"Real-Time Synthetic Financial Data Generation for Analytics and Model Testing"

Project Report

Introduction:

In data-driven world, businesses and researchers often require vast amounts of high-quality data to power analytics, validate pipelines, and train machine learning models. However, obtaining such data in real-time can be challenging due to privacy concerns and data scarcity. This project introduces a Python-based solution to generate synthetic financial datasets in real-time, enabling organizations to simulate real-world scenarios for analysis and experimentation.

Business Problem:

Companies often face a lack of real-world financial data for training and validating analytics models due to data sensitivity and privacy restrictions. Staticdatasets do not reflect the dynamic nature of real-world scenarios. Building machine learning models or testing pipelines without sufficient data can lead to inaccurate predictions and performance issues.

Objectives:

Develop a Python-based framework for real-time synthetic financial data generation. Simulate realistic financial metrics, including revenue, profit, and balance sheet elements. Save generated datasets in a structured format (CSV) for seamless integration into analytics workflows. Automate data generation with a configurable time interval. Provide an easily extendable and reusable solution for various industries and applications.

Required Tools:

Python Libraries: numpy: Numerical computations for generating random data. pandas: Data handling and manipulation. Faker: Simulate random values where necessary. os: Directory and file management. time: Control execution intervals.

IDE or Environment: Jupyter Notebook, VS Code, or any Python-compatible IDE. Operating System: Any platform that supports Python.

Methodology: Data Simulation Logic:

Simulate random financial metrics like revenue, expenses, and assets using numpy's random functions. Calculate derived metrics such as gross profit and net income from the base data.

Real-Time Data Generation:

Generate batches of synthetic financial data at a specified time interval (e.g., every 10 seconds). Use a Python while loop with the time.sleep() function to control execution frequency.

Data Storage:

Save each batch of data to a timestamped CSV file in a designated directory.
Implementation:

Implement the SyntheticDataGenerator class to encapsulate data generation logic. Define the main() function to handle the real-time execution loop.

Execution:

Execute the program and allow it to run indefinitely, generating and storing data until stopped by the user.

Assumptions: Financial metrics such as revenue and expenses follow a uniform random distribution for simplicity. The generated synthetic data does not represent any real-world company or industry. Users have sufficient disk space to store the generated CSV files.

Ethical Considerations: Ensure the synthetic data does not inadvertently replicate sensitive real-world financial data. Clearly label the generated datasets as synthetic to avoid misuse. Avoid generating data with personally identifiable information (PII) unless necessary and anonymized.

Achieved Output: Real-Time Data Generation:

The program generates financial data every 10 seconds with metrics such as revenue, gross profit, and net income. Automated Storage:

Saves each batch of data to a timestamped CSV file for easy retrieval and analysis. Extensible Framework:

The program provides a modular design, allowing users to customize metrics and batch sizes as needed.

Conclusion: The real-time synthetic financial data generator provides a practical solution for organizations requiring realistic datasets for testing and analytics. By automating data generation and storage, the program simulates dynamic financial scenarios, aiding in model training, validation, and pipeline testing. The modular design ensures adaptability for various use cases.

Recommendation: Enhance Customizability: Allow users to specify data distributions and ranges for each metric to align with domain-specific requirements. Integrate Visualization: Add real-time visualization capabilities to monitor trends in the generated data. Incorporate Machine Learning: Use the generated data to train predictive models and validate their performance in real-time.

Step Forward: Extend the framework to support multi-threaded or distributed execution for faster data generation. Implement a web-based interface to control data generation parameters dynamically. Explore integrations with real-time analytics platforms such as Power BI or Tableau for live monitoring.

Python Script

In [ ]:
```
pip install faker
```

In [3]:
```
# actual synthetic data - realtime

import numpy as np
import pandas as pd
from faker import Faker
import os
import time

class SyntheticDataGenerator:
    def __init__(self, num_records):
        self.num_records = num_records
        self.fake = Faker()

    def generate_raw_data(self):
        # Generate base components for calculations
        revenue = np.random.uniform(7e7, 8e7, self.num_records)
        cogs = np.random.uniform(3.5e7, 4.5e7, self.num_records)  # Cost of Goods S
        operating_expenses = np.random.uniform(1e7, 2e7, self.num_records)
        interest_expenses = np.random.uniform(1e5, 2e5, self.num_records)
        taxes = np.random.uniform(5e5, 1e6, self.num_records)
        inventory = np.random.uniform(6e6, 1e7, self.num_records)
        accounts_receivable = np.random.uniform(1e6, 1e7, self.num_records)
        cash_equivalents = np.random.uniform(1e6, 2e7, self.num_records)
        current_liabilities = np.random.uniform(1e7, 5e7, self.num_records)
        current_assets = cash_equivalents + inventory + np.random.uniform(1e6, 1e7,
        total_assets = current_assets + np.random.uniform(1e7, 5e7, self.num_record
        total_liabilities = np.random.uniform(1e7, 7e7, self.num_records)
        shareholders_equity = total_assets - total_liabilities

        # Derived metrics
        gross_profit = revenue - cogs
        operating_income = gross_profit - operating_expenses
        net_income = operating_income - interest_expenses - taxes

        # Compile raw data and calculations into a dictionary
        raw_data = {
            "Revenue": revenue,
            "Cost of Goods Sold (COGS)": cogs,
            "Operating Expenses": operating_expenses,
            "Interest Expenses": interest_expenses,
            "Taxes": taxes,
            "Inventory": inventory,
            "Accounts Receivable": accounts_receivable,
            "Cash Equivalents": cash_equivalents,
            "Current Liabilities": current_liabilities,
            "Current Assets": current_assets,
            "Total Assets": total_assets,
            "Total Liabilities": total_liabilities,
            "Shareholders' Equity": shareholders_equity,
            "Gross Profit": gross_profit,
            "Operating Income": operating_income,
            "Net Income": net_income,
        }

        return pd.DataFrame(raw_data)

    def save_to_csv(self, df, filename):
        df.to_csv(filename, index=False)
        print(f"Dataset saved to {filename}")

# Main program for real-time execution
def main():
    num_records = 5  # Set the number of records per batch
    output_dir = "realtime_synthetic_data"
    os.makedirs(output_dir, exist_ok=True)
    interval = 10  # Interval in seconds

    generator = SyntheticDataGenerator(num_records)

    print("Starting real-time synthetic data generation...")

    try:
        while True:
            # Generate raw data
            raw_data = generator.generate_raw_data()

            # Display a preview
            print(raw_data.head())

            # Save to a timestamped CSV file
            timestamp = pd.Timestamp.now().strftime("%Y%m%d_%H%M%S")
            filename = os.path.join(output_dir, f"financial_data_{timestamp}.csv")
            generator.save_to_csv(raw_data, filename)

            # Wait for the next batch
            print(f"Waiting {interval} seconds for the next batch...")
            time.sleep(interval)
    except KeyboardInterrupt:
        print("Real-time synthetic data generation stopped by user.")

if __name__ == "__main__":
    main()
```

```
Starting real-time synthetic data generation...
        Revenue  Cost of Goods Sold (COGS)  Operating Expenses  \
0  7.561577e+07               3.793104e+07        1.307109e+07
1  7.580648e+07               3.988506e+07        1.012989e+07
2  7.391017e+07               3.759752e+07        1.021109e+07
3  7.848843e+07               3.929551e+07        1.194127e+07
4  7.035596e+07               3.930720e+07        1.838231e+07

   Interest Expenses          Taxes     Inventory  Accounts Receivable  \
0      197102.903836  975766.192361  9.468619e+06         3.269570e+06
1      186732.418429  595121.014273  7.437361e+06         7.057148e+06
2      136256.375662  935319.828899  6.473158e+06         8.579237e+06
3      132359.558281  748945.058938  9.967327e+06         7.003279e+06
4      166156.040316  999579.715879  6.074137e+06         8.240948e+06

   Cash Equivalents  Current Liabilities  Current Assets  Total Assets  \
0      5.326819e+06         4.208496e+07    2.437581e+07  4.679138e+07
1      1.734859e+07         4.253221e+07    3.151418e+07  4.991367e+07
2      1.013733e+07         2.311162e+07    2.545250e+07  6.439539e+07
3      9.519797e+06         4.972360e+07    2.862479e+07  7.857642e+07
4      1.464380e+07         4.283029e+07    3.037097e+07  5.538435e+07

   Total Liabilities  Shareholders' Equity  Gross Profit  Operating Income  \
0       4.395679e+07          2.834592e+06  3.768473e+07      2.461364e+07
1       3.734537e+07          1.256829e+07  3.592142e+07      2.579153e+07
2       6.477214e+07         -3.767564e+05  3.631266e+07      2.610157e+07
3       3.442553e+07          4.415089e+07  3.919293e+07      2.725166e+07
4       5.728191e+07         -1.897557e+06  3.104876e+07      1.266645e+07

     Net Income
0  2.344077e+07
1  2.500967e+07
2  2.502999e+07
3  2.637035e+07
4  1.150072e+07
Dataset saved to realtime_synthetic_data\financial_data_20250119_034611.csv
Waiting 10 seconds for the next batch...
Real-time synthetic data generation stopped by user.
```

Explanation of the Script

The provided script is a Python program designed to generate synthetic financial data in real-time. The script:

Utilizes the numpy and pandas libraries for data manipulation and numerical operations. Incorporates Faker to simulate random values where applicable. Simulates real-world financial metrics, including revenue, cost of goods sold (COGS), gross profit, operating income, net income, and balance sheet elements like assets and liabilities. Continuously generates a batch of financial records at a specified interval (e.g., 10 seconds). Saves each batch to a timestamped CSV file for later use. Allows for real-time execution and can be terminated using Ctrl+C. The script achieves real-time synthetic data generation and storage, making it useful for data simulation, testing machine learning models, or validating data pipelines.

```
In [ ]:
```