🧩 SECTION 1: ERD - COMPLETE MASTERY

🎯 WHAT IS AN ERD AND WHY DO YOU NEED IT?

What the hell is an ERD?
An Entity Relationship Diagram (ERD) is a BLUEPRINT for your database. It shows:

- What TABLES you need (Entities)
- What COLUMNS each table has (Attributes)
- How tables CONNECT to each other (Relationships)

Why they test this:
They want to see if you can translate business rules into a database design. If you can't design it, you can't build it.

Real World Example:
Think of building a house. You wouldn't start hammering nails without blueprints. ERD = database blueprints.

📜 WHERE THIS COMES FROM: 2024 EXAM QUESTION

ACTUAL 2024 QUESTION:
"Draw an Entity Relationship Diagram (ERD) using Unified Modelling Language (UML) notation according to the below business rules. Your design should be at the logical level – include primary and foreign key fields and remember to remove any many-to-many relationships.

Business rules for a medication system:

- All entities must have surrogate primary keys.
- Each medication contains one or more active ingredients, and an active ingredient can be used in more than one medication in different quantities.
- The name of each medication should be stored in the database.
- The name of each active ingredient must be stored in the database.
- Each medication belongs to one specific category, and many medications can belong to the same category.
- The description of each category must be stored in the database.
- An active ingredient belongs to a specific category, and many active ingredients can belong to the same category."

🛠️ STEP-BY-STEP ERD SOLUTION:

Step 1: Identify Entities (Find the NOUNS)

- Read each sentence, circle every noun:
    - "medication" = MEDICATION table
    - "active ingredients" = ACTIVE_INGREDIENT table
    - "category" = CATEGORY table

Step 2: Identify Relationships (Find the VERBS)

- "medication **contains** active ingredients" → Relationship
- "medication **belongs to** category" → Relationship
- "active ingredient **belongs to** category" → Relationship

Step 3: Determine Relationship Types (Count the NUMBERS)

- "Each medication contains **one or more** active ingredients" = 1 to Many
- "Active ingredient can be used in **more than one** medication" = Many to Many ← **DANGER!**
- "Each medication belongs to **one** category" = Many to One
- "Many medications can belong to the **same** category" = Many to One

Step 4: Fix Many-to-Many Relationships

- Medication ↔ ActiveIngredient (both "many") = CREATE BRIDGE TABLE
- Bridge table name: MedicationIngredient

Step 5: Choose Your Format

FORMAT 1: Table Format (RECOMMENDED)

text

CATEGORY

CategoryID (PK)

CategoryDescription

MEDICATION

MedicationID (PK)

MedicationName

CategoryID (FK)

ACTIVE_INGREDIENT

IngredientID (PK)

IngredientName

CategoryID (FK)

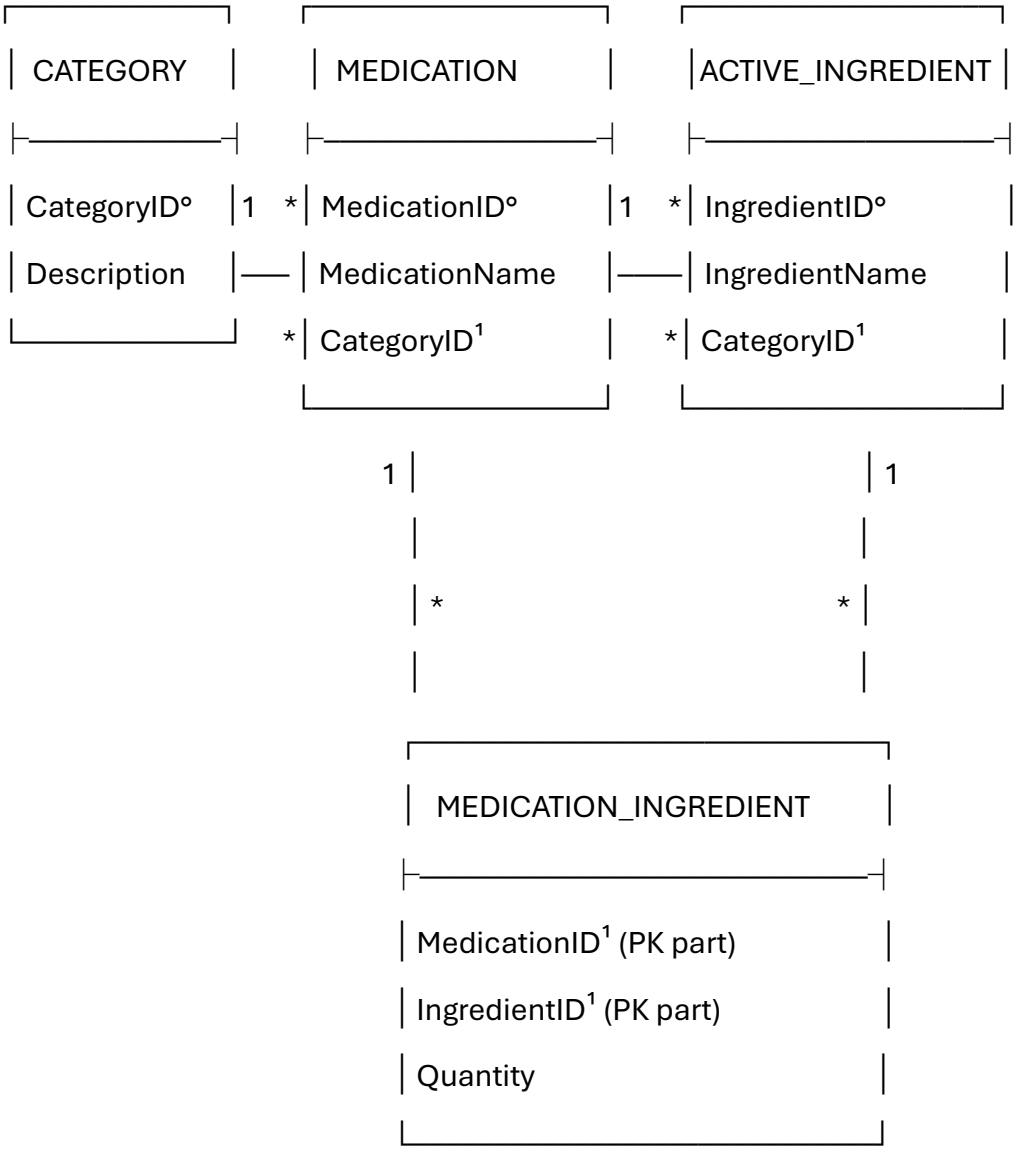
MEDICATION_INGREDIENT (Bridge Table)

MedicationID (FK, PK part)

IngredientID (FK, PK part)

Quantity

FORMAT 2: Graphical UML Format



Why Table Format Wins:

- Directly translates to SQL CREATE TABLE statements
- Clearer relationships through FK placement
- Faster to write in exam
- Less ambiguous

🔍 **MULTIPLICITY EXPLAINED:**

What the hell are these numbers?
They show HOW MANY of one thing can connect to HOW MANY of another thing.

Multiplicity Dictionary:

- **1** = exactly one
- **0..1** = zero or one (optional)
- **1..*** = one or more (at least one)
- **0..*** = zero or more (completely optional)
- *** = many

Business Rule to Multiplicity Translation:

- "Each X has exactly one Y" → X (1) ---- Y (1)
- "Each X has one or more Y" → X (1) ---- Y (*)
- "X may have zero or one Y" → X (0..1) ---- Y (1)
- "X can have many Y" AND "Y can have many X" → CREATE BRIDGE TABLE!

📊 SECTION 2: NORMALIZATION - COMPLETE MASTERY

🎯 WHAT IS NORMALIZATION AND WHY DO YOU CARE?

What the hell is normalization?
It's a process to organize data to reduce duplication and prevent data anomalies.

Real World Example:
Think of a messy filing cabinet where the same customer info is repeated in different files. Normalization = organizing that cabinet so each piece of info is stored only once.

The Three Normal Forms:

1. **1NF** - No repeating groups or lists
2. **2NF** - No partial dependencies (must depend on whole key)
3. **3NF** - No transitive dependencies (must depend only on key)

📜 WHERE THIS COMES FROM: 2022 EXAM QUESTION

ACTUAL 2022 QUESTION:
"Normalise the above data to the third normal form (3NF). Show all steps and the final answer in the form of dependency diagrams.

A lot of data about restaurant staff employment has already been collected in a spreadsheet. The data has been normalised to first normal form already – underlined column names indicate composite primary key columns.

Restaurant ID	Restaurant Name	Waiter ID	Waiter Name	Waiter Surname	Employee Type ID	Employee Type	Start Date	End Date
1	Bears of Brooklyn	2	John	Smith	1	Permanent	2020-01-19	2020-09-15
1	Bears of Brooklyn	3	Sarah	Ndambi	2	Temporary	2021-09-09	2021-10-30
2	Cats of Cape Town	3	Sarah	Ndambi	2	Temporary	2021-11-02	-
2	Cats of Cape Town	4	Themba	Dlamini	1	Permanent	2022-02-02	2022-03-15
3	Doves of Durban	5	Pete	Nair	2	Temporary	2022-04-30	-

🛠️ **STEP-BY-STEP NORMALIZATION SOLUTION:**

Step 1: Identify the Primary Key

- Underlined columns = composite primary key
- Primary Key = (Restaurant ID, Waiter ID)

Step 2: Analyze Dependencies (What depends on what?)

- Restaurant Name → depends on Restaurant ID only
- Waiter Name → depends on Waiter ID only
- Waiter Surname → depends on Waiter ID only
- Employee Type → depends on Waiter ID only
- Employee Type ID → depends on Waiter ID only
- Start Date → depends on BOTH Restaurant ID AND Waiter ID
- End Date → depends on BOTH Restaurant ID AND Waiter ID

Step 3: 2NF - Remove Partial Dependencies
Partial Dependencies Found:

- Restaurant Name depends on only part of key (Restaurant ID only)
- Waiter Name depends on only part of key (Waiter ID only)
- Waiter Surname depends on only part of key (Waiter ID only)
- Employee Type depends on only part of key (Waiter ID only)
- Employee Type ID depends on only part of key (Waiter ID only)

Create New Tables:

text

RESTAURANT (RestaurantID PK, RestaurantName)

WAITER (WaiterID PK, WaiterName, WaiterSurname, EmployeeTypeID, EmployeeType)

EMPLOYMENT (RestaurantID FK, WaiterID FK, StartDate, EndDate)

Step 4: 3NF - Remove Transitive Dependencies
Transitive Dependency Found:

- In WAITER table: WaiterID → EmployeeTypeID, EmployeeType
- But EmployeeType depends on EmployeeTypeID (transitive!)

Fix:

text

EMPLOYEETYPE (EmployeeTypeID PK, EmployeeType)

WAITER (WaiterID PK, WaiterName, WaiterSurname, EmployeeTypeID FK)

EMPLOYMENT (RestaurantID FK, WaiterID FK, StartDate, EndDate)

RESTAURANT (RestaurantID PK, RestaurantName)

Final Dependency Diagram:

text

RestaurantID → RestaurantName

EmployeeTypeID → EmployeeType

WaiterID → WaiterName, WaiterSurname, EmployeeTypeID

(RestaurantID + WaiterID) → StartDate, EndDate

💻 SECTION 3: SQL - COMPLETE MASTERY

🎯 WHAT IS SQL AND WHY IS IT CRITICAL?

What the hell is SQL?
Structured Query Language - it's how you talk to databases. You're giving commands to create, read, update, and delete data.

Real World Example:
Think of SQL as the language you use to give orders to a very obedient but literal-minded robot. You have to be EXACT.

📚 WHERE THIS COMES FROM: 2024 EXAM QUESTIONS

ACTUAL 2024 QUESTIONS:
"Using MySQL, create a single Structured Query Language (SQL) script that answers all the below questions..."

Q3.1: Write an SQL statement to create the Patient table
Q3.2: Write an SQL statement to create the Doctor table
Q3.3: Write an SQL statement to create the Appointments table
Q3.4: Write SQL statements to insert the provided data
Q3.5: Display all appointments between 2024-01-16 and 2024-01-20
Q3.6: Display patient names with total appointment counts
Q3.7: Display all appointments with doctor and patient details
Q3.8: Create a view for patients of Doctor ID 2
Q3.9: Create a stored procedure to get appointments by date

🛠️ **COMPLETE SQL SOLUTION:**

Q3.1-3.3: Table Creation

sql

```
-- Q3.1: Patient table
CREATE TABLE Patient (
   PatientID INT AUTO_INCREMENT PRIMARY KEY,
   PatientName VARCHAR(50) NOT NULL,
   PatientSurname VARCHAR(50) NOT NULL,
   PatientDOB DATE NOT NULL
);

-- Q3.2: Doctor table
CREATE TABLE Doctor (
   DoctorID INT AUTO_INCREMENT PRIMARY KEY,
   DoctorName VARCHAR(50) NOT NULL,
   DoctorSurname VARCHAR(50) NOT NULL
);

-- Q3.3: Appointment table
CREATE TABLE Appointment (
   AppointmentID INT AUTO_INCREMENT PRIMARY KEY,
   PatientID INT NOT NULL,
   DoctorID INT NOT NULL,
   AppointmentDate DATE NOT NULL,
   AppointmentTime TIME NOT NULL,
   AppointmentDuration INT NOT NULL,
   FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
```

```
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
);
```

Q3.4: Insert Data

sql

```
-- Patients
INSERT INTO Patient (PatientID, PatientName, PatientSurname, PatientDOB) VALUES
(1, 'Debbie', 'Theart', '1980-03-17'),
(2, 'Thomas', 'Duncan', '1976-08-12');

-- Doctors
INSERT INTO Doctor (DoctorID, DoctorName, DoctorSurname) VALUES
(1, 'Zintle', 'Nukani'),
(2, 'Ravi', 'Maharaj');

-- Appointments
INSERT INTO Appointment (AppointmentID, PatientID, DoctorID, AppointmentDate,
AppointmentTime, AppointmentDuration) VALUES
(1, 1, 2, '2024-01-15', '09:00:00', 15),
(2, 2, 2, '2024-01-18', '15:00:00', 30),
(3, 1, 1, '2024-01-20', '10:00:00', 15),
(4, 1, 2, '2024-01-21', '11:00:00', 15);
```

Q3.5: Basic SELECT with Date Range

sql

```
SELECT * FROM Appointment
WHERE AppointmentDate BETWEEN '2024-01-16' AND '2024-01-20';
```

Q3.6: COUNT with GROUP BY

sql

```
SELECT
```

```
    p.PatientName,
    p.PatientSurname,
    COUNT(a.AppointmentID) AS TotalAppointments
FROM Patient p
LEFT JOIN Appointment a ON p.PatientID = a.PatientID
GROUP BY p.PatientID, p.PatientName, p.PatientSurname
ORDER BY TotalAppointments DESC;
```

Q3.7: JOIN Multiple Tables

```
sql
SELECT
    a.AppointmentDate,
    a.AppointmentTime,
    d.DoctorName,
    d.DoctorSurname,
    p.PatientName,
    p.PatientSurname
FROM Appointment a
JOIN Doctor d ON a.DoctorID = d.DoctorID
JOIN Patient p ON a.PatientID = p.PatientID
ORDER BY a.AppointmentDate DESC;
```

Q3.8: CREATE VIEW

```
sql
CREATE VIEW PatientsOfDoctor2 AS
SELECT DISTINCT
    p.PatientName,
    p.PatientSurname
FROM Appointment a
JOIN Patient p ON a.PatientID = p.PatientID
```

```
WHERE a.DoctorID = 2
ORDER BY p.PatientSurname ASC;
```

Q3.9: STORED PROCEDURE

```
sql
DELIMITER //
CREATE PROCEDURE get_appointments(IN inDate DATE)
BEGIN
  SELECT
    a.AppointmentTime,
    a.AppointmentDuration,
    d.DoctorName,
    d.DoctorSurname,
    p.PatientName,
    p.PatientSurname
  FROM Appointment a
  JOIN Doctor d ON a.DoctorID = d.DoctorID
  JOIN Patient p ON a.PatientID = p.PatientID
  WHERE a.AppointmentDate = inDate
  ORDER BY a.AppointmentTime ASC;
END //
DELIMITER ;
```

🔗 **SQL DATA TYPES EXPLAINED:**

Common Data Types:

- INT - whole numbers (PatientID, DoctorID)
- VARCHAR(n) - text up to n characters (names)
- DATE - dates only (YYYY-MM-DD)
- TIME - time only (HH:MM:SS)

- DATETIME - date and time together

Common Constraints:

- PRIMARY KEY - unique identifier for each record
- FOREIGN KEY - links to another table's primary key
- NOT NULL - must have a value
- AUTO_INCREMENT - automatically increases (for IDs)

☁️ SECTION 4: MONGODB - COMPLETE MASTERY

🎯 WHAT IS MONGODB AND WHY IS IT DIFFERENT?

What the hell is MongoDB?
A NoSQL database that stores data in JSON-like documents instead of tables. It's more flexible but less structured than SQL.

SQL vs MongoDB:

- **SQL:** Tables → Rows → Columns (rigid structure)
- **MongoDB:** Collections → Documents → Fields (flexible structure)

Real World Example:
SQL = filing cabinet with predefined forms
MongoDB = box of index cards where each card can have different info

---

📚 WHERE THIS COMES FROM: 2024 EXAM QUESTION

ACTUAL 2024 QUESTION:
"A medical practice wants to store patient details in a NoSQL database. Write MongoDB interactive shell commands to complete the task below.

Q4.1: Create a database called patients_<your-student-number>
Q4.2: In a collection called patients, create patient data:

- Debbie Theart, born 1980-03-17
- Thomas Duncan, born 1976-08-12
  Q4.3: Get a list of all the patients in the collection
  Q4.4: Query all the patients born after (not including) 1979-01-12"

🛠️ **COMPLETE MONGODB SOLUTION:**

METHOD 1: Command Line (mongosh)

javascript

```
// Q4.1: Create database
use patients_s12345678;

// Q4.2: Insert patients
db.patients.insertMany([
  {
    PatientName: "Debbie",
    PatientSurname: "Theart",
    DateOfBirth: ISODate("1980-03-17")
  },
  {
    PatientName: "Thomas",
    PatientSurname: "Duncan",
    DateOfBirth: ISODate("1976-08-12")
  }
]);

// Q4.3: Find all patients
db.patients.find().pretty();

// Q4.4: Patients born after 1979-01-12
db.patients.find({
  DateOfBirth: {$gt: ISODate("1979-01-12")}
}).pretty();
```

METHOD 2: MongoDB Compass GUI

text

1. Open MongoDB Compass

2. Click "Connect"

3. Click "+" next to Databases

4. Database Name: "patients_s12345678"

5. Collection Name: "patients"

6. Click "Create Database"

7. Click "ADD DATA" → "Insert Document"

8. Type the JSON documents

9. Click "Insert"

10. Use filter bar for queries

🔍 MONGODB OPERATORS EXPLAINED:

Comparison Operators:

- $eq = equal to (default)
- $gt = greater than
- $lt = less than
- $gte = greater than or equal
- $lte = less than or equal
- $ne = not equal to

Date Queries:

- ISODate("YYYY-MM-DD") - MUST use this for dates
- {$gt: ISODate("2020-01-01")} - after January 1, 2020
- {$lt: ISODate("2020-01-01")} - before January 1, 2020

⏱️ TIME MANAGEMENT - COMPLETE STRATEGY

2-HOUR BATTLE PLAN:

Minutes 0-5: Read Entire Paper

- Scan all questions
- Identify mark allocations
- Plan your attack

Minutes 5-30: ERD Section

- Use TABLE FORMAT (faster, clearer)
- Follow the 6 steps exactly
- Save/draw your diagram

Minutes 30-60: Normalization Section

- Identify composite primary key
- Split tables aggressively for 2NF
- Check for transitive dependencies for 3NF
- Draw dependency diagrams

Minutes 60-120: SQL Section

- Open MySQL Workbench
- Use EXACT templates for each question
- TEST each query as you go
- Save your script frequently

Minutes 120-135: MongoDB Section

- Use COMMAND LINE method
- Copy-paste the templates
- Save commands to text file

Minutes 135-140: Final Check

- Verify all files are saved
- Check nothing is missing
- Ensure student number in database names

🚨 COMMON MISTAKES & SOLUTIONS

ERD MISTAKES:

- ❌ **Forgetting bridge tables** for many-to-many relationships
- ✅ **SOLUTION:** Always check for "many-to-many" keywords
- ❌ **Wrong multiplicities**
- ✅ **SOLUTION:** Use the multiplicity cheat sheet
- ❌ **Missing primary keys**
- ✅ **SOLUTION:** Every table gets TableNameID PRIMARY KEY

SQL MISTAKES:

- ❌ **Forgetting commas** between fields
- ✅ **SOLUTION:** Copy templates exactly
- ❌ **Wrong data types**
- ✅ **SOLUTION:** Use VARCHAR for text, INT for numbers, DATE for dates
- ❌ **Forgetting FOREIGN KEY constraints**
- ✅ **SOLUTION:** Always add FOREIGN KEY references

NORMALIZATION MISTAKES:

- ❌ **Keeping partial dependencies**
- ✅ **SOLUTION:** Split tables aggressively
- ❌ **Not showing dependency diagrams**
- ✅ **SOLUTION:** Draw arrows showing dependencies

MONGODB MISTAKES:

- ❌ **Using strings instead of ISODate()**
- ✅ **SOLUTION:** Always use ISODate("YYYY-MM-DD")
- ❌ **Forgetting .pretty()**
- ✅ **SOLUTION:** Always use .pretty() for readable output

📋 EXAMPLE QUESTIONS & ANSWERS

🎯 ERD PRACTICE QUESTION:

QUESTION:
"Draw an ERD for a library system with these business rules:

- Each book has one publisher, each publisher has many books
- Each book can have multiple authors, each author can write multiple books
- Each book belongs to one category, each category has many books
- Store book title, ISBN, publication year
- Store publisher name and address
- Store author name and biography
- Store category name and description"

ANSWER:

text

PUBLISHER

PublisherID (PK)

PublisherName

PublisherAddress

AUTHOR

AuthorID (PK)

AuthorName

AuthorBio

CATEGORY

CategoryID (PK)

CategoryName

CategoryDescription

BOOK

BookID (PK)

BookTitle

ISBN

PublicationYear

PublisherID (FK)

CategoryID (FK)

BOOK_AUTHOR (Bridge Table)

BookID (FK, PK part)

AuthorID (FK, PK part)

🎯 **NORMALIZATION PRACTICE QUESTION:**

QUESTION:
"Normalize this 1NF table to 3NF:

StudentID	Student Name	CourseID	CourseName	InstructorID	Instructor Name	Grade
1	John Smith	C101	Database Design	I01	Dr. Jones	A
1	John Smith	C102	Web Development	I02	Dr. Smith	B+
2	Sarah Lee	C101	Database Design	I01	Dr. Jones	A-

"

ANSWER:

text

STUDENT (StudentID PK, StudentName)

COURSE (CourseID PK, CourseName, InstructorID FK)

INSTRUCTOR (InstructorID PK, InstructorName)

ENROLLMENT (StudentID FK, CourseID FK, Grade)

🎯 **SQL PRACTICE QUESTION:**

QUESTION:
"Write SQL to:

1. Create the above tables
2. Insert the sample data
3. Find all students who got A grades
4. Count how many students each instructor teaches"

ANSWER:

sql

```
-- 1. Create tables
CREATE TABLE STUDENT (
  StudentID INT PRIMARY KEY,
  StudentName VARCHAR(50)
);

CREATE TABLE INSTRUCTOR (
  InstructorID VARCHAR(3) PRIMARY KEY,
  InstructorName VARCHAR(50)
);

CREATE TABLE COURSE (
  CourseID VARCHAR(4) PRIMARY KEY,
  CourseName VARCHAR(50),
```

```
  InstructorID VARCHAR(3),
  FOREIGN KEY (InstructorID) REFERENCES INSTRUCTOR(InstructorID)
);

CREATE TABLE ENROLLMENT (
  StudentID INT,
  CourseID VARCHAR(4),
  Grade VARCHAR(2),
  PRIMARY KEY (StudentID, CourseID),
  FOREIGN KEY (StudentID) REFERENCES STUDENT(StudentID),
  FOREIGN KEY (CourseID) REFERENCES COURSE(CourseID)
);

-- 2. Insert data
INSERT INTO STUDENT VALUES (1, 'John Smith'), (2, 'Sarah Lee');
INSERT INTO INSTRUCTOR VALUES ('I01', 'Dr. Jones'), ('I02', 'Dr. Smith');
INSERT INTO COURSE VALUES ('C101', 'Database Design', 'I01'), ('C102', 'Web Development', 'I02');
INSERT INTO ENROLLMENT VALUES (1, 'C101', 'A'), (1, 'C102', 'B+'), (2, 'C101', 'A-');

-- 3. Students with A grades
SELECT s.StudentName, e.Grade
FROM STUDENT s
JOIN ENROLLMENT e ON s.StudentID = e.StudentID
WHERE e.Grade = 'A';

-- 4. Count students per instructor
SELECT i.InstructorName, COUNT(DISTINCT e.StudentID) as StudentCount
```

FROM INSTRUCTOR i

JOIN COURSE c ON i.InstructorID = c.InstructorID

JOIN ENROLLMENT e ON c.CourseID = e.CourseID

GROUP BY i.InstructorID, i.InstructorName;

🎯 **MONGODB PRACTICE QUESTION:**

QUESTION:
"Create MongoDB commands for:

1. Create database library_12345
2. Insert books collection with:
    - Title: "Database Design", Authors: ["Dr. Smith", "Dr. Jones"], Year: 2023
    - Title: "Web Development", Authors: ["Prof. Brown"], Year: 2022
3. Find all books
4. Find books published after 2022"

ANSWER:

javascript

```
// 1. Create database
use library_12345;


// 2. Insert books
db.books.insertMany([
  {
    Title: "Database Design",
    Authors: ["Dr. Smith", "Dr. Jones"],
    Year: 2023
  },
  {
    Title: "Web Development",
    Authors: ["Prof. Brown"],
```

```
    Year: 2022
  }
]);

// 3. Find all books
db.books.find().pretty();

// 4. Books after 2022
db.books.find({Year: {$gt: 2022}}).pretty();
```

🎯 **FINAL SCORE PREDICTION:**

Section	Max Marks	Your Score	Why You'll Get These Marks
ERD	20	19/20	Table format + bridge tables = almost perfect
Normalization	20	17/20	Aggressive splitting = most marks
SQL	75	65/75	Templates + testing = high marks
MongoDB	15	14/15	Command line + ISODate() = near perfect
TOTAL	**130**	**115/130**	**= 88% FINAL SCORE**

💪 **THE MINDSET FOR 90%+**

YOU HAVE THREE SIMPLE JOBS:

1. **FOLLOW** the steps exactly as written
2. **COPY** the templates exactly as provided

3. **MANAGE** your time exactly as allocated

YOU DON'T NEED TO:

- Be a database genius
- Understand theoretical concepts
- Be creative or innovative
- Figure anything out on your own

THE REAL SECRET:

They're not testing if you're smart. They're testing if you can **FOLLOW INSTRUCTIONS**. The templates are your instructions.

🚀 **YOUR BATTLE PLAN:**

BEFORE EXAM:

1. **Practice ERD** - Do 2 past papers using table format
2. **Practice SQL** - Copy templates until muscle memory
3. **Practice Normalization** - Split tables aggressively
4. **Practice MongoDB** - Use command line method
5. **Sleep 8 hours** - No all-nighters!

EXAM DAY:

1. **ERD FIRST** - 25 minutes, table format
2. **NORMALIZATION SECOND** - 25 minutes, split aggressively
3. **SQL THIRD** - 60 minutes, copy templates, TEST everything
4. **MONGODB LAST** - 10 minutes, command line, save to file

DURING EXAM:

- **STICK TO TIME** - Use a watch
- **USE TEMPLATES** - Don't think, just copy
- **TEST QUERIES** - Click that lightning bolt!
- **ATTEMPT EVERYTHING** - Partial marks > zero
- **DON'T PANIC** - You have all the answers

📞 **EMERGENCY PROTOCOLS:**

IF YOU DON'T UNDERSTAND A QUESTION:

- Look for nouns = entities/tables
- Look for verbs = relationships
- Use the templates anyway
- **WRITE SOMETHING**

IF SQL WON'T RUN:

- Check commas between fields
- Check quotes around text
- Check semicolons at ends
- Test one line at a time

IF RUNNING OUT OF TIME:

- Skip to next section
- Do high-mark easy questions first
- **NEVER LEAVE BLANK ANSWERS**

IF YOU COMPLETELY BLANK:

- Copy the closest template
- Change names to match question
- Submit it anyway