**RESTAURANT MANAGEMENT SYSTEM**

**ABSTRACT**

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

Running a restaurant is hectic enough as it is, so why not make the day-to-day processes easier byhaving a system that will help ease the workload for you? There are so many day-to-day processesthat restaurants have to deal with. These can range from scheduling in employees, managing HR,monitoring employee attendance to preparing for payroll and to keep record of transactions anddatabase. In current marketplace, there is a great value for food, restaurants and its management.There is day by day increment on the number of restaurants and food places that are emergingtoday. It can be considered as a rapid growth in the field of business and food restaurants and itsmanagement system. The management system applied for every restaurant is different from theother one. Some restaurants may be bigger while the other may be smaller but every restaurant orhotel requires a management system and this is termed as Restaurant Management System.

RMS that is, Restaurant Management Systems are the crucial technologies that enables a singleoutlet or enterprise to better serve its customers and aid employees with food and beveragetransactions and controls. Restaurant management System is database program that keeps recordof all transaction carried out in the restaurant on daily bases. The Restaurant Management Systemhelps the restaurant management to keep adequate record of all transactions carried out and doesthat will still be carried out by the restaurant and maintain the database of the restaurant.

While investing in elements such as marketing and décor will go a long way in advancing thegrowth of a restaurant, investing in the right technology also plays a huge role, with restaurantmanagement software b eing among the “must haves”. Every restaurant out there, whether small, medium-sized, or large, will benefit greatly by switching from manual restaurant management processes to automated or software-based ones.

**TABLE OF CONTENTS**

**CHAPTER TATLE PAGE**

**ABSTRACT 4**

**LIST OF ABBREVIATIONS**

1. **INTODUCTION** 
   1. **SCOPE 6**
   2. **MOTIVATION 7**
   3. **ER DIAGRAM 8**
2. **SYSTEM DESIGN**
   1. **ARCHITECTURE DIAGRAM 9**
   2. **MODULE DESCRIPTION 10**
   3. **RELATIONAL ALGEBRA 13**
3. **QUERIES**
   1. **SQL QUERIES 14**
   2. **TRIGGERS 25**
   3. **UPDATE 27**
   4. **JOIN 28**
   5. **GROUPBY 29**
4. **ADVANCED IMPLEMENTATION**
   1. **NORMALIZATION 30**
   2. **NoSQL 36**
5. **CONCLUSION 38**
6. **FUTURE ENHANCEMENT 39**
7. **REFERENCES 40**

**CHAPTER 1**

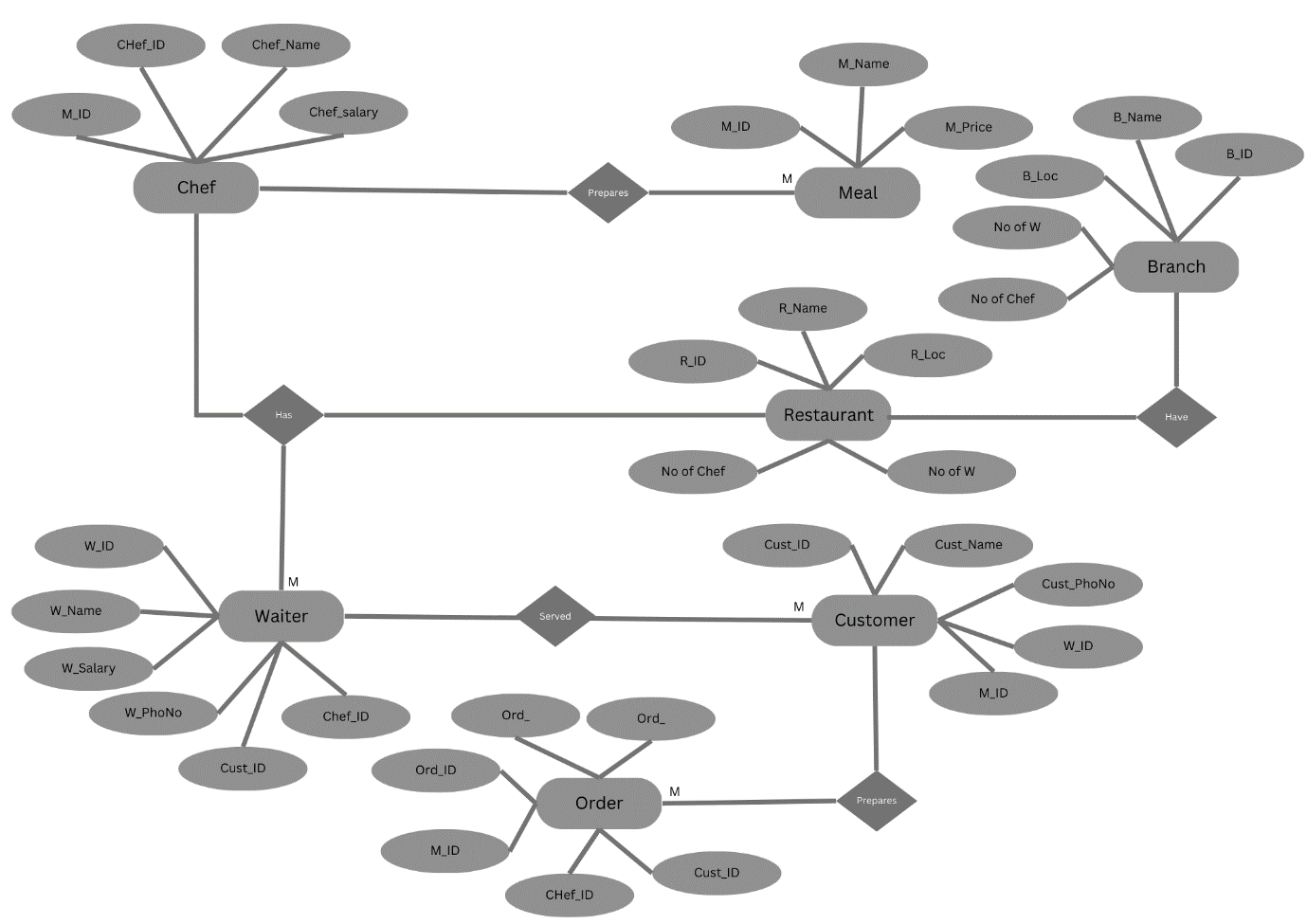
**SCOPE**

Scope of project in building a computerized system for silk route to handle billing restaurant records was to include the employees who are involved in the process of billing of a customer to storage of restaurant records and enables to view the records as desired. The employees are given limited access in order to safe guard the privacy and security of the records. The database is maintained in the whole project.

**MOTIVATION**

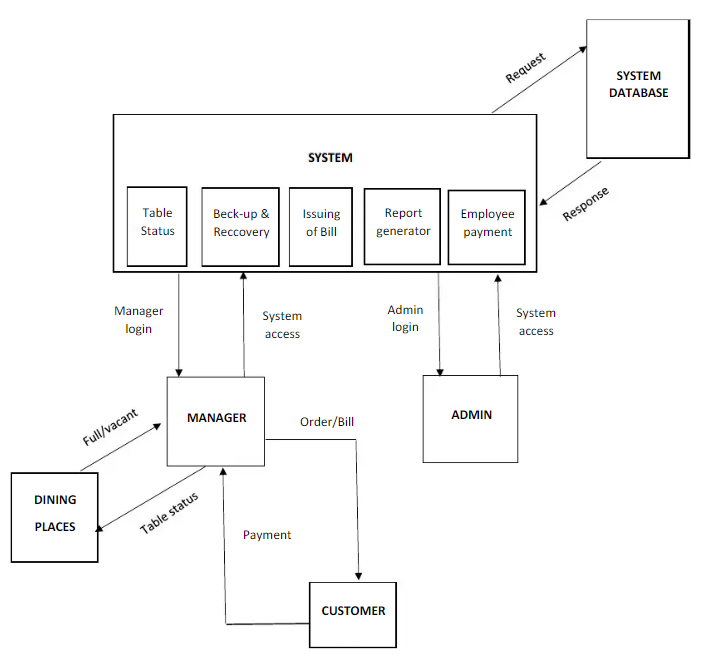
A restaurant management system is a software platform designed to help restaurants automate and streamline their operations. The system can help to improve the overall dining experience for customers, increase operational efficiency, and reduce costs. By automating tasks such as order taking, processing, inventory management, and table management, staff can work more efficiently and effectively. Features such as loyalty programs, customer feedback management, and data analytics can help restaurants build better relationships with customers and make data-driven decisions to improve their business. The system can also help to ensure compliance with health and safety regulations, optimize employee scheduling and performance, and reduce waste and associated costs. Overall, a restaurant management system can help restaurants run more smoothly, provide better service to customers, and make informed business decisions to improve profitability.

**ER DIAGRAM**



**CHAPTER 2**

**ARCHITECTURE DIG:**

****

**MODULE DESCRIPTION**

A Restaurant Management System (RMS) is a software application designed to help manage various aspects of restaurant operations. The system can help streamline various tasks such as menu management, inventory tracking, employee management, and order processing.

The key modules of a restaurant management system may include:

Menu Management: This module allows restaurant owners and managers to create and manage menus, add and edit dishes, set prices, and update availability.

Order Processing: This module helps manage the ordering process, including table allocation, order taking, and order processing. This module may also include a POS system to help process payments.

Inventory Management: This module allows restaurants to track inventory levels, monitor stock usage, and generate reports on inventory levels.

Table Management: This module helps manage table reservations, table assignments, and table turnover. It can also provide real-time information on table occupancy, wait times, and table status.

Employee Management: This module allows restaurant owners and managers to manage employee schedules, monitor attendance, and generate reports on employee performance.

Reporting and Analytics: This module provides real-time information on various aspects of restaurant operations, including sales, customer feedback, and inventory usage. It can also generate reports and analytics to help identify trends and make informed business decisions.

Overall, a restaurant management system can help improve operational efficiency, increase profitability, and enhance customer service.

Customers interact with our system directly in order to place order, modify order, get bill and give feedback. We do not store any information related to customers in our system. The process of order taking starts from customers placing order and then the other series of events begin.

Head Chef/Kitchen Manager Head Chef can mark a dish as prepared when a chef tells him to do so. He can approve the cancellation of an order whenever a customer edits or removes a dish from his order. He can also assign a dish to a particular chef based on the specialty of the chef.

Chefs don’t interact with the system. They just have to look at the dishes present in their queues and prepare the dishes accordingly. Chef’s name, address and specialty etc. are stored in the database.

Admin’s job is to manage the inventory and other information related to menu and chefs in the system.

Hall Managers will provide its input when he marks the bill as paid when customers pay for their order or get the bill printed. Moreover, he gets a notification whenever a particular order is complete, or some customer asks for help through the system. Hall manager can also see tables in the hall and their status i.e. empty or filled.

Customer: A person who visits the restaurant to dine in, order takeout, or place a delivery order. Customers can create accounts to save their contact and payment information, view order history, and earn loyalty rewards.

Server: A member of the restaurant staff who takes orders, serves food and beverages, and handles customer requests. Servers use POS systems to enter orders and process payments, and they may also manage table reservations, check availability of menu items, and make recommendations to customers.

Chef: A skilled culinary professional who prepares and cooks food in the kitchen. Chefs manage inventory, order supplies, and create recipes, as well as supervise kitchen staff and ensure food quality and safety.

Menu: A list of food and drink items that are available for purchase at the restaurant. Menus can be customized for different meal times, dietary preferences, and special occasions, and may include descriptions, prices, and photos of each item.

Table: A piece of furniture in the dining area where customers sit and eat their food. Tables may be assigned to specific servers or reserved in advance, and may have different seating capacities, configurations, and accessibility features.

Reservation: A booking made by a customer to reserve a table at a particular time and date. Reservations can be made online, by phone, or in person, and may require a deposit or prepayment, cancellation policy, or special requests.

Order: A request made by a customer for a specific food or drink item. Orders can be customized for different ingredients, cooking methods, and serving sizes, and may include special instructions or dietary restrictions.

**RELATIONAL ALGEBRA:**

* Retrieve all customer names and phone numbers:

π name, phone\_number (Customers)

* Retrieve all orders made by a customer with a given email:

σ email='example@email.com' (Customers ⋈ Orders)

* Retrieve the total amount for each order:

π order\_id, total\_amount (Orders)

* Retrieve the name and price of all menu items:

π name, price (Menu\_Items)

* Retrieve the name and price of all menu items ordered by a customer with a given email:

π name, price (Menu\_Items ⋈ (Order\_Items ⋈ σ email='example@email.com' (Customers ⋈ Orders)))

*Selection*

*BRANCH\_NAME="perryride" (Amount)*

*Projection*

*∏ NAME, CITY (CUSTOMER)*

*Union*

*∏ CUSTOMER\_NAME (Amount) ∪ ∏ CUSTOMER\_NAME (Debit)*

*Set*

*∏ CUSTOMER\_NAME (Amount) ∩ ∏ CUSTOMER\_NAME (Credit)*

**SQL Queries**

CREATE DATABASE restaurant\_management;



USE restaurant\_management;



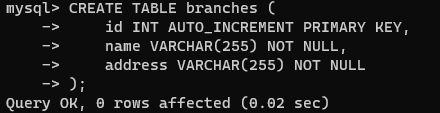
CREATE TABLE branches (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

address VARCHAR(255) NOT NULL

);



CREATE TABLE chef (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

branch\_id INT NOT NULL,

FOREIGN KEY (branch\_id) REFERENCES branches(id)

);

drop table if exists `SALE\_DETAIL`;

create table `SALE\_DETAIL`(

`Date` date NOT NULL,

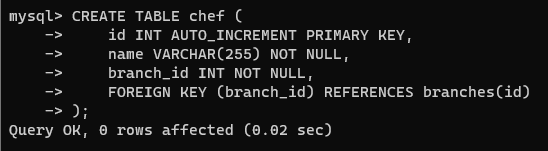
`Daily` int NOT NULL,

`Weekly` int DEFAULT NULL,

`Monthly` int DEFAULT NULL,

`Rname` varchar(30) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;



CREATE TABLE customer (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone VARCHAR(255) NOT NULL

);

drop table if exists `SUPPLIER`;

create table `SUPPLIER`(

`Fname` varchar(15) NOT NULL,

`Lname` varchar(15) NOT NULL,

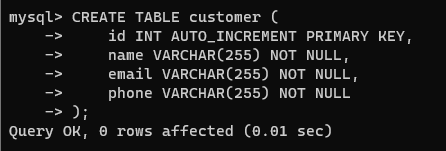
`Address` varchar(30) NOT NULL,

`Contact` varchar(20) NOT NULL,

`Details` varchar(500) DEFAULT NULL,

PRIMARY KEY (`Fname`,`Lname`,`Address`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

e

CREATE TABLE meal (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

price DECIMAL(10, 2) NOT NULL

);

drop table if exists `BOOKING`;

create table `BOOKING`(

`Booking\_Id` int NOT NULL AUTO\_INCREMENT,

`Table\_Num` varchar(30) NOT NULL,

`Date` varchar(30) NOT NULL,

`Time` varchar(30) NOT NULL,

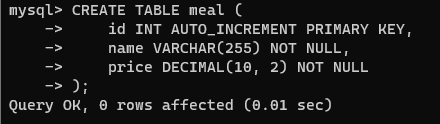
`Cust\_Id` int NOT NULL,

PRIMARY KEY (`Booking\_Id`),

FOREIGN KEY (`Cust\_Id`) REFERENCES `CUSTOMER`(`Customer\_Id`),

FOREIGN KEY (`Table\_Num`) REFERENCES `TABLES`(`Table\_Number`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;



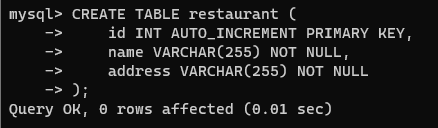
CREATE TABLE restaurant (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

address VARCHAR(255) NOT NULL

);



CREATE TABLE waiter (

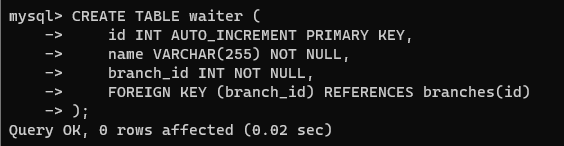
id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

branch\_id INT NOT NULL,

FOREIGN KEY (branch\_id) REFERENCES branches(id)

);



insert into `MENU`(`Name`,`Price`,`Type`,`Category`)

values

("Vegetable Pakora","3.00","Veg","Starters"),

("Vegetable Samosa","3.00","Veg","Starters"),

("Onion Bhaji ","3.00","Veg","Starters"),

("Potato and Mushroom Chaat","3.00","Veg","Starters"),

("Mushroom Garlic Fry","3.00","Veg","Starters"),

("Chicken Tikka","4.00","Non-Veg","Starters"),

("Tandoori Chicken","4.00","Non-Veg","Starters"),

("Chicken Garlic Fry","4.00","Non-Veg","Starters"),

("Chicken Tikka on Puree","4.00","Non-Veg","Starters"),

("Lamb Tikka","4.00","Non-Veg","Starters"),

("Tandoori King Prawn","6.95","Non-Veg","SeaFood"),

("King Prawn Rosun","5.95","Non-Veg","SeaFood"),

("King Prawn on Puree","5.95","Non-Veg","SeaFood"),

("Prawn Bhuna on Puree","3.95","Non-Veg","SeaFood"),

("Prawn Cocktail","3.95","Non-Veg","SeaFood"),

("Chi/Lam Sashlik Chi","9.95","Non-Veg","Tandoori Specials"),

("Tandoori Deluxe","12.95","Non-Veg","Tandoori Specials"),

("Tandoori Chicken Main","9.95","Non-Veg","Tandoori Specials"),

("Boiled Rice","2.50","Veg","Side Orders - Rice"),

("Keema Pilau Rice","3.50","Non-Veg","Side Orders - Rice"),

("Mushroom Rice","3.20","Veg","Side Orders - Rice"),

("Garlic Naan","3.00","Veg","Side Orders - Bread"),

("Stuffed Naan","3.50","Veg","Side Orders - Bread"),

("Chapati","1.00","Veg","Side Orders - Bread"),

("Green Salad","2.00","Veg","Side Orders - Sundries"),

("Spice Popadum","0.80","Veg","Side Orders - Sundries"),

("Chutney","0.70","Veg","Side Orders - Sundries"),

("Ras Malai","2.95","Veg","Dessert"),

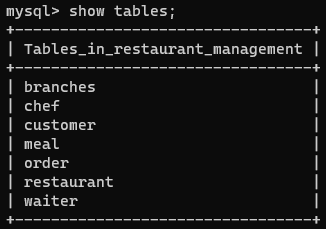
("Ice Cream","2.95","Veg","Dessert"),

("Gulab Jamun","2.95","Veg","Dessert"),

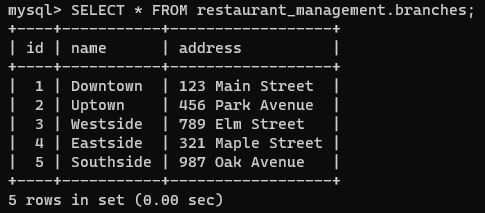
("Kulfi","2.95","Veg","Dessert"),

("Kheer","2.95","Veg","Dessert");

Show tables;



Select \* from restaurant\_management.branch;



Select \* from restaurant\_management.branches;

INSERT INTO branches (name, address) VALUES

('Downtown', '123 Main Street'),

('Uptown', '456 Park Avenue'),

('Westside', '789 Elm Street');

('Eastside', '321 Maple Street'),

('Southside', '987 Oak Avenue');



Select \* from restaurant\_management.chef;

INSERT INTO chef (name, branch\_id) VALUES

('John Smith', 1),

('Sarah Lee', 2),

('David Kim', 3);

('Jessica Brown', 1),

('Tyler Nguyen', 2),

('Hannah Kim', 3),

('Oliver Lee', 4);



Select \* from restaurant\_management.mela;

INSERT INTO meal (name, price) VALUES

('Hamburger', 9.99),

('Cheese Pizza', 12.99),

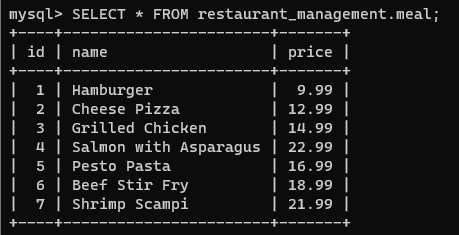
('Grilled Chicken', 14.99);

('Salmon with Asparagus', 22.99),

('Pesto Pasta', 16.99),

('Beef Stir Fry', 18.99),

('Shrimp Scampi', 21.99);



Select \* from restaurant\_management.customer;

INSERT INTO customer (name, email, phone) VALUES

('Jane Doe', 'jane@example.com', '555-1234'),

('Bob Smith', 'bob@example.com', '555-5678'),

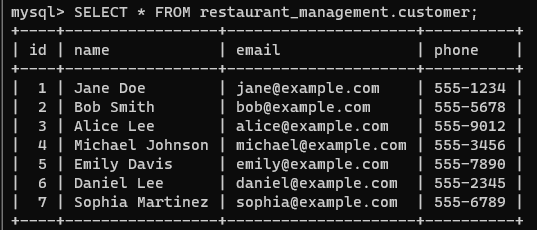
('Alice Lee', 'alice@example.com', '555-9012');

('Michael Johnson', 'michael@example.com', '555-3456'),

('Emily Davis', 'emily@example.com', '555-7890'),

('Daniel Lee', 'daniel@example.com', '555-2345'),

('Sophia Martinez', 'sophia@example.com', '555-6789');



Select \* from restaurant\_management.order;

INSERT INTO `order` (customer\_id, restaurant\_id, waiter\_id, meal\_id, quantity, total\_price) VALUES

(1, 1, 1, 1, 2, 19.98),

(2, 2, 2, 2, 1, 12.99),

(3, 3, 3, 3, 3, 44.97);

(1, 1, 1, 1, 1, 22.99),

(2, 2, 2, 2, 2, 33.98),

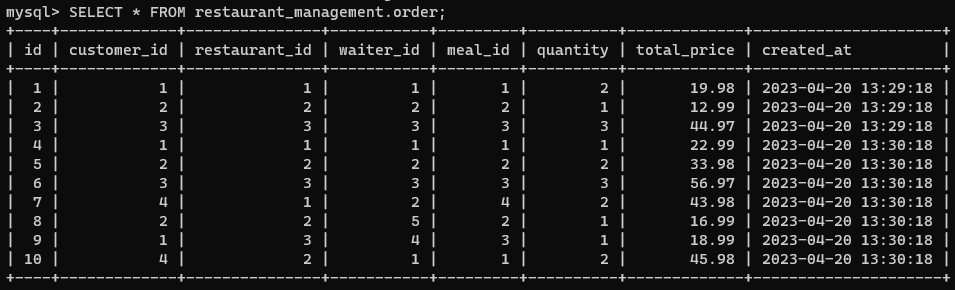
(3, 3, 3, 3, 3, 56.97),

(4, 1, 2, 4, 2, 43.98),

(2, 2, 5, 2, 1, 16.99),

(1, 3, 4, 3, 1, 18.99),

(4, 2, 1, 1, 2, 45.98);



INSERT INTO restaurant (name, address) VALUES

('Joe\'s Diner', '456 Main Street'),

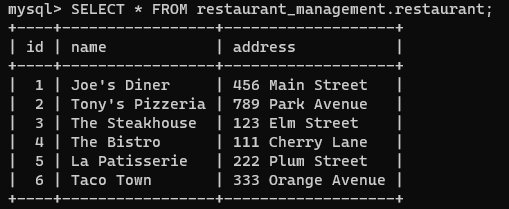
('Tony\'s Pizzeria', '789 Park Avenue'),

('The Steakhouse', '123 Elm Street');

('The Bistro', '111 Cherry Lane'),

('La Patisserie', '222 Plum Street'),

('Taco Town', '333 Orange Avenue');



INSERT INTO waiter (name, branch\_id) VALUES

('Rachel Kim', 1),

('Aiden Brown', 2),

('Leah Nguyen', 3),

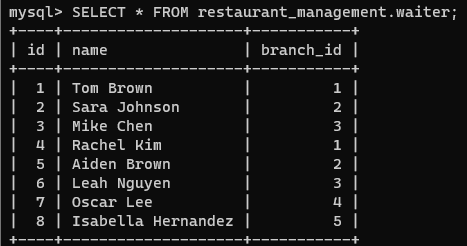
('Oscar Lee', 4),

('Isabella Hernandez', 5);

('Tom Brown', 1),

('Sara Johnson', 2),

('Mike Chen', 3);



**Trigger:**

Price

CREATE TRIGGER update\_totla\_price

AFTER INSERT ON ‘other’

FOR EACH ROW

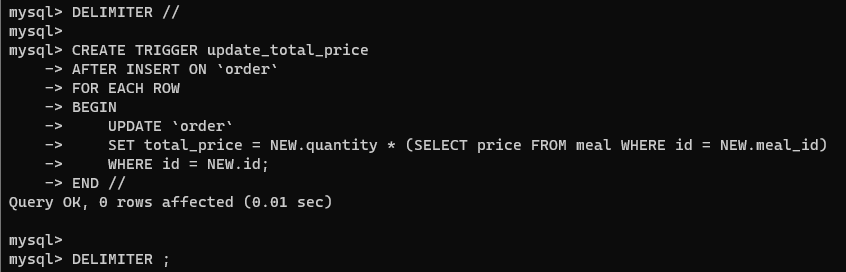
BEGIN

UPDATE ‘order’

SET total\_price = NEW.quantity \* (SELECT price FROM meal WHERE id = NEW.meal\_id)

WHERE ID = NEW.ID;

END//



**Insert:**

Create Trigger Loan Detail On Loan Instead Of Insert

As Begin

    Declare Name Char(128);

    Declare L\_Id Int;

    Declare Acc\_No  Int;

Delcare Balance Int;

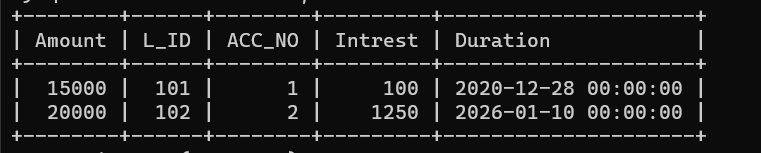
    Select Name = Name, L\_Id = L\_Id, Acc\_No = Acc\_No,

Balance=Balance

From Inserted;

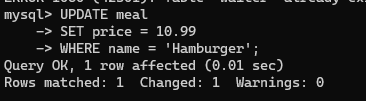
    If Name Is Null Set Name = Name;

    Insert Into Loan (Name, L\_Id, Acc\_No,Balance) Values (Name, L\_Id, Acc\_No, Balance);

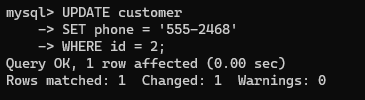
End;

**Update:**

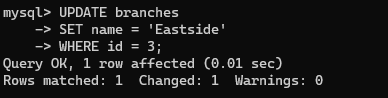
Update the price of the 'Hamburger' meal to $10.99:

****

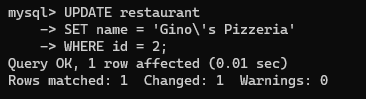
Update the phone number of the customer with id 2:



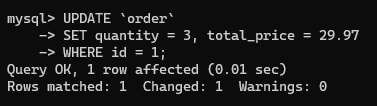
Update the name of the branch with id 3 to 'Eastside':



Update the name of the restaurant with id 2 to 'Gino's Pizzeria':

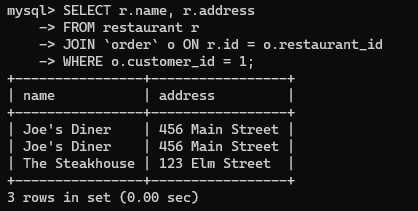


Update the quantity and total price of the order with id 1:

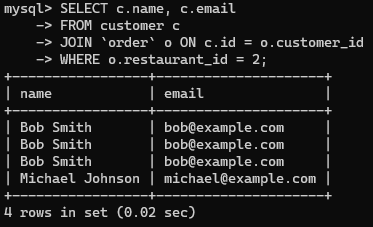


**JOIN QUERIES:**

1. Retrieve the name and address of the restaurant where the customer with id 1 placed an order:

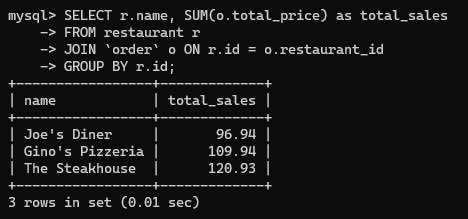
****

1. Retrieve the name and email of the customer who placed an order at the restaurant with id 2:

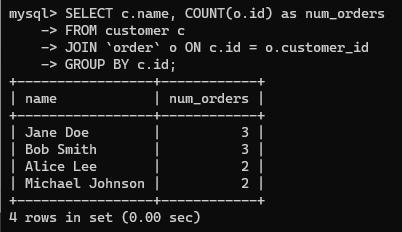
****

**Group by queries**

1. Retrieve the total sales for each restaurant:

****

1. Retrieve the total number of orders placed by each customer:



**CHAPTER 4**

**NORMALIZATION**

Normalization is a process in database management that helps to eliminate redundancy and data inconsistencies in a relational database. It involves breaking down a large table into smaller, more manageable tables, and establishing relationships between them. The goal of normalization is to ensure that each piece of data is stored in only one place in the database so that there are no duplicate entries that can cause problems with data integrity, consistency, and accuracy. By normalizing a database, you can improve its performance, make it easier to maintain and update and reduce the risk of data errors and inconsistencies. However, normalization can also result in more complex database designs and queries, so it's important to strike a balance between normalization and ease of use.

1NF:

A relation violates the first normal form if it contains a composite or multi-valued attribute, or the relation is in the first ordinary form if it does not contain any composite or multi-valued attribute. If each characteristic in a relation is an independently valued attribute, the connection is in its initial normal form.

select \* from Bank..Balance

--where continent is not null

order by ACC\_NO;

**Original table:**

| **Patent ID** | **Inventor Name** | **Invention Title** | **Invention Description** | **Filing Date** | **Patent Status** |
| --- | --- | --- | --- | --- | --- |
| 1234567 | John Smith | Widget | This invention is... | 01/01/2022 | Issued |
| 1234568 | Jane Doe | Gizmo | This invention is... | 01/02/2022 | Pending |
| 1234569 | John Smith | Doodad | This invention is... | 01/03/2022 | Issued |

**First normal form:**

| **Patent ID** | **Inventor ID** | **Invention Title** | **Invention Description** | **Filing Date** | **Patent Status** |
| --- | --- | --- | --- | --- | --- |
| 1234567 | 1 | Widget | This invention is... | 01/01/2022 | Issued |
| 1234568 | 2 | Gizmo | This invention is... | 01/02/2022 | Pending |
| 1234569 | 1 | Doodad | This invention is... | 01/03/2022 | Issued |

First Normal Form (1NF) - This form requires that each column in a table contains atomic (indivisible) values, meaning that each column should only contain a single value. Additionally, each row in the table should be unique, which is typically achieved through the use of a primary key.

In this first normalized table, we have removed the repeated information of Inventor Name and created an Inventor ID column. This prevents the possibility of having inconsistent or duplicated data.

**Second normal form:**

The premise of complete functional interdependence underpins the Second Normal Form (2NF). The second Normal Form is used for relations with composite keys, which are the primary keys composed up of two or more characteristics. A relation with a single-attribute primary key is in at least 2NF by defaults. The update anomalies might impact a relationship that is not in 2NF.

Select Address As Loc, DOB As Date, Sum(Cast(Balance As Int)) As Grand

Into Nf2

From Bank..Balance

Order By Balance;

Select \* From Nf2

| **Patent ID** | **Inventor ID** | **Invention Title** | **Invention Description** | **Filing Date** |
| --- | --- | --- | --- | --- |
| 1234567 | 1 | Widget | This invention is… | 01/01/2022 |
| 1234568 | 2 | Gizmo | This invention is… | 01/02/2022 |
| 1234569 | 1 | Doodad | This invention is… | 01/03/2022 |

| **Inventor ID** | **Inventor Name** |
| --- | --- |
| 1 | John Smith |
| 2 | Jane Doe |

Second Normal Form (2NF) – This form requires that all non-key attributes in a table are fully dependent on the primary key. In other words, each non-key attribute should be uniquely identified by the primary key, and there should be no redundancy in the table. To achieve this, tables can be split into multiple tables, with each table containing a subset of the original data.

**Third normal form:**

If there is no transitive dependency for non-prime characteristics and the relation is in second normal form, it is in third normal form.

In any non-trivial function dependence X -> Y, a relation is in 3NF if at least one of the following conditions holds:

The letter X is a super key.

Y is a prime attribute (every element of Y is a candidate key).

select Duration as date, Amount as Loan

into NF3

from NF2

select \* from NF3;

| **Patent ID** | **Inventor ID** | **Invention Title** | **Invention Description** | **Filing Date** |
| --- | --- | --- | --- | --- |
| 1234567 | 1 | Widget | This invention is... | 01/01/2022 |
| 1234568 | 2 | Gizmo | This invention is... | 01/02/2022 |
| 1234569 | 1 | Doodad | This invention is... | 01/03/2022 |

| **Inventor ID** | **Inventor Name** |
| --- | --- |
| 1 | John Smith |
| 2 | Jane Doe |

| **Patent ID** | **Status** |
| --- | --- |
| 1234567 | Issued |
| 1234568 | Pending |
| 1234569 | Issued |

Third Normal Form (3NF) - This form requires that all non-key attributes in a table are independent of each other. In other words, there should be no transitive dependencies, where one non-key attribute can be determined by another non-key attribute. To achieve this, tables can be further split into multiple tables, with each table containing only the data necessary to describe the entity being modeled.

Returning to the patent table example, in the original table, there was redundancy in the Inventor Name column, which violated 1NF. To normalize the table, we created a separate table for Inventor Name and assigned an Inventor ID to each inventor. This satisfies 1NF.

In the second normalized table, we identified the primary key (Patent ID) and removed redundancy by separating the data into two tables: one for patents and one for inventors. This satisfies 2NF.

In the third normalized table, we further separated the data into three tables to eliminate transitive dependencies. The Status column was moved to its own table, as it is not dependent on any other attribute in the table. This satisfies 3NF.

Overall, normalization helps to improve data quality and consistency, prevent data anomalies, and improve performance by reducing the amount of data that needs to be processed.

**NoSQL**

NoSQL is termed non-SQL. We won’t use SQL to write queries in No SQL. It is not relational but it gives us an organized way of storing data. The data is stored in the form of documents rather than tabular form. The best example for NoSql is Mongo DB. In SQL we will use the term key-value pairs but in Mongo DB we will use field-value pairs. Documents are stored and the group of documents is called “Collection”. The document will be in JSON format. The data is called a “Document” and the collection of documents is called a “Collection”.

Queries:

Creation:

db.covid.insertMany([

{id: 1, name: 'America', deaths: '4,47,68,172' },

{id: 2, name: 'India', deaths: '10,28,73,924' },

{id: 3, name: 'Argentina', deaths: '1,00,44,957' }

]);

db.covid.find({name: 'India'});



Insertion:

db.covid.insertMany([

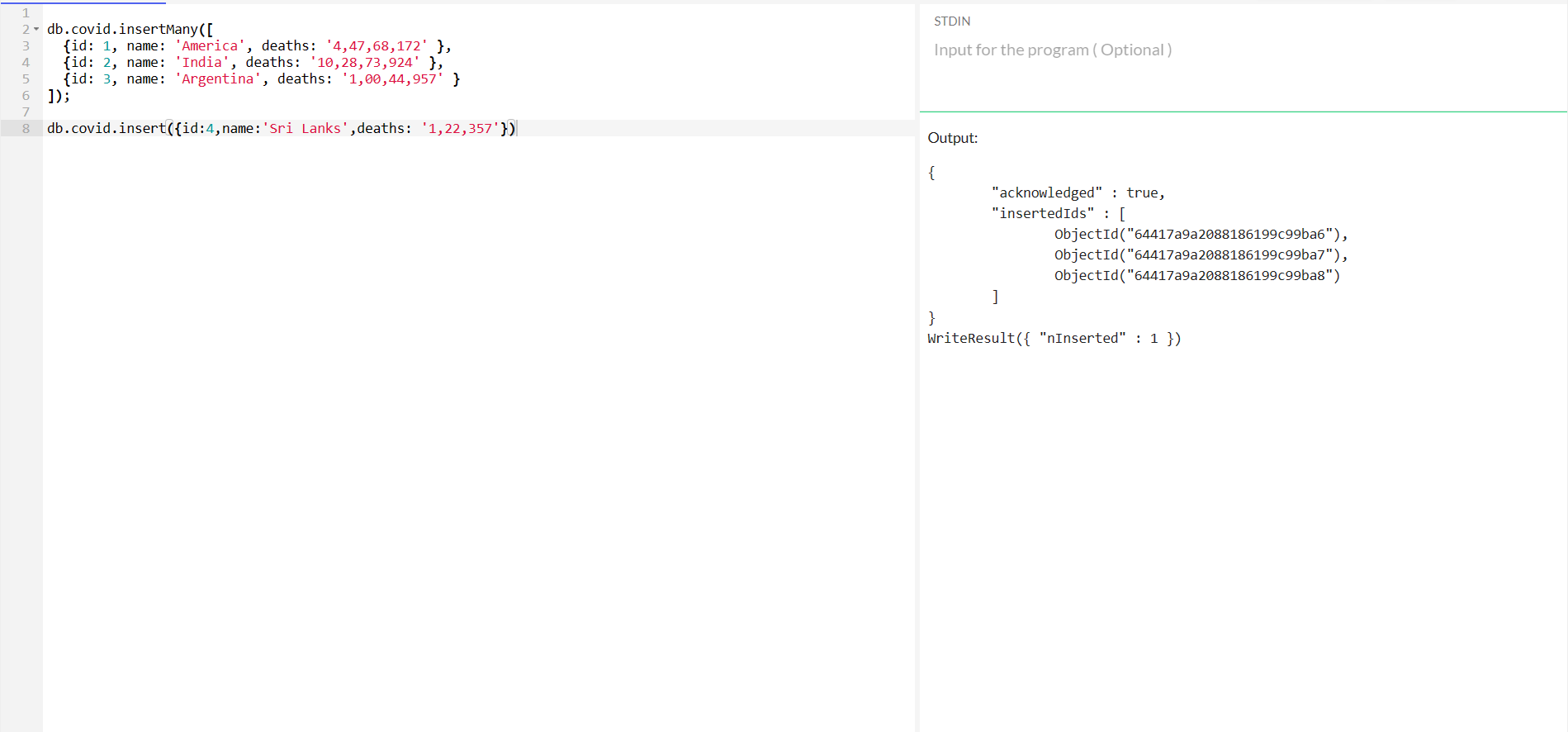
{id: 1, name: 'America', deaths: '4,47,68,172' },

{id: 2, name: 'India', deaths: '10,28,73,924' },

{id: 3, name: 'Argentina', deaths: '1,00,44,957' }

]);

db.covid.insert({id:4,name:'Sri Lanks',deaths: '1,22,357'})



Selection:

db.covid.insertMany([

{id: 1, name: 'America', deaths: '4,47,68,172' },

{id: 2, name: 'India', deaths: '10,28,73,924' },

{id: 3, name: 'Argentina', deaths: '1,00,44,957' }

]);

db.covid.find()



**CHAPTER 5**

**CONCLUSION**

A bank administration system is the virtualization of transactions in an accounting system. The banking system is manual, but when we use an online banking system, it is completely virtualized, avoiding traditional procedures and converting them to automatic processes. If a user can conduct a transaction in the bank management system, it is available at any location. Furthermore, the user may quickly link their Aadhar card to their account and change their branch location. Bank management systems save time and are more reliable than manual banking procedures. The Banking System is a Database Application system that is more efficient, quick, dependable, and user pleasant. Furthermore, there is no risk of data loss during processing using the suggested approach. This financial institution project will be a good method for a database conversation box for the user to make deposits and withdrawals of cash. It is a beneficial method for users. It makes it simple to deposit and withdraw money. It shortens the time it takes the consumer to save money. With usage this system, it provides every necessary data to management as well as the consumer; the user can simply sit in front of the system and observe all operations without any physical movement of the file. Management may respond to client requests as quickly as practicable. As a result, the project takes a user-friendly approach.

**CHAPTER 6**

**Future Enhancement**

This project can further be developed in multiple other aspects such as:

Improved security features: One potential enhancement would be to add more advanced security features to the database, such as two-factor authentication, biometric authentication, and encryption of sensitive data. Mobile app integration-Many banks are now offering mobile apps for their customers, allowing them to access their accounts, make transactions, and perform other banking tasks from their smartphones. Integrating a mobile app with the database could be a useful enhancement. Automated fraud detection: Fraud is a major concern for banks, and incorporating automated fraud detection algorithms into the database could help identify and prevent fraudulent transactions. Data analytics and reporting-Collecting and analysing data on customer behavior, transaction patterns, and other metrics can provide valuable insights for banks. Enhancing the database to support data analytics and reporting could help banks make more informed decisions. Multi-currency support: Banks that operate in multiple countries may need to support multiple currencies. Enhancing the database to support multiple currencies could make it easier to manage international transactions. Integration with third-party services: Many banks now offer services such as investment management, insurance, and mortgages. Integrating the database with third-party services could make it easier to manage these services and provide a more seamless experience for customers.

**CHAPTER 7**

**REFERENCES**

1. Toast POS: This is a popular restaurant management system that offers features such as menu management, tableside ordering, and inventory tracking.
2. Square for Restaurants: This is another comprehensive restaurant management system that offers features such as table management, online ordering, and employee management.
3. Upserve: This restaurant management system offers features such as online ordering, inventory management, and tableside ordering.
4. OpenTable: This restaurant management system is popular for its table management and reservation system.

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the

productivity

FUTURE DEVELOPMENTS

The main dispatcher system software may have a few areas to

improve on in the future. The current system allows anyone to view and modify

the database. Adding a user profiles with password will improve the overall

security of the system. As for eccentric feature, colour code different request

status, request type, or waiter id would make the request table easier to read.

In the future, it would be efficient to join this main dispatcher software to the

existing ordering software that restaurants use today to increase the