Modifying Data With SQL Cheatsheet

Adding a Row to a Table

Inserting a single row:

```
INSERT INTO <table> VALUES (<value 1>, <value 2>, ...);
```

This will insert values in the order of the columns prescribed in the schema.

Examples:

```
INSERT INTO users VALUES  (1, "chalkers", "Andrew", "Chalkl
INSERT INTO users VALUES  (2, "ScRiPtKiDdIe", "Kenneth", "L

INSERT INTO movies VALUES (3, "Starman", "Science Fiction",
INSERT INTO movies VALUES (4, "Moulin Rouge!", "Musical", 2
```

Inserting a single row with values in any order:

```
INSERT INTO <table> (<column 1>, <column 2>) VALUES (<value
INSERT INTO <table> (<column 2>, <column 1>) VALUES (<value
```

Examples:

```
INSERT INTO users (username, first_name, last_name) VALUES
```

```
INSERT INTO users (first_name, last_name, username) VALUES

INSERT INTO movies (title, genre, year_released) VALUES ("S
INSERT INTO movies (title, year_released, genre) VALUES ("M
```

Adding Multiple Rows to a Table

Inserting multiple rows in a single statement:

```
INSERT INTO <table> (<column 1>, <column 2>, ...)
             VALUES
                    (<value 1>, <value 2>, ...),
                    (<value 1>, <value 2>, ...),
                    (<value 1>, <value 2>, ...);
```

Examples:

```
INSERT INTO users (username, first_name, last_name)
    VALUES
                   ("chalkers", "Andrew", "Chalkley"),
                   ("ScRiPtKiDdIe", "Kenneth", "Love");

INSERT INTO movies (title, genre, year_released)
     VALUES
                    ("Starman", "Science Fiction", 1984),
                    ("Moulin Rouge!", "Musical", 2001);
```

Updating All Rows in a Table

An update statement for all rows:

```
UPDATE <table> SET <column> = <value>;
```

The `=` sign is different from an equality operator from a `WHERE` condition. It's an *assignment operator* because you're assigning a new value to something.

Examples:

```
UPDATE users SET password = "thisisabadidea";
UPDATE products SET price = 2.99;
```

Update multiple columns in all rows:

```
UPDATE <table> SET <column 1> = <value 1>, <column 2> = <va
```

Examples:

```
UPDATE users SET first_name = "Anony", last_name = "Moose";
UPDATE products SET stock_count = 0, price = 0;
```

Updating Specific Rows

An update statement for specific rows:

```
UPDATE <table> SET <column> = <value> WHERE <condition>;
```

Examples:

```
UPDATE users SET password = "thisisabadidea" WHERE id = 3;
UPDATE blog_posts SET view_count = 1923 WHERE title = "SQL
```

Update multiple columns for specific rows:

```
UPDATE <table> SET <column 1> = <value 1>, <column 2> = <va
```

Examples:

```
UPDATE users SET entry_url = "/home", last_login = "2016-01
UPDATE products SET status = "SOLD OUT", availability = "In
```

Removing Data from All Rows in a Table

To delete all rows from a table:

```
DELETE FROM <table>;
```

Examples:

```
DELETE FROM logs;
DELETE FROM users;
DELETE FROM products;
```

Removing Specific Rows

To delete specific rows from a table:

```
DELETE FROM <table> WHERE <condition>;
```

Examples:

```
DELETE FROM users WHERE email = "andrew@teamtreehouse.com";
DELETE FROM movies WHERE genre = "Musical";
DELETE FROM products WHERE stock_count = 0;
```

Transactions

Switch autocommit off and begin a transaction:

```
BEGIN TRANSACTION;
```

Or simply:

```
BEGIN;
```

To save all results of the statements after the start of the transaction to disk:

```
COMMIT;
```

To reset the state of the database to before the beginning of the transaction:

```
ROLLBACK;
```