	13	6518	Afternoon
4	12	5995	Afternoon

	HOUR	number_of_orders	time
5	18	5769	Evening
6	10	6177	Morning
7	21	6217	Evening
8	15	6454	Afternoon
9	22	5816	night
10	16	6675	Afternoon
11	20	6193	Evening
12	0	2394	night
13	23	4123	night
14	3	272	night
15	14	6569	Afternoon
16	9	4785	Morning
17	7	1231	Dawn
18	19	5982	Evening
19	4	206	night
20	8	2967	Morning
21	2	510	night
22	6	502	Dawn
23	5	188	Dawn

In [80]:

```
df = bigquery_client.query("""
WITH order_time
AS
(
SELECT
    X.HOUR,
    COUNT(X.order_id) as number_of_orders,
    CASE
        WHEN X.HOUR BETWEEN 5 AND 7 THEN 'Dawn'
        WHEN X.HOUR BETWEEN 8 AND 11 THEN 'Morning'
        WHEN X.hour BETWEEN 12 AND 17 THEN 'Afternoon'
        WHEN X.hour BETWEEN 18 and 21 THEN 'Evening'
        ELSE 'night'
        END AS time
FROM
            (
            SELECT
            order_id,
            EXTRACT (HOUR FROM order_purchase_timestamp) AS HOUR
            FROM
            `target.orders`
            )
            AS X
GROUP BY
    X.HOUR
)

SELECT
    time,
    SUM(number_of_orders) AS total_order
FROM
    order_time
GROUP BY time
ORDER BY total_order DESC

;

""").to_dataframe()
```

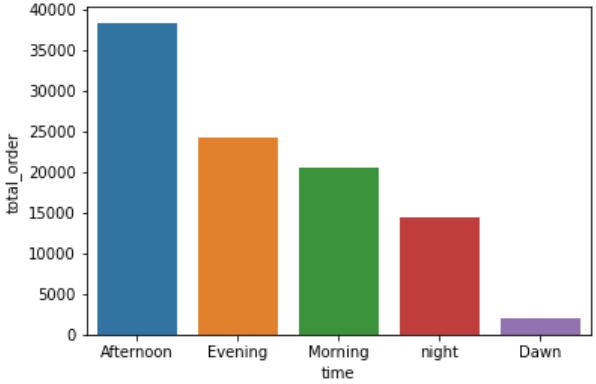
In [81]:
```
df
```

Out[81]:

	time	total_order
0	Afternoon	38361
1	Evening	24161
2	Morning	20507
3	night	14491
4	Dawn	1921

In [82]:
```
sns.barplot(x = df["time"],
            y = df["total_order"])
```

Out[82]:
```
<AxesSubplot:xlabel='time', ylabel='total_order'>
```



customers are purchasing during moring 8am to late evening 11pm.

afternoon orders are very high and evening , compare to morning , and night time.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

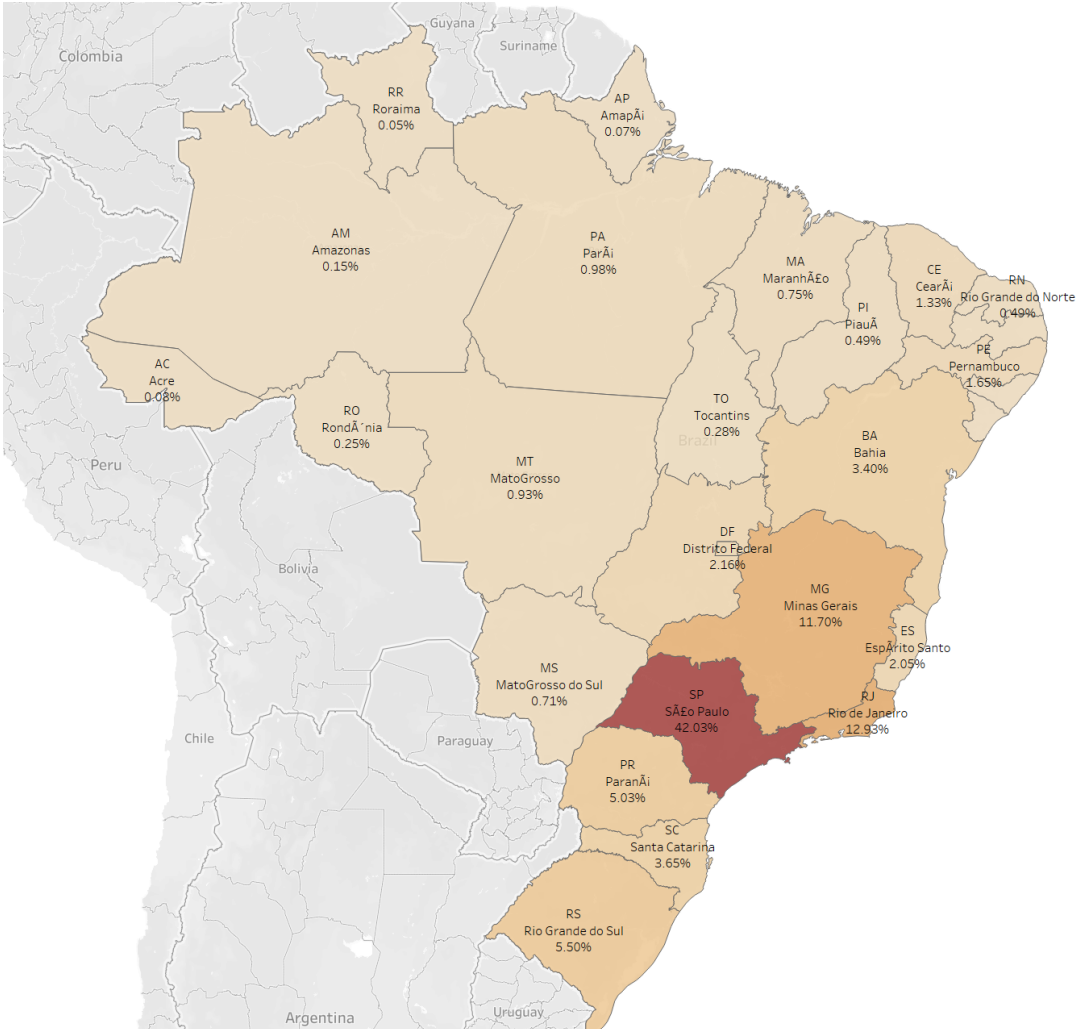
Number of orders per state

In [83]:
```
df = bigquery_client.query("""

select
    c.customer_state,
    count(o.order_id) as number_of_orders_per_state

from
    `target-360705.target.orders` as o
    join `target-360705.target.customers` as c
    on o.customer_id= c.customer_id
group by
    c.customer_state
order by number_of_orders_per_state desc

;
""").to_dataframe()
df = df.merge(brazil,on="customer_state")
df
```

Out[83]:

	customer_state	number_of_orders_per_state	State	Region
0	SP	41746	São Paulo	Southeast
1	RJ	12852	Rio de Janeiro	Southeast
2	MG	11635	Minas Gerais	Southeast
3	RS	5466	Rio Grande do Sul	South
4	PR	5045	Paraná	South
5	SC	3637	Santa Catarina	South
6	BA	3380	Bahia	Northeast
7	DF	2140	Distrito Federal	Center West
8	ES	2033	Espírito Santo	Southeast
9	GO	2020	Goiás	Center West
10	PE	1652	Pernambuco	Northeast

	customer_state	number_of_orders_per_state	State	Region
11	CE	1336	Ceará	Northeast
12	PA	975	Pará	North
13	MT	907	MatoGrosso	Center West
14	MA	747	Maranhão	Northeast
15	MS	715	MatoGrosso do Sul	Center West
16	PB	536	Paraíba	Northeast
17	PI	495	Piauí	Northeast
18	RN	485	Rio Grande do Norte	Northeast
19	AL	413	Alagoas	Northeast
20	SE	350	Sergipe	Northeast
21	TO	280	Tocantins	North
22	RO	253	Rondônia	North
23	AM	148	Amazonas	North
24	AC	81	Acre	North
25	AP	68	Amapá	North
26	RR	46	Roraima	North


Guyana
Suriname
Colombia
RR
Roraima
0.05%
AP
AmapÃi
0.07%
AM
Amazonas
0.15%
PA
ParÃi
0.98%
MA
MaranhÃ£o
0.75%
CE
CearÃi
1.33%
RN
Rio Grande do Norte
0.49%
PI
PiauÃ
0.49%
PE
Pernambuco
1.65%
AC
Acre
0.08%
RO
RondÃ´nia
0.25%
TO
Tocantins
0.28%
BA
Bahia
3.40%
Peru
MT
MatoGrosso
0.93%
DF
Distrito Federal
2.16%
Bolivia
MG
Minas Gerais
11.70%
ES
EspÃrito Santo
2.05%
MS
MatoGrosso do Sul
0.71%
SP
SÃ£o Paulo
42.03%
RJ
Rio de Janeiro
12.93%
Chile
Paraguay
PR
ParanÃi
5.03%
SC
Santa Catarina
3.65%
RS
Rio Grande do Sul
5.50%
Argentina
Uruguay

In [84]:
```
(df.groupby("Region")["number_of_orders_per_state"].sum()/df["number_of_orders_per_state"].sum())*100
```

Out[84]:
```
Region
Center West     5.814503
North           1.861405
Northeast       9.446808
South          14.227532
Southeast      68.649752
Name: number_of_orders_per_state, dtype: Float64
```

Revenue earning per states :

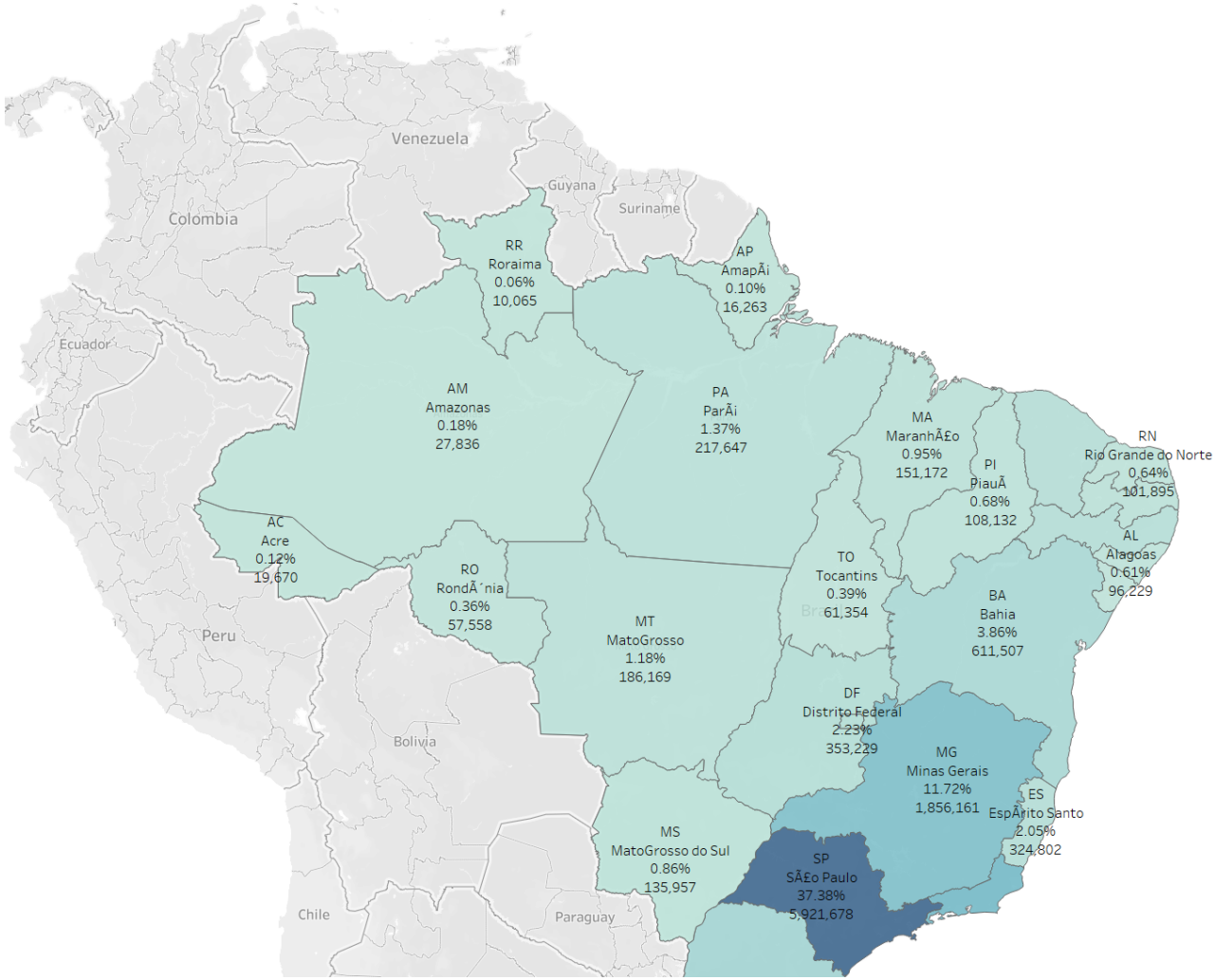
In [85]:

```
df = bigquery_client.query("""

select
    c.customer_state,
    sum(ois.Total_Amt_to_Pay) as revenue_per_state,
    avg(ois.Total_Amt_to_Pay) as avg_sale_per_state
from
    `target-360705.target.orders` as o
    join `target-360705.target.customers` as c
    on o.customer_id= c.customer_id
    join `target-360705.target.order_items_SALES` as ois
    on o.order_id = ois.order_id
group by
    c.customer_state
order by revenue_per_state desc

;
""").to_dataframe()
df = df.merge(brazil,on="customer_state")
df
```

Out[85]:

	customer_state	revenue_per_state	avg_sale_per_state	State	Region
0	SP	5921678.12	137.547109	São Paulo	Southeast
1	RJ	2129681.98	160.791391	Rio de Janeiro	Southeast
2	MG	1856161.49	154.860795	Minas Gerais	Southeast
3	RS	885826.76	157.145070	Rio Grande do Sul	South
4	PR	800935.44	155.310343	Paraná	South
5	BA	611506.67	175.720307	Bahia	Northeast
6	SC	610213.60	163.289698	Santa Catarina	South
7	DF	353229.44	159.760036	Distrito Federal	Center West
8	GO	347706.93	166.128490	Goiás	Center West

	customer_state	revenue_per_state	avg_sale_per_state	State	Region
9	ES	324801.91	154.815019	Espírito Santo	Southeast
10	PE	322237.69	190.899105	Pernambuco	Northeast
11	CE	275606.30	201.613972	Ceará	Northeast
12	PA	217647.11	216.996122	Pará	North
13	MT	186168.96	195.967326	MatoGrosso	Center West
14	MA	151171.99	196.838529	Maranhão	Northeast
15	PB	140987.81	257.747367	Paraíba	Northeast
16	MS	135956.67	185.733156	MatoGrosso do Sul	Center West
17	PI	108132.28	214.548175	Piauí	Northeast
18	RN	101895.08	204.198557	Rio Grande do Norte	Northeast
19	AL	96229.40	229.117619	Alagoas	Northeast
20	SE	73032.32	207.478182	Sergipe	Northeast
21	TO	61354.42	211.566966	Tocantins	North
22	RO	57558.02	223.093101	Rondônia	North
23	AM	27835.73	184.342583	Amazonas	North
24	AC	19669.70	239.874390	Acre	North
25	AP	16262.80	229.053521	Amapá	North
26	RR	10064.62	214.140851	Roraima	North


Venezuela
Guyana
Suriname
Colombia
Ecuador
Peru
Bolivia
Chile
Paraguay
RR
Roraima
0.06%
10,065
AP
AmapÃi
0.10%
16,263
AM
Amazonas
0.18%
27,836
PA
ParÃi
1.37%
217,647
MA
MaranhÃ£o
0.95%
151,172
PI
PiauÃ
0.68%
108,132
RN
Rio Grande do Norte
0.64%
101,895
AL
Alagoas
0.61%
96,229
AC
Acre
0.12%
19,670
RO
RondÂ´nia
0.36%
57,558
TO
Tocantins
0.39%
61,354
BA
Bahia
3.86%
611,507
MT
MatoGrosso
1.18%
186,169
DF
Distrito Federal
2.23%
353,229
MG
Minas Gerais
11.72%
1,856,161
ES
EspÃrito Santo
2.05%
324,802
MS
MatoGrosso do Sul
0.86%
135,957
SP
SÃ£o Paulo
37.38%
5,921,678

In [86]:
```
(df.groupby("Region")["revenue_per_state"].sum()/df["revenue_per_state"].sum())*100
```

Out[86]:
```
Region
Center West     6.457276
North           2.590280
Northeast      11.871072
South          14.497858
Southeast      64.583514
Name: revenue_per_state, dtype: float64
```

from above geo-map and query we can observe

São Paulo ,Rio de Janeiro , Minas Gerais ,Rio Grande do Sul and Paraná are top 5 highest orders states and also generating highest revenue.

more than 80% of orders are coming from south, southeast and nothest Brazil.

90% of the revenue is coming from south, southeast and nothest Brazil .

In [ ]:

In [ ]:

Top 20 cities where highest number of orders coming from:

In [87]:
```
bigquery_client.query("""

SELECT
    c.customer_city,
    count(distinct(o.order_id)) number_of_customers,
    sum(ois.Total_Amt_to_Pay) as revenue
FROM
    `target-360705.target.customers` as c
    left join  `target-360705.target.orders` as o
        on c.customer_id = o.customer_id
    left join  `target-360705.target.order_items_SALES` as ois
        on o.order_id = ois.order_id

group by
    c.customer_city
order by
    number_of_customers desc,
    revenue desc
limit 20
;
""").to_dataframe()
```

Out[87]:

	customer_city	number_of_customers	revenue
0	sao paulo	15540	2170227.12
1	rio de janeiro	6882	1154234.02
2	belo horizonte	2773	416733.39
3	brasilia	2131	352305.14
4	curitiba	1521	244739.87
5	campinas	1444	212541.70
6	porto alegre	1379	224064.09
7	salvador	1245	216772.40
8	guarulhos	1189	163575.82

	customer_city	number_of_customers	revenue
9	sao bernardo do campo	938	119024.85
10	niteroi	849	137919.38
11	santo andre	797	104721.52
12	osasco	746	94000.72
13	santos	713	111670.21
14	goiania	692	124060.53
15	sao jose dos campos	691	89987.31
16	fortaleza	654	118646.07
17	sorocaba	633	87148.84
18	recife	613	110023.52
19	florianopolis	570	99925.37

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Product and product category analysis :

top selling product categories and howmany orders were placed per category :

In [88]:
```
bigquery_client.query("""

SELECT
    product_category,
    count(distinct(ois.product_id)) as Number_of_products,
    count(distinct(ois.order_id)) as Number_of_orders,
    sum(Total_Amt_to_Pay) as revenue


FROM
    `target.order_items_SALES` as ois
    join `target.products`  as p
    on p.product_id = ois.product_id
GROUP BY
    p.product_category
ORDER BY
    revenue desc,Number_of_orders desc
LIMIT 20




;

""").to_dataframe()
```

Out[88]:

	product_category	Number_of_products	Number_of_orders	revenue
0	HEALTH BEAUTY	2444	8836	1441248.07
1	Watches present	1329	5624	1305541.61
2	bed table bath	3029	9417	1241681.72
3	sport leisure	2867	7720	1156656.48
4	computer accessories	1639	6689	1059272.40
5	Furniture Decoration	2657	6449	902511.79
6	housewares	2335	5884	778397.77

	product_category	Number_of_products	Number_of_orders	revenue
7	Cool Stuff	789	3632	719329.95
8	automotive	1900	3897	685384.32
9	Garden tools	753	3518	584219.21
10	toys	1411	3886	561372.55
11	babies	919	2885	480118.00
12	perfumery	868	3162	453338.71
13	telephony	1134	4199	394883.32
14	Furniture office	309	1273	342532.65
15	stationary store	849	2311	277741.71
16	pet Shop	719	1710	253876.65
17	PCs	30	181	232799.43
18	musical instruments	289	628	210137.37
19	None	610	1451	207705.09

health and beauty, Watches present, bed table bath, sport leisure, computer accessories, Furniture Decoration, housewares, Automotive are some of the top selling product categories.

health and beauty products are top selling having highest orders.

PCs and Musical Instruments category have relatively less number of products , but contributes in a high revenue.

Average price per product category along with average fright value :

In [89]:
```
bigquery_client.query("""
select
    p.product_category,
    min(ois.price) as min_price_product,
    max(ois.price) as max_price_product,
    avg(ois.price) as avg_price,
    avg(ois.freight_value) as avg_fright_value

from target.orders as o
    join target.order_items_SALES as ois
    on o.order_id = ois.order_id
    join target.products as p
    on ois.product_id = p.product_id
group by
    p.product_category
order by avg_price desc,avg_fright_value desc
limit 15

;
""").to_dataframe()
```

Out[89]:

	product_category	min_price_product	max_price_product	avg_price	avg_fright_value
0	PCs	34.50	6729.00	1141.459558	49.239227
1	HOUSE PASTALS OVEN AND CAFE	10.19	2899.00	624.285658	36.156053
2	ELECTRICES 2	13.90	2350.00	483.264786	45.011496
3	Agro Industria e Comercio	12.99	2990.00	351.167650	28.114973
4	musical instruments	4.90	4399.87	293.112673	27.528522
5	electrostile	6.50	4799.00	291.720396	24.225649
6	Kitchen portable and food coach	17.42	1099.00	282.214286	21.329286
7	CONSTRUCTION SECURITY TOOLS	8.90	3099.90	217.560178	20.552899
8	Watches present	8.99	3999.90	204.782542	16.814406
9	Furniture	6.90	650.00	185.867474	43.568526

	product_category	min_price_product	max_price_product	avg_price	avg_fright_value
10	climatization	10.90	1599.00	181.552874	22.527362
11	Cool Stuff	7.00	3109.99	169.937600	22.180880
12	Construction Tools Construction	0.85	2300.00	168.448263	22.406679
13	Furniture office	25.00	1189.90	167.868488	40.977820
14	Furniture Kitchen Service Area Dinner and Garden	9.60	1320.00	167.861250	43.054032

PCs,house pastals oven and cafe,agro industry and commerce,musical instruments,Kitchen portable and food coach are having highest average product price categories.

In [ ]:

In [ ]:

top 10 product category having the costliest items in it.

In [90]:
```
bigquery_client.query("""
select
    p.product_category,

    max(ois.price) as max_price_product

from target.orders as o
    join target.order_items_SALES as ois
    on o.order_id = ois.order_id
    join target.products as p
    on ois.product_id = p.product_id
group by
    p.product_category
order by
    max_price_product desc
limit 10

;
""").to_dataframe()
```

Out[90]:

	product_category	max_price_product
0	housewares	6735.00
1	PCs	6729.00
2	Art	6499.00
3	electrostile	4799.00
4	musical instruments	4399.87
5	Games consoles	4099.99
6	sport leisure	4059.00
7	Watches present	3999.90
8	None	3980.00
9	Garden tools	3930.00

top 10 product category having the cheapest items in it.

In [91]:
```
bigquery_client.query("""
select
    p.product_category,

    min(ois.price) as min_price_product

from target.orders as o
    join target.order_items_SALES as ois
    on o.order_id = ois.order_id
    join target.products as p
    on ois.product_id = p.product_id
where
     p.product_category is not null
group by
    p.product_category
order by
    min_price_product

limit 10

;
""").to_dataframe()
```

Out[91]:

	product_category	min_price_product
0	Construction Tools Construction	0.85
1	HEALTH BEAUTY	1.20
2	stationary store	2.29
3	pet Shop	2.90
4	housewares	3.06
5	automotive	3.49
6	Art	3.50
7	babies	3.54
8	song	3.85

	product_category	min_price_product
9	computer accessories	3.90

In [92]:
```
# costliest and cheapest product prices :
```

In [93]:
```
bigquery_client.query("""
select
    min(ois.price) as min_price_product,
    max(ois.price) as max_price_product
from target.orders as o
    join target.order_items_SALES as ois
    on o.order_id = ois.order_id
    join target.products as p
    on ois.product_id = p.product_id


;
""").to_dataframe()
```

Out[93]:

	min_price_product	max_price_product
0	0.85	6735.0

In [ ]:
```

```

price range : 0-10,11-100,101-500,501-1500,1501,3000 above,

count of orders per price range :

In [94]:

```
df = bigquery_client.query("""



SELECT
price_category,
COUNT(order_id) AS number_of_orders
FROM
(
SELECT
*,
CASE    WHEN price BETWEEN 0 and 10 THEN '0-10'
    WHEN price BETWEEN 11 and 100 THEN '11-100'
    WHEN price BETWEEN 101 and 500 THEN '101-500'
    WHEN price BETWEEN 501 and 1500 THEN '501-1500'
    WHEN price BETWEEN 1501 and 3000 THEN '1501-3000'
    ELSE 'Above 3000'
    END AS price_category

FROM
    target.order_items_SALES
) as x
GROUP BY x.price_category
ORDER BY number_of_orders DESC

;



""").to_dataframe()
```

In [95]:
```
df["number_of_orders_in_%"] = (df["number_of_orders"]/df["number_of_orders"].sum())*100
df
```

Out[95]:

	price_category	number_of_orders	number_of_orders_in_%
0	11-100	63418	61.916524
1	101-500	34738	33.915548
2	501-1500	2800	2.733708
3	0-10	904	0.882597
4	1501-3000	296	0.288992
5	Above 3000	269	0.262631

61% orders are between price range 10-100. 33% are from 101-500 price range.

In [ ]:

In [96]:
```
bigquery_client.query("""



select

p.product_category,
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)) as avg_delivery_time

from `target-360705.target.orders`  as o

join `target-360705.target.order_items_SALES` as ois
on o.order_id = ois.order_id

join `target-360705.target.products` as p
on p.product_id = ois.product_id

where o.order_delivered_customer_date is not null

group by
p.product_category

order by avg_delivery_time desc


;

""").to_dataframe()
```

Out[96]:

	product_category	avg_delivery_time
0	Furniture office	20.176791
1	Christmas articles	15.062992
2	insurance and services	15.000000

	product_category	avg_delivery_time
3	Fashion Calcados	14.984252
4	House Comfort 2	14.000000
5	CITTE AND UPHACK FURNITURE	13.891892
6	Fashion Underwear and Beach Fashion	13.525424
7	PCs	13.468927
8	ELECTRICES 2	13.414097
9	Room Furniture	13.343602
10	Games consoles	13.252896
11	House comfort	13.229426
12	Furniture	13.188889
13	Garden tools	13.165394
14	audio	13.028736
15	Casa Construcao	13.012170
16	computer accessories	12.728246
17	Blu Ray DVDs	12.661017
18	Fashion Men's Clothing	12.586207
19	musical instruments	12.584142
20	Furniture Decoration	12.555003
21	electronics	12.448657
22	telephony	12.444950
23	bed table bath	12.431112
24	IMAGE IMPORT TABLETS	12.392405
25	stationary store	12.295883
26	None	12.221984
27	Watches present	12.199012

	product_category	avg_delivery_time
28	babies	12.172969
29	Fashion Women's Clothing	12.135135
30	fixed telephony	12.080189
31	Cool Stuff	11.897879
32	automotive	11.802102
33	Construction Tools Tools	11.762887
34	Market Place	11.762590
35	sport leisure	11.735977
36	HEALTH BEAUTY	11.573256
37	climatization	11.449393
38	Furniture Kitchen Service Area Dinner and Garden	11.398340
39	perfumery	11.369435
40	Construction Tools Garden	11.360577
41	CONSTRUCTION SECURITY TOOLS	11.254658
42	toys	11.219713
43	song	11.157895
44	General Interest Books	11.153996
45	flowers	11.034483
46	Agro Industria e Comercio	11.028090
47	home appliances	10.888889
48	pet Shop	10.834286
49	Art	10.774359
50	Fashion Bags and Accessories	10.670114
51	housewares	10.604968
52	electrostile	10.513912

	product_category	avg_delivery_time
53	Construction Tools Construction	10.507134
54	cds music dvds	10.500000
55	Industry Commerce and Business	10.472574
56	SIGNALIZATION AND SAFETY	10.397163
57	Bags Accessories	10.351351
58	Drink foods	10.334802
59	technical books	10.257692
60	Fashion Sport	10.153846
61	Hygiene diapers	9.960000
62	drinks	9.952055
63	cine photo	9.906250
64	PC Gamer	9.714286
65	HOUSE PASTALS OVEN AND CAFE	9.397260
66	foods	9.351351
67	Construction Tools Illumination	9.306122
68	party articles	8.578947
69	Fashion Children's Clothing	8.142857
70	Kitchen portable and food coach	8.076923
71	Imported books	7.785714
72	La Cuisine	7.230769
73	Arts and Crafts	5.291667

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Average price and freight_value per product category and correlation :

In [97]:

```
df = bigquery_client.query("""

SELECT

p.product_category,
avg(ois.price) as avg_price,
avg(ois.freight_value) as avg_freight_value,
avg(date_diff(o.order_estimated_delivery_date,o.order_purchase_timestamp,day)) as avg_estimated_delivery_time,
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)) as avg_delivery_time

FROM
target.order_items_SALES as ois
join target.products as p
on p.product_id = ois.product_id
join target.orders as o
on o.order_id = ois.order_id
where
o.order_delivered_customer_date is not null
group by
    p.product_category
order by
    avg_freight_value desc
;


""").to_dataframe()
```

In [98]:
```
df
```

Out[98]:

	product_category	avg_price	avg_freight_value	avg_estimated_delivery_time	avg_delivery_time
0	PCs	1143.080169	49.387684	24.401130	13.468927
1	ELECTRICES 2	474.538194	44.877004	25.189427	13.414097
2	Furniture	186.615778	42.768556	25.911111	13.188889
3	CITTE AND UPHACK FURNITURE	116.848108	42.739730	20.567568	13.891892
4	Furniture Kitchen Service Area Dinner and Garden	169.432365	42.402324	23.049793	11.398340
5	Furniture office	166.575584	40.690646	31.492991	20.176791
6	HOUSE PASTALS OVEN AND CAFE	638.213151	36.388219	20.383562	9.397260
7	Room Furniture	139.297796	35.521114	24.590047	13.343602
8	Industry Commerce and Business	152.549620	28.444177	22.092827	10.472574
9	Bags Accessories	130.575627	28.157510	22.446911	10.351351
10	Agro Industria e Comercio	352.406236	27.940674	21.702247	11.028090
11	musical instruments	290.826537	27.358074	23.469256	12.584142
12	Construction Tools Illumination	150.263388	27.174204	20.187755	9.306122
13	SIGNALIZATION AND SAFETY	110.870993	26.005887	21.602837	10.397163
14	Casa Construcao	150.376491	24.104523	23.916836	13.012170
15	electrostile	288.826236	23.968429	23.797054	10.513912
16	Garden tools	119.617043	23.624673	24.495052	13.165394
17	La Cuisine	147.306923	23.576154	23.153846	7.230769
18	Construction Tools Garden	107.074279	22.678894	24.250000	11.360577
19	babies	137.632206	22.542409	23.160014	12.172969
20	climatization	180.534130	22.370283	24.817814	11.449393
21	Construction Tools Construction	168.126835	22.303139	21.294423	10.507134
22	Cool Stuff	167.264487	22.049051	23.747489	11.897879

	product_category	avg_price	avg_freight_value	avg_estimated_delivery_time	avg_delivery_time
23	Christmas articles	60.099213	21.843543	26.574803	15.062992
24	automotive	141.075083	21.827190	22.460651	11.802102
25	Kitchen portable and food coach	301.238462	21.790000	17.000000	8.076923
26	housewares	95.000667	21.460259	22.164512	10.604968
27	Furniture Decoration	93.158917	21.305847	24.432188	12.555003
28	CONSTRUCTION SECURITY TOOLS	221.354472	20.765714	23.496894	11.254658
29	pet Shop	113.475383	20.624777	22.756571	10.834286
30	insurance and services	141.645000	20.610000	31.000000	15.000000
31	Blu Ray DVDs	75.637458	20.280000	25.203390	12.661017
32	Construction Tools Tools	148.952062	19.777010	23.371134	11.762887
33	House comfort	139.073367	19.767082	22.438903	13.229426
34	party articles	112.226053	19.759211	22.605263	8.578947
35	sport leisure	117.070771	19.604119	23.065354	11.735977
36	Art	120.493949	19.316974	23.087179	10.774359
37	home appliances	105.315971	19.231151	22.850067	10.888889
38	HEALTH BEAUTY	134.593658	19.167183	22.980299	11.573256
39	computer accessories	116.814604	18.906119	24.540830	12.728246
40	toys	118.690906	18.885734	22.728953	11.219713
41	stationary store	93.460103	18.757719	23.790738	12.295883
42	Fashion Calcados	89.987244	18.753937	29.255906	14.984252
43	Fashion Sport	75.954615	18.697692	21.807692	10.153846
44	bed table bath	94.979133	18.521942	23.421521	12.431112
45	song	158.798684	18.171316	24.894737	11.157895
46	cine photo	101.231562	17.878750	20.187500	9.906250
47	Games consoles	138.593494	17.591737	24.192085	13.252896

	product_category	avg_price	avg_freight_value	avg_estimated_delivery_time	avg_delivery_time
48	None	115.431692	17.588308	23.032834	12.221984
49	Market Place	94.816978	17.541942	25.320144	11.762590
50	fixed telephony	158.303443	17.174528	26.419811	12.080189
51	Fashion Men's Clothing	84.654569	16.972586	25.008621	12.586207
52	Watches present	202.655120	16.794234	23.491355	12.199012
53	electronics	57.171469	16.746098	22.859400	12.448657
54	Drink foods	59.862247	16.587004	20.933921	10.334802
55	General Interest Books	84.569903	16.523489	22.645224	11.153996
56	PC Gamer	155.421429	16.208571	20.142857	9.714286
57	cds music dvds	53.333333	16.064167	25.583333	10.500000
58	technical books	71.412846	16.043115	20.930769	10.257692
59	perfumery	119.336009	16.003464	23.212820	11.369435
60	audio	142.853966	15.754885	22.462644	13.028736
61	telephony	69.799100	15.642580	23.182623	12.444950
62	Hygiene diapers	47.511600	15.595200	21.840000	9.960000
63	Arts and Crafts	75.583750	15.422083	11.458333	5.291667
64	Fashion Bags and Accessories	75.773521	15.408930	23.394002	10.670114
65	IMAGE IMPORT TABLETS	90.792532	14.803671	24.873418	12.392405
66	flowers	33.204828	14.720690	22.310345	11.034483
67	drinks	56.410685	14.691610	20.246575	9.952055
68	Fashion Underwear and Beach Fashion	76.378475	14.588305	23.466102	13.525424
69	foods	59.714144	14.416014	18.565315	9.351351
70	House Comfort 2	28.452917	13.610833	22.166667	14.000000
71	Fashion Women's Clothing	63.516486	13.563243	23.540541	12.135135
72	Imported books	77.533214	12.924643	18.285714	7.785714

	product_category	avg_price	avg_freight_value	avg_estimated_delivery_time	avg_delivery_time
73	Fashion Children's Clothing	74.278571	11.245714	23.571429	8.142857

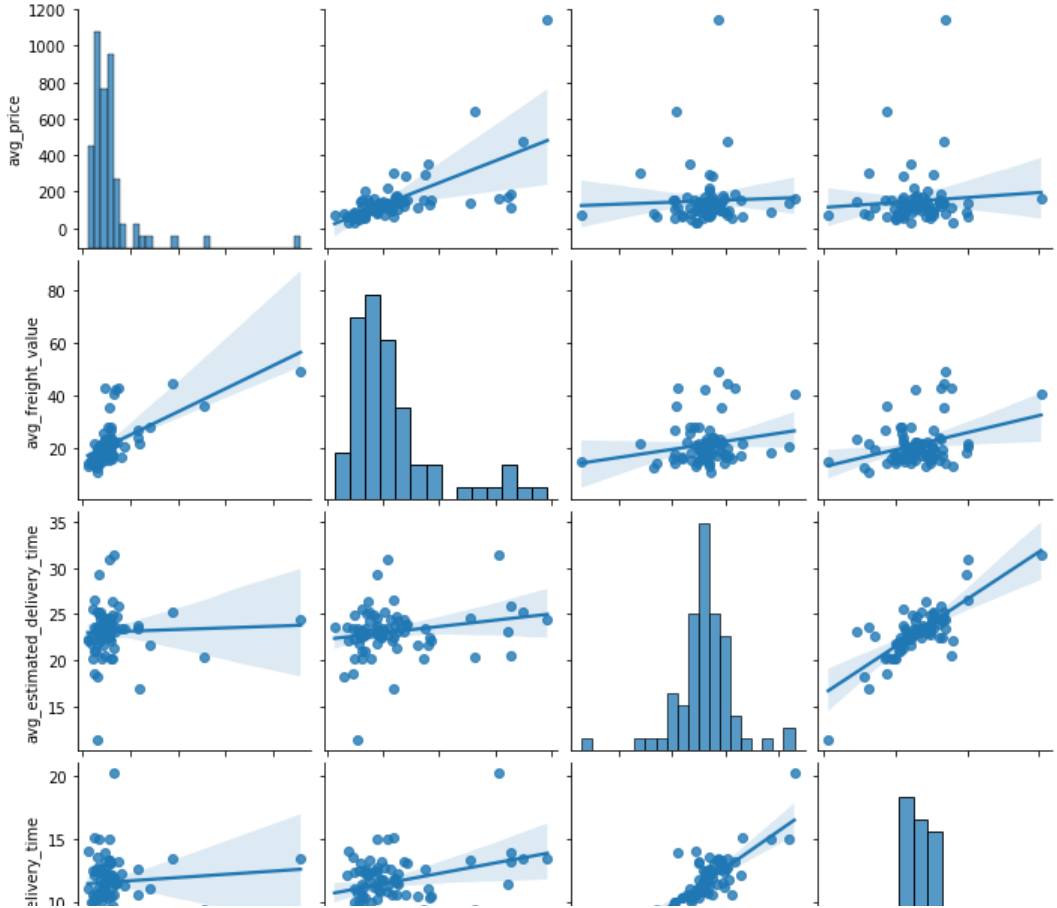
In [99]:
```
df.corr()
```

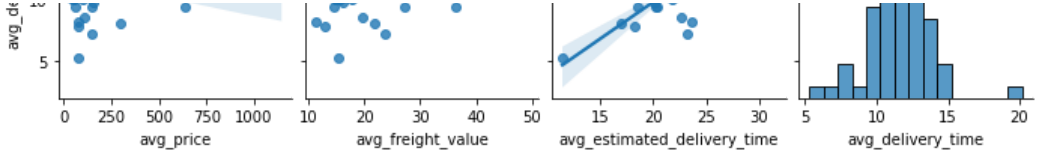
Out[99]:

	avg_price	avg_freight_value	avg_estimated_delivery_time	avg_delivery_time
avg_price	1.000000	0.646904	0.039161	0.074637
avg_freight_value	0.646904	1.000000	0.203909	0.325906
avg_estimated_delivery_time	0.039161	0.203909	1.000000	0.773978
avg_delivery_time	0.074637	0.325906	0.773978	1.000000

In [100]:
```
1 sns.pairplot(df,kind="reg")
```

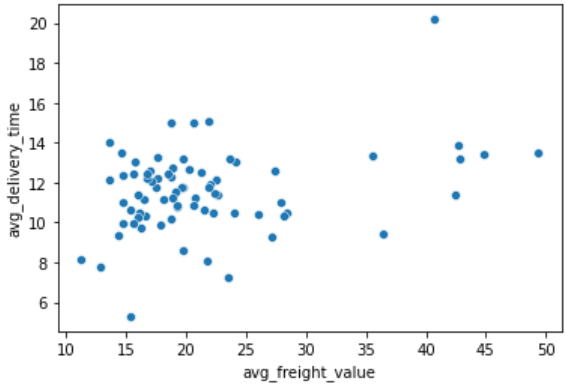
Out[100]: <seaborn.axisgrid.PairGrid at 0x2e1eb57f190>





In [101]:
```
sns.scatterplot(df["avg_freight_value"],df["avg_delivery_time"])
```

Out[101]:
```
<AxesSubplot:xlabel='avg_freight_value', ylabel='avg_delivery_time'>
```



avg_estimated_delivery_time and delivery_time have a positive correlation with avg_freight_value.

PCs/electronics, Furniture products,Kitchen Service Area Dinner and Garden equipments, Industry Commerce and Business , agro indsustrial commercial products,Bags Accessories, musical instruments, Construction Tools Illumination are some product categories having high average freighter value.

In [ ]:

Analysis delivery time

creating columns like, what is the time taken to complete delivery from purchase time.

what the expected time for delivery ?

how many days exceeded from expected delivery time ?

is the delivery was on time ?

In [102]:
```
bigquery_client.query("""


SELECT
*,
CASE WHEN days_exceeded_from_expected_delivery > 0 THEN 'delayed'
    ELSE 'on_time_delivery'
  END AS is_on_time
FROM
(
SELECT
    order_id,
    date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as days_exceeded_from_expected_del
    date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as days_taken_to_delivery,
    date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as estimated_time_to_delivery

FROM
    target.orders
WHERE
    order_status = 'delivered'
)
;




""").to_dataframe()
```

Out[102]:

	order_id	days_exceeded_from_expected_delivery	days_taken_to_delivery	estimated_time_to_delivery	
0	635c894d068ac37e6e03dc54eccb6189	-1	30	32	on
1	3b97562c3aee8bdedcb5c2e45a50d5e1	0	32	33	on
2	68f47f50f04c4cb6774570cfde3a9aa7	-1	29	31	on
3	276e9ec344d3bf029ff83a161c6b3ce9	4	43	39	

	order_id	days_exceeded_from_expected_delivery	days_taken_to_delivery	estimated_time_to_delivery
4	54e1a3c2b97fb0809da548a59f64c813	4	40	36
...	...	...	...	...
96473	ebca4856d6b3b849437fe99d11633d25	6	28	22
96474	a6f521d5e68e95961ac13d448a960fd7	-5	28	34
96475	f41397c4cf4c8a67f5f540472acbfe4f	5	28	22
96476	b674e463ea07d8a8fac9951be50283f1	10	28	18
96477	cd317fc34a4ea25ea8d1743f003712ac	11	28	16

96478 rows × 5 columns

In [ ]:

In [103]:
```
bigquery_client.query("""

SELECT
    DISTINCT
    x.delivery_on_time,
    COUNT(*) OVER (PARTITION BY x.delivery_on_time)
FROM
        (
        SELECT
        *,

        CASE
            WHEN order_delivered_customer_date IS NULL  THEN 'not_yet_delivered'
            WHEN (order_delivered_customer_date < order_estimated_delivery_date) THEN 'on_time_delivery'
            ELSE 'delayed'
            END AS delivery_on_time

        FROM
        target.orders
        ) AS x


;
""").to_dataframe()
```

Out[103]:

	delivery_on_time	f0_
0	on_time_delivery	88649
1	not_yet_delivered	2965
2	delayed	7827

In [ ]:

In [ ]:

In [104]:
```
dtime = bigquery_client.query("""


select
    order_id,
    order_status,
    date_diff(order_approved_at,order_purchase_timestamp,day) as approval_time,
    date_diff(order_delivered_carrier_date,order_approved_at,day) as time_taken_to_start_delivery_by_carrier,
    date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_taken_for_delivery,
    date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as estimated_time_for_delivery,
    date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as diff_estimated_delivery
from
    target.orders;
    """).to_dataframe()
```

In [ ]:

In [105]:
```
dtime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 7 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   order_id                               99441 non-null  object
 1   order_status                           99441 non-null  object
 2   approval_time                          99281 non-null  Int64
 3   time_taken_to_start_delivery_by_carrier  97644 non-null  Int64
 4   time_taken_for_delivery                96476 non-null  Int64
 5   estimated_time_for_delivery            99441 non-null  Int64
 6   diff_estimated_delivery                96476 non-null  Int64
dtypes: Int64(5), object(2)
memory usage: 5.8+ MB
```

In [106]:
```
dtime["approval_time"].mean(),dtime["approval_time"].median() # in days
```

Out[106]:
```
(0.2697998610005943, 0.0)
```

In [107]:
```
dtime["time_taken_to_start_delivery_by_carrier"].mean(),dtime["time_taken_to_start_delivery_by_carrier"].median()
```

Out[107]:
```
(2.315667117283192, 1.0)
```

In [108]:
```
dtime["time_taken_for_delivery"].mean(),dtime["time_taken_for_delivery"].median()  # in days
```

Out[108]:
```
(12.094085575687217, 10.0)
```

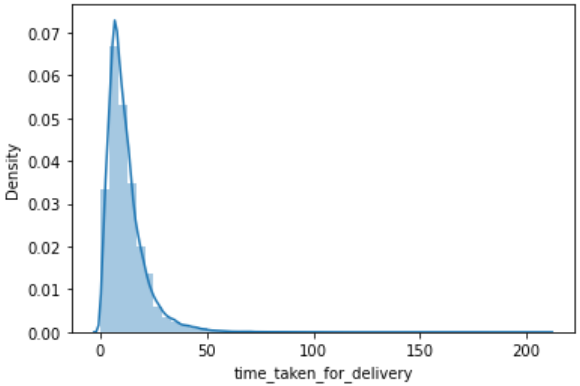
In [109]:
```
dtime["estimated_time_for_delivery"].mean(),dtime["estimated_time_for_delivery"].median() # in days
```

Out[109]:
```
(23.403958125923914, 23.0)
```

distribution of delivery time :

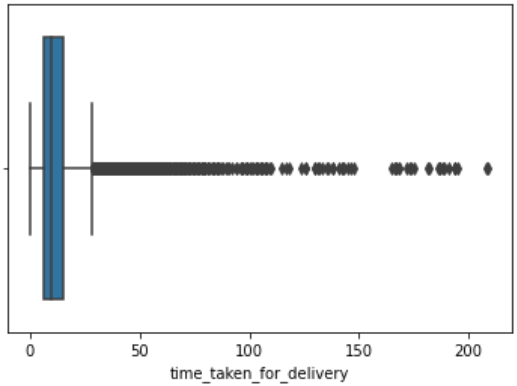
In [110]:
```
sns.distplot(dtime["time_taken_for_delivery"].dropna())
```

Out[110]:
```
<AxesSubplot:xlabel='time_taken_for_delivery', ylabel='Density'>
```



In [111]:
```
sns.boxplot(dtime["time_taken_for_delivery"].dropna())
```

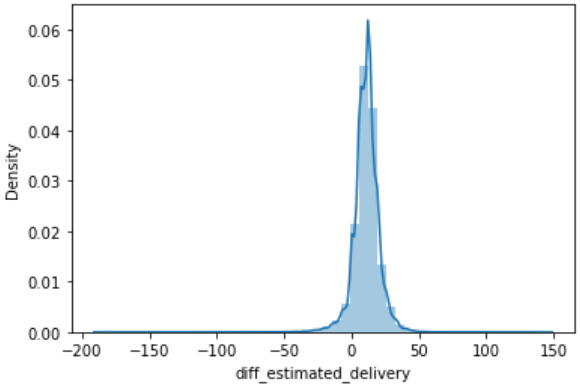
Out[111]:
```
<AxesSubplot:xlabel='time_taken_for_delivery'>
```



distibution of time difference between delivery time and estimated delivery time :

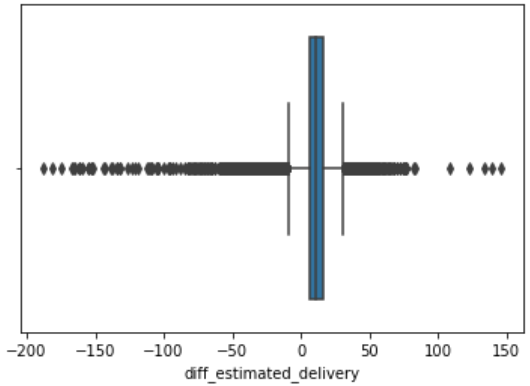
In [112]:
```
sns.distplot(dtime["diff_estimated_delivery"].dropna())
```

Out[112]:
```
<AxesSubplot:xlabel='diff_estimated_delivery', ylabel='Density'>
```



In [113]:
```
sns.boxplot(dtime["diff_estimated_delivery"].dropna())
```

Out[113]:
```
<AxesSubplot:xlabel='diff_estimated_delivery'>
```



In [114]:
```
state_delivery_time_freight_val = bigquery_client.query("""


/*Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery*/

SELECT
    c.customer_state,
    avg(date_diff(order_approved_at,order_purchase_timestamp,day)) as mean_of_approval_time,
    avg(date_diff(order_delivered_carrier_date,order_approved_at,day)) as mean_of_time_taken_to_start_delivery_by
    avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as mean_of_time_taken_for_delivery
    avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as mean_of_diff_estimated_del
    avg(ois.freight_value) avg_fright_value
FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
JOIN
`target-360705.target.order_items_SALES` as ois
on ois.order_id = o.order_id

group by
c.customer_state
order by
mean_of_time_taken_for_delivery

;
    """).to_dataframe()
```

In [115]:
```
state_delivery_time_freight_val.merge(brazil,how="inner",on="customer_state")
```

Out[115]:

	customer_state	mean_of_approval_time	mean_of_time_taken_to_start_delivery_by_carrier	mean_of_time_taken_for_delivery	mean_of_diff_
0	SP	0.246521	2.262769	8.264635	
1	PR	0.281365	2.270096	11.457694	
2	MG	0.262705	2.326563	11.505640	
3	DF	0.278607	2.275626	12.398799	
4	SC	0.290340	2.367738	14.456948	
5	RJ	0.236729	2.428419	14.751812	
6	RS	0.315350	2.232037	14.777277	
7	GO	0.328715	2.131566	15.046500	
8	MS	0.267760	2.246217	15.212707	
9	ES	0.280267	2.464046	15.225448	
10	TO	0.313793	2.394464	17.007018	
11	MT	0.355789	2.103376	17.406217	
12	PE	0.279028	2.314560	17.911152	
13	RN	0.284569	2.700803	18.733198	
14	BA	0.310632	2.314294	18.793073	
15	PI	0.259921	2.276768	18.991770	

	customer_state	mean_of_approval_time	mean_of_time_taken_to_start_delivery_by_carrier	mean_of_time_taken_for_delivery	mean_of_diff_
16	RO	0.333333	1.826772	19.007874	
17	PB	0.389397	2.445672	19.889098	
18	AC	0.451220	2.317073	20.604938	
19	CE	0.306735	2.401180	20.644647	
20	SE	0.264205	2.703704	20.944444	
21	MA	0.370274	2.534121	20.954362	
22	PA	0.322034	2.513026	23.323108	
23	AL	0.338095	2.431325	23.842365	
24	AM	0.238411	1.920530	25.986577	
25	AP	0.450704	2.271429	28.757143	
26	RR	1.617021	3.000000	28.975610	

In [116]:
```
np.corrcoef(state_delivery_time_freight_val["mean_of_time_taken_for_delivery"],
            state_delivery_time_freight_val["avg_fright_value"])
```

Out[116]:
```
array([[1.        , 0.77467453],
       [0.77467453, 1.        ]])
```

In [117]:
```
# positive high correlation between freight value and delivery time :
```

In [ ]:
```
```

In [118]:
```
bigquery_client.query("""


/*Group data by city, take mean of freight_value, time_to_delivery, diff_estimated_delivery*/

SELECT
    c.customer_city,
    avg(date_diff(order_approved_at,order_purchase_timestamp,day)) as mean_of_approval_time,
    avg(date_diff(order_delivered_carrier_date,order_approved_at,day)) as mean_of_time_taken_to_start_delivery_by
    avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as mean_of_time_taken_for_delivery
    avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as mean_of_diff_estimated_del
FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
WHERE
    o.order_delivered_customer_date is not null and
    o.order_delivered_carrier_date is not null and
    o.order_approved_at is not null
group by
c.customer_city
order by
mean_of_time_taken_for_delivery
limit 10

;
    """).to_dataframe()
```

Out[118]:

	customer_city	mean_of_approval_time	mean_of_time_taken_to_start_delivery_by_carrier	mean_of_time_taken_for_delivery	mean_of_diff
0	iomere	0.0	0.0	2.0	
1	acucena	0.0	0.0	3.0	
2	contenda	0.0	0.0	3.0	
3	siriji	0.0	0.0	3.0	

	customer_city	mean_of_approval_time	mean_of_time_taken_to_start_delivery_by_carrier	mean_of_time_taken_for_delivery	mean_of_diff
4	pedra bela	0.0	0.0	3.5	
5	bento de abreu	0.0	6.0	3.5	
6	sao joao da urtiga	0.0	0.0	4.0	
7	sao patricio	0.0	0.0	4.0	
8	divino das laranjeiras	0.0	0.0	4.0	
9	barao de juparana	0.0	1.0	4.0	

In [ ]:

In [ ]:

In [119]:
```
delivery_time = bigquery_client.query("""


SELECT
    o.order_id,
    c.customer_state,
    (date_diff(order_approved_at,order_purchase_timestamp,day)) as approval_time,
    (date_diff(order_delivered_carrier_date,order_approved_at,day)) as time_taken_to_start_delivery_by_carrier,
    (date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as time_taken_for_delivery,
    (date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)) as estimated_time_for_delivery,
    (date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as diff_estimated_delivery
FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
WHERE
    o.order_delivered_customer_date is not null and
    o.order_delivered_carrier_date is not null and
    o.order_approved_at is not null
ORDER BY
  c.customer_state
  ;
    """).to_dataframe()
```

In [120]:
```
delivery_time= delivery_time.merge(brazil,on="customer_state")
```

Calculating 95%Confidence Interval of Delivery time for each states :

In [121]:
```
delivery_time
```

Out[121]:

	order_id	customer_state	approval_time	time_taken_to_start_delivery_by_carrier	time_taken_for_delivery	e
0	c3ad507aba1f6b47354085e7e6bed0cf	AC	0	3	18	
1	5b829c37201779df841d79b1875ca7ef	AC	0	2	16	
2	48b4f1f96d5ae13b404fe7d041a8d393	AC	0	0	15	
3	0c956783114b7ac9633e16c494c191ce	AC	0	1	22	
4	e660da13091dbe48784500c1c3e2fb75	AC	0	3	22	
...	...	...	...	...	...	
96456	29145ea9a79826e9eb23fd0684f6777e	TO	1	8	30	
96457	334062cc253eacbccdb7825335f7b815	TO	0	0	15	
96458	d37b53698d3839a94cb33273bb8c0241	TO	0	1	12	
96459	abbf8270415671551b11b0e45c72eeb8	TO	0	5	29	
96460	5c8b55fb57c1225d199343bbc1ab6ee0	TO	0	4	34	

96461 rows × 9 columns

In [122]:
```
delivery_time["State"].unique()
```

Out[122]:
```
array(['Acre', 'Alagoas', 'Amazonas', 'Amapá', 'Bahia', 'Ceará',
       'Distrito Federal', 'Espírito Santo', 'Goiás', 'Maranhão',
       'Minas Gerais', 'MatoGrosso do Sul', 'MatoGrosso', 'Pará',
       'Paraíba', 'Pernambuco', 'Piauí', 'Paraná', 'Rio de Janeiro',
       'Rio Grande do Norte', 'Rondônia', 'Roraima', 'Rio Grande do Sul',
       'Santa Catarina', 'Sergipe', 'São Paulo', 'Tocantins'],
      dtype=object)
```

In [123]:
```
state_delivery_interval = {}
for state in ['Acre', 'Alagoas', 'Amazonas', 'Amapá', 'Bahia', 'Ceará',
       'Distrito Federal', 'Espírito Santo', 'Goiás', 'Maranhão',
       'Minas Gerais', 'MatoGrosso do Sul', 'MatoGrosso', 'Pará',
       'Paraíba', 'Pernambuco', 'Piauí', 'Paraná', 'Rio de Janeiro',
       'Rio Grande do Norte', 'Rondônia', 'Roraima', 'Rio Grande do Sul',
       'Santa Catarina', 'Sergipe', 'São Paulo', 'Tocantins']:
    btmeans = []
    data = delivery_time[delivery_time["State"]==state]["time_taken_for_delivery"]
    for i in range(1000):
        btsample = data.sample(50,replace=True)
        btmeans.append(np.mean(btsample))
        state_delivery_interval.update({state: [ np.mean(btmeans) - 1.96*(np.std(btmeans)),np.mean(btmeans) + 1.9
```

In [124]:
```
state_delivery_interval
```

Out[124]:
```
{'Acre': [17.768813188491205, 23.6785468115088],
 'Alagoas': [20.980534922399762, 27.23750507760024],
 'Amazonas': [21.970409453052962, 29.816510546947033],
 'Amapá': [20.766306424833093, 32.47169357516691],
 'Bahia': [15.622250726788305, 22.092069273211703],
 'Ceará': [17.18061485264945, 24.395225147350548],
 'Distrito Federal': [10.539943760778238, 14.535096239221764],
 'Espírito Santo': [12.377260970846564, 18.431539029153434],
 'Goiás': [12.496712178033068, 17.656727821966935],
 'Maranhão': [18.116914687871688, 24.25152531212831],
 'Minas Gerais': [9.712572110623608, 13.46814788937639],
 'MatoGrosso do Sul': [13.072987869997922, 17.224252130002075],
 'MatoGrosso': [15.271393223873627, 19.92180677612637],
 'Pará': [19.71447340875389, 27.05764659124611],
 'Paraíba': [16.96703600820457, 22.957923991795436],
 'Pernambuco': [14.943541767099857, 21.066378232900146],
 'Piauí': [14.838419866716432, 23.270900133283565],
 'Paraná': [9.669289411995365, 13.444590588004633],
 'Rio de Janeiro': [11.423285574127412, 18.30587442587259],
 'Rio Grande do Norte': [15.069781767797565, 22.477538232202434],
 'Rondônia': [16.763021508940128, 20.976618491059874],
 'Roraima': [21.303252806011212, 36.45786719398879],
 'Rio Grande do Sul': [12.372981610121204, 17.342698389878795],
 'Santa Catarina': [12.198192902328115, 16.582927097671885],
 'Sergipe': [16.535869413776467, 25.330210586223536],
 'São Paulo': [6.331933261416964, 10.177106738583038],
 'Tocantins': [15.196989736376977, 19.31073026362302]}
```

In [ ]:

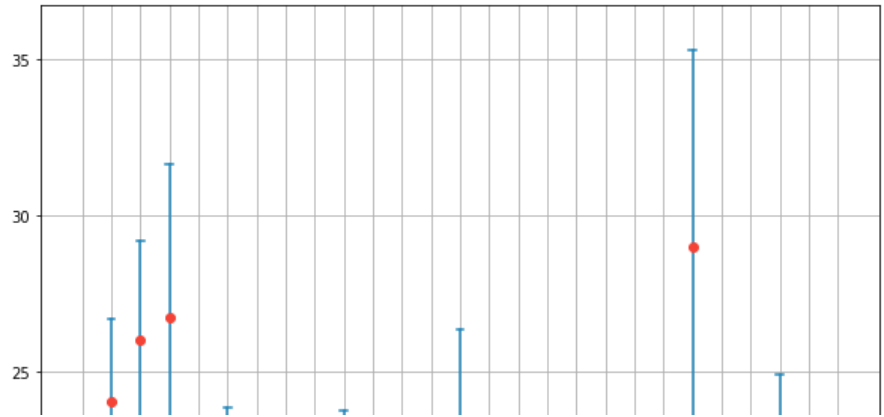
In [ ]:

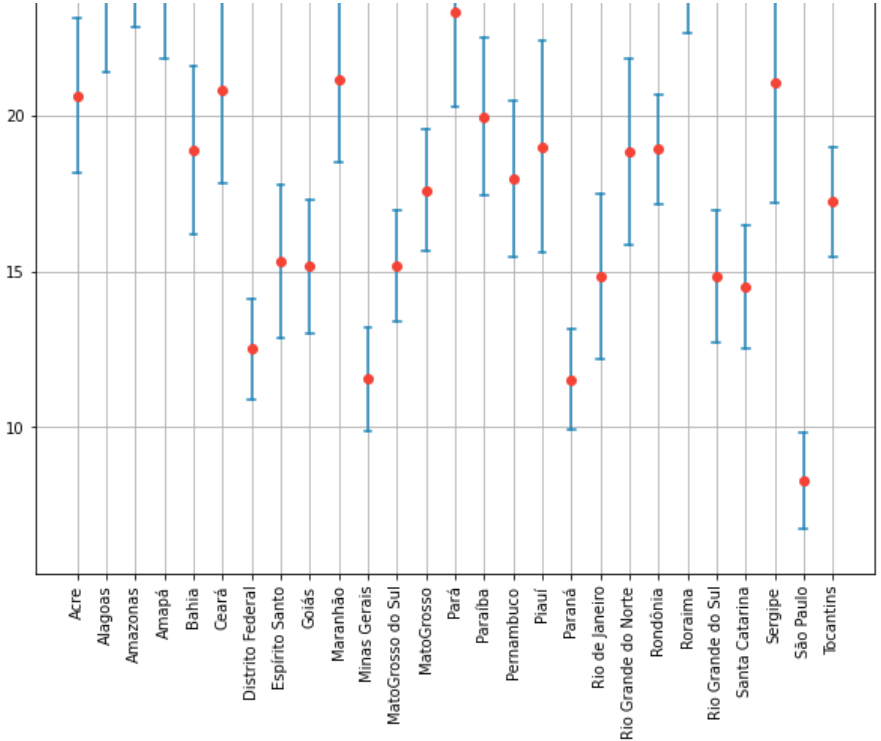
In [ ]:

In [125]:

```
def plot_confidence_interval(x, values,  color='#2187bb', horizontal_line_width=0.25,confidence=95):

    def CI(data,confidence, sample_size=75,trials = 3000):

        bootstrapped_mean= np.empty(trials)

        for i in range(trials):
            btssample = data.sample(n=sample_size,replace=True)
            bootstrapped_mean[i] = np.mean(btssample)
        sample_mean = np.mean(bootstrapped_mean)
        sample_std = np.std(data)
        standard_error = sample_std/np.sqrt(sample_size)
        talfa_by2 = t.ppf((1-((1-(confidence)/100)/2)),df = sample_size-1)
        margin_of_error = talfa_by2*standard_error

        return margin_of_error,sample_mean+margin_of_error,sample_mean-margin_of_error




    error,bottom,top = CI(values,confidence)

    left = x - horizontal_line_width / 2
    top = np.mean(values) - error
    right = x + horizontal_line_width / 2
    bottom = np.mean(values) + error

    plt.plot([x, x], [top, bottom], color=color)
    plt.plot([left, right], [top, top], color=color)
    plt.plot([left, right], [bottom, bottom], color=color)
    plt.plot(x, np.mean(values), 'o', color='#f44336')
```

In [126]:
```
plt.figure(figsize=(10,12))
plt.grid()
i = 1
for state in ['Acre', 'Alagoas', 'Amazonas', 'Amapá', 'Bahia', 'Ceará',
       'Distrito Federal', 'Espírito Santo', 'Goiás', 'Maranhão',
       'Minas Gerais', 'MatoGrosso do Sul', 'MatoGrosso', 'Pará',
       'Paraíba', 'Pernambuco', 'Piauí', 'Paraná', 'Rio de Janeiro',
       'Rio Grande do Norte', 'Rondônia', 'Roraima', 'Rio Grande do Sul',
       'Santa Catarina', 'Sergipe', 'São Paulo', 'Tocantins']:

    plot_confidence_interval(x = i,values=delivery_time[delivery_time["State"]==state]["time_taken_for_delivery"]
    i += 1
plt.xticks(list(range(1,28)),['Acre', 'Alagoas', 'Amazonas', 'Amapá', 'Bahia', 'Ceará',
       'Distrito Federal', 'Espírito Santo', 'Goiás', 'Maranhão',
       'Minas Gerais', 'MatoGrosso do Sul', 'MatoGrosso', 'Pará',
       'Paraíba', 'Pernambuco', 'Piauí', 'Paraná', 'Rio de Janeiro',
       'Rio Grande do Norte', 'Rondônia', 'Roraima', 'Rio Grande do Sul',
       'Santa Catarina', 'Sergipe', 'São Paulo', 'Tocantins'])
plt.xticks(rotation = 90)
plt.show()
```




20
15
10
Acre
Alagoas
Amazonas
Amapá
Bahia
Ceará
Distrito Federal
Espírito Santo
Goiás
Maranhão
Minas Gerais
MatoGrosso do Sul
MatoGrosso
Pará
Paraíba
Pernambuco
Piauí
Paraná
Rio de Janeiro
Rio Grande do Norte
Rondônia
Roraima
Rio Grande do Sul
Santa Catarina
Sergipe
São Paulo
Tocantins

In [127]:

```
df = bigquery_client.query("""

SELECT

    c.customer_state,

    avg(ois.freight_value) avg_fright_value,
    avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as time_taken_for_delivery,
    avg(date_diff(order_estimated_delivery_date,order_purchase_timestamp,day)) as estimated_time_for_delivery,
    avg(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) as diff_estimated_delivery,
    avg(date_diff(order_delivered_carrier_date,order_approved_at,day)) as time_taken_to_start_delivery_by_carrier
    avg(date_diff(order_approved_at,order_purchase_timestamp,day)) as approval_time
FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
JOIN
`target-360705.target.order_items_SALES` as ois
on ois.order_id = o.order_id
WHERE
    o.order_delivered_customer_date is not null and
    o.order_delivered_carrier_date is not null and
    o.order_approved_at is not null
GROUP BY
  c.customer_state

ORDER BY
  time_taken_for_delivery


  ;
    """).to_dataframe()
df.merge(brazil,how="inner",on="customer_state")
```

Out[127]:

	customer_state	avg_fright_value	time_taken_for_delivery	estimated_time_for_delivery	diff_estimated_delivery	time_taken_to_start_deliver

	customer_state	avg_fright_value	time_taken_for_delivery	estimated_time_for_delivery	diff_estimated_delivery	time_taken_to_start_deliver
0	SP	15.205497	8.263296	18.848318	10.239218	
1	PR	20.490765	11.457694	24.320543	12.499606	
2	MG	20.744201	11.504666	24.218358	12.363590	
3	DF	21.285139	12.398799	24.006470	11.284658	
4	SC	21.706368	14.456948	25.507902	10.717984	
5	RJ	21.003501	14.752319	26.019880	11.012318	
6	RS	21.807503	14.778279	28.243190	13.107703	
7	GO	22.900348	15.046500	26.693588	11.342144	
8	MS	24.101119	15.212707	25.577348	10.131215	
9	ES	22.078897	15.225448	25.232704	9.735365	
10	TO	37.810316	17.007018	28.694737	11.435088	
11	MT	27.878232	17.406217	31.464094	13.720257	
12	PE	32.586998	17.911152	30.677696	12.444853	
13	RN	36.364766	18.733198	32.044807	13.020367	
14	BA	26.392741	18.793073	29.119301	10.052102	
15	PI	38.777901	18.991770	29.775720	10.545267	
16	RO	42.050197	19.007874	38.677165	19.330709	
17	PB	41.777218	19.889098	32.678571	12.464286	
18	AC	41.286049	20.604938	40.864198	19.925926	
19	CE	33.023549	20.646657	31.014438	10.131459	

	customer_state	avg_fright_value	time_taken_for_delivery	estimated_time_for_delivery	diff_estimated_delivery	time_taken_to_start_deliver
20	SE	36.609912	20.944444	30.447368	9.222222	
21	MA	37.971935	20.965054	30.165323	9.000000	
22	PA	35.680552	23.323108	36.853783	13.244376	
23	AL	36.170961	23.842365	32.160099	8.091133	
24	AM	33.253423	25.986577	44.838926	18.530201	
25	AP	34.895571	28.757143	45.785714	16.642857	
26	RR	42.006829	28.975610	45.634146	16.414634	

after purchasing , the average time for approving the order by seller is 0.26 days and median time is 0 , means with in a day.

average time taken for a carrier to start the delivery is 2 and a half day.

average time taken to complete delivery is 12 days. and median of delivery time is 10 days.

estimated time delivery average is 23 days.

There is a positive correlation between freight value and delivery time.

long distance deliveries are having highers freight values and also takes more time for delivery.

states São Paulo ,Paraná,Minas Gerais, Distrito Federal ,Santa Catarina and Rio de Janeiro are some of the states having faster delivery time relatively.

Alagoas, Amazonas, Amapá ,Pará and Roraima are some states have very slow delivery time relatively.

Top 5 states with lowest average freight value

In [128]:
```
df = bigquery_client.query("""

SELECT

    c.customer_state,

    avg(ois.freight_value) avg_fright_value

FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
JOIN
`target-360705.target.order_items_SALES` as ois
on ois.order_id = o.order_id

GROUP BY
  c.customer_state

ORDER BY
  avg_fright_value asc
LIMIT 5
  ;
    """).to_dataframe()
df.merge(brazil,how="inner",on="customer_state")
```

Out[128]:

	customer_state	avg_fright_value	State	Region
0	SP	15.239253	São Paulo	Southeast
1	PR	20.525581	Paraná	South
2	MG	20.746072	Minas Gerais	Southeast
3	RJ	21.057323	Rio de Janeiro	Southeast
4	DF	21.258055	Distrito Federal	Center West

Top 5 states with highest average freight value

In [129]:
```
df = bigquery_client.query("""

SELECT

    c.customer_state,

    avg(ois.freight_value) avg_fright_value

FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
JOIN
`target-360705.target.order_items_SALES` as ois
on ois.order_id = o.order_id


GROUP BY
  c.customer_state

ORDER BY
  avg_fright_value desc
LIMIT 5
  ;
    """).to_dataframe()
df.merge(brazil,how="inner",on="customer_state")
```

Out[129]:

	customer_state	avg_fright_value	State	Region
0	RR	42.030213	Roraima	North
1	RO	41.830310	Rondônia	North
2	PB	41.422852	Paraíba	Northeast
3	AC	41.299512	Acre	North
4	PI	38.853631	Piauí	Northeast

Top 5 states with lowest average time to delivery

In [130]:
```
df = bigquery_client.query("""

SELECT

    c.customer_state,
    avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as time_taken_for_delivery

FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
GROUP BY
  c.customer_state

ORDER BY
    time_taken_for_delivery
LIMIT 5
;
    """).to_dataframe()
df.merge(brazil,how="inner",on="customer_state")
```

Out[130]:

	customer_state	time_taken_for_delivery	State	Region
0	SP	8.298061	São Paulo	Southeast
1	PR	11.526711	Paraná	South
2	MG	11.543813	Minas Gerais	Southeast
3	DF	12.509135	Distrito Federal	Center West
4	SC	14.479560	Santa Catarina	South

Top 5 states with highest average time to delivery

In [131]:
```
df = bigquery_client.query("""

SELECT

    c.customer_state,
    avg(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) as time_taken_for_delivery

FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
GROUP BY
  c.customer_state

ORDER BY
    time_taken_for_delivery DESC
LIMIT 5
;
    """).to_dataframe()
df.merge(brazil,how="inner",on="customer_state")
```

Out[131]:

	customer_state	time_taken_for_delivery	State	Region
0	RR	28.975610	Roraima	North
1	AP	26.731343	Amapá	North
2	AM	25.986207	Amazonas	North
3	AL	24.040302	Alagoas	Northeast
4	PA	23.316068	Pará	North

In [ ]:

Top 5 states where delivery is really fast compared to estimated date

In [132]:
```
df = bigquery_client.query("""

SELECT
    c.customer_state,
    avg(date_diff(order_estimated_delivery_date,
                  order_delivered_customer_date,
                  day)) as difference_betweem_estimated_and_delivery_dates,
    avg(date_diff(order_delivered_carrier_date,
                  order_approved_at,
                  day)) as time_taken_to_start_delivery_by_carrier
FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
WHERE
    o.order_delivered_customer_date is not null and
    o.order_delivered_carrier_date is not null and
    o.order_approved_at is not null
GROUP BY
  c.customer_state

ORDER BY
  difference_betweem_estimated_and_delivery_dates,
  time_taken_to_start_delivery_by_carrier
LIMIT 5
  ;
    """).to_dataframe()
df.merge(brazil,how="inner",on="customer_state")
```

Out[132]:

	customer_state	difference_betweem_estimated_and_delivery_dates	time_taken_to_start_delivery_by_carrier	State	Region
0	AL	7.947103	2.478589	Alagoas	Northeast
1	MA	8.743017	2.546089	Maranhão	Northeast
2	SE	9.173134	2.674627	Sergipe	Northeast
3	ES	9.618546	2.452632	Espírito Santo	Southeast

	customer_state	difference_betweem_estimated_and_delivery_dates	time_taken_to_start_delivery_by_carrier	State	Region
4	BA	9.934889	2.315111	Bahia	Northeast

Top 5 states where delivery is really slow compared to estimated date

In [133]:
```
df = bigquery_client.query("""

SELECT
    c.customer_state,
    avg(date_diff(order_estimated_delivery_date,
                  order_delivered_customer_date,
                  day)) as difference_betweem_estimated_and_delivery_dates,
    avg(date_diff(order_delivered_carrier_date,
                  order_approved_at,
                  day)) as time_taken_to_start_delivery_by_carrier
FROM
`target-360705.target.orders` as o
JOIN
`target-360705.target.customers` as c
on o.customer_id = c.customer_id
WHERE
    o.order_delivered_customer_date is not null and
    o.order_delivered_carrier_date is not null and
    o.order_approved_at is not null
GROUP BY
  c.customer_state

ORDER BY
  difference_betweem_estimated_and_delivery_dates DESC,
  time_taken_to_start_delivery_by_carrier
LIMIT 5
  ;
    """).to_dataframe()
df = df.merge(brazil,how="inner",on="customer_state")
```

In [134]:
```
df
```

Out[134]:

	customer_state	difference_betweem_estimated_and_delivery_dates	time_taken_to_start_delivery_by_carrier	State	Region
0	AC	19.762500	2.337500	Acre	North
1	RO	19.131687	1.802469	Rondônia	North
2	AP	18.731343	2.298507	Amapá	North
3	AM	18.606897	1.931034	Amazonas	North
4	RR	16.414634	2.829268	Roraima	North

In [ ]:

In [ ]:

In [ ]:

Payments related information :

In [ ]:

In [135]:
```
bigquery_client.query("""

SELECT
    * EXCEPT(is_generated, generation_expression, is_stored, is_updatable,is_hidden
    ,is_system_defined,is_partitioning_column,
    clustering_ordinal_position,collation_name,column_default,ordinal_position,table_catalog)
  FROM
    target-360705.target.INFORMATION_SCHEMA.COLUMNS
  WHERE
    table_name = 'payments';

""").to_dataframe()
```

Out[135]:

	table_schema	table_name	column_name	is_nullable	data_type
0	target	payments	order_id	YES	STRING
1	target	payments	payment_sequential	YES	INT64
2	target	payments	payment_type	YES	STRING
3	target	payments	payment_installments	YES	INT64
4	target	payments	payment_value	YES	FLOAT64

In [136]:
```
df = bigquery_client.query("""

SELECT
    payment_type,
    count(distinct(order_id)) Number_of_sales_per_payment_type,
    sum(payment_value) as Total_Payment_per_payment_type
FROM  target.payments
GROUP BY
    payment_type

;
""").to_dataframe()
```

In [137]:
```
df["in_percentage_payment"] =(df["Total_Payment_per_payment_type"]/df["Total_Payment_per_payment_type"].sum())*10
```

In [138]:
```
df
```

Out[138]:

	payment_type	Number_of_sales_per_payment_type	Total_Payment_per_payment_type	in_percentage_payment
0	credit_card	76505	12542084.19	78.344584
1	voucher	3866	379436.87	2.370166
2	not_defined	3	0.00	0.000000
3	debit_card	1528	217989.79	1.361681
4	UPI	19784	2869361.27	17.923569

78% payments are done using credit card and 17.92% are done with UPI .

In [ ]:
```
```

In [ ]:

In [ ]:

Distribution of payment installments and count of orders

In [ ]:

In [139]:
```
df = bigquery_client.query("""

SELECT
payment_installments,
count(distinct(order_id)) number_of_orders
from
target.payments
group by payment_installments

  ;
    """).to_dataframe()
```

In [140]:
```
df
```

Out[140]:

	payment_installments	number_of_orders
0	0	2
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315
11	11	23
12	12	133
13	13	16
14	14	15
15	15	74
16	16	5
17	17	8
18	18	27
19	20	17
20	21	3
21	22	1
22	23	1

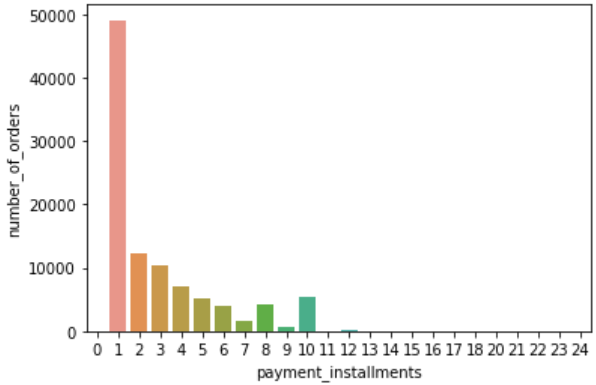
	payment_installments	number_of_orders
23	24	18

In [141]:
```
df.columns
```

Out[141]:
```
Index(['payment_installments', 'number_of_orders'], dtype='object')
```

In [142]:
```
sns.barplot(x = df["payment_installments"],
            y = df["number_of_orders"])
```

Out[142]:
```
<AxesSubplot:xlabel='payment_installments', ylabel='number_of_orders'>
```



majority of the orders are purchased at 1 payment installments.

more than 5 installments purchases are relatively very low.

Month over Month count of orders for different payment types

In [143]:
```
bigquery_client.query("""

SELECT

    FORMAT_TIMESTAMP("%b %Y", order_purchase_timestamp) as Month_year_purchase_date,
    extract(year from order_purchase_timestamp) as year,
    p.payment_type,
    count(o.order_id) number_of_orders
from
target.payments as p
join target.orders as o
on o.order_id = p.order_id
group by year,Month_year_purchase_date,p.payment_type
order by year,Month_year_purchase_date,p.payment_type

  ;
    """).to_dataframe()
```

Out[143]:

	Month_year_purchase_date	year	payment_type	number_of_orders
0	Dec 2016	2016	credit_card	1
1	Oct 2016	2016	UPI	63
2	Oct 2016	2016	credit_card	254
3	Oct 2016	2016	debit_card	2
4	Oct 2016	2016	voucher	23
5	Sep 2016	2016	credit_card	3
6	Apr 2017	2017	UPI	496
7	Apr 2017	2017	credit_card	1846
8	Apr 2017	2017	debit_card	27
9	Apr 2017	2017	voucher	202
10	Aug 2017	2017	UPI	938

	Month_year_purchase_date	year	payment_type	number_of_orders
11	Aug 2017	2017	credit_card	3284
12	Aug 2017	2017	debit_card	34
13	Aug 2017	2017	voucher	294
14	Dec 2017	2017	UPI	1160
15	Dec 2017	2017	credit_card	4377
16	Dec 2017	2017	debit_card	64
17	Dec 2017	2017	voucher	294
18	Feb 2017	2017	UPI	398
19	Feb 2017	2017	credit_card	1356
20	Feb 2017	2017	debit_card	13
21	Feb 2017	2017	voucher	119
22	Jan 2017	2017	UPI	197
23	Jan 2017	2017	credit_card	583
24	Jan 2017	2017	debit_card	9
25	Jan 2017	2017	voucher	61
26	Jul 2017	2017	UPI	845
27	Jul 2017	2017	credit_card	3086
28	Jul 2017	2017	debit_card	22
29	Jul 2017	2017	voucher	364
30	Jun 2017	2017	UPI	707
31	Jun 2017	2017	credit_card	2463
32	Jun 2017	2017	debit_card	27
33	Jun 2017	2017	voucher	239
34	Mar 2017	2017	UPI	590
35	Mar 2017	2017	credit_card	2016

	Month_year_purchase_date	year	payment_type	number_of_orders
36	Mar 2017	2017	debit_card	31
37	Mar 2017	2017	voucher	200
38	May 2017	2017	UPI	772
39	May 2017	2017	credit_card	2853
40	May 2017	2017	debit_card	30
41	May 2017	2017	voucher	289
42	Nov 2017	2017	UPI	1509
43	Nov 2017	2017	credit_card	5897
44	Nov 2017	2017	debit_card	70
45	Nov 2017	2017	voucher	387
46	Oct 2017	2017	UPI	993
47	Oct 2017	2017	credit_card	3524
48	Oct 2017	2017	debit_card	52
49	Oct 2017	2017	voucher	291
50	Sep 2017	2017	UPI	903
51	Sep 2017	2017	credit_card	3283
52	Sep 2017	2017	debit_card	43
53	Sep 2017	2017	voucher	287
54	Apr 2018	2018	UPI	1287
55	Apr 2018	2018	credit_card	5455
56	Apr 2018	2018	debit_card	97
57	Apr 2018	2018	voucher	370
58	Aug 2018	2018	UPI	1139
59	Aug 2018	2018	credit_card	4985
60	Aug 2018	2018	debit_card	277

	Month_year_purchase_date	year	payment_type	number_of_orders
61	Aug 2018	2018	not_defined	2
62	Aug 2018	2018	voucher	295
63	Feb 2018	2018	UPI	1325
64	Feb 2018	2018	credit_card	5253
65	Feb 2018	2018	debit_card	69
66	Feb 2018	2018	voucher	305
67	Jan 2018	2018	UPI	1518
68	Jan 2018	2018	credit_card	5520
69	Jan 2018	2018	debit_card	109
70	Jan 2018	2018	voucher	416
71	Jul 2018	2018	UPI	1229
72	Jul 2018	2018	credit_card	4755
73	Jul 2018	2018	debit_card	242
74	Jul 2018	2018	voucher	281
75	Jun 2018	2018	UPI	1100
76	Jun 2018	2018	credit_card	4813
77	Jun 2018	2018	debit_card	182
78	Jun 2018	2018	voucher	324
79	Mar 2018	2018	UPI	1352
80	Mar 2018	2018	credit_card	5691
81	Mar 2018	2018	debit_card	78
82	Mar 2018	2018	voucher	391
83	May 2018	2018	UPI	1263
84	May 2018	2018	credit_card	5497
85	May 2018	2018	debit_card	51

	Month_year_purchase_date	year	payment_type	number_of_orders
86	May 2018	2018	voucher	324
87	Oct 2018	2018	voucher	4
88	Sep 2018	2018	not_defined	1
89	Sep 2018	2018	voucher	15

In [ ]:

In [ ]:

In [ ]:

Insights and Recommendations :

Insights :

customers and seller informations :

`In [11] to  In [27]`

- We have 99,441 customers of data available.
- We have 96096 number of Unique Customers ids.
- 14994 different locations of customers
- Customers are from different 4119 cities and 27 states from Brazil.
- total 99441 customers are there in given data.
- from total 99441 orders , 1107 are shipped ,625 were canceled, 96478 are delivered.
- 68% customers are from southeast Brazil , 14% are from south Brazil and rest are other other regions of Brazil .
- Total 3095 sellers who are from 611 different cities and 23 states in Brazil and from 2246 different areas as per zip-code data.
- São Paulo state has the highest numbers of sellers in country.

analysis of sales and revenue as per time :

```
In [42]
```

- Time period for which the data is given is 25 months.

```
In [44,45,46]
```

- compare to 2017 , revenue has increased in 2018 by 21%.

```
In [49-50-51]
```

- Average number of order are higher during November month , september and october month average orders are comparatively low , in may and july and august have higher average orders compare to other months.

```
In [73]:
```

- Tuesday, monday and wednesdays have relatively higher number of orders.

increasing trend :

```
In [52] to In [64]
```

- there is a increasing trend in orders , trend sustains during 2018. There a slight fall we can observe during october 2017 following with a great hike in november month and again a fall in end of december 2017 and january 2018.
- we can observe the trend of increasing orders with time and also for revenue.

```
In [65] , In [66]
```

- we can observe there's 815% growth increased in terms of orders and 707% growth increment in terms of revenue in January from 2017 to 2018.
- growth rate for july and august in 2017 to 2018 is relatively very low!

```
In [67] to In [72]
```

- 2017-february, 2017-march,2017-november were the highest growing sale month compare to its previous month.

Customer_purchasing Behavior:

```
In [76] to In [82]
```

- customers are purchasing during moring 8am to late evening 11pm.
- afternoon and evening orders are very high , compare to morning , and night time.

Product Category :

```
In [28] to In [34]
```

- In products Data , total 32951 different products available in Target with 73 different product_category.

```
In [88]
```

- health and beauty, Watches present, bed table bath, sport leisure, computer accessories, Furniture Decoration, housewares, Automotive are some of the top selling product categories.
- health and beauty products are top selling having highest orders.
- PCs and Musical Instruments category have relatively less number of products , but contributes in a high revenue.

```
In [89]
```

- PCs,house pastals oven and cafe,agro industry and commerce,musical instruments,Kitchen portable and food coach are having highest average product price categories.

```
In [94] - In [95]
```

- 61% orders are between price range 10-100. 33% are from 101-500 price range.

```
In [97 to 101]
```

- PCs/electronics, Furniture products,Kitchen Service Area Dinner and Garden equipments, Industry Commerce and Business , agro indsustrial commercial products,Bags Accessories, musical instruments, Construction Tools Illumination are some product categories having high average freighter value.

Delivery time :

```
In [97] to In [101]
```

- avg_estimated_delivery_time and delivery_time have a positive correlation with avg_freight_value.

```
In [102] to In [127]
```

- after purchasing , the average time for approving the order by seller is 0.26 days and median time is 0 , means with in a day.
- average time taken for a carrier to start the delivery is 2 and a half day.
- average time taken to complete delivery is 12 days. and median of delivery time is 10 days.

```
In [109]
```

- estimated time delivery average is 23 days.
- There is a positive correlation between freight value and delivery time.

```
In [126]
```

- states São Paulo ,Paraná, Minas Gerais, Distrito Federal ,Santa Catarina and Rio de Janeiro are some of the states having faster delivery time relatively.
- Alagoas, Amazonas, Amapá ,Pará and Roraima are some states have very slow delivery time relatively.

Region and State vise Analysis :

```
In [128] to In [133]
Geo-maps In [85],In [83]
```

- from above geo-map and query we can observe
- São Paulo ,Rio de Janeiro , Minas Gerais ,Rio Grande do Sul and Paraná are top 5 highest orders states and also generating highest revenue.
- more than 80% of orders are coming from south, southeast and nothest Brazil. 90% of the revenue is coming from south, southeast and nothest Brazil .

Payment type related info :

```
In [135]- In [143]
```

- 78% payments are done using credit card and 17.92% are done with UPI .
- majority of the orders are purchased at 1 payment installments.
- more than 5 installments purchases are relatively very low.

Recommendations :

- from the distribution and statistical analysis we can observe the average time to complete the delivery is 12 days . which should be reduced to atleast half , as due to high competition in e-commerce market , its is vital to do so.
- In order to reduce the delivery time, if we look at the average time for carrier to start the delivery itself takes atleast 2 and a half days. and order approaval time is 0.26 days . These two should be optimized at as low as possible, that can result into delivering faster.
- If we look at Top states where delivery is really slow compared to estimated date , they are all from north Brazil region. Delivering faster in the north states may create and increase new customers and revenue from north.
- Increasing network in north brazil , having small towns can help increase the customer base. As north Brazil has the worlds largest river and most extesive rain forest, must be a good travel destination, introducing necessary servival/ camping/adventure products can help increase revenue and order from northen region .
- top selling items are between 10-100 dollars,introducing new different more products from top selling categories can increase revenue more.
- It was obsere an increasing trend in revenue and orders over time , yet during october and january sales are decreasing probably after Festival Sales. Introducing possible discount on not so running product can help sell more products during those low going months.

In [ ]:

- Thank you

In [ ]:

In [ ]:

In [ ]:

In [ ]: