Popular Video Games and Developers

Mason DeJesus

Mrunmai Gadbail

Nicholas Buse

Hassan Aftab - Appendix A

Thesis

Developers with fewer titles care more for their games than larger developers and thereby produce better products. The simplified version of this would be quality vs. quantity. During our analysis, presentation, and technical writeup, this is what we wished to prove or disprove.

Data Cleaning

Data cleaning on this dataset was interesting and challenging. Most of this dataset began as objects, so we had to convert quite a few columns to another datatype before we could do any analysis. We had to convert Release Date to datetime, after changing any TBD to null values. We changed the Rating to a float. We also changed six columns that included values such as 4.1k and 718 within the same column to integer columns. This was accomplished by changing “K” to three zeros for any cells containing “K”, multiplying by one thousand to move the decimal to the correct space, and then converting the entire column to an integer.

The following code was used to change the six columns to integers:

```
# Use a for loop to change to columns (Plays, Playing, Backlogs, Wishlist, Lists, and Reviews)
# Use an replace statement to change any value with 'K' to a float, multiply that by 1,000, and then change the entire column to integers

cols = ('Plays', 'Playing', 'Backlogs', 'Wishlist', 'Lists', 'Reviews')

for col in cols:
    # Apply a lambda function to replace 'K' and multiply by 1000 if 'K' is present
    game_df[col] = game_df[col].apply(lambda x: float(str(x).replace('K', '')) * 1000 if 'K' in str(x) else int(x))
```

We also had to separate developers from producers or studios that had been fed in from the dataset as a single string value and create a column for the number of platforms a title was

utilizing. Columns for year and month of Release Date were also added. The separation of developers was accomplished by dropping any brackets or apostrophes from the cell and splitting the value before a comma. Similar code was used to separate the Platforms column to a list and count its values for the total number of platforms columns.

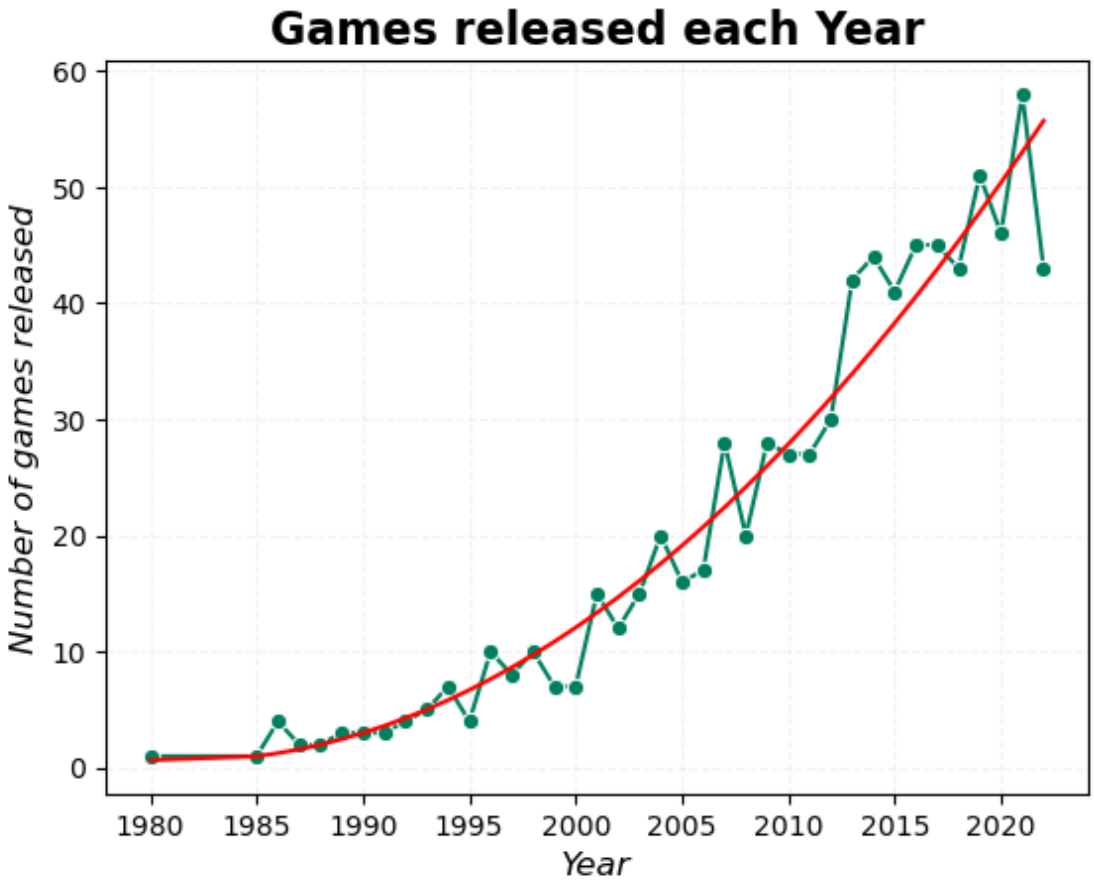
The change to developers was made with the following code:

```
# Seperate Developers from Producers
game_df['Developers']=game_df['Developers'].str.split(',').str[0]
# Remove [' and ]
game_df["Developers"] = game_df["Developers"].apply(lambda x: str(x).lstrip("['"))
game_df["Developers"] = game_df["Developers"].apply(lambda x: str(x).rstrip("']"))
```

Lastly, we discovered some bias in developers with a low number of reviews, influencing their average ratings values. It was then decided that we would only analyze titles with over two hundred ratings. We felt this would provide a better analysis more free of bias.

What are the most popular game titles?

The first part of this data story revolves around titles. The first line graph we can see is the number of games released per year. From the graph, we can easily see the number of games released per year has increased over the years as the gaming industry is developing rapidly and people have easier access to computers and gaming consoles. In this graph, we have not added 2023 as our database has data only till mid of 2023 due to which we had a big drop in number of games released.

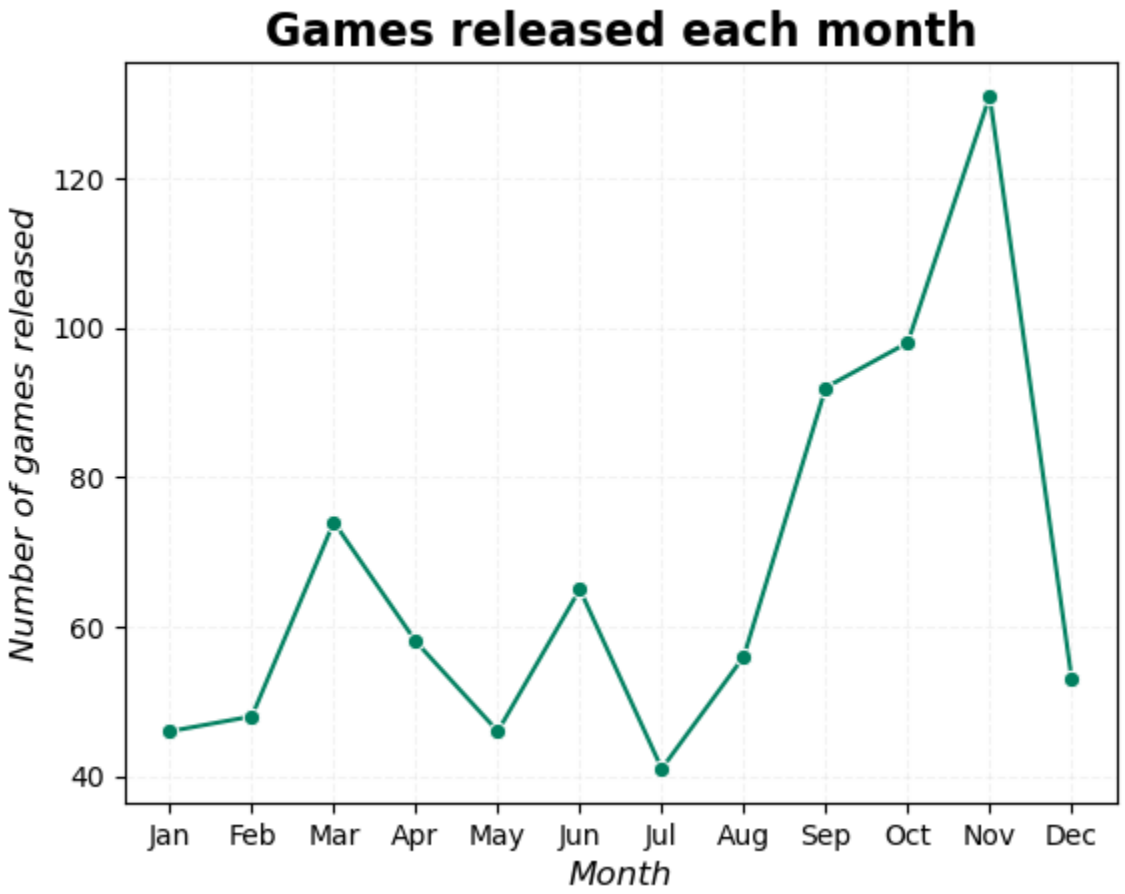


Code for line graph:

```
#bargraph for number of games realed per year

title_counts_y = reviews_df.groupby('Release_Year').size().reset_index(name='title_counts_y')
title_counts_y = title_counts_y[title_counts_y['Release_Year'] != title_counts_y['Release_Year'].max()]
plt.grid(color="lightgrey", linestyle="--", alpha=0.25)
sns.lineplot(data=title_counts_y, x='Release_Year', y='title_counts_y', marker='o')
x=reviews_df['Release_Year']
y=title_counts_y
xx=y['Release_Year']
yy=y['title_counts_y']
z=np.polyfit(xx,yy,2)
p=np.poly1d(z)
plt.plot(xx, p(xx),color="red")
plt.ylabel("Number of games released", fontstyle="italic", fontsize=12)
plt.title("Games released each Year", fontsize=16, fontweight="bold")
plt.xlabel("Year", fontstyle="italic", fontsize=12)
x_ticks=np.arange(1980,2025, step=5)
plt.xticks(x_ticks)
plt.show()
```

In the second line graph we can see the number of games released per month. In this graph, we have 3 peaks, one in March which is due to spring break, second peak in June which is due to summer vacation, and the third peak is in September, October, and November where people start

to shop for holiday gifts for Thanksgiving and Christmas. The largest peak is in November just before Christmas showing this as the most popular time to release games.

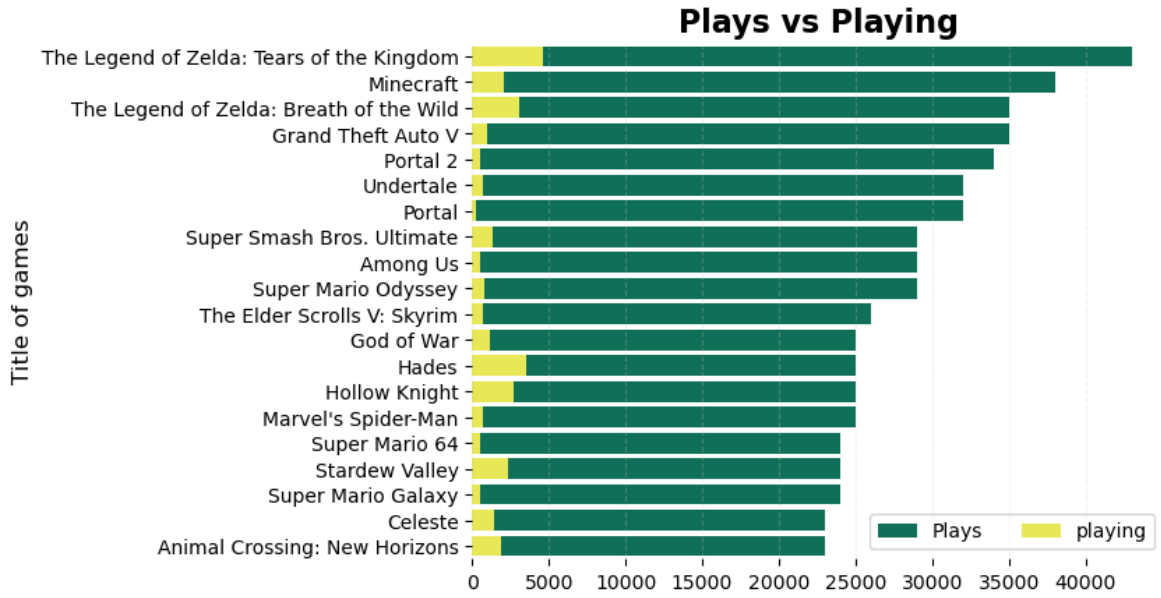


Code for line graph:

```
#Line graph for number of games released per month
title_counts_m = reviews_df.groupby('Release_Month').size().reset_index(name='title_count_m')
plt.ylabel("Number of games released", fontstyle="italic", fontsize=12)
plt.title("Games released each month", fontsize=16, fontweight="bold")
plt.xlabel("Month", fontstyle="italic", fontsize=12)
plt.grid(color="lightgrey", linestyle="--", alpha=0.25)
plt.xticks([1,2,3,4,5,6,7,8,9,10,11,12],["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])
sns.lineplot(data=title_counts_m, x='Release_Month', y='title_count_m', marker='o')
```

In the next bar chart, we can see the top 20 games plays vs playing, or copies sold vs concurrent users, where green indicates plays and yellow playing. The first game, The Legends of Zelda: Tears of Kingdom, which was released in May 2023, had the highest number of plays due to replayability. It is also a successor of the top 3rd game, The Legends of Zelda: Breath of the Wild, was released in March 2017. The 2nd game, Minecraft, was released in November 2011, but still has a high number of players as it has different modes to offer to its player. This chart

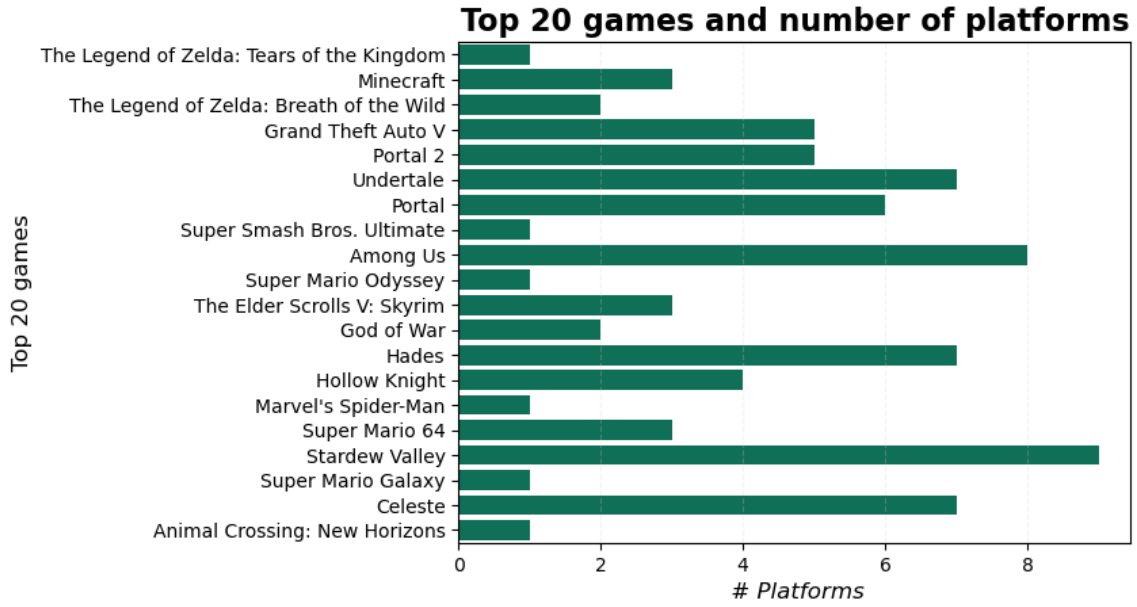
also shows some repeat names of games showing successful franchise games, such as Mario and Portal.



Code for bar chart:

```
#barchart for top 20 games for plays vs playing where we have sorted the dataframe by plays

x_ticks=np.arange(0,45000, step=5000)
Rating_df=reviews_df.sort_values(by="Plays", ascending=False).reset_index(drop=True).head(20)
sns.barplot(y="Title", x="Plays", data=Rating_df, label= "Plays", color="#007F5F")
sns.barplot(y="Title", x="Playing", data=Rating_df, label="playing", color="#FFFF3F")
plt.legend(ncol=2, loc="lower right", frameon=True)
sns.despine(left=True, bottom=True)
plt.grid(color="lightgrey", axis="x", linestyle="--", alpha=0.25)
plt.xlabel("", fontstyle="italic", fontsize=12)
plt.xticks(x_ticks)
plt.ylabel("Title of games", fontsize=12)
plt.title("Plays vs Playing", fontsize=16, fontweight="bold")
```

In the next chart, we can see the top 20 games and the number of platforms they are released on. We can easily see that some games are released on one platform and some games are released on nine platforms. From this chart, we can easily see the number of platforms and top games by plays/copies sold do not have a relation. Which concludes to become a top game, the number of platforms it is released on is not a deciding factor, with the top game being released on only one platform.



code for bar chart

```
# top 20 games and number of platform

Rating_df=reviews_df.sort_values(by="Plays", ascending=False).reset_index(drop=True).head(20)
sns.barplot(y="Title", x="Platform_Count", data=Rating_df, label= "Plays", color="#007F5F")
plt.grid(color="lightgrey", axis="x", linestyle="--", alpha=0.25)
plt.xlabel(" # Platforms", fontstyle="italic", fontsize=12)
plt.ylabel("Top 20 games", fontsize=12)
plt.title("Top 20 games a number of platforms", fontsize=16, fontweight="bold")
```

What are the Top 20 Developers?

The second part of this data story revolves around developers. The next question we wanted to solve was “What are the Top 20 Developers?” To answer that question, we broke it down into three sections, comparing different metrics, but all grouped by Developer. We broke the question into: “Top 20 Developers by Total Plays and Total Titles,” “Top 20 Developers by Average Rating,” and “Top 20 Developers by Average Rating, Average Playing, and Total Number of Plays.” To review this data and compare as needed, we began by creating a leaderboard that we would use for the respective graphs.

Developers	Number of Titles by Dev	Average Rating by Dev	Average Number Playing by Dev	Total Number of Plays by Dev
Nintendo	106	3.788679	301.216981	1011400.0
Capcom	37	3.837838	268.270270	260600.0
Ubisoft Entertainment	20	3.330000	200.200000	195400.0
Sony Interactive Entertainment	17	3.941176	302.647059	168400.0
Electronic Arts	14	3.650000	290.714286	156500.0
Square Enix	24	3.737500	378.500000	144700.0
Konami	16	3.950000	220.000000	130500.0
Naughty Dog	8	4.000000	192.125000	101200.0
Bandai Namco Entertainment	10	3.900000	413.600000	95400.0
Sega	19	3.352632	244.842105	92600.0
Bethesda Softworks	8	3.900000	514.125000	92200.0
Activision	9	3.455556	193.666667	81700.0
FromSoftware	6	4.250000	1133.500000	76100.0
Sonic Team	9	3.333333	81.888889	72100.0
2K Games	7	3.542857	216.142857	56000.0
Mojang Studios	3	3.800000	944.000000	54400.0
HAL Laboratory	8	3.887500	98.500000	54100.0
Devolver Digital	9	3.533333	326.444444	53900.0
Game Freak	7	3.757143	266.857143	53600.0
Square	7	4.042857	292.714286	52700.0

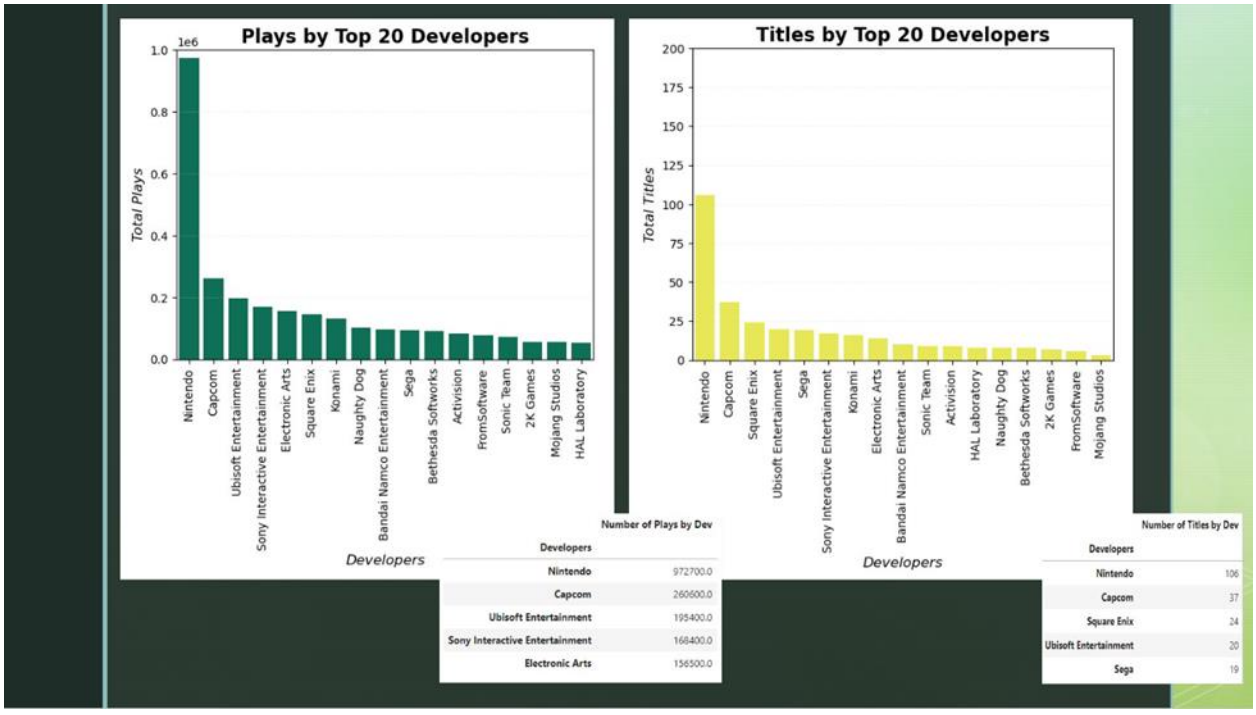
The leaderboard was made with the following code:

```
# Developer Summary Leaderboard

over_200_reviews =game_df[game_df['Reviews'] >= 200]

dev_avg_rating = over_200_reviews.groupby('Developers')['Rating'].mean()
dev_avg_playing = over_200_reviews.groupby('Developers')['Playing'].mean()
dev_sum_title = over_200_reviews.groupby('Developers')['Title'].count()
dev_sum_plays = over_200_reviews.groupby('Developers')['Plays'].sum()

dev_summary_data = {
    "Number of Titles by Dev": dev_sum_title,
    "Average Rating by Dev": dev_avg_rating,
    "Average Number Playing by Dev": dev_avg_playing,
    "Total Number of Plays by Dev": dev_sum_plays
}

dev_summary_data = pd.DataFrame(dev_summary_data)
dev_summary_data.sort_values(by="Total Number of Plays by Dev", ascending=False).head(20)
```

Top 20 Developers by Total Plays and Total Titles

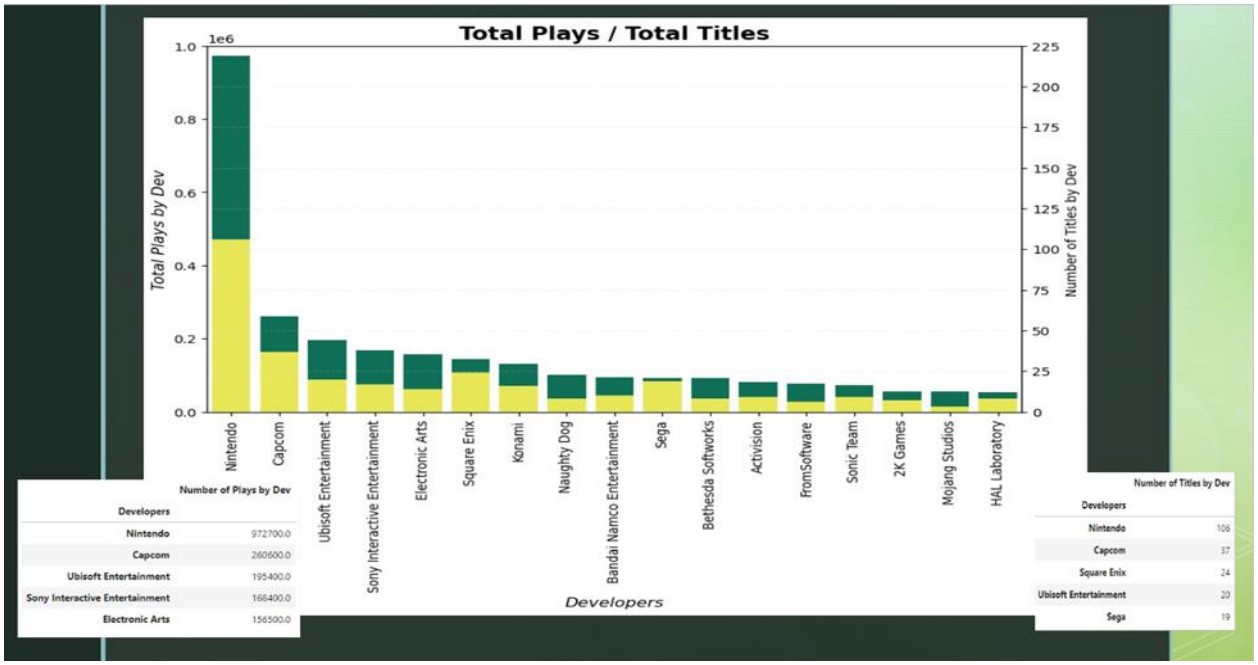
When we first thought of developers, we viewed them as a person or a craftsman, gaining experience like a carpenter, electrician, or perhaps a student going through a bootcamp learning how to code. We found ourselves thinking that the developer will only get better with every copy they sell or every title they produce. Their code will only get more efficient, their graphics more crisp, the stories they tell only more interesting as they learn the industry and how the market reacts to their games, what they like and dislike.

The first set of values we looked at compared developers by total plays, or “copies sold.” This was surely the metric we needed to find the answers in our data story. If you look at the mini leaderboard for Total Plays, you find names you would expect to see: Nintendo, Capcom, Ubisoft, E.A... These are big developers with loads of experience, but when we looked at those values compared to the number of Titles each had produced, all of a sudden our leaderboards shifted. Maybe this wasn’t the only metric of success. Maybe not all developers are created equally.



When we look at a Chart combining the two, the left or green being total plays, and the yellow or the right being total of titles produced, some developers immediately jump out. Our nice even ratios like Nintendo or Capcom aren’t quite reflected across all developers. Some, like Square Enix and Sega, produce a lot more titles than what their total plays reflect. Others, like

FromSoftware or Mojang Studios, have very few titles compared to total plays. This definitely got our analytical minds curious.

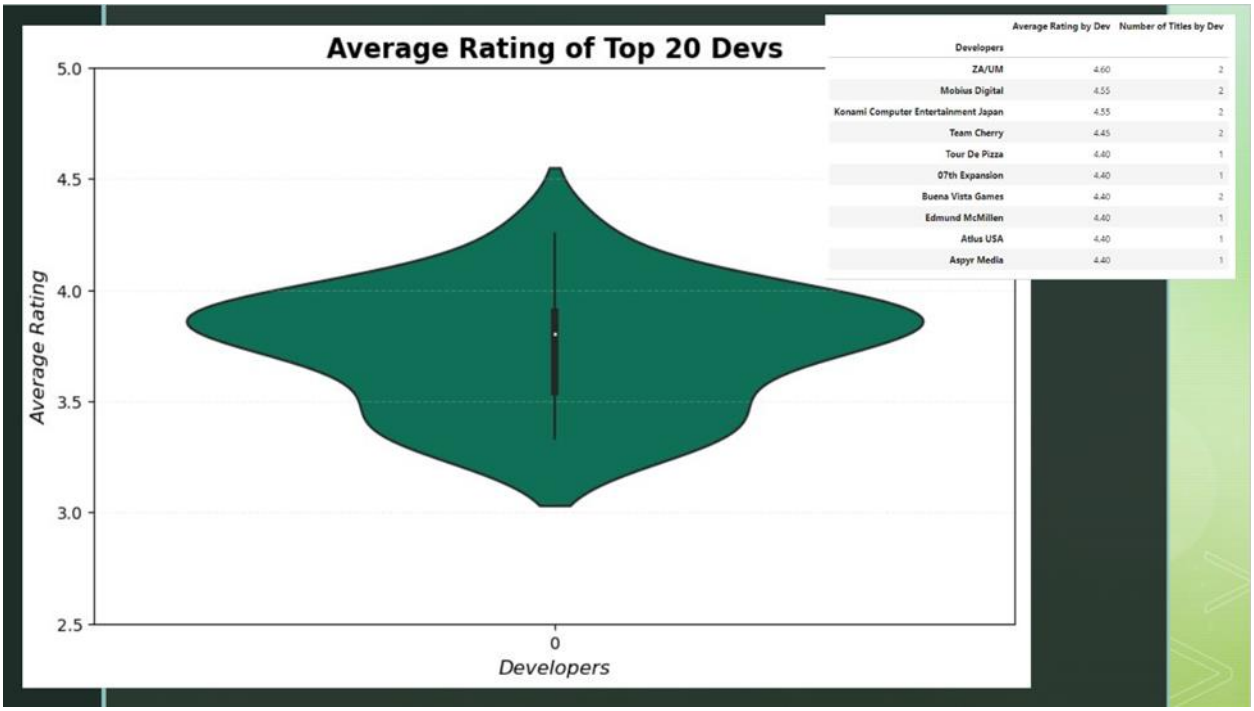


The double bar chart was made by combining two bar charts into one with the following code:

```
# Basic Bars
# https://seaborn.pydata.org/examples/part_whole_bars.html


# Dataset
top_20_by_plays =db[db['Total Number of Plays by Dev'] >= 54100]

fig, ax1 = plt.subplots(figsize=(10, 6))

sns.barplot(x=top_20_by_plays.index, y='Total Number of Plays by Dev', data=top_20_by_plays, color='#007f5f', ax=ax1)

# customizations
plt.xlabel("Developers", fontstyle="italic", fontsize=12)
plt.ylabel("Total Plays by Dev", fontstyle="italic", fontsize=12)
plt.title("Total Plays / Total Titles", fontsize=16, fontweight="bold")
plt.ylim(0, 1000000)
plt.xticks(rotation=90)

ax2=ax1.twinx()

sns.barplot(x=top_20_by_plays.index, y='Number of Titles by Dev', data=top_20_by_plays, color='#FFFF3F')
plt.ylim(0, 225)

plt.grid(color="lightgrey", axis="y", linestyle="--", alpha=0.25)


# show
plt.show()
```

Top 20 Developers by Average Rating

The second set of values we looked at compared developers by average rating. After looking at the data from the first section, we thought to ourselves, I wonder what developers would be at the top of our leaderboards if we compared the average rating of each developer across all of their titles. We then took the top 20 developers sorted by avg rating and created a violin plot.



Developers	Average Rating by Dev	Number of Titles by Dev
ZA/UM	4.60	2
Mobius Digital	4.55	2
Konami Computer Entertainment Japan	4.55	2
Team Cherry	4.45	2
Tour De Pizza	4.40	1
07th Expansion	4.40	1
Buena Vista Games	4.40	2
Edmund McMillen	4.40	1
Atlus USA	4.40	1
Aspyr Media	4.40	1

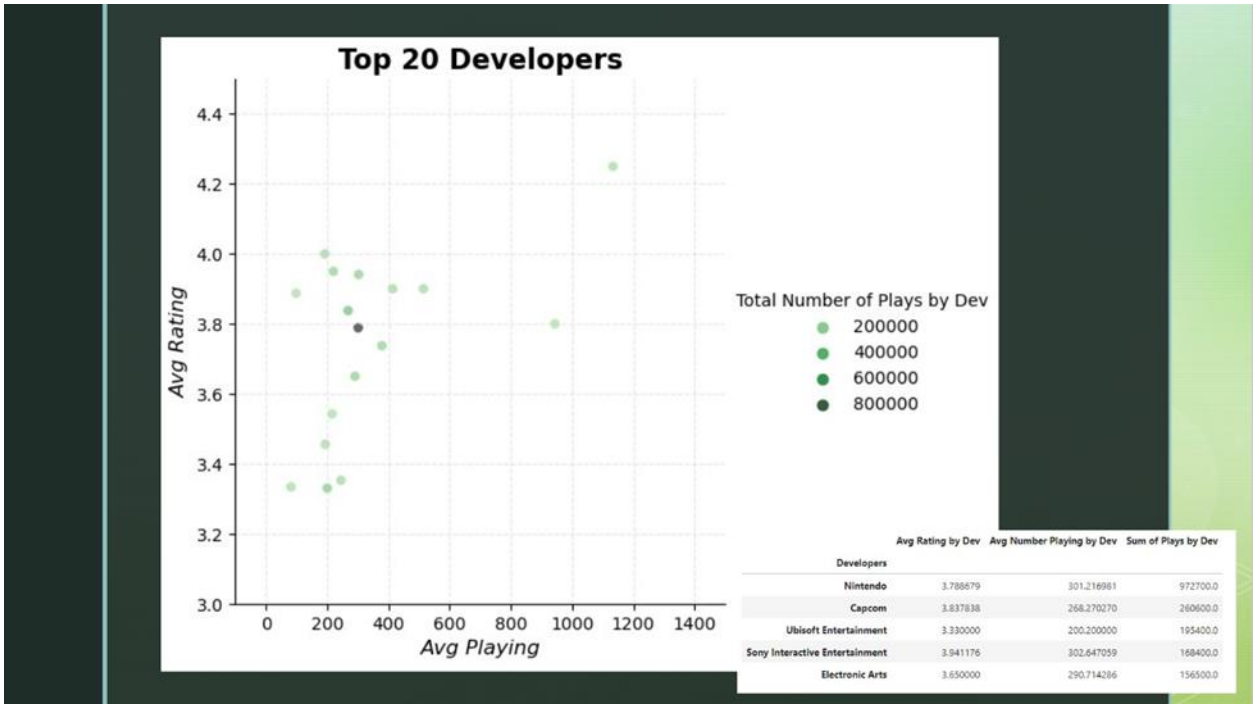
The data shows a min of 3.33, max of 4.25, and a mean of 3.74. It is a fairly even distribution skewing slightly high. If you look at the mini leaderboard, however, something immediately jumps out. All of the best performing developers by avg rating, have very few titles. We then thought, “what in the world is going on here?” Further analysis on this question was required.

The violin plot was made with the following code:

```
# Basic Violinplot
# https://seaborn.pydata.org/examples/wide_form_violinplot.html


# create the plot
plt.figure(figsize=(10,6))
sns.violinplot(data=top_20_by_plays['Average Rating by Dev'])

# customizations
plt.xlabel("Developers", fontstyle="italic", fontsize=12)
plt.ylabel("Average Rating", fontstyle="italic", fontsize=12)
plt.title("Average Rating of Top 20 Devs", fontsize=16, fontweight="bold")
plt.ylim(2.5, 5)

plt.grid(color="lightgrey", axis="y", linestyle="--", alpha=0.25)

# show
plt.show()
```

Top 20 Developers by Average Rating, Average Playing, and Total Number of Plays

For the final part of our developer question, we went back to our first grouping. The top 20 developers by Total Plays and Total Titles. Then, we compared each of their avg ratings on the Y axis, their avg playing on the X axis, and total number of plays illustrated by hue using a scatter plot.



Developers	Avg Rating by Dev	Avg Number Playing by Dev	Sum of Plays by Dev
Nintendo	3.788679	301.216981	972700.0
Capcom	3.837838	268.270270	260600.0
Ubisoft Entertainment	3.330000	200.200000	195400.0
Sony Interactive Entertainment	3.941176	302.647059	168400.0
Electronic Arts	3.650000	290.714286	156500.0

This told us a fairly consistent story of success. Although they didn't have the highest avg rating like the group with lower titles, these developers were definitely doing something right, and it was being mirrored across the grouping. Nintendo, the darkest point on the chart, and 4 out of the top 5 darkest points were safely in the middle of the pack, between 3.65 and 3.94. They had a tried and true formula of success. At this point, we thought it necessary to run regression models for the data set.

The scatter plot was made using the following code:

```
# Basic Scatter
# https://seaborn.pydata.org/examples/scatter_bubbles.html

x = top_20_by_plays['Average Number Playing by Dev']
y = top_20_by_plays['Average Rating by Dev']

# create the plot
plt.figure(figsize=(10,6))
sns.relplot(x=x, y=y, alpha=.75, hue=top_20_by_plays['Total Number of Plays by Dev'], palette='Greens_d')


# customizations
plt.xlabel("Avg Playing", fontstyle="italic", fontsize=12)
plt.ylabel("Avg Rating", fontstyle="italic", fontsize=12)
plt.title("Top 20 Developers", fontsize=16, fontweight="bold")
plt.xlim(-100, 1500)
plt.ylim(3,4.5)

plt.grid(color="lightgrey", axis="x", linestyle="--", alpha=0.5)
plt.grid(color="lightgrey", axis="y", linestyle="--", alpha=0.5)

# show
plt.show()
```

Regression

We chose Nintendo, since it had the most games, for the first regression. We removed the title "The Legend of Zelda: Tears of the Kingdom" as the amount of plays it was supposed to have was inaccurate as the game came out only 1 month before the data was collected for the dataset. We decided to give ourselves multiple options to work with by grabbing the 'Plays', 'Playing', and 'Rating' column and putting them into their own separate variables to be used in the regression later.

```
# Set it to Nintendo
game_dev_Nintendo = game_df_200[game_df_200['Developers']=='Nintendo']

# Wasn't showing the right amount of plays and came out only 1 month before the data was collected
game_dev_Nintendo = game_dev_Nintendo[game_dev_Nintendo['Title'] != 'The Legend of Zelda: Tears of the Kingdom']

# Setting the Plays, Playing, and Rating columns to variables
game_dev_Nintendo_plays = game_dev_Nintendo['Plays']
game_dev_Nintendo_playing = game_dev_Nintendo['Playing']
game_dev_Nintendo_rating = game_dev_Nintendo['Rating']

# Sorting the values and setting it to 99 for to keep with double digits
game_dev_Nintendo_sorted = game_dev_Nintendo.sort_values(by='Rating', ascending=False).head(99)

# Displaying the sorted frame
display(game_dev_Nintendo_sorted)
```

We then needed the variables to be set for our second regression, which was the small and indie developers that had only one title, while still following the 200 minimum number of ratings rule.

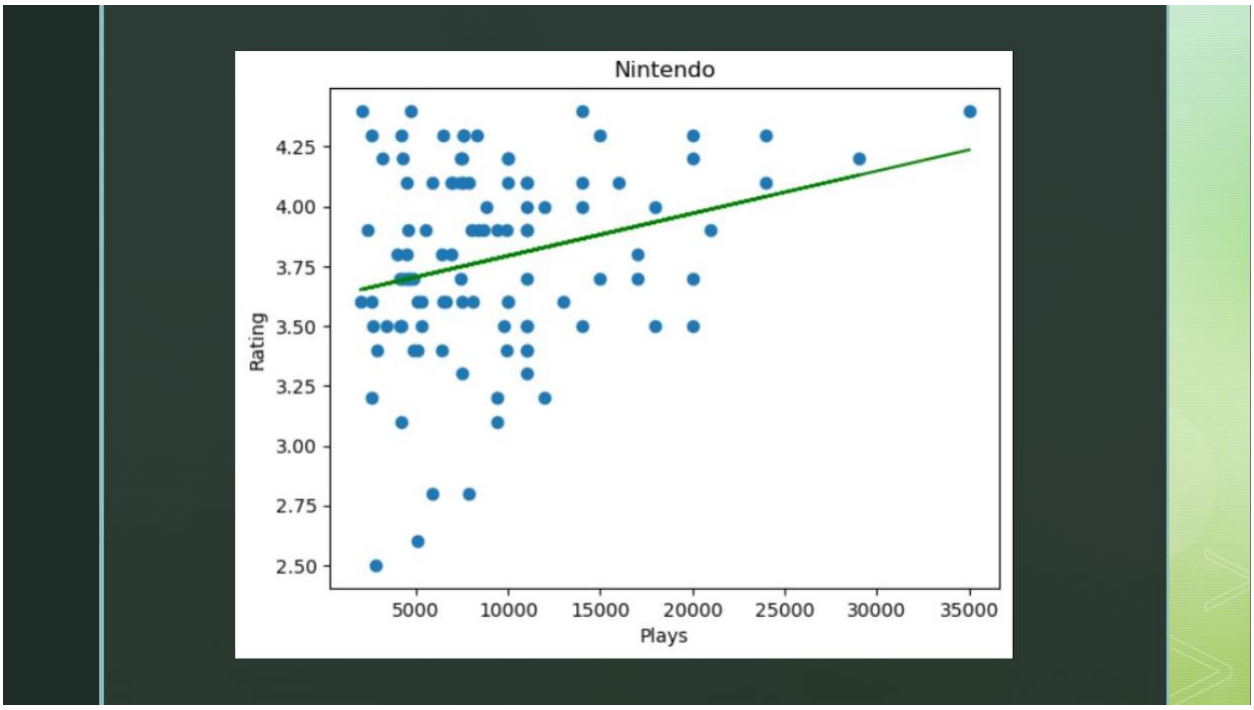
```
# Filtering developers with one game under them with the >=200 dataframe.
game_devs_one_off = game_df_200.groupby('Developers').filter(lambda x: len(x) == 1)

# Sorting the devs based on Ratings, which then displays the top 50.
top_devs_one_off = game_devs_one_off.sort_values(by='Rating', ascending=False).head(99)

# Puts everything together into a single dataframe along with adding 'Plays' for the regression later. Put in drop_duplicat
top_devs_df = top_devs_one_off.drop_duplicates()

# To prepare further for the regression, 'Plays' and 'Rating' need to be assigned as well.
top_devs_plays = top_devs_df['Plays']
top_devs_rating = top_devs_df['Rating']

# Displaying the top 50.
display(top_devs_df)
```

We then made the plots by using the regression plot format in the pymaceuticals module as a reference.

```
# Used the regression plot from the Pymaceuticals as a reference for all of this.
correlation = st.pearsonr(game_dev_Nintendo_plays, game_dev_Nintendo_rating)

(slope, intercept, rvalue, pvalue, stderr) = st.linregress(game_dev_Nintendo_plays, game_dev_Nintendo_rating)

regress_values = game_dev_Nintendo_plays * slope + intercept

# Set the title, labels, and variables that will make the regression.
plt.scatter(game_dev_Nintendo_plays, game_dev_Nintendo_rating)
plt.plot(game_dev_Nintendo_plays, regress_values, color= "green")
plt.title("Nintendo")
plt.xlabel("Plays")
plt.ylabel("Rating")
plt.grid(False)
plt.show()
```

The analysis of the Nintendo regression was based on the “Quantity” part of the thesis. Nintendo had an overwhelming amount of games, but the range of their ratings went from 2.5 to

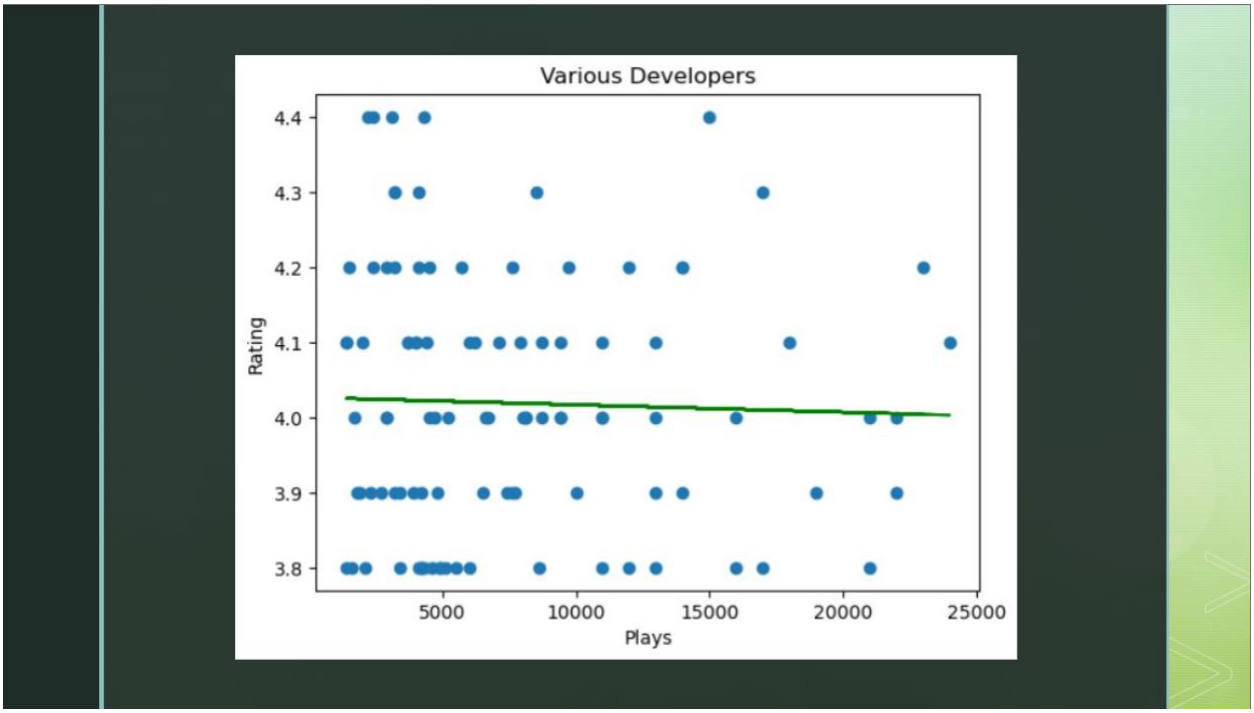
4.4. Many of their games also had somewhat decent reviews, but a low amount of plays. This could be due to their games being on their own separate console, separate from consoles that share games more like playstation or xbox. It could also be due to the fact that they take a safe route with their game; they stick to what the game is known for and try very little to take any risks with any of the franchises that they own. Breath of the Wild was a huge risk but paid off very well considering that it's their highest rated game and it has the most plays. The reason why was because they spent years working on that game before it was finally released. The last several Zelda games performed lower than previous Zelda games, and many of them started to go to the DS. Nintendo hasn't done much with the DS in recent years and they even shut down the Nintendo E-shop as well, which meant that system is no longer supported. They took a risk with the new Zelda game, but it paid off because they took their time. Their other games, however, have grown stale due to not taking risks or taking new routes with other franchises, yet they release so many games. Thus, they focus on quantity rather than quality.



As for the "Quality" side of things, this was the second regression which focused on various developers, all of which have only 1 game under their brand/studio/development provided the game follows the minimum 200 reviews rule. We also simply copied the format that was used for the Nintendo regression and transferred it over. We then did all of the setting of variables and made the necessary changes to the code to create the new second regression.

```
# Same thing as above, except that everything is set for the top individual devs.
correlation = st.pearsonr(top_devs_plays, top_devs_rating)

(slope, intercept, rvalue, pvalue, stderr) = st.linregress(top_devs_plays, top_devs_rating)

regress_values = top_devs_plays * slope + intercept

plt.scatter(top_devs_plays, top_devs_rating)
plt.plot(top_devs_plays, regress_values, color= "green")
plt.title("Various Developers")
plt.xlabel("Plays")
plt.ylabel("Rating")
plt.grid(False)
plt.show()
```

The downside of this graph was that it didn't do as good of a job at telling the audience that indie developers and small developers perform better than a big company like Nintendo, but we did come up with a great way to use the data. The data showed a higher range of ratings, 3.8 to 4.4, yet lower plays than Nintendo. We came to the conclusion that this meant that these games were for Niche audiences, the kind of games that would become or are cult classics, the passion projects of developers that wish to create their dream. These developers are small or alone, so the risk of failure weighs even heavier on them. As such, they have to put in all of their effort when they make these games, even if it only ever reaches a small crowd of players. They also lack the funds necessary for large-scale advertisements along with lacking the same name-recognition that a developer like Nintendo has. As such, the higher ratings are impressive and show their effort along with why they rate higher, in terms of range, than Nintendo.



Call to Action

Our original thesis for this project was: “Developers with fewer titles care more than larger developers and produce better products.” Per our analysis, this changed from “Quality vs Quantity into Customer vs Company Profit.” We decided to view this from two perspectives:

As a customer, indie and small developers make arguably better games, with higher avg ratings for titles, but sometimes they target an extremely niche market, represented by lower avg playing numbers. If they produce a game for your respective market, they are well worth checking out. They make games for a small but happy market.

As an investor or executive, Nintendo and larger developers’ strategy is a better investment. They treat the industry like an assembly line, consistently churning out popular products, while catering to an extremely large market.

Bias and Limitations

Working with this dataset, we discovered some inconsistencies that we believe contain bias and also realized some limitations. To remove some of the suspected bias in this dataset, we decided to only use data for analysis that had 200 or more reviews per title. With a topic which is based so heavily on popularity, we felt one limitation of this dataset is that it should be regularly updated with an API. A static dataset quickly becomes out of date, so it may not be accurate months after the dataset was created.

Another limitation of this dataset is documentation. We need more details on the column headers, such as Plays and Playing. The Plays column from the database we have assumed is the number of people who purchased games or total copies sold. Playing we assumed was a figure representing concurrent players. We would have also liked to see an additional column on whether the titles were linear, live, or limited to see if the title would maintain players for a longer timespan. The last limitation we felt was how the dataset procured its data. We didn’t see any documentation from where the creator sourced the data, so couldn’t confirm its accuracy.

Future Work

There is quite a bit that could have been done and that should be done for this dataset. The first things to mention would be changes to the dataset that would help with the accuracy of the data. Having an API feed that is connected to something like Steam, PSN, or some other game sites that track player movement would be amazing at keeping this dataset accurate and concurrent. An analysis of the bias or the addition of a rule requiring a minimum number of reviews before the rating is shown, such as, a game having a rating of 5.0, but only 9 reviews. Another useful addition would be a column that shows the number of playthroughs of a game/completions of a game. Separating game developers from game producers and putting them into their own separate columns is more of a quality of life fix that would have made cleaning up the data and analyzing the data a lot easier. The last addition would be a column that shows the approximate size of the development team. This would have allowed for some more unique analysis that would have provided results that tailor more to our thesis.

There were some alternate routes that we could have taken for the regressions. The first route we considered to take was a regression of Nintendo vs FromSoftware. The first problem with this route was the 200 minimum reviews rule and the developer column data cleaning, more specifically the fact that they both lowered the amount of games under FromSoftware from over twenty to six. FromSoftware wasn't consistently listed as the developer and the data cleaning we did removed the producers, which in some cases also removed FromSoftware. This would require for every developer cell that has FromSoftware in it to be put into a dataframe before the data cleaning is done. Then, we would have to ignore the 200 minimum reviews rule for both Nintendo and FromSoftware, which would have made for some amazing regressions that would have made proving our thesis easier, but we needed to maintain consistency, so we decided to forgo that route. Another route could've been the use of time; the use of release dates to show a growth or decline of the developers ratings. The problem is, release dates cannot be used on a regression unless they are converted into ordinal time, which is not a readable unit of time. That route also had to be abandoned. The final route, which was actually the first route before settling on the current regressions, was to use plays vs playing. This resulted in no real regression on either Nintendo nor the Various developers, so this was abandoned as well. These routes all would've been neat in theory, but they had far too many complications and conflicting aspects to be used.

Work Cited

Chaves, Matheus Fonseca. *Popular Video Games,* 11 July 2023, https://www.kaggle.com/datasets/matheusfonsecachaves/popular-video-games