**HALL TICKET NUMBER: 2403A51365**

**NAME: RAYAPUDI HARITHA**

**BATCH: 24BTCAICSB14**

**AssignmentNumber:16.1**

**Lab 16 – Database Design and Queries: Schema Design and SQL  
Generation  
Lab Objectives:  
• Understand schema design principles for relational databases.  
• Use AI-assisted tools to generate SQL schema definitions.  
• Write and refine SQL queries (CRUD + aggregate operations).  
Week8 -  
Monday**

**• Explore how AI can optimize queries and ensure normalization.  
Task 1 – Student Information System Schema  
Task:  
Design a database schema for a Student Information System and  
generate queries using AI.  
Instructions:  
• Tables: Students, Courses, Enrollments.  
• Define primary keys, foreign keys, and relationships.  
• Generate queries:  
o Insert a new student record.  
o Fetch all courses enrolled by a student.  
o Count number of students in each course.**

Prompt:-

Design a database schema for a Student Information System with the following tables: Students, Courses, and Enrollments. For each table, define: - Primary keys - Relevant attributes (e.g., student name, course title, enrollment date) - Data types for each attribute Define the relationships between these tables, including: - Which tables are related - What type of relationship exists (e.g., one-to-many) - Which attributes serve as foreign keys Based on this schema, generate the following SQL queries: 1. A query to insert a new student record into the Students table. 2. A query to fetch all courses enrolled by a specific student (you can use a placeholder for the student ID). 3. A query to count the number of students enrolled in each course. Present the database schema clearly, showing each table's structure, keys, and relationships. Then, provide the generated SQL queries with explanations for each.

Code:-

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Output:-

A screenshot of a computer program

AI-generated content may be incorrect.

Code observation:-

1. The code creates a Student Information System database with three main tables: Students (storing student details), Courses (storing course information), and Enrollments (managing student-course relationships).
2. The Students table has a primary key student\_id and stores essential student information like name, email, date of birth, and phone number.
3. The Courses table uses course\_id as primary key and contains course details including code, name, credits, and instructor.
4. The Enrollments table serves as a junction table implementing a many-to-many relationship between Students and Courses, allowing students to enroll in multiple courses and courses to have multiple students.
5. Sample data is inserted into all three tables to demonstrate the relationships and query capabilities.
6. The first query demonstrates inserting a new student (Alice Williams) and displaying her information.
7. The second query shows how to fetch all courses for a specific student (ID 1), using JOIN to combine data from Courses and Enrollments tables.
8. The third query counts students in each course using COUNT and GROUP BY, demonstrating how to generate enrollment statistics.
9. The relationships between tables are maintained through foreign keys (student\_id and course\_id in Enrollments table).
10. The code includes proper constraints like UNIQUE for email and course\_code, NOT NULL for required fields, and DEFAULT values for dates.

Task 2 – Hospital Management Database  
Task:  
Create schema and queries for a Hospital Management System.  
Instructions:  
• Tables: Doctors, Patients, Appointments.  
• Use AI to define constraints (unique IDs, valid dates).  
• Generate queries:  
o List all appointments for a specific doctor.  
o Retrieve patient history by patient ID.  
o Count total patients treated by each doctor.

Prompt:-

Design a database schema for a Hospital Management System with the following tables: Doctors, Patients, and Appointments.

For each table, define:

- Primary keys

- Relevant attributes (e.g., doctor name, patient name, appointment date and time)

- Data types for each attribute

- Constraints, including unique IDs and valid date/time formats for appointments.

Define the relationships between these tables, including:

- Which tables are related

- What type of relationship exists (e.g., one-to-many)

- Which attributes serve as foreign keys

Based on this schema, generate the following SQL queries:

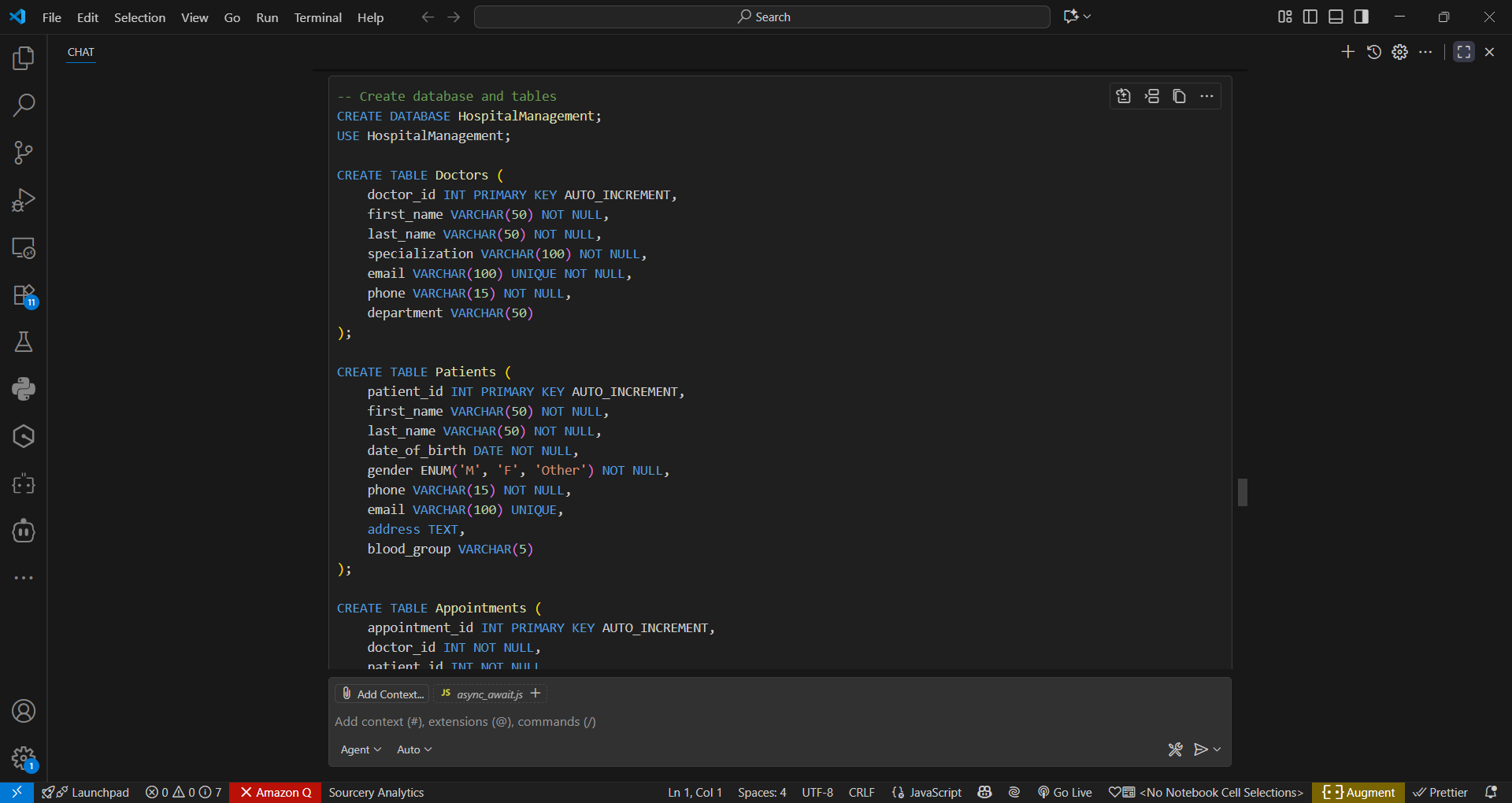
1. A query to list all appointments for a specific doctor (you can use a placeholder for the doctor ID).

2. A query to retrieve the complete history of appointments for a specific patient, ordered by date (use a placeholder for the patient ID).

3. A query to count the total number of distinct patients treated by each doctor.

Present the database schema clearly, showing each table's structure, keys, relationships, and constraints. Then, provide the generated SQL queries with explanations for each.

Code:-



A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Output:-

A screenshot of a computer

AI-generated content may be incorrect.

Code observation:-

1. The code creates a Hospital Management System database with three main tables: Doctors (storing doctor information), Patients (storing patient details), and Appointments (managing doctor-patient appointments).
2. The Doctors table includes essential information like doctor's name, specialization, contact details, and department, with doctor\_id as the primary key.
3. The Patients table stores patient information including name, date of birth, gender, contact details, and blood group, with patient\_id as the primary key.
4. The Appointments table acts as a bridge between Doctors and Patients, implementing a many-to-many relationship where doctors can see multiple patients and patients can visit multiple doctors.
5. The schema includes important constraints like UNIQUE for emails, NOT NULL for required fields, and a CHECK constraint to ensure appointment dates are not in the past.
6. Sample data is inserted into all tables to demonstrate real-world usage, including three doctors, three patients, and four appointments.
7. The first query demonstrates how to list all appointments for a specific doctor, showing patient names and appointment details.
8. The second query retrieves a patient's appointment history, displaying doctor information and appointment status ordered by date.
9. The third query counts the number of unique patients seen by each doctor, using COUNT(DISTINCT) to avoid counting repeat visits.
10. The relationships are maintained through foreign keys (doctor\_id and patient\_id in Appointments table), and appropriate data types (ENUM for status and gender, DATE and TIME for appointments) are used throughout.

Task 3 – Library Database  
Task:  
Design schema for a Library Management System.  
Instructions:  
• Tables: Books, Members, Loans.  
• Use AI to suggest indexing strategy for faster queries.  
• Generate queries:  
o Retrieve all books currently issued.  
o Find overdue books (loan date > 30 days).  
o Count number of books loaned by each member.

Prompt:-

Design a database schema for a Library Management System with the following tables: Books, Members, and Loans.

For each table, define:

- Primary keys

- Relevant attributes (e.g., book title, author, member name, loan date, return date)

- Data types for each attribute

Define the relationships between these tables, including:

- Which tables are related

- What type of relationship exists (e.g., one-to-many)

- Which attributes serve as foreign keys

Based on this schema, suggest an indexing strategy to improve query performance, especially for frequently accessed data like book titles, member IDs, and loan dates. Explain why these indexes would be beneficial.

Finally, generate the following SQL queries:

1. A query to retrieve all books that are currently issued (i.e., have a loan record but no return date).

2. A query to find all books that are overdue (i.e., the loan date is more than 30 days ago and there is no return date).

3. A query to count the total number of books loaned by each member.

Present the database schema clearly, showing each table's structure, keys, and relationships. Include the suggested indexing strategy with explanations. Then, provide the generated SQL queries with explanations for each.

Code:-

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Output:-

A screenshot of a computer program

AI-generated content may be incorrect.

code observation:-

1. The code creates a Library Management System database with three main tables: Books (storing book information), Members (storing member details), and Loans (tracking book borrowing records).
2. The Books table includes essential information about each book (ISBN, title, author, category) with book\_id as primary key and includes a status field to track availability.
3. The Members table stores member information (name, contact details, join date) with member\_id as primary key and tracks member status (Active/Inactive).
4. The Loans table implements the relationship between Books and Members, recording when books are borrowed and returned, with appropriate foreign keys to link to both tables.
5. The schema includes strategic indexes on frequently accessed fields (title, ISBN, email, dates) to improve query performance, especially for searches and joins.
6. Sample data demonstrates real-world usage with three books, three members, and various loan scenarios including current loans and returns.
7. The first query shows currently issued books (no return date) by joining all three tables to display comprehensive loan information.
8. The second query identifies overdue books by checking for loans older than 30 days without returns, calculating the number of overdue days.
9. The third query provides statistics on book borrowing patterns per member, showing both total loans and current loans using COUNT and CASE statements.
10. The relationships are maintained through foreign keys (book\_id and member\_id in Loans table), and appropriate data types and constraints (ENUM for status, UNIQUE for ISBN/email) ensure data integrity.

Task 4 – Real-Time Application: Online Shopping Database  
Scenario:  
Design a database for an E-commerce platform.  
Requirements:  
• Tables: Users, Products, Orders, OrderDetails.  
• Generate queries to:  
o Retrieve all orders by a user.  
o Find the most purchased product.  
o Calculate total revenue in a given month.  
• AI should also suggest normalization improvements and query  
optimization.

Prompt:-

Design a database schema for an Online Shopping platform with the following tables: Users, Products, Orders, and OrderDetails.

For each table, define:

* Primary keys
* Relevant attributes (e.g., user information, product details, order date, quantity, price)
* Data types for each attribute
* Foreign keys to represent relationships between tables.

Define the relationships between these tables, including:

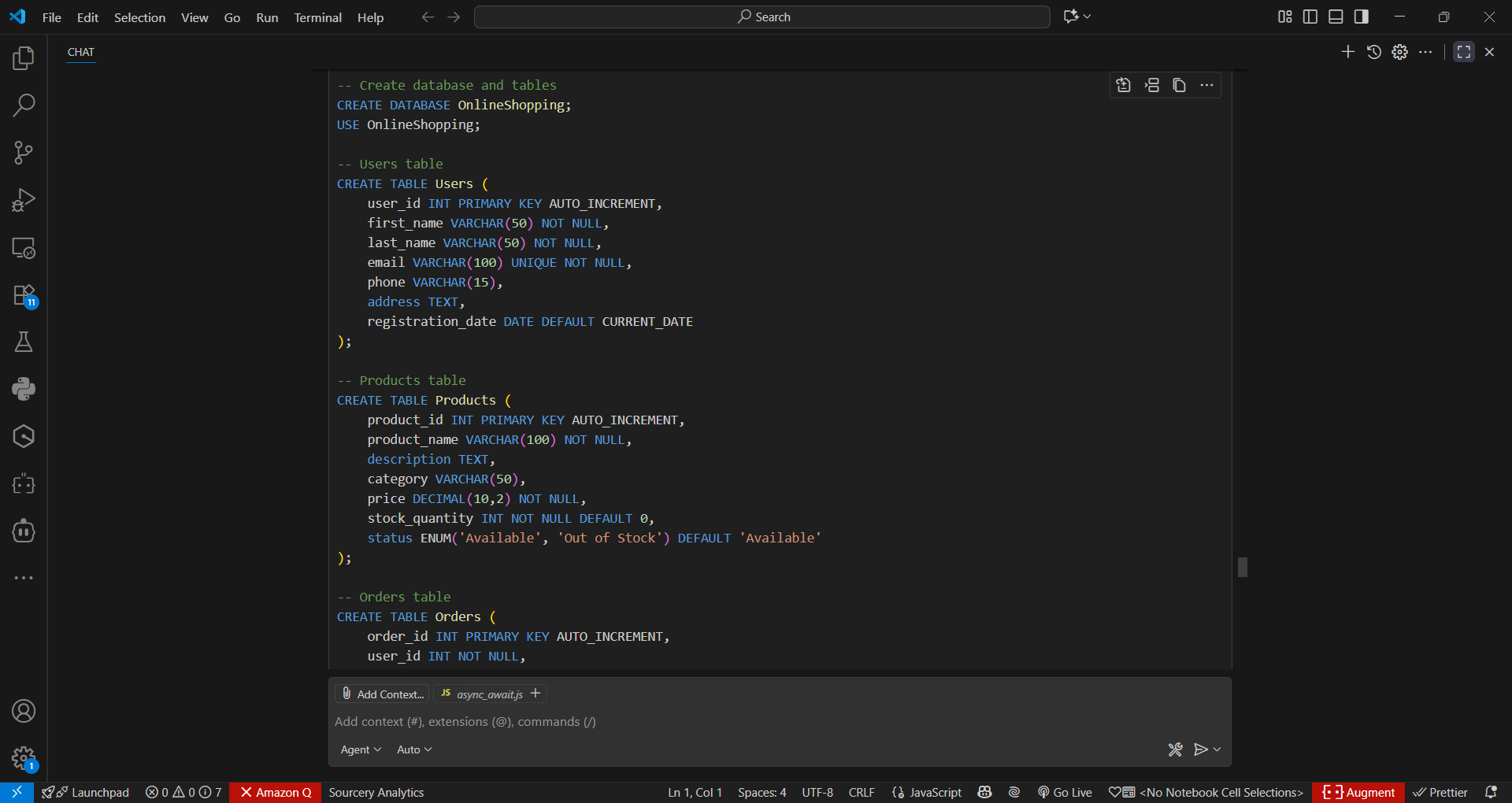
* Which tables are related
* What type of relationship exists (e.g., one-to-many, many-to-many - consider the relationship between Orders and Products, potentially through OrderDetails)
* Which attributes serve as foreign keys

Based on this schema, generate the following SQL queries:

1. A query to retrieve all orders placed by a specific user (you can use a placeholder for the user ID).
2. A query to find the most purchased product across all orders.
3. A query to calculate the total revenue generated in a given month (you can use a placeholder for the month and year).

In addition to the schema and queries, analyze the design and suggest potential normalization improvements to reduce data redundancy and improve data integrity. Also, suggest query optimization techniques for the generated queries, considering potential bottlenecks and how to improve performance (e.g., indexing, query structure).

Code:-



A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Output:-

A screenshot of a computer

AI-generated content may be incorrect.

Code observation:-

1. The code creates an Online Shopping System database with four main tables: Users (customer information), Products (item details and inventory), Orders (transaction records), and OrderDetails (linking orders with products).
2. The Users table stores customer information including name, contact details, and address, with user\_id as the primary key and a unique constraint on email.
3. The Products table manages inventory with product details, pricing, and stock levels, using product\_id as the primary key and including status tracking for availability.
4. The Orders table records transactions with order\_id as primary key, linking to Users through user\_id foreign key, and storing order status and total amount.
5. The OrderDetails table implements a many-to-many relationship between Orders and Products, storing quantity and price information for each item in an order.
6. The schema includes strategic indexes on frequently queried fields (email, product name, order date) to optimize search and join operations.
7. Query 1 demonstrates how to retrieve all orders for a specific user, including the products purchased in each order using GROUP\_CONCAT.
8. Query 2 identifies the most popular product by analyzing the total quantity sold across all orders, showing both sales volume and revenue.
9. Query 3 calculates monthly revenue statistics, including total orders, revenue, and average order value for a specific month.
10. The schema includes proper constraints (foreign keys, NOT NULL, UNIQUE), appropriate data types (DECIMAL for money, ENUM for statuses), and sample data demonstrating real-world usage patterns.