Introduction to SQL

CSC 242, Web Programming

SQL Statements

- Data Definition Language

    - CREATE

    - ALTER

    - DROP

- Data Manipulation Language

    - INSERT

    - UPDATE

    - DELETE

- Data Query Language

    - SELECT

- SQL statements end with a semicolon

- SQL keywords are not case sensitive

CREATE

- CREATE creates a new table
- The columns need to be given a datatype
- Syntax:
  ```
  CREATE TABLE table_name (
      column_name1 datatype,
      column_name2 datatype,
      ...
  );
  ```

CREATE Example

- Example create a new table named example with two columns:

```
CREATE TABLE example (
    column1 INTEGER,
    column2 TEXT
);
```

ALTER

- ALTER changes the structure of a table

- Syntax to add a column:

  ```
  ALTER TABLE table_name
  ADD COLUMN column_name datatype;
  ```

- Syntax to rename a table:

  ```
  ALTER TABLE table_name
  RENAME TO new_table_name;
  ```

ALTER Examples

- Example: add a new column to the example table:

  ```
  ALTER TABLE example
  ADD COLUMN column3 TEXT;
  ```

- Example: change a table name

  ```
  ALTER TABLE example
  RENAME TO Example;
  ```

DROP

- DROP deletes a table
- Syntax:

  ```
  DROP TABLE table_name;
  ```

DROP Example

- Example: delete the example table:

  ```
  DROP TABLE example;
  ```

INSERT

- INSERT inserts a row into a table

- Syntax:

  ```
  INSERT INTO table_name
  VALUES (value1, value2, ...);
  ```

- Syntax with explicit column names:

  ```
  INSERT INTO table_name (column1, column2, ...)
  VALUES (value1, value2, ...);
  ```

INSERT Example

- Example: insert a row into the first example table:

  ```
  INSERT INTO example (column1, column2)
  VALUES (23, 'Some text');
  ```

- Note: strings are in single quotes

UPDATE

- UPDATE changes the value of a column for a particular row in a table

- Syntax:

```
UPDATE table_name
SET column1=value1, column2=value2, ...
WHERE boolean_condition;
```

UPDATE Example

- Example: update the value of column2 from the previous insert

```
UPDATE example
SET column2 = 'New text'
WHERE column1 = 23;
```

- Note: strings are in single quotes

DELETE

- DELETE removes a row from a table

- Syntax:

  ```
  DELETE FROM table_name
  WHERE boolean_condition;
  ```

DELETE Example

- Example: remove the previously inserted row:

```
DELETE FROM example
WHERE column1 = 23;
```

SELECT

- SELECT retrieves information from a database

- Basic Syntax:

  SELECT column_name, column_name, ...
  FROM table_name;

- Syntax to select all columns from a table:

  SELECT * FROM table_name;

The DISTINCT keyword

- DISTINCT returns values with duplicates removed

- Syntax:

```
SELECT DISTINCT column_name, column_name, ...
FROM table_name;
```

The ORDER BY keyword

- ORDER BY sorts the records in ascending or descending order
- Syntax:

  ```
  SELECT column_name, column_name, ...
  FROM table_name
  ORDER BY column1 [ASC|DESC] , column2 [ASC|DESC], ... ;
  ```

The WHERE Clause

- WHERE clause filters information

- Syntax:

  ```
  SELECT column_name, column_name, ...
  FROM table_name
  WHERE column_name operator value;
  ```

WHERE Clause Operators

- =: equal
- <>: not equal
- >: greater than
- <: less than
- >=: greater than or equal to
- <=: less than or equal to
- AND: logical and
- OR: logical or
- NOT: logical not
- BETWEEN: between an inclusive range
- LIKE: search for a pattern
- IN: specify multiple possible values for a column

The AND and OR Operators

- The AND operator retrieves a record if both the first condition *and* the second condition are true

- The OR operator retrieves a record if either the first condition *or* the second condition are true

- Syntax:

```
SELECT column_name, column_name, ...
FROM table_name
WHERE column_name=value
AND columne_name=value;
```

The BETWEEN Operator

- The BETWEEN selects values within a range

- Syntax:

```
SELECT column_name, column_name, ...
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

The LIKE Operator

- The LIKE operator is used to search a column for a pattern

- Syntax:

  ```
  SELECT column_name, column_name, ...
  FROM table_name
  WHERE column_name LIKE pattern;
  ```

- The % is a wildcard that matches zero or more characters

The IN Operator

- The IN operator specifies multiple values in a WHERE clause

- Syntax:

```
SELECT column_name, column_name, ...
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

Aggregations

- Aggregations are functions that operate on a set of values (typically a column) and return a single value.

- Common aggregation functions:

    - `MIN` returns the minimum value

    - `MIN` returns the minimum value

    - `COUNT` returns the number of rows

    - `AVG` returns the average value of a numeric column

    - `SUM` returns the sum of a numeric column

Aggregation Example

```
SELECT min(column1), count(DISTINCT colum2)
FROM example;
```