Introduction to SQL

Lecture Aims

The aims of this Lecture are to

- Introduce SQL
- Practice writing SQL statements

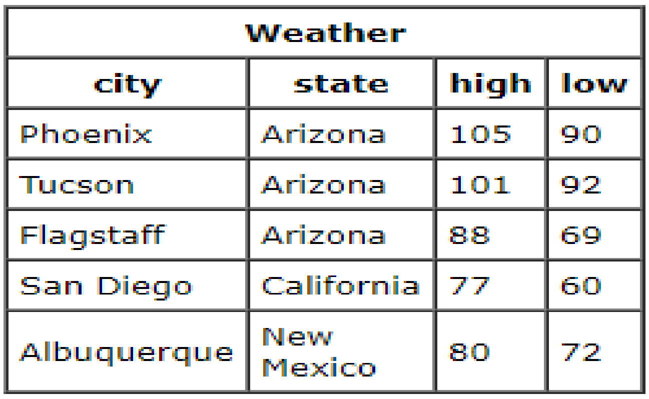
What is SQL?

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. 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.

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".

In the table 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 would 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

Examples of SQL statements that select information from the sample table above which is named empinfo for Employee Information

a:
```
select first, last, city
from empinfo;
```

first	last	city
John	Jones	Payson
Mary	Jones	Payson
Eric	Edwards	San Diego
Mary Ann	Edwards	Phoenix
Ginger	Howell	Cottonwood
Sebastian	Smith	Gila Bend
Gus	Gray	Bagdad
Mary Ann	May	Tucson
Erica	Williams	Show Low
Leroy	Brown	Pinetop
Elroy	Cleaver	Globe

b: Select the last name, city and age of all employees that are more than 30 years old

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

last	city	age
Jones	Payson	45
Edwards	San Diego	32
Edwards	Phoenix	32
Howell	Cottonwood	42
Gray	Bagdad	35
May	Tucson	52
Williams	Show Low	60

c: Return the first name, last name, city and state of all employees whose name starts with the letter J

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

first	last	city	state
John	Jones	Payson	Arizona

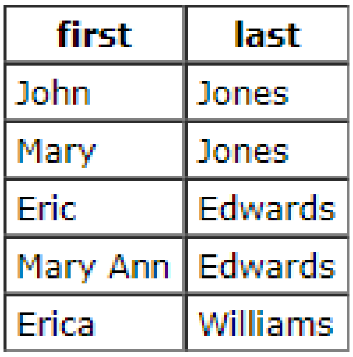
d: Return the whole table

```
select * from 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

e: Return the first and the last names of all employees whose surname ends with an s

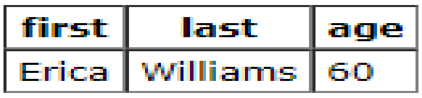
```
select first, last, from empinfo
       where last LIKE '%s';
```



first	last
John	Jones
Mary	Jones
Eric	Edwards
Mary Ann	Edwards
Erica	Williams

f: Return the first name, surname and age of all employees whose surname contains the letters illia

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



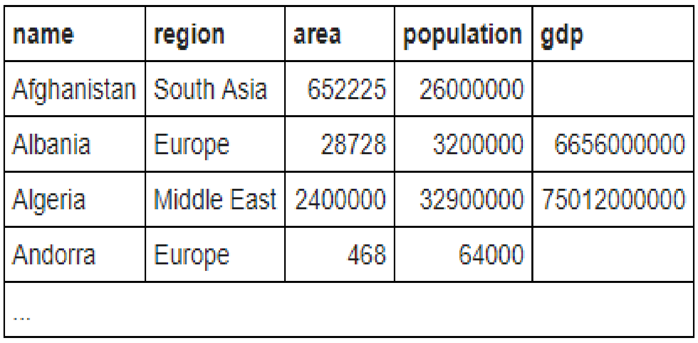
first	last	age
Erica	Williams	60

Exercise 1:

Write 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".

We next introduce **SQL IN** and **SQL BETWEEN** using a table named “world” which shows some geographical information of the countries in the world. The first few rows of the table world are:



name	region	area	population	gdp
Afghanistan	South Asia	652225	26000000	
Albania	Europe	28728	3200000	6656000000
Algeria	Middle East	2400000	32900000	75012000000
Andorra	Europe	468	64000	
...				

SQL IN

The word **IN** allows us to check if an item is in a list. The example below shows the name and population for the countries 'Brazil', 'Russia', 'India' and 'China'

```
SELECT name, population FROM world

WHERE name IN ('Brazil', 'Russia', 'India', 'China');
```

Exercise 3

Write an SQL statement that will show the name and the population for 'Sweden', 'Norway' and 'Denmark'

SQL BETWEEN

`BETWEEN` allows range checking (range specified is inclusive of boundary values). The example below shows countries with an area of 250,000-300,000 sq. km.

```
SELECT name, area FROM world

WHERE area BETWEEN 250000 AND 300000
```

Exercise 3

Modify the code above to show the country and the area for countries with an area between 200,000 and 250,000.

Exercise 4:

a: Go through the quiz in the following link: https://sqlzoo.net/wiki/SELECT_Quiz

b: Go through questions 1-7 of the quiz in the following link: https://sqlzoo.net/wiki/SELECT_from_WORLD_Tutorial

Aggregates

The functions SUM, COUNT, MAX and AVG are "aggregates", each may be applied to a numeric attribute resulting in a single row being returned by the query. An aggregate function takes many values and delivers just one value. For example the function SUM would aggregate the values 2, 4 and 5 to deliver the single value 11.

Distinct

By default the result of a SELECT may contain duplicate rows. We can remove these duplicates using the DISTINCT key word.

Examples

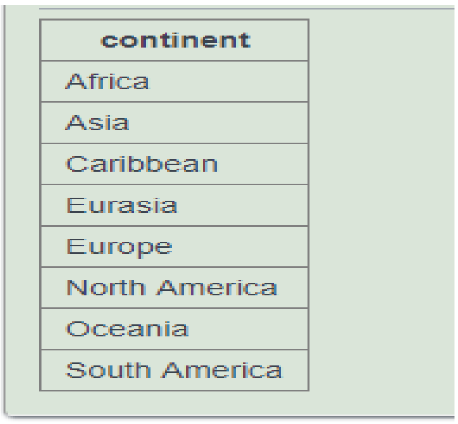
a: To show the total **population** of the world.

```
SELECT SUM(population)

 FROM world
```

b: A single search for listing all the continents

```
select distinct continent from world
```



continent
Africa
Asia
Caribbean
Eurasia
Europe
North America
Oceania
South America

c: Return the total gdp of Africa

```
SELECT sum(gdp)

 FROM world

WHERE continent='Africa';
```

d: Counting the number of countries with an area of at least 1000000 km^2

```
    SELECT count(name)
     FROM world
     WHERE area >= 1000000;
```

e: What is the total population of Estonia, Latvia and Lithuania?

```
SELECT SUM(population)
FROM world
WHERE name IN (‘Estonia’, ‘Latvia’, ‘Lithuania’);
```

Exercise 5

Go through the quiz in the following link: https://sqlzoo.net/wiki/SUM_and_COUNT_Quiz

Please note that most of the content in these notes is exact extract from the following references

References

[1] https://sqlzoo.net/wiki/SELECT_basics

[2] http://www.sqlcourse2.com/select2.html