Statistical Analysis of Steam Games

Mehfil

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.

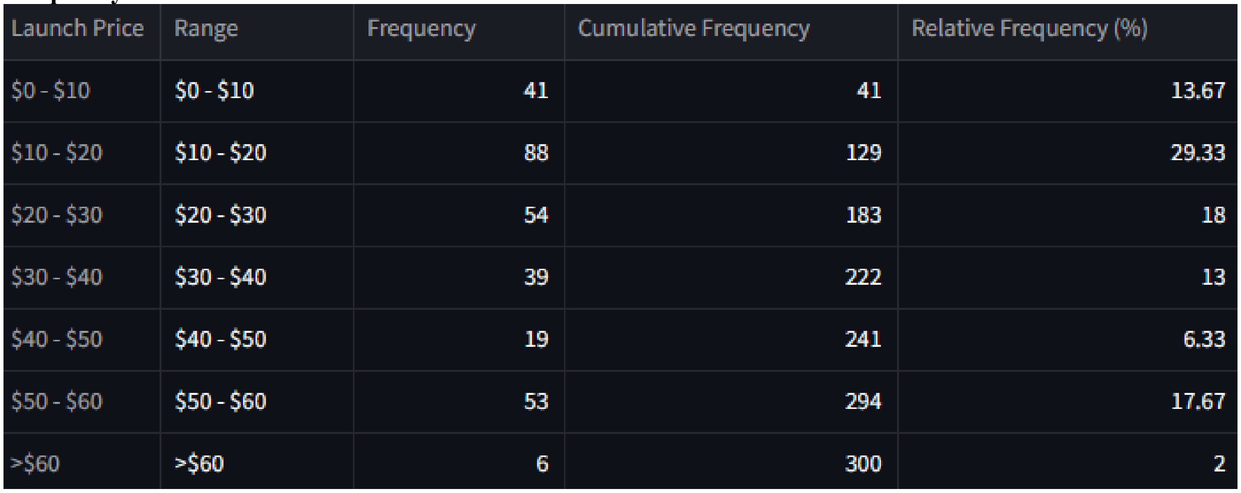
3. Data Description

www.reddit.com/r/gamedev/comments/165cii0

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.

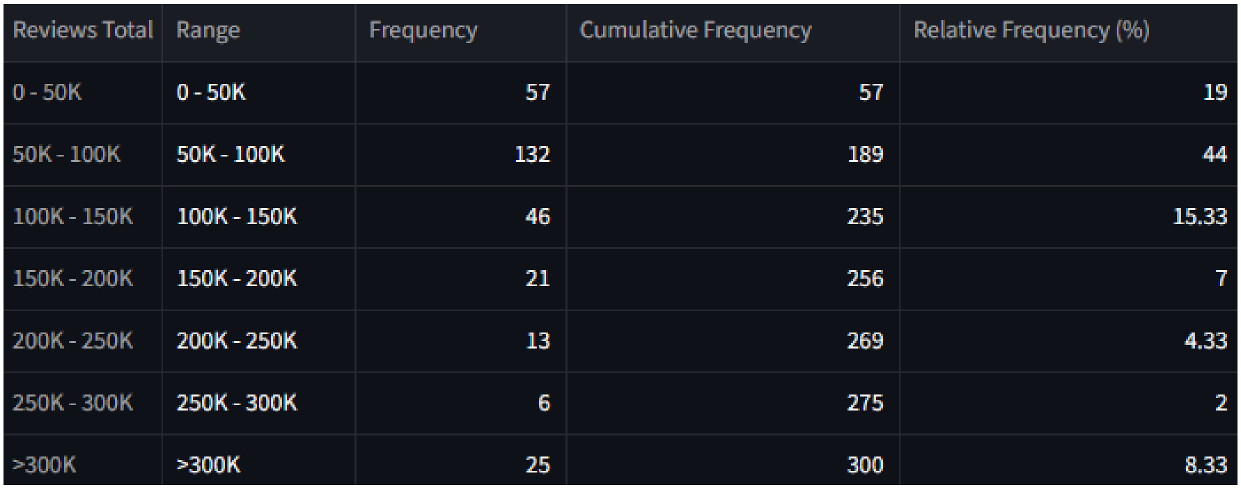
4. Results

Frequency Distribution Table



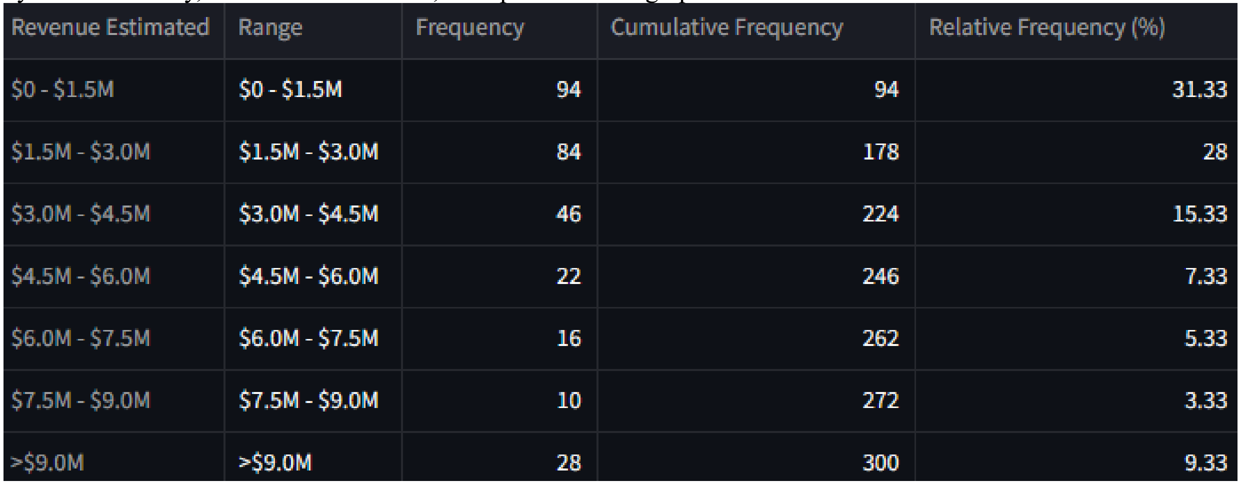
Launch Price	Range	Frequency	Cumulative Frequency	Relative Frequency (%)
$0 - $10	$0 - $10	41	41	13.67
$10 - $20	$10 - $20	88	129	29.33
$20 - $30	$20 - $30	54	183	18
$30 - $40	$30 - $40	39	222	13
$40 - $50	$40 - $50	19	241	6.33
$50 - $60	$50 - $60	53	294	17.67
>$60	>$60	6	300	2

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.



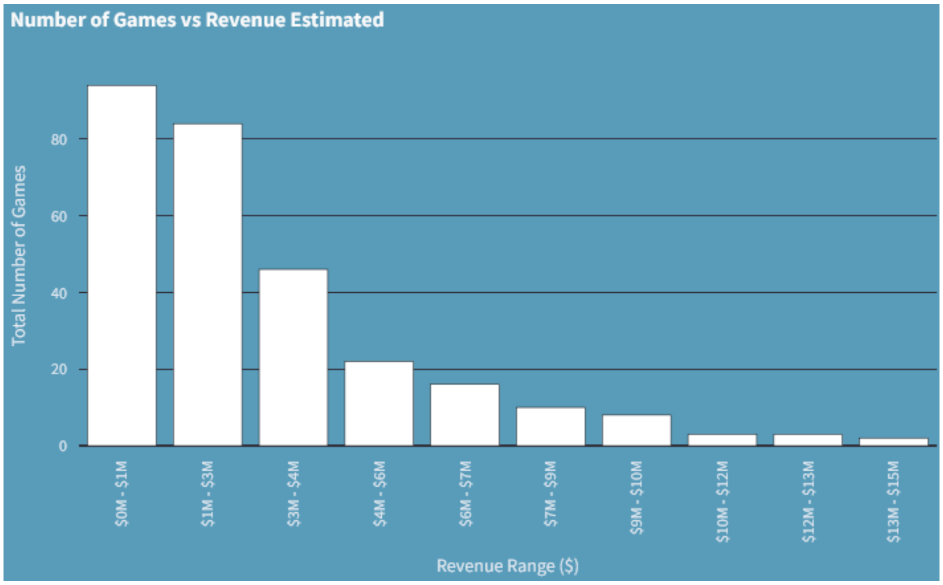
Reviews Total	Range	Frequency	Cumulative Frequency	Relative Frequency (%)
0 - 50K	0 - 50K	57	57	19
50K - 100K	50K - 100K	132	189	44
100K - 150K	100K - 150K	46	235	15.33
150K - 200K	150K - 200K	21	256	7
200K - 250K	200K - 250K	13	269	4.33
250K - 300K	250K - 300K	6	275	2
>300K	>300K	25	300	8.33

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.



Revenue Estimated	Range	Frequency	Cumulative Frequency	Relative Frequency (%)
$0 - $1.5M	$0 - $1.5M	94	94	31.33
$1.5M - $3.0M	$1.5M - $3.0M	84	178	28
$3.0M - $4.5M	$3.0M - $4.5M	46	224	15.33
$4.5M - $6.0M	$4.5M - $6.0M	22	246	7.33
$6.0M - $7.5M	$6.0M - $7.5M	16	262	5.33
$7.5M - $9.0M	$7.5M - $9.0M	10	272	3.33
>$9.0M	>$9.0M	28	300	9.33

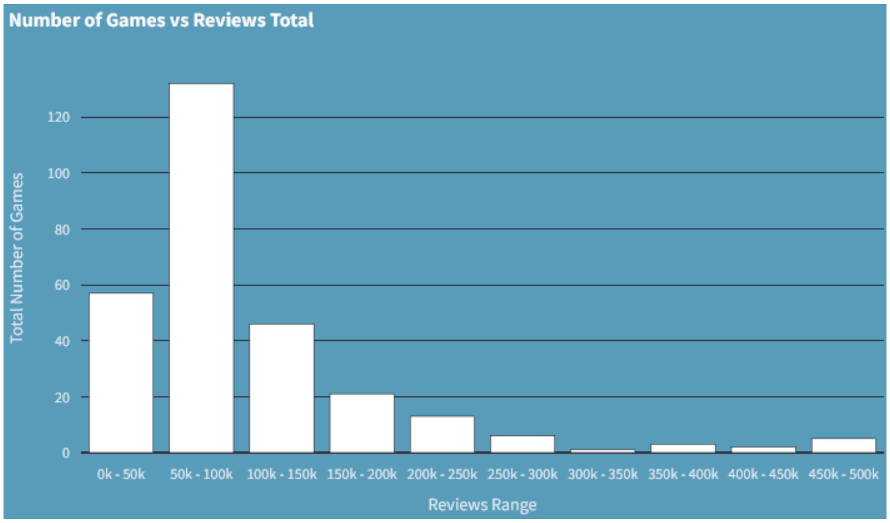
31.33% of the games earned only $0–1.5 million, while a notable 9.33% made over $9 million, with the highest reaching $110 million. This shows that despite a saturated market, successful games can still generate massive revenue.

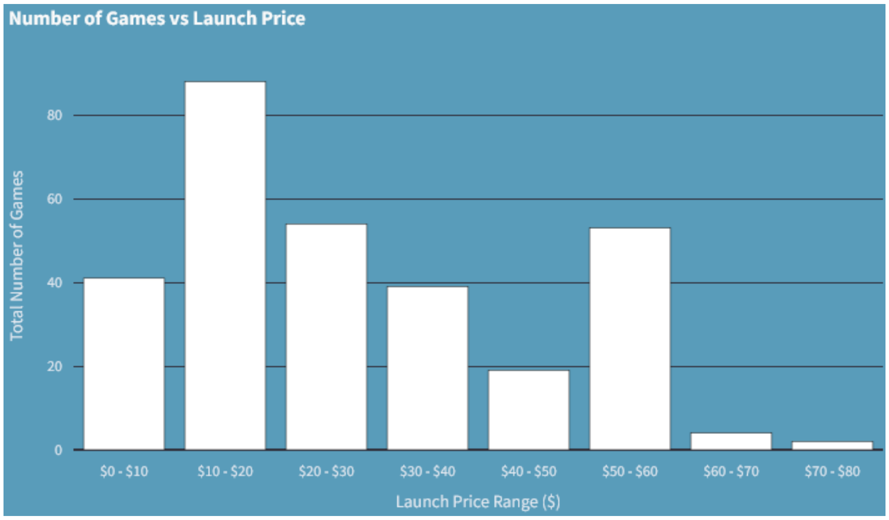


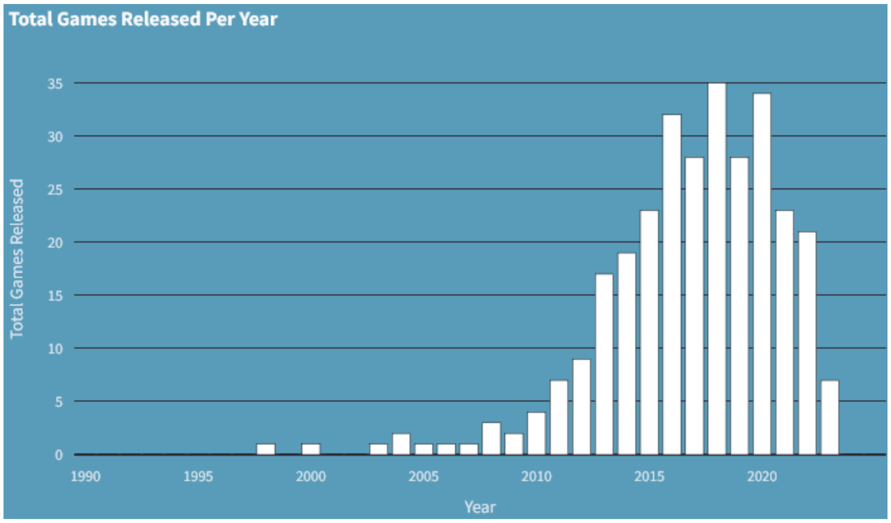
Bar Chart

Visual representation of the frequency distribution table.

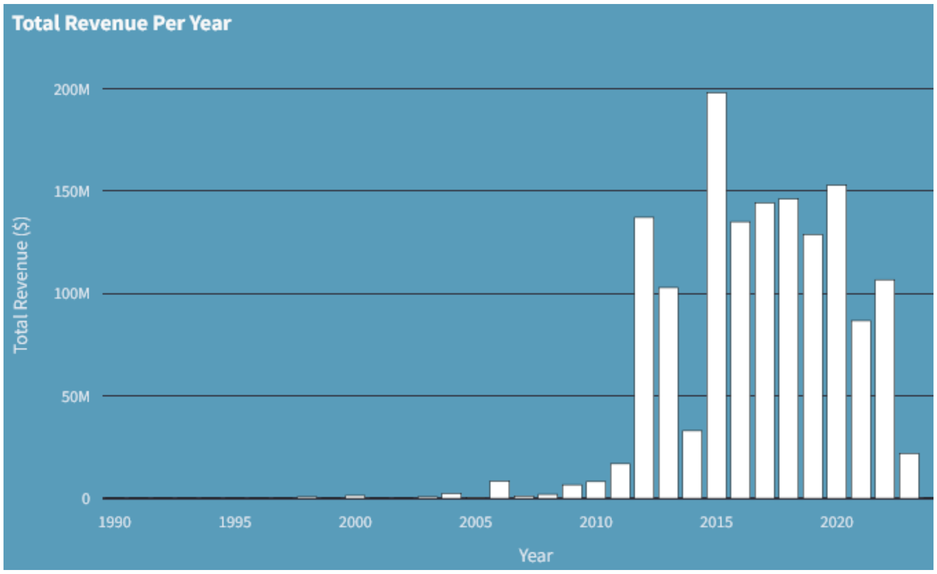
First three bar charts can be interpreted the same way as their tables.



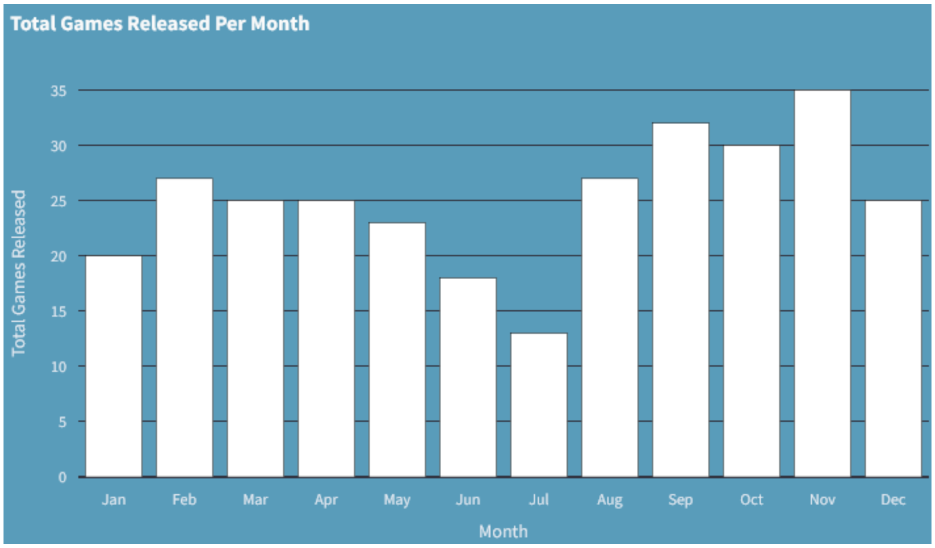




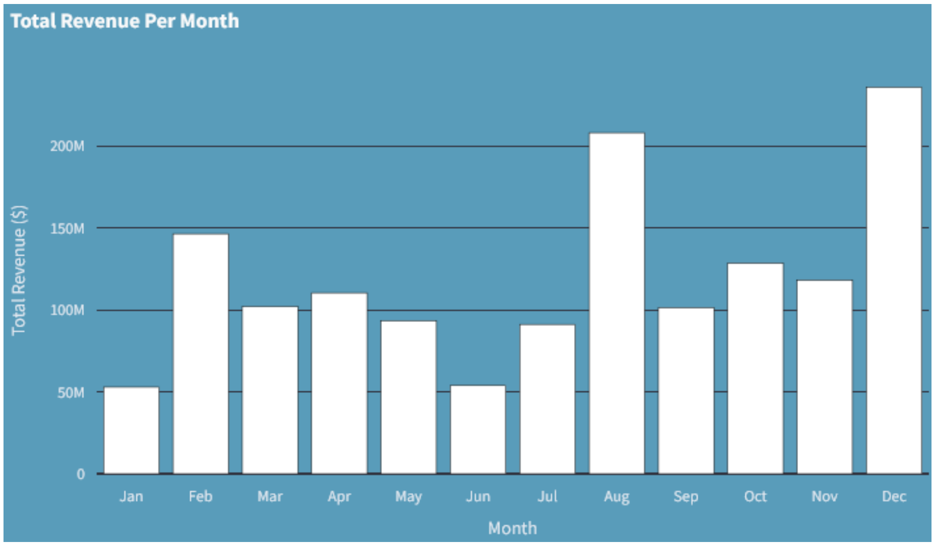
The bar chart shows an increase in the number of games released on Steam between 2008 and 2023, compared to earlier years. This trend reflects Steam's growing popularity as a distribution platform since its launch in 2003, with relatively few games published in its early years.



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 lockdown.

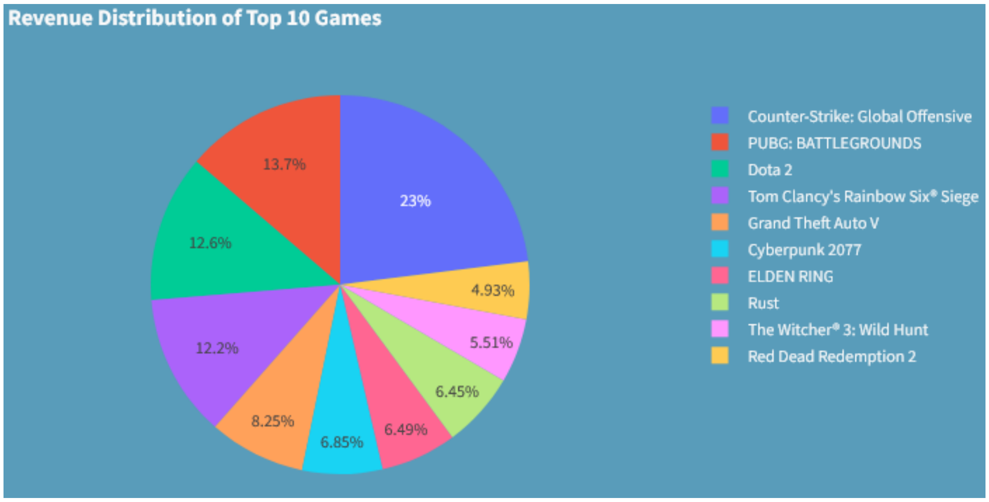


The monthly release chart shows fluctuating releases, with a peak in November just before the holiday season, as developers aim to boost sales during the holidays.

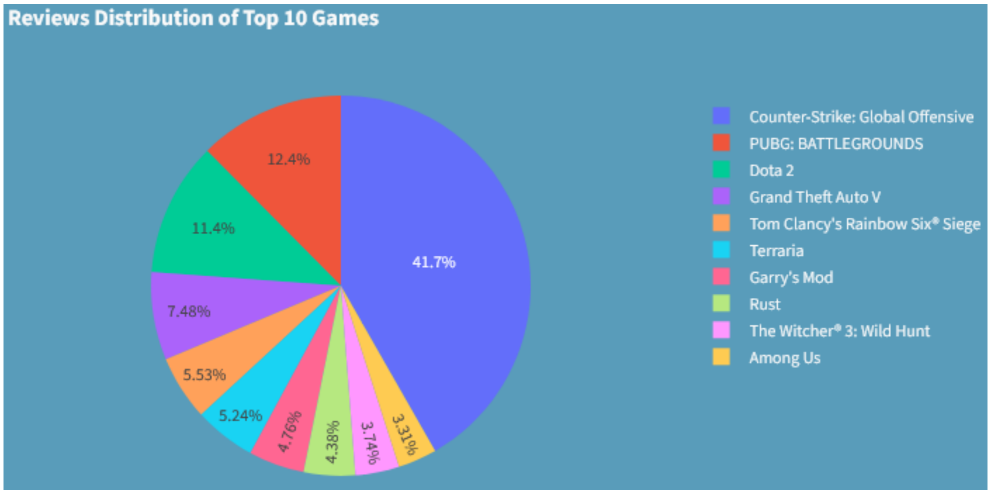


As a result, we see a peak in revenue generated in December, coinciding with the holiday season and increased game sales.

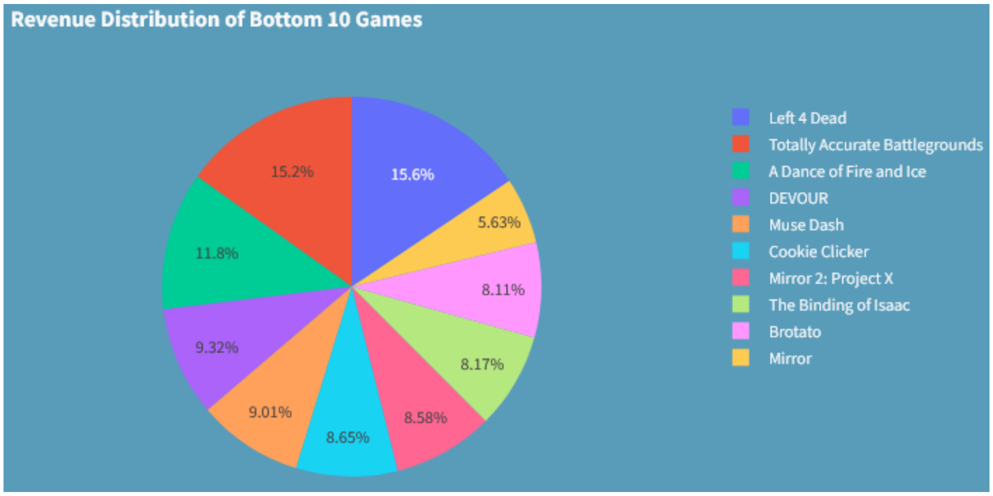
Pie Chart



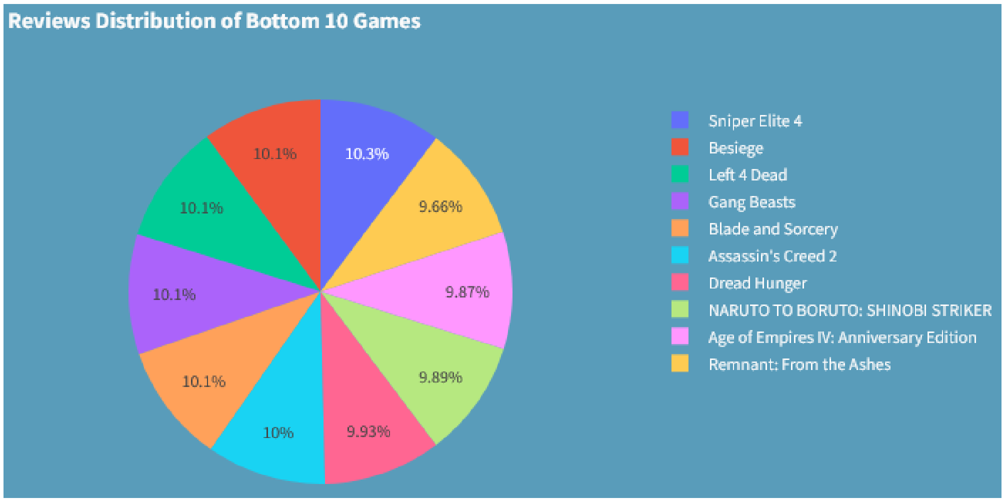
The pie chart shows that CS:GO, even among the top 10 games, accounts for 23% of the total revenue, highlighting its immense popularity. The remaining games show a gradual decrease in revenue, with RDR2 contributing the least at 4.93%.



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.



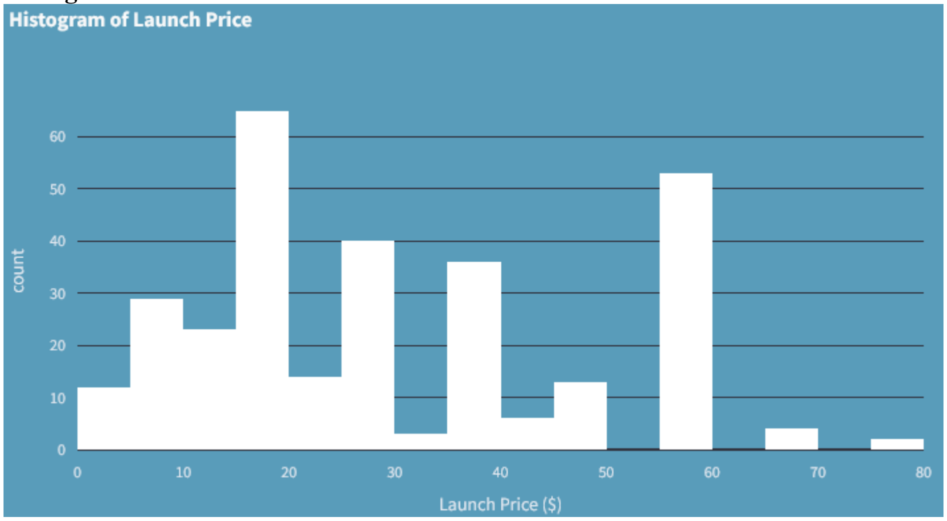
Among the bottom 10 games, the distribution is more balanced, with 2 games contributing over 15% and only 1 game generating less than 6%, indicating a narrower revenue gap within this group.



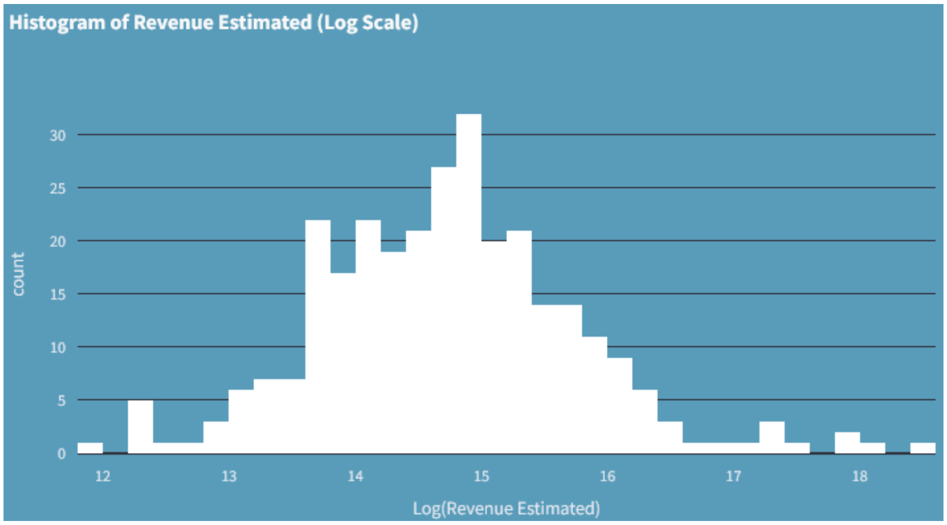
The review distribution among the bottom 10 games is almost even, with each title making up roughly 10% of the total, suggesting similar levels of player interaction.

***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.*

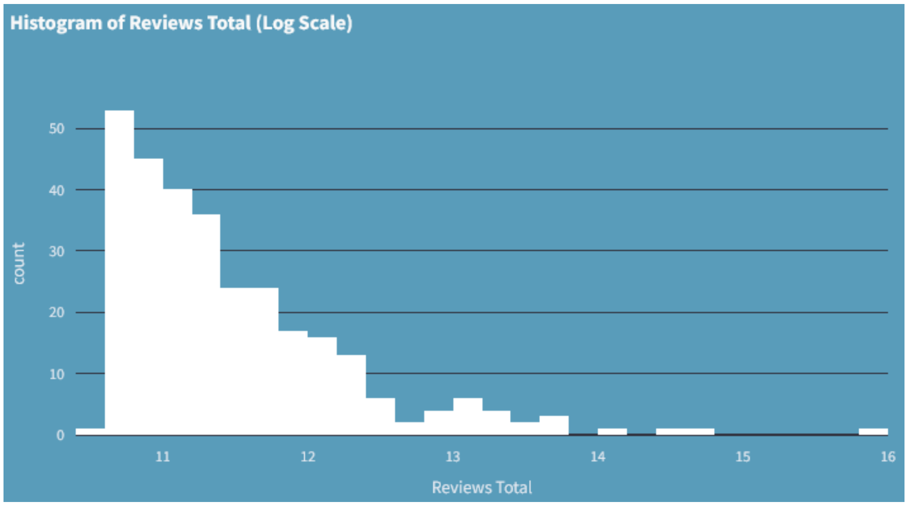
Histogram



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.



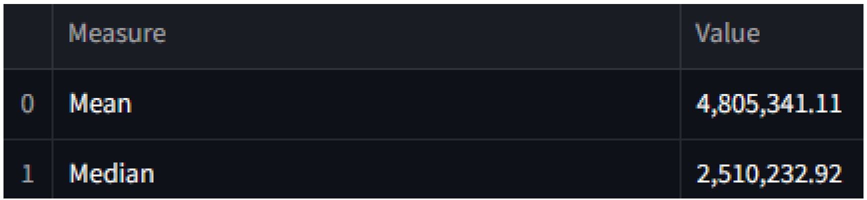
After log transforming the revenue data, the histogram shows an approximately normal distribution. Most games generate decent revenue, with a few making significantly less or more. The log transformation was needed to reduce the skew caused by extreme outliers, providing a clearer representation.



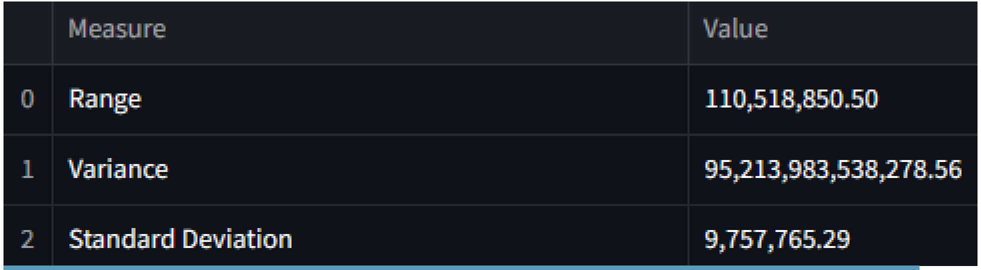
After log transforming the review data, right skewness remains, showing most games have few reviews, while a few games have many, indicating that only a small number of games have active communities leaving reviews.

Descriptive Statistics

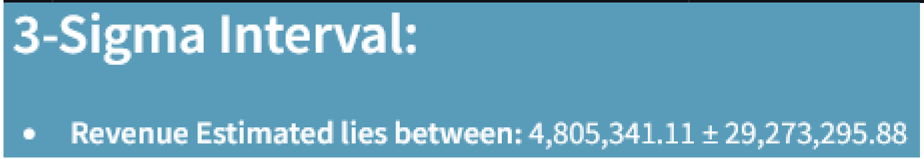
1. Revenue Estimated



	Measure	Value
0	Mean	4,805,341.11
1	Median	2,510,232.92

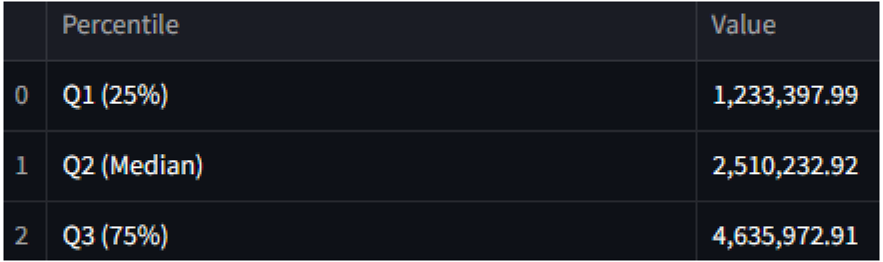


	Measure	Value
0	Range	110,518,850.50
1	Variance	95,213,983,538,278.56
2	Standard Deviation	9,757,765.29

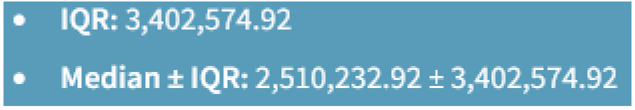


3-Sigma Interval:

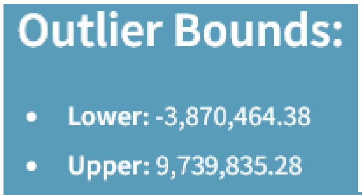
- **Revenue Estimated lies between:** 4,805,341.11 ± 29,273,295.88



	Percentile	Value
0	Q1 (25%)	1,233,397.99
1	Q2 (Median)	2,510,232.92
2	Q3 (75%)	4,635,972.91

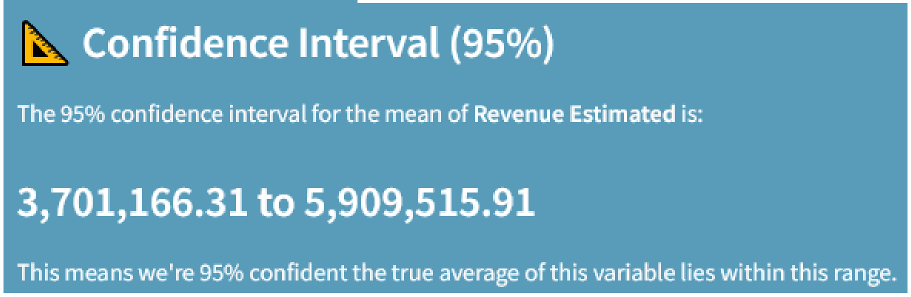


- **IQR:** 3,402,574.92
- **Median ± IQR:** 2,510,232.92 ± 3,402,574.92



Outlier Bounds:

- **Lower:** -3,870,464.38
- **Upper:** 9,739,835.28



📐 Confidence Interval (95%)

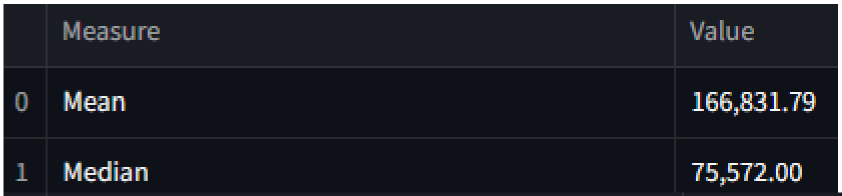
The 95% confidence interval for the mean of **Revenue Estimated** is:

3,701,166.31 to 5,909,515.91

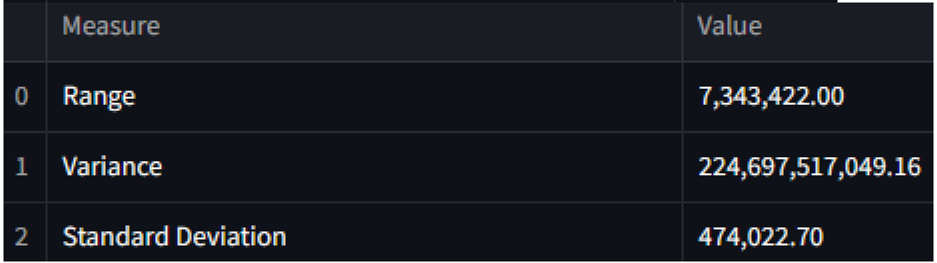
This means we're 95% confident the true average of this variable lies within this range.

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.

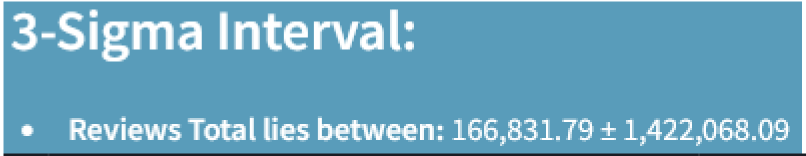
2. Reviews Total



	Measure	Value
0	Mean	166,831.79
1	Median	75,572.00

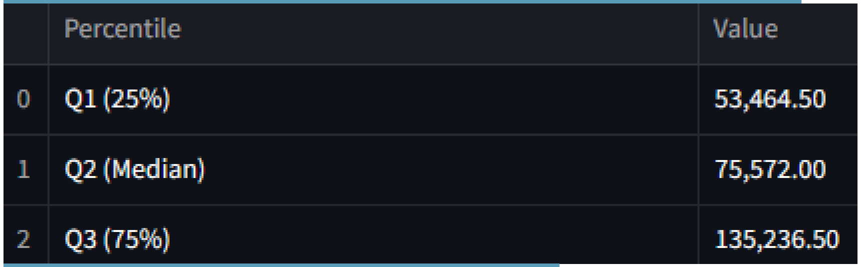


	Measure	Value
0	Range	7,343,422.00
1	Variance	224,697,517,049.16
2	Standard Deviation	474,022.70

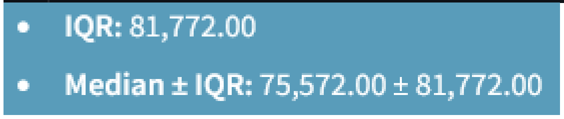


3-Sigma Interval:

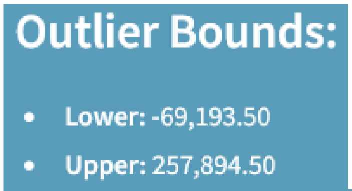
- **Reviews Total lies between:** 166,831.79 ± 1,422,068.09



	Percentile	Value
0	Q1 (25%)	53,464.50
1	Q2 (Median)	75,572.00
2	Q3 (75%)	135,236.50

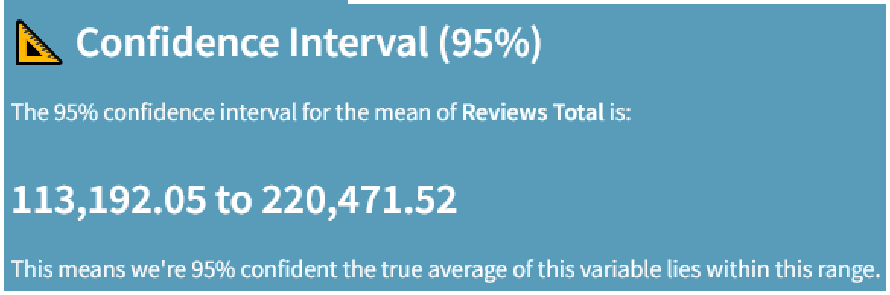


- **IQR:** 81,772.00
- **Median ± IQR:** 75,572.00 ± 81,772.00



Outlier Bounds:

- **Lower:** -69,193.50
- **Upper:** 257,894.50



📐 Confidence Interval (95%)

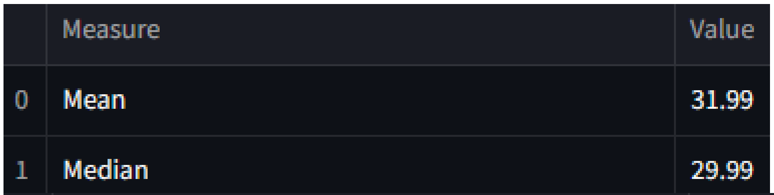
The 95% confidence interval for the mean of **Reviews Total** is:

113,192.05 to 220,471.52

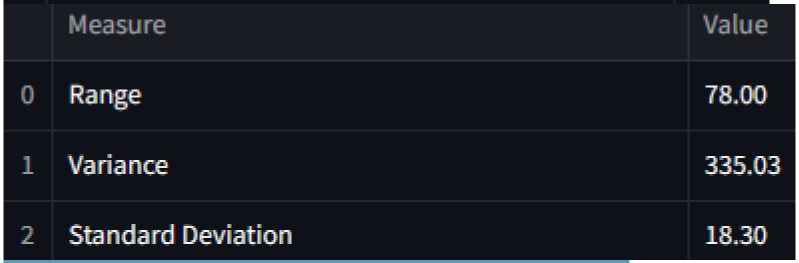
This means we're 95% confident the true average of this variable lies within this range.

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.

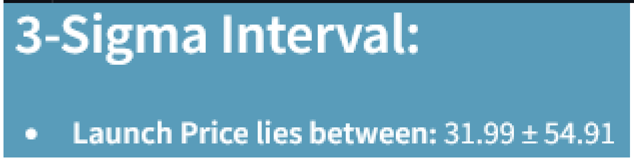
3. Launch Price



	Measure	Value
0	Mean	31.99
1	Median	29.99

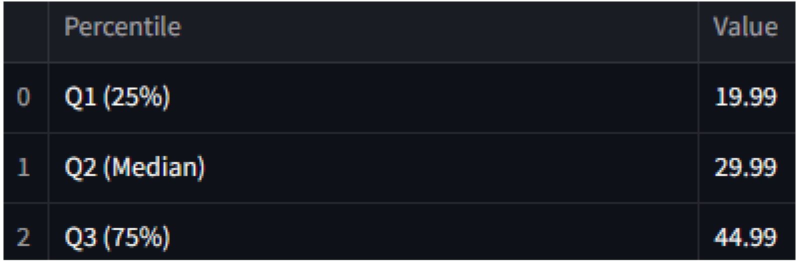


	Measure	Value
0	Range	78.00
1	Variance	335.03
2	Standard Deviation	18.30

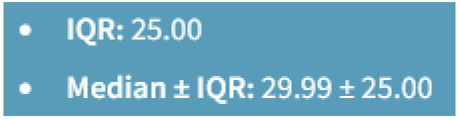


3-Sigma Interval:

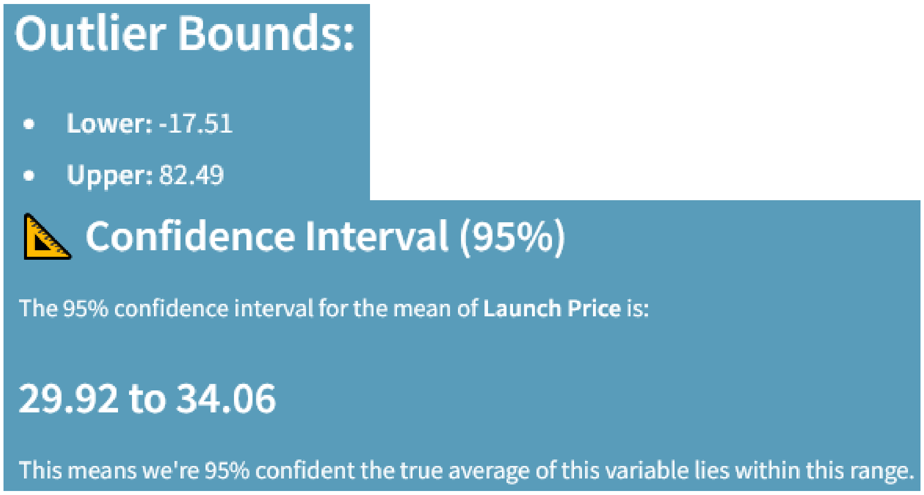
- **Launch Price lies between:** 31.99 ± 54.91



	Percentile	Value
0	Q1 (25%)	19.99
1	Q2 (Median)	29.99
2	Q3 (75%)	44.99



- **IQR:** 25.00
- **Median ± IQR:** 29.99 ± 25.00



Outlier Bounds:

- **Lower:** -17.51
- **Upper:** 82.49

📐 Confidence Interval (95%)

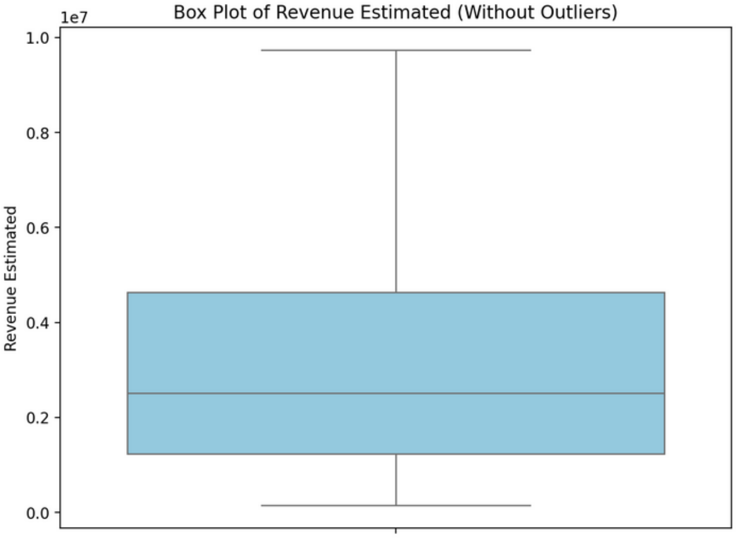
The 95% confidence interval for the mean of **Launch Price** is:

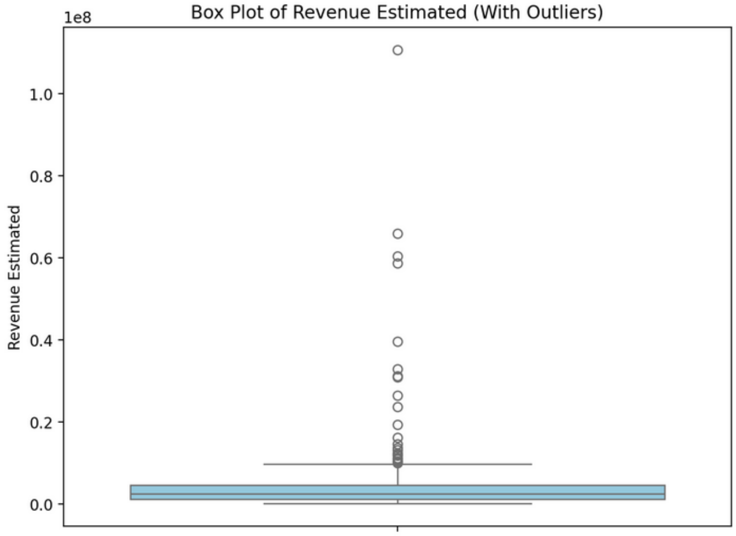
29.92 to 34.06

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

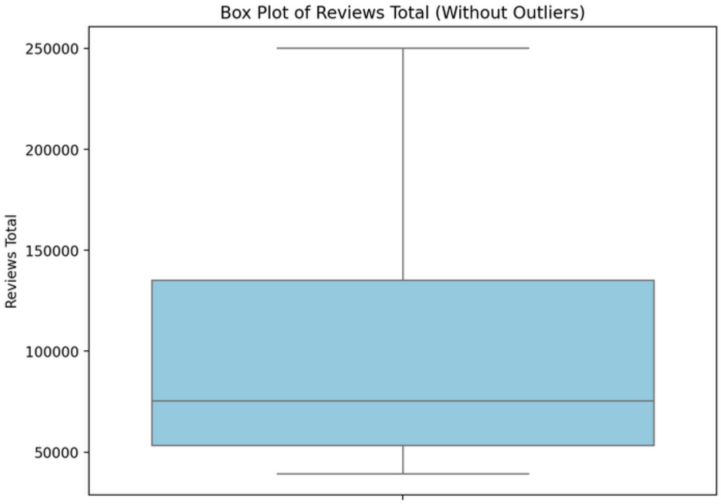
For launch price data under $80 with no true outliers, mean ± standard deviation is valid. However, due to skew from many low-priced games, median ± IQR remains more reliable for representing typical prices.

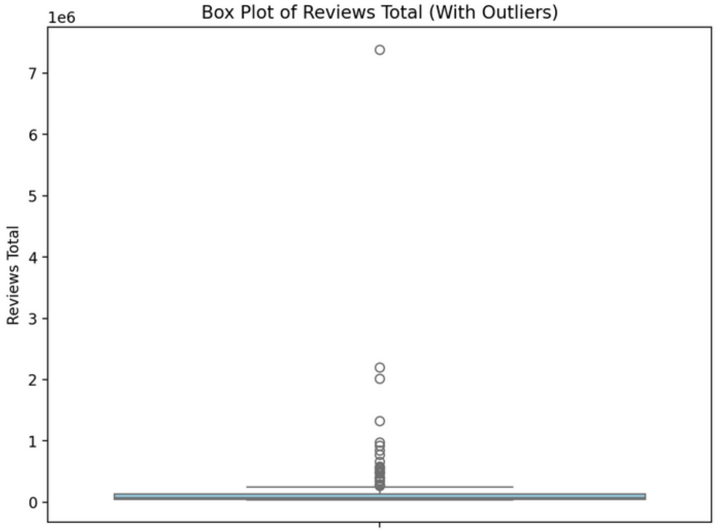
Box Plot



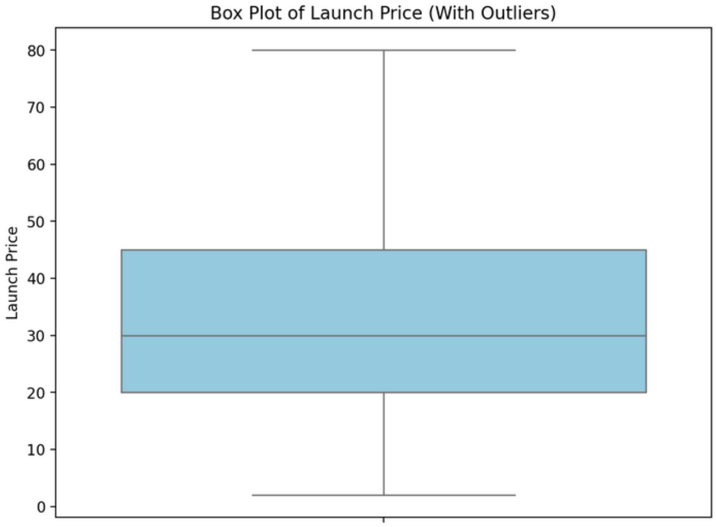


In the box plot with outliers, the data appears condensed near the bottom due to a few extreme high values stretching the scale. Without outliers, the plot more clearly shows the range between the lower and upper quartiles, where most of the revenue lies, providing a better view of the typical distribution.





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.

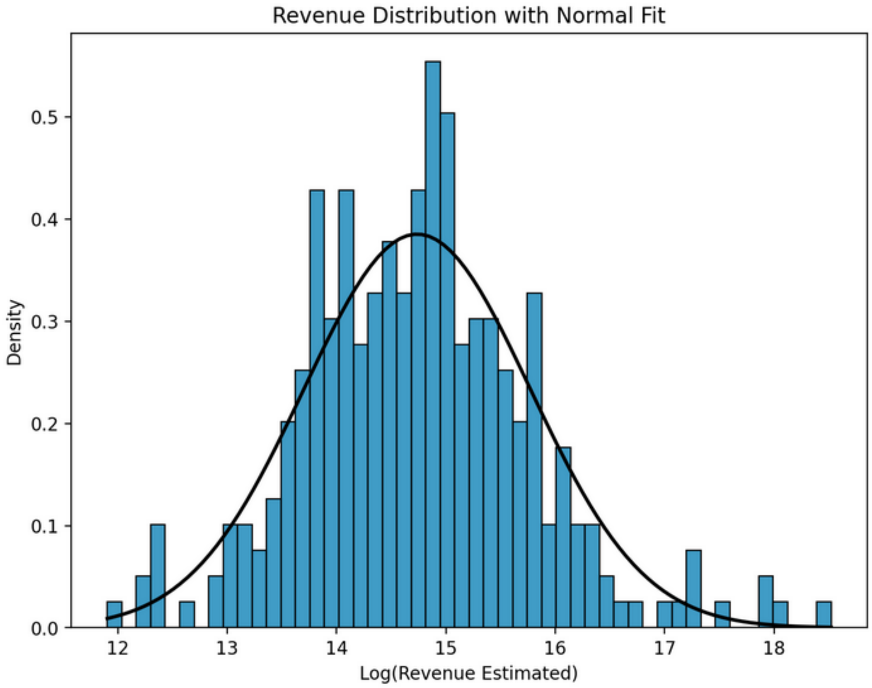


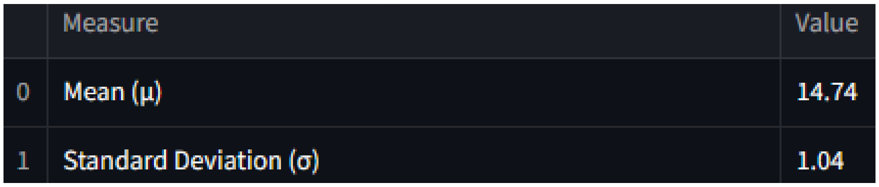
For launch price, the box plot remains identical with or without outliers, as no games exceed $80. This indicates a naturally limited range with no extreme values, making the distribution clearly visible without distortion.

Probability Distributions

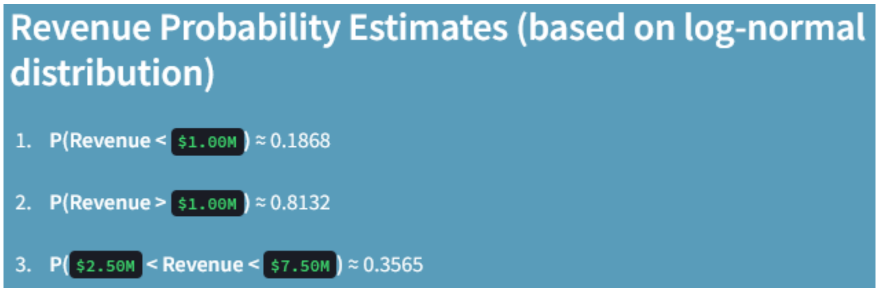
In the case of our data, a binomial distribution did not apply because there were no clear success/failure outcomes. Poisson distribution was unsuitable since the average rate was not constant over a fixed interval. Hypergeometric distribution also didn't fit, as it similarly relies on success/failure 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





	Measure	Value
0	Mean (μ)	14.74
1	Standard Deviation (σ)	1.04



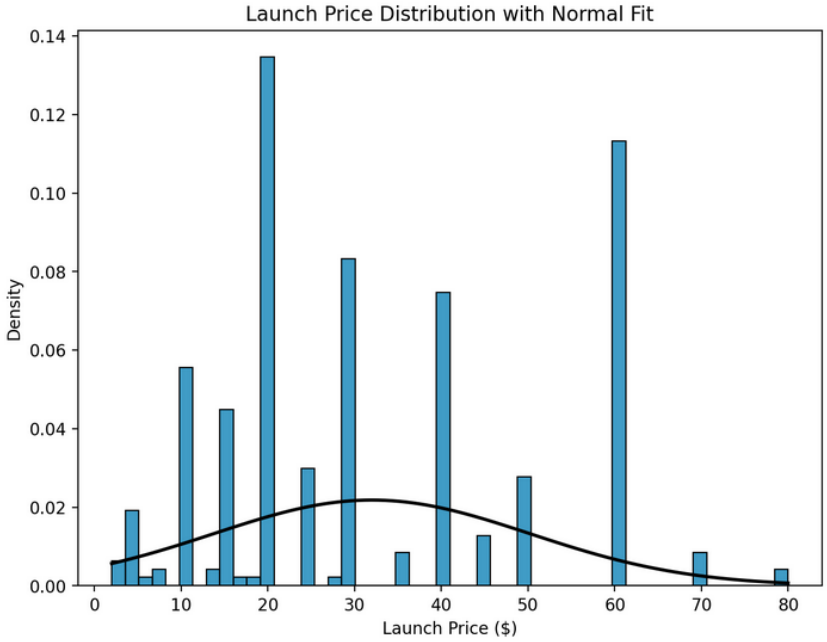
Revenue Probability Estimates (based on log-normal distribution)

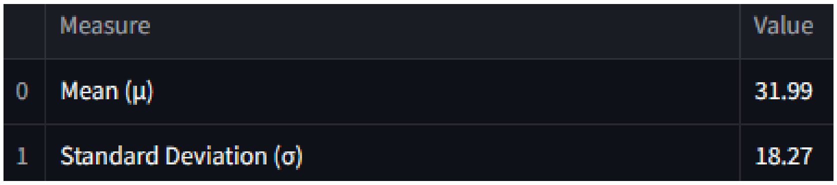
1. P(Revenue < $1.00M) ≈ 0.1868
2. P(Revenue > $1.00M) ≈ 0.8132
3. P($2.50M < Revenue < $7.50M) ≈ 0.3565

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

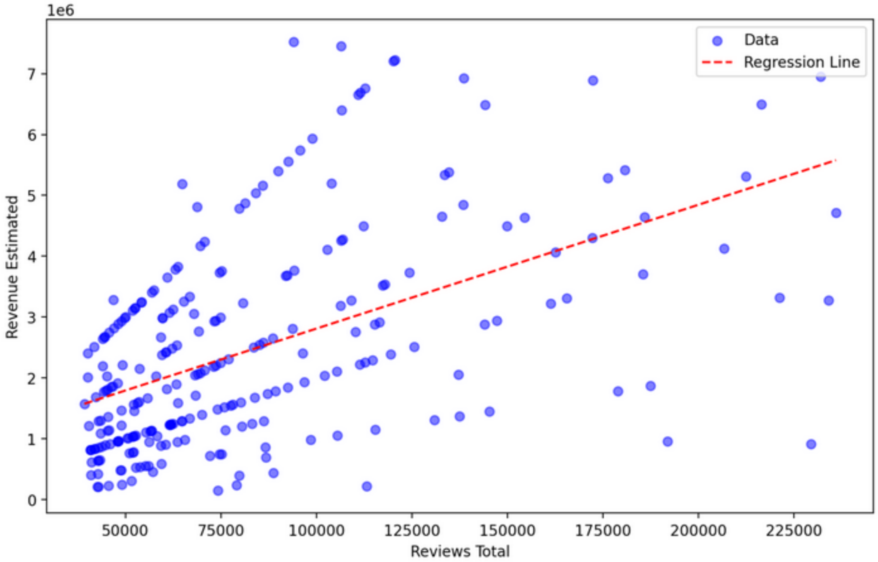
In the case of launch price, the data shows multiple sharp peaks, which suggests that the prices cluster around certain values rather than following a smooth, bell-shaped curve. A normal distribution assumes a continuous, symmetrical spread of data with a single peak. However, the presence of these spikes 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.



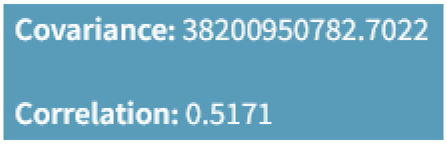


	Measure	Value
0	Mean (μ)	31.99
1	Standard Deviation (σ)	18.27

Linear Regression



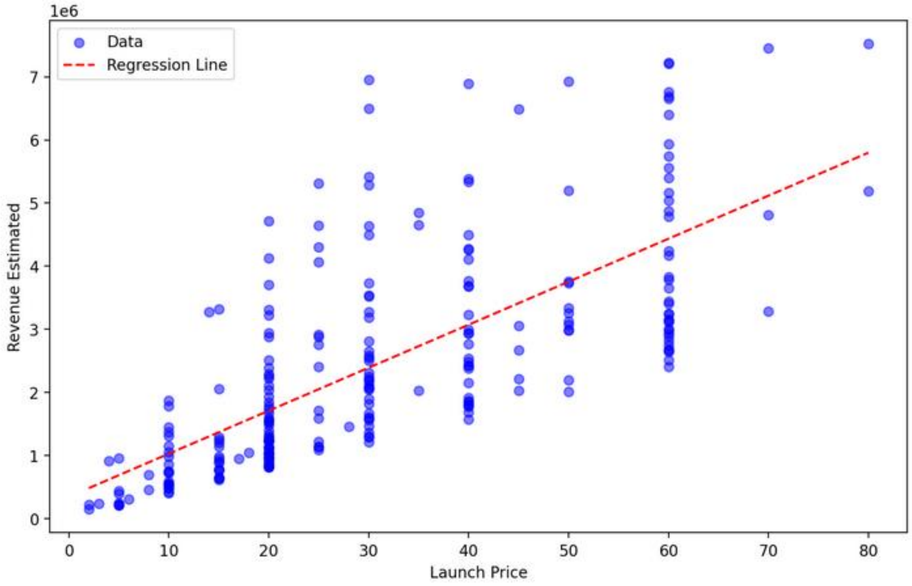
In the case of linear regression between revenue and total reviews, we observe a clear linear relationship after applying IQR filtering to remove outliers. Without this filtering, the graph would show a cluster of data points near the origin.



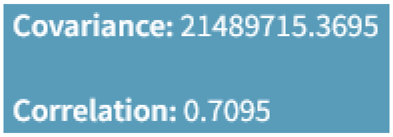
Covariance: 38200950782.7022

Correlation: 0.5171

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.

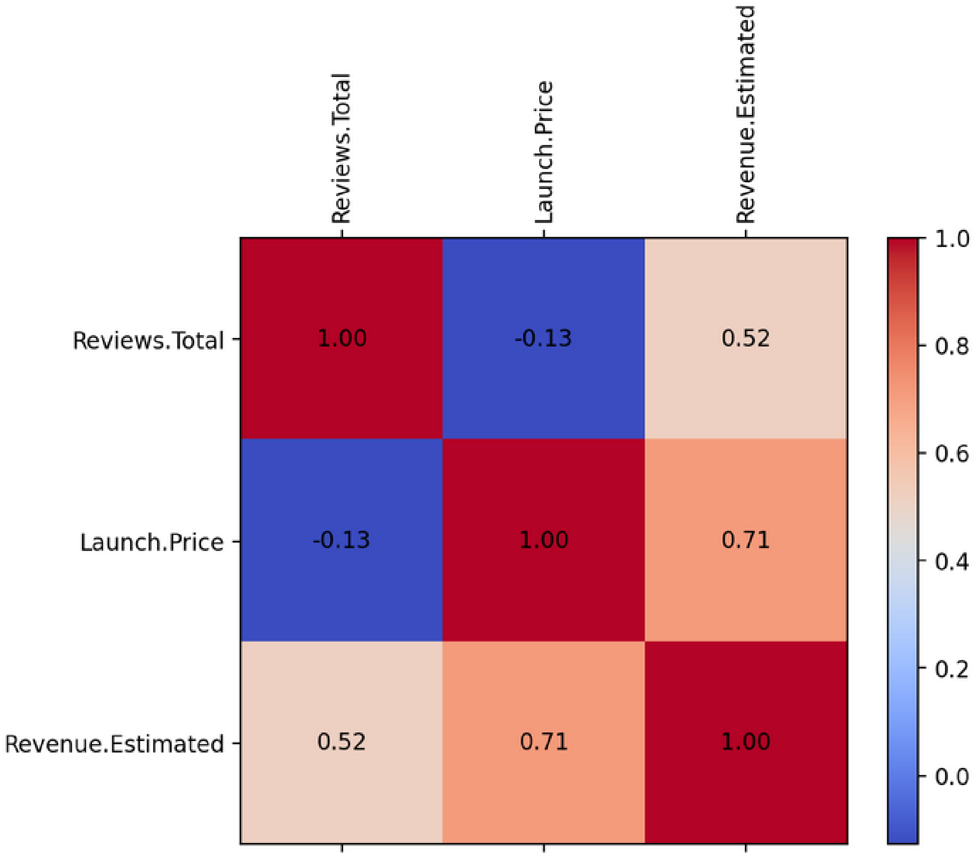


Covariance: 21489715.3695

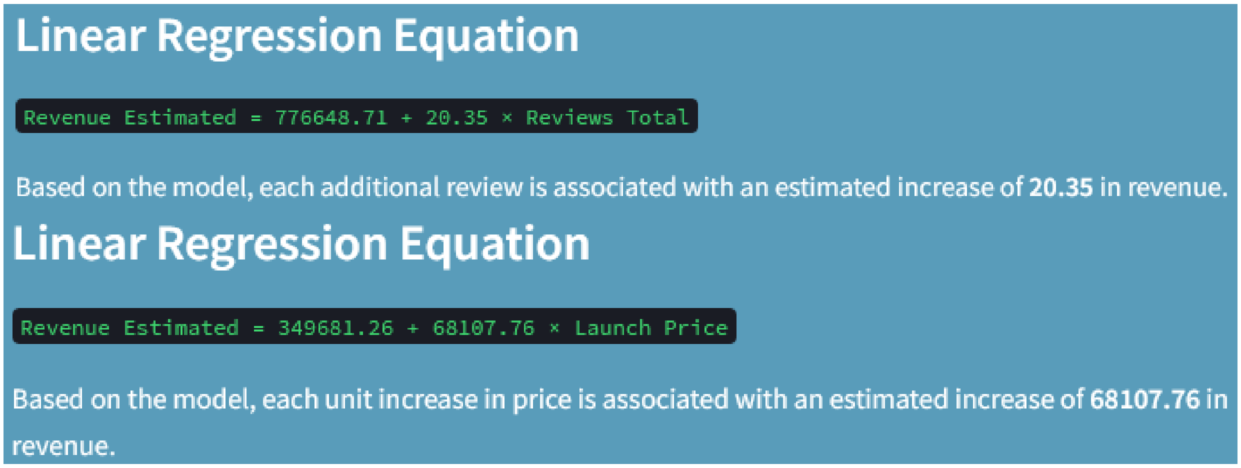
Correlation: 0.7095

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



Predictions



Linear Regression Equation

```
Revenue Estimated = 776648.71 + 20.35 × Reviews Total
```

Based on the model, each additional review is associated with an estimated increase of **20.35** in revenue.

Linear Regression Equation

```
Revenue Estimated = 349681.26 + 68107.76 × Launch Price
```

Based on the model, each unit increase in price is associated with an estimated increase of **68107.76** in revenue.

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

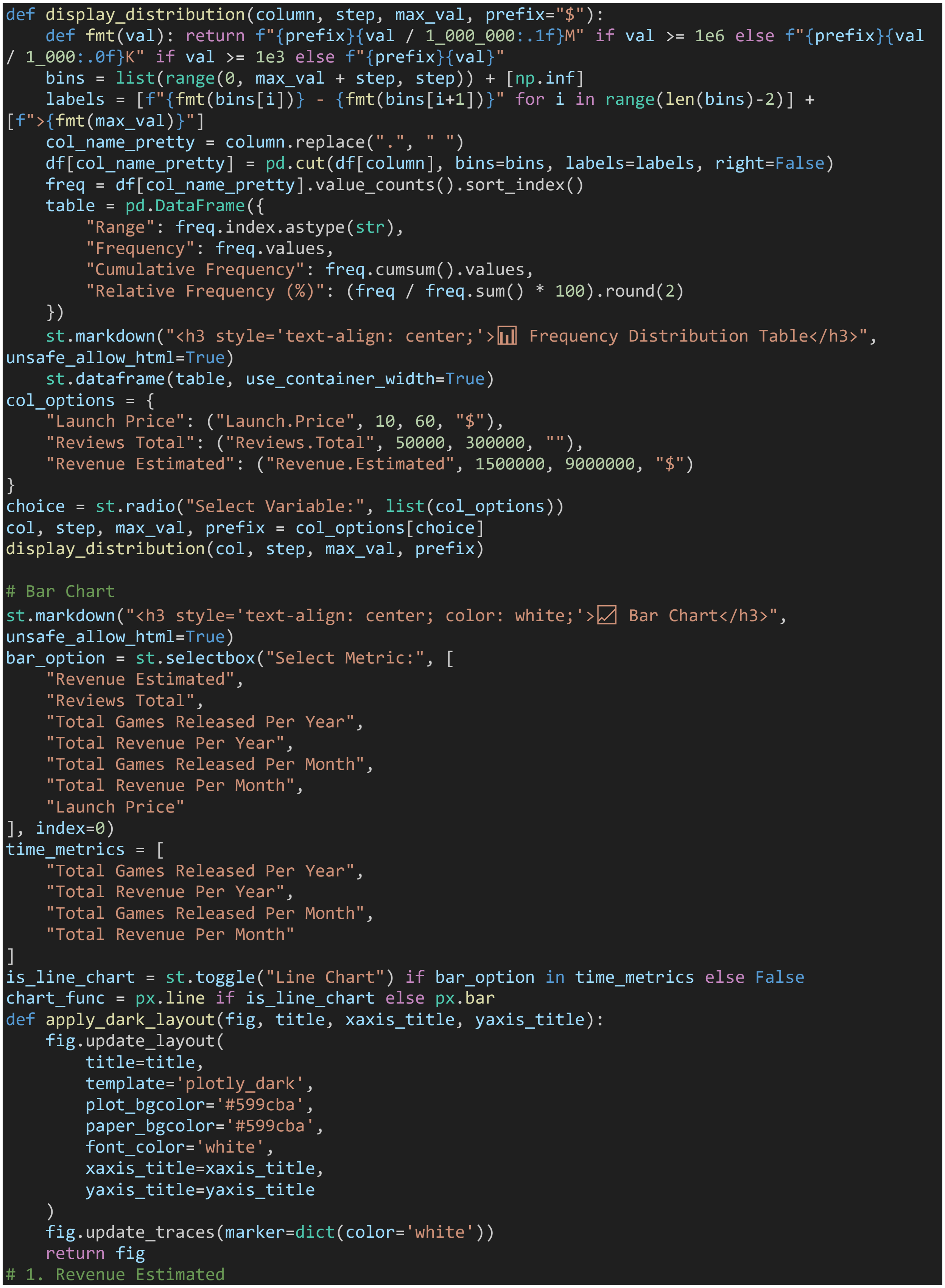
5. Codes

Libraries

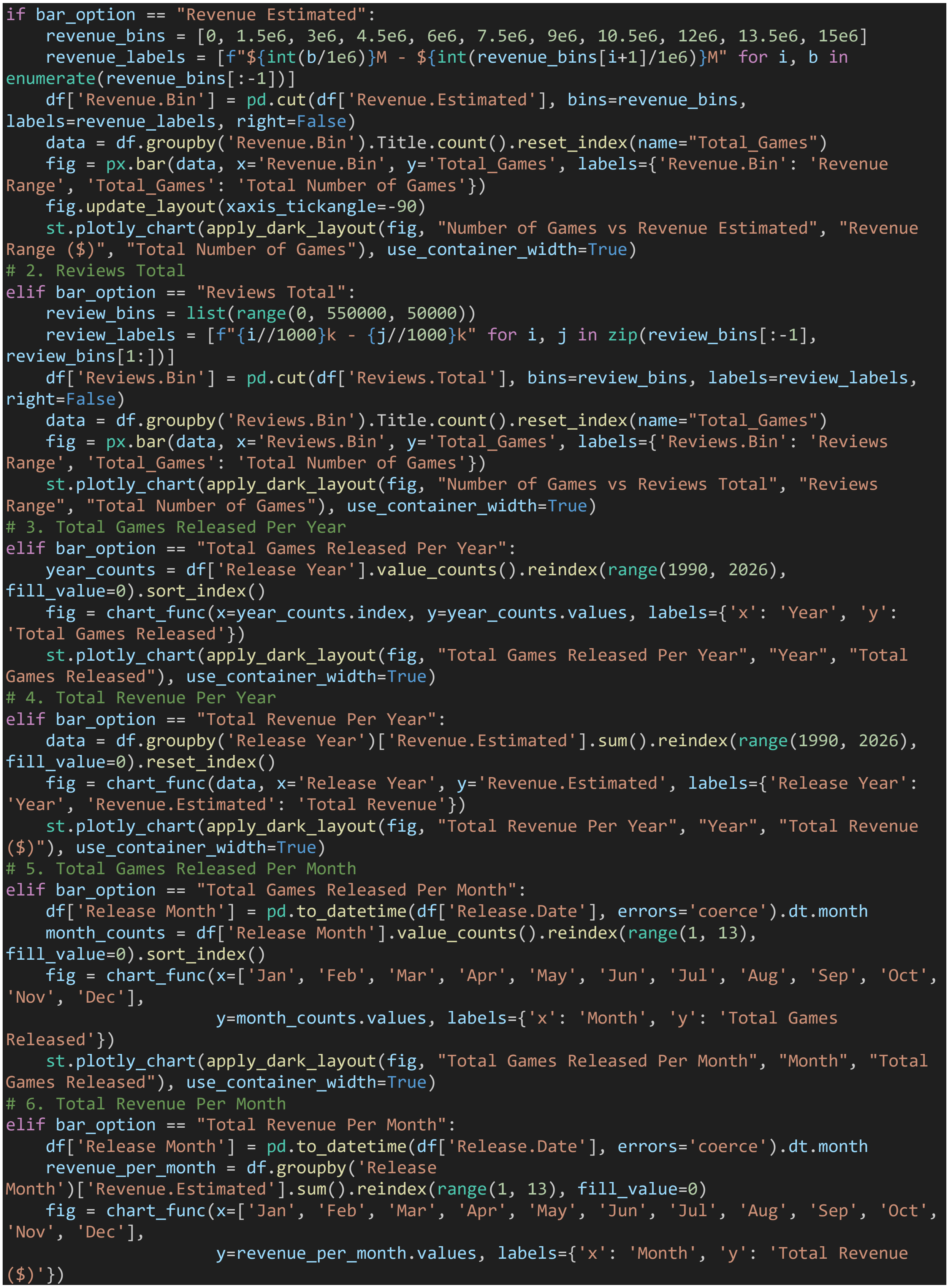
```
import streamlit as st  # For creating web apps with interactive widgets and layout
import pandas as pd     # For data manipulation and handling tabular data (DataFrames)
import numpy as np      # For numerical operations and working with arrays
import plotly.express as px  # For creating interactive 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/