Final Project Submission

Please fill out:

- Student name:
- Student pace: self paced / part time / full time
- Scheduled project review date/time:
- Instructor name:
- Blog post URL:

Student Name: Cassidy Exum

Student Pace: Flex/part time, 40 week pace

Schedule Project Review Date/Time:

Instructor Name: Morgan Jones

Blog Post URL: https://exumexaminesdata.blogspot.com/2022/06/data-analysis-and-solar-energy.html

Project Overview

Business Understanding

My goal is to analyze movie metrics such as rating, profit, runtime, budget, etc and come up with three proposals for microsofts new studio. Let's break down how I'm going to do this. Using Pandas and Sqlite I'll read in the different data sets. It seems like we have a plethora of data so I'm not too concerned about cleaning it, I can most likely just get rid of any bad rows of data. Making movies for a large studio is a business decision, it's not for the love of film, so we are going to proceed with that intnet in mind and keep budget and profit at the forefront. And now for analysis and proposals:

Our measure of success in this project will be profit and/or rating

Proposal 1 - What genres perform the best? What genre movie should we make?

For the first proposal I'm going to go through the data find the best genres. Genre is fairly general so it will leave us with plenty of movie options to make once we find genres that are successful.

Proposal 2 - Recent Success

For the second proposal we will look at only the last 2 years and figure out what movies have done the best. With this we will have a second approach that will follow recent trends.

Proposal 3 - Select one of the three reccomended directors

In [1]:
```
#import whats needed, set matplotlib inline
import pandas as pd
import numpy as np
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sea
import matplotlib.ticker as mtick
%matplotlib inline
```

In [2]:
```
#Acess the data -

file_path_1 = 'zippedData/bom.movie_gross.csv.gz'
file_path_2 = 'zippedData/rt.movie_info.tsv.gz'
file_path_3 = 'zippedData/rt.reviews.tsv.gz'
file_path_4 = 'zippedData/tmdb.movies.csv.gz'
file_path_5 = 'zippedData/tn.movie_budgets.csv.gz'
db = 'zippedData/im.db'

bom_movie_gross = pd.read_csv(file_path_1)
rt_movie_info = pd.read_csv(file_path_2, sep="\t", index_col = 0)
#rt_reviews = pd.read_csv(file_path_3, sep="\t")
#rt_review not working. Have enough data, we can ignore.
tmdb_movies = pd.read_csv(file_path_4, index_col = 0)
tn_movie_budgets = pd.read_csv(file_path_5, index_col = 0)

conn = sqlite3.connect(db)
```

In [3]:
```
#data scouting
bom_movie_gross.head()
```

Out[3]:

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

In [4]:
```
#data scouting
rt_movie_info.head()
```

Out[4]:

	synopsis	rating	genre	director	writer	theater_date	dvd_(
id							
1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	Sep 2

id	synopsis	rating	genre	director	writer	theater_date	dvd_(
3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	J. 2
5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	Ap 2
6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 1994	Au(
7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	NaN	

In [5]:
```
#data scouting
tmdb_movies.head()
```

Out[5]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_av
0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	

In [6]:
```
#data scouting
tn_movie_budgets.head()
```

Out[6]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279

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

In [7]:

```
#data scouting
pd.read_sql("""
SELECT *
FROM movie_basics
""", conn).head()
```

Out[7]:

	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

Ok, we generally know what the data looks like now.

The rt_movie_info table looks a bit useless. We have no title for the movies and thats essentially the primary key connecting all of these files/db's together. All the others seem good and useful so we will ignore that one for now.

Lets start by using the IMBD sqlite db to find the highest rated movies

In [8]:

```
# obtain highest rated films
highest_rated = pd.read_sql("""
SELECT original_title, genres, averagerating, numvotes
FROM movie_basics
JOIN movie_ratings
    USING(movie_id)
ORDER BY averagerating DESC
""", conn)
```

In [9]:

```
highest_rated.head(20)
```

Out[9]:

	original_title	genres	averagerating	numvotes

	original_title	genres	averagerating	numvotes
0	Exteriores: Mulheres Brasileiras na Diplomacia	Documentary	10.0	5
1	The Dark Knight: The Ballad of the N Word	Comedy,Drama	10.0	5
2	Freeing Bernie Baran	Crime,Documentary	10.0	5
3	Hercule contre Hermès	Documentary	10.0	5
4	I Was Born Yesterday!	Documentary	10.0	6
5	Dog Days in the Heartland	Drama	10.0	5
6	Revolution Food	Documentary	10.0	8
7	Fly High: Story of the Disc Dog	Documentary	10.0	7
8	All Around Us	Documentary	10.0	6
9	Atlas Mountain: Barbary Macaques - Childcaring...	Documentary	10.0	5
10	Requiem voor een Boom	Documentary	10.0	5
11	A Dedicated Life: Phoebe Brand Beyond the Group	Documentary	10.0	5
12	Ellis Island: The Making of a Master Race in A...	Documentary,History	10.0	6
13	Calamity Kevin	Adventure,Comedy	10.0	6
14	Pick It Up! - Ska in the '90s	Documentary	10.0	5
15	Renegade	Documentary	10.0	20
16	Gini Helida Kathe	Drama	9.9	417
17	The Wedding Present: Something Left Behind	Documentary	9.9	8
18	LA Foodways	Documentary	9.9	8
19	Moscow we will lose	Documentary	9.9	18

Honestly, not very useful... Way too many documentaries, and way too few votes. Lets look at the numvotes column and determine some number of votes that we feel a movie must have to be included

In [10]:

```
print(highest_rated['numvotes'].mean())
# Lets arbitrarily choose 2000
```

```
3523.6621669194105
```

In [11]:

```
# Average rating was 3523, lets use 2000
highest_rated = pd.read_sql("""
SELECT original_title, genres, averagerating, numvotes
FROM movie_basics
JOIN movie_ratings
    USING(movie_id)
WHERE numvotes > 2000
ORDER BY averagerating DESC
""", conn)
highest_rated.head(20)
```

Out[11]:

original_title	genres	averagerating	numvotes

	original_title	genres	averagerating	numvotes
0	Once Upon a Time ... in Hollywood	Comedy,Drama	9.7	5600
1	Ekvtime: Man of God	Biography,Drama,History	9.6	2604
2	Aloko Udapadi	Drama,History	9.5	6509
3	Peranbu	Drama	9.4	9629
4	Dag II	Action,Drama,War	9.3	100568
5	Aynabaji	Crime,Mystery,Thriller	9.3	18470
6	Wheels	Drama	9.3	17308
7	Natsamrat	Drama,Family	9.2	4297
8	C/o Kancharapalem	Drama	9.2	2195
9	CM101MMXI Fundamentals	Comedy,Documentary	9.2	41560
10	On vam ne Dimon	Documentary	9.2	2721
11	A Man Called Ahok	Drama	9.1	4162
12	Oggatonama	Drama	9.1	2973
13	Pariyerum Perumal	Drama	9.0	4854
14	Yowis Ben	Comedy,Drama	9.0	2992
15	Tylko nie mów nikomu	Documentary	8.9	2111
16	Godhi Banna Sadharana Mykattu	Drama,Family	8.9	2001
17	A Billion Lives	Documentary,History,News	8.9	2715
18	O.J.: Made in America	Biography,Crime,Documentary	8.9	14946
19	Burn the Stage: The Movie	Documentary,Music	8.8	2067

I think we also want to make the region US / language ENG and eliminate pure Documentaries

In [12]:
```
#Set region to US
highest_rated = pd.read_sql("""
SELECT original_title, genres, averagerating, numvotes, region
FROM movie_basics
INNER JOIN movie_ratings
    USING(movie_id)
INNER JOIN movie_akas
    USING(movie_id)
WHERE numvotes > 2000 AND genres NOT LIKE "%Documentary%" AND region = "US"
ORDER BY averagerating DESC
""", conn)
highest_rated.drop_duplicates(subset = "original_title", inplace = True)
```

In [13]:
```
highest_rated.head(20)
```

Out[13]:

	original_title	genres	averagerating	numvotes	region
0	Once Upon a Time ... in Hollywood	Comedy,Drama	9.7	5600	US
2	Peranbu	Drama	9.4	9629	US
3	Wheels	Drama	9.3	17308	US

	original_title	genres	averagerating	numvotes	region
4	Inception	Action,Adventure,Sci-Fi	8.8	1841066	US
8	Kill Bill: The Whole Bloody Affair	Action,Crime,Thriller	8.8	3406	US
9	Avengers: Endgame	Action,Adventure,Sci-Fi	8.8	441135	US
12	96	Drama,Romance	8.8	10903	US
13	Super Deluxe	Action,Crime,Drama	8.8	2254	US
14	Mahanati	Biography,Drama	8.7	6917	US
15	Interstellar	Adventure,Drama,Sci-Fi	8.6	1299334	US
19	Uri: The Surgical Strike	Action,Drama,War	8.6	30292	US
20	Yatra	Biography,Drama	8.6	2913	US
21	Rangasthalam	Action,Drama	8.6	15407	US
22	An Hour to Kill	Action,Comedy,Horror	8.6	2302	US
27	Intouchables	Biography,Comedy,Drama	8.5	677343	US
28	Whiplash	Drama,Music	8.5	616916	US
29	Thani Oruvan	Action,Crime,Thriller	8.5	13747	US
30	Capharnaüm	Drama	8.5	20215	US
31	Dhuruvangal Pathinaaru	Action,Crime,Mystery	8.5	8560	US
32	Avengers: Infinity War	Action,Adventure,Sci-Fi	8.5	670926	US

Now we have some actual data. Lets start looking at the profit info and start relating that to genres and things like that. Because I limited the previous table to US release, I'm going to limit our gross to domestic as well.

In [14]:
```
bom_movie_gross.sort_values('domestic_gross')
```

Out[14]:

	title	studio	domestic_gross	foreign_gross	year
1476	Storage 24	Magn.	100.0	NaN	2013
2321	The Chambermaid	FM	300.0	NaN	2015
2756	News From Planet Mars	KL	300.0	NaN	2016
2757	Satanic	Magn.	300.0	NaN	2016
1018	Apartment 143	Magn.	400.0	426000	2012
...	...	...	...	...	...
1975	Surprise - Journey To The West	AR	NaN	49600000	2015
2392	Finding Mr. Right 2	CL	NaN	114700000	2016
2468	Solace	LGP	NaN	22400000	2016
2595	Viral	W/Dim.	NaN	552000	2016
2825	Secret Superstar	NaN	NaN	122000000	2017

3387 rows × 5 columns

Ok now we are running into NaN issues. Lets figure out if we can remove them.

In [15]:
```
# Check NAN
bom_movie_gross.isna().sum()
```

Out[15]:
```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

In [16]:
```
#Drop foreign gross
bom_movie_gross.drop('foreign_gross', axis = 1, inplace = True)
```

In [17]:
```
bom_movie_gross = bom_movie_gross.dropna()
bom_movie_gross = bom_movie_gross.sort_values('domestic_gross', ascending=False)
```

In [18]:
```
bom_movie_gross
```

Out[18]:

	title	studio	domestic_gross	year
1872	Star Wars: The Force Awakens	BV	936700000.0	2015
3080	Black Panther	BV	700100000.0	2018
3079	Avengers: Infinity War	BV	678800000.0	2018
1873	Jurassic World	Uni.	652300000.0	2015
727	Marvel's The Avengers	BV	623400000.0	2012
...	...	...	...	...
1018	Apartment 143	Magn.	400.0	2012
2757	Satanic	Magn.	300.0	2016
2756	News From Planet Mars	KL	300.0	2016
2321	The Chambermaid	FM	300.0	2015
1476	Storage 24	Magn.	100.0	2013

3356 rows × 4 columns

In [19]:
```
gross_and_rating_df = bom_movie_gross.merge(highest_rated, how='inner', left_on=
```

In [20]:
```
gross_and_rating_df.head(20)
```

Out[20]:

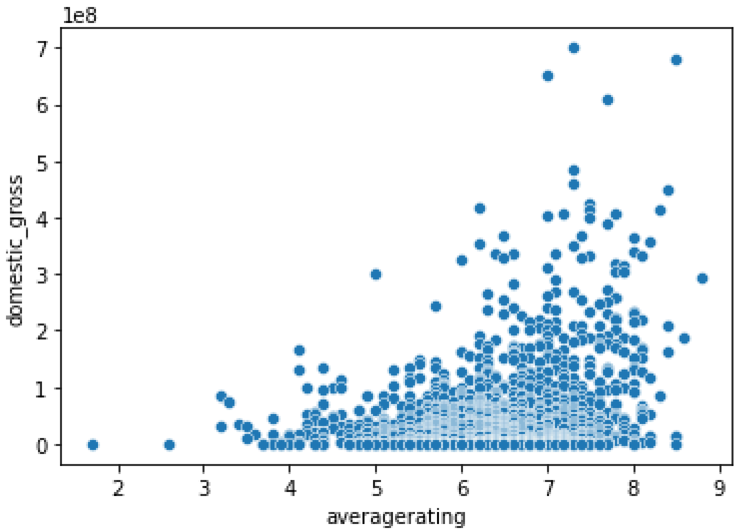
	title	studio	domestic_gross	year	original_title	genres	averagera
0	Black Panther	BV	700100000.0	2018	Black Panther	Action,Adventure,Sci-Fi	
1	Avengers: Infinity War	BV	678800000.0	2018	Avengers: Infinity War	Action,Adventure,Sci-Fi	
2	Jurassic World	Uni.	652300000.0	2015	Jurassic World	Action,Adventure,Sci-Fi	

	title	studio	domestic_gross	year	original_title	genres	averagera
3	Incredibles 2	BV	608600000.0	2018	Incredibles 2	Action,Adventure,Animation	
4	Finding Dory	BV	486300000.0	2016	Finding Dory	Adventure,Animation,Comedy	
5	Avengers: Age of Ultron	BV	459000000.0	2015	Avengers: Age of Ultron	Action,Adventure,Sci-Fi	
6	The Dark Knight Rises	WB	448100000.0	2012	The Dark Knight Rises	Action,Thriller	
7	The Hunger Games: Catching Fire	LGF	424700000.0	2013	The Hunger Games: Catching Fire	Action,Adventure,Sci-Fi	
8	Jurassic World: Fallen Kingdom	Uni.	417700000.0	2018	Jurassic World: Fallen Kingdom	Action,Adventure,Sci-Fi	
9	Toy Story 3	BV	415000000.0	2010	Toy Story 3	Adventure,Animation,Comedy	
10	Wonder Woman	WB	412600000.0	2017	Wonder Woman	Action,Adventure,Fantasy	
11	Captain America: Civil War	BV	408100000.0	2016	Captain America: Civil War	Action,Adventure,Sci-Fi	
12	The Hunger Games	LGF	408000000.0	2012	The Hunger Games	Action,Adventure,Sci-Fi	
13	Jumanji: Welcome to the Jungle	Sony	404500000.0	2017	Jumanji: Welcome to the Jungle	Action,Adventure,Comedy	
14	Frozen	BV	400700000.0	2013	Frozen	Adventure,Animation,Comedy	
15	Guardians of the Galaxy Vol. 2	BV	389800000.0	2017	Guardians of the Galaxy Vol. 2	Action,Adventure,Comedy	
16	The Secret Life of Pets	Uni.	368400000.0	2016	The Secret Life of Pets	Adventure,Animation,Comedy	
17	Despicable Me 2	Uni.	368100000.0	2013	Despicable Me 2	Adventure,Animation,Comedy	
18	Deadpool	Fox	363100000.0	2016	Deadpool	Action,Adventure,Comedy	
19	Inside Out	BV	356500000.0	2015	Inside Out	Adventure,Animation,Comedy	

In [21]:
```
y = gross_and_rating_df['domestic_gross']
x = gross_and_rating_df['averagerating']
```

```
sea.scatterplot(x, y);
```

```
/Users/cassidyexum/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seab
orn/_decorators.py:36: FutureWarning: Pass the following variables as keyword ar
gs: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error
or misinterpretation.
  warnings.warn(
```



In [22]:
```
genres_df = gross_and_rating_df['genres']
genres_df = genres_df.str.split(',')
genres_list_all = genres_df.tolist()
genres_dict_all = {}
for x in genres_list_all:
    for y in x:
        if y not in genres_dict_all:
            genres_dict_all[y] = 1
        else:
            genres_dict_all[y] += 1
```

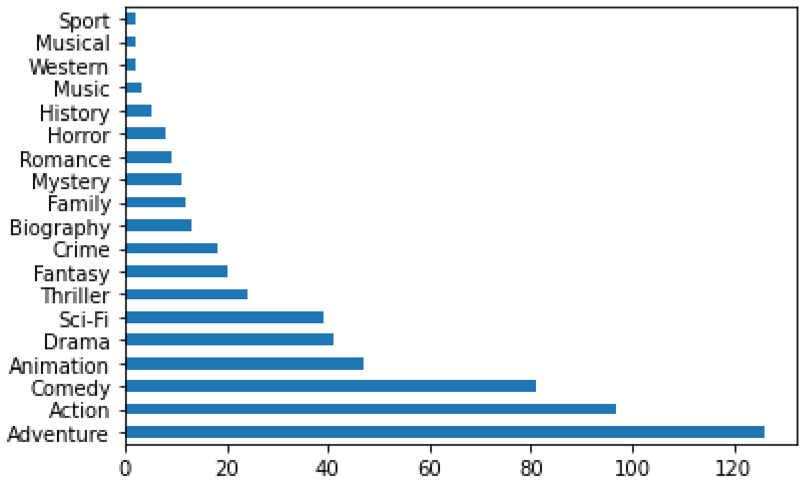
In [23]:
```
genres_df = gross_and_rating_df['genres'].iloc[:200]
genres_df = genres_df.str.split(',')
genres_list = genres_df.tolist()
genres_dict = {}
for x in genres_list:
    for y in x:
        if y not in genres_dict:
            genres_dict[y] = 1
        else:
            genres_dict[y] += 1
```

In [24]:
```
genres_series = pd.Series(genres_dict)
genres_series.sort_values(ascending = False, inplace=True)
genres_series
```

Out[24]:
```
Adventure    126
Action        97
Comedy        81
Animation     47
Drama         41
Sci-Fi        39
Thriller      24
Fantasy       20
```

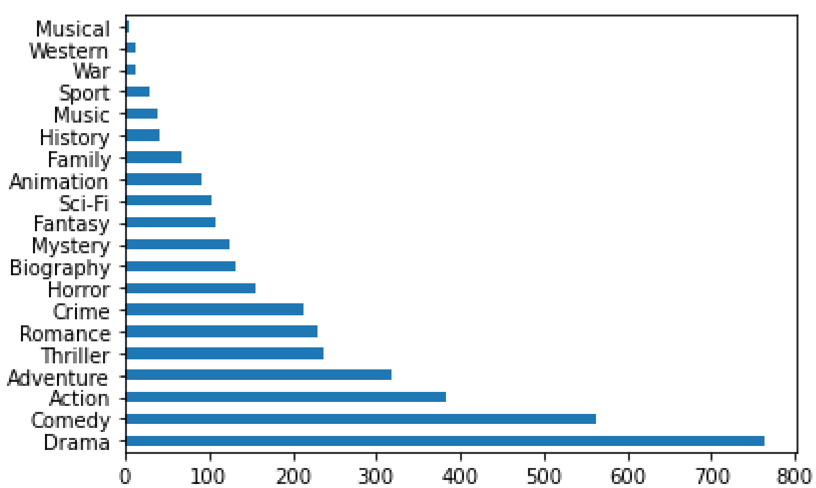
```
Crime            18
Biography        13
Family           12
Mystery          11
Romance           9
Horror            8
History           5
Music             3
Western           2
Musical           2
Sport             2
dtype: int64
```

In [25]:
```
genres_series.plot.barh();
#bar chart of top 200
```



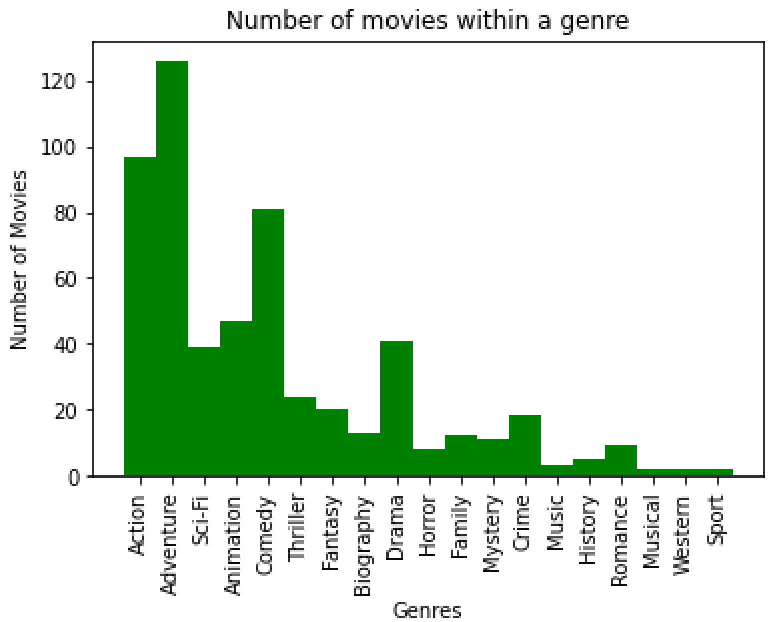
In [26]:
```
genres_series_all = pd.Series(genres_dict_all)
genres_series_all.sort_values(ascending = False, inplace=True)
genres_series_all.plot.barh()
#bar chart of all
```

Out[26]: <AxesSubplot:>



In [27]:
```
#scuffed bar chart
width = 1
plt.bar(genres_dict.keys(), genres_dict.values(), width, color='g')
plt.title('Number of movies within a genre')
plt.xlabel('Genres')
```

```python
plt.ylabel('Number of Movies')
plt.xticks(rotation = 90);
```



Reccomendation 1 - Make an action, adventure, scifi, comedy film

So it looks like rating plays some effect on domestic gross, but not too much of an effect. Something that we can see from the top 20 we posted earlier is the similarity in genres. Action, adventure, comedy, and SciFi are all constantly being repeated. So lets make this our first reccomendation. Make a movie that is an action, adventure, scifi, and comedy.

Recent Trends

Lets combine some tables and find films in the last 2 years (our data ends in 2018 so we will use 2016, 2017, and 2018) that have done exceptionally well.

tn_movie_budgets will be a great table to use for this.

In [28]:
```python
tn_movie_budgets.head()
```

Out[28]:

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

In [29]:
```
tn_movie_budgets_recent = tn_movie_budgets.sort_values('release_date', ascending
```

In [30]:
```
#converting to column to datetime and then recasting that column
release_date = pd.to_datetime(tn_movie_budgets_recent['release_date'])
```

In [31]:
```
tn_movie_budgets_recent['release_date'] = release_date
```

In [32]:
```
tn_movie_budgets_recent.sort_values('release_date', ascending = False, inplace =
```

In [33]:
```
tn_movie_budgets_recent.head(25)
```

Out[33]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
6	2020-12-31	Hannibal the Conqueror	$50,000,000	$0	$0
95	2020-12-31	Moonfall	$150,000,000	$0	$0
36	2020-02-21	Call of the Wild	$82,000,000	$0	$0
30	2019-12-31	Reagan	$25,000,000	$0	$0
81	2019-12-31	Army of the Dead	$90,000,000	$0	$0
72	2019-12-31	355	$75,000,000	$0	$0
13	2019-12-31	Rogue City	$13,000,000	$0	$0
16	2019-12-31	Eli	$11,000,000	$0	$0
44	2019-12-31	Down Under Cover	$40,000,000	$0	$0
8	2019-11-22	The Rhythm Section	$50,000,000	$0	$0
53	2019-11-08	Midway	$59,500,000	$0	$0
7	2019-11-08	Arctic Dogs	$50,000,000	$0	$0
30	2019-09-30	Unhinged	$29,000,000	$0	$0
9	2019-09-20	Ad Astra	$49,800,000	$0	$0
43	2019-09-13	The Goldfinch	$40,000,000	$0	$0
71	2019-08-30	PLAYMOBIL	$75,000,000	$0	$0
64	2019-08-14	Blinded by the Light	$15,000,000	$0	$0
16	2019-07-12	Crawl	$17,000,000	$0	$0
48	2019-06-21	Burn Your Maps	$8,000,000	$0	$0
39	2019-06-21	Kursk	$40,000,000	$0	$4,212,799
42	2019-06-14	Men in Black: International	$110,000,000	$3,100,000	$3,100,000
98	2019-06-14	Shaft	$30,000,000	$600,000	$600,000
81	2019-06-07	The Secret Life of Pets 2	$80,000,000	$63,795,655	$113,351,496
3	2019-06-07	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
35	2019-06-07	Late Night	$4,000,000	$246,305	$246,305

Everything before 2019-06-21 is too recent and has $0 as the listed gross. Lets drop those.

In [34]:
```
tn_movie_budgets_recent.drop(index=tn_movie_budgets_recent.index[:20], axis=0, i
```

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

Out[35]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
42	2019-06-14	Men in Black: International	$110,000,000	$3,100,000	$3,100,000
98	2019-06-14	Shaft	$30,000,000	$600,000	$600,000
3	2019-06-07	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
35	2019-06-07	Late Night	$4,000,000	$246,305	$246,305
25	2019-05-31	Godzilla: King of the Monsters	$170,000,000	$85,576,941	$299,276,941

In [36]:
```
#obtain all movies not documentaries and more than 2000 votes
recent_imdb = pd.read_sql("""
SELECT original_title, start_year, genres, averagerating, numvotes
FROM movie_basics
JOIN movie_ratings
    USING(movie_id)
WHERE numvotes > 2000 AND genres NOT LIKE "%Documentary%"
ORDER BY start_year DESC, averagerating DESC
""", conn)
recent_imdb.head(20)
```

Out[36]:

	original_title	start_year	genres	averagerating	numvotes
0	Once Upon a Time ... in Hollywood	2019	Comedy,Drama	9.7	5600
1	Avengers: Endgame	2019	Action,Adventure,Sci-Fi	8.8	441135
2	Super Deluxe	2019	Action,Crime,Drama	8.8	2254
3	Uri: The Surgical Strike	2019	Action,Drama,War	8.6	30292
4	Yatra	2019	Biography,Drama	8.6	2913
5	The Tashkent Files	2019	Drama,Mystery,Thriller	8.4	3175
6	Gully Boy	2019	Drama,Music	8.3	17483
7	Badla	2019	Crime,Drama,Mystery	8.1	9988
8	John Wick: Chapter 3 - Parabellum	2019	Action,Crime,Thriller	8.0	81568
9	Maharshi	2019	Action,Drama	8.0	2733
10	Balkanskiy rubezh	2019	Action,War	7.8	2958
11	Rocketman	2019	Biography,Drama,Music	7.7	24266
12	Lucifer	2019	Action,Crime,Drama	7.7	4412
13	Kesari	2019	Action,Drama,History	7.7	7557

	original_title	start_year	genres	averagerating	numvotes
14	Dolor y gloria	2019	Drama	7.7	2802
15	Madhura Raja	2019	Action,Comedy,Drama	7.7	2522
16	How to Train Your Dragon: The Hidden World	2019	Action,Adventure,Animation	7.6	60769
17	Once Upon a Time in London	2019	Crime	7.6	2752
18	The Boy Who Harnessed the Wind	2019	Drama	7.6	10725
19	Alita: Battle Angel	2019	Action,Adventure,Sci-Fi	7.5	88207

In [37]:
```
recents_merged_df = tn_movie_budgets_recent.merge(recent_imdb, how='inner', left
```

In [38]:
```
recents_merged_df.sort_values('release_date', ascending = False, inplace = True)
recents_merged_df.head(20)
```

Out[38]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	original_title
0	2019-06-07	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350	Dark Phoenix
1	2019-05-31	Godzilla: King of the Monsters	$170,000,000	$85,576,941	$299,276,941	Godzilla: King of the Monsters
2	2019-05-10	The Professor and the Madman	$25,000,000	$0	$5,227,233	The Professor and the Madman
3	2019-05-03	Long Shot	$40,000,000	$30,202,860	$43,711,031	Long Shot
4	2019-04-12	Hellboy	$50,000,000	$21,903,748	$40,725,492	Hellboy
6	2019-04-05	Pet Sematary	$21,000,000	$54,724,696	$109,501,146	Pet Sematary
8	2019-04-05	Shazam!	$85,000,000	$139,606,856	$362,899,733	Shazam!
9	2019-03-29	Unplanned	$6,000,000	$18,107,621	$18,107,621	Unplanned
10	2019-03-29	Dumbo	$170,000,000	$113,883,318	$345,004,422	Dumbo
11	2019-03-22	Us	$20,000,000	$175,006,930	$254,210,310	Us
12	2019-03-15	Five Feet Apart	$7,000,000	$45,729,221	$80,504,421	Five Feet Apart
13	2019-03-15	Captive State	$25,000,000	$5,958,315	$8,993,300	Captive State
14	2019-03-15	Wonder Park	$100,000,000	$45,216,793	$115,149,422	Wonder Park
15	2019-03-08	Captain Marvel	$175,000,000	$426,525,952	$1,123,061,550	Captain Marvel

	release_date	movie	production_budget	domestic_gross	worldwide_gross	original_title
16	2019-02-22	How to Train Your Dragon: The Hidden World	$129,000,000	$160,791,800	$519,258,283	How to Train Your Dragon: The Hidden World
17	2019-02-14	Alita: Battle Angel	$170,000,000	$85,710,210	$402,976,036	Alita: Battle Angel
18	2019-02-13	Happy Death Day 2U	$9,000,000	$28,051,045	$64,179,495	Happy Death Day 2U
19	2019-02-08	Cold Pursuit	$60,000,000	$32,138,862	$62,599,159	Cold Pursuit
20	2019-02-08	What Men Want	$20,000,000	$54,611,903	$69,911,903	What Men Want
21	2019-02-01	Velvet Buzzsaw	$21,000,000	$0	$0	Velvet Buzzsaw

Lets find the most profitable movies (domestic gross - budget) of the last 3 years (roughly 300 movies)

In [39]:
```
recents_df = recents_merged_df.drop(index=recents_merged_df.index[300:], axis=0)
```

In [40]:
```
recents_df
```

Out[40]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	original_ti
0	2019-06-07	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350	Dark Phoe
1	2019-05-31	Godzilla: King of the Monsters	$170,000,000	$85,576,941	$299,276,941	Godzi King of Monst
2	2019-05-10	The Professor and the Madman	$25,000,000	$0	$5,227,233	T Profes and Madn
3	2019-05-03	Long Shot	$40,000,000	$30,202,860	$43,711,031	Long S
4	2019-04-12	Hellboy	$50,000,000	$21,903,748	$40,725,492	Hellk
...	...	...	...	...	...	
304	2016-07-29	Jason Bourne	$120,000,000	$162,192,920	$416,168,316	Jas Bou
307	2016-07-22	Ice Age: Collision Course	$105,000,000	$64,063,008	$402,156,682	Ice A Collis Cou
306	2016-07-22	Lights Out	$5,000,000	$67,268,835	$148,806,510	Lights (
305	2016-07-22	Star Trek Beyond	$185,000,000	$158,848,340	$335,802,233	Star T Bey(
308	2016-07-15	Ghostbusters	$144,000,000	$128,350,574	$229,008,658	Ghostbust

300 rows × 10 columns

In [41]:
```
#remove $ and commas
recents_df['domestic_gross'] = recents_df['domestic_gross'].str.strip('$')
recents_df['production_budget'] = recents_df['production_budget'].str.strip('$')

recents_df['domestic_gross'] = recents_df['domestic_gross'].str.replace(',','')
recents_df['production_budget'] = recents_df['production_budget'].str.replace(',
```

In [42]:
```
recents_df
```

Out[42]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	original_ti
0	2019-06-07	Dark Phoenix	350000000	42762350	$149,762,350	Dark Phoe
1	2019-05-31	Godzilla: King of the Monsters	170000000	85576941	$299,276,941	Godzi King of Monst
2	2019-05-10	The Professor and the Madman	25000000	0	$5,227,233	T Profes and Madn
3	2019-05-03	Long Shot	40000000	30202860	$43,711,031	Long S
4	2019-04-12	Hellboy	50000000	21903748	$40,725,492	Hellk
...	...	...	...	...	...	
304	2016-07-29	Jason Bourne	120000000	162192920	$416,168,316	Jas Bou
307	2016-07-22	Ice Age: Collision Course	105000000	64063008	$402,156,682	Ice A Collis Cou
306	2016-07-22	Lights Out	5000000	67268835	$148,806,510	Lights (
305	2016-07-22	Star Trek Beyond	185000000	158848340	$335,802,233	Star T Beyc
308	2016-07-15	Ghostbusters	144000000	128350574	$229,008,658	Ghostbust

300 rows × 10 columns

In [43]:
```
#converting to INT
recents_df['domestic_gross'] = recents_df['domestic_gross'].astype(int)
recents_df['production_budget'] = recents_df['production_budget'].astype(int)
```

In [44]:
```
recents_df['profit'] = recents_df['domestic_gross'] - recents_df['production_bud
```

In [45]:
```
recents_df.sort_values('profit', ascending = False, inplace = True)
```

In [46]:
```
recents_df.head(20)
```

Out[46]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	original_ti
117	2018-02-16	Black Panther	200000000	700059566	$1,348,258,224	Bla Pantl

	release_date	movie	production_budget	domestic_gross	worldwide_gross	original_ti
217	2017-03-17	Beauty and the Beast	160000000	504014165	$1,259,199,706	Beauty a the Be
133	2017-12-20	Jumanji: Welcome to the Jungle	90000000	404508916	$964,496,193	Juma Welcome the Jun
164	2017-09-08	It	35000000	327481748	$697,457,969	
188	2017-06-02	Wonder Woman	150000000	412563408	$821,133,378	Won Wor
15	2019-03-08	Captain Marvel	175000000	426525952	$1,123,061,550	Capt Mar
90	2018-05-18	Deadpool 2	110000000	324591735	$786,680,557	Deadpoc
249	2016-12-21	Sing	75000000	270329045	$634,454,789	Si
223	2017-02-24	Get Out	5000000	176040665	$255,367,951	Get (
99	2018-04-06	A Quiet Place	17000000	188024361	$334,522,294	A Quiet Pla
48	2018-11-02	Bohemian Rhapsody	55000000	216303339	$894,985,342	Bohem Rhapsc
177	2017-07-07	Spider-Man: Homecoming	175000000	334201140	$880,166,350	Spider-M Homecomi
11	2019-03-22	Us	20000000	175006930	$254,210,310	
302	2016-08-05	Suicide Squad	175000000	325100054	$746,059,887	Suici Squ
67	2018-08-15	Crazy Rich Asians	30000000	174532921	$238,099,711	Crazy R Asia
143	2017-11-03	Thor: Ragnarok	180000000	315058289	$846,980,024	Th Ragnai
235	2017-01-20	Split	5000000	138141585	$278,964,806	S
251	2016-12-09	La La Land	20000000	151101803	$426,351,163	La La La
74	2018-07-13	Hotel Transylvania 3: Summer Vacation	65000000	167500092	$527,079,962	Hc Transylva 3: Sumn Vacat
221	2017-03-03	Logan	127000000	226277068	$615,461,394	Log

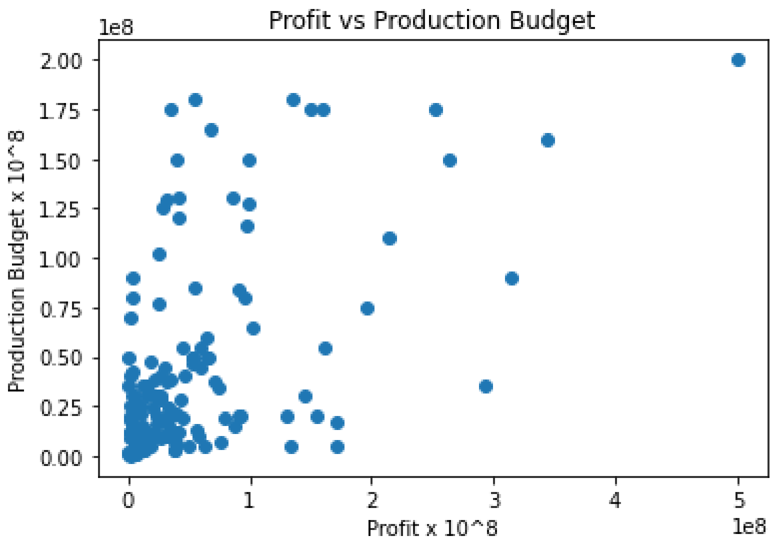
In [47]:
```
#remove all the negatives

recents_pos_df = recents_df[recents_df['profit']>0]
```

In [48]:
```
#bad plot
x = recents_pos_df['profit']
y = recents_pos_df['production_budget']

plt.scatter(x, y)
plt.title('Profit vs Production Budget')
plt.xlabel('Profit x 10^8')
```

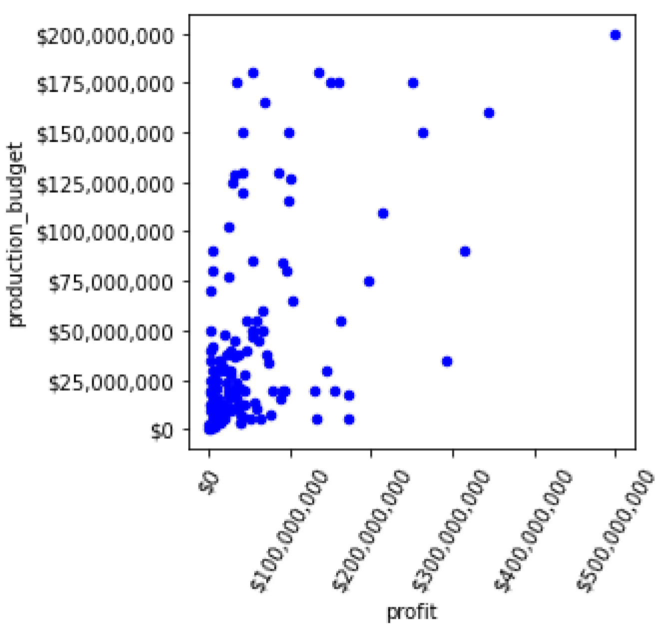
```
plt.ylabel('Production Budget x 10^8')
plt.show()
```



In [49]:
```
#good plot
fig, ax = plt.subplots(1, 1, figsize=(4, 4))
recents_pos_df.plot(kind='scatter', x='profit', y='production_budget',
        color='blue', ax=ax)

fmt = '${x:,.0f}'
tick = mtick.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)
ax.xaxis.set_major_formatter(tick)
plt.xticks(rotation=60)

plt.show()
```



So there isnt exactly a relation between budget and profit. But have some budget generally gets you some profit. So lets take the average budget and reccomend that as the target budget for our film studio.

In [50]:
```
recents_pos_df['production_budget'].mean()
```

Out[50]: 42959589.04109589

Reccomendation 2 - Target Budget = $43,000,000

People - Who can we get on a film that will draw a crowd?

How can we get this info?

- IMDB Sql database has a persons tables with a person_id and primary name.

One we have the person what do we want?

- lets generate a list of films for each person_id and determine with person_id has the highest average ratings of all the films they've been on.

In [51]:
```
persons_df = pd.read_sql("""
SELECT p.primary_name,
        mr.averagerating,
        COUNT(DISTINCT mb.primary_title) as num_movies
FROM directors as d
JOIN persons as p
    USING(person_id)
JOIN principals
    USING(person_id)
JOIN movie_basics as mb
    USING(movie_id)
JOIN movie_ratings as mr
    USING(movie_id)
WHERE numvotes > 3000
GROUP BY p.primary_name
HAVING num_movies > 5
""", conn)
```

In [52]:
```
persons_df
```

Out[52]:

	primary_name	averagerating	num_movies
0	A.R. Murugadoss	6.8	7
1	Adam Wingard	5.3	8
2	Alex Gibney	7.3	8
3	Anurag Kashyap	8.1	9
4	Baltasar Kormákur	6.6	6
5	Ben Wheatley	5.6	6
6	Clint Eastwood	6.5	7
7	Darren Lynn Bousman	6.4	7
8	David Gordon Green	6.4	8
9	Denis Villeneuve	8.3	6
10	Frank D'Angelo	6.6	6

	primary_name	averagerating	num_movies
11	François Ozon	6.4	6
12	Hirokazu Koreeda	7.4	6
13	James Wan	7.2	6
14	Jon M. Chu	6.2	7
15	Kevin Macdonald	7.2	6
16	Lasse Hallström	6.8	7
17	Luc Besson	6.5	6
18	Michael Winterbottom	6.6	6
19	Mike Flanagan	5.8	6
20	Noah Baumbach	6.1	6
21	Ridley Scott	8.0	7
22	Rob Reiner	6.5	6
23	Rohit Shetty	5.5	8
24	Ron Howard	5.3	6
25	Sarik Andreasyan	4.5	6
26	Stephen Frears	6.8	6
27	Steven C. Miller	4.9	7
28	Steven Soderbergh	7.0	8
29	Steven Spielberg	7.4	7
30	Takashi Miike	7.4	6
31	Tim Burton	6.5	6
32	Tim Story	7.4	7
33	Tyler Perry	4.7	12
34	Werner Herzog	7.2	6
35	Woody Allen	6.3	8
36	Álex de la Iglesia	6.6	6

In [53]:

```
persons_df.drop_duplicates(inplace = True)
persons_df.sort_values('averagerating', ascending = False, inplace=True)
persons_df.head()
```

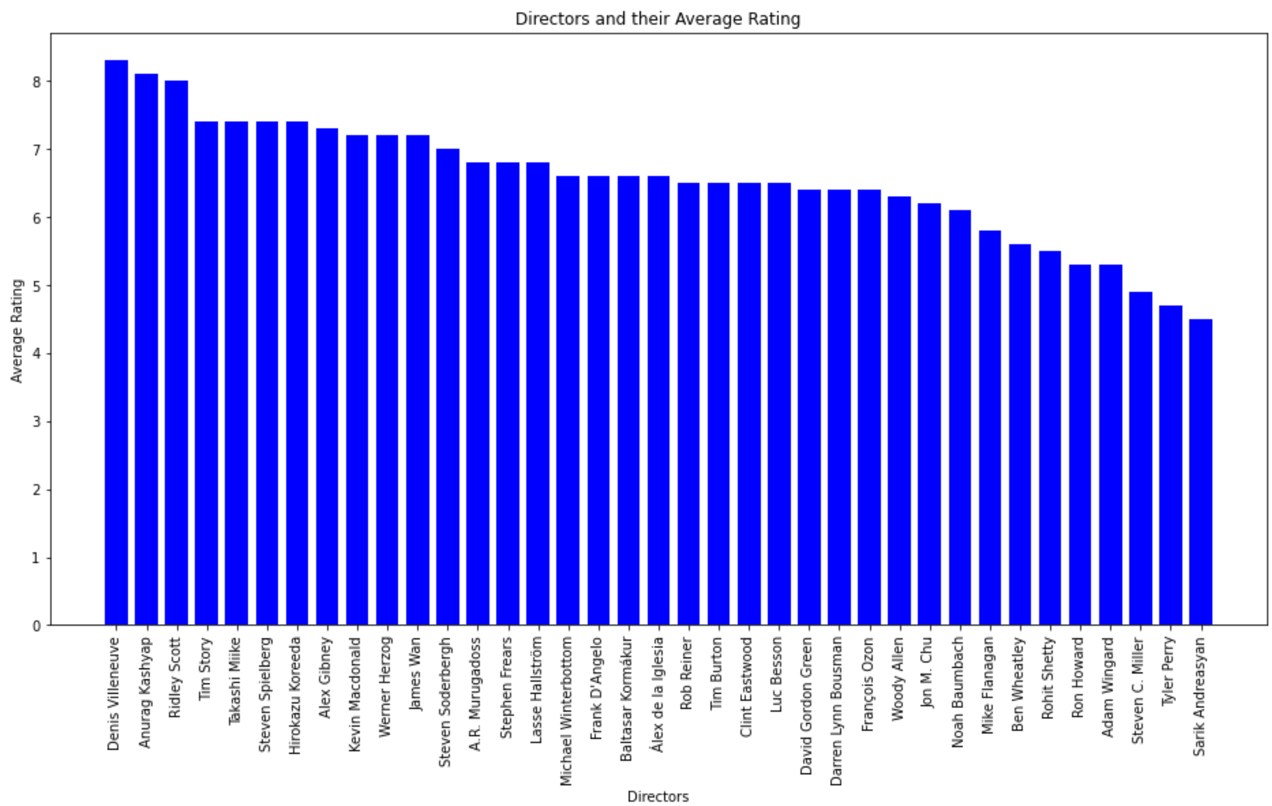
Out[53]:

	primary_name	averagerating	num_movies
9	Denis Villeneuve	8.3	6
3	Anurag Kashyap	8.1	9
21	Ridley Scott	8.0	7
32	Tim Story	7.4	7
30	Takashi Miike	7.4	6

In order to get rid of outliers we limited our results directors with more than 5 movies, and those movies needed to have more than 3000 votes. Then we started those directors by the average rating of their films. This is the list we should use to select a director for any film our studio wishes to make. Don't choose arbitrarily though, some directors are from non US regions, and directors typically stay within genres. So it important to do research here as well.

There is no relationship between the number of movies and the average rating

In [54]:
```
x = persons_df['primary_name']
y = persons_df['averagerating']
plt.figure(figsize=(16, 8))
plt.bar(x, y, width = .75, color='b')
plt.title('Directors and their Average Rating')
plt.xlabel('Directors')
plt.ylabel('Average Rating')
plt.xticks(rotation = 90);
```



Lets drop most of the list and for now, use one of the three directors at the top of this list: Denis Villeneuve, Anurag Kashyap, or Ridley Scott

Reccomendation 3 - Use one of these three directors: Denis Villeneuve, Anurag Kashyap, or Ridley Scott

Conclusion

We used rating and profit as our measures of success to determine what features like genre, budget, and director to use.

Reccomendation 1- Make an action, adventure, comedy film

Reccomendation 2- Use $43,000,000 as your target budget

Reccomendation 3- Select either Denis Villeneuve, Anurag Kashyap, or Ridley Scott as the director