Set operators

1. Retrieve a list of all employees who have a salary greater than 5000 or who work in the "Research" department, sorted by employee name in ascending order, using the UNION operator.

```
SELECT ename, sal, deptno
FROM emp
WHERE sal > 5000
UNION
SELECT ename, sal, deptno
FROM emp
WHERE deptno = 20
ORDER BY ename ASC;
```

2. Find the names of all employees who work in departments with more than 10 employees, but who do not have the letter "e" in their name, sorted by department name in descending order, using the EXCEPT operator.

```
SELECT ename, deptno
FROM emp
WHERE ename NOT LIKE '%e%'
  AND deptno IN (
    SELECT deptno
    FROM emp
    GROUP BY deptno
    HAVING COUNT(*) > 10
  )
EXCEPT
SELECT ename, deptno
FROM emp
WHERE ename LIKE '%e%'
ORDER BY deptno DESC;
```

3. Retrieve a list of all employees who work in departments with names that contain the word "Sales", and all employees who work in departments with names that contain the word "Marketing", sorted by department name in ascending order, using the UNION ALL operator.

```
SELECT ename, deptno
FROM emp
WHERE deptno IN (
  SELECT deptno
  FROM dept
  WHERE dname LIKE '%Sales%'
)
UNION ALL
SELECT ename, deptno
FROM emp
WHERE deptno IN (
  SELECT deptno
  FROM dept
  WHERE dname LIKE '%Marketing%'
)
ORDER BY deptno ASC;
```

4. Find the names of all employees who work in departments with more than 5 employees, but who do not have the letter "a" in their name, and are not managers, sorted by department name in descending order, using the EXCEPT operator.

```
SELECT ename, deptno
FROM emp
WHERE deptno IN (
  SELECT deptno
  FROM emp
  WHERE job != 'MANAGER'
  GROUP BY deptno
  HAVING COUNT(*) > 5
)
```

```sql
EXCEPT
SELECT ename, deptno
FROM emp
WHERE job = 'MANAGER' OR ename LIKE '%a%'
ORDER BY deptno DESC;
```

5. Retrieve a list of all employees who work in departments with names that contain the word "Research", and all employees who work in departments with names that contain the word "Development", sorted by employee name in ascending order, using the UNION operator.

```sql
SELECT ename, deptno
FROM emp
WHERE deptno IN (
  SELECT deptno
  FROM dept
  WHERE dname LIKE '%Research%'
)
UNION
SELECT ename, deptno
FROM emp
WHERE deptno IN (
  SELECT deptno
  FROM dept
  WHERE dname LIKE '%Development%'
)
ORDER BY ename ASC;
```

6. Find the names of all employees who work in departments with less than 5 employees, or who have the letter "j" in their name, sorted by employee name in ascending order, using the UNION operator.

```sql
SELECT ename
FROM emp
WHERE deptno IN (
  SELECT deptno
  FROM emp
  GROUP BY deptno
  HAVING COUNT(*) < 5
)
UNION
SELECT ename
FROM emp
WHERE ename LIKE '%j%'
ORDER BY ename ASC;
```

7. Display a list of all departments that have more than 3 employees, but do not have any employees with the letter "o" in their name, sorted by department name in ascending order, using the INTERSECT operator.

```sql
SELECT dname
FROM dept
WHERE deptno IN (
  SELECT deptno
  FROM emp
  GROUP BY deptno
  HAVING COUNT(*) > 3
)
INTERSECT
SELECT dname
FROM dept
WHERE deptno NOT IN (
  SELECT deptno
  FROM emp
  WHERE ename LIKE '%o%'
)
ORDER BY dname ASC;
```

8. Find the names of all employees who work in departments with more than 5 employees, and are either managers or have the letter "i" in their name, sorted by department name in

descending order, using the UNION operator.

```
select e.ename from emp e join dept d on e.deptno = d.deptno
where d.dname = (select dm.dname from dept dm where
e.deptno = dm.deptno
group by dm.dname having count(*) > 5)
and e.empno = e.mgr
union
select ename from emp where ename like '%i%';
```

9. Give a list of all departments that have more than 2 employees, and all employees who work in departments with names that contain the word "Accounting", sorted by department name in ascending order, using the UNION ALL operator.

```
select d.dname from emp e join dept d on e.deptno = d.deptno group by d.dname
having count(*) > 2
union all
select em.ename from emp em join
dept dm on em.deptno = dm.deptno
where
dm.dname = ( select de.dname from dept de where em.empno = de.deptno group by
de.dname having de.dname = 'Accounting');
```

10. Find the names of all employees who work in departments with less than 5 employees, and have the letter "m" in their name, or are managers, sorted by department name in ascending order, using the UNION operator.

```
select e.ename from emp e join dept d on e.deptno = d.deptno where
d.dname in (select de.dname from emp em join dept de on em.deptno = de.deptno
group by de.dname having count(*) < 5)
union
select ename from emp where ename like '%m%' or empno = mgr;
```

Sub-Query

11. Suppose you have a database containing information about employees and their salaries. Write a SQL query that uses a subquery to return the average salary of all employees who work in the same department as employee "John Smith".

```
SELECT AVG(sal) as avg_salary
FROM emp
WHERE deptno = (SELECT deptno FROM emp WHERE ename = 'John Smith');
```

12. Consider a database that contains information about students and their grades. Write a SQL query that uses a subquery to return the name of the student who has the highest grade in a given course.

```
SELECT name
FROM students
WHERE id = (SELECT student_id
            FROM grades
            WHERE course_id = <course_id>
            ORDER BY grade DESC
            LIMIT 1);
```

13. Suppose you have a database that contains information about products and their prices. Write a SQL query that uses a subquery to return the name of the product that has the highest price

```
SELECT name
FROM products
WHERE price = (SELECT MAX(price) FROM products);
```

Co-related Sub-Query

14. Find the top three highest-paid employees in each department. Use a correlated subquery to join the employees table with itself and find the top three salaries for each department.

```
SELECT e1.empno, e1.ename, e1.sal, e1.deptno
FROM employees e1
WHERE (
  SELECT COUNT(DISTINCT e2.sal)
  FROM employees e2
  WHERE e2.sal > e1.sal AND e2.deptno = e1.deptno
) < 3
ORDER BY e1.deptno, e1.sal DESC;
```

15. List the names of all employees who earn more than their department manager.

```
SELECT e.ename
FROM employees e
JOIN employees m ON e.mgr = m.empno
WHERE e.sal > m.sal
```

16. Calculate the average salary for all departments with at least one employee earning a salary greater than 3,000. (Try using exists). List the Department name and the average salary

```
SELECT d.dname AS DepartmentName, AVG(e.sal) AS AvgSalary
FROM dept d
JOIN emp e ON d.deptno = e.deptno
WHERE EXISTS (
  SELECT 1
  FROM emp
  WHERE deptno = d.deptno AND sal > 3000
)
GROUP BY d.dname
ORDER BY AvgSalary DESC;
```

Stored Procedure

17. Create a stored procedure that inserts a new employee into the database. The procedure should accept the employee's name, job title, department number, and salary as input and insert the new employee into the EMP table. The procedure should also validate that the department number exists in the DEPT table before inserting the employee.

```
CREATE PROCEDURE sp_InsertEmployee
    @Name VARCHAR(50),
    @JobTitle VARCHAR(50),
    @DeptNo INT,
    @Salary DECIMAL(10, 2)
AS
BEGIN
    SET NOCOUNT ON;
    IF NOT EXISTS (SELECT * FROM DEPT WHERE DeptNo = @DeptNo)
    BEGIN
        PRINT 'Department number does not exist.';
        RETURN;
    END;
    INSERT INTO EMP (Name, JobTitle, DeptNo, Salary)
    VALUES (@Name, @JobTitle, @DeptNo, @Salary);

    PRINT 'Employee inserted successfully.';
END;
```

18. Create a stored procedure that updates the salary of all employees in a given department by a specified percentage. The procedure should accept the department number and percentage increase as input and update the salary for all employees in that department.

```
CREATE PROCEDURE UpdateSalaryByDept
    @deptno INT,
    @percentage FLOAT
AS
BEGIN
    UPDATE EMP
    SET SAL = SAL * (1 + @percentage/100)
    WHERE DEPTNO = @deptno;
END
```

User defined Functions

19. Create a function that returns the total number of employees in a given department. The function should accept the department number as input and return the total number of employees in that department.

```
CREATE FUNCTION getNumEmployeesInDept(dept_num INT)
RETURNS INT
BEGIN
    DECLARE num_employees INT;
    SELECT COUNT(*) INTO num_employees FROM employees WHERE department_id = 
dept_num;
    RETURN num_employees;
END
```

20. Create a function that returns the name of the department with the highest average salary. The function should calculate the average salary for each department and return the name of the department with the highest average salary.st the Department name and the average salary

```
CREATE FUNCTION highest_avg_salary_dept()
RETURNS VARCHAR(30)
AS
BEGIN
    DECLARE @dept_name VARCHAR(30);

    SELECT TOP 1 @dept_name = d.dname
    FROM dept d
    JOIN emp e ON d.deptno = e.deptno
    GROUP BY d.dname
    ORDER BY AVG(e.sal) DESC;

    RETURN @dept_name;
END;
```