What is SQL?

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), 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. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, 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. This tutorial will provide you with the instruction on the basics of each of these commands as well as allow you to put them to practice using the SQL Interpreter.

SQL commands knowledge and all types of languages available in SQL like DDL, DML, DCL, TCL etc.

SQL consists of 4 types of supported languages internally called

- DDL (Data Definition Language)
  this type of language is responsible to create/change the definition of any object, in simple words "The language which can impact of definition of object known as DDL". Also we can say "The command which doesn't need COMMIT or ROLLBACK to complete the transaction known as DDL" It is also sub divided into various commands in SQL
  - CREATE COMMAND – to create objects in database
  - ALTER COMMAND – to alter/change the structure of objects in database
  - RENAME COMMAND – to rename the existing object to some other name
  - DROP COMMAND – to remove the object from database
  - TRUNCATE COMMAND – to remove the data completely including space allocated to that

- DML (Data Manipulation Language)
  this type of language is used to manage the data resides in object like Table etc. through predefined commands with in the same schema or outside the current schema. It also has various commands like
  - SELECT – to select the data from table
  - INSERT – to insert the rows into table
  - UPDATE – to update the existing records of the table
  - DELETE – to delete the existing record of table
  - MERGE – UPSERT operation like (Insert and Update), will understand this in detail later

- DCL (Data Control Language)
  this type of language is used to control the privileges given to the user(s). For example, if we want to give any kind of INSERT, UPDATE etc. privileges on one table to any user then this can be controlled through DCL. It consists the following
  - GRANT COMMAND – give permission to specified user OR to all users
  - REVOKE COMMAND – revoke permission form specified user OR form all users

- TCL (Transaction Control Language)
  this type of language is used to control the transactions done by user through DML operations. For example "If user A inserts some records in one table then to save those changes in database table can be done by TCL". It consists
  - COMMIT COMMAND – to save the work done by user in database
  - ROLLBACK COMMAND – to revoke the changes (UN-COMMITTED) data from database
  - SAVEPOINT COMMAND – it is just a watermark to identify the place which can be committed or not
  - SET TRANSACTION – change the transaction options like isolation level 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 "weather".

city, state, high, and low are the columns. The rows contain the data for this table:

Weather			
city	**state**	**high**	**low**
Phoenix	Arizona	105	90
Tucson	Arizona	101	92
Flagstaff	Arizona	88	69
San Diego	California	77	60
Albuquerque	New Mexico	80	72

Selecting Data

The **select** statement is used to query the database and retrieve selected data that match the criteria that you specify. Here is the format of a simple select statement:

```
select "column1"
  [,"column2",etc]
  from "tablename"
  [where "condition"];
  [] = optional
```

The column names that follow the select keyword determine which columns will be returned in the results. You can select as many column names that you'd like, or you can use a "*" to select all columns.

The table name that follows the keyword **from** specifies the table that will be queried to retrieve the desired results.

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

Conditional selections used in the **where** clause:

=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal to
LIKE	*See note below

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 wild card to match any possible character that might appear before or after the characters specified. For example:

```
select first, last, city
   from empinfo
   where first LIKE 'Er%';
```

This SQL statement will match any first names that start with 'Er'. **Strings must be in single quotes.**

Or you can specify,

```
select first, last
   from empinfo
   where last LIKE '%s';
```

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

```
select * from empinfo
   where first = 'Eric';
```

This will only select rows where the first name equals 'Eric' exactly.

Sample Table: empinfo					
first	**last**	**id**	**age**	**city**	**state**
John	Jones	99980	45	Payson	Arizona
Mary	Jones	99982	25	Payson	Arizona
Eric	Edwards	88232	32	San Diego	California
Mary Ann	Edwards	88233	32	Phoenix	Arizona
Ginger	Howell	98002	42	Cottonwood	Arizona
Sebastian	Smith	92001	23	Gila Bend	Arizona
Gus	Gray	22322	35	Bagdad	Arizona
Mary Ann	May	32326	52	Tucson	Arizona
Erica	Williams	32327	60	Show Low	Arizona
Leroy	Brown	32380	22	Pinetop	Arizona
Elroy	Cleaver	32382	22	Globe	Arizona

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 empinfo;
```

```
select last, city, age from empinfo
       where age > 30;

select first, last, city, state from empinfo
       where first LIKE 'J%';

select * from empinfo;

select first, last, from empinfo
       where last LIKE '%s';

select first, last, age from empinfo
       where last LIKE '%illia%';

select * from empinfo where first = 'Eric';
```

Select statement exercises

Enter select statements to:

1. Display the first name and age for everyone that's in the table.
2. Display the first name, last name, and city for everyone that's not from Payson.
3. Display all columns for everyone that is over 40 years old.
4. Display the first and last names for everyone whose last name ends in an "ay".
5. Display all columns for everyone whose first name equals "Mary".
6. Display all columns for everyone whose first name contains "Mary".

Creating Tables

Tables

1. Table names are in plural form, for example, persons, materials, addresses. If table name contains more than one word, they are separated with underscore in form {name1}_{name2}. Only the last one is in plural, for example person_addresses.
2. All tables have 3 or 4 character long aliases that are unique in a schema. Aliases are not directly used in name of table, but they are used to create column names. For example, persons - prs, materials - mat, addresses - adr.
3. Sometimes it is useful to distinguish some logical parts of an application. Therefore prefixes are used in table names. For example, sec_users, sec_roles, sec_rights all are related to security subsystem.

Columns (for tables)

1. All columns are in form {alias}_{colname}. For example prs_id, prs_name, prs_adr_id, adr_street_name. This guarantees that column names are unique in a schema, except denormalized columns from another table, which are populated using triggers or application logic.

2. All columns are in singular. If you think you need a column name in plural think twice whether it is the right design? Usually it means you are including multiple values in the same column and that should be avoided.
3. All tables have surrogate primary key column in form {alias}_id, which is the first column in the table. For example, prs_id, mat_id, adr_id.

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, enter the keywords **create table** followed by the table name, followed by an open parenthesis, followed by the first column name, followed by the data type for that column, followed by any optional constraints, and followed by a closing parenthesis. It is important to make sure you use an open parenthesis before the beginning table, and a closing parenthesis after the end of the last column definition. Make sure you seperate each column definition with a comma. All SQL statements should end with a ";".

The table and column names must start with a letter and can be followed by letters, numbers, or underscores - not to exceed a total of 30

characters in length. Do not use any SQL reserved keywords as names for tables or column names (such as "select", "create", "insert", etc).

Data types specify what the type of data can be for that particular column. If a column called "Last_Name", is to be used to hold names, then that particular column should have a "varchar" (variable-length character) data type.

Available data types in SQL

SQL provides some predefined data types through which we can store our data in tabular manner.

Data Type	Syntax	Explanation (if applicable)
NUMBER	NUMBER (p,s)	Where p is a precision value; s is a scale value. For example, NUMBER (6, 2) is a number that has 4 digits before the decimal and 2 digits after the decimal.
CHAR	CHAR(x)	Where x is the number of characters to store. This data type is space padded to fill the number of characters specified.
CHARACTER VARYING	VARCHAR2(x)	Where x is the number of characters to store. This data type does NOT space pad.
DATE	DATE	Stores year, month, and day values.
CLOB (Character Large Object)	CLOB	It is used to store bulky data and can be up to 2GB in size

Inserting into a Table

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 into "tablename"
 (first_column,...last_column)
  values (first_value,...last_value);
```

In the example below, the column name `first` will match up with the value `'Luke'`, and the column name `state` will match up with the value `'Georgia'`.

Example:

```
insert into employee
  (first, last, age, address, city, state)
  values ('Luke', 'Duke', 45, '2130 Boars Nest',
          'Hazard Co', 'Georgia');
```

Note: All strings should be enclosed between **single** quotes: `'string'`

Updating Records

The **update** statement is used to update or change records that match a specified criteria. 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
```

[The above example was line wrapped for better viewing on this Web page.]

Examples:

```
update phone_book
  set area_code = 623
  where prefix = 979;

update phone_book
  set last_name = 'Smith', prefix=555, suffix=9292
  where last_name = 'Jones';

update employee
  set age = age+1
  where first_name='Mary' and last_name='Williams';
```

Update statement exercises

After each update, issue a select statement to verify your changes.

1. Jonie Weber just got married to Bob Williams. She has requested that her last name be updated to Weber-Williams.
2. Dirk Smith's birthday is today, add 1 to his age.
3. All secretaries are now called "Administrative Assistant". Update all titles accordingly.
4. Everyone that's making under 30000 are to receive a 3500 a year raise.
5. Everyone that's making over 33500 are to receive a 4500 a year raise.
6. All "Programmer II" titles are now promoted to "Programmer III".
7. All "Programmer" titles are now promoted to "Programmer II".

Deleting Records

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

```
delete from "tablename"

where "columnname"
  OPERATOR "value"
[and|or "column"
  OPERATOR "value"];

[ ] = optional
```

[The above example was line wrapped for better viewing on this Web page.]

Examples:

```
delete from employee;
```

Note: if you leave off the where clause, **all records will be deleted!**

```
delete from employee
  where lastname = 'May';

delete from employee
  where firstname = 'Mike' or firstname = 'Eric';
```

To delete an entire record/row from a table, enter "`delete from`" followed by the table name, followed by the where `clause` which contains the conditions to delete. If you leave off the `where`clause, all records will be deleted.

Drop a Table

The **drop table** command is used to delete a table and all rows in the table.

To delete an entire table including all of its rows, issue the **drop table** command followed by the tablename. **drop table** is different from deleting all of the records in the table. Deleting all of the records in the table leaves the table including column and constraint information. Dropping the table removes the table definition as well as all of its rows.

```
drop table "tablename"
```

Example:

```
drop table myemployees_ts0211;
```

SELECT Statement with WHERE and FROM

The SELECT statement is used to query the database and retrieve selected data that match the criteria that you specify.

The SELECT statement has five main clauses to choose from, although, FROM is the only required clause. Each of the clauses have a vast selection of options, parameters, etc. The clauses will be listed below, but each of them will be covered in more detail later in the tutorial.

Here is the format of the SELECT statement:

```
SELECT [ALL | DISTINCT] column1[,column2] FROM table1[,table2] [WHERE
"conditions"] [GROUP BY "column-list"] [HAVING "conditions] [ORDER BY
"column-list" [ASC | DESC] ]
```

Example:

```
SELECT name, age, salary

FROM employee

WHERE age > 50;
```

SQL WHERE Clause

The WHERE clause is used to extract only those records that fulfill a specified criterion OR used to filter the records.

SQL WHERE SYNTAX

SELECT COL1, COL2 FROM CUSTOMER WHERE COL3 <<operator>> value;

Here <<operator>> could be {=, <, >, <>, IN, LIKE etc.}

Operators in the Where clause

The following operators can be used in the WHERE clause:

Operator	Description
=	Equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range can also work with numbers and string data type and date data type as well
LIKE	Search for a pattern (A% to match the data start with A / %A to match the data which ends with A / %A% to match the data wherever we have A in string)
IN	To specify multiple possible values for a column. For example SELECT * FROM TABLE_NAME WHERE ORDERNO IN (VALUE1,VALUE2……VALUE1000). We can only specify the maximum range of 1000 only

SQL ORDER BY Keyword

This is used to sort the order of data. It should be @ last in your SELECT statement. There are two types of ordering provided by Oracle. Ascending (abbreviation is ASC) and Descending (abbreviation is DESC). By default Oracle provides ASC order if nothing specified.

SQL ORDER BY syntax is

SELECT COL1, COL2 FROM CUSTOMER ORDER BY COL2 ASC/DESC;

ORDER BY can also be performed on several columns like

SELECT COL1, COL2 FROM CUSTOMER ORDER BY COL1 ASC, COL2 DESC;

Aliases

Alias are used to rename the query at run time. Means when query is in execute mode then it renames the actual table to another name.

1. Alias can be of table and for column as well.
2. We should keep alias very simple.
3. Alias should start with alphabet.

For example

SELECT * FROM CUSTOMER C

Here now "C" will denote the CUSTOMER table. We can also write the column name in SQL statement with alias also. For example

SELECT C.COL1, C.COL2 FROM CUSTOMER C WHERE C.COL3 = VALUE