

1. Import Necessary Library

```
In [1]: # Name: Md Rashed Karim  
# Student ID: 905242031  
import numpy as np
```

2. Numpy Array

i) Create 2 numpy 1D array (arr1,arr2) and populate data using random library function

```
In [19]: arr1 = np.random.randint(15,size=5)  
arr1
```

```
Out[19]: array([ 4, 10,  1, 10,  3])
```

```
In [21]: arr2 = np.random.randint(20, 50, size=5)  
arr2
```

```
Out[21]: array([42, 49, 29, 24, 37])
```

ii) Find out all the unique values from both arrays and print the output in a sorted manner

```
In [45]: np.sort(np.unique(arr1))
```

```
Out[45]: array([ 1,  3,  4, 10])
```

```
In [43]: np.sort(np.unique(arr2))
```

```
Out[43]: array([24, 29, 37, 42, 49])
```

iii) Find out the summation of all even numbers of these arrays

```
In [47]: even_arr1 = arr1[arr1 % 2 == 0].sum()  
even_arr1
```

```
Out[47]: 24
```

```
In [49]: even_arr2 = arr2[arr2 % 2 == 0].sum()  
even_arr2
```

```
Out[49]: 66
```

iv) Create a 2D array combining this 2 array

```
In [63]: d2 = np.vstack((arr1, arr2))  
d2
```

```
Out[63]: array([[ 4, 10,  1, 10,  3],  
               [42, 49, 29, 24, 37]])
```

v) Find the maximum value of each column

```
In [65]: column_max = np.max(d2, axis=0)  
column_max
```

```
Out[65]: array([42, 49, 29, 24, 37])
```

3. Pandas

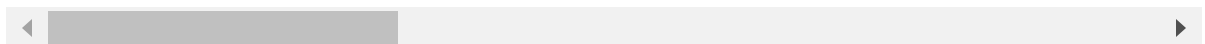
i) import data from provided file and display

```
In [167... import pandas as pd  
df = pd.read_csv("movie.csv")  
df
```

Out[167...

| | title | year | color | content_rating | duration | director_name | director_fb | ac |
|------|--|--------|-------|----------------|----------|-------------------|-------------|----------|
| 0 | Avatar | 2009.0 | Color | PG-13 | 178.0 | James Cameron | 0.0 | Pou |
| 1 | Pirates of the Caribbean: At World's End | 2007.0 | Color | PG-13 | 169.0 | Gore Verbinski | 563.0 | Joh D |
| 2 | Spectre | 2015.0 | Color | PG-13 | 148.0 | Sam Mendes | 0.0 | Christ V |
| 3 | The Dark Knight Rises | 2012.0 | Color | PG-13 | 164.0 | Christopher Nolan | 22000.0 | H. |
| 4 | Star Wars: Episode VII - The Force Awakens | NaN | NaN | NaN | NaN | Doug Walker | 131.0 | D W |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4911 | Signed Sealed Delivered | 2013.0 | Color | NaN | 87.0 | Scott Smith | 2.0 | Ma |
| 4912 | The Following | NaN | Color | TV-14 | 43.0 | NaN | NaN | Na |
| 4913 | A Plague So Pleasant | 2013.0 | Color | NaN | 76.0 | Benjamin Roberds | 0.0 | Boet |
| 4914 | Shanghai Calling | 2012.0 | Color | PG-13 | 100.0 | Daniel Hsia | 0.0 | f |
| 4915 | My Date with Drew | 2004.0 | Color | PG | 90.0 | Jon Gunn | 16.0 | J Au |

4916 rows × 22 columns



ii) Do the necessary for missing value handling apply appropriate technique

In [169...

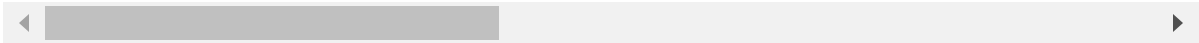
```
print("Missing values in the DataFrame:")
df.isnull()
```

Missing values in the DataFrame:

Out[169...

| | title | year | color | content_rating | duration | director_name | director_fb | actor1 | actc |
|------|-------|-------|-------|----------------|----------|---------------|-------------|--------|------|
| 0 | False | False | False | False | False | False | False | False | |
| 1 | False | False | False | False | False | False | False | False | |
| 2 | False | False | False | False | False | False | False | False | |
| 3 | False | False | False | False | False | False | False | False | |
| 4 | False | True | True | True | True | False | False | False | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 4911 | False | False | False | True | False | False | False | False | |
| 4912 | False | True | False | False | False | True | True | False | |
| 4913 | False | False | False | True | False | False | False | False | |
| 4914 | False | False | False | False | False | False | False | False | |
| 4915 | False | False | False | False | False | False | False | False | |

4916 rows × 22 columns



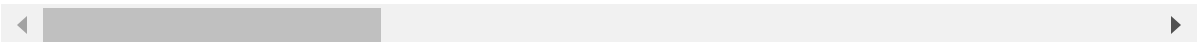
In [171...

```
df.dropna()
```

Out[171...

| | title | year | color | content_rating | duration | director_name | director_fb | ac |
|-------------|--|--------|-------|----------------|----------|-------------------|-------------|-------------|
| 0 | Avatar | 2009.0 | Color | PG-13 | 178.0 | James Cameron | 0.0 | Pou |
| 1 | Pirates of the Caribbean: At World's End | 2007.0 | Color | PG-13 | 169.0 | Gore Verbinski | 563.0 | Jo [|
| 2 | Spectre | 2015.0 | Color | PG-13 | 148.0 | Sam Mendes | 0.0 | Chris \\ |
| 3 | The Dark Knight Rises | 2012.0 | Color | PG-13 | 164.0 | Christopher Nolan | 22000.0 | H |
| 5 | John Carter | 2012.0 | Color | PG-13 | 132.0 | Andrew Stanton | 475.0 | I Se |
| ... | ... | ... | ... | ... | ... | ... | ... | |
| 4906 | Primer | 2004.0 | Color | PG-13 | 77.0 | Shane Carruth | 291.0 | S Ca |
| 4907 | Cavite | 2005.0 | Color | Not Rated | 80.0 | Neill Dela Llana | 0.0 | Gam |
| 4908 | El Mariachi | 1992.0 | Color | R | 81.0 | Robert Rodriguez | 0.0 | C Gall |
| 4910 | Newlyweds | 2011.0 | Color | Not Rated | 95.0 | Edward Burns | 0.0 | I I |
| 4915 | My Date with Drew | 2004.0 | Color | PG | 90.0 | Jon Gunn | 16.0 | At |

3706 rows × 22 columns



iii) Change inappropriate data type (if any)

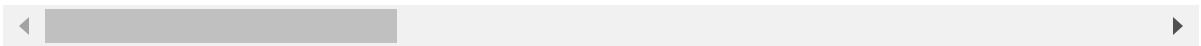
In [173...

```
df['year'] = df['year'].fillna(0).astype(int)
df
```

Out[173...

| | title | year | color | content_rating | duration | director_name | director_fb | acto |
|------|--|------|-------|----------------|----------|-------------------|-------------|------------|
| 0 | Avatar | 2009 | Color | PG-13 | 178.0 | James Cameron | 0.0 | Cc Pounc |
| 1 | Pirates of the Caribbean: At World's End | 2007 | Color | PG-13 | 169.0 | Gore Verbinski | 563.0 | John De |
| 2 | Spectre | 2015 | Color | PG-13 | 148.0 | Sam Mendes | 0.0 | Christo Wa |
| 3 | The Dark Knight Rises | 2012 | Color | PG-13 | 164.0 | Christopher Nolan | 22000.0 | Tc Har |
| 4 | Star Wars: Episode VII - The Force Awakens | 0 | NaN | NaN | NaN | Doug Walker | 131.0 | Do Walk |
| ... | ... | ... | ... | ... | ... | ... | ... | |
| 4911 | Signed Sealed Delivered | 2013 | Color | NaN | 87.0 | Scott Smith | 2.0 | E Mabi |
| 4912 | The Following | 0 | Color | TV-14 | 43.0 | NaN | NaN | Nata Z |
| 4913 | A Plague So Pleasant | 2013 | Color | NaN | 76.0 | Benjamin Roberds | 0.0 | E Boehn |
| 4914 | Shanghai Calling | 2012 | Color | PG-13 | 100.0 | Daniel Hsia | 0.0 | Al Ru |
| 4915 | My Date with Drew | 2004 | Color | PG | 90.0 | Jon Gunn | 16.0 | Jo Augu |

4916 rows × 22 columns



iv) Rename the column with meaningful name (at least one column)

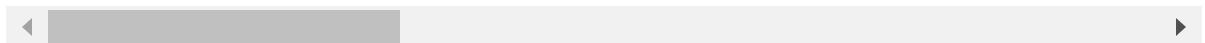
In [175...

```
df.rename(columns={'title': 'Movie Name'})
```

Out[175...

| | Movie Name | year | color | content_rating | duration | director_name | director_fb | acto |
|------|--|------|-------|----------------|----------|-------------------|-------------|------------|
| 0 | Avatar | 2009 | Color | PG-13 | 178.0 | James Cameron | 0.0 | C Pounc |
| 1 | Pirates of the Caribbean: At World's End | 2007 | Color | PG-13 | 169.0 | Gore Verbinski | 563.0 | John De |
| 2 | Spectre | 2015 | Color | PG-13 | 148.0 | Sam Mendes | 0.0 | Christo Wa |
| 3 | The Dark Knight Rises | 2012 | Color | PG-13 | 164.0 | Christopher Nolan | 22000.0 | Tc Har |
| 4 | Star Wars: Episode VII - The Force Awakens | 0 | NaN | NaN | NaN | Doug Walker | 131.0 | Do Walk |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 4911 | Signed Sealed Delivered | 2013 | Color | NaN | 87.0 | Scott Smith | 2.0 | E Mabi |
| 4912 | The Following | 0 | Color | TV-14 | 43.0 | NaN | NaN | Nata Z |
| 4913 | A Plague So Pleasant | 2013 | Color | NaN | 76.0 | Benjamin Roberds | 0.0 | E Boehn |
| 4914 | Shanghai Calling | 2012 | Color | PG-13 | 100.0 | Daniel Hsia | 0.0 | Al Ru |
| 4915 | My Date with Drew | 2004 | Color | PG | 90.0 | Jon Gunn | 16.0 | Jo Augu |

4916 rows × 22 columns



v) Select the highest budgeted movie

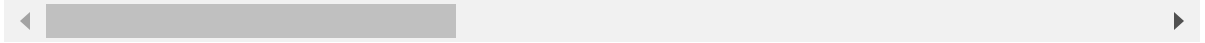
In [177...

```
df[df['budget'] == df['budget'].max()]
```

Out[177...

| | title | year | color | content_rating | duration | director_name | director_fb | actor1 |
|------|----------------|------|-------|----------------|----------|----------------|-------------|--------------|
| 3787 | Lady Vengeance | 2005 | Color | R | 112.0 | Chan-wook Park | 0.0 | Min-sik Choi |

1 rows × 22 columns



vi) Select the best 10 movies according their imdb score

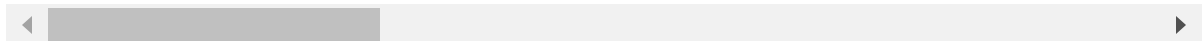
In [179...

```
df.head(10)
```


Out[179...

| | title | year | color | content_rating | duration | director_name | director_fb | actor1 |
|---|--|------|-------|----------------|----------|-------------------|-------------|-----------------|
| 0 | Avatar | 2009 | Color | PG-13 | 178.0 | James Cameron | 0.0 | CCI-Pounder |
| 1 | Pirates of the Caribbean: At World's End | 2007 | Color | PG-13 | 169.0 | Gore Verbinski | 563.0 | Johnny Depp |
| 2 | Spectre | 2015 | Color | PG-13 | 148.0 | Sam Mendes | 0.0 | Christoph Waltz |
| 3 | The Dark Knight Rises | 2012 | Color | PG-13 | 164.0 | Christopher Nolan | 22000.0 | Tom Hardy |
| 4 | Star Wars: Episode VII - The Force Awakens | 0 | NaN | NaN | NaN | Doug Walker | 131.0 | Doug Walker |
| 5 | John Carter | 2012 | Color | PG-13 | 132.0 | Andrew Stanton | 475.0 | Dary Sabara |
| 6 | Spider-Man 3 | 2007 | Color | PG-13 | 156.0 | Sam Raimi | 0.0 | J.K. Simmons |
| 7 | Tangled | 2010 | Color | PG | 100.0 | Nathan Greno | 15.0 | Brac Garrett |
| 8 | Avengers: Age of Ultron | 2015 | Color | PG-13 | 141.0 | Joss Whedon | 0.0 | Chris Hemsworth |
| 9 | Harry Potter and the Half-Blood Prince | 2009 | Color | PG | 153.0 | David Yates | 282.0 | Alar Rickmar |

10 rows × 22 columns



vii) Select all the movies such that the Facebook likes for actor 2 are greater than those for actor 1

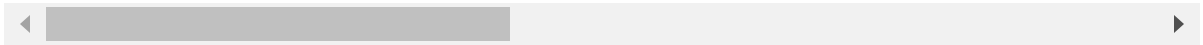
In [199...

```
# df.info()
df[df['actor2_fb'] > df['actor1_fb']]
```

Out[199...

| title | year | color | content_rating | duration | director_name | director_fb | actor1 | actor1_fb |
|-------|------|-------|----------------|----------|---------------|-------------|--------|-----------|
|-------|------|-------|----------------|----------|---------------|-------------|--------|-----------|

0 rows × 22 columns



viii) Find out the director who produced the highest number of movies

In [221...

```
director_movie_count = df.groupby('director_name')['title'].count()
director_movie_count
top_director = director_movie_count.idxmax()
max_movies = director_movie_count.max()

print(f'The director: "{top_director}" who produced {max_movies} movies.')
# df[df['director_name'] > df['actor1_fb']]
```

The director: "Steven Spielberg" who produced 26 movies.