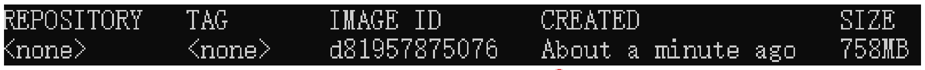
6220 Assignment1

Haoping Lin

January 2023

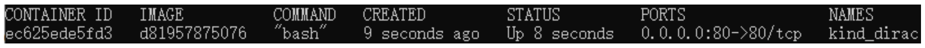
1 Question1

1. Create image d81957875076



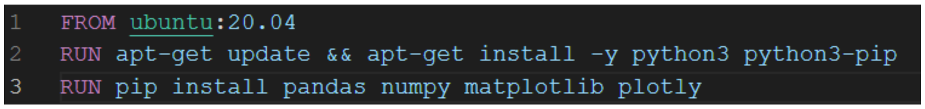
```
REPOSITORY   TAG      IMAGE ID       CREATED              SIZE
<none>       <none>   d81957875076   About a minute ago   758MB
```

Run container ec625ede5fd3 with above image



```
CONTAINER ID   IMAGE          COMMAND   CREATED         STATUS         PORTS                NAMES
ec625ede5fd3   d81957875076   "bash"    9 seconds ago   Up 8 seconds   0.0.0.0:80->80/tcp   kind_dirac
```

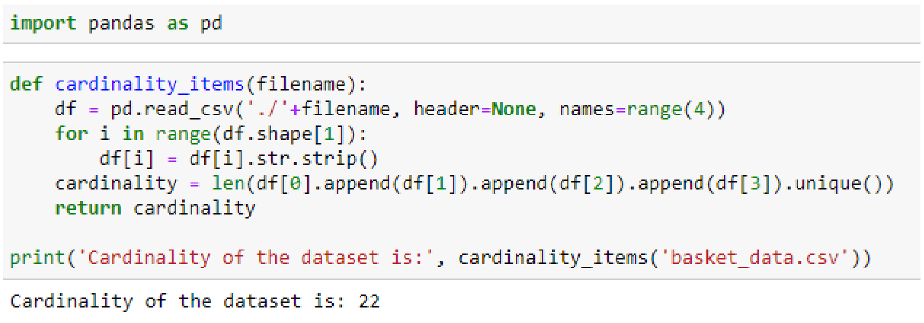
Dockerfile used to generate docker image



```
FROM ubuntu:20.04
RUN apt-get update && apt-get install -y python3 python3-pip
RUN pip install pandas numpy matplotlib plotly
```

2 Question2

1. Cardinality of the full set of unique items is 22.



```python
import pandas as pd
```

```python
def cardinality_items(filename):
    df = pd.read_csv('./'+filename, header=None, names=range(4))
    for i in range(df.shape[1]):
        df[i] = df[i].str.strip()
    cardinality = len(df[0].append(df[1]).append(df[2]).append(df[3]).unique())
    return cardinality

print('Cardinality of the dataset is:', cardinality_items('basket_data.csv'))
```

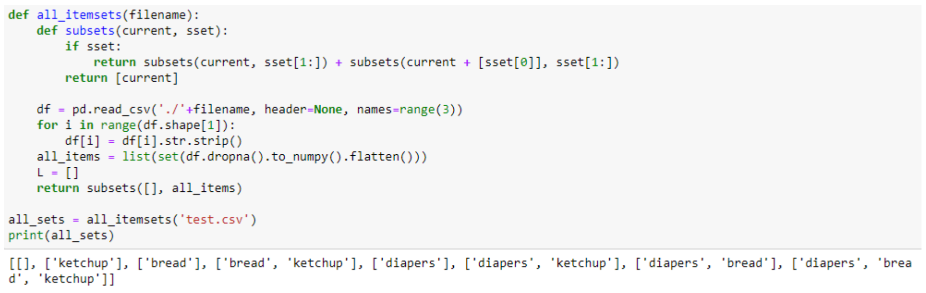
```
Cardinality of the dataset is: 22
```

Based on observation of data, there are some preprocessing needed before finding cardinality, that is some items have an extra space between the names. For example, 'butter', becomes ' butter'. This would result wrong result for this problem. So when finding cardinality, I removed the extra space to make sure ' butter' is 'butter'.

2. To find number of all possible subsets containing unique items, we can simply compute 2^N as result. This is because for each item, we have two choices, pick or not pick. And for N items, we have $\underbrace{2 * 2 * 2 * \cdots * 2}_{N}$ possible sets. If null set is ignored, simply minus one, which is $2^N - 1$.

3. **Notice: For subproblem 3 and 4, I manually created another data file called 'test.csv' with smaller cardinality of items.**

To find all possible subsets, I used recursion function to do so. Each item can be picked or not picked in each round. With all sets, we get a list of all possible subsets.



```
def all_itemsets(filename):
    def subsets(current, sset):
        if sset:
            return subsets(current, sset[1:]) + subsets(current + [sset[0]], sset[1:])
        return [current]

    df = pd.read_csv('./'+filename, header=None, names=range(3))
    for i in range(df.shape[1]):
        df[i] = df[i].str.strip()
    all_items = list(set(df.dropna().to_numpy().flatten()))
    L = []
    return subsets([], all_items)

all_sets = all_itemsets('test.csv')
print(all_sets)
```

```
[[], ['ketchup'], ['bread'], ['bread', 'ketchup'], ['diapers'], ['diapers', 'ketchup'], ['diapers', 'bread'], ['diapers', 'brea
d', 'ketchup']]
```

4. For this problem, I loop through the dataset D with given specific set S, and count how many times S shows up in D. By dividing number of records from number of S shows up, we get the probability that S occurs. In my test case, the probability is 0.1667.

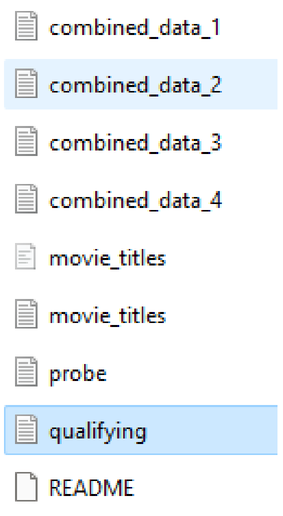
```
def prob_S(S, D):
    row_sets = []
    hit = 0
    row_num = df.shape[0]
    for index, row in df.iterrows():
        r = set(row.dropna())
        row_sets.append(r)
        if r == S:
            hit += 1
    return hit / row_num

df = pd.read_csv('./test.csv', header=None, names=range(3))
for i in range(df.shape[1]):
        df[i] = df[i].str.strip()
S = {'bread'}
print(prob_S(S, all_sets))
```

```
0.16666666666666666
```

3 Question3

3.1

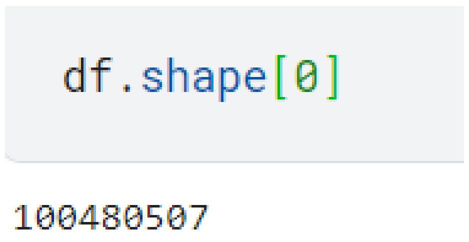
1. There is one more file called movie_titles.txt. This file is used for data formatting for subproblem 3 and 4.



combined_data_1
combined_data_2
combined_data_3
combined_data_4
movie_titles
movie_titles
probe
qualifying
README

3.2

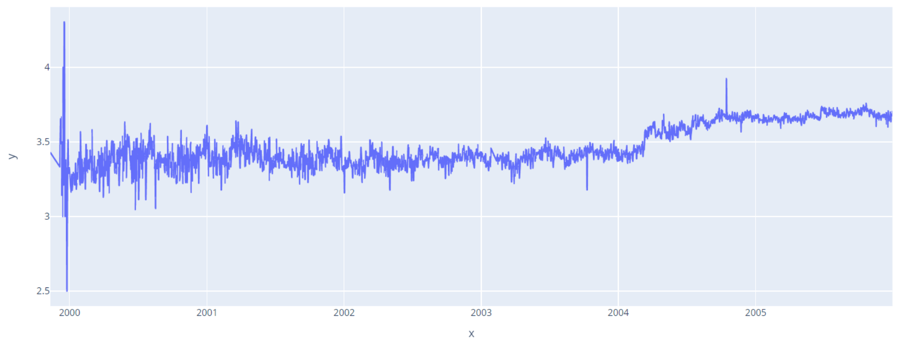
1. There are a total of 100480507 records in dataset. We can get number of rows in dataset using shape.



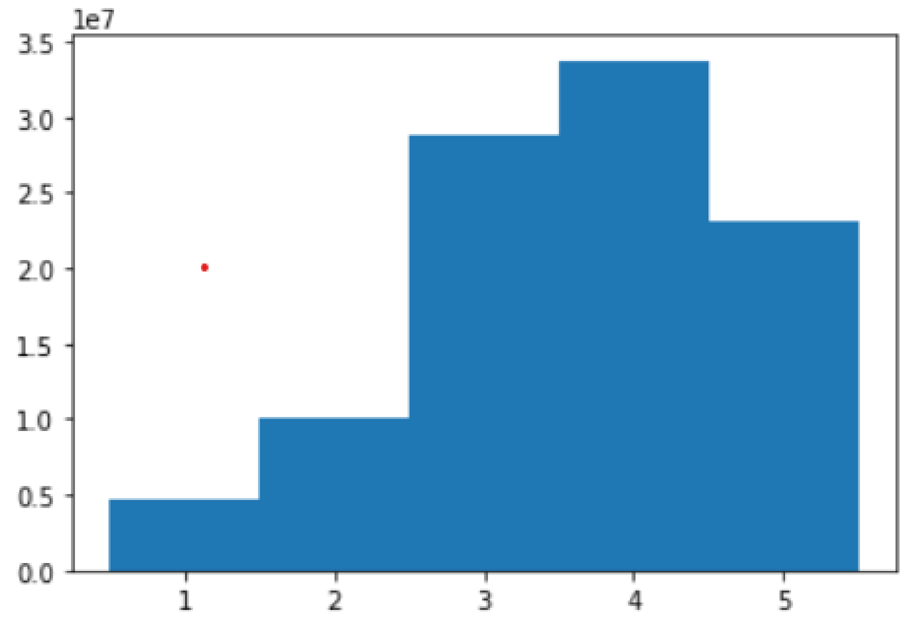
```
df.shape[0]
```

```
100480507
```

2. Below is the figure of movie rating over time. I first find mean of ratings of each year. And then plot the means with rating. From the graph, we can see that ratings around year 2000 are in huge fluctuation. Then, ratings tends to slowly increase over year.

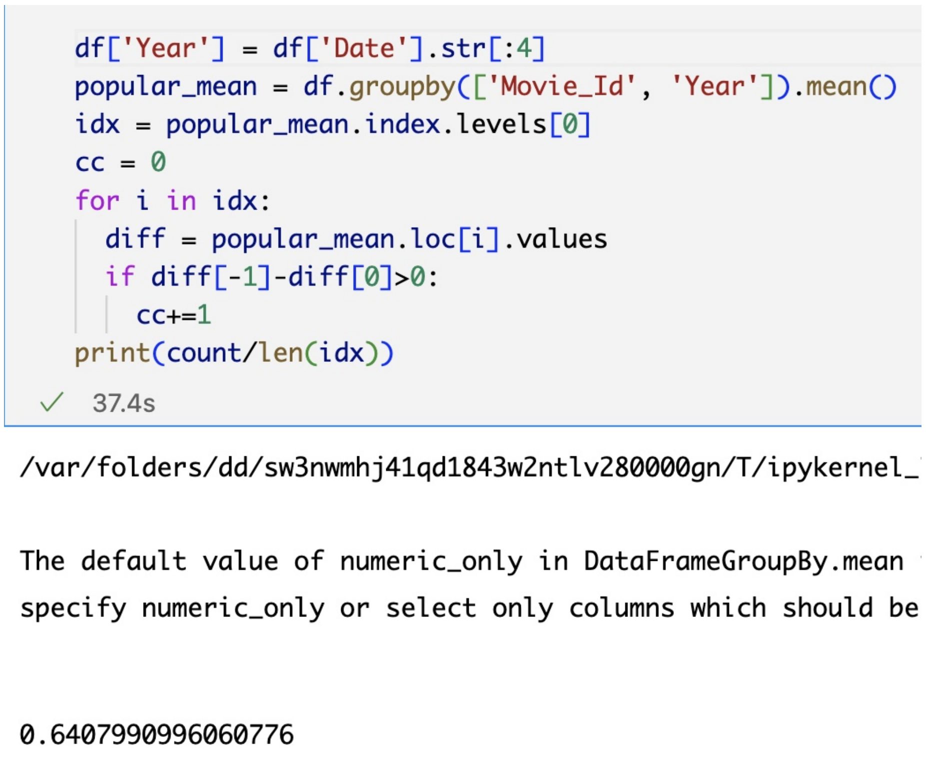


Below is the figure of rating distribution. From the graph, we can see that most ratings are above 4 stars. But there are still $5 * 10^7$ records with one star.



For this problem, it is hard to get rating over users. This is because users in the data file are represented as UserId, and there are huge amount of user in this file. If it more possible to get distribution of rating over user with a smaller size of data.

3. About 64% of movies have gotten more popular over time. I first create a new column 'Year', which conceal specific month and date in 'Date'. This operation will provide convenience in following steps. Then I find mean value of ratings based on each movie and each year. I compared ratings between last year rating and first year rating for each film, and if ratings become better, then it is more popular. Otherwise, movies become less popular. At last, count the number of more popular movies and divided by total number of films, we get percentage of films become popular over time.

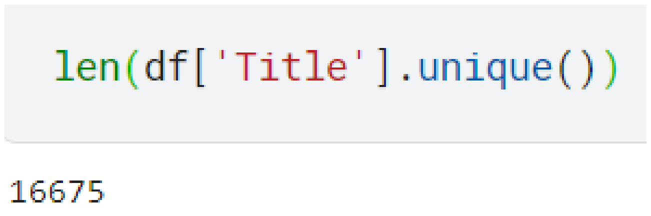


```
df['Year'] = df['Date'].str[:4]
popular_mean = df.groupby(['Movie_Id', 'Year']).mean()
idx = popular_mean.index.levels[0]
cc = 0
for i in idx:
    diff = popular_mean.loc[i].values
    if diff[-1]-diff[0]>0:
        cc+=1
print(count/len(idx))
```

✓ 37.4s

```
/var/folders/dd/sw3nwmhj41qd1843w2ntlv280000gn/T/ipykernel_

The default value of numeric_only in DataFrameGroupBy.mean
specify numeric_only or select only columns which should be


0.6407990996060776
```

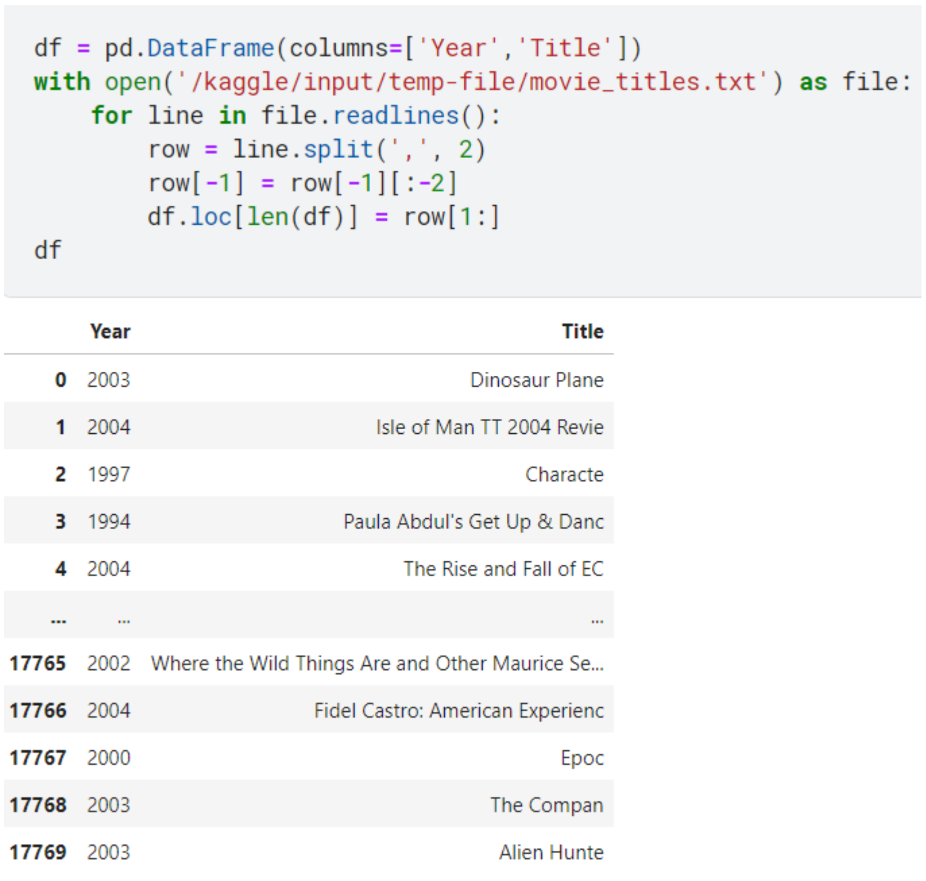
4. There are 16675 movies have been re-released. I find number of re-released movies by finding duplicate movie titles in this data file. If a movie title shows up twice or more times, then it is a re-released movie. Noticed that some movie titles are NULL, which means that NULL title also counts as a unique value. So, to be more precise, we remove NULL and the result becomes 16674. Below is the result figure.



```
len(df['Title'].unique())
```

```
16675
```

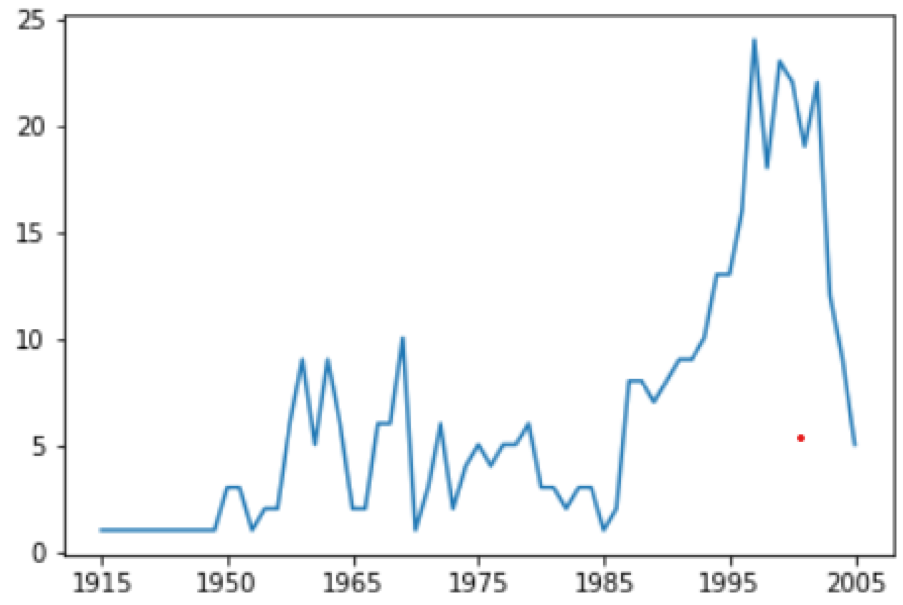
Below is the way I pre-processed the dataframe. A .csv file will use comma to split content in each row. However, I found that some movie titles contains comma as well, which resulted wrong format in reading data. Therefore, I convert the .csv file to .txt file, and read file with python. Along reading file line by line, I reformat the data to make it easy to use.



```python
df = pd.DataFrame(columns=['Year','Title'])
with open('/kaggle/input/temp-file/movie_titles.txt') as file:
    for line in file.readlines():
        row = line.split(',', 2)
        row[-1] = row[-1][:-2]
        df.loc[len(df)] = row[1:]
df
```

	Year	Title
0	2003	Dinosaur Plane
1	2004	Isle of Man TT 2004 Revie
2	1997	Characte
3	1994	Paula Abdul's Get Up & Danc
4	2004	The Rise and Fall of EC
...	...	...
17765	2002	Where the Wild Things Are and Other Maurice Se...
17766	2004	Fidel Castro: American Experienc
17767	2000	Epoc
17768	2003	The Compan
17769	2003	Alien Hunte

5. I tried to find trend of number of movies re-released over year. According figure below, we can see that during year 1985 and 1997, number of re-released movies quickly rise up. And during year 1998 and 2002, about 22 movies are re-released each year. According to this chart, we can see that before 1985, seldom movies were re-released. Year 1985 is a turning point where after 1985, large amount of movies were re-released every year. There could be multiple reasons behind, filming technology innovation, people love old movie, or film companies market strategy.



6. I come up with two ideas about this file. First, which is a classic problem, is to predict future movie rating based on these records. This can be done by training models with given files. However, to get a prediction model with high predict rate is complicated. Second, we can solve problem that what period is a movie tends to get higher rating after its release. This can be done by comparing period of high rating of a movie with its releasing time. And concatenate results of each movie together to get the answer.