1. Problem: −Create a list of all tables whose first two characters in the name of the table is JO −The tables must be owned by the current Oracle User.

Answer:

-- Create sample tables with names starting with "JO"

CREATE TABLE JOB\_DETAILS (

job\_id INT PRIMARY KEY,

job\_title VARCHAR(255) NOT NULL

);

CREATE TABLE JOURNAL\_ENTRIES (

entry\_id INT PRIMARY KEY,

entry\_text VARCHAR(255) NOT NULL

);

-- Create some other tables with different names

CREATE TABLE EMPLOYEES (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL

);

CREATE TABLE DEPARTMENTS (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Insert sample data into JOB\_DETAILS table

INSERT INTO JOB\_DETAILS (job\_id, job\_title)

VALUES

(1, 'Software Developer'),

(2, 'Data Analyst');

-- Insert sample data into JOURNAL\_ENTRIES table

INSERT INTO JOURNAL\_ENTRIES (entry\_id, entry\_text)

VALUES

(1, 'Entry 1 Text'),

(2, 'Entry 2 Text');

-- Insert sample data into EMPLOYEES table

INSERT INTO EMPLOYEES (employee\_id, first\_name, last\_name)

VALUES

(1, 'John', 'Doe'),

(2, 'Jane', 'Smith');

-- Insert sample data into DEPARTMENTS table

INSERT INTO DEPARTMENTS (department\_id, department\_name)

VALUES

(1, 'Sales'),

(2, 'Engineering');

SELECT table\_name

FROM all\_tables

WHERE table\_name LIKE 'JO%'

AND owner = USER;

1. • Problem: − Create a list that includes the first initial of every employee's first name, a space, and the last name of the employee.

Answer:

-- Create employees table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL

);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, first\_name, last\_name)

VALUES

(1, 'John', 'Doe'),

(2, 'Jane', 'Smith'),

(3, 'Emily', 'Johnson'),

(4, 'Michael', 'Brown'),

(5, 'Sarah', 'Davis');

SELECT

SUBSTRING(first\_name, 1, 1) || ' ' || last\_name AS employee\_name

FROM

employees;

1. Problem: − Create a list of every employee's first name concatenated to a space and the employee's last name, and the email of all employees where the email address contains the string 'IN.

Answer: -- Create employees table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL

);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, first\_name, last\_name, email)

VALUES

(1, 'John', 'Doe', 'john.doe@example.com'),

(2, 'Jane', 'Smith', 'jane.smith@inmail.com'),

(3, 'Emily', 'Johnson', 'emily.johnson@example.com'),

(4, 'Michael', 'Brown', 'michael.brown@internal.com'),

(5, 'Sarah', 'Davis', 'sarah.davis@intranet.com');

SELECT

first\_name || ' ' || last\_name AS full\_name,

email

FROM

employees

WHERE

email LIKE '%IN%';

1. Problem: − Create a list of 'smallest' last name and the 'highest' last name from the employees table.

Answer:

-- Create employees table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL

);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, first\_name, last\_name, email)

VALUES

(1, 'John', 'Doe', 'john.doe@example.com'),

(2, 'Jane', 'Smith', 'jane.smith@inmail.com'),

(3, 'Emily', 'Johnson', 'emily.johnson@example.com'),

(4, 'Michael', 'Brown', 'michael.brown@internal.com'),

(5, 'Sarah', 'Davis', 'sarah.davis@intranet.com');

SELECT

MIN(last\_name) AS smallest\_last\_name,

MAX(last\_name) AS highest\_last\_name

FROM

employees;

1. Problem: − Create a list of weekly salaries from the employees table where the weekly salary is between 700 and 3000 − The salaries should be formatted to include a $- sign and have two decimal points like: $9999.99.

Answer:

-- Create employees table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

weekly\_salary DECIMAL(10, 2) NOT NULL

); -- Insert sample data into employees table

INSERT INTO employees (employee\_id, first\_name, last\_name, email, weekly\_salary)

VALUES

(1, 'John', 'Doe', 'john.doe@example.com', 800.50),

(2, 'Jane', 'Smith', 'jane.smith@inmail.com', 2500.75),

(3, 'Emily', 'Johnson', 'emily.johnson@example.com', 3200.00),

(4, 'Michael', 'Brown', 'michael.brown@internal.com', 1500.00),

(5, 'Sarah', 'Davis', 'sarah.davis@intranet.com', 2900.25);

SELECT

'$' || TO\_CHAR(weekly\_salary, '9999.99') AS formatted\_salary

FROM

employees

WHERE

weekly\_salary BETWEEN 700 AND 3000;

1. Problem: − Create a list of every employee and his related job title sorted by job\_title.

Answer:

-- Create jobs table

CREATE TABLE jobs (

job\_id INT PRIMARY KEY,

job\_title VARCHAR(255) NOT NULL

);

-- Insert sample data into jobs table

INSERT INTO jobs (job\_id, job\_title)

VALUES

(1, 'Software Developer'),

(2, 'Data Analyst'),

(3, 'Project Manager'),

(4, 'HR Specialist'),

(5, 'Marketing Coordinator');

SELECT

e.first\_name || ' ' || e.last\_name AS full\_name,

j.job\_title

FROM

employees e

JOIN

jobs j ON e.job\_id = j.job\_id

ORDER BY

j.job\_title;

7. • Problem: −Create a list of every employee’s job, the salary ranges within the job, and the employee's salary −List the lowest and highest salary range within each job with a dash to separate the salaries like this: 100 – 200.

Answer:

-- Create jobs table

CREATE TABLE jobs (

job\_id INT PRIMARY KEY,

job\_title VARCHAR(255) NOT NULL

);

-- Insert sample data into jobs table

INSERT INTO jobs (job\_id, job\_title)

VALUES

(1, 'Software Developer'),

(2, 'Data Analyst'),

(3, 'Project Manager'),

(4, 'HR Specialist'),

(5, 'Marketing Coordinator');

SELECT

j.job\_title,

MIN(e.salary) || ' - ' || MAX(e.salary) AS salary\_range,

e.salary

FROM

employees e

JOIN

jobs j ON e.job\_id = j.job\_id

GROUP BY

j.job\_title, e.salary

ORDER BY

j.job\_title;

8. Problem: − Using an ANSII join method, create a list of every employee's first initial and last name, and department name − Make sure the tables are joined on all of the foreign keys declared between the two tables.

Answer:

-- Create departments table

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

); -- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

SELECT

SUBSTR(e.first\_name, 1, 1) || '. ' || e.last\_name AS employee\_name,

d.department\_name

FROM

employees e

JOIN

departments d ON e.department\_id = d.department\_id;

9. Problem: − Change the previous listing to join only on the department\_id column.

Answer:

-- Create departments table

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

SELECT

SUBSTR(e.first\_name, 1, 1) || '. ' || e.last\_name AS employee\_name,

d.department\_name

FROM

employees e

JOIN

departments d ON e.department\_id = d.department\_id;

10. Problem: − Create a list of every employee's last name, and the word nobody or somebody depending on whether or not the employee has a manager − Use the Oracle DECODE function to create the list.

Answer:

-- Create employees table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

department\_id INT,

manager\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id),

FOREIGN KEY (manager\_id) REFERENCES employees(employee\_id)

);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, first\_name, last\_name, email, department\_id, manager\_id)

VALUES

(1, 'John', 'Doe', 'john.doe@example.com', 1, NULL),

(2, 'Jane', 'Smith', 'jane.smith@inmail.com', 2, 1),

(3, 'Emily', 'Johnson', 'emily.johnson@example.com', 3, 1),

(4, 'Michael', 'Brown', 'michael.brown@internal.com', 4, 2),

(5, 'Sarah', 'Davis', 'sarah.davis@intranet.com', 5, 3);

SELECT

last\_name,

DECODE(manager\_id, NULL, 'nobody', 'somebody') AS manager\_status

FROM

employees;

11. Problem: − Create a list of every employee's first initial and last name, salary, and a yes or no to show whether or not an employee makes a commission − Fix this query to produce the result.

Answer:

-- Create employees table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL,

commission DECIMAL(10, 2), -- This column can be NULL if no commission

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, first\_name, last\_name, salary, commission, department\_id)

VALUES

(1, 'John', 'Doe', 5000.00, 500.00, 1),

(2, 'Jane', 'Smith', 6000.00, NULL, 2),

(3, 'Emily', 'Johnson', 5500.00, 700.00, 3),

(4, 'Michael', 'Brown', 4500.00, NULL, 4),

(5, 'Sarah', 'Davis', 7000.00, 1000.00, 5);

SELECT

SUBSTR(first\_name, 1, 1) || '. ' || last\_name AS employee\_name,

salary,

DECODE(commission, NULL, 'no', 'yes') AS makes\_commission

FROM

employees;

12. Problem: − Create a list of every employee's last name, department name, city, and state\_province − Include departments without employees − An outer join is required.

Answer:

-- Create departments table

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL,

city VARCHAR(255) NOT NULL,

state\_province VARCHAR(255) NOT NULL

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name, city, state\_province)

VALUES

(1, 'Engineering', 'New York', 'NY'),

(2, 'Human Resources', 'Los Angeles', 'CA'),

(3, 'Marketing', 'Chicago', 'IL'),

(4, 'Sales', 'Houston', 'TX'),

(5, 'Finance', 'Miami', 'FL');

SELECT

e.last\_name,

d.department\_name,

d.city,

d.state\_province

FROM

departments d

LEFT OUTER JOIN

employees e ON d.department\_id = e.department\_id

ORDER BY

d.department\_name, e.last\_name;

13. Problem: −Create a list of every employee's first and last names, and the first occurrence of: commission\_pct, manager\_id, or -1 −If an employee gets commission, display the commission\_pct column; if no commission, then display his manager\_id; if he has neither commission nor manager, then the number -1.

Answer:

-- Create employees table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL,

commission\_pct DECIMAL(5, 2), -- Can be NULL if no commission

manager\_id INT,

FOREIGN KEY (manager\_id) REFERENCES employees(employee\_id)

); -- Insert sample data into employees table

INSERT INTO employees (employee\_id, first\_name, last\_name, salary, commission\_pct, manager\_id)

VALUES

(1, 'John', 'Doe', 5000.00, 0.10, 2),

(2, 'Jane', 'Smith', 6000.00, NULL, NULL),

(3, 'Emily', 'Johnson', 5500.00, NULL, 4),

(4, 'Michael', 'Brown', 4500.00, NULL, NULL),

(5, 'Sarah', 'Davis', 7000.00, 0.05, NULL);

SELECT

first\_name,

last\_name,

COALESCE(commission\_pct, manager\_id, -1) AS first\_occurrence

FROM

employees;

14. Problem: − Create a list of every employee's last name, salary, and job\_grade for all employees working in departments with a department\_id greater than 50.

Answer:

-- Create departments table

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(51, 'Engineering'),

(52, 'Human Resources'),

(53, 'Marketing'),

(54, 'Sales'),

(55, 'Finance');

SELECT

e.last\_name,

e.salary,

e.job\_grade

FROM

employees e

JOIN

departments d ON e.department\_id = d.department\_id

WHERE

d.department\_id > 50;

15. Problem: − Produce a list of every employee's last name and department name − Include both employees without departments, and departments without employees.

Answer:

-- Create departments table

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');\_name VARCHAR(255) NOT NULL

);

SELECT

e.last\_name,

d.department\_name

FROM

employees e

FULL OUTER JOIN

departments d ON e.department\_id = d.department\_id

ORDER BY

e.last\_name, d.department\_name;

16. •Problem: −Create a treewalkinglist of every employee's last name, his manager’s last name, and his position in the company −The top level manager has position 1, this manager's subordinates position 2, their subordinates position 3, and so on −Start the listing with employee number 100.

Answer:

-- Create employees table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

manager\_id INT, -- NULL for top-level managers

department\_id INT,

FOREIGN KEY (manager\_id) REFERENCES employees(employee\_id)

);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, manager\_id, department\_id)

VALUES

(100, 'King', NULL, 1), -- Top-level manager

(101, 'Smith', 100, 2), -- Reports to King

(102, 'Johnson', 100, 3), -- Reports to King

(103, 'Williams', 101, 2), -- Reports to Smith

(104, 'Jones', 101, 2), -- Reports to Smith

(105, 'Brown', 102, 3); -- Reports to Johnson

-- Recursive CTE to generate the hierarchy

WITH RECURSIVE EmployeeHierarchy AS (

-- Anchor member: Start from employee number 100

SELECT

e.employee\_id,

e.last\_name AS employee\_last\_name,

NULL AS manager\_last\_name, -- Top-level manager has no manager

1 AS position -- Top-level position

FROM

employees e

WHERE

e.employee\_id = 100

UNION ALL

-- Recursive member: Find subordinates of the previous level

SELECT

e.employee\_id,

e.last\_name AS employee\_last\_name,

eh.employee\_last\_name AS manager\_last\_name,

eh.position + 1 AS position

FROM

employees e

INNER JOIN

EmployeeHierarchy eh ON e.manager\_id = eh.employee\_id

)

-- Select the required columns

SELECT

employee\_last\_name,

manager\_last\_name,

position

FROM

EmployeeHierarchy

ORDER BY

position, employee\_last\_name;

17. Problem: − Produce a list of the earliest hire date, the latest hire date, and the number of employees from the employees table

Answer:

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

hire\_date DATE NOT NULL

);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, hire\_date)

VALUES

(1, 'Smith', '2022-01-15'),

(2, 'Johnson', '2023-02-20'),

(3, 'Williams', '2021-05-30'),

(4, 'Jones', '2023-07-01'),

(5, 'Brown', '2022-11-05');

-- Query to find the earliest hire date, latest hire date, and number of employees

SELECT

MIN(hire\_date) AS earliest\_hire\_date,

MAX(hire\_date) AS latest\_hire\_date,

COUNT(\*) AS number\_of\_employees

FROM

employees;

18. Problem: − Create a list of department names and the departmental costs (salaries added up) − Include only departments whose salary costs are between 15000 and 31000, and sort the listing by the cost

Answer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL,

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, department\_id)

VALUES

(1, 'Smith', 5000, 1),

(2, 'Johnson', 6000, 1),

(3, 'Williams', 8000, 2),

(4, 'Jones', 7000, 2),

(5, 'Brown', 9000, 3),

(6, 'Davis', 10000, 3),

(7, 'Miller', 4000, 4),

(8, 'Wilson', 3500, 5);

-- Query to calculate departmental costs and filter based on criteria

SELECT

d.department\_name,

SUM(e.salary) AS total\_salary

FROM

employees e

JOIN

departments d ON e.department\_id = d.department\_id

GROUP BY

d.department\_name

HAVING

SUM(e.salary) BETWEEN 15000 AND 31000

ORDER BY

total\_salary;

19. Problem: − Create a list of department names, the manager id, manager name (employee last name) of that department, and the average salary in each department

Answer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL,

manager\_id INT,

FOREIGN KEY (manager\_id) REFERENCES employees(employee\_id)

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL,

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name, manager\_id)

VALUES

(1, 'Engineering', 1),

(2, 'Human Resources', 3),

(3, 'Marketing', 5),

(4, 'Sales', 7),

(5, 'Finance', 8);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, department\_id)

VALUES

(1, 'Smith', 12000, 1), -- Manager of Engineering

(2, 'Johnson', 6000, 1),

(3, 'Williams', 8000, 2), -- Manager of HR

(4, 'Jones', 7000, 2),

(5, 'Brown', 9000, 3), -- Manager of Marketing

(6, 'Davis', 10000, 3),

(7, 'Miller', 15000, 4), -- Manager of Sales

(8, 'Wilson', 3500, 5); -- Manager of Finance

-- Query to find the required information

SELECT

d.department\_name,

d.manager\_id,

m.last\_name AS manager\_name,

AVG(e.salary) AS average\_salary

FROM

departments d

JOIN

employees m ON d.manager\_id = m.employee\_id

JOIN

employees e ON d.department\_id = e.department\_id

GROUP BY

d.department\_name, d.manager\_id, m.last\_name

ORDER BY

d.department\_name;

20. Problem: − Show the highest average salary for the departments in the employees table − Round the result to the nearest whole number

Answer:

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL,

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, department\_id)

VALUES

(1, 'Smith', 12000, 1),

(2, 'Johnson', 6000, 1),

(3, 'Williams', 8000, 2),

(4, 'Jones', 7000, 2),

(5, 'Brown', 9000, 3),

(6, 'Davis', 10000, 3),

(7, 'Miller', 15000, 4),

(8, 'Wilson', 3500, 5);

-- Query to find the highest average salary and round it to the nearest whole number

SELECT

ROUND(MAX(avg\_salary)) AS highest\_average\_salary

FROM

(SELECT

AVG(salary) AS avg\_salary

FROM

employees

GROUP BY

department\_id) AS department\_avg\_salaries;

21. Problem: − Create a list of department names and their monthly costs (salaries added up)

Answer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL, -- Salary is assumed to be monthly

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, department\_id)

VALUES

(1, 'Smith', 12000, 1),

(2, 'Johnson', 6000, 1),

(3, 'Williams', 8000, 2),

(4, 'Jones', 7000, 2),

(5, 'Brown', 9000, 3),

(6, 'Davis', 10000, 3),

(7, 'Miller', 15000, 4),

(8, 'Wilson', 3500, 5);

-- Query to find department names and their total monthly costs

SELECT

d.department\_name,

SUM(e.salary) AS monthly\_cost

FROM

departments d

JOIN

employees e ON d.department\_id = e.department\_id

GROUP BY

d.department\_name

ORDER BY

d.department\_name;

22. Problem: − Create a list of department names, and job\_ids − Calculate the monthly salary cost for each job\_idwithin a department, for each department, and for all departments added together.

Answer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL, -- Salary is assumed to be monthly

job\_id INT NOT NULL,

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, job\_id, department\_id)

VALUES

(1, 'Smith', 12000, 101, 1), -- Job ID 101 in Engineering

(2, 'Johnson', 6000, 102, 1), -- Job ID 102 in Engineering

(3, 'Williams', 8000, 103, 2),-- Job ID 103 in Human Resources

(4, 'Jones', 7000, 104, 2), -- Job ID 104 in Human Resources

(5, 'Brown', 9000, 105, 3), -- Job ID 105 in Marketing

(6, 'Davis', 10000, 105, 3), -- Job ID 105 in Marketing

(7, 'Miller', 15000, 106, 4), -- Job ID 106 in Sales

(8, 'Wilson', 3500, 107, 5); -- Job ID 107 in Finance

-- Query to find department names, job\_ids, and calculate monthly salary costs

SELECT

d.department\_name,

e.job\_id,

SUM(e.salary) AS monthly\_salary\_cost

FROM

departments d

JOIN

employees e ON d.department\_id = e.department\_id

GROUP BY

d.department\_name, e.job\_id

UNION ALL

SELECT

'All Departments' AS department\_name,

NULL AS job\_id,

SUM(e.salary) AS monthly\_salary\_cost

FROM

employees e

ORDER BY

department\_name, job\_id;

23. Problem: −Create a list of department names, and job\_ids −Calculate the monthly salary cost for each job\_idwithin a department, for each department, for each group of job\_idsirrespective of the department, and for all departments added together (Hint: Cube)

Anwer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL, -- Salary is assumed to be monthly

job\_id INT NOT NULL,

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, job\_id, department\_id)

VALUES

(1, 'Smith', 12000, 101, 1), -- Job ID 101 in Engineering

(2, 'Johnson', 6000, 102, 1), -- Job ID 102 in Engineering

(3, 'Williams', 8000, 103, 2),-- Job ID 103 in Human Resources

(4, 'Jones', 7000, 104, 2), -- Job ID 104 in Human Resources

(5, 'Brown', 9000, 105, 3), -- Job ID 105 in Marketing

(6, 'Davis', 10000, 105, 3), -- Job ID 105 in Marketing

(7, 'Miller', 15000, 106, 4), -- Job ID 106 in Sales

(8, 'Wilson', 3500, 107, 5); -- Job ID 107 in Finance

-- Query to calculate monthly salary costs using CUBE

SELECT

COALESCE(d.department\_name, 'All Departments') AS department\_name,

COALESCE(e.job\_id, 'All Job IDs') AS job\_id,

SUM(e.salary) AS monthly\_salary\_cost

FROM

employees e

LEFT JOIN

departments d ON e.department\_id = d.department\_id

GROUP BY

CUBE(d.department\_name, e.job\_id)

ORDER BY

department\_name, job\_id;

24. Problem: − Expand the previous list to also show if the department\_idor job\_id was used to create the subtotals shown in the output (Hint: Cube, Grouping)

Answer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL, -- Salary is assumed to be monthly

job\_id INT NOT NULL,

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, job\_id, department\_id)

VALUES

(1, 'Smith', 12000, 101, 1), -- Job ID 101 in Engineering

(2, 'Johnson', 6000, 102, 1), -- Job ID 102 in Engineering

(3, 'Williams', 8000, 103, 2),-- Job ID 103 in Human Resources

(4, 'Jones', 7000, 104, 2), -- Job ID 104 in Human Resources

(5, 'Brown', 9000, 105, 3), -- Job ID 105 in Marketing

(6, 'Davis', 10000, 105, 3), -- Job ID 105 in Marketing

(7, 'Miller', 15000, 106, 4), -- Job ID 106 in Sales

(8, 'Wilson', 3500, 107, 5); -- Job ID 107 in Finance

-- Query to calculate monthly salary costs with grouping details

SELECT

COALESCE(d.department\_name, 'All Departments') AS department\_name,

COALESCE(e.job\_id, 'All Job IDs') AS job\_id,

SUM(e.salary) AS monthly\_salary\_cost,

GROUPING(d.department\_name) AS is\_department\_subtotal,

GROUPING(e.job\_id) AS is\_job\_id\_subtotal

FROM

employees e

LEFT JOIN

departments d ON e.department\_id = d.department\_id

GROUP BY

CUBE(d.department\_name, e.job\_id)

ORDER BY

department\_name, job\_id;

25. Problem: − Create a list that includes the monthly salary costs for each job title within a department − In the same list, display the monthly salary cost per city. (Hint: Grouping Sets)

Answer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL, -- Salary is assumed to be monthly

job\_id INT NOT NULL,

department\_id INT,

city VARCHAR(255),

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, job\_id, department\_id, city)

VALUES

(1, 'Smith', 12000, 101, 1, 'New York'), -- Job ID 101 in Engineering

(2, 'Johnson', 6000, 102, 1, 'New York'), -- Job ID 102 in Engineering

(3, 'Williams', 8000, 103, 2, 'Chicago'), -- Job ID 103 in Human Resources

(4, 'Jones', 7000, 104, 2, 'Chicago'), -- Job ID 104 in Human Resources

(5, 'Brown', 9000, 105, 3, 'San Francisco'), -- Job ID 105 in Marketing

(6, 'Davis', 10000, 105, 3, 'San Francisco'), -- Job ID 105 in Marketing

(7, 'Miller', 15000, 106, 4, 'Boston'), -- Job ID 106 in Sales

(8, 'Wilson', 3500, 107, 5, 'Los Angeles'); -- Job ID 107 in Finance

-- Query to calculate monthly salary costs using grouping sets

SELECT

d.department\_name,

e.job\_id,

e.city,

SUM(e.salary) AS monthly\_salary\_cost

FROM

employees e

LEFT JOIN

departments d ON e.department\_id = d.department\_id

GROUP BY

GROUPING SETS (

(d.department\_name, e.job\_id),

(e.city),

()

)

ORDER BY

d.department\_name, e.job\_id, e.city;

26. •Problem: −Create a list of employee names as shown and department ids −In the same report, list the department ids and department names. And finally, list the cities −The rows should not be joined, just listed in the same report. (Hint: Union)

Answer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL, -- Salary is assumed to be monthly

job\_id INT NOT NULL,

department\_id INT,

city VARCHAR(255),

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, last\_name, salary, job\_id, department\_id, city)

VALUES

(1, 'Smith', 12000, 101, 1, 'New York'),

(2, 'Johnson', 6000, 102, 1, 'New York'),

(3, 'Williams', 8000, 103, 2, 'Chicago'),

(4, 'Jones', 7000, 104, 2, 'Chicago'),

(5, 'Brown', 9000, 105, 3, 'San Francisco'),

(6, 'Davis', 10000, 105, 3, 'San Francisco'),

(7, 'Miller', 15000, 106, 4, 'Boston'),

(8, 'Wilson', 3500, 107, 5, 'Los Angeles');

-- Query to create the report with UNION

SELECT

last\_name AS name\_or\_id,

department\_id AS department\_id\_or\_null,

NULL AS department\_name\_or\_null,

NULL AS city\_or\_null

FROM

employees

UNION ALL

SELECT

NULL AS name\_or\_id,

department\_id AS department\_id\_or\_null,

department\_name AS department\_name\_or\_null,

NULL AS city\_or\_null

FROM

departments

UNION ALL

SELECT

NULL AS name\_or\_id,

NULL AS department\_id\_or\_null,

NULL AS department\_name\_or\_null,

city AS city\_or\_null

FROM

employees

ORDER BY

name\_or\_id, department\_id\_or\_null, department\_name\_or\_null, city\_or\_null;

27. Problem: − Create a list of each employee's first initial and last name, salary, and department name for each employee earning more than the average for his department.

Answer:

-- Assuming the departments table structure

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

-- Assuming the employees table structure

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

salary DECIMAL(10, 2) NOT NULL, -- Salary is assumed to be monthly

job\_id INT NOT NULL,

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Insert sample data into departments table

INSERT INTO departments (department\_id, department\_name)

VALUES

(1, 'Engineering'),

(2, 'Human Resources'),

(3, 'Marketing'),

(4, 'Sales'),

(5, 'Finance');

-- Insert sample data into employees table

INSERT INTO employees (employee\_id, first\_name, last\_name, salary, job\_id, department\_id)

VALUES

(1, 'John', 'Smith', 12000, 101, 1),

(2, 'Jane', 'Johnson', 6000, 102, 1),

(3, 'Emily', 'Williams', 8000, 103, 2),

(4, 'Michael', 'Jones', 7000, 104, 2),

(5, 'David', 'Brown', 9000, 105, 3),

(6, 'Sarah', 'Davis', 10000, 105, 3),

(7, 'Chris', 'Miller', 15000, 106, 4),

(8, 'Jessica', 'Wilson', 3500, 107, 5);

-- Query to get employees earning more than the average salary for their department

SELECT

LEFT(e.first\_name, 1) AS first\_initial,

e.last\_name,

e.salary,

d.department\_name

FROM

employees e

JOIN

departments d ON e.department\_id = d.department\_id

WHERE

e.salary > (

SELECT AVG(e2.salary)

FROM employees e2

WHERE e2.department\_id = e.department\_id

)

ORDER BY

e.last\_name, e.first\_name;