SVKM's NMIMS
School of Technology Management & Engineering, Navi Mumbai
A.Y. 2023 - 24
Course: Database Management Systems

Project Report

Program	B. Tech Computer Engineering	
Semester	Semester IV	
Name of the Project:	Restaurant Management	
Details of Project Members		
Batch	Roll No.	Name
2	A113	Aayush Singh
2	A121	Aditya Ladge
Date of Submission: 02/04/2024		

Contribution of each project Members:

Roll No.	Name:	Contribution
A113	Aayush Singh	Created database, backend contribution, optimized queries, and report.
A121	Aditya Ladge	Created ER Diagram, Relational Model, Normalization, and report

GitHub link of your project:

Note:

1. Create a readme file if you have multiple files
2. All files must be properly named (Example:R004_DBMSProject)
3. Submit all relevant files of your work (Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

Rubrics for the Project evaluation:

First phase of evaluation: Innovative Ideas (5 Marks) Design and Partial implementation (5 Marks)	10 marks
Final phase of evaluation Implementation, presentation and viva, Self-Learning and Learning Beyond classroom	10 marks

Project Report:

RESTAURANT MANAGEMENT

By

Aayush Singh, A113: 70022200285

Aditya Ladge, A121: 70022200297

Course: DBMS

AY: 2023-24

Table of Contents

Storyline:

The comprehensive database architecture encompasses tables for customers, restaurants, cart management, order details, and menu items.

Customers are provided a personalized experience through dedicated tables for each restaurant they engage with, ensuring tailored service and efficient order management. Meanwhile, restaurants are empowered with streamlined operations, with detailed records of customer orders and menu offerings.

The "customer" table captures essential customer information, including name, contact number, and address. Simultaneously, the "restaurant" table centralizes restaurant data, featuring unique identifiers and names for easy reference.

Our cart management system, represented by the "combined_cart" and respective "restaurantX_cart" tables, facilitates smooth transactions, enabling customers to add items from various menus seamlessly.

Moreover, the "combined_order_details" and "restaurantX_order_details" tables meticulously record order specifics, ensuring accuracy in transactions and allowing for comprehensive analytics.

Components of Database Design

Customer:
Attributes: customer_id (Primary Key), name, number, address

Restaurant:
Attributes: restaurant_id (Primary Key), name

Combined Cart:
Attributes: cart_id (Primary Key), customer_id (Foreign Key referencing customer(customer_id)), restaurant_id (Foreign Key referencing restaurant(restaurant_id)), item_id (Foreign Key referencing restaurant_menu(item_id)), quantity

Combined Order Details:
Attributes: order_id (Primary Key), customer_id (Foreign Key referencing customer(customer_id)), restaurant_id (Foreign Key referencing restaurant(restaurant_id)), total_amount

Restaurant Menu:
Attributes: item_id (Primary Key), restaurant_id (Foreign Key referencing restaurant(restaurant_id)), item_name, price

Restaurant Customer (restaurantX_customer):
Attributes: customer_id (Primary Key), name, number, address

Restaurant Menu (restaurantX_menu):
Attributes: menu_id (Primary Key), item_name, price

Restaurant Cart (restaurantX_cart):
Attributes: cart_id (Primary Key), customer_id (Foreign Key referencing restaurantX_customer(customer_id)), item_id (Foreign Key referencing restaurantX_menu(menu_id)), quantity

Restaurant Order Details (restaurantX_order_details):
Attributes: order_id (Primary Key), customer_id (Foreign Key referencing restaurantX_customer(customer_id)), total_amount

Relationships among various entities:

Customer - Combined Cart (One-to-Many):
A customer can have multiple items in their combined cart (participation: mandatory for customer, optional for combined cart).

Customer - Combined Order Details (One-to-Many):
A customer can have multiple orders in combined order details (participation: mandatory for customer, optional for combined order details).

Restaurant - Restaurant Menu (One-to-Many):
A restaurant can have multiple menu items (participation: mandatory for restaurant, optional for restaurant menu).

Restaurant - Combined Cart (One-to-Many):
A restaurant can have multiple carts (participation: mandatory for restaurant, optional for combined cart).

Restaurant - Combined Order Details (One-to-Many):
A restaurant can have multiple orders (participation: mandatory for restaurant, optional for combined order details).

Customer - Restaurant Customer (One-to-One):
Each customer can belong to only one restaurant (participation: mandatory for both customer and restaurant customer).

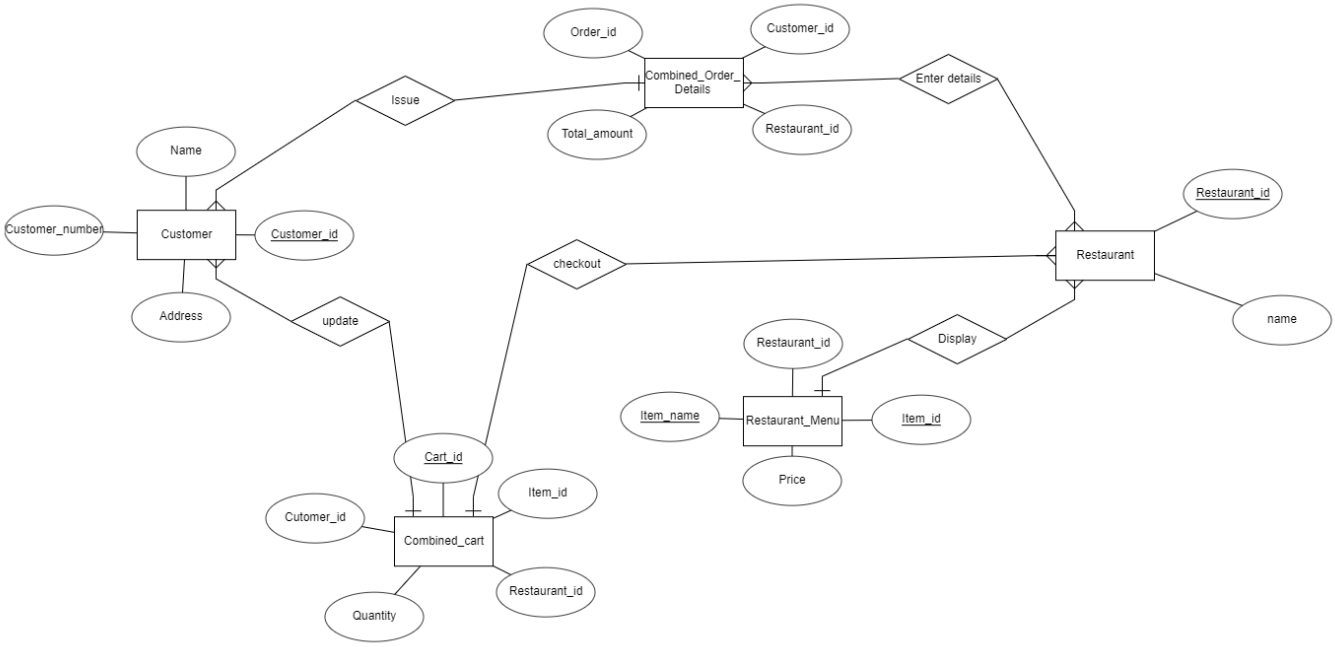
Restaurant - Restaurant Menu (One-to-Many):
A restaurant can have multiple menu items (participation: mandatory for restaurant, optional for restaurant menu).

Restaurant Customer - Restaurant Cart (One-to-Many):
A restaurant customer can have multiple items in their cart (participation: mandatory for restaurant customer, optional for restaurant cart).

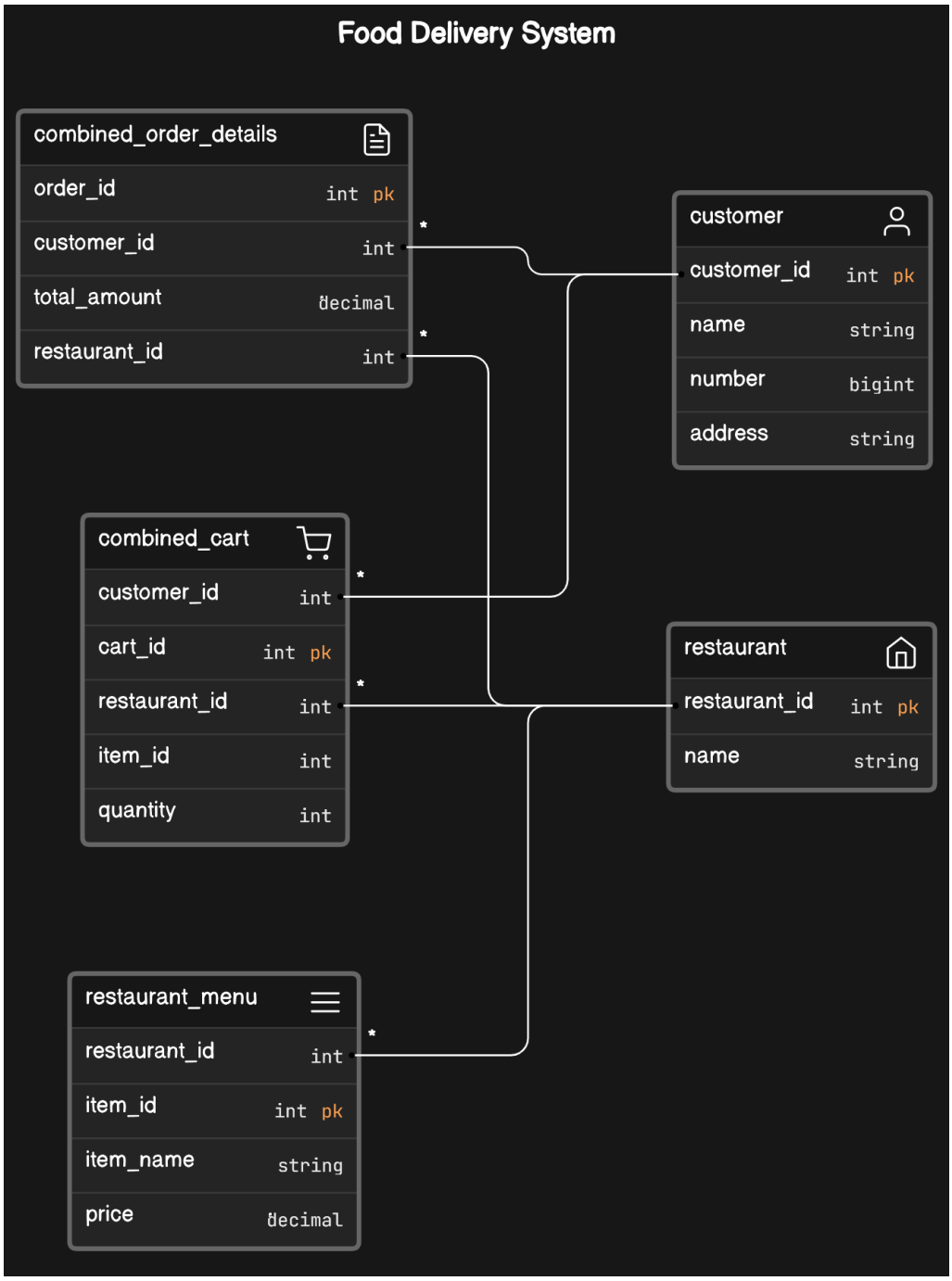
Restaurant Customer - Restaurant Order Details (One-to-Many):
A restaurant customer can have multiple orders (participation: mandatory for restaurant customer, optional for restaurant order details).

.

Entity Relationship Diagram



Relational Model:



Normalization

- **Table:** combined_cart
  1NF: Already in 1NF.
  2NF: No partial dependencies.
  3NF: Already in 3NF.
  BCNF: Already in BCNF.
- **Table:** combined_order_details
  1NF: Already in 1NF.
  2NF: No partial dependencies.
  3NF: Already in 3NF.
  BCNF: Already in BCNF.
- **Tables:** restaurant_menu, restaurant1_menu, restaurant2_menu, restaurant3_menu, restaurant4_menu, restaurant5_menu
  1NF: Already in 1NF.
  2NF: Already in 2NF.
  3NF: Already in 3NF.
  BCNF: Already in BCNF.
- **Tables:** customer, restaurant, restaurant1_customer, restaurant2_customer, restaurant3_customer, restaurant4_customer, restaurant5_customer
  1NF: Already in 1NF.
  2NF: Already in 2NF.
  3NF: Already in 3NF.
  BCNF: Already in BCNF.
- **Tables:** restaurant1_cart, restaurant2_cart, restaurant3_cart, restaurant4_cart, restaurant5_cart
  for restaurant1_cart
  1NF: Already in 1NF.
  2NF: Already in 2NF.
  3NF: Already in 3NF.
  BCNF: Already in BCNF.
- **Tables:** restaurant1_order_details, restaurant2_order_details, restaurant3_order_details, restaurant4_order_details, restaurant5_order_details
  1NF: Already in 1NF.
  2NF: No partial dependencies.
  3NF: Already in 3NF.
  BCNF: Already in BCNF.

Tables: (combined_cart, combined_order_details, restaurant_menu, customer, restaurant, restaurant1_customer, restaurant2_customer, restaurant3_customer, restaurant4_customer, restaurant5_customer, restaurant1_menu, restaurant2_menu, restaurant3_menu, restaurant4_menu, restaurant5_menu, restaurant1_cart, restaurant2_cart, restaurant3_cart, restaurant4_cart, restaurant5_cart, restaurant1_order_details, restaurant2_order_details, restaurant3_order_details, restaurant4_order_details, restaurant5_order_details)

SQL Queries:

customer (customer_id INT PRIMARY KEY, name VARCHAR(255), number bigint, address varchar(225));
restaurant (restaurant_id INT PRIMARY KEY, name VARCHAR(255));
combined_cart (cart_id INT PRIMARY KEY AUTO_INCREMENT, customer_id INT, restaurant_id INT, item_id INT, quantity INT, FOREIGN KEY (customer_id) REFERENCES customer(customer_id));
combined_order_details (order_id INT PRIMARY KEY AUTO_INCREMENT, customer_id INT, restaurant_id INT, total_amount DECIMAL(10, 2), FOREIGN KEY(customer_id) REFERENCES customer(customer_id));
restaurant_menu (item_id INT PRIMARY KEY AUTO_INCREMENT, restaurant_id INT, item_name VARCHAR(255), price DECIMAL(10, 2), FOREIGN KEY (restaurant_id) REFERENCES restaurant(restaurant_id));
restaurant1_customer (customer_id INT PRIMARY KEY, name VARCHAR(255), number bigint, address varchar(225));
restaurant1_menu (menu_id INT PRIMARY KEY, item_name VARCHAR(255), price DECIMAL(10, 2));
restaurant1_cart (cart_id INT PRIMARY KEY, customer_id INT, item_id INT, quantity INT);
restaurant1_order_details (order_id INT PRIMARY KEY, customer_id INT, total_amount DECIMAL(10, 2));
restaurant2_customer (customer_id INT PRIMARY KEY, name VARCHAR(255), number bigint, address varchar(225));
restaurant2_menu (menu_id INT PRIMARY KEY, item_name VARCHAR(255), price DECIMAL(10, 2));
restaurant2_cart (cart_id INT PRIMARY KEY, customer_id INT, item_id INT, quantity INT);
restaurant2_order_details (order_id INT PRIMARY KEY, customer_id INT, total_amount DECIMAL(10, 2));
restaurant3_customer (customer_id INT PRIMARY KEY, name VARCHAR(255), number bigint, address varchar(225));

```
restaurant3_menu (menu_id INT PRIMARY KEY, item_name VARCHAR(255), price DECIMAL(10, 2));
restaurant3_cart (cart_id INT PRIMARY KEY, customer_id INT, item_id INT, quantity INT);
restaurant3_order_details (order_id INT PRIMARY KEY, customer_id INT, total_amount DECIMAL(10, 2));
restaurant4_customer (customer_id INT PRIMARY KEY, name VARCHAR(255), number bigint, address varchar(225));
restaurant4_menu (menu_id INT PRIMARY KEY, item_name VARCHAR(255), price DECIMAL(10, 2));
restaurant4_cart (cart_id INT PRIMARY KEY, customer_id INT, item_id INT, quantity INT);
restaurant4_order_details (order_id INT PRIMARY KEY, customer_id INT, total_amount DECIMAL(10, 2));
restaurant5_customer (customer_id INT PRIMARY KEY, name VARCHAR(255), number bigint, address varchar(225));
restaurant5_menu (menu_id INT PRIMARY KEY, item_name VARCHAR(255), price DECIMAL(10, 2));
restaurant5_cart (cart_id INT PRIMARY KEY, customer_id INT, item_id INT, quantity INT);
restaurant5_order_details (order_id INT PRIMARY KEY, customer_id INT, total_amount DECIMAL(10, 2));
INSERT INTO restaurant1_customer (customer_id, name, number, address)
VALUES (1, 'Aayush Singh',9834537284,'Dahisar, Mumbai'), (2, 'Aditya Ladge',9385354324,'Vashi, Navi Mumbai'), (3, 'Adarsh Pandey',4999274283,'Kandivali ,Mumbai'), (4, 'Krish Surti',5673845932,'Malad, Mumbai'), (5, 'Tanya Sanyal',4582237438,'Hiranandani, Thane'), (6, 'Jeet Jain',4582647283,'Kharghar, Navi Mumbai'), (7, 'Jayesh Gawde',3532476842,'Malad, Mumbai'), (8, 'Vansh Doshi',4567313531,'Mulund, Mumbai');
select * from restaurant1_customer;
ALTER TABLE restaurant1_menu CHANGE menu_id item_id INT;
INSERT IGNORE INTO customer (customer_id, name, number, address)
SELECT customer_id, name, number, address
FROM restaurant1_customer;
INSERT INTO restaurant1_menu (item_id, item_name, price)
VALUES (1, 'Margherita Pizza', 10.99), (2, 'Pepporani Pizza 2', 12.99), (3, 'Corn and Cheese 3', 8.99);
INSERT INTO restaurant1_cart (cart_id, customer_id, item_id, quantity)
VALUES (2, 1, 2, 1);
INSERT INTO restaurant1_cart (cart_id, customer_id, item_id, quantity)
```

```
VALUES (1, 1, 1, 2);
select * from restaurant1_cart;
INSERT INTO combined_cart (customer_id, restaurant_id, item_id, quantity)
SELECT c.customer_id, 1, r1.item_id, r1.quantity
FROM restaurant1_customer c
JOIN restaurant1_cart r1 ON c.customer_id = r1.customer_id;
INSERT INTO restaurant1_order_details (order_id, customer_id, total_amount)
SELECT c.item_id,  c.customer_id, SUM((m.price * c.quantity)) AS total
FROM restaurant1_cart c
JOIN restaurant1_menu m ON c.cart_id = m.item_id
GROUP BY c.item_id, c.customer_id;
select * from restaurant1_order_details;
INSERT INTO restaurant2_customer (customer_id, name, number, address)
VALUES (1, 'Aayush Singh',9834537284,'Dahisar, Mumbai'), (2, 'Aditya Ladge',9385354324,'Vashi, Navi Mumbai'), (3, 'Adarsh Pandey',7839204024,'Kandivali ,Mumbai'), (4, 'Krish Surti',5673845932,'Malad, Mumbai'), (5, 'Tanya Sanyal',4582237438,'Hiranandani, Thane'), (6, 'Jeet Jain',4582647283,'Kharghar, Navi Mumbai'), (7, 'Jayesh Gawde',3859304234,'Malad, Mumbai'), (8, 'Vansh Doshi',7838492748,'Mulund, Mumbai');
select * from restaurant2_customer;
INSERT IGNORE INTO customer (customer_id, name, number, address)
SELECT customer_id, name, number, address
FROM restaurant2_customer;
ALTER TABLE restaurant2_menu CHANGE menu_id item_id INT;
INSERT INTO restaurant2_menu (item_id, item_name, price)
VALUES (1, 'Veg Burger', 14.99), (2, 'Chicken Burger 2', 9.99);
INSERT INTO restaurant2_cart (cart_id, customer_id, item_id, quantity)
VALUES (2, 6, 2, 2);
INSERT INTO restaurant2_cart (cart_id, customer_id, item_id, quantity)
VALUES (1, 6, 1, 3);
select * from restaurant2_cart;
INSERT INTO combined_cart (customer_id, restaurant_id, item_id, quantity)
SELECT c.customer_id, 2, r2.item_id, r2.quantity
FROM restaurant2_customer c
JOIN restaurant2_cart r2 ON c.customer_id = r2.customer_id;
INSERT INTO restaurant2_order_details (order_id, customer_id, total_amount)
SELECT c.cart_id, c.customer_id, SUM((m.price * c.quantity)) AS total
FROM restaurant2_cart c
JOIN restaurant2_menu m ON c.item_id = m.item_id
JOIN restaurant2_customer cus ON c.customer_id = cus.customer_id
```

```
GROUP BY c.cart_id, c.customer_id;
select * from restaurant2_order_details;
INSERT IGNORE INTO customer (customer_id, name, number, address)
SELECT customer_id, name, number, address
FROM restaurant3_customer;
INSERT INTO restaurant3_customer (customer_id, name, number, address)
VALUES (1, 'Aayush Singh',8956783924,'Borivali, Mumbai'), (4, 'Krish Surti',5673845932,'Malad, Mumbai'), (5, 'Tanya Sanyal',4582237438,'Hiranandani, Thane'), (6, 'Jeet Jain',4582647283,'Kharghar, Navi Mumbai'), (7, 'Jayesh Gawde',3532476842,'Malad, Mumbai'), (8, 'Vansh Doshi',4567313531,'Mulund, Mumbai');
select * from restaurant3_customer;
ALTER TABLE restaurant3_menu CHANGE menu_id item_id INT;
INSERT INTO restaurant3_menu (item_id, item_name, price)
VALUES (1, 'Garlic Bread', 7.99), (2, 'Farmhouse Pizza', 11.99), (3, 'Zinger Burger', 9.99);
INSERT INTO restaurant3_cart (cart_id, customer_id, item_id, quantity)
VALUES (8, 4, 2, 1);
INSERT INTO restaurant3_cart (cart_id, customer_id, item_id, quantity)
VALUES (5, 7, 3, 2);
select * from restaurant3_cart;
INSERT INTO combined_cart (customer_id, restaurant_id, item_id, quantity)
SELECT c.customer_id, 3, r3.item_id, r3.quantity
FROM restaurant3_customer c
JOIN restaurant3_cart r3 ON c.customer_id = r3.customer_id;
INSERT Into restaurant3_order_details (order_id, customer_id, total_amount)
SELECT c.cart_id, cus.customer_id, SUM(c.quantity * m.price) AS total
FROM restaurant3_cart c
JOIN restaurant3_menu m ON c.item_id = m.item_id
JOIN restaurant3_customer cus ON c.customer_id = cus.customer_id
GROUP BY c.cart_id, cus.customer_id;
select * from restaurant3_order_details;
INSERT IGNORE INTO customer (customer_id, name, number, address)
SELECT customer_id, name, number, address
FROM restaurant4_customer;
select * from restaurant4_customer;
Select * from customer;
INSERT INTO restaurant4_customer (customer_id, name, number, address)
VALUES (2, 'Aditya Ladge',9385354324,'Vashi, Navi Mumbai'), (3, 'Adarsh Pandey',4999274283,'Kandivali ,Mumbai'), (7, 'Jayesh Gawde',3532476842,'Malad, Mumbai');
select * from restaurant4_customer;
```

```
INSERT IGNORE INTO customer (customer_id, name, number, address)
SELECT customer_id, name, number, address
FROM restaurant4_customer;
ALTER TABLE restaurant4_menu CHANGE menu_id item_id INT;
INSERT INTO restaurant4_menu (item_id, item_name, price)
VALUES (1, 'Cheese and Corn', 8.99), (2, 'Veg Burger', 14.99);
INSERT INTO restaurant4_cart (cart_id, customer_id, item_id, quantity)
VALUES (2, 2, 2, 1);
INSERT INTO restaurant4_cart (cart_id, customer_id, item_id, quantity)
VALUES (1, 3, 1, 2);
select * from restaurant4_cart;
INSERT INTO combined_cart (customer_id, restaurant_id, item_id, quantity)
SELECT c.customer_id, 4, r4.item_id, r4.quantity
FROM restaurant4_customer c
JOIN restaurant4_cart r4 ON c.customer_id = r4.customer_id;
INSERT INTO restaurant4_order_details (order_id, customer_id, total_amount)
SELECT c.item_id,  c.customer_id, SUM((m.price * c.quantity)) AS total
FROM restaurant4_cart c
JOIN restaurant4_menu m ON c.cart_id = m.item_id
GROUP BY c.item_id, c.customer_id;
select * from restaurant4_order_details;
INSERT INTO restaurant5_customer (customer_id, name, number, address)
VALUES (1, 'Aayush Singh',9834537284,'Dahisar, Mumbai'), (5, 'Tanya Sanyal',4582237438,'Hiranandani, Thane'), (6, 'Jeet Jain',4582647283,'Kharghar, Navi Mumbai');
select * from restaurant5_customer;
ALTER TABLE restaurant5_menu CHANGE menu_id item_id INT;
INSERT INTO restaurant5_menu (item_id, item_name, price)
VALUES (1, 'Garlic Bread', 7.99), (2, 'Margherita Pizza', 10.99), (3, 'Farmhouse Pizza', 11.99);
INSERT INTO restaurant5_cart (cart_id, customer_id, item_id, quantity)
VALUES (2, 1, 1, 1);
INSERT into restaurant5_cart (cart_id, customer_id, item_id, quantity)
VALUES (3, 5, 3, 2);
select * from restaurant5_cart;
INSERT INTO combined_cart (customer_id, restaurant_id, item_id, quantity)
SELECT c.customer_id, 5, r5.item_id, r5.quantity
FROM restaurant5_customer c
JOIN restaurant5_cart r5 ON c.customer_id = r5.customer_id;
select * from restaurant5_cart;
```

```
INSERT INTO restaurant5_order_details (order_id, customer_id, total_amount)
SELECT c.item_id,  c.customer_id, SUM((m.price * c.quantity)) AS total
FROM restaurant5_cart c
JOIN restaurant5_menu m ON c.cart_id = m.item_id
GROUP BY c.item_id, c.customer_id;
select * from restaurant5_order_details;
INSERT INTO combined_order_details (customer_id, restaurant_id, total_amount)
SELECT customer_id, 1 AS restaurant_id, total_amount
FROM restaurant1_order_details;
INSERT INTO combined_order_details (customer_id, restaurant_id, total_amount)
SELECT customer_id, 2 AS restaurant_id, total_amount
FROM restaurant2_order_details;
INSERT INTO combined_order_details (customer_id, restaurant_id, total_amount)
SELECT customer_id, 3 AS restaurant_id, total_amount
FROM restaurant3_order_details;
INSERT INTO combined_order_details (customer_id, restaurant_id, total_amount)
SELECT customer_id, 4 AS restaurant_id, total_amount
FROM restaurant4_order_details;
INSERT INTO combined_order_details (customer_id, restaurant_id, total_amount)
SELECT customer_id, 5 AS restaurant_id, total_amount
FROM restaurant5_order_details;
select * from combined_order_details;
INSERT INTO restaurant (restaurant_id, name)
VALUES (1, 'Restaurant 1'), (2, 'Restaurant 2'), (3, 'Restaurant 3'), (4, 'Restaurant 4'), (5,
'Restaurant 5');
INSERT INTO restaurant_menu (restaurant_id, item_name, price)
SELECT 1 AS restaurant_id, item_name, price
FROM restaurant1_menu
UNION
SELECT 2 AS restaurant_id, item_name, price
FROM restaurant2_menu
UNION
SELECT 3 AS restaurant_id, item_name, price
FROM restaurant3_menu
UNION
SELECT 4 AS restaurant_id, item_name, price
FROM restaurant4_menu
UNION
SELECT 5 AS restaurant_id, item_name, price
FROM restaurant5_menu;
```

```
SELECT c.name, c.number, c.address,
    cod.order_id, cod.restaurant_id, r.name AS restaurant_name,
    m.item_name, cc.quantity, cod.total_amount
FROM customer c
INNER JOIN combined_order_details cod ON c.customer_id = cod.customer_id
INNER JOIN restaurant r ON cod.restaurant_id = r.restaurant_id
INNER JOIN combined_cart cc ON cod.order_id = cc.cart_id
INNER JOIN restaurant_menu m ON cc.item_id = m.item_id
WHERE c.customer_id = 5;
SELECT c.name, c.number, c.address,
    cod.order_id, cod.restaurant_id, r.name AS restaurant_name,
    m.item_name, cc.quantity, cod.total_amount
FROM customer c
INNER JOIN combined_order_details cod ON c.customer_id = cod.customer_id
INNER JOIN restaurant r ON cod.restaurant_id = r.restaurant_id
INNER JOIN combined_cart cc ON cod.order_id = cc.cart_id
INNER JOIN restaurant_menu m ON cc.item_id = m.item_id
WHERE c.name = 'Aayush Singh';
UPDATE restaurant SET name = 'Restaurant 5' WHERE restaurant_id = 3;
SELECT c.name AS customer_name, c.number, c.address,
    cod.order_id, r.name AS restaurant_name,
    m.item_name, cc.quantity, cod.total_amount
FROM customer c
INNER JOIN combined_order_details cod ON c.customer_id = cod.customer_id
INNER JOIN restaurant r ON cod.restaurant_id = r.restaurant_id
INNER JOIN combined_cart cc ON cod.order_id = cc.cart_id
INNER JOIN restaurant_menu m ON cc.item_id = m.item_id
WHERE cod.restaurant_id = 2;
UPDATE customer
SET name = 'Jeet Sharma'
WHERE customer_id = 6;
DELETE FROM combined_order_details WHERE order_id = 3;
select * from combined_order_details;
INSERT INTO restaurant1_cart (cart_id, customer_id, item_id, quantity)
VALUES (3, 2, 2, 3);
INSERT INTO combined_cart (customer_id, restaurant_id, item_id, quantity)
VALUES (2, 1, 2, 3);
INSERT INTO combined_order_details (order_id, customer_id, restaurant_id,
total_amount)
VALUES (3, 2, 1,
```

```
   (SELECT SUM((m.price * c.quantity))
    FROM restaurant1_cart c
    JOIN restaurant1_menu m ON c.item_id = m.item_id
    WHERE c.customer_id = 2 AND c.cart_id = 3)
);
INSERT INTO restaurant1_order_details (order_id, customer_id, total_amount)
VALUES (3, 2,
   (SELECT SUM((m.price * c.quantity))
    FROM restaurant1_cart c
    JOIN restaurant1_menu m ON c.item_id = m.item_id
    WHERE c.customer_id = 2 AND c.cart_id = 3)
);
select * from restaurant1_order_details;
select * from combined_order_details;
SELECT c.name, c.number, c.address,
      cod.order_id, cod.restaurant_id, r.name AS restaurant_name,
      m.item_name, cc.quantity, cod.total_amount
FROM customer c
INNER JOIN combined_order_details cod ON c.customer_id = cod.customer_id
INNER JOIN restaurant r ON cod.restaurant_id = r.restaurant_id
INNER JOIN combined_cart cc ON cod.order_id = cc.cart_id
INNER JOIN restaurant_menu m ON cc.item_id = m.item_id
WHERE c.name = 'Aditya Ladge';
```

Project demonstration

- MySQL Workbench

Self -Learning beyond classroom

- Hands-on experience in database management, including SQL queries and relational database management.
- Sharpening problem-solving skills through encounters with data inconsistencies, debugging code, and optimizing queries.
- Financial management skills, including order totals, budget management, and expense analysis.
- Enhancing critical thinking abilities through data analysis and data-driven decision-making.
- Adaptability and resilience to setbacks and unexpected changes.

- Enhancing attention to detail through managing data accuracy and maintaining database integrity.
- Real-world data modeling, translating a real-world scenario into a relational database schema.
- Understanding data normalization principles to prevent data redundancy and inconsistencies.
- Handling data manipulation at scale, managing and querying a large database.
- Demonstrating integration and aggregation skills, combining data from separate tables and calculating total order amounts.

Learning from the Project

Overall, this project provides a solid foundation in database management and design principles, which are essential skills for anyone working with databases in various domains such as software development, data analysis, and business intelligence.

Learned about pre-normalized databases.

Learned about cardinality (one-to-one, one-to-many, many-to-many) and participation (mandatory vs. optional) in relationships between entities.

Learned about basic database operations such as inserting, updating, deleting, and querying data.

Learned about the importance of maintaining data integrity through the use of primary keys, foreign keys, and constraints.

Become familiar with SQL syntax for creating tables, defining constraints, and establishing foreign key relationships.

Learned about the normalization process, including converting data into various normal forms (1NF, 2NF, 3NF, BCNF), which helps to minimize redundancy and ensure data integrity.

Challenges Faced

- Data Consistency and Integrity: Ensuring data consistency across multiple tables, especially with large datasets, requires careful planning and implementation.

- Query Optimization: Writing efficient SQL queries to retrieve and manipulate data from normalized tables is essential for maintaining system performance.
- Data Migration and Integration: Integrating data from multiple sources and migrating existing data into the new database schema can pose challenges.
- Documentation and Maintenance: Documenting the database design, schema, and business rules is essential for system understanding and maintenance.
- Data Normalization: Identifying and reducing duplicated information across tables and determining normalization level.
- ER Diagram Challenges: Capturing all entities and their relationships. Defined relationships between entities.
- Relational Model Challenges: Choosing appropriate data types for attributes. Defining constraints to ensure data consistency and prevent invalid entries.

Conclusion

In conclusion, this SQL project has provided a comprehensive exploration of database management and design principles. Through the creation of tables, establishment of relationships, and application of normalization techniques, valuable insights have been gained into organizing and managing data effectively.

Key takeaways from the project include a deeper understanding of SQL syntax, normalization techniques, and the importance of maintaining data integrity through primary keys, foreign keys, and constraints. Additionally, the project has enhanced proficiency in entity-relationship modeling, cardinality, and participation in relationships.

By practicing database operations such as inserting, updating, deleting, and querying data, practical skills have been honed that are essential in various domains such as software development, data analysis, and business intelligence.