? 1. One-to-One Relationship

? Explanation:

A One-to-One relationship means that one record in Table A is linked to only one record in Table B, and vice versa.

? Example:

Imagine you have two tables:

- Employees
- ID_Cards

Each employee has one ID card, and each ID card belongs to only one employee.

? SQL Implementation:

sql

```
CREATE TABLE employees (
employee_id INT PRIMARY KEY,
name VARCHAR(100)
);

CREATE TABLE id_cards (
card_id INT PRIMARY KEY,
employee_id INT UNIQUE,
issue_date DATE,
FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
);
```

? Benefits:

- Separating sensitive data for better security.
- Cleaner database structure.

? 2. One-to-Many Relationship

? Explanation:

A One-to-Many relationship means one record in Table A is related to many records in Table B, but each record in Table B is related to only one record in Table A.

? Example:

Imagine two tables:

- Teachers

- Students

One teacher can have many students, but each student is assigned to one teacher only.

? SQL Implementation:

```sql
CREATE TABLE teachers (
teacher_id INT PRIMARY KEY,
name VARCHAR(100)
);

CREATE TABLE students (
student_id INT PRIMARY KEY,
name VARCHAR(100),
teacher_id INT,
FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id)
);
```

? Benefits:

- Efficient management of parent-child data structures.

- Easier data retrieval and analysis.

? 3. Many-to-Many Relationship

? Explanation:

A *Many-to-Many* relationship means *multiple records* in *Table A* can relate to *multiple records* in *Table B*. You must use a *junction (intermediate) table* to manage this.

? Example:

Imagine two tables:

- `Students`

- `Courses`

Each student can enroll in many courses, and each course can have many students.

? SQL Implementation:

sql

CREATE TABLE students (

student_id INT PRIMARY KEY,

name VARCHAR(100)

);

CREATE TABLE courses (

course_id INT PRIMARY KEY,

title VARCHAR(100)

);

CREATE TABLE student_courses (

student_id INT,

course_id INT,

PRIMARY KEY (student_id, course_id),

FOREIGN KEY (student_id) REFERENCES students(student_id),

FOREIGN KEY (course_id) REFERENCES courses(course_id)

? Benefits:

- Allows flexible and powerful data linking.

- Ideal for systems that handle enrollments, memberships, tags, etc.