



Sales_data analysis PPT

Name : Shrihari Naik

Contents :

- Data Description
- Data Pre-Processing
- Feature Handling
- Question 1
- Question 2
- Question 3
- Question 4
- Conclusion



Data Description:

-

Import file date.xlsx:

- File imported using pandas library in python.

In [3]:

```
date = pd.read_excel("../input/sales-data/date.xlsx")  
date.head()
```

Out[3]:

	timestamp	date	day_name	day_of_month	month_of_year	time_of_day(hh:m
0	2010-12-01 08:26:00	2010-12-01	Wednesday	1	December	08:26:00
1	2010-12-01 08:26:00	2010-12-01	Wednesday	1	December	08:26:00
2	2010-12-01 08:26:00	2010-12-01	Wednesday	1	December	08:26:00

Import file sales_data.xlsx:

- File imported using pandas library in python.

```
In [2]: sales = pd.read_excel("../input/sales-data/sales_data.xlsx")
sales.head()
```

Out[2]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
			KNITTED UNION		2010-12-01			United Kingdom

File exploration :

- There are two files sales_data.xlsx and date.xlsx.
- Date.xlsx file contain the data of transaction time, date, month, day.
- sales_data.xlsx file contain the information about the customer id, product id, quantity sold, unit price, transaction date, timestamp, transaction country, transaction id.



Data Pre-Processing:

-

Concatenation of dataframes:

- I imported two files sales_date.xlsx and date.xlsx using pandas and stored it in the pandas dataframe.
- Then dataframe date is merge in sales.

```
[7]: df = [sales, date]
sales = pd.concat(df, axis=1, sort=False)
sales.head()
```

Out[7]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country	timestamp	date	day_name	day_of_month	month_of_year	time_of_day(hh:mm:ss)
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010-12-01 08:26:00	2010-12-01	Wednesday	1	December	08:26:00
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12-01 08:26:00	2010-12-01	Wednesday	1	December	08:26:00
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010-12-01 08:26:00	2010-12-01	Wednesday	1	December	08:26:00

Remove duplicates from dataframe:

- There are some duplicates value in the dataset.
- Duplicates values can harm the analysis, so need to drop duplicates values.

[67]:

```
print('sales_data duplicated:{}'.format(sales.duplicated().sum()))  
sales.drop_duplicates(inplace=True)
```

```
sales_data duplicated:5268
```

Remove unwanted rows:

- some of rows contain negative quantity sold.
- Sold quantity can not be negative, so that's why it's bad data.
- Using drop we can remove the unwanted rows.



```
sales[sales["quantity sold"] < 0].shape
```

Out[169] (10587, 14)

```
[170]: sales.drop(sales[sales["quantity sold"] < 0].index, inplace=True)
```

```
[171]: sales[sales["quantity sold"] < 0].shape
```

Out[171] (0, 14)



Feature Handling:

-

Drop features:

- Some of columns need to be dropped.
- Like timestamp because we already have date to use for.
- Time related data can be dropped. Because we have data for the whole year. So time data might not be important.

```
sales.drop(['timestamp', 'transaction timestamp', 'time_of_day(hh:mm:ss)'], axis=1, inplace=True)
```

Create new feature:

- New feature created from existing feature.
- total = quantity sold * unit price
- total can simplify the further analysis.

[15]:

```
sales["total"] = sales["quantity sold"] * sales["unit price"]
sales.head()
```

Out[15]:

	transaction id	product id	product description	quantity sold	unit price	customer id	transaction country	date	day_name	day_of_month	month_of_year	time_of_day(hh:mm:ss)	total
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.55	17850.0	United Kingdom	2010-12-01	Wednesday	1	December	08:26:00	15.30

Information about visualization :

- In this assignment we are going to use Matplotlib and Seaborn.
- Both are famous Python libraries for visualization.
- To understand how we draw the graph, we are going to put some snippets in the page for clarity.



Question 1 :

-

1. Is the company's performance improving or degrading overtime?

- To answer this question, I am using the graph between the “total vs date”.
- But first to reach there we need to groupby by date and then we can get the sum of total for respective day.




```
datewise_total = sales.groupby("date").agg({"total": "sum"})  
datewise_total.head(2)
```

Out[174]:

total	
date	
2010-12-01	58776.79
2010-12-02	47629.42

Resampling of Data :

- After that we can do resampling for more clarity. So we can use resample for purpose and resample data by month.
- We have to take sum of total in the particular month.



```
datewise_total = datewise_total.resample("M").sum()  
datewise_total.head()
```

Out[190]:

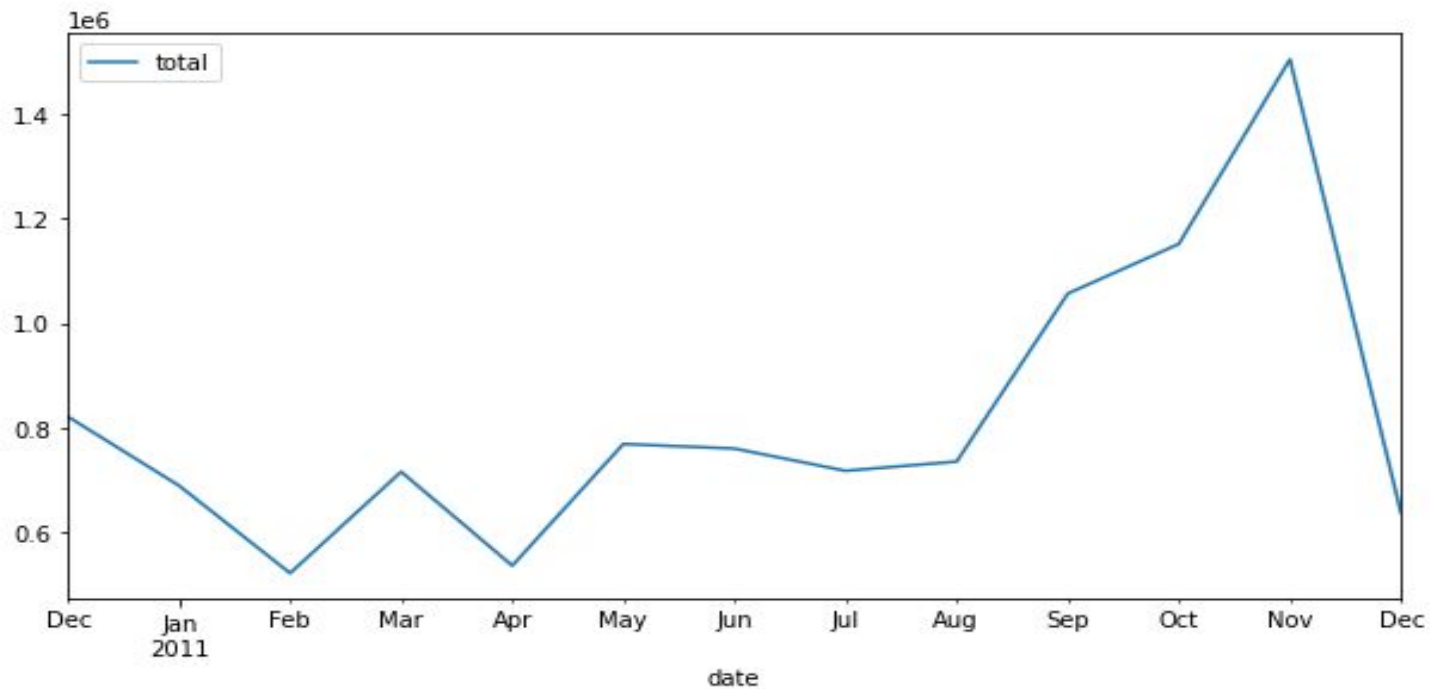
total	
date	
<hr/>	
2010-12-31	821452.730
2011-01-31	689811.610
2011-02-28	522545.560
2011-03-31	716215.260
2011-04-30	536968.491

Graph :

[24]:

```
datewise_total.plot(figsize=(10, 5))
```

date vs total



- From above graph it is clear that the total i.e quantity sold X unit price Increased in last few month.
- But highly decreased in Dec 2011.
- So it is clear that the sales of product are depending upon the month.
- In Feb 2011 the sales are lowest, and Highest in November 2011.
- To answer the above question maybe more data is required.

FBProphet:

- FBprophet is the time-series forecasting algorithm developed by Facebook.
- The instantiation of algorithm:

```
fb_model = Prophet()  
fb_model.add_seasonality(name="monthly", period=7, fourier_order=5)  
fb_model.fit(datewise_total_reset)
```

Use of FBProphet:

- So I used FBProphet forecasting algorithm for next two month.
- After that I got following result.


```
[259]: forecast = fb_model.predict(test_datewise_total)
forecast_result = forecast[["ds", "yhat"]]
forecast_result.rename(columns={"ds": "date", "yhat": "total"}, inplace=True)
forecast_result.set_index("date", inplace=True)
forecast_result.head()
```

Out[259]:

	total
date	
2011-12-10	95341.134440
2011-12-11	45645.841894
2011-12-12	67190.632461
2011-12-13	71945.924909
2011-12-14	65151.290052

Graph explanation:

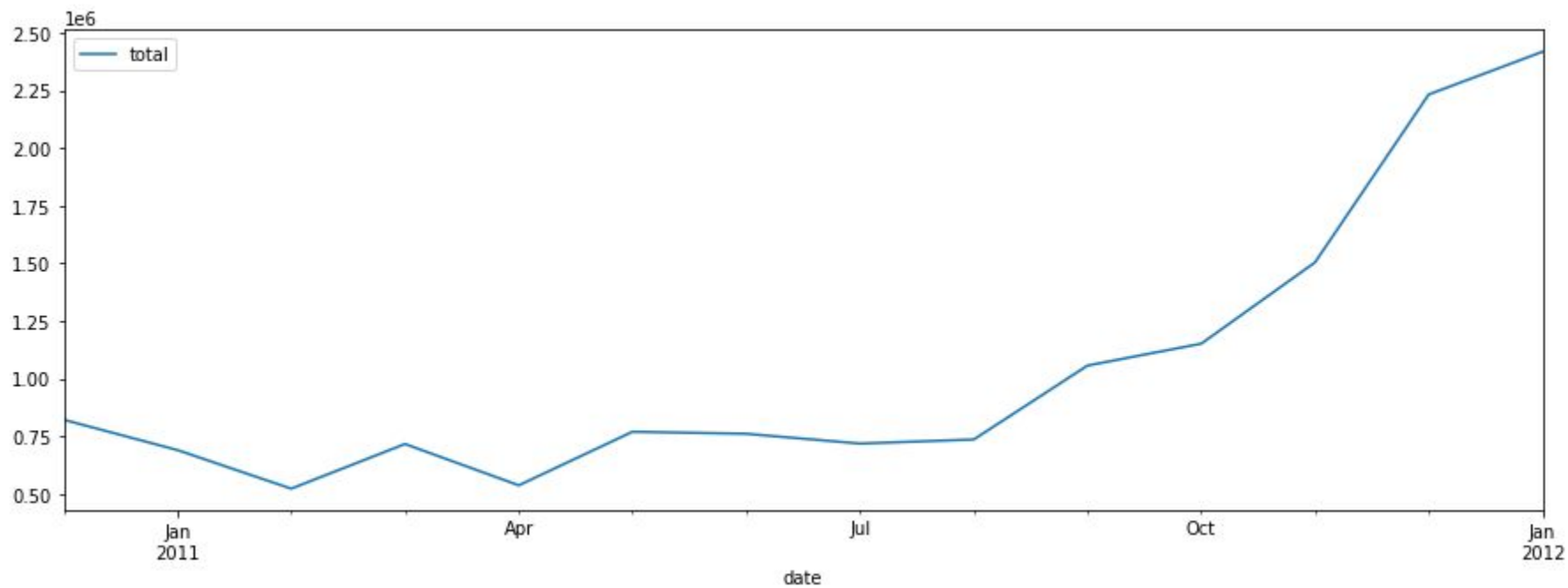
- The result of forecasting then concatenated with the datewise total dataframe to create a full graph.
- After that again resampled the data monthly.



```
df = pd.concat([datewise_total, forecast_result], axis=0, sort=False)
df = df.resample("M").sum()
df.plot(figsize=(15, 5))
```

Graph :

date vs total



Answer:

- From above graph it is clear that the companies performance increased continuously after the month of August.
- Before in Jan, Feb performance were decreased, but again in march it improved.
- In that period of time the performance were not constant.
- But after August it improving.



Question 2:

-

2. Examine and highlight important trends visible in the sales data and insights.

- To highlight important trends we are going to use the graph between the date and quantity sold.
- Quantity sold is chosen for the trend because it will clearly shows number of product bought by the customers.
- So for that we are going to do some preprocessing on the sales dataframe.

Pre-processing :

- We are going to use groupby.
- Here we are resampling the dataset by monthly, to understand graph more clearly.



```
datewise_quantity = sales.groupby(["date"]).agg({"quantity sold": "sum"})  
datewise_quantity = datewise_quantity.resample("M").sum()  
datewise_quantity.head()
```

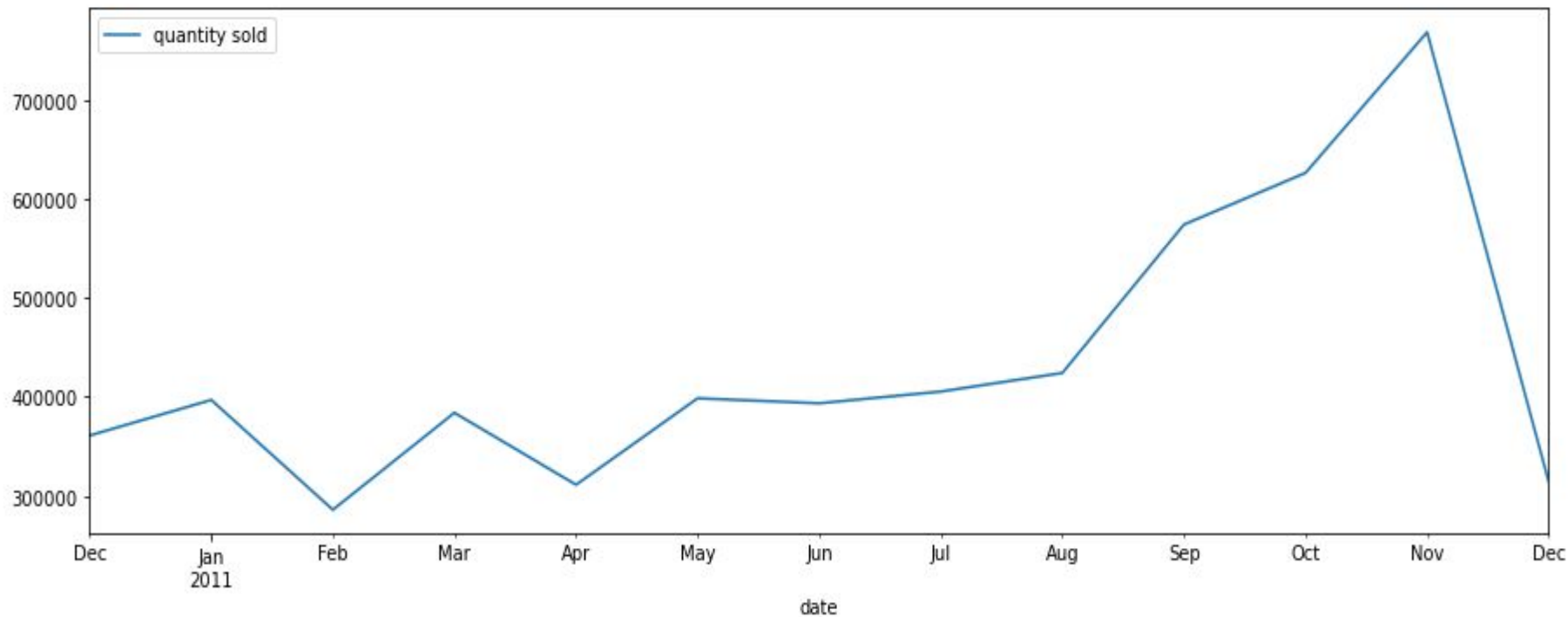
Out[79]:

quantity sold	
date	
2010-12-31	361094
2011-01-31	397030
2011-02-28	286074
2011-03-31	384023
2011-04-30	311314

Graph :

```
datewise_quantity.plot(figsize=(15,5))
```

date vs quantity_sold



Answer:

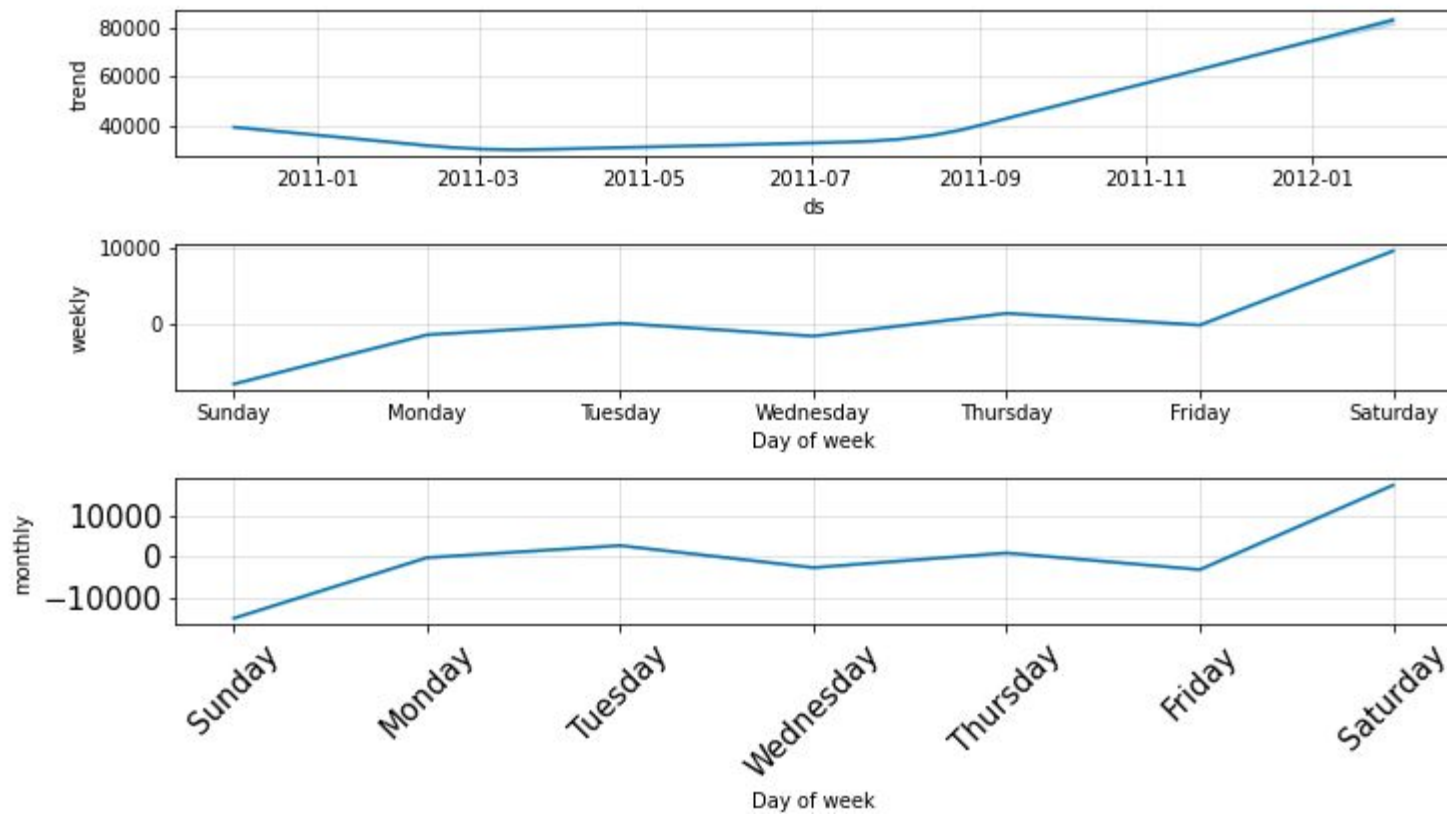
- From above graph it is clear that there are some ups-down in the few months.
- Company sold very less products in Feb.
- Graph goes to the peak in the month of November.
- If you see graph carefully, after May graph growing high.
- After April the growth rate is high, till November.
- After November growth rate is in the negative.

Use of forecasted dataset to show trend:

- As you previously forecasted the next two months total values.
- So we can use that dataset for finding trends.
- There is already available function in FBProphet to show the trends.
We can also refer to that.
- Using this graph it is easy to understand trend of the sales.

```
fig = plot_components(fb_model, forecast_result, figsize=(10, 5))  
ax = fig.gca()  
ax.tick_params(axis="x", labelsz=15, rotation=45)  
ax.tick_params(axis="y", labelsz=15)
```

Graph :



Answer :

- From above graph it is clear that after Dec 2010 sales goes down, till March.
- After march it showing the constant sales.
- But after August, sales picking growth.
- So that's good sign for our company.
- Second graph showing the weekly trend of the sales.
- And the third one showing the monthly trend of the sales.



Question 3:

-

3. How can we measure our performance in terms of customer acquisition and building customer loyalty?

a. What kind of customer do typically buy from us?

- Basically our 90 % customer from the United Kingdom.
- Basically most of the business in the United Kingdom.

```
customer_count = sales.groupby("transaction country").agg({"customer id":"nunique"})  
print("total customers : ", customer_count["customer id"].sum())  
print("total customers in UK : ", customer_count[customer_count.index=="United Kingdom"].values[0][0])
```

```
total customers : 4347  
total customers in UK : 3921
```

Pre-processing [for customer count]:

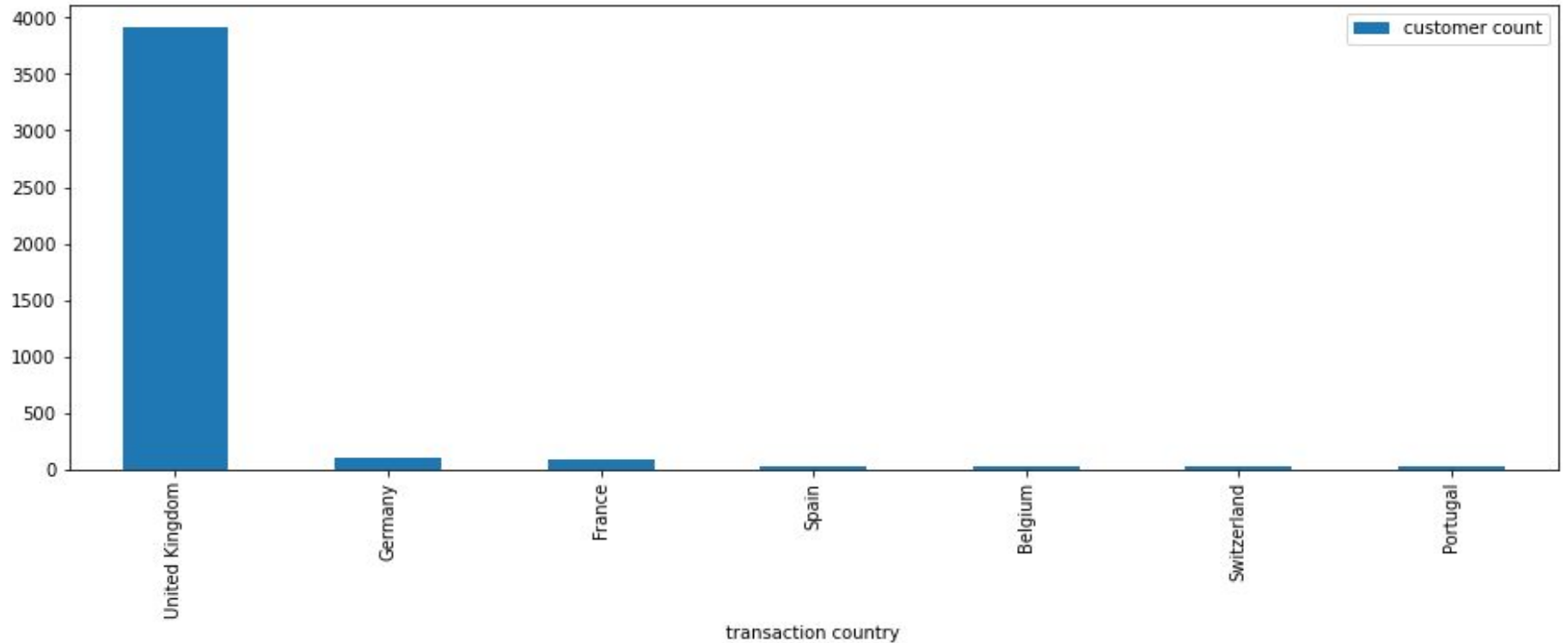
- For next graph need to do some preprocessing on dataframe sales.
- Here we use groupby on the sales data by transaction country and take the number of unique customer id from transaction country.
- After we used only top 7 countries for further analysis, for more clarity and also other countries has a few number of customers.



```
customer_count = sales.groupby("transaction country").agg({"customer id": "nunique"})
customer_count = customer_count.sort_values("customer id", ascending=False).head(7)
customer_count.rename(columns={"customer id": "customer count"}, inplace=True)
customer_count.plot.bar(figsize=(15, 5))
```

Graph :

Transaction country vs customer count



Pre - processing [for product description]:

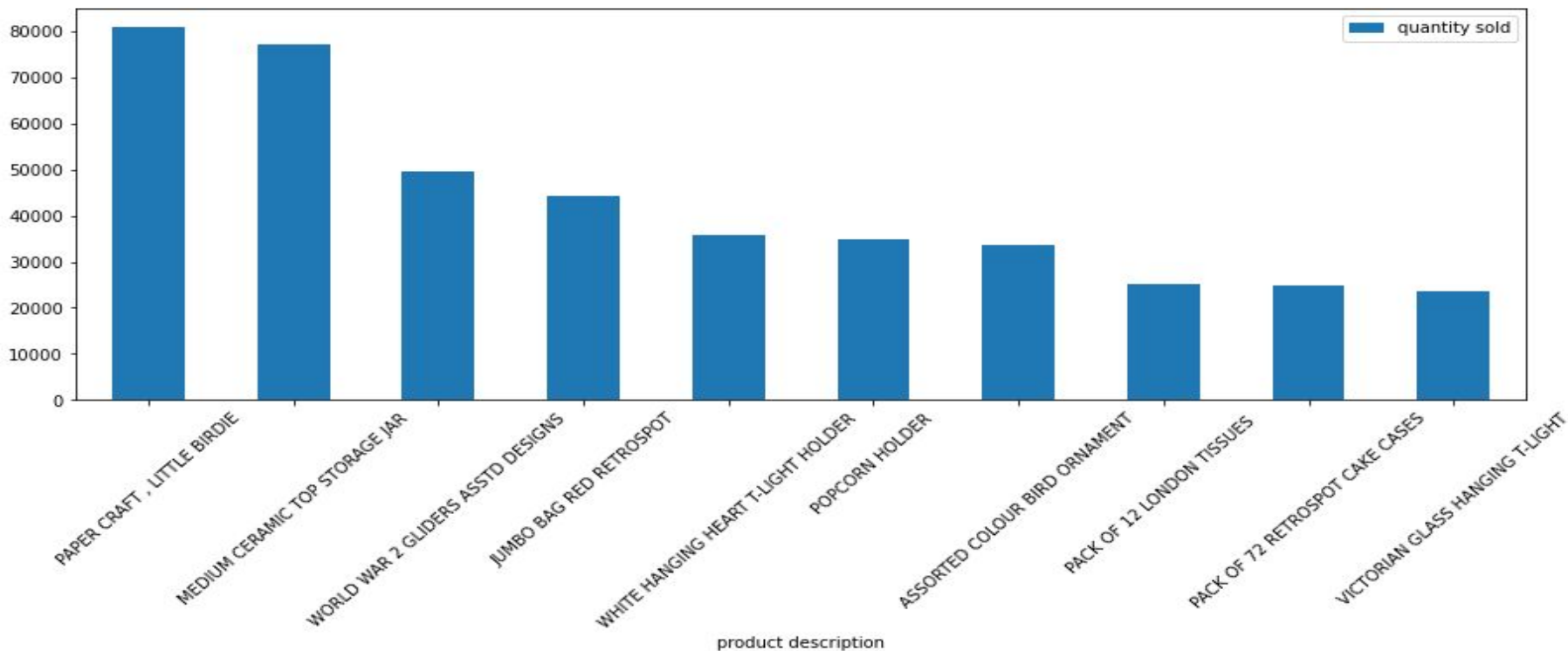
- For next graph we need to do some preprocessing on dataframe sales.
- First we have to separate the United kingdom sales data.
- On that data we have to use groupby by product description and the sum of quantity sold.
- Then we use sorting and then only use top 10 sold product for more clarity.

[84]:

```
sales_uk = sales[sales["transaction country"]=="United Kingdom"]
product_uk = sales_uk.groupby("product description").agg({"quantity sold":"sum"})
sort_uk_product = product_uk.sort_values("quantity sold", ascending=False).head(10)
sort_uk_product.plot.bar(figsize=(15, 5))
plt.xticks(rotation=45)
```

Graph:

product description vs quantity sold



Answer :

- So if we focus on the Uk customers, from following graph it clear that they are buying mostly the decorative items and interior design products.
- The people who can spend more amount of money on decorative items, can be financially stable customers.
- Or the customers who might be related to decoration businesses like marriages or party arrangement are buying from us.

b. Identify relationships and drivers of sales that might be hidden in the dataset.

- There are many customer who bought product from more than one customer.
- The number is around 153.

```
▶ sales_country_count[sales_country_count["transaction country count"]>1]["customer id"].count()
```

```
100 153
```


Pre-processing :

- For previous answer we had to do some pre-processing.
- Here we used groupby by customer id and aggregated transaction country by number of unique value and sum of total.
- We can also sorting for more clarity.

```
sales_country_count = sales.groupby("customer id").agg({"transaction country": "nunique", "total": "sum"}).reset_index()  
sales_country_count.rename(columns={"transaction country": "transaction country count"}, inplace=True)  
sales_country_count.sort_values("transaction country count", ascending=False).head()
```

4]:

	customer id	transaction country count	total
291	12705	3	9296.36
269	12678	3	20302.39
67	12429	3	3750.40
100	12471	3	25803.70
92	12457	3	3676.66

Top customers of the company :

- For top 10 customers of the company, we have to do some pre-processing.
- Most of them are from United Kingdom.
- Some customers ordered from more than one country.

Top 10 customers id with countries :

```
customer_freq = sales.groupby("customer id").agg({"date": "nunique", "transaction country": "unique"})
customer_freq.sort_values("date", ascending=False, inplace=True)
customer_freq["transaction country"] = customer_freq["transaction country"].apply(lambda x : x[0])
top_10_customer = customer_freq.head(10)
top_10_customer.rename(columns={"date": "number_of_day"}, inplace=True)
top_10_customer
```

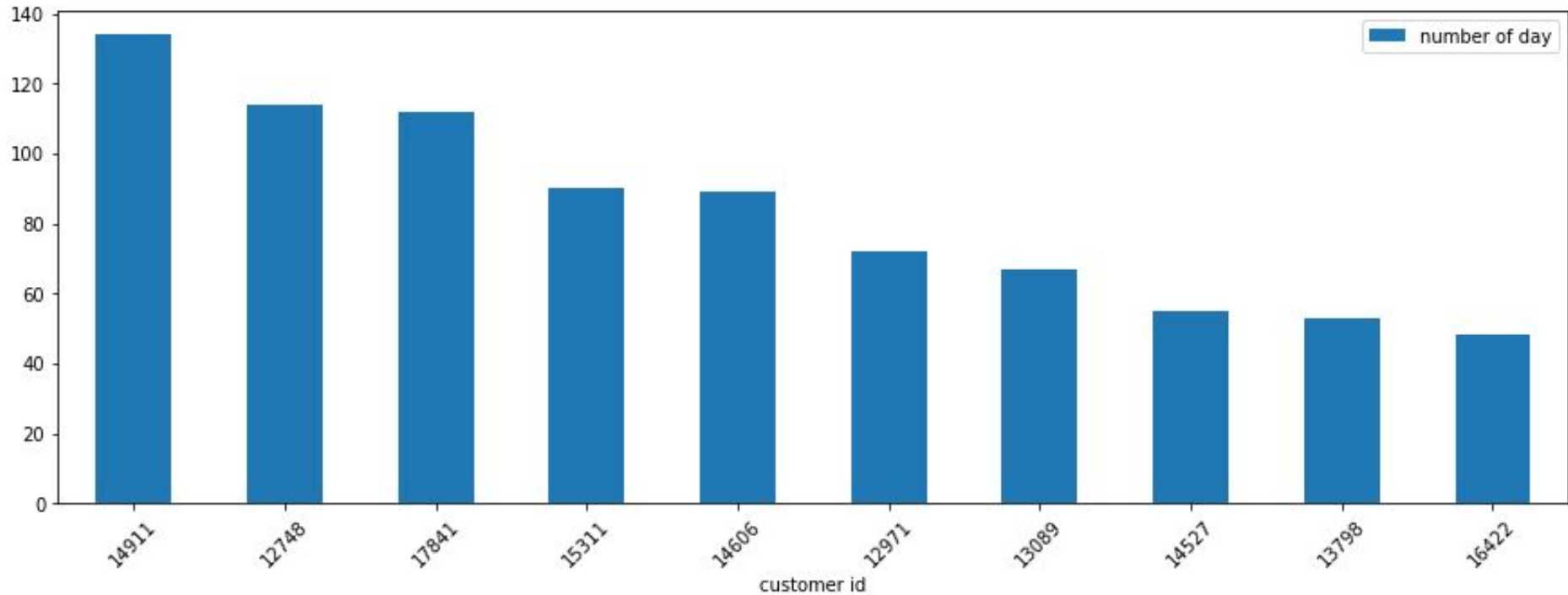
4]:

	number_of_day	transaction country
customer id		
14911	134	EIRE
12748	114	United Kingdom
17841	112	United Kingdom
15311	90	United Kingdom
14606	89	United Kingdom
12971	72	United Kingdom
13089	67	United Kingdom
14527	55	United Kingdom
13798	53	United Kingdom
16422	48	United Kingdom

Graph : [Top 10 customers of the company]

```
top_10_customer.drop("transaction country", axis=1).plot.bar(figsize=(15, 5))
```

customer id vs number of day





Question 4:


-

4. Can we take some initiatives based on the data to increase the sales? Also mention, based on data can we avoid out of stock situations?

- To increase the sales we can focus more on selling products like decorative items or interior design products.
- Different countries has different top selling products.
- According to that we can increase the different types of products.
- Like in United Kingdom, customers prefer decorative items to buy from our company.

Pre-processing :

- In the following code we are taking data particular product of quantity sold for example.
- For we have use groupby by date and aggregate quantity sold by using sum.
- Then we can sort data in descending order for more clarity.

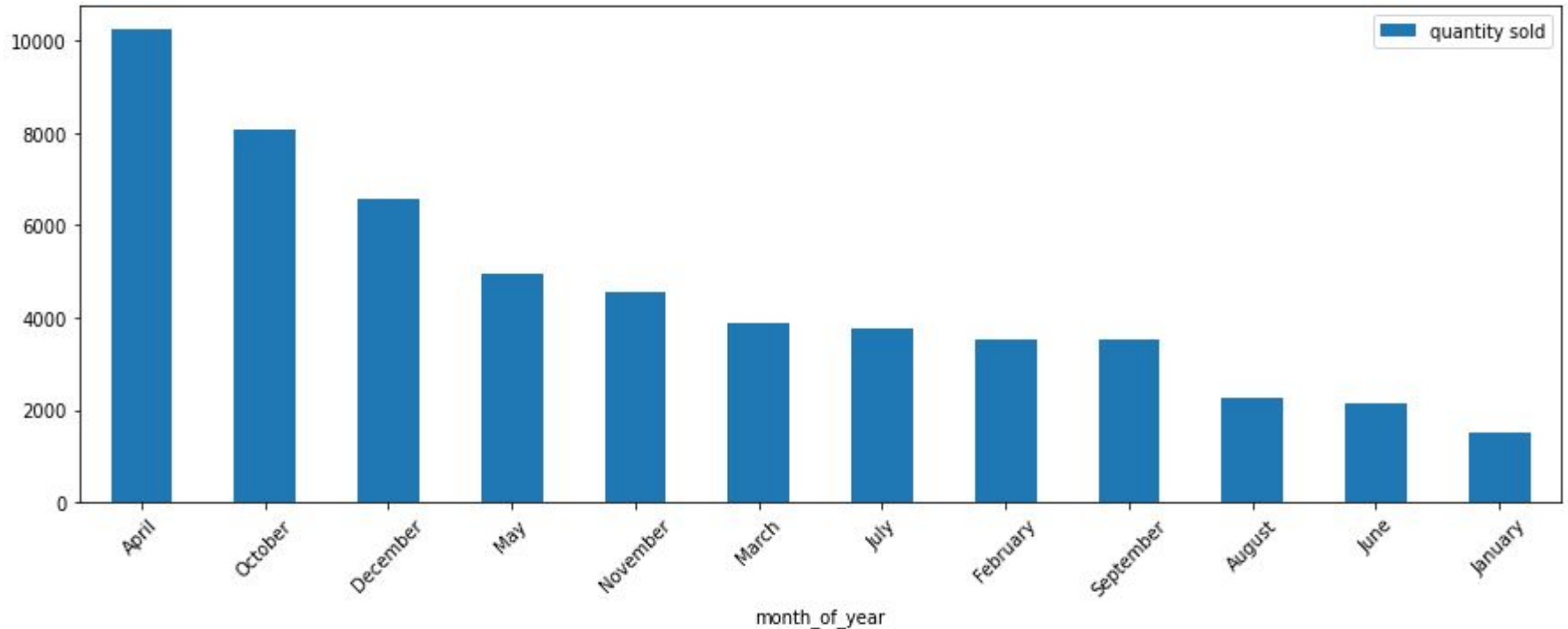


```
product_3 = sales[sales["product description"]=="WORLD WAR 2 GLIDERS ASSTD DESIGNS"]
product_3_count_date = product_3.groupby("date").agg({"quantity sold":"sum"})
product_3_count_month = product_3.groupby("month_of_year").agg({"quantity sold":"sum"})
product_3_count_month.sort_values("quantity sold", ascending=False, inplace=True)
product_3_count_month.plot.bar(figsize=(15, 5))
plt.xticks(rotation=45)
```

Graph :

- Following graph shows monthly selling of particular product.

month of year vs quantity sold



Answer :

- Above graph shows that product selling in each month is not constant.
- It varies according to month.
- So if we want to avoid out of stock situation we can increase the product quantity in respective month.
- For example, in April product had more number of buyers, and in January buyers are very less. So according we can manage our inventories.



Conclusion :

-

Conclusion :

- It is clear that the initially companies performance go through some ups - down.
- But in the next few month it improved lot.
- To maintain the growth company also has to focus on other countries. Because the 90 % percent companies business is in United Kingdom.
- And also focus on different types of products like Electronics, Furniture, etc.