EXERCISE 1: String Function

Q1. Concatenate first and last name full_name.

```
   ans: SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM employees;
```

Q2. Convert all employee names to lowercase:

```
   ans: SELECT LOWER(CONCAT(first_name, ' ', last_name)) AS
name_lowercase
FROM employees;
```

Q3. Extract first 3 letters of the employee's first name.

```
   ans: SELECT LEFT(first_name, 3) AS first3_letters
FROM employees;
```

Q4. Replace '@company.com' in email with '@org.com':

```
  ans:SELECT REPLACE(email, '@company.com', '@org.com') AS updated_email
FROM employees;
```

Q5. Trim spaces from a padded string:

```
   ans: SELECT TRIM('   Alice   ') AS trimmed_name;
```

Q6. Count characters in an employee’s full name:

```
  ans: SELECT LENGTH(CONCAT(first_name, ' ', last_name)) AS name_length
FROM employees;
```

Q7. Find position of '@' in email using INSTR()/CHARINDEX():

```
   Ans: SELECT INSTR(email, '@') AS at_position
FROM employees;
```

Q8. Add ‘Mr.’ or ‘Ms.’ before names based on gender (assume gender exists):

```
    Ans: SELECT
    CASE
        WHEN gender = 'M' THEN CONCAT('Mr. ', first_name, ' ', last_name)
        WHEN gender = 'F' THEN CONCAT('Ms. ', first_name, ' ', last_name)
    END AS titled_name
FROM employees;
```

Q9. Format project names to uppercase:

```
   Ans: SELECT UPPER(project_name) AS project_upper
FROM projects;
```

Q10. Remove any dashes from project names:

```
    Ans: SELECT REPLACE(project_name, '-', '') AS project_no_dash
FROM projects;
```

Q11.Create a label like “Emp: John Doe (HR)”:

```
  Ans: SELECT CONCAT('Emp: ', first_name, ' ', last_name, ' (',
department_name, ')') AS emp_label
```

```
FROM employees e
JOIN departments d ON e.department_id = d.department_id;

Q12.  Check email length for each employee:
   Ans: SELECT email, LENGTH(email) AS email_length
FROM employees;

Q13. Extract last name only from email (before @):
    Ans: SELECT SUBSTRING_INDEX(email, '@', 1) AS email_user_part
FROM employees;

Q14.Format: “LASTNAME, Firstname” using UPPER and CONCAT:
    Ans: SELECT CONCAT(UPPER(last_name), ', ', first_name) AS
formatted_name
FROM employees;

Q15: Add “(Active)” next to employee names who have current projects:
   Ans:SELECT
    CONCAT(first_name, ' ', last_name, ' (Active)') AS active_employee
FROM employees e
JOIN employee_projects ep ON e.employee_id = ep.employee_id
JOIN projects p ON ep.project_id = p.project_id
WHERE p.end_date IS NULL OR p.end_date > CURDATE();


EXERCISE 2:  Numeric Function
------------------------------
Q16.  Round salary to the nearest whole number
     Ans: SELECT first_name, last_name, salary, ROUND(salary) AS
salary_rounded
FROM employees;

Q17. Show only even salaries using MOD:
     Ans:SELECT first_name, last_name, salary
FROM employees
WHERE MOD(salary, 2) = 0;

Q18.Show difference between two project end/start dates using DATEDIFF:
  Ans:SELECT project_name,start_date,end_date,DATEDIFF(end_date,
start_date) AS duration_days
FROM projects
WHERE end_date IS NOT NULL;

Q19.Show absolute difference in salaries between two employees:
   Ans:SELECT ABS(
    (SELECT salary FROM employees WHERE employee_id = 101) -
    (SELECT salary FROM employees WHERE employee_id = 102)
) AS salary_difference;

Q20.Raise salary by 10% using POWER:
   Ans: SELECT first_name, last_name, salary,
       salary * POWER(1.10, 1) AS salary_with_increase
FROM employees;
```

Q21.Generate a random number for testing IDs:

```
   Ans: SELECT FLOOR(RAND() * 1000) AS random_id;
```

Q22.Use CEIL and FLOOR on a floating salary:

```
   Ans:SELECT first_name, last_name, salary,
       CEIL(salary) AS salary_ceiling,
       FLOOR(salary) AS salary_floor
FROM employees;
```

Q23.Use LENGTH() on phone numbers (assume column exists):

```
   Ans:SELECT first_name, last_name, phone, LENGTH(phone) AS phone_length
FROM employees;
```

Q24.Categorize salary: High/Medium/Low using CASE:

```
  Ans:SELECT first_name, last_name, salary,
       CASE
           WHEN salary >= 6000 THEN 'High'
           WHEN salary >= 4000 THEN 'Medium'
           ELSE 'Low'
       END AS salary_category
FROM employees;
```

Q25.Count digits in salary amount:

```
   Ans:SELECT first_name, last_name, salary,
       LENGTH(REPLACE(FORMAT(salary, 0), ',', '')) AS salary_digits
FROM employees;
```

EXERCISE 3: Date/Time Function

Q26.Show today’s date using CURRENT_DATE:

```
    Ans:SELECT CURRENT_DATE AS today_date;
```

Q27.Calculate how many days an employee has worked:

```
   Ans:SELECT first_name, last_name,
       DATEDIFF(CURRENT_DATE, hire_date) AS days_worked
FROM employees;
```

Q28.Show employees hired in the current year:

```
  Ans:SELECT first_name, last_name, hire_date
FROM employees
WHERE YEAR(hire_date) = YEAR(CURRENT_DATE);
```

Q29. Display current date and time using NOW():

```
  Ans:SELECT NOW() AS current_datetime;
```

Q30.Extract the year, month, and day from hire_date:

```
   Ans: SELECT first_name, last_name,
       YEAR(hire_date) AS hire_year,
       MONTH(hire_date) AS hire_month,
       DAY(hire_date) AS hire_day
FROM employees;
```

Q31.Show employees hired before 2020:
  Ans:SELECT first_name, last_name, hire_date
FROM employees
WHERE hire_date < '2020-01-01';

Q32.List projects that ended in the last 30 days:
  Ans: SELECT project_name, end_date
FROM projects
WHERE end_date IS NOT NULL
  AND end_date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY);

Q33.Calculate total days between project start and end dates:
  Ans:SELECT project_name,
       start_date,
       end_date,
       DATEDIFF(end_date, start_date) AS total_days
FROM projects
WHERE end_date IS NOT NULL;

Q34.Format date: '2025-07-23' to 'July 23, 2025' (use CONCAT):
   Ans:SELECT CONCAT(MONTHNAME('2025-07-23'), ' ',
              DAY('2025-07-23'), ', ',
              YEAR('2025-07-23')) AS formatted_date;
Q35.Add a CASE: if project still active (end_date IS NULL), show
'Ongoing':
  Ans:SELECT project_name,
       CASE
           WHEN end_date IS NULL THEN 'Ongoing'
           ELSE 'Completed'
       END AS project_status
FROM projects;

EXERCISE 4: Conditional Function

Q36. Use CASE to label salaries:

   Ans: SELECT first_name, last_name, salary,
       CASE
           WHEN salary >= 6000 THEN 'High'
           WHEN salary >= 4000 THEN 'Medium'
           ELSE 'Low'
       END AS salary_label
FROM employees;

Q37.Use COALESCE to show 'No Email' if email is NULL:
  Ans:SELECT first_name, last_name,
       COALESCE(email, 'No Email') AS email_display
FROM employees;

Q38.CASE: If hire_date < 2015, mark as 'Veteran':
  Ans:SELECT first_name, last_name, hire_date,
       CASE

```
            WHEN hire_date < '2015-01-01' THEN 'Veteran'
            ELSE 'New'
        END AS employee_status
FROM employees;
```

Q39.If salary is NULL, default it to 3000 using COALESCE:

```
   Ans: SELECT first_name, last_name,
        COALESCE(salary, 3000) AS adjusted_salary
FROM employees;
```

Q40. CASE: Categorize departments (IT, HR, Other):

```
     Ans: SELECT first_name, last_name, department_id,
        CASE department_id
            WHEN 3 THEN 'IT'
            WHEN 1 THEN 'HR'
            ELSE 'Other'
        END AS dept_category
FROM employees;
```

Q41.CASE: If employee has no project, mark as ‘Unassigned’:

```
     Ans:SELECT e.first_name, e.last_name,
        CASE
            WHEN ep.project_id IS NULL THEN 'Unassigned'
            ELSE 'Assigned'
        END AS project_status
FROM employees e
LEFT JOIN employee_projects ep ON e.employee_id = ep.employee_id;
```

Q42.Show tax band based on salary:

```
    Ans:SELECT first_name, last_name, salary,
        CASE
            WHEN salary >= 6000 THEN '25% Tax'
            WHEN salary >= 4000 THEN '15% Tax'
            ELSE '5% Tax'
        END AS tax_band
FROM employees;
```

Q43.Use nested CASE to label project duration:

```
    Ans: SELECT project_name,
        CASE
            WHEN end_date IS NULL THEN 'Ongoing'
            WHEN DATEDIFF(end_date, start_date) > 365 THEN 'Long-Term'
            ELSE 'Short-Term'
        END AS project_length
FROM projects;
```

Q44.Use CASE with MOD to show even/odd salary IDs:

```
    Ans:SELECT employee_id, salary,
        CASE
            WHEN MOD(employee_id, 2) = 0 THEN 'Even ID'
            ELSE 'Odd ID'
        END AS id_type
FROM employees;
```

Q45.Combine COALESCE + CONCAT for fallback names:

Ans: SELECT COALESCE(CONCAT(first_name, ' ', last_name), 'Unknown Employee') AS full_name
FROM employees;

Q46.CASE with LENGTH(): if name length > 10, label “Long Name”:

Ans: SELECT first_name, last_name,
CASE
WHEN LENGTH(CONCAT(first_name, last_name)) > 10 THEN 'Long Name'
ELSE 'Short Name'
END AS name_size
FROM employees;

Q47. CASE + UPPER(): if email has ‘TEST’, mark as dummy account:

Ans:SELECT email,
CASE
WHEN UPPER(email) LIKE '%TEST%' THEN 'Dummy Account'
ELSE 'Valid Account'
END AS email_status
FROM employees;

Q48.CASE: Show seniority based on hire year (e.g., Junior/Senior):

Ans: SELECT first_name, last_name, hire_date,
CASE
WHEN YEAR(hire_date) <= 2015 THEN 'Senior'
WHEN YEAR(hire_date) <= 2020 THEN 'Mid-Level'
ELSE 'Junior'
END AS seniority
FROM employees;

Q49.Use CASE to determine salary increment range:

Ans: SELECT first_name, last_name, salary,
CASE
WHEN salary >= 6000 THEN 'Increase by 5%'
WHEN salary >= 4000 THEN 'Increase by 10%'
ELSE 'Increase by 15%'
END AS increment_plan
FROM employees;

Q50.Use CASE with CURDATE() to determine anniversary month:

Ans:SELECT first_name, last_name, hire_date,
CASE
WHEN MONTH(hire_date) = MONTH(CURDATE()) THEN 'Anniversary Month'
ELSE 'Not Anniversary Month'
END AS anniversary_status
FROM employees;