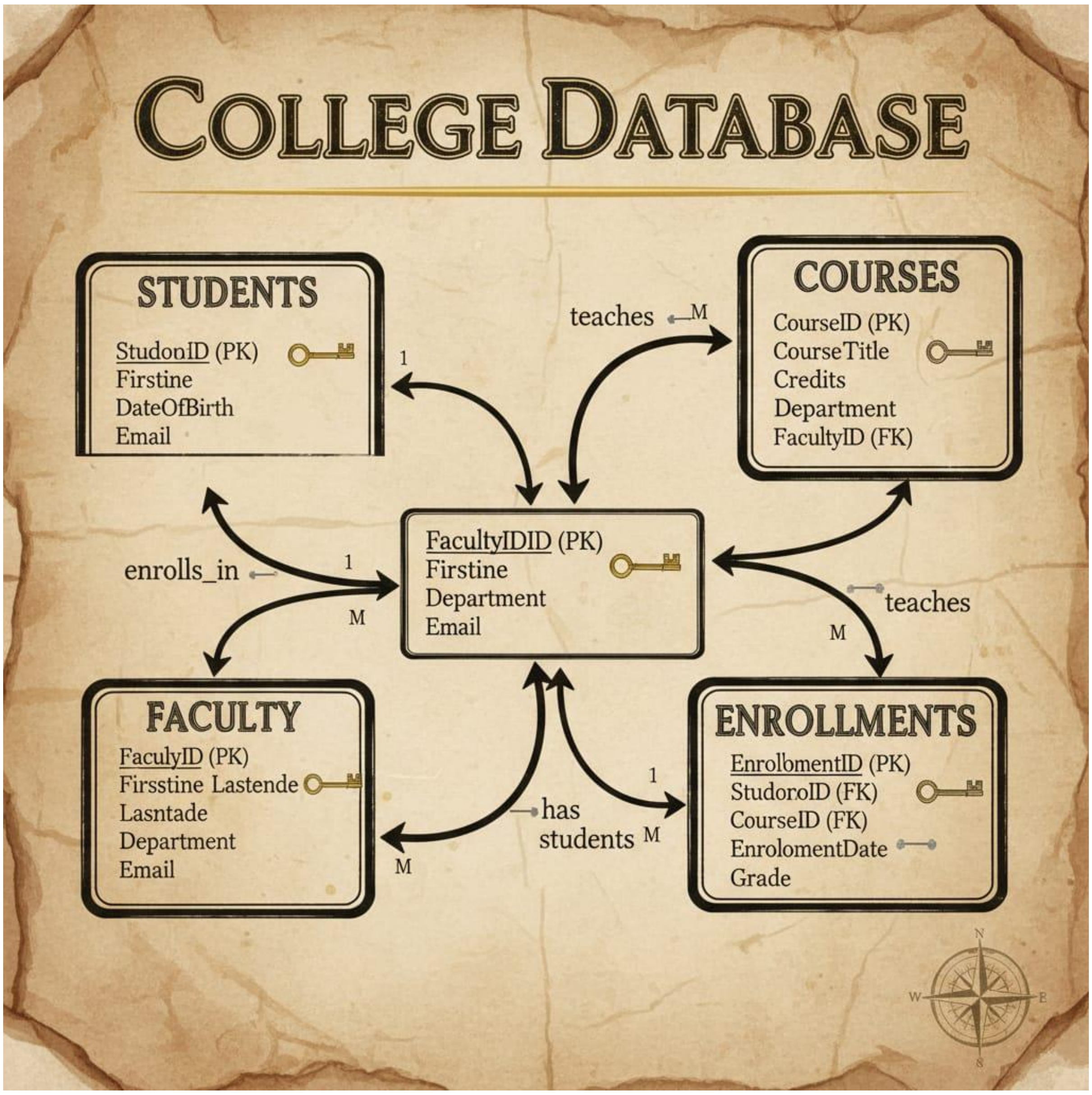


```
-- Students table
CREATE TABLE students (
    student_id SERIAL PRIMARY KEY,
    student_name VARCHAR(50) NOT NULL,
    dept VARCHAR(30),
    year_of_study INT,
    marks INT
);
```

```
-- Courses table
CREATE TABLE courses (
    course_id SERIAL PRIMARY KEY,
    course_name VARCHAR(50) NOT NULL,
    credits INT
);

-- Faculty table
CREATE TABLE faculty (
    faculty_id SERIAL PRIMARY KEY,
    faculty_name VARCHAR(50) NOT NULL,
    department VARCHAR(30)
);

-- Enrollments (Bridge table: Student ↔ Course)
CREATE TABLE enrollments (
    enroll_id SERIAL PRIMARY KEY,
    student_id INT REFERENCES students(student_id),
    course_id INT REFERENCES courses(course_id),
    semester VARCHAR(10)
);



commit
```

```
select * from students;



select * from courses;



select * from faculty;



select * from enrollments;

-- Students
INSERT INTO students(student_name, dept, year_of_study, marks)
VALUES ('Rohit', 'CSE', 2, 85),
       ('Anjali', 'ECE', 3, 76),
       ('Kiran', 'CSE', 1, 92);

-- Courses
INSERT INTO courses(course_name, credits)
VALUES ('Database Systems', 4),
       ('Operating Systems', 3),
       ('Electronics', 3);

-- Faculty
INSERT INTO faculty(faculty_name, department)
VALUES ('Dr. Rao', 'CSE'),
       ('Prof. Meena', 'ECE');

-- Enrollments
```

```
INSERT INTO enrollments(student_id, course_id, semester)
VALUES (1, 1, 'SEM-2'),
       (1, 2, 'SEM-2'),
       (2, 3, 'SEM-5'),
       (3, 1, 'SEM-1');
```

🎓 Workflow (Without Joins)

◈ Step 1: Student Admission (Insert new student)

```
INSERT INTO students(student_name, dept, year_of_study, marks)
VALUES ('Ravi', 'CSE', 1, 78);
```

📌 Scenario: A new student is getting admitted.

◈ Step 2: Filtering Students (WHERE)

Example – Normal:

```
SELECT * FROM students WHERE dept = 'CSE';
```

Real-Time Scenarios:

1. View all CSE students.
2. View only 1st year students.

```
SELECT student_name, marks FROM students WHERE year_of_study = 1;
```

3. Find students with marks < 40 and send them to remedial classes.

```
SELECT student_name, marks FROM students WHERE marks < 40;
```

◈ Step 3: Updating Records (UPDATE)

Example – Normal:

UPDATE students SET marks = 90 WHERE student_id = 2;

Real-Time Scenarios:

1. Update marks after Revaluation.

```
UPDATE students
SET marks = 85
WHERE student_name = 'Rohit';
```

2. Give Grace Marks (e.g., add +5 marks to all 3rd year CSE students).

```
UPDATE students
SET marks = marks + 5
WHERE dept = 'CSE' AND year_of_study = 3;
```

3. Change Department (student transfer).

```
UPDATE students
SET dept = 'IT'
WHERE student_name = 'Ravi';
```

◈ Step 4: Deleting Records (DELETE)

Example – Normal:

DELETE FROM students WHERE student_id = 3;

Real-Time Scenarios:

1. Delete a student who has dropped out.

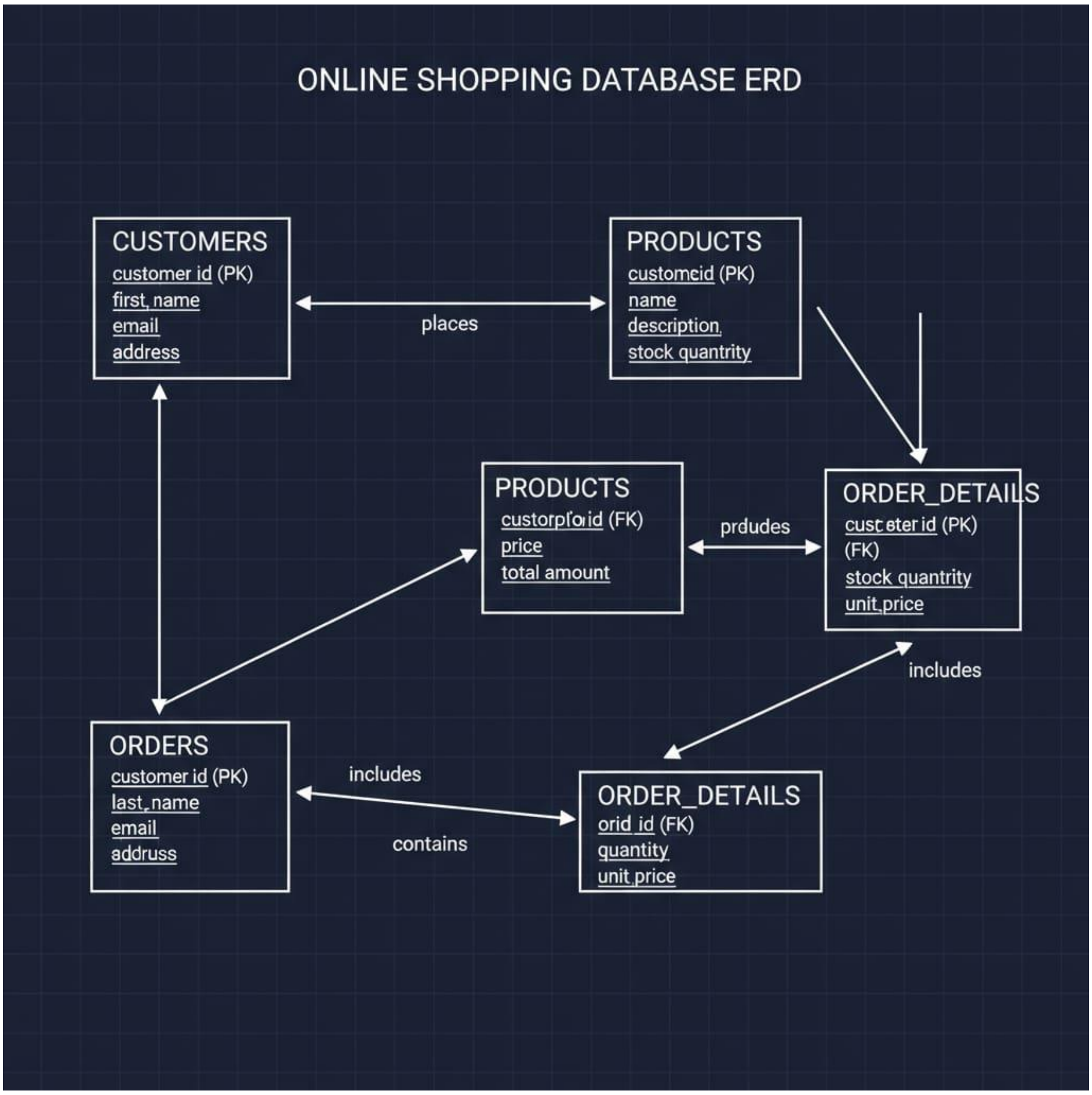
DELETE FROM students WHERE student_name = 'Kiran';

2. Remove a discontinued course from the courses table.

DELETE FROM courses WHERE course_name = 'Electronics';

3. Remove old student records (students who are in 4th year and completed degree).

DELETE FROM students WHERE year_of_study = 4;



```
-- Customer Table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(15),
    CreatedDate DATE DEFAULT CURRENT_DATE
);

-- Product Table
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    Price DECIMAL(10,2) NOT NULL,
    Stock INT CHECK (Stock >= 0)
);

-- Orders Table
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    TotalAmount DECIMAL(12,2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- Order Details (Many-to-Many relationship between Orders & Products)
CREATE TABLE OrderDetails (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT CHECK (Quantity > 0),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);



INSERT INTO Customers (CustomerID, FullName, Email, Phone)
VALUES (101, 'Ramesh Kumar', 'ramesh@example.com', '9876543210');

INSERT INTO Orders (OrderID, CustomerID, TotalAmount)
VALUES (5001, 101, 2500.00);

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity)
VALUES (1, 5001, 201, 2);

UPDATE Products SET Stock = Stock - 2 WHERE ProductID = 201;
```

Scenario: Generate report for MNC **Daily sales order**

```
SELECT OrderDate::date AS OrderDay, SUM(TotalAmount) AS DailySales
FROM Orders
GROUP BY OrderDay
ORDER BY OrderDay DESC;
```

Scenario: Generate report for MNC **Top selling Products**

```
SELECT p.ProductName, SUM(od.Quantity) AS TotalSold
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY TotalSold DESC
LIMIT 5;
```