SQL (Structured Query Language)

1. **Basic**

Some of most Important SQL Commands:

|  |  |
| --- | --- |
| 1st statement |  |
| SELECT  UPDATE  DELETE  INSERT | Search / Find records  Modify existing records  Delete records  Add new records |
| CREATE  ALTER  DROP | DATABASE  TABLE  INDEX  VIEW |

1. **SELECT**

Select Syntax: To search on table or between multiple tables  
  
SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)

HAVING conditionORDER BY column\_name(s);

SELECT CITY.NAME

FROM CITY

WHERE CITY.POPULATION > 120000 AND CITY.COUNTRYCODE = 'USA';

|  |  |
| --- | --- |
| SELECT \* FROM table\_name; | Select all |
| SELECT Name, Age FROM table\_name; | Select 2 columns |
| SELECT DISTINCT Country  FROM table\_name; | No duplicate |
| SELECT column1, column2, ... FROM table\_name WHERE condition; | Condition |
| SELECT column1, column2, ... FROM table\_name WHERE Address IS NULL; | IS NOT NULL – Ngược lại |
| SELECT column1, column2, ... FROM table\_name ORDER BY Country; | Sort by a column |
| ORDER BY Country ASC; | Sort by a field with increasing order  DESC – decreasing order |
| SELECT TOP 3 \* FROM Customers;  SELECT \* FROM Customers LIMIT 3;  SELECT TOP 50 PERCENT \* FROM Customers; | Get the first 3 records return  Get 50% records return |
| SELECT CITY, LENGTH(CITY)  FROM STATION  ORDER BY Length(CITY) DESC, CITY LIMIT 1; | Find the city with max length, use LIMIT 1 to return 1 answer |
| SELECT MIN(column\_name) FROM table\_name WHERE condition; | Find MIN() / MAX() value in a column   * Return: MIN(Price) 2.5 |
| SELECT COUNT(ProductID) FROM Products;  SELECT AVG(Price) FROM Products;  SELECT SUM(Quantity) FROM OrderDetails  WHERE Price = 18;  SELECT LENGTH(Name) FROM Products;  SELECT (COUNT(CITY) - COUNT(DISTINCT CITY))  FROM STATION; | COUNT, AVG, SUM, LENGTH  Find difference between total city & unique city |
| SELECT CONCAT(First Name, ' ', Last Name)  SELECT CONCAT('There are a total of ', count(ID), ' items.')  SELECT LOWER(Occupation) | Maro Chang  Write string as the output  Lowercase, uppercase |

1. **WHERE – Condition**

WHERE clause is used to filter records, to extract only records that fulfill a specified condition.

**Compare Value – IN - BETWEEN**

|  |  |
| --- | --- |
| SELECT \* FROM Customers WHERE Country ='Mexico';  WHERE Price % 2 = 0; | Even number |
| =  >  <  >=  <=  <>  IS NULL  IS NOT NULL | Compare |
| WHERE Country='Germany' AND City='Berlin';  WHERE City='Berlin' OR City='München';  WHERE NOT Country='Germany' AND NOT Country='USA'; | AND / OR / NOT |
| SELECT \* FROM Customers WHERE Country IN ('Germany', 'France', 'UK');  SELECT \* FROM Customers WHERE Country NOT IN ('Germany', 'France', 'UK');  SELECT \* FROM Customers WHERE Country IN (SELECT Country FROM Suppliers); | IN(value1, value2) shorthand for multiple OR |
| SELECT \* FROM Products WHERE Price BETWEEN 10 AND 20;  SELECT \* FROM Products WHERE NAME NOT BETWEEN 'ABC' AND 'LMN';  SELECT \* FROM Products WHERE Price BETWEEN 10 AND 20 AND CategoryID NOT IN (1,2,3);  SELECT \* FROM Orders WHERE OrderDate  BETWEEN '1996-07-01' AND '1996-07-31'; | BETWEEN Range  BETWEEN TEXT  BETWEEN + IN  BETWEEN DATE |

**LIKE – Pattern**

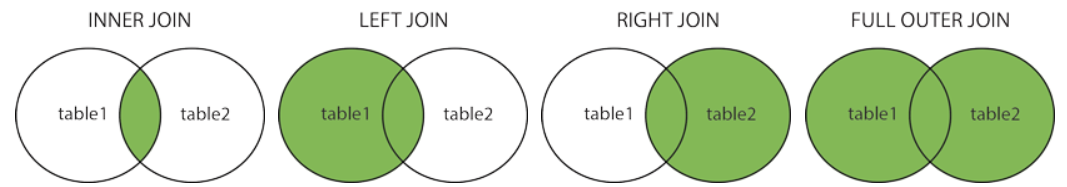
|  |  |
| --- | --- |
| WHERE columnN LIKE pattern;  SELECT \* FROM Customers WHERE CustomerName LIKE 'a%';  REGEXP '^[aeiou]';  REGEXP '[aeiou]$';  LIKE 'a%'  LIKE '%a'  LIKE '%or%'  LIKE '\_r%'  LIKE 'a\_%'  LIKE 'a%o'  WHERE City LIKE '[a-c]%';  WHERE ID LIKE '2#5'; | LIKE Pattern  WILDCARDS: Special characters % \_ ? [] ! - # ^  Start with vowels  End with vowels  Start with 'a'  End with 'a'  Contain 'or'  Have 'r' in second position  Start with 'a' and have atleast 2 characters  Start with 'a' and end with 'o'  Start with a, b, or c  -: is the range  2#5 finds 205, 215, 225, 235, 245, 255, 265, 275, 285, and 295  #: numeric |

**CASE – when – then - else**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CASE     WHEN condition1 THEN result1     WHEN condition2 THEN result2     WHEN conditionN THEN resultN     ELSE result END;  //----------------------------------------  SELECT OrderID, Score, CASE     WHEN Score > 90 THEN 'Grade A'     WHEN Score = 80 THEN 'Grade B'     ELSE 'Failed' END AS Grade FROM ScoreDetail;  //----------------------------------------  SELECT CustomerName, City, Country FROM Customers ORDER BY (CASE     WHEN City IS NULL THEN Country     ELSE City END); | Create a new column based on a previous col’s value   |  |  |  | | --- | --- | --- | | ID | Score | Grade | | 12 | 80 | B | | 31 | 92 | A | | 88 | 50 | Failed |   Sort with Case |

1. **JOIN**

* **(INNER) JOIN:** Returns records that have **matching values in both tables**
* **LEFT (OUTER) JOIN**: Returns all records from the **left table**, and the matched records from the right table
* **RIGHT (OUTER) JOIN**: Returns all records from the **right table**, and the matched records from the left table
* **FULL (OUTER) JOIN**: Returns all records when there is a **match** in either left or right table



|  |
| --- |
| SELECT column\_name(s) FROM table1 INNER JOIN table2ON table1.column\_name = table2.column\_name; |
| SELECT column\_name(s) FROM table1 LEFT JOIN table2ON table1.column\_name = table2.column\_name; |
| SELECT column\_name(s) FROM table1 RIGHT JOIN table2ON table1.column\_name = table2.column\_name; |
| SELECT column\_name(s) FROM table1 FULL OUTER JOIN table2ON table1.column\_name = table2.column\_nameWHERE condition; |

**UNION**

|  |  |
| --- | --- |
| SELECT column\_name(s) FROM table1 UNION SELECT column\_name(s) FROM table2;  SELECT City FROM Customers UNION SELECT City FROM Suppliers ORDER BY City;  SELECT City, Country FROM Customers WHERE Country='Germany' UNION ALL SELECT City, Country FROM Suppliers WHERE Country='Germany' ORDER BY City; | **combine** the result-set of **two or more SELECT statements.**  Every SELECT statement within UNION must have the same number of columns, and data types, same order. |

1. **GROUP BY - HAVING**

|  |  |
| --- | --- |
| SELECT column\_name(s) FROM table\_name WHERE condition GROUP BY column\_name(s)ORDER BY column\_name(s);  SELECT COUNT(StudentID), Grade FROM Student GROUP BY Grade; | GROUP BY groups by the same value. Use with COUNT(), MAX(), MIN(), SUM(), AVG()  Example: group student by Grade A, B, C D  Đếm số học sinh theo điểm số: 20 – Grade A 35 – Grade B  18 – Grade C |
| SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country HAVING COUNT(CustomerID) > 5; | Advance version of WHERE   * Groups customers by country, select countries that have more than 5 customers. * Use HAVING with COUNT(), MAX(), MIN(), SUM(), AVG() |

1. **UPDATE**

UPDATE Syntax: To update existing records in a table

|  |  |
| --- | --- |
| UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE condition; |  |
| UPDATE Customers SET ContactName='Juan'; | **The whole column will be updated** if we don’t use WHERE |
| UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1; |  |

1. **DELETE**

DELETE Syntax: To delete existing records in a table

|  |  |
| --- | --- |
| DELETE FROM table\_name WHERE condition; |  |
| DELETE FROM Customers  WHERE CustomerName='Alfreds Futterkiste'; |  |
| DELETE FROM Customers; | Delete **all records** |

1. **CREAT - DROP – BACKUP DB**

|  |  |
| --- | --- |
| CREATE DATABASE databasename;  DROP DATABASE databasename;  BACKUP DATABASE databasename TO DISK = 'filepath'; | Create DB  Delere DB  Backup DB |

1. **CRUP TABLE**

|  |  |
| --- | --- |
| CREATE TABLE table\_name (     column1 datatype,     column2 datatype,    .... );  DROP TABLE table\_name;  ALTER TABLE Customers ADD Email varchar(255);  ALTER TABLE Customers DROP COLUMN Email;  ALTER TABLE Customers RENAME COLUMN Email to Email Address; | Create Table  Delete Table  Add new column to table  Delete column from table  Rename column |
| CREATE TABLE Customer (     ID int NOT NULL UNIQUE PRIMARY KEY,     LastName varchar(255) NOT NULL,      FirstName varchar(255),     Age int,     City varchar(255) DEFAULT 'Sydney' ); | [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - cannot have a NULL value  UNIQUE – no duplicate values  DEFAULT – default value |

1. **VIEW**
2. **A**