**Title: My Fat Bastard Burritos Co. sales reporting project (Database – API – Dashboarding for insights)**I’m **Jainin Vekariya**, currently working as a **Supervisor at Fat Bastard Burrito Co.**, and a **Data Analyst by passion**. Over the past two years, I’ve grown alongside FBB’s journey — both personally and professionally. While working on the frontlines, most of the time I imagine how can I implement my technical skills to improve our operations, and one night I decided that I will tried to **build a backend process for FBB and will make accurate insights for sales data**.

In this document, I will explain **flow of project step by step**. The data which I used in this project is not accurate. I generated it using **nested SQL queries and logic**.

**Tools and Technologies:**

**MySQL Workbench** – Database design and management

**Python (Flask)** – Backend development and API creation

**Postman** – API testing and error handling

**GitHub** – Version control and code backup

**Power BI** – Data visualization and analytics

**Stage 1: Database Design with MySQL workbench**

To structure the core data, I designed a relational database named fbb\_database using **MySQL Workbench**. It includes following tables.

1. Employe\_table: contains data of employe with their position and total months of experience.
2. Employe\_hours: To store weekly schedule of employes with their total number of hours
3. Expense: Shows the additional expense of the store to manage profit number (monthly)
4. Menu\_item: It is one of the major table, because it stores every menu items with their category (burrito, bowls, drinks, sides), production cost, sale price, total quantity in inventory
5. Sales: It is second major table with records of daily transactions (15-minute intervals)

This structure assist backend process for accuracy, scalability, and clean data management for reporting. Here, I am attaching the dummy database for better understanding of database.

A screenshot of a computer

AI-generated content may be incorrect.

Fig 1.1 Inventory table

A screenshot of a computer

AI-generated content may be incorrect.

Fig 1.2 Menu table

A table with numbers and numbers

AI-generated content may be incorrect.

Fig 1.3 Sales table (15 minutes intervals)

**Stage 2: API development for Backend Automation**

I developed multiple **REST APIs in Python** to automate. For starting, there are a sever file which handle server for system. In this file, every single APIs relate to single function which will call by server. I used local server for run backend.

A screenshot of a computer program

AI-generated content may be incorrect.

For database connectivity, there are a separate file to call server for database and connect with **MySQL** library. Providing database config in form of dictionary which will use by connector to configure database.

A screen shot of a computer code

AI-generated content may be incorrect.

Following code, describe how sales record data logic handle for each 15 minutes intervals. This set of code stored around data for 2 years accurately which I used for dashboard visualization.

A screen shot of a computer screen

AI-generated content may be incorrect.

**Stage 3: Integration, Debugging, and Quality Checks**

Once APIs and the database were functional, I integrated all components. During testing, I encountered **database connection issues during API calls**, but I debugged and fixed the issue by properly managing the cursor and connection lifecycle. I also conducted **data quality checks** and ensured accuracy before final visualization.