Structured Query Language

Standard SQL commands: - Select - Insert - Update - Delete - Create - Drop

Relational db systems contain one or more tables. Each table is uniquely identified by their name and are composed of columns and rows.

Columns contain a column name, data type, and other attributes.

Rows contain the records/data for the cols.

Selecting Data

```
select "col1"
  [, "col2", etc]
from "tablename"
[where "condition"];
/* [] = optional */
```

`select`: Used to determine which cols. will be returned.

`from`: Specified the table that will be queried.

`where`: Optional clause which specifies the data values or rows that will be returned, based on the criteria `"condition"`. - Conditional selections - Equal `=` - Greater than `>` - Less than `<` - Greater than or equal to `>=` - Less than or equal to `<=` - Not equal to `<>` - Like - A pattern matching operator

Creating Tables

Creating tables:

```
create table "tablename"
("col1" "data type",
 "col2" "data type",
 "col3" "data type");
```

Creating tables w/ constraints:

```
create table "tablename"
("col1" "data type"
        [constraint],
 "col2" "data type"
        [constraint],
 "col3" "data type"
        [constraint]);
```

- Constraint: A rule associated w/ a column that the data entered into that column must follow.
  - Constraint examples: `unique`, `not null`, and `primary key`.
  - `unique`: No two records can have the same value.
  - `not null`: A col. cannot be left blank.
  - `primary key`: Defines a unique identification of each record.

Example for creating a table:

```
create table employee
(first    varchar(15),
 last     varchar(20),
 age      number(3),
 address  varchar(30),
```

```
 city     varchar(20),
 state    varchar(20));
```

Some common data types:

Data type	Description
`char(size)`	Fixed length char. str. Max 255 bytes
`varchar(size)`	Variable-length char. str.
`number(size)`	Num. value w/ max num. of col. digs.
`date`	Date value
`number(size,d)`	Num. val. w/ a max. num. digs., w/ max num. of `d` digs. to the right of the decimal

Inserting into Tables

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

Insert statement is used to insert a row of data into the table.

Updating Records

```
update "tablename"
set "columnname" =
    "newvalue"
 [,"nextcolumn" =
   "newvalue2"...]
where "columnname"
  OPERATOR "value"
 [and|or "column"
  OPERATOR "value"];
```

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';
```

Delete Records

```
delete from "tablename"

where "columnname"
  OPERATOR "value"
[and|or "column"
```

```
  OPERATOR "value"];

/* [ ] = optional */
```

Exaples:

```
delete from employee;

delete from employee
  where lastname = 'May';

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

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

Drop a table

Removes the table definition as well as all of it's records.

```
drop table "tablename"
```

Example:

```
drop table myemployees_ts0211;
```

Note: Diff. from deleting records since columns and column constraints are not kept.

Aggregate Functions

Used to compute against a returned column of numeric data from a `SELECT` statement.

MIN	returns the smallest value in a given column
MAX	returns the largest value in a given column
SUM	returns the sum of the numeric values in a given column
AVG	returns the average value of a given column
COUNT	returns the total number of values in a given column
COUNT(*)	returns the number of rows in a table

Group By clause

Gathers rows together that contain data in the specified column(s) and allows aggregate funcs. to be performed on the 1+ columns.

```
select column_i, aggrfunc(column_j)
from "list-of-tables"
group by "column-list";
```

Note: You must always use an aggregate function when using Group By.

Example:

```
/* Gather maximum salary for the people in each department */
select max(salary), dept
from employee
group by dept;
```

Having clause

Provides some conditions for which row should be selected

```
select column_i, aggrfunc(column_j)
from "list-of-tables"
group by "column-list"
having "condition";
```

Note: Having clause must always follow the group by clause.

Example:

```
/*
 * Selects avg(salary) for all employees in each dept and returns all
 * avg salaries which are greater than 20000.
 */
select avg(salary), dept
from employee
group by dept
having avg(salary) > 20000;
```

Order By clause

Optional clause which allows you to display your query in a sorted order.

```
select column_i, aggrfunc(column_j)
from "list-of-tables"
order by "column-list" [ASC|DESC]
/* [] = optional */
```

Example:

```
/*
 * Ordered employee info from Sales dept. in descending order based on
 * salary and age.
 */
select employee_id, dept, name, age, salary
from employee_info
where dept = 'Sales'
order by salary, age DESC;
```