Initial Movie Data Exploration

Overview

This project identifed and analyzed relevant indicators that needs to be emphazized during the creation of Microsoft Movie Studio(MMS).A retrospective data derived from Internet Movie Database (IMDB), Box Office Mojo(BOM) were used to select varaibles.These varables were selected on the basis of the genral business model framework.Descriptive statstical analysis showed the importance of evidence based decison making for the continuous success in the movie industry.The MMS can leverage these analysis during the process of studio establishment.

Business Problem

Data driven buisness decission making has been an importnat step for many companies. The microsoft also needs to consider differnt evidences during the process of movie studio establishment. Looking different sources of data relevant to the movie indsutry makes microsoft productive at the differnt levels of the business. With the help of this preliminary study, MMS can select what types of movies have high domestic grooss income without comrpomising the choice and interest of the public, which was analysed by the average rating by the public.

Data Understanding

IMDB is an online database of information related to movies. This databse contains detail information about a movie including the title ratings and basics. Data realted to revenue from each movie was generated uing the Boxofficie Mojo(BMO) dataset. Datas including Movie type(Genres), Domestic gross income, Average rating and Movie lenth in minutes were selected from the dataset.

In [85]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [93]:
```
!ls zippedData/
```

```
bom.movie_gross.csv.gz
imdb.name.basics.csv.gz
imdb.title.akas.csv.gz
imdb.title.basics.csv.gz
imdb.title.crew.csv.gz
imdb.title.principals.csv.gz
imdb.title.ratings.csv.gz
rt.movie_info.tsv.gz
rt.reviews.tsv.gz
tmdb.movies.csv.gz
tn.movie_budgets.csv.gz
```

In [8]:
```
bom=pd.read_csv('zippedData/bom.movie_gross.csv.gz')
```

In [9]:
```
bom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
```

In [23]:
```
#Create a new column and call it title_year
bom['title_year'] = bom['title'] + " " + bom['year'].astype(str)
```

In [24]:
```
bom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
 5   title_year      3387 non-null   object 
dtypes: float64(1), int64(1), object(4)
memory usage: 158.9+ KB
```

In [25]:
```
bom.head()
```

Out[25]:

	title	studio	domestic_gross	foreign_gross	year	title_year
0	Toy Story 3	BV	415000000.0	652000000	2010	Toy Story 3 2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010	Alice in Wonderland (2010) 2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010	Harry Potter and the Deathly Hallows Part 1 2010
3	Inception	WB	292600000.0	535700000	2010	Inception 2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010	Shrek Forever After 2010

In [26]:
```
tit_basics= pd.read_csv('zippedData/imdb.title.basics.csv.gz')
```

In [27]:
```

tit_basics.rename(columns={"primary_title":"title"} , inplace=True)
```

In [11]:
```
tit_basics.head()
```

Out[11]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

In [28]:
```
# Create a new column name it title_year
tit_basics['title_year']= tit_basics['title'] + " " + tit_basics['start_year'
```

In [103]:
```
tit_basics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   tconst           146144 non-null  object 
 1   title            146144 non-null  object 
 2   original_title   146123 non-null  object 
 3   start_year       146144 non-null  int64  
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object 
 6   title_year       146144 non-null  object 
dtypes: float64(1), int64(1), object(5)
memory usage: 7.8+ MB
```

In [29]:
```
tit_basics.head()
```

Out[29]:

	tconst	title	original_title	start_year	runtime_minutes	genres	title
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	Sun
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	Or Befo S
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	The Side Win
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama	Bada
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	War

Data Preparation

Data Cleaning

Across all data sets, to make the workflow easier, column names were normalzied when needed.Unnecessary columns were dropped and also removed all missing values.

Merging Datasets

Mergring took place at two places to bring the three datasets in a single table.A new varaiable were created at both steps before merging them. The new variable, Movie_Data was created to join BOM and title basics datasets using the columns title_year and title. The second merge was perofmred by creating a new varable called Movie_Analysis,in referece to joining the variable Movie_Data and title rating using the common column 'tconst' which is a unique key, to prevent making duplicate rows.

In [51]:

```
# Merge the bom and title basics on animal title_year and year
Movie_Data=pd.merge(bom,tit_basics, on=['title_year', 'title'])
```

In [31]:

```
Movie_Data.head()
```

Out[31]:

	title	studio	domestic_gross	foreign_gross	year	title_year	tconst	original_title	s
0	Toy Story 3	BV	415000000.0	652000000	2010	Toy Story 3 2010	tt0435761	Toy Story 3	
1	Inception	WB	292600000.0	535700000	2010	Inception 2010	tt1375666	Inception	
2	Shrek Forever After	P/DW	238700000.0	513900000	2010	Shrek Forever After 2010	tt0892791	Shrek Forever After	
3	The Twilight Saga: Eclipse	Sum.	300500000.0	398000000	2010	The Twilight Saga: Eclipse 2010	tt1325004	The Twilight Saga: Eclipse	
4	Iron Man 2	Par.	312400000.0	311500000	2010	Iron Man 2 2010	tt1228705	Iron Man 2	

In [32]:

```
# Drop the column foreign gross and save the change.
Movie_Data.drop(columns='foreign_gross', inplace=True)
```

In [108]:
```
Movie_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1873 entries, 0 to 1872
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   title            1873 non-null   object 
 1   studio           1871 non-null   object 
 2   domestic_gross   1863 non-null   float64
 3   year             1873 non-null   int64  
 4   title_year       1873 non-null   object 
 5   tconst           1873 non-null   object 
 6   original_title   1873 non-null   object 
 7   start_year       1873 non-null   int64  
 8   runtime_minutes  1863 non-null   float64
 9   genres           1871 non-null   object 
dtypes: float64(2), int64(2), object(6)
memory usage: 161.0+ KB
```

In [54]:
```
tit_ratings= pd.read_csv('zippedData/imdb.title.ratings.csv.gz')
```

In [55]:
```
tit_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   tconst         73856 non-null  object 
 1   averagerating  73856 non-null  float64
 2   numvotes       73856 non-null  int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
```

In [61]:
```
# Merge Movie data and title ratings  on tconstant.
Movie_Analysis=pd.merge(Movie_Data,tit_ratings, on='tconst')
```

In [60]:
```
Movie_Analysis.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1847 entries, 0 to 1846
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   title            1847 non-null   object 
 1   studio           1845 non-null   object 
 2   domestic_gross   1837 non-null   float64
 3   foreign_gross    1269 non-null   object 
 4   year             1847 non-null   int64  
 5   title_year       1847 non-null   object 
 6   tconst           1847 non-null   object 
 7   original_title   1847 non-null   object 
 8   start_year       1847 non-null   int64  
 9   runtime_minutes  1843 non-null   float64
 10  genres           1845 non-null   object 
 11  averagerating    1847 non-null   float64
 12  numvotes         1847 non-null   int64  
dtypes: float64(3), int64(3), object(7)
memory usage: 202.0+ KB
```

In [41]:
```
# Remove all mssing values and save the change.
Movie_Analysis.dropna(how= 'all', inplace= True)
```

In [42]:
```
Movie_Analysis.head()
```

Out[42]:

	title	studio	domestic_gross	year	title_year	tconst	original_title	start_year	runti
0	Toy Story 3	BV	415000000.0	2010	Toy Story 3 2010	tt0435761	Toy Story 3	2010	
1	Inception	WB	292600000.0	2010	Inception 2010	tt1375666	Inception	2010	
2	Shrek Forever After	P/DW	238700000.0	2010	Shrek Forever After 2010	tt0892791	Shrek Forever After	2010	
3	The Twilight Saga: Eclipse	Sum.	300500000.0	2010	The Twilight Saga: Eclipse 2010	tt1325004	The Twilight Saga: Eclipse	2010	
4	Iron Man 2	Par.	312400000.0	2010	Iron Man 2 2010	tt1228705	Iron Man 2	2010	

In [64]:
```
Movie_Analysis['genres']
```

Out[64]:
```
0         Adventure,Animation,Comedy
1            Action,Adventure,Sci-Fi
2         Adventure,Animation,Comedy
3            Adventure,Drama,Fantasy
4            Action,Adventure,Sci-Fi
                    ...
1842                           Drama
1843                    Comedy,Drama
1844                           Drama
1845             Action,Drama,Thriller
1846                          Comedy
Name: genres, Length: 1847, dtype: object
```

The .explode() function was used to access the list of movies and to be counted in each genrea.

In [44]:
```
Movie_Analysis['genres'].str.split(",").explode()
```

Out[44]:
```
0       Adventure
0       Animation
0          Comedy
1          Action
1       Adventure
          ...
1844        Drama
1845       Action
1845        Drama
1845     Thriller
1846       Comedy
Name: genres, Length: 4569, dtype: object
```

In [45]:
```
Movie_Analysis['genres']=Movie_Analysis['genres'].str.split(",")
```

In [125]:
```
Movie_Analysis['genres']
```

Out[125]:
```
0       [Adventure, Animation, Comedy]
1          [Action, Adventure, Sci-Fi]
2       [Adventure, Animation, Comedy]
3          [Adventure, Drama, Fantasy]
4          [Action, Adventure, Sci-Fi]
                    ...
1842                          [Drama]
1843                  [Comedy, Drama]
1844                          [Drama]
1845           [Action, Drama, Thriller]
1846                         [Comedy]
Name: genres, Length: 1847, dtype: object
```

In [46]:
```
Movie_exploded=Movie_Analysis.explode('genres')
```

In [173]:
```
Movie_exploded
```

Out[173]:

	title	studio	domestic_gross	year	title_year	tconst	original_title	start_year
0	Toy Story 3	BV	415000000.0	2010	Toy Story 3 2010	tt0435761	Toy Story 3	2010
0	Toy Story 3	BV	415000000.0	2010	Toy Story 3 2010	tt0435761	Toy Story 3	2010
0	Toy Story 3	BV	415000000.0	2010	Toy Story 3 2010	tt0435761	Toy Story 3	2010
1	Inception	WB	292600000.0	2010	Inception 2010	tt1375666	Inception	2010
1	Inception	WB	292600000.0	2010	Inception 2010	tt1375666	Inception	2010
...	...	...	...	...	...	...	...	...
1844	A Paris Education	KL	21600.0	2018	A Paris Education 2018	tt6593240	Mes provinciales	2018
1845	The Quake	Magn.	6200.0	2018	The Quake 2018	tt6523720	Skjelvet	2018
1845	The Quake	Magn.	6200.0	2018	The Quake 2018	tt6523720	Skjelvet	2018
1845	The Quake	Magn.	6200.0	2018	The Quake 2018	tt6523720	Skjelvet	2018
1846	An Actor Prepares	Grav.	1700.0	2018	An Actor Prepares 2018	tt5718046	An Actor Prepares	2018

4569 rows × 12 columns

In [47]:
```
#group by the column genres with the mean function

Movie_Genresdetail=Movie_exploded.groupby('genres').mean().reset_index()
```

In [48]:
```
Movie_Genresdetail.sort_values(by='domestic_gross', ascending= True, inplace=
```

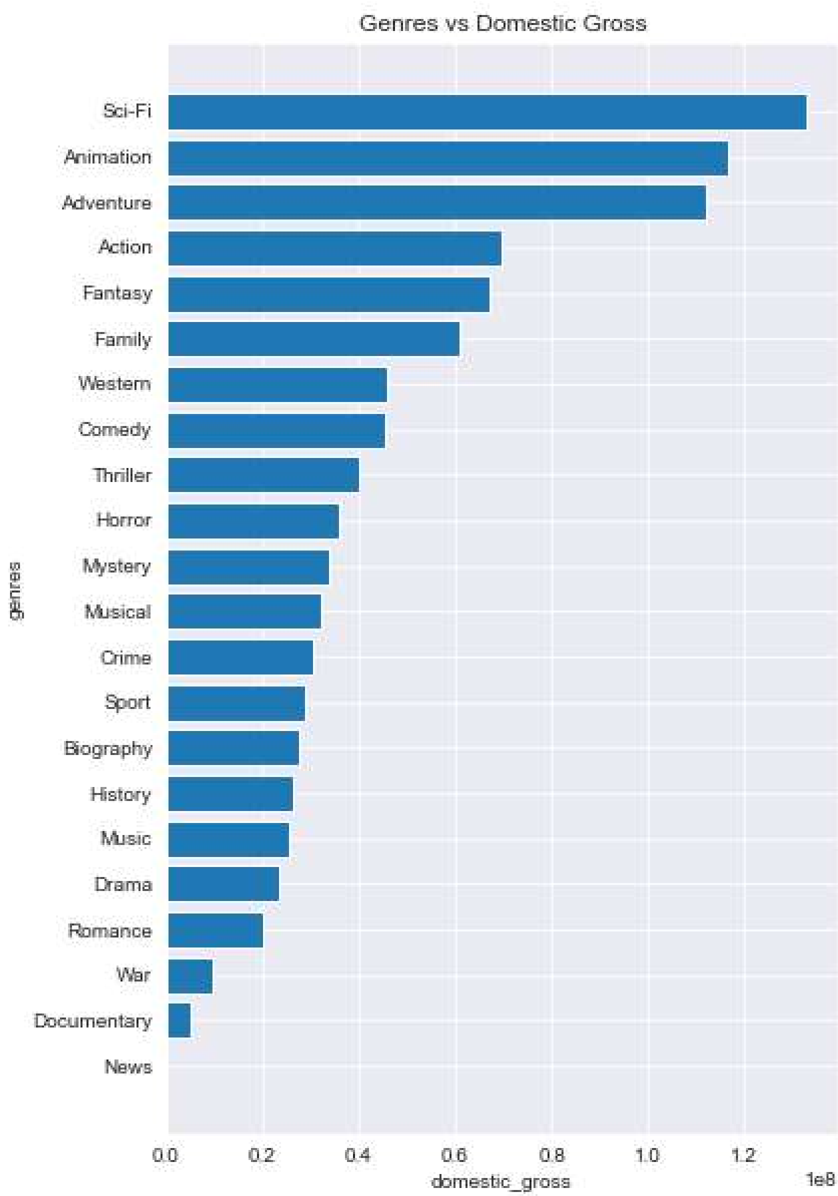
Analysis

Genres and Domestic Gross income

Sci-Fi movies genreate the highest domestic gross income, whereas the least income is from Documentary movies.

In [209]:

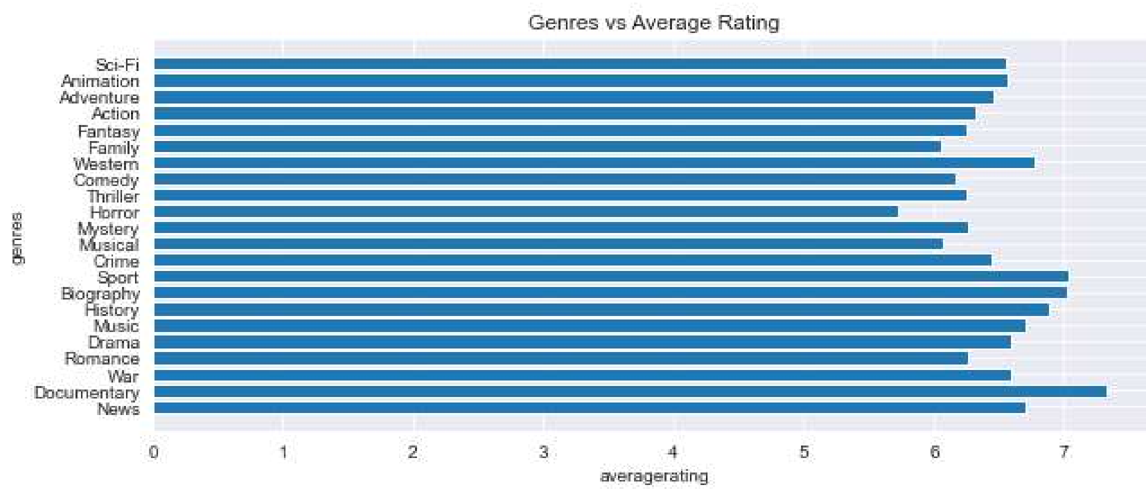
```
fig,ax=plt.subplots()
fig.set_figheight(10)
ax.set_ylabel('genres')
ax.set_xlabel('domestic_gross')
#ax.set_xticklabels(labels=Movie_Genresdetail['genres'], rotation = 45,ha='ri

ax.barh(y=Movie_Genresdetail['genres'], width=Movie_Genresdetail['domestic_gr

ax.set_title('Genres vs Domestic Gross')

plt.savefig("./images.jpg")
```



Genres and Average Rating

Among all movie types, the genre Documnetary has the highest ratings.

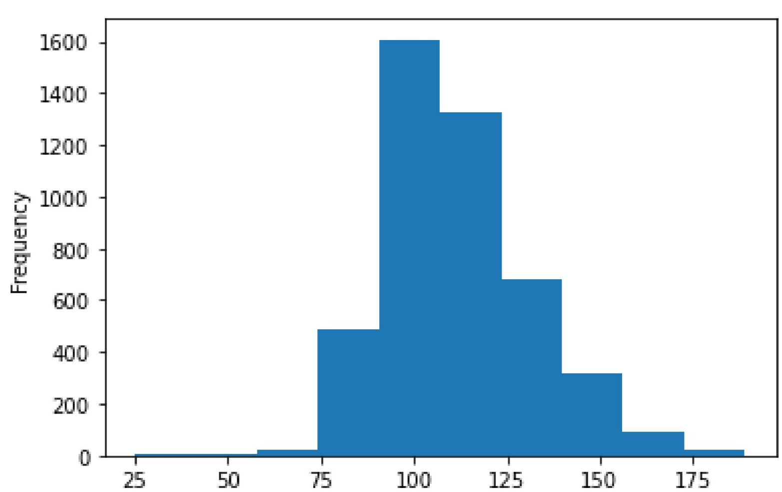
In [214]:
```
fig,ax=plt.subplots()
fig.set_figwidth(10)
ax.set_ylabel('genres')
ax.set_xlabel('averagerating')
#ax.set_xticklabels(labels=Movie_Genresdetail['genres'], rotation = 45,ha='ri

ax.barh(y=Movie_Genresdetail['genres'], width=Movie_Genresdetail['averagerati

ax.set_title('Genres vs Average Rating')

plt.savefig("./images.jpg")
```



Highest Frequency was observed with movies taht have a length of hundred minutes.

In [84]:
```
Movie_exploded['runtime_minutes'].plot.hist()
ax.set_ylabel('frequency')
ax.set_xlabel('runtime_minutes')
ax.set_title('Frequency of Runtime in Minutes')
```

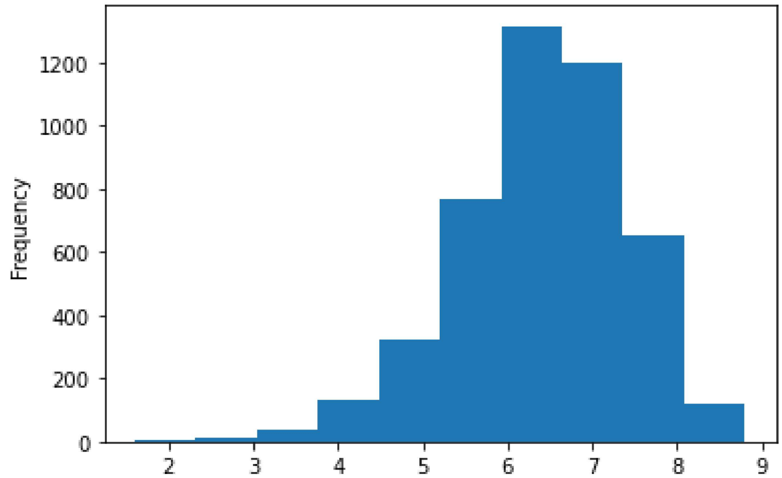
Out[84]: Text(0.5, 1.0, 'Frequency of Runtime in Minutes')



Among the rating scores, the highest frequency was six.

In [81]:
```
Movie_exploded['averagerating'].plot.hist()
ax.set_ylabel('frequency')
ax.set_xlabel('averagerating')
ax.set_title('Frequency of Average Rating')
```

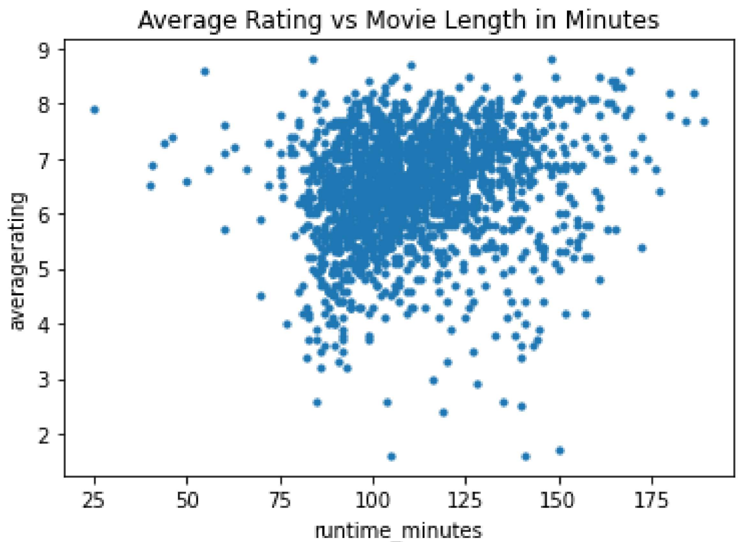
Out[81]: Text(0.5, 1.0, 'Frequency of Average Rating')



The Association Between Movie Rating and Runtime in Minutes

The below scatter plots show a slight positive asscoation between the average rating vs the lenght of movie at 100 to 120 minutes long.

In [83]:
```
x=Movie_Analysis['runtime_minutes']
y=Movie_Analysis['averagerating']
fig, ax=plt.subplots()
ax.scatter(x=x , y=y, s=10,)
ax.set_ylabel('averagerating')
ax.set_xlabel('runtime_minutes')
ax.set_title('Average Rating vs Movie Lenght in Minutes')

plt.savefig("./images.jpg")
```



In [112]:

```
x=Movie_Analysis['runtime_minutes']
y=Movie_Analysis['averagerating']
fig, ax=plt.subplots()
sns.set(rc={'figure.figsize':(17.7,8.27)})
sns.regplot(x=x , y=y, scatter_kws={"s": 7})
#ax.set_ylabel('averagerating')
#ax.set_xlabel('runtime_minutes')
#ax.set_title('Average Rating vs Movie Lenght in Minutes')

#plt.savefig("./images.jpg")
```

Out[112]: <AxesSubplot:xlabel='runtime_minutes', ylabel='averagerating'>

Conclusions

Based on the above analysis, the follwing recommendations are forwarded to the Microsoft.

1. Engage in producing Sci-Fi movies.
2. Length of movies must be considered during production.
3. Movie length and average rating tends to have a relationship.
4. Generate data to address the conflicting results between higher rating movies vs high domestic gross income.

Next Steps

1.Organize and recruit various team of expertise. Sicne the movie industry is new to the micorsoft, it is inevitable to expect some level of challenge.Therefore, to address and overcome multiple issues in the movie industry, expertie who can extraplote multiple infromation form the data realted to the production, marketing and managment are mandatory.

2.Further study should be conducted.Muliple data sets must be collected and utilized for further anlysis to achieve the mission of MMS.

3.Microsoft should expand the technology dimenstion in to the movie production.The long history of Microsfot's role in the tech industry could bring a major advancement in videography.