Advanced SQL Queries - Part 1

JOIN and UNION

Lesson Objectives (Part 1)

- Understand how **JOIN** works
- Execute **UNION** between tables
- Run queries that combine multiple tables
- Recognize typical use cases

JOIN - Introduction

Used to combine rows from two or more tables based on a common condition

Types of JOIN

- `INNER JOIN`: rows with a match in both tables
- `LEFT JOIN`: all rows from the left table + matches
- `RIGHT JOIN`: all rows from the right table + matches
- `FULL JOIN`: all rows from both tables
- `CROSS JOIN`: Cartesian product
- `SELF JOIN`: join of the table with itself

Generic Syntax - INNER JOIN

```
SELECT t1.col1, t2.col2
FROM table1 t1
INNER JOIN table2 t2
ON t1.common_col = t2.common_col;
```

Example - INNER JOIN

```
SELECT u.name, l.loan_date
FROM Loans l
INNER JOIN Users u
ON l.user_id = u.id;
```

Generic Syntax - LEFT JOIN

```
SELECT t1.col1, t2.col2
FROM table1 t1
LEFT JOIN table2 t2
ON t1.common_col = t2.common_col;
```

Example - LEFT JOIN

```
SELECT b.title, l.loan_date
FROM Books b
LEFT JOIN Loans l
ON b.id = l.book_id;
```

Generic Syntax - RIGHT JOIN

```
SELECT t1.col1, t2.col2
FROM table1 t1
RIGHT JOIN table2 t2
ON t1.common_col = t2.common_col;
```

SQLite - RIGHT JOIN

📌 **Compatibility:** `RIGHT JOIN` requires SQLite ≥ 3.39.0 (July 2022). On older versions, swap the two tables and use `LEFT JOIN` instead — the result is equivalent.

```sql
-- Equivalent of: table1 RIGHT JOIN table2
SELECT t2.col2, t1.col1
FROM table2 t2
LEFT JOIN table1 t1
ON t2.common_col = t1.common_col;
```

Example - RIGHT JOIN

```
SELECT b.title, l.loan_date
FROM Books b
RIGHT JOIN Loans l
ON b.id = l.book_id;
```

SQLite - RIGHT JOIN Example

📌 **Alternative with LEFT JOIN** (SQLite < 3.39.0): swap the tables so the one that must appear in full becomes the left table.

```
SELECT b.title, l.loan_date
FROM Loans l
LEFT JOIN Books b
ON l.book_id = b.id;
```

Generic Syntax - FULL JOIN

```
SELECT t1.col1, t2.col2
FROM table1 t1
FULL JOIN table2 t2
ON t1.common_col = t2.common_col;
```

SQLite - FULL JOIN

📌 **Compatibility:** `FULL JOIN` requires SQLite ≥ 3.39.0 (July 2022). On older versions, emulate it with a `UNION` of two `LEFT JOIN`s – one for each direction.

```
SELECT t1.col1, t2.col2
FROM table1 t1 LEFT JOIN table2 t2 ON t1.common_col = t2.common_col
UNION
SELECT t1.col1, t2.col2
FROM table2 t2 LEFT JOIN table1 t1 ON t2.common_col = t1.common_col;
```

Example - FULL JOIN

```
SELECT b.title, l.loan_date
FROM Books b
FULL JOIN Loans l
ON b.id = l.book_id;
```

SQLite - FULL JOIN Example

📌 **Workaround with UNION of two LEFT JOINs** (SQLite < 3.39.0):

```
SELECT b.title, l.loan_date
FROM Books b LEFT JOIN Loans l ON b.id = l.book_id
UNION
SELECT b.title, l.loan_date
FROM Loans l LEFT JOIN Books b ON l.book_id = b.id;
```

Generic Syntax - CROSS JOIN

```
SELECT *
FROM table1
CROSS JOIN table2;
```

Example - CROSS JOIN

```
SELECT u.name, b.title
FROM Users u
CROSS JOIN Books b;
```

Generic Syntax - SELF JOIN

```
SELECT a.col1, b.col1
FROM table a
JOIN table b
ON a.common_col = b.common_col_condition;
```

Example - SELF JOIN

```
SELECT a.name, b.name
FROM Users a
JOIN Users b
ON a.user_type = b.user_type
WHERE a.id < b.id;
```

UNION - Introduction

Used to combine results from two or more SELECT statements with the same number of columns and compatible types

Components of `UNION`

- Two or more `SELECT` queries
- Identical number and type of columns
- `UNION` removes duplicates
- `UNION ALL` keeps duplicates

Generic Syntax - UNION

```
SELECT col1 FROM table1
UNION
SELECT col1 FROM table2;
```

Example - UNION

```
SELECT email FROM Users WHERE user_type = 'student'
UNION
SELECT email FROM Users WHERE user_type = 'lecturer';
```