# FIT 1043 Introduction to Data Science

# Assignment 1

Lang Zolyn

30719704

---

## Introduction

The film industry has been acting as an entertainment platform for people all over the world since a long time ago. It can be seen that the industry has been growing rapidly in the past years which is likely due to the acceleration of online mobile distribution and lower admission prices. In order to find out what factor contributes most to the success of each movie, it is necessary for us to analyze using several techniques such as data wrangling, functions and graphs that will manipulate and enable data visualization.

## Importing the necessary libraries

In [18]:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
ticket_seating=pd.read_csv('FIT1043-ticket-seating.csv')
ticket_trx=pd.read_csv('FIT1043-ticket-trx.csv')
merged_ticket = ticket_trx.merge(ticket_seating, left_on=["Transaction.Number"
,"Transaction.Sequence.Number"], right_on = ["Transaction.Number","Transactio
n.Sequence.Number"], how='left')
pd.set_option('display.max_columns', 50)
merged_ticket.head()
```

Out[18]:

| | Transaction.Number | Transaction.Sequence.Number | Transaction.Date.Time | Type.Of.Transaction |
|---|---|---|---|---|
| 0 | 689235 | 5 | 2/4/2017 0:34 | Refund Portion |
| 1 | 689235 | 6 | 2/4/2017 0:34 | Refund Portion |
| 2 | 691991 | 5 | 2/4/2017 0:33 | Refund Portion |
| 3 | 691991 | 6 | 2/4/2017 0:33 | Refund Portion |
| 4 | 692271 | 1 | 1/4/2017 6:01 | Ticket Refunded |

The above function is to merge the two files provided using the unique identifiers to make sure there are no duplicated columns. Then, the first 5 rows are displayed to have an overview look at the merged files.

## Description of data

```
In [299]: merged_ticket.isnull().values.any()
```

```
Out[299]: True
```

The data set contains null value.

```
In [300]: merged_ticket.shape
```

```
Out[300]: (88477, 49)
```

The data set contains 88477 rows and 49 columns.

```
In [301]: print(merged_ticket['Session.Screening.Time'].dtypes)
          print(merged_ticket['Transaction.Date.Time'].dtypes)
```

```
          object
          object
```

The data type of session screening time and transaction date time are both object.

```
In [303]: n_uni_seat = {'Seat.Number':{'Unique Values':'nunique'}}
          merged_ticket.agg(n_uni_seat)
```

Out[303]:

| | Seat.Number |
|---|---|
| Unique Values | 28 |

There are 28 unique values in the column ‘Seat.Number’.

In [304]:
```
merged_ticket.describe()
```

Out[304]:

| | Transaction.Number | Transaction.Sequence.Number | Ticket.Type.Code | Admits | Gross. |
|---|---|---|---|---|---|
| count | 88477.000000 | 88477.000000 | 88477.000000 | 88477.000000 | 88 |
| mean | 726713.927156 | 5.204211 | 31.250246 | 0.988359 | |
| std | 20475.126596 | 23.963779 | 9.905059 | 0.152143 | |
| min | 689235.000000 | 1.000000 | 1.000000 | -1.000000 | |
| 25% | 708132.000000 | 1.000000 | 36.000000 | 1.000000 | |
| 50% | 726091.000000 | 2.000000 | 36.000000 | 1.000000 | |
| 75% | 744559.000000 | 2.000000 | 36.000000 | 1.000000 | |
| max | 761953.000000 | 392.000000 | 47.000000 | 1.000000 | |

8 rows × 25 columns

*There are several useful fields such as the transaction number, admits, sales taxes and full prices. From the data frame above, we can see that there is a total of 88477 transactions. The highest price for one ticket is 10, the average admits is almost 1.0 and the maximum entertainment tax and good and services tax are 1.9 and 0.46. Also, the average of the user's best-averaged order time(turnaround) is around 54 seconds, this indicates that the user is quite efficient.*

## *1. Which movie generated the highest revenue in the dataset?*

*In order to figure out which movie generated the highest revenue, we need to group the movies accordingly first by using the group by function. In this case, I chose to use Film HO Code to represent my movie since the original movie name is replaced by this code. Moving on, I will just assume that the gross box office is the revenue because it already includes the refunded amount and it is also the full price including the two taxes, which is the entertainment and goods and services tax (GST). To visualize the data better, I used a bar graph to represent my data.*

In [305]:
```
Revenue_calc = {'Gross.Box.Office':{'Revenue':'sum'}}
```

*The above code is to calculate the sum of gross box office for the revenue.*

In [306]:

```
groupbyfilm = merged_ticket.groupby('Film.HO.Code').agg(Revenue_calc)
groupbyfilm = groupbyfilm.reset_index()
groupbyfilm.columns = groupbyfilm.columns.droplevel(0)
groupbyfilm.rename(columns = {'':'Film HO Code'},inplace = True)
groupbyfilm
```

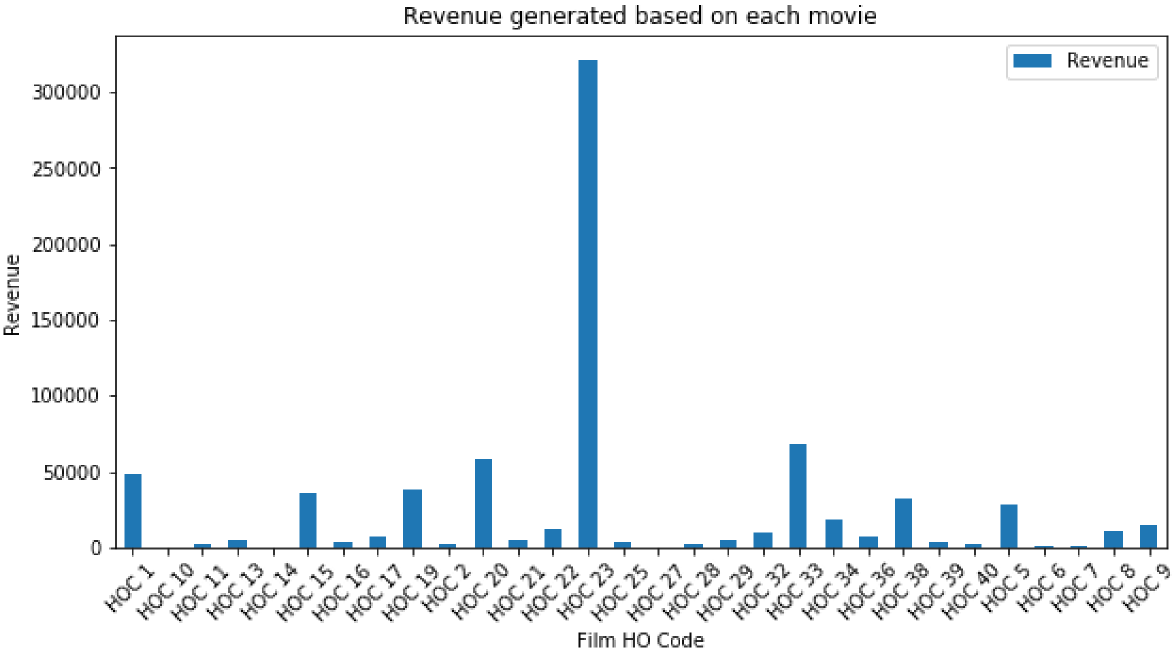
Out[306]:

|  | Film HO Code | Revenue |
|---|---|---|
| 0 | HOC 1 | 48343.0 |
| 1 | HOC 10 | 71.5 |
| 2 | HOC 11 | 2675.0 |
| 3 | HOC 13 | 4872.0 |
| 4 | HOC 14 | 80.0 |
| 5 | HOC 15 | 35775.5 |
| 6 | HOC 16 | 3780.0 |
| 7 | HOC 17 | 7786.0 |
| 8 | HOC 19 | 37826.0 |
| 9 | HOC 2 | 2638.0 |
| 10 | HOC 20 | 58725.0 |
| 11 | HOC 21 | 4376.0 |
| 12 | HOC 22 | 12488.5 |
| 13 | HOC 23 | 321248.5 |
| 14 | HOC 25 | 3969.5 |
| 15 | HOC 27 | 36.0 |
| 16 | HOC 28 | 2540.0 |
| 17 | HOC 29 | 4446.0 |
| 18 | HOC 32 | 9260.5 |
| 19 | HOC 33 | 68564.0 |
| 20 | HOC 34 | 18225.0 |
| 21 | HOC 36 | 7410.0 |
| 22 | HOC 38 | 31590.0 |
| 23 | HOC 39 | 3375.5 |
| 24 | HOC 40 | 2219.0 |
| 25 | HOC 5 | 28629.5 |
| 26 | HOC 6 | 1682.0 |
| 27 | HOC 7 | 1404.0 |
| 28 | HOC 8 | 11321.5 |
| 29 | HOC 9 | 15309.0 |

*The code below is to construct a bar chart by grouping the film, setting the size to the plot, use column Film HO Code for x axis label and setting the title of the chart which is revenue generated based on the movies.*

```
In [307]: ax=groupbyfilm.plot.bar(figsize=(10,5))
          ax.set_xticklabels(groupbyfilm['Film HO Code'],rotation=45)
          plt.xlabel('Film HO Code')
          plt.ylabel('Revenue')
          plt.title('Revenue generated based on each movie')

Out[307]: Text(0.5, 1.0, 'Revenue generated based on each movie')
```



*According to the bar graph and the data frame, film HOC 23 generates the highest revenue in the data set which is 321248.5 This means that the film is extremely famous and wholesome as it manages to generate a very high amount of revenue compared to other movies.*

## 2. Which day (as in Monday, Tuesday, etc and not date) is the least popular day to watch a movie?

*In order to determine which day is the least popular day to watch a movie, I will need to use the session screening time and the admits to confirm the number of moviegoers based on that particular day, I choose to use admits instead of just counting the seat numbers is because admits includes the refunded portion (in this case it would be more simple to use admits right away).*

In [308]:

```python
merged_ticket['Session.Screening.Time'] = pd.to_datetime(merged_ticket['Session.Screening.Time'])
print(merged_ticket['Session.Screening.Time'].dtypes)
```

```
datetime64[ns]
```

*The above code is to change the type of 'Session.Screening.Time' to datetime.*

In [309]:

```python
 merged_ticket['Day of Week'] = merged_ticket['Session.Screening.Time'].dt.weekday_name
```

*The code above converts the date into weekday names in English.*

In [310]:

```python
refund=merged_ticket['Type.Of.Transaction']=="Refund Portion"
no_of_refunds=len(merged_ticket[refund])
no_of_refunds
```

Out[310]:

```
515
```

*From the above code, we know that in total there are 515 refunded tickets.*

In [311]:

```python
viewers={'Admits':{'Number of viewers':'sum'}}
number_of_viewers = merged_ticket.groupby('Day of Week').agg(viewers).reset_index()
number_of_viewers.columns =number_of_viewers.columns.droplevel(0)
number_of_viewers.rename(columns = {'':'Day of Week'},inplace = True)
number_of_viewers.sort_values(by=['Number of viewers'], inplace=True)
number_of_viewers
```

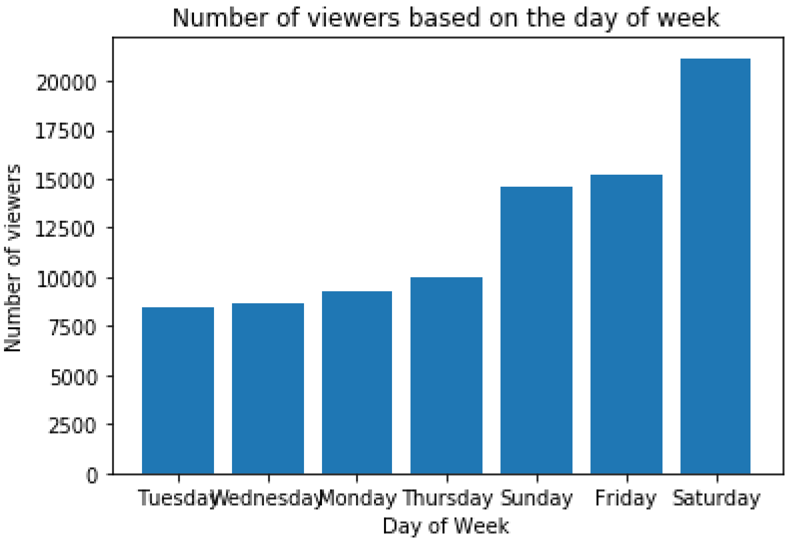
Out[311]:

| | Day of Week | Number of viewers |
|---|---|---|
| 5 | Tuesday | 8510 |
| 6 | Wednesday | 8696 |
| 1 | Monday | 9318 |
| 4 | Thursday | 10031 |
| 3 | Sunday | 14562 |
| 0 | Friday | 15166 |
| 2 | Saturday | 21164 |

*From the code above, admits are summed to obtain the actual number of moviegoers who viewed the movie and not the refunded ones. Next, I grouped the number of moviegoers based on the particular day of the week and display them in a data frame neatly.*

In [312]:

```
plt.bar(number_of_viewers['Day of Week'], number_of_viewers['Number of viewers'])
plt.xlabel('Day of Week')
plt.ylabel('Number of viewers')
plt.title('Number of viewers based on the day of week')
plt.show()
```



*Based on the bar graph above, we can see that Tuesday is the least popular day to watch a movie as the number of viewers on that particular day is only 8510. This could be due to many factors such as the free time of the people and mood, often on Tuesday people will be busier compared to the weekends as Monday is the first day of the week and most likely their work will be piled up on Tuesday, it can be also due to the preference of movies of the moviegoers.*

## *3. What is the most popular time of the day for movie goers?*

*The code below is to obtain the specific hour from the session screening time.*

In [314]:

```
merged_ticket['Timing'] = merged_ticket['Session.Screening.Time'].apply(lambda time: time.hour)
```

*The code above is to group the timing into different categories such as morning, afternoon, evening and night. After that, the number of people who watched the movie at that particular time is grouped according to their admits.*

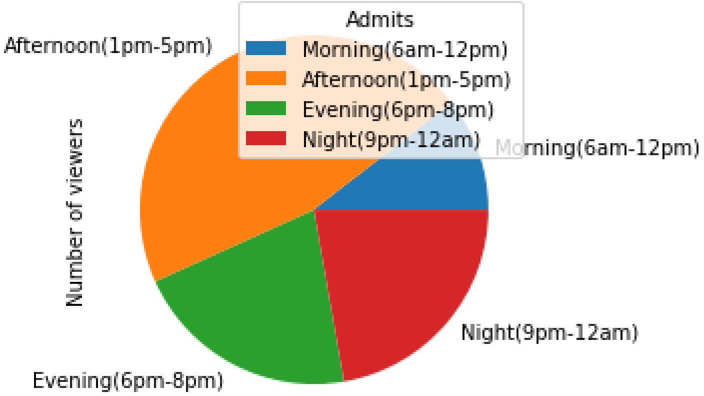
In [315]:
```
no_of_viewers['Number of viewers'] = no_of_viewers[1]- no_of_viewers[-1]
no_of_viewers
```

Out[315]:

| Admits | -1 | 1 | Number of viewers |
|---|---|---|---|
| Morning(6am-12pm) | 34 | 9263 | 9229 |
| Afternoon(1pm-5pm) | 207 | 40602 | 40395 |
| Evening(6pm-8pm) | 120 | 18506 | 18386 |
| Night(9pm-12am) | 154 | 19591 | 19437 |

*The code above is to obtain the actual number of viewers at that particular time by excluding all the refunded portion.*

In [316]:
```
no_of_viewers.plot.pie(y=['Number of viewers'])
plt.show()
```



*According to the pie chart, the most popular time of the day for moviegoers is Afternoon. This can be due to various reasons, such as the individual's productivity and free time, people often tend to be more productive in the morning compared to the afternoon, so they might need some entertainment during the afternoon (after a long day of work), and of course, watching a movie is a great option!*

## *4. Using the column 'Order.Time..Secs', determine (other than WEB user) which user has the best averaged order time (turnaround).*

*The column 'Order.Time.Secs' is basically how fast (efficient) the user is, I assume that the shorter the order time is, the more efficient the user is because less time will be taken.*

In [317]:

```
total={'Order.Time..Secs.':{'Order time (sum)':'sum'}}
efficiency = merged_ticket.groupby('User').agg(total).reset_index()
efficiency.columns =efficiency.columns.droplevel(0)
efficiency.rename(columns = {'':'User'},inplace = True)
new_efficiency=efficiency.drop([0,7],axis=0)
```

In [318]:

```
total2={'Order.Time..Secs.':{'Order time (count)':'count'}}
efficiency2 = merged_ticket.groupby('User').agg(total2).reset_index()
efficiency2.columns =efficiency2.columns.droplevel(0)
efficiency2.rename(columns = {'':'User'},inplace = True)
new_efficiency2=efficiency2.drop([0,7],axis=0)
```

In [319]:

```
removing_user =new_efficiency2.drop("User", axis=1)
```

*I decided to remove the WEB and no show user because it is not needed in this case.*

In [320]:

```
averagefunc= pd.concat([new_efficiency,removing_user ], axis = 1)
```

*To make my graph and data frame neat and clean, I choose to remove the sum and count column for my final data frame, so it can just focus on our main findings which are the best-averaged order time based on the different users.*

In [321]:

```
averagefunc['Best averaged order time'] = averagefunc['Order time (sum)'] /averagefunc['Order time (count)']
averagefunc.sort_values(by=['Best averaged order time'], inplace=True)
averagefunc
removing_count =averagefunc.drop("Order time (sum)", axis=1)
removing_count
removing_sum=removing_count.drop("Order time (count)", axis=1)
removing_sum
```

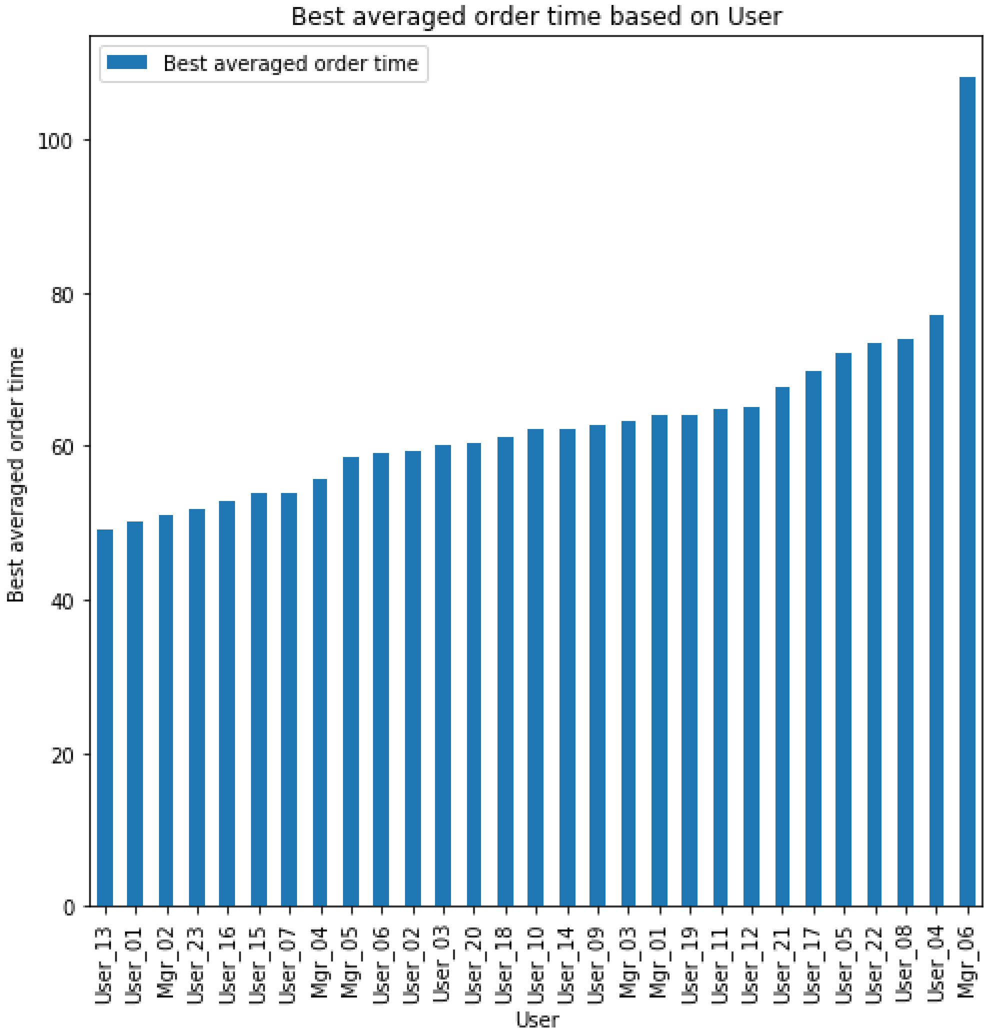
Out[321]:

| | User | Best averaged order time |
|---|---|---|
| 20 | User_13 | 49.155963 |
| 8 | User_01 | 50.144928 |
| 2 | Mgr_02 | 51.057462 |
| 30 | User_23 | 51.778272 |
| 23 | User_16 | 52.850312 |
| 22 | User_15 | 53.827586 |
| 14 | User_07 | 53.854359 |
| 4 | Mgr_04 | 55.588410 |
| 5 | Mgr_05 | 58.430380 |
| 13 | User_06 | 58.981793 |
| 9 | User_02 | 59.410838 |
| 10 | User_03 | 60.104647 |
| 27 | User_20 | 60.247510 |
| 25 | User_18 | 61.253000 |
| 17 | User_10 | 62.209181 |
| 21 | User_14 | 62.269274 |
| 16 | User_09 | 62.739976 |
| 3 | Mgr_03 | 63.139241 |
| 1 | Mgr_01 | 63.928012 |
| 26 | User_19 | 63.969216 |
| 18 | User_11 | 64.889959 |
| 19 | User_12 | 64.961033 |
| 28 | User_21 | 67.582164 |
| 24 | User_17 | 69.664723 |
| 12 | User_05 | 72.029255 |
| 29 | User_22 | 73.419619 |
| 15 | User_08 | 73.897017 |
| 11 | User_04 | 77.066309 |
| 6 | Mgr_06 | 108.167665 |

*The code below is to plot a bar graph for the best-averaged order time based on user, I have sorted the best averaged time in my data frame so that the result can be seen clearly.*

```
In [322]: ax=removing_sum.plot.bar(figsize=(8,8))
          ax.set_xticklabels(removing_sum['User'],rotation=90)
          plt.xlabel('User')
          plt.ylabel('Best averaged order time')
          plt.title('Best averaged order time based on User')
```

```
Out[322]: Text(0.5, 1.0, 'Best averaged order time based on User')
```



*From the bar chart above, it is obvious that user_13 has the best-averaged order time (turnaround) which is only 49.155963 seconds. This means that user_13 is the most efficient user compared to the others. This might be due to various reasons such as time consciousness and the person's habits, some people will take a shorter time in accomplishing a task as they are a very time continuous and productive person.*

## *Business insights*

*The cinema management generates revenue from the number of tickets purchased, this means that to gain higher revenue, we must focus on the film that can attract the moviegoers most, in this case, HOC 23 is a very good example. The cinema should add more screening time and session for that particular movie. However, it doesn't mean that they should completely abandon the other movies that have lesser viewers such as HOC 27 and HOC 10 and HOC 14. In order to boost sales in those other movies, the cinema can choose to promote it in many different ways such as playing the trailers of the movies prior to the screening time of HOC 23 or they can also hand out vouchers and apply them to buy one get one scheme. The cinema may also determine the preference of the majority of the customer by screening more movies of the same genre as film HOC 23. Moving on, the cinema should promote implement special deals and promotions on Tuesday to attract more customers since it has been proven that Tuesday is the least popular day to watch a movie. Next, since the afternoon is the most popular hours for moviegoers, the cinema can arrange more screening sessions during the afternoon compared to the other hours to increase revenue. Lastly, the cinema management should enforce a system where the most efficient user would be presented with incentives—it may be in monetary form or psychological form— so that users such as user_05 user_08 user_22 and user_04 which took more than an average of 70 seconds to handle a transaction may be encouraged to be more efficient; which will increase the productivity of the whole ticket purchasing process and lead to more profit. Moreover, the data also shows that morning is the least popular time for moviegoers. As mentioned above, the cinema should have a variety of discount schemes for showtimes during this period, for instance, a cheaper ticket price. In conclusion, there are many ways that the cinema can make use of the data provided to improvise and plan their marketing strategies.*