Names: Kunda Tabitha
ID: 27684
Database Management System

1. Concatenate first and last name as full_name
```
SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM employees;
```

2. Convert all employee names to lowercase
```
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
```
SELECT SUBSTR(first_name, 1, 3) AS first_three FROM employees;
```

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

5. Trim spaces from a padded string
```
SELECT TRIM('   Tax Department   ') AS trimmed_text FROM dual;
```

6. Count characters in an employee’s full name
```
SELECT first_name, last_name, LENGTH(CONCAT(first_name, last_name)) AS full_name_length FROM employees;
```

7. Find position of '@' in email
```
SELECT email, INSTR(email, '@') AS at_position FROM employees;
```

8. Add ‘Mr.’ or ‘Ms.’ before names based on assumed gender column
```
SELECT
  CASE WHEN gender = 'M' THEN CONCAT('Mr. ', first_name)
       WHEN gender = 'F' THEN CONCAT('Ms. ', first_name)
  END AS titled_name
FROM employees;
```

9. Format project names to uppercase
```
SELECT UPPER(project_name) AS upper_name FROM projects;
```

10. Remove any dashes from project names
```
SELECT REPLACE(project_name, '-', '') AS clean_name FROM projects;
```

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

12. Check email length for each employee
```
SELECT email, LENGTH(email) AS email_length FROM employees;
```

13. Extract last name from email (before @)
```
SELECT SUBSTR(email, 1, INSTR(email, '@') - 1) AS name_from_email FROM employees;
```

14. Format as “LASTNAME, Firstname”
```
SELECT CONCAT(UPPER(last_name), ', ', INITCAP(first_name)) AS formatted_name FROM employees;
```

15. Add “(Active)” next to employee names who have current projects
```
SELECT first_name,  CASE WHEN ep.project_id IS NOT NULL THEN CONCAT(first_name, ' (Active)')
```

```
        ELSE first_name
    END AS active_status
FROM employees e
LEFT JOIN employee_projects ep ON e.employee_id = ep.employee_id;
```

16. Round salary to nearest whole number

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

17. Show only even salaries

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

18. Difference between project end and start dates

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

19. Absolute difference in salaries between two employees

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

20. Raise salary by 10% using POWER

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

21. Generate a random number for testing IDs

```
SELECT ROUND(DBMS_RANDOM.VALUE(1, 1000)) AS random_id FROM dual;
```

22. Use CEIL and FLOOR on a floating salary

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

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

```
SELECT phone_number, LENGTH(phone_number) AS phone_length FROM employees;
```

24. Categorize salary: High/Medium/Low

```
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

```
SELECT salary, LENGTH(TRUNC(salary)) AS digits_in_salary FROM employees;
```

26. Show today’s date

```
SELECT CURRENT_DATE AS today FROM dual;
```

27. Calculate how many days an employee has worked

```
SELECT employee_id, (CURRENT_DATE - hire_date) AS days_worked FROM employees;
```

28. Show employees hired in the current year

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

29. Display current date and time

```
SELECT CURRENT_TIMESTAMP FROM dual;
```

30. Extract year, month, and day from hire_date

```
SELECT employee_id,
      EXTRACT(YEAR FROM hire_date) AS hire_year,
      EXTRACT(MONTH FROM hire_date) AS hire_month,
      EXTRACT(DAY FROM hire_date) AS hire_day
FROM employees;
```

31. Show employees hired before 2020

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

32. List projects that ended in the last 30 days

```
SELECT * FROM projects
WHERE end_date >= (CURRENT_DATE - 30);
```

33. Calculate total days between project start and end

```
SELECT project_name, (end_date - start_date) AS total_days FROM projects;
```

34. Format date to “July 23, 2025”

```
SELECT CONCAT(TO_CHAR(DATE '2025-07-23', 'Month DD, YYYY')) AS formatted_date FROM dual;
```

35. Add CASE: if project still active, show “Ongoing”

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

36. Use CASE to label salaries

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

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

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

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

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

39. Default salary to 3000 if NULL

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

40. Categorize departments

```
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 projects as “Unassigned”

```
SELECT e.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. Show tax band based on salary

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

43. Use nested CASE to label project duration

```
SELECT project_name,
    CASE
      WHEN (end_date - start_date) > 365 THEN 'Long-term'
      WHEN (end_date - start_date) BETWEEN 180 AND 365 THEN 'Mid-term'
      ELSE 'Short-term'
    END AS duration_type
FROM projects
WHERE end_date IS NOT NULL;
```

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

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

45. Combine COALESCE + CONCAT for fallback names

```
SELECT COALESCE(CONCAT(first_name, ' ', last_name), 'No Name') AS full_name
FROM employees;
```

46. CASE with LENGTH(): label “Long Name”

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

47. CASE + UPPER(): mark dummy accounts

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

48. Show seniority based on hire year

```
SELECT first_name,
    CASE
      WHEN EXTRACT(YEAR FROM hire_date) < 2015 THEN 'Senior'
```

```
        ELSE 'Junior'
      END AS seniority
FROM employees;
```

 49. Determine salary increment range

```
SELECT salary,
      CASE
        WHEN salary < 3000 THEN 'Increase by 20%'
        WHEN salary BETWEEN 3000 AND 5000 THEN 'Increase by 10%'
        ELSE 'Increase by 5%'
      END AS increment_plan
FROM employees;
```

 50. Determine anniversary month

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