Comprehensive Analysis and Insights on TARGET

Introduction:

Target is a globally renowned brand and a prominent retailer in the United States. This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews. The datasets that have been provided are:

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

Data Sets Description:

The case study utilizes the following data sets:

1. Customers: This table contains information about customers, including unique customer IDs, zip code prefixes, city, and state. It provides insights into customer demographics and geographic distribution.

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	09790	sao bernardo do campo	SP
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	01151	sao paulo	SP
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	08775	mogi das cruzes	SP
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP
5	879864dab9bc3047522c92c82e1212b8	4c9374451666 7ad3b8f1fb645a3116a4	89254	jaragua do sul	SC
6	fd826e7cf63160e536e0908c76c3f441	addec96d2e059c80c30fe6871d30d177	04534	sao paulo	SP
7	5e274e7a0c3809e14aba7ad5aae0d407	57b2a98a409812fe9618067b6b8ebe4f	35182	timoteo	MG
8	5adf08e34b2e993982a47070956c5c65	1175e95fb47ddff9de6b2b06188f7e0d	81560	curitiba	PR
9	4b7139f34592b3a31687243a302fa75b	9afe194fb833f79e300e37e580171f22	30575	belo horizonte	MG

2. Geolocation: The geolocation table includes data related to zip code prefixes, latitude, longitude, city, and state. It serves as a reference for mapping zip codes to specific geographical locations.

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
0	01037	-23.54562128115268	-46.63929204800168	sao paulo	SP
1	01046	-23.546081127035535	-46.64482029837157	sao paulo	SP
2	01046	-23.54612896641469	-46.64295148361138	sao paulo	SP
3	01041	-23.5443921648681	-46.63949930627844	sao paulo	SP
4	01035	-23.541577961711493	-46.64160722329613	sao paulo	SP
5	01012	-23.547762303364266	-46.63536053788448	são paulo	SP
6	01047	-23.546273112412678	-46.64122516971552	sao paulo	SP
7	01013	-23.546923208436723	-46.6342636964915	sao paulo	SP
8	01029	-23.543769055769133	-46.63427784085132	sao paulo	SP
9	01011	-23.547639550320632	-46.63603162315495	sao paulo	SP

3. Order Items: This table consists of details regarding individual order items, such as order IDs, product IDs, seller IDs, shipping limit dates, prices, and freight values. It provides granular information about the products purchased and their associated sellers.

order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
428e8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19 09:45:35	58.90	13.29
f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017-05-03 11:05:13	239.90	19.93
9ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	2018-01-18 14:48:30	199.00	17.87
acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	2018-08-15 10:10:18	12.99	12.79
fb26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab4004ba5c87	2017-02-13 13:57:51	199.90	18.14
c3aef777c65dbb7d2a0634bc1ea	1	ef92defde845ab8450f9d70c526ef70f	6426d21aca402a131fc0a5d0960a3c90	2017-05-23 03:55:27	21.90	12.69
8431b9d7675808bcb819fb4a32	1	8d4f2bb7e93e6710a28f34fa83ee7d28	7040e82f899a04d1b434b795a43b4617	2017-12-14 12:10:31	19.90	11.85
9fe39319847cbb9d288c5617fa6	1	557d850972a7d6f792fd18ae1400d9b6	5996cddab893a4652a1572bfb58ab8db	2018-07-10 12:30:45	810.00	70.75
a1728c9d785b8e2b08b904576c	1	310ae3c140ff94b03219ad0adc3c778f	a416b6a846a11724393025641d4edd5e	2018-03-26 18:31:29	145.95	11.65
d0442cb953dcd1d21e1fb923495	1	4535b0e1091c278dfd193e5a1d63b39f	ba143b0f0110f0dc71ad71b4466ce92	2018-07-06 14:10:56	53.99	11.40

4. Order Reviews: The order reviews table contains information about customer reviews, including review IDs, order IDs, review scores, review comment titles, and timestamps. It offers insights into customer satisfaction and feedback.

review_id	order_id	review_score	review_comment_title	review_creation_date	review_answer_timestamp
7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	None	18/01/18 0:00	18/01/18 21:46
80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	None	10/03/18 0:00	11/03/18 3:05
228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	None	17/02/18 0:00	18/02/18 14:36
e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	None	21/04/17 0:00	21/04/17 22:02
f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	None	01/03/18 0:00	02/03/18 10:26
15197aa66ff4d0650b5434f1b46cda19	b18dcdf73be66366873cd26c5724d1dc	1	None	13/04/18 0:00	16/04/18 0:39
07f9bee5d1b850860defd761afa7ff16	e48aa0d2dcec3a2e87348811bcfdf22b	5	None	16/07/17 0:00	18/07/17 19:30
7c6400515c67679fbee952a7525281ef	c31a859e34e3adac22f376954e19b39d	5	None	14/08/18 0:00	14/08/18 21:36
a8f6f76f433de0aefbb97da197c554c	9c214ac970e84273583ab523dfafd09b	5	None	17/05/17 0:00	18/05/17 12:05
8670d52e15e00043ae7de4c01cc2fe06	b9bf720beb4ab3728760088589c62129	4	I recommend	22/05/18 0:00	23/05/18 16:45

5. Orders: This table captures comprehensive information about orders, including order IDs, customer IDs, order statuses, purchase timestamps, approval timestamps, delivery dates, and estimated delivery dates. It enables tracking the order lifecycle and evaluating delivery performance.

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carri
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-10-02 11:07:15	2017-10-04 1
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07-26 03:24:27	2018-07-26 1
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	2018-08-08 08:55:23	2018-08-08 1
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11-18 19:45:59	2017-11-22 1
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02-13 22:20:29	2018-02-14 1
5	a4591c265e18cb1dcee52889e2d8acc3	503740e9ca751ccdda7ba28e9ab8f608	delivered	2017-07-09 21:57:05	2017-07-09 22:10:13	2017-07-11 1
6	136cce7faa42fdb2cefd53fdc79a6098	ed0271e0b7da060a393796590e7b737a	invoiced	2017-04-11 12:22:08	2017-04-13 13:25:17	
7	6514b8ad8028c9f2cc2374ded245783f	9bdf08b4b3b52b5526ff42d37d47f222	delivered	2017-05-16 13:10:30	2017-05-16 13:22:11	2017-05-22 1
8	76fcfe8662f89321a7c93bfd2b54852dc33	f54a9f0e6b351c431402bb461ea51999	delivered	2017-01-23 18:29:09	2017-01-25 02:50:47	2017-01-26 1
9	e69bfb5eb88e0ed6a785585b27e16dbf	31ad1d1b63eb6962463f764d4e6e0c9d	delivered	2017-07-29 11:55:02	2017-07-29 12:05:32	2017-08-10 1

ner_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
la929d	delivered	2017-10-02 10:56:33	2017-10-02 11:07:15	2017-10-04 19:55:00	2017-10-10 21:25:13	2017-10-18 00:00:00
02d7ef	delivered	2018-07-24 20:41:37	2018-07-26 03:24:27	2018-07-26 14:31:00	2018-08-07 15:27:45	2018-08-13 00:00:00
f67089	delivered	2018-08-08 08:38:49	2018-08-08 08:55:23	2018-08-08 13:50:00	2018-08-17 18:06:29	2018-09-04 00:00:00
f64d82	delivered	2017-11-18 19:28:06	2017-11-18 19:45:59	2017-11-22 13:39:59	2017-12-02 00:28:42	2017-12-15 00:00:00
7aad2c	delivered	2018-02-13 21:18:39	2018-02-13 22:20:29	2018-02-14 19:46:34	2018-02-16 18:17:02	2018-02-26 00:00:00
b8f608	delivered	2017-07-09 21:57:05	2017-07-09 22:10:13	2017-07-11 14:58:04	2017-07-26 10:57:55	2017-08-01 00:00:00
7b737a	invoiced	2017-04-11 12:22:08	2017-04-13 13:25:17	None	None	2017-05-09 00:00:00
47f222	delivered	2017-05-16 13:10:30	2017-05-16 13:22:11	2017-05-22 10:07:46	2017-05-26 12:55:51	2017-06-07 00:00:00
e51999	delivered	2017-01-23 18:29:09	2017-01-25 02:50:47	2017-01-26 14:16:31	2017-02-02 14:08:10	2017-03-06 00:00:00
1e0c9d	delivered	2017-07-29 11:55:02	2017-07-29 12:05:32	2017-08-10 19:45:24	2017-08-16 17:14:30	2017-08-23 00:00:00

6. Payments: The payments table provides data on payment transactions, including order IDs, payment sequential numbers, payment types, payment installments, and payment values. It allows analysis of various payment methods and patterns.

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45
5	298fcdf1f73eb413e4d26d01b25bc1cd	1	credit_card	2	96.12
6	771ee386b001f06208a7419e4fc1bbd7	1	credit_card	1	81.16
7	3d7239c394a212faae122962df514ac7	1	credit_card	3	51.84
8	1f78449c87a54faf9e96e88ba1491fa9	1	credit_card	6	341.09
9	0573b5e23cbd798006520e1d5b4c6714	1	UPI	1	51.95

7. Products: This table contains details about products, such as product IDs, product categories,

name length, description length, photos quantity, weight, length, height, and width. It offers insights into the characteristics and attributes of the products.

	product_id	product category	product_name_length	product_description_length	product_photos_qty	product_weight_g	product_length_
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumery	40	287	1	225	
1	3aa071139cb16b67ca9e5dea641aaa2f	Art	44	276	1	1000	
2	96bd76ec8810374ed1b65e291975717f	sport leisure	46	250	1	154	
3	ce87bcfe19066a932b7673e239eb23d	babies	27	261	1	371	
4	9dc1a7de274444849c219cff195d0b71	housewares	37	402	4	625	
5	41d3672d4792049fa1779bb35283ed13	musical instruments	60	745	1	200	
6	732bd381ad09e530fe0a5f457d81becb	Cool Stuff	56	1272	4	18350	
7	2548af3e6e77a690cf3eb6368e9ab61e	Furniture Decoration	56	184	2	900	
8	37cc742be07708b53a98702e77a21a02	home appliances	57	163	1	400	
9	8c92109888e8cdf9d66dc7e463025574	toys	36	1156	1	600	

product category	product_name_length	product_description_length	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm
perfumery	40	287	1	225	16	10	14
Art	44	276	1	1000	30	18	20
sport leisure	46	250	1	154	18	9	15
babies	27	261	1	371	26	4	26
iusewares	37	402	4	625	20	17	13
musical estruments	60	745	1	200	38	5	11
Cool Stuff	56	1272	4	18350	70	24	44
Furniture Decoration	56	184	2	900	40	8	40
home ppliances	57	163	1	400	27	13	17
toys	36	1156	1	600	17	10	12

8. Sellers: The sellers table includes information about sellers, such as seller IDs, zip code prefixes, city, and state. It provides insights into the distribution and locations of sellers.

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP
1	d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	SP
2	ce3ad9de960102d0677a81f5d0bb7b2d	20031	rio de janeiro	RJ
3	c0f3eea2e14555b6faeea3dd58c1b1c3	04195	sao paulo	SP
4	51a04a8a6bdcb23deccc82b0b80742cf	12914	braganca paulista	SP
5	c240c4061717ac1806ae6ee72be3533b	20920	rio de janeiro	RJ
6	e49c26c3edfa46d227d5121a6b6e4d37	55325	brejao	PE
7	1b938a7ec6ac5061a66a3766e0e75f90	16304	penapolis	SP
8	768a86e36ad6aae3d03ee3c6433d61df	01529	sao paulo	SP
9	ccc4bbb5f32a6ab2b7066a4130f114e3	80310	curitiba	PR

Analyzing the Data:

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

   a. Data type of all columns in the "customers" table.

      Query:

      df = spark.sql("""describe customers;""").show()

      Output:

```
+--------------------+---------+-------+
|            col_name|data_type|comment|
+--------------------+---------+-------+
|         customer_id|   string|   null|
|  customer_unique_id|   string|   null|
|customer_zip_code...|   string|   null|
|       customer_city|   string|   null|
|      customer_state|   string|   null|
+--------------------+---------+-------+
```

   b. Get the time range between which the orders were placed.

      Query:

      spark.sql("""select date_format(order_purchase_timestamp,'yyyy-MM-dd') as Day, min(order_purchase_timestamp) as Start_time_of_orders, max(order_purchase_timestamp) as End_time_of_orders from orders group by date_format(order_purchase_timestamp,'yyyy-MM-dd') order by day;""").show(10)

      Output:

```
+----------+--------------------+--------------------+
|       Day|Start_time_of_orders| End_time_of_orders|
+----------+--------------------+--------------------+
|2016-09-04| 2016-09-04 21:15:19|2016-09-04 21:15:19|
|2016-09-05| 2016-09-05 00:15:34|2016-09-05 00:15:34|
|2016-09-13| 2016-09-13 15:24:19|2016-09-13 15:24:19|
|2016-09-15| 2016-09-15 12:16:38|2016-09-15 12:16:38|
|2016-10-02| 2016-10-02 22:07:52|2016-10-02 22:07:52|
|2016-10-03| 2016-10-03 09:44:50|2016-10-03 22:51:30|
|2016-10-04| 2016-10-04 09:06:10|2016-10-04 23:59:01|
|2016-10-05| 2016-10-05 00:32:31|2016-10-05 23:14:34|
|2016-10-06| 2016-10-06 00:06:17|2016-10-06 23:49:18|
|2016-10-07| 2016-10-07 00:54:40|2016-10-07 23:18:38|
+----------+--------------------+--------------------+
```

   c. Count the number of Cities and States in our dataset.

      Query:

spark.sql("""select count(distinct geolocation_city) as Number_of_cities, count(distinct geolocation_state) as Number_of_states from geolocation;""").show()

Output:

```
+----------------+----------------+
|Number_of_cities|Number_of_states|
+----------------+----------------+
|            8011|              27|
+----------------+----------------+
```

2. In-depth Exploration:

a. cIs there a growing trend in the no. of orders placed over the past years?

Query:

with cte as (select extract(year from order_purchase_timestamp) as year from orders) select year, count(year) as count from cte group by year;

Visualization code:

```
df = spark.sql("""with cte as (select extract(year from
order_purchase_timestamp) as year from orders)
select year, count(year) as Number_of_orders from cte group by year;""")
df.show()

x_values = df.select('year').rdd.flatMap(lambda x: x).collect()
y_values = df.select('Number_of_orders').rdd.flatMap(lambda x: x).collect()

plt.bar(x_values, y_values)
plt.xlabel('Number of Orders')
plt.ylabel('Year')
plt.title('Number of Orders Vs Year')
plt.show()
```

Output:

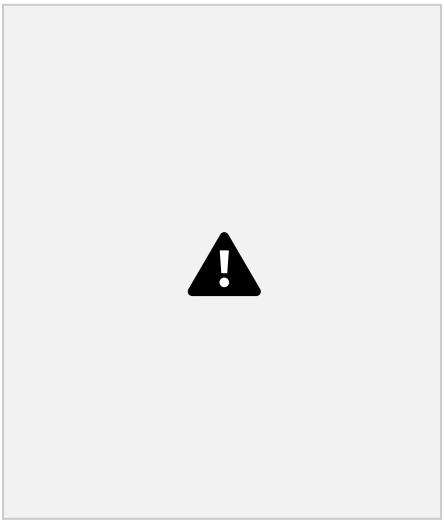
```
+----+----------------+
|year|Number_of_orders|
+----+----------------+
|2018|           54011|
|2016|             329|
|2017|           45101|
+----+----------------+
```

b. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Query:

select date_format(order_purchase_timestamp, 'yyyy-MM') as Date, count(order_id) as Number_of_orders from orders group by Date order by Date limit 10;

Output:



c. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

Query:

select

count(case when hour(order_purchase_timestamp) <= 6 then 1 else null end) as Dawn,

count(case when hour(order_purchase_timestamp) between 7 and 12 then 1

else null end) as Mornings,

count(case when hour(order_purchase_timestamp) between 13 and 18 then 1 else null end) as Afternoon,

count(case when hour(order_purchase_timestamp) between 19 and 23 then 1 else null end) as Night

from orders;

Output:

3. Evolution of E-commerce orders in the Brazil region:
    a. Get the month on month no. of orders placed in each state.

Query:

with a as (select date_format(DATE_TRUNC('month', o.order_purchase_timestamp), 'yyyy-MM') as Dates,

c.customer_state as States,

count(o.order_id) as Orders_count

from orders o inner join customers c on o.customer_id = c.customer_id

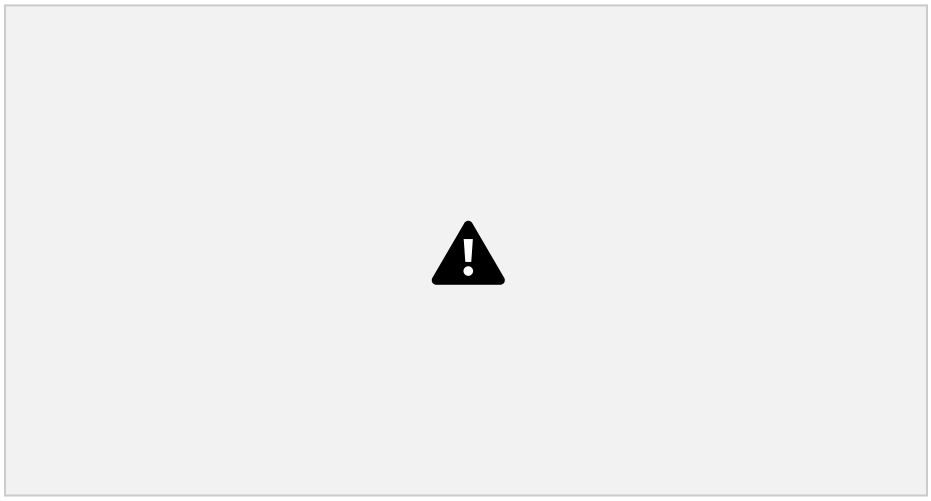
group by date_format(DATE_TRUNC('month', o.order_purchase_timestamp), 'yyyy-MM'), c.customer_state

order by date_format(DATE_TRUNC('month', o.order_purchase_timestamp), 'yyyy-MM'), c.customer_state)

select dates, states, orders_count, lag(orders_count) over(order by dates,states) as Prev_years_orders_count

from a;

Output:



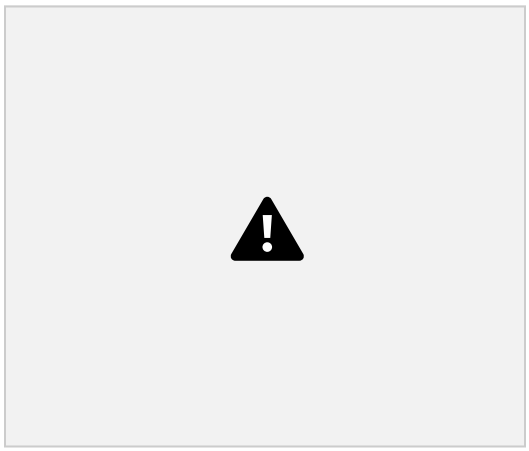
b. How are the customers distributed across all the states?

Query:

select customer_state, count(customer_state) as customers_count

from customers group by customer_state order by customers_count desc limit 10;

Output:



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

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

Query:

select

round((((round(sum(case when extract(year from o.order_purchase_timestamp)

= 2018 and

extract(month from o.order_purchase_timestamp) <= 8 then p.payment_value else null end),2))

-(round(sum(case when extract(year from o.order_purchase_timestamp) = 2017 and

extract(month from o.order_purchase_timestamp) <= 8 then p.payment_value else null end),2)))/

(round(sum(case when extract(year from o.order_purchase_timestamp) = 2017 and

*extract(month from o.order_purchase_timestamp) <= 8 then p.payment_value else null end),2))*100),2) as percent_increase*

from payments p inner join orders o on o.order_id = p.order_id;

Output:

b. Calculate the Total & Average value of order price for each state.

Query:

select sub.s as states, round(sum(sub.p),2) as price_total,round(avg(sub.p),2) as price_average from (select c.customer_state as s, p.payment_value p from customers c inner join orders o on c.customer_id = o.customer_id inner join payments p on p.order_id = o.order_id) as sub group by sub.s;

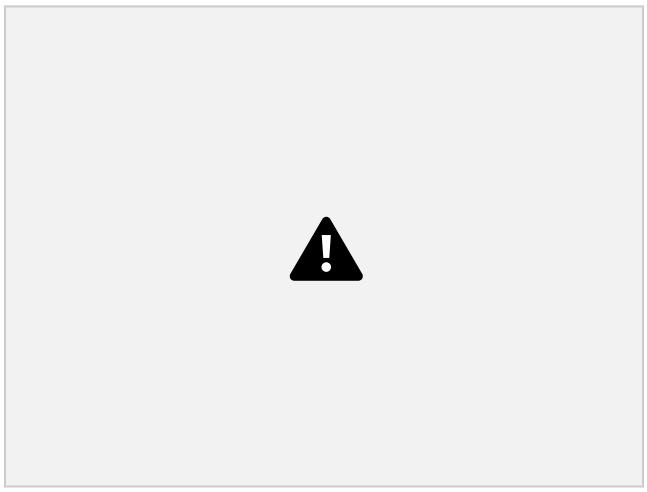
Output:

c. Calculate the Total & Average value of order freight for each state.

Query:

select sub.s as state, round(sum(sub.f),2) as freight_total, round(avg(sub.f),2) as freight_average from (select c.customer_state as s, oi.freight_value as f from orders o inner join order_items oi on o.order_id = oi.order_id inner join customers c on c.customer_id = o.customer_id) as sub group by sub.s;

Output:

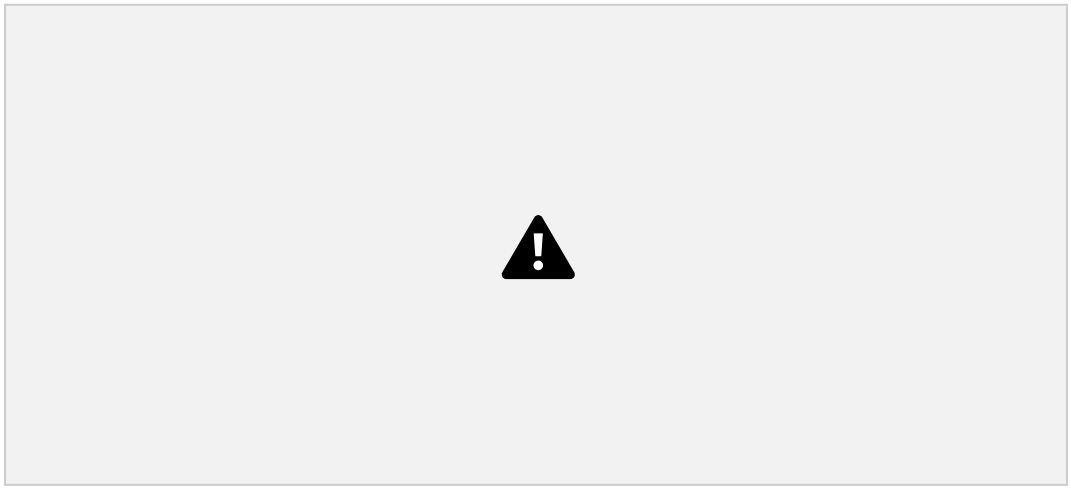


5. Analysis based on sales, freight and delivery time.
   a. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
   Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
   Do this in a single query.
   Query:

   select order_id,
   datediff(order_delivered_customer_date,order_purchase_timestamp)
   as time_to_deliver,

datediff(order_estimated_delivery_date,order_delivered_customer_date) as diff_estimated_delivery from orders limit 10;

Output:

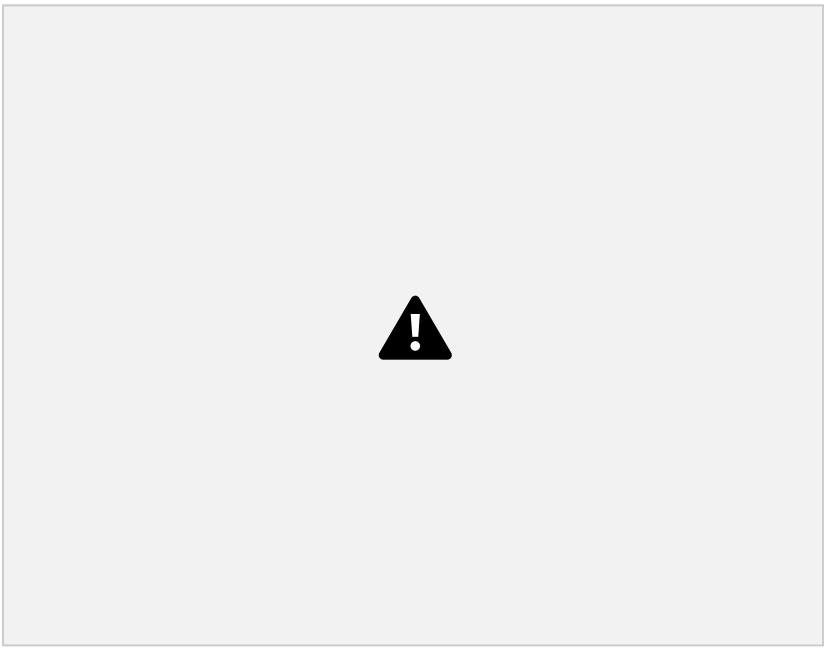


b. Find out the top 5 states with the highest & lowest average freight value. Query:

select c.customer_state as top5_states_with_highest_average_freight_value from orders o
inner join order_items oi on o.order_id = oi.order_id inner join customers c on o.customer_id = c.customer_id group by c.customer_state order by avg(oi.freight_value) desc limit 5;

select c.customer_state as top5_states_with_lowest_average_freight_value from orders o inner join order_items oi on o.order_id = oi.order_id inner join customers c on o.customer_id = c.customer_id
group by c.customer_state order by avg(oi.freight_value) limit 5

Output:

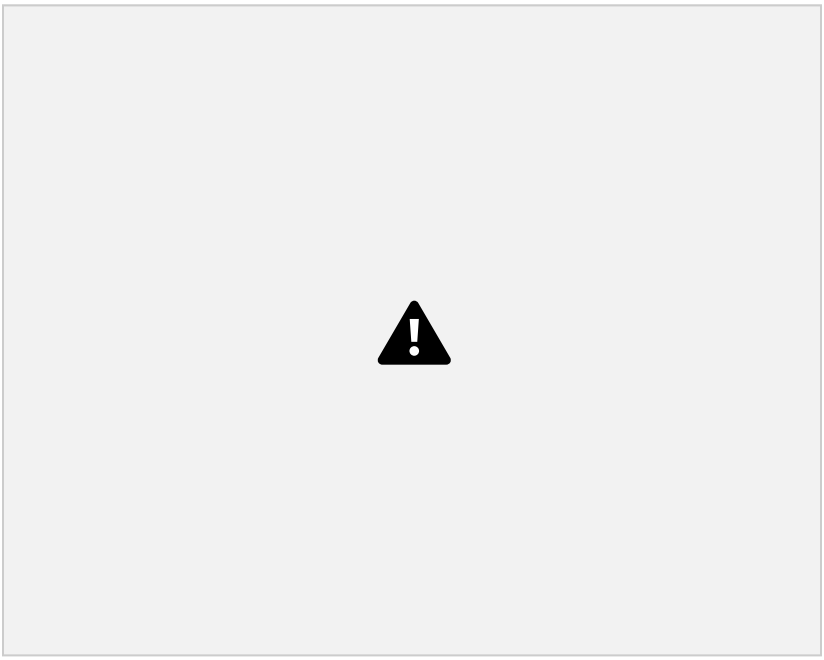


c. Find out the top 5 states with the highest & lowest average delivery time. Query:

```
select c.customer_state as top5_states_with_highest_average_delivery_time
from orders o inner join customers c on o.customer_id = c.customer_id
group by c.customer_state
order by
avg(datediff(o.order_estimated_delivery_date,o.order_purchase_timestamp))
desc limit 5;

select c.customer_state as top5_states_with_lowest_average_delivery_time
from orders o inner join customers c on o.customer_id = c.customer_id
group by c.customer_state
order by
avg(datediff(o.order_estimated_delivery_date,o.order_purchase_timestamp))
limit 5;
```

Output:

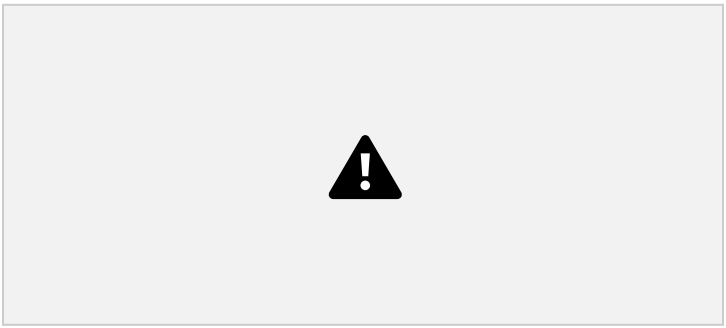


d. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Query:

select c.customer_state as top5_states_with_fastest_order_delivery
from customers c inner join orders o on c.customer_id = o.customer_id
group by c.customer_state
order by
avg(datediff(o.order_estimated_delivery_date,o.order_delivered_customer_date)
) desc limit 5;

Output:



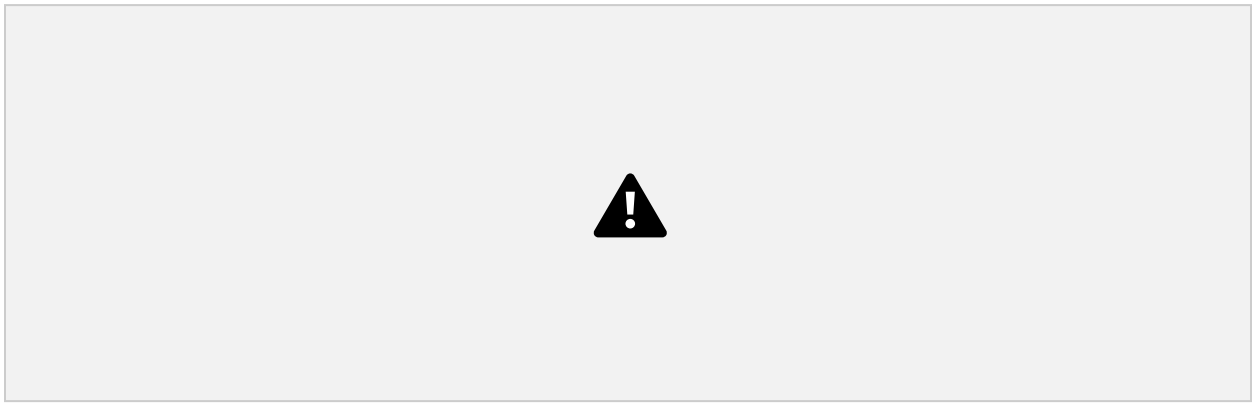
6. Analysis based on the payments:

a. Find the month on month no. of orders placed using different payment types.

Query:

```
select date_format(DATE_TRUNC('month', o.order_purchase_timestamp),
 'yyyy-MM') as Month,
payment_type as Payment_type, count(payment_type) as
 Number_of_orders_placed,
 lag(count(payment_type))
 over(order by date_format(DATE_TRUNC('month',
 o.order_purchase_timestamp), 'yyyy-MM'),
 count(payment_type)) as Number_of_orders_placed_in_previousYear
from orders o inner join payments p on o.order_id = p.order_id
 group by DATE_TRUNC('month', o.order_purchase_timestamp),
 payment_type order by Month, Number_of_orders_placed;
```

Output:

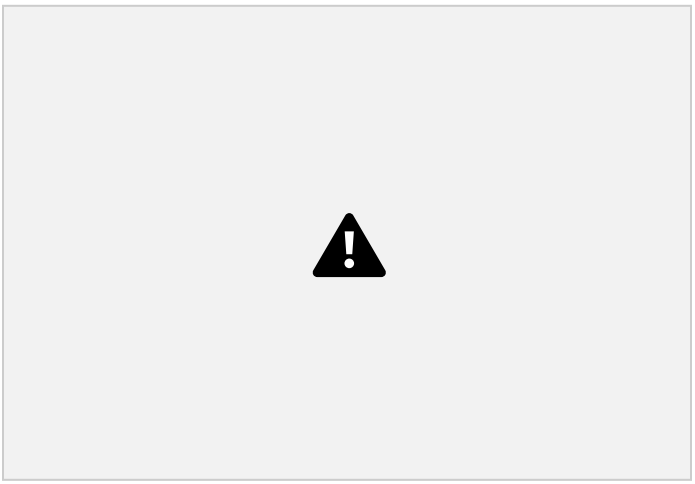


b. Find the no. of orders placed on the basis of the payment installments that have been paid.

Query:

```
select payment_installments,
 count(order_id) as Number_of_orders
from payments
 group by payment_installments
 order by payment_installments limit 10;
```

Output:



Miscellaneous Questions:

1. Is there any relation between delivery time and reviews?

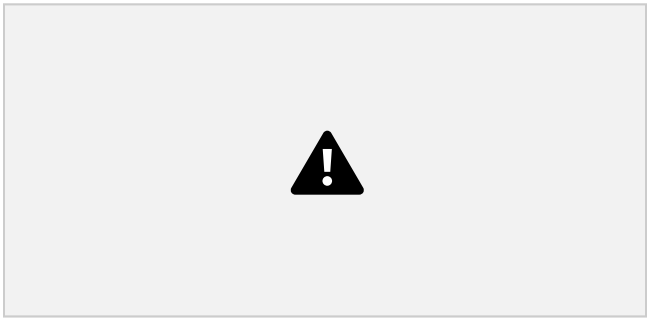
   Query:

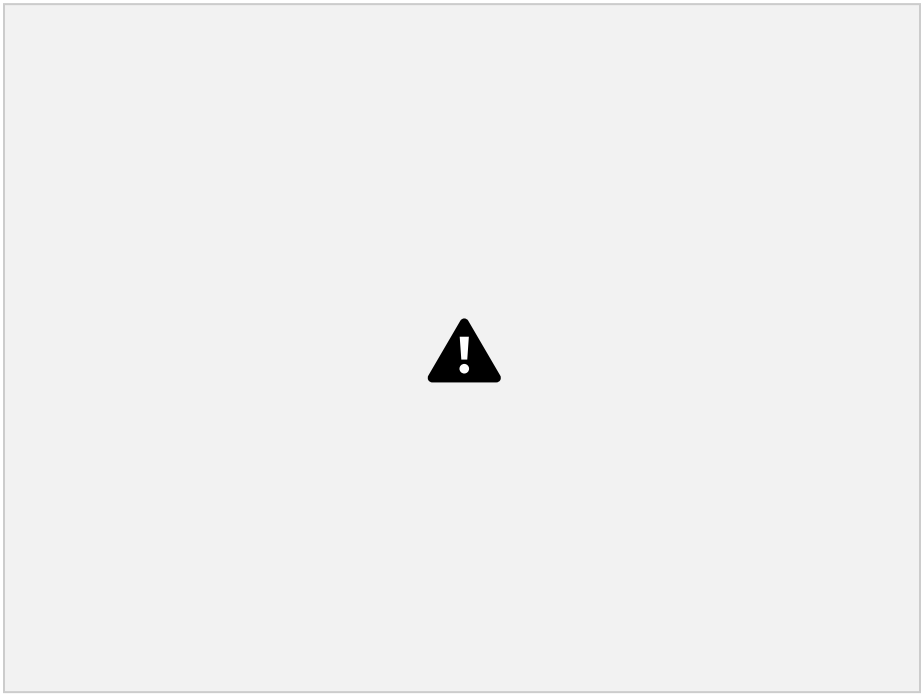
   ```
   df = spark.sql("""select ore.review_score as order_review,
   round(avg(datedif (o.order_delivered_customer_date,o.order_purchase_timestamp)),2) as
   average_delivery_time
   from orders o inner join order_reviews ore on
    o.order_id = ore.order_id group by ore.review_score order by average_delivery_time""")

   df.show()
   df.toPandas()
   x_values = [float(row['order_review']) for row in df.collect()]
   y_values = [int(row['average_delivery_time']) for row in df.collect()]

   plt.bar(range(len(y_values)), x_values)
   plt.yticks(range(len(y_values)), y_values)
   plt.xlabel('Order Reviews')
   plt.ylabel('Average Delivery Time')
   plt.title('Order Reviews Vs Delivery Time')
   plt.show()
   ```

   Output:





2. How many products are there in a specific product category?

Query:

spark.sql("""***select `product category`, count(product_id) as Products_count from products group by `product category` limit 10;***""").show()

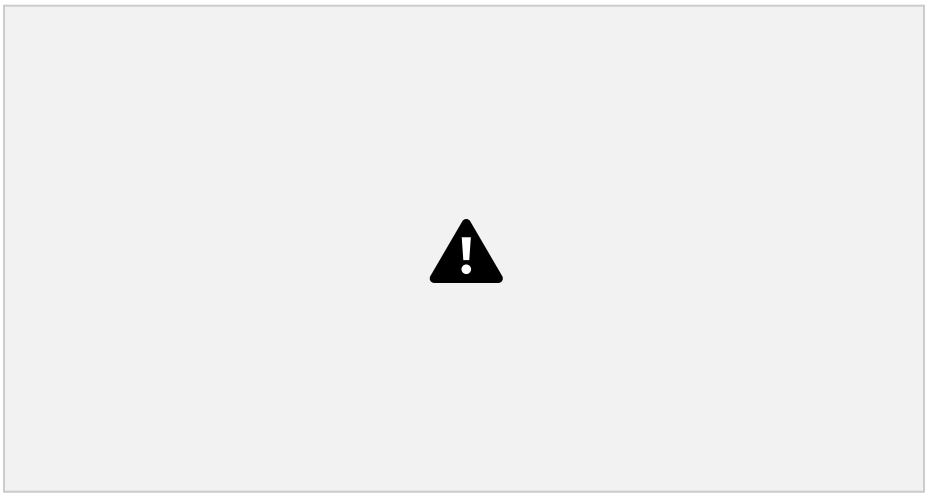
Output:

3. How many sellers are there in each city and state?

Query:

spark.sql("""***select seller_state, seller_city, count(seller_id) from sellers group by seller_state, seller_city limit 10;***""").show()

Output:

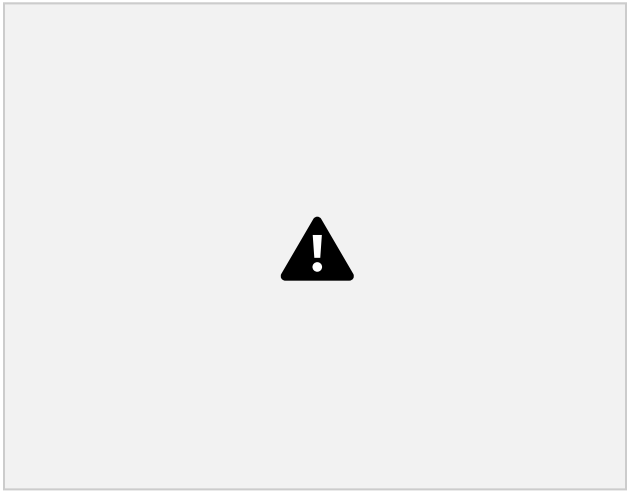


4. How many customers are there from each state?

   Query:

   spark.sql("""***select customer_state, count(customer_id) from customers group by customer_state limit 10;***""").show()

   Output:

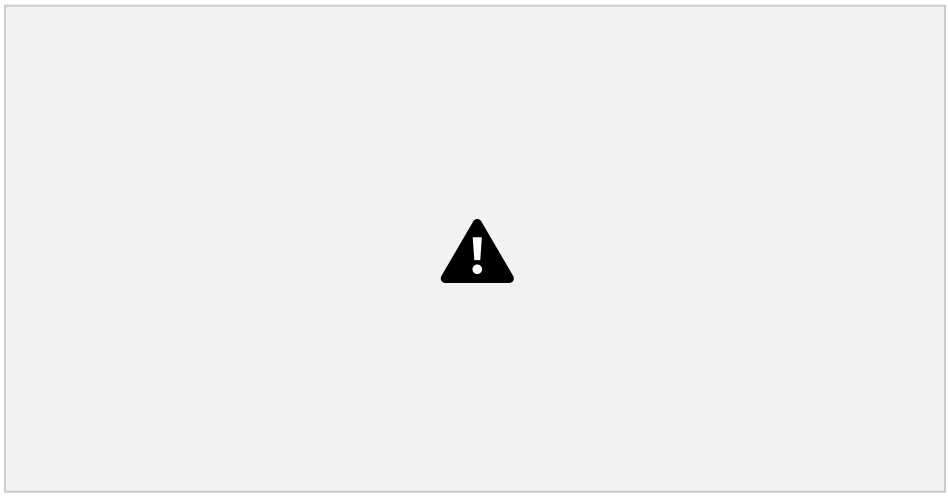


5. How many zip_code_prefixes are present in each state?

   Query:

   spark.sql("""***select geolocation_state, count(geolocation_zip_code_prefix) from geolocation group by geolocation_state limit 10;***""").show()

   Output:



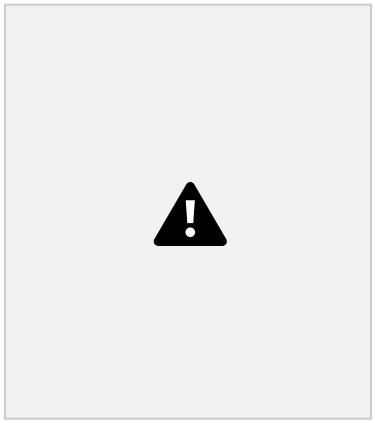
6. What are the states that are generating less revenue?

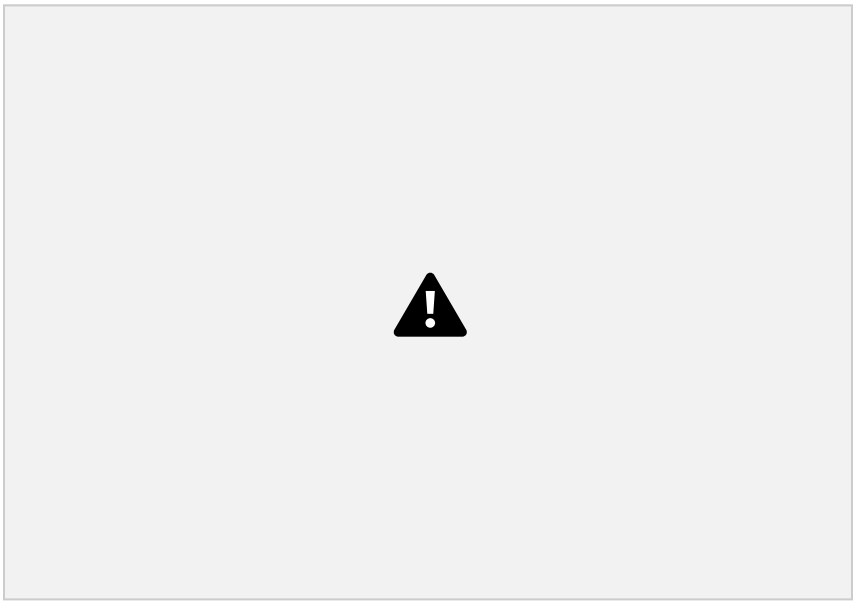
Query:

```
df = spark.sql("""select c.customer_state as cus_state, round(sum(p.payment_value),2) as summ
from payments p inner join orders o on p.order_id = o.order_id inner join customers c on
c.customer_id = o.customer_id group by c.customer_state having sum(p.payment_value) <
100000 order by summ;""")
df.show()
df.toPandas()
x_values = df.select('cus_state').rdd.flatMap(lambda x: x).collect()
y_values = df.select('summ').rdd.flatMap(lambda x: x).collect()

plt.bar(x_values, y_values)
plt.xlabel('Brazil States')
plt.ylabel('Revenue')
plt.title('States Vs Revenue')
plt.show()
```

Output:





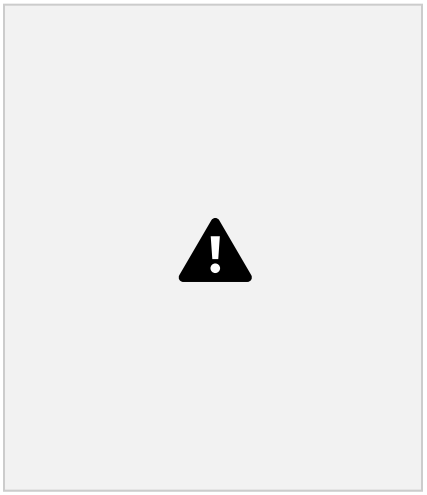
7. What is the revenue generated on weekdays and weekends? Query:

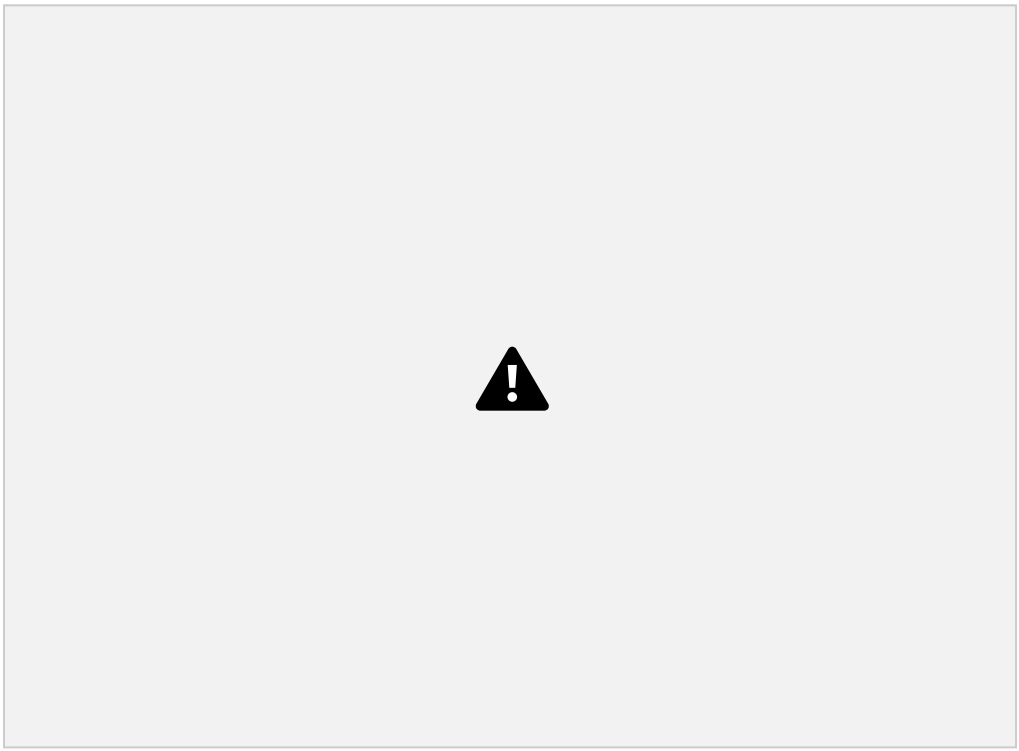
```
df = spark.sql("""select year(order_purchase_timestamp) as year,
date_format(order_purchase_timestamp, "EEEE") as day,
count(order_purchase_timestamp) as countt from orders
group by year, date_format(order_purchase_timestamp, "EEEE") order by year""")

df.toPandas()
x_values = df.select('day').rdd.flatMap(lambda x: x).collect()
y_values = df.select('countt').rdd.flatMap(lambda x: x).collect()

plt.bar(x_values, y_values)
plt.xlabel('Days')
plt.ylabel('Number of Orders')
plt.title('Days vs Number of Orders')
plt.show()
```

Output:





8. What is the percent of yearly increase in the number of orders?

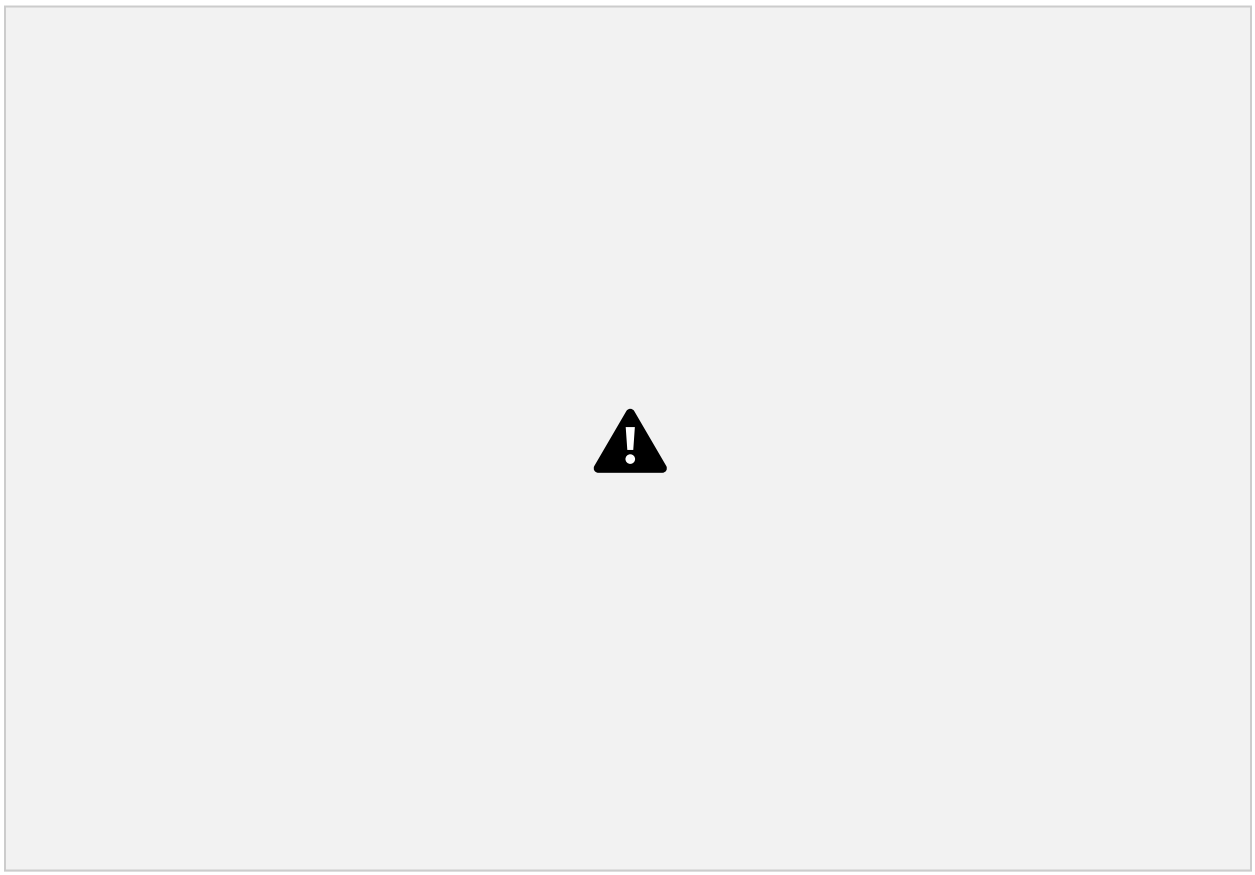
Query:

```
df = spark.sql("""with cte as (select extract(year from order_purchase_timestamp) as year from orders), b as
(select year, count(year) as Number_of_orders from cte group by year) select year,
percent_increase from (select year, Number_of_orders,
lag(Number_of_orders) over(order by year) as prev,
round(((Number_of_orders-prev)/prev)*100,2) as percent_increase from b) as sub where
sub.prev is not null;""")

df.show()
df.toPandas()
x_values = df.select('year').rdd.flatMap(lambda x: x).collect()
y_values = df.select('percent_increase').rdd.flatMap(lambda x: x).collect()


plt.bar(x_values, y_values)
plt.xticks([2017,2018])
plt.xlabel('Year')
plt.ylabel('Percent Increase')
plt.title('Yearly Percent Increase in Number of Orders')
plt.show()
```

Output:



9. What are the number of orders that took more than 50 days from the estimated days to deliver? Query:

spark.sql("""***select count(countt)as Number_of_extra_days_from_estimated_delivery_time from (select datedif (order_estimated_delivery_date,order_delivered_customer_date) as dif _estimated_delivery, count(order_id) as countt from orders group by datedif (order_estimated_delivery_date,order_delivered_customer_date) having datedif (order_estimated_delivery_date,order_delivered_customer_date) > 50 order by dif _estimated_delivery desc);***""").show()

Output:

10. Is there any pattern between revenue and time required to deliver the product?

Query:

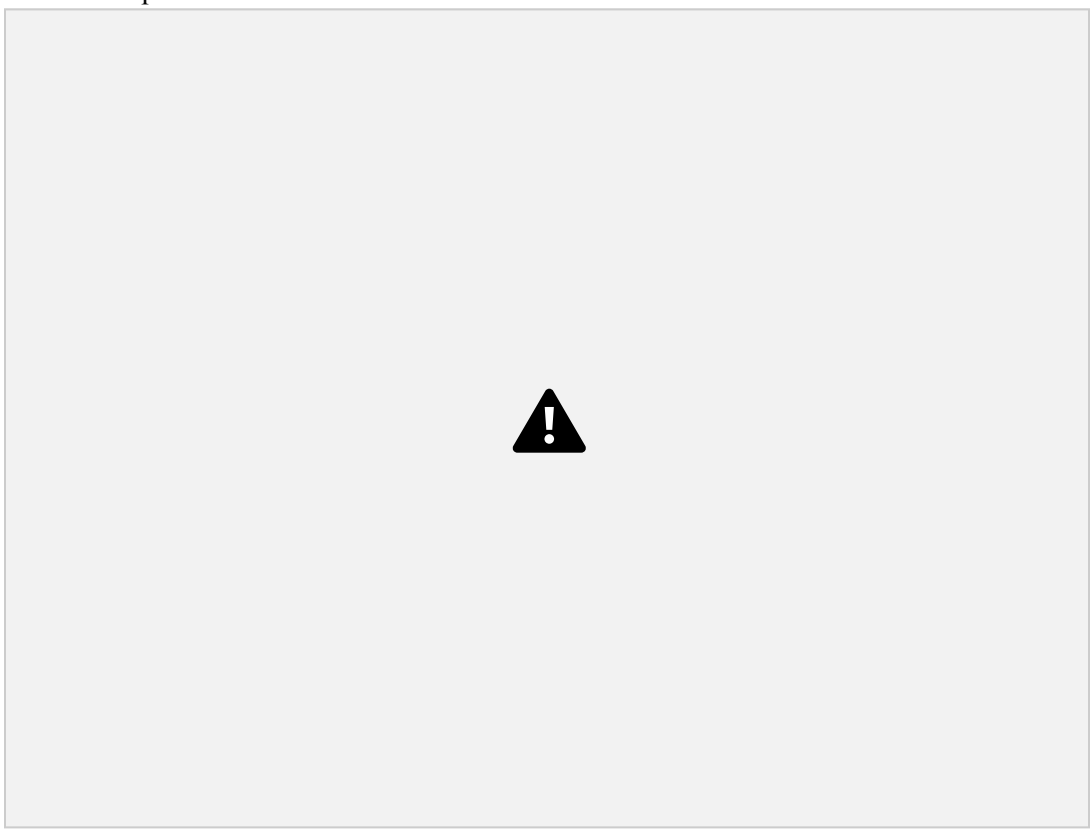
df = spark.sql("""***select c.customer_state, round(avg(datedif (o.order_delivered_customer_date,o.order_purchase_timestamp)),2) as average_delivery_time,***

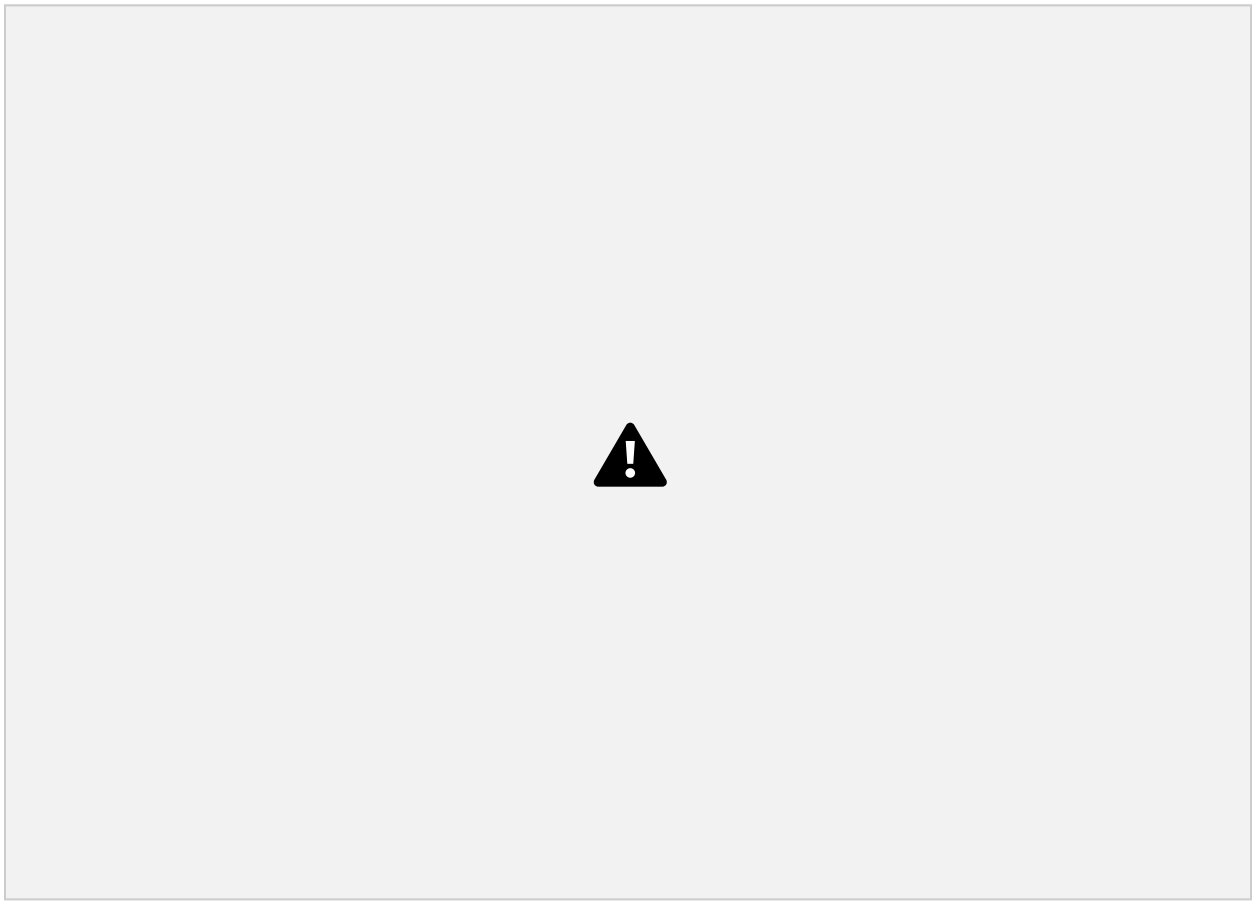
round(sum(p.payment_value),2) as total_revenue_generated from customers c inner join orders o on
c.customer_id = o.customer_id inner join
payments p on o.order_id = p.order_id group by c.customer_state order by average_delivery_time;""")

```
df.show()
df.toPandas()
x_values = [float(row['average_delivery_time']) for row in df.collect()]
y_values = [int(row['total_revenue_generated']) for row in df.collect()]

plt.barh(range(len(y_values)), x_values)
plt.yticks(range(len(y_values)), y_values)
plt.xlabel('Number of days to deliver')
plt.ylabel('Revenue Generated')
plt.title('Delivery Time Vs Revenue Generated')
plt.show()
```

Output:



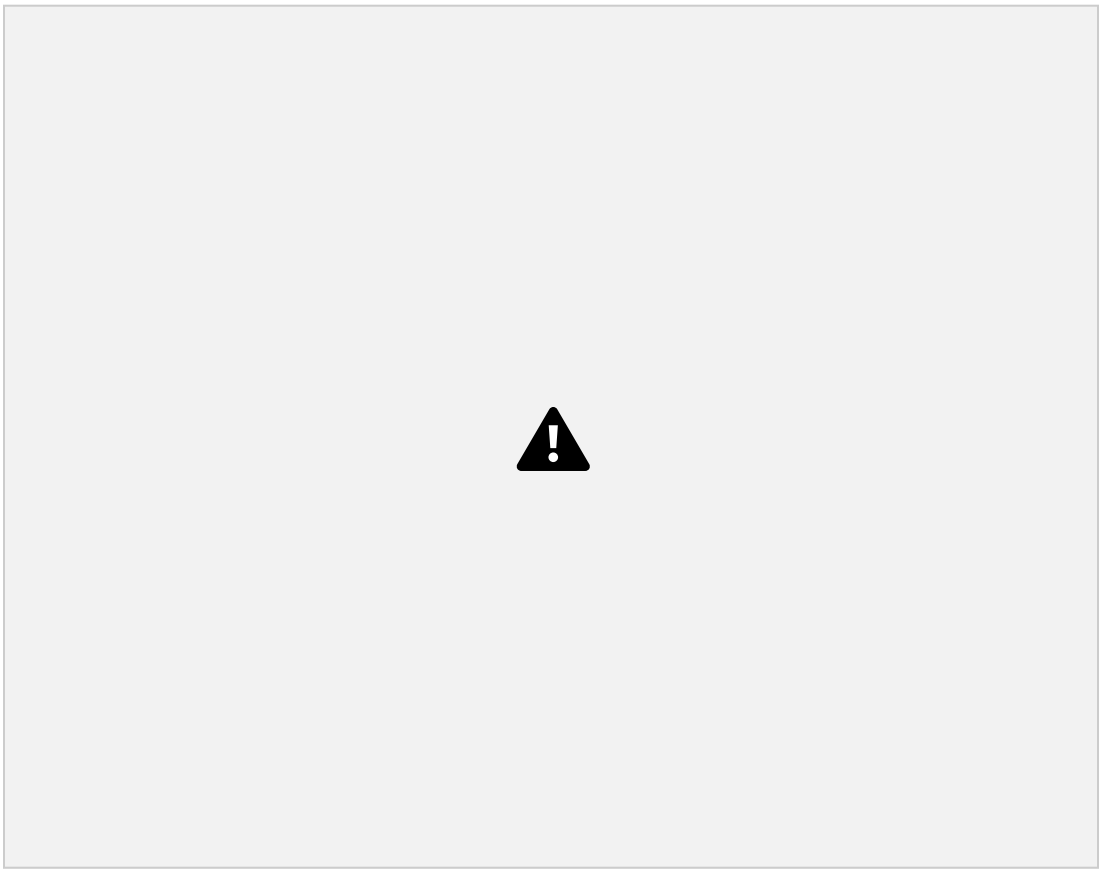


11. How is the price of the products and revenue generated are related?

Query:

spark.sql("""***SELECT p.`product category`, COUNT(oi.order_id) AS order_count,***
round(sum(oi.price),2) as revenue,
round(avg(oi.price),2) as average_of_price
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.`product category` order by revenue desc;""").show()

Output:



12. How is the Delivery time and product reviews correlated?

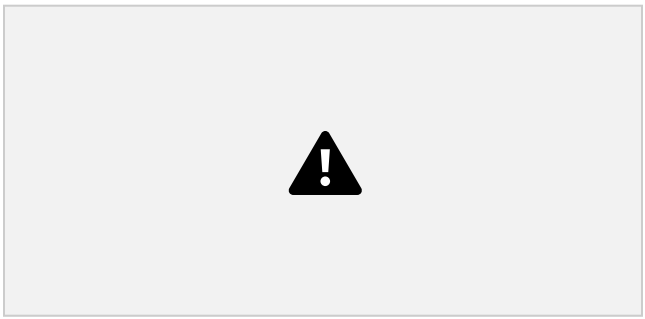
    Query:

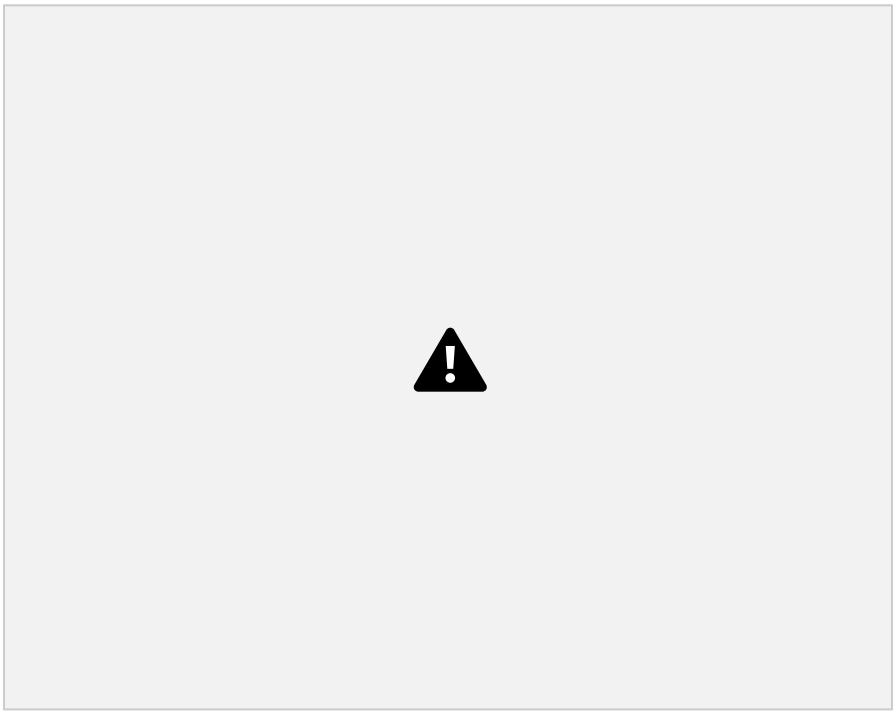
    df = spark.sql("""***select ore.review_score as order_review,***
    round(avg(datedif (o.order_delivered_customer_date,o.order_purchase_timestamp)),2) as average_delivery_time
    from orders o inner join order_reviews ore on
    o.order_id = ore.order_id group by ore.review_score order by average_delivery_time""")

    df.show()
    df.toPandas()
    x_values = [float(row['order_review']) for row in df.collect()]
    y_values = [int(row['average_delivery_time']) for row in df.collect()]

    plt.bar(range(len(y_values)), x_values)
    plt.yticks(range(len(y_values)), y_values)
    plt.xlabel('Order Reviews')
    plt.ylabel('Average Delivery Time')
    plt.title('Order Reviews Vs Delivery Time')
    plt.show()

    Output:





Insights:

The primary objective of this analysis was to gain actionable insights and valuable information from the data sets provided and optimize business solutions to increase revenue or to get profit. The insights that have been found from the above analysis are listed below:

1. Fewer orders are placed at dawn than at other times of day
2. Revenue of Below 1 Lakh is generated from states like rs, pr sc, ba, df, es, go in 3 years
3. Less revenue is generated from states with less number of cities
4. More Revenue is generated on Monday, Tuesday and Wednesday
5. There is 13608.51% increase in Number of orders from 2016 to 2017, but there is only 19.76% increase in number of orders from 2017 to 2018
6. There are 32 orders that took more than 50 days to deliver from the estimated delivery days.
7. Less revenue is generated when there is an increase in the number of days to deliver.
8. Products with moderate price range generate more revenue than products with less or more price range.
9. Delivery time and product reviews are inversely correlated.

Recommendations:

These are the recommendations that can be provided for the further growth of the company by considering the given insights.

1. Offers and coupons can be used in cities that produce less than 100000 in revenue to draw in more customers and maximize revenue.
2. Products can continue to have sales or offers over the weekend to increase weekend sales.
3. Advertisements are needed to draw in more customers as in 2016 to 2017 4. Concentrate on states with more cities that are generating more revenue, maximize the reach in these cities.
5. Consider providing special promotions or incentives on Monday, Tuesday, and Wednesday to encourage more sales and take advantage of consumer purchasing trends since these days generate more revenue.
6. Improving Customer Satisfaction: Identify the factors that made the 32 orders delay by more than 50 days and make sure that the delays in delivering a product is reduced to increase customer satisfaction.
7. Improve Delivery Speed: It's crucial to optimize your delivery operations to cut down on delivery time because revenue declines as delivery days increase.
8. Optimize pricing Strategy: Since products with a moderate price range bring in more money,to increase sales, consider adjusting the product prices to maximize revenue.
9. Enhance product reviews: As there is inverse correlation between delivery time and reviews, focus on improving the delivery experience of the customers by providing tracking information and encourage them to leave a positive review.