**SQL MEGA PROJECT**

**Create regions table**

CREATE TABLE regions (

region\_id INT PRIMARY KEY,

region\_name VARCHAR (25) DEFAULT NULL

);

**Create countries table**

CREATE TABLE countries (

country\_id CHAR (2) PRIMARY KEY,

country\_name VARCHAR (40) DEFAULT NULL,

region\_id INT NOT NULL,

FOREIGN KEY (region\_id) REFERENCES regions (region\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

**Create locations table**

CREATE TABLE locations (

location\_id INT PRIMARY KEY,

street\_address VARCHAR (40) DEFAULT NULL,

postal\_code VARCHAR (12) DEFAULT NULL,

city VARCHAR (30) NOT NULL,

state\_province VARCHAR (25) DEFAULT NULL,

country\_id CHAR (2) NOT NULL,

FOREIGN KEY (country\_id) REFERENCES countries (country\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

**Create jobs table**

CREATE TABLE jobs (

job\_id INT PRIMARY KEY,

job\_title VARCHAR (35) NOT NULL,

min\_salary DECIMAL (8, 2) DEFAULT NULL,

max\_salary DECIMAL (8, 2) DEFAULT NULL

);

**Create departments table**

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR (30) NOT NULL,

location\_id INT DEFAULT NULL,

FOREIGN KEY (location\_id) REFERENCES locations (location\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

**Create employees table**

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR (20) DEFAULT NULL,

last\_name VARCHAR (25) NOT NULL,

email VARCHAR (100) NOT NULL,

phone\_number VARCHAR (20) DEFAULT NULL,

hire\_date DATE NOT NULL,

job\_id INT NOT NULL,

salary DECIMAL (8, 2) NOT NULL,

manager\_id INT DEFAULT NULL,

department\_id INT DEFAULT NULL,

FOREIGN KEY (job\_id) REFERENCES jobs (job\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (department\_id) REFERENCES departments (department\_id) ON DELETE CASCADE ON UPDATE CASCADE,

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

);

**Create dependents table**

CREATE TABLE dependents (

dependent\_id INT PRIMARY KEY,

first\_name VARCHAR (50) NOT NULL,

last\_name VARCHAR (50) NOT NULL,

relationship VARCHAR (25) NOT NULL,

employee\_id INT NOT NULL,

FOREIGN KEY (employee\_id) REFERENCES employees (employee\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

**Insert data into regions table**

INSERT INTO regions(region\_id, region\_name) VALUES (1, 'Europe');

INSERT INTO regions(region\_id, region\_name) VALUES (2, 'Americas');

INSERT INTO regions(region\_id, region\_name) VALUES (3, 'Asia');

INSERT INTO regions(region\_id, region\_name) VALUES (4, 'Middle East and Africa');

**Insert data into countries table**

(Continue with the provided data for countries, locations, jobs, departments, employees, and dependents)

**1**

1. SELECT Statements:

A. To get data from all the rows and columns in the employees table:

SELECT \*

FROM employees;

1. B. To select data from the employee id, first name, last name, and hire date of all rows in the employees table:

SELECT employee\_id, first\_name, last\_name, hire\_date

FROM employees;

1. C. To get the first name, last name, salary, and new salary:

SELECT first\_name, last\_name, salary, salary \* 2 AS new\_salary

FROM employees;

1. D. Increase the salary two times and named as New\_SALARY from employees table:

SELECT \*, salary \* 2 AS new\_salary

FROM employees;

2. ORDER BY Statements:

A. Returns the data from the employee id, first name, last name, hire date, and salary column of the employees table:

SELECT employee\_id, first\_name, last\_name, hire\_date, salary

FROM employees;

**2.** B. To sort employees by first names in alphabetical order:

SELECT \*

FROM employees

ORDER BY first\_name;

2. C. To sort the employees by the first name in ascending order and the last name in descending order:

SELECT \*

FROM employees

ORDER BY first\_name ASC, last\_name DESC;

2. D. To sort employees by salary from high to low:

SELECT \*

FROM employees

ORDER BY salary DESC;

2. E. To sort the employees by values in the hire\_date column from:

SELECT \*

FROM employees

ORDER BY hire\_date;

2. F. Sort the employees by the hire dates in descending order:

SELECT \*

FROM employees

ORDER BY hire\_date DESC;

**3. DISTINCT Statements:**

A. Selects the salary data from the salary column of the employees table and sorts them from high to low:

SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC;

3. B. Select unique values from the salary column of the employees table:

SELECT DISTINCT salary

FROM employees;

3. C. Selects the job id and salary from the employees table:

SELECT DISTINCT job\_id, salary

FROM employees;

3. D. To remove the duplicate values in job id and salary:

SELECT DISTINCT job\_id, salary

FROM employees;

3. E. Returns the distinct phone numbers of employees:

SELECT DISTINCT phone\_number

FROM employees;

**4. TOP N Statements:**

A. Returns all rows in the employees table sorted by the first\_name column.

SELECT \*

FROM employees

ORDER BY first\_name;

4. B. To return the first 5 rows in the result set returned by the SELECT clause:

SELECT TOP 5 \*

FROM employees;

4. C. To return five rows starting from the 4th row:

SELECT \*

FROM employees

ORDER BY column\_list OFFSET 3 ROWS FETCH NEXT 5 ROWS ONLY;

4. D. Gets the top five employees with the highest salaries.

SELECT TOP 5 \*

FROM employees

ORDER BY salary DESC;

4. E. To get employees who have the 2nd highest salary in the company.

SELECT TOP 1 \*

FROM employees

ORDER BY salary DESC OFFSET 1 ROW;

5. WHERE CLAUSE and COMPARISON OPERATORS:

A. Query finds employees who have salaries greater than 14,000 and sorts the result sets based on the salary in descending order.

SELECT \*

FROM employees

WHERE salary > 14000

ORDER BY salary DESC;

5. B. Query finds all employees who work in the department id 5.

SELECT \*

FROM employees

WHERE department\_id = 5;

5. C. Query finds the employee whose last name is Chen.

SELECT \*

FROM employees

WHERE last\_name = 'Chen';

5. D. To get all employees who joined the company after January 1st, 1999.

SELECT \*

FROM employees

WHERE hire\_date > '1999-01-01';

5. E. To find the employees who joined the company in 1999.

SELECT \*

FROM employees

WHERE hire\_date >= '1999-01-01' AND hire\_date < '2000-01-01';

5. F. Statement finds the employee whose last name is Himuro.

SELECT \*

FROM employees

WHERE last\_name = 'Himuro';

5. G. The query searches for the string Himuro in the last\_name column of the employees table.

SELECT \*

FROM employees

WHERE last\_name LIKE '%Himuro%';

5. H. To find all employees who do not have phone numbers:

SELECT \*

FROM employees

WHERE phone\_number IS NULL;

5. I. Returns all employees whose department id is not 8.

SELECT \*

FROM employees

WHERE department\_id <> 8;

5. J. Finds all employees whose department id is not eight and ten.

SELECT \*

FROM employees

WHERE department\_id NOT IN (8, 10);

5. K. To find the employees whose salary is greater than 10,000.

SELECT \*

FROM employees

WHERE salary > 10000;

5. L. Finds employees in department 8 and have the salary greater than 10,000.

SELECT \*

FROM employees

WHERE department\_id = 8 AND salary > 10000;

5. M. The statement below returns all employees whose salaries are less than 10,000.

SELECT \*

FROM employees

WHERE salary < 10000;

5. N. Finds employees whose salaries are greater than or equal 9,000.

SELECT \*

FROM employees

WHERE salary >= 9000;

5. O. Finds employees whose salaries are less than or equal to 9,000.

SELECT \*

FROM employees

WHERE salary <= 9000;

6. ALTER TABLE Queries for the courses Table:

A. Adds a new column named credit\_hours to the courses table.

ALTER TABLE courses

ADD credit\_hours INT;

6. B. Adds the fee and max\_limit columns to the courses table and places these columns after the course\_name column.

ALTER TABLE courses

ADD fee DECIMAL, max\_limit INT

AFTER course\_name;

6. C. Changes the attribute of the fee column to NOT NULL.

ALTER TABLE courses

ALTER COLUMN fee DECIMAL NOT NULL;

6. D. To remove the fee column of the courses table.

ALTER TABLE courses

DROP COLUMN fee;

6. E. Removes the max\_limit and credit\_hours columns of the courses table.

ALTER TABLE courses

DROP COLUMN max\_limit, DROP COLUMN credit\_hours;

7. SQL FOREIGN KEY Constraint Examples:

A. To add an SQL FOREIGN KEY constraint to the project\_milestones table to enforce the relationship between the projects and project\_milestones tables.

ALTER TABLE project\_milestones

ADD CONSTRAINT fk\_project\_milestones\_project

FOREIGN KEY (project\_id) REFERENCES projects(project\_id);

7. B. Suppose the project\_milestones already exists without any predefined foreign key, and you want to define a FOREIGN KEY constraint for the project\_id column.

ALTER TABLE project\_milestones

ADD CONSTRAINT fk\_project\_milestones\_project

FOREIGN KEY (project\_id) REFERENCES projects(project\_id);

Logical Operators

1. AND Operator

SELECT \* FROM employees

WHERE salary > 50000 AND department\_id = 5;

2. NOT Operator

SELECT \* FROM employees

WHERE NOT department\_id = 10;

3. OR Operator

SELECT \* FROM employees

WHERE job\_id = 'IT\_PROG' OR job\_id = 'HR\_REP';

Special Operators

4. ANY Operator

SELECT \* FROM employees

WHERE salary > ANY (SELECT salary FROM employees WHERE department\_id = 5);

5. BETWEEN Operator

SELECT \* FROM employees

WHERE hire\_date BETWEEN '2005-01-01' AND '2010-12-31';

6. EXISTS Operator

SELECT \* FROM employees

WHERE EXISTS (SELECT \* FROM departments WHERE department\_id = 20);

7. IN Operator

SELECT \* FROM employees

WHERE department\_id IN (5, 10, 15);

8. LIKE Operator

SELECT \* FROM employees

WHERE last\_name LIKE 'S%';

9. ALL Operator

SELECT \* FROM employees

WHERE salary > ALL (SELECT salary FROM employees WHERE department\_id = 5);

Part 1

A.

SELECT \* FROM employees

WHERE salary > 5000 AND salary < 7000;

B.

SELECT \* FROM employees

WHERE salary = 7000 OR salary = 8000;

C.

SELECT \* FROM employees

WHERE phone\_number IS NULL;

D.

SELECT \* FROM employees

WHERE salary BETWEEN 9000 AND 12000;

E.

SELECT \* FROM employees

WHERE department\_id IN (8, 9);

F.

SELECT \* FROM employees

WHERE first\_name LIKE 'jo%';

G.

SELECT \* FROM employees

WHERE first\_name LIKE '\_h%';

H.

SELECT \* FROM employees

WHERE salary > ALL (SELECT salary FROM employees WHERE department\_id = 8);

Part 2

A.

SELECT \* FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees GROUP BY department\_id);

B.

SELECT \* FROM employees

WHERE employee\_id IN (SELECT DISTINCT employee\_id FROM dependents);

C.

SELECT \* FROM employees

WHERE salary BETWEEN 2500 AND 2900;

D.

SELECT \* FROM employees

WHERE salary NOT BETWEEN 2500 AND 2900;

E.

SELECT \* FROM employees

WHERE hire\_date BETWEEN '1999-01-01' AND '2000-12-31';

F.

SELECT \* FROM employees

WHERE hire\_date NOT BETWEEN '1989-01-01' AND '1999-12-31';

G.

SELECT \* FROM employees

WHERE hire\_date BETWEEN '1990-01-01' AND '1993-12-31';

Part 3

A.

SELECT \* FROM employees

WHERE first\_name LIKE 'Da%';

B.

SELECT \* FROM employees

WHERE first\_name LIKE '%er';

C.

SELECT \* FROM employees

WHERE last\_name LIKE '%an%';

D.

SELECT \* FROM employees

WHERE first\_name LIKE 'Jo\_\_%';

E.

SELECT \* FROM employees

WHERE first\_name LIKE 'S%\_' AND first\_name NOT LIKE 'Sh%';

Part 4

A.

SELECT \* FROM employees

WHERE department\_id = 5;

B.

SELECT \* FROM employees

WHERE department\_id = 5 AND salary <= 5000

C.

SELECT \* FROM employees

WHERE department\_id NOT IN (1, 2, 3);

D.

SELECT \* FROM employees

WHERE first\_name NOT LIKE 'D%';

E.

SELECT \* FROM employees

WHERE salary NOT BETWEEN 1000 AND 5000;

Part 5

A. To get the employees who do not have any dependents:

SELECT \*

FROM employees

WHERE employee\_id NOT IN (SELECT DISTINCT employee\_id FROM dependents);

B. To find all employees who do not have phone numbers:

SELECT \*

FROM employees

WHERE phone\_number IS NULL;

C. To find all employees who have phone numbers:

SELECT \*

FROM employees

WHERE phone\_number IS NOT NULL;