

**Module Code – ITSCA2-B12**

**Assessment Type – Project**

**Campus Name – Mbombela**

**Student Number – NS.2022.R4G0J9**

## Question 1 (30 marks)

```
import sys

portfolio_accounts = []

sp500_top25 = {
    "Apple Inc.": 175.50,
    "Microsoft Corporation": 399.20,
    "Amazon.com Inc.": 154.30,
    "Alphabet Inc. Class A": 138.70,
    "NVIDIA Corporation": 690.45,
    "Meta Platforms, Inc.": 472.60,
    "Tesla, Inc.": 201.50,
    "Berkshire Hathaway Inc. Class B": 341.80,
    "UnitedHealth Group Incorporated": 492.60,
    "Johnson & Johnson": 160.30,
    "JPMorgan Chase & Co.": 181.40,
    "Visa Inc. Class A": 275.80,
    "Procter & Gamble Co.": 156.20,
    "Mastercard Incorporated Class A": 438.90,
    "Exxon Mobil Corporation": 105.60,
    "Home Depot, Inc.": 366.50,
    "Chevron Corporation": 145.70,
    "AbbVie Inc.": 163.20,
    "Pfizer Inc.": 29.40,
    "The Walt Disney Company": 110.60,
    "Comcast Corporation Class A": 42.30,
    "PepsiCo, Inc.": 173.80,
    "Cisco Systems, Inc.": 51.60,
    "Merck & Co., Inc.": 109.70,
    "Coca-Cola Company": 60.90
}

main_menu = ("-----\n"
             "Stock Portfolio Manager\n"
             "-----\n"
             "1. Add new portfolio\n"
             "2. Display all portfolios\n"
             "3. Display specific portfolio\n"
             "0. Exit\n\n"
             "Enter Choice: ")
```

```

account_menu = ("\\n-----\\n"
                "Manage Portfolio Account\\n"
                "-----\\n"
                "1. Add stock to portfolio\\n"
                "2. Buy shares\\n"
                "3. Sell shares\\n"
                "4. Deposit money\\n"
                "5. Withdraw money\\n"
                "6. Display available balance\\n"
                "0. Back\\n\\n"
                "Enter Choice: ")

def find_portfolio(id_number):
    """Find portfolio by ID and return it."""
    for account in portfolio_accounts:
        if account["ID Number"] == id_number:
            return account
    return None

def add_new_portfolio():
    fName = input("Enter first name: ")
    sName = input("Enter last name: ")
    idNum = input("Enter ID number: ")

    if find_portfolio(idNum):
        print("Portfolio already exists!")
    else:
        portfolio_accounts.append({
            "Name": fName,
            "Surname": sName,
            "ID Number": idNum,
            "Balance": 10000.00, # Default starting balance
            "Stocks": {} # Dictionary to hold stocks and their share
count
        })
    print("Portfolio successfully created.")

def display_all_portfolios():
    if not portfolio_accounts:
        print("No portfolios available.")
    else:
        for account in portfolio_accounts:

```

```

        print(f"\nID: {account['ID Number']} }\nName:
{account['Name']} } {account['Surname']} }\nBalance:
${account['Balance']:.2f}")

        if account["Stocks"]:
            print("Stocks:")
            for stock, shares in account["Stocks"].items():
                print(f" - {stock}: {shares} shares")

        else:
            print("No stocks added.")

def display_specific_portfolio():
    idNum = input("Enter Account Holder ID: ")
    account = find_portfolio(idNum)

    if account:
        print(
            f"\nID: {account['ID Number']} }\nName: {account['Name']} }
{account['Surname']} }\nBalance: ${account['Balance']:.2f}")
        print("Stocks:", account["Stocks"] if account["Stocks"] else
"No stocks added.")
        manage_account(account)
    else:
        print("Account does not exist.")

def manage_account(account):
    while True:
        try:
            choice = int(input(account_menu))
        except ValueError:
            print("Invalid input. Please enter a number.")
            continue # Restart loop to re-display the menu

        if choice == 1:
            print("\nAvailable Stocks:")
            for stock, price in sp500_top25.items():
                print(f"{stock}: ${price:.2f}")

            stock_name = input("\nEnter stock name: ")
            if stock_name in sp500_top25:
                try:
                    shares = int(input("Enter number of shares: "))
                    cost = sp500_top25[stock_name] * shares
                    if account["Balance"] >= cost:
                        account["Stocks"][stock_name] = shares
                        account["Balance"] -= cost
                        print(f"Added {shares} shares of {stock} at ${price:.2f}. New balance: ${account['Balance']:.2f}")
                    else:
                        print("Insufficient funds. Transaction failed.")

                except ValueError:
                    print("Invalid input. Please enter a number of shares.")


    else:
        print("Exiting program. Goodbye!")

```

```

                account["Stocks"][stock_name] =
account["Stocks"].get(stock_name, 0) + shares
                account["Balance"] -= cost
                print(f"Added {shares} shares of
{stock_name}.")
            else:
                print("Insufficient funds.")
        except ValueError:
            print("Invalid number of shares. Please enter a
valid integer.")
        else:
            print("Stock not found.")

    elif choice == 2:
        stock_name = input("Enter stock name: ")
        if stock_name in account["Stocks"]:
            try:
                shares = int(input("Enter number of shares to buy:
"))
                cost = sp500_top25[stock_name] * shares
                if account["Balance"] >= cost:
                    account["Stocks"][stock_name] += shares
                    account["Balance"] -= cost
                    print(f"Bought {shares} additional shares of
{stock_name}.")
                else:
                    print("Insufficient funds.")
            except ValueError:
                print("Invalid number of shares. Please enter a
valid integer.")
            else:
                print("Stock not found in portfolio.")

    elif choice == 3:
        stock_name = input("Enter stock name: ")
        if stock_name in account["Stocks"]:
            try:
                shares = int(input("Enter number of shares to
sell: "))
                if account["Stocks"][stock_name] >= shares:
                    account["Stocks"][stock_name] -= shares
                    account["Balance"] += sp500_top25[stock_name]
                    * shares
                    print(f"Sold {shares} shares of
{stock_name}.")
            except ValueError:
                print("Invalid number of shares. Please enter a
valid integer.")
            else:
                print("Stock not found in portfolio."))


```

```

        if account["Stocks"][stock_name] == 0:
            del account["Stocks"][stock_name] #
Remove stock if zero shares left
        else:
            print("Insufficient shares to sell.")
    except ValueError:
        print("Invalid number of shares. Please enter a
valid integer.")
    else:
        print("Stock not found in portfolio.")

elif choice == 4:
    try:
        amount = float(input("Enter deposit amount: "))
        account["Balance"] += amount
        print(f"Deposited ${amount:.2f}. New balance:
${account['Balance']:.2f}")
    except ValueError:
        print("Invalid amount. Please enter a valid number.")

elif choice == 5:
    try:
        amount = float(input("Enter withdrawal amount: "))
        if account["Balance"] >= amount:
            account["Balance"] -= amount
            print(f"Withdrew ${amount:.2f}. New balance:
${account['Balance']:.2f}")
        else:
            print("Insufficient balance.")
    except ValueError:
        print("Invalid amount. Please enter a valid number.")

elif choice == 6:
    print(f"Available Balance: ${account['Balance']:.2f}")

elif choice == 0:
    print("Returning to main menu.")
    break

else:
    print("Invalid input. Please re-enter.") # Handles
out-of-range numbers

# Main loop

```

```
while True:
    try:
        choice = int(input(main_menu))
    except ValueError:
        print("Invalid input. Please enter a number.")
        continue # Restart loop to re-display the menu

    if choice == 1:
        add_new_portfolio()

    elif choice == 2:
        display_all_portfolios()

    elif choice == 3:
        display_specific_portfolio()

    elif choice == 0:
        print("Exiting...")
        sys.exit()

    else:
        print("Invalid input. Please enter a valid menu option.") # Handles out-of-range numbers
```

## Stock Portfolio App Outputs

### Main Menu

```
-----  
Stock Portfolio Manager  
-----  
1. Add new portfolio  
2. Display all portfolios  
3. Display specific portfolio  
0. Exit
```

### Add New Portfolio (Automatically adds \$10 000 to account)

<pre>----- Stock Portfolio Manager ----- 1. Add new portfolio 2. Display all portfolios 3. Display specific portfolio 0. Exit  Enter Choice: 1 Enter first name: Angel Enter last name: Siwele Enter ID number: 00026070305060 Portfolio successfully created.</pre>	<pre>----- Stock Portfolio Manager ----- 1. Add new portfolio 2. Display all portfolios 3. Display specific portfolio 0. Exit  Enter Choice: 1 Enter first name: Bryan Enter last name: Ndlovu Enter ID number: 005250205060 Portfolio successfully created.</pre>
--	--

## Display All Portfolios

```
-----  
Stock Portfolio Manager  
-----  
1. Add new portfolio  
2. Display all portfolios  
3. Display specific portfolio  
0. Exit
```

Enter Choice: 2

ID: 00026070305060

Name: Angel Siwele

Balance: \$10000.00

No stocks added.

ID: 005250205060

Name: Bryan Ndlovu

Balance: \$10000.00

No stocks added.

## Display Specific Portfolio (Enters portfolio and allows you to make changes to portfolio)

```
-----  
Stock Portfolio Manager  
-----  
1. Add new portfolio  
2. Display all portfolios  
3. Display specific portfolio  
0. Exit
```

Enter Choice: 3

Enter Account Holder ID: 00026070305060

ID: 00026070305060

Name: Angel Siwele

Balance: \$10000.00

Stocks: No stocks added.

## Add Stock to Portfolio (Displays a list of stocks and their prices to choose from)

-----  
Manage Portfolio Account  
-----

1. Add stock to portfolio
2. Buy shares
3. Sell shares
4. Deposit money
5. Withdraw money
6. Display available balance
0. Back

Enter Choice: 1

Available Stocks:

Apple Inc.: \$175.50  
Microsoft Corporation: \$399.20  
Amazon.com Inc.: \$154.30  
Alphabet Inc. Class A: \$138.70  
NVIDIA Corporation: \$690.45  
Meta Platforms, Inc.: \$472.60  
Tesla, Inc.: \$201.50  
Berkshire Hathaway Inc. Class B: \$341.80  
UnitedHealth Group Incorporated: \$492.60  
Johnson & Johnson: \$160.30  
JPMorgan Chase & Co.: \$181.40  
Visa Inc. Class A: \$275.80  
Procter & Gamble Co.: \$156.20  
Mastercard Incorporated Class A: \$438.90  
Exxon Mobil Corporation: \$105.60  
Home Depot, Inc.: \$366.50  
Chevron Corporation: \$145.70  
AbbVie Inc.: \$163.20  
Pfizer Inc.: \$29.40  
The Walt Disney Company: \$110.60  
Comcast Corporation Class A: \$42.30  
PepsiCo, Inc.: \$173.80  
Cisco Systems, Inc.: \$51.60  
Merck & Co., Inc.: \$109.70  
Coca-Cola Company: \$60.90

Enter stock name: Apple Inc.  
Enter number of shares: 5  
Added 5 shares of Apple Inc..

-----  
Manage Portfolio Account  
-----

1. Add stock to portfolio
2. Buy shares
3. Sell shares
4. Deposit money
5. Withdraw money
6. Display available balance
0. Back

Enter Choice: 1

Available Stocks:

Apple Inc.: \$175.50  
Microsoft Corporation: \$399.20  
Amazon.com Inc.: \$154.30  
Alphabet Inc. Class A: \$138.70  
NVIDIA Corporation: \$690.45  
Meta Platforms, Inc.: \$472.60  
Tesla, Inc.: \$201.50  
Berkshire Hathaway Inc. Class B: \$341.80  
UnitedHealth Group Incorporated: \$492.60  
Johnson & Johnson: \$160.30  
JPMorgan Chase & Co.: \$181.40  
Visa Inc. Class A: \$275.80  
Procter & Gamble Co.: \$156.20  
Mastercard Incorporated Class A: \$438.90  
Exxon Mobil Corporation: \$105.60  
Home Depot, Inc.: \$366.50  
Chevron Corporation: \$145.70  
AbbVie Inc.: \$163.20  
Pfizer Inc.: \$29.40  
The Walt Disney Company: \$110.60  
Comcast Corporation Class A: \$42.30  
PepsiCo, Inc.: \$173.80  
Cisco Systems, Inc.: \$51.60  
Merck & Co., Inc.: \$109.70  
Coca-Cola Company: \$60.90

Enter stock name: Coca-Cola Company  
Enter number of shares: 10  
Added 10 shares of Coca-Cola Company.

## Display Balance

```
-----  
Manage Portfolio Account  
-----  
1. Add stock to portfolio  
2. Buy shares  
3. Sell shares  
4. Deposit money  
5. Withdraw money  
6. Display available balance  
0. Back
```

Enter Choice: 6

Available Balance: \$8513.50

## Deposit Money into Account

```
-----  
Manage Portfolio Account  
-----  
1. Add stock to portfolio  
2. Buy shares  
3. Sell shares  
4. Deposit money  
5. Withdraw money  
6. Display available balance  
0. Back
```

Enter Choice: 4

Enter deposit amount: 5000

Deposited \$5000.00. New balance: \$13513.50

## Buy Shares

```
-----  
Manage Portfolio Account  
-----  
1. Add stock to portfolio  
2. Buy shares  
3. Sell shares  
4. Deposit money  
5. Withdraw money  
6. Display available balance  
0. Back
```

```
Enter Choice: 2  
Enter stock name: Apple Inc.  
Enter number of shares to buy: 10  
Bought 10 additional shares of Apple Inc..
```

## Sell Shares

```
-----  
Manage Portfolio Account  
-----  
1. Add stock to portfolio  
2. Buy shares  
3. Sell shares  
4. Deposit money  
5. Withdraw money  
6. Display available balance  
0. Back
```

```
Enter Choice: 3  
Enter stock name: Apple Inc.  
Enter number of shares to sell: 5  
Sold 5 shares of Apple Inc..
```

-----  
Manage Portfolio Account  
-----

1. Add stock to portfolio
2. Buy shares
3. Sell shares
4. Deposit money
5. Withdraw money
6. Display available balance
0. Back

Enter Choice: 6

Available Balance: \$12636.00

Withdraw Money

- Manage Portfolio Account  
-----
1. Add stock to portfolio
  2. Buy shares
  3. Sell shares
  4. Deposit money
  5. Withdraw money
  6. Display available balance
  0. Back

Enter Choice: 5

Enter withdrawal amount: 4000

Withdrew \$4000.00. New balance: \$8636.00

## Display all Portfolios

```
-----  
Stock Portfolio Manager  
-----  
1. Add new portfolio  
2. Display all portfolios  
3. Display specific portfolio  
0. Exit
```

Enter Choice: 2

ID: 00026070305060  
Name: Angel Siwele  
Balance: \$8636.00  
Stocks:  
- Apple Inc.: 10 shares  
- Coca-Cola Company: 10 shares

ID: 005250205060  
Name: Bryan Ndlovu  
Balance: \$10000.00  
No stocks added.

```
-----  
Stock Portfolio Manager  
-----  
1. Add new portfolio  
2. Display all portfolios  
3. Display specific portfolio  
0. Exit
```

```
-----  
Stock Portfolio Manager  
-----
```

1. Add new portfolio
2. Display all portfolios
3. Display specific portfolio
0. Exit

```
Enter Choice: 3
```

```
Enter Account Holder ID: 00026070305060
```

```
ID: 00026070305060
```

```
Name: Angel Siwele
```

```
Balance: $8636.00
```

```
Stocks: {'Apple Inc.': 10, 'Coca-Cola Company': 10}
```

Exit Account Menu, Exit Main Menu (and app)

```
-----  
Manage Portfolio Account  
-----  
1. Add stock to portfolio  
2. Buy shares  
3. Sell shares  
4. Deposit money  
5. Withdraw money  
6. Display available balance  
0. Back
```

Enter Choice: 0  
Returning to main menu.

```
-----  
Stock Portfolio Manager  
-----  
1. Add new portfolio  
2. Display all portfolios  
3. Display specific portfolio  
0. Exit
```

Enter Choice: 0  
Exiting...  
An exception has occurred, use %tb to see the full traceback.

**SystemExit**

## Question 2 (40 marks)

```
import pandas as pd

# Load CSV file into Pandas Data Frame
df = pd.read_csv("Books_Data.csv", delimiter=";")
# Check dataset dimensions
dataset_size = df.shape

# Standardize column names (using snake_case for readability)
df.columns = [
    "index", "publishing_year", "book_name", "author",
    "language_code",
    "author_rating", "book_average_rating", "book_ratings_count",
    "genre",
    "gross_sales", "publisher_revenue", "sale_price", "sales_rank",
    "publisher", "units_sold"
]

# Handle missing values (removing rows with null values)
df.dropna(inplace=True)

# Fix "publishing_year" column
df["publishing_year"] = df["publishing_year"].astype(int)
df = df[(df["publishing_year"] > 1000) & (df["publishing_year"] <= 2025)]

# Normalize "author_rating" categories
rating_mapping = {
    "Novice": "Beginner",
    "Intermediate": "Intermediate",
    "Expert": "Expert"
}
df["author_rating"] = df["author_rating"].map(rating_mapping)

# Normalize "genre" categories
genre_mapping = {
    "genre fiction": "Fiction",
    "fiction": "Fiction",
    "nonfiction": "Non-Fiction",
    "children": "Children"
}
df["genre"] = df["genre"].map(genre_mapping)
```

```
# Compute operating costs
df["operating_cost"] = df["gross_sales"] - df["publisher_revenue"]

# Compute average book rating per publisher
avg_rating_per_publisher =
df.groupby("publisher") ["book_average_rating"].mean()

# Compute statistics for gross sales
gross_sales_stats = {
    "mean": df["gross_sales"].mean(),
    "median": df["gross_sales"].median(),
    "std_dev": df["gross_sales"].std()
}

# Remove "sales_rank" column
df.drop(columns=["sales_rank"], inplace=True)

# Create new dataframe for top 10 books with rating >= 4
top_books = df[df["book_average_rating"] >= 4].nlargest(10,
"book_average_rating")

# Display results
print("Dataset Size:", dataset_size)
print("\nCleaned Data Preview:\n", df.head())
print("\nAverage Rating Per Publisher:\n",
avg_rating_per_publisher.head())
print("\nGross Sales Statistics:", gross_sales_stats)
print("\nTop 10 Books with Rating >= 4:\n", top_books["book_name"])

# Create new CSV file with cleaned data
df.to_csv("Books_Data_Cleaned.csv", index=False)
```

## The Dimensions (size) of the Dataset

Dataset Size: (1070, 15)

## Preview of Cleaned Data

```
Cleaned Data Preview:
   index publishing_year          book_name \
0        0            1975           Beowulf
1        1            1987  Batman: Year One
2        2            2015  Go Set a Watchman
3        3            2008 When You Are Engulfed in Flames
4        4            2011 Daughter of Smoke & Bone

                           author language_code \
0             Unknown, Seamus Heaney      en-US
1  Frank Miller, David Mazzucchelli, Richmond Lew...      eng
2                  Harper Lee      eng
3                  David Sedaris      en-US
4                  Laini Taylor      eng

  author_rating  book_average_rating  book_ratings_count    genre \
0     Beginner                 3.42           155903.0  Fiction
1 Intermediate                4.23           145267.0  Fiction
2     Beginner                 3.31           138669.0  Fiction
3 Intermediate                4.04           150898.0  Fiction
4 Intermediate                4.04           198283.0  Fiction

   gross_sales  publisher_revenue  sale_price          publisher \
0      34160.0            20496.0       4.88  HarperCollins Publishers
1     12437.5            7462.5       1.99  HarperCollins Publishers
2      47795.0            28677.0       8.69  Amazon Digital Services, Inc.
3      41250.0            24750.0       7.50  Hachette Book Group
4      37952.5            22771.5       7.99  Penguin Group (USA) LLC

  units_sold  operating_cost
0      7000.0            13664.0
1      6250.0            4975.0
2      5500.0            19118.0
3      5500.0            16500.0
4      4750.0            15181.0
```

## Total Showing the Average Rating Per Publisher

```
Average Rating Per Publisher:
  publisher
Amazon Digital Services, Inc.        4.002811
Hachette Book Group                  3.949818
HarperCollins Christian Publishing  4.170000
HarperCollins Publishers            3.995846
HarperCollins Publishing            4.130000
Name: book_average_rating, dtype: float64
```

Note: Shows Amazon Digital Publisher has the highest ratings at 4.0, closely followed by Hachette Book Group

### Gross Sales Statistics

```
Gross Sales Statistics: {'mean': np.float64(1892.2366161616164), 'median': 809.745, 'std_dev': 4038.0081893161355}
```

Dataframe with the names of the top 10 books with a rating of 4 and above



```
Top 10 Books with Rating >= 4:  
331           Words of Radiance  
777           A Court of Mist and Fury  
479   The Essential Calvin and Hobbes: A Calvin and ...  
31           The Way of Kings  
249           Calvin and Hobbes  
733           Queen of Shadows  
433   The Hobbit and The Lord of the Rings  
965   A Storm of Swords: Part 2 Blood and Gold  
153           The House of Hades  
491           Heir of Fire  
Name: book_name, dtype: object
```

## Question 3 (30 marks)

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load cleaned dataset from Question 2
file_path = "Books_Data_Cleaned.csv" # Ensure this is your cleaned
dataset
df = pd.read_csv(file_path)

# Bar graph showing Top 10 Highest Rated Books
top_10_books = df.nlargest(10, "book_average_rating")
plt.figure(figsize=(12, 6))
sns.barplot(data=top_10_books, x="book_average_rating",
y="book_name", palette="viridis")
plt.xlabel("Average Rating")
plt.ylabel("Book Name")
plt.title("Top 10 Highest Rated Books")
plt.show()

# Pie chart showing Distribution of Sales Across Genres
plt.figure(figsize=(8, 6))
genre_sales = df.groupby("genre")["gross_sales"].sum()
genre_sales.plot(kind="pie", autopct='%.1f%%', cmap="Set3",
startangle=140)
plt.title("Distribution of Sales Across Genres")
plt.ylabel("") # Hide y-label
plt.show()

# Line graph showing Sales Distribution of Fiction Books Over Time
fiction_df = df[df["genre"] == "Fiction"]
plt.figure(figsize=(10, 5))
sns.lineplot(data=fiction_df, x="publishing_year", y="gross_sales",
marker="o", ci=None)
plt.xlabel("Publishing Year")
plt.ylabel("Gross Sales")
plt.title("Sales Distribution of Fiction Books Over Time")
plt.grid()
plt.show()

# Bar graph showing Sales Distribution Across Various Publishers
plt.figure(figsize=(12, 6))
```

```

publisher_sales =
df.groupby("publisher") ["gross_sales"].sum().nlargest(10)
sns.barplot(x=publisher_sales.values, y=publisher_sales.index,
palette="coolwarm")
plt.xlabel("Total Sales")
plt.ylabel("Publisher")
plt.title("Sales Distribution Across Various Publishers")
plt.show()

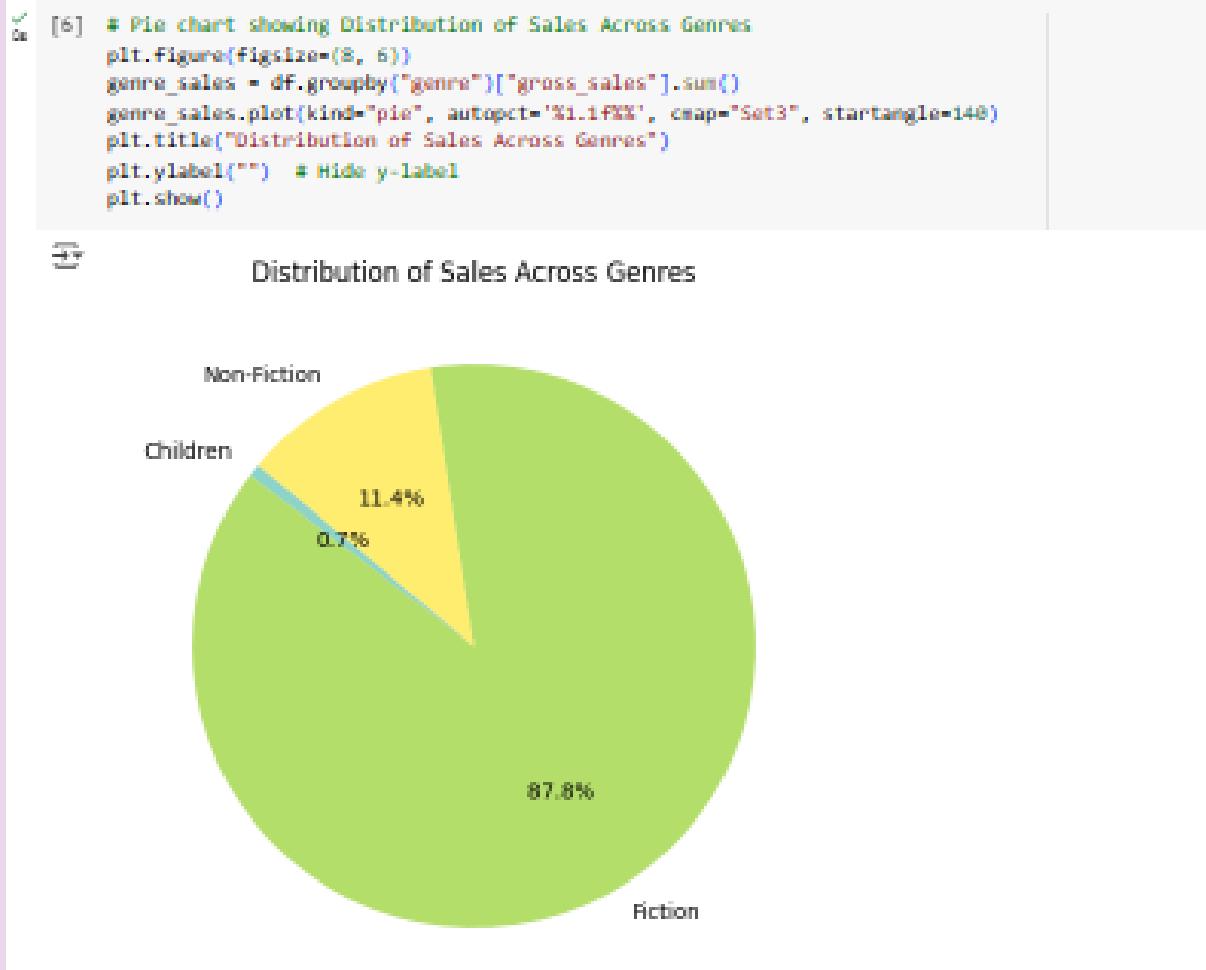
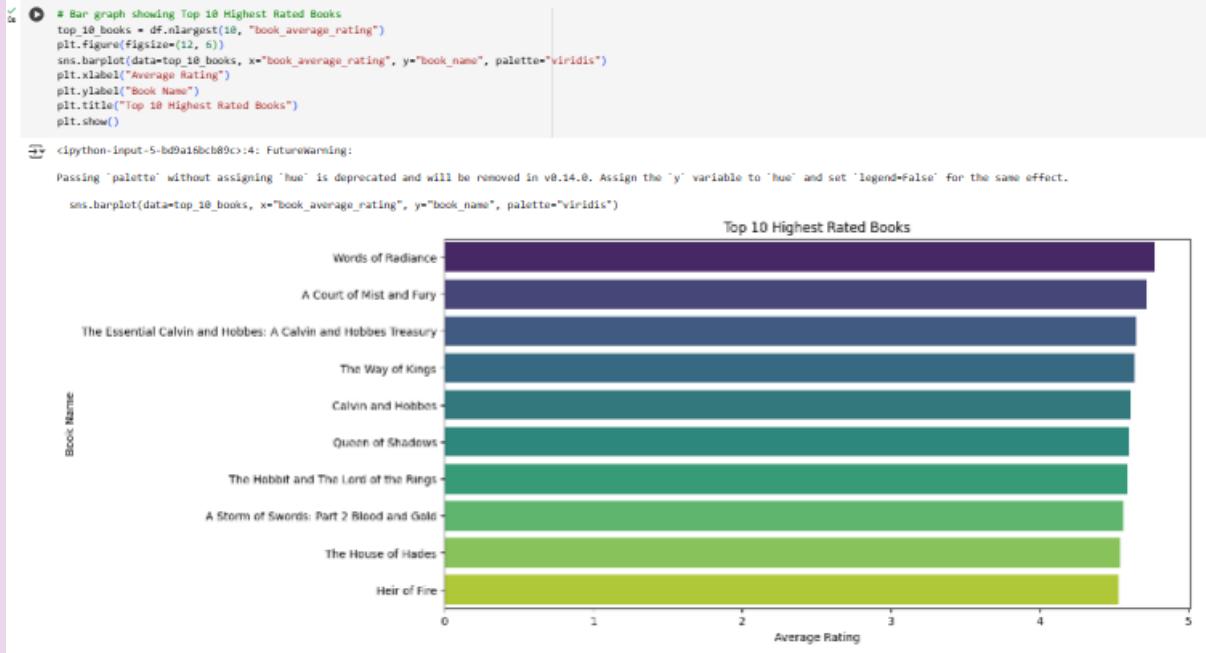
# Bar graph showing Sales Distribution Across Genres for Top
Publisher
top_publisher = df.groupby("publisher") ["gross_sales"].sum().idxmax()
top_publisher_data = df[df["publisher"] == top_publisher]
top_publisher_genre_sales =
top_publisher_data.groupby("genre") ["gross_sales"].sum()

plt.figure(figsize=(8, 6))
top_publisher_genre_sales.plot(kind="bar", color="skyblue")
plt.xlabel("Genre")
plt.ylabel("Total Sales")
plt.title(f"Sales Distribution Across Genres for {top_publisher}")
plt.xticks(rotation=45)
plt.show()

# Scatterplot showing Relationship Between Book Ratings and Sales
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x="book_average_rating", y="gross_sales",
alpha=0.6)
plt.xlabel("Book Average Rating")
plt.ylabel("Gross Sales")
plt.title("Relationship Between Book Ratings and Sales")
plt.show()

# Boxplot showing Relationship Between Book Ratings and Author
Ratings
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x="author_rating", y="book_average_rating",
palette="Set2")
plt.xlabel("Author Rating")
plt.ylabel("Book Average Rating")
plt.title("Relationship Between Book Ratings and Author Ratings")
plt.show()

```



```

[7] # Line graph showing Sales Distribution of Fiction Books Over Time
fiction_df = df[df["genre"] == "Fiction"]
plt.figure(figsize=(10, 5))
sns.lineplot(data=fiction_df, x="publishing_year", y="gross_sales", marker="o", ci=None)
plt.xlabel("Publishing Year")
plt.ylabel("Gross Sales")
plt.title("Sales Distribution of Fiction Books Over Time")
plt.grid()
plt.show()

<ipython-input-7-d41f24143bbd>:4: FutureWarning:
The 'ci' parameter is deprecated. Use 'errorbar=None' for the same effect.

sns.lineplot(data=fiction_df, x="publishing_year", y="gross_sales", marker="o", ci=None)

```



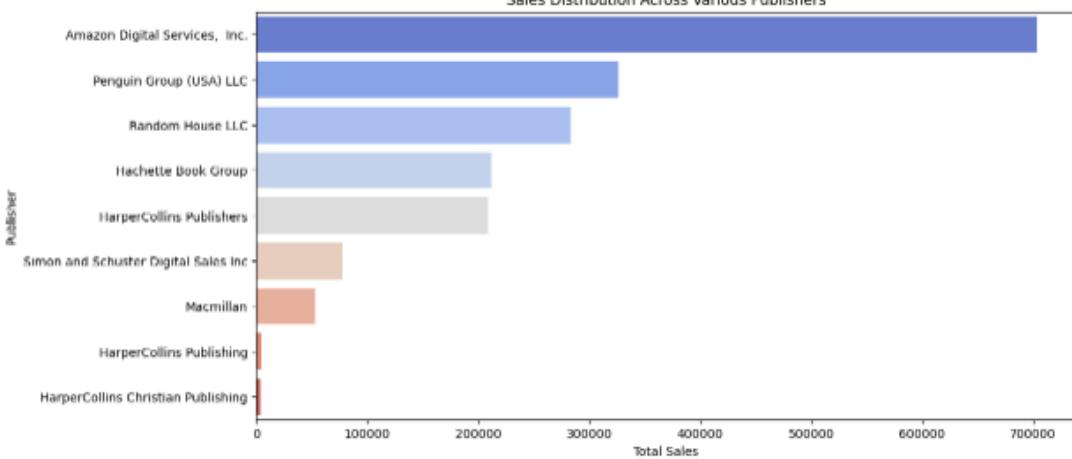
```

[8] # Bar graph showing Sales Distribution Across Various Publishers
plt.figure(figsize=(12, 6))
publisher_sales = df.groupby("publisher")["gross_sales"].sum().nlargest(10)
sns.barplot(x=publisher_sales.values, y=publisher_sales.index, palette="coolwarm")
plt.xlabel("Total Sales")
plt.ylabel("Publisher")
plt.title("Sales Distribution Across Various Publishers")
plt.show()

<ipython-input-8-1e6a1a6b77c3>:4: FutureWarning:
Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'y' variable to 'hue' and set 'legend=False' for the same effect.

sns.barplot(x=publisher_sales.values, y=publisher_sales.index, palette="coolwarm")

```

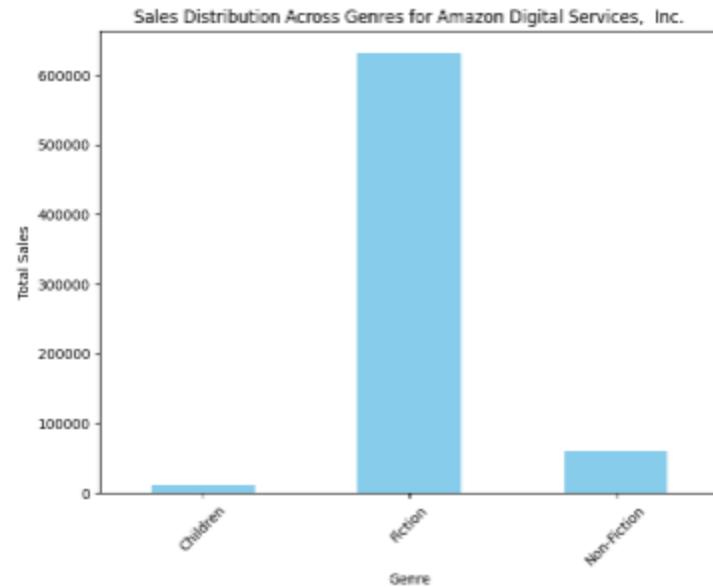


```

# Bar graph showing Sales Distribution Across Genres for Top Publisher
top_publisher = df.groupby("publisher")["gross_sales"].sum().idxmax()
top_publisher_data = df[df["publisher"] == top_publisher]
top_publisher_genre_sales = top_publisher_data.groupby("genre")["gross_sales"].sum()

plt.figure(figsize=(8, 6))
top_publisher_genre_sales.plot(kind="bar", color="skyblue")
plt.xlabel("Genre")
plt.ylabel("Total Sales")
plt.title(f"Sales Distribution Across Genres for {top_publisher}")
plt.xticks(rotation=45)
plt.show()

```



```

# Scatterplot showing Relationship Between Book Ratings and Sales
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x="book_average_rating", y="gross_sales", alpha=0.6)
plt.xlabel("Book Average Rating")
plt.ylabel("Gross Sales")
plt.title("Relationship Between Book Ratings and Sales")
plt.show()

```



```
<ipython-input-11-a3887ea9fee5>:3: FutureWarning:  
Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'x' variable to 'hue' and set 'legend=False' for the same effect.  
sns.boxplot(data=df, x="author_rating", y="book_average_rating", palette="Set2")
```

