Complex Stored Procs
DCL
Nested Queries

Complex Stored procedure

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Stored Procedure Syntax
```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Execute a Stored Procedure
```
EXEC procedure_name;
```

Example

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
```

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
```

Execute the stored procedure above as follows:

Example
```
EXEC SelectAllCustomers;
```

Link :https://www.programiz.com/sql/stored-procedures#google_vignette

Stored Procedure Parameters: Input, Output, Optional

1) With Input

```
      CREATE PROCEDURE uspUpdateEmpSalary
(
     @empId int
     ,@salary money
)
AS
BEGIN
    UPDATE dbo.Employee
    SET Salary = @salary
    WHERE EmployeeID = @empId
END
```

2) With Output

```
CREATE PROCEDURE uspGetManagerID
   @empId int,
   @managerId int OUTPUT
AS
BEGIN
   SELECT @managerId = ManagerID
   FROM dbo.Employee
   WHERE EmployeeID = @empId
END

DECLARE @managerID int

EXECUTE uspGetManagerID @empId = 2, @managerId OUTPUT

PRINT @managerId
```

3) Optional Parameter

```
CREATE PROCEDURE uspUpdateEmpSalary
(
     @empId int
     ,@salary money = 1000
)
AS
BEGIN
    UPDATE dbo.Employee
    SET Salary = @salary
    WHERE EmployeeID = @empId
END
```

DCL (Data Control Language)
DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

List of DCL commands:
GRANT: This command gives users access privileges to the database.
Syntax:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER,
ANOTHER_USER;
```

REVOKE: This command withdraws the user's access privileges given by using the GRANT command.

Syntax:

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

TCL (Transaction Control Language)
Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure.

BEGIN: Opens a Transaction.

COMMIT: Commits a Transaction.

Syntax:

```
COMMIT;
```

ROLLBACK: Rollbacks a transaction in case of any error occurs.

Syntax:

```
ROLLBACK;
```

SAVEPOINT: Sets a save point within a transaction.

Syntax:

```
SAVEPOINT SAVEPOINT_NAME;
```

Nested Queries

Nested queries are a way to perform more complex queries by embedding one query within another. A nested query is a query that appears inside another query, and it helps retrieve data from multiple tables or apply conditions based on the results of another query. The result of inner query is used in execution of outer query

2 Types
1) Independent Subqueries
2) corelated subqueries

1) Independent Subqueries

In independent nested queries, query execution starts from innermost query to outermost queries. The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query

Excersise
1) 4 tables
   a) Student (S_ID , S_NAME,S_ADDRESS,S_PHONE,S_AGE)
   b) Courses (C_ID,C_NAME)
   c) Student_Courses (ID, S_ID C_ID)
   d) MARKS (ID , SEMESTER , MARKS)

1) Find all students who have scored more than 90 in Data Structures in first Sem.

a) Break in smaller parts
b) find course_id for Data Structures from courses table
c) find all Student_Cources 'id' enrolled in Data Structures based on course_id
d) filter the previous id find all id who has scored

more than 90 in first sem from Marks .
e) Find all S_id from student_Courses and get data from Student

2) Corelated Subqueries

A correlated subquery is one way of reading every row in a table and comparing values in each row against related data. It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query.

General form

```
SELECT column1, column2, ....
FROM table1 outer
WHERE column1 operator
                    (SELECT column1, column2
                     FROM table2
                     WHERE expr1 =
                               outer.expr2);
```

Scenario

Select Salary of employees whose salary is less that the average salary of the department .

```
SELECT last_name, salary, department_id
 FROM employees outer
 WHERE salary <
                (SELECT AVG(salary)
                 FROM employees
                 WHERE department_id =
                       outer.department_id group by department_id);
```

UNION

`UNION` merges the results of two SELECT statements. Important: `UNION` statements only return UNIQUE values

UNION ALL
Same as union but returns all data of all table

MINUS
It returns values that are present in one table that are not present in another table.

Intersect

Difference between Intersect and Join

INTERSECT just compares 2 sets and picks only distinct equivalent values from both sets. It's important to note that NULL marks are considered as equals in INTERSECT set operator. Also sets should contain equal number of columns with implicitly convertible types.

https://sqliteonline.com/