Assignment # 5

[Find errors and complete the steps]

[Note]

Use the PyCharm software if google colab is not working properly

Automated File Management and Data Export System

1. **Moving Files**: This code uses glob to find all CSV files and shutil.move to move them to a backup folder.
2. **Export Function**: The export_data function takes a pandas DataFrame and exports it to a file in the specified format (CSV or JSON).
3. **Example DataFrame**: A simple DataFrame is created for demonstration and exported to both CSV and JSON formats.

```
import os
import glob
import shutil
import pandas as pd

# Move all CSV files to a backup folder
csv_files = glob.glob("*.csv")
for file in csv_files:
    shutil.move(file, "backup_folder/")
    print(f"Moved file: {file}")

# Automating Export
def export_data(df, filename, format):
    if format == "csv":
        df.to_csv(filename, index=False)
        print(f"Data exported to {filename} in CSV format.")
    elif format == "json":
```

```
        df.to_json(filename, orient="records")
        print(f"Data exported to {filename} in JSON format.")
    else:
        print("Unsupported format.")

# Example usage:
# Creating a sample dataframe
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'Los Angeles', 'Chicago']}

df = pd.DataFrame(data)

# Exporting to CSV
export_data(df, "output.csv", "csv")

# Exporting to JSON
export_data(df, "output.json", "json")
```

[Your Task]

1. Create a one folder with the name of “csv_files” and copy 3 to 4 csv files into it
2. Create a second folder with the name of ”backup_folder”
3. Install the dependencies
4. Run the above code
5. Show the output

Real-Time Stock Market Data Collection and Analysis Using Python and SQLite

- Fetches live stock data from Yahoo Finance.
- Stores the data in an SQLite database.

```
import yfinance as yf
import sqlite3
import pandas as pd
import time
```

```python
# Database setup
db_name = "stocks.db"
conn = sqlite3.connect(db_name)
cursor = conn.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS stock_data (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                symbol TEXT,
                timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
                open REAL,
                high REAL,
                low REAL,
                close REAL,
                volume INTEGER)''')
conn.commit()

# Function to fetch stock data
def fetch_stock_data(symbol):
    try:
        stock = yf.Ticker(symbol)
        data = stock.history(period="1d", interval="1m")

        if data.empty:
            print(f"No data found for {symbol}. Skipping...")
            return None  # Return None if no data is available

        latest = data.iloc[-1]  # Get the most recent price data
        return {
            "symbol": symbol,
            "open": latest["Open"],
            "high": latest["High"],
            "low": latest["Low"],
            "close": latest["Close"],
            "volume": latest["Volume"]
        }
    except Exception as e:
        print(f"Error fetching data for {symbol}: {e}")
        return None

# Function to store data in SQLite
def store_data(symbol):
    stock_data = fetch_stock_data(symbol)
    if stock_data:  # Only store if data is available
```

```
        cursor.execute('''INSERT INTO stock_data (symbol, open, high, low,
close, volume)
                          VALUES (?, ?, ?, ?, ?, ?)''',
                       (stock_data["symbol"], stock_data["open"],
stock_data["high"],
                        stock_data["low"], stock_data["close"],
stock_data["volume"]))
        conn.commit()
        print(f"Stored data for {symbol}")

# Function to analyze stock data
def analyze_stock(symbol):
    df = pd.read_sql_query("SELECT * FROM stock_data WHERE symbol=? ORDER BY
timestamp DESC LIMIT 100", conn, params=(symbol,))
    print(df)

# Example Usage
symbol = "AAPL"  # Apple stock
for _ in range(5):  # Fetch data 5 times with intervals
    store_data(symbol)
    time.sleep(60)  # Wait for 1 minute before fetching again

analyze_stock(symbol)

# Close database connection
conn.close()
```

[Your Task]

1. Install the dependencies
2. Replace "AAPL" with any stock symbol of your choice
3. Compile and run the code
4. Print and show the output

Augmented Reality Transformation – Perform linear algebra operations like scaling, rotation, and translation.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

URL = "http://books.toscrape.com/"
HEADERS = {"User-Agent": "Mozilla/5.0"}

def get_books(url):
    response = requests.get(url, headers=HEADERS)
    soup = BeautifulSoup(response.text, "html.parser")

    books = soup.find_all("article", class_="product_pod")
    book_list = []

    for book in books:
        title = book.h3.a["title"]
        price = book.find("p", class_="price_color").text
        stock = book.find("p", class_="instock availability").text.strip()

        book_list.append({"Title": title, "Price": price, "Availability":
stock})

    return book_list

books_data = get_books(URL)
df = pd.DataFrame(books_data)
df.to_csv("books.csv", index=False)
print("Data saved to books.csv")
```

[Your Task]

```
1. Install the dependencies
2. Replace "given website link" with any HTML website link
3. Compile and run the code
4. Print and show the output
```