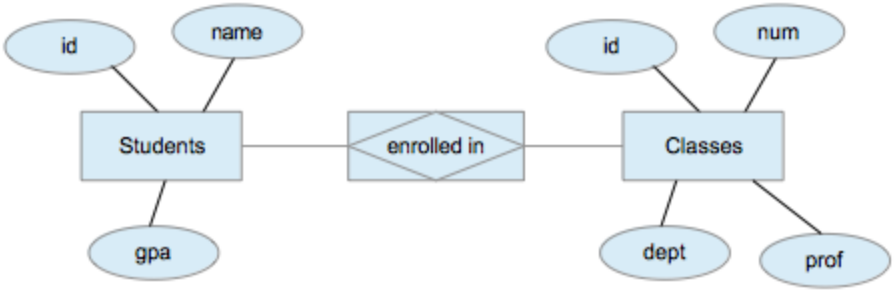
SQL

Structured Query Language (SQL) is the standard way of interacting with a relational database management system. SQL commands can typically be broken up into two major categories: data definition language command (DDL) and data manipulation language commands (DML). This handout will provide an overview of the major SQL commands. For more details, you should review the SQL language reference for the particular RDBMS that you're using.

Entity Relationship Diagram used for these examples:



DDL Commands:
This is a basic introduction to the fundamental DDL commands. Each of them has additional clauses that allow for the definition of more sophisticated database structures.

- *Create a new database in MySQL*
  - Example:
    ```
    CREATE database <dbname>;
    ```

- *Create a table*
  - The CREATE table[1] command takes the name of the new table and an ordered list of column names and each column's associated data type at a minimum.
  - Some potential data types include:
    - bit
    - tinyint, smallint, mediumint, integer
    - real, double, float, decimal
    - date, time, datetime, timestamp
    - varchar (<length>)
      - variable-length strings

[1]For the complete syntax of the CREATE table statement, see http://dev.mysql.com/doc/refman/5.5/en/create-table.html

        - <length> is the max length of the field
      - BLOB
        - Binary Large OBject
        - Treated as binary strings
      - Text
        - Treated as character strings

    - Trivial Example:
```
CREATE table students (
   id varchar(4),
   name varchar(20),
   gpa float);
```

    - You can indicate that a particular column may never contain a NULL value (lack of some particular value) by using the NOT NULL tag on a column definition.
```
 CREATE table students (
   id varchar(4) NOT NULL,
   name varchar(20),
   gpa float);
```

    - You can indicate that a field is part of the primary key of a table by using the PRIMARY KEY tag.
```
CREATE table students (
   id varchar(4) PRIMARY KEY,
   name varchar(20),
   gpa float);
```

- *Delete databases or tables*
  - DROP database <databasename>
  - DROP table <tablename>

DML Commands:

- *Insert data into a table*
  - Basic version accepts a table name and a list of values to insert into each column of the table, one value for each column. Order of values passed to the INSERT command must match the order of columns in the table.

    - Example:
```
INSERT into student values('123', 'John Doe', 3.2);
```

  - You can supply an ordered list of column names after the table name if you only want to insert into a subset of the columns. The values must appear in the same

order as the column list.

- Example:
```
INSERT into student(id, name) values ("9876", "No gpa
person");
```

- Data values used as part of an INSERT statement can be the result of a select statement. More on this later as needed.

- *Retrieve data from a table or tables*
  - The SELECT statement[2] allows us to retrieve data from one or more tables.
  - SELECT statements can get quite complex fairly fast, particularly as you add in joins and subqueries.
  - Basic version combines the selection and projection operators from relational algebra.
```
SELECT [DISTINCT|ALL] {* | [colExpr [AS newName], […]}
FROM  tableName [alias], […]
[WHERE condition]
```

    - DISTINCT will cause the return of the dataset with duplicates removed.
    - * is a wildcard representing all columns from the result set
      - If the select joins data from more than one table, you can use tableName.* to select all of the columns from a particular table.
    - colExpr [AS newName]
      - colExpr can be a calculated value
      - AS newName gives this colExpr a new name
      - [alias] gives the tableName a "short cut" name. Usually used to increase brevity of overall SELECT statement
    - condition of WHERE can combine multiple predicates
      - combined using logical connectives of AND, OR, and NOT.
      - 5 general types of predicates
        - comparison
```
SELECT *
FROM students
WHERE gpa >= 3.2;
```
        - range (inclusive)
```
SELECT *
FROM students
WHERE gpa BETWEEN 3.0 AND 3.5;
```
        - membership in a set
```
SELECT *
```

[2]The complete syntax of the SELECT statement can be found at http://dev.mysql.com/doc/refman/5.5/en/select.html
.

```
FROM classes
WHERE prof IN ('Doe', 'Smith');
```

- pattern match
  - more on this later
- to test whether a value is null or not.

```
SELECT *
FROM classes
WHERE gpa IS NULL;
```

- Sorting the output
  - use ORDER BY clause
  - can order by more than one column name
  - default order is ascending. Use DESC to indicate descending ordering.

```
SELECT *
FROM students
ORDER BY gpa DESC;

SELECT *
FROM students
ORDER BY gpa DESC, name ASC;
```

- Aggregate functions
  - perform a function on a column in the result set
  - Functions:
    - COUNT(<column>)
      - returns the number of values in <column>
    - SUM(<column>)
      - returns the sum of the values in <column>
    - AVG(<column>)
      - returns the average of the values in <column>
    - MIN(<column>)
      - returns the minimum value of the values in <column>
    - MAX(<column>)
      - returns the maximum value of the values in <column>
  - Examples:

```
SELECT count(*)
FROM students;

SELECT avg(gpa)
FROM students;
```

  - Except for count, each operate on a single field.
  - count, min and max can be applied to numeric and non-numeric typed columns
  - sum and avg can only be applied to numeric typed columns
  - distinct can be used in conjunction with sum, avg, and count. However, it

has no effect on min and max.
    - Some uses of aggregate functions require a GROUP BY clause. More on this later.
- Grouping results
    - accomplished with the GROUP BY clause
    - allows us to apply an aggregate function by group rather than on the entire result set
    - HAVING clause allows us to restrict which groups appear in the final result set
    - Examples:

```
SELECT prof, count(*)
FROM classes
GROUP BY prof;

SELECT prof, count(*)
FROM classes
GROUP BY prof
HAVING count(*) >= 2;
```

Some useful utility commands:

- `desc <tablename>;`
    - executing this command on a particular table will show you the structure of the table (column names, data types, etc.)
- `show tables;`
    - lists the tables that are in the current database
- `show databases;`
    - lists the databases in this MySQL instance
- `use <dbname>;`
    - access or move into a particular database
- `\. <fileName>`
    - will execute all of the SQL commands that are in scrip file <fileName>
- `tee <filename> & notee`
    - tee <filename> will keep a log file of session until notee is executed.