NAMES: MUSONI NSHUTI SAM

ID: 28857

COURSE: DATABASE MANAGEMENT SYSTEMS

String Function Exercises (15)

1. Concatenate first and last name as full_name

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name

FROM EMPLOYEE;
```

2. Convert all employee names to lowercase

```
SELECT LOWER(first_name) AS first_name_lower,

            LOWER(last_name) AS last_name_lower

FROM EMPLOYEE;
```

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

```
SELECT LEFT(first_name, 3) AS first_3_letters

FROM EMPLOYEE;
```

4. Replace '@company.com' in email with '@org.com'

```
SELECT REPLACE(email, '@company.com', '@org.com') AS updated_email

FROM EMPLOYEE;
```

5. Trim spaces from a padded string

```
-- Assume column `padded_string` exists or use any column

SELECT TRIM(padded_string) AS trimmed_string

FROM some_table;
```

6. Count characters in an employee’s full name

```
SELECT LENGTH(CONCAT(first_name, ' ', last_name)) AS name_length

FROM EMPLOYEE;
```

7. Find position of '@' in email using INSTR or CHARINDEX

```
-- INSTR (My)

SELECT INSTR(email, '@') AS at_position

FROM EMPLOYEE;

-- Or, CHARINDEX ( Server)

-- SELECT CHARINDEX('@', email) AS at_position FROM EMPLOYEE;
```

8. Add ‘Mr.’ or ‘Ms.’ before names based on gender

```
SELECT

   CASE

      WHEN gender = 'M' THEN CONCAT('Mr. ', first_name, ' ', last_name)

      WHEN gender = 'F' THEN CONCAT('Ms. ', first_name, ' ', last_name)

      ELSE CONCAT(first_name, ' ', last_name)

   END AS titled_name

FROM EMPLOYEE;
```

9. Format project names to uppercase

```
SELECT UPPER(project_name) AS project_upper

FROM PROJECTS;
```

10. Remove any dashes from project names

```
SELECT REPLACE(project_name, '-', '') AS project_cleaned

FROM PROJECTS;
```

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

```
SELECT
```

```
    CONCAT('Emp: ', first_name, ' ', last_name, ' (', d.department_name, ')') AS label
FROM EMPLOYEE e
JOIN DEPARTMENTS d ON e.department_id = d.department_id;
```

12. Check email length for each employee

```
SELECT LENGTH(email) AS email_length
FROM EMPLOYEE;
```

13. Extract last name only from email (before @)

```
SELECT SUBSTRING_INDEX(email, '@', 1) AS email_username
FROM EMPLOYEE;
```

14. Format: "LASTNAME, Firstname" using UPPER and CONCAT

```
SELECT CONCAT(UPPER(last_name), ', ', first_name) AS formatted_name
FROM EMPLOYEE;
```

15. Add "(Active)" next to employee names who have current projects

```
SELECT
    CONCAT(first_name, ' ', last_name,
                CASE
                    WHEN EXISTS (
                        SELECT 1 FROM EMPLOYEE_PROJECTS ep
                        JOIN PROJECTS p ON ep.project_id = p.project_id
                        WHERE ep.employee_id = e.employee_id AND (p.end_date IS NULL OR
p.end_date >= CURRENT_DATE)
                    )
                    THEN ' (Active)'
                    ELSE ''
```

```
            END
    ) AS name_status
FROM EMPLOYEE e;
```

Numeric Function Exercises (10)

16. Round salary to the nearest whole number

```
SELECT employee_id, ROUND(salary) AS rounded_salary
FROM EMPLOYEE;
```

17. Show only even salaries using MOD

```
SELECT employee_id, salary
FROM EMPLOYEE
WHERE MOD(ROUND(salary), 2) = 0;
```

18. Show difference between two project end/start dates using DATEDIFF

```
SELECT project_id, DATEDIFF(end_date, start_date) AS project_duration_days
FROM PROJECTS;
```

19. Show absolute difference in salaries between two employees

```
-- Example: compare employee 101 and 102
SELECT ABS(e1.salary - e2.salary) AS salary_difference
FROM EMPLOYEE e1
JOIN EMPLOYEE e2 ON e1.employee_id = 101 AND e2.employee_id = 102;
```

20. Raise salary by 10% using POWER (not typical, but interpreted as exponential growth)

```
-- Power used as: salary * POWER(1.10, 1) = salary increased by 10%
SELECT employee_id, salary,
       salary * POWER(1.10, 1) AS increased_salary
FROM EMPLOYEE;
```

21. Generate a random number for testing IDs

```
SELECT employee_id, ROUND(RAND() * 10000) AS test_id
FROM EMPLOYEE;
```

22. Use CEIL and FLOOR on a floating salary

```
SELECT employee_id, salary,
       CEIL(salary) AS salary_ceil,
       FLOOR(salary) AS salary_floor
FROM EMPLOYEE;
```

23. Use LENGTH() on phone numbers (assume column exists)

```
-- Assuming a column: phone_number VARCHAR(20)
SELECT employee_id, phone_number,
       LENGTH(phone_number) AS phone_length
FROM EMPLOYEE;
```

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

```
SELECT employee_id, salary,
       CASE
           WHEN salary >= 10000 THEN 'High'
           WHEN salary >= 5000 THEN 'Medium'
           ELSE 'Low'
       END AS salary_category
FROM EMPLOYEE;
```

25. Count digits in salary amount

```
-- Remove decimal and count only digits

SELECT employee_id, salary,

        LENGTH(REPLACE(CAST(salary AS CHAR), '.', '')) AS digit_count

FROM EMPLOYEE;
```

<u>Date/Time Function Exercises (10)</u>

26. Show today's date using CURRENT_DATE

```
SELECT CURRENT_DATE AS today;
```

27. Calculate how many days an employee has worked

```
SELECT employee_id, first_name, last_name,

        DATEDIFF(CURRENT_DATE, hire_date) AS days_worked

FROM EMPLOYEE;
```

28. Show employees hired in the current year

```
SELECT employee_id, first_name, last_name, hire_date

FROM EMPLOYEE

WHERE YEAR(hire_date) = YEAR(CURRENT_DATE);
```

29. Display current date and time using NOW()

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

30. Extract the year, month, and day from hire_date

```
SELECT employee_id,

        YEAR(hire_date) AS hire_year,

        MONTH(hire_date) AS hire_month,

        DAY(hire_date) AS hire_day
```

```
FROM EMPLOYEE;
```

31. Show employees hired before 2020

```
SELECT employee_id, first_name, last_name, hire_date

FROM EMPLOYEE

WHERE hire_date < '2020-01-01';
```

32. List projects that ended in the last 30 days

```
SELECT project_id, project_name, end_date

FROM PROJECTS

WHERE end_date IS NOT NULL

   AND end_date >= CURRENT_DATE - INTERVAL 30 DAY;
```

33. Calculate total days between project start and end dates

```
SELECT project_id, project_name,

          DATEDIFF(end_date, start_date) AS total_days

FROM PROJECTS

WHERE end_date IS NOT NULL;
```

34. Format date ‘2025-07-23’ to ‘July 23, 2025’ using CONCAT

```
SELECT CONCAT(MONTHNAME('2025-07-23'), ' ', DAY('2025-07-23'), ', ', YEAR('2025-07-23')) AS
formatted_date;
```

35. Add a CASE: if project still active (end_date IS NULL), show ‘Ongoing’

```
SELECT project_id, project_name,

          CASE

             WHEN end_date IS NULL THEN 'Ongoing'

             ELSE 'Completed'

          END AS status
```

```
FROM PROJECTS;
```

<u>Conditional Function Exercises (15)</u>

36. Use CASE to label salaries

```
SELECT employee_id, salary,
       CASE
           WHEN salary >= 10000 THEN 'High'
           WHEN salary >= 5000 THEN 'Medium'
           ELSE 'Low'
       END AS salary_label
FROM EMPLOYEE;
```

37. Use COALESCE to show ‘No Email’ if email is NULL

```
SELECT employee_id, COALESCE(email, 'No Email') AS email_display
FROM EMPLOYEE;
```

38. CASE: If hire_date < 2015, mark as ‘Veteran’

```
SELECT employee_id, first_name, hire_date,
       CASE
           WHEN hire_date < '2015-01-01' THEN 'Veteran'
           ELSE 'New Hire'
       END AS status
FROM EMPLOYEE;
```

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

```
SELECT employee_id, COALESCE(salary, 3000) AS salary_with_default
```

```
FROM EMPLOYEE;
```

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

```
SELECT d.department_id, d.department_name,
            CASE
               WHEN d.department_name = 'IT' THEN 'IT'
               WHEN d.department_name = 'HR' THEN 'HR'
               ELSE 'Other'
            END AS department_category
FROM DEPARTMENTS d;
```

41. CASE: If employee has no project, mark as 'Unassigned'

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

42. CASE: Show tax band based on salary

```
SELECT employee_id, salary,
            CASE
               WHEN salary >= 12000 THEN 'Band A'
               WHEN salary >= 8000 THEN 'Band B'
               WHEN salary >= 5000 THEN 'Band C'
               ELSE 'Band D'
            END AS tax_band
FROM EMPLOYEE;
```

43. Use nested CASE to label project duration

```
SELECT project_id, DATEDIFF(end_date, start_date) AS duration,
            CASE
               WHEN end_date IS NULL THEN 'Ongoing'
               WHEN DATEDIFF(end_date, start_date) > 365 THEN 'Long-Term'
               WHEN DATEDIFF(end_date, start_date) > 180 THEN 'Mid-Term'
               ELSE 'Short-Term'
            END AS duration_label
FROM PROJECTS;
```

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

```
SELECT employee_id, salary,
            CASE
               WHEN MOD(employee_id, 2) = 0 THEN 'Even ID'
               ELSE 'Odd ID'
            END AS id_parity
FROM EMPLOYEE;
```

45. Combine COALESCE + CONCAT for fallback names

```
SELECT employee_id,
            CONCAT(COALESCE(first_name, 'Unknown'), ' ', COALESCE(last_name, 'Name')) AS
full_name
FROM EMPLOYEE;
```

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

```
SELECT employee_id, first_name,
            CASE
               WHEN LENGTH(first_name) > 10 THEN 'Long Name'
               ELSE 'Short Name'
            END AS name_length_category
FROM EMPLOYEE;
```

47. CASE + UPPER(): if email has 'TEST', mark as dummy account

```
SELECT employee_id, email,
            CASE
               WHEN UPPER(email) LIKE '%TEST%' THEN 'Dummy Account'
               ELSE 'Valid Account'
            END AS account_type
FROM EMPLOYEE;
```

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

```
SELECT employee_id, hire_date,
            CASE
               WHEN YEAR(hire_date) <= YEAR(CURRENT_DATE) - 10 THEN 'Senior'
               WHEN YEAR(hire_date) <= YEAR(CURRENT_DATE) - 5 THEN 'Mid-Level'
               ELSE 'Junior'
            END AS seniority_level
FROM EMPLOYEE;
```

49. Use CASE to determine salary increment range

```
SELECT employee_id, salary,
            CASE
```

```
        WHEN salary < 3000 THEN 'Increment: 20%'
        WHEN salary < 7000 THEN 'Increment: 15%'
        WHEN salary < 10000 THEN 'Increment: 10%'
        ELSE 'Increment: 5%'
    END AS increment_range
FROM EMPLOYEE;
```

50. Use CASE with CURDATE() to determine anniversary month

```
SELECT employee_id, hire_date,
    CASE
        WHEN MONTH(hire_date) = MONTH(CURDATE()) THEN 'Anniversary Month'
        ELSE 'Not This Month'
    END AS anniversary_status
FROM EMPLOYEE;
```