Steam

The Gaming world

**Team Members:**

Zheng Qi

Oscar Hui

Chad Molenaar

Joe Quinn

Contents

[ETL Project Proposal 2](#_Toc66024473)

[Project Overview 2](#_Toc66024474)

[Extract - Proposed data sources: 3](#_Toc66024475)

[Transform - Proposed clean-up and analysis: 3](#_Toc66024476)

[Load - Data storage: 4](#_Toc66024477)

[Potential limitations: 5](#_Toc66024478)

[Steam Data Generation Project 6](#_Toc66024479)

[Executive Summary (ETL Process) 6](#_Toc66024480)

[Workflow Process – Initial ETL Overview 6](#_Toc66024481)

[Dependencies 7](#_Toc66024482)

[Part One: - Genre’s and Feedback 8](#_Toc66024483)

[Extraction Process 8](#_Toc66024484)

[Transformation Process 8](#_Toc66024485)

[Loading Process 9](#_Toc66024486)

[Summary ERD’s from the EDA (Part One) 10](#_Toc66024487)

[Part Two: - Steam Game Price Index 12](#_Toc66024488)

[Extraction Process 12](#_Toc66024489)

[Transformation Process 12](#_Toc66024490)

[Loading Process 13](#_Toc66024491)

[Summary ERD’s from the EDA (Part Two) 14](#_Toc66024492)

[: - 14](#_Toc66024493)

# ETL Project Proposal

## Project Overview

* What is your question of interest?

How can both gamers and game publishers access data throughSQL/MongoDBto better understand and/or for analyses for information such as:

* + Steam game sales?
  + Most successful publishers?
  + Most popular genres?
  + Size of online player base?
  + Regional and GDP per capita players.
* What is the expected outcome of your ETL project? (i.e., how will this data ETL project add value?) Think about how your ETL project will enrich the data and how this data might create value.

As a group, we are looking to allow both to have the information available to them that will help them analyse and extract data. Through the extraction, transformation and data storage processes used, anyone will be able to save resources such as time, money, and trusted information. An important value-add for both gamers and game developers.

This data will essentially be a summary of multiple different data sets to give an accurate picture of the game sales of Steam.

Game publishers can use this data to gain a better understanding of what the most popular genres are so they can choose to develop a game targeting a wider audience or aim at a more niche fanbase. The size of the online player base can help both gamers and publishers.

Gamers can use this data to look at the popularity of different publishers so they can get a better idea of how good the game will be before making a purchase. In addition to this, gamers can use this to help them decide on which game to buy. This is especially useful for users who want to be a part of an online community and play in larger online games.

## Extract - Proposed data sources:

* Where is the data located?
  + Data sources are for Part One are steamspy and steamstore.
  + For Part Two
* What are the data set formats?
  + Part One: - Json.
  + Part Two: - .csv file
* How will you get this data?
  + Combination of API and downloaded data from steamspy and steamstore, .
* What are your assumptions about the data?
  + Potentially messy and may require pretty print. Utilising JSON can have its pitfalls as it comes out as a dictionary.
  + CSV files tend to hold a lot of duplicates, misspellings, and null values as examples**.**
  + We are additionally holding the assumption that it will hold the Regional and GDP data we are looking for; however, this will be identified with further investigation of other potential databases/sets.
* What does it cover?
  + We are expecting to locate common values or headers (such as geographical data or demographical data) that will enable us to answer the above questions through the cleansing and transformation process.
* What is it missing?
  + Accurate data on owners numbers or I.D.’s for games is an expectation of missing data.

## Transform - Proposed clean-up and analysis:

* What are the transformations you will apply to the data? (e.g. filtering, aggregation, derived columns)
  + Filtering, aggregation.
* What steps will you take to clean the data and ensure its validity? (e.g. messy data, duplicated data, incorrectly formatted data)
  + Drop the duplicated data. Convert the data to the right format.

* How will you identify potential issues with your data sources? (e.g. exploratory data analysis, simple statistics etc)

* + Exploratory data analysis.
* How will the data be integrated? (e.g. joins, merges)

* + Mergers and joins.
* How will you apply these transformations (e.g. jupyter notebook, python script)
  + Pandas, Jupyter notebook, Json, all are listed under dependencies.
* IMPORTANT → Why did you apply these transformations? How did this enrich your data?
  + The data requested through API is quite messy and non-structured. To investigate the relationship between different factors, the data needs to be converted from non-structured data to structured data.

## Load - Data storage:

* What type of database (relational, document) will you store the data?
  + Originally we had planned on using an SQL database, however given the parameters and data found, we decided to use Mongo.
* Why did you choose this database over another database?
  + Structured data for easier comparison was the original thought for SQL, however due to the use of Json, we thought that Mongo would be a more appropriate system to normalize the data we are storing with the query mechanism.
* What are your expected tables / documents and relationships between tables / documents in your database? **TBA**

### Potential limitations:

* What are the potential limitations of your above proposed steps?
  + As the publishing time of different games is different, the aggregation results may be limited to a particular time.
* How can you control these potential issues?

* + We will not be considering the recent published game and specify the selection to a certain period.

# Steam Data Generation Project

## Executive Summary (ETL Process)

This is ETL project is focused on the gaming sales platform Steam. As per our title page, it really has become one of the major platforms for “the gaming world”. We have found the process of extracting, transforming and loading data to allow others to search and visualise relevant data (accessibility) has many crossovers, as it is not as simple as just extracting, transforming and loading as individual processes (as shown in the workflow process). Through each step, there has been some form of general transformation and additional extraction.

## Workflow Process – Initial ETL Overview

As we can see below, the overall simplicity of an ETL process, looks straight forward. But as we continued the process after the initial adding of dependencies, some sort of transformation was needed immediately to obtain the related/relevant data. This inherently has allowed the group to take two differing approaches to maximise the raw data and ensure better transformation process’.

### Dependencies

|  |  |
| --- | --- |
| **Part One -** | **Part Two -** |
| **Extraction** | **Extraction** |
| import csv  import datetime as dt  import json  import os  import statistics  import time  # *third-party imports*  Import numpy as np  import pandas as pd  import requests | import pandas as pd  import requests |
| **Transform & Load** | **Transform & Load** |
| import csv  import json  import pandas as pd  import pymongo  import matplotlib.pyplot as plt  import scipy.stats as st | import numpy as np  import pandas as pd  import requests  import time  from matplotlib import pyplot as plt  from pprint import pprint  from iso3166 import countries  import pymongo |

**Table 1:** Dependencies for both Part One and Part Two shows two very differing approaches the group has taken.

The two differing approaches were taken to ensure that the Rubic (initial) was meet. Primarily utilising several different tools/programs and dependencies would give the group a greater understanding and effective working knowledge of the ETL process. Given the diffing dependencies, we can make a relative distinction between the two process’ that would enable the extraction and transformation processes.

## Part One: - Genre’s and Feedback

### Extraction Process

<https://github.com/oscahui/ETL-project/blob/main/SteamGame-ETL-DataExtraction.ipynb>

# Customisations - ensure tables show all columns.

# Create the link for steamspy api.

# Create an empty list to store the first 10 pages of games.

# The games are stored in steamspy API by the order of popularity.

# Compile the data together to a new dataframe.

# Export the dataframe to Data folder.

# Create a function to continuously retrieve data from: -

<http://store.steampowered.com/api/appdetails/>.

#create a new list to iterate through each row of app\_list, confined by start and stop.

# retrieve app data for a row, handled by supplied parser, and append to list.

# Write the data to a json file due to the unstructured nature.

# examine the data file.

### Transformation Process

<https://github.com/oscahui/ETL-project/blob/main/SteamGame-ETL-TransAndLoad.ipynb>

# Customisations - ensure tables show all required columns: -

--- pd.set\_option("max\_columns", 100)

# Load the app\_list data extracted from steamspy API.

# Load the app\_data data extracted from steamspy API.

Data Merging

# Rename the app id column for merging.

# Merge the two dataframes based on the unique app id.

Data selection

# Select the game from the app list

Data filtration

# Drop the useless columns.

Data Cleaning

# Drop the duplicated games.

Data Splitting

# drop the index columns.

Data aggregating

# Aggregate the data of price/release\_data/recommendations together with previous dataframe.

# Drop the old unsplit columns.

# Rename the column for easier reading.

Data filtering and operating

# Filtering the null value.

# convert the price to readable value.

# reset the index.

### Loading Process

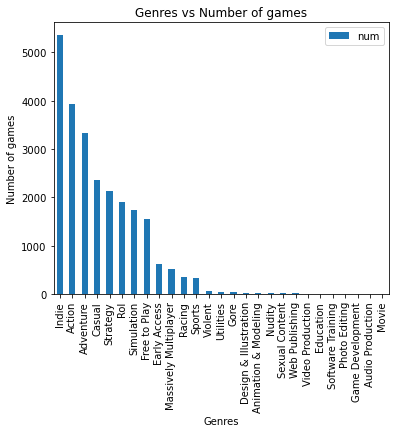
# Initialize PyMongo to work with MongoDBs.

# Define the 'steam\_game\_database' database in Mongo.

# Drop the game collection if exists.

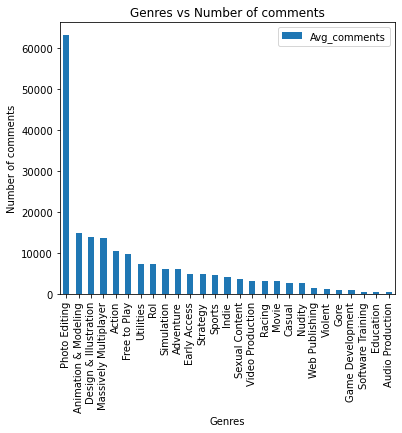
# Load the data frame to game collection in database.

### Summary ERD’s from the EDA (Part One)



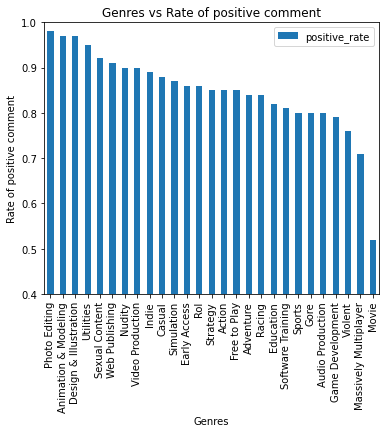
**Summary ERD of Genres Vs Number of games.**

The summary provides both gamers and developers a great tool to analyse. For a developer as an example – they may wish to make a new style of game in a popular (or unpopular) genre. For a gamer – they may wish to analyse which are the most popular and make decisions from there. It is a great example of adding value to both.



**Summary ERD of Genres Vs Number of comments.**

The content that garners the most comments, or the least and anywhere in between can be a useful tool, especially if they are positive. This not only will allow a developer to read the commentary for their analysis but allow them to sift through positive and negative ones of other developers, especially if it is the genre you are working with.



**Summary ERD of Genres Vs positive comments.**

As a developer, you can clearly see that the ERD shows the varying, yet surprising values attached to each genre. A particularly important added benefit for those targeting specific positive feedback for their business or for analysing competitors. In particular – if you joined a genre such as sexual content and nudity, it would show a clear trend for analysts.

## Part Two: - Steam Game Price Index

### Extraction Process

<https://github.com/oscahui/ETL-project/blob/main/SteamGame-ETL-DataExtraction.ipynb>

# Customisations - ensure tables show all columns.

# Create the link for steamspy api.

# Create an empty list to store the first 10 pages of games.

# The games are stored in steamspy API by the order of popularity.

# Compile the data together to a new dataframe.

# Export the dataframe to Data folder.

# Create a function to continuously retrieve data from: -

<http://store.steampowered.com/api/appdetails/>.

#create a new list to iterate through each row of app\_list, confined by start and stop.

# retrieve app data for a row, handled by supplied parser, and append to list.

# Write the data to a json file due to the unstructured nature.

# examine the data file.

### Transformation Process

<https://github.com/oscahui/ETL-project/blob/main/SteamGame-ETL-TransAndLoad.ipynb>

# Customisations - ensure tables show all required columns: -

--- pd.set\_option("max\_columns", 100)

# Load the app\_list data extracted from steamspy API.

# Load the app\_data data extracted from steamspy API.

Data Merging

# Rename the app id column for merging.

# Merge the two dataframes based on the unique app id.

Data selection

# Select the game from the app list

Data filtration

# Drop the useless columns.

Data Cleaning

# Drop the duplicated games.

Data Splitting

# drop the index columns.

Data aggregating

# Aggregate the data of price/release\_data/recommendations together with previous dataframe.

# Drop the old unsplit columns.

# Rename the column for easier reading.

Data filtering and operating

# Filtering the null value.

# convert the price to readable value.

# reset the index.

### Loading Process

# Initialize PyMongo to work with MongoDBs.

# Define the 'steam\_game\_database' database in Mongo.

# Drop the game collection if exists.

# Load the data frame to game collection in database.

### Summary ERD’s from the EDA (Part Two)