**SQL Notes**

What is SQL?

* SQL (pronounced “ess-que-el”) stands for Structured Query Language
* SQL is used to communicate with a database.
* It is the standard language for relational database management systems.
* SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL server, Access, Ingres, etc.
* The standard SQL commands such as “Select”, “Insert”, “Update”, “Delete”, “Create”, and “Drop” can be used to accomplish almost everything that one needs to do with a database.
* SQL programming can be used to perform multiple actions on data such as :
  + - Querying
    - Inserting
    - Updating
    - Deleting
    - Extracting etc.

Table Basics

A relational database system contains one or more objects called tables. The data or information for the database are stored in these tables. Tables are uniquely identified by their names and are comprised of columns and rows. Columns contain the column name, data type, and any other attributes for the column. Rows contain the records or data for the columns. Here is a sample table called “Employee”

|  |  |  |  |
| --- | --- | --- | --- |
| Employee Name | Employee Id | Manager Name | Division |
| Than Than Swe | 901 | Maung Maung | 1 |
| Hla Hla Win | 902 | Maung Maung | 1 |
| Tun Tun | 903 | Su Su | 1 |
| Su Su | 904 | Soe Soe | 1 |
| San Thidar | 905 |  |  |

SELECTING THE DATA

Selecting statement is used to query the database and retrieve selected data that matches the criteria.

Here’s a format of a select query:

Select column1, column2 from tablename where [condition]; // where condition is optional

The where clause (optional) specifies which data values or rows will be returned or displayed, based on the criteria described after the keywords where:

* = equal
* > greater than
* < less than
* >= greater than
* <= less than
* <> not equal
* LIKE like operator

The LIKE pattern matching operator can also be used in the conditional selection of the where clause. Like is a very powerful operator that allows you to select only rows that are “like” what you specify. The percent sign “%” can be used as a wildcard to match any possible character that might appear before or after the characters specified.

This SQL statement will match any first names that start with ‘Er’. Strings must be in single quotes. Or you can specify:

For example:

Select first, last ,city from emp\_info where first LIKE ‘Er%’;

This statement will match any last names that end in a ‘s’.

For example:

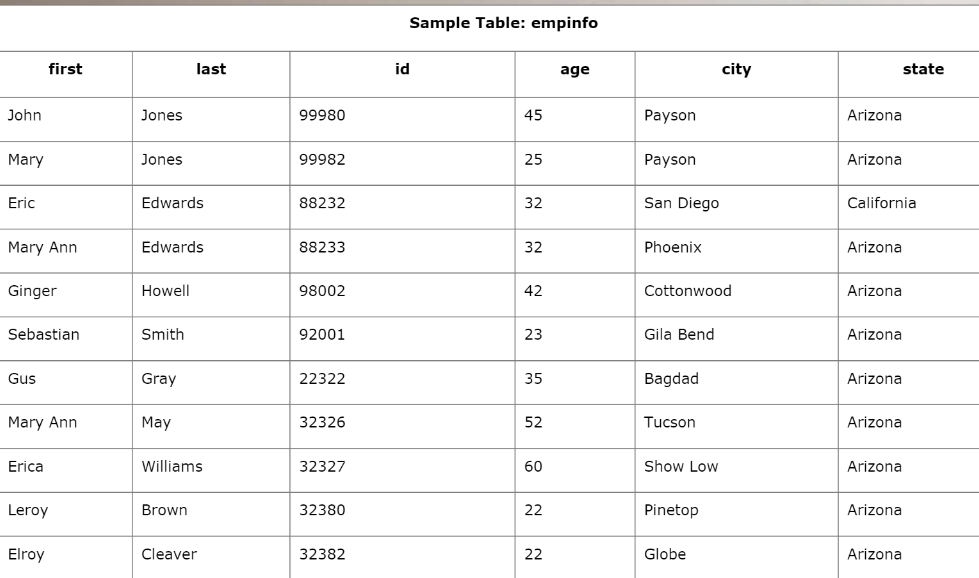
Select first, last from emp\_info where last LIKE ‘%’;

This will only select rows where the first name equals ‘Eric’ exactly

For example:

Select \* from emp\_info where first =’Eric”;

Table 1:



Enter the following sample select statements in the SQL interpreter form at the bottom of this page. Before you press “submit”, write down your expected results. Press “submit” and compare the results.

Select first, last, city from emp\_info;

Select last, city, age from emp\_info where age>30;

Select first, last, city, state from emp\_info where first LIKE ‘J%’;

Select \* from emp\_info;

Select first, last from emp\_info where last LIKE ‘%s’;

Select first, last, age from emp\_info where last LIKE ‘%illia’;

Select \* from emp\_info where first =’Eric’;

Creating tables

The create table statement is used to create a new table. Here is the format of a simple create table statement:

Create table “tablename”

(“column1” “data type”,

“column2” “data type”,

“column3” “data type”));

Format of create table if you were to use optional constraints:

Create table “tablename”

(“column1” “data type” [constraint],

“column2” “data type” [constraint],

“column3” “data type” [constraint]); [ ]= optional

Note: You may have as many columns as you’d like, and the constraints are optional.

Example:

Create table employee (

first varchar (15),

last varchar (20),

age number (3),

address varchar (30),

city varchar (20),

state varchar (20));

To create a new table,

1. Enter the keywords create table followed by the table name,
2. Followed by an open parenthesis,
3. Followed by the first column name,
4. Followed by the data type for that column,
5. Followed by any optional constraints, and followed by a closing parenthesis.

The table and column names:

1. Must start with a letter
2. And can be followed by letters, numbers, or underscores
3. Not to exceed a total of 30 characters in length.

Here are the most common data types:

1. Char (size): fixed-length character string. Size is specified in parenthesis, max 255 bytes.
2. Varchar (size): variable length character string. Max size is specified in parenthesis.
3. Number (size): Number value with a max number of column digits specified in parenthesis.
4. Date: date value
5. Number (size,d): Number value with a maximum number of digits of “size” total, with a maximum number of “Digit” digits to the right of the decimal.

What are constraints?

When tables are created, it is common for one or more columns to have constraints associated with them. A constraint is basically a rule associated with a column that the data entered into that column must follow.

* For example, a “unique” constraint specifies that no two records can have the same value in a particular column. They must all be unique.
* The other two most popular constraints are “not null” which specifies that a column can’t be left blank, and “primary key”. They will be covered and supported in the future release of the Advanced SQL tutorial – that is, if “response” is good.

Create table exercise

You have just started a new company. It is time to hire some employees. You will need to create a table that will contain the following information about your new employees:

First name, last name, title, age, and salary.

After you create the table, you should receive a small form on the screen with the appropriate column names. If you are missing any columns, you need to double check your SQL statement and recreate the table. Once it’s created successfully, go to the “Insert” lesson.

Insert

The insert statement is used to insert or add a row of data into the table.

To insert records into a table, enter the key words insert into followed by the table name, followed by an open parenthesis, followed by a list of column names separated by commas, followed by a closing parenthesis, followed by the keyword values, followed by the list of values enclosed in parenthesis. The values that you enter will be held in the rows and they will match up with the column names that you specify.

Strings should be enclosed in single quotes, and numbers should not.

Insert statement exercises

It is time to insert data into your new employee table.

Your first three employees are the following:

Jonie Weber, Secretary, 28, 19500.00

Potsy Weber, Programmer, 32, 45300.00

Dirk Smith, Programmer, 45, 5020.000

UPDATE

The update statement is used to update or change records that match a specified criterion. This is accomplished by carefully constructing a where clause.

Update “tablename” set “columnname” = “newvalue”

[,”nextcolumn” =

“newvalue2”…]

Where “columnname”

Operator “value”

[and | or “column” Operator “value”];

[] = optional

Deleting records

The delete statement is used to delete records or rows from the table.

Example:

Delete from employee;

Note: if you leave off the where clauses, all records will be deleted.

Delete from employee where last name =’Mary’;

Delete from employee where first name =’Mike’ or first name =’Eric’;

Drop a table

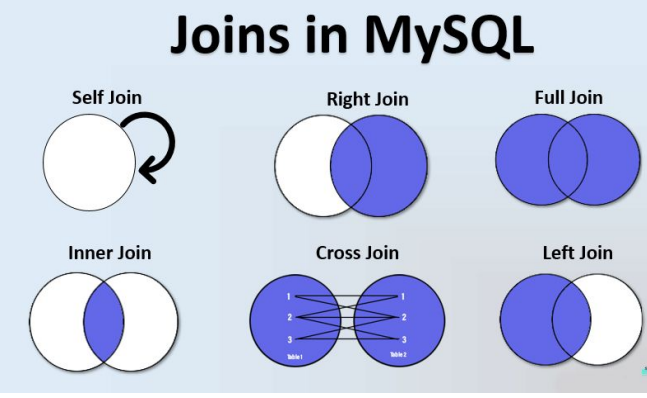
1. The drop table command is used to delete a table and all rows in the table.
2. To delete an entire table including all of its rows, issue the drop table command followed by the table name.
3. Drop table is different from deleting all of the records in the table.
4. Deleting all of the records in the table leaves the table including column and constraint information.
5. Dropping the table removes the table definition as well as all of its rows.

Sorting

1. In order to sort a table, we need to use the ORDER BY clause.
2. In order to get the results sorted based on certain columns, we need to use this.
3. Usage of asc or desc has to be defined to have the results in ascending or descending orders. Default value being asc (ascending).

JOINS

1. In most of the real-world problems, we might need data from multiple tables, that’s where Joins comes into picture.



JOINS Left Join

The left join keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

JOIN Right Join

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1).

JOIN2 – Inner Join

The INNER join keyword selects records that have matching values in both tables.

Aggregate Functions

Sometimes we examine and analyze data of varying magnitudes, hence we realize the need of grouping similar types of values together and loo them at as one bunch.

For example: Consider a table containing data consisting of the marks scored by students in their 12th board exams. While you would want to now how the students performed in all the subjects put together, it is equally important to see how they performed in each subject. You can gain even further insights if you group these students by state. Hence, it is imperative that you learn the usage of aggregate functions in your queries.

1. Groupby() 🡪 to aggregate values of a column C1 ‘grouped by’ a certain column C2.
2. Count() 🡪 Count the number of rows.
3. Min((), max() 🡪 finding the minimum and maximum values for a particular column.
   1. For example: select min(score) from employee;
4. Avg()🡪 Find the average

Ordering

If we want to display the retrieved records in a particular order, for example, in increasing order of income, joining date, alphabetical order, etc. This is commonly useful when you are making a report or presenting the data to someone else.

SELECT firstName from employees order by firstName asc limit 3;

HAVING Clause

You have already learnt how to filter individual values based on a given condition. But how do you do this on grouped values?

Suppose your manager asks you to count all the employees whose salaries are more than the average salary in that particular department.

Now, intuitively, you know that two aggregate functions would be used here, namely, count() and avg(). You decide to apply the 'where' condition on the average salary of the department, but to your surprise, the query fails. This is exactly what the having clause is for. The 'having' clause is typically used when you have to apply a filter condition on an 'aggregated value'. This is because the 'where' clause is applied before aggregation takes place, and thus, it is not useful when you want to apply a filter on an aggregated value.

Exercise:

Keywords in SQL

List the keywords below in order of their occurrence in a query:

1. Group by
2. Order by
3. Select
4. Where
5. From
6. Limit
7. Having

String Functions

Used to manipulate the string data and make it more understandable for analysis.

For example: amitabhbachchan, or Amitabh Bachchan, which one of

them is more readable, obviously the later one right.

concat → Concatenates two strings

substr/substring → Extracts a substring from a string

upper → Converts a string to uppercase

lower → Converts a string to lowercase

character\_length → Calculates the length of the string variable.

Date and Time Functions

Used to manipulate the date and time columns

For example: If you want to change the date format, or just want to

see the exact day, or so on.

datediff → Return the number of days between the two date values

date\_format → Format a date variable

day → Return the day of the month for a date.

quarter → Return the quarter of the year for a date.

and so on….

REGEX

So far you have already known about the wildcards like “LIKE” Operator, but in cases wildcards may fall short for some advanced use case, regular expressions come into picture.

Regex, or regular expressions, is a sequence of characters, used to search and locate specific sequences of characters that match a pattern.

Example 1: Match beginning of the name

Select \* from customer where email REGEXP’^w’

Example 2: Find the customers, which email address having a ‘z’, ‘v’ or ‘p’

Select \* from customer

WHERE email REGEXP “[zvp]”;

NESTED Queries

Generating insights from data, you may need to refer to these multiple tables in a query. There are two ways to deal with such types of queries. These include the following:

1. JOINS
2. Nested queries/ subqueries

A subquery is called an inner query while the query that contains the subquery is called an outer query. A subquery can be used anywhere that expression is used and must be closed in parentheses.

Views

Views are virtual tables that do not store any data of their own but display data stored in other tables.

Advantages:

1. Hide the complexity of data.
2. Act as aggregated tables.
3. If you are doing a user level access control, you can give an user access to a view without giving access to the tables behind it.
4. It can allow for massive performance improvements.

Disadvantages:

1. If done wrong, it can result in performance issues.
2. You may not be able to update the view, forcing you back to the original tables.