**More SQL commands**

**SELECT … WHERE**

This statement does select based on condition defined after the *WHERE* clause.

Syntax:

SELECT col1, col2, … FROM table\_name

WHERE condition;

Example:  
SELECT hackerName, hackerSkill FROM hackers

WHERE hackerID=9001;

**SELECT … DISTINCT**

The DISTINCT clause is used to remove duplicates in the results returned from the SELECT statement.

Syntax:

SELECT DISTINCT col1, col2, … FROM table\_name

WHERE condition;

Example:

SELECT DISTINCT hackerAge, hackerSkill FROM hackers

WHERE hackerName=’Pablo’;

**SELECT … WHERE … IN**

The WHERE clause may include the IN keyword, which specifies a condition where a column value must **at least include** one of the listed items in the (…).

Syntax:

SELECT col1, col2, … FROM table\_name

WHERE col3 IN (value1, value2);

Example:

SELECT hackerName, hackerSkill FROM hackers

WHERE hackerID in (9001, 1337, 80);

**SELECT … LIKE**

The LIKE clause is used in **conjunction** with the WHERE clause, to search for **patterns in the column** specified in the WHERE clause.

Syntax:

SELECT col1, col2, … FROM table\_name

WHERE col3 LIKE ‘pattern’;

Example:

SELECT hackerID, hackerSkill FROM hackers

WHERE hackerName LIKE ‘ominousS%’;

// % means zero, one or multiple characters

\_ means a single character.

**SELECT … ORDER BY**

Used to sort results either in **ascending (default)** or **descending order.**

Syntax:  
SELECT \* FROM table\_name

ORDER BY col1 [ASC|DESC] col2 ORDER BY col2 [ASC|DESC];

Example:

SELECT \* FROM hackers

ORDER BY hackerName;

// This will sort the results in **alphabetical** **order** based on the hackerName’s column.

**SELECT … GROUP BY**

The GROUP BY clause will **group rows** into **separate** groups based on the **unique values** of the **specified** columns. This clause is typically used in conjunction with **aggregate functions** to perform calculations on **each group**.

**Aggregate functions**

* SUM – **sums** the values in the collection
* AVG – calculates the **average** of the values in the collection.
* COUNT – counts the **number** of elements in the collection.
* MIN – retrieves the **minimum** value in the collection.
* MAX - retrieves the **maximum** value in the collection.

Syntax:

SELECT col1, col2, ..., aggregate\_function(col) AS new\_col1

FROM table

GROUP BY col1, col2, ...

Examples:

SELECT hackerAge, hackerSkill, SUM(hackerPorts) AS PortQuantity

FROM hackers

GROUP BY hackerAge, hackerSkill;

// This query will group all the rows into separate groups for columns based on the unique combinations in the **hackerAge** and **hackerSkill** columns, and will calculate the sum of the columns hackerPorts present in every group.

As a result every row will have a 3 columns, hackerAge, hackerSkill, PortQuantity.

SELECT MIN(amount) AS min\_amt

MAX(amount) AS max\_amt

FROM loan;

**HAVING Clause**

HAVING clause specifies conditions **applying to the groups** of GROUP BY query rather than **individual rows**, **filtering out** results returned from the GROUP BY clause should be shown.

Syntax:

SELECT col, FROM table\_name WHERE condition GROUP BY col HAVING condition ORDER BY col;

Example:

SELECT hackerAge, hackerSkill, SUM(hackerPorts) AS PortQuantity

FROM hackers

GROUP BY hackerAge, hackerSkill

HAVING SUM(hackerPorts) > 9001;

// HAVING clause must **follow** the **GROUP BY** clause and also **precede** the **ORDER BY** clause **if** **used**.

**ALIAS**

ALIAS clause is used to **temporary change** to name of a table/column just for the purpose of the SQL query, the **actual** table/column name **will no**t change after the query has executed.

Syntax:  
SELECT col1 AS new\_col

FROM table\_name;

SELECT col1

FROM table\_name AS new\_table\_name;

Example:  
SELECT hackerID AS hD

FROM hackers

WHERE hD LIKE ‘900\*’;

//ALIAS are typically used to make the query more readable or to make references to shorter object names