Data Analysis with SQL, Python, Jupyter Notebook

To analyze the case - study data after cleaning the data via MS Excel, tools like SQL, Python along with Pandas library and Jupyter Notebook is used.

Since SQL cannot be run directly in Jupyter Notebook, we create a dataframe for the CSV file using Pandas library and then create a table in SQL and connect the dataframe with the table.

The following steps are involved to import and connect the data in SQL database:

import pandas as pd

import sqlite3

data = pd.read\_csv("Data Analysis - Case Study - Data.csv", header = 0)

For analyzing the data in a much easier way, the columns with improper data format are removed to avoid confusion.

data = data.drop(['order\_time', 'store\_reach\_time', 'start\_delivery\_time', 'completed\_cancelled\_time', 'order\_rating'], axis=1)

Now the data framework is connected to SQL server via SQLITE3 available in Jupyter Notebook. A database named ‘dunzo\_da\_assig’ is created to store the table.

db\_conn = sqlite3.connect("da\_assig.db")

c = db\_conn.cursor()

Now a table named ‘data’ is created with matching columns as of dataframe.

c.execute(

"""

CREATE TABLE data (

order\_timestamp TEXT,

order\_timeslot TEXT,

order\_month\_num INTEGER,

order\_month TEXT,

order\_day\_of\_week TEXT,

user\_id INTEGER,

pickup\_geo TEXT,

drop\_geo TEXT,

order\_id INTEGER,

products TEXT,

num\_products INTEGER,

store\_reach\_timestamp TEXT,

start\_delivery\_timestamp TEXT,

completed\_timestamp TEXT,

completion\_flag TEXT,

difference\_delivery\_time INTEGER,

order\_ratings INTEGER,

product\_amount INTEGER,

delivery\_charges INTEGER,

discount INTEGER,

PRIMARY KEY (order\_id)

);

"""

)

The dataframe and table will be connected using the to\_sql function:

data.to\_sql('data', db\_conn, if\_exists='append', index=False)

**Completion\_Rate:**

To determine the completion\_rate from the available data, following query is used:

pd.read\_sql("SELECT

SUM(CASE completion\_flag WHEN 'YES' THEN 1 ELSE 0 END) \* 100.0 / COUNT(completion\_flag) as completion\_rate

from data",db\_conn)

**Completion Rate across day of week and timeslot:**

Day of week:

pd.read\_sql("SELECT count(\*) as Num\_orders, order\_day\_of\_week

from data

where completion\_flag = 'YES'

group by order\_day\_of\_week

order by Num\_orders DESC",db\_conn)

Timeslot:

pd.read\_sql("SELECT count(\*) as Num\_orders, order\_timeslot

from data

where completion\_flag = 'YES'

group by order\_timeslot

order by Num\_orders DESC",db\_conn)

**Segmenting the users based on their order behavior:**

Number of products purchased by users vary at different times. To segment users based on number of products purchased:

pd.read\_sql("SELECT count(\*) Num\_Customers, CASE

WHEN num\_products BETWEEN 1 AND 4 THEN 'Essential buyer' WHEN num\_products BETWEEN 5 AND 10 THEN 'Mid buyer'

WHEN num\_products BETWEEN 11 AND 17 THEN 'Heavy buyer'

ELSE 'Bulk buyer' END as Types\_Customers

from data

group by Types\_Customers

order by Num\_Customers DESC",db\_conn)

The key metrics available from the following data are Average revenue per user (shows how much each customer is spending for purchase via the platform), Retention rates (calculates how many customers would be loyal to the company), Number of events (number of times a customer orders from the platform, or uses the app), Conversion rate (how many times customers actually buys any order via the app/website)

**Retention Rate:**

To calculate the retention rate for each month, the below query is used:

pd.read\_sql("Select first,

SUM(CASE WHEN month\_number = 0 THEN 1 ELSE 0 END) AS January,

SUM(CASE WHEN month\_number = 1 THEN 1 ELSE 0 END) AS February,

SUM(CASE WHEN month\_number = 2 THEN 1 ELSE 0 END) AS March,

SUM(CASE WHEN month\_number = 3 THEN 1 ELSE 0 END) AS April,

SUM(CASE WHEN month\_number = 4 THEN 1 ELSE 0 END) AS May,

SUM(CASE WHEN month\_number = 5 THEN 1 ELSE 0 END) AS June,

SUM(CASE WHEN month\_number = 6 THEN 1 ELSE 0 END) AS July,

SUM(CASE WHEN month\_number = 7 THEN 1 ELSE 0 END) AS August,

SUM(CASE WHEN month\_number = 8 THEN 1 ELSE 0 END) AS September

from (SELECT m.user\_id,m.order\_month\_num,n.first as first,m.order\_month\_num-first as month\_number

from (SELECT user\_id, order\_month\_num FROM data GROUP BY user\_id,order\_month\_num) m,

(SELECT user\_id, min(order\_month\_num) AS first FROM data GROUP BY user\_id) n

where m.user\_id=n.user\_id) as with\_month\_number

group by first

order by first",db\_conn)

**Average Revenue per account**:

pd.read\_sql("SELECT order\_month,

sum(product\_amount + delivery\_charges) AS monthly\_revenue, count(DISTINCT user\_id) AS users,

(sum(product\_amount + delivery\_charges) / count(DISTINCT user\_id)) AS average\_revenue

FROM data

GROUP BY order\_month",db\_conn)

**Customer LifeTime Value:**

To calculate Customer LifeTime Value(LTV), we need few other values to be calculated.

Average purchase value(APV) = total revenue / total number of purchases

Average purchase frequency rate(APFR) = total users / total number of purchases

Average customer value(ACV) = APV \* APFR

Average customer lifespan(ACL) = sum of user lifespan / total number of users

Customer Lifetime Value(LTV) = ACL \* ACV

pd.read\_sql("With purchases\_by\_month as

(SELECT user\_id, order\_month, SUM(product\_amount + delivery\_charges) as total\_purchases\_that\_month

FROM data

GROUP BY 1, 2),

ACV as (SELECT AVG(total\_purchases\_that\_month) as ACV

From purchases\_by\_month),

max\_and\_min\_purchases as (SELECT user\_id,

MAX(product\_amount + delivery\_charges) as most\_recent\_purchase, MIN(product\_amount + delivery\_charges) as first\_purchase

FROM data Group by 1),

Lifespan\_calculation as (SELECT

(most\_recent\_purchase - first\_purchase) / 30.0 as months\_in\_between\_purchases FROM max\_and\_min\_purchases),

average\_lifespan as (SELECT AVG(months\_in\_between\_purchases) as ACL

FROM Lifespan\_calculation)

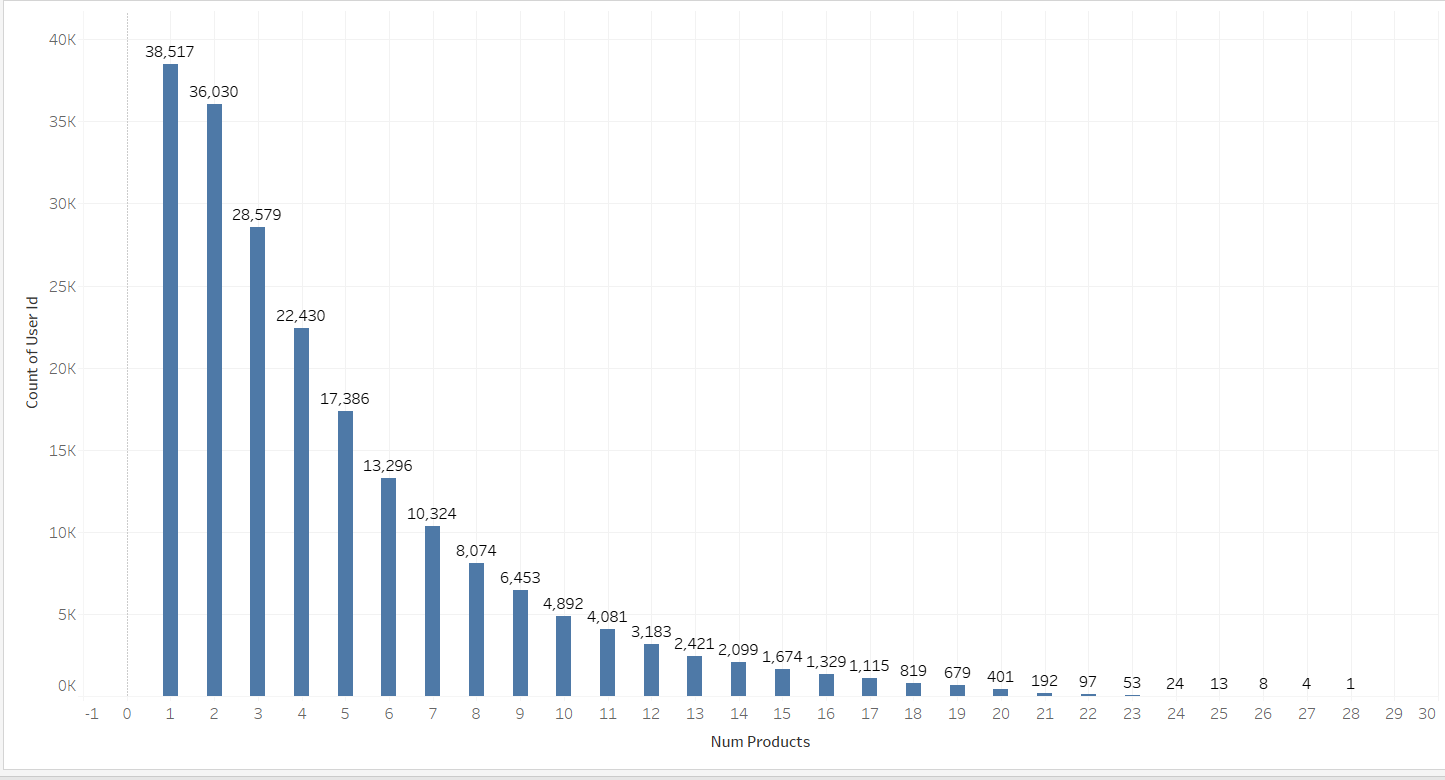
SELECT ACV \* ACL as LTV

From average\_lifespan

join ACV on 1 = 1",db\_conn)

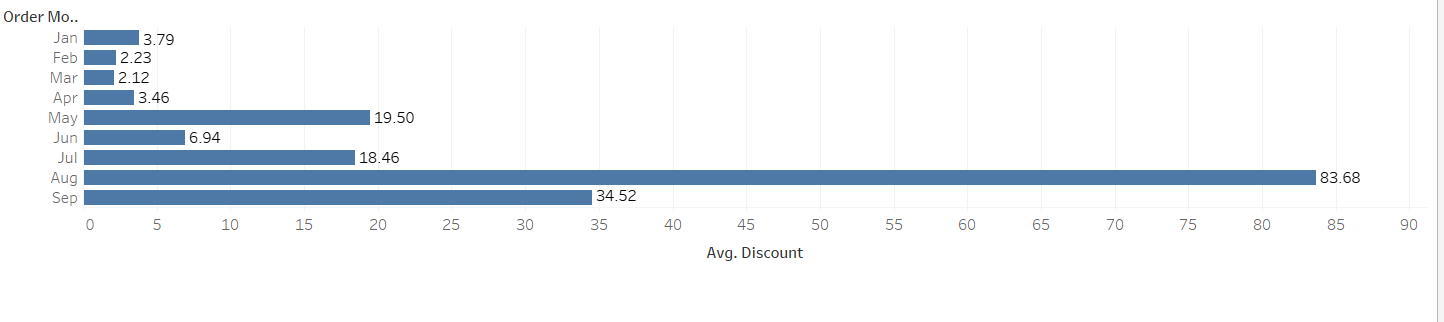
**Relationship between columns via Visualization:**

1. Number of users vs Number of products purchased



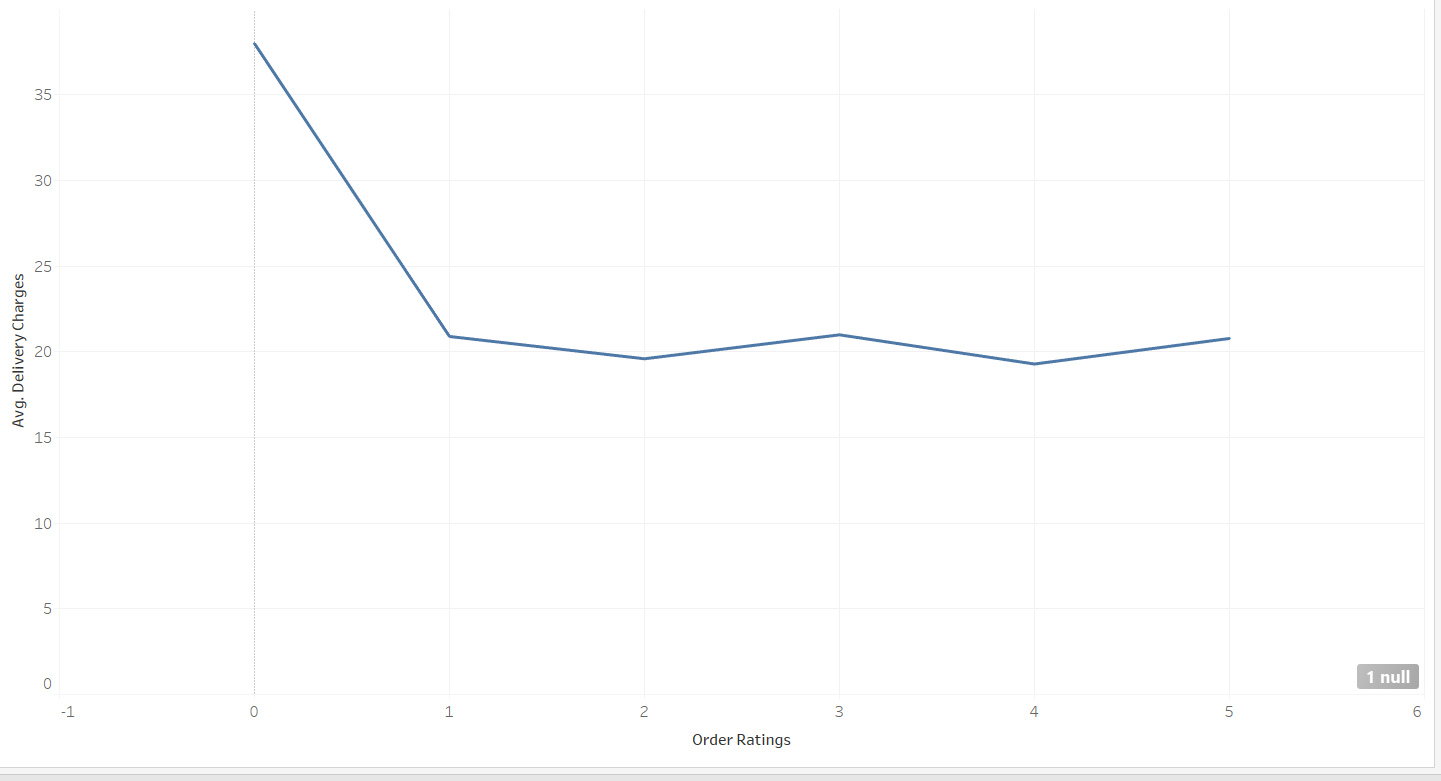
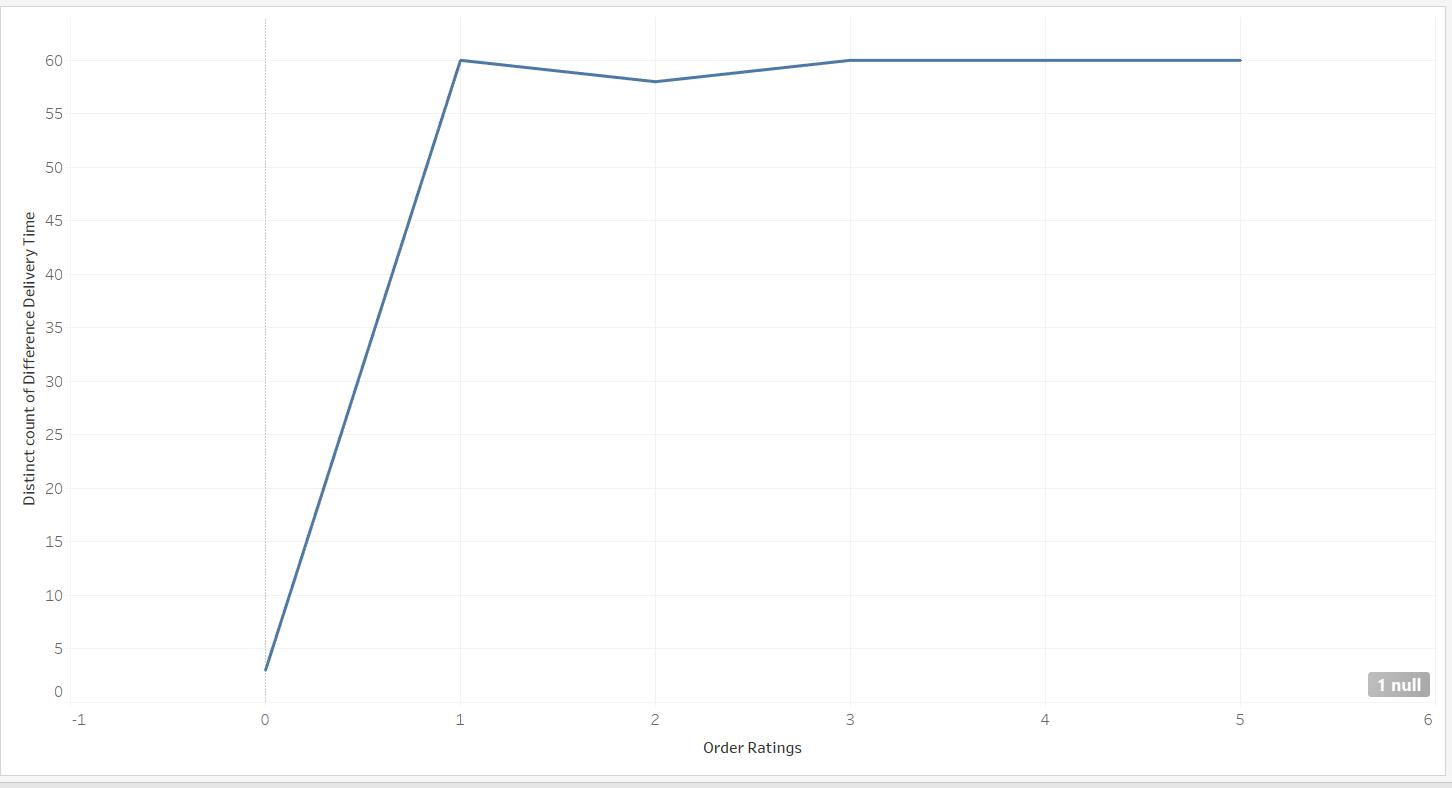
The graph shows a constant decrease in the number of products purchased with respect to the number of users.

1. Discount rate vs Month

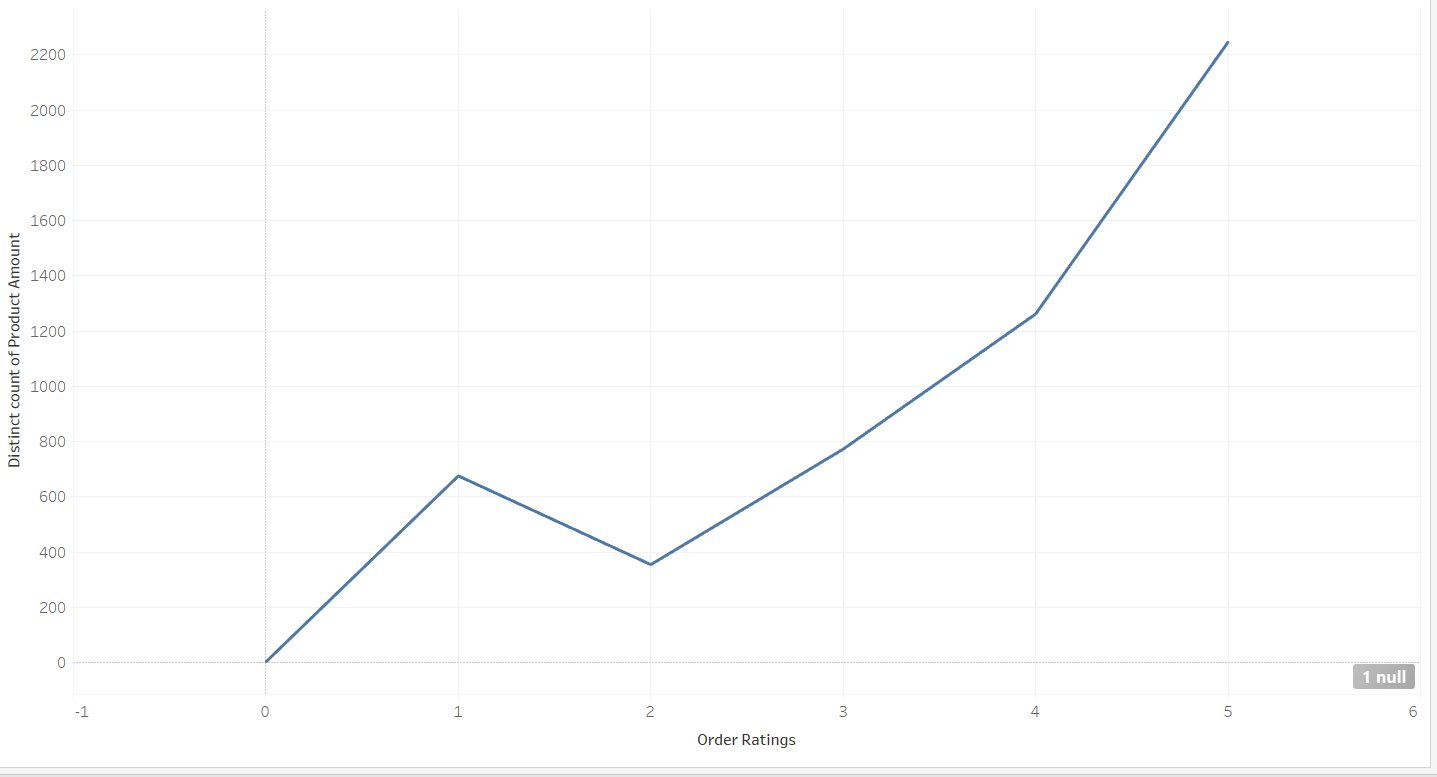
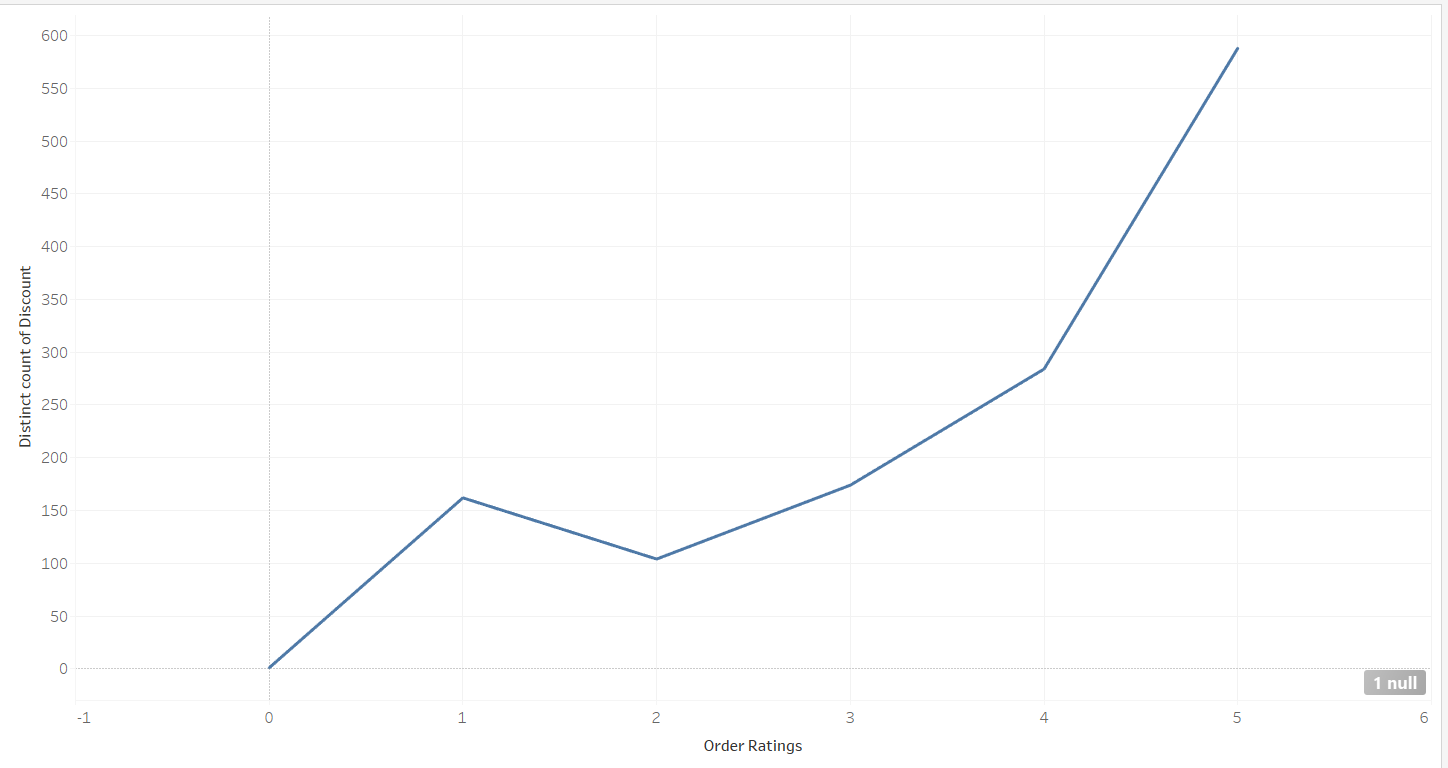


We could see that the discount rates were lower at the start of the year while as the year progressed, there were a decent amount of discounts provided. August records the highest amount of discounts provided.

1. Order ratings correlation with other data-points



From the above 2 graphs, it could be concluded that the delivery charges and difference in delivery time does not impact on ratings.



But there are 2 data points which are impacting the order ratings - discount rates and total product amount.

Higher the discount, higher the rating

Likewise, higher the value of the products, higher the rating.

**Propose how can we improve ratings as well as customer satisfaction:**

From the analysis, it could be concluded that the discount rates and amounts of products have an impact. Hence more discount offers should be provided for new customers.

To raise the retention rate, more offers like combo offers or discount offers should be provided to loyal customers.

To prevent the churn rate, a survey should be held to know the difficulties faced by customers. Also any reason why the product was canceled sometimes.