SQL -Structured Query Languages

-storing, manipulating and retrieving data

-Basically info from multiple **related** data bases to give out a specific query.

-A Relational Database Management Systems (**RDBMS**): MySQL, MS, Access, Oracle, postgres ect. -Dif systems can have slightly different language.

-Data base- collection of table

-Tables – rows(Record)- observations of the samples; and columns(Field) characteristics about observations

Data Types-

*-Columns must be same data type*

-String – combo or letters (input it with quotes or will not work)

-Integer- whole number

-Floating point- decimal numbers

- Boolean- True and False

Writing SQL statements (query)

* Not case sensitive. Only MySQL- type table names as they exist in databases
* Keywords cant be split across lines or abbreviated.
* Keywords usually in uppercase – looks nice.
* Clauses usually in sep lines for readability and ease of editing
* Tabs and indents to enhance readably
* Semicolon to end query

Ex:

SELECT \*

FROM (table) Jobs

WHERE Min\_salary > 1000

ORDER BY Job\_title;

Keywords and Clauses- used to find specific data

**DISTINCT**- Brings only distinct values in table. (no dups)

**WHERE**- Filters data from Columns

* Can be combined with **AND**(All need to be TRUE)**, OR**( One needs to be TRUE)**, NOT**(NOT TRUE) operators.
* If using 2 different operators in one query, use ( ) to separate the statements.

**ORDER BY**

* **ASC**- Ascending order. 1,2,3
* **DESC**- Descending order. 3,2,1

TIP: If you type out multiple SELECT values, you can use numbers later on for the position.

SELECT Table\_ID, Table\_date, Table\_item

FROM random

ORDER BY 2 Asc;

* “2” will be Table\_date

**IS NULL** -field with no value

-Use **IS NULL** in **WHERE** clause; **IS NOT NULL**- gives you everything with a value

SELECT Min\_salary

FROM (table) Jobs

WHERE Min\_salary **IS NULL**

**ROWNUM**- in oracle

**LIMIT-** In other SQL systems

-Both specify the number of records to return

SELECT \*

FROM countries

WHERE ROWNUM < 6; *OR LIMIT 6*

**MIN** -returns the smallest value of the selected field

SELECT MIN(salary)

FROM employees;

**MAX-** returns the largest value of the selected field.

SELECT MAX(salary)

FROM employees;

**COUNT**- returns the number of records that matches a specified criteria.

SELECT COUNT(\*)

FROM employees;

-For unique, use COUNT(DISTINCT)

SELECT COUNT(DISTINCT first\_name)

FROM employees;

**AVG**- returns the average value of a numeric field (number)

SELECT AVG(salary)

FROM employees;

**SUM**- total sum of a numeric field (number)- must specify what you’re summing.

SELECT SUM(salary)

FROM employees;

Ex: Report:

* **Calculate number of employees, minimum, maximum and average salary from employees table**

SELECT COUNT(employee\_id), MAX(salary), MIN(salary), AVG(salary)

FROM employees;

**ROUND-** returns a numeric value, rounded to the specified length or precision.

SELECT ROUND(AVG(salary), 2)

FROM employees;

-“2” being the amount of decimal points

Arithmetic Operations

* **Calculate the sum of maximum and minimum salary in employees table.**

SELECT MAX(salary)+MIN(salary)

FROM employees;

* **Calculate the average salary in employees table**

SELECT SUM(salary)/COUNT(employee\_id)

FROM employees;

* **Calculate 10 times average salary of employees in employee table**

SELECT AVG(salary)\*10

FROM employees;

**Like -**Used with **WHERE**

**-** search for a specified pattern/string/number in a field.

- “%” - rep 0,1, multi numbers

- “\_”- rep 1 character

|  |  |
| --- | --- |
| WHERE first\_name LIKE 'a%' | Finds any names that start with "a" |
| WHERE first\_name LIKE '%a' | Finds any names that end with "a" |
| WHERE first\_name LIKE '%ar%' | Finds any names that have "a" in any position |
| WHERE first\_name LIKE '\_a%' | Finds any names that have "a" in the second position |
| WHERE first\_name LIKE 'a\_%' | Finds any names that start with "a" and are at least 2 characters in length |
| WHERE first\_name LIKE 'a\_\_%' | Finds any names that start with "a" and are at least 3 characters in length |
| WHERE first\_name LIKE 'a%b' | Finds any names that start with "a" and ends with "b" |

**Select all employee records whose first name start with ‘S’.**

SELECT \*

FROM employees

WHERE first\_name LIKE 'S%';

* **Select all employee records whose first name contains with ‘er’.**

SELECT \*

FROM employees

WHERE first\_name LIKE '%er%';

**IN-** allows you to specify multiple values in **WHERE** clause. It  is a shorthand for multiple **OR** conditions.

* Can use **NOT IN**
* **Find all the records where the employees first names are ‘Peter’ or ‘Steven’.**

SELECT \*

FROM employees

WHERE first\_name IN ('Peter', 'Steven');

**BETWEEN**- selects values within a specified range. -Inclusive of all data type. **NOT BETWEEN-** for op

* **Find all the employee records where employees salaries are between 11000 and 13000**

SELECT \*

FROM employees

WHERE salary BETWEEN 11000 AND 13000;

Aliases -to make column names more readable (only for fields)

**-AS**

-Only exists for the duration of the query

SELECT country\_id AS cid, country\_name, region\_id

FROM countries

SELECT C.\*

FROM countries  C

SELECT C.country\_id AS cid, C.country\_name, C.region\_id

FROM countries  C

Comments- Single line comments start with –

* Multi line starts with /\* and ends with \*/

SELECT country\_id--, country\_name, region\_id

FROM countries;

SELECT country\_id

/\*

This is my second comment

And I love it.

\*/

FROM countries;

Concatenation – It concatenates columns or character strings to other columns. It is represented by two vertical bars ( **||** ) - *(Shift \\)- above “enter”*.

SELECT first\_name || last\_name AS full\_name

FROM employees;

-**put ‘ ‘ for space b/w names**

SELECT first\_name || '  ' || last\_name AS full\_name

FROM employees;

SELECT first\_name || '  ' || last\_name AS "Full Name"

FROM employees;

**JOIN**: clause is used to combine records from two or more tables in a database

Types

1. **INNER JOIN** returns records that have matching values in both tables.

SELECT table1.column, table2.column

FROM table1

INNER JOIN table2

ON table1.column\_name = table2.column\_name;

Ex:

SELECT employees.first\_name, employees.employee\_id, job\_history.start\_date, job\_history.end\_date

FROM employees

INNER JOIN job\_history

ON employees.employee\_id = job\_history.employee\_id;

1. **LEFT JOIN** returns all records from the left table, and the matched records from the right table.

SELECT table1.column, table2.column

FROM table1

LEFT JOIN table2

ON table1.column\_name = table2.column\_name;

Ex:

SELECT E.first\_name, E.employee\_id, J.start\_date, J.end\_date

FROM employees  E

LEFT JOIN job\_history  J

ON E.employee\_id = J.employee\_id;