➿ 1. Non-Equijoin

📘 Concept:

- A **non-equijoin** joins two tables without using the `=` operator, often using `>`, `<`, `BETWEEN` etc.
- Useful when ranges are involved (like salary in job grades).

□ Practice Queries:

```
-- 1. Match employees to job grade by salary range
SELECT e.first_name, e.salary, j.grade_level
FROM employees e
JOIN job_grades j ON e.salary BETWEEN j.lowest_sal AND j.highest_sal;

-- 2. Employees with salary > lowest_sal of any grade
SELECT e.first_name, e.salary, j.grade_level
FROM employees e
JOIN job_grades j ON e.salary > j.lowest_sal;

-- 3. Match only employees with salaries in top 3 grade levels
SELECT e.first_name, e.salary, j.grade_level
FROM employees e
JOIN job_grades j ON e.salary BETWEEN j.lowest_sal AND j.highest_sal
WHERE j.grade_level IN ('G2','G1','F3');

-- 4. List employees whose salary is not within any grade range
SELECT first_name, salary
FROM employees e
WHERE NOT EXISTS (
  SELECT 1 FROM job_grades j
  WHERE e.salary BETWEEN j.lowest_sal AND j.highest_sal
);

-- 5. Show grade ranges and employee count for each
SELECT j.grade_level, COUNT(e.employee_id) AS total_employees
FROM job_grades j
LEFT JOIN employees e ON e.salary BETWEEN j.lowest_sal AND j.highest_sal
GROUP BY j.grade_level;
```

🔄 2. Outer Joins

📘 Concept:

- **LEFT OUTER JOIN**: Returns all from left + matched from right.
- **RIGHT OUTER JOIN**: All from right + matched from left.
- **FULL OUTER JOIN**: All from both sides (MySQL doesn't directly support this — use `UNION`).

□ Practice Queries:

```
-- 1. List all customers and their sales, even if no sale
SELECT c.customer_name, s.sale_id
FROM customers c
LEFT JOIN sales s ON c.customer_id = s.customer_id;

-- 2. List all products and if they were sold
SELECT p.product_name, s.sale_id
FROM products p
LEFT JOIN sales s ON p.product_id = s.product_id;

-- 3. Departments with or without employees
SELECT d.department_name, e.first_name
FROM departments d
LEFT JOIN employees e ON d.department_id = e.department_id;

-- 4. Employees and their managers (including those without managers)
SELECT e.first_name AS employee, m.first_name AS manager
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id;

-- 5. Customers with no sales (only those with NULL sale)
SELECT c.customer_name
FROM customers c
LEFT JOIN sales s ON c.customer_id = s.customer_id
WHERE s.sale_id IS NULL;
```

🔁 3. Self Joins

📓 Concept:

- Join a table to itself.
- Common use: hierarchical data (e.g., employees & managers).

□ Practice Queries:

```
-- 1. Employee and their manager
SELECT e.first_name AS Employee, m.first_name AS Manager
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id;

-- 2. Employees who are managers
SELECT DISTINCT m.first_name AS Manager
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id;

-- 3. Employee-manager pairs with same department
SELECT e.first_name AS Employee, m.first_name AS Manager
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id
```

```
WHERE e.department_id = m.department_id;

-- 4. Managers without subordinates
SELECT first_name FROM employees
WHERE employee_id NOT IN (
  SELECT DISTINCT manager_id FROM employees WHERE manager_id IS NOT NULL
);

-- 5. Count of subordinates per manager
SELECT m.first_name AS Manager, COUNT(e.employee_id) AS SubordinateCount
FROM employees m
LEFT JOIN employees e ON m.employee_id = e.manager_id
GROUP BY m.first_name;
```

⚙ 4. Built-in Functions in MySQL

Divided by type:

✨ A. Character Functions

📘 Examples: `UPPER()`, `LOWER()`, `LENGTH()`, `TRIM()`, `CONCAT()`

□ Queries:

```
-- 1. Convert customer names to uppercase
SELECT UPPER(customer_name) FROM customers;

-- 2. Length of each email
SELECT email, LENGTH(email) FROM customers;

-- 3. Trim padded string
SELECT padded_string, TRIM(padded_string) FROM text_samples;

-- 4. Concatenate first and last name
SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM employees;

-- 5. Find customers with 'corp' in name (case-insensitive)
SELECT customer_name FROM customers
WHERE LOWER(customer_name) LIKE '%corp%';
```

🔢 B. Number Functions

📘 Examples: `ROUND()`, `CEIL()`, `FLOOR()`, `MOD()`, `POWER()`

□ Queries:

```
-- 1. Round all product prices
SELECT product_name, ROUND(price) FROM products;

-- 2. Get ceiling of all negative values
SELECT id, CEIL(negative_value) FROM numeric_data;

-- 3. Modulo of value1 and value2
SELECT id, MOD(value1, value2) FROM numeric_data;

-- 4. Power of int_value squared
SELECT id, POWER(int_value, 2) FROM numeric_data;

-- 5. Floor salary of all employees
SELECT first_name, FLOOR(salary) FROM employees;
```

🧮 C. Date Functions

📘 Examples: `CURDATE()`, `DATEDIFF()`, `YEAR()`, `MONTH()`, `NOW()`

🧪 Queries:

```
-- 1. Days since employee hired
SELECT first_name, DATEDIFF(CURDATE(), hire_date) AS days_worked
FROM employees;

-- 2. Sale year and month
SELECT sale_id, YEAR(sale_date) AS sale_year, MONTH(sale_date) AS sale_month
FROM sales;

-- 3. Current date/time
SELECT NOW() AS current_datetime;

-- 4. Employees hired in 2020
SELECT * FROM employees WHERE YEAR(hire_date) = 2020;

-- 5. Products older than 1 year
SELECT product_name, created_date
FROM products
WHERE created_date < (CURDATE() - INTERVAL 1 YEAR);
```

🔄 D. Conversion Functions

📘 Examples: `CAST()`, `CONVERT()`

🧪 Queries:

```
-- 1. Convert string_number to DECIMAL
SELECT id, CAST(string_number AS DECIMAL(10,2)) FROM mixed_formats;
```

```
-- 2. Convert number_as_text to INT
SELECT id, CONVERT(number_as_text, SIGNED) FROM mixed_formats;

-- 3. Convert hire_date to CHAR
SELECT employee_id, CONVERT(hire_date, CHAR) FROM employees;

-- 4. Convert decimal to CHAR
SELECT id, CAST(value1 AS CHAR) FROM numeric_data;

-- 5. Force string to date (where valid)
SELECT id, STR_TO_DATE(date_string, '%Y-%m-%d') FROM mixed_formats;
```

⚙ E. General Functions

📓 Examples: `IFNULL()`, `COALESCE()`, `ISNULL()`, `VERSION()`

□ Queries:

```
-- 1. Replace NULL commission with 0
SELECT first_name, IFNULL(commission_pct, 0) FROM employees;

-- 2. First non-null among 3 columns
SELECT id, COALESCE(string_number, date_string, number_as_text) FROM
mixed_formats;

-- 3. Check for NULL expiry dates
SELECT product_name, ISNULL(expiry_date) FROM products;

-- 4. MySQL version
SELECT VERSION();

-- 5. Handle NULL phone numbers
SELECT customer_name, COALESCE(phone, 'N/A') FROM customers;
```

🧮 F. Elements of Date Format

📓 Concept:

MySQL uses format specifiers like `%d`, `%m`, `%Y`, `%H`, `%i`, `%s` in `DATE_FORMAT()` and `STR_TO_DATE()`.

□ Queries:

```
-- 1. Format hire date
SELECT first_name, DATE_FORMAT(hire_date, '%M %d, %Y') FROM employees;

-- 2. Custom sale date format
SELECT sale_id, DATE_FORMAT(sale_date, '%d/%m/%Y %h:%i %p') FROM sales;
```

```
-- 3. Convert string to date
SELECT id, STR_TO_DATE(date_string, '%d-%m-%Y') FROM mixed_formats
WHERE date_string LIKE '%-%-%';

-- 4. Day of week of event
SELECT event_name, DAYNAME(start_datetime) FROM event_log;

-- 5. Month-Year from created_date
SELECT product_name, DATE_FORMAT(created_date, '%b-%Y') FROM products;
```