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

* The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.
* SELECT Syntax :
* **SELECT column1, column2, ... FROM table\_name;**
* **SELECT \* FROM table\_name;**
* **SELECT TOP 100 \* FROM table\_name;**

1. ***The SQL SELECT DISTINCT Statement***

* The SELECT DISTINCT statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.
* SELECT DISTINCT Syntax :
* **SELEC DISTINCT column1, column2, ... FROM table\_name;**
* **SELECT DISTINCT \* FROM table\_name;**
* **SELECT DISTINCT TOP 100 \* FROM table\_name;**

1. ***The SQL WHERE Clause***

* The WHERE clause is used to filter records. It is used to extract only those records that fulfill a specified condition.
* WHERE Syntax :
* **SELECT column1, column2, ... FROM table\_name WHERE condition;**
* **SELECT \* FROM table\_name WHERE condition;**
* Operators in The WHERE Clause :

|  |  |
| --- | --- |
| **Operator** | **Description** |
| = | Equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| <> or != | Not equal |
| BETWEEN | Between a certain range |
| LIKE | Search for a pattern |
| IN | To specify multiple possible values for a column |

1. ***The SQL AND, OR and NOT Operators***

* The WHERE clause can be combined with AND, OR, and NOT operators.
* The AND and OR operators are used to filter records based on more than one condition:
* The AND operator displays a record if all the conditions separated by AND are TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.
* The NOT operator displays a record if the condition(s) is NOT TRUE.
* AND Syntax :
  + **SELECT column1, column2, ...  
    FROM table\_name  
    WHERE condition1 AND condition2 AND condition3 ...;**
* OR Syntax :
  + **SELECT column1, column2, ...  
    FROM table\_name  
    WHERE condition1 OR condition2 OR condition3 ...;**
* NOT Syntax :
  + **SELECT column1, column2, ...  
    FROM table\_name  
    WHERE NOT condition;**

1. **The SQL NULL and NOT NULL Values**

* It is not possible to test for NULL values with comparison operators, such as =, <, or <>. We will have to use the IS NULL and IS NOT NULL operators instead.
* IS NULL Syntax :
  + **SELECT column\_namesFROM table\_name  
    WHERE column\_name IS NULL;**
* IS NOT NULL Syntax :
  + **SELECT column\_namesFROM table\_name  
    WHERE column\_name IS NOT NULL;**

1. ***The SQL ORDER BY Keyword***
   * The ORDER BY keyword is used to sort the result-set in ascending or descending order.
   * ORDER BY Syntax :

* **SELECT column1, column2, ...  
  FROM table\_name  
  ORDER BY column1, column2, ... ASC|DESC;**

1. ***The SQL INSERT INTO Statement***
   * The INSERT INTO statement is used to insert new records in a table.
   * INSERT INTO Syntax (Specify both the column names and the values to be inserted) :
     + **INSERT INTO table\_name (column1, column2, column3, ...)  
       VALUES (value1, value2, value3, ...);**

* INSERT INTO Syntax (If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.) :
  + **INSERT INTO table\_name  
    VALUES (value1, value2, value3, ...);**

1. ***The SQL UPDATE Statement***
   * The UPDATE statement is used to modify the existing records in a table.
   * UPDATE Syntax :
     + **UPDATE table\_name  
       SET column1 = value1, column2 = value2, ...  
       WHERE condition;**
   * **Note:** Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!
2. ***The SQL DELETE Statement***

* The DELETE statement is used to delete existing records in a table.
* DELETE Syntax :
  + **DELETE FROM table\_name WHERE condition;**
* **Note:** Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

1. ***The SQL MIN() and MAX() Functions***

* The MIN() function returns the smallest value of the selected column.
* The MAX() function returns the largest value of the selected column.
* MIN() Syntax :
  + **SELECT MIN(column\_name)  
    FROM table\_name  
    WHERE condition;**
* MAX() Syntax :
  + **SELECT MAX(column\_name)  
    FROM table\_name  
    WHERE condition;**

1. ***The SQL COUNT(), AVG() and SUM() Functions***

* The COUNT() function returns the number of rows that matches a specified criterion.
* The AVG() function returns the average value of a numeric column.
* The SUM() function returns the total sum of a numeric column.
* COUNT() Syntax :
  + **SELECT COUNT(column\_name)   
    FROM table\_name  
    WHERE condition;**
* AVG() Syntax :
  + **SELECT AVG(column\_name) AS AVG\_VALUE  
    FROM table\_name  
    WHERE condition;**
* SUM() Syntax :
* **SELECT SUM(column\_name) AS TOTAL\_VALUE  
  FROM table\_name  
  WHERE condition;**

1. ***The SQL LIKE Operator***

* The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
* There are two **wildcards** often used in conjunction with the LIKE operator:
  + The **percent sign (%)** represents zero, one, or multiple characters
  + The **underscore sign (\_)** represents one, single character
* LIKE Syntax :
  + **SELECT column1, column2, ...  
    FROM table\_name  
    WHERE columnN LIKE pattern;**
* Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

|  |  |
| --- | --- |
| ***LIKE OPERATOR*** | ***Description*** |
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_%' | Finds any values that start with "a" and are at least 2 characters in length |
| WHERE CustomerName LIKE 'a\_\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

1. ***The SQL IN Operator***
   * The IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.
   * IN Syntax :

* **SELECT column\_name(s)  
  FROM table\_name  
  WHERE column\_name IN (value1, value2, ...);**
* **SELECT column\_name(s)  
  FROM table\_name  
  WHERE column\_name IN (*SELECT* STATEMENT);**

1. ***The SQL BETWEEN Operator***
   * The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.
   * BETWEEN Syntax :

* **SELECT column\_name(s)  
  FROM table\_name  
  WHERE column\_name BETWEEN value1 AND value2;**

1. ***SQL Aliases***
   * SQL aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable. An alias only exists for the duration of that query. An alias is created with the AS keyword.
   * Alias Column Syntax :
     + **SELECT column\_name AS alias\_name  
       FROM table\_name;**
   * Alias Table Syntax :
     + **SELECT column\_name(s)  
       FROM table\_name AS alias\_name;**