DATA ANALYTICS PORTFOLIO

PREPARED BY: SAYAK DAS

python R R Programming Microsoft SQL Server Power BI X tableau

PROFESSIONAL BACKGROUND

I am an MTECH graduate and a certified data analyst from **IIT Kanpur** currently seeking for an entry-level role as a data analyst. I have 2 years of work experience as a sales and marketing executive from 2015 to 2017. I have done **virtual internship** in data analytics and **live projects** for **Trainity.**

I completed my MTECH IN 2020 and did my BTECH in mechanical engineering in 2015. My skills are **Predictive Modelling in both Python and R, Machine learning,** data visualization tools like **Power Bi and Tableau**,RDBMS tools like **SQL server** , **MYSQL and Excel.**

In my previous job, I have handled several reputed clients and converted many hot calls into leads, and did PESTEL analysis to capture new markets. My domain was water and wastewater treatment.

With my business acumen and **data-driven decision-making** skills, I am looking forward to work in an organization to leverage data-driven insights to **optimize business processes** and **deliver impactful solutions.**

I Have done several data science and analytics works which I have uploaded to my **Github.** All my Trainity works are uploaded in GITHUB, https://github.com/sayakakash

PROJECT 1 :AUTOMOBILE PROJECT

PROBLEM STATEMENT

Analyzing the Impact of Car Features on Price

Data dictionary

- **Make:** the make or brand of the car
- **Model:** the specific model of the car
- **Year:** the year the car was released
- **Engine Fuel Type**: the type of fuel used by the car (gasoline, diesel, etc.)
- **Engine HP:** the horsepower of the car's engine
- **Engine Cylinders:** the number of cylinders in the car's engine
- **Transmission Type**: the type of transmission (automatic or manual)
- **Driven_Wheels:** the type of wheels driven by the car (front, rear, all)
- **Number of Doors:** the number of doors the car has
- **Market Category:** the market category the car belongs to (Luxury, Performance, etc.)
- **Vehicle Size:** the size of the car
- **Vehicle Style:** the style of the car (Sedan, Coupe, etc.)
- **Highway MPG:** the estimated miles per gallon the car gets on the highway
- **City MPG:** the estimated miles per gallon the car gets in the city
- **Popularity:** a ranking of the popularity of the car (based on the number of times it has been viewed on Edmunds.com)
- **MSRP:** the manufacturer's suggested retail price of the car

OBJECTIVE

- Find the car category which captured the market most
- Plot a scatter graph to show the horsepower vs price relationship and show outliers in it by fitting a trend line
- Compare models to predict MSRP and reduce RMSE value.
- Find maker who has the highest average price
- Show the relationship between highway mileage and cylinder relationship
- Find the lowest price and highest price cars.
- Show top 5 features which are most correlated to price
- Build a Dashboard in Excel to present to the stakeholder

RESULTS

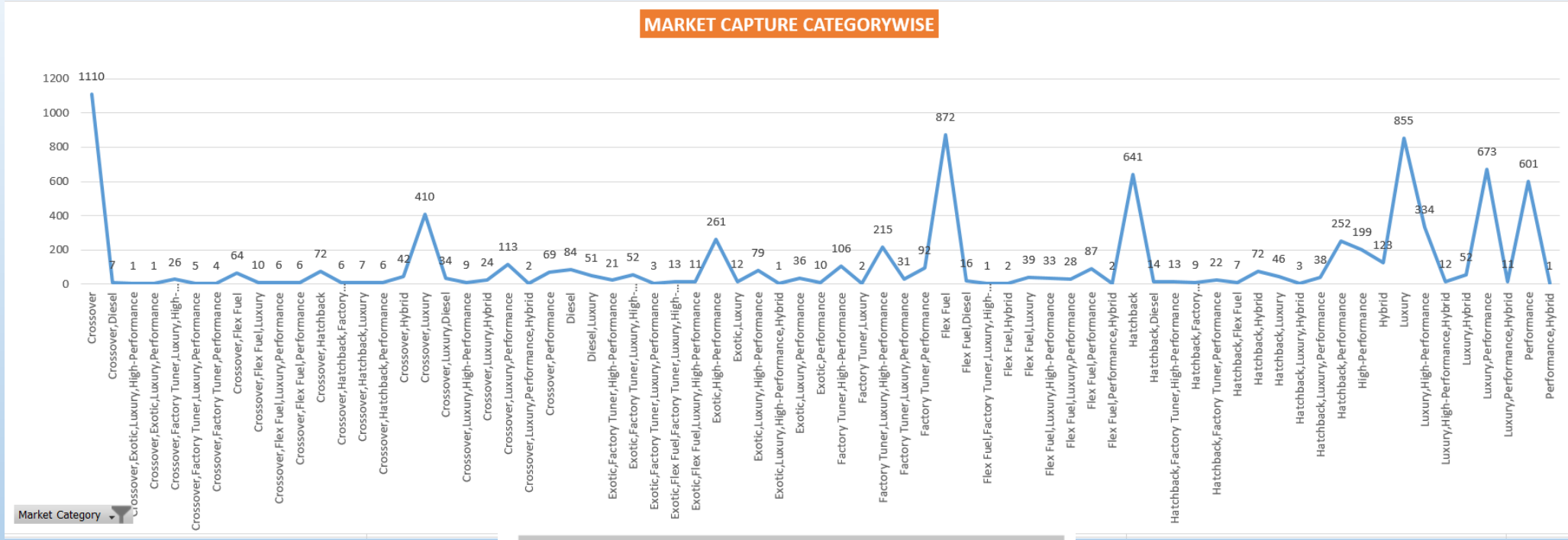
Data cleaning in python is done

```
df_cleaned.isnull().sum()
```

```
Make                 0
Model                0
Year                 0
Engine Fuel Type     0
Engine HP            0
Engine Cylinders     0
Transmission Type    0
Driven_Wheels        0
Number of Doors      0
Market Category      0
Vehicle Size         0
Vehicle Style        0
highway MPG          0
city mpg             0
Popularity           0
MSRP                 0
dtype: int64
```

Car category which captured the market most

EXCEL ANALYSIS



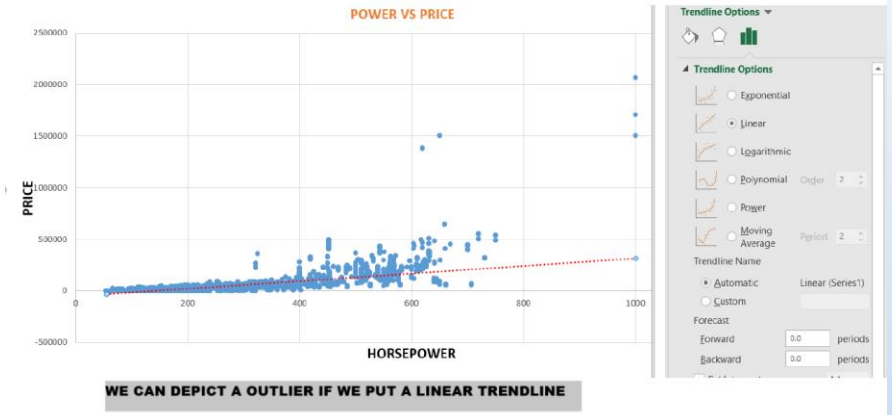
Crossover captured the market most

```
highest_capture = df['Market Category'].value_counts().idxmax()
print("category with the highest market capture:", highest_capture)
```

```
category with the highest market capture: Crossover
```

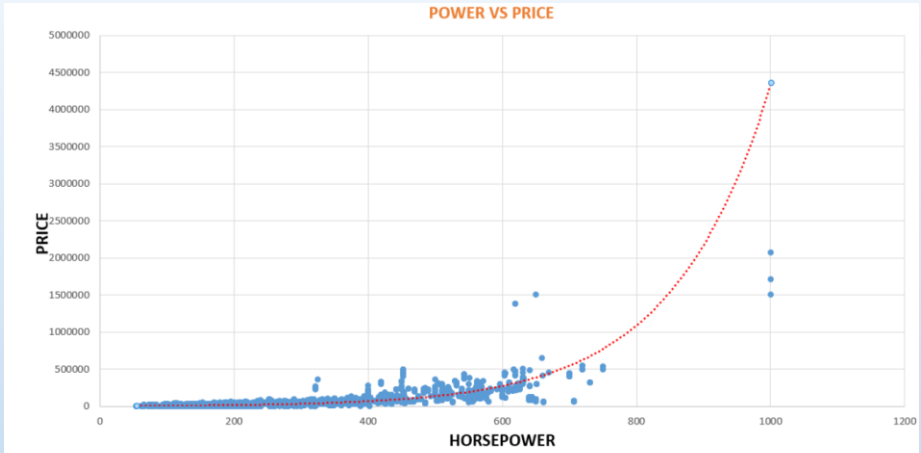
Python code

a scatter graph to show the horsepower vs price relationship and show outliers in it by fitting a trend line.



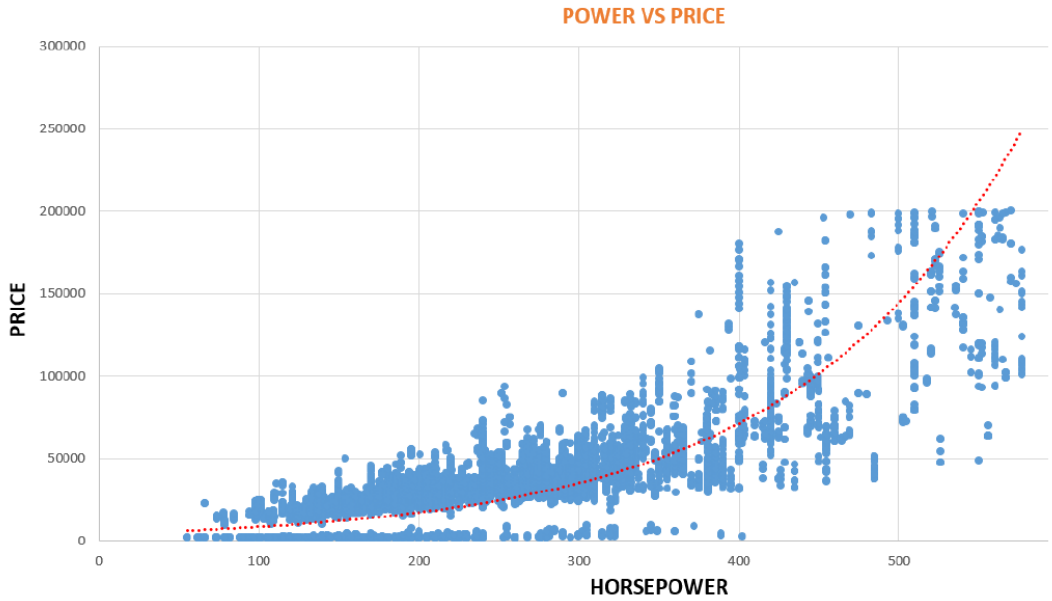
WE CAN DEPICT A OUTLIER IF WE PUT A LINEAR TRENDLINE

I HAVE FILTERED OUT THE NA VALUES



AN EXPONENTIAL OUTLIER GIVES A BETTER FIT IN THIS CASE AND SHOWS THE OUTLIERS

SINCE WE HAVE CHECKED THE TREND AND SEEN THE OUTLIERS WE CAN FILTER OUT THE RANGE AND DO GENERAL ANALYSIS



GENERAL TREND SAYS AS HORSEPOWER INCREASES PRICE ALSO INCREASES

Excel analysis

Compare models to predict MSRP and reduce RMSE value

```
df_cleaned=pd.get_dummies(df_cleaned)
```

```
# making train_dataset
train=df_cleaned[0:8000]
```

```
# making test dataset
test=df_cleaned[8000:]
```

```
x_train=train.drop('MSRP',axis=1)
```

```
y_train=train['MSRP']
```

```
x_test=test.drop('MSRP',axis=1)
```

```
final_pred=test['MSRP']
```

```
x_train=pd.get_dummies(x_train)
```

```
x_train.shape
```

```
(8000, 1079)
```

```
x_test=pd.get_dummies(x_test)
```

```
x_test.shape
```

```
(3894, 1079)
```

```
from sklearn.linear_model import LinearRegression
lreg=LinearRegression()
lreg.fit(x_train,y_train)
```

▾ LinearRegression

LinearRegression()

```
from sklearn.metrics import mean_squared_error

# Predict on the test set
y_pred = lreg.predict(x_test)

# Calculate the mean squared error
mse = mean_squared_error(final_pred, y_pred)

# Calculate the RMSE
rmse = np.sqrt(mse)

print("RMSE SCORE:", rmse)
```

```
RMSE SCORE: 108177329700.84694
```

In this case, the large RMSE score may indicate that the linear regression model is not capturing the underlying patterns and relationships in the data effectively. It could be a result of various factors such as the model's inability to capture nonlinear relationships, presence of outliers, or inadequate feature selection.Therefore I will perform gradient boost algorithm

GRADIENT BOOST

```
# Initialize the Gradient Boosting regressor
gb_regressor = GradientBoostingRegressor()

# Train the model on the training set
gb_regressor.fit(x_train, y_train)

# Make predictions on the test set
y_pred = gb_regressor.predict(x_test)

# Calculate RMSE
rmse = mean_squared_error(final_pred, y_pred, squared=False)
print("RMSE Score:", rmse)
```

```
RMSE Score: 45536.418841537954
```

We can see a huge reduction over RMSE score.Hence Grdient Boost performs well in this dataset

Steps involved:

- Using the cleaned data set
- Splitting the data
- Creating dummies for categorical features
- Fitting the model on train data
- Making predictions on test data
- Evaluating the model performance on RMSE

EXCEL Analysis

Maker who has the highest average price

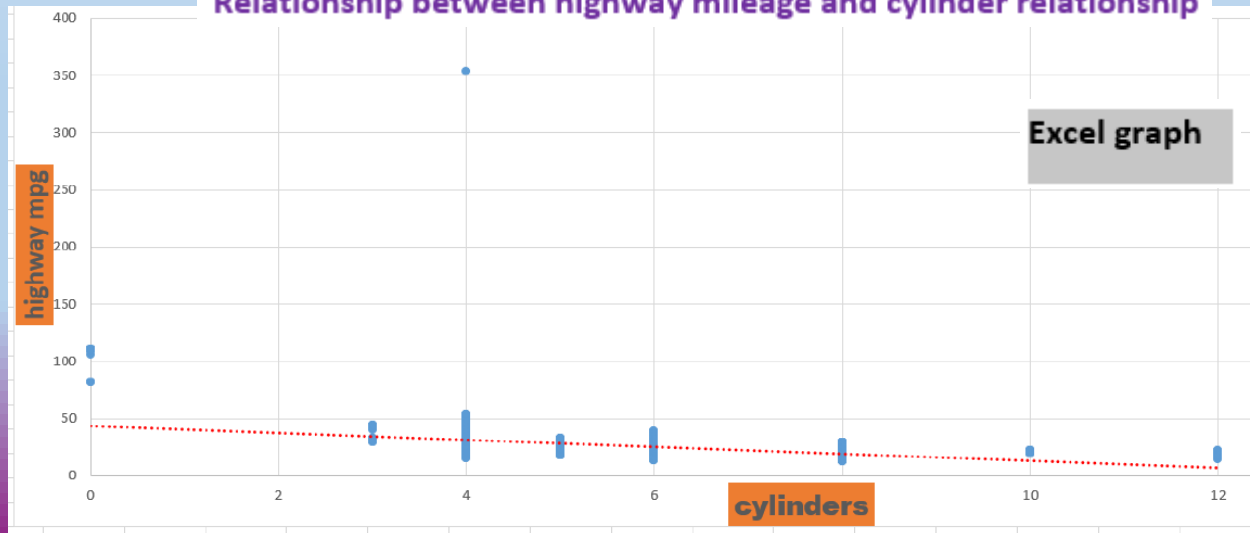
BUGATTI COUPE HAS THE HIGHEST AVERAGE PRICE

```python
import pandas as pd

# Group the data by maker and calculate the average price
maker_avg_price = df.groupby('Make')['MSRP'].mean()

# Find the maker with the highest average price
highest_avg_price_maker = maker_avg_price.idxmax()

# Print the result
print("Maker with the highest average price:", highest_avg_price_maker)
```

```
Maker with the highest average price: Bugatti
```

Python code

Relationship between highway mileage and cylinder relationship



Excel graph

```python
import matplotlib.pyplot as plt
import numpy as np

# Select the relevant columns from the DataFrame
highway_mpg = df_cleaned['highway MPG']
cylinders = df_cleaned['Engine Cylinders']

# Create the scatter plot
plt.scatter(cylinders, highway_mpg, alpha=0.5)

# Fit a linear regression line
fit = np.polyfit(cylinders, highway_mpg, 1)
fit_fn = np.poly1d(fit)
plt.plot(cylinders, fit_fn(cylinders), 'r-', linewidth=2)

# Set the labels and title
plt.xlabel('Number of Cylinders')
plt.ylabel('Highway MPG')
plt.title('Relationship between Highway MPG and Number of Cylinders')

# Display the plot
plt.show()
```

Python code

The general trend says as the cylinder increases mileage decreases

lowest price and highest price cars

BRANDS	Total MSRP for each brand
Acura	34887.5873
Alfa Romeo	61600
Aston Martin	197910.3763
Audi	53452.1128
Bentley	247169.3243
BMW	61546.76347
Bugatti	1757223.667
Buick	28206.61224
Cadillac	56231.31738
Chevrolet	28350.38557
Chrysler	26722.96257
Dodge	22390.05911
Ferrari	238218.8406
FIAT	22670.24194
Ford	27399.26674
Genesis	46616.66667
GMC	30493.29903
Honda	26674.34076
HUMMER	36464.41176
Hyundai	24597.0363
Infiniti	42394.21212
Kia	25310.17316
Lamborghini	331567.3077
Land Rover	67823.21678
Lexus	47549.06931
Lincoln	42839.82927
Lotus	69188.27586
Maserati	114207.7069
Maybach	546221.875
Mazda	20039.38298

=INDEX(O4:O11505, MATCH(MAX(P4:P11505), P4:P11505, 0))

BRANDS	Total MSRP for each brand
Acura	34887.5873
Alfa Romeo	61600
Aston Martin	197910.3763
Audi	53452.1128
Bentley	247169.3243
BMW	61546.76347
Bugatti	1757223.667

Bugatti HIGHEST MSRP

=INDEX(O4:O11505, MATCH(MIN(P4:P11505), P4:P11505, 0))

BRANDS	Total MSRP for each brand
Acura	34887.5873
Alfa Romeo	61600
Aston Martin	197910.3763
Audi	53452.1128
Bentley	247169.3243
BMW	61546.76347
Bugatti	1757223.667
Buick	28206.61224
Cadillac	56231.31738

Bugatti HIGHEST MSRP

Plymouth LOWEST MSRP

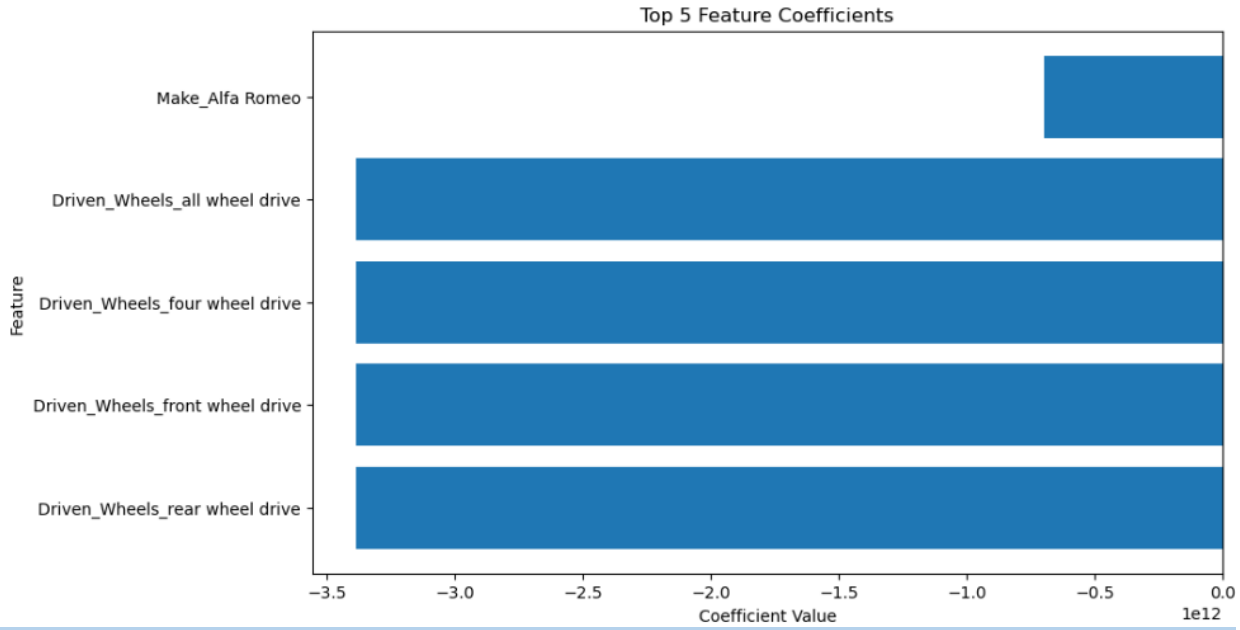
Excel analysis

BUGATTI Has Highest Price and PLYMOUTH Lowest

Python code

Top 5 features which are most correlated to price

```
import matplotlib.pyplot as plt

# Select the top 5 feature coefficients
top_5_features = feature_coefficients.head(5)

# Create a bar chart for the top 5 feature coefficients
plt.figure(figsize=(10, 6))
plt.barh(top_5_features['Feature'], top_5_features['Coefficient'])
plt.xlabel('Coefficient Value')
plt.ylabel('Feature')
plt.title('Top 5 Feature Coefficients')
plt.show()
```




```
# Create a dataframe with the selected numeric features and the target variable
numeric_features = df_cleaned[['Engine HP', 'Engine Cylinders','highway MPG', 'city mpg', 'Popularity', 'MSRP']]

# Calculate the correlation matrix
correlation_matrix = numeric_features.corr()

# Extract the correlation values of the numeric features with the target variable
correlation_with_msrp = correlation_matrix['MSRP'].drop('MSRP')

# Sort the correlation values in descending order
correlation_with_msrp_sorted = correlation_with_msrp.abs().sort_values(ascending=False)

# Print the sorted correlation values
print(correlation_with_msrp_sorted)
```

```
Engine HP           0.661752
Engine Cylinders    0.529566
city mpg            0.167916
highway MPG         0.167470
Popularity          0.048312
Name: MSRP, dtype: float64
```

Among numeric features engine horsepower has most impact on price

CARS DASHBOARD

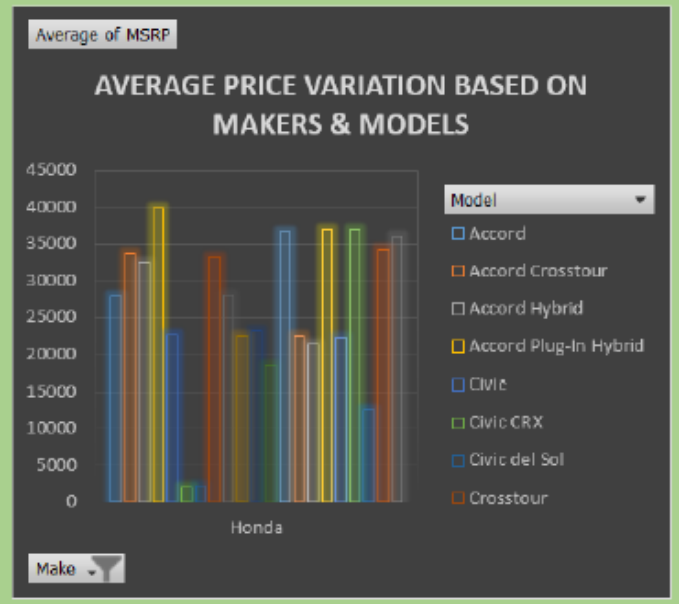
MSRP
2715
2879

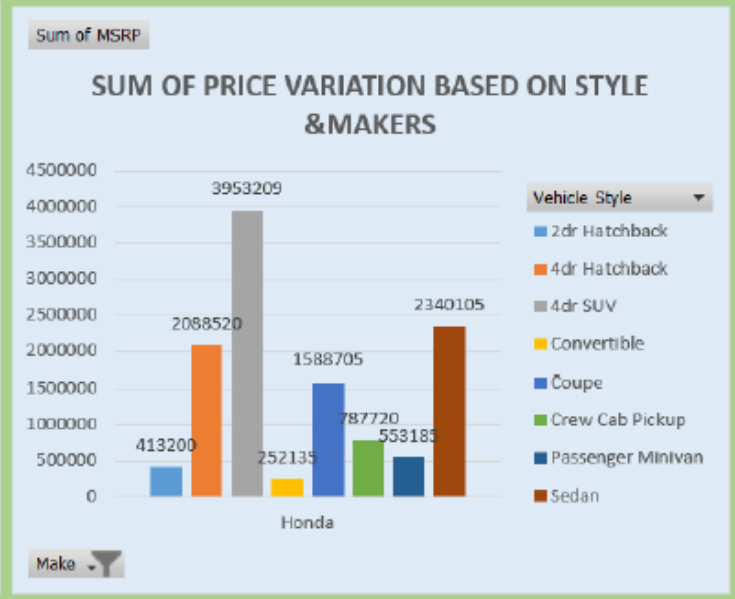
Make
Honda
HUMMER
Hyundai

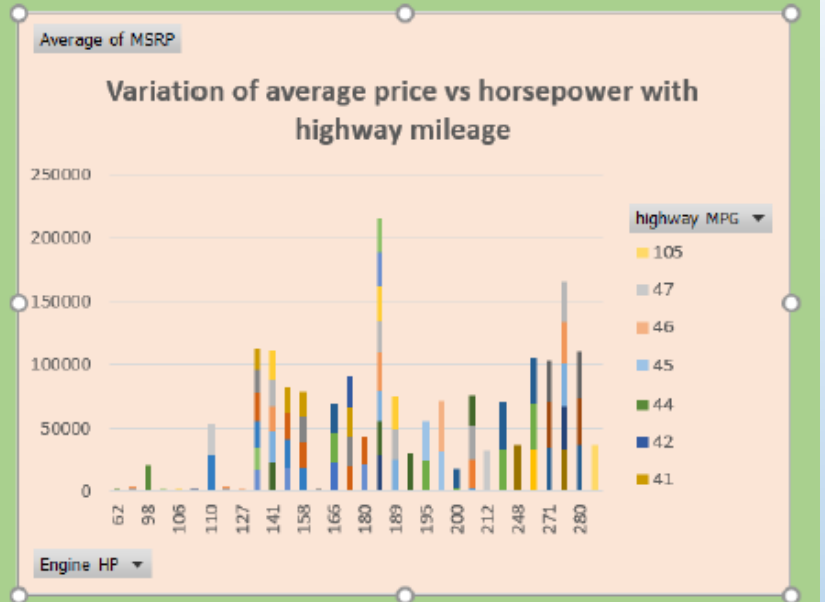
Model
Accord
Accord Cr...
Accord Hy...

Engine HP
62
92
98

Engine HP
62
92
98







DASHBOARD

PROJECT 2 :BANK LOAN PROJECT

PROBLEM STATEMENT

Perform exploratory data analysis(EDA) to understand how consumer attributes and loan attributes influence the tendency of default.

OBJECTIVE

- Finding outliers using the IQR method
- **Univariate analysis**
  - Analysis of the" **target"** feature
  - Plot histogram to show the distribution of income
- **Segmented univariate Analysis**
  - Histogram plot to show loan default is more in which income range
  - Histogram plot to show loan amount requested based on gender
- **Bivariate analysis**
  - Scatter plot and heat map to show the relationship between Credit Amount and Target
- Finding the top 10 **correlations** for the Client with payment difficulties
- Relation between the amount requested and the amount credited

Data dictionary

- SK_ID_CURR: Unique identifier for each client.
- TARGET: Binary variable indicating if the client had payment difficulties (1) or not (0).
- NAME_CONTRACT_TYPE: Type of the loan contract, such as cash loans or revolving loans.
- CODE_GENDER: Gender of the client (M: male, F: female).
- FLAG_OWN_CAR: Flag indicating if the client owns a car (Y: yes, N: no).
- FLAG_OWN_REALTY: Flag indicating if the client owns real estate (Y: yes, N: no).
- CNT_CHILDREN: Number of children the client has.
- AMT_INCOME_TOTAL: Total income of the client.
- AMT_CREDIT: Amount of credit requested by the client.
- AMT_ANNUITY: Annuity amount of the loan.
- AMT_GOODS_PRICE: Price of the goods for which the loan is taken.
- NAME_TYPE_SUITE: Who was accompanying the client during the application
- NAME_INCOME_TYPE: Client's income type, such as working, student, retired, etc.
- NAME_EDUCATION_TYPE: Level of education of the client.
- NAME_FAMILY_STATUS: Marital status of the client.
- NAME_HOUSING_TYPE: Type of housing where the client lives.
- REGION_POPULATION_RELATIVE: Relative population of the region where the client lives.
- DAYS_BIRTH: Client's age in days at the time of application.
- DAYS_EMPLOYED: Number of days employed by the client at the time of application.
- DAYS_REGISTRATION: Number of days the client has been registered in the application system.
- DAYS_ID_PUBLISH: Number of days since the client's application was published.
- OWN_CAR_AGE: Age of the client's car.
- FLAG_MOBIL: Flag indicating if the client has a mobile phone.
- FLAG_EMP_PHONE: Flag indicating if the client has an employer-provided phone.
- FLAG_WORK_PHONE: Flag indicating if the client has a work phone.
- FLAG_CONT_MOBILE: Flag indicating if the client's mobile phone can be contacted.
- FLAG_PHONE: Flag indicating if the client has a registered phone.
- FLAG_EMAIL: Flag indicating if the client has an email address.
- OCCUPATION_TYPE: Occupation of the client.
- CNT_FAM_MEMBERS: Number of family members of the client.
- REGION_RATING_CLIENT: Rating of the region where the client lives.
- REGION_RATING_CLIENT_W_CITY: Rating of the region where the client lives with taking into account the city.

And more.....columns

RESULTS

```
ser = pd.Series(df_1['AMT_CREDIT'])
print(ser.describe())
```

```
count    3.075110e+05
mean     5.990260e+05
std      4.024908e+05
min      4.500000e+04
25%      2.700000e+05
50%      5.135310e+05
75%      8.086500e+05
max      4.050000e+06
Name: AMT_CREDIT, dtype: float64
```

```
IQR=8.086500e+05-2.700000e+05
lower_limit=2.700000e+05-1.5*IQR
lower_limit
```

```
-537975.0
```

```
upper_limit=8.086500e+05+1.5*IQR
upper_limit
```

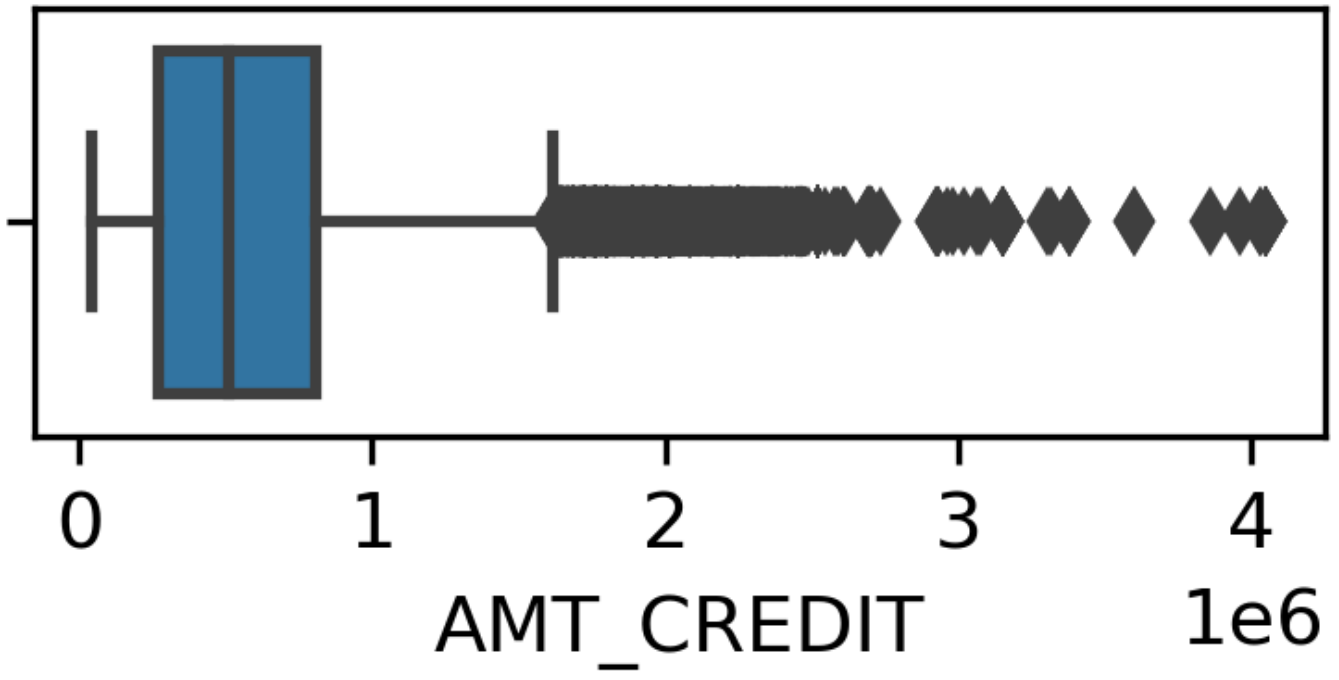
```
1616625.0
```

```
ser[ser >upper_limit]
```

```
60        1663987.5
135       1755000.0
189       2250000.0
235       1710000.0
314       1800000.0
             ...
307216    1827549.0
307252    1724220.0
307401    1718473.5
307422    1971072.0
307476    1762110.0
Name: AMT_CREDIT, Length: 6562, dtype: float64
```

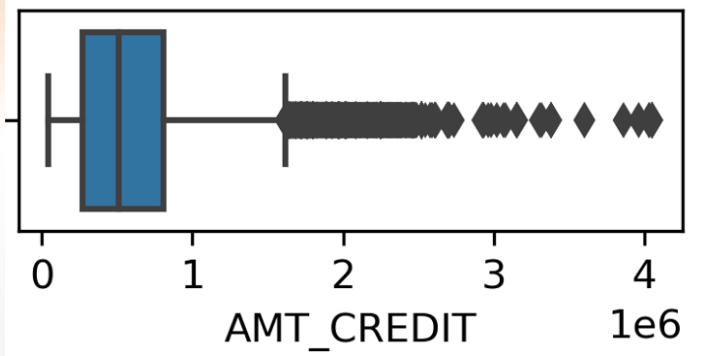
Finding outliers using IQR method

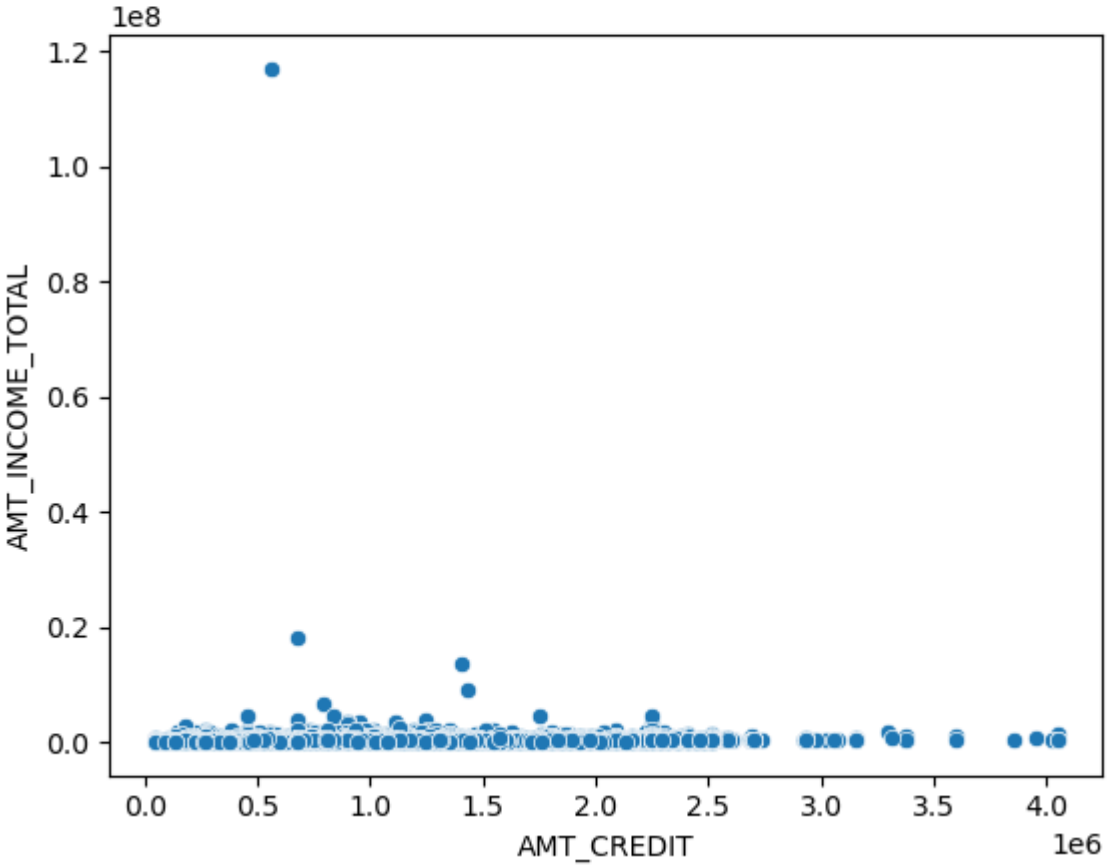
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(3,1),dpi=300)
sns.boxplot(x='AMT_CREDIT',data=df_1,orient='h')
plt.show()
```



Anything greater than approx. 1.7 lakh might be treated as outlier

```
import seaborn as sns
import matplotlib.pyplot as plt

# Select only the columns of interest
df_subset = df_1[['AMT_INCOME_TOTAL', 'AMT_CREDIT']]

# Create the scatter plot
sns.scatterplot(data=df_subset, x='AMT_CREDIT', y='AMT_INCOME_TOTAL')

# Show the plot
plt.show()
```



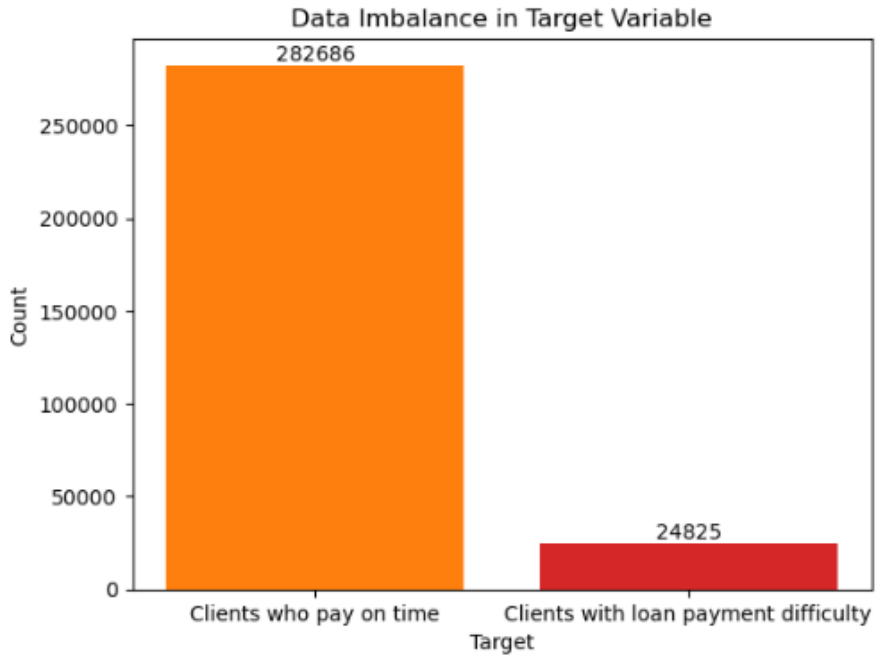


On analyzing both the box plot and scatter plot we can say with low income level amount credit > 2.5 * 10 E6 will increase the risk of default.

Univariate analysis - Analysis of the" target" feature

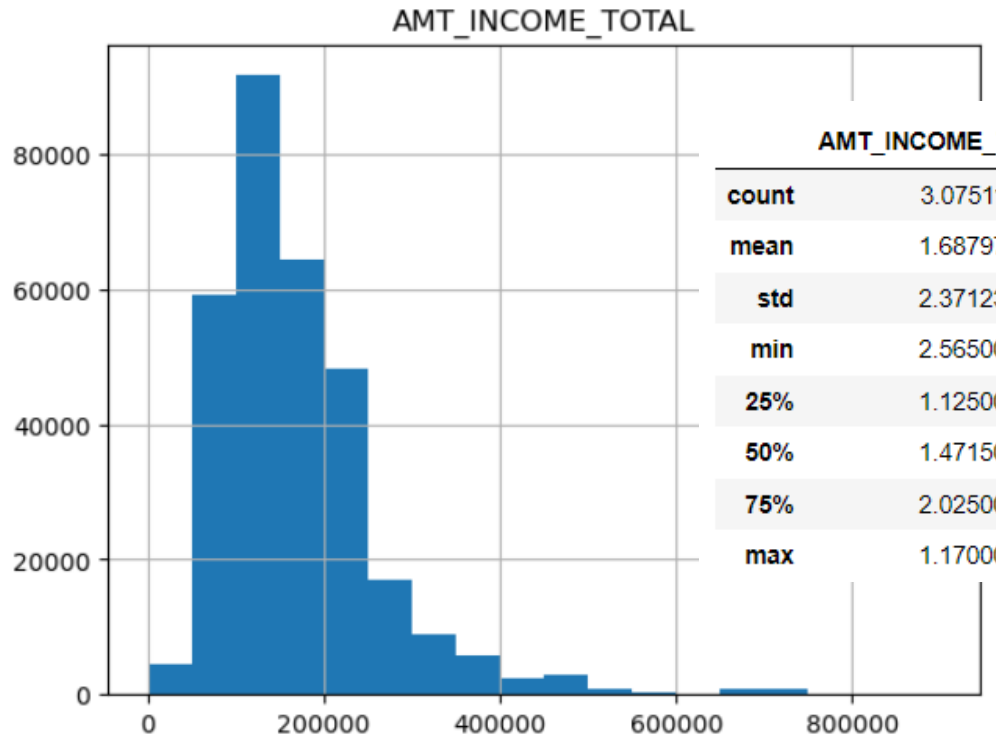
```
import matplotlib.pyplot as plt

# Count the occurrences of 0 and 1 in TARGET separately
count_0 = df_1[df_1['TARGET'] == 0]['TARGET'].count()
count_1 = df_1[df_1['TARGET'] == 1]['TARGET'].count()

# Define the colors for each category
colors = ['C1', 'C3']

# Create a bar plot
plt.bar(['Clients who pay on time', 'Clients with loan payment difficulty'], [count_0, count_1], color=colors)

# Set the labels and title
plt.xlabel('Target')
plt.ylabel('Count')
plt.title('Data Imbalance in Target Variable')

# Add the actual count on top of each bar
for i, count in enumerate([count_0, count_1]):
    plt.text(i, count, str(count), ha='center', va='bottom')

# Display the plot
plt.show()
```

The code is creating two new dataframes 'zero' and 'one' containing the rows from the original dataframe 'df_1' where the value in the 'TARGET' column is 0 and 1, respectively. Then, it is creating a bar plot using matplotlib to display the count of clients who pay on time and clients with loan payment difficulty separately, using the 'TARGET' column of each of these dataframes. The subplot function is used to display both the plots side by side. The first subplot displays the count of clients who pay on time and the second subplot displays the count of clients with loan payment difficulty.



Univariate analysis - Plot histogram to show the distribution of income

```
# SHOW THE DISTRIBUTION OF AMT_INCOME_TOTAL
amount_Income = df_1[['AMT_INCOME_TOTAL']]

# DEFINE THE BINS
bins = [0, 50000, 100000, 150000,200000, 250000, 300000, 350000, 400000, 450000, 500000, 550000,600000,650000,750000,800000,85000

# PLOT A HISTOGRAM TO SEE THE DISTRIBUTION OF INCOME
amount_Income.hist(bins= bins, range=[2.565000e+04,1.170000e+08])

plt.show()
amount_Income.describe()
```



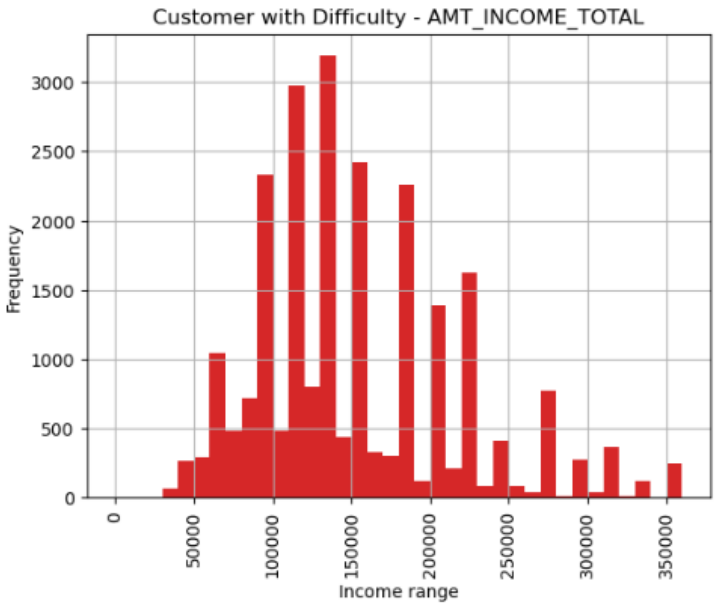
	AMT_INCOME_TOTAL
count	3.075110e+05
mean	1.687979e+05
std	2.371231e+05
min	2.565000e+04
25%	1.125000e+05
50%	1.471500e+05
75%	2.025000e+05
max	1.170000e+08

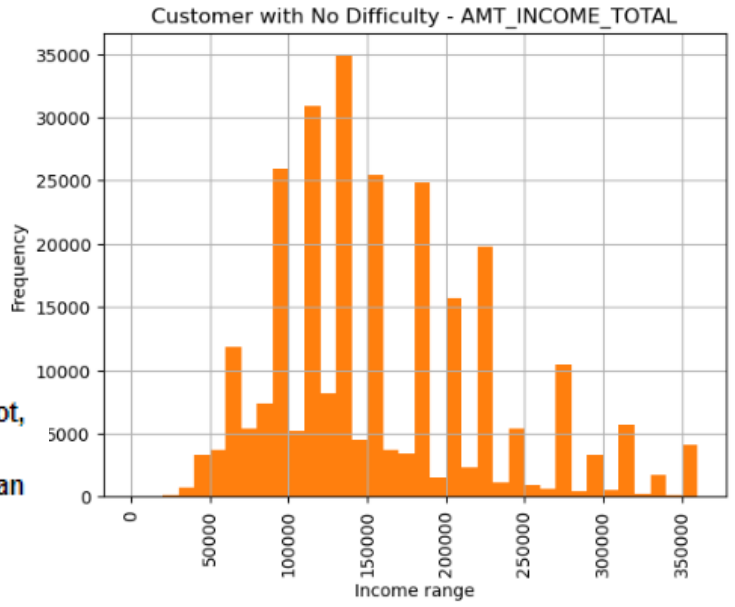
This is a left-skewed distribution or a left-tailed distribution.This means that there are more data points on the left side of the distribution.In a left-skewed distribution, the mean is typically less than the median.

- **Segmented univariate Analysis**
  - Histogram plot to show loan default is more in which income range
  - Histogram plot to show loan amount requested based on gender

Segmented univariate analysis

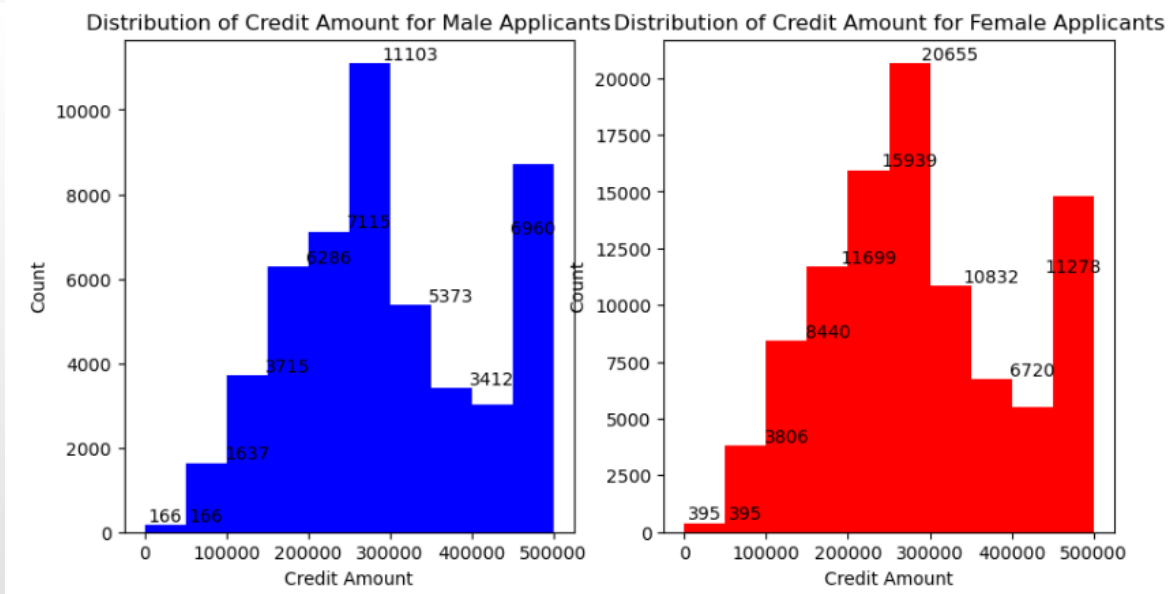
```
# 3.A. UNIVARIATE ANALYSIS - AMT_INCOME_TOTAL
AMT_INCOME_TOTAL_one=one[['AMT_INCOME_TOTAL']]
AMT_INCOME_TOTAL_zero = zero[['AMT_INCOME_TOTAL']]

min = AMT_INCOME_TOTAL_one.describe().min()
max = AMT_INCOME_TOTAL_one.describe().max()
range1=[min['AMT_INCOME_TOTAL'], max['AMT_INCOME_TOTAL']]
bins = [0, 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000, 130000, 140000, 150000, 160000,
AMT_INCOME_TOTAL_one.hist(bins=bins, range=range1, color = ['C3'])
plt.title("Customer with Difficulty - AMT_INCOME_TOTAL")
plt.xlabel("Income range")
plt.ylabel("Frequency")
plt.xticks(rotation = 90, fontsize=10)

min = AMT_INCOME_TOTAL_zero.describe().min()
max = AMT_INCOME_TOTAL_zero.describe().max()
range2=[min['AMT_INCOME_TOTAL'], max['AMT_INCOME_TOTAL']]
AMT_INCOME_TOTAL_zero.hist(bins=bins, range=range2, color = ['C1'])
plt.title("Customer with No Difficulty - AMT_INCOME_TOTAL")
plt.xlabel("Income range")
plt.ylabel("Frequency")
plt.xticks(rotation = 90, fontsize=10)

plt.show()
```





This code is plotting separate histograms for the income distribution of loan applicants who have defaulted on their loans and those who have not, using the "hist" function in matplotlib. The histograms will show the count of loan applicants within a particular income range. The "bins" argument specifies the number of bins in the histogram and the "color" argument specifies the color of the histogram bars. The resulting plots can help identify any significant differences in the income distribution of loan applicants between those who have defaulted on their loans and those who have not.

```python
# MALE vs FEMALE applicant
male_applicants = df_1[df_1['CODE_GENDER'] == 'M']
female_applicants = df_1[df_1['CODE_GENDER'] == 'F']
bins = [0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, 500000]
range_male = [male_applicants['AMT_CREDIT'].min(), male_applicants['AMT_CREDIT'].max()]
range_female = [female_applicants['AMT_CREDIT'].min(), female_applicants['AMT_CREDIT'].max()]

plt.figure(figsize=(10,5))
plt.subplot(121)
plt.hist(male_applicants['AMT_CREDIT'], bins=bins, range=range_male, color='blue')
plt.xlabel('Credit Amount')
plt.ylabel('Count')
plt.title('Distribution of Credit Amount for Male Applicants')

for i in range(len(bins)-1):
    plt.text((bins[i]+bins[i+1])/2, male_applicants['AMT_CREDIT'].value_counts(bins=bins)[bins[i]], male_applicants['AMT_CREDIT']

plt.subplot(122)
plt.hist(female_applicants['AMT_CREDIT'], bins=bins, range=range_female, color='red')
plt.xlabel('Credit Amount')
plt.ylabel('Count')
plt.title('Distribution of Credit Amount for Female Applicants')

for i in range(len(bins)-1):
    plt.text((bins[i]+bins[i+1])/2, female_applicants['AMT_CREDIT'].value_counts(bins=bins)[bins[i]], female_applicants['AMT_CRED

plt.show()
```

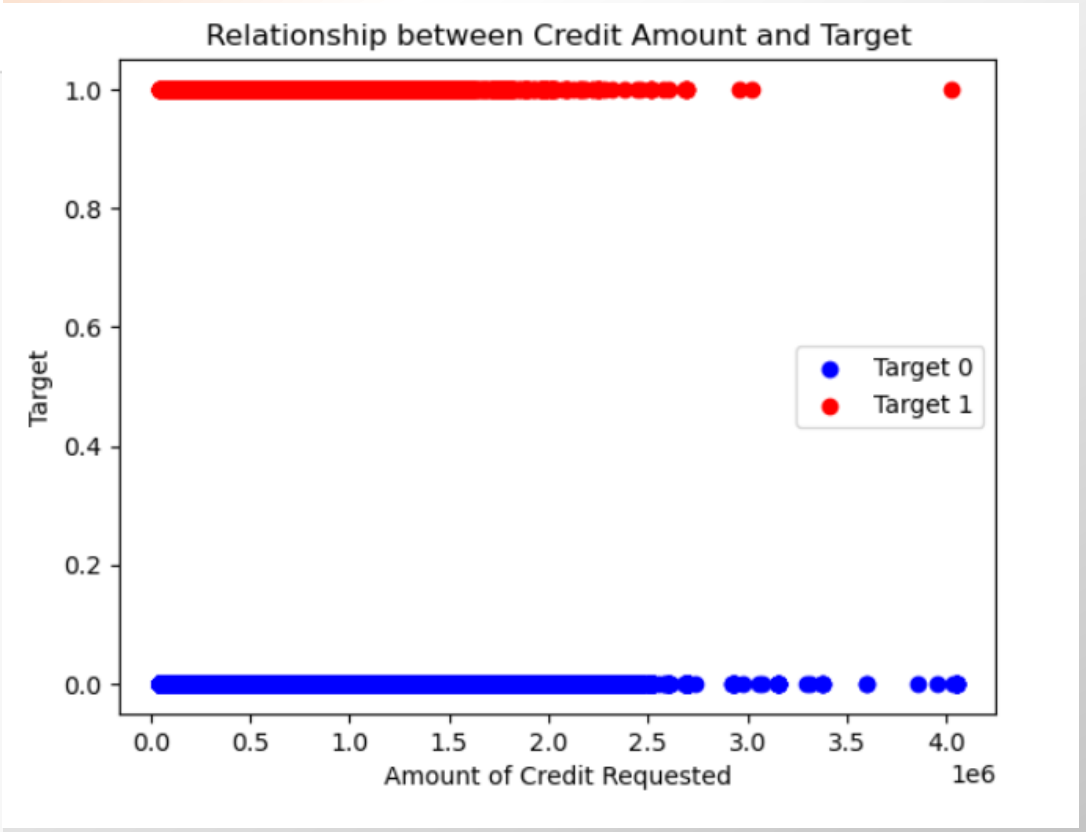


The above code will give us two histograms side-by-side, one showing the distribution of credit amounts for male applicants and the other showing the distribution for female applicants. We can use this to visually compare the credit amounts requested by men and women and see if there are any significant differences.

Bivariate analysis-Scatter plot to show the relationship between Credit Amount and Target

bivariate analysis

```
# credit vs target
import matplotlib.pyplot as plt

# Get data for target = 0
target_0 = df_1[df_1['TARGET'] == 0]
x_0 = target_0['AMT_CREDIT']
y_0 = target_0['TARGET']

# Get data for target = 1
target_1 = df_1[df_1['TARGET'] == 1]
x_1 = target_1['AMT_CREDIT']
y_1 = target_1['TARGET']

# Create scatter plot
plt.scatter(x_0, y_0, color='blue', label='Target 0')
plt.scatter(x_1, y_1, color='red', label='Target 1')
plt.xlabel('Amount of Credit Requested')
plt.ylabel('Target')
plt.title('Relationship between Credit Amount and Target')
plt.legend()
plt.show()
```



By visualizing the graph we can gain insights into how credit amount affects the likelihood of a loan being repaid.The target variable is represented by different colors, where blue dots represent loans that were repaid on time (target=0) and orange dots represent loans that were not repaid on time (target=1).There are more blue dots (repaid loans) than orange dots (defaulted loans) across most credit amounts.This suggests that the majority of loans are repaid on time regardless of the credit amount.However, we can also see that as the credit amount increases,the proportion of orange dots (defaulted loans)DECREASES as well.This indicates that as the credit amount gets larger,there is a LOWER risk of default.In conclusion, this graph highlights the relationship between credit amount and loan repayment status.

Finding the top 10 correlations for the Client with payment difficulties

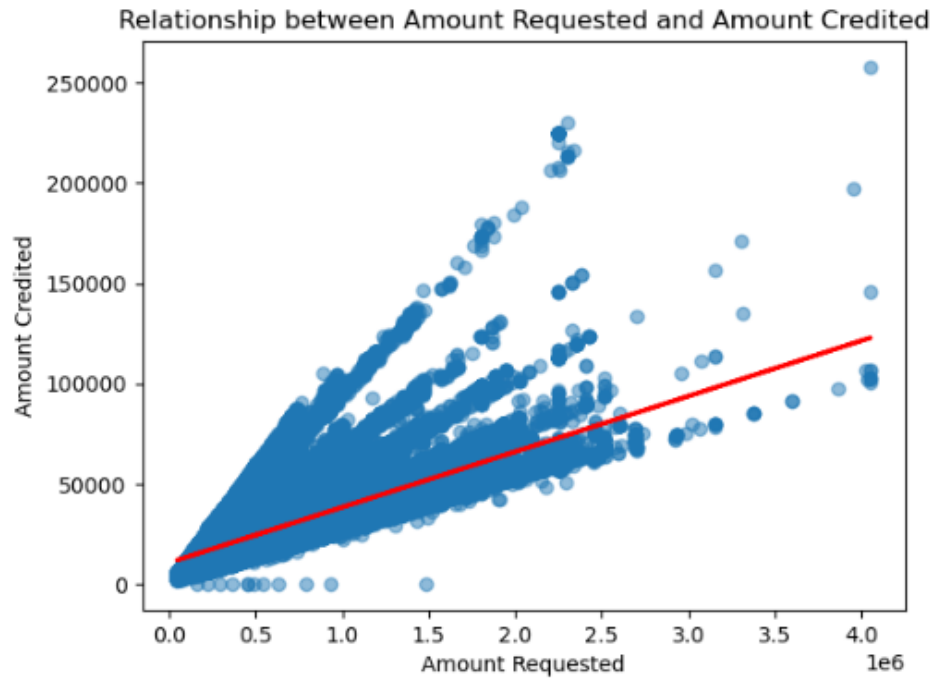
```python
#top 10 correlation for the Client with payment difficulties and all other cases (Target variable)
# Segregate data frames based on 'TARGET' variable
df_payment_difficulty = df_1[df_1['TARGET'] == 1]
df_no_payment_difficulty = df_1[df_1['TARGET'] == 0]

# Calculate correlation matrix for clients with payment difficulties
corr_payment_difficulty = df_payment_difficulty.corr()

# Identify top 10 correlations for clients with payment difficulties
top_corr_payment_difficulty = corr_payment_difficulty.unstack().sort_values(ascending=False).drop_duplicates()[1:11]

# Calculate correlation matrix for clients without payment difficulties
corr_no_payment_difficulty = df_no_payment_difficulty.corr()

# Identify top 10 correlations for clients without payment difficulties
top_corr_no_payment_difficulty = corr_no_payment_difficulty.unstack().sort_values(ascending=False).drop_duplicates()[1:11]

# Analyze insights from the top correlations
print('Top correlations for clients with payment difficulties:')
print(top_corr_payment_difficulty)
```

```
Top correlations for clients with payment difficulties:
YEARS_BEGINEXPLUATATION_MEDI  YEARS_BEGINEXPLUATATION_AVG     0.999964
YEARS_BUILD_MEDI              YEARS_BUILD_AVG                 0.999939
YEARS_BEGINEXPLUATATION_MODE  YEARS_BEGINEXPLUATATION_AVG     0.999792
                              YEARS_BEGINEXPLUATATION_MEDI    0.999774
YEARS_BUILD_MODE              YEARS_BUILD_MEDI                0.999676
                              YEARS_BUILD_AVG                 0.999632
FLOORSMIN_MEDI                FLOORSMIN_AVG                   0.999119
BASEMENTAREA_AVG              BASEMENTAREA_MEDI               0.999011
FLOORSMAX_MEDI                FLOORSMAX_AVG                   0.998769
LIVINGAPARTMENTS_AVG          LIVINGAPARTMENTS_MEDI           0.998711
dtype: float64
```

The variables with the highest correlations are related to the age and construction of the property, followed by variables related to floors, living space, and basement area. It is interesting to note that the variables related to social circles have a high correlation for clients without payment difficulties, but not for those with payment difficulties.

Relation between the amount requested and the amount credited

```python
import matplotlib.pyplot as plt
import numpy as np

#Select the columns of interest
amt_requested = df_1['AMT_CREDIT']
amt_credited = df_1['AMT_ANNUITY']

#Create a scatter plot
plt.scatter(amt_requested, amt_credited, alpha=0.5)

#Fit a polynomial regression line
fit = np.polyfit(amt_requested, amt_credited, 1)
fit_fn = np.poly1d(fit)
plt.plot(amt_requested, fit_fn(amt_requested), 'r-', linewidth=2)

#Set the labels and title
plt.xlabel('Amount Requested')
plt.ylabel('Amount Credited')
plt.title('Relationship between Amount Requested and Amount Credited')

#Display the plot
plt.show()
```

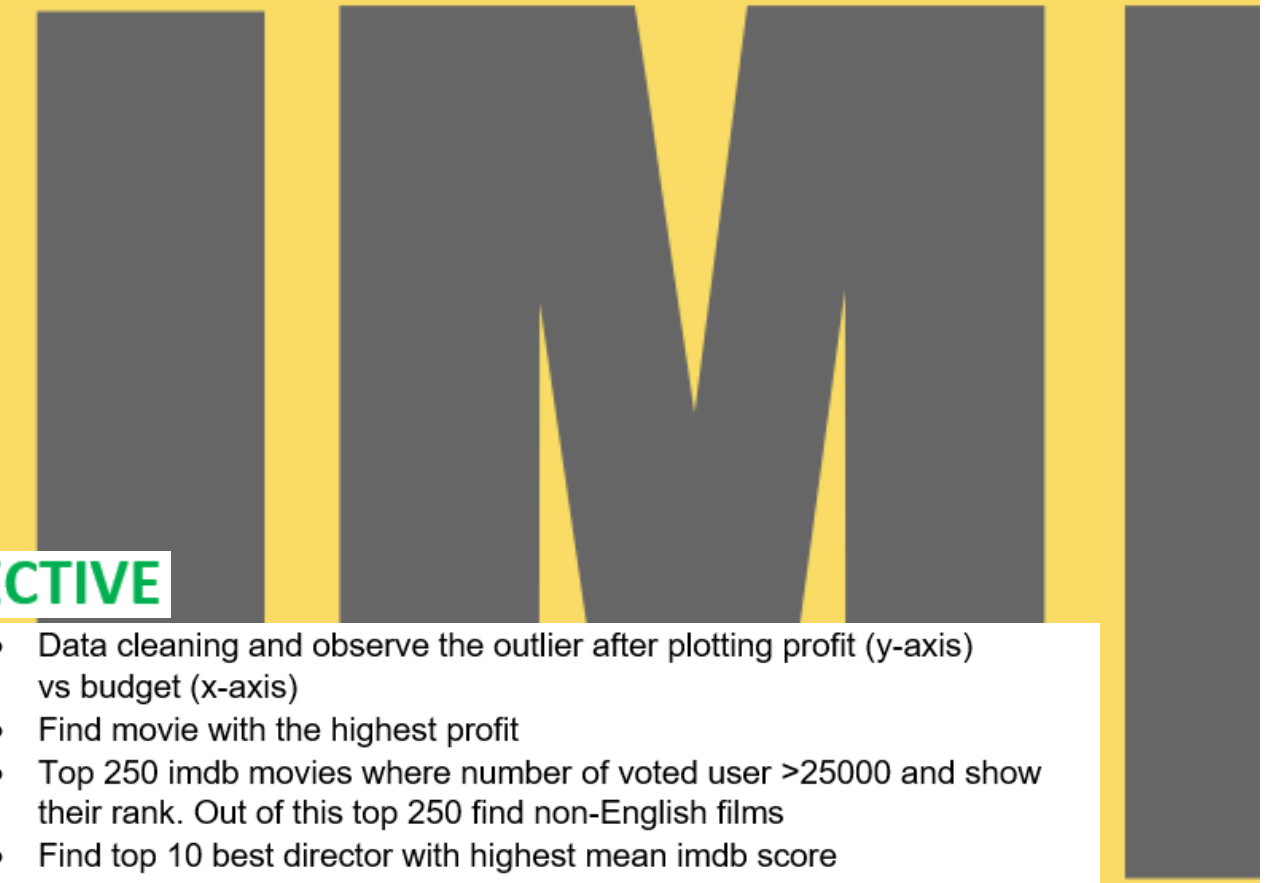


Therefore, amount requested and amount credited shows a linear relation and an upward trend.

PROJECT 3 :IMDB MOVIE ANALAYSIS

PROBLEM STATEMENT

Perform tasks to draw some data insights



OBJECTIVE

- Data cleaning and observe the outlier after plotting profit (y-axis) vs budget (x-axis)
- Find movie with the highest profit
- Top 250 imdb movies where number of voted user >25000 and show their rank. Out of this top 250 find non-English films
- Find top 10 best director with highest mean imdb score
- Find top 10 genres
- Show movies of Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' after extraction
- Count movies that belong to each decade
- Number of critic reviews for the movie and number of voted user
- Present a Power Bi Dashboard

Data dictionary

- color: Indicates whether the movie is in color or black and white.
- director_name: Name of the movie director.
- num_critic_for_reviews: Number of critic reviews for the movie.
- duration: Duration of the movie in minutes.
- director_facebook_likes: Number of Facebook likes for the director.
- actor_3_facebook_likes: Number of Facebook likes for the third main actor.
- actor_2_name: Name of the second main actor.
- actor_1_facebook_likes: Number of Facebook likes for the first main actor.
- gross: Gross earnings of the movie.
- genres: Genre(s) of the movie.
- num_voted_users: Number of users who voted for the movie.
- cast_total_facebook_likes: Total number of Facebook likes for all the cast members of the movie.
- facenumber_in_poster: Number of faces shown in the movie poster.
- plot_keywords: Keywords associated with the movie plot.
- movie_imdb_link: Link to the movie's IMDb page.
- num_user_for_reviews: Number of user reviews for the movie.
- language: Language of the movie.
- country: Country of origin of the movie.
- content_rating: Content rating of the movie.
- budget: Budget of the movie.
- title_year: Year of release of the movie.
- actor_2_facebook_likes: Number of Facebook likes for the second main actor.
- imdb_score: IMDb rating score for the movie.
- aspect_ratio: Aspect ratio of the movie.
- movie_facebook_likes: Number of Facebook likes for the movie's Facebook page.

RESULTS

Data cleaning and finding outliers

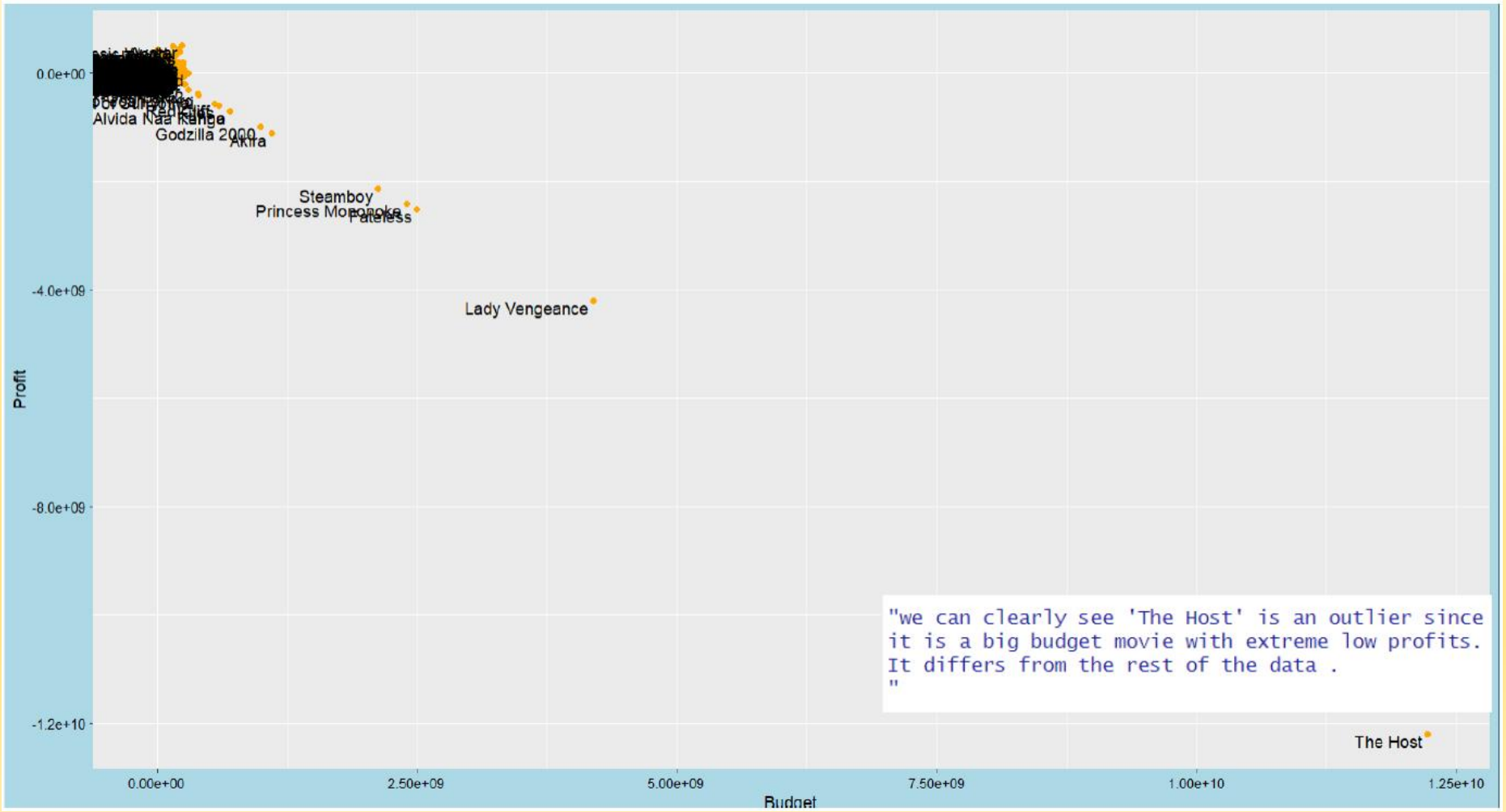
```
#### DATA CLEANING ####

#converting all numeric columns NaN value to 0
numeric_col_fill <- c('num_critic_for_reviews', 'duration', 'actor_3_facebook_likes', 'gross',
                      'budget', 'aspect_ratio', 'facenumber_in_poster', 'actor_1_facebook_likes',
                      'actor_2_facebook_likes',
                      'num_user_for_reviews', 'title_year','director_facebook_likes')
mydata[, numeric_col_fill] <- lapply(mydata[, numeric_col_fill], function(x) ifelse(is.na(x), 0, x))

# Converting all string columns NaN or null value to "NONE" string
string_col_fill <- c("actor_3_name", "plot_keywords", "content_rating", "director_name", "color",
                     "actor_2_name", "genres", "movie_imdb_link", "actor_1_name", "language")
mydata[, string_col_fill] <- lapply(mydata[string_col_fill], function(x) ifelse(is.na(x) | x == "", "NONE", x))
colSums(is.na(mydata))
```

```
                    color                  director_name      num_critic_for_reviews                  duration
                        0                              0                          0                         0
director_facebook_likes      actor_3_facebook_likes                0            actor_2_name     actor_1_facebook_likes
                        0                              0                          0                         0
                    gross                         genres            actor_1_name                movie_title
                        0                              0                          0                         0
        num_voted_users cast_total_facebook_likes                0            actor_3_name      facenumber_in_poster
                        0                              0                          0                         0
          plot_keywords                movie_imdb_link      num_user_for_reviews                  language
                        0                              0                          0                         0
                country                 content_rating                    budget                title_year
                        0                              0                          0                         0
 actor_2_facebook_likes                     imdb_score                0            aspect_ratio      movie_facebook_likes
                        0                              0                          0                         0
```

Data is cleaned now

```r
install.packages("ggplot2")
library(ggplot2)
ggplot(clean_profit, aes(x = budget, y = profit)) +
  geom_point(color = "orange") +

  geom_text(aes(label = movie_title), vjust = 1, hjust = 1) +

  xlab("Budget") +
  ylab("Profit") +

  theme(plot.background = element_rect(fill = "lightblue"),
        plot.title = element_text(color = "black"),
        axis.title = element_text(color = "black"),
        axis.text = element_text(color = "black"))
```

outlier after plotting profit (y-axis) vs budget (x-axis)



Find movie with the highest profit

```
clean_profit= mydata
clean_profit$profit <-clean_profit$gross -clean_profit$budget
clean_profit <-clean_profit[order(-clean_profit$profit),]
View(clean_profit)
 max_profit_index <- which.max(clean_profit$profit)
row_zero <- clean_profit[max_profit_index, ]
row_zero
color director_name num_critic_for_reviews duration director_facebook_likes actor_3_facebook_likes     actor_2_name
Color James Cameron                   723      178                         0                    855 Joel David Moore
actor_1_facebook_likes     gross                          genres actor_1_name movie_title num_voted_users cast_total_facebook_likes
                  1000 760505847 Action|Adventure|Fantasy|Sci-Fi  CCH Pounder     Avatar           886204                          4834
actor_3_name facenumber_in_poster                              plot_keywords                                                movie_imdb_link
   Wes Studi                     0 avatar|future|marine|native|paraplegic http://www.imdb.com/title/tt0499549/?ref_=fn_tt_tt_1
num_user_for_reviews language country content_rating   budget title_year actor_2_facebook_likes imdb_score aspect_ratio
                3054  English     USA          PG-13 2.37e+08       2009                    936        7.9         1.78
movie_facebook_likes    profit
               33000 523505847
```

Avatar is the movie with the highest profit

TRAINITY IMDB ANALYSIS.R* | Top_Foreign_Lang_Film

Filter

	movie_title	language
4499	The Good, the Bad and the Ugly	Italian
4030	City of God	Portuguese
4748	Seven Samurai	Japanese
2374	Spirited Away	Japanese
3871	Airlift	Hindi
4260	The Lives of Others	German
4922	Children of Heaven	Persian
1299	Amélie	French
1330	Baahubali: The Beginning	Telugu
2324	Princess Mononoke	Japanese
2971	Das Boot	German
3686	Rang De Basanti	Hindi
4106	Oldboy	Korean

Showing 1 to 14 of 33 entries, 2 total columns

There are 33 non English movie in top 250

Console | Background Jobs

R 4.3.0 · ~/

```
> Top_Foreign_Lang_Film <- top_250[top_250$language != "English", c("movie_title", "language")]
> View(Top_Foreign_Lang_Film)
> count_FOREIGN <- nrow(Top_Foreign_Lang_Film)
> count_FOREIGN
[1] 33
```

Top 250 imdb movies where number of voted user >25000 and show their rank.

```
### Top 250: ####
Create a new column IMDb_Top_250 and store the top 250 movies with the highest IMDb Rating
(corresponding to the column: imdb_score). Also make sure that for all of these movies, the num_voted_users
is greater than 25,000. Also add a Rank column containing the values 1 to 250 indicating the ranks of the
corresponding films.
"clean_top250= mydata
high_votes <- clean_top250[clean_top250$num_voted_users > 25000,]
high_votes_sorted <- high_votes[order(-high_votes$imdb_score),]
high_votes_sorted$IMDb_Top_250 <-high_votes_sorted$movie_title
View(high_votes_sorted)
```

TRAINITY IMDB ANALYSIS.R | high_votes_sorted

Filter

country	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes	IMDb_Top_250
USA	R	25000000	1994	745	9.3	1.85	108000	The Shawshank Redemption
USA	R	6000000	1972	10000	9.2	1.85	43000	The Godfather
USA	PG-13	185000000	2008	13000	9.0	2.35	37000	The Dark Knight
USA	R	13000000	1974	14000	9.0	1.85	14000	The Godfather: Part II
USA	TV-MA	0	0	1000	9.0	1.78	61000	Fargo
USA	PG-13	94000000	2003	857	8.9	2.35	16000	The Lord of the Rings: The Return of the King
USA	R	22000000	1993	795	8.9	1.85	41000	Schindler's List
USA	R	8000000	1994	902	8.9	2.35	45000	Pulp Fiction
Italy	Approved	1200000	1966	34	8.9	2.35	20000	The Good, the Bad and the Ugly
USA	Not Rated	350000	1957	259	8.9	1.66	40000	12 Angry Men
USA	PG-13	160000000	2010	27000	8.8	2.35	175000	Inception
New Zealand	PG-13	93000000	2001	5000	8.8	2.35	21000	The Lord of the Rings: The Fellowship of the Ring

```r
high_votes_sorted$Rank <- 1:nrow(high_votes_sorted)
top_250 <- head(high_votes_sorted[order(-high_votes_sorted$imdb_score),], 250)
top_250
View(top_250)
```

TRAINITY IMDB ANALYSIS.R* × top_250 ×

Filter

	content_rating	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes	IMDb_Top_250	Rank
	R	25000000	1994	745	9.3	1.85	108000	The Shawshank Redemption	1
	R	6000000	1972	10000	9.2	1.85	43000	The Godfather	2
	PG-13	185000000	2008	13000	9.0	2.35	37000	The Dark Knight	3
	R	13000000	1974	14000	9.0	1.85	14000	The Godfather: Part II	4
	TV-MA	0	0	1000	9.0	1.78	61000	Fargo	5
	PG-13	94000000	2003	857	8.9	2.35	16000	The Lord of the Rings: The Return of the King	6
	R	22000000	1993	795	8.9	1.85	41000	Schindler's List	7
	R	8000000	1994	902	8.9	2.35	45000	Pulp Fiction	8
	Approved	1200000	1966	34	8.9	2.35	20000	The Good, the Bad and the Ugly	9
	Not Rated	350000	1957	259	8.9	1.66	40000	12 Angry Men	10
	PG-13	160000000	2010	27000	8.8	2.35	175000	Inception	11
nd	PG-13	93000000	2001	5000	8.8	2.35	21000	The Lord of the Rings: The Fellowship of the Ring	12
	TV-MA	0	0	4	8.8	16.00	55000	Daredevil	13

Showing 1 to 13 of 250 entries, 30 total columns

The topmost movie in top 250 is The Shawshank Redemption

Find top 10 best director with highest mean imdb score

```
library(dplyr)
clean_director <- mydata
director_scores <- aggregate(imdb_score ~ director_name, clean_director, mean)
clean_director$top10director <- director_scores$imdb_score[match(clean_director$director_name, director_scores$director_name)]
clean_director <- clean_director %>% arrange(desc(top10director))
View(clean_director)
```

```
library(dplyr)
top_10_unsorted <- clean_director %>%
  top_n(10, top10director) %>%
  select(director_name, top10director) %>%
  arrange(desc(top10director))
View(top_10_unsorted )
```

TRAINITY IMDB ANALYSIS.R* × top_10_unsorted ×

Filter

	director_name	top10director
1	John Blanchard	9.5
2	Mitchell Altieri	8.7
3	Sadyk Sher-Niyaz	8.7
4	Cary Bell	8.7
5	Mike Mayhall	8.6
6	Charles Chaplin	8.6
7	Raja Menon	8.5
8	Ron Fricke	8.5
9	Damien Chazelle	8.5
10	Majid Majidi	8.5

John Blanchard with score mean IMDB score of 9.5 is the best director

Find top 10 genres

```
> clean_genres=mydata
> genres_scores <- aggregate(imdb_score ~ genres,clean_genres, mean)
> clean_genres$top10genres<- genres_scores$imdb_score[match(clean_genres$genre, genres_scores$genre)]
> clean_genres <- clean_genres %>% arrange(desc(top10genres))
> View(clean_genres)
```

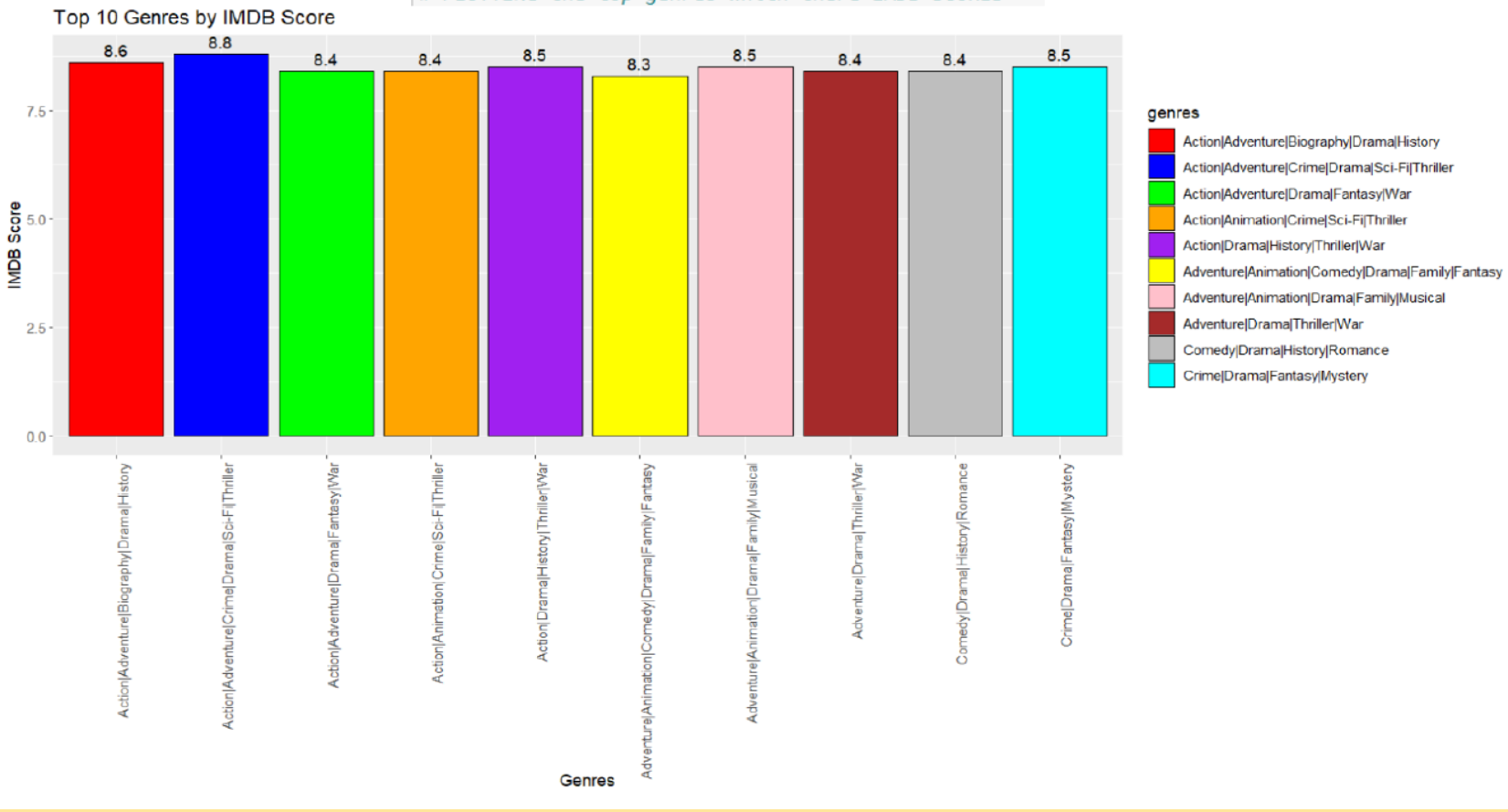
```
#top 10 genres for whom the mean of imdb_score is the highest
top_10_genres <- clean_genres[1:10, c("genres", "top10genres")]
View(top_10_genres)
```

TRAINITY IMDB ANALYSIS.R* × | top_10_genres ×

Filter

	genres	top10genres
1	Action|Adventure|Crime|Drama|Sci-Fi|Thriller	8.8
2	Action|Adventure|Biography|Drama|History	8.6
3	Adventure|Animation|Drama|Family|Musical	8.5
4	Crime|Drama|Fantasy|Mystery	8.5
5	Action|Drama|History|Thriller|War	8.5
6	Action|Adventure|Drama|Fantasy|War	8.4
7	Adventure|Drama|Thriller|War	8.4
8	Comedy|Drama|History|Romance	8.4
9	Action|Animation|Crime|Sci-Fi|Thriller	8.4
10	Adventure|Animation|Comedy|Drama|Family|Fantasy	8.3

```
# PLOTTING the top genres whith there IMDB SCORES
```



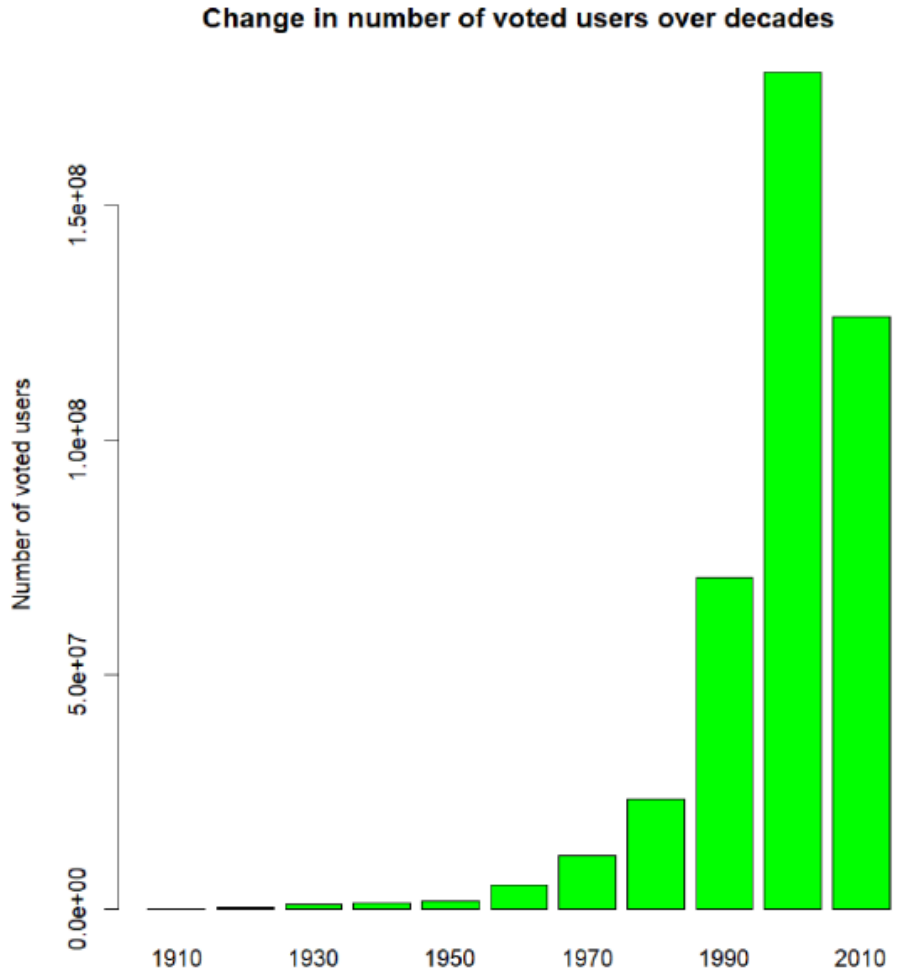
Show movies of Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' after extraction

```
mydata$Meryl_Streep <- ifelse(mydata$actor_1_name == "Meryl Streep", mydata$movie_title, "")
mydata$Leo_Caprio <- ifelse(mydata$actor_1_name == "Leonardo DiCaprio", mydata$movie_title, "")
mydata$Brad_Pitt <- ifelse(mydata$actor_1_name == "Brad Pitt", mydata$movie_title, "")

# Combine movie names with actor names
mydata$Combined <- paste(
  ifelse(mydata$actor_1_name == "Meryl Streep", mydata$movie_title, ""),
  ifelse(mydata$actor_1_name == "Leonardo DiCaprio", mydata$movie_title, ""),
  ifelse(mydata$actor_1_name == "Brad Pitt", mydata$movie_title, ""),
  sep = ", "
)

# Create a new data frame with desired columns
new_data <- subset(mydata, select = c(Meryl_Streep, Leo_Caprio, Brad_Pitt, Combined))

# View the new data frame
View(new_data)
```

	Meryl_Streep	Leo_Caprio	Brad_Pitt	Combined
1107	The River Wild			The River Wild , ,
2782	The Iron Lady			The Iron Lady , ,
1926	The Hours			The Hours , ,
1409	The Devil Wears Prada			The Devil Wears Prada , ,
1576	Out of Africa			Out of Africa , ,
1675	One True Thing			One True Thing , ,
1484	Lions for Lambs			Lions for Lambs , ,
1205	Julie & Julia			Julie & Julia , ,
3642	Julia			Julia , ,
411	It's Complicated			It's Complicated , ,
1619	Hope Springs			Hope Springs , ,
1753	Florence Foster Jenkins			Florence Foster Jenkins , ,
3136	A Prairie Home Companion			A Prairie Home Companion , ,
27		Titanic		, Titanic ,
309		The Wolf of Wall Street		, The Wolf of Wall Street ,
180		The Revenant		, The Revenant ,
1561		The Quick and the Dead		, The Quick and the Dead ,
1423		The Man in the Iron Mask		, The Man in the Iron Mask ,
51		The Great Gatsby		, The Great Gatsby ,
3477		The Great Gatsby		, The Great Gatsby ,
362		The Departed		, The Departed ,
991		The Beach		, The Beach ,
258		The Aviator		, The Aviator ,
453		Shutter Island		, Shutter Island ,

Count movies that belong to each decade

```
"Observe the change in number of voted users over decades using a bar chart.
Create a column called decade which represents the decade to which every movie belongs to.
For example, the title_year year 1923, 1925 should be stored as 1920s.
Sort the column based on the column decade, group it by decade and find
the sum of users voted in each decade. Store this in a new data frame called df_by_decade."
# Defining a function to calculate the decade
get_decade <- function(year) {
  return(floor(year / 10) * 10)
}
# Creating a new column 'decade' in the dataframe 'df_clean'
mydata$decade <- sapply(mydata$title_year, get_decade)
# Calculating the sum of 'num_voted_users' grouped by 'decade'
df_by_decade <- aggregate(num_voted_users ~ decade, data = mydata, sum)
barplot(df_by_decade$num_voted_users, names.arg = df_by_decade$decade,
        xlab = 'Decade', ylab = 'Number of voted users',
        main = 'Change in number of voted users over decades',
        border = 'black', col = 'green')
```



1990 has the highest movie in decades

Number of critic reviews for the movie and number of voted user

```
highest_mean_critic_reviews <- mean_data %>%
  top_n(1, mean_critic_reviews)
highest_mean_critic_reviews
```

```
> highest_mean_critic_reviews
# A tibble: 1 × 3
  actor_1_name   mean_critic_reviews mean_user_reviews
  <chr>                        <dbl>             <dbl>
1 Phaldut Sharma                 738              1885
```

Highest mean(num_critic_for_reviews)

```
highest_mean_user_reviews <- mean_data %>%
  top_n(1, mean_user_reviews)
highest_mean_user_reviews
```

```
> highest_mean_user_reviews
# A tibble: 1 × 3
  actor_1_name    mean_critic_reviews mean_user_reviews
  <chr>                         <dbl>             <dbl>
1 Heather Donahue                 360              3400
```

Highest mean(num_user_for_reviews)

IMDB MOVIE ANALYSIS

MOVIES

movie_title
10th & WolfÂ
11:14Â
12 Angry MenÂ
12 MonkevsÂ

MOVIE BY LANGUAGE

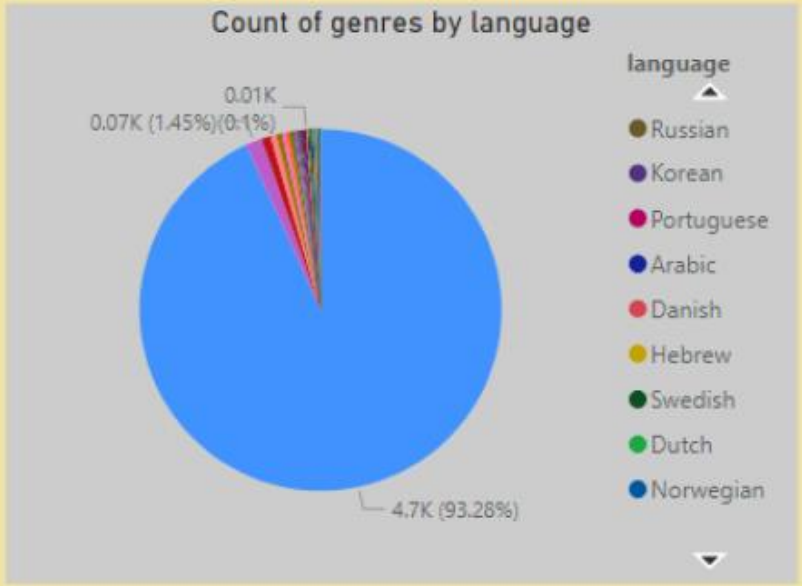
language
Aboriginal
Arabic
Aramaic
Bosnian

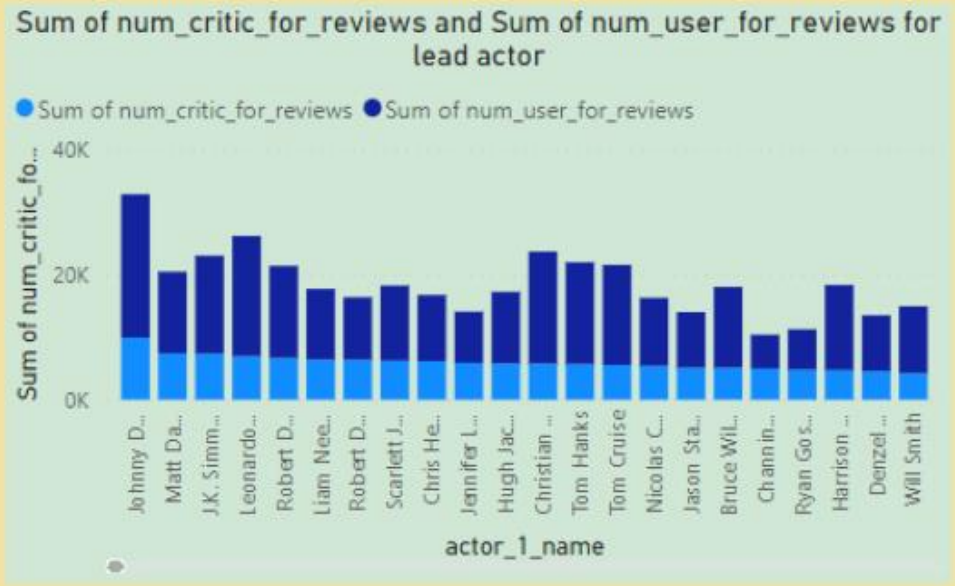
MOVIE BY GENRES

genres
Action
Action|Adventure
Action|Adventure|Animation|Comedy|Crime|Family|Fantasy

MOVIE BY LEAD ACTORS

actor_1_name
50 Cent
A.J. Buckley
Ã"lafur Darri Ã"lafsson
ÂlIex Angulo





Power Bi Dashboard

PROJECT 4 :Hiring Process Analytics

PROBLEM STATEMENT

Perform exploratory data analysis(EDA) to understand how MNCs analyze the trend in hiring of employees.

Data dictionary

- **application_id: Unique identifier for each job application.**
- **Interview Taken on: Date on which the interview was conducted.**
- **Status: Status of the hiring process, indicating whether the candidate was selected, rejected, or in another stage of the process.**
- **event_name: Name or identifier of the hiring event or interview session.**
- **Department: Department or division within the organization for which the position is being hired.**
- **Post Name: Name or title of the job position.**
- **Offered Salary: Salary or compensation offered to the selected candidate.**

OBJECTIVE

- Find how many males and females are hired
- Find how many males and females are rejected
- Find the Average salary for service department people who got hired
- Present a line graph to show the average salary
- Show class interval for each salary bucket with the help of z scores
- Present a pie chart to show people working in different departments
- Show department wise post count
- Present an excel dashboard

RESULTS

Find how many males and females are hired

Find how many males and females are rejected

```
=COUNTIFS(C:C,"Hired",D:D,"male")
```

n on	Status	event_name	Department	Post Name	Offered Salary
4 11:40	Hired	Male	Service Department	c8	56553
4 08:08	Hired	Female	Service Department	c5	22075
4 08:08	Rejected	Male	Service Department	c5	70069
4 16:28	Rejected	Female	Operations Department	i4	3207
4 16:32	Hired	Male	Operations Department	i4	29668
4 07:44	Hired	Male	Sales Department	-	85914
4 16:27	Rejected	Male	Sales Department	i7	69904
4 13:17	Rejected	Male	Sales Department	i7	11758
4 13:09	Hired	Female	Service Department	i4	15156
4 13:11	Rejected	Female	Service Department	i4	49515
4 09:00	Rejected	Male	Service Department	n10	26990
4 10:48	Hired	Female	Service Department	b9	200000
4 10:50	Hired	Male	Service Department	b9	86787
4 09:31	Hired	Male	Finance Department	b9	2308
4 12:48	Hired	Female	Service Department	i7	56688
4 12:48	Hired	-	Service Department	i7	81757
4 08:07	Hired	Male	Service Department	i5	15134
4 08:11	Rejected	-	Service Department	i5	100

2563

hired male = 2563

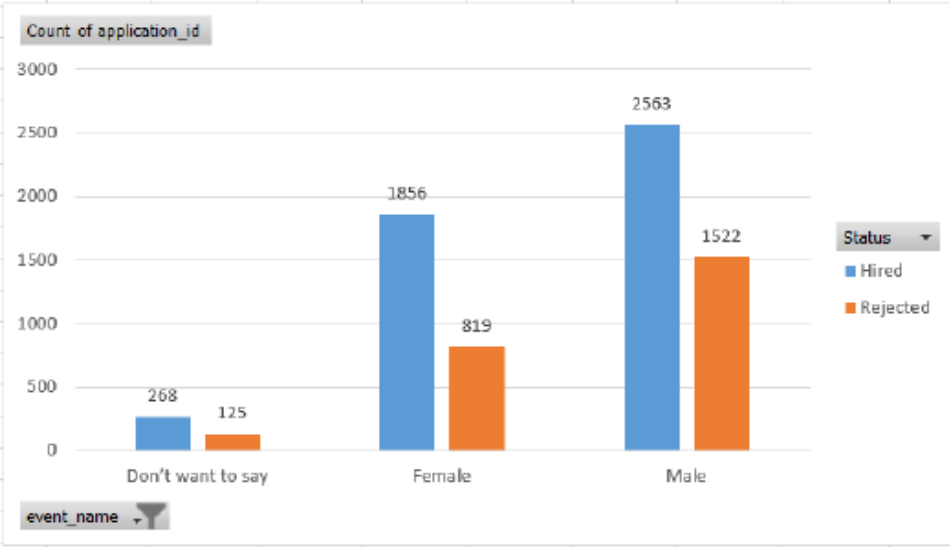
I6 =COUNTIFS(C:C,"Hired",D:D,"Female")

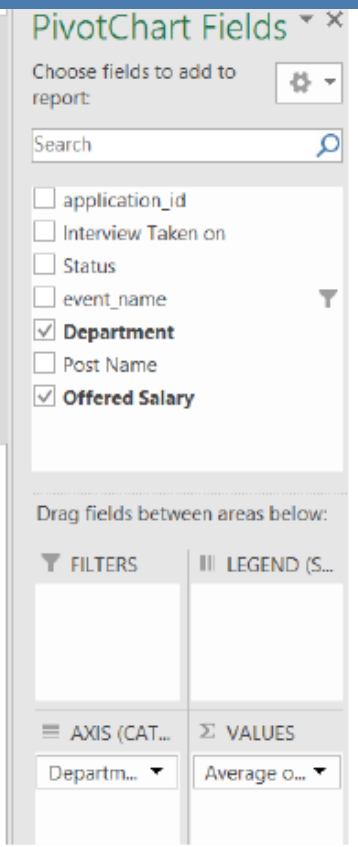
	A	B	C	D	E	F	G	H	I
1	application_id	Interview Taken on	Status	event_name	Department	Post Name	Offered Salary		
2	383422	01-05-2014 11:40	Hired	Male	Service Department	c8	56553		
3	907518	06-05-2014 08:08	Hired	Female	Service Department	c5	22075		
4	176719	06-05-2014 08:08	Rejected	Male	Service Department	c5	70069		
5	429799	02-05-2014 16:28	Rejected	Female	Operations Department	i4	3207		
6	253651	02-05-2014 16:32	Hired	Male	Operations Department	i4	29668		=COUNTIFS(C:C,"Hired",D:D,"Female")
7	289907	01-05-2014 07:44	Hired	Male	Sales Department	-	85914		
8	959124	06-05-2014 16:27	Rejected	Male	Sales Department	i7	69904		
9	86642	09-05-2014 13:17	Rejected	Male	Sales Department	i7	11758		
10	751029	02-05-2014 13:09	Hired	Female	Service Department	i4	15156		
11	434547	02-05-2014 13:11	Rejected	Female	Service Department	i4	49515		
12	518854	01-05-2014 09:00	Rejected	Male	Service Department	n10	26990		
13	649039	07-05-2014 10:48	Hired	Female	Service Department	b9	200000		
14	199526	07-05-2014 10:50	Hired	Male	Service Department	b9	86787		
15	539803	15-05-2014 09:31	Hired	Male	Finance Department	b9	2308		
16	191009	09-05-2014 12:48	Hired	Female	Service Department	i7	56688		
17	195323	09-05-2014 12:48	Hired	-	Service Department	i7	81757		
18	51318	02-05-2014 08:07	Hired	Male	Service Department	i5	15134		
19	742283	02-05-2014 08:11	Rejected	-	Service Department	i5	100		
20	513166	01-05-2014 22:53	Hired	Female	Operations Department	i1	73579		
21	791372	01-05-2014 22:54	Rejected	Male	Operations Department	i1	50351		
22	47857	01-05-2014 22:55	Rejected	Female	Operations Department	i1	38462		

We can also use this formula of COUNTIFS for hired female

hired female = 1856

Count of application_id	Column Labels		
Row Labels	Hired	Rejected	Grand Total
Don't want to say	268	125	393
Female	1856	819	2675
Male	2563	1522	4085
Grand Total	**4687**	**2466**	**7153**





WE CAN ALSO COUNT FEMALES AND MALES HIRED AND REJECTED THROUGH THE PIVOT TABLE AND PLOT IN THE BAR GRAPH

Find the Average salary for service department people who got hired

I10 =COUNTIFS(C:C,"Hired",E:E,"Service Department")

	A	B	C	D	E	F	G	H	I
1	application_id	Interview Taken on	Status	event_name	Department	Post Name	Offered Salary		
2	383422	01-05-2014 11:40	Hired	Male	Service Department	c8	56553		
3	907518	06-05-2014 08:08	Hired	Female	Service Department	c5	22075		
4	176719	06-05-2014 08:08	Rejected	Male	Service Department	c5	70069		
5	429799	02-05-2014 16:28	Rejected	Female	Operations Department	i4	3207		
6	253651	02-05-2014 16:32	Hired	Male	Operations Department	i4	29668		
7	289907	01-05-2014 07:44	Hired	Male	Sales Department	-	85914		
8	959124	06-05-2014 16:27	Rejected	Male	Sales Department	i7	69904		
9	86642	09-05-2014 13:17	Rejected	Male	Sales Department	i7	11758		
10	751029	02-05-2014 13:09	Hired	Female	Service Department	i4	15156		=COUNTIFS(C:C,"Hired",E:E,"Service Department")
11	434547	02-05-2014 13:11	Rejected	Female	Service Department	i4	49515		

I10 =COUNTIFS(C:C,"Hired",E:E,"Service Department")

	A	B	C	D	E	F	G	H	I
1	application_id	Interview Taken on	Status	event_name	Department	Post Name	Offered Salary		
2	383422	01-05-2014 11:40	Hired	Male	Service Department	c8	56553		
3	907518	06-05-2014 08:08	Hired	Female	Service Department	c5	22075		
4	176719	06-05-2014 08:08	Rejected	Male	Service Department	c5	70069		
5	429799	02-05-2014 16:28	Rejected	Female	Operations Department	i4	3207		
6	253651	02-05-2014 16:32	Hired	Male	Operations Department	i4	29668		
7	289907	01-05-2014 07:44	Hired	Male	Sales Department	-	85914		
8	959124	06-05-2014 16:27	Rejected	Male	Sales Department	i7	69904		
9	86642	09-05-2014 13:17	Rejected	Male	Sales Department	i7	11758		
10	751029	02-05-2014 13:09	Hired	Female	Service Department	i4	15156		1332
11	434547	02-05-2014 13:11	Rejected	Female	Service Department	i4	49515		
12	518854	01-05-2014 09:00	Rejected	Male	Service Department	n10	26990		

=67331965/1332

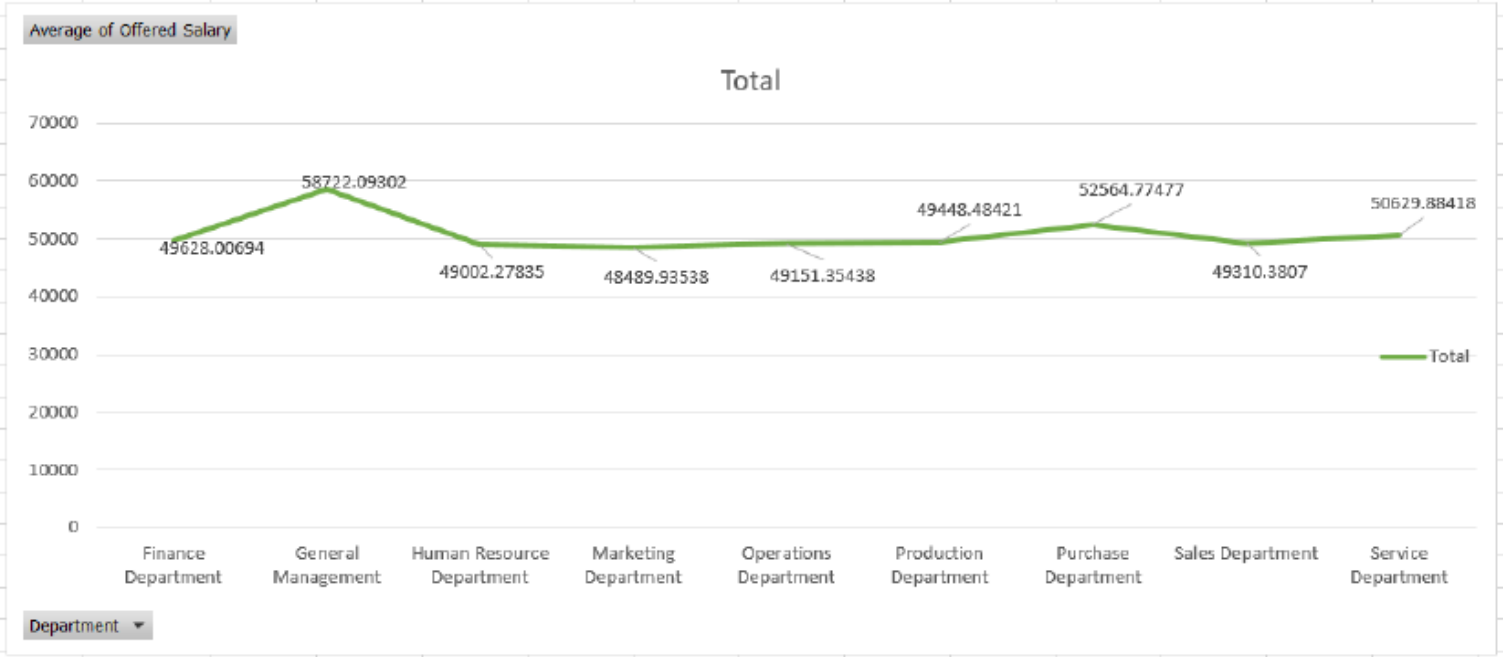
50549.52327

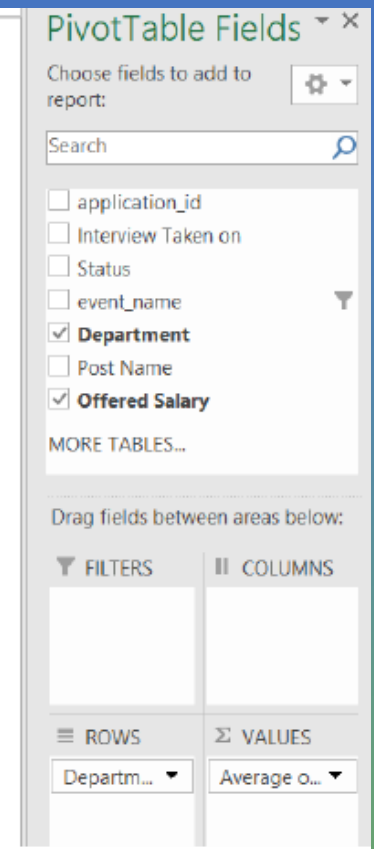
AVERAGE SALARY FOR SERVICE DEPARTMENT PEOPLE WHO GOT HIRED

=50549.52327

Present a line graph to show the average salary

Row Labels	Average of Offered Salary
Finance Department	49628.00694
General Management	58722.09302
Human Resource Department	49002.27835
Marketing Department	48489.93538
Operations Department	49151.35438
Production Department	49448.48421
Purchase Department	52564.77477
Sales Department	49310.3807
Service Department	50629.88418
Grand Total	**49983.02902**





AVERAGE SALARY DEPARTMENT-WISE AND OVERALL

Show class interval for each salary bucket with the help of z scores

Class Intervals: The class interval is the difference between the upper-class limit and the lower class limit. Draw the class intervals for salary in the company

M1 | SAMPLE_LIST

	E	F	G	H	I	J	K	L	M	N
1	**Department**	**Post Name**	**Offered Salary**						SAMPLE_LIST	
2	Service Department	c8	56553						1	
3	Service Department	c5	22075						1	
4	Service Department	c5	70069						1	
5	Operations Department	i4	3207		Upper limit = Sample mean + Z-score * Standard deviation of sample / SQRT(s)				1	
6	Operations Department	i4	29668		Lower limit = Sample mean - Z-score * Standard deviation of sample / SQRT(s)				1	
7	Sales Department	-	85914						1	
8	Sales Department	i7	69904						1	
9	Sales Department	i7	11758						1	
10	Service Department	i4	15156						1	
11	Service Department	i4	49515						1	
12	Service Department	n10	26990						1	
13	Service Department	b9	200000						1	
14	Service Department	b9	86787						1	
15	Finance Department	b9	2308						1	
16	Service Department	i7	56688						1	
17	Service Department	i7	81757						1	
18	Service Department	i5	15134						1	
19	Service Department	i5	100						1	
20	Operations Department	i1	73579						1	
21	Operations Department	i1	50351						1	
22	Operations Department	i1	38462						1	
23	Operations Department	i1	82510						1	
24	Service Department	i6	52554						1	
25	Operations Department	i7	3423						1	
26	Service Department	i1	88744						1	
27	Service Department	i1	70979						1	
28	Operations Department	i6	99574						1	

CREATING SAMPLE LIST WITH 72 SAMPLES WITH 100 DATA POINT EACH

P2 =AVERAGEIFS(G2:G7169,M2:M7169,O2)

	E	F	G	H	I	J	K	L	M	N	O	P
1	**Department**	**Post Name**	**Offered Salary**						SAMPLE_LIST		BUCKET	SAMPLE AVERAGE
2	Service Department	c8	56553						1		1	52317.60204
3	Service Department	c5	22075						1		2	49918.64
4	Service Department	c5	70069						1		3	51335.55
5	Operations Department	i4	3207		Upper limit = Sample mean + Z-score * Standard deviation of sample / SQRT(s)				1		4	48601.88
6	Operations Department	i4	29668		Lower limit = Sample mean - Z-score * Standard deviation of sample / SQRT(s)				1		5	51097.83
7	Sales Department	-	85914						1		6	48760.47
8	Sales Department	i7	69904						1		7	52679.08
9	Sales Department	i7	11758						1		8	40395.58
10	Service Department	i4	15156						1		9	48645.33
11	Service Department	i4	49515						1		10	51866.79
12	Service Department	n10	26990						1		11	52317.92
13	Service Department	b9	200000						1		12	49169.67
14	Service Department	b9	86787						1		13	49548.03
15	Finance Department	b9	2308						1		14	49696.05
16	Service Department	i7	56688						1		15	50040.84
17	Service Department	i7	81757						1		16	51010.5
18	Service Department	i5	15134						1		17	52719.23
19	Service Department	i5	100						1		18	51732.75
20	Operations Department	i1	73579						1		19	50179.99
21	Operations Department	i1	50351						1		20	47805.67
22	Operations Department	i1	38462						1		21	49422.52
23	Operations Department	i1	82510						1		22	49305.9
24	Service Department	i6	52554						1		23	49957.22
25	Operations Department	i7	3423						1		24	52773.35
26	Service Department	i1	88744						1		25	54540.2

SAMPLE AVERAGE FOR EACH BUCKET

Q2 | =STDEV.P(G2:G100)

	F	G	H	I	J	K	L	M	N	O	P	Q
1	**Post Name**	**Offered Salary**						SAMPLE_LIST		BUCKET	SAMPLE AVERAGE	ST_DEV
2	c8	56553						1		1	52317.6[illegible]4	32554.13
3	c5	22075						1		2	49918.64	28963.45
4	c5	70069						1		3	51335.55	44786.69
5	i4	3207		Upper limit = Sample mean + Z-score * Standard deviation of sample / SQRT(s)				1		4	48601.88	26686.81
6	i4	29668		Lower limit = Sample mean - Z-score * Standard deviation of sample / SQRT(s)				1		5	51097.83	26799.33
7	-	85914						1		6	48760.47	
8	i7	69904						1		7	52679.08	
9	i7	11758						1		8	40395.58	
10	i4	15156						1		9	48645.33	
11	i4	49515						1		10	51866.79	
12	n10	26990						1		11	52317.92	
13	b9	200000						1		12	49169.67	
14	b0	86787						1		12	40548.02	

STANDARD DEVIATION FOR BUCKET 1

R1 | Z SCORE

	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	**Post Name**	**Offered Salary**						SAMPLE_LIST		BUCKET	SAMPLE AVERAGE	ST_DEV	Z SCORE
2	c8	56553						1		1	52317.60204	32554.13	0.130103
3	c5	22075						1		2	49918.64	28963.45	-0.92899
4	c5	70069						1		3	51335.55	44786.69	0.545289
5	i4	3207		Upper limit = Sample mean + Z-score * Standard deviation of sample / SQRT(s)				1		4	48601.88	26686.81	-1.50858
6	i4	29668		Lower limit = Sample mean - Z-score * Standard deviation of sample / SQRT(s)				1		5	51097.83	26799.33	-0.69575
7	-	85914						1		6	48760.47		1.032016
8	i7	69904						1		7	52679.08		0.54022
9	i7	11758						1		8	40395.58		-1.24591
10	i4	15156						1		9	48645.33		-1.14153
11	i4	49515						1		10	51866.79		-0.08609
12	n10	26990						1		11	52317.92		-0.77802
13	b9	200000						1		12	49169.67		4.536518
14	b9	86787						1		13	49548.03		1.058833
15	b9	2308						1		14	49696.05		-1.5362
16	i7	56688						1		15	50040.84		0.13425
17	i7	81757						1		16	51010.5		0.904321
18	i5	15134						1		17	52719.23		-1.14221
19	i5	100						1		18	51732.75		-1.60402
20	i1	73579						1		19	50179.99		0.653109
21	i1	50351						1		20	47805.67		-0.06041
22	i1	38462						1		21	49422.52		-0.42562
23	i1	82510						1		22	49305.9		0.927452
24	i6	52554						1		23	49957.22		0.007262
25	i7	3423						1		24	52773.35		-1.50195

Z SCORES FOR BUCKET 1

T1 =AVERAGE(R2:R100)

	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	**Offered Salary**						SAMPLE_LIST		BUCKET	SAMPLE AVERAGE	ST_DEV	Z SCORE	AVG Z SCORE FOR BUCKET 1	-0.01623
2	56553						1		1	52317.60204	32554.13	0.130103		
3	22075						1		2	49918.64	28963.45	-0.92899		
4	70069						1		3	51335.55	44786.69	0.545289		
5	3207		Upper limit = Sample mean + Z-score * Standard deviation of sample / SQRT(s)				1		4	48601.88	26686.81	-1.50858		
6	29668		Lower limit = Sample mean - Z-score * Standard deviation of sample / SQRT(s)				1		5	51097.83	26799.33	-0.69575		
7	85914						1		6	48760.47		1.032016		
8	69904						1		7	52679.08		0.54022		
9	11758													

SO WE WILL TAKE AVERAGE Z SCORE FOR BUCKET 1 TO CALCULATE UPPER AND LOWER RANGE

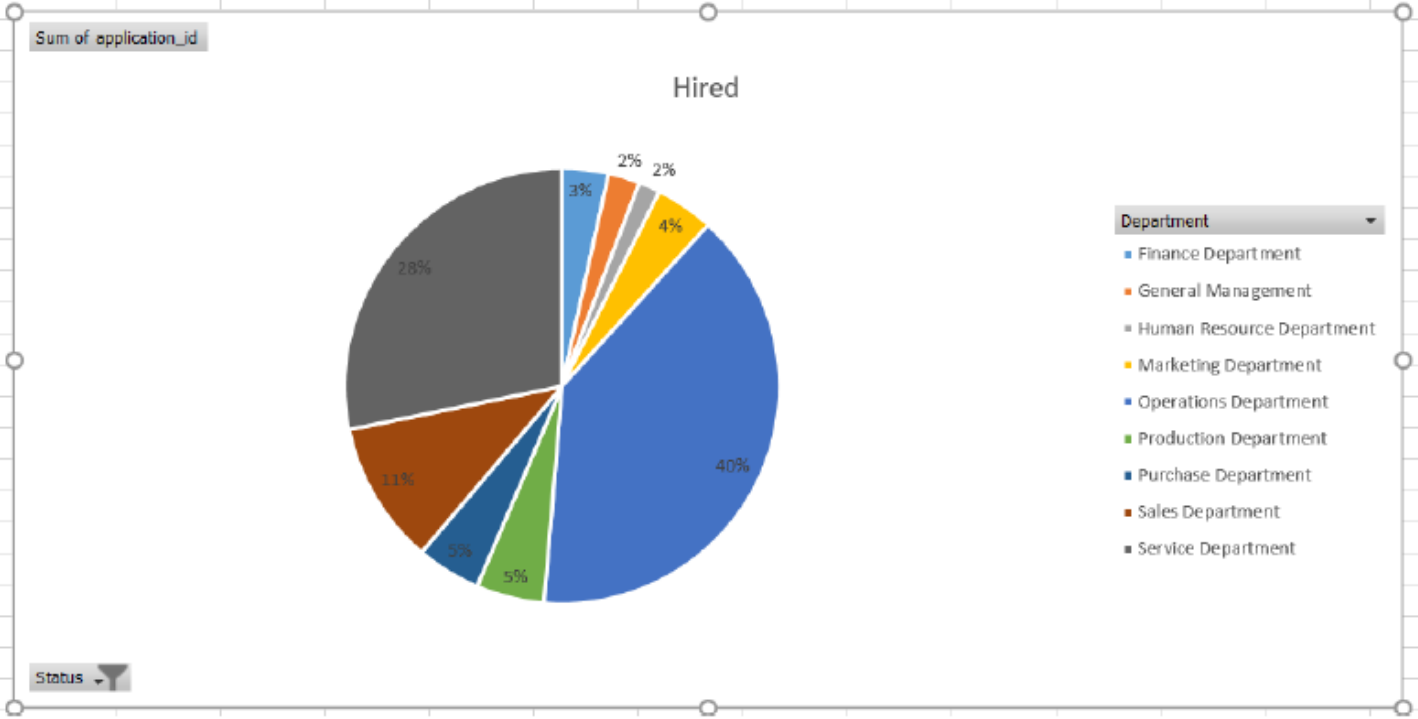
V8

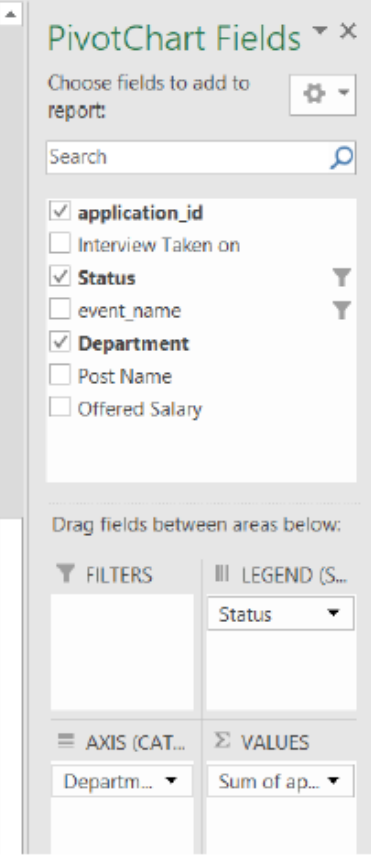
	O	P	Q	R	S	T	U	V
1	BUCKET	SAMPLE AVERAGE	ST_DEV	Z SCORE	AVG Z SCORE FOR BUCKET 1	-0.01623	STANDARD ERROR=Z-score * Standard deviation of sample / SQRT(s)	Upper limit _ BUCKET 1= Sample mean + STANDARD ERROR
2	1	52317.60204	32554.13	0.130103			-52.84606267	52370.4481
3	2	49918.64	28963.45	-0.92899				
4	3	51335.55	44786.69	0.545289				
5	4	48601.88	26686.81	-1.50858				
6	5	51097.83	26799.33	-0.69575				Lower limit _BUCKET1= Sample mean - STANDARD ERROR
7	6	48760.47		1.032016				52264.75598
8	7	52679.08		0.54022				
9	8	40395.58		-1.24591				
10	9	48645.33		-1.14153				
11	10	51866.79		-0.08609				
12	11	52317.92		-0.77802				
13	12	49169.67		4.536518				
14	13	49548.03		1.058833				
15	14	49696.05		-1.5362				
16	15	50040.84		0.13425				
17	16	51010.5		0.904321				
18	17	52719.23		-1.14221				
19	18	51732.75		-1.60402				
20	19	50179.99		0.653109				
21	20	47805.67		-0.06041				

UPPER LIMIT AND LOWER LIMIT OF BUCKET 1. SIMILARLY, WE CAN FIND OTHERS

Present a pie chart to show people working in different departments

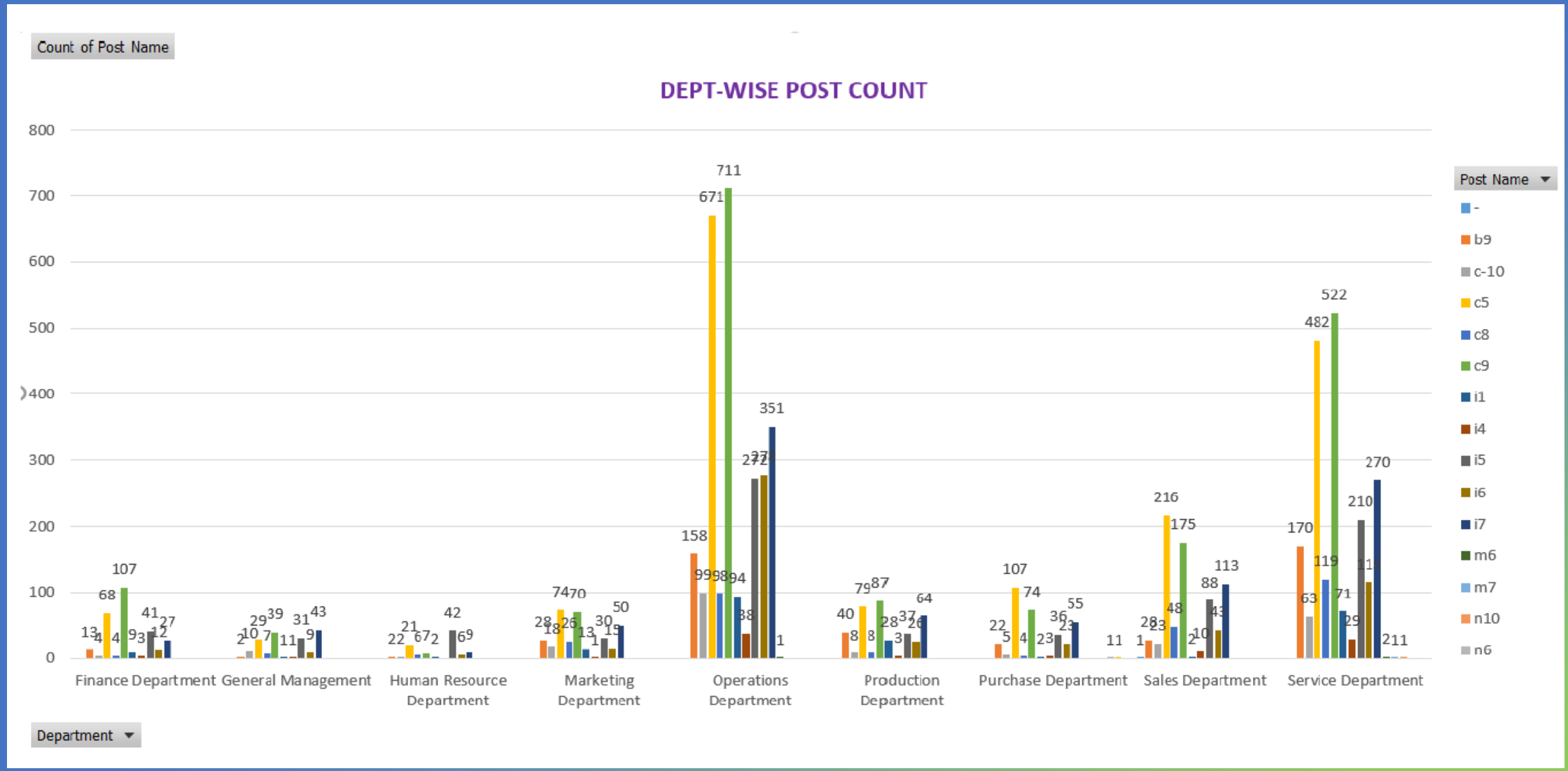
Sum of application_id	Column Labels	
Row Labels	Hired	Grand Total
Finance Department	82096471	82096471
General Management	55696016	55696016
Human Resource Department	38271751	38271751
Marketing Department	103030245	103030245
Operations Department	941934194	941934194
Production Department	120614576	120614576
Purchase Department	113416207	113416207
Sales Department	252490716	252490716
Service Department	670683246	670683246
Grand Total	**2378233422**	**2378233422**



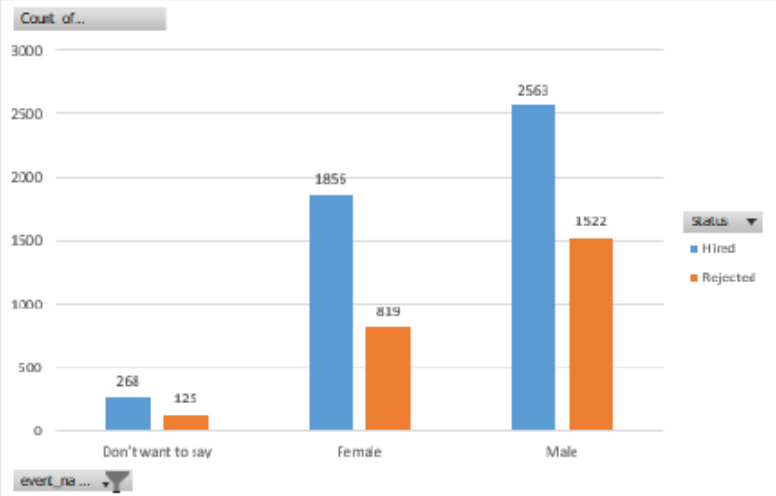


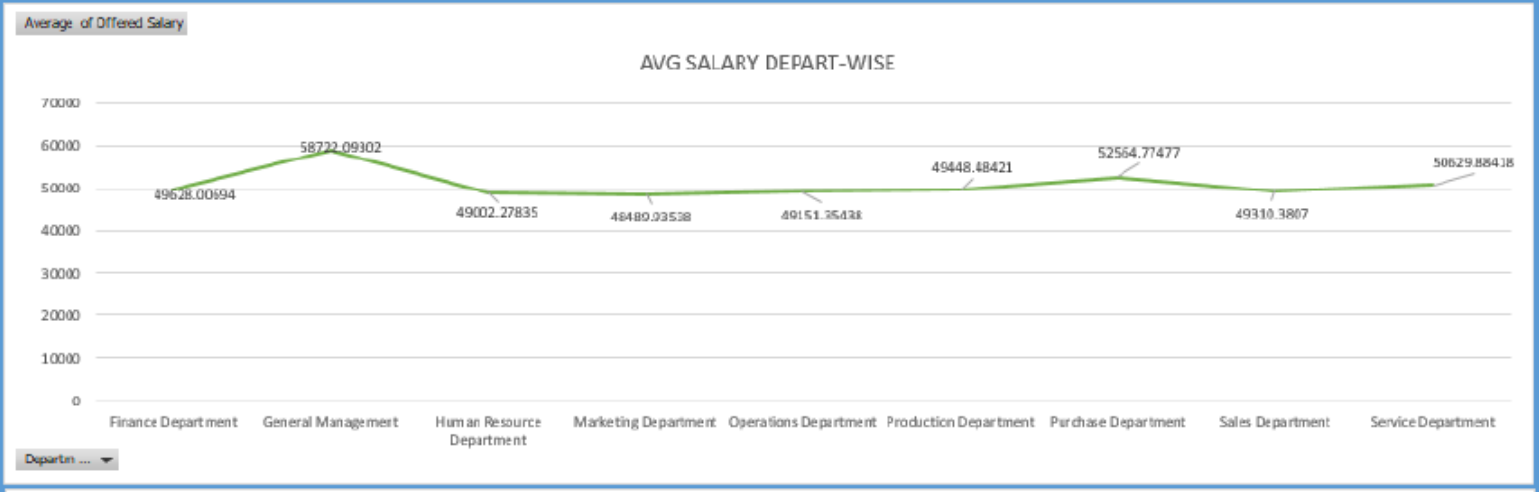
PROPORTION OF PEOPLE WORKING IN DIFFERENT DEPARTMENT

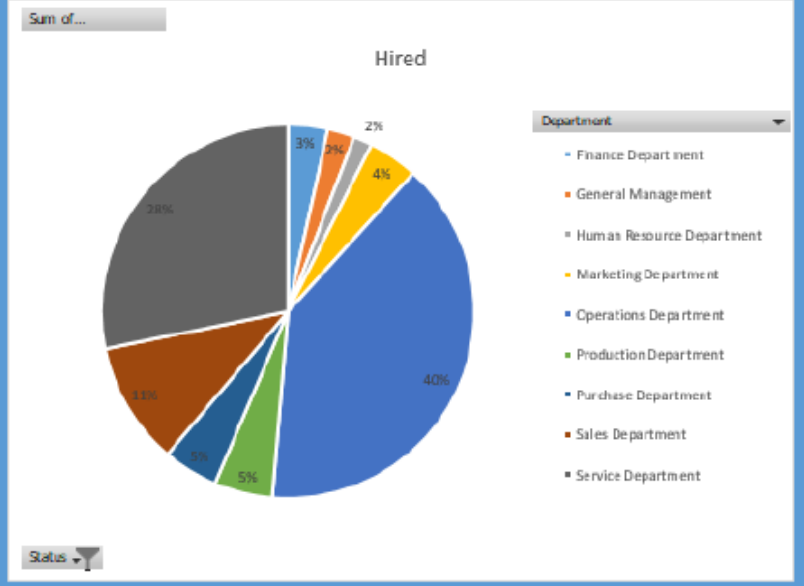
Show department wise post count

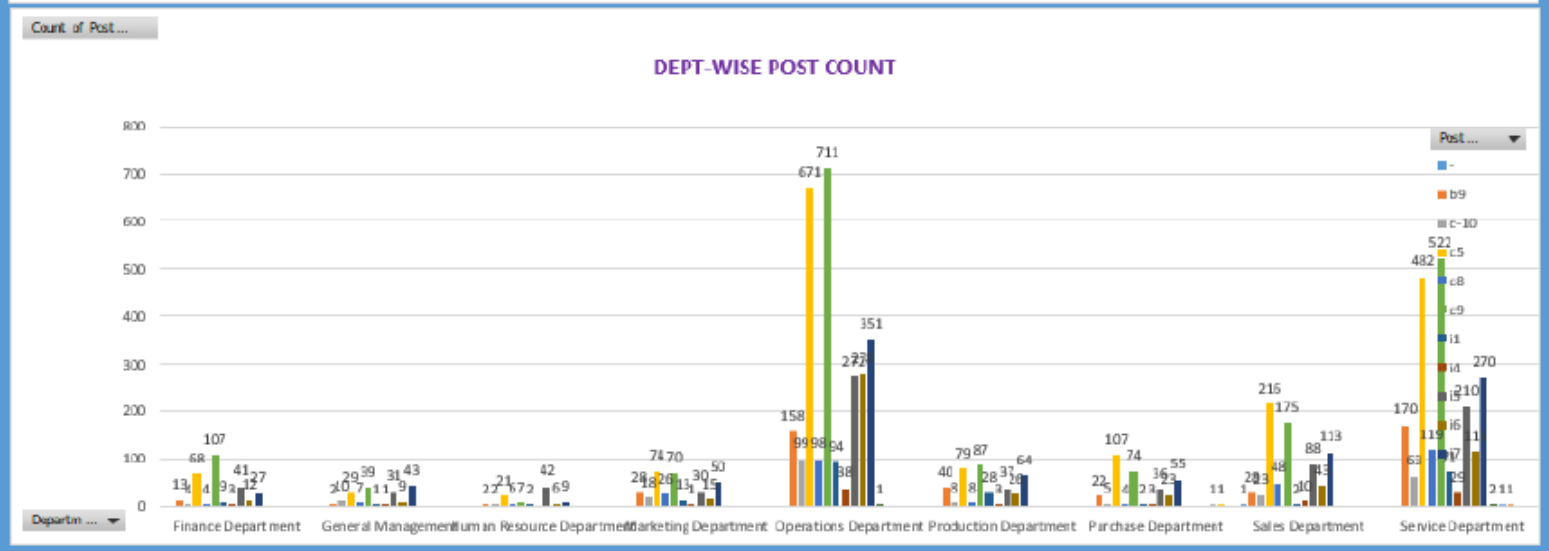


DASHBOARD









Excel Dashboard

PROJECT 5:Operation Analytics and Investigating Metric Spike

PROBLEM STATEMENT

Find analysis report for job data and investigating metric spike

Data dictionary

- **user_id: Unique identifier for each user in the system. It is used to associate operations and activities with specific users.**
- **created_at: Timestamp indicating the date and time when the operation was created or initiated.**
- **company_id: Identifier for the company associated with the operation. It links the operation to a specific company within the system.**
- **language: Indicates the language used for the operation. It represents the preferred language setting for the user or the language of the operation itself.**
- **activated_at: Timestamp indicating the date and time when the operation was activated or made active. It may refer to the moment when the operation becomes operational or accessible to users.**
- **state: Represents the current state or status of the operation. It could indicate whether the operation is active, inactive, completed, in progress, or any other relevant state.**

OBJECTIVE

- Job data
  - Calculate the number of jobs reviewed per hour per day
  - Built a Regression model to predict future jobs review
  - Calculate 7-day rolling average of throughput
  - Calculate the percentage share of each language in the last 30 days
- Investigating metric spike
  - Calculate the weekly user engagement
  - Calculate the user growth for product
  - Calculate the weekly engagement per device
  - Calculate the email engagement metrics

RESULTS

Calculate the number of jobs reviewed per hour per day

```sql
--Number of jobs reviewed: Amount of jobs reviewed over time .
--task: Calculate the number of jobs reviewed per hour per day for November 2020?

SELECT CAST(ds AS date) AS date, DATEPART(hour, CAST(ds AS datetime)) AS hour, COUNT(*) AS jobs_reviewed, SUM(time_spent)/3600.0 AS total_hours
FROM job_data
WHERE ds >= '2020-11-01' AND ds < '2020-12-01'
GROUP BY CAST(ds AS date), DATEPART(hour, CAST(ds AS datetime))
ORDER BY CAST(ds AS date), DATEPART(hour, CAST(ds AS datetime))
```

100 %

Results | Messages

	date	hour	jobs_reviewed	total_hours
1	2020-11-25	0	1	0.012500
2	2020-11-26	0	1	0.015555
3	2020-11-27	0	1	0.028888
4	2020-11-28	0	2	0.009166
5	2020-11-29	0	1	0.005555
6	2020-11-30	0	2	0.011111

Built a Regression model to predict future jobs review

```
import pandas as pd
from sklearn.linear_model import LinearRegression

# create a DataFrame with the data
data = {'date': ['2020-11-25', '2020-11-26', '2020-11-27', '2020-11-28', '2020-11-29', '2020-11-30'],
        'hour': [0, 0, 0, 0, 0, 0],
        'jobs_reviewed': [1, 1, 1, 2, 1, 2],
        'total_hours': [0.012500, 0.015555, 0.028888, 0.009166, 0.005555, 0.011111]}
df = pd.DataFrame(data)

# create a LinearRegression object
lr = LinearRegression()

# fit the model
lr.fit(df[['jobs_reviewed']], df['total_hours'])

# print the intercept and coefficients
print(lr.intercept_)
print(lr.coef_)

# make predictions
predicted_hours = lr.predict(df[['jobs_reviewed']])

# add the predicted hours to the DataFrame
df['predicted_hours'] = predicted_hours

# print the updated DataFrame
print(df)

# make a prediction for a value of 5 jobs_reviewed
prediction = lr.predict([[5]])

# print the prediction
print("Predicted total hours for 5 jobs reviewed:", prediction[0])
```

```
0.0211105
[-0.005486]
         date  hour  jobs_reviewed  total_hours  predicted_hours
0  2020-11-25     0              1     0.012500         0.015624
1  2020-11-26     0              1     0.015555         0.015624
2  2020-11-27     0              1     0.028888         0.015624
3  2020-11-28     0              2     0.009166         0.010138
4  2020-11-29     0              1     0.005555         0.015624
5  2020-11-30     0              2     0.011111         0.010138
Predicted total hours for 5 jobs reviewed: -0.006319500000000006
```

intercept is 0.0211105 coefficient for 'jobs_reviewed' is -0.005486.

The predicted total hours for 5 jobs reviewed is negative because the model is linear, and it can produce negative predicted values if the slope of the line is negative and the intercept is large enough.

The slope and intercept coefficients represent the relationship between the independent variable (in this case, 'jobs_reviewed') and the dependent variable (in this case, 'total_hours').

The slope coefficient (also known as the regression coefficient) represents the change in the dependent variable that is associated with a one-unit increase in the independent variable. In this case, the slope coefficient is -0.005486, which means that for each additional job reviewed, we would expect the total hours worked to decrease by 0.005486 hours, on average.

The predicted total hours for 5 jobs reviewed is -0.006319500000000006, which is a negative value. This is not a meaningful prediction, and it suggests that the linear regression model may not be appropriate for this particular dataset.

Calculate 7-day rolling average of throughput

```sql
--Throughput: It is the no. of events happening per second.
--task: Calculate 7 day rolling average of throughput?
--For throughput, do you prefer daily metric or 7-day rolling and why?

--7 day rolling average of throughput
  SELECT
  ds,
  (
    SELECT AVG(time_spent)
    FROM job_data
    WHERE ds BETWEEN DATEADD(day, -6, t.ds) AND t.ds
  ) AS rolling_avg_throughput
FROM
  job_data t
GROUP BY
  ds
ORDER BY
  ds ASC;
```

100 %

Results | Messages

	ds	rolling_avg_throughput
1	2020-11-25	45
2	2020-11-26	50
3	2020-11-27	68
4	2020-11-28	47
5	2020-11-29	43
6	2020-11-30	37

Weekly Average throughput is 48.33

Calculate the percentage share of each language in the last 30 days

```sql
--Percentage share of each language: Share of each language for different contents.
SELECT
event,
language,
COUNT(*) AS count,
100.0 * COUNT(*) / SUM(COUNT(*)) OVER (PARTITION BY event) AS percentage
FROM job_data
GROUP BY event, language;
```

100 %

Results | Messages

	event	language	count	percentage
1	decision	French	1	33.333333333333
2	decision	Hindi	1	33.333333333333
3	decision	Persian	1	33.333333333333
4	skip	English	1	50.000000000000
5	skip	Persian	1	50.000000000000
6	transfer	Arabic	1	33.333333333333
7	transfer	Italian	1	33.333333333333
8	transfer	Persian	1	33.333333333333

Calculate the weekly user engagement

```sql
--(Investigating metric spike)

--the weekly user engagement(in hour)
  SELECT
    DATEADD(week, DATEDIFF(week, 0, e.occurred_at), 0) AS week_start,
    COUNT(DISTINCT e.user_id) AS weekly_engaged_users
FROM  [Table-2 events] e
INNER JOIN [Table-1 users] u ON e.user_id = u.user_id
WHERE e.event_type = 'engagement'
GROUP BY DATEADD(week, DATEDIFF(week, 0, e.occurred_at), 0)
ORDER BY week_start ASC;
```

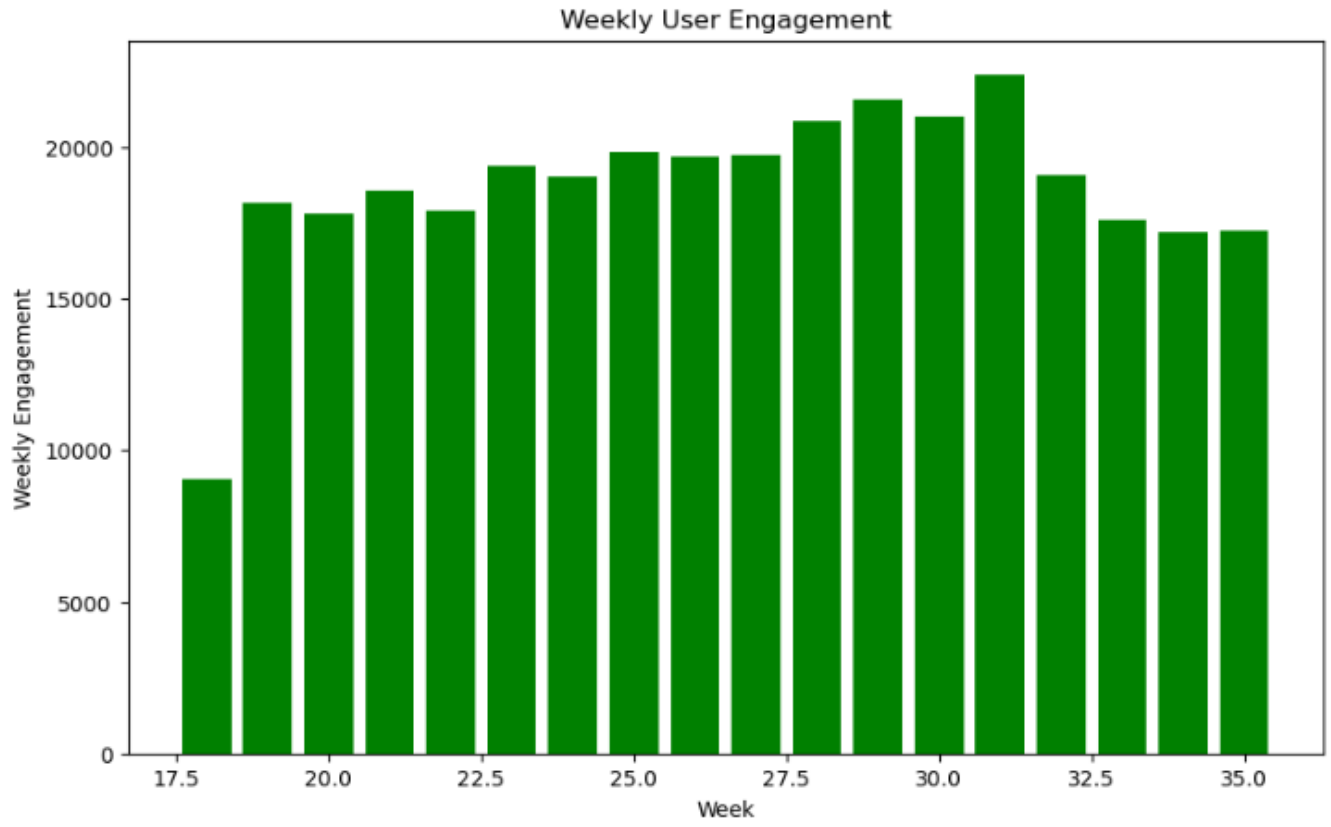
99 %

Results | Messages

	week_start	weekly_engaged_users
1	2014-04-28 00:00:00.000	663
2	2014-05-05 00:00:00.000	1068
3	2014-05-12 00:00:00.000	1113
4	2014-05-19 00:00:00.000	1154
5	2014-05-26 00:00:00.000	1121
6	2014-06-02 00:00:00.000	1186
7	2014-06-09 00:00:00.000	1232
8	2014-06-16 00:00:00.000	1275
9	2014-06-23 00:00:00.000	1264
10	2014-06-30 00:00:00.000	1302
11	2014-07-07 00:00:00.000	1372
12	2014-07-14 00:00:00.000	1365
13	2014-07-21 00:00:00.000	1376

Plotting the weekly user engagement in python

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.bar(weekly_engagement['Weekly_Activity'], weekly_engagement['Weekly_Engagement'], color='green')
plt.xlabel('Week')
plt.ylabel('Weekly Engagement')
plt.title('Weekly User Engagement')
plt.show()
```



Calculate the user growth for product

```sql
--user growth for product
SELECT
  YEAR(created_at) AS year,
  COUNT(DISTINCT ue.user_id) AS new_users,
  COUNT(DISTINCT CASE WHEN ua.user_id IS NOT NULL THEN ue.user_id END) AS activated_users,
  (COUNT(DISTINCT CASE WHEN ua.user_id IS NOT NULL THEN ue.user_id END) * 100.0) / COUNT(DISTINCT ue.user_id) AS user_growth
FROM [master].[dbo].[Table-1 users] AS ue
LEFT JOIN [master].[dbo].[Table-2 events] AS ua
  ON ue.user_id = ua.user_id
  AND ua.event_name = 'home_page'
  AND YEAR(ua.occurred_at) BETWEEN 2013 AND 2014
WHERE YEAR(ue.created_at) BETWEEN 2013 AND 2014
GROUP BY YEAR(created_at)
ORDER BY YEAR(created_at);
```

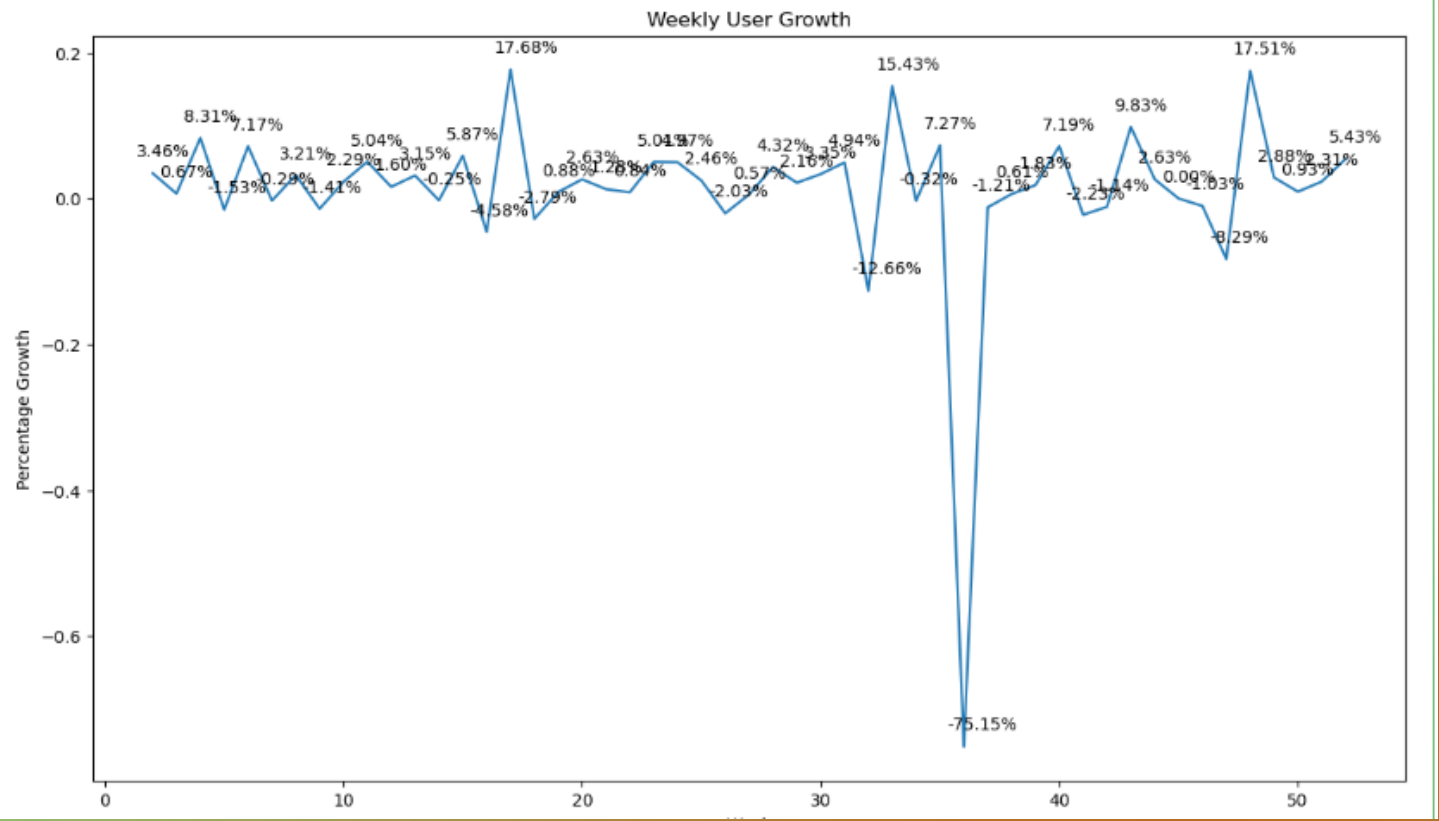
99 %

Results | Messages

	year	new_users	activated_users	user_growth
1	2013	6981	1276	18.278183641312
2	2014	12085	4690	38.808440215142

USER GROWTH FOR A PARTICULAR PRODUCT ‘HOME PAGE’

```python
import pandas as pd
import matplotlib.pyplot as plt

# merge user_df and event_df on user_id
merged_df = pd.merge(user_df, event_df, on='user_id', how='inner')

# plot growth over time
fig, ax = plt.subplots(figsize=(14, 8))
ax.plot(weekly_signups['created_at'], weekly_signups['growth'])
ax.set_title('Weekly User Growth')
ax.set_xlabel('Week')
ax.set_ylabel('Percentage Growth')

# add labels to each data point
for i, row in weekly_signups.iterrows():
    ax.annotate('{:.2f}%'.format(row['growth']*100), xy=(row['created_at'], row['growth']), xytext=(-10,10), textcoords='offset p

plt.show()
```



Plotting the weekly user growth in python

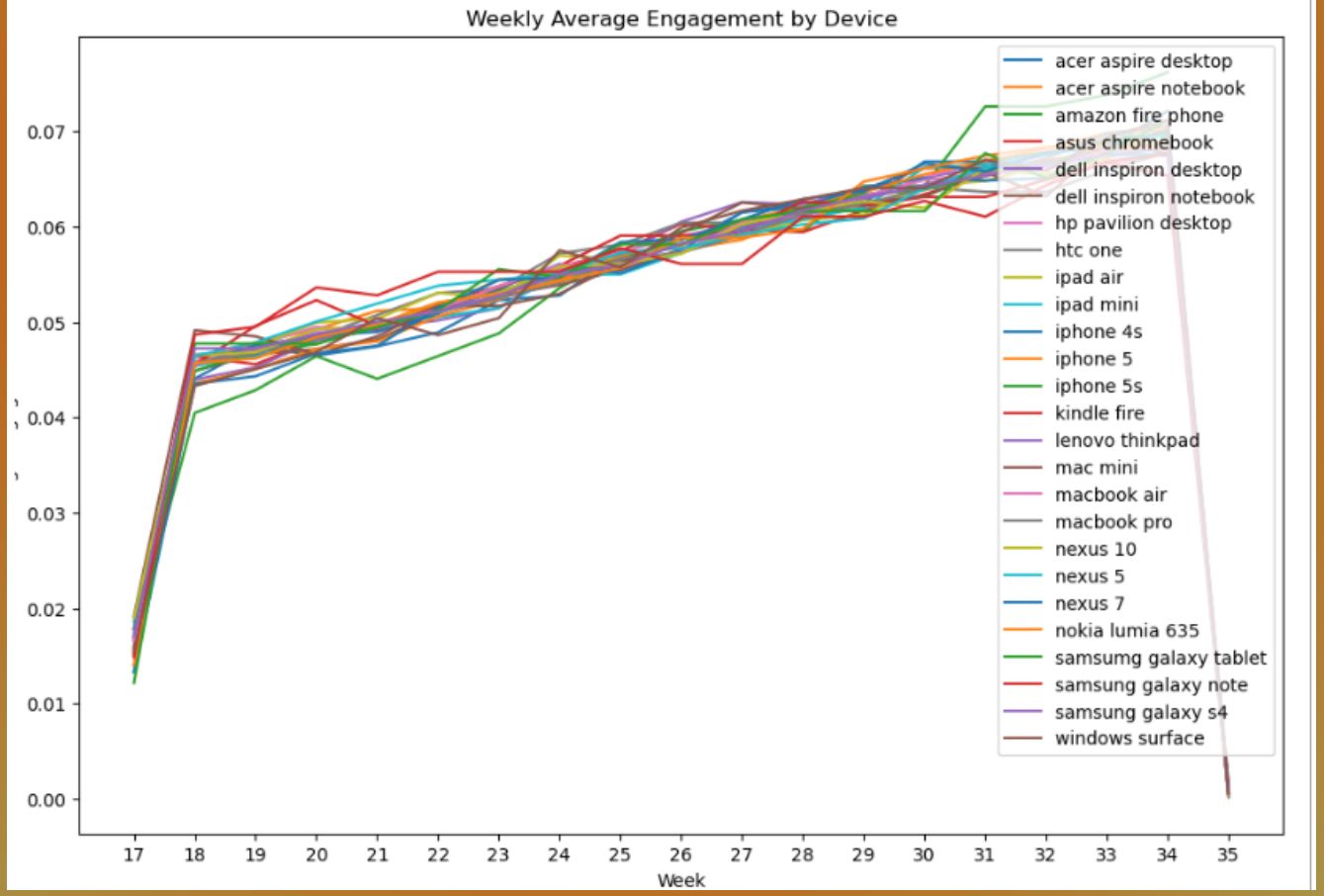
Calculate the weekly engagement per device

```
# the weekly engagement per device
merged_df = email_events_df.merge(user_df, on='user_id', how='left').merge(event_df, on='user_id', how='left')
merged_df['occurred_at_x'] = pd.to_datetime(merged_df['occurred_at_x'])
merged_df['week'] = merged_df['occurred_at_x'].apply(lambda x: x.strftime('%U'))
weekly_engagement = merged_df.groupby(['device', 'week']).agg({'user_id': pd.Series.nunique}).reset_index()
weekly_engagement['avg_engagement'] = weekly_engagement.groupby('device')['user_id'].apply(lambda x: x / x.sum())
weekly_engagement['avg_engagement']
```

```
0      0.016949
1      0.044068
2      0.047458
3      0.046489
4      0.047458
         ...
480    0.067023
481    0.065836
482    0.069395
483    0.071174
484    0.000593
Name: avg_engagement, Length: 485, dtype: float64
```

Plotting the weekly engagement per device in python

```python
import matplotlib.pyplot as plt

# Plot the line graph
fig, ax = plt.subplots(figsize=(12, 8))

for device, data in weekly_engagement.groupby('device'):
    ax.plot(data['week'], data['avg_engagement'], label=device)

# Add titles and labels
ax.set_title('Weekly Average Engagement by Device')
ax.set_xlabel('Week')
ax.set_ylabel('Average Engagement')
ax.legend()

# Display the plot
plt.show()
```



Calculate the email engagement metrics

```python
# the email engagement metrics
import pandas as pd

# read the csv file into a DataFrame
email_events_df = pd.read_csv('D:\\data ANALYTICS AND SCIENCE\\Table-3 email_events.csv')

# calculate the total number of users who engaged with the email service
total_users = len(email_events_df['user_id'].unique())

# filter the DataFrame to include only sent emails
sent_emails_df = email_events_df[email_events_df['action'] == 'sent_weekly_digest']

# calculate the total number of sent emails
total_sent_emails = len(sent_emails_df)

# filter the DataFrame to include only opened emails
opened_emails_df = email_events_df[email_events_df['action'] == 'email_open']

# calculate the total number of opened emails
total_opened_emails = len(opened_emails_df)

# filter the DataFrame to include only clicked emails
clicked_emails_df = email_events_df[email_events_df['action'] == 'email_clickthrough']

# calculate the total number of clicked emails
total_clicked_emails = len(clicked_emails_df)

# calculate the open rate
open_rate = total_opened_emails / total_sent_emails * 100

# calculate the click-through rate
click_through_rate = total_clicked_emails / total_sent_emails * 100

# print the results
print(f"Total number of users who engaged with the email service: {total_users}")
print(f"Total number of sent emails: {total_sent_emails}")
print(f"Total number of opened emails: {total_opened_emails}")
print(f"Total number of clicked emails: {total_clicked_emails}")
print(f"Open rate: {open_rate:.2f}%")
print(f"Click-through rate: {click_through_rate:.2f}%")
```

```
Total number of users who engaged with the email service: 6179
Total number of sent emails: 57267
Total number of opened emails: 20459
Total number of clicked emails: 9010
Open rate: 35.73%
Click-through rate: 15.73%
```

PROJECT 6 :Instagram User Analytics

PROBLEM STATEMENT

Provide insights on some questions for the Instagram campaign

OBJECTIVE

- Marketing
  - Find the 5 oldest users of Instagram from the database provided
  - Find the users who have never posted a single photo on Instagram
  - Find who gets the most likes on a single photo
  - Identify and suggest the top 5 most commonly used hashtags on the platform
  - What day of the week do most users register on? Provide insights on when to schedule an ad campaign
- Investor Metrics
  - how many times does average user posts on Instagram?
  - Provide data on users (bots) who have liked every single photo on the site (since any normal user would not be able to do this).

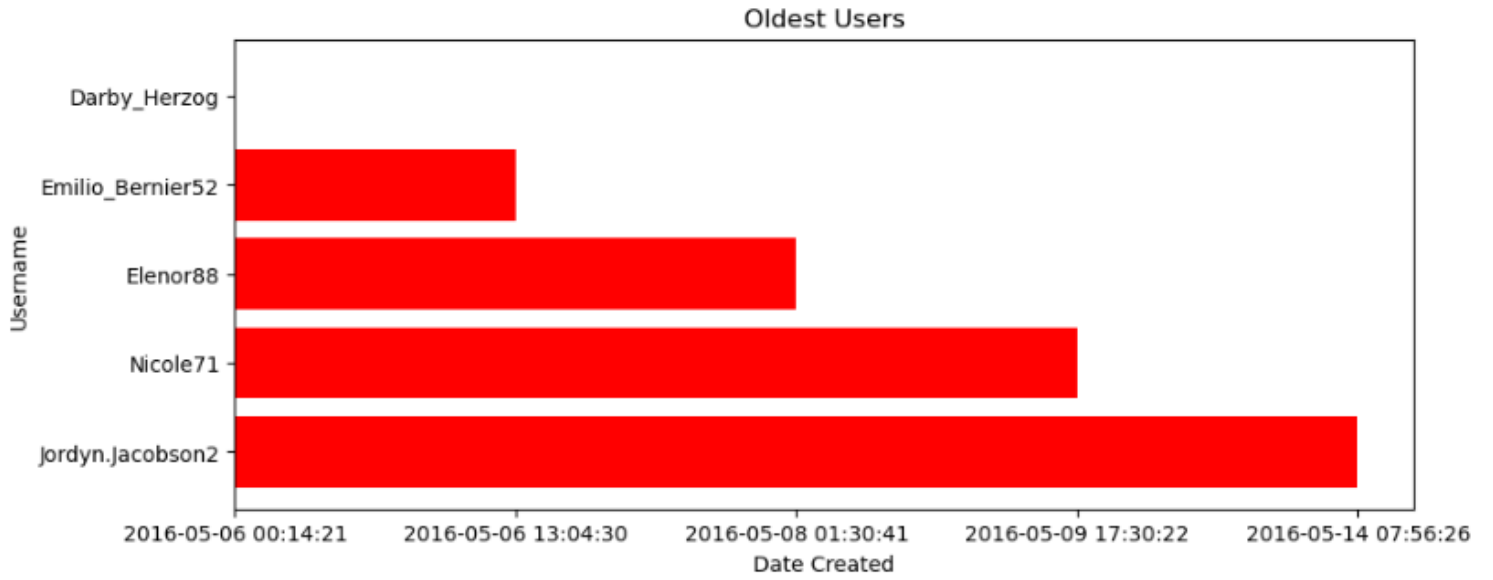
Find the 5 oldest users of Instagram from the database provided

```sql
-- 5 oldest users of the Instagram from the database
SELECT  *
from users
ORDER BY created_at ASC
LIMIT 5
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	username	created_at
80	Darby_Herzog	2016-05-06 00:14:21
67	Emilio_Bernier52	2016-05-06 13:04:30
63	Elenor88	2016-05-08 01:30:41
95	Nicole71	2016-05-09 17:30:22
38	Jordyn.Jacobson2	2016-05-14 07:56:26

```python
# Creating bar chart
import matplotlib.pyplot as plt
df['created_at'] = pd.to_datetime(df['created_at'])
df_sorted = df.sort_values(by='created_at', ascending=True).head(5)
labels = df_sorted['username']
values = df_sorted['created_at'].apply(lambda x: x.strftime('%Y-%m-%d %H:%M:%S'))
fig, ax = plt.subplots(figsize=(10, 4))
ax.barh(labels, values,color='red')
ax.invert_yaxis()
ax.set_xlabel('Date Created')
ax.set_ylabel('Username')
ax.set_title('Oldest Users')
plt.show()
```



Find the users who have never posted a single photo on Instagram

```sql
-- 2) users who have never posted a single photo on INSTAGRAM (1st way)
select users.username
from users
left join photos on users.id= photos.user_id
where photos.user_id is null;

-- 2) users who have never posted a single photo on INSTAGRAM (2nd way)
SELECT *
FROM users
WHERE id NOT IN (
  SELECT DISTINCT user_id
  FROM Photos
);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

id	username	created_at
5	Aniya_Hackett	2016-12-07 01:04:39
7	Kasandra_Homenick	2016-12-12 06:50:08
14	Jadyn81	2017-02-06 23:29:16
21	Rocio33	2017-01-23 11:51:15
24	Maxwell.Halvorson	2017-04-18 02:32:44

What day of the week do most users register on? Provide insights on when to schedule an ad campaign

```sql
-- day of the week do most users register on
SELECT DATE_FORMAT(created_at, '%W') AS registration_day,
count(*) as most_popularday
FROM users
GROUP BY registration_day
ORDER BY most_popularday DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

registration_day	most_popularday
Thursday	16
Sunday	16
Friday	15
Tuesday	14
Monday	14

Find who gets the most likes on a single photo

```sql
-- who gets the most likes on a single photo
SELECT photos.id,username,
COUNT(*) AS total_likes
FROM photos
INNER JOIN likes
ON likes.photo_id = photos.id
INNER join users
on photos.user_id=users.id
group by photos.id
ORDER BY total_likes DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	username	total_likes
145	Zack_Kemmer93	48

Identify and suggest the top 5 most commonly used hashtags on the platform

```sql
-- top 5 most commonly used hashtags on the platform
select tag_name,id,
count(*) as each_hastag_count
from tags t
left join photo_tags pt on t.id=pt.tag_id
group by tag_name
order by each_hastag_count desc
limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

tag_name	id	each_hastag_count
smile	21	59
beach	20	42
party	17	39
fun	13	38
concert	18	24

how many times does average user posts on Instagram?

User Engagement

```sql
-- how many times does average user posts on Instagram.
SELECT SUM(num_posts) AS total_num_posts, SUM(num_posts) / (SELECT COUNT(DISTINCT username) FROM users) AS avg_post
FROM (
  SELECT COUNT(*) AS num_posts
  FROM photos
  INNER JOIN users ON photos.user_id = users.id
  GROUP BY username
) AS subquery;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_num_posts	avg_post
257	2.5700

Provide data on users (bots) who have liked every single photo on the site (since any normal user would not be able to do this).

Bots & Fake Accounts

```sql
-- data on users (bots) who have liked every single photo on the site
SELECT users.id,username, COUNT(users.id) As total_likes_by_user
FROM users
JOIN likes ON users.id = likes.user_id
GROUP BY users.id
HAVING total_likes_by_user = (SELECT COUNT(*) FROM photos);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	username	total_likes_by_user
5	Aniya_Hackett	257
14	Jaclyn81	257
21	Rocio33	257
24	Maxwell.Halvorson	257
36	Ollie_Ledner37	257

ANALYSIS

- **Rewarding Most Loyal Users:** 'Darby_Herzog', 'Emilio_Bernier52', 'Elenor88', 'Nicole71', 'Jordyn.Jacobson2
- **Remind Inactive Users to Start Posting:** There are 25 inactive users
- **Declaring Contest Winner:** 'Zack_Kemmer93', with user id '145', got max like 48.
- **Hashtag Researching:** smile,beach,party,fun,concert are top 5 hastag used
- **Launch AD Campaign:** Thursday and Sunday are two days on which most registration occurred
- **User Engagement:** average post person is 2.57
- **Bots & Fake Accounts:** There are 13 bot users.

PROJECT 7 :ABC Call Volume Trend Analysis

PROBLEM STATEMENT

Analyse Inbound calls of an ABC company from the insurance category.

Data dictionary

- Agent_Name: The name of the call center agent who handled the call.
- Agent_ID: The unique identifier of the call center agent.
- Customer_Phone_No: The phone number of the customer who made the call.
- Queue_Time(Secs): The duration, in seconds, that the customer spent in the call queue before being connected to an agent.
- Date_&_Time: The date and time when the call was initiated.
- Time: The time of day when the call was initiated.
- Time_Bucket: A categorization of the call time into specific time intervals or buckets.
- Duration(hh:mm:ss): The total duration of the call in hours, minutes, and seconds.
- Call_Seconds (s): The total duration of the call in seconds.
- Call_Status: The status of the call, indicating if it was answered, missed, or any other relevant status.
- Wrapped_By: The identifier or name of the agent who wrapped up or concluded the call.
- Ringing: The duration of time the call rang before being answered.
- IVR_Duration: The duration of time the caller spent interacting with the interactive voice response (IVR) system.

OBJECTIVE

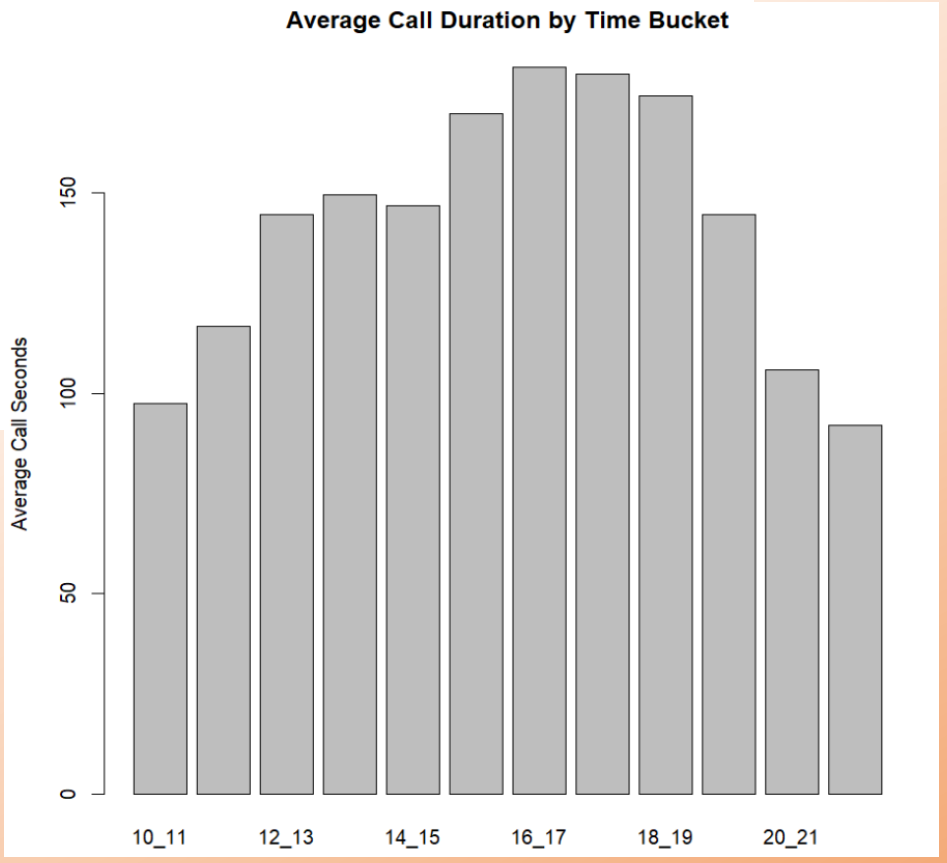
- Calculate the average call time duration for all incoming calls received by agents (in each Time_Bucket).
- Show the total volume/ number of calls coming in via charts/ graphs [Number of calls v/s Time]. You can select time in a bucket form (i.e. 1-2, 2-3, ….)
- As you can see current abandon rate is approximately 30%. Propose a manpower plan required during each time bucket [between 9am to 9pm] to reduce the abandon rate to 10%. (i.e. You have to calculate minimum number of agents required in each time bucket so that at least 90 calls should be answered out of 100.)

RESULTS

Calculate the average call time duration for all incoming calls received by agents (in each Time_Bucket).

```
# the average call time duration for all incoming calls received by agents (in each Time_Bucket).
average_time_duration <- aggregate(Call_Seconds..s. ~ Time_Bucket, data = call_centre, FUN = mean)
average_time_duration
```

Time_Bucket	Call_Seconds..s.
10_11	97.42402
11_12	116.78374
12_13	144.72502
13_14	149.54096
14_15	146.96932
15_16	169.89682
16_17	181.43935
17_18	179.72451
18_19	174.32468
19_20	144.58255
20_21	105.94914
9_10	92.01033

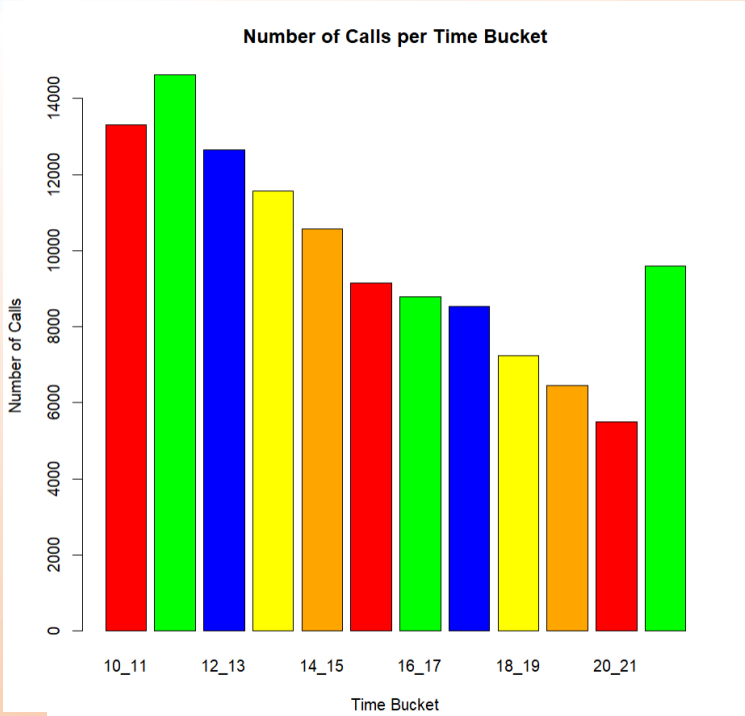


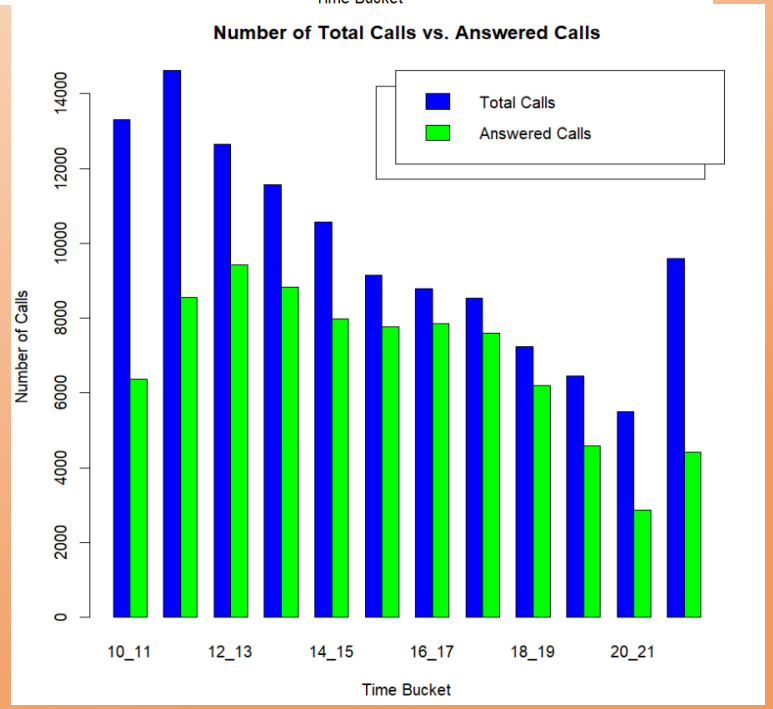

```
# Plotting  the average_time_duration in a bar graph
barplot(average_time_duration$Call_Seconds..s., names.arg = average_time_duration$Time_Bucket,
        xlab = "Time Bucket", ylab = "Average Call Seconds",
        main = "Average Call Duration by Time Bucket")
```

Show the total volume/ number of calls coming in via charts/ graphs [Number of calls v/s Time]. You can select time in a bucket form (i.e. 1-2, 2-3, ….)

```
# Create a bar graph to show the number of calls per time bucket
colors <- c("red", "green", "blue", "yellow", "orange")
barplot(calls_per_bucket, xlab = "Time Bucket", ylab = "Number of Calls",
        main = "Number of Calls per Time Bucket", col = colors)


# Create a bar plot for number of total calls vs. answered calls
barplot(height = t(call_counts[, "Call_Status"]), beside = TRUE,
        names.arg = call_counts[, "Time_Bucket"],
        xlab = "Time Bucket", ylab = "Number of Calls",
        col = c("blue", "green"),
        legend.text = c("Total Calls", "Answered Calls"),
        main = "Number of Total Calls vs. Answered Calls")

# Add a legend
legend("topright", legend = c("Total Calls", "Answered Calls"), fill = c("blue", "green"))
```

	Time_Bucket	Call_Status.total_calls	Call_Status.answered_calls
1	10_11	13313	6368
2	11_12	14626	8560
3	12_13	12652	9432
4	13_14	11561	8829
5	14_15	10561	7974
6	15_16	9159	7760
7	16_17	8788	7852
8	17_18	8534	7601
9	18_19	7238	6200
10	19_20	6463	4578
11	20_21	5505	2870
12	9_10	9588	4428





As you can see current abandon rate is approximately 30%. Propose a manpower plan required during each time bucket [between 9am to 9pm] to reduce the abandon rate to 10%. (i.e. You have to calculate minimum number of agents required in each time bucket so that at least 90 calls

```
"As you can see current abandon rate is approximately 30%.
Propose a manpower plan required during each time bucket [between 9am to 9pm] to
reduce the abandon rate to 10%. (i.e. You have to calculate minimum number of
agents required in each time bucket so that at least 90 calls should be answered out of 100.) "

# Define the current and desired abandon rates
abandon_rate_current <- 0.30
abandon_rate_desired <- 0.10
# Define the average occupancy rate of agents
occupancy_rate <- 0.60
# Calculate the total number of calls and the number of answered calls for each time bucket
call_counts <- aggregate(Call_Status ~ Time_Bucket, data = call_centre, FUN = function(x) c(total_calls = length(x), answered_calls = sum(x == "answered")))
call_counts

# Calculate the number of calls and the number of answered calls in each time bucket
total_calls <- call_counts$Call_Status[, "total_calls"]
total_calls
answered_calls <- call_counts$Call_Status[, "answered_calls"]
answered_calls
# Calculate the number of agents required in each time bucket
agents_required <- round((answered_calls / total_calls) / (1 - abandon_rate_desired) * 100)/ occupancy_rate
p=ceiling(agents_required)
# Print the minimum number of agents required in each time bucket
cat("Minimum Number of Agents Required per Time Bucket:\n")
cat("Time Bucket\t\tAgents Required\n")
cat("--------------------------------\n")
for (i in 1:length(agents_required)) {
  cat(call_counts$Time_Bucket[i], "\t\t", p[i], "\n")
}
```

10_11	89
11_12	109
12_13	139
13_14	142
14_15	140
15_16	157
16_17	165
17_18	165
18_19	159
19_20	132
20_21	97
9_10	85

Manpower required in each time bucket with 10% abandon rate

PROJECT 8 : Data Analytics Process

(Plan, Prepare, Process, Analyze, Share, Act)

CASE: Buying a birthday surprise for a friend.

PLAN

- What gifts to buy?
- Where I can buy the gift from?
- When to buy?

I have to decide what suitable gifts I can buy, where I can purchase them from, and when I can buy i.e. weekdays or weekends if it is offline. If the birthday is within 2 or 3 days I have to buy it immediately.

PREPARE

- What is the price range of my gift?
- Decide for online or offline.
- How to manage the cost of the gift?

I will fix the price range through the money I can afford, then I will check when is the birthday. If I don't have time, i.e., if the birthday is within 1 or 2 days then I will skip online buying and go offline. If the birthday is not very soon I can go for offline buying on weekends or weekdays. I will decide the price and range and spend it from my savings. I will start searching for what to buy.

Process

- Check prices online first(considering birthday is away by 1 week)
- Check prices offline (considering birthday is after 1 or 2 days)
- Check prices offline (considering birthday is away by 1 week)

After thinking about my friend's choice I decided to buy a shirt and a pair of trousers since he like dresses very much. Now I will look for various brands and collect data after wish listing during offline buying or comparing various brands' prices during online buying (after fixing from which store to buy.)

Analyse

- Which brand does my friend like the most?
- Which colour shirt and trouser he likes?
- What size fit him most?
- Whether the cost is within my prefixed range or not?
- Any coupons or offers available

I will select 2 , or 3 brands like Us polo, Ucb, Pepe jeans ,etc. now select color and size which one I can buy best and whether it is from a recent collection or not. Then I will compare the price and finally, I will try to get a coupon or credit card discount if I buy online or to get some other means of discount if it is offline buying.I will compare price from different brands.

Share

I can select a few products which suit my price range and ask the retailer to comment on the products for any additional feedback on that product if it is offline and if it is online I will check the reviews. I can also check the best fit according to the size my friend needs through the retailer or AI of an online app.

Act

- The last step is to buy the product after I am fully satisfied and present the gift to my friend.

- ✓ The analysis of 8 projects was successfully carried out and all insights were data driven.
- ✓ The used software for the projects are:
  - PROJECT 1: AUTOMOBILE PROJECT: **EXCEL AND PYTHON**
  - PROJECT 2: BANK LOAN PROJECT: **PYTHON**
  - PROJECT 3: IMDB MOVIE ANALYSIS**: R STUDIO AND POWER BI**
  - PROJECT 4: HIRING PROCESS: S ANALYTICS: **EXCEL**
  - PROJECT 5: OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE:
  - **SQL SERVER AND PYTHON**
  - PROJECT 6: INSTAGRAM USER ANALYTICS: **MY SQL**
  - PROJECT 7**:** ABC CALL VOLUME TREND ANALYSIS: **R STUDIO**
  - PROJECT 8: DATA ANALYTICS PROCESS: **POWERPOINT**

CONCLUSION

AUTOMOBILE: The effect of several features like mileage, horsepower, drive wheel can help manufacturers predict the accurate price, which is shown in the analysis. It will help in profitability if manufacturing cost, operational cost, etc., is provided and hence the maker can take data-driven decisions.

BANK LOAN: Studying customer behavior like credit capacity, age, income, job will give the idea of loan default rate to banks. The analysis is carried out to show the relations.

IMDB MOVIE ANALYSIS: Stakeholders will gain complete data-driven insights from the exploratory data analysis that is performed here.

HIRING PROCESS: S ANALYTICS: The analysis will help in maintaining the employee information and help in future hiring processes. It gave insights into the hiring process overall like male-female number , people in each department, and their count , etc.

OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE: Metrics like user engagement, email engagement metrics, weekly engagement per device, user growth for product, etc. is found out in the analysis. Metric spikes can indicate unexpected changes or issues in a system. They can help identify performance bottlenecks, errors, or unusual behaviour that may require investigation and troubleshooting.

INSTAGRAM USER ANALYTICS: User's engagement and interaction with Instagram will give business insights for marketing, product & development teams. These insights are then used by teams across the business to launch a new marketing campaign, decide on features to build for an app, track the success of the app by measuring user engagement and improve the experience altogether while helping the business grow. A complete analysis was carried out, showing various marketing and user engagement analysis.

ABC CALL VOLUME TREND ANALYSIS: Analysis for a call center data was shown which can help in manpower prediction for future and help in customer retention by answering their problems through availability of inbound calls.