# Switch Games Database

## MSIN0166: Data Engineering

Group 6


Word count: 3998

(excluding front page, content page and references)

## Contents

## 1. Introduction

Nintendo(WIKIPEDIA, 2015)is a Japanese company that mainly focuses on developing video game software and hardware. This company is one of the three majorities of the video game industry, and currently, it is the modern video game industry pioneer. Their most popular product is Switch, a console released in March 2017, which is a revolutionary console. Switch's features are its design, which adopts the integrated design of the home console and handheld console. This design arguably redefines gaming. Players can play Switch games anywhere, such as at bus stops, planes, trains, or even in a doctor's waiting room due to its portability.

Five years past from 2017, the Switch has sold about 107.2 billion units in its lifetime. This impressive selling result made Nintendo Switch turn to be the seventh video game platform to sell more than 100 million units. Other platforms such as PS2, DS, Game Boy, PS4, PS1, and Wii are well-known. This indicates that Nintendo Switch is promoted to be one of the most popular products in the gaming world. A lot has changed for Nintendo since the Switch arrived in 2017; Nintendo Switch's derivatives have also kept up with trends and come into the public eye. One of the most important is Switch Games. Players can experience a variety of games with Switch, top switch games are simply stunning and profitable.

Metacritic(WIKIPEDIA, 2001) is a website that collects reviews on games, which will integrate the scores of each reviews and make a final mark on the games. The final score is worthy as a reference by many players.

This project will scrape data from Metacritic about the top hundred switch games and data from YouTube. We aim to use those data to create a database, which will be convenient for further data expletory and pipeline studies.

## 2. Data Extraction

We used two methods to grab data to ensure the information is realistic and up to date in the database. The first method is called web scraping, and we used the first web crawler and second web crawler in this project. The web crawler is a simulation way as a browser to open a webpage to extract the data we want from that page. As long as the data can be accessed through a browser, we can acquire it through the web crawler. The first web we call here stands for the first web page we reach. The second web stands for the web page we open after clicking through the link from the first page. API is the second method we have used to help in data extraction. An API(LLC, 2022) is simply a programming code that will allow data transfer between software. This method is formal and under license, as most platforms provide API to the public. We will show how we played with these two methods to catch data in more detail.

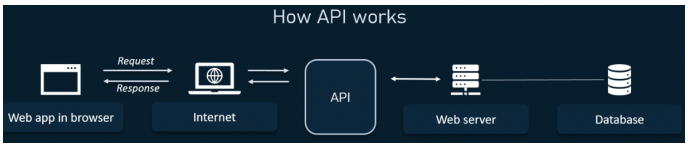


Figure 1: API WORK Flow Diagram

## 2.1 *Metacritic*

Metacritic's official website will be the primary data source in our project as they can provide millions of data under different filters. We would like to catch the data about games that can play on Switch, and we will not set a time section in this project. These selections will bring us to a web page that lists all Switch games under user scores from top to bottom. This web page will be the first web page we want to focus on under the first web crawler. We aim to collect the top 200 game names that stand as the primary key for the game table, which we will introduce in the Relational Database section. To prepare the work, we need to understand the URL of the web page. Since we want to collect the first 200 games and the web page only contains 100 games per page. Therefore we need a total of 2 web pages during the first web crawler. In this project, we found that the URL difference between the first and second of the first web page is the page number printed in the URL. We used a loop code to achieve two web pages. As we have analyzed the web page, we are ready to work.

The first stage is to retrieve data. To archive this aim, we need to send a request to the target web page through an HTTP. We used a package called requests from the python library and included headers in the request. After completing this step, we will get a response; The web page's content has been obtained. The second stage aims to parse the content we just archived, which could be done by using another package called beautifulSoup. The content we get is in HTML format; we parse with it and extract critical data such as game name, game ranking, release date, and the second weblink through regular expressions. The final step is to save the data. We hold the data in CSV format locally.

We have done the data collection for the first web page. As we have the link to the second web page for each game, we are ready to work on the second web crawler. Similar to working as the first web crawler. Firstly, we have to understand the structure of the HTML page. The data such as user score and the number of players are the main points we want to focus on. We used Xpath instead of regular expressions as we wanted to be more specific and accurate to the resulting data. By the end, we collected data over the game name, game ranking, release date, and scores of 14 columns of relational data of the top 200 Switch games listed by the user score from Metacritic.

| # | Column |
|---|---|
| 0 | ranking |
| 1 | game_name |
| 2 | link |
| 3 | rating_score |
| 4 | release_date |
| 5 | metascore |
| 6 | critic_reviews |
| 7 | user_score |
| 8 | user_score_rating_num |
| 9 | developer |
| 10 | game_genre |
| 11 | num_players |
| 12 | Cheats |
| 13 | rating_number |

Figure 2: Columns From Web Scraping

### 2.2 *YouTube*

YouTube is the second primary data source in our project as it is the most popular and largest video sharing platform in the world. Nintendo Switch has its official account on YouTube, where people can find the newest information on YouTube, such as the new game released and game updates. They can comment, share and favorite the videos instead of just watching. They could even upload their own clipping videos to comment, introduce, reaction to the relational games. Therefore, we could find millions of relational videos by keyword searching. YouTube is also providing the API to the public. The data from YouTube could be one way to reflect the popularity of the Switch games as it gives data such as the number of views, likes, comments and favorites.

Therefore, we will use YouTube V3 API through the second method. In this part, the major function to use is called the V3 Video function, which returns us the relational video information. However, it requires video IDs for each relational video. To obtain video IDs, we could use the function called Search. The name of each game will be the keywords as we suppose most game relational videos must contain the game name in the title to relate and increase exposure opportunities on YouTube. We have to clean up the data we scraped by web scraping before the API work starts. Such as dropping all the nulls and merging the two datasets by game name as we need to create the top 100 Switch games list from the data we got from Metacritic. We used a loop to achieve the video IDs of the top 50 relational videos from the top 100 games in the code. After we got the video IDs, we could use the V3 Video function to generate the relational video data, including view count, like count, favorite count and comment count of each game.

By the end of this Data Extraction stage, we gathered 3 CSV datasets related to the top 100 Switch games ranked by Metacritic. The next stage is the process of Data Storage which AWS S3 does.

## 3. Data Storage


Figure 3: Amazon S3 Work Flow

This project will focus on using AWS; the S3(Amazon, 2016b) service will be used as the primary data storage. We will put the 3 CSV files into S3 storage in this project. To access AWS, we first need an account. After a few simple steps to create the AWS account, we will be able to view the AWS services with hundreds of them around several different science regions. We need the service called S3, as already mentioned above. Creating a bucket will be the first step under this service, which just sets a name, and the rest are set by default and recommended. The Aws region will follow the selection areas by the default; we just need to make sure it belongs to EU(London) EU-west-2. After creating the bucket, we will be ready to go through the code to upload local CSV to AWS S3 storage. Several key points are needed, such as username, access key id and secret access key, bucket name, region, etc. These are available to check from AWS, such as the access key id and secret access key by clicking the Security credentials.

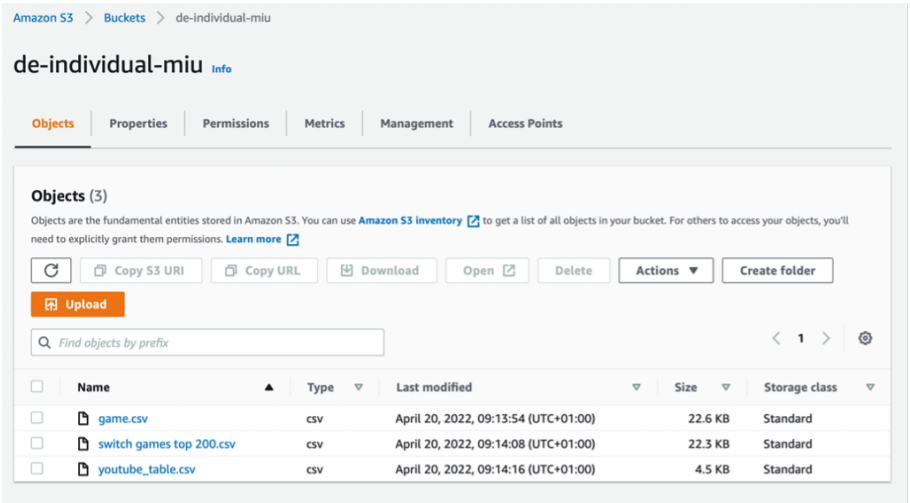


Figure 4: Screenshot of Amazon S3 Under Bucket de-individual-miu

As the codes are set correctly, we have uploaded the CSV files to the AWS bucket under the S3 storge called de-individual-miu. Up to this stage, we are ready to connect our data from S3 to a database such as PostgreSQL. We will claim this in the next section.

## 4. Database

### 4.1 *Connect to S3 Storage*

The first step in this stage is to read the data from S3 storage. This could be done by using a defined function. The main points in the programming code, such as the bucket name, access key id and the secret access key are the same as when we uploaded the CSV file to S3. Since the defined function and the main points are correct, we read CSV successfully in Python.

### 4.2 *Clean and Prepare Data*

The first CSV we have read contains 200 entries and 5 columns which is the data we gained from the Metacritic first web. The game_name, ranking, and release_date will be the columns we are interested in. The second CSV is the data we obtained from the second web from Metacritic. It contains 200 entries as well, but ten columns in total. Taking a general look at these data, we found some columns are pending as the same results and some show unclear results. We will drop these columns after we merge CSV1 and CSV2. To be noticed, The column called game_genre is a column that shows multiple kinds of genres, which separates by commas. We used the function split to separate them and take the first and second into account of the whole data named game_genre1 and game_genre2. Another column, release_date, is a time string which contains the date, month and year in one column. We used the same idea to separate them. After dealing with these issues, we need to generate three Data Frames based on the current data, corresponding to the three tables in the Schema we designed in section4.3. These three tables have named a game, developer, uer_review_web.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 20 to 239
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   ranking       200 non-null    object
 1   game_name     200 non-null    object
 2   link          200 non-null    object
 3   rating_score  200 non-null    object
 4   release_date  200 non-null    object
dtypes: object(5)
memory usage: 9.4+ KB
```

Figure 5: Information On CSV1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   game_name          200 non-null    object
 1   metascore          200 non-null    int64
 2   critic_reviews     200 non-null    int64
 3   user_score         200 non-null    object
 4   user_score_rating  199 non-null    float64
 5   developer          200 non-null    object
 6   game_genre         200 non-null    object
 7   num_players        186 non-null    object
 8   Cheats             200 non-null    object
 9   rating_number      192 non-null    object
dtypes: float64(1), int64(2), object(7)
memory usage: 15.8+ KB
```

Figure 6: Information On CSV2

The last CSV is the YouTube data which contains 100 entries with six columns. It is the most straightforward data to deal with; it directly corresponds to the fourth table in the Schema. All we need to append a new column named YouTube id. The id column is also required in the other three tables, such as game_id, developer_id, and uer_review_web_id, which will help us connect tables in further studies, such as SQL queries. By the end of this step, all the data we need has been cleaned.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   youtube_id       100 non-null    int64
 1   game_name        100 non-null    object
 2   view_count       100 non-null    int64
 3   like_count       100 non-null    int64
 4   favourite_count  100 non-null    int64
 5   comment_count    100 non-null    int64
dtypes: int64(5), object(1)
memory usage: 4.8+ KB
```

Figure 7: Information On CSV3

## 4.3 *Schema*

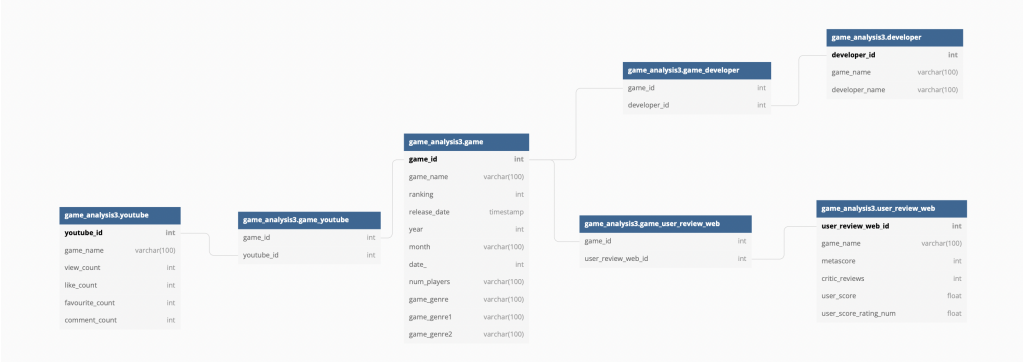


Figure 8: Schema Diagram

Database server contains multiple databases, which could contain serval schemas with various tables. The Database Schema Diagram we designed contains seven tables. Four tables are the major tables, and three are the connect tables. The table called game is the central table, developer,user_review_web, and youtube tables are the main table. The rest tables, game_developer, game_user_review_web and game_youtube, are all connection tables. These tables are helping us to join tables in further studies, as the game_id play as the primary key in the game table and the foreign key in the three connection tables. For instance, from the Schema figure, we can see the game table could connect to the youtube table due to the connection table between them. We also designed the type for each attribute in each table. For example, the game table contains 11 columns. For attributes such as game$_n$ame, we define it as varchar, release_data is like timestamp, ranking like integer, etc. As long as we define these correct, we will be able to create the Schema in our database. We have named the Schema game_analysis3 in this project.

### 4.4 *Import Data to PostgreSQL*

In this project, The database server that we will use is PostgreSQL, a relational database management system based on AWS. This time the service we choose from AWS is called Amazon Relational Database Service (RDS)(Amazon, 2016a). From my understanding, RDS is a relational database with multiple functions. Using this database saves us much time; it could help us with database installation of backup/restore etc. RDS supports MySQL, Amazon Aurora, PostgreSQL, Oracle, Microsoft SQL Server and MariaDB engines. Obviously, PostgreSQL will be our target. For the database creation, we can follow the way that Week 9 lecture slides are straightforward. We used the programming code to connect to the database. The keys to joining the correct database are database name, user name, password, host, and port. These keys could be found under the database information. Before inserting the prepared data into the database, we must convert the four tables ( game, developer, uer_review_web and youtube_table) from Data Frames into tuples. We also need to generate three connect tables which are game_developer,game_user_review_web and game_youtube. These tables include two columns, game_id with developer_id,user_review_web_id and youtube_id, from 0 to 100, which correspond to the three connect tables described in the Schema. We also need to transform these three Data Frames into tuples before inserting them into the database. Everything will be ready to query from the database as we successfully used execute function in Python to insert the actual data into the specific Schema in our database.

## 5. Data Cleaning

We will mainly focus on cleaning data. Firstly, we have to merge all the four data tables (game, game_developer, youtube, user_review_web) into one final combined data called GAME. In this GAME, we will need to delete all the columns that play as id, such as game_id,developer_id etc. We need to correct the column type for the left columns, such as ranking, year, date_, and user_score are in the kind of object. We used Pandas' astype functions to change columns to integer and float. We also generate a new column called quality based on the metascore by the end of data cleaning. If the metascore is between 90 to 97, we define the game as quality 1; the score between 86 to 90 will be quality 2; 0 to 86 will be quality 3. The description above is all the data cleaning work we have done; one thing to be noticeable is after we dropped the null values, only 95 entries are left in the GAME table.

The following step, we will make a test set to meet a few conditions:

1. The test set must be random.
2. The test set must be representative of the dataset.
3. The size of the test set has to be around 20% of the dataset.

From checking the distribution of the quality, we found it is not balanced. We used the function called stratify to help with this problem. The function random$_s$tate and test$_s$ize will help in conditions 1 and 3, respectively. If everything is set correctly, we can ensure the test set follows the stratified sampling based on the quality category. The test set section has done now.

## 6. Exploratory Data Analysis

This section will focus on exploratory analysis of the data we have cleaned and prepared on the database to explore business thought. We will first focus on 5 queries by using SPARK SQL and display 5 data Visualization on the data. Finally, we will pipeline the machine learning models.

### 6.1 *SQL*

Query 1: Retrieve a list of all games from Nintendo (developer)with their ranking order.

| game_name | ranking | developer_name |
|---|---|---|
| The Legend of Zel... | 1 | Nintendo |
| Super Mario Odyssey | 2 | Nintendo |
| INSIDE | 13 | Nintendo |
| Ori and the Blind... | 19 | Nintendo |
| 13 Sentinels: Aeg... | 29 | Nintendo |
| SteamWorld Dig 2 | 42 | Nintendo |

Figure 9: SQL Table For Query1

Nintendo developed six popular games from the top 100 Switch games from the above result. The ranked first and second games are all from Nintendo. The lowest-ranked game from Nintendo developed still ranked 42nd among the top 100 games.

Query 2: Retrieve the Top 10 average like count by each developer.

| developer_name | avg_like_count |
|---|---|
| Draw Me A Pixel | 2.917945E7 |
| Team Salvato | 1.239445E7 |
| DotEmu,Seaven Stu... | 1.22156E7 |
| Studio MDHR | 9971000.0 |
| Supergiant Games | 9577200.0 |
| Capy Games | 8756000.0 |
| Infinite Fall | 8331350.0 |
| Matt Makes Games ... | 6634350.0 |
| Nintendo,HAL Labs... | 4862500.0 |
| Giant Sparrow | 4637400.0 |

Figure 10: SQL Table For Query2

As we can see, The game's average like count from the first and second developers is outweighed by the other eight developers. The first developer, Draw Me A Pixel, is even more on average like count than the first developer, with almost 2.4 times.

Query 3: Retrieve the sum number of view count over 5 billion by each developer.

| developer_name | num_view_count |
|---|---|
| Team Salvato | 5441313750 |
| tobyfox,8-4 | 9214791250 |
| Infinite Fall | 9488786250 |
| Nintendo,HAL Labs... | 10663402500 |
| Matt Makes Games ... | 14174572500 |
| Draw Me A Pixel | 19840091250 |
| Next Level Games,... | 23616037500 |
| Studio MDHR | 29074003750 |
| Capy Games | 37127110000 |
| Supergiant Games | 49420496250 |
| Nintendo | 53728223500 |

Figure 11: SQL Table For Query3

Eleven developers have over 5 billion views count on their games. Nintendo ranked first, and its view count number is about ten times that of the last developer, Team Salvato.

Query 4: Retrieve a list of average metascore by each year for developer Nintendo (order by year).

| year | avg_metascore |
|---|---|
| 17 | 93.5909090909091 |
| 18 | 92.0 |
| 19 | 90.0 |
| 22 | 89.0 |

Figure 12: SQL Table For Query4

The average metascore for Nintendo's games is decreasing from 2017 to 2022, about 1 point per year. The highest point was about 93.6 in 2017, about 4.6 higher than five years later.

Query 5: List all the games with the number of players, which are up to 4, and their corresponding to the release year and the first game genre.

| game_name | year | game_genre1 |
|---|---|---|
| Dragon Quest Buil... | 19 | Role-Playing |
| Guacamelee! 2 | 18 | Action |
| Super Mario Maker 2 | 19 | Action |
| Divinity: Origina... | 19 | Role-Playing |
| Super Mega Baseba... | 20 | Sports |
| Monster Hunter Rise | 21 | Role-Playing |
| Overcooked! All Y... | 21 | Miscellaneous |
| Super Mario 3D Wo... | 21 | Action |
| Heroes of Hammerw... | 20 | Role-Playing |
| Sonic Mania Plus | 18 | Action |
| Diablo III: Etern... | 18 | Role-Playing |

Figure 13: SQL Table For Query5

There are eleven games where the number of players is up to four, and most of them belong to role-playing and action games. Their release year is between 2018 to 2020.

## 6.2 *Data Visualization*

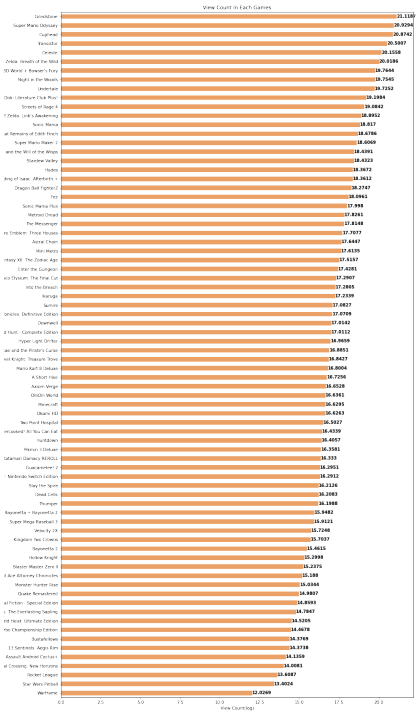


Figure 14: View Count In Each Games

The first graph is a horizontal bar chart showing all the games' YouTube view counts. Grindstone is the top-ranked overall game with about two times the view count as the last game, Warframe. The figure shows that the first tier group is in the top 9 games with a salty view count than the 10th games. The decreasing slope is not high overall for the second-tier group, and the view count is not too different between adjacent ranked games.

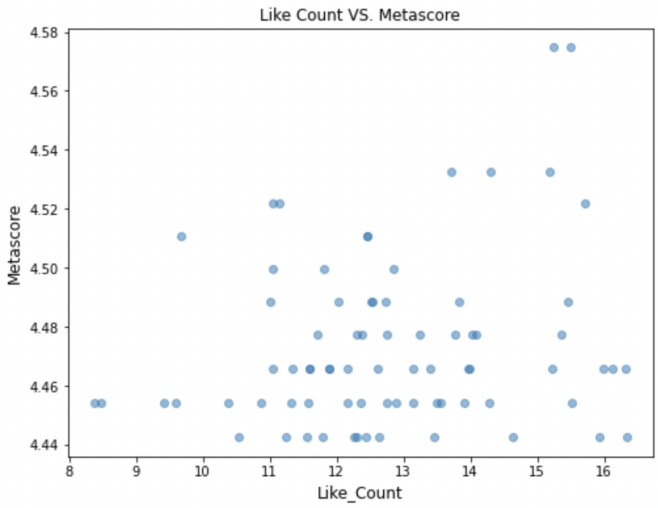


Figure 15: Like Count VS. Metascore

The second graph is a scatter graph between like count and Metascore. as the like count increases, the Metascore also increases. These two terms are showing as a positive correlation. There are some outliers displayed because of the size of the data, and the two attributes are from different platforms. If we have more data in the future study, we would like to drop the outliers, making the figure clearer.

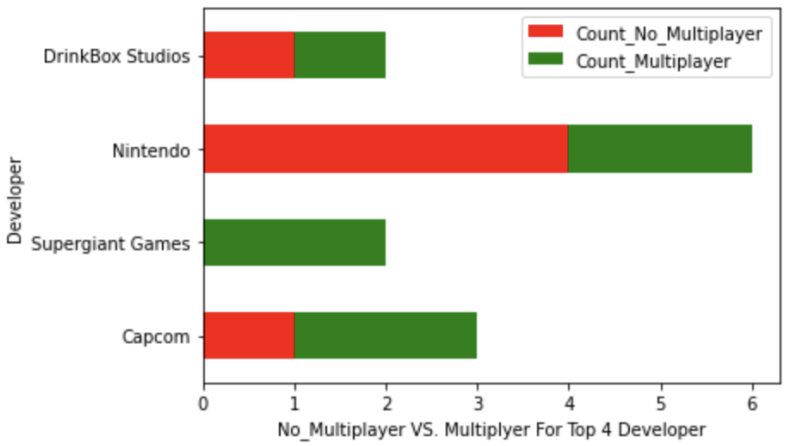


Figure 16: Number of Player Tyeps for Top 4 Developer

The third graph shows the comparison in the type of the number of players from the top 4 developers. The top 4 developers are the top 4 developers that released the most games in our data list. The leading developer is Nintendo, with six games in the count.

Among four games are not allowed multiplayer, and two games are allowed. The second developer is Capcom owned three games; 2 games belong to the multiplayer type, 1 is the single-player game. The rest two developer have the exact count of games with 2. DrinkBox Studios are balanced as one game type is a single-player one game type is a multiplayer game. However, the developer Supergiant Games only has two games ranked on the list with multiplayer type.

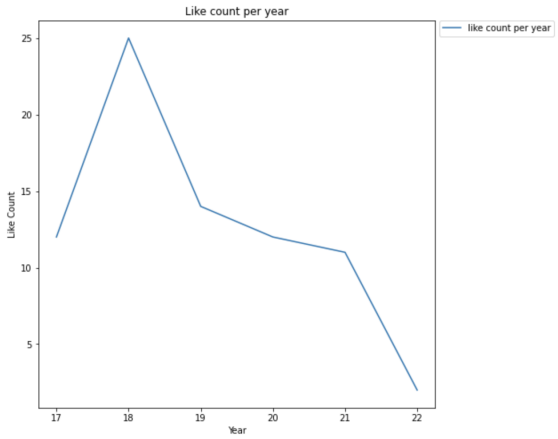


Figure 17: Scatter Graph with Gross Profit and Vote

The fourth figure is a line graph which shows the like count flow on YouTube every year from 2017 to 2022. The like count of Switch games reached its peaked in 2018 from 2017. However, the number of like counts dramatically decreased in the next year. The like count trend between 2019 to 2021 has a smooth decrease slop. The lowest like count is in 2022, but we believe this is the limitation as now is only April, which two-thirds of the year has not reached yet.

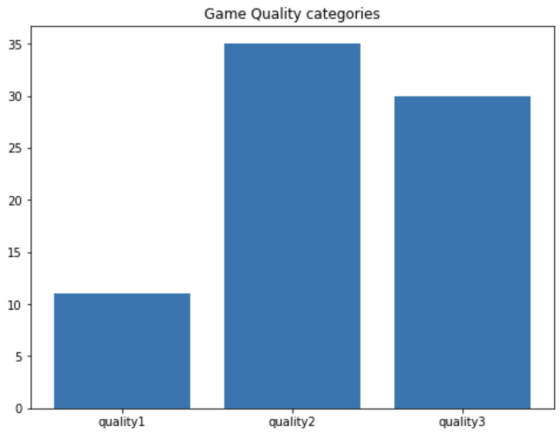


Figure 18: Scatter Graph with Gross Profit and Vote

The Figure is showing the game quality distribution. We can see from the graph, most of the games are belong to quality 2 and 3 with quality 2 ranked as the top.

### 6.3 *Machine Learning Pipeline*

Streaming workflows with pipelines is essential to apply in the project. It is helpful because when dealing with the training sets such as character normalization, we will need to resue all the parameters to avoid data leakage. The Pipeline can eventually work with many algorithmic models, such as classification and normalization, to form a typical machine learning workflow. In the Pipeline, we can directly call the function to the train and predict our models. The parameters could be selected from the pipeline work combined with the grid search. The above two points are beneficial that bring from apply Pipeline to the machine learning.

We have already cleaned and separated the data into train and test sets in section 5. Therefore, we only need to separate the train and test set into x and y variables before working on models. We have to make sure the index of the x and y on the train must be identical, which applies to the test sets as well. In our project, the length of the dataset is only 95 as we have dropped null values. The train set size is about 76 entries, and only 19 entries are in the test sets.

The Pipeline will be applied to the train and test sets as, firstly, use the function called StandardScaler to standardize each attribute from the datasets. Secondly, we will use another function called PCA to compress the original dimensions. Thirdly, we are ready to use the models. It is worth mentioning that the first and second steps act as a transformer which includes the fit and transform methods, but the last step is the Estimator, in which the transform is not a required method is this step. As we are setting the code correctly, the final step is to call the Pipeline.fit function to the training set and the function Pipeline.score to the testing set to predict and score.

We applied six models to the datasets: Linear Regression, Logistic Regression, Random Forest, Decision Tree, Extra-Trees, and Gradient Boosting. Except for the first model, the rest models obtained a test accuracy of roughly $70\%$ to $80\%$. The Linear Regression model got less than $50\%$. We believe the reasonable cause of the extreme value problem is the small size of the dataset. The size of 95 entries is extremely small to be a dataset to play on machine learning.

## 7. Conclusion

We have collected data using different methods from different resources to create a database that allowed us to develop pipeline studies for the future. We used web scraped to gain data from Metacritic. We also used another method called YouTube API to scrape data. These data are being well designed into the database under different tables. We have done some data visualization to show how well the data is being used and queried using SQL from the database. For each query, we have saved them into a data storage format called parquet. By the end of the project, we also showed some model examples.

### 7.1 *Limitations*

In this project, we have not touched the part of the pipeline due to the data type. The data from the website is not daily data. The website will update its data once a year because our section is the year ranking. We highly recommend changing the data from the website to a daily or weekly ranking for future reference. We can focus on the real-time updated date as we have a short-time updated data. If we achieve this point in the future study, we would like to apply the ELT pipeline, which requires airflow and docker to work together. For example, we will be using airflow connections to manage and connect to our database. The data will be saved to the S3 storage as in this project. Used PostgreSQLOperator to operate the database and used Xcom to pass data between different tasks. Those are the future work if we have more time.

## 8. Appendix

Link to the repository:https://github.com/dmiao123/DE-Individual-

## References

Amazon (Feb. 2016a). *Amazon RDS*. URL: https://aws.amazon.com/free/database/?trk=43bd568d-279c-4acf-be5b-052eb10c34b9%5C&sc_channel=ps%5C&sc_campaign=acquisition%5C&sc_medium=ACQ-P%7CPS-GO%7CBrand%7CDesktop%7CSU%7CDatabase%7CSolution%7CGB%7CEN%7CText%5C&s_kwcid=AL!4422!3!549070929503!e!!g!!amazon%20database%20server%5C&ef_id=CjwKCAjwjZmTBhB4EiwAynRmDxdIal3MN-D0rPOF6Uat9_kbN8HrZMeODgm4oSJvhAtApMwxTpR1fxoC_FIQAvD_BwE:G:s%5C&s_kwcid=AL!4422!3!549070929503!e!!g!!amazon%20database%20server.

— (Feb. 2016b). *Amazon S3*. URL: https://aws.amazon.com/cn/s3/.

LLC, MuleSoft (Jan. 2022). *What is an API? (Application Programming Interface)*. URL: https://www.mulesoft.com/resources/api/what-is-an-api.

WIKIPEDIA (Jan. 2001). *Metacritic*. URL: https://en.wikipedia.org/wiki/Metacritic.

— (2015). "Nintendo". In: *From Wikipedia, the free encyclopedia*.