**SVKM’s NMIMS**

**School of Technology Management & Engineering (Indore Campus)**

**Computer Engineering Department (B Tech/MBATech CE and B Tech AIDS Sem IV)**

**Database Management System**

**Project Report**

|  |  |  |
| --- | --- | --- |
| Program: | BTech CE | |
| Semester | IV | |
| Name of the Project: | Flex Fusion: Fitness website database system | |
|  | | |
| Details of Project Members |  |  |
| Batch | Roll No. | Name |
| 2 | D077 | Samriddhi Motiani |
| 2 | D041 | Khyati Anand |
| 2 | D082 | Shubh Sethi |
| Date of Submission: 12 April 2025 | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| Roll No. | Name: | Contribution |
| D077 | Samriddhi Motiani |  |
| D041 | Khyari Anand |  |
| D082 | Shubh Sethi |  |

**Project Report**

**Flex Fusion-Fitness website database system**

by

Samriddhi Motiani, Roll number: D077

Khyati Anand, Roll number: D041

Shubh Sethi, Roll number: D082

Course: DBMS

AY: 2024-25

|  |
| --- |
| **Table of Contents** |
|  |
| **Sr no. Topic** |
| 1 Storyline |
| 2 Components of Database Design |
| 3 Entity Relationship Diagram |
| 4 Relational Model |
| 5 Normalization |
| 6 SQL Queries |
| 7 Learning from the Project |
| 8 Project Demonstration |
| 9 Self-learning beyond classroom |
| 10 Challenges faced |
| 11 Conclusion |

**I. Storyline**

In today’s fast-paced world, fitness has become an essential aspect of people’s lives. Our project aims to develop a **fitness management system** that provides users with access to **personalized diet plans, workout programs, and mindfulness content**. The system allows users to purchase fitness-related products, enroll in fitness programs, track their diet plans, and access wellness resources.

The system includes:

* **User registration** where individuals sign up with their details.
* **Product purchasing** where users can browse and buy fitness-related products.
* **Diet plans and recipes** for users to personalize their nutrition.
* **Fitness programs** that users can enroll in to achieve their goals.
* **Mindfulness content** such as guided meditation and relaxation techniques.

The database ensures smooth data management, enabling efficient tracking of users, products, orders, and fitness plans.

**II. Components of Database Design**

**Table 1: User**

**Purpose :** Stores user details and handles login.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| UserID | INT | Primary Key. Unique ID for each user |
| Name | VARCHAR(50) | Full name of the user(First name and Last name) |
| Email | VARCHAR(50) | Unique email used for login |
| Password | VARCHAR(100) | Encrypted/stored password |

**Primary Key:** UserID

**Table 2: Products**

**Purpose :** Stores details of products like

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| ProductID | INT | Primary Key. Unique ID for each product |
| ProductName | VARCHAR(100) | Name of the product |
| Description | VARCHAR(255) | Product description |
| Price | DECIMAL(8,2) | Cost of the product |
| Stock | INT | Availability of products |
| Category\_ID | INT | Foreign key(from table Categories) |

**Primary Key:** ProductID

**Table 3: Categories**

**Purpose :** Illustrates all the product categories in store.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| CategoryID | INT | Primary Key. Unique Category ID |
| CategoryName | VARCHAR(100) | Name of product category |

**Primary Key:** CategoryID

**Table 4: Orders**

**Purpose :** All the order details of a purchase.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| OrderID | INT | Primary Key. Unique ID for each order |
| OrderDate | DATE | Date on which order is placed |
| TotalAmount | DECIMAL(8,2) | Amount of the order placed |
| PaymentStatus | VARCHAR(100) | Status of the payment:UPI,Netbanking,COD,etc. |

**Primary Key:** OrderID

**Table 5: OrderDetails**

**Purpose :** Stores details of order placed.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| OrderID | INT | Foreign Key(from table Orders) |
| ProductID | INT | Foreign key(from table Products) |
| Quantity | INT | Quantity of products purchased |
| UnitPrice | DECIMAL(8,2) | Price per product |

**Primary Key:**None(weak entity set)

**Table 6: Programs**

**Purpose :** Holds the details of all programs offered,

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| ProgramID | INT | Primary Key. Unique ProgramID |
| Title | VARCHAR(100) | Type of program enrolling in(ex:Yoga) |
| Description | VARCHAR(255) | Program description |
| Duration | INT | Duration of program in weeks |
| EnrollmentDate | DATE | Start date of fitness program |

**Primary Key:** ProgramID

**Table 7: DietPlans**

**Purpose :** Stores details of particular diet plans offered.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| DietPlanID | INT | Primary Key. Unique Diet plan ID |
| UserID | INT | Foreign key(from table User) |
| Title | VARCHAR(100) | Type of diet plan |
| Description | VARCHAR(255) | Details of diet plan |
| Duration | INT | Duration diet plan in days |
| DailyCalories | INT | Calorie intake per day |

**Primary Key:** DietPlanID

**Table 8: DietRecipes**

**Purpose :** Stores recipe name,instructions,ingredients of various dishes in diet.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| DietPlanID | INT | Foreign key(from table DietPlans) |
| RecipeName | VARCHAR(100) | Name of the recipe |
| Instructions | VARCHAR(255) | Steps to make a dish |
| Ingredients | VARCHAR(255) | Items needed to make a dish |

**Primary Key:** None(weak entity set)

**Table 9: MindfullnessContent**

**Purpose :** Organizes courses into categories (e.g., Data Science, Design).

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| ContentID | INT | Primary Key. Unique content ID |
| ContentType | VARCHAR(50) | Name of the content(ex:meditation) |
| Description | TEXT | Description of a various exercises |
| Date Published | DATE | Date when content was posted |

**Primary Key:** ContentID

* **Relationships Among Entities**

1. **User – Orders** :

|  |
| --- |
| **Relation**: A user can place multiple orders. **Cardinality:** 1 to many |

1. **User – Programs :**

|  |
| --- |
| **Relation:** A user can enroll in many programs **Cardinality:** 1 to many |

1. **User-MinfullnessContent :**

|  |
| --- |
| **Relation:** An user can access many mindfulness programs **Cardinality:** 1 to many |

1. **User-DietPlans:**

|  |
| --- |
| **Relation:** An user can access many diet plans **Cardinality:** 1 to many |

1. **Orders-Products** :

|  |
| --- |
| **Relation:** Orders can include many products  **Cardinality:** many to many |

1. **Products-Categories :**

|  |
| --- |
| **Relation:** A group of products are of a particular category **Cardinality:** many to one |

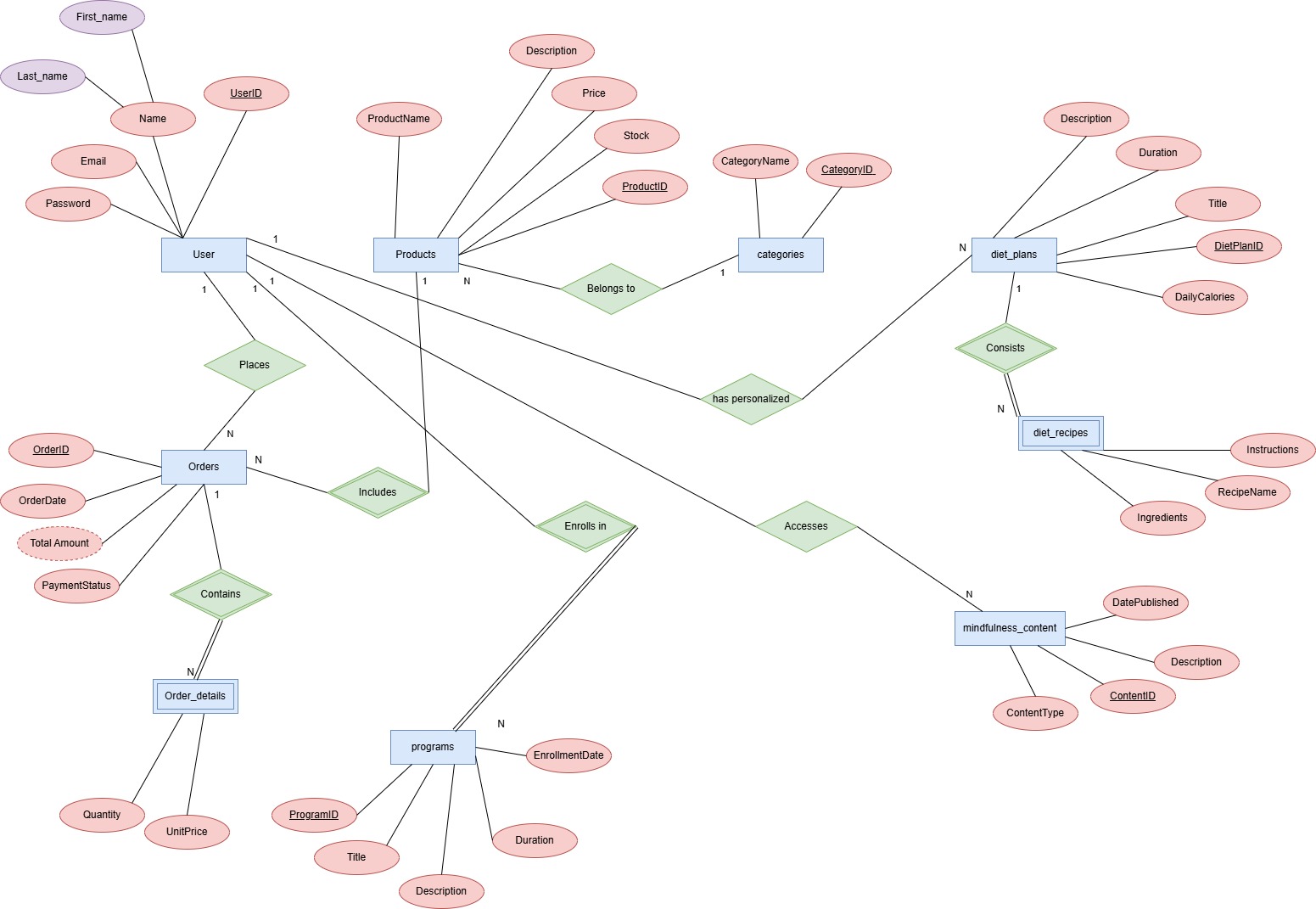
1. **DietPlans-DietRecipes :**

|  |
| --- |
| **Relation:** A diet plan consists of various recipes **Cardinality:** one to many |

1. **Order-OrderDetails :**

|  |
| --- |
| **Relation:** An order has some order details essential for identification **Cardinality:** 1 to many |

**III. Entity Relationship Diagram**



**IV. Relational Model**

Various tables obtained are:-

1. User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | First\_name | Last\_name | Email | Password |
|  |  |  |  |  |

* Schema:User(UserID,First\_name,Last\_name,Email,Pasword)

1. Orders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_ID | UserID(Foreign key) | OrderDate | TotalAmount | PaymentStatus |
|  |  |  |  |  |

* Schema: Orders(Order\_ID,UserID,OrderDate,TotalAmount,PaymentStatus)

1. order\_details(weak entity)

|  |  |  |  |
| --- | --- | --- | --- |
| OrderID (Foreign key) | ProductID(Foreign key) | Quantity | UnitPrice |
|  |  |  |  |

* Schema: order\_details(OrderID,Product\_ID,Quantity,UnitPrice)

1. Products

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ProductID | ProductName | Stock | Price | Description | CategoryID(Foreign key) |
|  |  |  |  |  |  |

* Schema: Products(ProductID,ProductName,Stock,Price,Description,CategoryID)

1. Categories

|  |  |
| --- | --- |
| CategoryID | CategoryName |
|  |  |

* Schema: Categories(CategoryID,CategoryName)

1. Diet\_plans

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DietPlan\_ID | User\_ID(Foreign key) | Title | Duration | DailyCalories | Description |
|  |  |  |  |  |  |

* Schema:diet\_plans(DietPlan\_ID,User\_ID,Title,Duartion,DailyCalories,Description)

1. Diet\_recipes(weak entity)

|  |  |  |  |
| --- | --- | --- | --- |
| DietPlanID(Foreign key) | Instructions | RecipeName | Ingredients |
|  |  |  |  |

* Schema:diet\_recipes(DietPlanID,Instructions,RecipeName,Ingredients)

1. Mindfulness\_content

|  |  |  |  |
| --- | --- | --- | --- |
| ContentID | ContentType | Description | DatePublished |
|  |  |  |  |

Schema:Mindfulness\_content(ContentID,ContentType,Description,DatePublished)

1. Programs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ProgramID | Title | Description | EnrollmentDate | Duration |
|  |  |  |  |  |

* Schema: programs(ProgramID,Title,Description,EnrollmentDate,Duration)

1. Places(one to many)

Two tables will be required:-

1. User table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | First\_name | Last\_name | Email | Password |
|  |  |  |  |  |

1. Combined table for entity Orders and reltionship set “Places”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | OrderID | OrderDate | TotalAmount | PaymentStatus |
|  |  |  |  |  |

1. Contains(one to many)

Two table will be required:-

1. Orders table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_ID | UserID(Foreign key) | OrderDate | TotalAmount | PaymentStatus |
|  |  |  |  |  |

1. Combined table of entity set order\_details and relationship set “Contains”

|  |  |  |  |
| --- | --- | --- | --- |
| OrderID | ProductID | Quantity | UnitPrice |
|  |  |  |  |

1. Includes(one to many)

Two tables will be required:-

1. Products table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ProductID | ProductName | Stock | Price | Description | CategoryID |
|  |  |  |  |  |  |

1. Combined table of entity set order\_details and relationship set “Includes”

|  |  |  |  |
| --- | --- | --- | --- |
| ProductID | OrderID | Quantity | UnitPrice |
|  |  |  |  |

1. Belongs to(many to one)

Two tables will be required:-

1. Combined table of entity Products and relationship set “Belongs to”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CategoryID | ProductID | Stock | Price | Description | ProductName |
|  |  |  |  |  |  |

1. Categories table

|  |  |
| --- | --- |
| CategoryID | CategoryName |
|  |  |

1. has personalized (one to many)

Two tables will be required:-

1. User table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | First\_name | Last\_name | Email | Password |
|  |  |  |  |  |

1. Combined table of entity diet\_plans and relationship set “has paersonalized”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| UserID | DietPlanID | Title | Duration | DailyCalories | Description |
|  |  |  |  |  |  |

1. Consists(one to many)

Two tables will be requiresd:-

1. diet\_plans table:-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DietPlan\_ID | User\_ID | Title | Duration | DailyCalories | Description |
|  |  |  |  |  |  |

1. Combined table of entity diet\_recipes and relationship set “Consists”

|  |  |  |  |
| --- | --- | --- | --- |
| DietPlanID | Instructions | RecipeName | Ingredients |
|  |  |  |  |

1. Accesses(one to many)

Two tables will be required:-

1. User table:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | First\_name | Last\_name | Email | Password |
|  |  |  |  |  |

1. MindfullnessContent

|  |  |
| --- | --- |
| UserID | ContentID |
|  |  |

1. Enrolls in(many to many)

Two tables will be required:-

1. User table:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | First\_name | Last\_name | Email | Password |
|  |  |  |  |  |

1. Combined table of entity programs and relationship “Enroll in”

|  |  |  |
| --- | --- | --- |
| UserID | ProgramID | EnrollmentDate |
|  |  |  |

**V. Database Normalization**

The database includes these main entities:  
• User  
• Products  
• Categories  
• Orders  
• Order Details  
• Programs  
• Diet Plans  
• Diet Recipes  
• Mindfulness Content

### **Normalization in DBMS**

**Definition:** Normalization is a process in relational database design used to minimize redundancy and dependency by organizing fields and table data. It involves decomposing a complex database into simpler, smaller tables without losing data integrity.

### **Normal Forms (NFs):**

Normalization is carried out through a series of "normal forms". Each form addresses a specific type of problem:

#### **1. First Normal Form (1NF):**

* Ensures that each column has atomic (indivisible) values.
* No repeating groups or arrays.

**Rule:** All entries in a column must be of the same data type and each column must contain only one value per row.

#### **2. Second Normal Form (2NF):**

* Must be in 1NF.
* Removes partial dependencies (non-prime attributes dependent on part of a composite primary key).

**Rule:** Every non-key attribute must be fully functionally dependent on the primary key.

#### **3. Third Normal Form (3NF):**

* Must be in 2NF.
* Removes transitive dependencies.

After applying the principles of normalization (1NF, 2NF, 3NF, and BCNF), the original database schema has been refined to eliminate redundancy, maintain atomicity, and ensure data integrity. The result is a more structured and efficient database with **11 well-normalized tables**.

### **1. User Table**

* **Purpose:** Stores basic information about each user (member, trainer, or admin).
* **Normalization Action:** Already satisfied all normal forms; no changes required.
* **Fields:** UserID (PK), Name, Email, Password, UserType, DateOfRegistration

### **2. Product Table**

* **Purpose:** Holds details about fitness-related products.
* **Normalization Action:** Already satisfied all the normal forms;no changes required.
* **Fields:** ProductID (PK), ProductName, Price, Stock, CategoryID (FK),description.

### **3. Category Table**

* **Purpose:** Stores category names and allows grouping of products.
* **Normalization Action:** Maintains atomicity and dependency with Product table.
* **Fields:** CategoryID (PK), CategoryName

### **4. Order Table**

* **Purpose:** Records order metadata.
* **Normalization Action:** Removed the TotalAmount field, which can be derived from order details.
* **Fields:** OrderID (PK), UserID (FK), OrderDate

### **5. OrderDetail Table**

* **Purpose:** Represents the many-to-many relationship between Orders and Products.
* **Normalization Action:** Composite primary key (OrderID, ProductID) ensures 2NF and 3NF.
* **Fields:** OrderID (PK, FK), ProductID (PK, FK), Quantity, UnitPrice

### **6. Program Table**

* **Purpose:** Contains all available fitness programs like Yoga, HIIT, etc.
* **Normalization Action:** Retains unique attributes only related to the program.
* **Fields:** ProgramID (PK), Title, Description, Duration

### **7. ProgramEnrollment Table**

* **Purpose:** Tracks which user has enrolled in which program.
* **Normalization Action:** Fully normalized till 3NF with a composite primary key.
* **Fields:** UserID (PK, FK), ProgramID (PK, FK), EnrollmentDate

### **8. DietPlan Table**

* **Purpose:** Records personalized diet plans for users.
* **Normalization Action:** Removed aggregate fields like DailyCalories, which are derivable.
* **Fields:** DietPlanID (PK), UserID (FK), Title, Description, Duration

### **9. Recipe Table**

* **Purpose:** Stores basic information about each recipe in a diet plan.
* **Normalization Action:** Split from previously combined fields in diet\_recipes.
* **Fields:** RecipeID (PK), DietPlanID (FK), RecipeName

### **10. Ingredient Table**

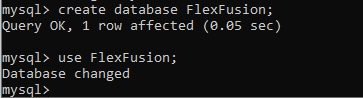
* **Purpose:** Contains atomic ingredient entries for each recipe.
* **Normalization Action:** Created to resolve non-atomic values in the original Ingredients field.
* **Fields:** RecipeID (FK), IngredientName

### **11. Instruction Table**

* **Purpose:** Provides step-by-step cooking instructions for each recipe.
* **Normalization Action:** Split from non-atomic Instructions field into a normalized structure.
* **Fields:** RecipeID (FK), Instruction

**VI. SQL QUERIES**

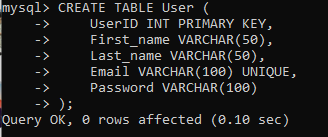
**Create and use database**



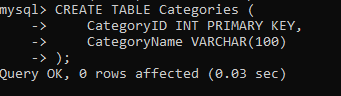
**DATA DEFINATION LANGUAGE(DDL) COMMANDS**

**Create tables**

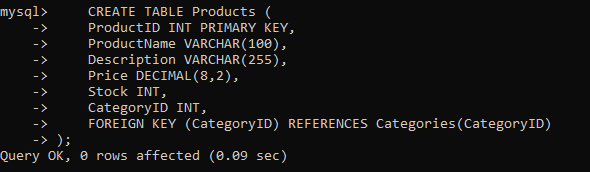
1)User



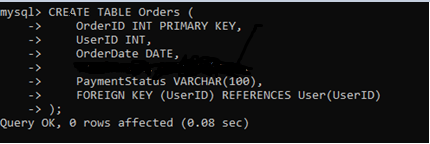
2)Categories



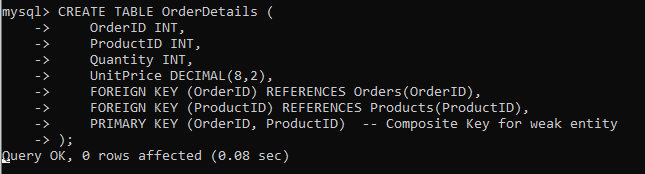
3)Products



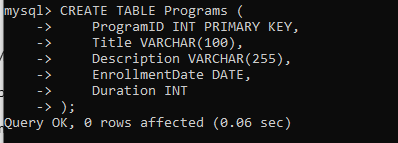
4)Orders



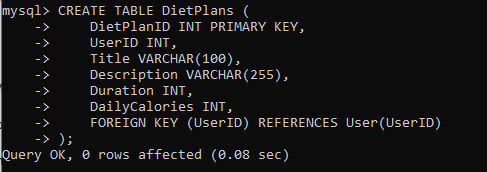
5)OrderDetails(weak entity)



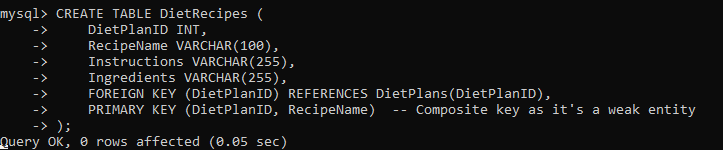
6)Programs



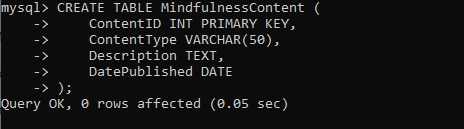
7)DietPlans



8) DietRecipe(weak entity)



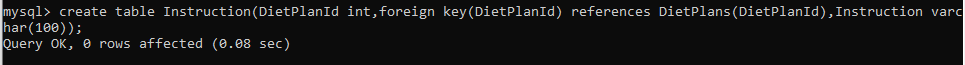
9)MindfullessContent



10)Ingredient\_table

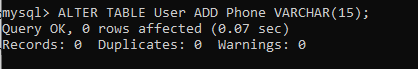


11)Instruction

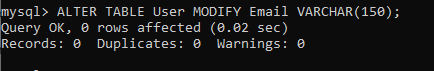


**Alter table**

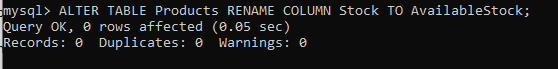
1) **Add a new column Phone to the User table:**



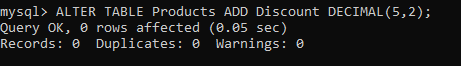
2)Modify the data type of Email in User to accommodate longer emails:



3) Rename column Stock to AvailableStock in Products:



4) Add a Discount column to Products table



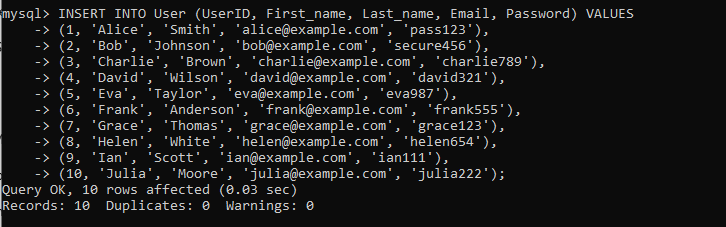
**Drop table**

1) **Drop the** PaymentStatus **column from** Orders

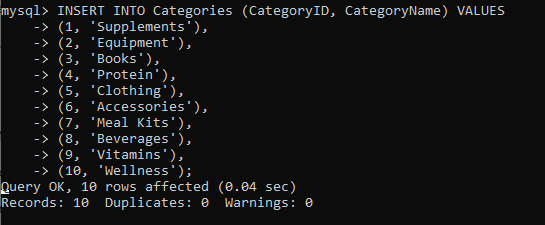
**DATA MANIULATION LANGUAGE(DML) COMMANDS**

**Insert values into tables**

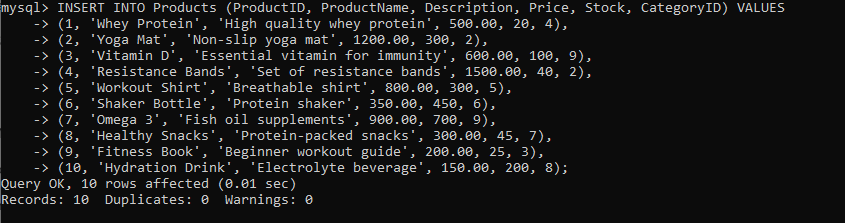
1)User



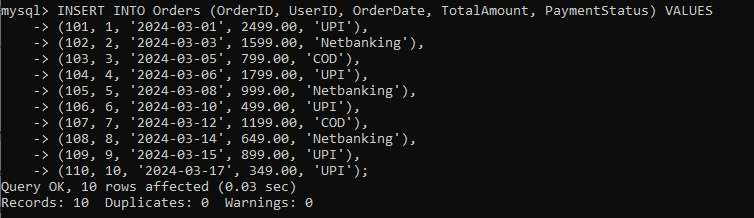
2)Categories



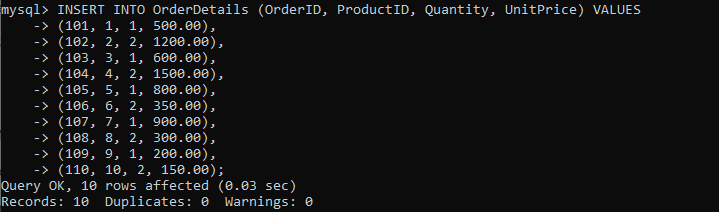
3)Products



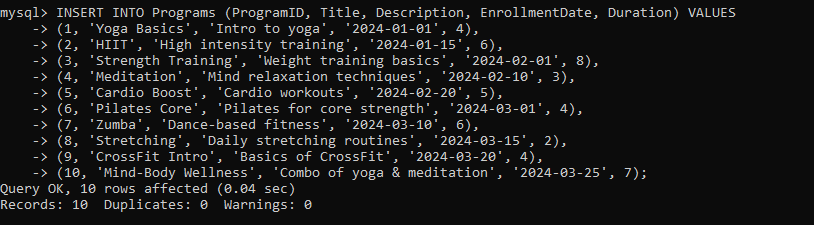
4)Orders



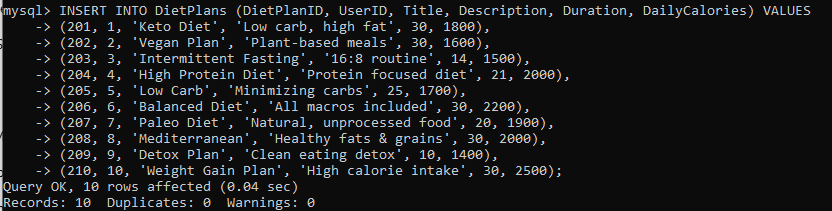
5)OrderDetails



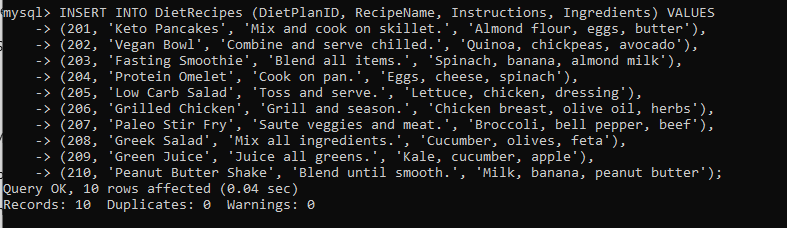
6) Programs



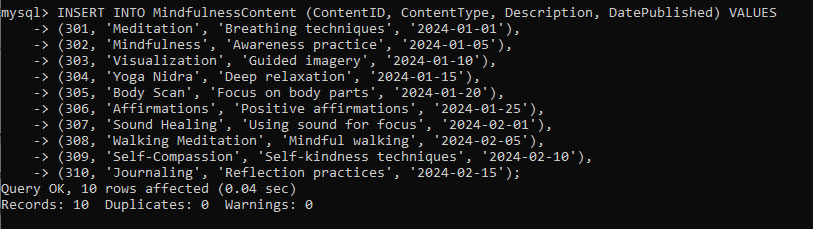
7)DietPlans



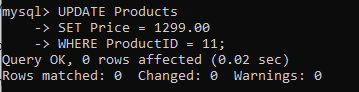
8)DietRecipes

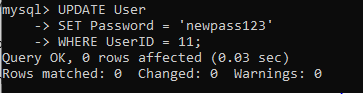


9)MindfullnessContent

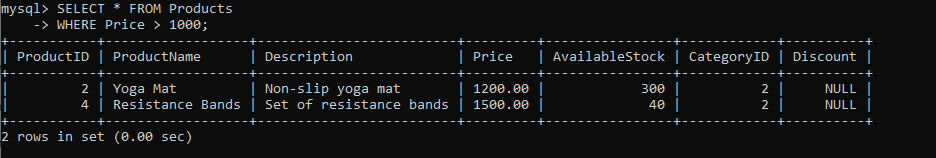


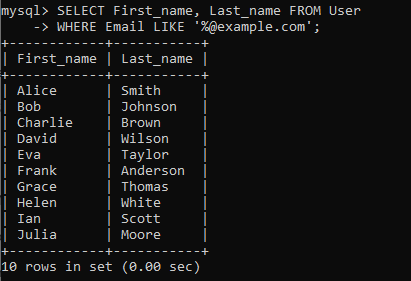
**Update command**

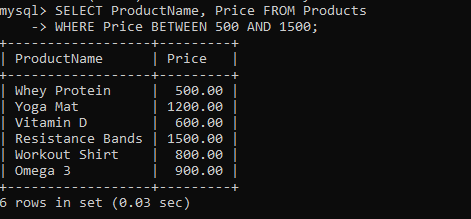




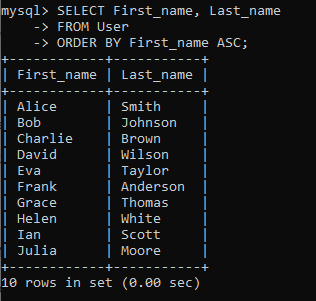
**Select command**

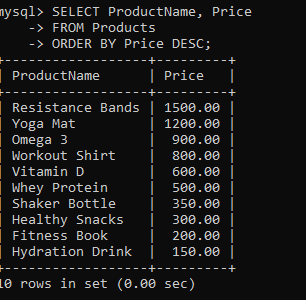






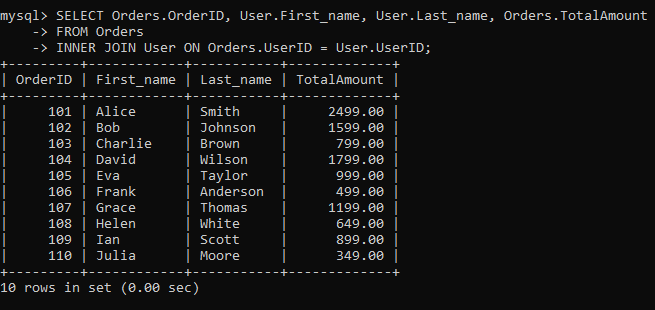
**Order by**



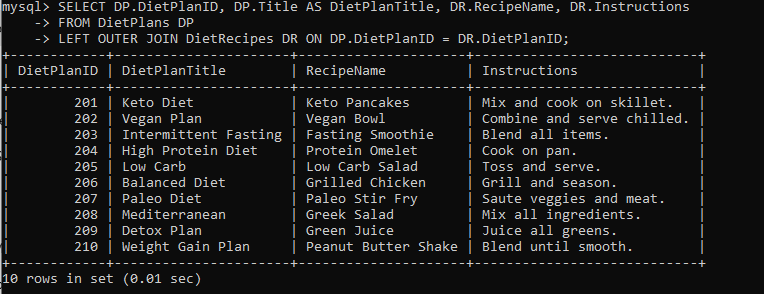


**Joins**

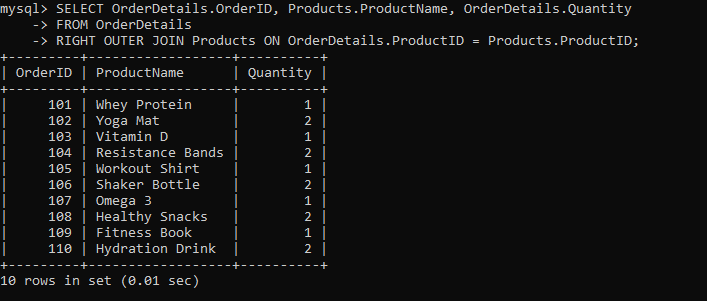
### **1. INNER JOIN:**Returns only matching rows from both tables.



### **2.LEFT OUTER JOIN:**Returns all rows from the left table and matched rows from the right table. If no match, NULLs are shown.

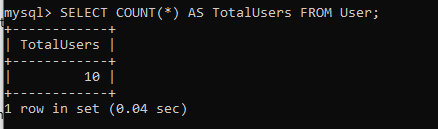


### **3. RIGHT OUTER JOIN:**Returns all rows from the right table and matched rows from the left table. If no match, NULLs are shown.

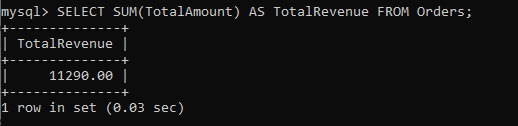


**Aggregate commands**

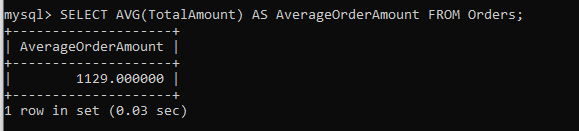
1. Count total number of users



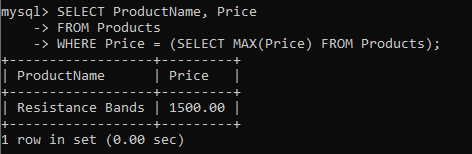
2. Find the total revenue generated from all orders



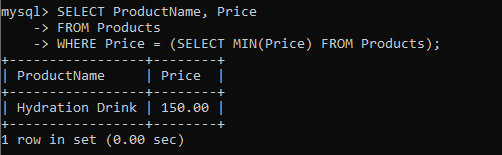
**3. Find the average amount spent per order**



4. Find the product with the maximum price

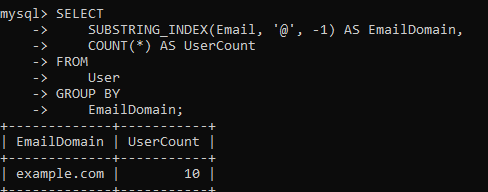


5. Find the product with the minimum price

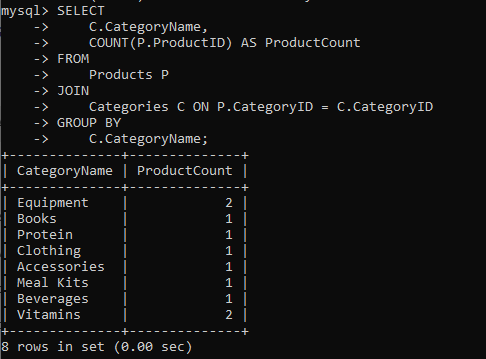


**Group by commands**

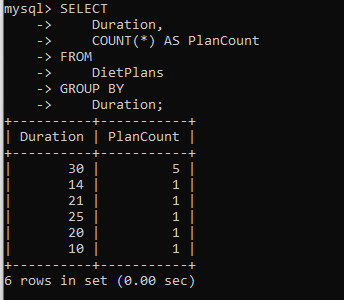
1.Total number of users per domain in their email



### **2.Number of products available in each category**

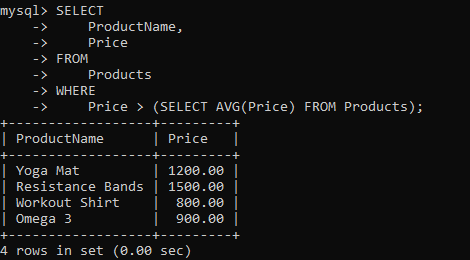


1. Count of diet plans grouped by duration

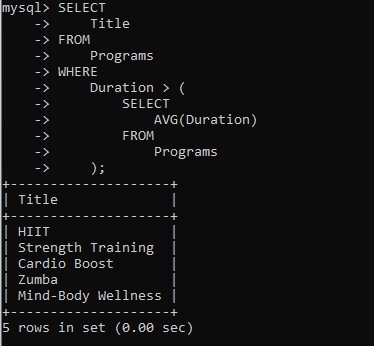


**Group by commands**

1. Get all products with a price higher than the average price of all products.



2. Get the names of programs that have a duration longer than the average duration of all programs.

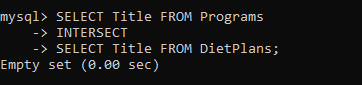


**Set Operations**

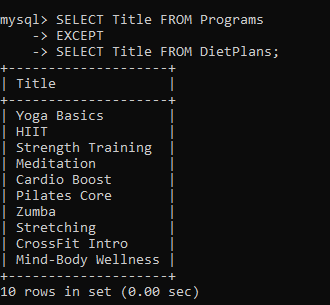
1. UNION — Get all titles from both Programs and Diet Plans (no duplicates)



2. INTERSECT — Find common titles between Programs and Diet Plans



3.EXCEPT / MINUS — Get titles of Programs that are not in Diet Plans

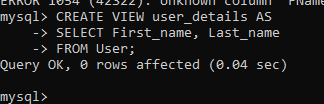


**DCL (Data Control Language) commands**

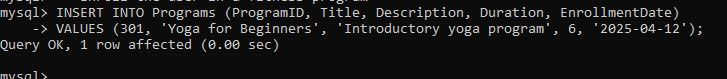








**Control Transaction commands**





**VII. LEARNING FROM PROJECT**

This project provided a solid foundation for understanding how database systems work in a practical context. Here are some key learnings:

* **Hands-on SQL Practice:** Writing and executing SQL queries helped reinforce theoretical concepts.
* **Database Design Principles:** We applied principles like normalization and ER modeling to design an efficient and scalable schema.
* **Team Collaboration:** Working in a group improved our ability to divide tasks and collaborate effectively.
* **Problem-solving and Debugging:** Encountering and solving issues related to foreign keys, data types, and table relationships honed our debugging skills.
* **Business Logic Mapping:** Translating real-world fitness operations into database logic helped us think like full-stack developers.

**VIII. PROJECT DEMONSTRATION**

**IX. SELF LEARNING BEYOND CLASSROOM**

Throughout the development of the *Flex Fusion* fitness database system, we encountered scenarios that required us to explore topics beyond the DBMS curriculum. These self-driven explorations enriched our learning experience:

1. **User Role Management (DCL Usage):**  
   We learned how to apply SQL’s Data Control Language (GRANT, REVOKE) to implement role-based access control, giving specific privileges to users like administrators or guests. This simulated real-world database security and user management.
2. **Transaction Management and ACID Properties:**  
   Implementing transactions using BEGIN, COMMIT, and ROLLBACK helped us understand the importance of atomic operations and consistency in case of failure, especially in processes like updating stock or assigning diet plans.
3. **Creation and Use of Views:**  
   We explored SQL views to simplify data access for users by abstracting complex queries. For instance, creating views like user\_details or active\_programs helped present filtered, relevant data.
4. **Normalization Beyond 3NF:**  
   While classroom lessons covered 1NF to 3NF, we explored **Boyce-Codd Normal Form (BCNF)** to further refine our schema, especially when resolving anomalies in multi-attribute dependencies.
5. **Real-World Schema Design Decisions:**  
   We analyzed how popular platforms like MyFitnessPal or Cure.fit structure their data, and adapted features like mindfulness content, categorized products, and personalized fitness plans into our database design.
6. **Query Optimization Techniques:**  
   We practiced writing efficient queries using subqueries, JOINs, GROUP BY, and aggregate functions to simulate dashboard metrics such as most popular programs, revenue per category, and user engagement trends.
7. **Version Control with SQL Files:**  
   We introduced basic version control by organizing and naming our SQL files properly, making collaboration smoother and reducing conflicts during updates.

These experiences pushed us to think like database designers and backend developers, making the project more realistic and industry-aligned.

**X. CHALLENGES FACED**

**1. Data Redundancy:** Avoiding redundancy while keeping queries readable was a challenge, especially for derived attributes like TotalAmount in orders.

**2. Atomicity Issues:** Designing recipe storage in the DietRecipes table without violating 1NF required rethinking how to structure ingredients and instructions.

3. **Version Control of Queries:** Maintaining consistency in the evolving database schema when multiple members worked in parallel was tricky.

**4. Tool Familiarity:** Adapting to tools like MySQL Workbench, and learning how to visually design ER diagrams and run complex joins was a learning curve.

**X. CONCLUSION**

The **Flex Fusion** fitness website database system project enabled us to translate a real-world application into a structured database solution. Our major takeaways include:

* Strong understanding of **database schema design**, **entity relationships**, and **normalization**.
* Proficiency in writing comprehensive **SQL queries** covering DDL, DML, DCL, and transaction control.
* Awareness of **practical data issues** like redundancy, atomicity, and referential integrity.
* Insight into how databases support larger systems like e-commerce platforms, health apps, and content platforms.

This experience has significantly improved our capability to approach backend development and data management with confidence and competence.