**VISVESVARAYA TECHNOLOGICAL UNIVERSITY,**
**BELAGAVI 590018**

Report on

# E-Commerce Data Analysis

By

Diksha Jain (1BM17CS024)
Battula Pragati (1BM17CS019)
Chandana Kolli (1BM17CS022)

Under the Guidance of

**Dr. Pallavi G B**
Assistant Professor,
Department of CSE
BMS College of Engineering

Work carried out at

Department of Computer Science and Engineering
BMS College of Engineering
(Autonomous college under VTU)
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019
2020-2021

**BMS COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**


## *CERTIFICATE*

This is to certify that the BIG DATA and ANALYTICS mini Project titled "**E-commerce Data Analysis"** has been carried out by Diksha Jain (1BM17CS024), Battula Pragati (1BM17CS019),Chandana Kolli(1BM17CS022), during the academic year 2020-2021.

Signature of the guide
**Dr. Pallavi G B**
Assistant Professor,
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

# BMS COLLEGE OF ENGINEERING

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## *DECLARATION*

We, Diksha Jain (1BM17CS024), Battula Pragati (1BM17CS019), Chandana Kolli (1BM17CS022), students of 7th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this assignment work entitled "E-commerce Data Analysis" has been carried out by us under the guidance of **Dr. Pallavi G B,** Assistant Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester Aug-Dec 2020. We also declare that to the best of our knowledge and belief, the assignment reported here is not from part of any other report by any other students.

**Signature of the Candidates**

Diksha Jain (1BM17CS024)

Battula Pragati (1BM17CS019)

Chandana Kolli (1BM17CS022)

# CONTENTS

# 1. Objective

This era unlike any, is faced with explosive growth in the size of data generated/captured. Data growth has undergone a renaissance, influenced primarily by ever cheaper computing power and the ubiquity of the internet. This has led to a paradigm shift in the E-commerce sector; as data is no longer seen as the by-product of their business activities, but as their biggest asset providing: key insights to the needs of their customers, predicting trends in customer's behaviour, democratizing of advertisement to suits consumers varied taste, as well as providing a performance metric to assess the effectiveness in meeting customer's needs. Such unique insight can be applied to improve customer service, guide business strategy, and provide democratized services to customers.

# 2. Introduction

## 2.1 E-commerce

Ecommerce hosts a platform that lets businesses sell their products or services throughout the world through an electronic medium. Earlier, ecommerce was referred to as Electronic Commerce. As the definition of ecommerce implies, it includes all sorts of businesses that use the internet for data exchange and/ or money exchange

## 2.2 Big Data analytics in Ecommerce

E-commerce firms are one of the fastest groups of BDA adopters due to their need to stay on top of their game. In most cases, e-commerce firms deal with both structured and unstructured data. In the BDA environment, the challenge is to deal with both types of data in order to generate meaningful insights to increase conversions. A recent study by BSA Software Alliance in the United States (USA) indicates that BDA contributes to 10 % or more of the growth for 56 % of firms. Therefore, 91 % of Fortune 1000 companies are investing in BDA projects, an 85 % increase from the previous year. While the use of emerging internet-based technologies provides e-commerce firms with transformative benefits (e.g., real-time customer service, dynamic pricing, personalized offers or improved interaction), BDA can further solidify these impacts by enabling informed decisions based on critical insights. Specifically, in the e-commerce context, “big data enables merchants to track each user’s behavior and connect the dots to determine the most effective ways to convert one-time customers into repeat buyers”.

## 2.3 Different sources of Big Data in Ecommerce

Big data is often generated by machines, people, and organizations.

a. Machine generated data are often referred to data generated from real time sensors.

b. Human generated data are referred to data generated through use of social media data, status updates, tweets, photos, and others.

c. Organizational generated data is referred to more traditional types of data, including business transaction information.

Data gets generated through following sources:

a. Transaction or business activity data: Structured data from retail transactions, customer profiles, distribution frequency and volume, product consumption and service usage, nature and frequency of customer complaints.

b. Click-stream data: Click-stream data from the web, social media content, online advertisements (tweets, blogs, Facebook wall postings, etc.)

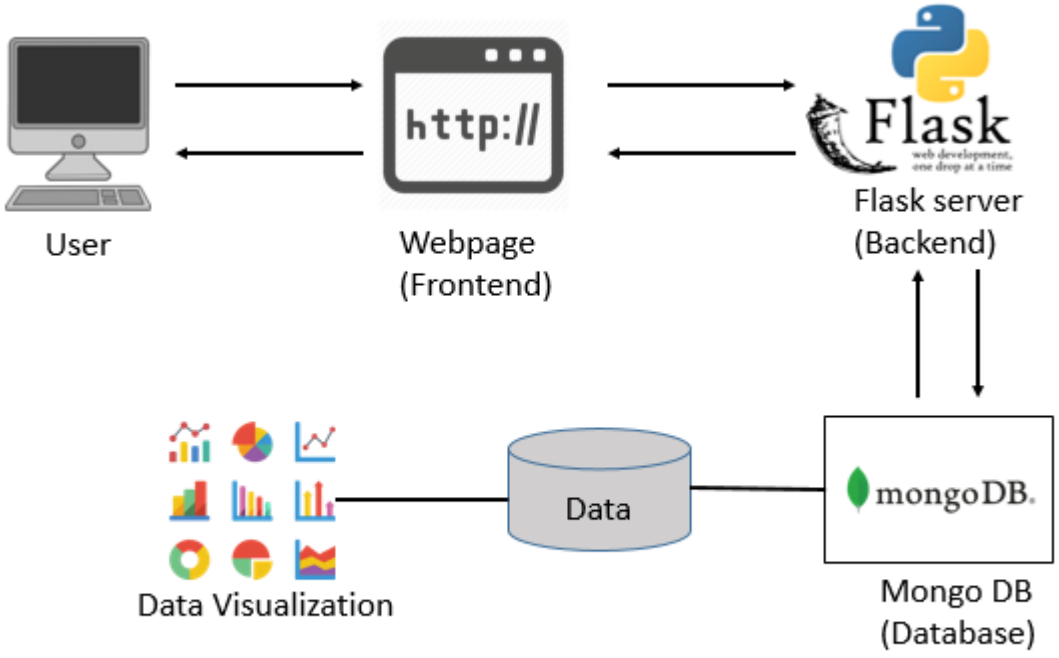
c. Video data: Video data from retail and other settings d. Voice data: Voice data from phone calls, call centers, customer service.

Big data can be either structured, semi-structured, or unstructured. Real value of data comes from combining these streams of big data sources with each other and analyzing them to generate new insights

## 2.4 Dataset Used

Dataset used is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

## 3. Design Modules




**Frontend:** User interface for backend interaction i.e. Create, Read, Update, Delete methods were implemented.

**Backend:** Establishes communication between the database and user interface. Flask server was used as the Backend.

**Database:** To store large data set in one place with easy access. Mongo DB was used.

**Data Visualization:** Visual representation of data enables for easy interpretation of data which was implemented using Jupyter notebook in our project.

# 4. Detailed Description of Modules

## 4.1 Frontend

Within computer programming, the acronym CRUD stands for create, read, update and delete. These are the four basic functions of persistent storage. Also, each letter in the acronym can refer to all functions executed in relational database applications and mapped to a standard HTTP method and mongo DB queries.


**Create**: The create module here creates a new record in the mongo DB database. The user enters all the required information like, Invoice number, stock code, description, date etc. The entered information in the user interface is added to the database. A success message is displayed after the addition of the new record.

```
@app.route("/create", methods=['GET','POST'])
def create():
    #adding an invoice
    if request.method == 'POST':
        InvoiceNo=request.form["InvoiceNo"]
        StockCode=request.form["StockCode"]
        Description=request.form["Description"]
        Quantity=request.form["Quantity"]
        InvoiceDate=request.form["InvoiceDate"]
        UnitPrice=request.form["UnitPrice"]
        CustomerID=request.form["CustomerID"]
        Country=request.form["Country"]
        customers.insert({ "InvoiceNo":InvoiceNo, "StockCode":StockCode, "Description":Description, "Quantity":Quantity,
                           "InvoiceDate":InvoiceDate, "UnitPrice":UnitPrice, "CustomerID":CustomerID, "Country":Country })
        flash('Invoice Added!')
        return render_template('/create.html')
    return render_template('/create.html')
```

**Update**: This module modifies the field quantity of the existing records in the database. The user enters Invoice number and stock code of the item whose quantity has to be modified with new value of the field quantity. The invoice number and stock code are cross checked with the database and the quantity is updated. A corresponding success or failure message is displayed.

```
@app.route("/update",methods=['GET','POST'])
def update ():
    if request.method == 'POST':
        InvoiceNo=request.values.get("InvoiceNo")
        StockCode=request.values.get("StockCode")
        Quantity=request.values.get("Quantity")
        print(InvoiceNo, StockCode, Quantity)
        invoice = customers.find_one({"InvoiceNo":InvoiceNo, "StockCode":StockCode})
        print(invoice)
        newvalues = {"$set": {"Quantity": Quantity}}
        temp=customers.update_one({"InvoiceNo":InvoiceNo, "StockCode":StockCode}, newvalues)
        print(newvalues)
        if(temp.modified_count > 0):
            flash('Invoice updated!')
        else:
            flash("Failed to update the invoice. Enter valid data.")
        return render_template('update.html',invoice=invoice,h=heading,t=title)
    return render_template('/update.html')
```

**Read**: This module displays all the items under a invoice. The user enters the invoice number and corresponding items are fetched and displayed to the user in a tabular format.

```
@app.route("/read")
def read ():
    #Display the Invoice
    InvoiceNo=request.values.get("InvoiceNo")
    form = customers.find({'InvoiceNo':InvoiceNo})
    print(InvoiceNo)
    print(form)
    invoice = []
    for i in form:
        invoice.append(i)
    print(invoice)
    return render_template('/read.html',invoice=invoice,t=title,h=heading)
```

**Delete**: The module deletes a item from the database. The user enters invoice number and stock code in the user interface. Thus, the item is deleted from the database. A corresponding success or failure message is displayed.

```
@app.route("/delete",methods=['GET','POST'])
def remove ():
    #Deleting a Task with various references
    if request.method == 'POST':
        InvoiceNo=request.values.get("InvoiceNo")
        StockCode=request.values.get("StockCode")
        result=customers.delete_one({"InvoiceNo":InvoiceNo,"StockCode":StockCode})
        print(result.deleted_count)
        if(result.deleted_count > 0):
            flash('Item Deleted!')
        else:
            flash('Failed to delete the item.Enter valid data.')
    return render_template('delete.html')
```


## 4.2 Backend

**Flask:** Flask is a micro web framework written in Python. It's based on the Werkzeug WSGI toolkit and Jinja2 template engine. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers,

form validation, and upload handling, various open authentication technologies and several common framework related tools.

It'll act as the central configuration object for the entire application. It is used to set up pieces of the application required for extended functionality, e.g., a database connection and help with authentication.

It is used to set up the routes that will become the application's points of interaction.

## 4.3 Database

**MongoDB:** MongoDB is a cross- platform document- oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. The MongoDB NoSQL database can underpin many Big Data systems, not only as a real-time, operational data store but in offline capacities as well. With MongoDB, organizations are serving more data, more users, and more insight with greater ease — and creating more value worldwide.

## 4.4 Data Visualization

Data Visualization is a discipline of trying to understand data by placing it in a virtual context so that patterns trends, and correlations that might not otherwise be detected can be exposed. In our project we have used matplotlib and plotly libraries provided by python to visualize data.

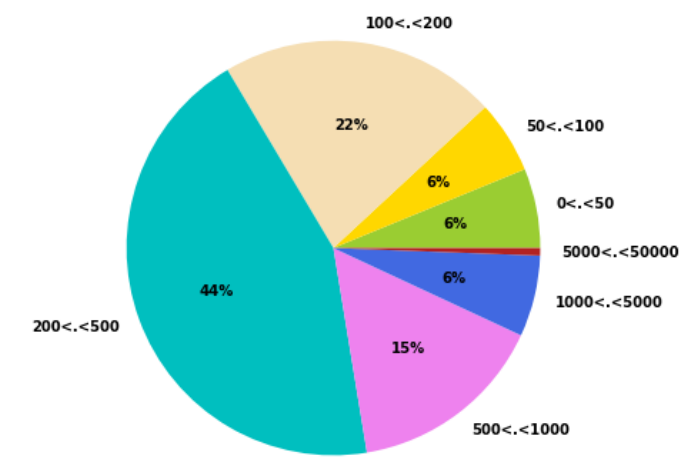
i. **Number of orders per country:** This graph gives us the total number of orders received from all countries present in the dataset. Since the dataset is based on a UK based ecommerce company, United Kingdom was the country with most orders, followed by Germany and France. This Choropleth map was plotted using plotly library.

```
fig = go.Figure(data=go.Choropleth(locations = countries.index,locationmode = 'country names',
    z = countries,text = countries.index,
    colorscale=[[0, 'rgb(224,255,255)'],
            [0.01, 'rgb(166,206,227)'], [0.02, 'rgb(31,120,180)'],
            [0.03, 'rgb(178,223,138)'], [0.05, 'rgb(51,160,44)'],
            [0.10, 'rgb(251,154,153)'], [0.20, 'rgb(255,255,0)'],
            [1, 'rgb(227,26,28)']],
    reversescale=False,marker_line_color='darkgray',marker_line_width=0.5,colorbar_title = 'num orders',
))
fig.update_layout(   title_text='Number of orders per country',
    geo=dict(showframe=True,showcoastlines=True,projection_type='mercator'
    ),
    annotations = [dict(x=0.55,y=0.1,xref='paper',yref='paper',
    )]
)
fig.show()
```

ii. **Purchases divided according to total prices:** This graph shows the analysis of purchases made by customers in terms of percentage based on total price range. The following pie chart was plotted using matplotlib library which shows the maximum number or purchases were made in the price range of 200 to 500 pounds with 44% of customers.

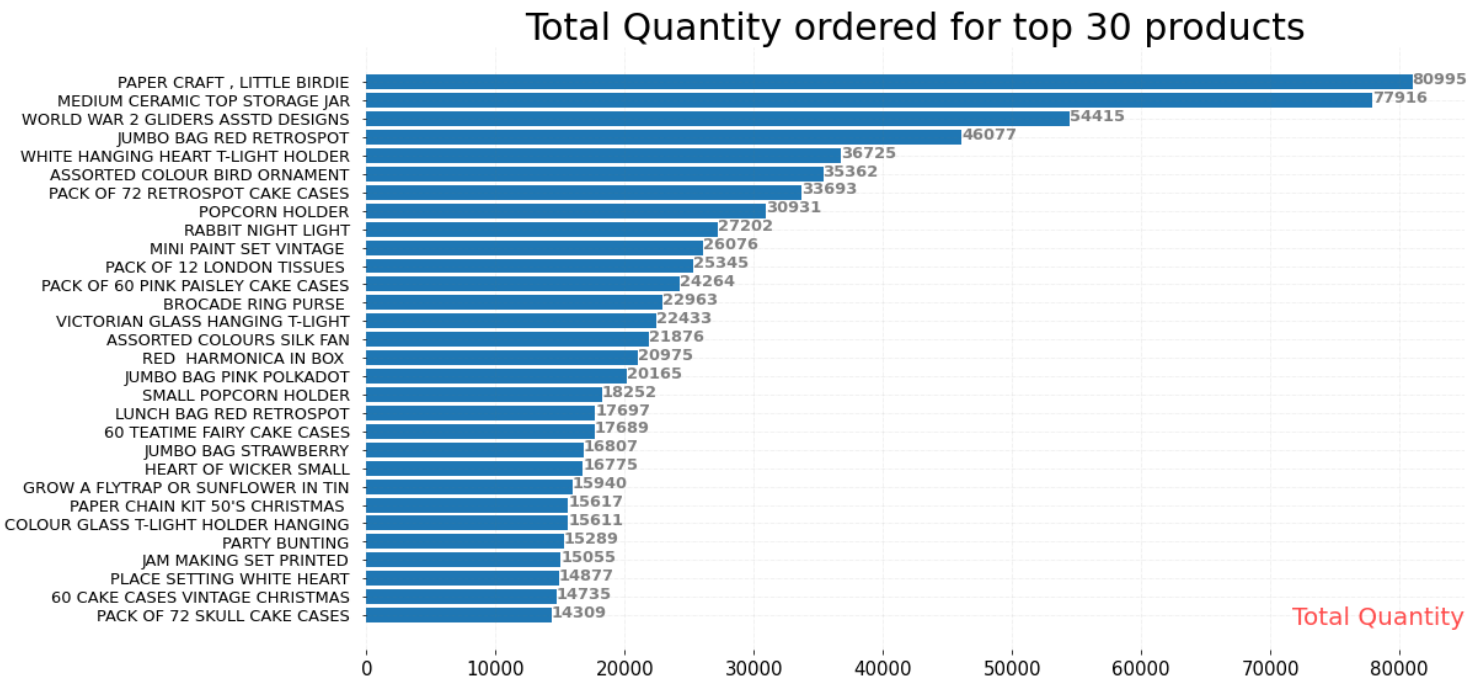
```
#calulating the total price and grouping them into differnt price ranges
remaining_entries = df_cleaned[(df_cleaned['Quantity'] < 0) & (df_cleaned['StockCode'] != 'D')]
df_cleaned['TotalPrice'] = df_cleaned['UnitPrice'] * (df_cleaned['Quantity'] - df_cleaned['QuantityCanceled'])
df_cleaned.sort_values('CustomerID')[:5]
temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['TotalPrice'].sum()
basket_price = temp.rename(columns = {'TotalPrice':'Basket Price'})

basket_price = basket_price[basket_price['Basket Price'] > 0]
basket_price.sort_values('CustomerID')[:10]
price_range = [0, 50, 100, 200, 500, 1000, 5000, 50000]
count_price = []
for i, price in enumerate(price_range):
    if i == 0: continue
    val = basket_price[(basket_price['Basket Price'] < price) &
                       (basket_price['Basket Price'] > price_range[i-1])]['Basket Price'].count()
    count_price.append(val)
#representing in pichart
plt.rc('font', weight='bold')
f, ax = plt.subplots(figsize=(11, 6))
colors = ['yellowgreen', 'gold', 'wheat', 'c', 'violet', 'royalblue','firebrick']
labels = [ '{}<.<{}'.format(price_range[i-1], s) for i,s in enumerate(price_range) if i != 0]
sizes  = count_price
explode = [0.0 if sizes[i] < 100 else 0.0 for i in range(len(sizes))]
ax.pie(sizes, explode = explode, labels=labels, colors = colors,
       autopct = lambda x:'{:1.0f}%'.format(x) if x > 1 else '',
       shadow = False, startangle=0)
ax.axis('equal')
f.text(0.5, 1.01, "purchases divided according to total prices", ha='center', fontsize = 18);
```



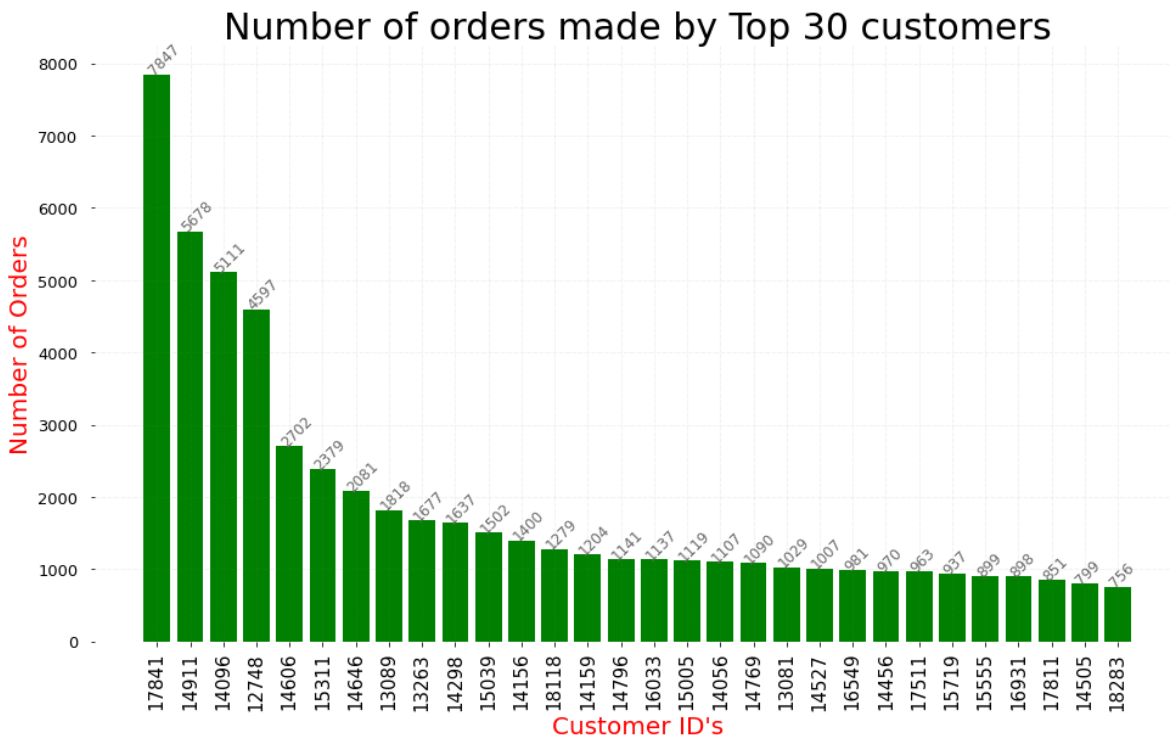
iii. **Total quantity of products ordered:** The following graph shows the total quantity of products ordered for top 30 products. The bar graph was visualized using matplotlib library, indicating the product-paper craft, little birdie- was bought the most with a total quantity of 80995.

```
#grouping description and quantity based on description
temp=df_cleaned[['Description','Quantity']].groupby(['Description']).sum().reset_index()
sorted=temp.sort_values(by=["Quantity"],ascending=False)
top30=sorted[:30]
#plotting horizontal bar graph
plt.rc('font', weight='normal')
fig, ax = plt.subplots(figsize=(16, 9))
ax.barh(top30["Description"],top30["Quantity"])
for s in ['top', 'bottom', 'left', 'right']:
    ax.spines[s].set_visible(False)
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 13)
ax.xaxis.set_tick_params(pad = 5)
ax.yaxis.set_tick_params(pad = 10)
ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha = 0.2)
ax.invert_yaxis()
for i in ax.patches:
    plt.text(i.get_width()+0.2, i.get_y()+0.5,str(round((i.get_width()), 2)), fontsize = 13, fontweight ='bold',
             color ='grey')
ax.set_title('Total Quantity ordered for top 30 products', loc ='center',fontsize=30 )
fig.text(0.9, 0.15, 'Total Quantity', fontsize = 20, color ='red', ha ='right', va ='bottom', alpha = 0.7)
plt.show()
```



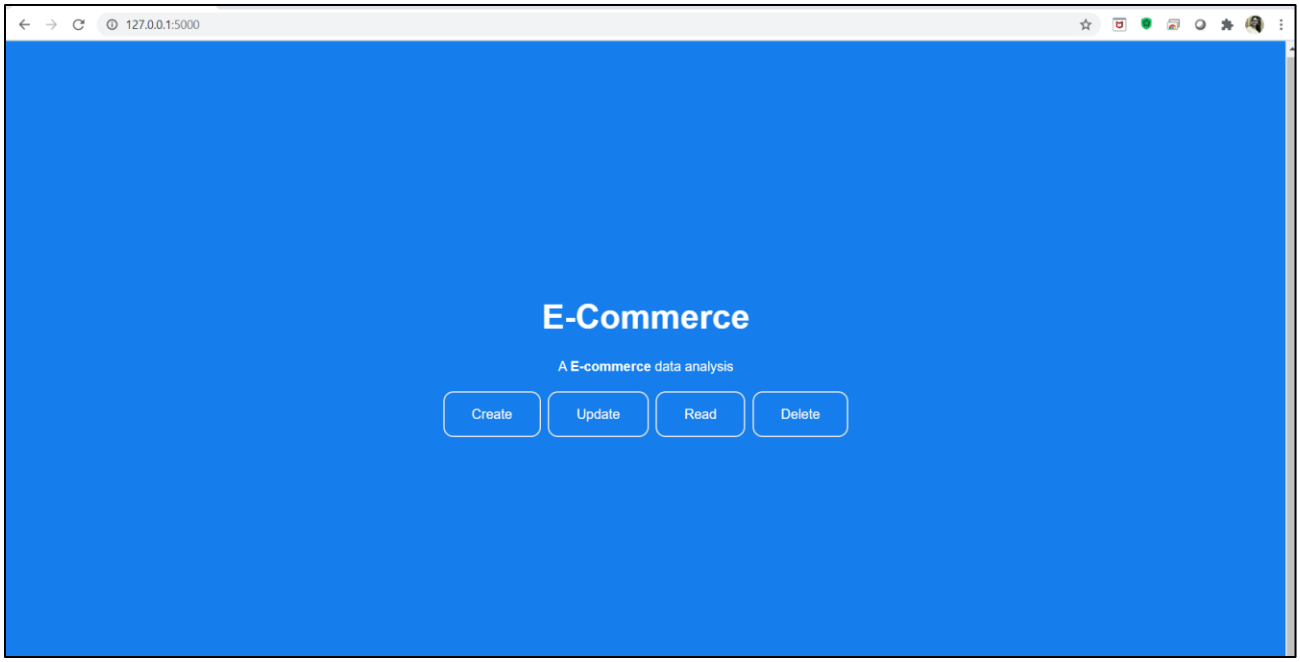
iv. **Number of orders made by customers:** The below graph displays the number of orders made by top 30 customers, by keeping a count of invoice date of each customer. The bar chart was plotted using matplotlib and the customer with ID 17841 has made most purchases i.e. 7847.

```
#grouping customerId and invoice date by customerId to get number of orders made by a customer
temp=df_cleaned[['CustomerID','InvoiceDate']].groupby(['CustomerID']).count().reset_index()
sorted=temp.sort_values(by=["InvoiceDate"],ascending=False)
top30=sorted[:30]
#plotting bar graph
plt.rc('font', weight='normal')
fig, ax = plt.subplots(figsize=(16, 9))
ax.bar(top30["CustomerID"],top30["InvoiceDate"],color="green")
for s in ['top', 'bottom', 'left', 'right']:
    ax.spines[s].set_visible(False)
plt.xticks(fontsize = 15,rotation=90)
plt.yticks(fontsize = 13)
ax.xaxis.set_tick_params(pad = 5)
ax.yaxis.set_tick_params(pad = 10)
ax.grid(b = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha = 0.2)
ax.set_title('Number of orders made by Top 30 customers', loc ='center',fontsize=30 )
plt.ylabel('Number of Orders', fontsize = 20, color ='red')
plt.xlabel("Customer ID's",fontsize = 20, color ='red')
for i in ax.patches:
  plt.text(i.get_x()+.05, i.get_height()+.7, str(round((i.get_height()), 2)), fontsize=12,color='dimgrey',rotation=45)
plt.show()
```

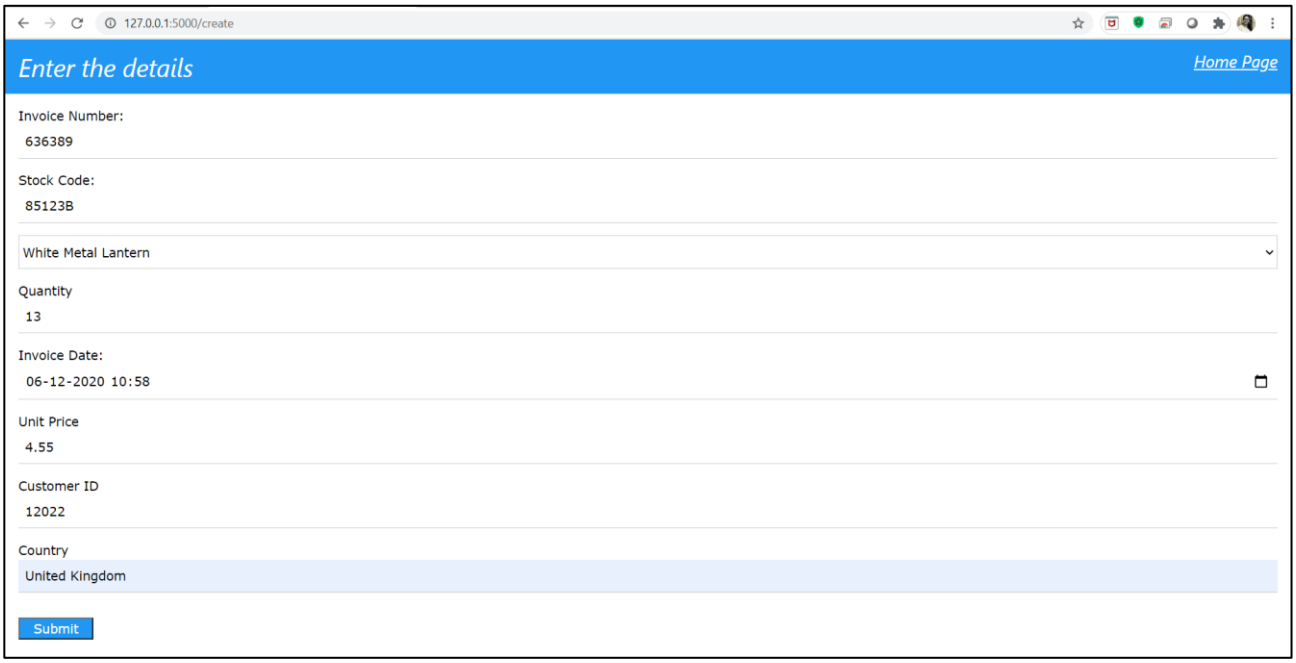


# 5. Screen Shots

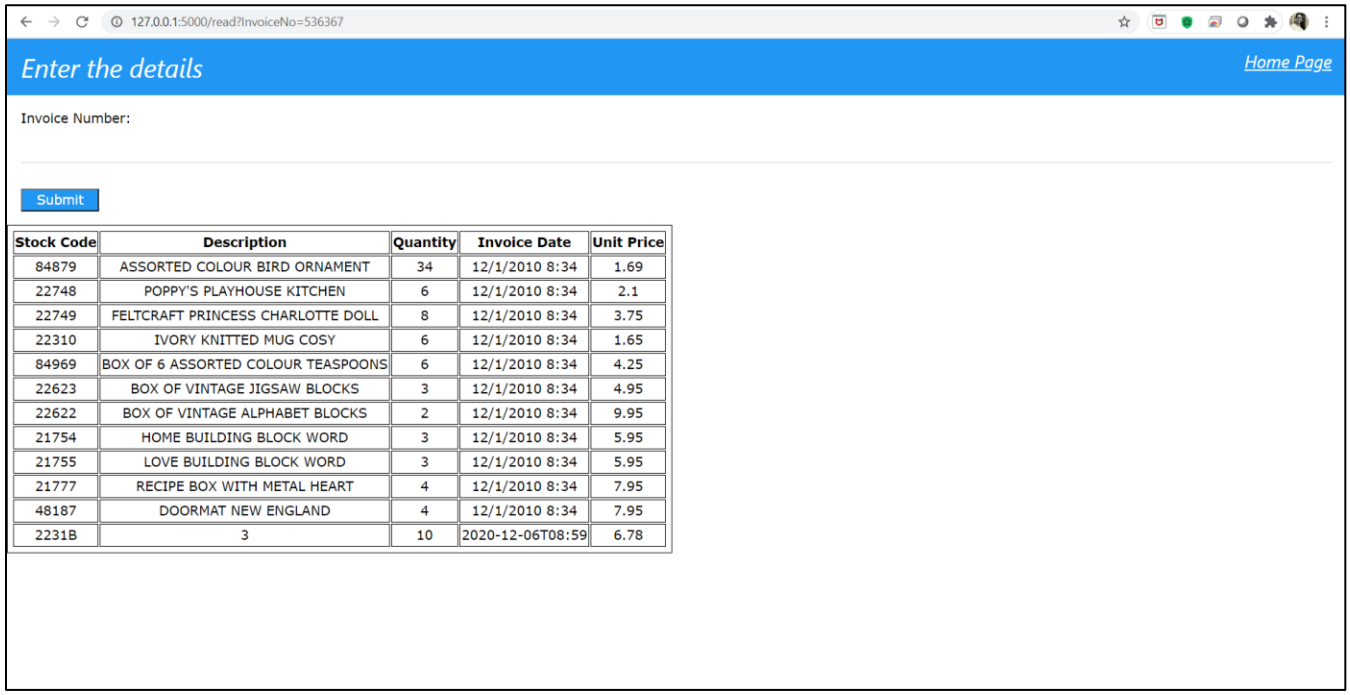
## 5.1 Home page




## 5.2 Create page i.e. insert details into database

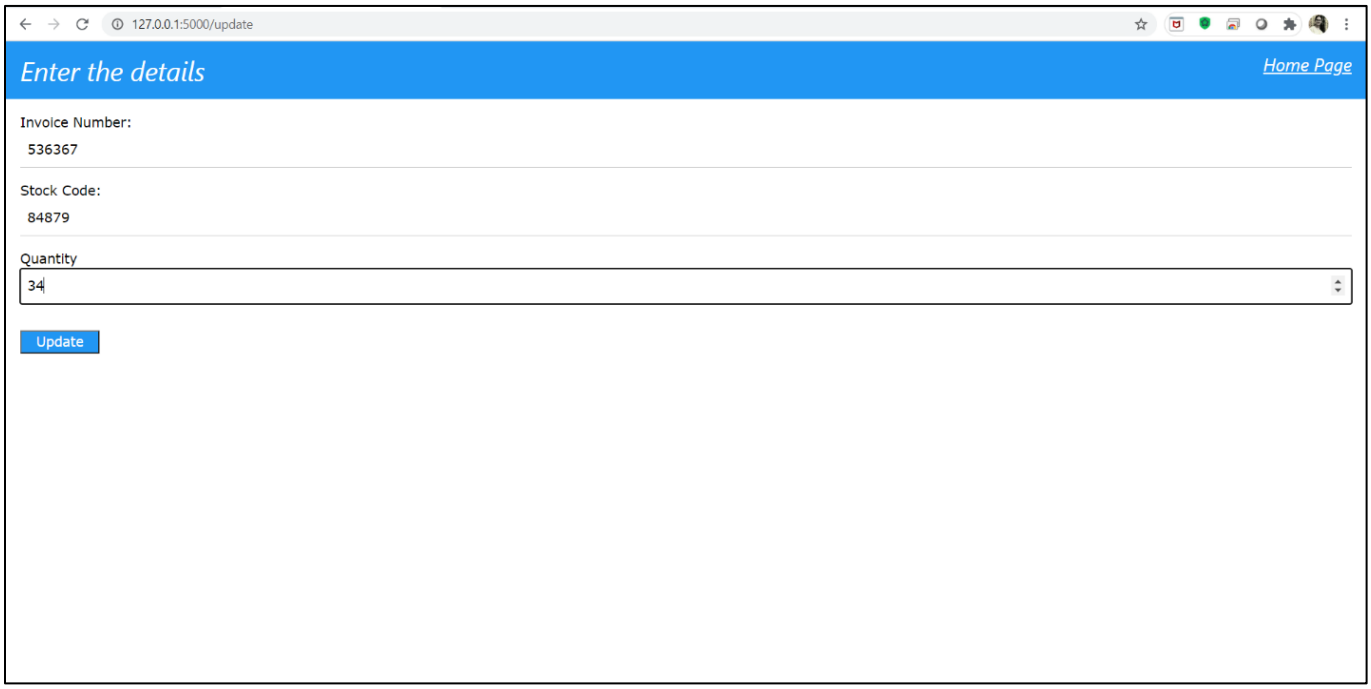


**5.3** Read details for the invoice no. 536367

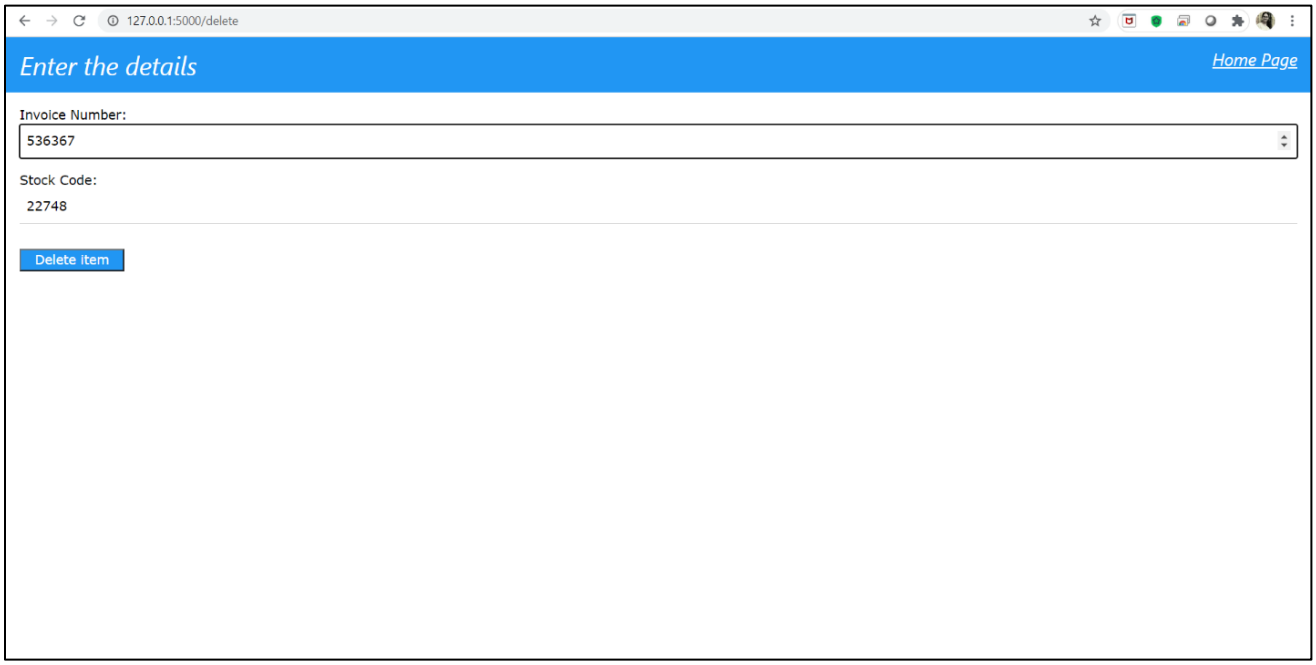



| Stock Code | Description | Quantity | Invoice Date | Unit Price |
|---|---|---|---|---|
| 84879 | ASSORTED COLOUR BIRD ORNAMENT | 34 | 12/1/2010 8:34 | 1.69 |
| 22748 | POPPY'S PLAYHOUSE KITCHEN | 6 | 12/1/2010 8:34 | 2.1 |
| 22749 | FELTCRAFT PRINCESS CHARLOTTE DOLL | 8 | 12/1/2010 8:34 | 3.75 |
| 22310 | IVORY KNITTED MUG COSY | 6 | 12/1/2010 8:34 | 1.65 |
| 84969 | BOX OF 6 ASSORTED COLOUR TEASPOONS | 6 | 12/1/2010 8:34 | 4.25 |
| 22623 | BOX OF VINTAGE JIGSAW BLOCKS | 3 | 12/1/2010 8:34 | 4.95 |
| 22622 | BOX OF VINTAGE ALPHABET BLOCKS | 2 | 12/1/2010 8:34 | 9.95 |
| 21754 | HOME BUILDING BLOCK WORD | 3 | 12/1/2010 8:34 | 5.95 |
| 21755 | LOVE BUILDING BLOCK WORD | 3 | 12/1/2010 8:34 | 5.95 |
| 21777 | RECIPE BOX WITH METAL HEART | 4 | 12/1/2010 8:34 | 7.95 |
| 48187 | DOORMAT NEW ENGLAND | 4 | 12/1/2010 8:34 | 7.95 |
| 2231B | 3 | 10 | 2020-12-06T08:59 | 6.78 |

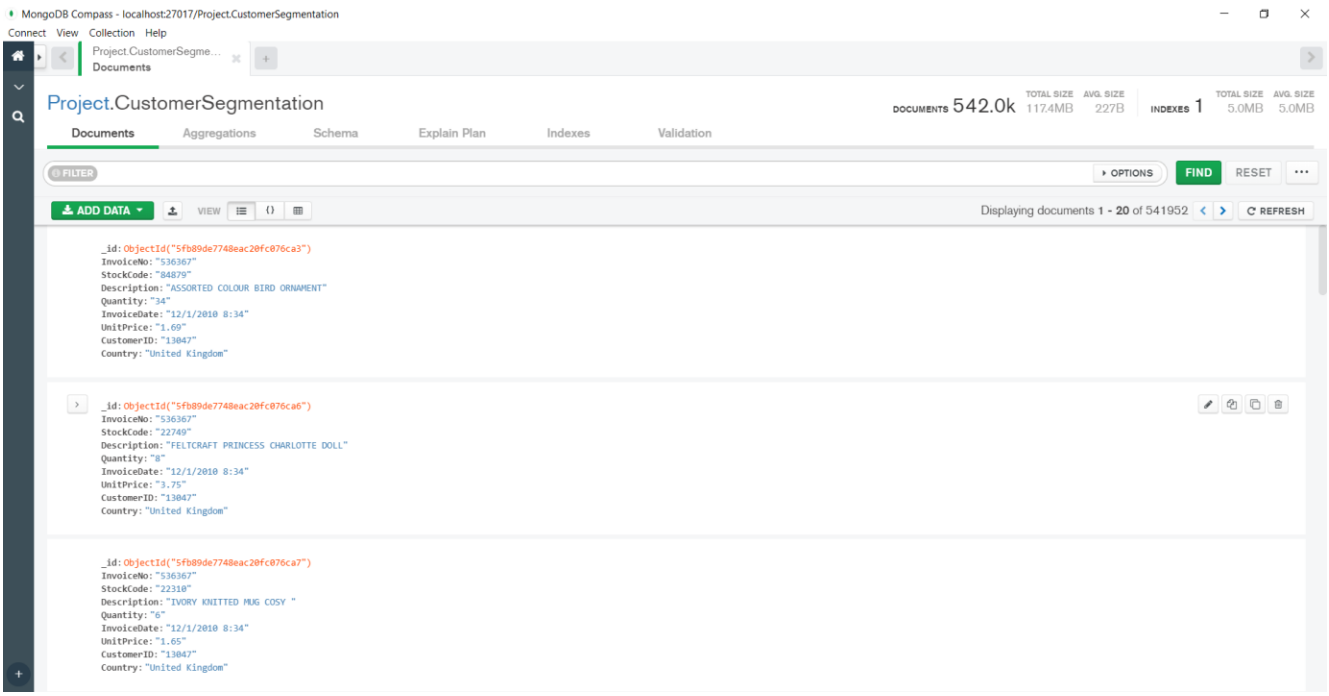
**5.4** Update page i.e. update quantity for a given invoice number and stock code



**5.5** Delete a unique entry from the database based on invoice number and stock code




**5.6** Mongo DB database



## 6. New Learnings

In this project, we implemented Mongo DB database along with Python (Flask) as backend. Flask is a light-weight web framework. The main Flask extensions used are Flask Mongo Engine and Flask RESTful. The first lets us build the templates for our data and the latter is designed for building an API making the process simpler. An API allows us to serve data over the web, by exposing endpoints to make requests. The visualization helps in easy presentation of the data to the common layman.

- Learnt the usage of flask python web framework.
- Usage of mongo DB in big data analytics.
- Learnt on pandas and numpy in data visualization and analysis in python.
- We learnt about the visualization tool Matplotlib and plotly provided by python.

## 7. Future Enhancements

Using this dataset, companies can identify the several segments of customers allowing them to target the potential user base. We can identify the division of customer base into several groups of individuals that share a similarity in different ways that are relevant to marketing such as gender, age, interests, and miscellaneous spending habits. Through the data collected, companies can gain a deeper understanding of customer preferences as well as the requirements for discovering valuable segments that would reap them maximum profit. This way, they can strategize their marketing techniques more efficiently and minimize the possibility of risk to their investment.

- A machine learning model can be implemented to gain further insights into the data.
- Analyses for this dataset could include time series, clustering, classification and more.