Microsoft Movie Studio Needs Analysis

Overview

This project analyzes the provided movie data to assist Microsoft Movie Studio to make a sound decision in producing movies. Movie business is fiercely competitive, therefore it is imperative to analyze the movie data correctly. In this attempt analysis of genres, directors, and writers will be provided to aid Microsoft Movie Studio create quality and profitable movies.

Business Problem

Microsoft Movie Studio can produce quality and profitable movies based on the analysis provided. In order for Microsoft Movie Studio to continue making movies, they need to be profitable and quality in kind. My desire is that by analyzing profitable genres, quality directors and writers, help Microsoft Movie Studio to produce profitable and quality movies well into the future

Data Understanding

The Numbers data provide world wide gross of each movie. IMDB data provides ratings for each movie and information regarding which directors and writers were involved in those movies.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import sqlite3
import datetime as dt
```

In [2]:
```
#zippedData/tn.movie_budgets is the Numbers data I use to get world wide gross.
movies_budget = pd.read_csv("zippedData/tn.movie_budgets.csv.gz")
```

In [3]:
```
#These are list of pandas tables I use to come up with the directors and writers with best ratings.
conn = sqlite3.connect("im.db")
movies_info = pd.read_sql("SELECT * FROM movie_basics;", conn)
directors_info = pd.read_sql('SELECT * FROM directors;', conn)
writers_info = pd.read_sql("SELECT * FROM writers;", conn)
ratings_info = pd.read_sql("SELECT * FROM movie_ratings;", conn)
main_info = pd.read_sql("SELECT * FROM principals;", conn)
persons_info = pd.read_sql("SELECT * FROM persons;", conn)
```

In [4]:
```
movies_budget.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 5782 non-null   int64 
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
```

In [5]:
```
movies_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   movie_id         146144 non-null  object 
 1   primary_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 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

In [6]:
```
directors_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 291174 entries, 0 to 291173
Data columns (total 2 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   movie_id   291174 non-null  object
 1   person_id  291174 non-null  object
dtypes: object(2)
memory usage: 4.4+ MB
```

In [7]:
```
writers_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 255873 entries, 0 to 255872
Data columns (total 2 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   movie_id   255873 non-null  object
 1   person_id  255873 non-null  object
dtypes: object(2)
memory usage: 3.9+ MB
```

In [8]:

```
ratings_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   movie_id       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 [9]:

```
main_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1028186 entries, 0 to 1028185
Data columns (total 6 columns):
 #   Column      Non-Null Count    Dtype 
---  ------      --------------    ----- 
 0   movie_id    1028186 non-null  object
 1   ordering    1028186 non-null  int64 
 2   person_id   1028186 non-null  object
 3   category    1028186 non-null  object
 4   job         177684 non-null   object
 5   characters  393360 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.1+ MB
```

In [10]:

```
persons_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 606648 entries, 0 to 606647
Data columns (total 5 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   person_id           606648 non-null  object
 1   primary_name        606648 non-null  object
 2   birth_year          82736 non-null   float64
 3   death_year          6783 non-null    float64
 4   primary_profession  555308 non-null  object
dtypes: float64(2), object(3)
memory usage: 23.1+ MB
```

Profitable Genres

In [11]:

```
#this pandas table and movies_info1 pandas table will be merged to come up with genres with most
#world wide gross
movies_budget.head()
```

Out[11]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747

In [12]:
```
movies_info.head()
```

Out[12]:

	movie_id	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

Directors and Writers with Good ratings

In [13]:
```
#ratings_info1 pandas table will be combined with multiple pandas tables below to come up with
# the directors and writers with good ratings
ratings_info.head()
```

Out[13]:

	movie_id	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559
2	tt1042974	6.4	20
3	tt1043726	4.2	50352
4	tt1060240	6.5	21

In [14]:
```
movies_info.head()
```

Out[14]:

	movie_id	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 [15]:
```
directors_info.head()
```

Out[15]:

	movie_id	person_id
0	tt0285252	nm0899854
1	tt0462036	nm1940585
2	tt0835418	nm0151540
3	tt0835418	nm0151540
4	tt0878654	nm0089502

In [16]:
```
writers_info.head()
```

Out[16]:

	movie_id	person_id
0	tt0285252	nm0899854
1	tt0438973	nm0175726
2	tt0438973	nm1802864
3	tt0462036	nm1940585
4	tt0835418	nm0310087

In [17]:
```
persons_info.head()
```

Out[17]:

	person_id	primary_name	birth_year	death_year	primary_profession
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manager,producer
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_department
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneous,actor,writer
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,art_department
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,set_decorator

In [18]:
```
main_info.head()
```

Out[18]:

	movie_id	ordering	person_id	category	job	characters
0	tt0111414	1	nm0246005	actor	None	["The Man"]
1	tt0111414	2	nm0398271	director	None	None
2	tt0111414	3	nm3739909	producer	producer	None
3	tt0323808	10	nm0059247	editor	None	None
4	tt0323808	1	nm3579312	actress	None	["Beth Boothby"]

Data Cleaning and Engineering

Profitable Genres

In [19]:
```
#using datetime method to change the format of release date
movies_budget['release_date'] = pd.to_datetime(movies_budget['release_date'])
```

In [20]:
```
movies_budget['year'] = movies_budget['release_date'].dt.year
```

In [21]:
```
#adding a column 'year' and converting it to integer type
movies_budget['year'] = movies_budget['year'].astype(np.int64)
```

In [22]:
```
movies_budget.head()
```

Out[22]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	year
0	1	2009-12-18	Avatar	$425,000,000	$760,507,625	$2,776,345,279	2009
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875	2011
2	3	2019-06-07	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350	2019
3	4	2015-05-01	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963	2015
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747	2017

In [23]:
```
movies_info.head()
```

Out[23]:

	movie_id	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 [24]:
```
#Joining two pandas tables(movies_info1, movies_budget) using pandas merge method
profitable_genres = pd.merge(movies_info, movies_budget, left_on = ['primary_title', 'start_year'],
                    right_on = ['movie', 'year'], how='inner')
```

In [25]:
```
#importing warnings to hide warning comments
import warnings
warnings.filterwarnings('ignore')
```

In [26]:
```
#adding a column ww_gross with doing some data cleaning
profitable_genres['ww_gross'] = profitable_genres['worldwide_gross'].str.replace('$',
                                '').str.replace(',','')
```

In [27]:
```
profitable_genres['ww_gross'] = profitable_genres['ww_gross'].astype('int')
```

In [28]:
```
#adding a column genre_list with doing some data cleaning
profitable_genres['genre_list'] = profitable_genres['genres'].str.split(',')
```

In [29]:
```
#using explode method to separating out each genre
profitable_genres = profitable_genres.explode('genre_list')
```

Production Cost

In [30]:
```
#adding a column production_amount with doing some data cleaning to
#calculate average production cost for each genre
profitable_genres['production_cost'] = profitable_genres['production_budget'].str.replace('$',
                                         '').str.replace(',','')
```

In [31]:
```
profitable_genres['production_cost'] = profitable_genres['production_cost'].astype('int')
```

Directors and Writers with Good ratings

In [32]:
```
#joining two tables to get each director's movies
good_talents = """
SELECT movie_id, primary_title, person_id, category
FROM movie_basics
JOIN principals
    USING(movie_id)
JOIN persons
    USING(person_id)
ORDER BY category
;
"""
good_talents = pd.read_sql(good_talents, conn)
```

In [33]:
```
#joining two tables to ratings with matching directors
good_talents = pd.merge(good_talents, ratings_info, left_on = ['movie_id'],
                        right_on = ['movie_id'], how='inner')
```

In [34]:
```
good_talents = pd.merge(good_talents, persons_info, left_on = ['person_id'],
                        right_on = ['person_id'], how='inner')
```

In [35]:
```
good_talents.head()
```

Out[35]:

	movie_id	primary_title	person_id	category	averagerating	numvotes	primary_name	birth_year	death_year	
0	tt0063540	Sunghursh	nm0474801	actor	7.0	77	Dilip Kumar	1922.0	NaN	
1	tt0063540	Sunghursh	nm0474876	actor	7.0	77	Sanjeev Kumar	1938.0	1985.0	
2	tt0063540	Sunghursh	nm0756379	actor	7.0	77	Balraj Sahni	1913.0	1973.0	
3	tt0063540	Sunghursh	nm0904537	actress	7.0	77	Vyjayanthimala	1933.0	NaN	actress,music_depar
4	tt0063540	Sunghursh	nm0006210	composer	7.0	77	Naushad	1919.0	2006.0	composer,soundtrac

In [36]:
```
good_directors = good_talents[good_talents.category == 'director']
```

In [37]:

```
#dropping all dead directors
good_directors = good_directors[good_directors['death_year'].isna()]
good_directors.head()
```

Out[37]:

	movie_id	primary_title	person_id	category	averagerating	numvotes	primary_name	birth_year	death_year	primary_profes
29	tt1767372	She's Funny That Way	nm0000953	director	6.1	22179	Peter Bogdanovich	1939.0	NaN	actor,director,w
132	tt0100275	The Wandering Soap Opera	nm0765384	director	6.5	119	Valeria Sarmiento	1948.0	NaN	editor,director,w
133	tt1928329	Lines of Wellington	nm0765384	director	6.2	1235	Valeria Sarmiento	1948.0	NaN	editor,director,w
134	tt7490368	The Black Book	nm0765384	director	5.4	69	Valeria Sarmiento	1948.0	NaN	editor,director,w
326	tt0146592	Pál Adrienn	nm1030585	director	6.8	451	Ágnes Kocsis	1971.0	NaN	director,writer,prod

In [38]:

```
good_writers = good_talents[good_talents.category == 'writer']
```

In [39]:
```
#dropping all dead writers
good_writers = good_writers[good_writers['death_year'].isna()]
good_writers.head()
```

Out[39]:

	movie_id	primary_title	person_id	category	averagerating	numvotes	primary_name	birth_year	death_year	prim
7	tt0063540	Sunghursh	nm0347899	writer	7.0	77	Gulzar	1936.0	NaN	music_department,v
8	tt0069204	Sabse Bada Sukh	nm0347899	writer	6.1	13	Gulzar	1936.0	NaN	music_department,v
9	tt0357717	Haar Jeet	nm0347899	writer	5.1	9	Gulzar	1936.0	NaN	music_department,v
10	tt1946280	Noukadubi	nm0347899	writer	7.6	626	Gulzar	1936.0	NaN	music_department,v
11	tt2063745	Kya Dilli Kya Lahore	nm0347899	writer	7.5	1741	Gulzar	1936.0	NaN	music_department,v

Data Analysis

Profitable Genres

Most genres' world wide gross fall short of those at top of graph. Three genres stand out among many genres. Animation, Adventure, Sci-Fi are most noticeable.

In [40]:
```
profitable_genres.head()
```

Out[40]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	id	release_date	movie	productior
0	tt0249516	Foodfight!	Foodfight!	2012	91.0	Action,Animation,Comedy	26	2012-12-31	Foodfight!	$45
0	tt0249516	Foodfight!	Foodfight!	2012	91.0	Action,Animation,Comedy	26	2012-12-31	Foodfight!	$45
0	tt0249516	Foodfight!	Foodfight!	2012	91.0	Action,Animation,Comedy	26	2012-12-31	Foodfight!	$45
1	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	37	2013-12-25	The Secret Life of Walter Mitty	$91
1	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	37	2013-12-25	The Secret Life of Walter Mitty	$91

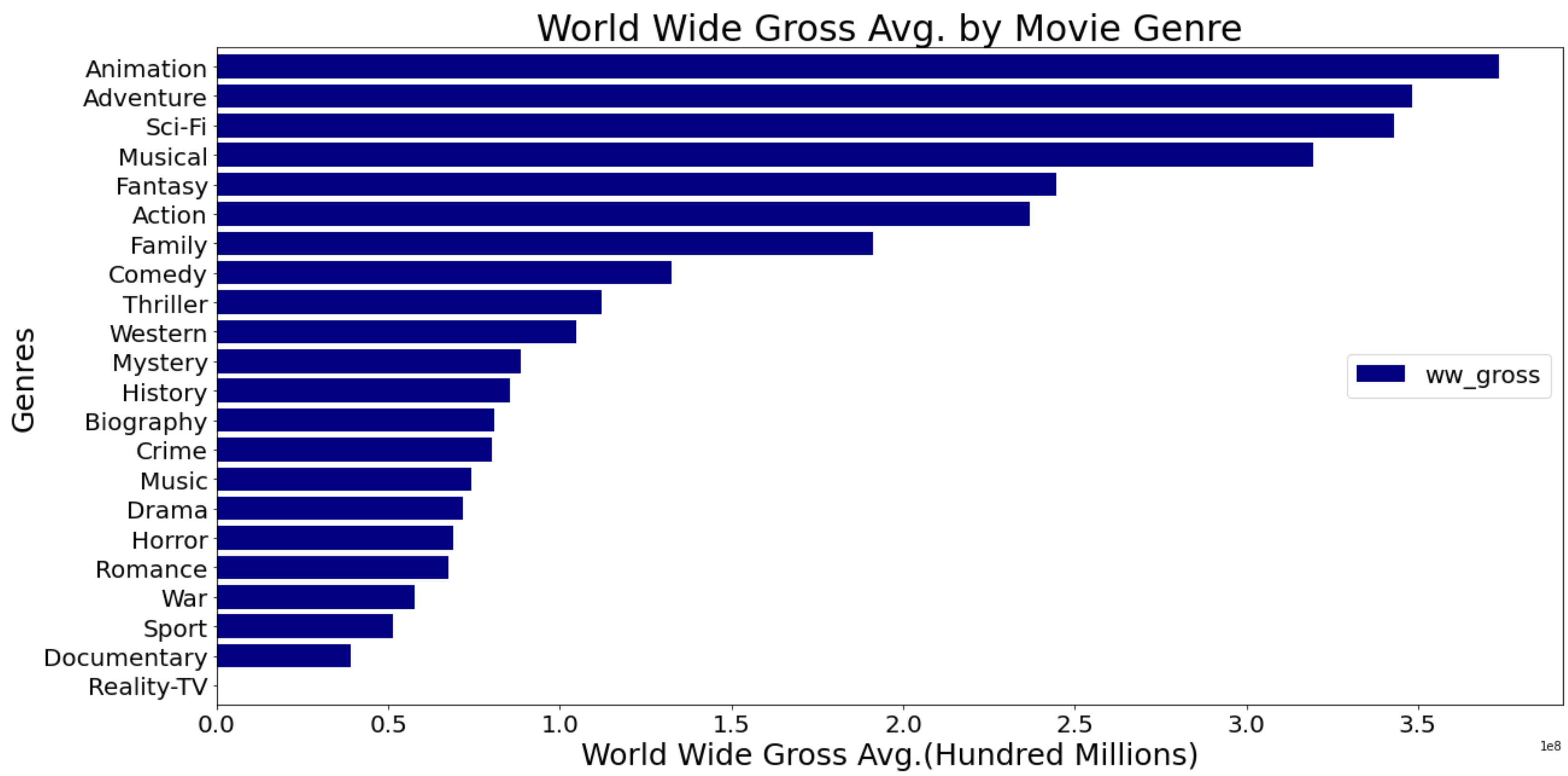
In [41]:
```
#using groupby method to get the average world wide gross of each genre
profitable_genres_avg = profitable_genres.groupby('genre_list').mean()['ww_gross'].reset_index()
```

In [42]:
```
profitable_genres_avg = profitable_genres_avg.sort_values(['ww_gross'], ascending=True)
```

In [43]:
```
profitable_genres_avg = profitable_genres_avg.set_index('genre_list')
```

In [44]:

```
#display in horizontal bar graph the size of average world wide gross of each genre
profitable_genres_avg.plot(figsize=(20,10),kind='barh', color='navy', width=.8)
profitable_genres_avg.sort_values('ww_gross',inplace=False)
plt.legend(fontsize = 20, loc='center right')
plt.title('World Wide Gross Avg. by Movie Genre', fontsize='30')
plt.xlabel('World Wide Gross Avg.(Hundred Millions)', fontsize=25)
plt.ylabel('Genres', fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```



Production Cost by Genre

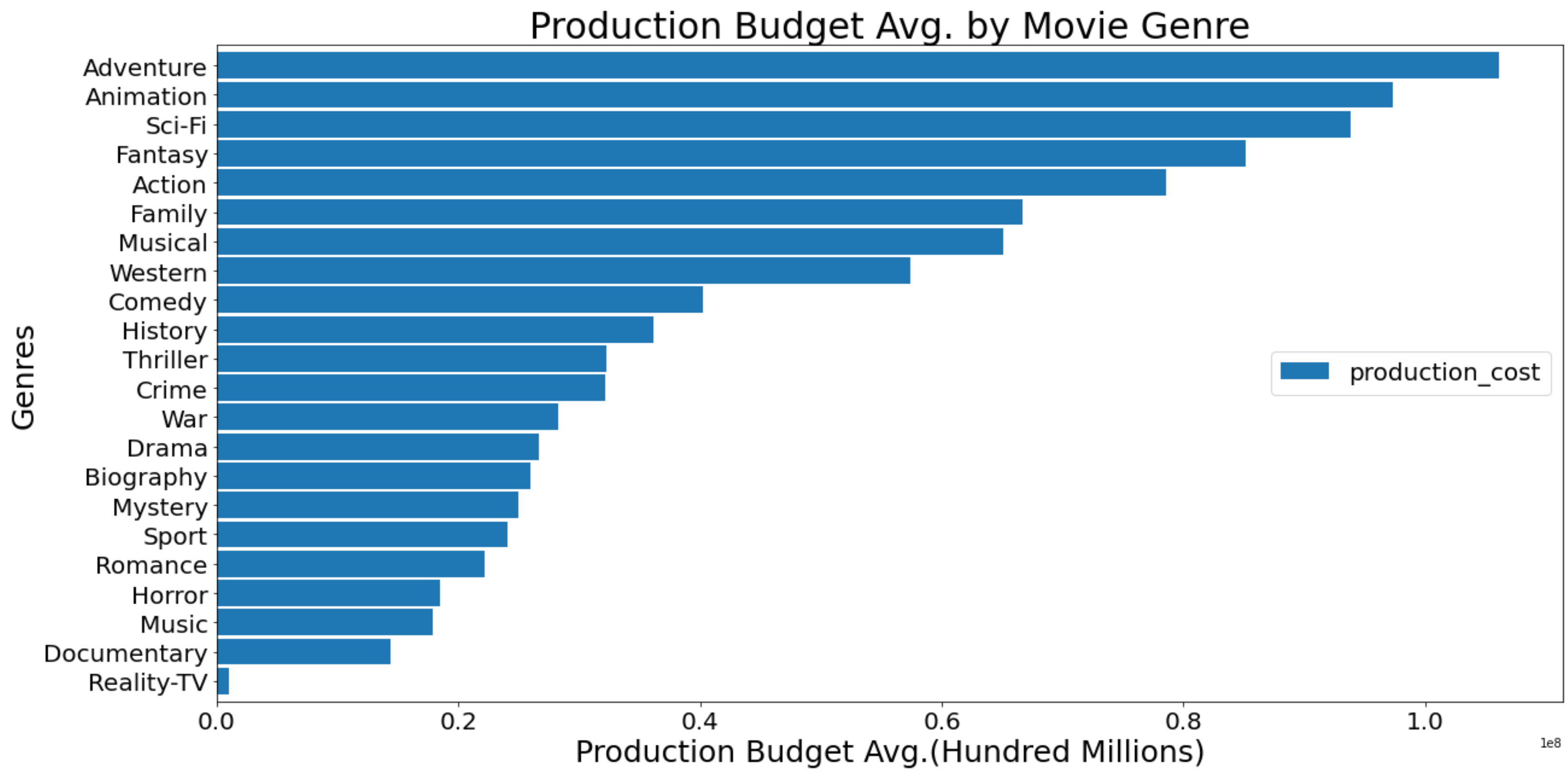
Production cost is another factor that affects how a movie could ultimately made. By looking at these costs, the conclusion could be made that more gross a movie has generally it is more expensive to make.

In [45]:
```
#using groupby method to get average production cost for each genre
pcost_avg = profitable_genres.groupby('genre_list').mean()['production_cost'].reset_index()
```

In [46]:
```
pcost_avg = pcost_avg.sort_values(['production_cost'], ascending=True)
```

In [47]:
```
pcost_avg = pcost_avg.set_index('genre_list')
```

In [48]:
```
pcost_avg.plot(figsize=(20,10),kind='barh', width=.9)
pcost_avg.sort_values('production_cost',inplace=False)
plt.legend(fontsize = 20, loc='center right')
plt.title('Production Budget Avg. by Movie Genre', fontsize='30')
plt.xlabel('Production Budget Avg.(Hundred Millions)', fontsize=25)
plt.ylabel('Genres', fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```



Directors and Writers with Good ratings

Movies with good directors and writers achieve higher ratings. Having these directors and writers in a movie will produce better quality film.

In [49]:
```
#using groupby method to get average rating for each director
directors_avg = good_directors.groupby('primary_name').mean()['averagerating'].reset_index()
```

In [50]:
```
directors_avg = directors_avg.sort_values(['averagerating'], ascending=False)
```

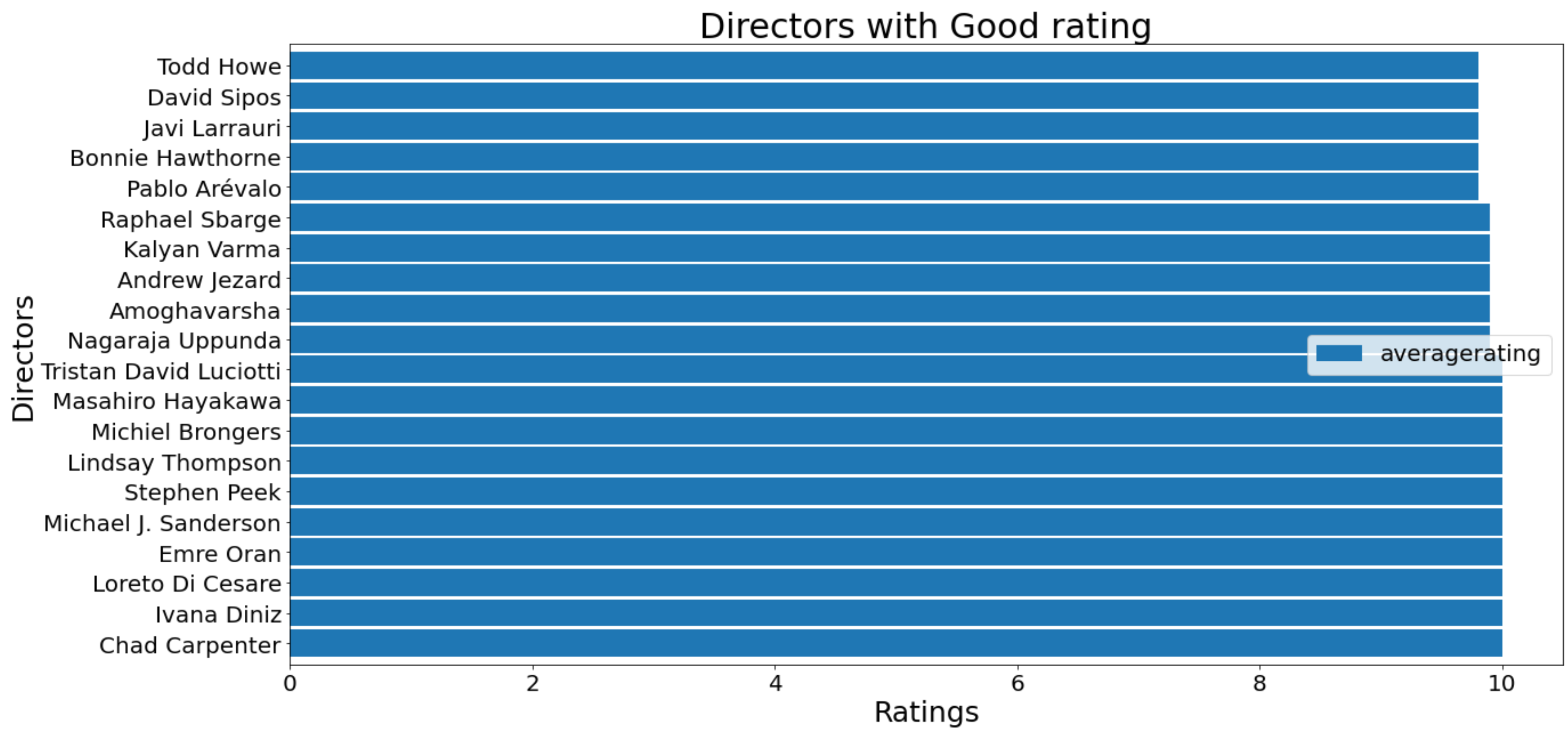
In [51]:
```
#choosing directors with rating of 9.8 or above and selecting first 20 directors
directors_avg = directors_avg.head(20)
directors_avg
```

Out[51]:

	primary_name	averagerating
7753	Chad Carpenter	10.0
19493	Ivana Diniz	10.0
28494	Loreto Di Cesare	10.0
13854	Emre Oran	10.0
32657	Michael J. Sanderson	10.0
45257	Stephen Peek	10.0
28232	Lindsay Thompson	10.0
33154	Michiel Brongers	10.0
31194	Masahiro Hayakawa	10.0
48283	Tristan David Luciotti	10.0
34457	Nagaraja Uppunda	9.9
2374	Amoghavarsha	9.9
2889	Andrew Jezard	9.9
25344	Kalyan Varma	9.9
39312	Raphael Sbarge	9.9
36485	Pablo Arévalo	9.8
6091	Bonnie Hawthorne	9.8
20956	Javi Larrauri	9.8
11475	David Sipos	9.8
47596	Todd Howe	9.8

In [52]:
```
directors_avg = directors_avg.set_index('primary_name')
```

In [53]:
```
directors_avg.plot(figsize=(20,10),kind='barh', width=.9)
directors_avg.sort_values('averagerating',inplace=False)
plt.legend(fontsize = 20, loc='center right')
plt.title('Directors with Good rating', fontsize='30')
plt.xlabel('Ratings', fontsize=25)
plt.ylabel('Directors', fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```



In [54]:
```
#using groupby method to get average rating for each writer and just picking first 20 writers
writers_avg = good_writers.groupby('primary_name').mean()['averagerating'].reset_index()
```

In [55]:
```
writers_avg = writers_avg.sort_values(['averagerating'], ascending=False)
```

In [56]:
```
writers_avg = writers_avg.head(20)
writers_avg
```

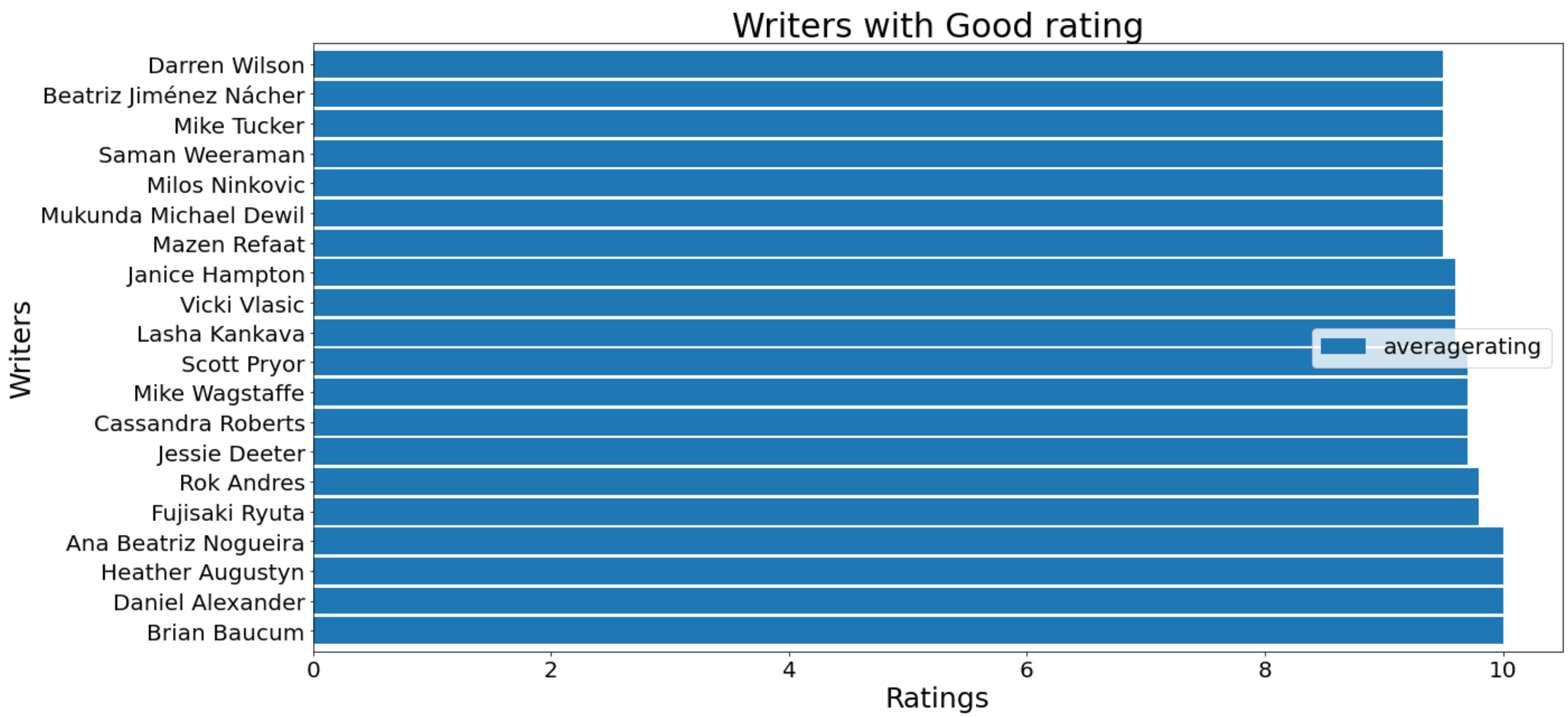
Out[56]:

	primary_name	averagerating
4501	Brian Baucum	10.0
7023	Daniel Alexander	10.0
12307	Heather Augustyn	10.0
1704	Ana Beatriz Nogueira	10.0
10688	Fujisaki Ryuta	9.8
28180	Rok Andres	9.8
15117	Jessie Deeter	9.7
5244	Cassandra Roberts	9.7
23058	Mike Wagstaffe	9.7
29555	Scott Pryor	9.7
18806	Lasha Kankava	9.6
33775	Vicki Vlasic	9.6
14139	Janice Hampton	9.6
22073	Mazen Refaat	9.5
23596	Mukunda Michael Dewil	9.5
23155	Milos Ninkovic	9.5
28975	Saman Weeraman	9.5
23055	Mike Tucker	9.5
3581	Beatriz Jiménez Nácher	9.5
7355	Darren Wilson	9.5

In [57]:
```
writers_avg = writers_avg.set_index('primary_name')
```

In [58]:

```
writers_avg.plot(figsize=(20,10),kind='barh', width=.9)
writers_avg.sort_values('averagerating',inplace=False)
plt.legend(fontsize = 20, loc='center right')
plt.title('Writers with Good rating', fontsize='30')
plt.xlabel('Ratings', fontsize=25)
plt.ylabel('Writers', fontsize=25)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```



Conclusions

This analysis enables to make three recommendations to produce profitable and quality movies for Microsoft Movie Studios.

- In first movie production taking what's already popular is a crucial step to ensure financial gain, which enables Microsoft Movie Studio to continue making quality movies well into the future. In this regard **a recommendation is given to choose 3 most world wide grossing genres: Animation, Adventure and Sci-Fi.**
- First movie production is about making initial investment to make a quality movie. In this regard a recommendation is given to choose a director of good quality. Directorship is one of the most important aspects of movie making. **Any of the 20 directors shown would work out since their**
rating is at or above 9.8
- Again first movie production is about making initial investment to make a quality movie. In this regard a recommendation is given to choose a writer of good quality. **Any of the 20 writers shown would work out since their rating is at or above 9.5**

Next Steps

Further analysis of certain aspects of movie making and trend could help to create potential top grossing films.

- By observation of recent film data it is noticeable in last 10 years multiple Action/Adventure/Sci-Fi genre films placed as top grossing movies. This might inform that with recent lightning speed technological advancement, there might be a trend building where people throng toward Sci-Fi genre movies.
- While top grossing genres did well in the box office it is an obstacle to overcome in that those same genres also have high production cost. This is an element that should be considered in producing such genre movies.