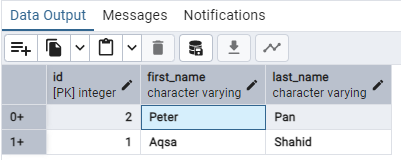
# Lab 12 - Postgresql

Student's name: Aqsa Shahid

Lab- 12A Tuesday, April 29, 2025 (introduction)

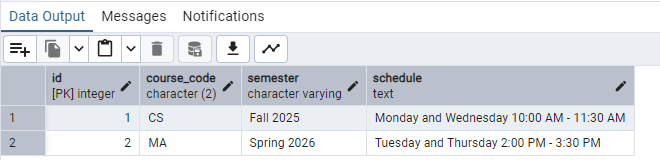
Class Example:

User Table

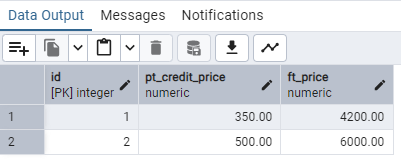


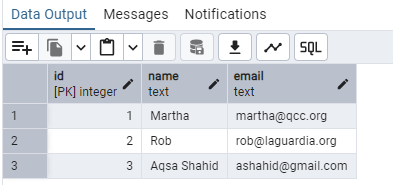
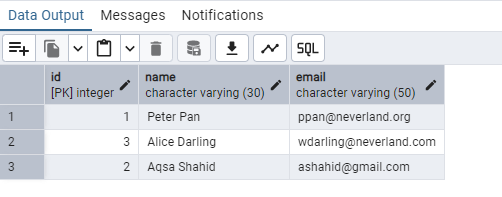
Lab exercise: Manually create a database, table, and columns

Courses Table



Tuition Table



Lab- 12B Thursday, May 1st, 2025 (CRUD)

EXERCISE:

1. Create the Table

--1. CREATE A TABLE

CREATE TABLE books(

id SERIAL PRIMARY KEY,

title VARCHAR(100) NOT NULL,

author VARCHAR(100) NOT NULL,

year\_published INT);

2. Insert Records

--2. INSERT DATA

INSERT INTO books(title, author,year\_published)

VALUES

('The Great Gatsby','F. Scott Fitzgerald','1925'),

('Animal Farm','George Orwell','1945'),

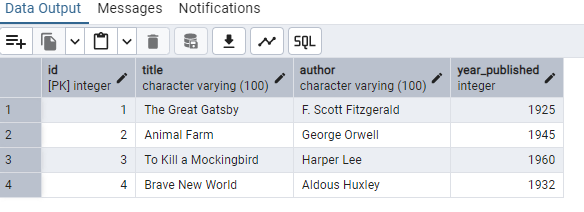
('To Kill a Mockingbird','Harper Lee','1960'),

('Brave New World','Aldous Huxley','1932');

3. Read Data

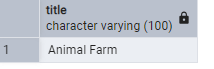
--3. READ

SELECT \* FROM books



SELECT title FROM books

WHERE author = 'George Orwell';



4. Update Data

-- UPDATE, year\_published of Brave New World to 1931

UPDATE books

SET year\_published='1931'

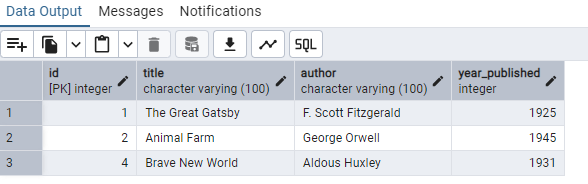
WHERE author='Aldous Huxley';

5. Delete Data

--5. DELETE

DELETE FROM books

WHERE title ='To Kill a Mockingbird';



Bonus Challenge (Optional)

• Add a new column genre to the books table.

ALTER TABLE books ADD COLUMN genre VARCHAR(50);

• Update the genre for each book.

UPDATE books

SET genre = 'Realist'

WHERE id = 1;

UPDATE books

SET genre = 'Fiction'

WHERE id = 2;

UPDATE books

SET genre = 'Mystery'

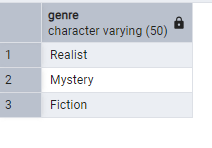
WHERE id = 4;

• Select all books grouped by genre.

SELECT genre

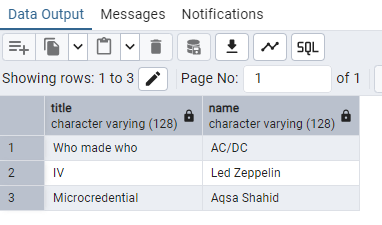
FROM books

GROUP BY genre



Lab- 12C FRIDAY, May 2nd, 2025

Class Practice:



Exercise:

Create Tables:

1. Students:

-- First check if table exists and drop it if needed

DROP TABLE IF EXISTS students;

--Create 'students' table

CREATE TABLE students (

id SERIAL PRIMARY KEY,

name VARCHAR(100),

major VARCHAR(100),

);

Courses:

-- First check if courses exists and drop it if needed

DROP TABLE IF EXISTS courses;

--Create 'courses' table

CREATE TABLE courses (

id SERIAL PRIMARY KEY,

title VARCHAR(100),

department VARCHAR(100)

);

Enrollments:

-- Create 'enrollments' table

CREATE TABLE enrollments(

id SERIAL PRIMARY KEY,

grade VARCHAR(2),

student\_id INTEGER REFERENCES students(id) ON DELETE CASCADE,

course\_id INTEGER REFERENCES courses(id) ON DELETE CASCADE

);

INSERT DATA:

1. Students:

-- Insert data into students table

INSERT INTO students (name, major) VALUES

('Alice Brown', 'Computer Science'),

('Peter Pan', 'Mathematics'),

('Annie Chen', 'Physics'),

('Aqsa Shahid', 'Mathematics'

);

1. Courses:

-- Insert data into courses table

INSERT INTO courses (title, department) VALUES

('Database Systems', 'Computer Science'),

('Linear algebra', 'Mathematics'),

('Quantum mechanics', 'Physics'),

('Python programming', 'computer science'),

('Calculus III', 'mathematics'

);

1. Enrollments:

INSERT INTO enrollments(id, grade, student\_id, course\_id)

VALUES

(1,'A',4, 2 ),

(2,'B',3, 3 ),

(3,'C',2, 5 ),

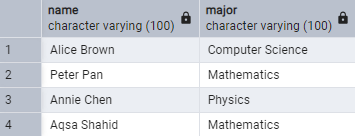
(4,'B+',1, 4 );

• Queries to practice

-- Query 1: Read all students and their major

SELECT students.name, students.major

FROM students;



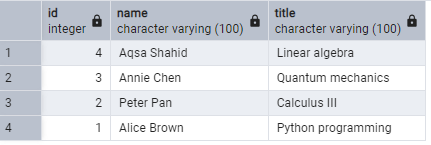
-- Query 2: Read the titles of courses each student is enrolled in

SELECT students.id, students.name, courses.title

FROM students

JOIN enrollments ON students.id = enrollments.student\_id

JOIN courses ON enrollments.course\_id = courses.id;



-- Query 3: Read all students with their grades and courses

SELECT students.id, students.name, courses.title, courses.department, enrollments.grade

FROM students

JOIN enrollments ON students.id = enrollments.student\_id

JOIN courses ON enrollments.course\_id = courses.id;

