**Synthetic Data Factory**

Automated, Statistically-Driven Synthetic Data Generation

and Validation Platform

**Project Overview:**

The Synthetic Data Factory (SDF) is an intelligent web application that enables users to generate realistic, privacy-safe, and statistically validated synthetic datasets with ease. It allows developers, data scientists, testers, and researchers to create custom datasets for analytics, testing, and machine learning without relying on sensitive real-world data.

This system bridges the gap between data accessibility and privacy by learning the structure, distribution, and relationships within uploaded datasets or defined schemas, and then reproducing that same statistical behaviour in newly generated synthetic data.

By integrating AI-based data pattern detection, Faker-driven content generation, and statistical validation techniques, SDF ensures every dataset is both realistic and unique, making it ideal for simulation, system testing, research, and training environments.

**Scenario 1: Data Science and Machine Learning Development**

Data scientists and machine learning engineers often struggle to find large, high-quality datasets that are both diverse and privacy-compliant. The Synthetic Data Factory (SDF) addresses this challenge by generating realistic synthetic data that mirrors the statistical patterns of real-world datasets without exposing sensitive information. For instance, a financial technology startup developing a loan prediction model can use SDF to create synthetic data representing customer profiles, income ranges, credit scores, and loan histories. Instead of relying on confidential client data, they can upload a small sample dataset or define the required schema, and SDF will generate thousands of synthetic records maintaining the same relationships and distributions.

With SDF’s built-in visualization and validation tools, teams can ensure that the generated data accurately reflects the original structure. This allows safe, large-scale model training and testing while remaining compliant with data privacy standards. Through its integration of Streamlit, Faker, and statistical validation frameworks, the Synthetic Data Factory not only accelerates data preparation but also empowers organizations to innovate responsibly. Ultimately, SDF transforms data scarcity into data abundance driving progress in AI, analytics, and decision intelligence securely and efficiently.

**Scenario 2: Application Development and Database Testing**

In software development and testing, teams often require large, diverse, and realistic datasets to validate performance, database operations, and system integration. However, using production data introduces serious privacy and compliance challenges. The Synthetic Data Factory (SDF) overcomes this by generating synthetic datasets that replicate the structure and statistical behaviour of real data without revealing any sensitive information.

For example, a banking enterprise can use SDF to generate synthetic customer profiles, account details, and transaction histories to test APIs, data pipelines, and application performance under production-like conditions. Through its intelligent combination of Streamlit, Faker, and validation frameworks, SDF enables secure, scalable, and automated data population for development environments.

Its integrated MySQL connectivity module supports direct schema detection and synthetic data insertion, simplifying test data management. By ensuring realistic simulation, privacy compliance, and data consistency, SDF helps teams accelerate release cycles, enhance reliability, and minimize time and cost in test data generation.

**Architecture Overview:**

The Synthetic Data Factory is a modular, intelligent data generation platform designed to create realistic, privacy-safe, and statistically balanced synthetic datasets for AI, analytics, and application testing. Built using Streamlit as the front-end framework, SDF integrates a seamless, interactive interface with a robust backend powered by Python, Faker, NumPy, Pandas, and PyMySQL for advanced data handling and simulation.

The system follows a multi-layered architecture consisting of three core backend modules: UniversalDataGenerator, DatabaseHandler, and DataValidator, each serving a distinct role. The *UniversalDataGenerator* module generates synthetic datasets either from uploaded CSV files or user-defined column schemas. The *DatabaseHandler* enables direct integration with MySQL databases, allowing users to generate and insert synthetic data directly based on table structures. The *DataValidator* module ensures statistical fidelity by comparing real and synthetic datasets using distribution metrics and visual validations.

The Streamlit-based front-end ties these components together through an intuitive tabbed interface featuring CSV extension, custom data creation, and database connectivity sections. With integrated validation, progress tracking, and responsive visual design, SDF provides users with a complete synthetic data ecosystem enabling them to generate, validate, and visualize data securely and efficiently for diverse enterprise and research applications.

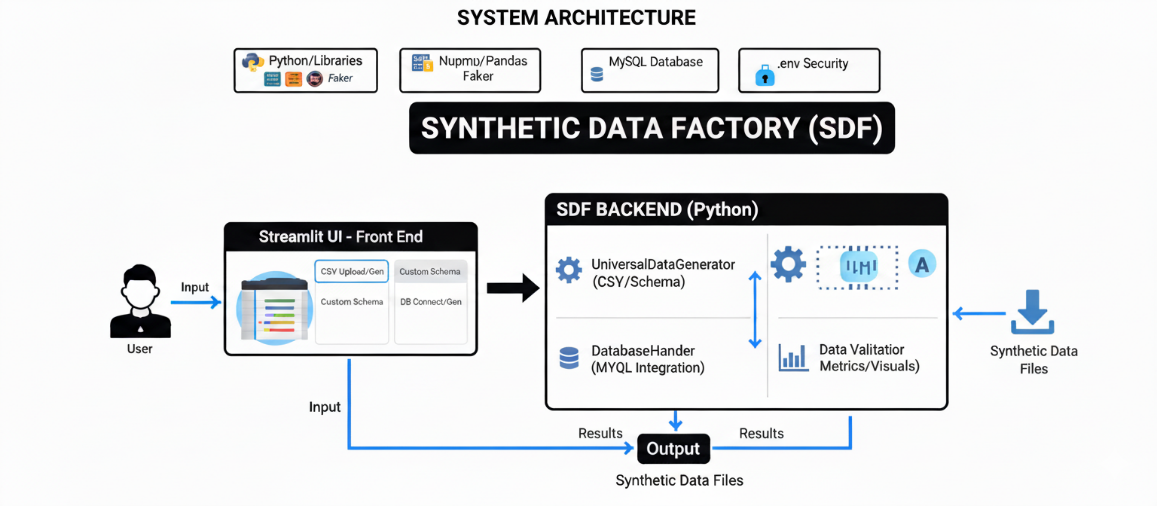


Figure 1: System architecture diagram

**Core Technologies:**

* **Streamlit (Frontend Framework):** Provides an intuitive and interactive web interface for dataset generation, schema customization, and visualization. It handles real-time user inputs, displays synthetic data previews, and integrates dynamic components like tabs and progress indicators.
* **Faker (Synthetic Data Engine):** Powers the data generation process by creating realistic yet artificial values for various data types such as names, addresses, transactions, and demographic details. It supports both predefined templates and custom schema definitions.
* **NumPy & Pandas (Data Processing Core):** Enable efficient data manipulation, statistical computation, and tabular structure handling. These libraries ensure scalable and optimized performance during large-scale synthetic dataset creation and validation.
* **PyMySQL (Database Integration):** Allows direct connection with MySQL databases for real-time schema extraction and automatic insertion of generated synthetic data, enabling seamless backend testing and database population.
* **Matplotlib & Seaborn (Visualization Framework):** Used to compare and validate real vs. synthetic datasets through visual metrics such as histograms, distributions, and correlation heatmaps, ensuring statistical integrity and data balance.
* **Data Validation & Comparison Module:** Ensures that synthetic datasets maintain consistency, diversity, and similarity to real datasets by applying distribution checks, statistical summaries, and visualization-based validation.
* **Python Backend (Core Logic Layer):** Manages communication between all modules data generation, validation, and database operations that’s ensuring smooth execution, modular scalability, and high-performance processing.

**Component-Wise Architecture:**

|  |  |
| --- | --- |
| Component | Description |
| **User Interface (Streamlit)** | Interactive dashboard for CSV upload, schema creation, and database connection with real-time data display. |
| **Universal Data Generator** | Creates realistic synthetic datasets using Faker, NumPy, and Pandas based on sample or custom schema. |
| **Database Handler (MySQL)** | Connects to MySQL, detects table schemas, and inserts generated data directly for testing and simulation. |
| **Data Validator & Visualizer** | Compares real vs. synthetic data through statistical checks and visualizations using Matplotlib and Seaborn. |
| **Custom Schema Builder** | Let users define columns, types, and record limits for domain-specific synthetic data generation. |
| **Visualization & Reporting** | Generates charts and summaries for quick validation of data quality and distribution patterns. |
| **Backend Logic Controller** | Manages coordination between data generation, validation, and visualization modules. |
| **Help & Documentation** | Provides usage guidance and workflow support within the Streamlit interface. |

**Pre-requisites:**

1. **Python Environment Setup**: The Synthetic Data Factory (SDF) is developed using Python 3.9+, leveraging its extensive data handling and visualization libraries. Create and activate a dedicated virtual environment (e.g., synthetic\_data\_factory) to manage dependencies and maintain a clean setup.

Official Documentation: <https://www.python.org/downloads/>

1. **Streamlit Installation and Configuration**: Streamlit provides the front-end interface for data generation, visualization, and validation. Install Streamlit to build and run the interactive web dashboard.

Docs: <https://docs.streamlit.io/library/get-started/installation>

Tutorial: <https://www.youtube.com/watch?v=JwSS70SZdyM>

1. **Library Installation and Core Dependencies**: SDF relies on several Python libraries for data generation, analysis, and visualization. Install all dependencies listed in requirements.txt.

Key Libraries:

streamlit – Interactive UI and data display

faker – Synthetic data generation

pandas – Data manipulation and structure handling

numpy – Numerical and statistical computation

matplotlib & seaborn – Data visualization and validation plots

pymysql – MySQL database connectivity and data insertion

1. **Database Setup**: For projects requiring live database testing, install and configure MySQL Server. SDF connects via PyMySQL, enabling table schema reading and direct data insertion.

MySQL Installation: <https://dev.mysql.com/downloads/>

PyMySQL Docs: <https://pypi.org/project/PyMySQL/>

1. **Development Environment**:

Recommended IDEs for efficient development and debugging

Visual Studio Code: <https://code.visualstudio.com/>

PyCharm (Community Edition): <https://www.jetbrains.com/pycharm/download/>

1. **Optional Learning Resources**:

Enhance understanding of the tools and concepts used

Streamlit Components Gallery: <https://streamlit.io/components>

Faker Library Guide: <https://faker.readthedocs.io/>

**Project Flow:**

**1. Environment Setup and Dependency Configuration**

* **Activity 1.1:** Create and activate a virtual environment and install required dependencies.
* **Activity 1.2:** Organize project structure and add UI styling with styles.css.
* **Activity 1.3:** Initialize Streamlit app and validate integration between modules.

**2. Core Logic Development (Synthetic Data Generation Engine)**

* **Activity 2.1:** Implement UniversalDataGenerator for CSV and schema-based synthetic data.
* **Activity 2.2:** Develop validation module to analyze and compare real vs synthetic data.
* **Activity 2.3:** Integrate MySQL database for schema retrieval and synthetic data insertion.

**3. Streamlit UI Implementation and User Interaction**

* **Activity 3.1:** Design multi-tab layout with navigation and professional styling.
* **Activity 3.2:** Configure input system for columns, rows, and database details with session state.
* **Activity 3.3:** Display synthetic data and visualizations with computed metrics.

**4. Testing, Optimization, and Deployment**

* **Activity 4.1:** Test UI components and verify accurate data generation.
* **Activity 4.2:** Validate database integration and large dataset performance.
* **Activity 4.3:** Prepare deployment and perform end-to-end validation for stability and performance.

**MILESTONE 1: Environment Setup and Dependency Configuration**

This foundational milestone establishes the technical environment required for building and deploying the Synthetic Data Factory (SDF). It ensures that all dependencies, frameworks, and integrations are configured correctly for seamless execution of synthetic data generation, visualization, and validation workflows.

**Activity 1.1: Python Environment and Dependency Installation**

* Create and activate a dedicated virtual environment for SDF to maintain dependency isolation.

**A screen shot of a computer

AI-generated content may be incorrect.**

Figure 2: Creating & Activating Environment

* Install all required dependencies listed in requirements.txt using:

**A screen shot of a computer

AI-generated content may be incorrect.**

Figure 3: Installing requirements

* Verify compatibility across Python 3.9+ and confirm successful installation of each module.
* Test library imports and ensure data-related packages (NumPy, Pandas) and visualization tools (Matplotlib, Seaborn) work correctly.

**Activity 1.2: Project Structure Initialization**

* Organize project modules (streamlit\_app.py, data\_generator.py, database\_handler.py, validation.py) into a structured hierarchy for clarity and maintainability.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4: Folder Structure

* Add styles.css to define Streamlit UI design, theme consistency, and responsive layout behaviour.
* Verify file paths and imports to ensure modular independence and scalability.

**Activity 1.3: Streamlit Application Initialization**

* Set up streamlit\_app.py as the central entry point of the system using:

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 5: Streamlit Configuration

* Ensure consistent theming, responsive layout, and component rendering across devices.
* Validate integration between all modules, confirming smooth data flow between UI and backend functions.

**MILESTONE 2: Core Logic Development (Synthetic Data Generation Engine)**

This milestone builds the analytical and generation backbone of SDF. The goal is to automate realistic, statistically valid data generation using the UniversalDataGenerator and connect it with analysis, validation, and storage pipelines.

**Activity 2.1: Universal Data Generator Implementation (data\_generator.py)**

* Develop the UniversalDataGenerator class to handle both CSV-based and custom schema-based synthetic data generation.

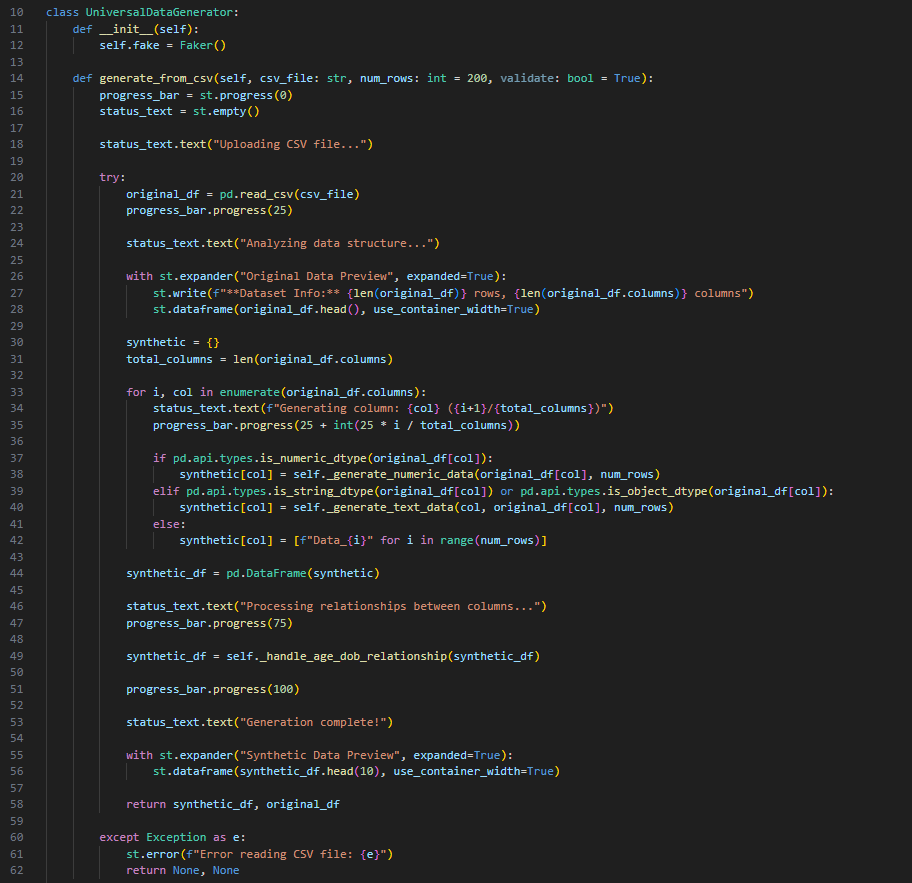


Figure 6: Dataset Generator from CSV

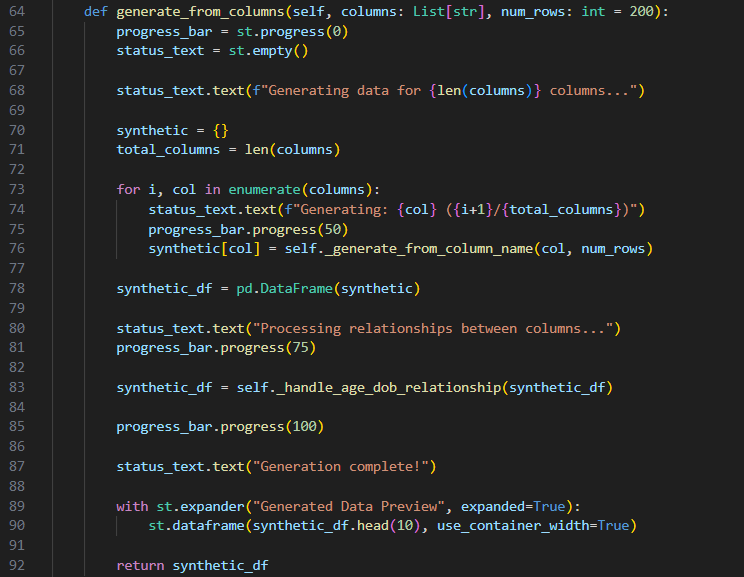


Figure 7: Data Generator from Column Names

* Implement configurable parameters for number of rows, data types, and randomization logic.

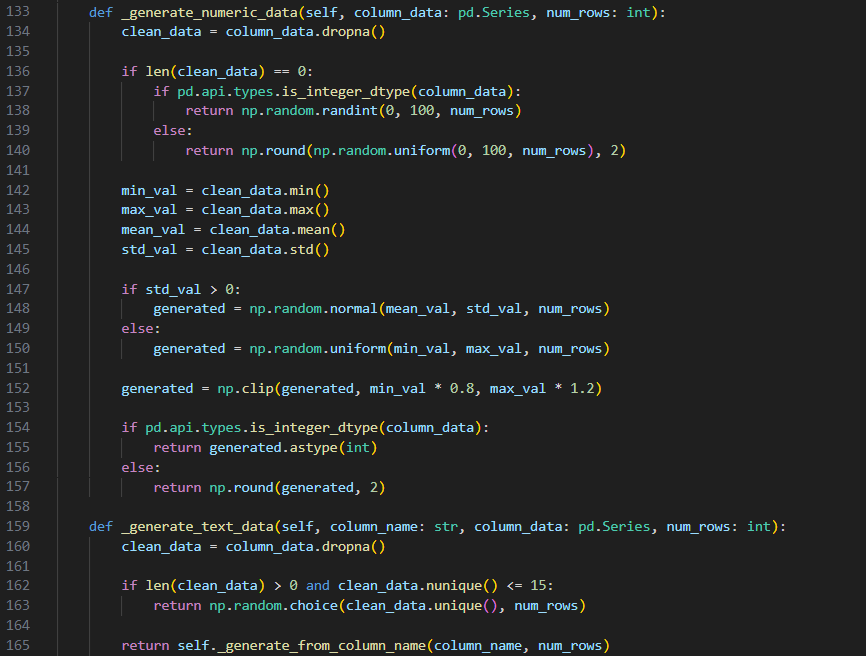


Figure 8: Numeric data processing

* Integrate Faker to simulate text, numeric, categorical, and temporal data types.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 9: Faker Mapping

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 10: Faker Mapping

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 11: Faker Mapping

**Activity 2.2: Data Analysis and Validation Module (validation.py)**

* Develop analytical functions to calculate core statistical metrics such as mean, median, mode, variance, and correlation.

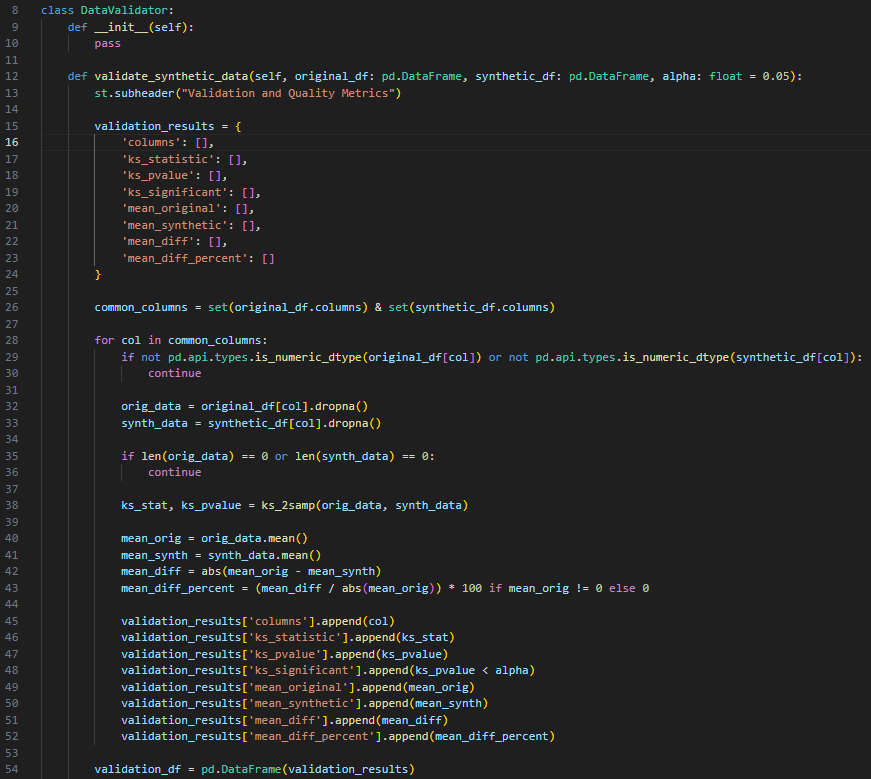


Figure 12: Validation of dataset

* Implement comparison logic for real vs synthetic data distributions to validate fidelity.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 13: Comparing Logic

* Structure results using **Pandas DataFrames** for compatibility with the visualization layer.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 14: Plotting Function

* Integrate automated summary reports highlighting statistical differences and similarities.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 15: Score Calculation

**Activity 2.3: Database Integration (database\_handler.py)**

* Implement secure and optimized **MySQL** database integration using **PyMySQL**.
* Allow users to:
  + Retrieve existing table schemas automatically.
  + Generate data conforming to those schemas.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 16: SQL operation function

* Insert synthetic records directly into connected databases.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 17: Creating and Inserting the dummy data to database

* Include robust error handling for connection failures, authentication issues, and SQL exceptions.

A computer screen shot of a program code

AI-generated content may be incorrect.

Figure 18: Exception Handling

**MILESTONE 3: Streamlit UI Implementation and User Interaction**

This milestone focuses on building an **interactive, intuitive user interface** that connects backend intelligence with real-time visualization. The UI allows users to configure schemas, generate data, validate results, and interact with synthetic datasets dynamically.

**Activity 3.1: Layout Design and Navigation**

* Develop a multi-tabbed interface for:

CSV Extension

Dummy Data Creation

SQL Operations

**A screen shot of a computer

AI-generated content may be incorrect.**

Figure 19: Navigation buttons

* Used columns, custom Markdown, and HTML formatting for adding custom style and structure.

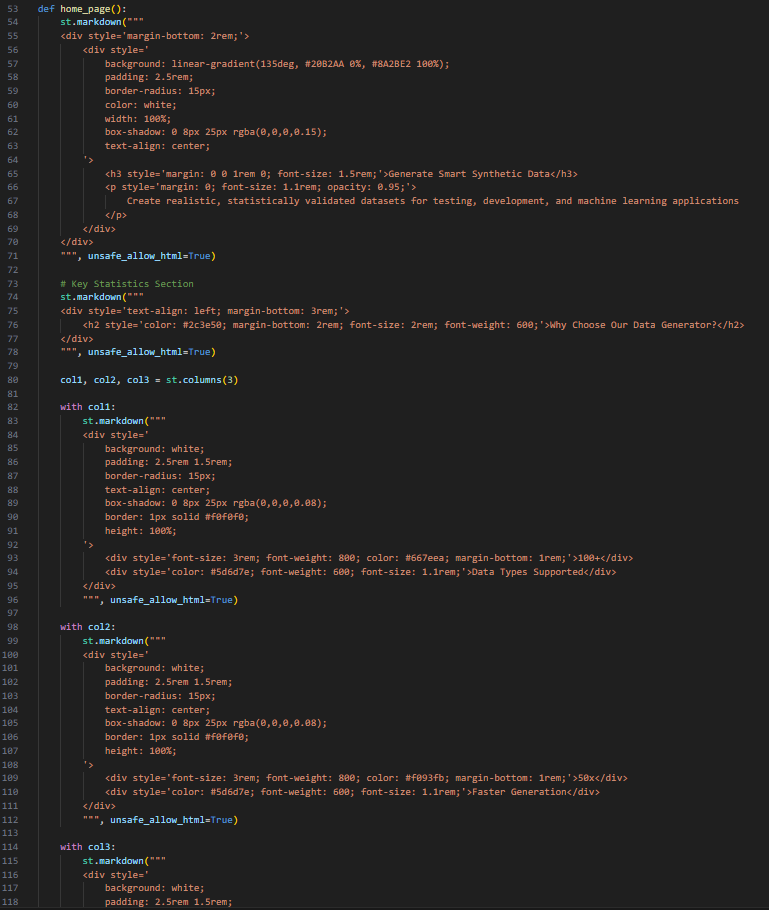


Figure 20: Column separation output

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 21: Markdowns

* Integrate visual separators and typography for a professional, data-centric dashboard experience.

**Activity 3.2: Input System and Schema Configuration**

Added text area for user input including:

* Column names and types format for CSV extension page

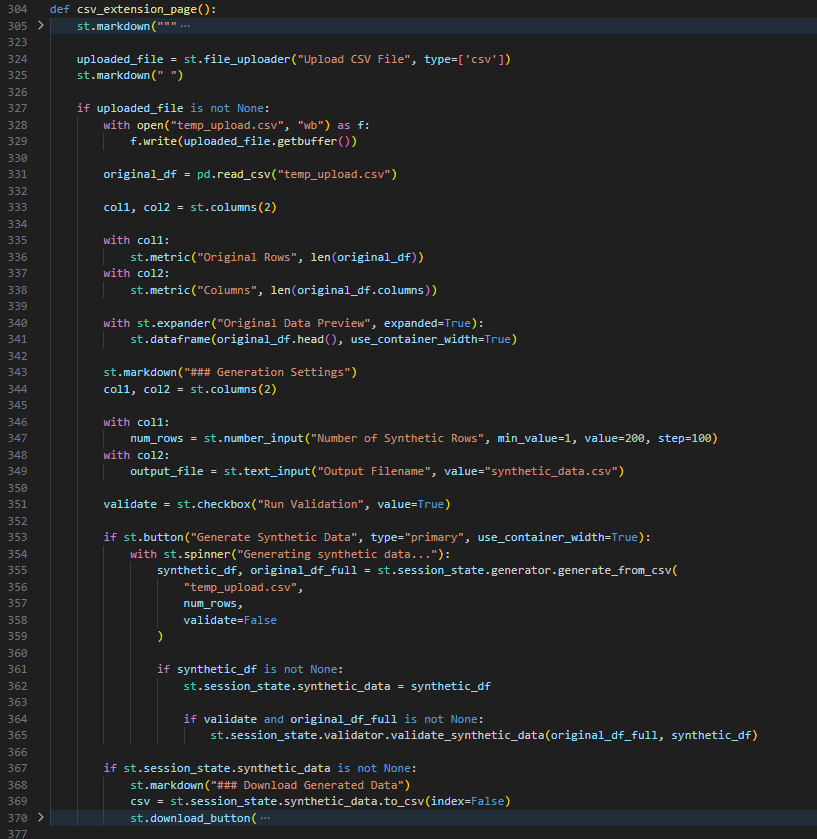


Figure 22: Input to CSV extension

* Row column name for generation for Dummy data creation

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 23: Input to the Dummy data creation

* Database details (host, user, password, table)

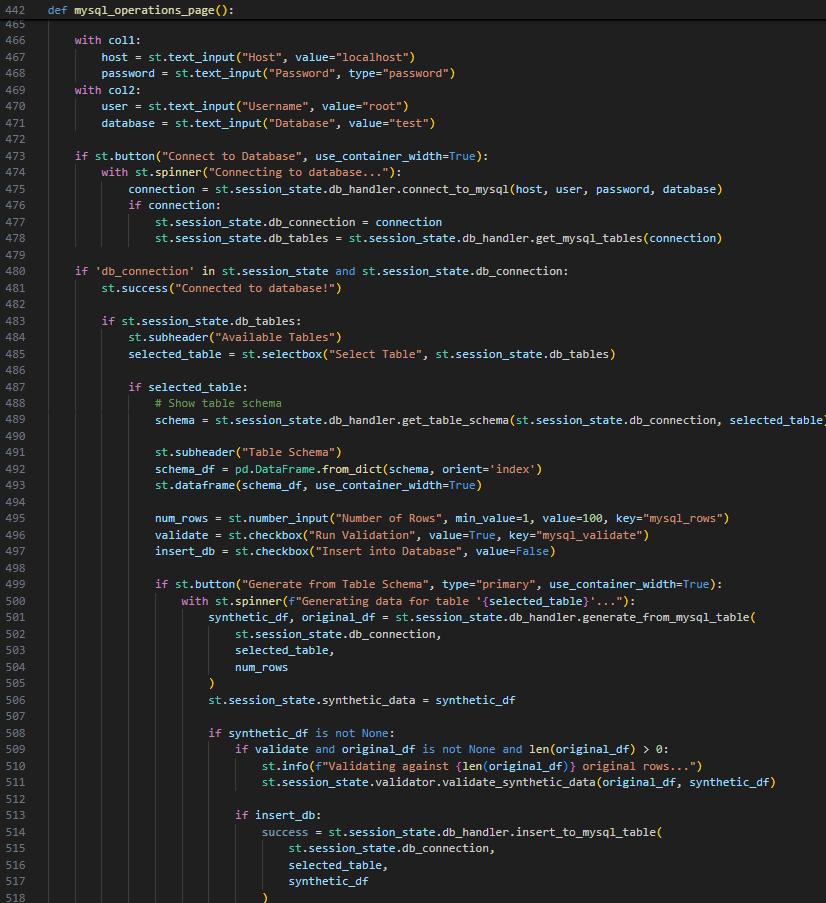


Figure 24: Database detail input section

* Integrate progress indicators and success messages using Streamlit’s feedback components (st.spinner, st.success, etc.).
* Persist inputs via session state for consistent user experience.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 25: Session State

**Activity 3.3: Data Visualization and Result Display**

* Display generated synthetic data dynamically using **Pandas DataFrames** in Streamlit tables.

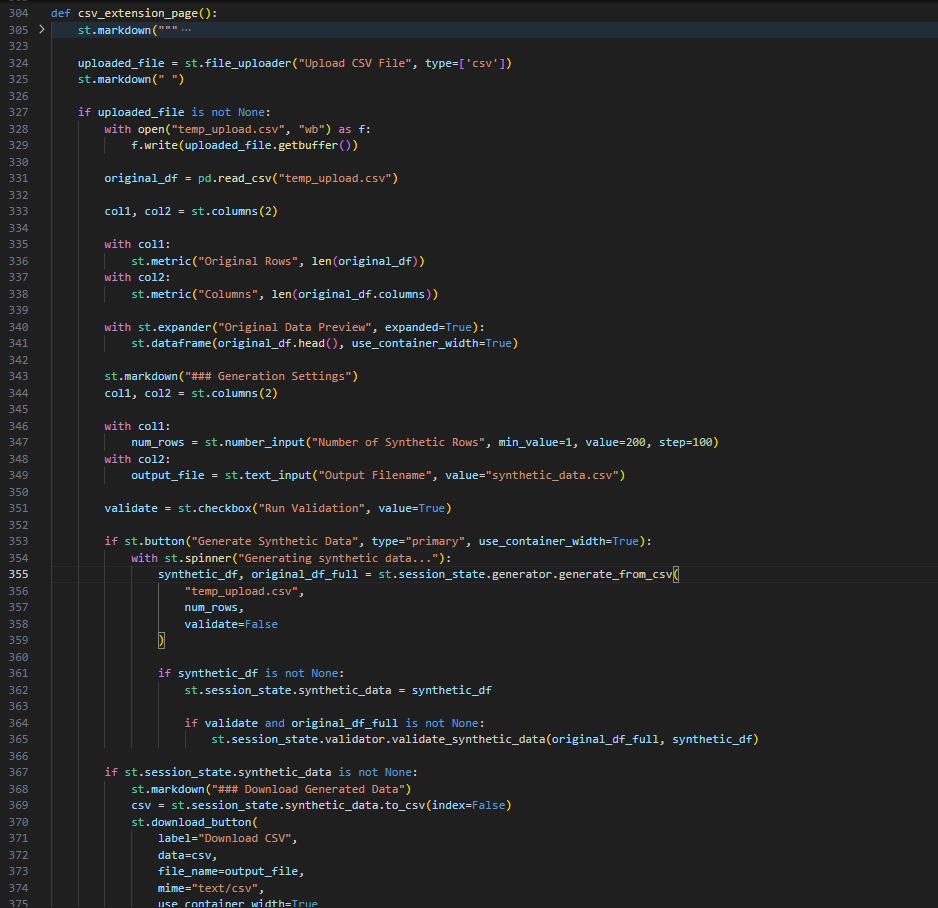


Figure 26: Streamlit Output section for CSV extension

* Integrate Matplotlib and Plotly for generating statistical charts histograms with correlation of the inputted dataset.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 27: Plotting the hist graph

* Show computed metrics like mean, standard deviation, and column count in summary cards.

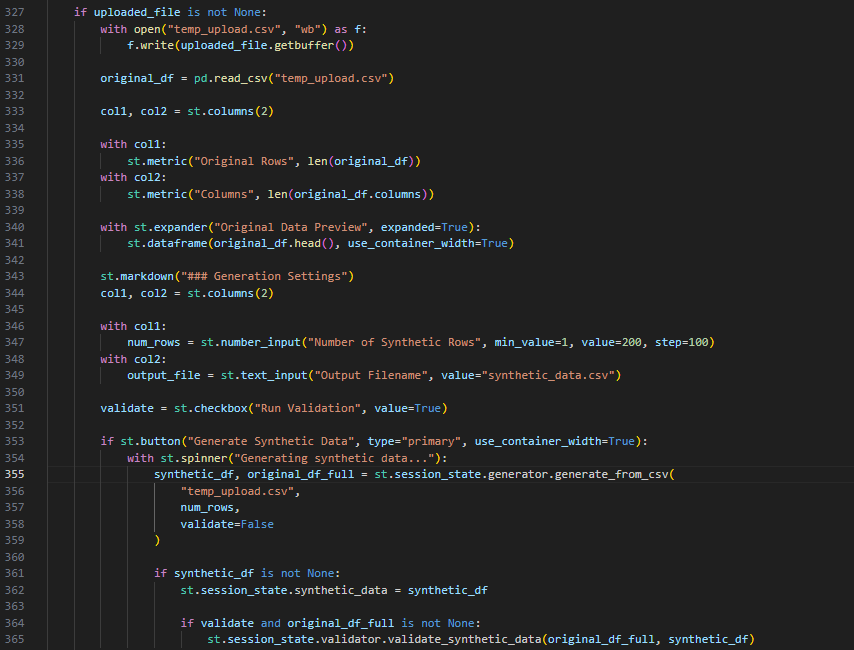


Figure 28: Evaluation Matrix

**MILESTONE 4: Testing, Optimization, and Final Deployment**

This final milestone ensures reliability, speed, and stability of the Synthetic Data Factory through rigorous front-end testing, optimization, and deployment preparation.

**Activity 4.1: Functional Testing**

* Test all UI components including buttons, input forms, chart rendering, and data preview tables.

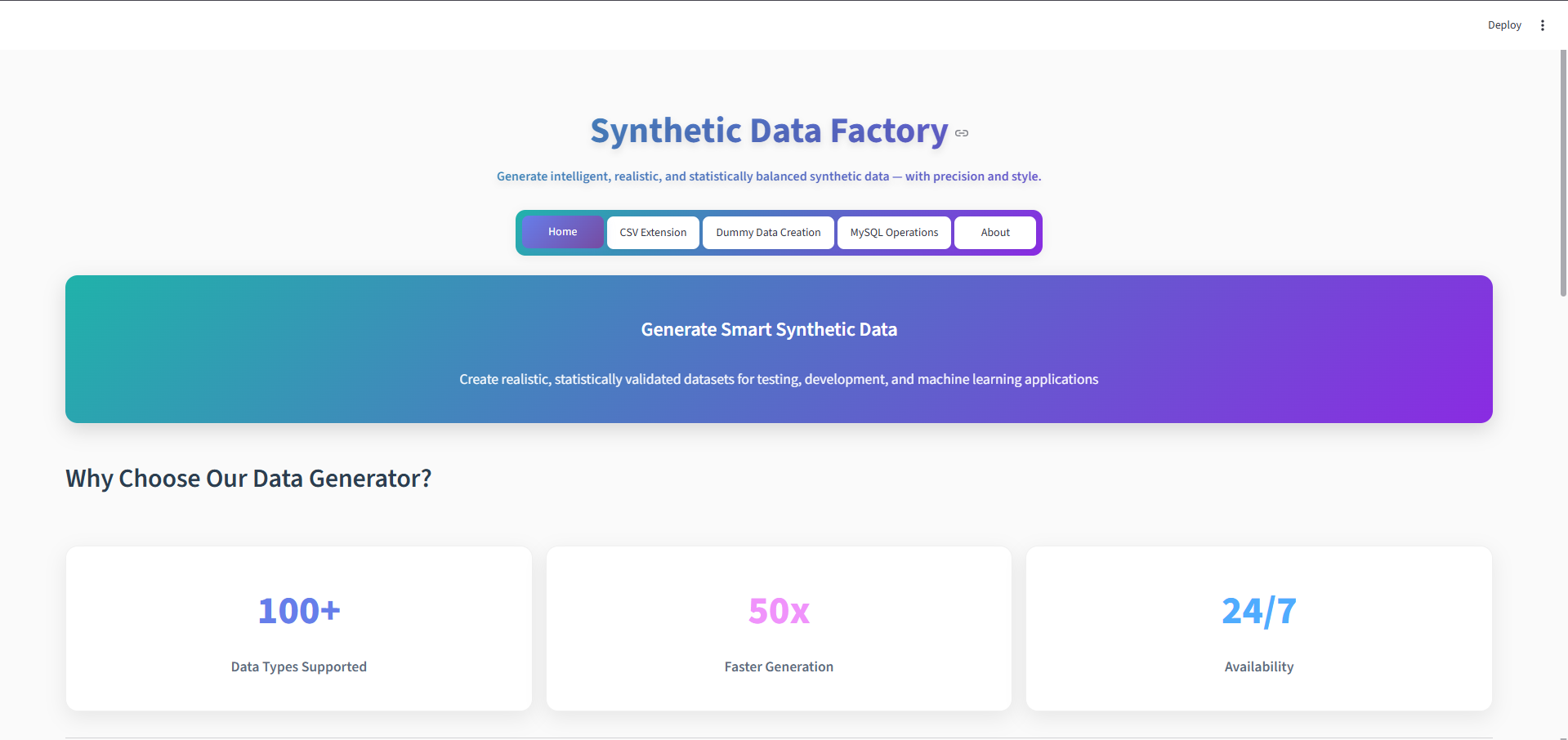


Figure 29: Home Page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 30: CSV Extention Page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 31: Dummy Data Creation Page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 32: SQL Operation Page

* Ensure data generation is accurate and consistent for both CSV and database modes.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 33: Data Generation from CSV Extension

A screenshot of a computer

AI-generated content may be incorrect.

Figure 34: Real VS Synthetic Data

A screenshot of a computer

AI-generated content may be incorrect.

Figure 35: Data Creation from Dummy Data creation Model

* Validate interactive chart responsiveness across different browsers and resolutions.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 36: Chart Visualization

**Activity 4.2: Integration and Performance Validation**

* Test MySQL connection handling and data insertion with large datasets.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 37: Connection to the Database

A screenshot of a computer

AI-generated content may be incorrect.

Figure 38: Data Generation from SQL Model

* Configuring the system generation of 10K+ rows efficiently in the Database (MYSQL).

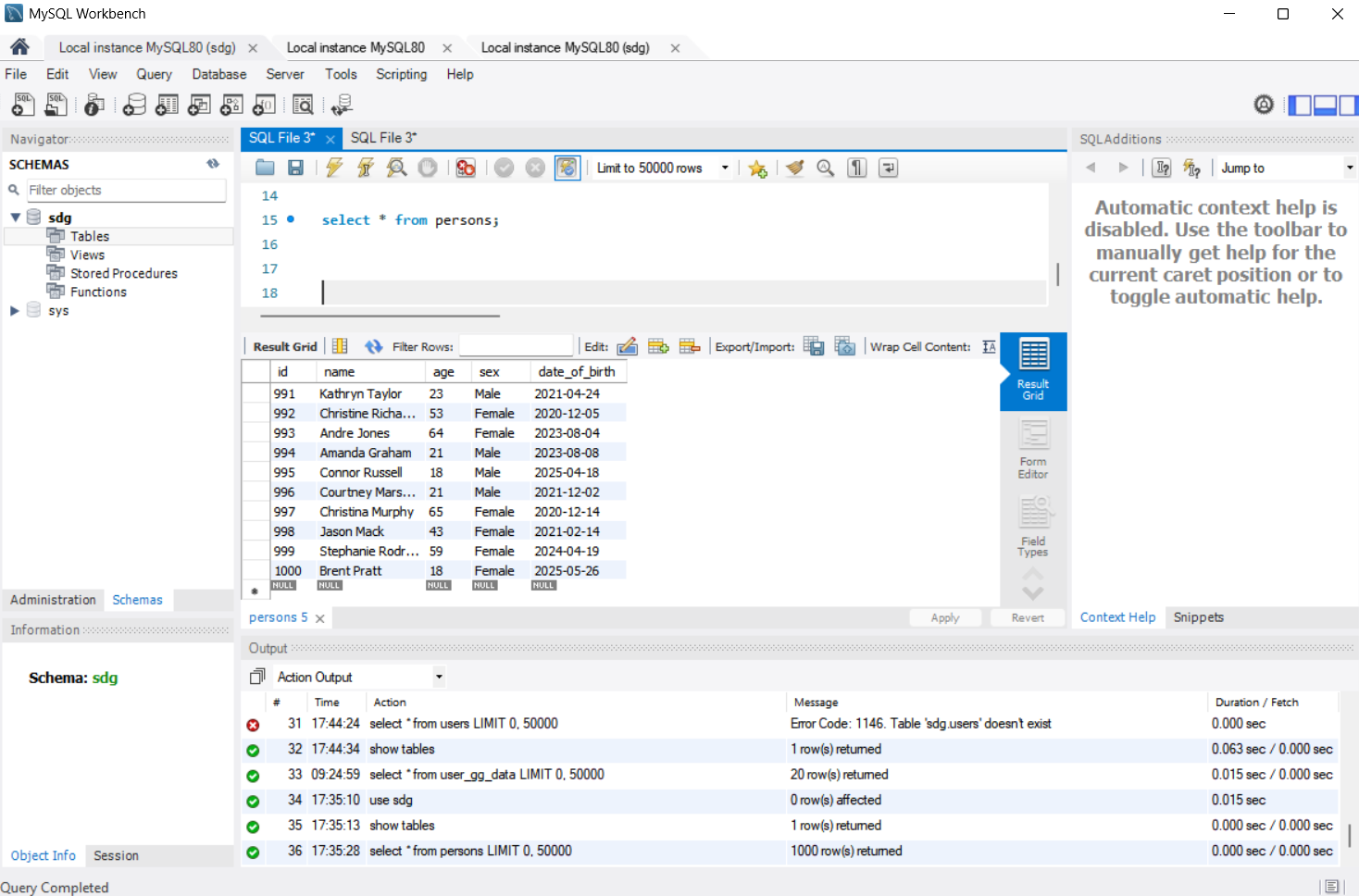


Figure 39: SQL data inserting verification

* Validate that Streamlit’s refresh and rerun mechanisms preserve data state correctly with session states.

**Activity 4.3: Deployment Preparation and Final Validation**

* The project was prepared for deployment with all dependencies documented in requirements.txt and environment setup instructions completed.
* The application was successfully deployed on Streamlit Cloud (or a local server) with proper configuration to ensure scalability and stability.
* Comprehensive end-to-end validation was performed across all modules, confirming UI consistency, functionality, and seamless integration.
* The deployed environment was verified to maintain data integrity, high performance, and responsiveness under real-world usage conditions.

**Conclusion**

The Synthetic Data Factory (SDF) represents a significant advancement in intelligent data generation, transforming how organizations create, validate, and deploy synthetic datasets for testing, development, and analytics. By integrating AI-powered data generation techniques with robust statistical validation and database compatibility, the system ensures that every dataset is realistic, consistent, and tailored to the user’s specific requirements. Whether it’s extending existing CSV files, creating custom datasets from scratch, or populating MySQL databases, SDF empowers users to produce high-quality synthetic data efficiently and accurately.

Built on a Streamlit interface, the application delivers a seamless and interactive user experience. Users can intuitively navigate between multiple data generation modes, configure datasets, monitor validation metrics, and download or directly insert synthetic data into databases. The combination of Python’s data science libraries, intelligent column detection, and database integration makes SDF both technically sophisticated and practically applicable for real-world workflows.

Looking ahead, Synthetic Data Factory has strong potential for further expansion. Features such as automated pattern detection, advanced anomaly generation, integration with additional database systems, and enhanced analytics dashboards could transform SDF into a comprehensive data simulation platform. Ultimately, this project demonstrates how AI-driven synthetic data generation can bridge gaps in data availability, accelerate development cycles, and empower organizations to make data-driven decisions with confidence and precision.