Final Project Submission

Please fill out:

- Student name: Jacob M. Hansen
- Student pace: part time
- Scheduled project review date/time: 11AM, July 15th, 2022
- Instructor name: Morgan Jones
- Blog post URL:https://medium.com/@jacob.m.hansen/the-care-principle-in-data-science-e0a700d0c4f6 (https://medium.com/@jacob.m.hansen/the-care-principle-in-data-science-e0a700d0c4f6)

Objective

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they don't know anything about creating movies.

- Explore what types of films are currently doing the best at the box office.
- Translate those findings into actionable insights that the head of Microsoft's new movie studio can use to help decide what type of films to create.

Questions for Data Analysis

- Over the course of the past decade which movies have had the largest box office success in terms of both domestic and worldwide sales?
- What is the relationship between production costs and expected worldwide gross revenue?
- What can Microsoft expect to pay in movie production costs?
- What are some attributes of movie success? Consider audience attention spans when it comes to movie length. Also consider the time of year of when movies are frequently released.

Workflow Process

- Identify and import needed data science packages and libraries
- Read and examine the provided databases. Identify the data types and structure of the information in each database.
- Begin exploratory analysis by cleaning, filtering and grouping data based upon initial questions.
- Further explore, clean, and filter data. Use aggregate functions to identify trends and patterns.

- Create Exploratory data visualizations to further identify trends and patterns.
- Create Explanatory data visualization to communicate findings.

In [1]:
```
#import packages and libraries

import pandas as pd
import numpy as np

import matplotlib as plt
import seaborn as sns

#Use 'magic' inline function to display in notebook
%matplotlib inline
```

Read Datasets, Identify Content and Structure

Read the provided datasets: Box Office Mojo, Movie Budgets, and IMDB. Create a dataframe for each dataset using the Pandas library. Identify the content and structure of each dataset. Identify the length of data and data types. Examine for missing data and make an initial plan to manage any missing values.

Exploratory Analysis of the Box Office Mojo data

Use Pandas

In [2]:
```
# Read the Box Office Mojo dataset

bom_df = pd.read_csv('bom.movie_gross.csv')
```

In [3]:
```
type(bom_df)
```

Out[3]:
```
pandas.core.frame.DataFrame
```

In [4]:
```
# Identify content structure and length

bom_df.shape
```

Out[4]:
```
(3387, 5)
```

In [5]:
```
# Explore the content

bom_df
```

Out[5]:

	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

	title	studio	domestic_gross	foreign_gross	year
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
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

In [6]:
```
# Identify missing data.

bom_df.isna().sum()
```

Out[6]:
```
title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
```

In [7]:
```
# Identify datatypes related to missing data.

bom_df.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 [8]:
```
# Examine the fundamental stats of domestic gross

bom_df.describe().astype(int)
```

Out[8]:

	domestic_gross	year
count	3359	3387
mean	28745845	2013

	domestic_gross	year
std	66982498	2
min	100	2010
25%	120000	2012
50%	1400000	2014
75%	27900000	2016

In [9]:
```
# Examine the year 2010 for trends. Drop missing data to do so.

bom_df_2010 = bom_df[(bom_df['year'] == 2010)]

bom_df_2010 = bom_df_2010.dropna()
```

In [10]:
```
# Examine domestic gross for 2010 compare to overall BOM dataset.

bom_df_2010.describe().astype(int)
```

Out[10]:

	domestic_gross	year
count	308	308
mean	32963448	2010
std	60668552	0
min	800	2010
25%	280250	2010
50%	4000000	2010
75%	40400000	2010
max	415000000	2010

In [11]:
```
#Identify year 2010 domestic gross mean and median

print("Domestic 2010 gross box office mean",
      bom_df_2010['domestic_gross'].mean())

print("Domestic 2010 gross box office median",
      bom_df_2010['domestic_gross'].median())
```

```
Domestic 2010 gross box office mean 32963448.373376623
Domestic 2010 gross box office median 4000000.0
```

In [12]:
```
# Examine another year for comparison to 2010. Choose 2018 to see diff
# time period.

bom_df_2018 = bom_df[(bom_df['year'] == 2018)]

bom_df_2018.dropna()
```

Out[12]:

	title	studio	domestic_gross	foreign_gross	year
3079	Avengers: Infinity War	BV	678800000.0	1,369.5	2018
3080	Black Panther	BV	700100000.0	646900000	2018
3081	Jurassic World: Fallen Kingdom	Uni.	417700000.0	891800000	2018
3082	Incredibles 2	BV	608600000.0	634200000	2018
3083	Aquaman	WB	335100000.0	812700000	2018
...	...	...	...	...	...
3275	I Still See You	LGF	1400.0	1500000	2018
3286	The Catcher Was a Spy	IFC	725000.0	229000	2018
3309	Time Freak	Grindstone	10000.0	256000	2018
3342	Reign of Judges: Title of Liberty - Concept Short	Darin Southa	93200.0	5200	2018
3353	Antonio Lopez 1970: Sex Fashion & Disco	FM	43200.0	30000	2018

173 rows × 5 columns

In [13]:
```
bom_df_2018.describe().astype(int)
```

Out[13]:

	domestic_gross	year
count	308	308
mean	36010421	2018
std	85733961	0
min	1300	2018
25%	175250	2018
50%	2700000	2018
75%	35950000	2018
max	700100000	2018

In [14]:
```
#Identify year 2018 domestic gross mean and median. Compare to 2010 me

print("Domestic 2018 gross box office mean",
      bom_df_2018['domestic_gross'].mean())

print("Domestic 2018 gross box office median",
```

```
      bom_df_2018['domestic_gross'].median())
```

```
Domestic 2018 gross box office mean 36010421.75
Domestic 2018 gross box office median 2700000.0
```

Box Office Mojo Dataset summary

The Box Office Mojo dataset contains data about movie titles, which studio released the film, and gross revenue broken out into domestic and foreign returns. The movie release years range from 2010 to 2018. The median domestic gross for all years was USD 1.4 million, but this can vary quite a bit year to year. Many of the datatypes are not numeric and there are several columns, particularly foreign gross, that are missing data.

The Numbers dataset

Read and explore the movie budgets database. Identify the data structure, data types and whether there is any missing data. Identify relevant data that could be joined with the Box Office Mojo dataset.

Use Pandas

In [15]:
```
budgets_df = pd.read_csv('tn.movie_budgets.csv')
```

In [16]:
```
budgets_df[:10]
```

Out[16]:

	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
5	6	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	$306,000,000	$936,662,225	$2,053,311,220
6	7	Apr 27, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,134,200
7	8	May 24, 2007	Pirates of the Caribbean: At Worldâs End	$300,000,000	$309,420,425	$963,420,425
8	9	Nov 17, 2017	Justice League	$300,000,000	$229,024,295	$655,945,209
9	10	Nov 6, 2015	Spectre	$300,000,000	$200,074,175	$879,620,923

In [17]:
```
budgets_df.tail()
```

Out[17]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

In [18]:
```
budgets_df.shape
```

Out[18]:
```
(5782, 6)
```

In [19]:
```
#Identify any missing data
budgets_df.isna().sum()
```

Out[19]:

```
id                    0
release_date          0
movie                 0
production_budget     0
domestic_gross        0
```

In [20]:
```
#Find data types
budgets_df.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
```

The Numbers database summary:

This df consists of 6 columns and 5782 rows, there is no missing data. Production budgets and gross revenue are currently stored as objects or strings, these will need to be changed to integers or floats to do any data aggregation. The movie column can serve as possible index id for a dataframe join with the bom_df for further analysis and data visualizations.

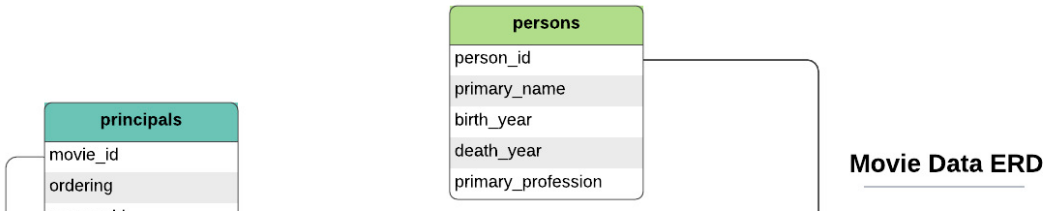
Explore the IMDB database.

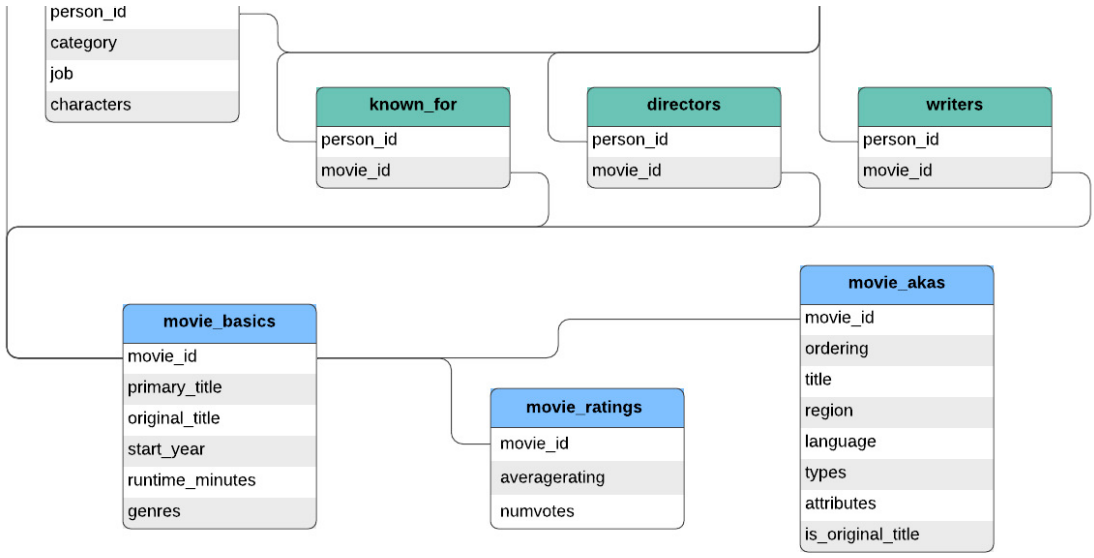
Explore the imdb tables and associated data for relevant information related to primary business questions. Identify primary and foreign keys to join, filter, group, and limit data. Create a dataframe that can be joined or merged with bom_df and budgets_df for further analysis and data visualizations.

Use sqlite3

In [21]:
```
import sqlite3
conn = sqlite3.connect('im.db')
```

In [22]:
```
#import the entity relational diagram (ERD) for this database
```





Read and Examine individual tables

In [23]:
```
#Examine and identify datatypes for movie_basics table

movie_basics_df = pd.read_sql("""SELECT * FROM movie_basics;""", conn)

movie_basics_df
```

Out[23]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genre
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Dram
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Dram
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Dram
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Dram
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantas
...	...	...	...	...	...	.
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Dram
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentar
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comed

	movie_id	primary_title	original_title	start_year	runtime_minutes	genre
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	Non
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentar

In [24]:
```
movie_basics_df.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 [25]:
```
#Examine and identify datatypes for movie_ratings table

movie_ratings_df = pd.read_sql("""SELECT * FROM movie_ratings;""", con

movie_ratings_df.head()
```

Out[25]:

	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 [26]:
```
movie_ratings_df.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 [27]:
```
# Examine and identify data types for movie_akas table
```

```
movie_akas_df = pd.read_sql("""SELECT * FROM movie_akas;""", conn)

movie_akas_df.head()
```

Out[27]:

	movie_id	ordering	title	region	language	types	attributes	is_original_title
0	tt0369610	10	Джурасик свят	BG	bg	None	None	0.0
1	tt0369610	11	Jurashikku warudo	JP	None	imdbDisplay	None	0.0
2	tt0369610	12	Jurassic World: O Mundo dos Dinossauros	BR	None	imdbDisplay	None	0.0
3	tt0369610	13	O Mundo dos Dinossauros	BR	None	None	short title	0.0
4	tt0369610	14	Jurassic World	FR	None	imdbDisplay	None	0.0

In [28]:
```
movie_akas_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 331703 entries, 0 to 331702
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   movie_id           331703 non-null  object 
 1   ordering           331703 non-null  int64  
 2   title              331703 non-null  object 
 3   region             278410 non-null  object 
 4   language           41715 non-null   object 
 5   types              168447 non-null  object 
 6   attributes         14925 non-null   object 
 7   is_original_title  331678 non-null  float64
dtypes: float64(1), int64(1), object(6)
memory usage: 20.2+ MB
```

In [29]:
```
# Merge individual imdb tables into one dataframe using movie_id

imdb_df = pd.merge(pd.merge(
    movie_basics_df, movie_ratings_df,on='movie_id'),
    movie_akas_df, on='movie_id').drop_duplicates(subset = 'movie_id')

imdb_df = imdb_df.reset_index(drop = True)

imdb_df
```

Out[29]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime

	movie_id	primary_title	original_title	start_year	runtime_minutes	
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biograph
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comed
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,
...	...	...	...	...	...	

In [30]:
```
imdb_df.isna().sum()
```

Out[30]:
```
movie_id                0
primary_title           0
original_title          0
start_year              0
runtime_minutes      6747
genres                640
averagerating           0
numvotes                0
ordering                0
title                   0
region              11444
language            63871
types               45395
attributes          67562
is_original_title       0
dtype: int64
```

IMDB dataset summary

This dataset contains a wide range of data. I examined three specific tables and joined them into one dataframe. The data provides details on movie categories such as runtime, genres, rating, and languages. The dataframe is the lengthiest with nearly 70,000 entries. there is a range of missing data in several categories. Some of these could be easily managed using the median or mode, but it is likely some data will need to be dropped to analyze efficiently. The imdb dataframe could be joined to the other datasets using the primary_title column.

Join or Merge Dataframes.

Join or merge the three dataframes from imdb, box office mojo, and production budgets into one dataframe for data cleaning, further analysis and data visualizations. Use the movie titles as primary and foreign keys.

Use Pandas

In [31]:
```
# Merge the box office mojo dataset and budget dataset

big_movie_df = pd.merge(bom_df, budgets_df, left_on='title', right_on=

big_movie_df.columns
```

Out[31]:
```
Index(['title', 'studio', 'domestic_gross_x', 'foreign_gross', 'year
', 'id',
       'release_date', 'movie', 'production_budget', 'domestic_gross_
y',
       'worldwide_gross'],
      dtype='object')
```

In [32]:
```
#drop column 'movie', keep 'title' as the primary key

big_movie_df.drop('movie', axis=1, inplace=True)
```

In [33]:
```
big_movie_df.head()
```

Out[33]:

	title	studio	domestic_gross_x	foreign_gross	year	id	release_date	production_budge
0	Toy Story 3	BV	415000000.0	652000000	2010	47	Jun 18, 2010	$200,000,00
1	Inception	WB	292600000.0	535700000	2010	38	Jul 16, 2010	$160,000,00
2	Shrek Forever After	P/DW	238700000.0	513900000	2010	27	May 21, 2010	$165,000,00
3	The Twilight Saga: Eclipse	Sum.	300500000.0	398000000	2010	53	Jun 30, 2010	$68,000,00
4	Iron Man 2	Par.	312400000.0	311500000	2010	15	May 7, 2010	$170,000,00

In [34]:
```
#Merge big_movie_df with imdb_df using title and primary_title

super_df = pd.merge(big_movie_df, imdb_df, left_on='title', right_on='

super_df.head()
```

Out[34]:

	title_x	studio	domestic_gross_x	foreign_gross	year	id	release_date	production_budge
0	Toy Story 3	BV	415000000.0	652000000	2010	47	Jun 18, 2010	$200,000,00
1	Inception	WB	292600000.0	535700000	2010	38	Jul 16, 2010	$160,000,00
2	Shrek Forever After	P/DW	238700000.0	513900000	2010	27	May 21, 2010	$165,000,00

	title_x	studio	domestic_gross_x	foreign_gross	year	id	release_date	production_budge
3	The Twilight Saga: Eclipse	Sum.	300500000.0	398000000	2010	53	Jun 30, 2010	$68,000,00
4	Iron Man 2	Par.	312400000.0	311500000	2010	15	May 7, 2010	$170,000,00

Clean Data.

Identify duplicate columns and drop based on datatype and missing data. For any remaining missing data convert NaN to either mean, median, or mode or consider keeping as a missing category. Consider carefully how much data to drop entirely.

In [35]:
```
#Identify datatypes

super_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1398 entries, 0 to 1397
Data columns (total 25 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   title_x            1398 non-null   object 
 1   studio             1398 non-null   object 
 2   domestic_gross_x   1397 non-null   float64
 3   foreign_gross      1202 non-null   object 
 4   year               1398 non-null   int64  
 5   id                 1398 non-null   int64  
 6   release_date       1398 non-null   object 
 7   production_budget  1398 non-null   object 
 8   domestic_gross_y   1398 non-null   object 
 9   worldwide_gross    1398 non-null   object 
 10  movie_id           1398 non-null   object 
 11  primary_title      1398 non-null   object 
 12  original_title     1398 non-null   object 
 13  start_year         1398 non-null   int64  
 14  runtime_minutes    1371 non-null   float64
 15  genres             1392 non-null   object 
 16  averagerating      1398 non-null   float64
 17  numvotes           1398 non-null   int64  
 18  ordering           1398 non-null   int64  
 19  title_y            1398 non-null   object 
 20  region             1299 non-null   object 
 21  language           193 non-null    object 
 22  types              933 non-null    object 
 23  attributes         56 non-null     object 
 24  is_original_title  1398 non-null   float64
dtypes: float64(4), int64(5), object(16)
memory usage: 284.0+ KB
```

In [36]:
```
#Identify missing data
```

In [36]:
```
super_df.isna().sum()
```

Out[36]:
```
title_x                0
studio                 0
domestic_gross_x       1
foreign_gross        196
year                   0
id                     0
release_date           0
production_budget      0
domestic_gross_y       0
worldwide_gross        0
movie_id               0
primary_title          0
original_title         0
start_year             0
runtime_minutes       27
genres                 6
averagerating          0
numvotes               0
ordering               0
title_y                0
region                99
language            1205
types                465
attributes          1342
is_original_title      0
dtype: int64
```

In [37]:
```
# Drop following columns: 'id', 'ordering', 'language', 'types',
# 'attributes', 'is_original-title'
super_df.drop(['id', 'ordering', 'language', 'types',
               'attributes', 'is_original_title', 'region', 'start_yea
              axis=1, inplace = True)
```

In [38]:
```
super_df
```

Out[38]:

	title_x	studio	domestic_gross_x	foreign_gross	year	release_date	production_budge
0	Toy Story 3	BV	415000000.0	652000000	2010	Jun 18, 2010	$200,000,00(
1	Inception	WB	292600000.0	535700000	2010	Jul 16, 2010	$160,000,00(
2	Shrek Forever After	P/DW	238700000.0	513900000	2010	May 21, 2010	$165,000,00(
3	The Twilight Saga: Eclipse	Sum.	300500000.0	398000000	2010	Jun 30, 2010	$68,000,00(

	title_x	studio	domestic_gross_x	foreign_gross	year	release_date	production_budge
4	Iron Man 2	Par.	312400000.0	311500000	2010	May 7, 2010	$170,000,00(
...	...	...	...	...	...	...	..
1393	Gotti	VE	4300000.0	NaN	2018	Jun 15, 2018	$10,000,00(
1394	Bilal: A New Breed of Hero	VE	491000.0	1700000	2018	Feb 2, 2018	$30,000,00(
1395	Mandy	RLJ	1200000.0	NaN	2018	Sep 14, 2018	$6,000,00(
1396	Mandy	RLJ	1200000.0	NaN	2018	Sep 14, 2018	$6,000,00(
1397	Lean on	A24	1200000.0	NaN	2018	Apr 6, 2018	$8,000,00(

In [39]:
```
#Keep title_x and drop duplicate column movie titles

super_df.drop(['title_y', 'primary_title', 'original_title'], axis=1,
```

In [40]:
```
#Drop duplicate gross revenue columns 'domestic_gross_x' and 'foreign_

super_df.drop(['domestic_gross_x', 'foreign_gross'], axis=1, inplace =
```

In [41]:
```
#drop 'movie_id', use super_df index for new id

super_df.drop(['movie_id'], axis = 1, inplace = True)
```

In [42]:
```
super_df.columns
```

Out[42]:
```
Index(['title_x', 'studio', 'year', 'release_date', 'production_budge
t',
       'domestic_gross_y', 'worldwide_gross', 'runtime_minutes', 'gen
res',
       'averagerating', 'numvotes'],
      dtype='object')
```

In [43]:
```
super_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1398 entries, 0 to 1397
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
```

In [44]:
```
#convert production budget to integer datatype

super_df["production_budget"] = super_df["production_budget"].replace(
    "[$,]", "", regex=True).astype(int)
```

In [45]:
```
#convert domestic gross and worldwide gross to integer datatype
super_df["domestic_gross_y"] = super_df["domestic_gross_y"].replace(
    "[$,]", "", regex=True).astype(int)

super_df["worldwide_gross"] = super_df["worldwide_gross"].replace(
    "[$,]", "", regex=True).astype(int)
```

In [46]:
```
# Confirm data types for further filtering
super_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1398 entries, 0 to 1397
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title_x            1398 non-null   object
 1   studio             1398 non-null   object
 2   year               1398 non-null   int64
 3   release_date       1398 non-null   object
 4   production_budget  1398 non-null   int64
 5   domestic_gross_y   1398 non-null   int64
 6   worldwide_gross    1398 non-null   int64
 7   runtime_minutes    1371 non-null   float64
 8   genres             1392 non-null   object
 9   averagerating      1398 non-null   float64
 10  numvotes           1398 non-null   int64
dtypes: float64(2), int64(5), object(4)
memory usage: 131.1+ KB
```

In [47]:
```
#Examine aggregates on runtime_minutes, identify to replace with mean

super_df['runtime_minutes'].describe()
```

Out[47]:
```
count    1371.000000
mean      107.261853
std        19.654739
min         3.000000
25%        95.000000
50%       105.000000
75%       118.000000
max       192.000000
Name: runtime_minutes, dtype: float64
```

In [48]:
```
#Use median to replace missing data in runtime_minutes
```

```
super_df['runtime_minutes'] = super_df['runtime_minutes'].fillna(
    super_df['runtime_minutes'].median())
```

In [49]:
```
# Missing data in genres is categorical, create a 'missing' category

super_df['genres'] = super_df['genres'].fillna('missing')
```

In [50]:
```
# Verify all missing data is accounted for.

super_df.isna().sum()
```

Out[50]:
```
title_x              0
studio               0
year                 0
release_date         0
production_budget    0
domestic_gross_y     0
worldwide_gross      0
runtime_minutes      0
genres               0
averagerating        0
numvotes             0
dtype: int64
```

In [51]:
```
#convert release date into month released only

super_df['release_date'].dtypes
```

Out[51]:
```
dtype('O')
```

In [52]:
```
super_df['release_date'] = pd.to_datetime(super_df['release_date'])

super_df['release_date'].head()
```

Out[52]:
```
0   2010-06-18
1   2010-07-16
2   2010-05-21
3   2010-06-30
4   2010-05-07
Name: release_date, dtype: datetime64[ns]
```

In [53]:
```
# Create a month released column by month name and month number.

super_df['month_released'] = pd.to_datetime(super_df['release_date']).

super_df['month_num'] = pd.to_datetime(super_df['release_date']).dt.mo

super_df.columns
```

Out[53]:

```
Index(['title_x', 'studio', 'year', 'release_date', 'production_budge
t',
```

In [54]:
```
# Drop release date and keep months released now.

super_df.drop(['release_date'], axis = 1, inplace = True)
```

In [55]:
```
super_df.columns
```

Out[55]:
```
Index(['title_x', 'studio', 'year', 'production_budget', 'domestic_gr
oss_y',
       'worldwide_gross', 'runtime_minutes', 'genres', 'averagerating
',
       'numvotes', 'month_released', 'month_num'],
      dtype='object')
```

In [56]:
```
# Rename columns for clarity

super_df.rename(columns = {'title_x': 'Movie_Title',
                           'domestic_gross_y': 'domestic_gross'}, inpl

print(super_df.columns)
```

```
Index(['Movie_Title', 'studio', 'year', 'production_budget', 'domesti
c_gross',
       'worldwide_gross', 'runtime_minutes', 'genres', 'averagerating
',
       'numvotes', 'month_released', 'month_num'],
      dtype='object')
```

In [57]:
```
#create new column, separate out foreign_gross

super_df['foreign_gross'] = super_df['worldwide_gross'] - super_df['do

super_df['foreign_gross'].head()
```

Out[57]:
```
0    653874642
1    542948447
2    517507886
3    405571077
4    308723058
Name: foreign_gross, dtype: int64
```

In [58]:
```
# create new column reflecting production percentage of gross revenue.
# 'production_percent' is the production_budget divided by worldwide_g

super_df['production_percent'] = (super_df['production_budget'] /
                                  super_df['worldwide_gross'])* 100

super_df[['Movie_Title', 'production_percent']].head()
```

Out[58]:

	Movie_Title	production_percent

	Movie_Title	production_percent
0	Toy Story 3	18.711183
1	Inception	19.149645
2	Shrek Forever After	21.818336

In [59]:
```
super_df
```

Out[59]:

	Movie_Title	studio	year	production_budget	domestic_gross	worldwide_gross	runtime_
0	Toy Story 3	BV	2010	200000000	415004880	1068879522	
1	Inception	WB	2010	160000000	292576195	835524642	
2	Shrek Forever After	P/DW	2010	165000000	238736787	756244673	
3	The Twilight Saga: Eclipse	Sum.	2010	68000000	300531751	706102828	
4	Iron Man 2	Par.	2010	170000000	312433331	621156389	
...	...	...	...	...	...	...	
1393	Gotti	VE	2018	10000000	4286367	6089100	
1394	Bilal: A New Breed of Hero	VE	2018	30000000	490973	648599	
1395	Mandy	RLJ	2018	6000000	1214525	1427656	
1396	Mandy	RLJ	2018	6000000	1214525	1427656	
1397	Lean on Pete	A24	2018	8000000	1163056	2455027	

1398 rows × 14 columns

In [60]:
```
# There are duplicate movie titles, drop duplicate rows by movie title

super_df = super_df.drop_duplicates(subset= 'Movie_Title', keep="first
```

Data Cleaning Summary

The three initial dataframes were joined into one large dataframe. That dataframe contained duplicate data in several columns such as the movie title or gross revenue. Those columns were dropped. Many columns were missing data. Based on our business questions focusing on movie costs, revenue, and specific movie attributes, more columns were dropped for a more manageable dataframe. Data was missing in several columns such as runtime_minutes. The missing data was replaced with the median in each case due to outliers in data skewing the mean. Some data was labeled as a missing category.

Data Analysis

Focus on key business questions and what defines a successful film. Begin with descriptive statisitics. Extract data related to production costs and identify relationship to domestic revenue and worldwide revenue. Consider whether the median cost to produce a film has changed over the years. Identify key movie attributes that may lead to box office success.

In [61]:
```
#Sort dataframe by worldwide gross

super_df = super_df.sort_values(by = 'worldwide_gross', ascending = Fa
                                ignore_index = True)

super_df.head()
```

Out[61]:

	Movie_Title	studio	year	production_budget	domestic_gross	worldwide_gross	runtime_mir
0	Avengers: Infinity War	BV	2018	300000000	678815482	2048134200	
1	Jurassic World	Uni.	2015	215000000	652270625	1648854864	
2	Furious 7	Uni.	2015	190000000	353007020	1518722794	
3	Avengers: Age of Ultron	BV	2015	330600000	459005868	1403013963	
4	Black Panther	BV	2018	200000000	700059566	1348258224	

In [62]:
```
# Drop rows where worldwide gross and domestic gross is zero.

super_df = super_df[:1154]


#Had trouble with python warning, took a slice instead to drop offendi
```

Defining movie success

It is now important to consider our business question and what makes a movie successful. For a large public company such as Microsoft they are likely going to care very much about production costs and revenue, especially if their competitors are large established movie studios. Sort the dataframe by worldwide gross to further filter by amount of revenue earned for the most successful films. Take the top 50 grossing films of each year available. Compare the descriptive statistics of the top 50 films of each year to the larger dataframe.

In [63]:
```
#Identify top 50 films of each year

df_2010 = super_df[super_df['year'] == 2010][:50]
```

```
df_2011 = super_df[super_df['year'] == 2011][:50]

df_2012 = super_df[super_df['year'] == 2012][:50]

df_2013 = super_df[super_df['year'] == 2013][:50]

df_2014 = super_df[super_df['year'] == 2014][:50]

df_2015 = super_df[super_df['year'] == 2015][:50]

df_2016 = super_df[super_df['year'] == 2016][:50]

df_2017 = super_df[super_df['year'] == 2017][:50]

df_2018 = super_df[super_df['year'] == 2018][:50]

# Concatenate each movie year dataframe into one dataframe

top_df = pd.concat([df_2010, df_2011, df_2012, df_2013,
           df_2014, df_2015, df_2016, df_2017,
           df_2018])

top_df = top_df.reset_index(drop = True)
```

In [64]:
```
# Sort top 50 movies of each year by worldwide gross revenue

top_df = top_df.sort_values(by = 'worldwide_gross', ascending = False,
                            ignore_index = True)

# Compare top 50 from each year to overall super_df.

top_df.describe().astype(int)
```

Out[64]:

	year	production_budget	domestic_gross	worldwide_gross	runtime_minutes	averagerat
count	450	450	450	450	450	
mean	2014	93217777	132092268	353442938	114	
std	2	67813267	105363308	294687399	19	
min	2010	3000000	8178001	73866088	50	
25%	2012	40000000	63573607	153155012	99	
50%	2014	75000000	100269433	238949263	111	
75%	2016	140000000	162946882	443582754	127	
max	2018	410600000	700059566	2048134200	180	

In [65]:
```
super_df.describe().astype(int)
```

Out[65]:

	year	production_budget	domestic_gross	worldwide_gross	runtime_minutes	averagerat
count	1154	1154	1154	1154	1154	1

	year	production_budget	domestic_gross	worldwide_gross	runtime_minutes	averagerat
mean	2013	48912907	63462033	159921566	108	
std	2	57127476	86891558	241574575	17	
min	2010	50000	0	7943	41	
25%	2011	11000000	9291413	20441424	96	
50%	2014	28000000	35116412	69814368	106	
75%	2016	60000000	78042832	181743225	118	

In [66]:
```
# Print median of production costs Top 50 per year and super_df

print("The median production cost for super_df is USD",
      super_df['production_budget'].median())
print("The median production cost for top_df is USD",
      top_df['production_budget'].median())

# Domestic gross median compare

print("The median domestic gross for super_df is USD",
      super_df['domestic_gross'].median())
print("The median domestic gross for top_df is USD",
      top_df['domestic_gross'].median())

# Worldwide gross compare

print("The median worldwide gross for super_df is USD",
      super_df['worldwide_gross'].median())
print("The median worldwide gross for top_df is USD",
      top_df['worldwide_gross'].median())
```

```
The median production cost for super_df is USD 28000000.0
The median production cost for top_df is USD 75000000.0
The median domestic gross for super_df is USD 35116412.5
The median domestic gross for top_df is USD 100269433.5
The median worldwide gross for super_df is USD 69814368.0
The median worldwide gross for top_df is USD 238949263.0
```

In [67]:
```
# Identify the skewness of the dataset for the top 50 films per year.

print("The mean production costs for top_df is USD",
      top_df['production_budget'].mean().astype(int))
print("The mean worldwide gross for top_df is USD",
      top_df['domestic_gross'].mean().astype(int))
```

```
The mean production costs for top_df is USD 93217777
The mean worldwide gross for top_df is USD 132092268
```

Analysis

The top 50 films of each year cost substantially more to produce compared to all films in the data set. But they also bring far more revenue both domestically and worldwide. In the

top_50 dataset the mean is skewed to the right compared to the median, meaning there may be extreme outliers. Also, based on the median revenues in top_50 it appears that a worldwide release and not just a domestic release is critical to covering costs and earning a healthier return on investment.

In [68]:
```
# Identify the top 5 films out of the top_50. Compare to the mean and
# and gross revenue.

top_df[:5]
```

Out[68]:

	Movie_Title	studio	year	production_budget	domestic_gross	worldwide_gross	runtime_mir
0	Avengers: Infinity War	BV	2018	300000000	678815482	2048134200	
1	Jurassic World	Uni.	2015	215000000	652270625	1648854864	
2	Furious 7	Uni.	2015	190000000	353007020	1518722794	
3	Avengers: Age of Ultron	BV	2015	330600000	459005868	1403013963	
4	Black Panther	BV	2018	200000000	700059566	1348258224	

In [69]:
```
# Identify films near the median for budget.

#top_df[top_df['production_budget'] <= 78000000 | top_df['production_b

super_df.query("74000000 < production_budget < 76000000")
```

Out[69]:

	Movie_Title	studio	year	production_budget	domestic_gross	worldwide_gross	runtime_r
20	Despicable Me 3	Uni.	2017	75000000	264624300	1034727750	
29	The Secret Life of Pets	Uni.	2016	75000000	368384330	886750534	
70	Sing	Uni.	2016	75000000	270329045	634454789	
151	Now You See Me	LG/S	2013	75000000	117723989	342769200	
199	Grown Ups	Sony	2010	75000000	162001186	272223430	
244	Inferno	Sony	2016	75000000	34343574	219519367	
259	Immortals	Rela.	2011	75000000	83504017	211562435	
471	Killers	LGF	2010	75000000	47059963	95572749	
497	Sucker Punch	WB	2011	75000000	36392502	89758389	

In [70]:
```
super_df.query("100000000 < domestic_gross < 101000000")
```

Out[70]:

	Movie_Title	studio	year	production_budget	domestic_gross	worldwide_gross	runtime_r
119	Kingsman: The Golden Circle	Fox	2017	104000000	100234838	408803696	
134	Fifty Shades Freed	Uni.	2018	55000000	100407760	371350619	
136	Edge of Tomorrow	WB	2014	178000000	100206256	370541256	
251	Bad Teacher	Sony	2011	19000000	100292856	215448997	
258	Due Date	WB	2010	65000000	100539043	211739043	
267	Yogi Bear	WB	2010	80000000	100246011	204774690	
268	Arrival	Par.	2016	47000000	100546139	203127894	
375	A Wrinkle in Time	BV	2018	103000000	100478608	133401882	

Analysis

Outlier films such as Black Panther and Avengers: Infinity War costs substantially more to produce -in the hundreds of millions dollars. The risk worked for those studios as they were able to earn in the billions with worldwide releases. Microsoft would have to spend a substantial amount with no guarantees of success for a first film of this magnitude. Example films that hew more closely to the median and could be a lower investment risk for Microsoft, but still garner a top 50 film, are films such as Despicable Me 3 and Kingsman: The Golden Circle.

In [71]:
```
# Identify most popular months for movie releases for top_50

top_df['month_released'].value_counts()
```

Out[71]:
```
November     59
July         57
December     55
June         48
May          40
October      37
March        31
February     31
August       26
September    25
April        22
January      19
Name: month_released, dtype: int64
```

In [72]:
```
# Identify the most popular genres

top_df['genres'].value_counts()
```

Out[72]:

```
Adventure,Animation,Comedy    51
Action,Adventure,Sci-Fi       41
Action,Adventure,Fantasy      23
Action,Adventure,Drama        14
Action,Comedy,Crime           13
                              ..
Adventure,Mystery,Sci-Fi       1
Horror,Sci-Fi,Thriller         1
Comedy,Fantasy                 1
Action,Adventure,Western       1
Action,Comedy,Drama            1
```

In [73]:
```
# Identify mean, median, ranges of runtime minutes

top_df['runtime_minutes'].describe().astype(int)
```

Out[73]:
```
count    450
mean     114
std       19
min       50
25%       99
50%      111
75%      127
max      180
Name: runtime_minutes, dtype: int64
```

In [74]:
```
# Compare median rating for a top 50 film compared to overall dataset

print(top_df["averagerating"].median())
print(super_df["averagerating"].median())
```

```
6.65
6.5
```

Analysis

For a top 50 film the months of November, December, June, and July are the most popular times to release a film. The median runtime for those films is 111 minutes, maximum runtimes are 3 hours long. Many of the films fall into Action and Adventure categories. There does not appear to be much of a difference in ratings between a top 50 film and the overall dataset.

Summary of initial data analysis

After cleaning and filtering a new dataframe was created that took the top 50 films of each year. The median cost of the top 50 films is USD 75 million. This is significantly higher than the original dataframe's median cost of USD 28 million. Worldwide revenue median for the top 50 of each year was also significantly higher at USD 240 million compared to all films at under USD 70 million. For a top 50 film in the ensuing years Microsoft can expect to pay a median of 26% of gross revenue, but that can range much higher with the interquartile range being 17-39%. Films are often released in Nov, Dec, June, and July. They majority range between 99-127 minutes. Many of them are action and adventure films. A movie studio

could release a lower cost film or a higher cost film but ratings may not vary much in the end between the two.

Limitations -

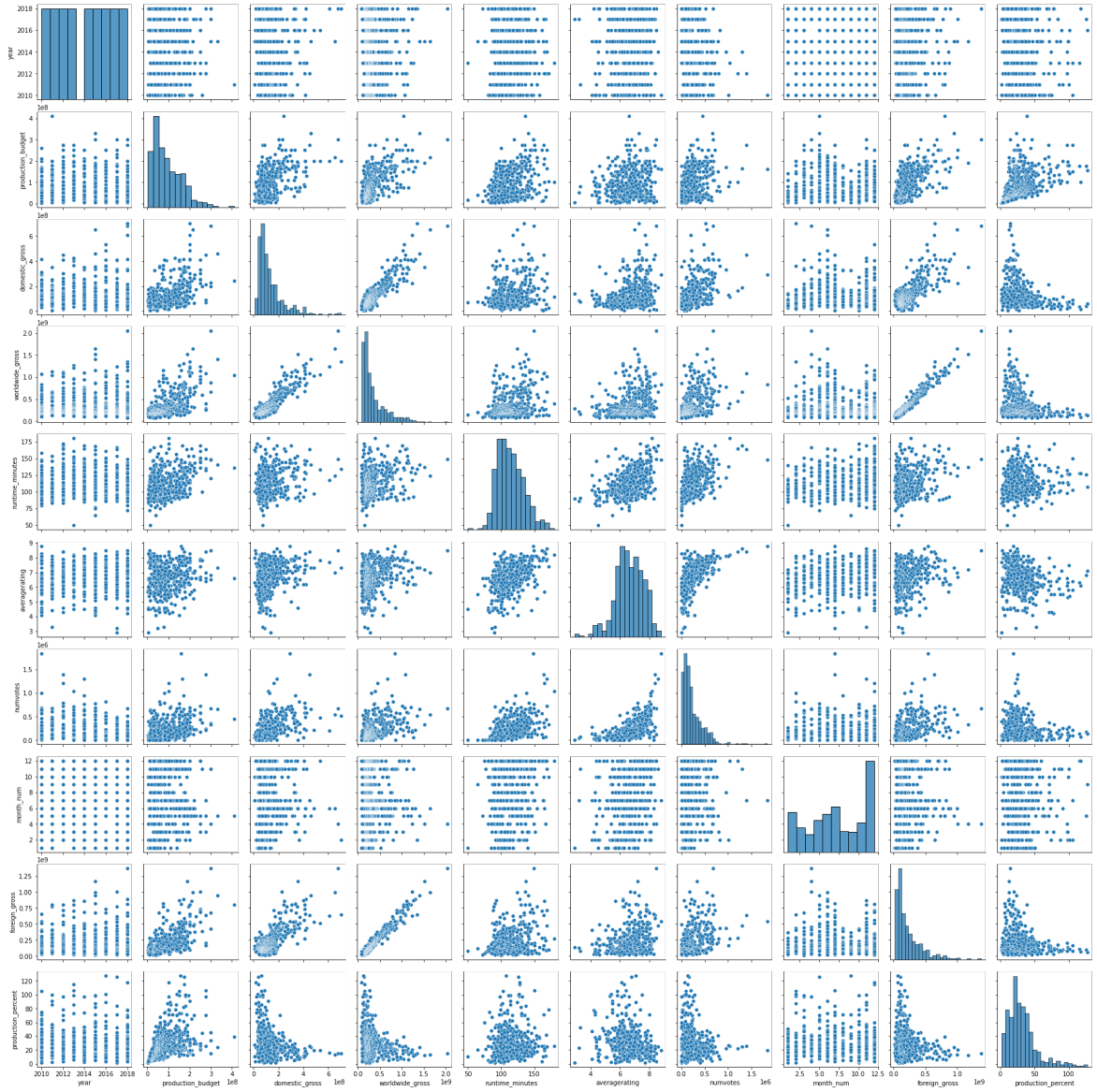
The dataset only looks at films from 2010 to 2018. It does not account for changes to movie going habits and revenue streams since the covid19 pandemic took hold in 2020. The movie industry largely paused many releases in 2020. Currently the movie industry is still adjusting and online releases have become more popular. Future data analyses could take this into account for a more nuanced look into revenue and movie attributes.

Data Visualization - Exploratory

Analyze the dataframe and identify trends and patterns. Begin with exploratory visualizations examining costs, revenue, and data categories that will affect these elements, such as month of release and runtime in minutes.

In [75]:
```
import matplotlib.pyplot as plt
%matplotlib inline
#plt.rcParams['figure.figsize'] = (10, 10)


#plt.style.available (remove # for plot style options)
```

In [76]:
```
#Run a pairplot using seaborn, identify trends and patterns for furthe

sns.pairplot(top_df)

plt.show()
```

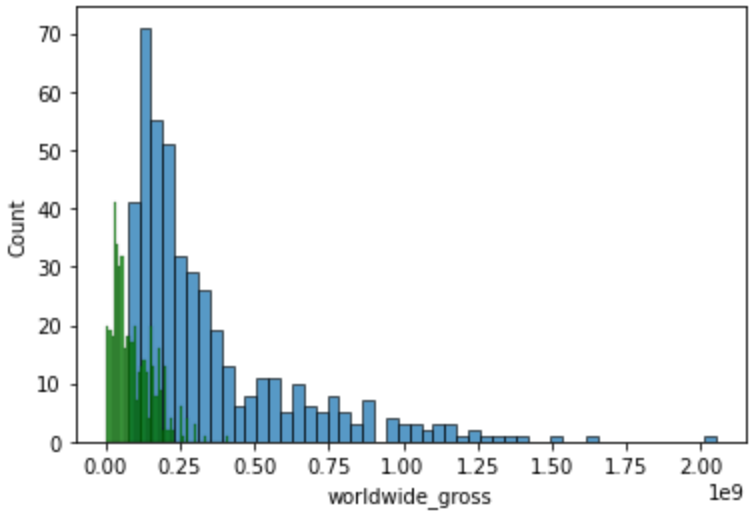


Trends and patterns for exploratory visualizations

Based on the pairplot above, further explore the relationship between worldwide gross, domestic gross and production budgets. Also explore further the runtime minutes category and month released. Examine the production percentages of revenue. Primarily use histograms, bar charts, scatter plots, and boxplots.

In [77]:
```
# Examine production budgets and their relationship to worldwide gross
#using histogram.

sns.histplot(top_df, x = 'worldwide_gross', bins = 50)
sns.histplot(top_df, x = 'production_budget', bins = 50, color = 'gree
```

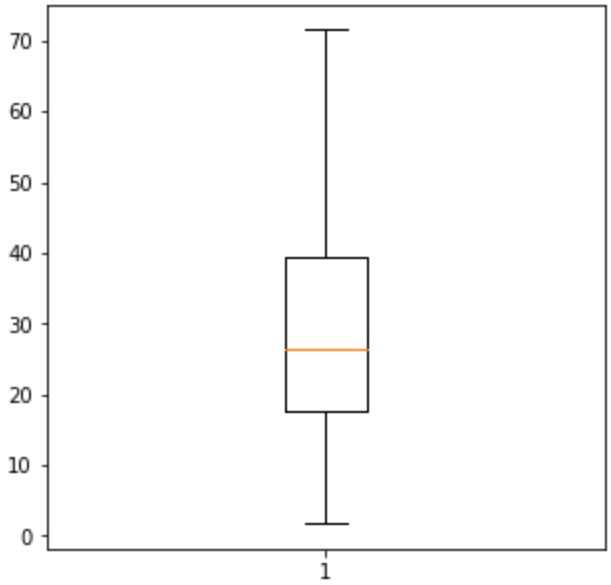
Out[77]:
```
<AxesSubplot:xlabel='worldwide_gross', ylabel='Count'>
```



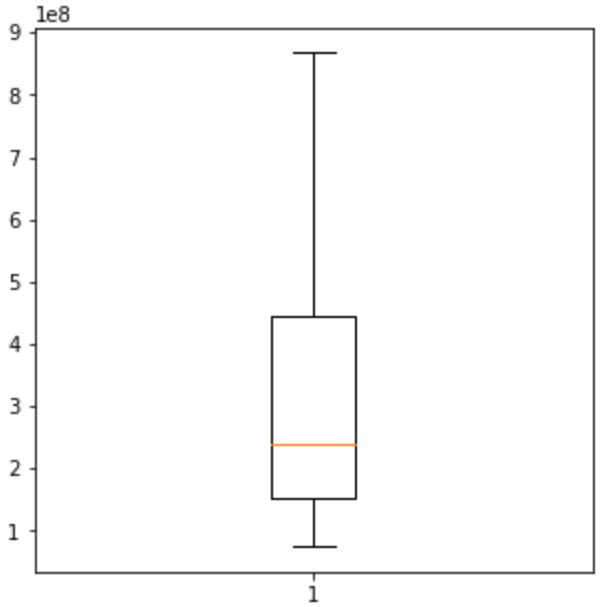
In [78]:
```
#Find the median and mean percentage production cost.

print("The median percentage of production costs for all films is",
     top_df["production_percent"].median().astype(int), "percent.")

print("The mean percentage of production costs for all films is",
     top_df["production_percent"].mean().astype(int), "percent.")
```

```
The median percentage of production costs for all films is 26 percen
t.
The mean percentage of production costs for all films is 31 percent.
```

In [79]:
```
#Create a boxplot to show median and interquartile ranges for producti
#percentages of worldwide revenue. Do not show outliers.

fig, ax = plt.subplots(figsize=(5, 5))
plt.boxplot(top_df['production_percent'], showfliers = False);
```



In [80]:
```
# Examine worldwide revenue median and ranges

fig, ax = plt.subplots(figsize=(5, 5))
plt.boxplot(top_df['worldwide_gross'], showfliers = False);
```

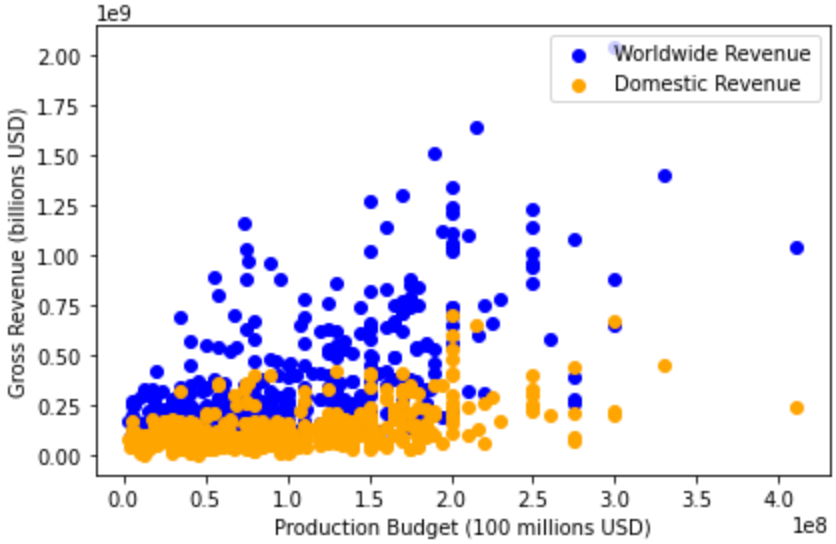


In [81]:
```
# Create reference stat dataframe of costs and gross

top_df[['production_budget', 'domestic_gross', 'worldwide_gross',
          'production_percent']].describe().astype(int)
```

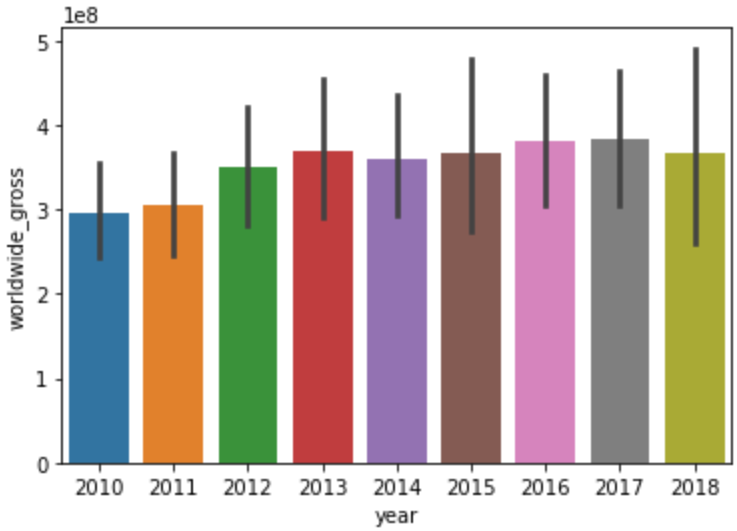
Out[81]:

	production_budget	domestic_gross	worldwide_gross	production_percent
count	450	450	450	450
mean	93217777	132092268	353442938	31
std	67813267	105363308	294687399	21
min	3000000	8178001	73866088	1
25%	40000000	63573607	153155012	17
50%	75000000	100269433	238949263	26
75%	140000000	162946882	443582754	39
max	410600000	700059566	2048134200	127

In [82]:
```
# Examine production budgets and their relationship to worldwide gross
#scatter plot.

fig, ax = plt.subplots()

ax.scatter(top_df['production_budget'], top_df['worldwide_gross'],
           c ='blue')
ax.scatter(top_df['production_budget'], top_df['domestic_gross'],
           c ='orange')


ax.set_xlabel('Production Budget (100 millions USD)')
ax.set_ylabel('Gross Revenue (billions USD)')



ax.legend(['Worldwide Revenue', 'Domestic Revenue'], loc='upper right'
plt.tight_layout()
```



In [83]:
```
sns.barplot(x ='year', y = 'worldwide_gross', data = top_df);
```



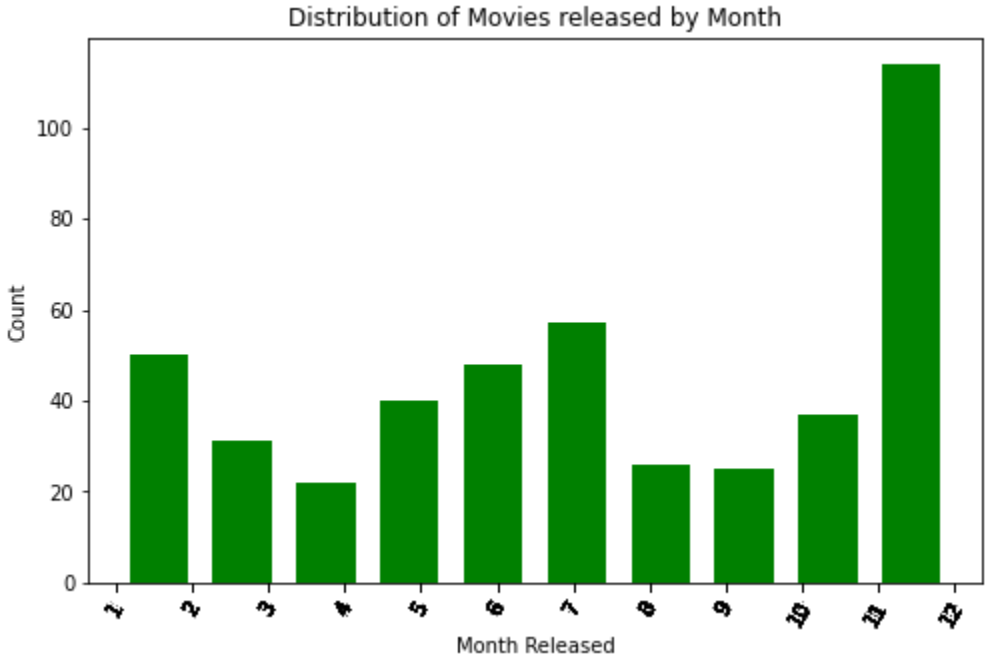
Exploratory Visual Summary Analysis of Costs and Revenue:

It appears that there is a wide range of production budgets with no guarantee of a profit. Median production cost, however, hovers in the mid 20% range of worldwide gross but the mean is above 30% suggesting there are outliers that can leave a production company not making much of a profit or falling into the red. Barring the outliers most production costs fall between 18% to 40% of revenue based on the boxplots. The domestic revenue median hovers above USD 100 million while the worldwide revenue median floats above USD 220 million. This suggests a worldwide release is significant when it comes to ROI and covering productions costs. It also appears there has been a small, but steady increase in gross revenue from 2010 to 2018.

In [84]:
```
top_df.columns
```

Out[84]:
```
Index(['Movie_Title', 'studio', 'year', 'production_budget', 'domesti
c_gross',
       'worldwide_gross', 'runtime_minutes', 'genres', 'averagerating
',
       'numvotes', 'month_released', 'month_num', 'foreign_gross',
       'production_percent'],
      dtype='object')
```

In [85]:
```
#Show histogram for which month films are released

fig, ax = plt.subplots(figsize = (8, 5))
ax.hist(top_df['month_num'], color='green', rwidth = 0.7)
ax.set_xlabel('Month Released');
ax.set_ylabel('Count')
ax.set_title('Distribution of Movies released by Month')
plt.xticks(super_df['month_num'],
                    rotation=60);
```

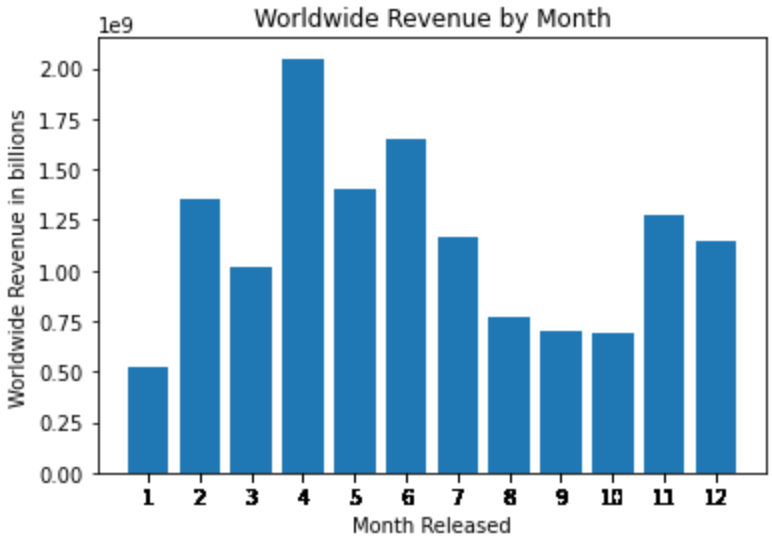


In [86]:
```
top_df['month_released'].value_counts()
```

Out[86]:
```
November     59
July         57
December     55
June         48
May          40
October      37
March        31
February     31
August       26
September    25
April        22
January      19
Name: month_released, dtype: int64
```

In [87]:
```
# Examine month released and their relationship to revenue.
fig, ax = plt.subplots()

plt.bar(top_df['month_num'], top_df['worldwide_gross'])

ax.set_xlabel('Month Released');
ax.set_ylabel('Worldwide Revenue in billions')
ax.set_title('Worldwide Revenue by Month')
```

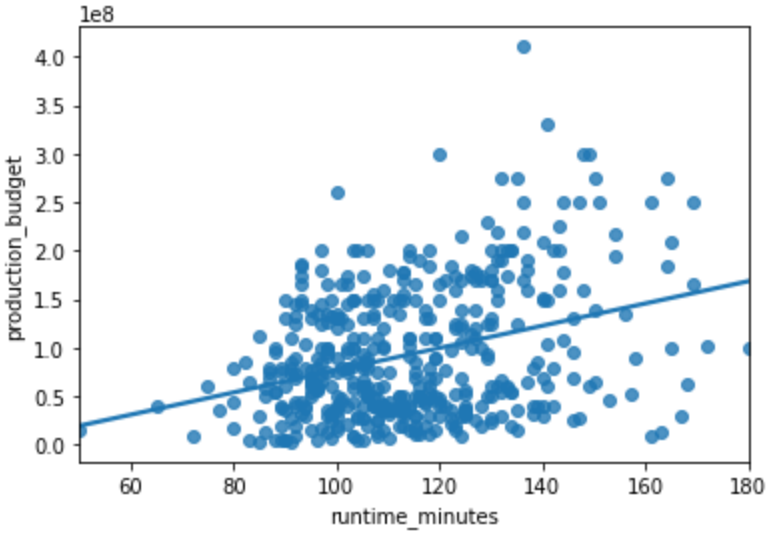
```
plt.xticks(super_df['month_num']);
```



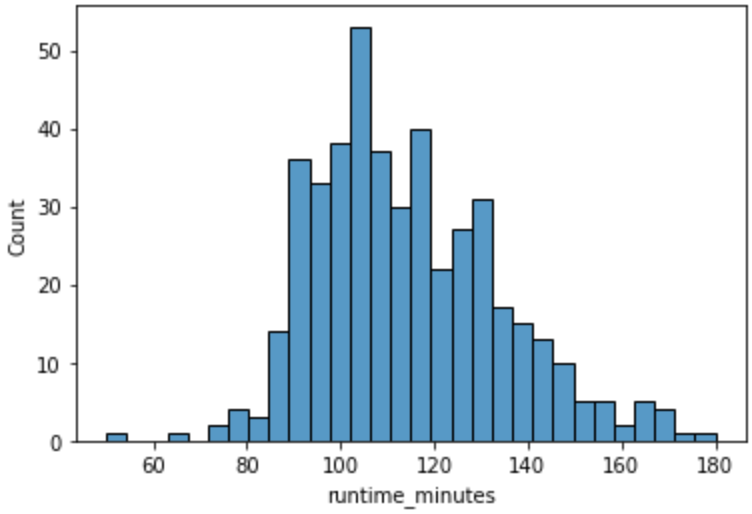
Exploratory Visual Summary Analysis of Movie Release Timing:

It appears that more films are released in the fall months of October, November, and December followed by another grouping in June and July. Revenue wise May, June, July provide the largest returns followed by the fall months grouping.

In [88]:
```
# Examine relationship between runtime minutes and production budget i
sns.regplot(x = "runtime_minutes",
            y = "production_budget",
            ci = None,
            data = top_df);
```



In [89]:
```
#Identify the peak movie runtime, use histogram

sns.histplot(top_df, x = 'runtime_minutes', bins = 30);
```



Exploratory Visual Summary Analysis of Movie Runtimes in Minutes:

Based on the scatter plot tracking runtime and production budget, generally the longer the film the increase in costs. Most films are centered around the 90-110 minute mark.

Data Visualization - Explanatory

Create three to four high quality data visualizations that show the following:

- The relationship between movie production costs and gross revenue both domestically and worldwide.
- The median and range of production costs across the years 2010 to 2018.
- The month of release that is most likely to draw more viewers and thus higher revenue to offset production costs.
- Runtime in minutes related to production costs.

In [90]:
```
top_df.describe().astype(int)
```

Out[90]:

	year	production_budget	domestic_gross	worldwide_gross	runtime_minutes	averagerat
count	450	450	450	450	450	
mean	2014	93217777	132092268	353442938	114	
std	2	67813267	105363308	294687399	19	
min	2010	3000000	8178001	73866088	50	
25%	2012	40000000	63573607	153155012	99	
50%	2014	75000000	100269433	238949263	111	

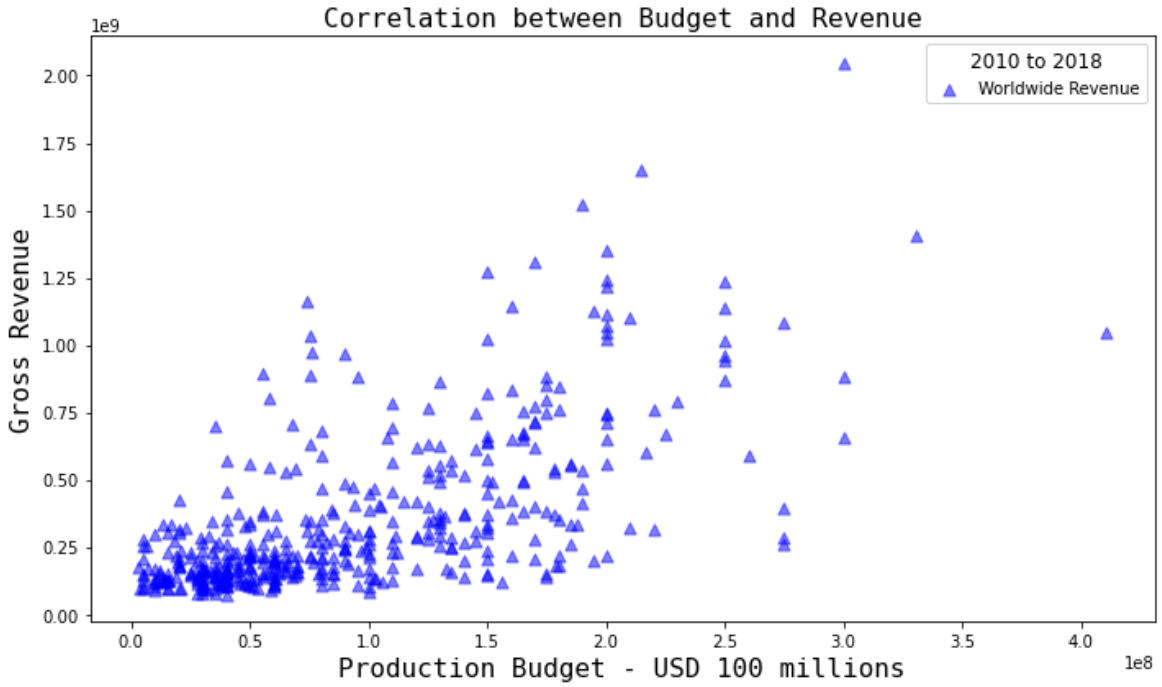
	year	production_budget	domestic_gross	worldwide_gross	runtime_minutes	averagera

Use Seaborn and MatPlotLib

Business Question - Relationship between movie cost and revenue.

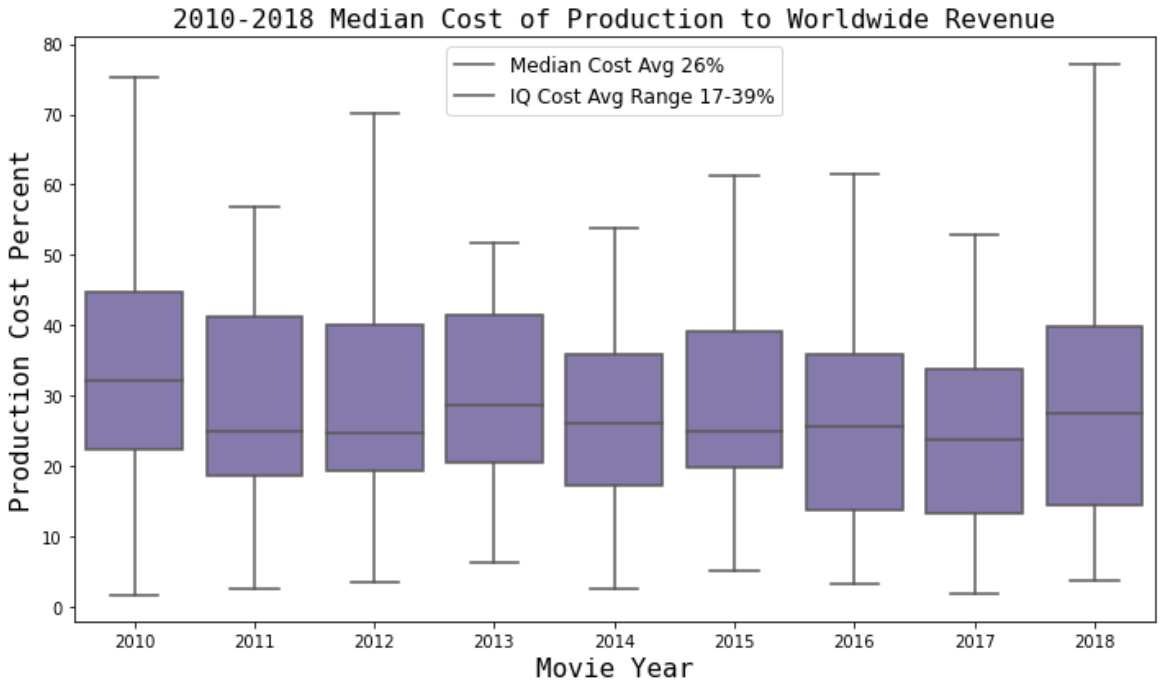
Display the big picture. Show the overall correlation between production cost and revenue. Show the trends for both domestic and worldwide.

In [110]:
```
# Show relationship between production costs and revenue. Use scatterp

fig, ax = plt.subplots(figsize=(10,6))


ax.scatter(top_df['production_budget'], top_df['worldwide_gross'],
            c ='blue',
            alpha=0.5,
            s=50,
            marker='^')



#set labels
plt.xlabel('Production Budget - USD 100 millions', size=16, family='mo
plt.ylabel('Gross Revenue', size=16, family='monospace')
plt.title('Correlation between Budget and Revenue', size=16,
          family='monospace', weight=500)


# Create Legend
ax.legend(['Worldwide Revenue', 'Domestic Revenue'], title = '2010 to
          title_fontsize = "12",loc='upper right')

plt.tight_layout();

#Save Fig
plt.savefig('images/scatter_budget_revenue.png')
```



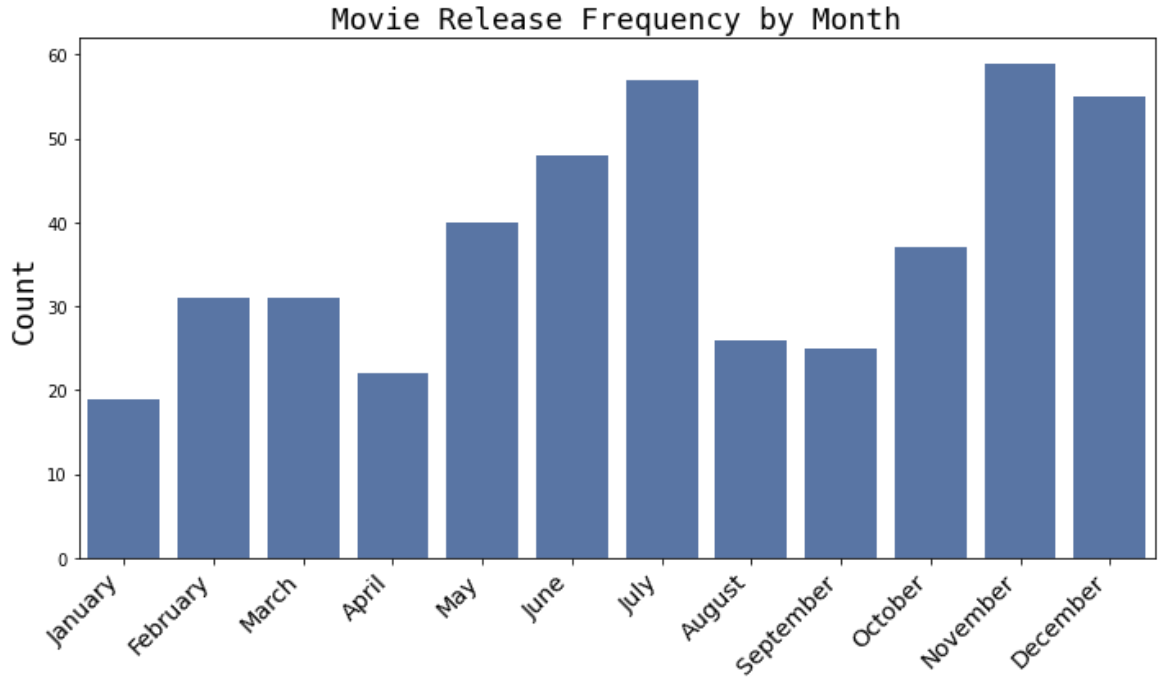
In [111]:
```
# Show the median and interquartile range of production budget percent
#worldwide gross

sns.set_palette("Set1", 8, .75)
```

```
plt.figure(figsize=(10,6))

box = sns.boxplot(x = top_df['year'], y = top_df['production_percent']
                  showfliers = False, color = 'm')


# Add labels and title

plt.xlabel('Movie Year', size=16, family='monospace', weight=500)
plt.ylabel('Production Cost Percent', size=16, family='monospace')
plt.title('2010-2018 Median Cost of Production to Worldwide Revenue',
          family='monospace', weight=500)

# Add text box labeling median, IQ range for all years

plt.legend(['Median Cost Avg 26%', 'IQ Cost Avg Range 17-39%'], fontsi
           loc='upper center')


plt.tight_layout();

#Save Fig

plt.savefig('images/median_budget_cost.png')
```



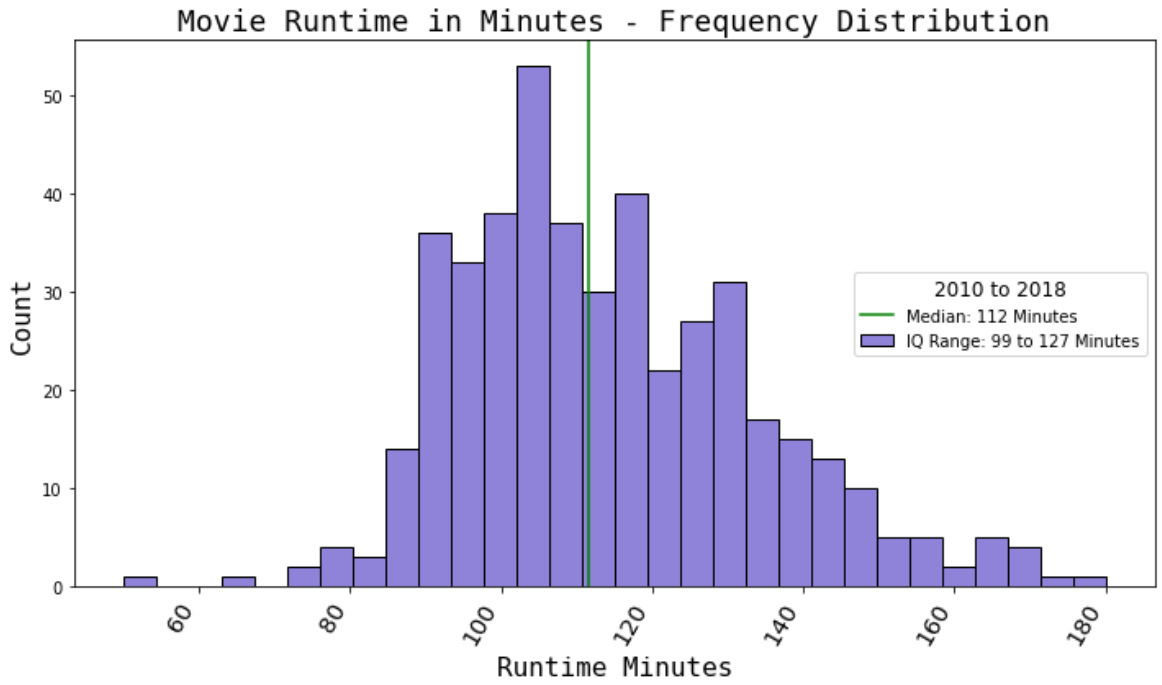
In [112]:
```
# Show most common months for movie release.


plt.figure(figsize=(10,6))


sns.countplot(x ='month_released', data = top_df, order = ['January',
                                                           'March', '
                                                           'June', 'J
```

```
                                                                'September
                                                                'November'
              color = 'b')

plt.xticks(
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)

#Set Title and Axes Labels

plt.xlabel(None)
plt.ylabel('Count', size=18, family='monospace')
plt.title('Movie Release Frequency by Month', size=18, family='monospa

plt.tight_layout();

#Save Fig

plt.savefig('images/movie_month.png')
```



In [113]:
```
# Create one more visualization for runtime minutes
plt.figure(figsize=(10,6))

sns.histplot(x ='runtime_minutes', data = top_df, bins = 30, color = '

plt.xticks(
    rotation=60,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)
```

```
# Display median vertical line

plt.axvline(x=top_df.runtime_minutes.median(),
            color='green')

# Label the axes

plt.xlabel('Runtime Minutes', size=16, family='monospace', weight=500)
plt.ylabel('Count', size=16, family='monospace')
plt.title('Movie Runtime in Minutes - Frequency Distribution', size=18
          family='monospace', weight=500)

# Add detail

plt.legend(['Median: 112 Minutes', 'IQ Range: 99 to 127 Minutes'], tit
          title_fontsize = "12",loc='center right')

plt.tight_layout();

# Save Fig

plt.savefig('images/movie_minutes.png')
```



Conclusions and Recommendations based on Explanatory Data Visualizations

The three datasets from Box Office Mojo, the production budget data set, and the IMDB database were merged into one dataframe that took the top 50 films from each year, 2010 to 2018, for data cleaning, query and visual analysis. Production costs, possible ranges of revenue, and factors that play into a film's likely success were examined. The end goal was to identify a pathway for Microsoft to break into the movie business.

Conclusions

- For the top 50 films per year the median production budget was USD 75 million, most films budgets ranged between USD 40 to 140 million. These costs reflect a median of 26% of revenue but can vary between 17-39%.
- Despite the steep initial production costs, the top 50 films each year enjoyed a domestic revenue median of USD 100 million. To further pad revenue worldwide releases brought in a median of USD 240 million with an interquartile range of USD 153 million to USD 443 million. A real risk remains, however, of creating a movie that barely covers production costs and possibly delivers a blow to the company's brand.
- Timing is important when releasing a film. Many films are released around October, November, and December followed by June and July. May, June, and July see some of the highest grossing films come out.
- Most films range between 99 to 127 minutes with a median of 112 minutes.

Recommendations

- Microsoft should set aside a fund of USD 75 million per film in order to break into the top 50 films in a calendar year. The initial budget can be set lower towards the USD 40 million range but no lower. Be prepared with a cushion of funds up to that USD 75 million mark, especially because Microsoft is new to the industry. What may be a lower cost film for other studios, Microsoft will possibly have to spend more initially to establish connections with movie directors, actors, distribution networks, marketing structures and the development of movie infrastructure that it may currently lack. Microsoft could easily spend in the hundreds of millions of dollars to create just one box office hit, but it is likely worth starting with a few films that have productions costs set to the median.
- To ensure production costs are covered and to expand the return on initial investments, Microsoft should consider a worlwide release of their films instead of to domestic audiences only. The potential to match domestic revenue and beyond is a real possibility. Beware the risks, however, no movie is guaranteed box office success. Microsoft should consider development of three movie ideas to take to production to spread risk, despite higher initial investments for three films the chance of delivering a successful film is higher.
- Microsoft should release a film sometime at the end of May or early June. This is the beginning of the summer movie season when many films are released and when some of the highest grossing films come out. November and December are also good times but so many films come out during this time it's possible Microsoft's film could get lost in the mix.
- For an initial release the movie should range between 99 to 127 minutes. The sweet spot is around 112 minutes for many films. Longer films may require a greater attentional effort from audiences. Longer films also appear to slowly, but steadily, add to production costs.

In [ ]: