1. ***The SQL UNION Operator***

* The UNION operator is used to combine the result-set of two or more SELECT statements.
* Notes :
* Every SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in every SELECT statement must also be in the same order
* UNION Syntax :
* **SELECT column\_name(s) FROM table1  
  UNION  
  SELECT column\_name(s) FROM table2;**
* UNION ALL Syntax :
* **SELECT column\_name(s) FROM table1  
  UNION ALL  
  SELECT column\_name(s) FROM table2;**
* Notes :
* The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:
* The column names in the result-set are usually equal to the column names in the first SELECT statement.
* UNION Example :
* **SELECT City FROM Customers  
  UNION  
  SELECT City FROM Suppliers  
  ORDER BY City;**
* UNION ALL Example :
* **SELECT City FROM Customers  
  UNION ALL  
  SELECT City FROM Suppliers  
  ORDER BY City;**
* UNION With Where Example :
* **SELECT City, Country FROM Customers  
  WHERE Country='Germany'  
  UNION  
  SELECT City, Country FROM Suppliers  
  WHERE Country='Germany'  
  ORDER BY City;**
* UNION ALL With Where Example :
* **SELECT City, Country FROM Customers  
  WHERE Country='Germany'  
  UNION ALL  
  SELECT City, Country FROM Suppliers  
  WHERE Country='Germany'  
  ORDER BY City;**
* Another UNION Example :
* **SELECT 'Customer' AS Type, ContactName, City, Country  
  FROM Customers  
  UNION  
  SELECT 'Supplier', ContactName, City, Country  
  FROM Suppliers;**

1. ***The SQL GROUP BY Statement***

* The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".
* The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.
* GROUP BY Syntax :
* **SELECT column\_name(s)  
  FROM table\_name  
  WHERE condition  
  GROUP BY column\_name(s)ORDER BY column\_name(s);**
* Example-1 :
* **SELECT COUNT(CustomerID), Country  
  FROM Customers  
  GROUP BY Country;**
* Example-2 :
* **SELECT COUNT(CustomerID), Country  
  FROM Customers  
  GROUP BY Country  
  ORDER BY COUNT(CustomerID) DESC;**
* Example-3 :
* **SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders**

**FROM Orders  
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID  
GROUP BY ShipperName;**

1. ***The SQL HAVING Clause***

* The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.
* HAVING Syntax :
* **SELECT column\_name(s)  
  FROM table\_name  
  WHERE condition  
  GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);**
* Example-1 :
  + **SELECT COUNT(CustomerID), Country  
    FROM Customers  
    GROUP BY Country  
    HAVING COUNT(CustomerID) > 5;**
* Example-2 :
  + **SELECT COUNT(CustomerID), Country  
    FROM Customers  
    GROUP BY Country  
    HAVING COUNT(CustomerID) > 5  
    ORDER BY COUNT(CustomerID) DESC;**
* Example-3 :
* **SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
  FROM (Orders  
  INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID)  
  GROUP BY LastName  
  HAVING COUNT(Orders.OrderID) > 10;**
* Example-4 :
  + **SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
    FROM Orders  
    INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
    WHERE LastName = 'Davolio' OR LastName = 'Fuller'  
    GROUP BY LastName  
    HAVING COUNT(Orders.OrderID) > 25;**

1. ***The SQL EXISTS Operator***

* The EXISTS operator is used to test for the existence of any record in a subquery. The EXISTS operator returns TRUE if the subquery returns one or more records.
* EXISTS Syntax :
* **SELECT column\_name(s)  
  FROM table\_name  
  WHERE EXISTS  
  (SELECT column\_name FROM table\_name WHERE condition);**
* Example-1 :
* **SELECT SupplierName  
  FROM Suppliers  
  WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20);**
* Example-2 :
* **SELECT SupplierName  
  FROM Suppliers  
  WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price = 22);**

1. ***The SQL ANY & ALL Operators***

* The ANY and ALL operators allow you to perform a comparison between a single column value and a range of other values.
* The ANY operator :
* Returns a boolean value as a result
* Returns TRUE if ANY of the subquery values meet the condition
* ANY means that the condition will be true if the operation is true for any of the values in the range.
* ANY Syntax :
* **SELECT column\_name(s)  
  FROM table\_name  
  WHERE column\_name operator ANY  
    (SELECT column\_name  FROM table\_name  WHERE condition);**
* Note : The operator must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).
* The ALL operator:
* Returns a boolean value as a result
* Returns TRUE if ALL of the subquery values meet the condition
* Is used with SELECT, WHERE and HAVING statements
* ALL means that the condition will be true only if the operation is true for all values in the range.
* ALL Syntax With SELECT :
* **SELECT ALL column\_name(s)  
  FROM table\_name  
  WHERE condition;**
* ALL Syntax With WHERE or HAVING :
* **SELECT column\_name(s)  
  FROM table\_name  
  WHERE column\_name operator ALL  
    (SELECT column\_name  FROM table\_name  WHERE condition);**
* Note : The operator must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).
* ANY Example-1 :
* **SELECT ProductName  
  FROM Products  
  WHERE ProductID = ANY  
    (SELECT ProductID  
    FROM OrderDetails  
    WHERE Quantity = 10);**
* ALL Example-1 :
* **SELECT ALL ProductName  
  FROM Products  
  WHERE TRUE;**
* ALL Example-2 :
* **SELECT ProductName  
  FROM Products  
  WHERE ProductID = ALL  
    (SELECT ProductID  
    FROM OrderDetails  
    WHERE Quantity = 10);**

1. ***The SQL SELECT INTO Statement***

* The SELECT INTO statement copies data from one table into a new table.
* SELECT INTO Syntax (Copy all columns into a new table) :
* **SELECT \*  
  INTO newtable [IN externaldb]  
  FROM oldtableWHERE condition;**
* SELECT INTO Syntax (Copy only some columns into a new table) :
* **SELECT column1, column2, column3, ...  
  INTO newtable [IN externaldb]  
  FROM oldtableWHERE condition;**
* Note : The new table will be created with the column-names and types as defined in the old table. You can create new column names using the AS clause.
* SELECT INTO Examples :
* The following SQL statement creates a backup copy of Customers:
  + **SELECT \* INTO CustomersBackup2017  
    FROM Customers;**
* The following SQL statement uses the IN clause to copy the table into a new table in another database:
  + **SELECT \* INTO CustomersBackup2017 IN 'Backup.mdb'  
    FROM Customers;**
* The following SQL statement copies only a few columns into a new table:
  + **SELECT CustomerName, ContactName INTO CustomersBackup2017  
    FROM Customers;**
* The following SQL statement copies only the German customers into a new table:
  + **SELECT \* INTO CustomersGermany  
    FROM Customers  
    WHERE Country = 'Germany';**
* The following SQL statement copies data from more than one table into a new table:
  + **SELECT Customers.CustomerName, Orders.OrderID  
    INTO CustomersOrderBackup2017  
    FROM Customers  
    LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;**
* SELECT INTO can also be used to create a new, empty table using the schema of another. Just add a WHERE clause that causes the query to return no data:
  + **SELECT \* INTO newtable  
    FROM oldtable  
    WHERE 1 = 0;**

1. ***The SQL INSERT INTO SELECT Statement***

* The INSERT INTO SELECT statement copies data from one table and inserts it into another table. The INSERT INTO SELECT statement requires that the data types in source and target tables match.
* **Note:** The existing records in the target table are unaffected.
* INSERT INTO SELECT Syntax (Copy all columns from one table to another table) :
* **INSERT INTO table2  
  SELECT \* FROM table1WHERE condition;**
* INSERT INTO SELECT Syntax (Copy only some columns from one table into another table) :
* **INSERT INTO table2 (column1, column2, column3, ...)  
  SELECT column1, column2, column3, ...  
  FROM table1  
  WHERE condition;**

1. ***The SQL CASE Statement***

* The CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause. If there is no ELSE part and no conditions are true, it returns NULL.
* CASE Syntax :
* **CASE  
      WHEN condition1 THEN result1  
      WHEN condition2 THEN result2  
      WHEN conditionN THEN resultN  
      ELSE result  
  END;**
* Example-1 :
* **SELECT OrderID, Quantity,  
  CASE  
      WHEN Quantity > 30 THEN 'The quantity is greater than 30'  
      WHEN Quantity = 30 THEN 'The quantity is 30'  
      ELSE 'The quantity is under 30'  
  END AS QuantityText  
  FROM OrderDetails;**
* Example-2 :
* **SELECT CustomerName, City, Country  
  FROM Customers  
  ORDER BY  
  (CASE  
      WHEN City IS NULL THEN Country  
      ELSE City  
  END);**