

ACCA7002 Database Systems

Tutorial Tasks

Tutorial 4:

Open PostgreSQL in your computers or visit

<https://onecompiler.com/postgresql/3zpdm24rn>

4.1 Creating Tables with Primary and Foreign Keys

Create two tables - Students and Courses. Assign appropriate primary keys and establish a foreign key relationship between them.

-- Create Students table

```
CREATE TABLE Students (  
    student_id SERIAL PRIMARY KEY,  
    student_name VARCHAR(100)  
    -- Other attributes...  
);
```

-- Create Courses table

```
CREATE TABLE Courses (  
    course_id SERIAL PRIMARY KEY,  
    course_name VARCHAR(100)  
    -- Other attributes...  
);
```

-- Establish foreign key relationship

```
ALTER TABLE Courses  
ADD CONSTRAINT fk_student_id  
FOREIGN KEY (student_id)  
REFERENCES Students(student_id);
```

4.2 CRUD SQL Statements

Perform CRUD (Create, Read, Update, Delete) operations on the Students and Courses tables.

Create 10 records for student table and 10 records for Course table

-- Create a new student

```
INSERT INTO Students (student_name)
VALUES ('John Doe');
```

-- Create a new course

```
INSERT INTO Courses (course_name)
VALUES ('Math');
```

-- Read all students

```
SELECT * FROM Students;
```

-- Read all courses

```
SELECT * FROM Courses;
```

-- Update a student's name

```
UPDATE Students
SET student_name = 'Jane Smith'
WHERE student_id = 1;
```

-- Update a course's name

```
UPDATE Courses
SET course_name = 'History'
WHERE course_id = 1;
```

-- Delete a student

```
DELETE FROM Students
WHERE student_id = 1;
```

-- Delete a course

```
DELETE FROM Courses
WHERE course_id = 1;
```

4.3 Simple SQL Queries

Write simple SQL queries to retrieve information from the Students and Courses tables.

-- Retrieve all students

```
SELECT * FROM Students;
```

-- Retrieve all courses

```
SELECT * FROM Courses;
```

-- Retrieve students taking a specific course

```
SELECT s.student_name  
FROM Students s  
JOIN Courses c ON s.student_id = c.student_id  
WHERE c.course_name = 'Math';
```

4.4 Table Joins

Perform an equi join between the Students and Courses tables.

```
SELECT s.student_name, c.course_name  
FROM Students s  
JOIN Courses c ON s.student_id = c.student_id;
```

Perform a left join between the Students and Courses tables.

```
SELECT s.student_name, c.course_name  
FROM Students s  
LEFT JOIN Courses c ON s.student_id = c.student_id;
```

Right Join

Perform a right join between the Students and Courses tables.

```
SELECT s.student_name, c.course_name  
FROM Students s  
RIGHT JOIN Courses c ON s.student_id = c.student_id;
```

4.5 Grouping

Calculate the number of students enrolled in each course.

```
SELECT c.course_name, COUNT(*) AS student_count
FROM Students s
JOIN Courses c ON s.student_id = c.student_id
GROUP BY c.course_name;
```

4.6 Indexes

Create an index on the course_name column in the Courses table.

```
CREATE INDEX course_name_index ON Courses
(course_name);
```

4.7 Drop Tables

Drop the Students and Courses tables.

```
DROP TABLE Students, Courses;
```