**DBMS Final project part 4- Final draft**

**TOPIC- Restaurant management database**

**Introduction-**

My project focuses on creating a “Restaurant Management Database” using PostgreSQL. The database will streamline the management of various aspects of restaurant operations including menu items, staff management, customer feedback and sales. I work at ‘The View’ and observed the activities over there. So, I thought of creating a database that would capture and track the activities of managing a restaurant. My minimal goal was to design the database with core features needed for managing a restaurant and showcase essential database functionality. My optimistic goal was to expand the database to include additional features such as customer’s feedback, Inventory management, reservations, timesheet etc. and some advanced queries to analyze sales data, customer ratings, and inventory status.

**Exploring-**

This is the first ever database I created. I didn’t have any experience with databases, and I was excited to do one. Firstly, I thought of creating some database related to banking activities and later this idea popped up in my mind and I finally went with it. I have seen how the restaurant works closely, and I considered it as my base and framed my project.

**Building-**

The restaurant management database was designed to track various activities of restaurant operations, including menu items, sales, staff schedules, and customer feedback. It incorporates tables for sections, menu items, sales, staff, timesheets, and more, with foreign key relationships ensuring data integrity. I thought of not having too many rows in the data. So, after creating the tables I used ‘Chatgpt’ to insert some mock data and edited them as per my requirements for testing, allowing realistic simulation of orders, staff working hours, and sales. Queries were created to generate insights, such as revenue by section, popular items, and top customers by spending. Triggers were set up to automate tasks like validating sale totals and updating staff sick leave based on hours worked, ensuring the database remained accurate and efficient.

**Discovering-**

This was my first step in learning how to build a database. I didn’t have any previous knowledge or experience in databases before this. But in the process of building this project I’m quite confident that I have learnt and explored a way much. I chose to build the database in PostgreSQL because there were more similarities between Oracle and PostgreSQL and thought of exploring it too.

As I mentioned my optimistic goal was to expand the database to include additional features such as customer’s feedback, Inventory management, reservations, timesheet. But I have included only customer feedback and timesheet, because I thought If have too many tables with more information I was mess up. So, I took a few of them and tried to build the database. I wanted to create a trigger which would calculate the total amount based on quantity column of order item table and price from menu item table instead of hand entering the value. I tried but there were always some errors in the code that I was not able to figure out. So, I dropped that trigger. So, as a next improvement for my project I will try to implement that in the future.

**Topics from the class-**

**Normalization:**

* The database design follows normalization principles to minimize redundancy and ensure data integrity. For example, the menu items and sections are stored in separate tables, and staff details are stored on their own table. This structure reduces duplication and ensures that data is stored logically.

**Primary Keys and Foreign Keys:**

* Primary Keys: Each table in the database has a primary key to uniquely identify rows. For example, the staff\_id in the staff table, sale\_id in the sales table, and menuitem\_id in the menu\_items table are all primary keys.
* Foreign Keys: These establish relationships between tables. For instance, restaurant\_id in the sales table is a foreign key that links to the restaurant table. Similarly, menuitem\_id in the sale\_items table links to the menu\_items table, ensuring referential integrity across the database.

**Joins:**

* Joins are used to combine data from multiple tables based on a common key. For instance, to get detailed sales data for each menu item, the query joins the sales table with the sale\_items table using sale\_id and with the menu\_items table using menu\_item\_id. This allows us to retrieve related data from multiple tables in a single query, such as revenue by section or popular items.

**Triggers:**

* Triggers automate tasks based on certain events. For example, the trigger\_sick\_leave automatically calculates sick leave hours for staff based on their hours worked, and the update\_sales\_and\_totals\_simple trigger automatically updates the sales information based on the customer\_orders table. Triggers help enforce business rules and maintain data accuracy without manual intervention.

**Cascading:**

* Cascading ensures that when a row is updated or deleted, changes are automatically reflected in related tables. For example, if a section entry is deleted, cascading deletes can automatically remove related menu item data from the menu\_items table.

**Correctness and completeness:**

The project aligns with the concepts covered in class materials, functioning as intended. The CREATE TABLE commands were manually written, while ChatGPT was utilized to generate initial scripts for data insertion and trigger creation, which were then customized to meet project requirements. All code and related resources are organized and included in the shared directory.