**Statistical Analysis of Steam Games**

A logo of a company

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 23F-0574 | Ali Meekal | BSCS-4B |
| 2 | 23F-0559 | Syed Muhammad | BSCS-4B |
| 3 | 23F-0602 | Muhammad Emad | BSCS-4B |
| 4 | 23F-0627 | Muteeba Shahzad | BSCS-4B |
| 5 | 23F-0685 | Qasim Alvi | BSCS-4B |

## 1. Problem Statement

This project seeks to understand how various factors—specifically a game's launch price and the number of user reviews—affect its estimated revenue on the Steam platform. By analysing these variables, the study aims to uncover meaningful patterns or relationships that can inform predictions about a game's financial performance. Such insights can be valuable to developers, publishers, and marketers in making data-driven decisions regarding pricing strategies and promotional efforts.

## 2. Objective

## The objective of this project is to analyze the impact of launch price and user review count on the estimated revenue of video games on Steam. This analysis will involve graphical and tabular data representations, descriptive statistics, probability distributions, and regression modeling. The goal is to identify trends and develop predictive models to better estimate a game’s potential revenue.

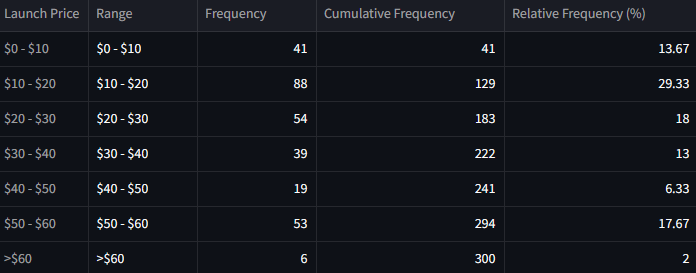
## 3. Data Description

[www.reddit.com/r/gamedev/comments/165cii0](http://www.reddit.com/r/gamedev/comments/165cii0)

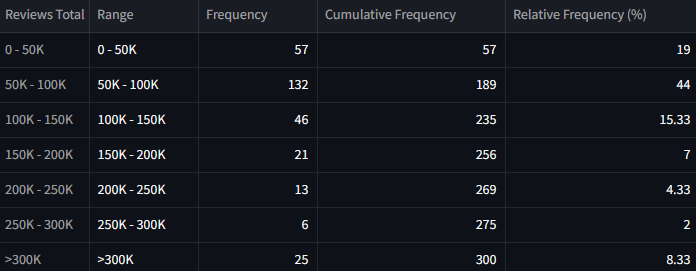
This dataset originally contains information on approximately 65,000 video games released on the Steam platform up to July 28, 2023. It includes five key variables: game title, release date, total reviews, launch price, and estimated revenue (calculated using the Boxleiter method). For this analysis, we focused on a subset of around 300 games, as the remaining ~63,000 titles exhibited extremely low revenue and review counts, contributing minimally to the overall trends and potentially introducing noise. Furthermore, user review data was cleaned to eliminate review bombs and entries from users who received the game for free. While the filtered subset provides a more meaningful basis for analysis, we acknowledge the presence of outliers even within the selected 300 games and will address them accordingly during the analysis.

## 4. Results

**Frequency Distribution Table**



Most indie games are priced between $10 and $20 to attract more buyers with affordability. AAA games, priced at $50–$60, follow as the third-largest group, reflecting a premium pricing strategy.



Most games have 50k–100k reviews, showing moderate player engagement. A few exceptional titles, highly loved by the community, exceed 300k reviews, with peaks reaching up to 7 million.

A screenshot of a computer

AI-generated content may be incorrect.  
31.33% of the games earned only $0–1.5 million, while a notable 9.33% made over $9 million, with the highest reaching $110 million. This shows that despite a saturated market, successful games can still generate massive revenue.

A graph of a business

AI-generated content may be incorrect.**Bar Chart**

Visual representation of the frequency distribution table.

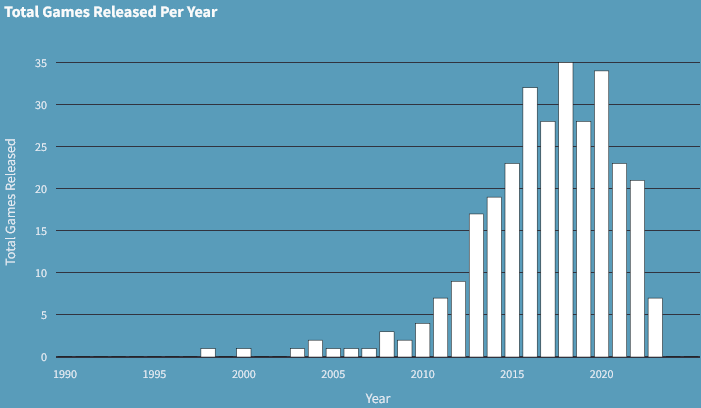
First three bar charts can be interpreted the same way as their tables.

A graph of a bar

AI-generated content may be incorrect.

A graph of a price increase

AI-generated content may be incorrect.

The bar chart shows an increase in the number of games released on Steam between 2008 and 2023, compared to earlier years. This trend reflects Steam’s growing popularity as a distribution platform since its launch in 2003, with relatively few games published in its early years.

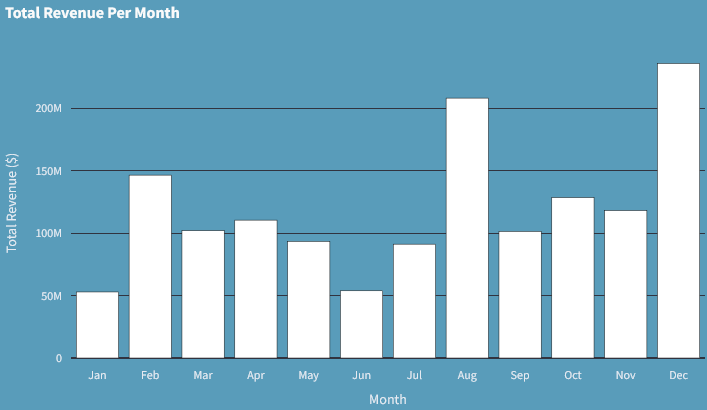
The total revenue chart shows relatively consistent revenue between 2015 and 2020, with 2015 marking the peak, driven by major releases like GTA V on PC. 2020 saw an increase due to the lockdownA graph of a number of years

AI-generated content may be incorrect..

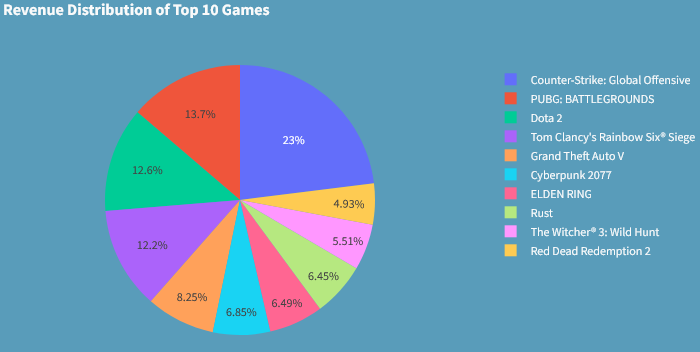
A graph of a graph

AI-generated content may be incorrect.

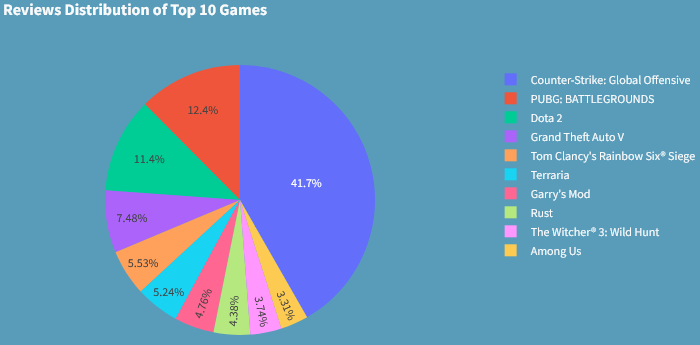
The monthly release chart shows fluctuating releases, with a peak in November just before the holiday season, as developers aim to boost sales during the holidays.



As a result, we see a peak in revenue generated in December, coinciding with the holiday season and increased game sales.

**Pie Chart**

The pie chart shows that CS:GO, even among the top 10 games, accounts for 23% of the total revenue, highlighting its immense popularity. The remaining games show a gradual decrease in revenue, with RDR2 contributing the least at 4.93%.

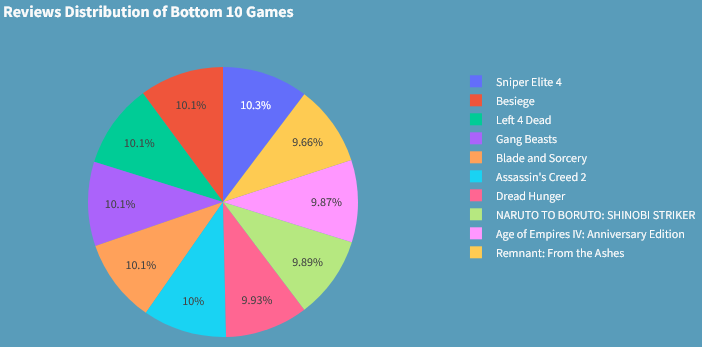


The reviews pie chart shows CS:GO holding a dominant 41.7% share, reflecting high player engagement, while Among Us has the lowest at 3.31%, with a gradual decline among the rest.

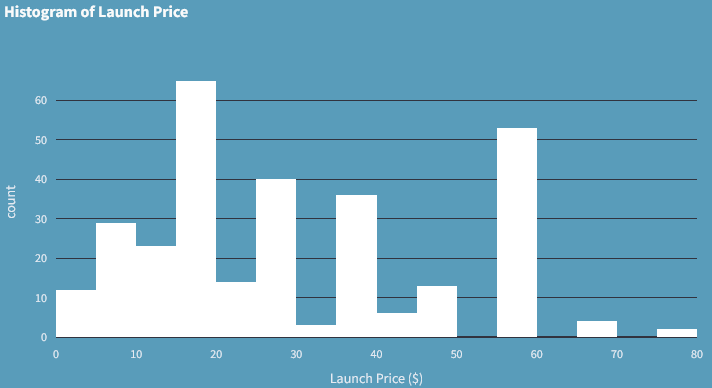
A colorful pie chart with numbers and a blue background

AI-generated content may be incorrect.

Among the bottom 10 games, the distribution is more balanced, with 2 games contributing over 15% and only 1 game generating less than 6%, indicating a narrower revenue gap within this group.

The review distribution among the bottom 10 games is almost even, with each title making up roughly 10% of the total, suggesting similar levels of player interaction.

***Note****: The top 10 and bottom 10 games are selected separately based on revenue and reviews, so the titles may vary between the two categories.*

**Histogram**

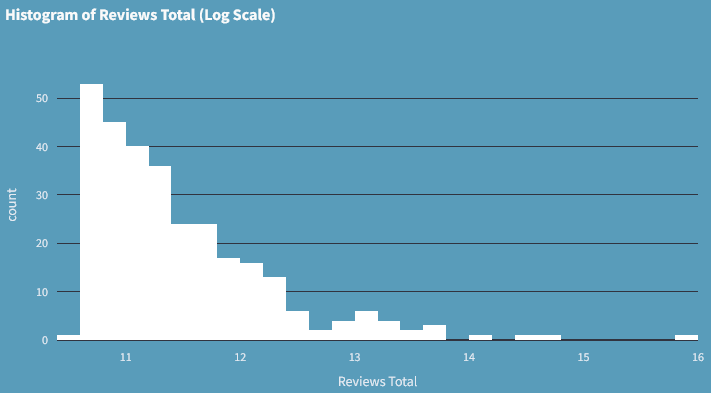
The histogram shows most games are priced between $0 and $50, with a smaller peak at $60, representing a common price point. Few games exceed $60, resulting in a somewhat right-skewed distribution.

A graph of a graph

AI-generated content may be incorrect.

After log transforming the revenue data, the histogram shows an approximately normal distribution. Most games generate decent revenue, with a few making significantly less or more. The log transformation was needed to reduce the skew caused by extreme outliers, providing a clearer representation.

After log transforming the review data, right skewness remains, showing most games have few reviews, while a few games have many, indicating that only a small number of games have active communities leaving reviews.

**Descriptive Statistics**

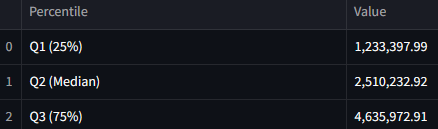
**1. Revenue Estimated**

A black rectangular object with a black background

AI-generated content may be incorrect. A black rectangular object with white lines

AI-generated content may be incorrect.

A blue background with white text

AI-generated content may be incorrect.  A blue background with white numbers

AI-generated content may be incorrect.

A blue background with white text

AI-generated content may be incorrect. A blue background with white text

AI-generated content may be incorrect.

For descriptive statistics of revenue, the median ± IQR is preferred because it minimizes the impact of outliers and provides a better estimate for skewed data (non log-transformed). Negative values, while mathematically necessary due to the formula, will be ignored since they don't apply to revenue.

**2. Reviews Total**

A black rectangular object with a black background

AI-generated content may be incorrect.

A black rectangular object with white lines

AI-generated content may be incorrect.A blue background with white text

AI-generated content may be incorrect.A black rectangular object with white lines

AI-generated content may be incorrect.A blue background with white text

AI-generated content may be incorrect.

A blue background with white text

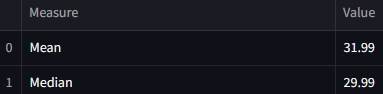
AI-generated content may be incorrect.

A blue background with white text

AI-generated content may be incorrect.

For descriptive statistics of total reviews, the median ± IQR is preferred as it better handles outliers and provides a more accurate estimate for skewed data. Negative values will be ignored since they don’t apply to review counts.

**3. Launch Price**



A screenshot of a video game

AI-generated content may be incorrect.

A blue background with white text

AI-generated content may be incorrect.A black rectangular object with white lines

AI-generated content may be incorrect.

A blue background with white text

AI-generated content may be incorrect.

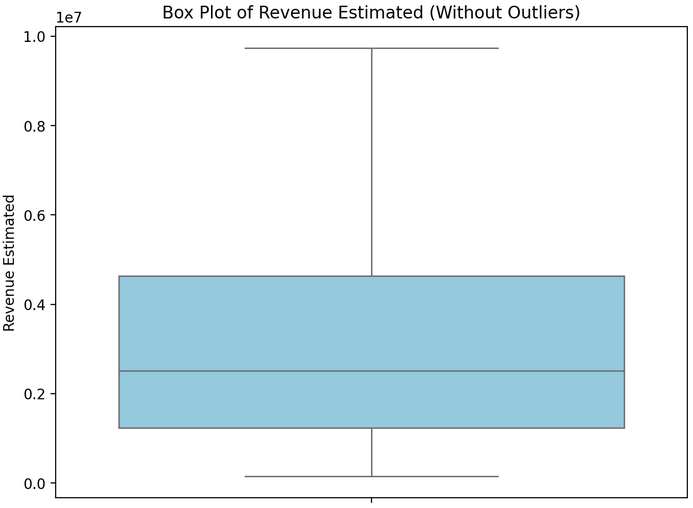
A blue background with white text

AI-generated content may be incorrect.A blue background with white text

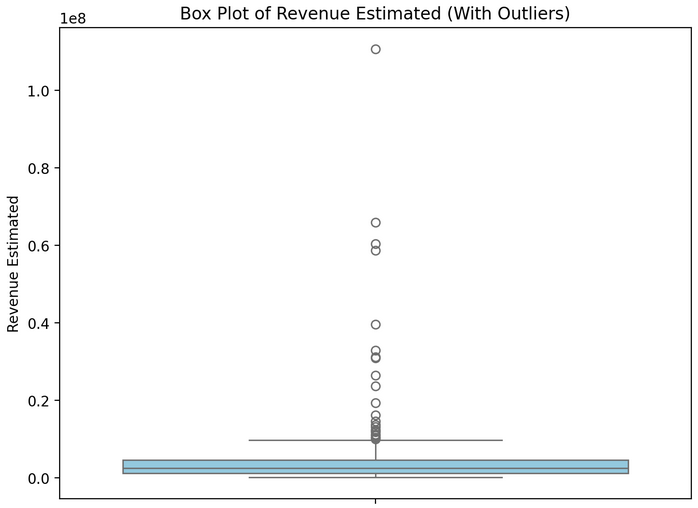
AI-generated content may be incorrect.

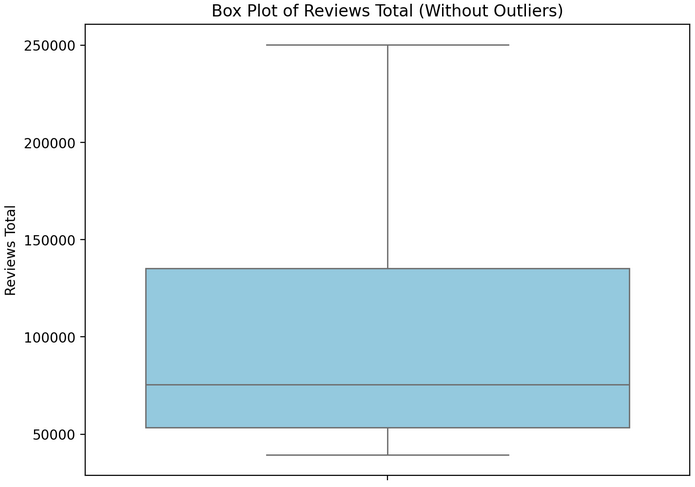
For launch price data under $80 with no true outliers, mean ± standard deviation is valid. However, due to skew from many low-priced games, median ± IQR remains more reliable for representing typical prices.

**Box Plot**



In the box plot with outliers, the data appears condensed near the bottom due to a few extreme high values stretching the scale. Without outliers, the plot more clearly shows the range between the lower and upper quartiles, where most of the revenue lies, providing a better view of the typical distribution.





A diagram of a box

AI-generated content may be incorrect.

In the box plot for total reviews, the condensation near the bottom is even more pronounced due to a few games with extremely high review counts.

A diagram with a blue rectangle

AI-generated content may be incorrect.

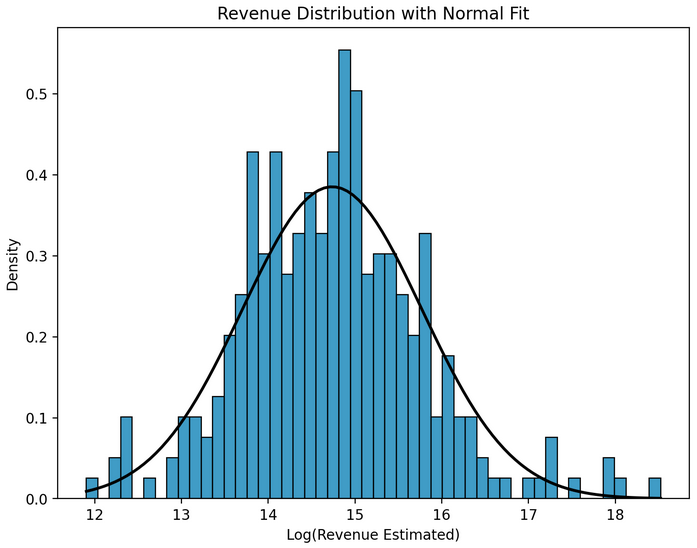
For launch price, the box plot remains identical with or without outliers, as no games exceed $80. This indicates a naturally limited range with no extreme values, making the distribution clearly visible without distortion.

**Probability Distributions**

In the case of our data, a binomial distribution did not apply because there were no clear success/failure outcomes. Poisson distribution was unsuitable since the average rate was not constant over a fixed interval. Hypergeometric distribution also didn’t fit, as it similarly relies on success/failure outcomes in a fixed population. The data was also not uniform, as values were unevenly distributed. The only suitable model was the normal distribution, which emerged after log-transforming the revenue data, making it symmetric and bell-shaped.

**Revenue Estimated**

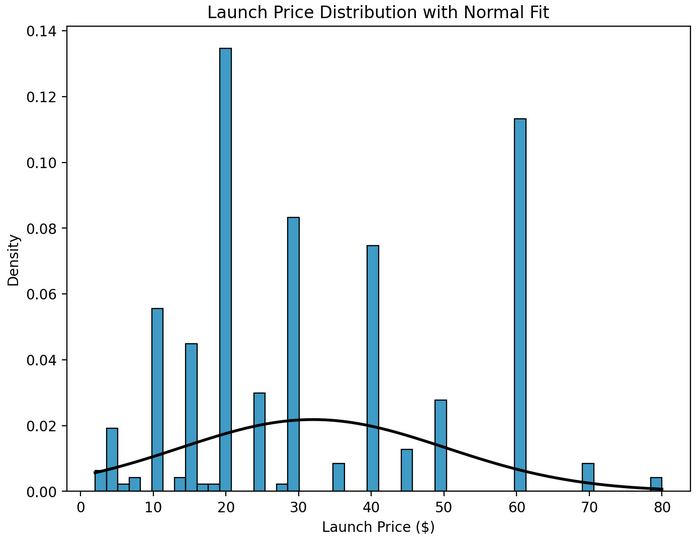
A screenshot of a blue screen

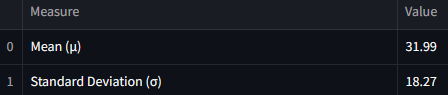
AI-generated content may be incorrect.  
A black rectangular object with white lines

AI-generated content may be incorrect.Based on the log-normal distribution of revenue, there's an estimated 18.68% chance that a game earns less than $1 million, while around 81.32% of games earn more than $1 million. Additionally, approximately 35.65% of games fall within the $2.5 million to $7.5 million revenue range.

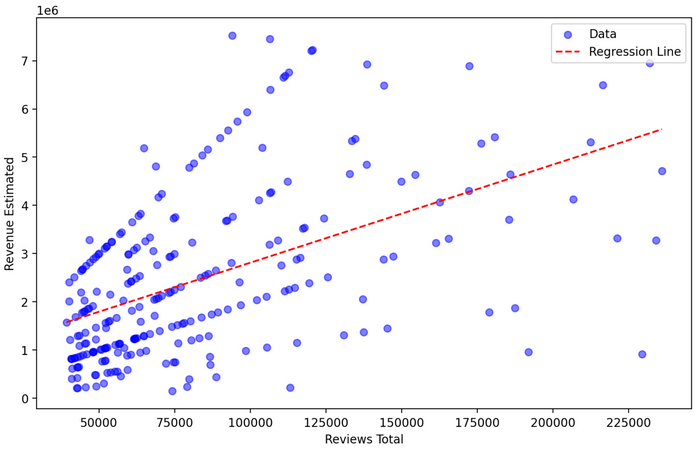
**Launch Price**

In the case of launch price, the data shows multiple sharp peaks, which suggests that the prices cluster around certain values rather than following a smooth, bell-shaped curve. A normal distribution assumes a continuous, symmetrical spread of data with a single peak. However, the presence of these spikes indicates that the data is likely influenced by specific price points (e.g., $60 being a common retail price), assuming of normality invalid. Therefore, a normal distribution isn't a suitable model for this data.





**Linear Regression**



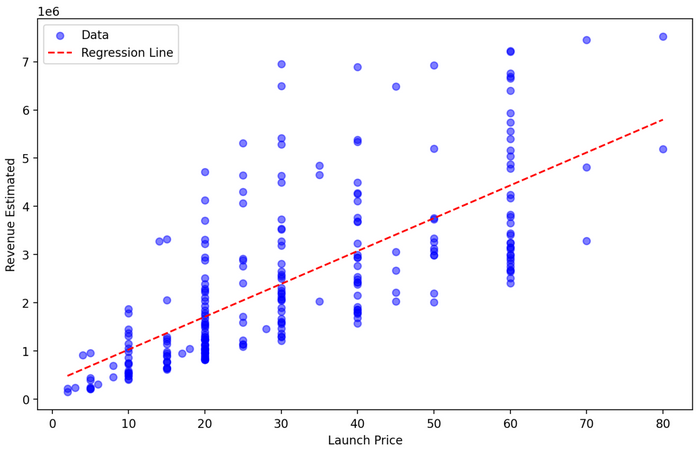
In the case of linear regression between revenue and total reviews, we observe a clear linear relationship after applying IQR filtering to remove outliers. Without this filtering, the graph would show a cluster of data points near the origin.

A blue background with white numbers

AI-generated content may be incorrect.The covariance is very large, making it an inaccurate measure for assessing the relationship between the two variables. To get a more meaningful understanding, we computed the correlation, which is 0.5171, indicating a moderate positive relationship between revenue and total reviews.

In the case of launch price, after applying IQR filtering, we observe a clear linear trend.

A blue background with white text

AI-generated content may be incorrect.The correlation of 0.7095 indicates a strong positive linear relationship between the variables. Meaning as launch price increases, revenue is estimated to increase at a strong rate.

**Correlation Matrix**

A red and blue squares with white text

AI-generated content may be incorrect.

**Predictions**

A blue background with white text

AI-generated content may be incorrect.

A blue screen with white text

AI-generated content may be incorrect.

By plugging the data into the linear regression formula, we have inferred these results.

**5. Codes**

**Libraries**

import streamlit as st  # For creating web apps with interactive widgets and layout

import pandas as pd     # For data manipulation and handling tabular data (DataFrames)

import numpy as np      # For numerical operations and working with arrays

import plotly.express as px  # For creating interactive plots and charts

import seaborn as sns            # For creating statistical visualizations

import matplotlib.pyplot as plt  # For plotting graphs and figures

import scipy.stats as stats      # For performing statistical tests and analysis

from sklearn.linear\_model import LinearRegression  # Imports the LinearRegression model class used to perform simple or multiple linear regression

**Home**

st.set\_page\_config(page\_title="Steam Games Analysis", layout="centered")    # Titles & Styling

st.markdown("""

    <style>

    .main, .css-18e3th9, .css-1d391kg, .stApp {

        background-color: #599cba;

        color: white;

    }

    h1, h3 { color: white; }

    .nav-item { font-size: 18px; margin-right: 20px; display: inline-block; }

    .nav-item a { color: white; text-decoration: none; }

    .nav-item a:hover { color: #ffa500; }

    </style>

""", unsafe\_allow\_html=True)

st.markdown("<h1 style='text-align: center;'>🎮 Steam Games Analysis</h1>", unsafe\_allow\_html=True)

st.markdown("<h3 style='text-align: center;'>📉 Graphical & Tabular Representations</h3>", unsafe\_allow\_html=True)

# Load data

df = pd.read\_csv("games.csv")

for col in ['Launch.Price', 'Reviews.Total', 'Revenue.Estimated']:

    df[col] = pd.to\_numeric(df[col], errors='coerce')

df['Release Year'] = pd.to\_datetime(df['Release.Date'], errors='coerce').dt.year

df = df[(df['Release Year'] >= 1990) & (df['Release Year'] <= 2025)]

df.dropna(subset=['Launch.Price', 'Reviews.Total', 'Revenue.Estimated'], inplace=True)

# Frequency Distribution

def display\_distribution(column, step, max\_val, prefix="$"):

    def fmt(val): return f"{prefix}{val / 1\_000\_000:.1f}M" if val >= 1e6 else f"{prefix}{val / 1\_000:.0f}K" if val >= 1e3 else f"{prefix}{val}"

    bins = list(range(0, max\_val + step, step)) + [np.inf]

    labels = [f"{fmt(bins[i])} - {fmt(bins[i+1])}" for i in range(len(bins)-2)] + [f">{fmt(max\_val)}"]

    col\_name\_pretty = column.replace(".", " ")

    df[col\_name\_pretty] = pd.cut(df[column], bins=bins, labels=labels, right=False)

    freq = df[col\_name\_pretty].value\_counts().sort\_index()

    table = pd.DataFrame({

        "Range": freq.index.astype(str),

        "Frequency": freq.values,

        "Cumulative Frequency": freq.cumsum().values,

        "Relative Frequency (%)": (freq / freq.sum() \* 100).round(2)

    })

    st.markdown("<h3 style='text-align: center;'>📊 Frequency Distribution Table</h3>", unsafe\_allow\_html=True)

    st.dataframe(table, use\_container\_width=True)

col\_options = {

    "Launch Price": ("Launch.Price", 10, 60, "$"),

    "Reviews Total": ("Reviews.Total", 50000, 300000, ""),

    "Revenue Estimated": ("Revenue.Estimated", 1500000, 9000000, "$")

}

choice = st.radio("Select Variable:", list(col\_options))

col, step, max\_val, prefix = col\_options[choice]

display\_distribution(col, step, max\_val, prefix)

# Bar Chart

st.markdown("<h3 style='text-align: center; color: white;'>📈 Bar Chart</h3>", unsafe\_allow\_html=True)

bar\_option = st.selectbox("Select Metric:", [

    "Revenue Estimated",

    "Reviews Total",

    "Total Games Released Per Year",

    "Total Revenue Per Year",

    "Total Games Released Per Month",

    "Total Revenue Per Month",

    "Launch Price"

], index=0)

time\_metrics = [

    "Total Games Released Per Year",

    "Total Revenue Per Year",

    "Total Games Released Per Month",

    "Total Revenue Per Month"

]

is\_line\_chart = st.toggle("Line Chart") if bar\_option in time\_metrics else False

chart\_func = px.line if is\_line\_chart else px.bar

def apply\_dark\_layout(fig, title, xaxis\_title, yaxis\_title):

    fig.update\_layout(

        title=title,

        template='plotly\_dark',

        plot\_bgcolor='#599cba',

        paper\_bgcolor='#599cba',

        font\_color='white',

        xaxis\_title=xaxis\_title,

        yaxis\_title=yaxis\_title

    )

    fig.update\_traces(marker=dict(color='white'))

    return fig

# 1. Revenue Estimated

if bar\_option == "Revenue Estimated":

    revenue\_bins = [0, 1.5e6, 3e6, 4.5e6, 6e6, 7.5e6, 9e6, 10.5e6, 12e6, 13.5e6, 15e6]

    revenue\_labels = [f"${int(b/1e6)}M - ${int(revenue\_bins[i+1]/1e6)}M" for i, b in enumerate(revenue\_bins[:-1])]

    df['Revenue.Bin'] = pd.cut(df['Revenue.Estimated'], bins=revenue\_bins, labels=revenue\_labels, right=False)

    data = df.groupby('Revenue.Bin').Title.count().reset\_index(name="Total\_Games")

    fig = px.bar(data, x='Revenue.Bin', y='Total\_Games', labels={'Revenue.Bin': 'Revenue Range', 'Total\_Games': 'Total Number of Games'})

    fig.update\_layout(xaxis\_tickangle=-90)

    st.plotly\_chart(apply\_dark\_layout(fig, "Number of Games vs Revenue Estimated", "Revenue Range ($)", "Total Number of Games"), use\_container\_width=True)

# 2. Reviews Total

elif bar\_option == "Reviews Total":

    review\_bins = list(range(0, 550000, 50000))

    review\_labels = [f"{i//1000}k - {j//1000}k" for i, j in zip(review\_bins[:-1], review\_bins[1:])]

    df['Reviews.Bin'] = pd.cut(df['Reviews.Total'], bins=review\_bins, labels=review\_labels, right=False)

    data = df.groupby('Reviews.Bin').Title.count().reset\_index(name="Total\_Games")

    fig = px.bar(data, x='Reviews.Bin', y='Total\_Games', labels={'Reviews.Bin': 'Reviews Range', 'Total\_Games': 'Total Number of Games'})

    st.plotly\_chart(apply\_dark\_layout(fig, "Number of Games vs Reviews Total", "Reviews Range", "Total Number of Games"), use\_container\_width=True)

# 3. Total Games Released Per Year

elif bar\_option == "Total Games Released Per Year":

    year\_counts = df['Release Year'].value\_counts().reindex(range(1990, 2026), fill\_value=0).sort\_index()

    fig = chart\_func(x=year\_counts.index, y=year\_counts.values, labels={'x': 'Year', 'y': 'Total Games Released'})

    st.plotly\_chart(apply\_dark\_layout(fig, "Total Games Released Per Year", "Year", "Total Games Released"), use\_container\_width=True)

# 4. Total Revenue Per Year

elif bar\_option == "Total Revenue Per Year":

    data = df.groupby('Release Year')['Revenue.Estimated'].sum().reindex(range(1990, 2026), fill\_value=0).reset\_index()

    fig = chart\_func(data, x='Release Year', y='Revenue.Estimated', labels={'Release Year': 'Year', 'Revenue.Estimated': 'Total Revenue'})

    st.plotly\_chart(apply\_dark\_layout(fig, "Total Revenue Per Year", "Year", "Total Revenue ($)"), use\_container\_width=True)

# 5. Total Games Released Per Month

elif bar\_option == "Total Games Released Per Month":

    df['Release Month'] = pd.to\_datetime(df['Release.Date'], errors='coerce').dt.month

    month\_counts = df['Release Month'].value\_counts().reindex(range(1, 13), fill\_value=0).sort\_index()

    fig = chart\_func(x=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],

                     y=month\_counts.values, labels={'x': 'Month', 'y': 'Total Games Released'})

    st.plotly\_chart(apply\_dark\_layout(fig, "Total Games Released Per Month", "Month", "Total Games Released"), use\_container\_width=True)

# 6. Total Revenue Per Month

elif bar\_option == "Total Revenue Per Month":

    df['Release Month'] = pd.to\_datetime(df['Release.Date'], errors='coerce').dt.month

    revenue\_per\_month = df.groupby('Release Month')['Revenue.Estimated'].sum().reindex(range(1, 13), fill\_value=0)

    fig = chart\_func(x=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],

                     y=revenue\_per\_month.values, labels={'x': 'Month', 'y': 'Total Revenue ($)'})

    st.plotly\_chart(apply\_dark\_layout(fig, "Total Revenue Per Month", "Month", "Total Revenue ($)"), use\_container\_width=True)

# 7. Launch Price

elif bar\_option == "Launch Price":

    price\_bins = [0, 10, 20, 30, 40, 50, 60, 70, 80]

    price\_labels = [f"${price\_bins[i]} - ${price\_bins[i+1]}" for i in range(len(price\_bins)-1)]

    df['Launch.Price.Bin'] = pd.cut(df['Launch.Price'], bins=price\_bins, labels=price\_labels, right=False)

    data = df.groupby('Launch.Price.Bin').Title.count().reset\_index(name="Total\_Games")

    fig = px.bar(data, x='Launch.Price.Bin', y='Total\_Games',

                 labels={'Launch.Price.Bin': 'Launch Price Range', 'Total\_Games': 'Total Number of Games'})

    st.plotly\_chart(apply\_dark\_layout(fig, "Number of Games vs Launch Price", "Launch Price Range ($)", "Total Number of Games"), use\_container\_width=True)

# Pie Chart

st.markdown("<h3 style='text-align: center; color: white;'>🍰 Pie Chart</h3>", unsafe\_allow\_html=True)

chart\_option = st.selectbox("Select Metric:", [

    "Revenue (Top 10 Games)", "Revenue (Bottom 10 Games)",

    "Reviews (Top 10 Games)", "Reviews (Bottom 10 Games)"

], index=0)

metric\_map = {

    "Revenue (Top 10 Games)": ('Revenue.Estimated', False, "Revenue Distribution of Top 10 Games"),

    "Revenue (Bottom 10 Games)": ('Revenue.Estimated', True, "Revenue Distribution of Bottom 10 Games"),

    "Reviews (Top 10 Games)": ('Reviews.Total', False, "Reviews Distribution of Top 10 Games"),

    "Reviews (Bottom 10 Games)": ('Reviews.Total', True, "Reviews Distribution of Bottom 10 Games")

}

col, asc, title = metric\_map[chart\_option]

pie\_data = df.sort\_values(by=col, ascending=asc).head(10)

fig\_pie = px.pie(pie\_data, names='Title', values=col, title=title, template='plotly\_dark')

fig\_pie.update\_layout(plot\_bgcolor='#599cba', paper\_bgcolor='#599cba', font\_color='white')

st.plotly\_chart(fig\_pie, use\_container\_width=True)

# Histogram

st.markdown("<h3 style='text-align: center; color: white;'>📊 Histogram</h3>", unsafe\_allow\_html=True)

hist\_option = st.selectbox("Select Metric:", [

    "Launch Price", "Revenue Estimated", "Reviews Total"

], index=0)

if hist\_option == "Launch Price":

    bin\_size = 5

    fig\_hist = px.histogram(df, x='Launch.Price',

                            nbins=int((df['Launch.Price'].max() - df['Launch.Price'].min()) / bin\_size),

                            title="Histogram of Launch Price",

                            labels={'Launch.Price': 'Launch Price ($)'},

                            template='plotly\_dark')

    fig\_hist.update\_traces(xbins=dict(start=0, end=80, size=bin\_size), marker=dict(color='white'))

elif hist\_option == "Revenue Estimated":

    col\_log = "Revenue.Estimated.Log"

    df[col\_log] = np.log(df['Revenue.Estimated'] + 1)

    fig\_hist = px.histogram(df, x=col\_log, nbins=50,

                            title="Histogram of Revenue Estimated (Log Scale)",

                            labels={col\_log: 'Log(Revenue Estimated)'},

                            template='plotly\_dark')

    fig\_hist.update\_traces(marker=dict(color='white'))

else:

    col\_log = "Reviews.Total.Log"

    df[col\_log] = np.log(df['Reviews.Total'] + 1)

    fig\_hist = px.histogram(df, x=col\_log, nbins=50,

                            title="Histogram of Reviews Total (Log Scale)",

                            labels={col\_log: 'Reviews Total'},

                            template='plotly\_dark')

    fig\_hist.update\_traces(marker=dict(color='white'))

fig\_hist.update\_layout(plot\_bgcolor='#599cba', paper\_bgcolor='#599cba', font\_color='white')

st.plotly\_chart(fig\_hist, use\_container\_width=True)

**Descriptive**

st.markdown("<h1 style='text-align: center;'>🧾 Descriptive Statistics</h1>", unsafe\_allow\_html=True)   # Titles & Styling

st.markdown("""

    <style>

    .main, .css-18e3th9, .css-1d391kg, .stApp {

        background-color: #599cba;

        color: white;

    }

    h1, h3 { color: white; }

    .nav-item { font-size: 18px; margin-right: 20px; display: inline-block; }

    .nav-item a { color: white; text-decoration: none; }

    .nav-item a:hover { color: #ffa500; }

    </style>

""", unsafe\_allow\_html=True)

# Load data & select variable

df = pd.read\_csv("games.csv")

df.columns = df.columns.str.strip()

column\_map = {

    "Revenue Estimated": "Revenue.Estimated",

    "Reviews Total": "Reviews.Total",

    "Launch Price": "Launch.Price"

}

display\_col = st.selectbox("Select Variable:", list(column\_map.keys()))

col = pd.to\_numeric(df[column\_map[display\_col]], errors='coerce').dropna()

# Central Tendency and Dispersion

mean, median, std, var = col.mean(), col.median(), col.std(), col.var()

rng, q1, q3 = col.max() - col.min(), col.quantile(0.25), col.quantile(0.75)

iqr = q3 - q1

# Confidence Interval (95%)

n = len(col)

se = std / (n \*\* 0.5)

z = stats.norm.ppf(0.975)

ci\_lower, ci\_upper = mean - z \* se, mean + z \* se

# 3-Sigma and Outlier Bounds

lower\_3sd, upper\_3sd = mean - 3\*std, mean + 3\*std

lower\_out, upper\_out = q1 - 1.5\*iqr, q3 + 1.5\*iqr

# Central Tendency Output

st.subheader("📌 Central Tendency")

st.dataframe(pd.DataFrame({"Measure": ["Mean", "Median"], "Value": [f"{mean:,.2f}", f"{median:,.2f}"]}), use\_container\_width=True)

# Dispersion Output

st.subheader("📊 Dispersion")

st.dataframe(pd.DataFrame({

    "Measure": ["Range", "Variance", "Standard Deviation"],

    "Value": [f"{rng:,.2f}", f"{var:,.2f}", f"{std:,.2f}"]

}), use\_container\_width=True)

# 3-Sigma Output

st.markdown(f"### 3-Sigma Interval:\n- \*\*{display\_col} lies between:\*\* {mean:,.2f} ± {3\*std:,.2f}")

# IQR & Quartiles Output

st.subheader("📊 IQR and Outliers")

st.dataframe(pd.DataFrame({

    "Percentile": ["Q1 (25%)", "Q2 (Median)", "Q3 (75%)"],

    "Value": [f"{q1:,.2f}", f"{median:,.2f}", f"{q3:,.2f}"]

}), use\_container\_width=True)

st.markdown(f"- \*\*IQR:\*\* {iqr:,.2f}  \n- \*\*Median ± IQR:\*\* {median:,.2f} ± {iqr:,.2f}")

# Outlier Bounds Output

st.markdown(f"### Outlier Bounds:\n- \*\*Lower:\*\* {lower\_out:,.2f}  \n- \*\*Upper:\*\* {upper\_out:,.2f}")

# Confidence Interval Output

st.subheader("📐 Confidence Interval (95%)")

st.markdown(

    f"""

    The 95% confidence interval for the mean of \*\*{display\_col}\*\* is:

    ### \*\*{ci\_lower:,.2f} to {ci\_upper:,.2f}\*\*

    This means we're 95% confident the true average of this variable lies within this range.

    """

)

# Box Plot

st.markdown("<h1 style='text-align: center;'>📦 Box Plot</h1>", unsafe\_allow\_html=True)   # Titles & Styling

show\_outliers = st.checkbox("Show Outliers", value=False)

fig, ax = plt.subplots(figsize=(8, 6))

sns.boxplot(data=col, ax=ax, color='skyblue', showfliers=show\_outliers)

ax.set\_title(f"Box Plot of {display\_col} ({'With' if show\_outliers else 'Without'} Outliers)")

ax.set\_ylabel(display\_col)

st.pyplot(fig)

**Distributions**

st.markdown("<h1 style='text-align: center;'>📊 Probability Distributions</h1>", unsafe\_allow\_html=True)    # Titles & Styling

st.markdown("""

    <style>

    .main, .css-18e3th9, .css-1d391kg, .stApp {

        background-color: #599cba;

        color: white;

    }

    h1, h3 { color: white; }

    .nav-item { font-size: 18px; margin-right: 20px; display: inline-block; }

    .nav-item a { color: white; text-decoration: none; }

    .nav-item a:hover { color: #ffa500; }

    </style>

""", unsafe\_allow\_html=True)

# Load data

df = pd.read\_csv("games.csv")

df.columns = df.columns.str.strip()

df["Revenue.Estimated"] = pd.to\_numeric(df["Revenue.Estimated"], errors="coerce")

df["Reviews.Total"] = pd.to\_numeric(df["Reviews.Total"], errors="coerce")

df["Launch.Price"] = pd.to\_numeric(df["Launch.Price"], errors="coerce")

df.dropna(subset=["Revenue.Estimated", "Reviews.Total", "Launch.Price"], inplace=True)

df["Transformed\_Revenue"] = np.log1p(df["Revenue.Estimated"])   # Revenue Estimated (log transformation)

# Reusable plot + stats display

def plot\_and\_display\_distribution(data, title, label):

    mean, std = norm.fit(data)

    x = np.linspace(data.min(), data.max(), 100)

    fig, ax = plt.subplots(figsize=(8, 6))

    sns.histplot(data, bins=50, stat="density", color="#007bb3", ax=ax)

    ax.plot(x, norm.pdf(x, mean, std), 'k', lw=2)

    ax.set(title=title, xlabel=label, ylabel="Density")

    st.pyplot(fig)

    stats\_df = pd.DataFrame({"Measure": ["Mean (μ)", "Standard Deviation (σ)"],

                             "Value": [f"{mean:.2f}", f"{std:.2f}"]})

    st.dataframe(stats\_df, use\_container\_width=True)

    return mean, std

# Revenue distribution (log-transformed)

st.subheader("📈 Log-Transformed Revenue Estimated Distribution")

mean\_rev, std\_rev = plot\_and\_display\_distribution(df["Transformed\_Revenue"],

                                                  "Revenue Distribution with Normal Fit",

                                                  "Log(Revenue Estimated)")

# Revenue probability estimates

p1 = norm.cdf(np.log1p(1\_000\_000), loc=mean\_rev, scale=std\_rev)

p2 = 1 - p1

p3 = norm.cdf(np.log1p(7\_500\_000), loc=mean\_rev, scale=std\_rev) - norm.cdf(np.log1p(2\_500\_000), loc=mean\_rev, scale=std\_rev)

st.write("### Revenue Probability Estimates (based on log-normal distribution)")

st.write(f"1. \*\*P(Revenue < `$1.00M`)\*\* ≈ {p1:.4f}")

st.write(f"2. \*\*P(Revenue > `$1.00M`)\*\* ≈ {p2:.4f}")

st.write(f"3. \*\*P(`$2.50M` < Revenue < `$7.50M`)\*\* ≈ {p3:.4f}")

# Launch Price distribution

st.subheader("💰 Launch Price Distribution")

mean\_price, std\_price = plot\_and\_display\_distribution(df["Launch.Price"],

                                                      "Launch Price Distribution with Normal Fit",

                                                      "Launch Price ($)")

**Regression**

st.markdown("<h1 style='text-align: center;'>📈 Linear Regression</h1>", unsafe\_allow\_html=True)    # Titles & Styling

st.markdown("""

    <style>

        .main, .css-18e3th9, .css-1d391kg, .stApp {

            background-color: #599cba;

            color: white;

        }

        h1, h3 { color: white; }

        .nav-item { font-size: 18px; margin-right: 20px; display: inline-block; }

        .nav-item a { color: white; text-decoration: none; }

        .nav-item a:hover { color: #ffa500; }

    </style>

""", unsafe\_allow\_html=True)

# Load data & select feature

df = pd.read\_csv('games.csv')

col\_map = {

    "Reviews.Total": "Reviews Total",

    "Launch.Price": "Launch Price",

    "Revenue.Estimated": "Revenue Estimated"

}

features = list(col\_map.keys())

display\_feature = st.selectbox("Select Feature", [col\_map[f] for f in features[:-1]])

selected\_feature = [k for k, v in col\_map.items() if v == display\_feature][0]

# Apply IQR

def iqr\_filter(data, cols):

    for col in cols:

        Q1, Q3 = data[col].quantile([0.25, 0.75])

        IQR = Q3 - Q1

        data = data[(data[col] >= Q1 - 1.5 \* IQR) & (data[col] <= Q3 + 1.5 \* IQR)]

    return data

df\_filtered = iqr\_filter(df.copy(), features)

# Linear Regression & Plot

X = df\_filtered[[selected\_feature]]

y = df\_filtered["Revenue.Estimated"]

model = LinearRegression().fit(X, y)

y\_pred = model.predict(X)

sorted\_idx = X[selected\_feature].argsort()

X\_sorted, y\_sorted, y\_pred\_sorted = X.iloc[sorted\_idx], y.iloc[sorted\_idx], y\_pred[sorted\_idx]

st.subheader(f"{display\_feature} vs Revenue Estimated (IQR Filtered)")

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

ax.scatter(X, y, alpha=0.5, color='blue', label='Data')

ax.plot(X\_sorted, y\_pred\_sorted, '--', color='red', label='Regression Line')

ax.set\_xlabel(display\_feature)

ax.set\_ylabel("Revenue Estimated")

ax.legend()

st.pyplot(fig)

# Covariance and Correlation

cov = np.cov(X[selected\_feature], y)[0, 1]

corr = X[selected\_feature].corr(y)

st.markdown(f"\*\*Covariance:\*\* {cov:.4f}")

st.markdown(f"\*\*Correlation:\*\* {corr:.4f}")

# Correlation Matrix

st.subheader("Correlation Matrix (IQR Filtered)")

corr\_matrix = df\_filtered[features].corr()

fig2, ax2 = plt.subplots()

cax = ax2.matshow(corr\_matrix, cmap='coolwarm')

fig2.colorbar(cax)

ticks = range(len(features))

ax2.set\_xticks(ticks); ax2.set\_yticks(ticks)

ax2.set\_xticklabels(features, rotation=90)

ax2.set\_yticklabels(features)

for (i, j), val in np.ndenumerate(corr\_matrix.values):

    ax2.text(j, i, f"{val:.2f}", ha='center', va='center')

st.pyplot(fig2)

# Linear Regression Equation

B0, B1 = model.intercept\_, model.coef\_[0]

st.subheader("Linear Regression Equation")

st.markdown(f"`Revenue Estimated = {B0:.2f} + {B1:.2f} × {display\_feature}`")

explanation = {

    "Reviews.Total": "each additional review is associated with",

    "Launch.Price": "each unit increase in price is associated with"

}

st.markdown(f"Based on the model, {explanation[selected\_feature]} an estimated increase of \*\*{B1:.2f}\*\* in revenue.")

**6. Conclusion**

The analysis of Steam games reveals that both launch price and total user reviews significantly influence a game's estimated revenue. Descriptive statistics, visualizations, and regression modeling show that while most games fall within modest revenue ranges, a select few generate exceptionally high returns, leading to skewed distributions—especially in revenue and review counts. Log transformation helped normalize these distributions, allowing for more accurate statistical modeling. The correlation between total reviews and revenue (r = 0.5171) suggests a moderate positive relationship, while the stronger correlation with launch price (r = 0.7095) indicates that pricing strategy plays a more substantial role in predicting revenue. Overall, the findings underscore the importance of setting an optimal launch price and fostering user engagement to maximize financial success on the Steam platform.

[*https://steam-analysis.streamlit.app/*](https://steam-analysis.streamlit.app/)