SQL Basics Part 5

This PDF about JOINS

```
/*
	=============
	== 20 JOIN ==
	=============
	- JOINS are used to combine data from two or more tables based on a related
column between them
	- they help us answer questions like:
		1- get all customers and their orders
		2- list employees and departments names
	- without JOINs, you'd have to manually fetch data from each table

	----------------
	-- JOIN Types --
	----------------
	- INNER JOIN:
	- LEFT JOIN:
	- RIGHT JOIN:
	- FULL JOIN:
*/

/*
	====================
	== 01 INNER Join ==
	====================
	- the most common Join use between developers
	- used to combine rows from two or more tables based on related columns
between them
	- it only return rows where there's a match in both tables
		If a row exists in one table but not in the other → it will be
excluded.
		It’s the most commonly used join in SQL.

	- syntax
		SELECT table1.column1, table1.column2, table2.column1, table2.column2
		FROM table1
		INNER JOIN table2
			ON table1.common_column = table2.common_column;
*/

-- list all customers ID, first name, order ID, amount, exclude the customers who
have not placed aby order
SELECT
	c.customer_id,
	c.first_name,
	o.order_id,
	o.amount
FROM customers AS c
INNER JOIN orders AS o
ON c.customer_id = o.customer_id
ORDER BY c.customer_id ASC;
```

```sql
-- we'll display customer name, country, total score, and order amount
SELECT
    c.first_name,
    c.country,
    c.score,
    o.order_id,
    o.amount
FROM customers AS c
INNER JOIN orders AS o
    ON c.customer_id = o.customer_id
ORDER BY o.amount DESC;


-- show full orders details
SELECT
    c.first_name,
    c.country,
    o.order_id,
    o.order_date,
    o.amount
FROM customers AS c
INNER JOIN orders AS o
    ON c.customer_id = o.customer_id
ORDER BY o.amount DESC;

-- count orders per customer
SELECT
    c.first_name,
    COUNT(o.order_id) AS total_orders
FROM customers AS c
INNER JOIN orders AS o
    ON c.customer_id = o.customer_id
GROUP BY c.first_name
ORDER BY total_orders DESC;
```

```sql
/*
    ==================
    == 21 LEFT JOIN ==
    ==================
    - all rows from the left table matching from the right table
        if there's no match, it still show the left table, and write the right table
rows as NULL
    - syntax
        LEFT JOIN = All Left Table Rows + Matching Right Table Rows
                If no match → NULL for right table columns
*/

use MyDatabase;

-- list customer ID, first name, oreder ID, amount. exclude the customers who have
no placed any orders
SELECT
```

```sql
    c.customer_id,
    c.first_name,
    o.order_id,
    o.amount
FROM customers AS c
LEFT JOIN orders AS o
    ON c.customer_id = o.customer_id
ORDER BY c.customer_id;


-- Example 2 - Find Customers Without Orders (LEFT JOIN + WHERE)
SELECT
    c.customer_id,
    c.first_name,
    c.country
FROM customers AS c
LEFT JOIN orders AS o
ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;


-- Example 3 - Count Total Orders per Customer (LEFT JOIN + GROUP BY)
SELECT
    c.customer_id,
    c.first_name,
    COUNT(o.order_id) AS total_orders
FROM customers AS c
LEFT JOIN orders AS o
ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.first_name
ORDER BY total_orders DESC;


-- Example 4 - Show Customers and Latest Order Amount (LEFT JOIN + MAX)
SELECT
    c.customer_id,
    c.first_name,
    MAX(o.amount) AS latest_order
FROM customers AS c
LEFT JOIN orders AS o
ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.first_name
ORDER BY latest_order DESC;
```

```sql
/*
    ===================
    == 22 RIGHT JOIN ==
    ===================
    - A RIGHT JOIN returns all rows from the right table (e.g., orders),
        and the matching rows from the left table (e.g., customers).
    - If there's no match in the left table, the result will contain NULL for
columns from the left table.
*/

USE MyDatabase;
```

```sql
-- Show me all orders even if there is no matching customer
SELECT
    c.customer_id,
    c.first_name,
    o.order_id,
    o.amount
FROM customers AS c
RIGHT JOIN orders AS o
ON c.customer_id = o.customer_id
ORDER BY o.order_id;


-- Example 1 - Get All Orders with Customer Names ✅ (Basic RIGHT JOIN)
SELECT
    o.order_id,
    o.amount,
    c.customer_id,
    c.first_name
FROM customers AS c
RIGHT JOIN orders AS o
ON c.customer_id = o.customer_id
ORDER BY o.order_id;


-- Example 2 - Find Orders Without a Customer (RIGHT JOIN + WHERE)
SELECT
    o.order_id,
    o.amount,
    o.customer_id
FROM customers AS c
RIGHT JOIN orders AS o
ON c.customer_id = o.customer_id
WHERE c.customer_id IS NULL;


-- Example 3 - Count Total Orders Per Customer (RIGHT JOIN + GROUP BY)
SELECT
    o.customer_id,
    c.first_name,
    COUNT(o.order_id) AS total_orders
FROM customers AS c
RIGHT JOIN orders AS o
ON c.customer_id = o.customer_id
GROUP BY o.customer_id, c.first_name
ORDER BY total_orders DESC;
```

```sql
/*
    ==================
    == 23 FULL JOIN ==
    ==================
    - A FULL JOIN (or FULL OUTER JOIN) combines the results of LEFT JOIN and RIGHT
JOIN
        It returns all rows from both tables.
        If there's a match → it shows data from both tables.
        If there's no match → it shows NULL for the missing values.
*/
```

```sql
USE MyDatabase;


-- Find Customers Without Orders or Orders Without Customers [MySQL Syntax]
SELECT
    c.customer_id,
    c.first_name,
    o.order_id,
    o.amount
FROM customers AS c
LEFT JOIN orders AS o
ON c.customer_id = o.customer_id

UNION

SELECT
    c.customer_id,
    c.first_name,
    o.order_id,
    o.amount
FROM customers AS c
RIGHT JOIN orders AS o
ON c.customer_id = o.customer_id
ORDER BY customer_id;


-- Find Only Unmatched Rows
SELECT
    c.customer_id,
    c.first_name,
    o.order_id,
    o.amount
FROM customers AS c
LEFT JOIN orders AS o
ON c.customer_id = o.customer_id
WHERE o.customer_id IS NULL

UNION

SELECT
    c.customer_id,
    c.first_name,
    o.order_id,
    o.amount
FROM customers AS c
RIGHT JOIN orders AS o
ON c.customer_id = o.customer_id
WHERE c.customer_id IS NULL;


-- Find Only Unmatched Rows in SQL Server
SELECT
    c.customer_id,
    c.first_name,
    o.order_id,
    o.amount
FROM customers AS c
```

```sql
FULL JOIN orders AS o
ON c.customer_id = o.customer_id
WHERE c.customer_id IS NULL OR o.customer_id IS NULL;
```

```sql
/*
    ==========================
    == 24 UNION & UNION ALL ==
    ==========================
    - UNION used to combine the rows from both tables
    - used to combines the rows together

    -----------
    -- UNION --
    -----------
    - operator used to combine the results of two or more SELECT statements into a
single result
    - it removes duplicate rows
        The number of columns in all SELECT statements must be the same.
        The data types of the columns must be compatible.
        Column names in the final result are taken from the first SELECT.
    - Syntax
        SELECT country FROM table1
        UNION
        SELECT country FROM table2;

    ---------------
    -- UNION ALL --
    ---------------
    - the UNION ALL operator is used to combine results from multiple SELECT without
removing duplicates
    - Syntax
        SELECT country FROM table1
        UNION ALL
        SELECT country FROM table2;
*/


USE MyDatabase;

CREATE TABLE students_2024 (
    id INT,
    name VARCHAR(50),
    country VARCHAR(50)
);

CREATE TABLE students_2025 (
    id INT,
    name VARCHAR(50),
    country VARCHAR(50)
);

-- Insert data into students_2024
INSERT INTO students_2024 (id, name, country) VALUES
(1, 'Ali', 'Egypt'),
(2, 'Sara', 'USA'),
(3, 'John', 'UK'),
(4, 'Mona', 'Canada');
```

```sql
-- Insert data into students_2025
INSERT INTO students_2025 (id, name, country) VALUES
(3, 'John', 'UK'),
(4, 'Mona', 'Canada'),
(5, 'Youssef', 'Egypt'),
(6, 'Emma', 'France');


-- UNION removes duplicate records automatically
SELECT name, country FROM students_2024
UNION
SELECT name, country FROM students_2025
ORDER BY country ASC;


-- Count Unique Students
SELECT COUNT(*) AS unique_students
FROM (
    SELECT name FROM students_2024
    UNION
    SELECT name FROM students_2025
) AS unique_list;


-- UNION ALL keeps duplicates.
SELECT name, country FROM students_2024
UNION ALL
SELECT name, country FROM students_2025;


-- Count How Many Students in Both Tables
SELECT COUNT(*) AS total_students
FROM (
    SELECT name FROM students_2024
    UNION ALL
    SELECT name FROM students_2025
) AS all_students;
```
