**Final Project : Restaurant Management System**

Name: - Savankumar Hasmukhbhai Patel

Master of Professional Studies in Analytics

Student ID Number: - 002806850

Professor: - Farnaz Derakhshan

Date: - 2024-03-30

**Introduction:**

The Restaurant Management System (RMS) is a comprehensive software solution created to optimize different aspects of restaurant operations. It is used by restaurant managers, waitstaff, kitchen staff, and customers to improve efficiency, streamline processes, and enhance overall customer experience. The application assists in order management, inventory tracking, reservation handling, and reporting/analytics. It follows a freemium model, offering basic features for free while advanced functionalities are available under subscription plans. For those passionate about technology and intrigued by the challenges of the restaurant industry, developing an RMS offers an exciting opportunity to combine these interests and contribute to improving restaurant operations.

Since I have a strong interest in both technology and the restaurant business, creating a Restaurant Management System (RMS) is a wonderful fit for my skills and interests. Having held a variety of positions in the hospitality industry, I have direct experience with the difficulties restaurant owners and employees encounter in efficiently running their businesses while offering top-notch customer service. My desire to develop a solution that tackles these issues and improves the whole eating experience for customers and restaurant employees is fueled by this personal connection. My experience in software development also gives me the ability to create and execute an RMS that is user-friendly, effective, and customized to meet the unique requirements of restaurants.

**Business Analysis:**

1. **Manager/Administrator Persona:**

* **Role:** Oversees overall operations as a restaurant manager or administrator.
* **Usage:** Relies on the RMS to manage menu items, update prices, schedule employees, monitor inventory levels, and generate reports.
* **Importance:** Needs efficient tools for management, decision-making, and performance analysis to ensure smooth operations and maximize profitability.

1. **Waitstaff Persona:**

* **Role:** Front-of-house staff handling order taking, table management, and payments.
* **Usage:** Uses the RMS to input orders, assign tables, track order status, and process transactions.
* **Importance:** Requires a streamlined system to efficiently serve customers, manage table turnover, and ensure accurate order fulfillment.

1. **Kitchen Staff Persona:**

* **Role:** Back-of-house staff involved in food preparation, inventory management, and stock replenishment.
* **Usage:** Uses the RMS to view orders, manage inventory levels, update stock status, and receive alerts for low stock items.
* **Importance:** Needs a system to coordinate food preparation efficiently, reduce waste, and maintain sufficient inventory levels to meet demand.

**Business Rules/Logic:**

* Orders require a valid table number and customer selection for processing.
* Menu items must have unique identifiers and prices.
* Reservation dates/times should not conflict with existing bookings.
* Employees cannot have overlapping shifts in their schedules.
* Inventory levels must be updated in real-time with order processing and ingredient usage.
* Reports must be accurately generated based on predefined parameters such as sales, inventory, and employee performance.

**Table Design and Analysis:**

In this RMS, a total of 5 tables are there to build the entire transportation management system. I have listed those tables below with their attributes.

1. **Customer:**

Customer\_id (Primary key): Unique identifier for all the customers.

Name: Name of customer.

PhoneNumber: Phone number of customer.

Email: Email id of customer.

1. **Employee:**

EmployeeID (Primary Key): Unique identifier for all the Employee.

Name: Name of employee

Role: Job role of employee

1. **Menu Category:**

CategoryID (Primary\_Key): Unique identifier for all the Menu Category.

Name: Name of Item Category.

Description: Description of menu category.

ImageURL: URL for image of menu category.

1. **Item:**

ItemID (Primary Key): Unique identifier for all the Menu Item.

Name: Name of Particular Menu Item.  
Description: Description of Menu Item.

Price : Price of that food item.

CategoryID (Foreign Key): Unique identifier for all the Menu Category.

1. **Order:**

OrderID (Primary Key): Unique identifier for all Order.

Name: Type of food order.

Employee ID (Foreign Key): Unique identifier for all the Employee.

Customer ID (Foreign Key): Unique identifier for all the customers.

**ER Diagram**

Link: [RMS\_ER\_Diagram](https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1#R%3Cmxfile%20scale%3D%221%22%20border%3D%220%22%3E%3Cdiagram%20id%3D%22R2lEEEUBdFMjLlhIrx00%22%20name%3D%22Page-1%22%3E7V3ZcuI4FP0aHkl5YUkeG7J1lm462%2FTMS0rBivG0sRgjCOTrRwIZjCWMjJd4q0pVsBDX9j1HR9a9ktzQ%2B%2BPFlQsmo3tkQLuhKcaioZ83NE1VFI38oyVLVtLpnK1LTNcyWNm24NH6hN5PWenMMuB0pyJGyMbWZLdwiBwHDvFOGXBd9LFb7R3Zu2edABNyBY9DYPOlf1kGHq1LT9vKtvwaWubIOzO55fU3Y%2BBVZgXTETDQx04RXOBL5GB2iQPojoEDHUy%2BuQfuH%2Bg22hcjjOmdfmtol%2BTvndY%2BMREybQgm1vRkiMakeDglVS7fwdiyqZ99hnrMEDmdftHQ%2By5CeP1pvOhDm4LlwbC%2Bpss932784FK7Ej%2FoN%2BfLuz8Kdq4%2F9Bd8f28Oum9NxoY5sGfMv98xHDP%2F4KXndOKqCf2IwRst6k0xcDHjhq6QAoI2BpZDbkw%2FV1fHtg0mU2tVfV0ysmzjDizRDHuGvKPeu7WAxsOaGrQuYckdMUYPqXHq5Ed2MfRrYFumQz4PyZ3TM%2FZcOCXXcgemmNXgfcPcNYcuhgtfEfPVFURjiN0lqcK%2BPWWUYO2k2fY48rFlneaxbuRj3IZMgNHI3NjeokM%2BMIAigKVzYO0Fitw6toD9QFogcMwVZruQUL8aLpo8AdeEmBVMkEU9ejGH6zawcr5l231kI4qsgxzoVVvdXLtH%2Fsjt9pWTdqNNLqBPjtXtMfmj1V3cR84Uu4Qh9FSQAPUBKVg9jCbsPDZ89y7DZc6kn98QxqRJ7YM1lNSHsWbg6pLQ6mkh2%2BKQHdxGwRaRe323V2I2sgwDOusmSQUYbPEWQCn0%2F8bnQTCCDVESD10aDx8AekT%2FM2Nbr0S2BmzCfQdg2EMzx5hyoG6u83ic20K5%2FX6eQ6w9kV3X7U0nYGg55t36l50AGdppkGGxv3FqiZJDylwG7OhUWd%2BVFPW9I6nvrbT0vcsh%2B5hHfZf0f0fa%2FwcUOMzf0fU8aC2DFnvK4foDjGEOkT2o5olDH1G9j6eClLkMuHBWq3c66r0ZwX%2BZfHuxjyOxzZd6n0n7v%2BzqraocrOdwOnStCbaQk0OEk1JxeQqUWcWNLvilwcHN%2FfnL5c2vM%2BX1ZvDSVLu1aiej2q0MVVsM5WkOmzDv7lAaVkulxa7gH6wGrjUsyVN2XPSrJ9CtWqCTEejTLxdoPhx6WYywSCgza81uqnwo8%2FL2tU%2FOaSJ3Wcx4d%2FIsKLN2iwdcfAbzYjyx0RLu780rlHLuBlLOqqJKSrSaXuiDT01WqP%2BNkORU5RNbeck6C%2FrfMqWdk%2BuPC593FvTHnvAWszM%2BMvmcXudc%2FOyzyicpK6T0EUZa0ZX%2By%2FPPKp%2BoLHICuprRMbEr%2BOhYyVPQdXAsZPCp1pKdjGRnmnQWYxmp%2B81ZKEx%2B7FsigRa7go95PCC7JAIdF%2FwyC%2FTr%2Bc380sQfrz3rz3Lw36f29OJu2OADHhom9IYyRBdHyEQOsC%2B2pT2XXg40GEzbOndohS8dEv8LMV6y6BiYYUSKRnhss2%2FhwsK%2FfZ%2F%2FpqZO2uzofMEsrw6WxynrFM3cIZTotrHXvRyqSP0SSh4X2gBbc3%2BlFB6dNb799mdT0oLWi3iqHrNseW3XC1qeidJK3hqznXUyEVuzd8LO7gkFIdKuoK1%2FwYBaq0OnchqiJRI6FXEstdCpVu7QqVaHTjeu4EOnnv5XKnQagRIhT3V6xEZ64KlOxlwWHKlDp2kpvSh0KlL61EKnWrlCp1odOt24onKh0wjgR9TwGCNzGXPJsqH58O36eaye)

A screenshot of a diagram

Description automatically generated

**Relationships:**

1. Employee to Order (One to many).
2. Customer to Order (One to many).
3. Order to Item (One to many).
4. Menu Category to Item (One to many).

**Database Implementation:**

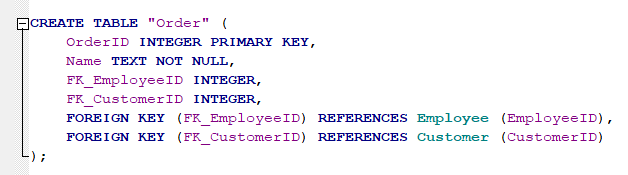
It's now time to put what we have seen and discussed in this report's earlier sections into practice. Using a DB Browser, we will build a database, construct tables, and add records to those tables.

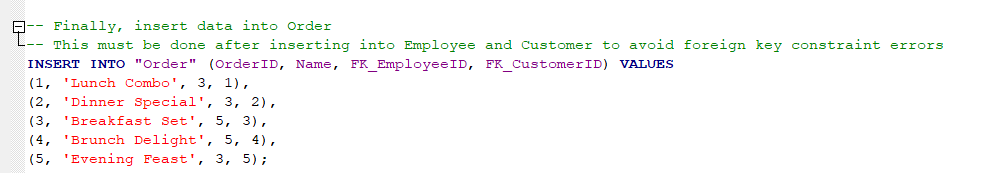
I created a project called rms\_final\_project. Once the database is set up, the next step involves creating the tables as outlined in the Table Design and Analysis section, utilizing the CREATE TABLE command. Furthermore, data will be added to these tables through the INSERT command.

To illustrate this process, an example involving the creation of the Order table is provided. This example is particularly instructive as it demonstrates not only how to establish tables and populate them with data but also how to designate primary and foreign keys. The methods used for the Order table serve as a template for constructing other tables and inserting data into them using similar commands.

Figure1.

*Create a Order Table and Insert the data into it using SQL commands.*



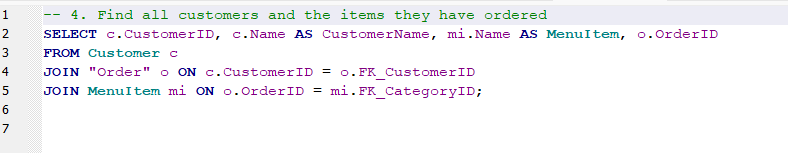


**Analytics, Report and Metrics**

Next, we will execute SQL queries on our RMS (Restaurant Management System) database. By joining two or more tables, we aim to extract valuable information.

Figure 2.

*retrive all customer and items they have order*



A menu with black text

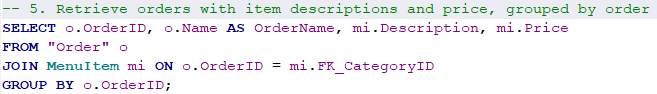
Description automatically generated

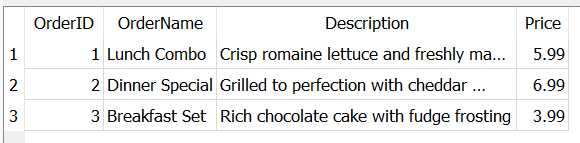
In summary, the SQL query successfully pulls together data from multiple tables to provide a detailed view of customer orders, effectively showing which customers ordered which menu items. This type of query is essential in databases where information is normalized and spread across different tables but needs to be reported in a unified format.

Now let’s retrieve orders with item description and price, grouped by order. Refer Figure below for more information.

Figure 3.

*retrive all orders with item description and price, grouped by order.*



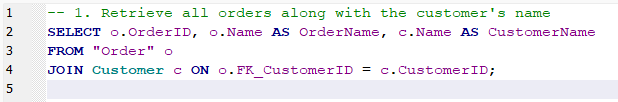


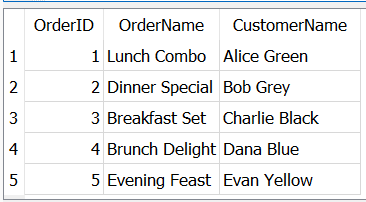
The SQL command provided is set up to gather and organize information about orders, including each order's name, item descriptions, and prices, and to group this information by each order's unique ID. The resulting display neatly summarizes this data, showing each order with details like 'Lunch Combo' or 'Dinner Special' alongside their descriptions and prices, making it clear and easy to read.

Now We will Retrieve all orders along with the customer's name

Figure 4.

*Retrieve all orders along with the customer's name*





The SQL query shown pulls together a list of orders from a database, matching each order with the customer's name. The result is a neat table that displays the order ID, the name of the order, and the name of the customer who made it. This setup allows anyone looking at the table to quickly see who ordered what, like Alice Green's 'Lunch Combo' or Bob Grey's 'Dinner Special'.

**Security Concerns**

In our restaurant system, we keep a lot of information like names, jobs, phone numbers, and email addresses for our workers and customers. This information is private and if someone who shouldn't see it gets access, it could cause big problems, like someone stealing an identity.

Our system also keeps track of what people order. If this information got out, it could tell someone a lot about our customers' habits. And even the details about our menu items, like their descriptions and pictures, need to be kept safe because they are part of how we run our restaurant.

We need to make sure that only the people who need to see this information can get to it, and that it's locked up tight when we send it through the internet or store it. We also need to keep checking our system to find any weak spots before they become a problem. Keeping everything safe and sound is key to making sure everyone can trust our system.

**Architecture**

**Client/Server Architecture Solution:** In our Restaurant Management System (RMS) project, different users such as staff, managers, chefs, and waiters interact with the system through user-friendly interfaces. These could be websites accessed via computers or apps on phones and tablets. All our data is stored on a main server that handles requests and sends out the right information. Having a client/server setup means we can keep everything organized from one spot, make the system bigger when we need to, and keep our data more secure. It also makes it easier to fix or update parts of the system without messing with the rest.

**Cloud-Based Hosting:** By hosting our RMS on the cloud, we’re using remote servers run by companies like AWS, Azure, or Google Cloud to hold our application and data. These services give us strong infrastructure tools like virtual servers, storage, and secure networks. Cloud hosting is great because we can scale it to handle busy times or slow times, change things around as needed, reach our system from anywhere, and pay only for what we use, saving money compared to having servers in our restaurant.

**Storage Requirements:** The amount of storage our RMS needs depends on how much data we have, how often it changes, and what kind of data it is—like text for orders, pictures for menu items, or videos for training. We have to store lots of details about menus, orders, inventory, employees, and sales, so we need enough room to keep all this without any hiccups. Using smart storage methods like organizing data well, squishing it down, and dividing it into chunks helps us use space better and makes the system run faster.

**Project Wrap-up and Future Consideration**

As the RMS project comes to a close, the experience has been immensely enlightening and has significantly expanded my skill set. To start with, I've honed my understanding of business analysis, identifying the different user roles within a restaurant environment and mapping out their interactions with the RMS. I've also become adept at designing Entity-Relationship Diagrams (ERDs), learning how to detail entities, attributes, and the connections between them, and how to create and populate database tables efficiently.

Navigating through the project, I've established relationships among various database elements, ensuring smooth data flow and management. Additionally, I've tackled architectural design, enhancing the system's functionality and scalability—this has improved my ability to craft resilient frameworks that support the specific needs of our RMS.

Importantly, the project has underlined the critical importance of security. After setting up the database, it became clear that protecting sensitive data is a top priority, given the risks of unauthorized access and potential data breaches.

Going forward, I'm excited to scale up the RMS to support more users and to further improve its features. Future upgrades are set to include sophisticated order tracking, advanced table management algorithms, seamless online reservation and delivery systems, and in-depth business analytics to offer an exceptional experience for both restaurant staff and patrons. These enhancements will propel the RMS to new heights, ensuring it remains a vital tool in the restaurant industry.

**References**

* Database Academy, Database Components,

<https://database.academy/database-tutorials/database-architecture/>

* Database Academy, Database Approach,

<https://database.academy/database-tutorials/database-architecture/>

* DB Browser for SQLite. (n.d.),(February 21st, 2024), <https://sqlitebrowser.org/dl/>
* Savankumar H. Patel, (February 25th, 2024), savan\_rms.pdf,

Database Management System, Northeastern University