Python Lab-10

Name- Jatin Karthik

Matriculation no.-313301

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
import seaborn as sns
from sklearn.decomposition import NMF
import math as Math
```

In [2]:
```python
All_data=pd.read_csv(r"D:\Hildesheim Universitaet\Semester 1\ML LAB\Exercise_10\ml-100k\ml-100k\u.data",sep=r"\s+",
                    names=["userId","itemId","rating","timestamp"])
All_data
```

Out[2]:

	userId	itemId	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
...	...	...	...	...
99995	880	476	3	880175444
99996	716	204	5	879795543
99997	276	1090	1	874795795
99998	13	225	2	882399156
99999	12	203	3	879959583

100000 rows × 4 columns

In [3]:
```python
user_data=pd.read_csv(r"D:\Hildesheim Universitaet\Semester 1\ML LAB\Exercise_10\ml-100k\ml-100k\u.user",
                      names=["userId","age","gender","occupation","zipcode"],sep="|")
#userdata=userdata.replace(['F','M'],[1,0])
user_data
```

Out[3]:

	userId	age	gender	occupation	zipcode
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213
...	...	...	...	...	...
938	939	26	F	student	33319
939	940	32	M	administrator	02215
940	941	20	M	student	97229
941	942	48	F	librarian	78209
942	943	22	M	student	77841

943 rows × 5 columns

In [4]:
```python
item_data=pd.read_csv(r"D:\Hildesheim Universitaet\Semester 1\ML LAB\Exercise_10\ml-100k\ml-100k\u.item",names=["movieid", "movietitle","releasedate","videoreleasedate",
               "IMDbURL","unknown","Action","Adventure", "Animation",
               "Children's","Comedy", "Crime", "Documentary","Drama","Fantasy",
               "Film-Noir","Horror","Musical", "Mystery","Romance","Sci-Fi",
               "Thriller", "War", "Western"],sep="|",encoding="latin-1")
item_data.head()
```

Out[4]:

	movieid	movietitle	releasedate	videoreleasedate	IMDbURL	unknown	Action	Adventure	Animation	Children's	...	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	0	0	0	0	0	0	0	0	0
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	...	0	0	0	0	0	0	0	1	0	0
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0

5 rows × 24 columns

In [5]:
```python
print(All_data.info())
print(user_data.info())
print(item_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userId     100000 non-null  int64
 1   itemId     100000 non-null  int64
 2   rating     100000 non-null  int64
 3   timestamp  100000 non-null  int64
dtypes: int64(4)
memory usage: 3.1 MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 943 entries, 0 to 942
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   userId      943 non-null    int64
 1   age         943 non-null    int64
 2   gender      943 non-null    object
 3   occupation  943 non-null    object
 4   zipcode     943 non-null    object
dtypes: int64(2), object(3)
memory usage: 37.0+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1682 entries, 0 to 1681
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   movieid           1682 non-null   int64
 1   movietitle        1682 non-null   object
 2   releasedate       1681 non-null   object
 3   videoreleasedate  0 non-null      float64
 4   IMDbURL           1679 non-null   object
 5   unknown           1682 non-null   int64
 6   Action            1682 non-null   int64
 7   Adventure         1682 non-null   int64
 8   Animation         1682 non-null   int64
 9   Children's        1682 non-null   int64
 10  Comedy            1682 non-null   int64
 11  Crime             1682 non-null   int64
 12  Documentary       1682 non-null   int64
 13  Drama             1682 non-null   int64
 14  Fantasy           1682 non-null   int64
 15  Film-Noir         1682 non-null   int64
 16  Horror            1682 non-null   int64
 17  Musical           1682 non-null   int64
 18  Mystery           1682 non-null   int64
 19  Romance           1682 non-null   int64
 20  Sci-Fi            1682 non-null   int64
 21  Thriller          1682 non-null   int64
 22  War               1682 non-null   int64
 23  Western           1682 non-null   int64
dtypes: float64(1), int64(20), object(3)
memory usage: 315.5+ KB
None
```

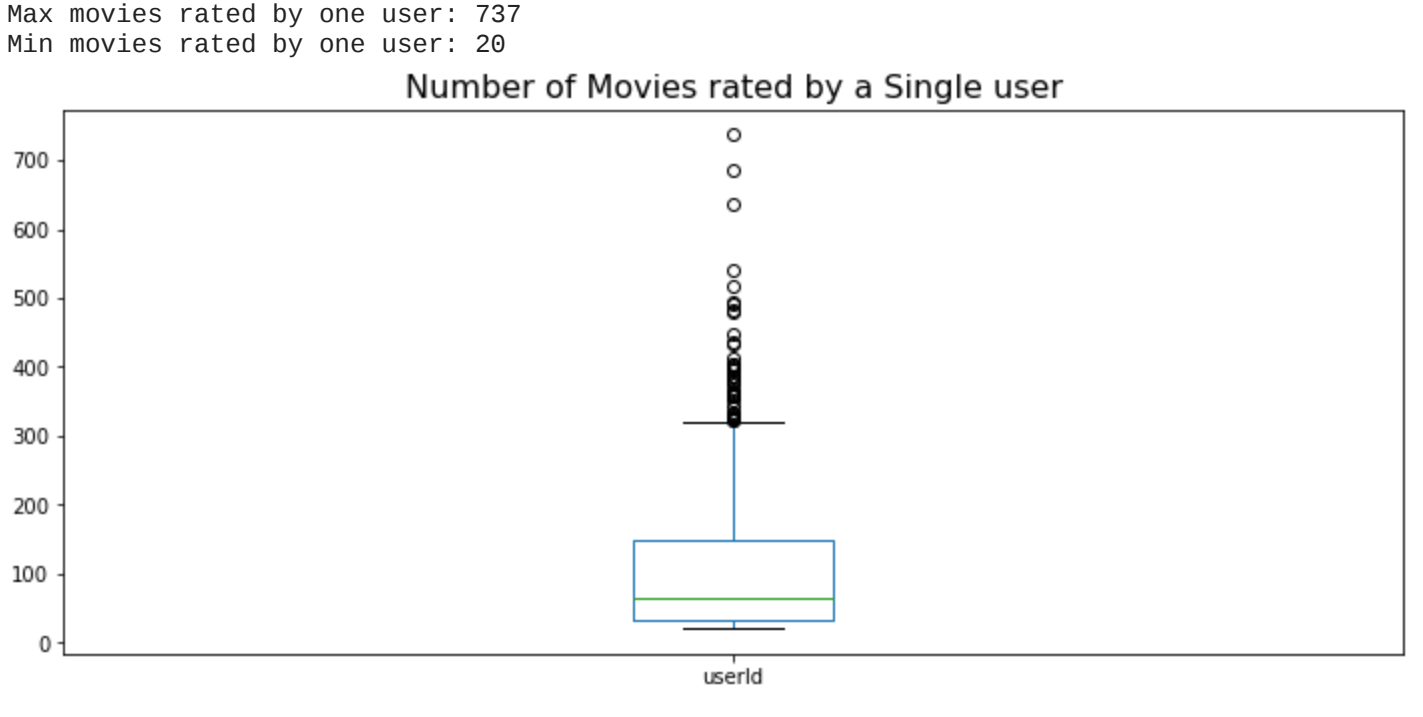
1) Showcase how the ratings vary across users, as an example consider whether the plot is able to tell if most ratings are only from a handful of users.

The below graph shows where the median or mean of the most ratings were by users. I also printed the maximum movies rated by one user and the minimum one.

In [6]:
```python
merge1=pd.merge(All_data, user_data, on='userId', how='left')
```

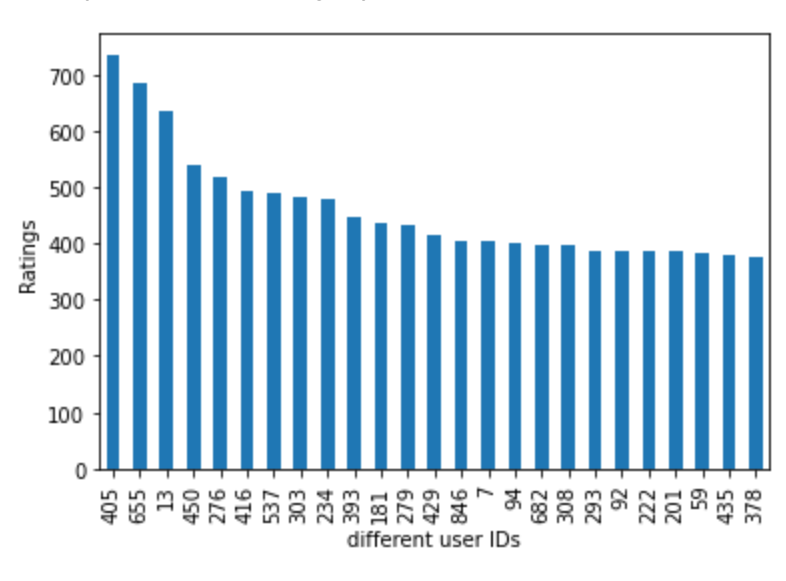
In [7]:
```python
movies_rated = All_data.groupby("userId").size().sort_values(ascending=False)
print(f"Max movies rated by one user: {max(movies_rated)}\nMin movies rated by one user: {min(movies_rated)}")

All_data.userId.value_counts().plot.box(figsize=(12, 5))
plt.title("Number of Movies rated by a Single user", fontsize=16)
plt.show()
```

```
Max movies rated by one user: 737
Min movies rated by one user: 20
```



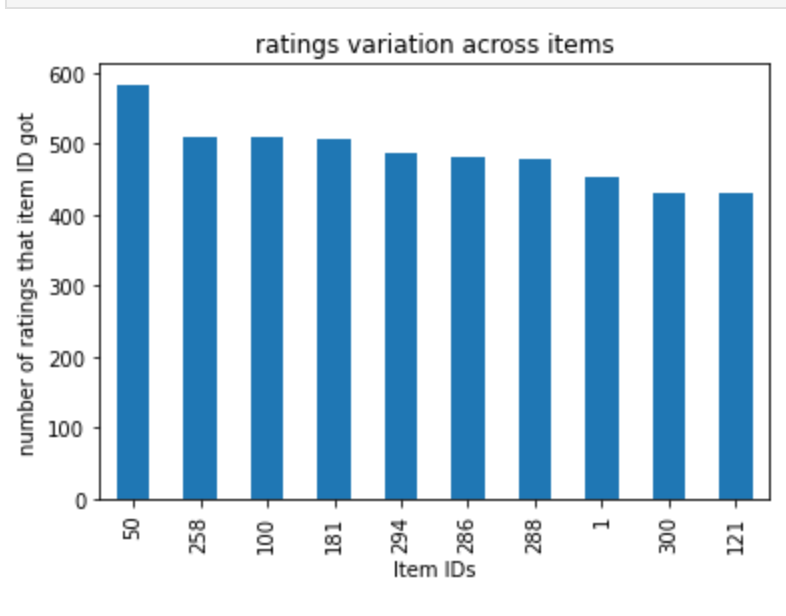
In [8]:
```python
#showing only the top 10 most users who rated
ID_plot=movies_rated[0:25]
ID_plot.plot(kind='bar')
plt.xlabel('different user IDs')
plt.ylabel('Ratings')
```

Out[8]: `Text(0, 0.5, 'Ratings')`

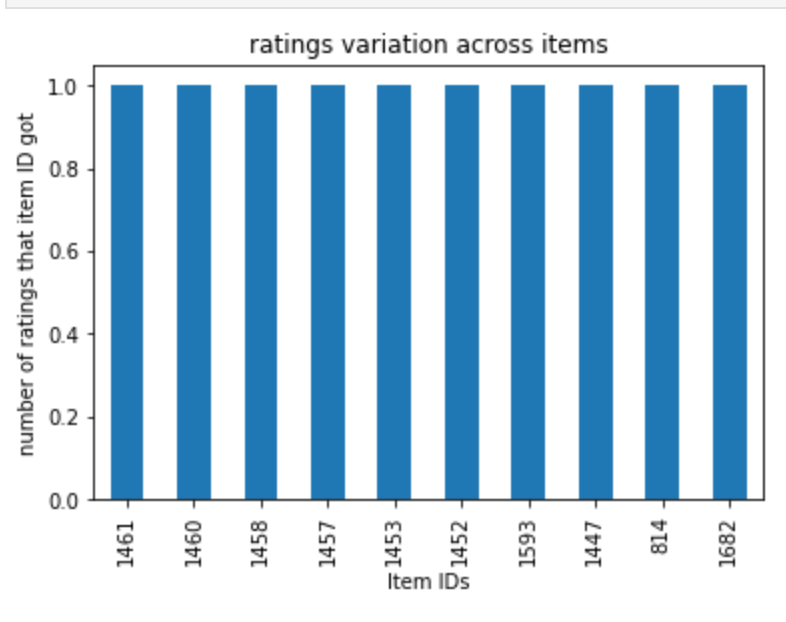


2) Showcase how the ratings vary across items.

In [22]:
```python
item_list=All_data["itemId"].value_counts()
items=All_data.groupby('itemId').size().sort_values(ascending=False)
item_list10=items[0:10]

# I show only the top 10 most highly rated items here
b_plot=item_list10
b_plot.plot(kind='bar')
plt.title("ratings variation across items")
plt.xlabel('Item IDs')
plt.ylabel('number of ratings that item ID got')
plt.show()
```



In [23]:
```python
item_list_last10=items[-10:]

#I show only the least 10 most rated items here which is all 1
b_plot=item_list_last10
b_plot.plot(kind='bar')
plt.title("ratings variation across items")
plt.xlabel('Item IDs')
plt.ylabel('number of ratings that item ID got')
plt.show()
```

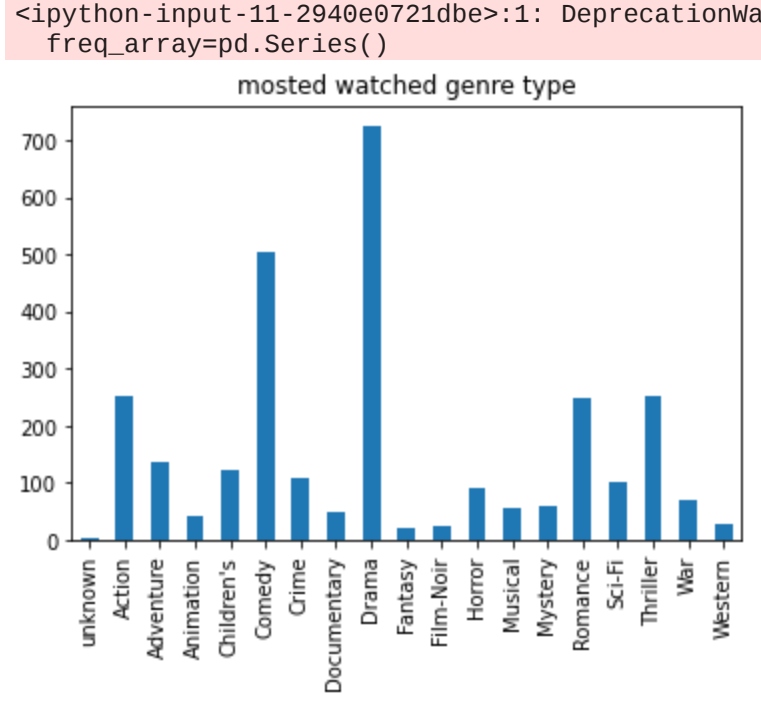


3) Are there genres that are more highly rated than others?

Yes, drama is the most highly rated genre

In [11]:
```python
freq_array=pd.Series()
for col in item_data.columns[5:]:
    freq_array[col]=sum(item_data[col].values)
bar_plot=freq_array
bar_plot.plot(kind='bar')
plt.title("mosted watched genre type")
plt.show()
```

```
<ipython-input-11-2940e0721dbe>:1: DeprecationWarning: The default dtype for empty Series will be 'object' instead of 'float64' in a future version. Specify a dtype explicitly to silence this warning.
  freq_array=pd.Series()
```



4) What age groups prefer what genres based on ratings? You can bin respective ages to your preference

In [ ]:

In [ ]:

In [12]:
```python
merge1.sort_values(by='age')
```

Out[12]:

	userId	itemId	rating	timestamp	age	gender	occupation	zipcode
20780	30	242	5	885941156	7	M	student	55436
37707	30	873	1	875061066	7	M	student	55436
50845	30	313	5	885941156	7	M	student	55436
16651	30	231	2	875061066	7	M	student	55436
3845	30	181	4	875060217	7	M	student	55436
...	...	...	...	...	...	...	...	...
30870	481	514	4	885829045	73	M	retired	37771
28980	481	524	5	885829045	73	M	retired	37771
79034	481	199	5	885828543	73	M	retired	37771
61198	481	70	5	885828389	73	M	retired	37771
45783	481	163	4	885828389	73	M	retired	37771

100000 rows × 8 columns

In [13]:
```python
print(sorted(merge1['age'].unique()))
```

```
[7, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 6
8, 69, 70, 73]
```

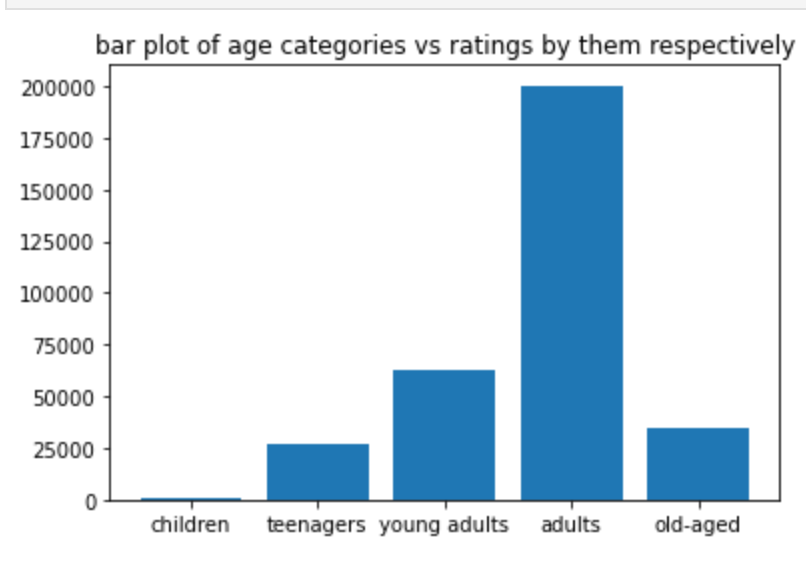
My age group preferenc eis as follows:- "children"--0-12 yrs, "teenagers"--13-19 yrs, "young adults"--20-25 yrs, "adults"-- 26-50 yrs,"old-aged"--50-74 yrs

In [14]:
```python
bins = pd.IntervalIndex.from_tuples([(0, 12), (13, 19), (20, 25), (26,50),(50,74)])
temp1=merge1.groupby(pd.cut(merge1["age"],bins, labels=["children", "teenagers", "young adults", "adults","old-aged"])).sum()
temp1
```

Out[14]:

age	userId	itemId	rating	timestamp	age
(0, 12]	23694	37989	346	89104808397	908
(13, 19]	3710022	3534800	26611	6737172558577	136968
(20, 25]	8824618	7863350	62768	16109434303043	422336
(26, 50]	26255045	23402810	200431	49890840449957	2024878
(50, 74]	3953840	4061819	34839	8379476974700	532576

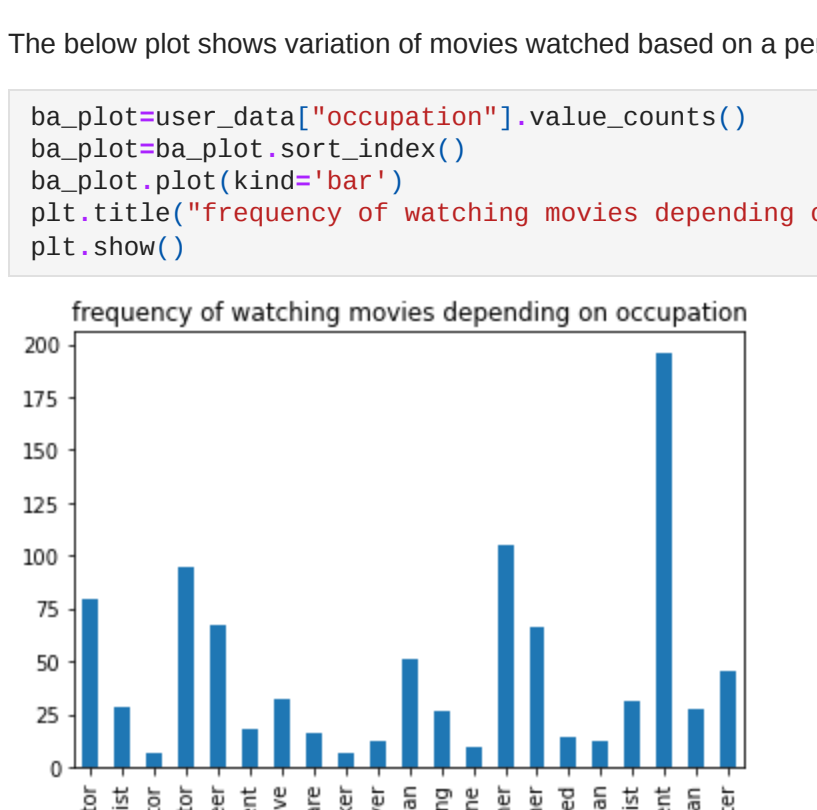
In [15]:
```python
categories={'children':346, 'teenagers':26611, 'young adults':62768, 'adults':200431, 'old-aged':34839}
names = list(categories.keys())
values = list(categories.values())

plt.bar(range(len(categories)), values, tick_label=names)
plt.title('bar plot of age categories vs ratings by them respectively')
plt.show()
```



In [ ]:

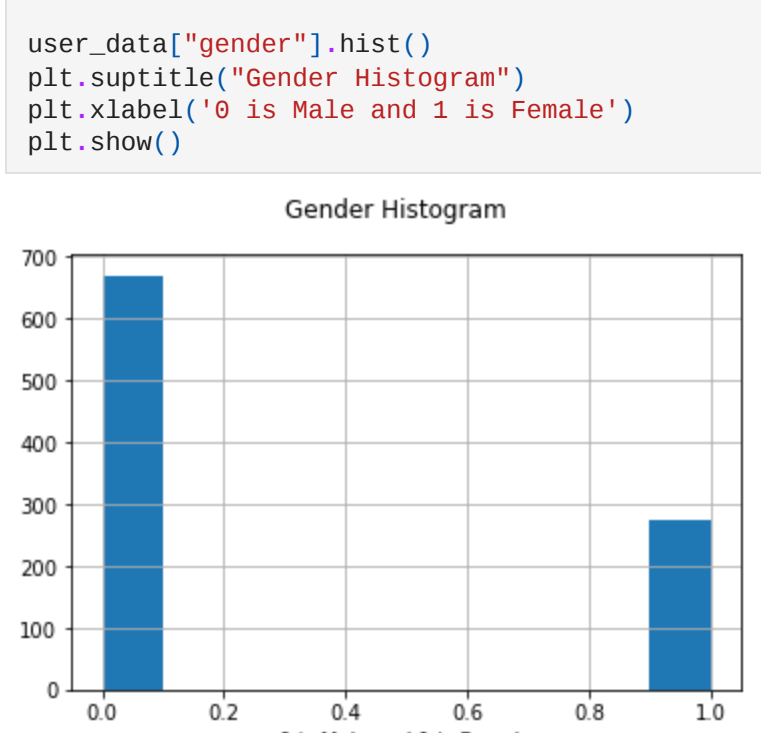
I have displayed just a few more graphs to more explore the dataset. This graph shows which numbered rating were used the mos. The bar plot shows that 4.0 was the most used rating.

In [16]:
```python
All_data["rating"].hist()
plt.suptitle("Rating Histogram")
plt.xlabel('ratings out of 5')
plt.ylabel('number of such ratings')
plt.show()
```



The below plot shows variation of movies watched based on a person's occupation. We see that students watch the most movies

In [17]:
```python
ba_plot=user_data["occupation"].value_counts()
ba_plot=ba_plot.sort_index()
ba_plot.plot(kind='bar')
plt.title("frequency of watching movies depending on occupation")
plt.show()
```



The below graph shows how the gender distribution stands wrt movie watching

In [18]:
```python
user_data=user_data.replace(['F','M'],[1,0])
user_data.head()

user_data["gender"].hist()
plt.suptitle("Gender Histogram")
plt.xlabel('0 is Male and 1 is Female')
plt.show()
```


In [ ]: