assignment

1. **Concatenate first and last name as full_name**

```
sql

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

2. **Convert all employee names to lowercase**

```
sql

SELECT LOWER(first_name) AS first_name_lower, LOWER(last_name) AS
last_name_lower FROM employees;
```

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

```
sql

SELECT SUBSTRING(first_name, 1, 3) AS first_3 FROM employees;
```

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

```
sql

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

5. **Trim spaces from a padded string**

```
sql

SELECT TRIM('   Hello SQL   ') AS trimmed_string;
```

6. **Count characters in an employee's full name**

```
sql

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

7. **Find position of '@' in email**

```
sql

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

8. **Add 'Mr.' or 'Ms.' before names based on gender**

Assume a gender column exists (you may need to add it)

```
sql

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 employees;
```

9. **Format project names to uppercase**

```
sql

SELECT UPPER(project_name) AS upper_project_name FROM projects;
```

10. **Remove any dashes from project names**

```
sql

SELECT REPLACE(project_name, '-', '') AS clean_name FROM projects;
```

11. **Create a label like "Emp: John Doe (HR)"**

```
sql

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

12. **Check email length**

```
sql

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

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

```
sql

SELECT SUBSTRING(email, 1, INSTR(email, '@') - 1) AS email_prefix FROM
employees;
```

14. **Format: "LASTNAME, Firstname"**

```
sql

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

15. **Add "(Active)" next to employee names with current projects**

```
sql

SELECT
  CONCAT(first_name, ' ', last_name,
         CASE
           WHEN p.end_date IS NULL OR p.end_date > CURRENT_DATE THEN '
(Active)'
```

```
           ELSE ''
         END) AS employee_status
FROM employees e
JOIN employee_projects ep ON e.employee_id = ep.employee_id
JOIN projects p ON ep.project_id = p.project_id;
```

Numeric Function Exercises

16. **Round salary to nearest whole**

```
sql

SELECT ROUND(salary) AS rounded_salary FROM employees;
```

17. **Show only even salaries**

```
sql

SELECT * FROM employees WHERE MOD(salary, 2) = 0;
```

18. **Difference between project end and start dates**

```
sql
SELECT project_name, DATEDIFF(end_date, start_date) AS duration_days FROM
projects;
```

19. **Absolute difference in salaries between 2 employees**

```
sql

SELECT ABS(e1.salary - e2.salary) AS salary_difference
FROM employees e1, employees e2
WHERE e1.employee_id = 101 AND e2.employee_id = 102;
```

20. **Raise salary by 10% using POWER**

```
sql

SELECT salary, salary * POWER(1.10, 1) AS increased_salary FROM employees;
```

21. **Generate a random number**

```
sql

SELECT RAND() AS random_number;
```

22. **Use CEIL and FLOOR on a floating salary**

```
sql

SELECT salary, CEIL(salary), FLOOR(salary) FROM employees;
```

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

```
sql

SELECT LENGTH(phone_number) AS phone_length FROM employees;  -- Assumes
'phone_number' exists
```

24. **Categorize salary**

```
sql

SELECT
  salary,
  CASE
    WHEN salary >= 5000 THEN 'High'
    WHEN salary >= 3000 THEN 'Medium'
    ELSE 'Low'
  END AS salary_category
FROM employees;
```

25. **Count digits in salary amount**

```
sql

SELECT salary, LENGTH(REPLACE(salary, '.', '')) AS digit_count FROM
employees;
```

Date/Time Function Exercises

26. **Today's date**

```
sql

SELECT CURRENT_DATE;
```

27. **Days employee has worked**

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

28. **Employees hired this year**

```
sql

SELECT * FROM employees WHERE YEAR(hire_date) = YEAR(CURRENT_DATE);
```

29. **Current date and time**

```
sql

SELECT NOW();
```

30. **Extract year, month, and day**

```
sql
```

```
SELECT
  hire_date,
  YEAR(hire_date) AS year,
  MONTH(hire_date) AS month,
  DAY(hire_date) AS day
FROM employees;
```

31. **Employees hired before 2020**

```
sql

SELECT * FROM employees WHERE hire_date < '2020-01-01';
```

32. **Projects that ended in last 30 days**

```
sql

SELECT * FROM projects
WHERE end_date IS NOT NULL
  AND end_date BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY) AND
CURRENT_DATE;
```

33. **Total days between project start and end**

```
sql

SELECT 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'**

```
sql

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

35. **CASE: if project still active**

```
sql

SELECT
  project_name,
  CASE
    WHEN end_date IS NULL THEN 'Ongoing'
    ELSE 'Completed'
  END AS project_status
FROM projects;
```

Conditional Function Exercises

36. **CASE to label salaries**

Already answered in #24.

37. **COALESCE for NULL email**

sql

```
SELECT COALESCE(email, 'No Email') AS safe_email FROM employees;
```

38. **Veteran if hired before 2015**

sql

```
SELECT first_name, last_name,
  CASE
    WHEN hire_date < '2015-01-01' THEN 'Veteran'
    ELSE 'New'
  END AS experience_label
FROM employees;
```

39. **Default NULL salary to 3000**

sql

```
SELECT COALESCE(salary, 3000) AS adjusted_salary FROM employees;
```

40. **Categorize departments**

sql

```
SELECT department_name,
  CASE
    WHEN department_name = 'Information Technology' THEN 'IT'
    WHEN department_name = 'Human Resources' THEN 'HR'
    ELSE 'Other'
  END AS dept_category
FROM departments;
```

41. **Mark employees with no project**

sql
```
SELECT e.employee_id, first_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;
```

42. **Tax band based on salary**

sql

```
SELECT salary,
  CASE
    WHEN salary >= 6000 THEN 'Band A'
    WHEN salary >= 4000 THEN 'Band B'
    ELSE 'Band C'
  END AS tax_band
FROM employees;
```

43. **Label project duration**

```
sql

SELECT project_name,
  CASE
    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
WHERE end_date IS NOT NULL;
```

44. **Even/odd employee_id using CASE + MOD**

```
sql

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

45. **COALESCE + CONCAT fallback name**

```
sql

SELECT CONCAT(COALESCE(first_name, 'Unknown'), ' ', COALESCE(last_name,
'Employee')) AS name FROM employees;
```

46. **Label long name using LENGTH**

```
sq
SELECT first_name,
  CASE
    WHEN LENGTH(first_name) > 10 THEN 'Long Name'
    ELSE 'Short Name'
  END AS name_type
FROM employees;
```

47. **Dummy account if email has 'TEST'**

```
sql

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

48. **Seniority based on hire year**

```
sql

SELECT first_name,
  CASE
    WHEN YEAR(hire_date) <= 2015 THEN 'Senior'
    ELSE 'Junior'
  END AS seniority
FROM employees;
```

49. **Determine salary increment range**

```
sql
SELECT salary,
  CASE
    WHEN salary < 4000 THEN '10% Increment'
    WHEN salary < 6000 THEN '7% Increment'
    ELSE '5% Increment'
  END AS increment_range
FROM employees;
```

50. **Anniversary month check**

```
sql

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