Topics:

1. **UNION / UNION ALL**
2. **DROP vs DELETE vs TRUNCATE**
3. **Subqueries** *(exploratory task – they search and try it)*
4. **Transaction & Batch Script** *(exploratory and guided)*
5. **Hands-on comparison with real effect on data*

Practice Scenario: Training & Job Application System

Your institute is managing two main datasets:

- **Trainees**: People who complete training at your institute.
- **Job Applicants**: External applicants who apply directly to job posts.

Your goal is to:

- Compare the data of both groups.
- Clean or restructure the database safely.
- Explore more advanced SQL topics on your own (subqueries, transactions).

Tables

```
-- Trainees Table

CREATE TABLE Trainees (

   TraineeID INT PRIMARY KEY,

   FullName VARCHAR(100),

   Email VARCHAR(100),

   Program VARCHAR(50),

   GraduationDate DATE

);
```

```
-- Job Applicants Table
CREATE TABLE Applicants (
    ApplicantID INT PRIMARY KEY,
    FullName VARCHAR(100),
    Email VARCHAR(100),
    Source VARCHAR(20), -- e.g., "Website", "Referral"
    AppliedDate DATE
);
```

Sample Data

```
-- Insert into Trainees
INSERT INTO Trainees VALUES
(1, 'Layla Al Riyami', 'layla.r@example.com', 'Full Stack .NET', '2025-04-30'),
(2, 'Salim Al Hinai', 'salim.h@example.com', 'Outsystems', '2025-03-15'),
(3, 'Fatma Al Amri', 'fatma.a@example.com', 'Database Admin', '2025-05-01');

-- Insert into Applicants
INSERT INTO Applicants VALUES
(101, 'Hassan Al Lawati', 'hassan.l@example.com', 'Website', '2025-05-02'),
(102, 'Layla Al Riyami', 'layla.r@example.com', 'Referral', '2025-05-05'), -- same person as
trainee
(103, 'Aisha Al Farsi', 'aisha.f@example.com', 'Website', '2025-04-28');
```

Part 1: UNION Practice

1. **List all unique people who either trained or applied for a job.**
    - Show their full names and emails.
    - Use UNION (not UNION ALL) to avoid duplicates.
2. **Now use UNION ALL. What changes in the result?**
    - Explain why one name appears twice.
3. **Find people who are in both tables.**
    - You must use INTERSECT if supported, or simulate it using INNER JOIN on Email.

Part 2: DROP, DELETE, TRUNCATE Observation

Let's test destructive commands.

4. Try DELETE FROM Trainees WHERE Program = 'Outsystems'.
    - Check if the table structure still exists.
5. Try TRUNCATE TABLE Applicants.
    - What happens to the data? Can you roll it back?
6. Try DROP TABLE Applicants.
    - What happens if you run a SELECT after that?

Write your observations after each command.

Part 3: Self-Discovery & Applied Exploration

In this section, you'll independently **research**, **experiment**, and **apply** advanced SQL concepts. Follow the guided prompts below.

Subquery Exploration

Goal: Understand what a subquery is and how it's used inside SQL commands.

1. Research:
    - What is a subquery in SQL?
    - Where can we use subqueries? (e.g., in SELECT, WHERE, FROM)
2. Task:
    - Write a query to find all trainees whose emails appear in the applicants table.
    - You must use a subquery inside a WHERE clause.
3. Extra Challenge:
    - Write a DML statement (like UPDATE or DELETE) that uses a subquery in the WHERE clause.
    - Example: Delete all applicants whose email matches someone in the trainees table.

Batch Script & Transactions

Goal: Understand how to safely execute multiple SQL statements as a unit.

4. Research:
    - What is a **SQL transaction**?
    - How to write transaction blocks in your database tool (BEGIN TRANSACTION, COMMIT, ROLLBACK)?
5. Task:
    - Write a script that:
        - Starts a transaction
        - Tries to insert two new applicants
        - The second insert should have a **duplicate ApplicantID** (to force failure)
        - Rollback the whole transaction if any error occurs

6. Add this logic:

```
BEGIN TRANSACTION;

INSERT INTO Applicants VALUES (104, 'Zahra Al Amri', 'zahra.a@example.com', 'Referral', '2025-05-10');

INSERT INTO Applicants VALUES (104, 'Error User', 'error@example.com', 'Website', '2025-05-11'); -- Duplicate ID

COMMIT;

-- Or use ROLLBACK if needed
```

ACID Properties Exploration

Goal: Learn the theory behind reliable transactions.

7. Research and summarize each of the **ACID** properties:
    - **Atomicity**
    - **Consistency**
    - **Isolation**
    - **Durability**
8. For each property, write a **real-life example** that explains it in your own words.
9.

GitHub Instructions

- Create a new repository: SQL-AdvancedPractice
- Write your scripts + observation notes in a single .sql file
- Use meaningful commit messages like:
    - Practiced UNION vs UNION ALL
    - Tested DELETE vs DROP vs TRUNCATE