```
# Importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


amz = pd.read_csv('Amazon - Movies and TV Ratings.csv')
print(amz.head())

#checking the shape of the data sets
print(amz.shape)

print(amz.describe().T)

#ploting the mean frequency
print()
amz.describe().T['mean'].plot(bins=25, kind='hist', color = 'indianred')
plt.show()
#ploting the count frequency
print()
amz.describe().T['count'].plot(bins=25, kind='hist', color = 'blue')
plt.show()

# Movie that has maxmium ratings
print()
print(amz.drop('user_id',axis=1).sum().sort_values(ascending=False)[:1].to_frame(
))

# Average rating of each movie
print()
print(amz.drop('user_id',axis=1).mean())

#top 5 movies with the maximum rating
print()
print(amz.drop('user_id',axis=1).mean().sort_values(ascending=False)[0:5].to_fram
e())

# Top 5 movies with least audience
print(amz.describe().T['count'].sort_values(ascending=True)[:5].to_frame())

#importing libiraies for model building
from surprise import Reader
from surprise import Dataset
from surprise import accuracy
```

```
from surprise import SVD
from surprise.model_selection import train_test_split

movie_data = amz.melt(id_vars =
amz.columns[0],value_vars=amz.columns[1:],var_name="Movies",value_name="Rating")
print(movie_data)

#creating a dataset for training and testing
rd = Reader(rating_scale=(-1,10))
data = Dataset.load_from_df(movie_data.fillna(0),reader=rd)
print(data)

train_data,test_data = train_test_split(data,test_size=0.20)

#Using SVD (Singular Value Descomposition)
svd = SVD()

svd.fit(train_data)

pred = svd.test(test_data)

print(accuracy.rmse(pred))

print(accuracy.mae(pred))

u_id='AH3QC2PC1VTGP'
mv = 'Movie206'
r_id = 5.0
print(svd.predict(u_id, mv, r_ui=r_id, verbose= True))

from surprise.model_selection import cross_validate

print(cross_validate(svd, data, measures = ['RMSE', 'MAE'], cv = 3, verbose =
True))

def repeat(ml_type,dframe,min_,max_):
    rd = Reader()
    data = Dataset.load_from_df(dframe,reader=rd)
    print(cross_validate(ml_type, data, measures = ['RMSE', 'MAE'], cv = 3,
verbose = True))
    print("#"*10)
    u_id = 'AH3QC2PC1VTGP'
    m_id = 'Movie206'
    ra_u = 5.0
    print(ml_type.predict(u_id,mv,r_ui=ra_u,verbose=True))
```

```
    print("#"*10)
    print()

amz= amz.iloc[:3000, :50]
movie_data = amz.melt(id_vars =
amz.columns[0],value_vars=amz.columns[1:],var_name="Movies",value_name="Rating")

repeat(SVD(),movie_data.fillna(0),-1,10)
repeat(SVD(),movie_data.fillna(movie_data.mean()),-1,10)
repeat(SVD(),movie_data.fillna(movie_data.median()),-1,10)

#trying grid search and find optimum hyperparameter value for n_factors
from surprise.model_selection import GridSearchCV

param_grid = {'n_epochs':[20,30], 'lr_all':[0.005,0.001], 'n_factors':[50,100]}

gs = GridSearchCV(SVD,param_grid,measures=['rmse','mae'],cv=3)
gs.fit(data)

print(gs.best_score)

print(gs.best_score["rmse"])
print(gs.best_params["rmse"])
```