Problem statement

In [88]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import calendar
```

What 'good' looks like?

- Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
  - Data type of columns in a table
  - Time period for which the data is given
  - Cities and States covered in the dataset
- In-depth Exploration:
  - 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?
  - What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?
- Evolution of E-commerce orders in the Brazil region:
  - Get month on month orders by region, states
  - How are customers distributed in Brazil
- Impact on Economy: Analyze the money movemented by e-commerce by looking at order prices, freight and others.
  - Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)
  - Mean & Sum of price and freight value by customer state
- Analysis on sales, freight and delivery time
  - Calculate days between purchasing, delivering and estimated delivery
  - Create columns:
    - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
    - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
  - Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery
  - Sort the data to get the following:
    - Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

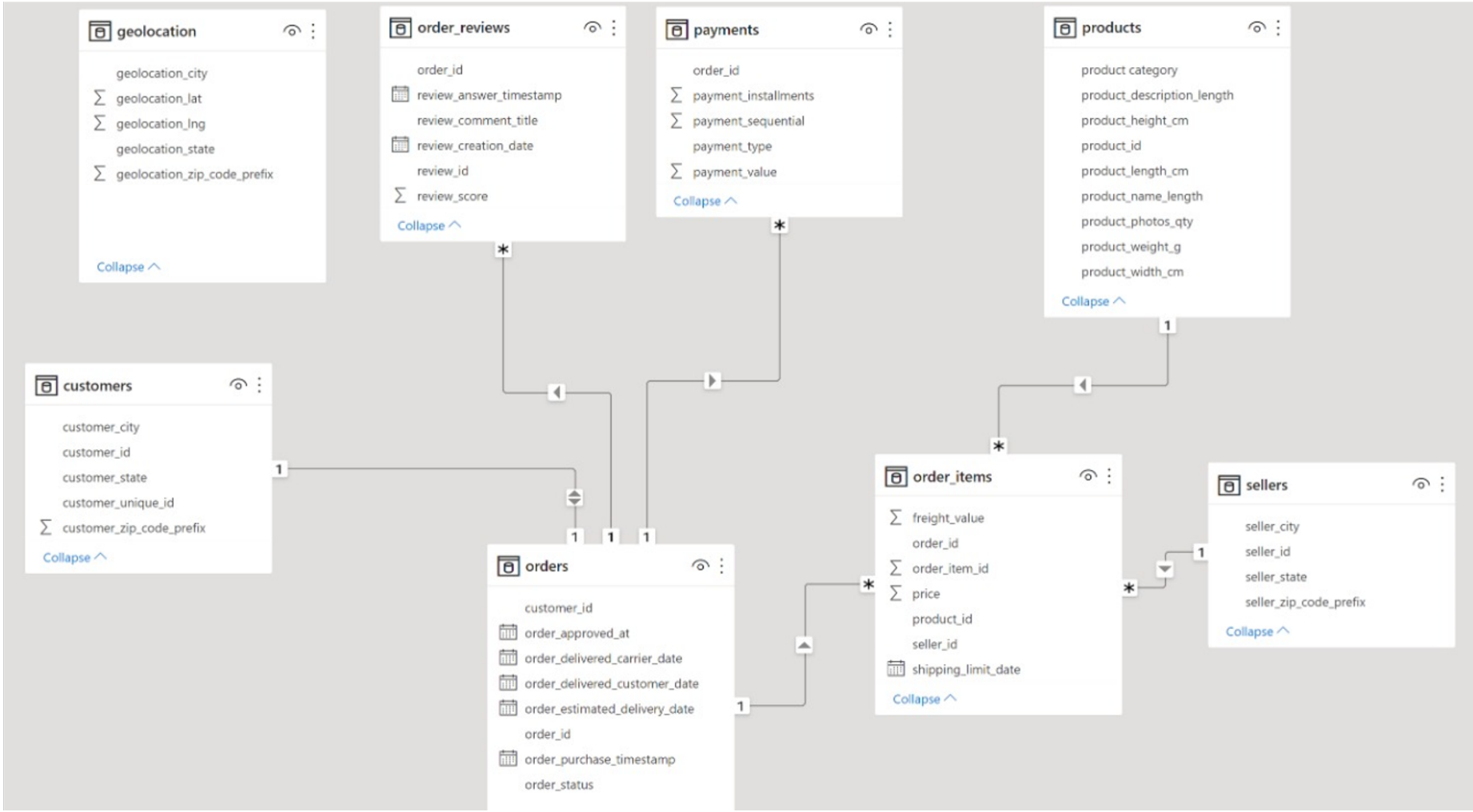
    - Top 5 states with highest/lowest average time to delivery
    - Top 5 states where delivery is really fast/ not so fast compared to estimated date
- Payment type analysis:
    - Month over Month count of orders for different payment types
    - Distribution of payment installments and count of orders

Evaluation *[...]*

Installation *[...]*

PROJECT SQL

EXPLORING EACH TABLE ONE BY ONE


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
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

ORDERS

The **orders.csv** contain following features:

Features	Description
order_id	A unique id of order made by the consumers.
customer_id	Id of the consumer who made the purchase.
order_status	status of the order made i.e delivered, shipped etc.
order_purchase_timestamp	Timestamp of the purchase.
order_delivered_carrier_date	delivery date at which carrier made the delivery.
order_delivered_customer_date	date at which customer got the product.
order_estimated_delivery_date	estimated delivery date of the products.

In [21]:

```
QUERY = """

SELECT * FROM Target.orders LIMIT 1000



  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[21]:

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_deliv
0	7a4df5d8cff4090e541401a20a22bb80	725e9c75605414b21fd8c8d5a1c2f1d6	created	2017-11-25 11:10:33+00:00	NaT	
1	35de4050331c6c644cddc86f4f2d0d64	4ee64f4bfc542546f422da0aeb462853	created	2017-12-05 01:07:58+00:00	NaT	
2	b5359909123fa03c50bdb0cfed07f098	438449d4af8980d107bf04571413a8e7	created	2017-12-05 01:07:52+00:00	NaT	
3	dba5062fbda3af4fb6c33b1e040ca38f	964a6df3d9bdf60fe3e7b8bb69ed893a	created	2018-02-09 17:21:04+00:00	NaT	
4	90ab3e7d52544ec7bc3363c82689965f	7d61b9f4f216052ba664f22e9c504ef1	created	2017-11-06 13:12:34+00:00	NaT	
...	...	...	...	...	...	
995	3d3742a96f24a8fe4e2e57628807e476	b0e2c3d5cc9ecbf6a89693a15231ac7e	shipped	2017-04-16 06:31:07+00:00	2017-04-16 06:41:44+00:00	2017-04
996	bdf0dff2c16a74482f5ffce31586700e	ab98d2da5363a55f4c3fac106aa2ab8f	shipped	2017-04-09 17:51:22+00:00	2017-04-09 18:01:58+00:00	2017-04
997	d37d275bf870e1d9a50b888717929495	9f9a83d15d7255a3747dd9175b875ea0	shipped	2017-07-16 13:08:37+00:00	2017-07-16 13:22:58+00:00	2017-07
998	30c03e784f1c82b58bdbdf321132067f	e1892c28d1493eb6be3661e8daea168f	shipped	2017-07-16 14:44:13+00:00	2017-07-16 14:55:07+00:00	2017-07
999	6bcd251ea229d32710e7690cb3641ba6	ab3a9c34beacec9fb8bbc4e76765ee1e	shipped	2017-07-07 06:34:08+00:00	2017-07-11 04:15:19+00:00	2017-07

1000 rows × 8 columns

Observation

- contains all the details of the orders like start date , status , delivery date etc along with missing values.

In [ ]:
```
1
```

CUSTOMERS

The **customers.csv** contain following features:

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

In [23]:
```
1
2  QUERY = """
3
4  select * from `Target.customers`;
5
6
7
8    """
9
10 Query_Results = bigquery_client.query(QUERY)
11 Query_Results.to_dataframe()
12
```

Out[23]:

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
0	0735e7e4298a2ebbb46649346570476a	fcb003b1bdc0df64b4d065d9bb94f8c4	59650	acu	RN
1	903b3d86e3990db01619a4ebe3edef4e	46824822b15da44e983b021d0e945379	59650	acu	RN
2	38c97666e962d4fea7fd6a83e69f20cd	b6108acc674ae5c99e29adc1047d1049	59650	acu	RN
3	77c2f46cf580f4874c9a5751c2d88474	402cce5c0509000eed9e77fece8056e2	63430	ico	CE
4	4d3ef4cfffb8ad4767c199c36a4cfee6	6ba00666ab7eada5ceec279b259e44b5	63430	ico	CE
...	...	...	...	...	...
99436	0cba07be7f89557dbdf43a0e58663ff9	314e67684fb54502d819d66c0e3beb9b	25780	sao jose do vale do rio preto	RJ
99437	38efc316910029abb94b32e27de7bd11	c76f390a72ba18ccac9e4ae30a25b188	25780	sao jose do vale do rio preto	RJ
99438	bbc245da02e51a5ff7b595c31628af6b	9022903214ece785db32637ebda2f605	25780	sao jose do vale do rio preto	RJ
				sao jose do vale do rio	

Observations *[...]*

- contains the city name and the state name on which we can perform agg. functions to generate insights

ORDER REVIES

The **reviews.csv** contain following features:

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

In [13]:

```
QUERY = """

SELECT * FROM Target.order_reviews LIMIT 1000



  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[13]:

	review_id	order_id	review_score	review_comment_title	review_creation_date	review_answer_
0	be7e2989673cb2a147b87ba73277da9e	777c67eab7c0712ccde8ffbb22715adb	1	None	0001-04-17 00:00:00+00:00	0001-04-17 07:4
1	e12151267e4594d69eda14a871e2a1ab	4338a4463f7f9193d2a03a83a4b68ba0	1	None	0001-04-17 00:00:00+00:00	0001-04-17 09:0
2	41d614b133efebcd10352001bc024071	b8aaeda740b17cf925d3f2a13dcd1d7f	1	None	0001-04-17 00:00:00+00:00	0002-04-17 03:4
3	c950324a42c5796d06f569f77d8b2e88	b159d0ce7cd881052da94fa165617b05	1	None	0001-04-17 00:00:00+00:00	0001-04-17 10:2
4	76823ada94c8861ecebcfbc7c270853f	2a3007ed051b02a0e0dd0709c0ec9cd6	1	None	0001-04-17 00:00:00+00:00	0002-04-17 13:5
...	...	...	...	...	...	
95	b321ab5a94150664805153075900edf6	9cb197309d8105d9d407533fce50ae34	1	None	0031-08-18 00:00:00+00:00	0031-08-18 19:3
96	2c573bdf789cae8761dc8357cc6d2663	919baca007d9525b6668c18f79a33197	1	I didn't receive my product	0031-08-18 00:00:00+00:00	0031-08-18 15:4
97	f238b2f2738a38e3058911efccd46f9c	d4dcec44106e7301b362ee4b771b7ff3	1	You guys need to improve	0031-08-18 00:00:00+00:00	0001-09-18 00:0
98	7a11bf826668febba0800ec35884958c	52018484704db3661b98ce838612b507	1	Much FRÃ ¡ Gil	0031-08-18 00:00:00+00:00	0021-09-18 13:5
99	ddaab20f072344136b791751a563b92a	a4652eb1954cd0c3121a95ba49122673	1	None	0031-08-18 00:00:00+00:00	0002-09-18 19:0

)00 rows × 6 columns

Observations

- carries the review for each order
- the review comments etc

In [ ]:
```
1
```

PAYMENTS

In [25]:
```
QUERY = """

SELECT * FROM Target.payments LIMIT 1000



  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[25]:

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	1a57108394169c0b47d8f876acc9ba2d	2	credit_card	0	129.94
1	744bade1fcf9ff3f31d860ace076d422	2	credit_card	0	58.69
2	8bcbe01d44d147f901cd3192671144db	4	voucher	1	0.00
3	fa65dad1b0e818e3ccc5cb0e39231352	14	voucher	1	0.00
4	6ccb433e00daae1283ccc956189c82ae	4	voucher	1	0.00
...	...	...	...	...	...
995	20aa4f85b1ed8d98e5c742c1a003db9c	2	voucher	1	50.00
996	3bfd863aeb764ee13e7a3909ca8be2a5	2	voucher	1	50.00
997	c659935fdc19ef6b13aa33fb990a2f81	2	voucher	1	50.00
998	b5a1d012931f59ed905df201cf2aba80	2	voucher	1	50.00
999	0c1457493ca1492c4a4417fc278605b0	2	voucher	1	50.00

1000 rows × 5 columns

Observations

- contains the payment type and installments which can be used to talk about the economic status and financial stability of the people in Brazil and even with each and every region

In [ ]:
```
1
```

ORDER ITEMS

In [14]:
```
QUERY = """

SELECT * FROM Target.order_items LIMIT 1000



  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[14]:

	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freig
	e36e258656850b92657ac5f67b6d5	1	44d53f1240d6332232e4393c06500475	b64d51f0435e884e8de603b1655155ae	2018-07-09 13:31:36+00:00	3.0	
	f9ccaff7267fd0cf076e795b1fae8b69	1	44d53f1240d6332232e4393c06500475	b64d51f0435e884e8de603b1655155ae	2018-08-14 14:04:44+00:00	3.0	
	9bdf061e22288609201ec60deb42fb	1	5304ff3fa35856a156e1170a6022d34d	cf6f6bc4df3999b9c6440f124fb2f687	2017-05-12 19:05:20+00:00	3.5	
	193e64eb9a46b7f3197762f242b20a	1	98224bfc1eaadb3a394ec334c60453ff	ce616e1913288884e7742faac9d981db	2018-06-28 01:30:49+00:00	3.5	
	d6357ffe41aa6d2998852a710c70a0	1	98224bfc1eaadb3a394ec334c60453ff	ce616e1913288884e7742faac9d981db	2018-06-12 19:15:14+00:00	3.5	
	...	...	...	...	...	...	
	ce6f9094188f36623b250901635869	3	4a545fda8038a360bd93a5f1ac419a4e	9b013e03b2ab786505a1d3b5c0756754	2018-08-31 04:04:01+00:00	17.0	
	ef358c39e6d1537243a3ede175c8c0	1	51b24bbd390b66c9f67579153fe7ec5f	01fd077212124329bac32490e8ef80d9	2018-05-15 11:30:43+00:00	17.0	
	ef358c39e6d1537243a3ede175c8c0	2	51b24bbd390b66c9f67579153fe7ec5f	01fd077212124329bac32490e8ef80d9	2018-05-15 11:30:43+00:00	17.0	
	ef358c39e6d1537243a3ede175c8c0	3	51b24bbd390b66c9f67579153fe7ec5f	01fd077212124329bac32490e8ef80d9	2018-05-15 11:30:43+00:00	17.0	
	ef358c39e6d1537243a3ede175c8c0	4	51b24bbd390b66c9f67579153fe7ec5f	01fd077212124329bac32490e8ef80d9	2018-05-15 11:30:43+00:00	17.0	

s × 7 columns

Observations

- contains relevant information like the price of products and the frieght value which can be used to talk about the total amount of purchase the distance value between each purchase etc.

In [ ]:
```
1
```

PRODUCTS

In [24]:

```
QUERY = """

SELECT * FROM Target.products LIMIT 1000



  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[24]:

	product_id	product_category	product_name_length	product_description_length	product_photos_qty	product_weight_g
0	5eb564652db742ff8f28759cd8d2652a	None	<NA>	<NA>	<NA>	<NA>
1	09ff539a621711667c43eba6a3bd8466	babies	60	865	3	<NA>
2	2f763ba79d9cd987b2034aac7ceffe06	electronics	45	1198	2	595
3	a69f15dfb803d485e8933e80b9742c1c	Watches present	53	506	6	150
4	e1cfc87f543782b8a78b59fc8571df92	Garden tools	39	524	4	369
...	...	...	...	...	...	...
995	97afd188bb555122d11925deb0052174	sport leisure	42	578	1	350
996	c04ebe208ae260f39b376cd6ffbd0b14	telephony	57	578	2	200
997	96ff741d2c7879406ec3054d6b2a11c8	Watches present	49	578	6	200
998	2b3b11d70c32fbfc8e850fb68985a000	musical instruments	57	578	3	1965
999	baa1af9bc7467e619e2bb2876386ae55	sport leisure	49	578	2	1050

1000 rows × 9 columns

Observations

- tlaks about the product category , and the product details
- the product dimentsions can be used wrt to taking the idea of space consumptions and delivery vehicle and means of transportation of we can analyse further on that.

In [ ]:
```
1
```

SELLERS

In [17]:
```
QUERY = """

SELECT * FROM Target.sellers LIMIT 1000



  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[17]:

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	4be2e7f96b4fd749d52dff41f80e39dd	69900	rio branco	AC
1	327b89b872c14d1c0be7235ef4871685	69005	manaus	AM
2	4221a7df464f1fe2955934e30ff3a5a1	48602	bahia	BA
3	651530bf5c607240ccdd89a30c9c9712	44600	ipira	BA
4	2b402d5dc42554061f8ea98d1916f148	44900	irece	BA
...	...	...	...	...
995	b57e8460909fa137df7951b4a3b5ea84	90640	porto alegre	RS
996	48985c61529077fa4f1e38bcff0f2ed3	90020	porto alegre	RS
997	06e5eefc71ec47ae763c5c6f8db7064f	91350	porto alegre	RS
998	f1ed6bd0a9b11b581f16c851c6a5a527	90240	porto alegre	RS
999	a23266650e7c84bb93fbbba502137478	90220	porto alegre	RS

Observations

- the seller city and state along with customer table can be used to take insights on state specific revenue , connectivity etc.

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

- Data type of columns in a table
- Time period for which the data is given
- Cities and States covered in the dataset

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table
    2. Time period for which the data is given
    3. Cities and States covered in the dataset

Time period for which the data is given

can be found by finding the date time

In [48]:
```
QUERY = """



SELECT
    EXTRACT(YEAR FROM (MIN(order_purchase_timestamp))) AS Starting_Year,
    EXTRACT(MONTH FROM (MIN(order_purchase_timestamp))) AS First_Month,
    EXTRACT(DAY FROM (MIN(order_purchase_timestamp))) AS First_Day,

    EXTRACT(YEAR FROM (MAX(order_purchase_timestamp))) AS Ending_Year,
    EXTRACT(MONTH FROM (MAX(order_purchase_timestamp))) AS Last_Month,
    EXTRACT(DAY FROM (MAX(order_purchase_timestamp))) AS Last_Day
FROM Target.orders
LIMIT 1000



  """

bigquery_client.query(QUERY).to_dataframe()

```

Out[48]:

	Starting_Year	First_Month	First_Day	Ending_Year	Last_Month	Last_Day
0	2016	9	4	2018	10	17

- so the time duration started from 2016/9/4 and ended at 2018/10/17
- which shows that we can only infer details and make connections based on full 2 years.

In [ ]:
```
1
```

Cities and States covered in the dataset

- For this we can actually use customer table and seller table

In [668]:

```
QUERY = """



SELECT
  COUNT(DISTINCT customer_city) AS count_cities ,
  COUNT(DISTINCT customer_state) AS count_states

FROM Target.customers




  """

customer_city_states = bigquery_client.query(QUERY).to_dataframe()
customer_city_states
```

Out[668]:

	count_cities	count_states
0	4119	27

In [126]:

```
QUERY = """



SELECT
  customer_city,
  customer_state
FROM Target.customers




  """

customer_city_states = bigquery_client.query(QUERY).to_dataframe()
customer_city_states
```

Out[126]:

	customer_city	customer_state
0	acu	RN
1	acu	RN
2	acu	RN
3	ico	CE
4	ico	CE
...	...	...
99436	sao jose do vale do rio preto	RJ
99437	sao jose do vale do rio preto	RJ
99438	sao jose do vale do rio preto	RJ
99439	sao jose do vale do rio preto	RJ
99440	vila bela da santissima trindade	MT

99441 rows × 2 columns

In [67]:
```
c_city_list = set(customer_city_states.customer_city.values.tolist())
c_state_list = set(customer_city_states.customer_state.values.tolist())
```

In [118]:
```
c_city_list
```

Out[118]:
```
{'sao miguel do guama',
 'mutuipe',
 'sabino',
 'santo antonio das missoes',
 'pedro ii',
 'cajazeiras',
 'bias fortes',
 'alto sao joao',
 'seara',
 'igarape-acu',
 'valparaiso de goias',
 'grao para',
 'santo andre',
 'sao jose do calcado',
 'confins',
 'pirauba',
 'manoel ribas',
 'santa fe do sul',
 'canarana',
```

In [119]:
```
len(c_city_list)
```

Out[119]:
```
4119
```

In [69]:
```
1 c_state_list
```

Out[69]:
```
{'AC',
 'AL',
 'AM',
 'AP',
 'BA',
 'CE',
 'DF',
 'ES',
 'GO',
 'MA',
 'MG',
 'MS',
 'MT',
 'PA',
 'PB',
 'PE',
 'PI',
 'PR',
 'RJ',
 'RN',
 'RO',
 'RR',
 'RS',
 'SC',
 'SE',
 'SP',
 'TO'}
```

In [120]:
```
1 len(c_state_list)
```

Out[120]:
```
27
```

Observations

- There are 27 states inside the customer table which comprises of around 4119 cities.
- this shows the geographical span of Brazil and their retail market

In [ ]:

```
1
```

```
1
```

In [115]:

```
QUERY = """



SELECT
  COUNT(DISTINCT seller_city),
  COUNT(DISTINCT seller_state)

FROM Target.sellers




  """

seller_city_states = bigquery_client.query(QUERY).to_dataframe()
seller_city_states
```

Out[115]:

	f0_	f1_
0	611	23

In [64]:

```
QUERY = """



SELECT
  seller_city,
  seller_state
FROM Target.sellers




  """

seller_city_states = bigquery_client.query(QUERY).to_dataframe()
seller_city_states
```

Out[64]:

	seller_city	seller_state
0	rio branco	AC
1	manaus	AM
2	bahia	BA
3	ipira	BA
4	irece	BA
...	...	...
3090	vargem grande paulista	SP
3091	carapicuiba / sao paulo	SP
3092	marechal candido rondon	SP
3093	sao sebastiao da grama/sp	SP
3094	ribeirao preto / sao paulo	SP

3095 rows × 2 columns

In [92]:
```
s_city_list = set(seller_city_states.seller_city.values.tolist())
s_state_list = set(seller_city_states.seller_state.values.tolist())
```

In [93]:
```
s_city_list
```

Out[93]:
```
{'04482255',
 'abadia de goias',
 'afonso claudio',
 'aguas claras df',
 'alambari',
 'alfenas',
 'almirante tamandare',
 'alvares machado',
 'alvorada',
 'americana',
 'amparo',
 'ampere',
 'anapolis',
 'andira-pr',
 'andradas',
 'angra dos reis',
 'angra dos reis rj',
 'ao bernardo do campo',
 'aparecida',
```

In [121]:
```
len(s_city_list)
```

Out[121]:
```
611
```

In [94]:
```
1 s_state_list
```

Out[94]:
```
{'AC',
 'AM',
 'BA',
 'CE',
 'DF',
 'ES',
 'GO',
 'MA',
 'MG',
 'MS',
 'MT',
 'PA',
 'PB',
 'PE',
 'PI',
 'PR',
 'RJ',
 'RN',
 'RO',
 'RS',
 'SC',
 'SE',
 'SP'}
```

In [122]:
```
1 len(s_state_list)
```

Out[122]:
```
23
```

Observations

- there are 23 seller states which comprises of 611 cities from which the products are manufactured or sent to customers.
- we can see that the number of sellers cities are just 1/6th of the customer spread cities.
- if we had the population of each we could have talked about the saturation of sellers or surplus of them there by tweaking the market demand - supply in Brazil

In [140]:

```
QUERY = """



SELECT
  DISTINCT
  c.customer_state,
  c.customer_city,
  s.seller_city,
  s.seller_state

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.sellers` s
ON ot.seller_id = s.seller_id




  """

city_states = bigquery_client.query(QUERY).to_dataframe()
city_states
```

Out[140]:

	customer_state	customer_city	seller_city	seller_state
0	SE	aracaju	sao luis	MA
1	SE	aracaju	sao paulo	SP
2	SE	boquim	sao jose do rio preto	SP
3	AM	manaus	sao goncalo	RJ
4	RR	boa vista	ribeirao preto	SP
...	...	...	...	...
36766	TO	goianorte	goiania	GO

	customer_state	customer_city	seller_city	seller_state
36767	TO	conceicao do tocantins	ribeirao preto	SP
36768	TO	araguaina	guariba	SP
36769	TO	araguaina	campinas	SP
36770	TO	palmas	None	None

36771 rows × 4 columns

In [136]:
```
1 len(set(city_states.customer_state))
```

Out[136]: 27

In [137]:
```
1 len(set(city_states.customer_city))
```

Out[137]: 4119

In [138]:
```
1 len(set(city_states.seller_city))
```

Out[138]: 612

In [139]:
```
1 len(set(city_states.seller_state))
```

Out[139]: 24

Observations

- just conformed the data with pandas

2 In-depth Exploration:

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

2. In-depth Exploration:

    1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

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

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?

In [ ]:
```
1
```

- is the number of sellers and customers increasing with year ??
- is the number of orders being made increasing per year ??
- is the payment aggregate increased per year ?
- each customer id used multiple credit card purchase in a specific time period ??

1.is the number of sellers and customers increasing with year ??

In [208]:

```
QUERY = """


SELECT
    x2.customer_id,
    x2.seller_id,
    x2.Year_of_purchase,
    x2.CUSTOMER_COUNT,
    LAST_VALUE(CUSTOMER_COUNT) OVER (PARTITION BY x2.Year_of_purchase ORDER BY x2.Year_of_purchase
     RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS CUM_CUSTOMER_COUNT
FROM
    (SELECT
        x1.customer_id,
        x1.seller_id,
        x1.Year_of_purchase,
        RANK() OVER (PARTITION BY x1.Year_of_purchase ORDER BY x1.customer_id) AS CUSTOMER_COUNT
    FROM
        (SELECT

          c.customer_id,
          s.seller_id,
          order_purchase_timestamp,
          EXTRACT(YEAR from (order_purchase_timestamp)) AS Year_of_purchase

        FROM Target.orders o
        LEFT JOIN `Target.customers` c
        ON o.customer_id = c.customer_id
        LEFT JOIN `Target.order_items` ot
        ON o.order_id = ot.order_id
        LEFT JOIN `Target.sellers` s
        ON ot.seller_id = s.seller_id) AS x1) AS x2


  """
Query_Results = bigquery_client.query(QUERY)
Each_year_Purchases = Query_Results.to_dataframe()
Each_year_Purchases
```

Out[208]:

	customer_id	seller_id	Year_of_purchase	CUSTOMER_COUNT	CUM_CUSTOMER_COUNT
0	00474d2582fd72663036795b7ab8cfc1	cca3071e3e9bb7d12640c9fbe2301306	2016	1	387
1	00474d2582fd72663036795b7ab8cfc1	cca3071e3e9bb7d12640c9fbe2301306	2016	1	387
2	01415cfeb907d8ce0e17075b4c097fe9	c43d924a0f1688ee9fae0efcd6f539d4	2016	3	387
3	01be2c0c2d55c597b5ec011e26e4b0d1	620c87c171fb2a6dd6e8bb4dec959fc6	2016	4	387
4	01f7b7a4e25cda9ce48c0f7263f7452d	ce27a3cc3c8cc1ea79d11e561e9bebb6	2016	5	387
...	...	...	...	...	...
113420	fffa0238b217e18a8adeeda0669923a3	093805f8f2aeb63881444571e1f48f30	2017	51382	51386
113421	fffc22669ca576ae3f654ea64c8f36be	None	2017	51383	51386
113422	ffffa3172527f765de70084a7e53aae8	23613d49c3ac2bd302259e55c06c050c	2017	51384	51386
113423	ffffa3172527f765de70084a7e53aae8	23613d49c3ac2bd302259e55c06c050c	2017	51384	51386
113424	ffffe8b65bbe3087b653a978c870db99	None	2017	51386	51386

113425 rows × 5 columns

In [214]:
```
1 Each_year_Purchases['CUM_CUSTOMER_COUNT'].unique()
```

Out[214]:
```
<IntegerArray>
[387, 61652, 51386]
Length: 3, dtype: Int64
```

In [222]:
```
1 Each_year_Purchases.Year_of_purchase.value_counts()
```

Out[222]:
```
2018    61652
2017    51386
2016      387
Name: Year_of_purchase, dtype: Int64
```

Observations

- there is an increse in the number of customers over the years.
- it signifies that the e-commerece in Brazil in ol the roll and the profit , inhouse production and IT growth assiciated with it in on a growth.

In [ ]:
```
1
```

```
1 2.is the number of orders being made increasing per year ??
```

In [ ]:
```
1
```

Can we see some seasonality with peaks at specific months?

In [674]:

```
QUERY = """


SELECT
  x.Month_year_purchase_date,
  COUNT(x.Month_year_purchase_date) AS count_purchases_in_each_month
FROM
(SELECT

  c.customer_id,
  s.seller_id,
  customer_city,
  seller_city,
  FORMAT_DATE("%b %Y", DATE (order_purchase_timestamp)) AS Month_year_purchase_date

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.sellers` s
ON ot.seller_id = s.seller_id
) AS x
GROUP BY x.Month_year_purchase_date
ORDER BY count_purchases_in_each_month




  """
Query_Results = bigquery_client.query(QUERY)
season_purchase = Query_Results.to_dataframe()
season_purchase
```

Out[674]:

	Month_year_purchase_date	count_purchases_in_each_month
0	Dec 2016	1
1	Oct 2018	4

	Month_year_purchase_date	count_purchases_in_each_month
2	Sep 2016	7
3	Sep 2018	16
4	Oct 2016	379
5	Jan 2017	966
6	Feb 2017	1998
7	Apr 2017	2697
8	Mar 2017	3041
9	Jun 2017	3611
10	May 2017	4176
11	Jul 2017	4576
12	Sep 2017	4873
13	Aug 2017	4948
14	Oct 2017	5385
15	Dec 2017	6357
16	Jun 2018	7085
17	Jul 2018	7111
18	Aug 2018	7308
19	Feb 2018	7706
20	May 2018	7945
21	Apr 2018	7980
22	Mar 2018	8240
23	Jan 2018	8257
24	Nov 2017	8758

In [278]:
```
1 season_purchase.amout_purchases_in_each_month.sum()
```

Out[278]: 113425

Observations

- In the data we can see that there is consistant increase in the nuber of orders along side the year and each month.
- the most interesting fact is the largest number of orders are around Nov 2017 and Jan 2018 which shows the importance of Festive season like Christmas and New year in Brazil.
- capitalising on those will be great revenue generating opportunity.

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

In [159]:

```
QUERY = """


SELECT
  customer_id,
  order_purchase_timestamp,
  EXTRACT(HOUR FROM (order_purchase_timestamp)) AS Hour,
    CASE
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 6 THEN 'Dawn'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp))  <= 12  THEN 'Morning'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 18 THEN 'Afternoon'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 24 THEN 'Night'
    END AS order_timing
FROM Target.orders;

  """

Query_Results = bigquery_client.query(QUERY)
order_timing = Query_Results.to_dataframe()
order_timing
```

Out[159]:

	customer_id	order_purchase_timestamp	Hour	order_timing
0	58803ced08e52f5e3b028ef81d8c9563	2018-03-05 03:47:11+00:00	3	Dawn
1	fd53366df6136213ab393be0c5e2802f	2018-01-25 03:24:05+00:00	3	Dawn
2	45ead47bd4ee6bb5b5108506c44a4a7a	2018-01-18 04:37:44+00:00	4	Dawn
3	827d7240a35889668129a7b10aa0081b	2017-02-08 05:56:31+00:00	5	Dawn
4	b879ae1c5f704d2571496b0f0fb8c47c	2018-04-21 03:25:21+00:00	3	Dawn
...	...	...	...	...
99436	86c608f34ba6e73337c7fb6bb4db336b	2017-03-21 23:15:29+00:00	23	Night
99437	ad8939b7dfebf1e59e4145ed73bb0e33	2017-08-13 23:23:00+00:00	23	Night
99438	b8cbb2fa6090f51abb6076e2b26ca3ae	2017-11-09 23:51:34+00:00	23	Night
99439	4956cfa41268f0de55f989555c81ddde	2018-01-01 23:50:58+00:00	23	Night
99440	fc0f896db68de04d75532460d7c4b244	2017-03-07 23:02:58+00:00	23	Night

99441 rows × 4 columns

In [675]:
```
1 order_timing.order_timing.value_counts()
```

Out[675]:
```
Afternoon    38135
Night        28331
Morning      27733
Dawn          5242
Name: order_timing, dtype: int64
```

Observations

- The Brazilian people have bought products more in the after noon followed by night and then morning.
- so the advertising and push notification tema can focus more on these timings as they can be very active and motivated arounf afternoon.

Confirming the values obtained from query with sql count

In [168]:

```
QUERY = """


SELECT
  COUNT(x.customer_id)
FROM
(SELECT
  customer_id,
  order_purchase_timestamp,
  EXTRACT(HOUR FROM (order_purchase_timestamp)) AS Hour,
    CASE
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 6 THEN 'Dawn'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp))  <= 12  THEN 'Morning'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 18 THEN 'Afternoon'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 24 THEN 'night'
    END AS order_timing
FROM Target.orders) AS x
WHERE x.order_timing = 'Dawn'


  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[168]:

	f0_
0	5242

In [169]:

```
QUERY = """


SELECT
  COUNT(x.customer_id)
FROM
(SELECT
  customer_id,
  order_purchase_timestamp,
  EXTRACT(HOUR FROM (order_purchase_timestamp)) AS Hour,
    CASE
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 6 THEN 'Dawn'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp))  <= 12  THEN 'Morning'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 18 THEN 'Afternoon'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 24 THEN 'night'
    END AS order_timing
FROM Target.orders) AS x
WHERE x.order_timing = 'Morning'


  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[169]:

	f0_
0	27733

In [ ]:

```
1
```

In [170]:

```
QUERY = """


SELECT
  COUNT(x.customer_id)
FROM
(SELECT
  customer_id,
  order_purchase_timestamp,
  EXTRACT(HOUR FROM (order_purchase_timestamp)) AS Hour,
    CASE
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 6 THEN 'Dawn'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp))  <= 12  THEN 'Morning'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 18 THEN 'Afternoon'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 24 THEN 'night'
    END AS order_timing
FROM Target.orders) AS x
WHERE x.order_timing = 'Afternoon'


  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[170]:

	f0_
0	38135

In [ ]:

```
1
```

In [171]:
```
QUERY = """


SELECT
  COUNT(x.customer_id)
FROM
(SELECT
  customer_id,
  order_purchase_timestamp,
  EXTRACT(HOUR FROM (order_purchase_timestamp)) AS Hour,
    CASE
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 6 THEN 'Dawn'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp))  <= 12  THEN 'Morning'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 18 THEN 'Afternoon'
      WHEN EXTRACT(HOUR FROM (order_purchase_timestamp)) <= 24 THEN 'night'
    END AS order_timing
FROM Target.orders) AS x
WHERE x.order_timing = 'night'


  """

Query_Results = bigquery_client.query(QUERY)
Query_Results.to_dataframe()

```

Out[171]:

	f0_
0	28331

In [215]:
```
order_timing.order_timing.value_counts()
```

Out[215]:
```
Afternoon    38135
Night        28331
Morning      27733
Dawn          5242
Name: order_timing, dtype: int64
```

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

- Get month on month orders by region, states

- How are customers distributed in Brazil

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

    1. Get month on month orders by region, states

    2. How are customers distributed in Brazil

In [ ]:
```
1
```

Get month on month orders by region, states

In [286]:

```
QUERY = """


SELECT
  COUNT(o.order_id) AS counter,
  o.order_id,
  c.customer_id,
  customer_state,
  customer_city,
  FORMAT_DATE("%b %Y", DATE (order_purchase_timestamp)) AS Month_year_purchase_date

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.sellers` s
ON ot.seller_id = s.seller_id
GROUP BY customer_id , customer_state , customer_city , Month_year_purchase_date, order_id
ORDER BY counter DESC


  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[286]:

	counter	order_id	customer_id	customer_state	customer_city	Month_year_purchase_dat
0	21	8272b63d03f5f79c56e9e4120aec44ef	fc3d1daec319d62d49bfb5e1f83123e9	SP	sao paulo	Jul 201
1	20	ab14fdcfbe524636d65ee38360e22ce8	bd5d39761aa56689a265d95d8d32b8be	GO	goiania	Aug 201
2	20	1b15974a0141d54e36626dca3fdc731a	be1b70680b9f9694d8c70f41fa3dc92b	SP	sao paulo	Feb 201
3	15	9ef13efd6949e4573a18964dd1bbe7f5	adb32467ecc74b53576d9d13a5a55891	GO	goiania	Jan 201
4	15	428a2f660dc84138d969ccd69a0ab6d5	10de381f8a8d23fff822753305f71cae	PR	uniao da vitoria	Nov 201
...	...	...	...	...	...	.
99436	1	ebb921d271b8b17989e08220637b921f	9ad3cb19919815536d4ae2c46d69279c	SP	indaiatuba	Jan 201

	counter	order_id	customer_id	customer_state	customer_city	Month_year_purchase_dat
99437	1	33e13c48e5883f2a3ae2269bf57650c7	cef54c3ad569af033c4ecdaa69c9e140	MG	goiana	Dec 201
99438	1	ad9a8214948a5bbd4fa03af2ea598a20	921e7175793c295e163d9bd7c8b1c1ee	SC	alto bela vista	May 201
99439	1	24f686e03e134195c83dc3e2b6bf4cd0	8feae4bd81cd90e377f43588b900d3c2	MG	tres coracoes	May 201
99440	1	9a31fd9d697e9670777501f720773fd9	c0a4ca941f313300f1694f4ab73b3651	RO	porto velho	Feb 201

99441 rows × 6 columns

Observations

- we can see that for each month and each region the number of orders in counter.
- since there are just 611 seller cities compared to 4000 customer cities from which the orders come from , we can increase the nuber of sellers inorder to handle more diverse orders especially in cities which has more traffic.

In [ ]:
```
1
```

How are customers distributed in Brazil

- based on Product category
- based on the payment type

In [305]:

```
QUERY = """


SELECT

  p.product_category,
  SUM(ot.price) AS Aggregate_expenditure_on_item,
  COUNT(o.order_id) AS Amount_of_customers

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.products` p
ON ot.product_id = p.product_id
GROUP BY product_category
ORDER BY Amount_of_customers DESC




  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type
```

	product_category	Aggregate_expenditure_on_item	Amount_of_customers
0	bed table bath	1036988.68	11115
1	HEALTH BEAUTY	1258681.34	9670
2	sport leisure	988048.97	8641
3	Furniture Decoration	729762.49	8334
4	computer accessories	911954.32	7827
...	...	...	...
69	cds music dvds	730.00	14

	product_category	Aggregate_expenditure_on_item	Amount_of_customers
70	La Cuisine	2054.99	14
71	PC Gamer	1545.95	9
72	Fashion Children's Clothing	569.85	8
73	insurance and services	283.29	2

Observations

- Brazilian people spend more on bed table bath , then beauty products and sports leisure the most.
- in the top 3 there are 3 items related to aesthetics and self care.
- tapping the potential for that will be more profitable considering the fact that the e-commerce is on a rise.

In [308]:

```
QUERY = """


SELECT

  p.payment_type,
  SUM(ot.price) AS Aggregate_expenditure_on_item,
  COUNT(o.order_id) AS Amount_of_customers

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.payments` p
ON o.order_id = p.order_id
GROUP BY p.payment_type
ORDER BY Amount_of_customers DESC




  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type
```

Out[308]:

	payment_type	Aggregate_expenditure_on_item	Amount_of_customers
0	credit_card	1.097436e+07	87286
1	UPI	2.391526e+06	23037
2	voucher	6.594736e+05	6407
3	debit_card	1.837587e+05	1698
4	not_defined	NaN	3
5	None	1.349700e+02	3

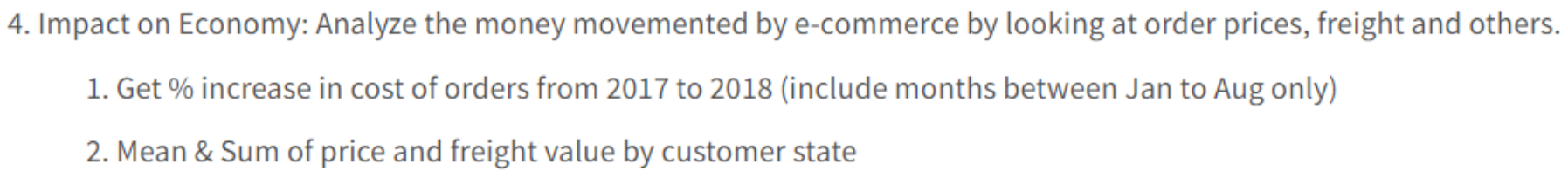
Observation

- we can see that 80 percent of people has used credit card as the mode of payment
- the value generated from it around 10 times more than than the very next that is UPI.
- from this we can say that the people are spontaneous spenders who repay it with long term basis.
- the chances that people think twice before buying is less with credit card purchases.
- In hand cash purchases are very small therefore the technological advancements should be high and also upi and methods are reaaching even the minute pockets of the country there by increasing the connectivity

4 Impact on Economy: Analyze the money movemented by e-commerce by looking at order prices, freight and others.

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

Mean & Sum of price and freight value by customer state



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

In [702]:

```
QUERY = """



SELECT
    xx.year,
    xx.months,
    xx.total_costs,
    100*(xx.total_costs-xx.lag)/xx.lag as pct_increased

FROM
    (SELECT
        x.year,
        x.months,
        x.total_costs,
        lag(total_costs) over(order by year, months) lag
    FROM(
        SELECT
        extract(year from shipping_limit_date) year,
        extract(month from shipping_limit_date) months,
        sum(price) as total_costs
FROM `Target.order_items`
GROUP BY year, months)x

WHERE (x.year = 2017 and x.months >= 1) or (x.year = 2018 and x.months <= 8)
ORDER BY year, months) xx

  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type
```

Out[702]:

	year	months	total_costs	pct_increased
0	2017	1	80124.74	NaN
1	2017	2	245982.01	206.998825
2	2017	3	343242.60	39.539717
3	2017	4	308148.32	-10.224337

	year	months	total_costs	pct_increased
4	2017	5	505655.46	64.094829
5	2017	6	469001.05	-7.248890
6	2017	7	465282.10	-0.792951
7	2017	8	560093.34	20.377152
8	2017	9	617046.12	10.168444
9	2017	10	658020.23	6.640364
10	2017	11	883351.63	34.243841
11	2017	12	898920.98	1.762531
12	2018	1	845279.18	-5.967354
13	2018	2	819228.85	-3.081861
14	2018	3	1030527.84	25.792425
15	2018	4	958729.70	-6.967123
16	2018	5	1084358.55	13.103678
17	2018	6	870736.84	-19.700284
18	2018	7	811132.08	-6.845324
19	2018	8	1072147.70	32.179176

Observations

- Mostly percentage change of total price gets negative on april, june and july which is cyclical. 2017 Jan - 2017 Feb sees the highest percentage increase in overall time span.

In [ ]:
```
1
```

Mean & Sum of price and freight value by customer state

In [ ]:
```
1
```

In [319]:

```
QUERY = """



SELECT
    *,
    ROUND(((x.price + x.freight_value) * x.quantity),2) as total_amount_payable
FROM
(SELECT

      DISTINCT
      order_id,
      seller_id,
      product_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
ORDER BY quantity DESC


  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type
```

Out[319]:

	order_id	seller_id	product_id	shipping_limit_date	price	frei
0	1b15974a0141d54e36626dca3fdc731a	8e6d7754bc7e0f22c96d255ebda59eba	ee3d532c8a438679776d222e997606b3	2018-03-01 02:50:48+00:00	100.00	
1	ab14fdcfbe524636d65ee38360e22ce8	ce27a3cc3c8cc1ea79d11e561e9bebb6	9571759451b1d780ee7c15012ea109d4	2017-08-30 14:30:23+00:00	98.70	
2	428a2f660dc84138d969ccd69a0ab6d5	f326006815956455b2859abd58fe7e39	89b190a046022486c635022524a974a8	2017-11-30 10:30:51+00:00	65.49	
3	9ef13efd6949e4573a18964dd1bbe7f5	0b36063d5818f81ccb94b54adfaebbf5	37eb69aca8718e843d897aa7b82f462d	2017-02-03 21:44:49+00:00	51.00	

	order_id	seller_id	product_id	shipping_limit_date	price	freig
4	9bdc4d4c71aa1de4606060929dee888c	e7d5b006eb624f13074497221eb37807	44a5d24dd383324a421569ca697b13c2	2018-02-28 11:48:12+00:00	29.99	
...	...	...	...	...	...	
102420	e93fe3b7d6cf93d390620d65c8d98e68	5f5b43b2bffa8656e4bc6efeb13cc649	d0688f07b0aa9bfd7535de718425d8f5	2017-12-22 12:45:26+00:00	59.00	
102421	ebe6eddebae955b95268b2c70cdc019e	4a3ca9315b744ce9f8e9374361493884	84f456958365164420cfc80fbe4c7fab	2018-02-12 12:56:18+00:00	92.00	
102422	ee24e7d976a47cc46be7304fe5654ae9	6560211a19b47992c3666cc44a7e94c0	6c50c87e36d8641d67fbcceed5cfec5e	2018-08-02 10:24:28+00:00	55.00	
102423	f55dcacbda42aa97458735683d9fbb76	8b9d6eec4a7eb7d0f9d579ce0b38324d	2ff995aead9c63a1f37a07b3664ead37	2018-08-09 15:10:23+00:00	74.99	
102424	fc9cb1d496b03a40d603876228b48bff	955fee9216a65b617aa5c0531780ce60	54d9ac713e253fa1fae9c8003b011c2a	2018-05-08 23:09:56+00:00	35.00	

102425 rows × 8 columns

In [ ]:
```
1
```

In [323]:

```
QUERY = """



SELECT
    *,
    ROUND(((x.price + x.freight_value) * x.quantity),2) as total_amount_payable
FROM
(SELECT

      DISTINCT
      c.customer_city,
      c.customer_state,
      o.order_id,
      ot.seller_id,
      ot.product_id,
      ot.shipping_limit_date,
      ot.price,
      ot.freight_value,
      COUNT(product_id) OVER (PARTITION BY o.order_id, product_id) AS quantity

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id) as x
ORDER BY quantity DESC


  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type
```

Out[323]:

	customer_city	customer_state	order_id	seller_id	product_i
0	goiania	GO	ab14fdcfbe524636d65ee38360e22ce8	ce27a3cc3c8cc1ea79d11e561e9bebb6	9571759451b1d780ee7c15012ea109d

	customer_city	customer_state	order_id	seller_id	product_i
1	sao paulo	SP	1b15974a0141d54e36626dca3fdc731a	8e6d7754bc7e0f22c96d255ebda59eba	ee3d532c8a438679776d222e997606b
2	goiania	GO	9ef13efd6949e4573a18964dd1bbe7f5	0b36063d5818f81ccb94b54adfaebbf5	37eb69aca8718e843d897aa7b82f462
3	uniao da vitoria	PR	428a2f660dc84138d969ccd69a0ab6d5	f326006815956455b2859abd58fe7e39	89b190a046022486c635022524a974a
4	santos	SP	73c8ab38f07dc94389065f7eba4f297a	1f50f920176fa81dab994f9023523100	422879e10f46682990de24d770e7f83
...	...	...	...	...	.
103195	goiania	GO	1770ee9dc600eccda7bfb6c56562b91d	None	Non
103196	brasilia	DF	c037dd9db2ac27aa44381219612e91ba	None	Non
103197	garibaldi	RS	f0eadb6ba0382197c2f94afe7d95e60e	None	Non
103198	sao paulo	SP	6111350dd1f8b825ac82e9caea339f07	None	Non
103199	sao joao de meriti	RJ	0a5c74ccc786ced7903270de9d6c170a	None	Non

103200 rows × 10 columns

Observations

- the average customer to seller freight value increases as the distance increases.
- since the distance of the order in not given its difficult to double check this inference but the frieght value is increasing for big orders .
- this erratic behaviour can be associated the size of the product , quantity or the distance between the seller city and customer city

In [ ]:
```
1
```

5 Analysis on sales, freight and delivery time

- Calculate days between purchasing, delivering and estimated delivery
- Create columns:

    - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
    - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
- Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery
- Sort the data to get the following:
    - Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
    - Top 5 states with highest/lowest average time to delivery
    - Top 5 states where delivery is really fast/ not so fast compared to estimated date

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery
2. Create columns:
    - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
    - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery
4. Sort the data to get the following:
    1. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
    2. Top 5 states with highest/lowest average time to delivery
    3. Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
1
```

In [ ]:
```
1
```

Calculate days between purchasing, delivering and estimated delivery / time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

In [385]:

```
QUERY = """


SELECT

    order_id,
    order_status,
    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,
    date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) AS day_exceeded_for_delivery


FROM `Target.orders`
WHERE order_status = 'delivered'


  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[385]:

	order_id	order_status	days_taken_to_delivery	estimated_time_to_delivery	day_exceeded_for_delivery
0	635c894d068ac37e6e03dc54eccb6189	delivered	30	32	-1
1	3b97562c3aee8bdedcb5c2e45a50d5e1	delivered	32	33	0
2	68f47f50f04c4cb6774570cfde3a9aa7	delivered	29	31	-1
3	276e9ec344d3bf029ff83a161c6b3ce9	delivered	43	39	4
4	54e1a3c2b97fb0809da548a59f64c813	delivered	40	36	4
...	...	...	...	...	...
96473	ebca4856d6b3b849437fe99d11633d25	delivered	28	22	6
96474	a6f521d5e68e95961ac13d448a960fd7	delivered	28	34	-5
96475	f41397c4cf4c8a67f5f540472acbfe4f	delivered	28	22	5
96476	b674e463ea07d8a8fac9951be50283f1	delivered	28	18	10
96477	cd317fc34a4ea25ea8d1743f003712ac	delivered	28	16	11

96478 rows × 5 columns

In [692]:

```
QUERY = """


SELECT
    *,
    CASE
        WHEN day_exceeded_for_delivery >= 0
        THEN 'order_delayed'
        ELSE 'on_time'
    END AS delivery_status
FROM
(SELECT

    order_id,
    order_status,
    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,
    date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) AS day_exceeded_for_delivery


FROM `Target.orders`
WHERE order_status = 'delivered')


  """
Query_Results = bigquery_client.query(QUERY)
oder_delivery_delay = Query_Results.to_dataframe()
oder_delivery_delay
```

Out[692]:

	order_id	order_status	days_taken_to_delivery	estimated_time_to_delivery	day_exceeded_for_delivery	delivery_s
0	b60b53ad0bb7dacacf2989fe27ad567a	delivered	12	7	5	order_de
1	276e9ec344d3bf029ff83a161c6b3ce9	delivered	43	39	4	order_de
2	1a0b31f08d0d7e87935b819ede91a321	delivered	6	36	-29	on
3	cec8f5f7a13e5ab934a486ec9eb713c8	delivered	20	61	-40	on
4	54e1a3c2b97fb0809da548a59f64c813	delivered	40	36	4	order_de
...	...	...	...	...	...	

	order_id	order_status	days_taken_to_delivery	estimated_time_to_delivery	day_exceeded_for_delivery	delivery_s
96473	7c67e69218256034fdfb7f4507c85906	delivered	7	9	-1	on
96474	6ece547d00a145a6d65f564467f72d22	delivered	22	23	-1	on
96475	0b7acd2a88da22f2cf5365bff495f1e6	delivered	13	15	-1	on

Observations

- we can see that in most of the times the order was delivered on time which is rmearkable but the amount of time its delayed needs to be reduced.
- when the oreders are delayed there is a chance that people write bad reviews and give poor strs which may affect the sales and seller in effect.

In [693]:
```
1 oder_delivery_delay.delivery_status.value_counts()
```

Out[693]:
```
on_time          87190
order_delayed     9288
Name: delivery_status, dtype: int64
```

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

In [407]:

```
QUERY = """


SELECT
    *,
    CASE
        WHEN day_exceeded_for_delivery >= 0
        THEN 'order_delayed'
        ELSE 'on_time'
    END AS delivery_status
FROM
(
SELECT
    c.customer_city,
    o.order_id,
    o.order_status,
    freight_value,
    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,
    date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) AS day_exceeded_for_delivery


FROM `Target.orders` o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
WHERE order_status = 'delivered'
) AS x


  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[407]:

	customer_city	order_id	order_status	freight_value	days_taken_to_delivery	estimated_time_to_delivery	day_ex
0	sao paulo	b60b53ad0bb7dacacf2989fe27ad567a	delivered	9.30	12	7	

	customer_city	order_id	order_status	freight_value	days_taken_to_delivery	estimated_time_to_delivery	day_ex
1	boa viagem	276e9ec344d3bf029ff83a161c6b3ce9	delivered	30.94	43	39	
2	sao paulo	1a0b31f08d0d7e87935b819ede91a321	delivered	15.65	6	36	
3	rio de janeiro	cec8f5f7a13e5ab934a486ec9eb713c8	delivered	14.52	20	61	
4	rio de janeiro	cec8f5f7a13e5ab934a486ec9eb713c8	delivered	14.52	20	61	
...	...	...	...	...	...	...	
110192	marica	6ece547d00a145a6d65f564467f72d22	delivered	17.63	22	23	
110193	lorena	0b7acd2a88da22f2cf5365bff495f1e6	delivered	13.85	13	15	
110194	lorena	0b7acd2a88da22f2cf5365bff495f1e6	delivered	13.85	13	15	
110195	uberaba	56329c4faab8b220947aaf471b09e4a0	delivered	18.83	14	16	
110196	juiz de fora	999b8d758771edb14fdf3f1a280d8a60	delivered	15.10	18	20	

110197 rows × 8 columns

In [ ]:
```
1
```

Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5 /Top 5 states with highest/lowest average time to delivery /Top 5 states where delivery is really fast/ not so fast compared to estimated date

In [421]:

```
QUERY = """


SELECT
    c.customer_city,
    c.customer_state,
    AVG(freight_value) AS avg_freight_val_per_state


FROM `Target.orders` o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
GROUP BY customer_city,customer_state
ORDER BY avg_freight_val_per_state DESC
LIMIT 6




  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[421]:

	customer_city	customer_state	avg_freight_val_per_state
0	itupiranga	PA	203.38
1	amarante	PI	193.84
2	almino afonso	RN	170.11
3	canapi	AL	147.32
4	marilac	MG	142.49
5	sao martinho	RS	142.33

In [ ]:

```
1
```

Top 5 states with highest/lowest average time to delivery

In [458]:

```
QUERY = """


SELECT
    x.customer_state,
    SUM(days_taken_to_delivery) / COUNT(x.order_id) AS per_state_average_time_to_delivery,
    SUM(estimated_time_to_delivery) / COUNT(x.order_id) AS per_state_estimated_time_to_delivery

FROM
(
SELECT
    o.order_id,
    c.customer_state,
    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,
    date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) AS day_exceeded_for_delivery


FROM `Target.orders` o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
) AS x
GROUP BY customer_state
ORDER BY per_state_average_time_to_delivery
LIMIT 10

  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[458]:

	customer_state	per_state_average_time_to_delivery	per_state_estimated_time_to_delivery
0	SP	8.021769	18.896801
1	PR	11.207016	24.378780
2	MG	11.251589	24.307640
3	DF	12.160677	24.195787

	customer_state	per_state_average_time_to_delivery	per_state_estimated_time_to_delivery
4	SC	14.164961	25.508212
5	RJ	14.165655	26.085350
6	RS	14.389217	28.318392
7	GO	14.508525	26.647485
8	MS	14.850909	25.678788
9	ES	14.931095	25.280477

In [ ]:
```
1
```

In [455]:

```
QUERY = """


SELECT
    x.customer_state,
    SUM(days_taken_to_delivery) / COUNT(x.order_id) AS per_state_average_time_to_delivery,
    SUM(estimated_time_to_delivery) / COUNT(x.order_id) AS per_state_estimated_time_to_delivery

FROM
(
SELECT
    o.order_id,
    c.customer_state,
    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,
    date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) AS day_exceeded_for_delivery


FROM `Target.orders` o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
) AS x
GROUP BY customer_state
ORDER BY per_state_average_time_to_delivery DESC
LIMIT 10

  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[455]:

	customer_state	per_state_average_time_to_delivery	per_state_estimated_time_to_delivery
0	AP	27.414634	45.487805
1	AM	25.493976	45.114458
2	RR	24.615385	45.980769

	customer_state	per_state_average_time_to_delivery	per_state_estimated_time_to_delivery
3	AL	22.970852	32.210762
4	PA	22.635945	36.941014
5	MA	20.412756	30.500602
6	SE	20.171795	30.305128
7	AC	20.108696	40.695652
8	CE	19.694687	30.962340
9	PB	19.455446	32.528053

Observation

- we can see that all the states have a very efficint delivery system on an average.
- this shows the efficiency of the retail company and connectivity in Braxil

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

In [465]:

```
QUERY = """


SELECT
    x.customer_state,
    SUM(days_taken_to_delivery) / COUNT(x.order_id) AS per_state_average_time_to_delivery,
    SUM(estimated_time_to_delivery) / COUNT(x.order_id) AS per_state_estimated_time_to_delivery,
    ((SUM(estimated_time_to_delivery) / COUNT(x.order_id)) - SUM(days_taken_to_delivery) / COUNT(x.order_id)) AS exp

FROM
(
SELECT
    o.order_id,
    c.customer_state,
    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,
    date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) AS day_exceeded_for_delivery


FROM `Target.orders` o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
) AS x
GROUP BY customer_state
ORDER BY expected_Vs_Real
LIMIT 10

  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[465]:

	customer_state	per_state_average_time_to_delivery	per_state_estimated_time_to_delivery	expected_Vs_Real
0	AL	22.970852	32.210762	9.239910
1	MA	20.412756	30.500602	10.087846

	customer_state	per_state_average_time_to_delivery	per_state_estimated_time_to_delivery	expected_Vs_Real
2	SE	20.171795	30.305128	10.133333
3	ES	14.931095	25.280477	10.349382
4	MS	14.850909	25.678788	10.827879
5	SP	8.021769	18.896801	10.875031
6	BA	18.096572	29.134520	11.037948
7	CE	19.694687	30.962340	11.267653
8	SC	14.164961	25.508212	11.343252
9	PI	18.200368	29.937500	11.737132

In [466]:

```
QUERY = """


SELECT
    x.customer_state,
    SUM(days_taken_to_delivery) / COUNT(x.order_id) AS per_state_average_time_to_delivery,
    SUM(estimated_time_to_delivery) / COUNT(x.order_id) AS per_state_estimated_time_to_delivery,
    ((SUM(estimated_time_to_delivery) / COUNT(x.order_id)) - SUM(days_taken_to_delivery) / COUNT(x.order_id)) AS exp

FROM
(
SELECT
    o.order_id,
    c.customer_state,
    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,
    date_diff(order_delivered_customer_date, order_estimated_delivery_date,day) AS day_exceeded_for_delivery


FROM `Target.orders` o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
) AS x
GROUP BY customer_state
ORDER BY expected_Vs_Real DESC
LIMIT 10

  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[466]:

	customer_state	per_state_average_time_to_delivery	per_state_estimated_time_to_delivery	expected_Vs_Real
0	RR	24.615385	45.980769	21.365385
1	AC	20.108696	40.695652	20.586957

	customer_state	per_state_average_time_to_delivery	per_state_estimated_time_to_delivery	expected_Vs_Real
2	RO	18.535211	38.697183	20.161972
3	AM	25.493976	45.114458	19.620482
4	AP	27.414634	45.487805	18.073171
5	MT	17.144476	31.539188	14.394712
6	PA	22.635945	36.941014	14.305069
7	RS	14.389217	28.318392	13.929175
8	RN	18.483083	32.225564	13.742481
9	PE	17.162983	30.827072	13.664088

Observation

- All the states have a delivery status of atleast a week before. This builds the reputation and trust in customers.

6 Payment type analysis:

- Month over Month count of orders for different payment types
- Distribution of payment installments and count of orders

6. Payment type analysis:

1. Month over Month count of orders for different payment types

2. Distribution of payment installments and count of orders

Month over Month count of orders for different payment types

In [536]:

```
QUERY = """


SELECT
  FORMAT_DATE("%b %Y", DATE (order_purchase_timestamp)) AS Month_year_purchase_date,
  p.payment_type,
  COUNT(o.order_id) AS Amount_of_customers

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.payments` p
ON o.order_id = p.order_id
GROUP BY p.payment_type ,Month_year_purchase_date
ORDER BY Amount_of_customers DESC









  """
Query_Results = bigquery_client.query(QUERY)
payment_type_per_month = Query_Results.to_dataframe()
payment_type_per_month.head(50)
```

Out[536]:

	Month_year_purchase_date	payment_type	Amount_of_customers
0	Nov 2017	credit_card	6816
1	Mar 2018	credit_card	6473
2	May 2018	credit_card	6323
3	Jan 2018	credit_card	6244
4	Apr 2018	credit_card	6234
5	Feb 2018	credit_card	5985
6	Aug 2018	credit_card	5609

	Month_year_purchase_date	payment_type	Amount_of_customers
7	Jun 2018	credit_card	5537
8	Jul 2018	credit_card	5341
9	Dec 2017	credit_card	4879
10	Oct 2017	credit_card	4080
11	Aug 2017	credit_card	3753
12	Sep 2017	credit_card	3709
13	Jul 2017	credit_card	3495
14	May 2017	credit_card	3213
15	Jun 2017	credit_card	2721
16	Mar 2017	credit_card	2288
17	Apr 2017	credit_card	2061
18	Nov 2017	UPI	1793
19	Jan 2018	UPI	1758
20	Mar 2018	UPI	1584
21	Feb 2018	UPI	1550
22	Apr 2018	UPI	1525
23	Feb 2017	credit_card	1522
24	May 2018	UPI	1501
25	Jul 2018	UPI	1443
26	Dec 2017	UPI	1331
27	Aug 2018	UPI	1282
28	Jun 2018	UPI	1264
29	Oct 2017	UPI	1188
30	Aug 2017	UPI	1077
31	Sep 2017	UPI	1056
32	Jul 2017	UPI	982

	Month_year_purchase_date	payment_type	Amount_of_customers
33	May 2017	UPI	881
34	Jun 2017	UPI	814
35	Jan 2017	credit_card	702
36	Mar 2017	UPI	673
37	Apr 2017	UPI	572
38	Jan 2018	voucher	476
39	Feb 2017	UPI	451
40	Mar 2018	voucher	427
41	Nov 2017	voucher	419
42	Jul 2017	voucher	402
43	Apr 2018	voucher	393
44	Jun 2018	voucher	388
45	May 2018	voucher	369
46	Aug 2017	voucher	355
47	Feb 2018	voucher	339
48	Dec 2017	voucher	338
49	Sep 2017	voucher	338

Observations

- We have seen that before that it was at the month of NOV 2017 there was the most orders happening. Here we can see that they have used Credit for the payment.
- this signifies the payments methods during festive seasons.

In [478]:
```
# QUERY = """


# SELECT
#   COUNT(o.order_id) AS counter,
#   o.order_id,
#   c.customer_id,
#   customer_state,
#   customer_city,
#   FORMAT_DATE("%b %Y", DATE (order_purchase_timestamp)) AS Month_year_purchase_date

# FROM Target.orders o
# LEFT JOIN `Target.customers` c
# ON o.customer_id = c.customer_id
# LEFT JOIN `Target.order_items` ot
# ON o.order_id = ot.order_id
# LEFT JOIN `Target.sellers` s
# ON ot.seller_id = s.seller_id
# GROUP BY customer_id , customer_state , customer_city , Month_year_purchase_date, order_id
# ORDER BY counter DESC


#   """
# Query_Results = bigquery_client.query(QUERY)
# df = Query_Results.to_dataframe()
# df
```

In [ ]:
```

```

Distribution of payment installments and count of orders

In [695]:

```
QUERY = """


SELECT
  payment_installments,
  FORMAT_DATE("%b %Y", DATE (order_purchase_timestamp)) AS Month_year_purchase_date,
  p.payment_type,
  COUNT(o.order_id) AS count_of_orders

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.payments` p
ON o.order_id = p.order_id
GROUP BY p.payment_type ,Month_year_purchase_date,payment_installments
ORDER BY payment_installments DESC









  """
Query_Results = bigquery_client.query(QUERY)
payment_type_per_month = Query_Results.to_dataframe()
payment_type_per_month
```

Out[695]:

	payment_installments	Month_year_purchase_date	payment_type	count_of_orders
0	24	Nov 2017	credit_card	29
1	24	Dec 2017	credit_card	1
2	24	Jan 2018	credit_card	3
3	24	Jun 2018	credit_card	1
4	23	Jun 2018	credit_card	1
...	...	...	...	...

	payment_installments	Month_year_purchase_date	payment_type	count_of_orders
385	1	Oct 2016	debit_card	2
386	1	Dec 2016	credit_card	1
387	0	May 2018	credit_card	2
388	0	Apr 2018	credit_card	1
389	<NA>	Sep 2016	None	3

390 rows × 4 columns

Observations

- the maximum duration for credit card installments are 24 months ir 2 years.

INSIGHTS

- EXPLORING EACH TABLE ONE BY ONE
  - Orders - contains all the details of the orders like start date , status , delivery date etc along with missing values.
  - customers - contains the city name and the state name on which we can perform agg. functions to generate insights
  - order reviews - carries the review for each order

```
- the review comments etc
```

  - payments - contains the payment type and installments which can be used to talk about the economic status and financial stability of the people in Brazil and even with each and every region
  - order items - contains relevant information like the price of products and the frieght value which can be used to talk about the total amount of purchase the distance value between each purchase etc.
  - Products - talks about the product category , and the product details

```
- the product dimentsions can be used wrt to taking the idea of space consumptions and delivery veh
icle and means of transportation of we can analyse further on that.
```

  - sellers - the seller city and state along with customer table can be used to take insights on state specific revenue , connectivity etc.

- 1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
  - Time period for which the data is given
    - so the time duration started from 2016/9/4 and ended at 2018/10/17
    - which shows that we can only infer details and make connections based on full 2 years.
  - Cities and States covered in the dataset
    - There are 27 states inside the customer table which comprises of around 4119 cities.
    - this shows the geographical span of Brazil and their retail market
    - there are 23 seller states which comprises of 611 cities from which the products are manufactured or sent to customers.
    - we can see that the number of sellers cities are just 1/6th of the customer spread cities.
    - if we had the population of each we could have talked about the saturation of sellers or surplus of them there by tweaking the market demand - supply in Brazil

---

- 2 In-depth Exploration:
  - 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?
    - there is an increse in the number of customers over the years.
    - it signifies that the e-commerece in Brazil in ol the roll and the profit , inhouse production and IT growth assiciated with it in on a growth.
  - Can we see some seasonality with peaks at specific months?
    - In the data we can see that there is consistant increase in the nuber of orders along side the year and each month.
    - the most interesting fact is the largest number of orders are around Nov 2017 and Jan 2018 which shows the importance of Festive season like Christmas and New year in Brazil.
    - capitalising on those will be great revenue generating opportunity.
  - What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?
    - The Brazilian people have bought products more in the after noon followed by night and then morning.
    - so the advertising and push notification tema can focus more on these timings as they can be very active and motivated arounf afternoon.

- 3 Evolution of E-commerce orders in the Brazil region:
  - Get month on month orders by region, states
    - we can see that for each month and each region the number of orders in counter.
    - since there are just 611 seller cities compared to 4000 customer cities from which the orders come from , we can increase the nuber of sellers inorder to handle more diverse orders especially in cities which has more traffic.
  - How are customers distributed in Brazil

    - Brazilian people spend more on bed table bath , then beauty products and sports leisure the most.
    - in the top 3 there are 3 items related to aesthetics and self care.
    - tapping the potential for that will be more profitable considering the fact that the e-commerce is on a rise.
    - we can see that 80 percent of people has used credit card as the mode of payment
    - the value generated from it around 10 times more than than the very next that is UPI.
    - from this we can say that the people are spontaneous spenders who repay it with long term basis.
    - the chances that people think twice before buying is less with credit card purchases.
    - In hand cash purchases are very small therefore the technological advancements should be high and also upi and methods are reaaching even the minute pockets of the country there by increasing the connectivity

---

- 4 Impact on Economy: Analyze the money movemented by e-commerce by looking at order prices, freight and others.
  - Mean & Sum of price and freight value by customer state
    - the average customer to seller freight value increases as the distance increases.
    - since the distance of the order in not given its difficult to double check this inference but the frieght value is increasing for big orders .
    - this erratic behaviour can be associated the size of the product , quantity or the distance between the seller city and customer city

---

- 5 Analysis on sales, freight and delivery time
  - Calculate days between purchasing, delivering and estimated delivery / time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
    - we can see that in most of the times the order was delivered on time which is rmearkable but the amount of time its delayed needs to be reduced.
    - when the oreders are delayed there is a chance that people write bad reviews and give poor strs which may affect the sales and seller in effect.
  - Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5 /Top 5 states with highest/lowest average time to delivery /Top 5 states where delivery is really fast/ not so fast compared to estimated date
    - we can see that all the states have a very efficint delivery system on an average.
    - this shows the efficiency of the retail company and connectivity in Brazil
    - All the states have a delivery status of atleast a week before. This builds the reputation and trust in customers.

- 6 Payment type analysis:
  - Month over Month count of orders for different payment types
    - We have seen that before that it was at the month of NOV 2017 there was the most orders happening. Here we can see that they have used Credit for the payment. this signifies the payments methods during festive seasons.

- Distribution of payment installments and count of orders
  - the maximum duration for credit card installments are 24 months ir 2 years.m

RECOMMENDATION

- the most revenue generating state is SP and the least is AC.
  - the difference is around 10000 times which is bad for a country.
  - so make more sellers make the profitable products.
  - market more items that are not famous and evenly distribute the production to have a even growth

---

- its visible that there are products that can only be sold with good description like that of pc and beauty products.
- make sure such items are presented with organised description and wordings.

- the most number of products sold havce a star or 5 which shows the quality and satisfaction of the people in Brazil, the seeler quality and customer preference and choice. therefore increasing the quality and standars of low star products need to addressed quickly

---

- we can see that all the states have a very efficint delivery system on an average. this shows the efficiency of the retail company and connectivity in Brazil All the states have a delivery status of atleast a week before. This builds the reputation and trust in customers.

- the average customer to seller freight value increases as the distance increases. since the distance of the order in not given its difficult to double check this inference but the frieght value is increasing for big orders . this erratic behaviour can be associated the size of the product , quantity or the distance between the seller city and customer city

---

- the most profitable item in SP is bed bath table and bath. this product an be promoted in other staes as well inorder to boost the supply and reduce the monopoly of targeted growth

---

- very less number of orders from 11pm to 9am , Target should focus and increase customer base at that specific time by profiding discounts , push notifications and advertisements

---

- ecommerce market between 2016 , 2017 and 2018 is in a rise and Target should look to maximise this opportunity to leverage the boost of sales and healthy expansion.

Passion Works

Volume of products with category

In [529]:

```
QUERY = """



SELECT
    product_category,
    number_of_orders,
    (volume_of_the_product / number_of_orders) AS avg_volume_per_product
FROM


(SELECT

  p.product_category,
  SUM(p.product_length_cm * p.product_height_cm * p.product_width_cm) AS volume_of_the_product,
  SUM(product_name_length) AS product_name_length,
  SUM(product_description_length) AS product_description_lenght,
  COUNT(p.product_id) AS number_of_orders

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.products` p
ON ot.product_id = p.product_id
GROUP BY product_category
) AS x
ORDER BY  avg_volume_per_product DESC







  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type.head(10)
```

Out[529]:

	product_category	number_of_orders	avg_volume_per_product
0	Furniture office	1691	74018.172679
1	CITTE AND UPHACK FURNITURE	38	69557.710526
2	Furniture Kitchen Service Area Dinner and Garden	281	51949.128114
3	ELECTRICES 2	238	47954.546218
4	Furniture	109	47933.403670
5	Room Furniture	503	42625.393638
6	Industry Commerce and Business	268	41983.410448
7	PCs	203	41926.640394
8	Bags Accessories	1092	39505.455128
9	HOUSE PASTALS OVEN AND CAFE	76	39317.210526

Observation

- shows the volume of products per category.
- this can be used to suggest the shipping methods like light carriers or full load truch etc.

product description and name

In [525]:

```
QUERY = """



SELECT
    product_category,
    number_of_orders,
    (volume_of_the_product / number_of_orders) AS avg_volume_per_product,
    (product_name_length/ number_of_orders) AS avg_name_lenght_per_product,
    (product_description_lenght/ number_of_orders) AS avg_description_lenght_per_product
FROM


(SELECT

  p.product_category,
  SUM(p.product_length_cm * p.product_height_cm * p.product_width_cm) AS volume_of_the_product,
  SUM(product_name_length) AS product_name_length,
  SUM(product_description_length) AS product_description_lenght,
  COUNT(p.product_id) AS number_of_orders

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.products` p
ON ot.product_id = p.product_id
GROUP BY product_category
) AS x
ORDER BY avg_description_lenght_per_product DESC , number_of_orders









  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type.head(10)
```

Out[525]:

	product_category	number_of_orders	avg_volume_per_product	avg_name_lenght_per_product	avg_description_lenght_per_product
0	PCs	203	41926.640394	55.576355	2602.344828
1	song	38	6967.236842	45.447368	1472.684211
2	HOUSE PASTALS OVEN AND CAFE	76	39317.210526	51.552632	1368.105263
3	technical books	267	3137.254682	40.382022	1323.310861
4	CONSTRUCTION SECURITY TOOLS	194	6789.737113	52.711340	1320.288660
5	foods	510	4689.052941	48.796078	1273.635294
6	Construction Tools Garden	238	13808.462185	48.008403	1210.932773
7	Drink foods	278	6398.241007	43.579137	1192.528777
8	Industry Commerce and Business	268	41983.410448	51.007463	1166.757463
9	Furniture office	1691	74018.172679	44.415139	1156.361916

Observation

- its visible that there are products that can only be sold with good description like that of pc and beauty products.
- make sure such items are presented with organised description and wordings.

product category with pictures

In [534]:

```
QUERY = """



SELECT

  p.product_category,
  SUM(product_photos_qty) AS number_of_pics,
  COUNT(p.product_id) AS number_of_orders

FROM Target.orders o
LEFT JOIN `Target.customers` c
ON o.customer_id = c.customer_id
LEFT JOIN `Target.order_items` ot
ON o.order_id = ot.order_id
LEFT JOIN `Target.products` p
ON ot.product_id = p.product_id
GROUP BY product_category
ORDER BY number_of_pics DESC




  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type.head(10)
```

Out[534]:

	product_category	number_of_pics	number_of_orders
0	Furniture Decoration	20820	8334
1	sport leisure	18959	8641
2	HEALTH BEAUTY	17680	9670
3	bed table bath	16639	11115
4	housewares	16183	6964
5	Watches present	15019	5991

	product_category	number_of_pics	number_of_orders
6	computer accessories	13594	7827
7	telephony	13193	4545
8	toys	11463	4117
9	automotive	10645	4235

Observation

- we can conclude that keeping more picture for aestheically appearing prodicts can be really effective so that people can look athe products and but

Products with stars

In [551]:

```
QUERY = """



SELECT
    COUNT(DISTINCT(order_id)) AS number_of_orders,
    review_score AS Stars


FROM Target.order_reviews
GROUP BY review_score
ORDER BY review_score DESC







  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type.head(10)
```

Out[551]:

	number_of_orders	Stars
0	57076	5
1	19098	4
2	8160	3
3	3148	2
4	11393	1

Observations

- the most number of products sold havce a star or 5 which shows the quality and satisfaction of the people in Brazil, the seeler quality and customer preference and choice.

5 star and review comments

In [575]:

```
QUERY = """



SELECT
    DISTINCT(review_comment_title),
    COUNT(review_comment_title) AS count_of_each_review_title


FROM Target.order_reviews
WHERE review_score = 5
GROUP BY review_comment_title









  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type
```

Out[575]:

	review_comment_title	count_of_each_review_title
0	None	0
1	I recommend	583
2	I loved	58
3	Excellent	190
4	Commitment	2
...	...	...
1519	High quality	1
1520	Excellent pillow	1
1521	Good quality	1

	review_comment_title	count_of_each_review_title
1522	Overecommon	1
1523	It was in record time	1

1524 rows × 2 columns

Observation

- None of the people who gave 5 stars did not leave the review comments blank.
- most of them suggested i reccommend

2 star and review comments

In [577]:

```
QUERY = """



SELECT
    DISTINCT(review_comment_title),
    COUNT(review_comment_title) AS count_of_each_review_title


FROM Target.order_reviews
WHERE review_score = 2
GROUP BY review_comment_title
ORDER BY count_of_each_review_title DESC









  """
Query_Results = bigquery_client.query(QUERY)
People_type = Query_Results.to_dataframe()
People_type
```

Out[577]:

	review_comment_title	count_of_each_review_title
0	I recommend	24
1	Good	13
2	Defective product	9
3	Bad	9
4	Regular	8
...	...	...
351	No arrived	1
352	XXX	1
353	too slow	1
354	I'm waiting	1

	review_comment_title	count_of_each_review_title
355	None	0

356 rows × 2 columns

In [ ]:
```
1
```

most profitable product per state

In [687]:

```
QUERY = """



SELECT
   xx.product_category,
   xx.seller_state,
   SUM(xx.total_amount_payable) as Revenue_generated_per_product_per_state
FROM (

SELECT
    *,
    ROUND(((x.price + x.freight_value) * x.quantity),2) as total_amount_payable,

FROM
(SELECT


  p.product_category,
  ot.price,
  ot.freight_value,
  s.seller_state,
  COUNT(p.product_id) OVER (PARTITION BY ot.order_id, p.product_id) AS quantity


FROM `Target.order_items` ot
INNER JOIN `Target.sellers` s
ON ot.seller_id = s.seller_id
INNER JOIN `Target.products` p
ON ot.product_id = p.product_id) AS x) AS xx
GROUP BY seller_state , product_category
ORDER BY SUM(xx.total_amount_payable) DESC
  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[687]:

	product_category	seller_state	Revenue_generated_per_product_per_state
0	bed table bath	SP	1282792.50

	product_category	seller_state	Revenue_generated_per_product_per_state
1	Watches present	SP	1071760.87
2	Furniture Decoration	SP	948606.67
3	HEALTH BEAUTY	SP	914799.48
4	sport leisure	SP	812188.55
...	...	...	...
508	cine photo	MG	42.72
509	Arts and Crafts	MG	33.69
510	telephony	ES	32.10
511	Arts and Crafts	RJ	29.66
512	Drink foods	PR	22.52

513 rows × 3 columns

Observation

- the most profitable item in SP is bed bath table and bath.
- this product an be promoted in other staes as well inorder to boost the supply and reduce the monopoly of targeted growth

most profitable state per revenue generated

In [690]:

```
QUERY = """



SELECT

   xx.seller_state,
   SUM(xx.total_amount_payable) as Revenue_generated_per_product_per_state
FROM (

SELECT
    *,
    ROUND(((x.price + x.freight_value) * x.quantity),2) as total_amount_payable,

FROM
(SELECT


  p.product_category,
  ot.price,
  ot.freight_value,
  s.seller_state,
  COUNT(p.product_id) OVER (PARTITION BY ot.order_id, p.product_id) AS quantity


FROM `Target.order_items` ot
INNER JOIN `Target.sellers` s
ON ot.seller_id = s.seller_id
INNER JOIN `Target.products` p
ON ot.product_id = p.product_id) AS x) AS xx
GROUP BY seller_state
ORDER BY SUM(xx.total_amount_payable) DESC
  """
Query_Results = bigquery_client.query(QUERY)
df = Query_Results.to_dataframe()
df
```

Out[690]:

	seller_state	Revenue_generated_per_product_per_state
0	SP	12515952.56

	seller_state	Revenue_generated_per_product_per_state
1	PR	1736710.31
2	MG	1484402.92
3	RJ	1064683.72
4	SC	849432.11
5	RS	534151.23
6	BA	365500.10
7	DF	136394.92
8	PE	123820.17
9	GO	92155.43
10	ES	77296.44
11	MA	50703.56
12	CE	25289.31
13	MT	22655.21
14	PB	18769.89
15	RN	12489.13
16	MS	10163.27
17	RO	5474.98
18	PI	2965.32
19	SE	2346.05
20	PA	1393.11
21	AM	1258.80
22	AC	299.84

Observation

- the most revenue generating state is SP and the least is AC.
- the difference is around 10000 times which is bad for a country.

- so make more sellers make the profitable products.
- market more items that are not famous and evenly distribute the production to have a even growth

In [ ]:
```
1
```