Database_Management_System

Superhero Mission Tracker - Code Explanation

Overview

This Python application is an **interactive superhero mission tracking dashboard** that demonstrates advanced database concepts, data visualization, and interactive widgets. It creates a simulated database of superhero teams, heroes, missions, and assignments, then provides an interactive interface to explore and analyze the data.

Purpose

The primary goals of this application are:

1. **Educational**: Demonstrate SQL concepts including triggers, joins, subqueries, window functions, and aggregations
2. **Data Visualization**: Show how to create interactive charts and dashboards using Plotly
3. **Interactivity**: Provide a real-time dashboard that responds to user selections and data changes
4. **Mock Data Generation**: Illustrate how to generate realistic test data using the Faker library

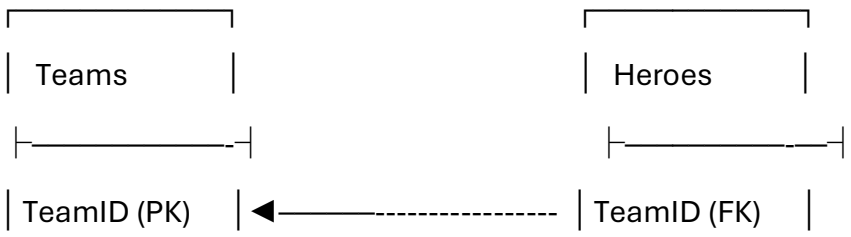
Dependencies

Library	Purpose
sqlite3	In-memory SQLite database for storing data

pandas	Data manipulation and SQL query results handling
faker	Generating realistic fake names, companies, and phrases
random	Random data selection and number generation
datetime	Date handling for mission dates
plotly. express	Interactive data visualizations
ipywidgets	Interactive UI controls (dropdowns, buttons)
IPython. display	Rendering outputs in Jupyter notebooks

Database Schema

The application creates four interconnected tables:



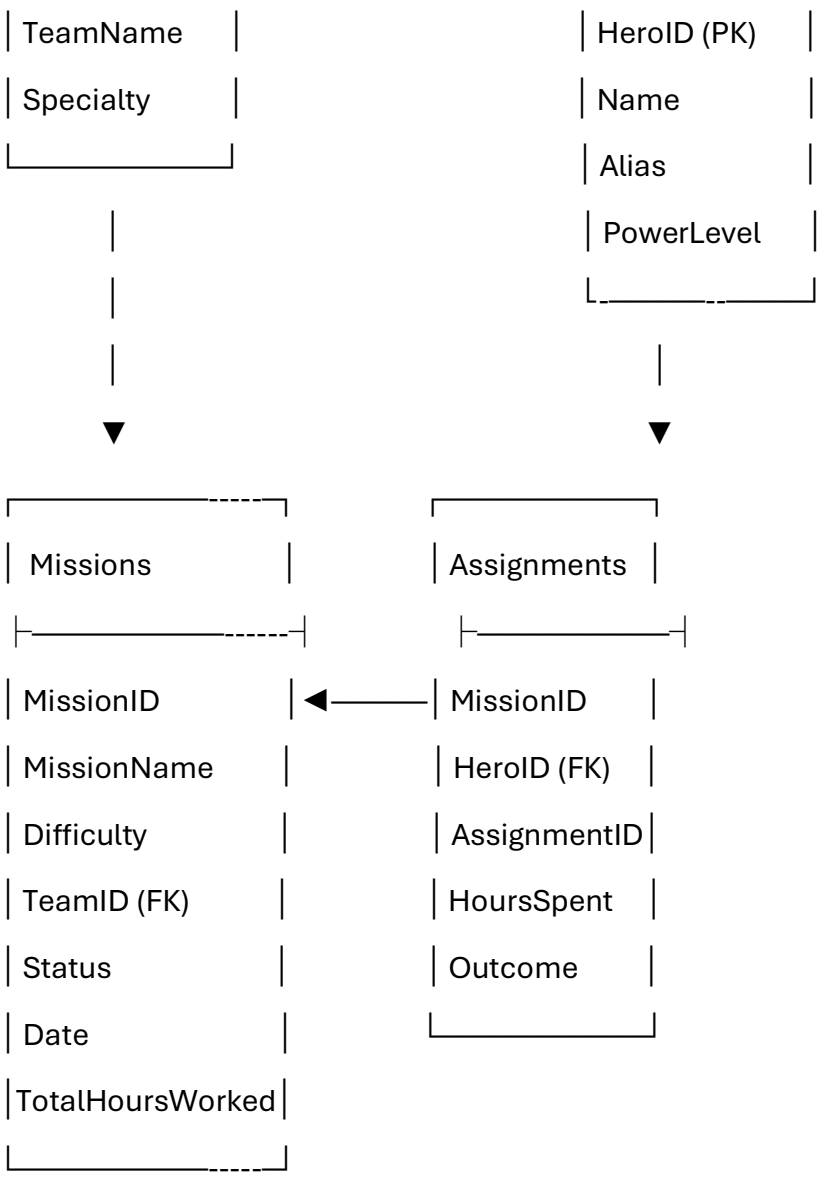


Table Descriptions

- **Teams**: Superhero teams with specialties (Combat, Stealth, Tech, Mystic, Speed)
- **Heroes**: Individual heroes with power levels (1-10), linked to teams
- **Missions**: Tasks with difficulty ratings, status, and dates
- **Assignments**: Links heroes to missions with hours spent and outcomes (Win/Loss/Draw)

Key Features

1. Database Trigger

```
CREATE TRIGGER update_total_hours
AFTER INSERT ON Assignments
FOR EACH ROW
BEGIN
  UPDATE Missions
  SET TotalHoursWorked = TotalHoursWorked + NEW.HoursSpent
  WHERE MissionID = NEW. MissionID;
END;
```

Purpose: Automatically updates the TotalHoursWorked field in the Missions table whenever a new assignment is added. This demonstrates database automation and maintains data consistency without manual updates.

2. Mock Data Generation

The generate_mock_data() function creates:

- **100 teams** with random company names and specialties
- **500 heroes** with fake names, usernames, and power levels
- **1,000 missions** with catch phrases as names and random dates throughout 2024
- **3,000 assignments** linking heroes to missions

3. Advanced SQL Queries

Function	SQL Concept	Description
query_hero_missions()	INNER JOIN	Retrieves hero aliases with their mission names and outcomes
query_team_avg_power()	Subquery + HAVING	Finds teams with above-average hero power levels

query_hero_rank_by_success()	Window Function (ROW_NUMBER)	Ranks heroes by wins with optional filtering
query_mission_heatmap()	GROUP BY + Date Functions	Aggregates missions per team per month
query_total_hours_per_mission()	Triggered Data	Shows missions ordered by total hours worked

4. Interactive Dashboard

The dashboard provides three interactive visualizations:

1. **Scatter Plot**: Hero Power Level vs Wins
    a. Size: Total hours spent
    b. Color: Team specialty
    c. Hover: Hero alias
2. **Bar Chart**: Top 20 Heroes by Wins
    a. Colored by specialty
    b. Filterable by team and specialty
3. **Heatmap**: Mission Intensity
    a. X-axis: Month
    b. Y-axis: Team name
    c. Color intensity: Number of missions

5. Interactive Controls

- **Team Dropdown**: Filter data by specific team or view all
- **Specialty Dropdown**: Filter by team specialty type
- **Add Random Assignment Button**: Inserts new data and refreshes the dashboard in real-time

How It Works

Execution Flow

1. Initialize Faker for fake data generation
2. Create in-memory SQLite database connection
3. Create schema (tables + trigger)
4. Generate mock data (teams → heroes → missions → assignments)
5. Set up interactive widgets
6. Render initial dashboard
7. Listen for user interactions and update accordingly

Real-Time Updates

When the "Add Random Assignment" button is clicked:

1. A new random assignment is created
2. The database trigger automatically updates TotalHoursWorked
3. The dashboard refreshes to reflect the new data

Use Cases

This application can serve as:

- **Learning Tool**: Understanding SQL joins, triggers, and window functions
- **Template**: Starting point for building data dashboards
- **Demonstration**: Showcasing Python data visualization capabilities
- **Testing Environment**: Generating mock data for database testing

Technical Notes

- **In-Memory Database**: Uses sqlite3. connect(':memory:') for fast, temporary storage
- **Dark Theme**: Uses Plotly's plotly_dark template for visual appeal

- **NaN Handling**: Fills missing TotalHours values with 0 before plotting
- **Jupyter Integration**: Designed to run in Jupyter notebooks with widget support

Sample Output

When executed, the application displays:

1. Sample query results (Teams Above Avg Power)
2. Thematic superhero images for engagement
3. Interactive dropdown filters
4. Three responsive Plotly charts

Potential Enhancements

- Add date range filters for mission analysis
- Include hero performance trends over time
- Add export functionality for reports
- Implement data persistence with file-based SQLite
- Add more outcome metrics (success rate, average hours per mission)