

**Overview:**

A fintech startup wants to launch a cloud-based platform that helps users manage personal finances, track expenses, automate savings, categorize transactions, and provide insights using dashboards and predictive analytics. The application should handle structured banking data, semi-structured user uploads, and real-time notifications using Azure services and big data capabilities.

**Theme:**

Design the backend schema for user profiles, transactions, categories, savings goals, and alerts. Simulate ingestion of expense logs using Python and SQL queries.

|  |  |
| --- | --- |
| S.NO | **TASKS** |
| 1 | Design logical and physical data models for user, account, transaction, and goals |
| 2 | Normalize the schema, define integrity constraints, and implement DDL using SQL |
| 3 | Populate sample user accounts, expenses, and perform CRUD operations |
| 4 | Write SQL queries: top 5 spending 4 categories, savings percentage, monthly burn rate |
| 5 | Simulate ingestion of transaction logs from 5 CSV using Python and perform transformations |
| 6 | Parse user-uploaded JSON/XML budget plans and convert to structured rows |

|  |  |
| --- | --- |
| S.NO | **Expected Deliverables** |
| 1 | ER diagrams and relational schema scripts |
| 2 | SQL scripts for data population and analytics |
| 3 | Python ETL script for transaction ingestion |
| 4 | JSON/XML transformation code |

**Team Details:**1. Koppula Shanmitha

2. Mayank Sharma

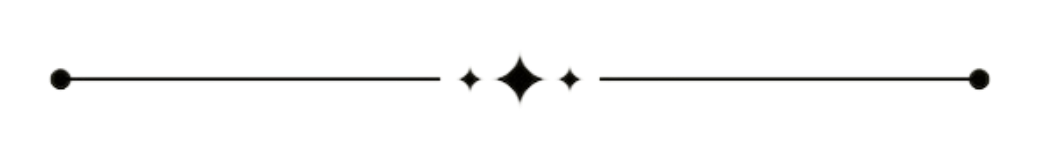
3. Megavath Ajay

4. Miniga Prasad Naidu

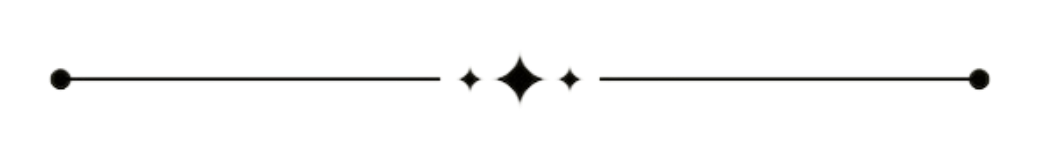
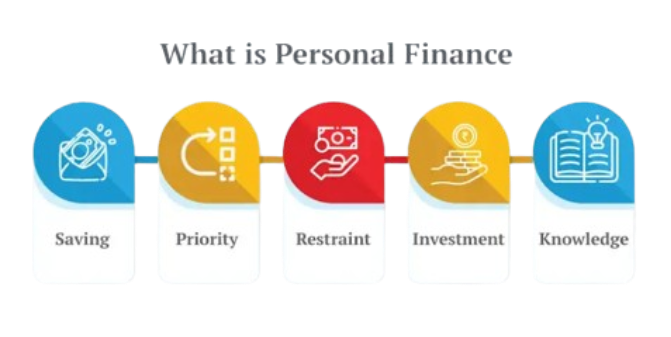
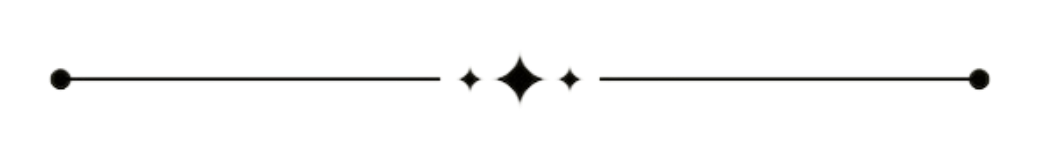
5. Mulugoju Sathwika

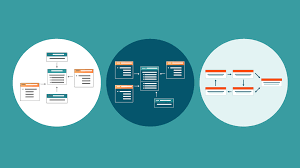
6. Musini Bhavana

7. Neha P Achar

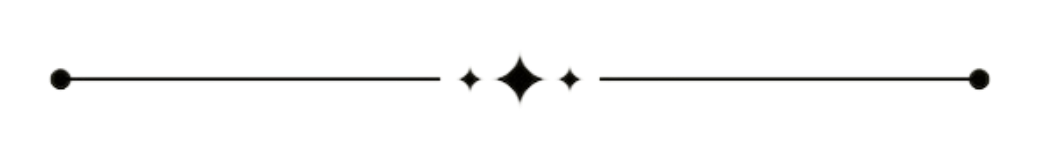
8. NithishKumar Vadivel  


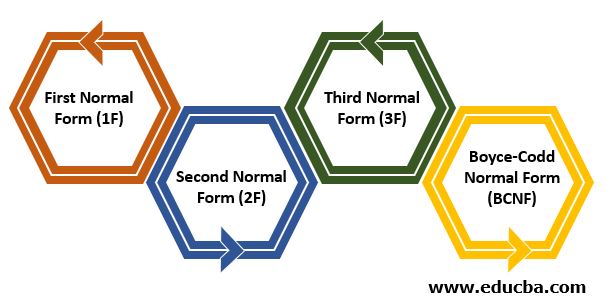
**Project Report**

* **Problem Statement**   
  Managing personal finances is a common challenge for individuals, with scattered banking data, irregular tracking of expenses, and a lack of actionable insights. Users often struggle to maintain savings goals, understand their spending patterns, or categorize expenses efficiently. Existing tools lack real-time processing, flexibility in data formats, and deep financial analytics.
* **Objective** 
* The objective of this project is to develop a cloud-based personal finance platform, FinSure, leveraging Azure services.   
  The platform will:
* Ingest structured and semi-structured data (CSV, JSON, XML)
* Categorize transactions and track savings goals
* Provide financial KPIs through SQL queries
* Enable real-time data processing and dashboard insights
* Simulate ETL processes using Python
* **Task – 1**

**Design logical and physical data models for user, account, transaction, and goals  
  
Performed By :** NithishKumar Vadivel

**Description**:

* Objective: Create a robust schema to support expense tracking, goals, and reporting.
* Actions Taken:
  + Identified core entities: Users, Accounts, Transactions, Categories, SavingsGoals, Alerts.
  + Designed an Entity-Relationship (ER) diagram showing relationships and cardinalities.
  + Translated logical model into a physical model with proper datatypes and keys.
* Tools Used: draw.io for ERD, MySQL Workbench for schema drafting.
* Outcome:
  + Scalable schema design capable of handling large datasets.
  + Normalized and ready for relational implementation.
* Challenges Faced:
  + - Ensuring normalization without compromising performance
  + - Handling complex relationships between entities like transactions and goals
* Key Learnings:
  + - Gained deeper understanding of relational modeling principles
  + - Improved skills in translating real-world finance structures into schema
* **Task - 2**

**Normalize the schema, define integrity constraints, and implement DDL using SQL  
  
Performed By :** Neha P Achar  
  
**Description:**

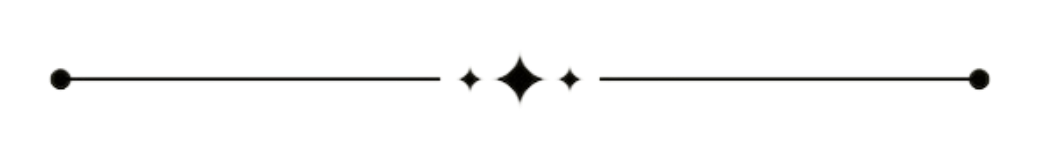
* Objective: Ensure data consistency and eliminate redundancy.
* Actions Taken:
  + Applied 1NF, 2NF, 3NF to all entities.
  + Added constraints: NOT NULL, UNIQUE, FOREIGN KEY, AUTO\_INCREMENT.
  + Created DDL scripts (CREATE TABLE) for all six entities.
* Tools Used: MySQL, SQL Server, or SQLite.
* Outcome:
  + A clean, normalized database schema with enforced relationships.
  + Executable SQL scripts ready for DB creation.
* Challenges Faced:

- Implementing constraints across multiple foreign keys

- Ensuring data consistency while normalizing

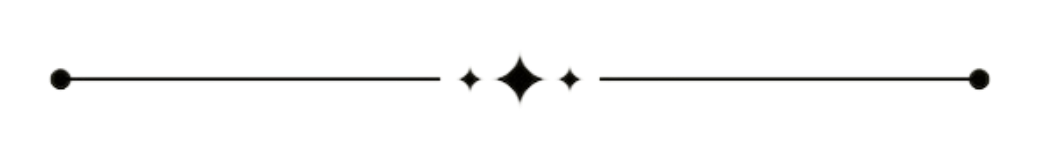
* Key Learnings:

- Enhanced command over SQL DDL statements and constraint handling

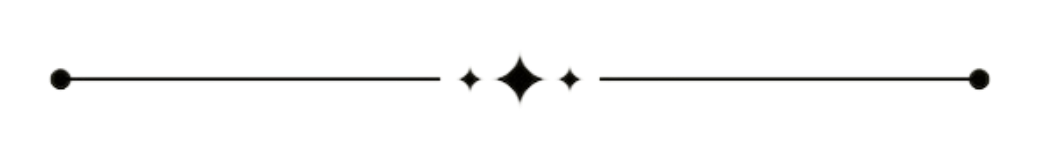
- Practiced using ERD tools to validate schema structure

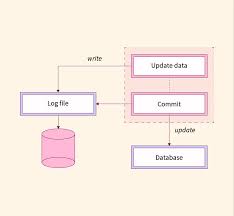
* **Task – 3**

**Populate sample user accounts, expenses, and perform CRUD operations  
  
Performed By :** Musini Bhavana  
  
**Description:**

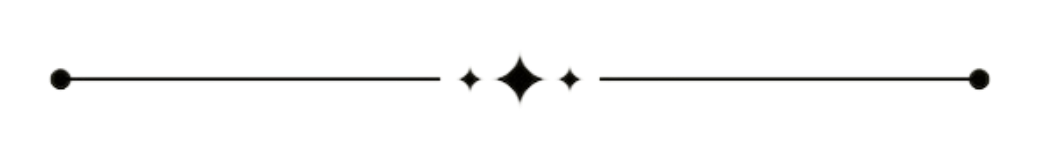
* Objective: Simulate user actions for data entry and management.
* Actions Taken:
  + Used **Faker** Python to generate dummy users, accounts, and transactions.
  + Wrote SQL queries for:
    - INSERT - add sample users, accounts
    - UPDATE - update account balances
    - DELETE - remove expired goals or test data
    - SELECT - fetch user-specific transaction histories
* Tools Used: Python (Faker, sqlite3), MySQL
* Outcome:
  + Realistic dataset created for testing queries and analysis.
  + Basic backend operations simulated successfully.
* Challenges Faced:
  + Creating realistic and varied sample data
  + Ensuring that CRUD operations worked with all constraints in place
* Key Learnings:
  + - Practiced using SQL DML commands for different operations
  + - Learned how to debug insert/update errors in complex schemas
* **Task – 4**

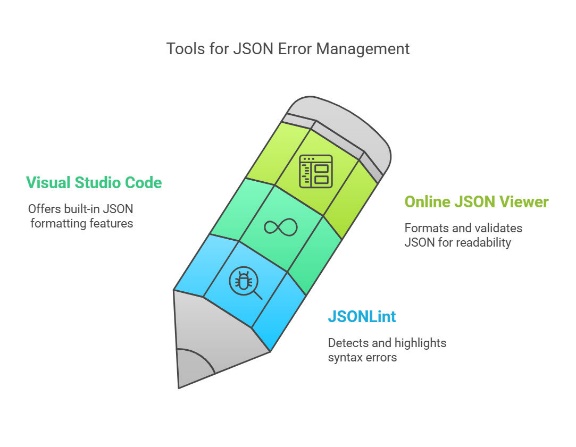
**Write SQL queries: top 5 spending 4 categories, savings percentage, monthly burn rate  
  
Performed By :** Mulugoju Sathwika  
  
**Description:**

* Objective: Provide financial insights using SQL.
* Actions Taken:
  + Wrote advanced queries using JOIN, GROUP BY, SUM, and date functions.
  + Created key financial views:
    - Top 5 spending categories
    - Monthly burn rate over time
    - User-wise savings percentage
  + Verified accuracy of results against dummy data.
* Tools Used: MySQL/PostgreSQL
* Outcome:
  + Clear, queryable KPIs to assist in personal financial decision-making.
  + Reusable SQL views created for dashboard use.
* Challenges Faced:
  + Writing aggregate queries across multiple joins
  + Handling date filtering and grouping
* Key Learnings:
  + Improved query writing and optimization skills
  + Learned to derive financial insights using SQL
* **Task – 5**

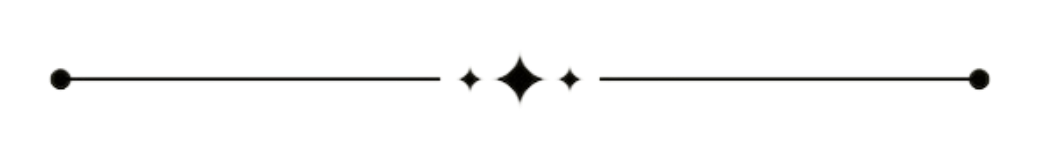
 **Simulate ingestion of transaction logs from 5 CSV using Python and perform transformations  
  
Performed By:** Miniga Prasad Naidu

**Description:**

* Objective: Simulate ingestion of transaction logs (bank statements).
* Actions Taken:
  + Used pandas to load CSV data.
  + Cleaned data (null handling, format conversion).
  + Performed transformations: categorized entries, extracted dates, converted currencies (if any).
  + Inserted into the Transactions table using DB connection.
* Tools Used: Python (pandas, csv, mysql.connector)
* Outcome:
  + End-to-end ETL simulation completed.
  + Efficient handling of batch transaction uploads.
* Challenges Faced:
  + Handling inconsistent CSV formatting
  + Mapping unstructured data to relational format
* Key Learnings:
  + Gained hands-on with pandas and csv modules
  + Learned to build simple ETL pipelines in Python
* **Task – 6**

**Parse user-uploaded JSON/XML budget plans and convert to structured rows  
  
Performed By :** Mayank Sharma

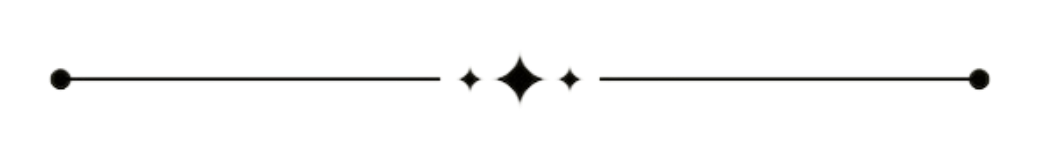
**Description :**

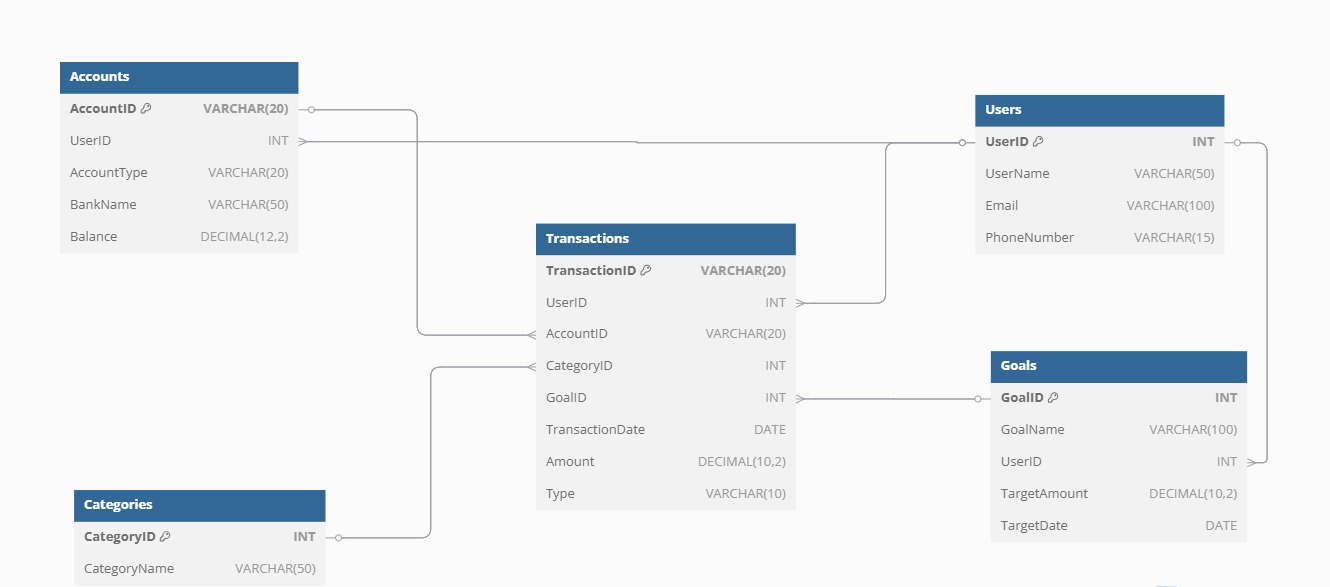
* Objective: Support structured uploads from different user tools.
* Actions Taken:
  + Created parsers in Python for both JSON and XML formats.
  + Extracted data like category names, spending limits, and deadlines.
  + Validated data and inserted structured rows into the Categories and SavingsGoals tables.
* Tools Used: Python (json, xml.etree.ElementTree)
* Outcome:
  + Flexible ingestion pipeline created for varied formats.
  + Enabled broader user compatibility for planning tools.
* Challenges Faced:
  + Parsing nested JSON and XML elements accurately
  + Ensuring conversion matches schema constraints
* Key Learnings:
  + Learned how to use xml.etree and json modules in Python
  + Developed confidence in multi-format data handling
* **Logical Data Model**

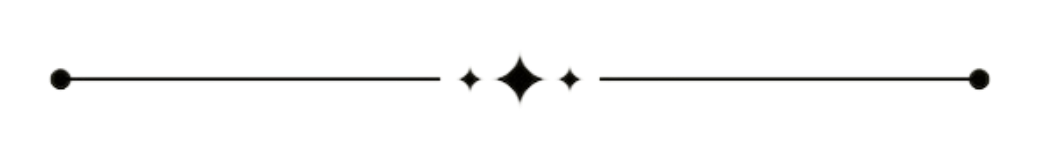
| **Entity** | **Attributes** | **Relationships** |
| --- | --- | --- |
| **User** | user\_id, name, email, phone | One user → Many accounts, savings goals, alerts |
| **Account** | account\_id, user\_id, bank\_name, account\_type, balance | Each account belongs to one user |
| **Transaction** | transaction\_id, account\_id, category\_id, amount, date, desc | Many transactions → One account, one category |
| **Category** | category\_id, name, type (income/expense) | One category → Many transactions |
| **SavingsGoal** | goal\_id, user\_id, name, target\_amount, deadline | Each goal belongs to one user |
| **Alert** | alert\_id, user\_id, message, created\_at | Each alert belongs to one user |

* **Physical Data Model**

| **Entity** | **Attributes** | **Relationships** |
| --- | --- | --- |
| **User** | user\_id, name, email, phone | One user → Many accounts, savings goals, alerts |
| **Account** | account\_id, user\_id, bank\_name, account\_type, balance | Each account belongs to one user |
| **Transaction** | transaction\_id, account\_id, category\_id, amount, date, desc | Many transactions → One account, one category |
| **Category** | category\_id, name, type (income/expense) | One category → Many transactions |
| **SavingsGoal** | goal\_id, user\_id, name, target\_amount, deadline | Each goal belongs to one user |
| **Alert** | alert\_id, user\_id, message, created\_at | Each alert belongs to one user |



*  **ER Diagram**



* **Schema Design  
    
   1. User Table**CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

full\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone\_number VARCHAR(15),

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);  
  
**2. Account Table**CREATE TABLE Accounts (

account\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

account\_type VARCHAR(50) CHECK (account\_type IN ('Savings', 'Credit', 'Wallet')),

balance DECIMAL(12,2) DEFAULT 0.0,

currency VARCHAR(10) DEFAULT 'INR',

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);  
  
**3. Category Table**CREATE TABLE Categories (

category\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) UNIQUE NOT NULL,

type VARCHAR(20) CHECK (type IN ('Expense', 'Income'))

);  
  
**4. Transaction Table**CREATE TABLE Transactions (

transaction\_id INT PRIMARY KEY AUTO\_INCREMENT,

account\_id INT,

category\_id INT,

amount DECIMAL(12,2) NOT NULL,

transaction\_type VARCHAR(10) CHECK (transaction\_type IN ('Credit', 'Debit')),

description TEXT,

txn\_date DATE NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (account\_id) REFERENCES Accounts(account\_id),

FOREIGN KEY (category\_id) REFERENCES Categories(category\_id)

);  
  
**5. Savings Goal Table**CREATE TABLE SavingsGoals (

goal\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

goal\_name VARCHAR(100) NOT NULL,

target\_amount DECIMAL(12,2) NOT NULL,

current\_amount DECIMAL(12,2) DEFAULT 0.0,

due\_date DATE,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);  
**6. Savings Goal Table**CREATE TABLE Alerts (

alert\_id INT PRIMARY KEY AUTO\_INCREMENT,

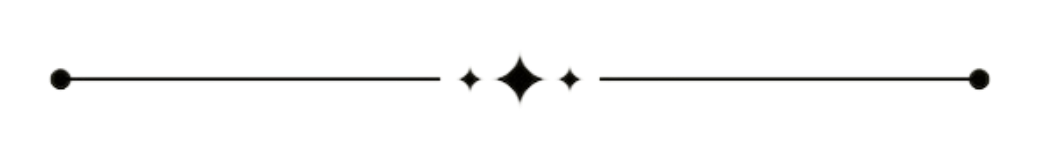
user\_id INT,

message TEXT NOT NULL,

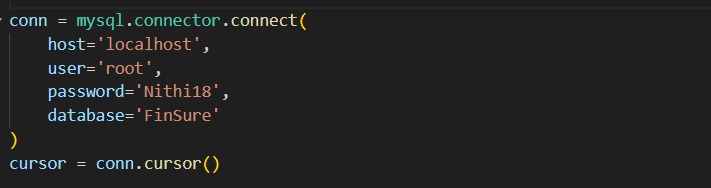
alert\_type VARCHAR(50) CHECK (alert\_type IN ('Goal', 'Overspending', 'Budget')),

is\_read BOOLEAN DEFAULT FALSE,

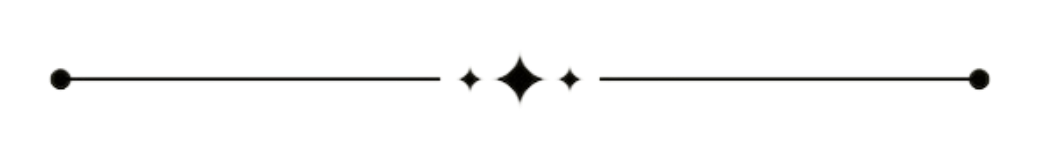
created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

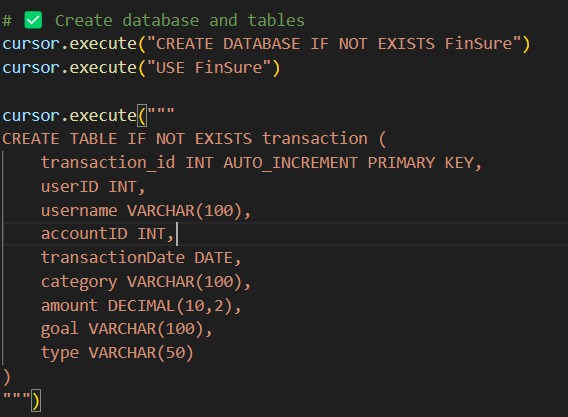
 FOREIGN KEY (user\_id) REFERENCES Users(user\_id) );

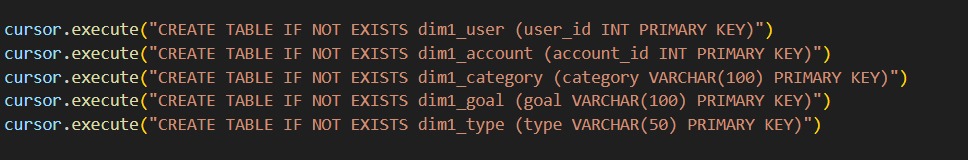
* **Integration with Python**  
  1. Used pandas to read CSV files of bank statements  
    
    
  2. Used MySQL connector to connect and insert data into database



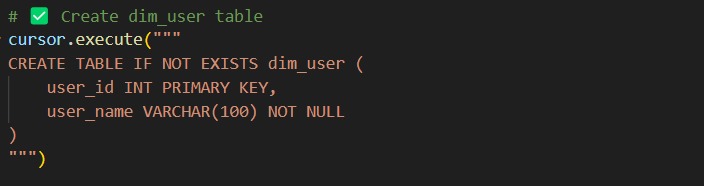
3. Loading CSV files and Date formating

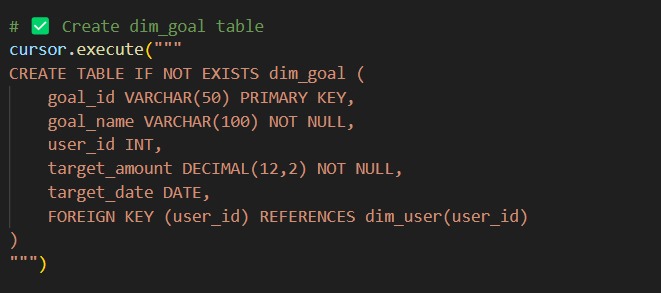


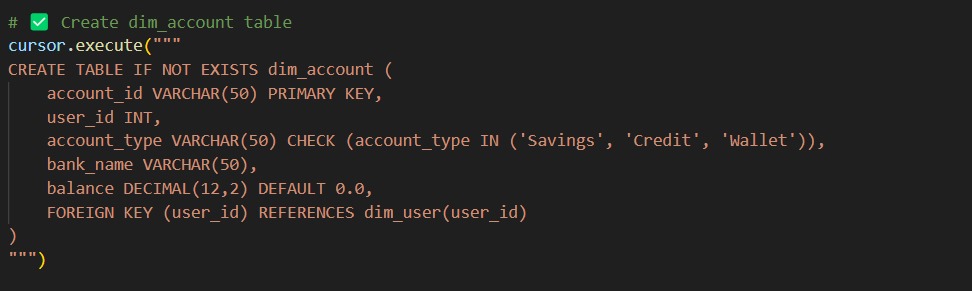
* **Core SQL Queries**   
    
  1. Database Creation and Table formation

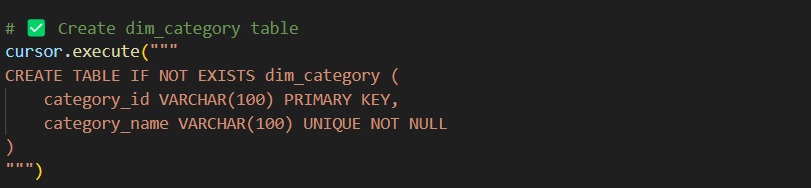


2. Creation of required tables

3. Creating User Dimensional table  
  


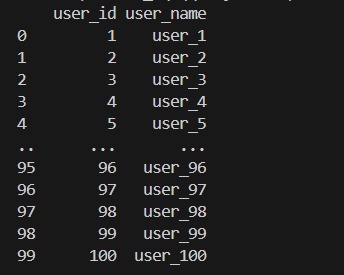
4. Creating Goal dimensional under normalization table  
  


5. Creating Dimensional Account table  
  


6. Creating Dimensional Category table  
  


7. Output- Dimensional table

User Table:



Category Table:

A black screen with white text

AI-generated content may be incorrect.

Accounts Table:

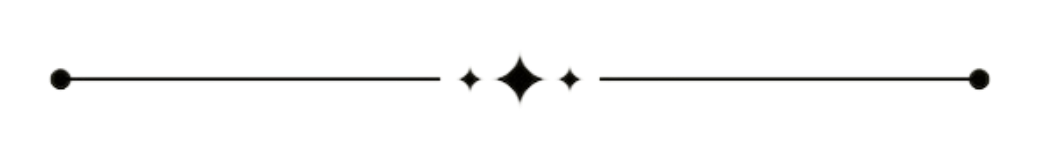
A screenshot of a computer screen

AI-generated content may be incorrect.

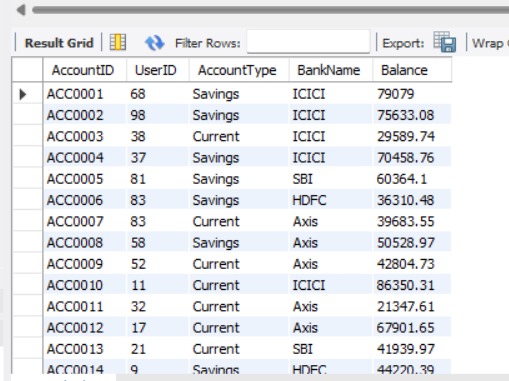
Goal Table:

A screenshot of a computer screen

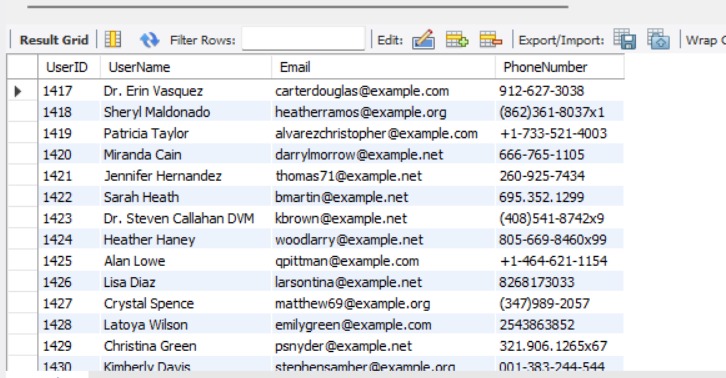
AI-generated content may be incorrect.



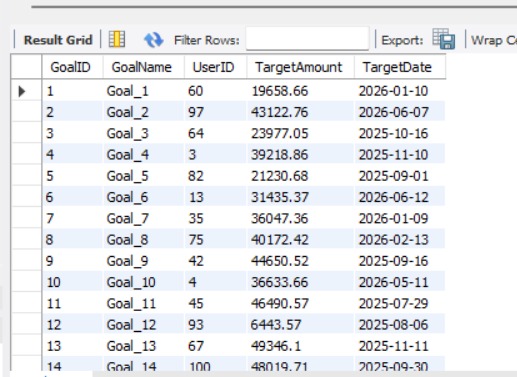
* **Core SQL Views**  
  1. Account table View

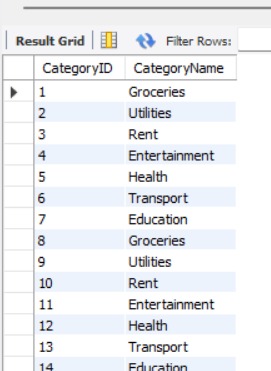


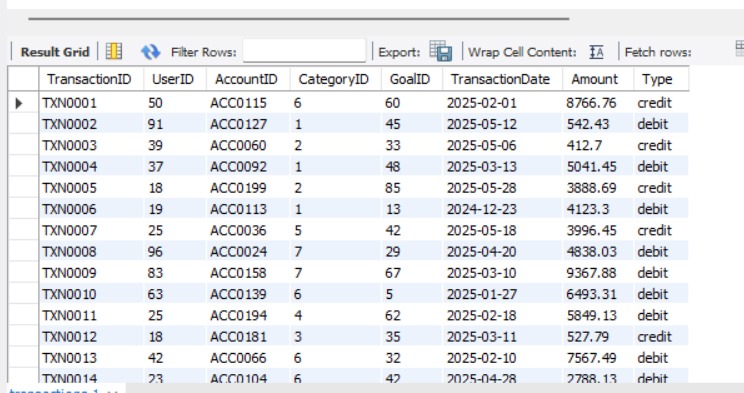
2. User table View

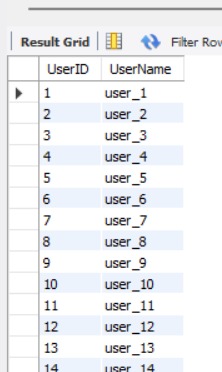


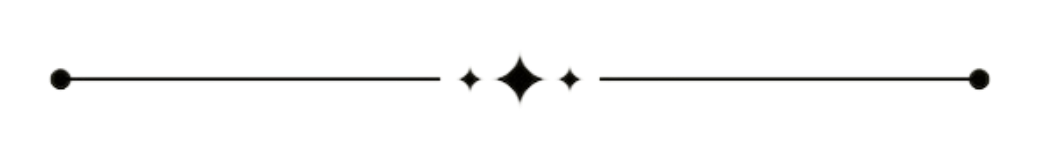
3. Goal table View

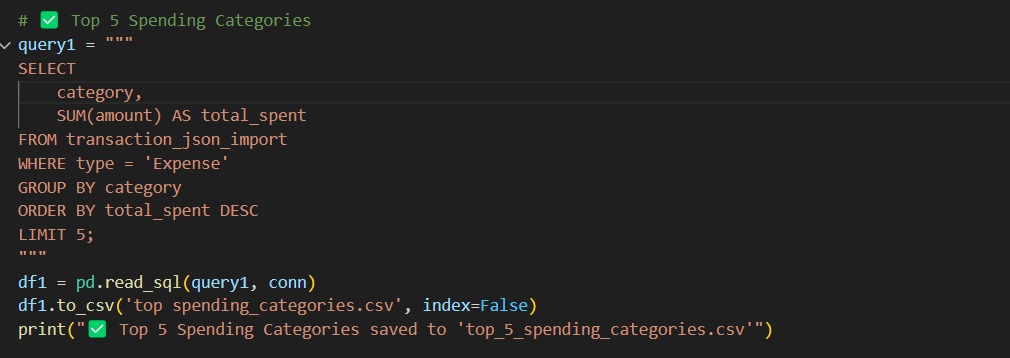


4. Category table View  
  


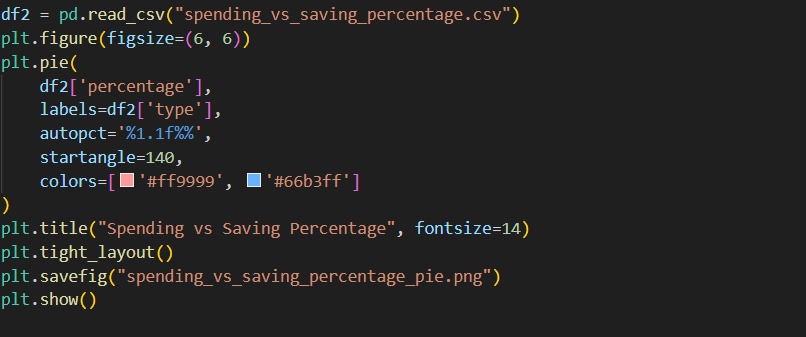
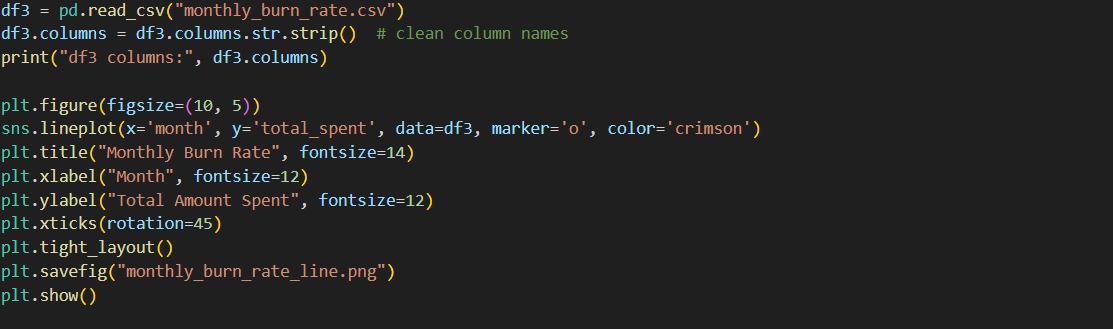
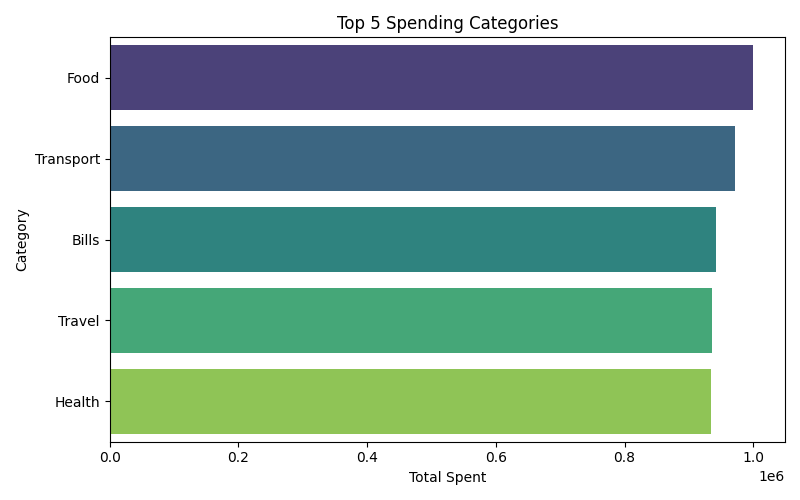
5. Transaction table View  
  
  
  
  
6. User ID table View



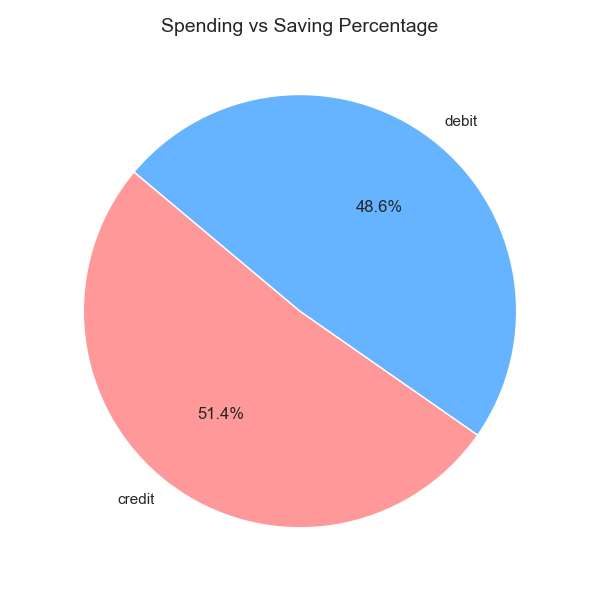


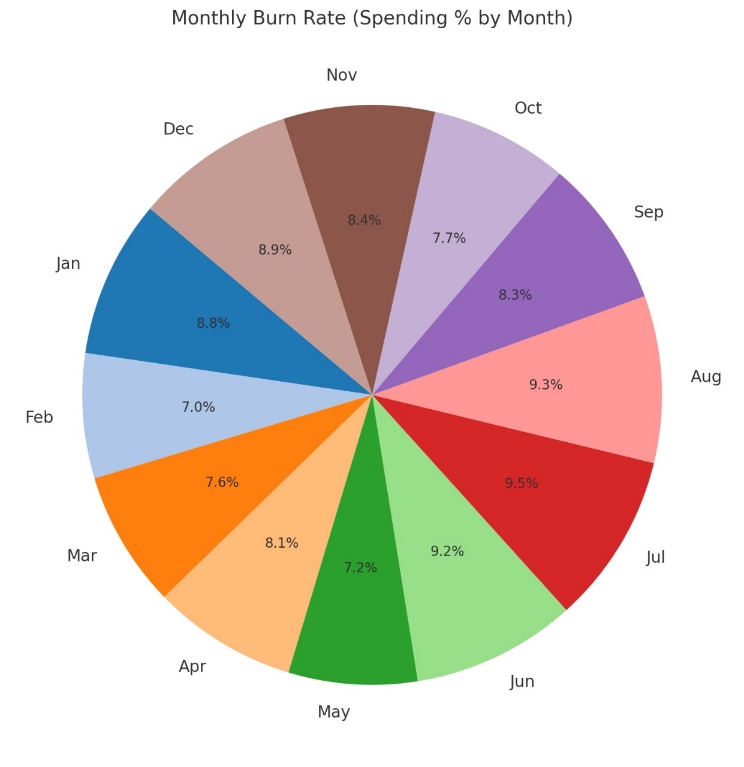
* **JSON Modules**1. Top 5 Spending Categories  
    
  2. Savings Percentage

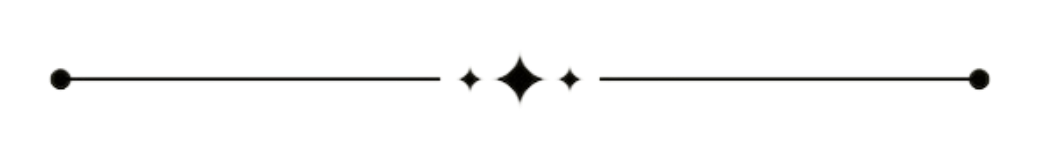
  
3. Monthly Burn Rate

* **Output Plotting Modules**1. Top 5 Spending Categories  
  2. Savings Percentage  
    
  3. Monthly Burn Rate
* **Output Visualization**1. Top 5 Spending Categories

2. Savings Percentage



3. Monthly Burn Rate

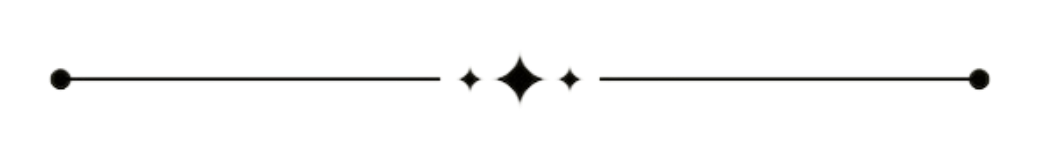


* **Future Scope**1. Integration with UPI and banking APIs for real-time sync

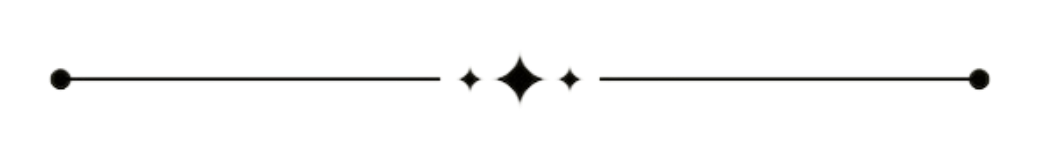
2. Azure Data Factory for scheduled ingestion pipeline

3. Power BI dashboards for advanced visualization

4. AI-based anomaly detection for unusual spending

5. Mobile app for budget alerts and visualization

* **Conclusion**

The FinSure platform successfully demonstrates a cloud-based approach to personal finance management. With SQL-backed data models, ETL pipelines in Python, and flexibility to handle various input formats, it sets a solid foundationfor scalable and intelligent financial tracking. This sprint validated data modeling, ingestion, and analytics pipelines using modern Azure-aligned practices.

* **Appendix**

**Appendix A – Sample Transaction CSV Data**  
This file contains sample transaction data used for ingestion testing in **Task 5**

****

**Appendix B – Sample JSON Budget Plan**

This JSON file represents a structured user-uploaded budget plan parsed **Task 6**



**Appendix C – Output CSV file**

* ****
* ****
* ****

