Creating a Profitable Video Game Title: An Analysis Using SQL

by Emerson Fleming

July 14th, 2024

# Introduction

**Problem:**

1. You are a rising video game developer hoping to create the most profitable title of the year. How will you solve this problem?
   1. You are given 2 datasets to build your findings upon. This is to simulate the reality that clean and optimal data will not always be available in a real setting.
      1. **game\_sales:** A dataset containing regional video games sales for each title included as well as the overall rating given by Metacritic.
         1. Columns:

img

title

console

genre

publisher

developer

critic\_score

total\_sales

na\_sales

jp\_sales

pal\_sales

other\_sales

release\_date

last\_update

* + 1. **game\_info:** A dataset containing information from 475,000 titles.
       1. Columns:

slug

name

metacritic

released

tba

updated

suggestions\_count

platforms

developers

genres

publishers

esrb\_rating

**Hypothesis:**

The most profitable video game to create relative to cost is an indie survival PC title.

**Null Hypothesis:**

The most profitable video game to create relative to cost is not an indie survival PC title.

Initial brainstorming questions:

1. Which video game genre is the most expensive to create?
   1. Which games have the highest profit to cost ratio?
2. Which genres have the highest sales?
3. Which platforms should the game be made for?
4. Does the name of the title impact sales?
   1. If so, in what way?
5. Which developer is the most popular?
   1. What genres does this developer typically create?
   2. What are their games generally rated?
   3. Which developer has the best reviews?
      1. What kinds of video games does this developer typically make?
6. Is it feasible to consider release date?
   1. When should the game be released to generate the most sales?
7. Can genre popularity trends be identified over time?
8. Which video games require the most time to create?

# Exploratory Analysis

1. The datasets are first opened in Microsoft Excel and unnecessary columns are deleted.
   1. “game\_info”:
      1. We will delete:
         1. “id”
         2. “updated”
         3. “esrb\_rating” (as this column is mostly empty)
   2. “game\_sales”:
      1. We will delete:
         1. “img”
         2. “last\_update”
2. The data is then imported to Microsoft SQL Server.
   1. Columns of interest are selected individually for analysis within the “game\_sales” table.
   2. The columns are then filtered to show titles created within 5 years.
      1. Overall, the results do not demonstrate a sufficient observation number for analysis given the size of both datasets. Therefore, the year scope must be expanded.
   3. In the results below, the data column appears incorrect for the “game\_sales” table.
      1. ‘Final Fantasy Type 0’ is a PSP game released in 2011 according to Google.
      2. Additionally, ‘Tokyo Jungle’ is a PS3 game released in 2012.
   4. Furthermore, the majority of sales columns contain insufficient data for analysis.

SELECT title, console, genre, publisher, developer, critic\_score, total\_sales, na\_sales, jp\_sales, pal\_sales, other\_sales, release\_date,

LEFT(release\_date, 4) AS year

FROM game\_sales

WHERE LEFT(release\_date, 4) > 2019

A screenshot of a computer

Description automatically generated

1. The date column is assessed from “game\_sales” by counting the number of rows and grouping by “year.”
   1. The results demonstrate few titles are included after year 2020. Therefore, recent genre and console trends cannot be properly included for analysis.

WITH CTE AS (

SELECT title, console, genre, publisher, developer, critic\_score, total\_sales, na\_sales, jp\_sales, pal\_sales, other\_sales, release\_date,

LEFT(release\_date, 4) AS year

FROM game\_sales

)

SELECT DISTINCT year,

COUNT(\*) AS count

FROM CTE

GROUP BY year

ORDER by year DESC

A table of numbers with numbers on it

Description automatically generated

1. The “game\_sales” table will be examined more granularly to consider the possibility of working around accurate release date information.
   1. In the query below, the “year” filter scope has been increased to encompass 7th generation console releases——which includes the Microsoft Xbox 360 and Sony PlayStation 3.
   2. Furthermore, target consoles released within 2005 to present day have been selected.
      1. However, the query below cannot be used to select consoles within the ideal release scope——as 'PC' releases will include titles older than 2005 due to the inaccurate release data. Therefore, the release data issue must later be addressed.

SELECT title, console, genre, publisher, developer, critic\_score, total\_sales, na\_sales, jp\_sales, pal\_sales, other\_sales,

LEFT(release\_date, 4) AS year

FROM game\_sales

WHERE LEFT(release\_date, 4) >= 2005

AND console IN ('Wii', 'WiiU', 'X360', 'XOne', 'XS', 'ZXS', 'PS3', 'PS4', 'PS5', 'iOS', 'PC')

A screenshot of a computer

Description automatically generated

1. The “game\_info” table is assessed for results.
   1. This table does not create new rows for individual releases by console.
      1. However, the “game\_sales” and “game\_info” tables will be joined together——which will solve this issue.
   2. The data range is increased to match the “game\_sales” table, where the most information is derived.
   3. Overall, the data column in the “game\_info” table appears more accurate.
      1. Upon cross-checking 10 title release dates using Google, the date column in this dataset appears more accurate.
      2. Therefore, a join is performed to retain the date column from the “game\_info” table.

SELECT "name", suggestions\_count, platforms, developers, genres, publishers,

LEFT(released, 4) AS year

FROM game\_info

WHERE LEFT(released, 4) >= 2005

AND "platforms" LIKE '%PlayStation 5%' OR "platforms" LIKE '%Xbox Series%'

A close up of a computer screen

Description automatically generated

# Data Cleaning

1. The “released” column from the “game\_info” table is changed to “date” data type.

ALTER TABLE game\_info ADD released\_converted date;

Update game\_info

SET released\_converted = CONVERT(date, released)

SELECT "name", suggestions\_count, platforms, developers, genres, publishers,

YEAR(released\_converted) AS year\_2

FROM game\_info

WHERE YEAR(released\_converted) >= 2005

1. Duplicates are found and dropped across both datasets.
   1. Duplicates are only dropped under logical circumstances. In this case, duplicate observations are dropped as repeat release observations for the same console are unnecessary.
   2. Therefore, duplicates within “game\_sales” are located.

WITH CTE AS(

SELECT \*,

ROW\_NUMBER() OVER (

PARTITION BY title,

console,

publisher,

developer

ORDER BY title) AS duplicate\_finder

FROM game\_sales)

SELECT \*

FROM CTE

WHERE duplicate\_finder > 1

* 1. Duplicates within “game\_sales” are then deleted.

WITH CTE AS(

SELECT \*,

ROW\_NUMBER() OVER (

PARTITION BY title,

console,

publisher,

developer

ORDER BY title) AS duplicate\_finder

FROM game\_sales)

DELETE

FROM CTE

WHERE duplicate\_finder > 1

* 1. Duplicates within the “game\_info” dataset are located.

WITH CTE AS (

SELECT \*,

ROW\_NUMBER() OVER(

PARTITION BY "name",

developers

ORDER BY id) AS duplicate\_finder

FROM game\_info)

SELECT \*

FROM CTE

WHERE duplicate\_finder > 1

* 1. Duplicates within the “game\_info” dataset are deleted.

WITH CTE AS (

SELECT \*,

ROW\_NUMBER() OVER(

PARTITION BY "name",

developers

ORDER BY id) AS duplicate\_finder

FROM game\_info)

DELETE

FROM CTE

WHERE duplicate\_finder > 1

1. An inner join is performed across the “game\_sales” and “game\_info” tables.
   1. A join is created using video game titles to create a join upon.
      1. Additionally, an inner join is used to drop titles without matches from both tables.
      2. A WHERE clause is used to filter for rows with at least 1 column with regional sales; observations with entirely absent regional sales columns cannot be used for analysis.
      3. The “platforms” column from the “game\_info” table appears more accurate than the “console” column from the “game\_sales” table.
         1. ‘Sudden Strike 4’ was released for macOS||iOS||Linux||PC||PlayStation 4
         2. ‘Cities:Skylines’ was released for Linux||macOS||PC||Nintendo Switch||Xbox One||PlayStation 4
            1. However, the ‘platforms’ column cannot be used from “game\_info” as the “game\_sales” table contains the most important sales data.

WITH CTE AS(

SELECT title, console, genre, publisher, developer, critic\_score, total\_sales, na\_sales, jp\_sales, pal\_sales, other\_sales,

YEAR(release\_date\_converted) AS year

FROM game\_sales

WHERE YEAR(release\_date\_converted) >= 2005),

CTE\_2 AS(SELECT "name", suggestions\_count, platforms, developers, genres, publishers,

YEAR(released\_converted) AS year\_accurate

FROM game\_info

WHERE YEAR(released\_converted) >= 2005)

SELECT CTE.title, genre, console, publisher, developer, critic\_score, total\_sales, na\_sales, jp\_sales, pal\_sales, other\_sales,

CTE\_2.platforms, year\_accurate

FROM CTE

INNER JOIN CTE\_2

ON CTE.title = CTE\_2.name

WHERE CTE.total\_sales IS NOT NULL

A screenshot of a computer screen

Description automatically generated

1. Missing data must now be addressed.
   1. Many missing values exist for regional sales data that require imputation.
   2. Statistically modeling datasets with missing values is not possible without addressing the issue.
      1. Overall, the data provided is difficult to statistically model as the independent variables——which include title, console, genre and publisher——are all nominal and categorical. Therefore, dummy variables would be necessary to assign binary values to nominal data. In this case, a myriad of nominal variables exist. Therefore, statistical modelling is not easily feasible.
   3. Within SQL, imputation is primarily performed using mean or median of the entire dataset.
      1. However, this will lead to more skewed results than using different imputation methods.
      2. Therefore, the data is imputed using R Studio.

WITH CTE AS(

SELECT title, console, genre, publisher, developer, critic\_score, total\_sales, na\_sales, jp\_sales, pal\_sales, other\_sales,

YEAR(release\_date\_converted) AS year

FROM game\_sales

WHERE YEAR(release\_date\_converted) >= 2005),

CTE\_2 AS(SELECT "name", suggestions\_count, platforms, developers, genres, publishers,

YEAR(released\_converted) AS year\_accurate

FROM game\_info

WHERE YEAR(released\_converted) >= 2005)

SELECT CTE.title, genre, console, publisher, developer, critic\_score, total\_sales, na\_sales, jp\_sales, pal\_sales, other\_sales,

CTE\_2.platforms, year\_accurate

FROM CTE

INNER JOIN CTE\_2

ON CTE.title = CTE\_2.name

WHERE CTE.total\_sales IS NOT NULL

1. Initially, the data is exported to Microsoft Excel.
   1. NULL values are replaced with empty values in Microsoft Excel. Otherwise, they will not be detected as empty within R Studio.

A screenshot of a computer

Description automatically generated

1. Within R Studio:
   1. The necessary packages are run.

knitr::opts\_chunk$set(echo = TRUE)

library(VIM)

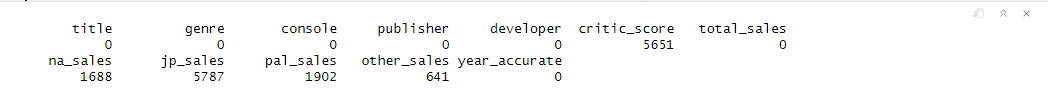
library(mice)

library(dplyr)

library(datarium)

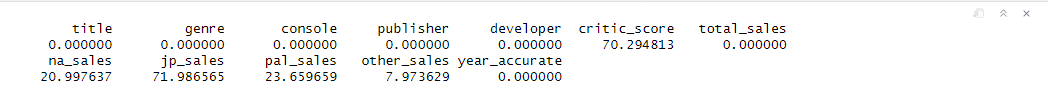
game\_data <- read.csv("C:/Users/emers/Desktop/Video Game Analysis/Video\_Game\_Analysis\_Cleaned.csv")

* 1. NA values are located and counted.

sapply(game\_data, function(x) sum(is.na(x)))

* 1. Percentages of NA values for each column are assessed.

sapply(game\_data, function(x) mean(is.na(x)) \* 100)



* 1. A visualization to understand the proportion of missing data is created.
     1. Overall, the “jp\_sales” and “critic\_score” columns are both primarily missing.
     2. Imputing data should not affect the mean and data distribution; it is always better to add more data if possible.
        1. The “critic\_score” column has a high percentage of missing values. Therefore, it will be dropped.
     3. Generally, imputation performs most effectively if missing values are <= 5% in a column.
     4. Additionally, imputation should not be used if the data is more than 20% missing.
        1. A large percentage of regional sales data is missing. Additionally, 70% of the “jp\_sales” column is absent. Therefore, an exception will be made due to the lack of usable data. Regional sales will then be recombined to create a new total sales column.

A blue and red squares with black text

Description automatically generated

* 1. Missing data is imputed using the kNN() function.
     1. The K Nearest-Neighbor method is utilized, as this method is the standard for data imputation due to its accuracy and efficacy.

game\_data\_knn <- kNN(game\_data, variable = c("jp\_sales", "critic\_score", "pal\_sales", "na\_sales", "other\_sales"), k = 6)

* 1. The imputation results are evaluated using an aggregation plot to ensure success.

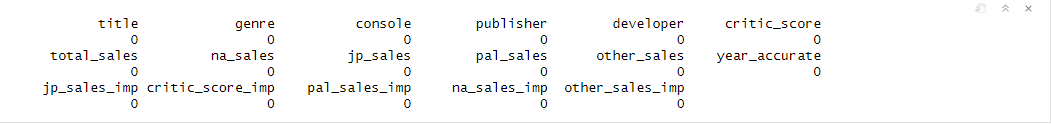
aggr(game\_data\_knn)

A blue and black chart

Description automatically generated with medium confidence

* 1. NA values are reassessed using the imputed data to ensure absence of missing values.

sapply(game\_data\_knn, function(x) sum(is.na(x)))



* 1. A boxplot is created to visualize possible distribution differences between the original and imputed data.
     1. Imputed data values should maintain the same distribution as the raw data.

par(mfrow=c(1,2))

boxplot(game\_data\_knn$jp\_sales, main = "Japan Sales: Imputed Values")

A comparison of a diagram

Description automatically generated with medium confidenceboxplot(game\_data$jp\_sales, main = "Japan Sales: Raw Data")

* 1. A t-test is used to compare sample means between the original and imputed data.
     1. The results demonstrate a low p-value indicative of no association between the “jp\_sales” variables across both datasets.

A screen shot of a computer code

Description automatically generatedt.test(game\_data\_knn$jp\_sales, game\_data$jp\_sales)

* 1. A new column is created containing the combination of values for each regional sales column. This column represents the new “total sales” column.

game\_data\_final <- mutate(game\_data\_comp, total\_sales\_cleaned = na\_sales + jp\_sales + pal\_sales + other\_sales

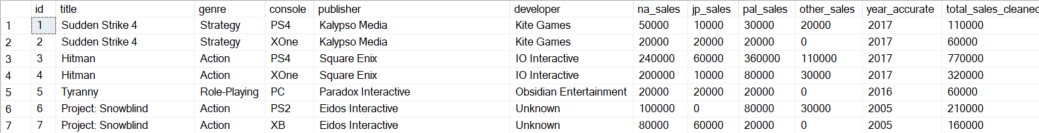
A screenshot of a computer screen

Description automatically generated

1. Before the data is reuploaded to Microsoft SQL Server, the sales columns are converted to actual sales figures.
   1. The sales-related columns are converted to millions sold for intuitive reporting.
   2. Additionally, the “critic\_score” and original “total\_sales” columns are dropped.
2. The imputed data is reuploaded to Microsoft SQL Server for analysis.

SELECT \*

FROM game\_data\_imputed



# Critical Analysis

1. The most ideal genre to create is assessed using the cleaned and imputed dataset.
   1. Total sales by genre are assessed across the dataset.
      1. Overall, action, sports and shooter games are the most popular.

WITH CTE AS (

SELECT genre,

SUM(total\_sales\_cleaned) AS total\_sales\_by\_genre,

FORMAT(SUM(total\_sales\_cleaned), 'C') AS total\_sales\_by\_genre\_$

FROM game\_data\_imputed

GROUP BY genre)

SELECT genre,

total\_sales\_by\_genre\_$

FROM CTE

ORDER BY total\_sales\_by\_genre DESC



1. Highest-selling individual releases are then assessed.
   1. Action and shooter games are among the most popular individual releases.
      1. Interestingly, sports games appear less popular as individual releases.
         1. This must be considered, as the goal is to create one single title.
   2. Additionally, the Sony PlayStation 3 and Microsoft Xbox 360 console releases are the most popular.
      1. This indicates 7th generation consoles——which include the PlayStation 3 and Xbox 360——may be more popular than their 8th generation counterparts——which include the PlayStation 4 and Xbox One.

WITH CTE AS (

SELECT title, genre, console, publisher, developer, year\_accurate, total\_sales\_cleaned,

FORMAT(total\_sales\_cleaned, 'C') AS total\_sales\_$

FROM game\_data\_imputed

)

SELECT title, genre, console, publisher, developer, year\_accurate, total\_sales\_$

FROM CTE

A screenshot of a computer

Description automatically generated ORDER BY total\_sales\_cleaned DES

1. A query is created to assess genre count across releases for developers with the highest percentage of sales.
   1. Within the query, duplicates are addressed——which currently exist as titles separated into individual rows based on console.

SELECT DISTINCT title,

developer,

genre

FROM game\_data\_imputed

WHERE developer = 'Rockstar North'

A screenshot of a computer

Description automatically generated

* 1. Therefore, a CTE is created to contain only first instances of game titles using the query below.

WITH CTE AS(

SELECT \*,

ROW\_NUMBER() OVER (

PARTITION BY title

ORDER BY title) AS duplicate\_finder

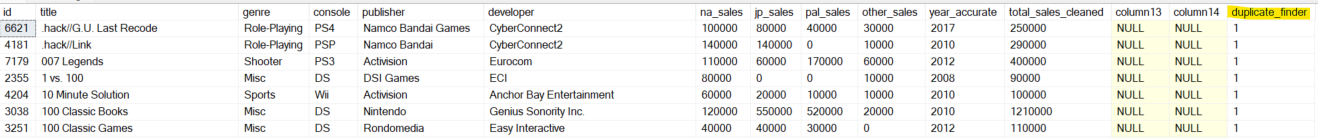
FROM game\_data\_imputed

)

SELECT \*

FROM CTE

WHERE duplicate\_finder = 1



* 1. A query is then created to count the number of genres by developer and join the tables together using CTEs.
     1. According to the query below, ‘EA%’ has the highest number of total sales by developer——and most frequently produces sports titles.
        1. The table below indicates developers with the highest percentage of sales most frequently create sports or action games.

WITH CTE AS (

SELECT \*,

ROW\_NUMBER() OVER (

PARTITION BY title

ORDER BY title) AS duplicate\_finder

FROM game\_data\_imputed

),

CTE\_2 AS (

SELECT \*

FROM CTE

WHERE duplicate\_finder = 1

),

CTE\_3 AS (

SELECT developer,

genre,

COUNT(genre) AS genre\_count

FROM CTE\_2

GROUP BY developer, genre

),

CTE\_4 AS (

SELECT developer,

CAST(100 \* SUM(total\_sales\_cleaned)/(SELECT SUM(total\_sales\_cleaned) FROM game\_data\_imputed) AS DECIMAL(7,2)) AS percentage\_of\_total\_sales\_by\_developer

FROM game\_data\_imputed

GROUP BY developer

)

SELECT

CTE\_3.developer,

CTE\_3.genre,

CTE\_4.percentage\_of\_total\_sales\_by\_developer,

CTE\_3.genre\_count

FROM CTE\_4

INNER JOIN CTE\_3

ON CTE\_3.developer = CTE\_4.developer

WHERE genre\_count >= 10

A screenshot of a computer

Description automatically generated ORDER BY CTE\_4.percentage\_of\_total\_sales\_by\_developer DESC, genre\_count DESC

1. To assess consoles to be included for the release, the percentage of total sales by console are evaluated.
   1. Overall, the Microsoft Xbox 360 and Sony PlayStation 3 are the most popular.
      1. Console releases generate a far greater percentage of sales than PC titles.
      2. Additionally, ninth generation consoles——which include the Microsoft Xbox Series X/S and Sony PlayStation 5 consoles——were only released in 2020. As discussed, the latest usable data from the “game\_sales” table is from 2020. Therefore, ninth generation console sales are not possible to effectively analyze.
      3. The Sony PlayStation 4 and Microsoft Xbox One sold far less titles than their predecessors.
         1. Therefore, developing a PC-only title may generate the highest return on investment given the generational decrease in console sales.

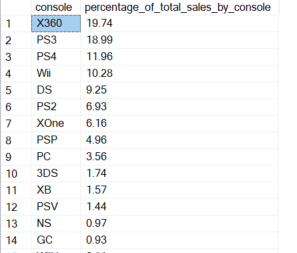
SELECT console,

CAST(100 \* SUM(total\_sales\_cleaned)/(SELECT SUM(total\_sales\_cleaned) FROM game\_data\_imputed) AS DECIMAL(7,2)) AS percentage\_of\_total\_sales\_by\_console

FROM game\_data\_imputed

GROUP BY console

ORDER BY CAST(100 \* SUM(total\_sales\_cleaned)/(SELECT SUM(total\_sales\_cleaned) FROM game\_data\_imputed) AS DECIMAL(7,2)) DESC



1. Next, the percentage of total sales by console and developer are assessed.
   1. ‘Infinity Ward’ and ‘Treyarch’ are well-known developers of shooter games; ‘EA%’ generally produces sports titles.
      1. Therefore, the query further demonstrates the popularity of these titles.
      2. The query also indicates the popularity of sports and shooter games for 7th generation consoles.

SELECT developer,

console,

CAST(100 \* SUM(total\_sales\_cleaned)/(SELECT SUM(total\_sales\_cleaned) FROM game\_data\_imputed) AS DECIMAL(7,2)) AS percentage\_of\_total\_sales\_by\_developer

FROM game\_data\_imputed

GROUP BY console, developer

ORDER BY CAST(100 \* SUM(total\_sales\_cleaned)/(SELECT SUM(total\_sales\_cleaned) FROM game\_data\_imputed) AS DECIMAL(7,2)) DESC

A screenshot of a computer

Description automatically generated

1. A query is created to identify trends by visualizing most sold releases for each year included in the dataset.
   1. Overall, the query demonstrates the popularity of shooter games has decreased over time.
   2. Additionally, sports and action/action-adventure titles are likely to sell the most copies.

WITH CTE\_1 AS (

SELECT developer AS developer,

genre,

title,

year\_accurate,

CAST(100 \* SUM(total\_sales\_cleaned)/(SELECT SUM(total\_sales\_cleaned) FROM game\_data\_imputed) AS DECIMAL(7,2)) AS percentage\_of\_total\_sales

FROM game\_data\_imputed

GROUP BY year\_accurate, developer, genre, title),

CTE\_2 AS (SELECT year\_accurate,

genre,

title,

developer,

percentage\_of\_total\_sales,

RANK() OVER (PARTITION BY year\_accurate ORDER BY percentage\_of\_total\_sales DESC) AS year\_rank

FROM CTE\_1)

SELECT year\_accurate,

genre,

title,

developer

FROM CTE\_2

WHERE year\_rank = 1

ORDER BY year\_accurate DESC

A screenshot of a video game

Description automatically generated

1. A query is created to visualize the top 3 highest selling genres by year.
   1. The data demonstrates sports and action games are among the highest consistently performing genres across the years.
   2. 2020 represents the latest year available. Therefore, sports games are likely the most consistently high performing titles.
      1. Other genres such as ‘Role-Playing’ and ‘Shooter’ titles are likely to trend.

WITH CTE\_1 AS (

SELECT genre,

year\_accurate,

CAST(100 \* SUM(total\_sales\_cleaned)/(SELECT SUM(total\_sales\_cleaned) FROM game\_data\_imputed) AS DECIMAL(7,2)) AS percentage\_of\_total\_sales

FROM game\_data\_imputed

GROUP BY year\_accurate, genre

),

CTE\_2 AS (SELECT year\_accurate,

genre,

percentage\_of\_total\_sales,

RANK() OVER (PARTITION BY year\_accurate ORDER BY percentage\_of\_total\_sales DESC) AS year\_rank

FROM CTE\_1

)

SELECT year\_accurate,

genre,

year\_rank

FROM CTE\_2

WHERE percentage\_of\_total\_sales != 0.00

AND year\_rank IN (1, 2, 3)

ORDER BY year\_accurate DESC

A screenshot of a table

Description automatically generated

# Conclusion

Overall, only genre and console analysis could be properly conducted. The analysis demonstrates console releases generate the highest number of sales. Additionally, sports and action titles are among the most predictably popular; the most popular developers typically create these genres. Furthermore, shooter and role-playing-game titles trend more and are less predictable when estimating potential sales.

The decrease in console sales from 7th to 8th generation must also be considered; a far larger percentage of sales (according to the data) consist of 7th generation console sales.

Based on findings of this analysis only, a sports video game for PC and current-generation consoles represents the safest title to create. However, further research must be conducted on the finance associated with video game creation.

Ultimately, the hypothesis that indie survival PC games are the most popular relative to cost cannot be proven or disproven. Therefore, further research must be conducted to answer the questions in blue.