



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

1.1. Data type of columns in a table

SQL: describe customers;

Customers:

Result Grid		 Filter Rows:	Export:			Wrap Cell
	Field	Type	Null	Key	Default	Extra
▶	customer_id	text	YES		NULL	
	customer_unique_id	text	YES		NULL	
	customer_zip_code_prefix	text	YES		NULL	
	customer_city	text	YES		NULL	
	customer_state	text	YES		NULL	

SQL: describe geolocation;

Field	Type	Null	Key	Default	Extra
geolocation_zip_code_prefix	text	YES		NULL	
geolocation_lat	double	YES		NULL	
geolocation_lng	double	YES		NULL	
geolocation_city	text	YES		NULL	
geolocation_state	text	YES		NULL	

SQL: describe order_items;

Field	Type	Null	Key	Default	Extra
order_id	text	YES		NULL	
order_item_id	int	YES		NULL	
product_id	text	YES		NULL	
seller_id	text	YES		NULL	
shipping_limit_date	datetime	YES		NULL	
price	double	YES		NULL	
freight_value	double	YES		NULL	

SQL: describe order_reviews;

Field	Type	Null	Key	Default	Extra
review_id	text	YES		NULL	
order_id	text	YES		NULL	
review_score	int	YES		NULL	
review_comment_title	text	YES		NULL	
review_creation_date	datetime	YES		NULL	
review_answer_timestamp	datetime	YES		NULL	

SQL: describe orders;

	Field	Type	Null	Key	Default	Extra
►	order_id	text	YES		NULL	
	customer_id	text	YES		NULL	
	order_status	text	YES		NULL	
	order_purchase_timestamp	datetime	YES		NULL	
	order_approved_at	datetime	YES		NULL	
	order_delivered_carrier_date	datetime	YES		NULL	
	order_delivered_customer_date	datetime	YES		NULL	
	order_estimated_delivery_date	datetime	YES		NULL	

SQL: describe payments;

	Field	Type	Null	Key	Default	Extra
►	order_id	text	YES		NULL	
	payment_sequential	int	YES		NULL	
	payment_type	text	YES		NULL	
	payment_installments	int	YES		NULL	
	payment_value	double	YES		NULL	

SQL: describe products;

	Field	Type	Null	Key	Default	Extra
►	product_id	text	YES		NULL	
	product_category	text	YES		NULL	
	product_name_length	int	YES		NULL	
	product_description_length	int	YES		NULL	
	product_photos_qty	int	YES		NULL	
	product_weight_g	int	YES		NULL	
	product_length_cm	int	YES		NULL	
	product_height_cm	int	YES		NULL	
	product_width_cm	int	YES		NULL	

SQL: describe sellers;

	Field	Type	Null	Key	Default	Extra
►	seller_id	text	YES		NULL	
	seller_zip_code_prefix	text	YES		NULL	
	seller_city	text	YES		NULL	
	seller_state	text	YES		NULL	

1.2. Time period for which the data is given

SQL: select min(shipping_limit_date),max(shipping_limit_date)

from order_items;

	min(shipping_limit_date)	max(shipping_limit_date)
▶	2016-09-19 00:15:34	2020-04-09 22:35:08

Insight: seems like in order_items table the data is available for the time frame 19-09-2016 to 09-04-2020

SQL: select min(review_creation_date),max(review_creation_date),
min(review_answer_timestamp), max(review_answer_timestamp)

from order_reviews;

	min(review_creation_date)	max(review_creation_date)	min(review_answer_timestamp)	max(review_answer_timestamp)
▶	2001-01-18 00:00:00	2031-12-17 00:00:00	2001-01-18 04:31:00	2031-12-17 20:35:00

Insight: seems like the order_reviews table has data for the time frame 18-01-2001 and 17-12-2031

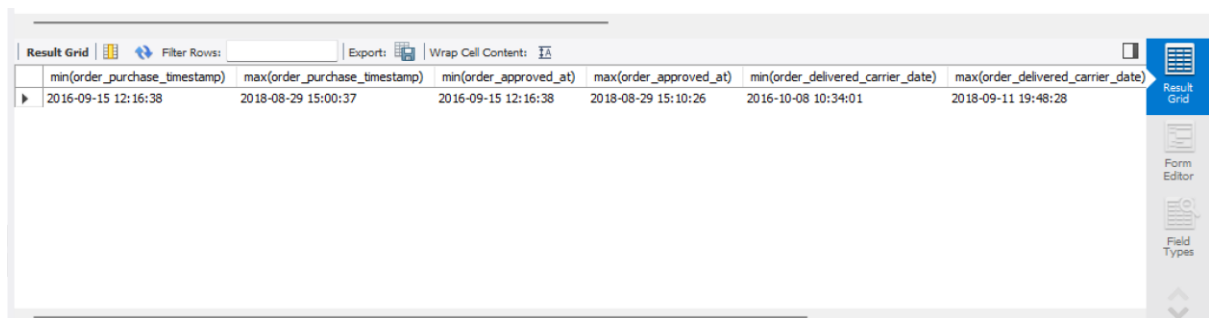
SQL: select min(order_purchase_timestamp),max(order_purchase_timestamp),

min(order_approved_at), max(order_approved_at),

min(order_delivered_carrier_date),max(order_delivered_carrier_date),

min(order_delivered_customer_date),max(order_delivered_customer_date)

from orders;



The screenshot shows a database interface with a 'Result Grid' displaying the results of a SQL query. The query selects the minimum and maximum values for six different timestamps from the 'orders' table. The results are as follows:

	min(order_purchase_timestamp)	max(order_purchase_timestamp)	min(order_approved_at)	max(order_approved_at)	min(order_delivered_carrier_date)	max(order_delivered_carrier_date)
▶	2016-09-15 12:16:38	2018-08-29 15:00:37	2016-09-15 12:16:38	2018-08-29 15:10:26	2016-10-08 10:34:01	2018-09-11 19:48:28

The interface includes a 'Filter Rows' section, an 'Export' button, and a 'Wrap Cell Content' checkbox. On the right side, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'.

Insight: seems like in the orders table there is data for the data frame 15-09-2016 and 17-10-2018

1.3. Cities and States of customers ordered during the given period

Answer:

- a. Query to find out the zip code prefix for customers who placed orders:

```
select customer_zip_code_prefix from customers
where customer_id in
(select customer_id from orders);
```

	customer_zip_code_prefix
▶	04571
	13175
	83085
	91340
	32070
	11440
	08340
	48700
	20250
	72225
	26115
	02435

- b. Query to find the city and states above customers placed each orders from:

```
select zip.customer_id, geo.geolocation_city, geo.geolocation_state
from
    (select customer_zip_code_prefix, customer_id from customers
    where customer_id in
        (select customer_id from orders)
    ) as zip
join
    geolocation as geo
on zip.customer_zip_code_prefix = geo.geolocation_zip_code_prefix
```

	customer_id	geolocation_city	geolocation_state
▶	30d149ba8c3793107e4829a8aaa4406f	sao paulo	SP
	e674bf9e1fd5863bbb4c72f95fdb39f	sao paulo	SP
	30d149ba8c3793107e4829a8aaa4406f	sao paulo	SP
	e414cf673d7803bcddf9263feed7c535	são paulo	SP
	e674bf9e1fd5863bbb4c72f95fdb39f	sao paulo	SP
	e674bf9e1fd5863bbb4c72f95fdb39f	sao paulo	SP
	e414cf673d7803bcddf9263feed7c535	sao paulo	SP
	e674bf9e1fd5863bbb4c72f95fdb39f	sao paulo	SP
	e674bf9e1fd5863bbb4c72f95fdb39f	sao paulo	SP
	e414cf673d7803bcddf9263feed7c535	sao paulo	SP
	e674bf9e1fd5863bbb4c72f95fdb39f	sao paulo	SP
	e414cf673d7803bcddf9263feed7c535	sao paulo	SP

- c. Query to find out distinct cities from where the orders were made:

```
select distinct(geo.geolocation_city)
from
    (select customer_zip_code_prefix, customer_id from customers
    where customer_id in
        (select customer_id from orders)
    ) as zip
join
    geolocation as geo
on zip.customer_zip_code_prefix = geo.geolocation_zip_code_prefix
```

	geolocation_city
▶	sao paulo
	são paulo
	sp
	são paulo
	osasco
	carapicuíba
	carapicuíba
	barueri
	santana de parnaíba
	santana de parnaíba
	itapevi
	jandira

Insight: As can be seen above the same city names are entered using different character sets and as a result the data consistency / cleanliness is impacted.

Recommendation: standardize the way through which the city name can be entered. This could be done by either auto-suggestion, using a single character set or maybe tokenize the location if possible.

- d. Query to find out the distinct states from where the orders were made:

```
select distinct( geo.geolocation_state)
from
    (select customer_zip_code_prefix, customer_id from customers
     where customer_id in
        (select customer_id from orders)
    ) as zip
join
    geolocation as geo
on zip.customer_zip_code_prefix = geo.geolocation_zip_code_prefix
```

	geolocation_state
▶	SP
	RJ
	ES
	MG
	BA
	SE
	PE
	AL
	PB
	AC

Insight: As can be seen all the orders came from these 10 states.

Recommendation: Need to focus on acquiring customers outside these 10 states as well to expand the business

2. In-depth Exploration:

- 2.1. 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?

Answer:

- a. Query to find the total number of orders placed each year in brazil:

```
select count(order_id) , year(order_purchase_timestamp) as yr
from orders
group by yr
order by yr
```

	count(order_id)	yr
►	272	2016
	43411	2017
	52778	2018

Insight: as can be seen from the above result between 2016 to 2018 there is a clear growing trend on e-commerce in Brazil.

- b. Query to find out number of product categories sold per year:

```
select year(od.order_purchase_timestamp) as yr, count("pd.product category")
as categories_sold
from
orders as od
join
order_items as oi
on od.order_id = oi.order_id
join
products as pd
on oi.product_id = pd.product_id
group by yr
order by yr
```

	yr	categories_sold
►	2016	311
	2017	47907
	2018	58658

Insight: As can be seen from above output between 2016 and 2018 there are more and more product categories sold each year which reaffirms our hypothesis that there is a growing trend on e-commerce in Brazil.

- c. Query to find the total number of orders placed each month each year:

```
select year(order_purchase_timestamp) as yr
,month(order_purchase_timestamp) as mth, count(order_id)
from orders
group by yr,mth
order by yr,mth
```

	yr	mth	count(order_id)
	2017	6	3135
	2017	7	3872
	2017	8	4193
	2017	9	4149
	2017	10	4478
	2017	11	7288
	2017	12	5513
	2018	1	7069
	2018	2	6556
	2018	3	7003
	2018	4	6798
	2018	5	6749

Insight: From the above output it seems like the last quarter (October, November and December) of the year (especially the month of November) each records the highest number of monthly sales each year.

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

Answer:

Assumption: using a 24hr time format

Dawn is between 2 :00 to 5:59 hrs

Morning is between 6:00 to 11:59 hrs

Afternoon is between 12:00 to 17:59 hrs

Night is between 18:00 to 01:59 hrs

Query to find the total sales done during each quarter of the day:

```
select case
  when hr between 2 and 6 then "Dawn"
  when hr between 7 and 11 then "Morning"
  when hr between 12 and 18 then "Afternoon"
  else "Night"
end as dayq,
sum(sub.ct) as order_count
from
(select hour(order_purchase_timestamp) as hr, count(order_id) as ct
from orders
group by hr
)as sub
group by dayq;
```

	dayq	order_count
▶	Morning	21114
	Night	30974
	Afternoon	42756
	Dawn	1617

Insight: From the above query output we can conclude that Brazilian customers tend to make maximum purchases during afternoon.

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

3.1. Get month on month orders by states

```
select
year(od.order_purchase_timestamp) as yr ,
month(od.order_purchase_timestamp) as mth,
cst.customer_state as state,
count(order_id) as ct
from
orders as od
join
customers as cst
on od.customer_id = cst.customer_id
group by yr, mth, state

order by yr, mth, ct desc
```

	yr	moth	state	ct
	2016	10	SP	95
	2016	10	RJ	43
	2016	10	MG	35
	2016	10	PR	19
	2016	10	RS	17
	2016	10	SC	10
	2016	10	GO	7
	2016	10	PE	6
	2016	10	CE	6
	2016	10	DF	6
	2016	10	RN	4
	2016	10	MA	4
	2016	10	PA	4
	2016	10	ES	3
	2016	10	SE	3
	2016	10	BA	3
	2016	10	RR	1
	2016	10	MT	1
	2016	10	PB	1
	2016	10	AL	1
	2016	10	PI	1
	2016	12	PR	1
	2017	1	SP	282
	2017	1	MG	101
	2017	1	RJ	91
	2017	1	PR	62
	2017	1	RS	52
	2017	1	SC	26
	2017	1	BA	24
	2017	1	GO	18
	2017	1	DF	13

Insight: as seen in the result MG, PR, RJ, RS , SC and SP almost every month drive the highest sales being in the top 6 of highest number of orders each month.

Recommendation: RR, AC, AP and RO are mostly present in the bottom five each month on the number of orders per month. SO more focus needs to be made to be made on these states.

3.2. Distribution of customers across the states in Brazil

```
select
customer_state,
count(customer_id) as customer_count
from
customers
group by customer_state
order by customer_count desc
```

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

For a deeper analysis on the distribution of customer on each city in each state:

```
select
customer_state,
customer_city,
count(customer_id) as customer_count
from
customers
group by customer_state , customer_city
order by customer_state , customer_city, customer_count desc
```

	customer_state	customer_city	customer_count
▶	AC	brasileia	1
	AC	cruzeiro do sul	3
	AC	epitaciolandia	1
	AC	manoel urbano	1
	AC	porto acre	1
	AC	rio branco	70
	AC	senador guiomard	2
	AC	xapuri	2
	AL	agua branca	1
	AL	anadia	2
	AL	arapiraca	29
	AL	atalaia	1
	AL	barra de santo a...	2
	AL	barra de sao miguel	2
	AL	batalha	3
	AL	belem	3
	AL	boca da mata	2
	AL	cacimbinhas	1
	AL	cajueiro	1
	AL	campo alegre	2
	AL	canapi	1
	AL	coite do noia	1
	AL	colonia leopoldina	2

Insight: The bottom 10 states have combined less than one the count of customer in one state SP alone , Which means that the concentration on customers are lopsided in favour of a few states.

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

- 4.1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table

```
select
sum(pt.payment_value)
from
orders as od
join
payments as pt
on od.order_id = pt.order_id
where year(od.order_purchase_timestamp) = 2017 and
month(od.order_purchase_timestamp) between 1 and 8
```

2017 total order value between Jan and Aug = 3471714.1599999317

	sum(pt.payment_value)
▶	3471714.1599999317

```
select
sum(pt.payment_value)
from
orders as od
join
payments as pt
on od.order_id = pt.order_id
where year(od.order_purchase_timestamp) = 2018 and
month(od.order_purchase_timestamp) between 1 and 8
```

2018 total order value between Jan and Aug = 8451969.199999993

	sum(pt.payment_value)
▶	8451969.199999993

Insight : % increase in cost of orders from 2017 to 2018 = $(8451969.199 - 3471714.16) / (3471714.16) * 100 = 143.452 \%$

4.2. Mean & Sum of price and freight value by customer state

```

select
cst.customer_state as state,
avg(oi.price) as avg_price,
sum(oi.price) as total_price,
avg(oi.freight_value) as avg_freight,
sum(oi.freight_value) as total_freight
from
orders as od
join
order_items as oi
on od.order_id = oi.order_id
join
customers as cst
on od.customer_id = cst.customer_id
group by state
order by avg_price, total_price, avg_freight, total_freight

```

	state	avg_price	total_price	avg_freight	total_freight
▶	SP	109.09221332130235	5065915.110001317	15.114605594676377	701876.93999999869
	PR	117.90821561338618	666063.5100000185	20.47181625066402	115645.29000000104
	RS	118.77434023813751	728205.4800000021	21.614624041755373	132519.26000000022
	MG	120.20063651849837	1552271.019999888	20.62668654173796	266373.03000000404
	ES	120.73862921348153	268643.4499999964	22.028979775280796	49014.47999999977
	SC	123.75479014153447	507147.13000000827	21.506627623230877	88134.16000000013
	GO	124.2146245059273	282836.69999999646	22.56286780851995	51375.64999999993
	RJ	124.43219472529407	1759844.529999834	20.911051403521185	295745.0000000001
	DF	125.90166029723889	296498.4099999976	21.072161358810963	49624.93999999982
	BA	134.0168721151253	493584.1400000065	26.487556339940014	97553.66999999907
	AM	135.92539877300632	22155.84000000003	33.31061349693253	5429.630000000003
	MS	142.33041923551153	115429.96999999984	23.35090012330457	18937.580000000005
	PE	144.2666036655193	251889.4899999967	32.693333333333285	57082.55999999992
	MA	146.34466833541924	116929.38999999997	38.507321652065094	30767.350000000001
	MT	146.76144648023077	152191.61999999933	27.99691417550633	29032.800000000065
	SE	150.86450666666704	56574.19000000014	36.57317333333333	13714.939999999988
	RR	153.42326086956518	7057.469999999998	43.088043478260886	1982.0500000000006
	CE	154.1595719298228	219677.38999999748	32.73864561403507	46652.56999999998
	TO	156.1371290322583	48402.51000000007	37.43503225806452	11604.86
	RN	157.5924376199617	82105.660000000003	35.71808061420345	18609.12
	PI	161.99043977055464	84721.00000000007	39.115086042065045	20457.190000000017
	AP	165.12111111111096	13374.809999999989	34.160493827160494	2767
	PA	165.5318690702073	174470.5899999985	35.62901328273243	37552.97999999998
	RO	167.33611721611751	45682.760000000008	41.330549450549434	11283.239999999996
	AC	175.06560439560428	15930.969999999988	40.0479120879121	3644.3600000000001
	AL	184.67381733021068	78855.71999999996	35.87065573770489	15316.769999999986
	PB	192.12767918088716	112586.81999999988	43.091689419795244	25251.730000000014

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
select distinct order_status from orders
```

	order_status
▶	delivered
	canceled

Insight: there are 2 types of order status 'delivered' and 'canceled'. So we have to do the analysis only on the delivered orders

```
select  
datediff(order_delivered_customer_date, order_purchase_timestamp) as  
delivery_days,  
datediff(order_estimated_delivery_date, order_delivered_carrier_date) as  
delivery_deviation
```

```
from orders  
where order_status = "delivered"  
order by delivery_days
```

	delivery_days	delivery_deviation
▶	0	11
	1	22
	1	11
	1	10
	1	8
	1	27
	1	12
	1	14
	1	20
	1	18
	1	14
	1	8
	1	6
	1	18
	1	13
	1	13
	1	31
	1	10
	1	5
	1	4

```
select
```

```
max(datediff(order_delivered_customer_date, order_purchase_timestamp)) as  
max_delivery_days,
```

```
max(datediff(order_estimated_delivery_date, order_delivered_carrier_date)) as  
max_delivery_deviation,
```

```
min(datediff(order_delivered_customer_date, order_purchase_timestamp)) as  
min_delivery_days,
```

```
min(datediff(order_estimated_delivery_date, order_delivered_carrier_date)) as  
min_delivery_deviation
```

```
from orders
```

```
where order_status = "delivered"
```

	max_delivery_days	max_delivery_deviation	min_delivery_days	min_delivery_deviation
▶	210	193	0	-99

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- $\text{time_to_delivery} = \text{order_purchase_timestamp} - \text{order_delivered_customer_date}$
- $\text{diff_estimated_delivery} = \text{order_estimated_delivery_date} - \text{order_delivered_customer_date}$

```
select  
order_id,  
datediff( order_purchase_timestamp, order_delivered_customer_date) as  
time_to_delivery,  
datediff(order_estimated_delivery_date,  
order_delivered_customer_date) as diff_estimated_delivery
```

```
from orders  
where order_status = "delivered"
```


	order_id	time_to_delivery	diff_estimated_delivery
▶	e481f51cbdc54678b7cc49136f2d6af7	-8	8
	53cdb2fc8bc7dce0b6741e2150273451	-14	6
	47770eb9100c2d0c44946d9cf07ec65d	-9	18
	949d5b44dbf5de918fe9c16f97b45f8a	-14	13
	ad21c59c0840e6cb83a9ceb5573f8159	-3	10
	a4591c265e18cb1dcee52889e2d8acc3	-17	6
	6514b8ad8028c9f2cc2374ded245783f	-10	12
	76c6e866289321a7c93b82b54852dc33	-10	32
	e69bfb5eb88e0ed6a785585b27e16dbf	-18	7
	e6ce16cb79ec1d90b1da9085a6118aeb	-13	9
	34513ce0c4fab462a55830c0989c7edb	-6	20
	82566a660a982b15fb86e904c8d32918	-12	29
	5ff96c15d0b717ac6ad1f3d77225a350	-5	9
	432aaf21d85167c2c86ec9448c4e42cc	-11	9
	dcb36b511fcac050b97cd5c05de84dc3	-14	13
	403b97836b0c04a622354cf531062e5f	-18	17
	116f0b09343b49556bbad5f35bee0cdf	-13	21
	85ce859fd6dc634de8d2f1e290444043	-6	14
	83018ec114eee8641c97e08f7b4e926f	-13	15
	203096f03d82e0dffbc41ebc2e2bcbf7	-21	-11

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

```

select cst.customer_state,
avg(oit.freight_value) as mean_freight_value,
avg(datediff( order_purchase_timestamp, order_delivered_customer_date)) as
mean_time_to_delivery,
avg(datediff(order_estimated_delivery_date, order_delivered_customer_date))
as meandiff_estimated_delivery
from
orders as od
join
customers as cst
on od.customer_id = cst.customer_id
join
order_items as oit
on od.order_id = oit.order_id
where od.order_status = "delivered"
group by cst.customer_state

```

	customer_state	mean_freight_value	mean_time_to_delivery	meandiff_estimated_delivery
►	SP	15.114601485947812	-8.6611	11.2057
	RS	21.614624041755373	-15.1352	14.1324
	SC	21.507359043202364	-14.9463	11.5685
	BA	26.48755633994001	-19.1925	10.9826
	MS	23.35090012330457	-15.4599	11.2293
	RJ	20.9127036775106	-15.0748	12.0085
	PI	39.11508604206504	-19.3174	11.5277
	MG	20.62719197707767	-11.9183	13.3417
	ES	22.0289797752808	-15.5874	10.6463
	RO	41.33054945054944	-19.6557	20.0403
	PR	20.47181625066402	-11.8931	13.4861
	DF	21.072161358810963	-12.8938	12.2004
	MT	27.99691417550633	-17.9074	14.5718

- Sort the data to get the following:
- Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Top 5 highest avg freight value:

```

select
cst.customer_state,
avg(oit.freight_value) as avg_freight
from
orders as od
join
customers as cst
on od.customer_id = cst.customer_id
join
order_items as oit
on od.order_id = oit.order_id
group by cst.customer_state
order by avg_freight desc
limit 5

```

	customer_state	avg_freight
►	PB	43.091689419795244
	RR	43.088043478260886
	RO	41.330549450549434
	AC	40.0479120879121
	PI	39.115086042065045

Top 5 lowest avg freight value:

```
select
cst.customer_state,
avg(oit.freight_value) as avg_freight
from
orders as od
join
customers as cst
on od.customer_id = cst.customer_id
join
order_items as oit
on od.order_id = oit.order_id
group by cst.customer_state
order by avg_freight
limit 5
```

	customer_state	avg_freight
▶	SP	15.114605594676377
	PR	20.47181625066402
	MG	20.62668654173796
	RJ	20.911051403521185
	DF	21.072161358810963

6. Top 5 states with highest/lowest average time to delivery

Top 5 highest average time to delivery:

```
select
cst.customer_state,
avg(datediff( od.order_purchase_timestamp,
od.order_delivered_customer_date)) as mean_time_to_delivery
from
orders as od
join
customers as cst
on od.customer_id = cst.customer_id
group by cst.customer_state
order by mean_time_to_delivery desc
limit 5
```

	customer_state	mean_time_to_delivery
▶	SP	-8.6992
	PR	-11.9380
	MG	-11.9454
	DF	-12.8990
	SC	-14.9075

Top 5 lowest average time to delivery:

```
select
cst.customer_state,
avg(datediff( od.order_purchase_timestamp,
od.order_delivered_customer_date)) as mean_time_to_delivery
from
orders as od
join
customers as cst
on od.customer_id = cst.customer_id
group by cst.customer_state
order by mean_time_to_delivery
limit 5
```

	customer_state	mean_time_to_delivery
►	RR	-29.3415
	AP	-27.1791
	AM	-26.3586
	AL	-24.5013
	PA	-23.7252

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

Top 5 slowest delivery speed:

```
select
cst.customer_state,
avg(datediff( order_estimated_delivery_date, order_delivered_customer_date))
as delivery_speed
from
orders as od
join
customers as cst
on od.customer_id = cst.customer_id
group by cst.customer_state
order by delivery_speed
limit 5
```

	customer_state	delivery_speed
►	AL	8.7078
	MA	9.5461
	SE	10.0209
	ES	10.4962
	BA	10.7945

Top 5 fastest delivery speed:

```
select
cst.customer_state,
avg(datediff( order_estimated_delivery_date, order_delivered_customer_date))
as delivery_speed
from
orders as od
join
customers as cst
on od.customer_id = cst.customer_id
group by cst.customer_state
order by delivery_speed desc
limit 5
```

	customer_state	delivery_speed
▶	AC	20.7250
	RO	20.1029
	AP	19.6866
	AM	19.5655
	RR	17.2927

6. Payment type analysis:

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

```
select
year(od.order_purchase_timestamp) as yr,
month(od.order_purchase_timestamp) as mth,
py.payment_type as py_type,
count(od.order_id) as order_count

from
orders as od
join
payments as py
on od.order_id = py.order_id
group by yr,mth,py_type
order by yr,mth
```

	yr	nth	py_type	order_count
►	2016	10	credit_card	214
	2016	10	debit_card	2
	2016	10	UPI	51
	2016	10	voucher	20
	2016	12	credit_card	1
	2017	1	credit_card	542
	2017	1	debit_card	9
	2017	1	UPI	186
	2017	1	voucher	60
	2017	2	credit_card	1257
	2017	2	debit_card	13
	2017	2	UPI	359
	2017	2	voucher	108
	2017	3	credit_card	1908
	2017	3	debit_card	30
	2017	3	UPI	565
	2017	3	voucher	197
	2017	4	credit_card	1772
	2017	4	debit_card	25
	2017	4	UPI	474
	2017	4	voucher	165
	2017	5	credit_card	2732
	2017	5	debit_card	29
	2017	5	UPI	740
	2017	5	voucher	285

Insight: as can be seen from above result each month credit card is the most popular mode of payment. Over the months it can be seen that the popularity of UPI as a payment mode is increasing.

Recommendation: debit card usage is very low each month. So special offers or cashbacks can be given to boost that mode of payment

6.2. Count of orders based on the no. of payment installments

```
select  
payment_installments,  
count(order_id) as order_count  
from payments  
group by payment_installments  
order by order_count desc
```

	payment_installments	order_count
►	1	52546
	2	12413
	3	10461
	4	7098
	10	5328
	5	5239
	8	4268
	6	3920
	7	1626
	9	644
	12	133
	15	74
	18	27
	11	23
	24	18
	20	17
	13	16
	14	15
	17	8
	16	5
	21	3
	0	2
	22	1
	23	1

Insight: as can be seen from the out put majority of the orders gets paid within 10 instalments