SQL Cheatsheet

Part 1: Starting with SQL

Glossary

Structured query language (SQL): A programming language for working with relational databases.

Database: A structured collection of data.

Schema: A database structure that defines how to organize data.

Table: A fundamental component of a database that organizes data into rows (records) and columns (fields).

Data type: A label that tells the database what kind of information can be stored in a particular column. Common data types include PRIMARY KEY, INT or INTEGER (numbers), VARCHAR (characters), NOT NULL (missing or unknown values), and DATE (year, month, and day).

Primary key: A unique identifier for a record in a table that ensures each row is distinct.

Foreign key: A field in a table that links to the primary key of another table, which establishes a relationship between the two tables.

SQL statement: A request for information from a database. SQL statements are colloquially referred to as queries. Some SQL statements include CREATE, INSERT, UPDATE, SELECT, and DELETE.

Common SQL statements

CREATE statements

With the CREATE statements, you can create new databases and database tables. The CREATE TABLE statement also defines the schema of each table by creating columns and their respective names and types.

Note: If you copy and paste these sample commands into the database management system of your choice, the single quotation marks might prevent you from running the commands. If you test the commands and experience an error, make sure that you manually replace the quotation marks at the beginning and end of each string. Use single straight quotation marks for the commands to run properly.

```
CREATE DATABASE database_name;
CREATE TABLE table_name (column1 datatype, column2 datatype, column3 datatype);
```

Note: After you create a database, select the database by entering the USE database_name; query. Then, create the database table.

Examples of CREATE statements:

```
CREATE DATABASE web_store;
CREATE TABLE orders (orderID INT PRIMARY KEY, itemID INT NOT NULL, department
VARCHAR(50), customerName VARCHAR(50), QTY INT NOT NULL, price VARCHAR(10),
orderDate DATE);
```

Table Orders

orderID	itemID	department	customerName	Qty	price	orderDate

INSERT statements

The INSERT INTO statement adds new data into a database.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Example of the INSERT INTO statement:

```
INSERT INTO orders (orderID, itemID, department, customerName, QTY, price,
orderDate)
VALUES (13293, 01234, 'electronics', 'Nikki Wolf', 1, '19.99', '2023/09/12'),
(64345, 02789, 'home', 'John Stiles', 2, '5.98', '2023/10/25'), (18946, 09251,
'food', 'Martha Rivera', 1, '14.99', '2023/09/04'), (23590, 03334, 'clothing',
'Diego Ramirez', 1, '39.95', '2023/10/18'), (41502, 09251, 'food', 'Zhang Wei', 2,
'29.98', '2023/09/02');
```

Table Orders

orderID	itemID	department	customerName	Qty	price	orderDate
13293	1234	electronics	Nikki Wolf	1	19.99	9/12/23
64345	2789	home	John Stiles	2	5.98	10/25/23
18946	9251	food	Martha Rivera	1	14.99	9/4/23
23590	3334	clothing	Diego Ramirez	1	39.95	10/18/23
41502	9251	food	Zhang Wei	2	29.98	9/2/23

UPDATE statements

The UPDATE statement updates data in a database.

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

Example of the UPDATE statement:

```
UPDATE orders
SET QTY = 1, price = '2.99'
WHERE orderID = 64345;
```

Table Orders

orderID	itemID	department	customerName	Qty	price	orderDate
13293	1234	electronics	Nikki Wolf	1	19.99	9/12/23
64345	2789	home	John Stiles	1	2.99	10/25/23
18946	9251	food	Martha Rivera	1	14.99	9/4/23
23590	3334	clothing	Diego Ramirez	1	39.95	10/18/23
41502	9251	food	Zhang Wei	2	29.98	9/2/23

Part 2: Manipulating data in a single table

SELECT statements

The SELECT statement shows the requested information from a specific table. With a SELECT query, you can retrieve data, filter it, aggregate it, and more.

Retrieve all data

To return all information from the table, use the SELECT * statement with the FROM clause, which identifies the source table that contains the data.

```
SELECT * FROM table_name;
```

Example of the SELECT statement:

```
SELECT * FROM orders;
```

Table Orders

orderID	itemID	department	customerName	Qty	price	orderDate
13293	1234	electronics	Nikki Wolf	1	19.99	9/12/23
64345	2789	home	John Stiles	2	5.98	10/25/23
18946	9251	food	Martha Rivera	1	14.99	9/4/23
23590	3334	clothing	Diego Ramirez	1	39.95	10/18/23
41502	9251	food	Zhang Wei	2	29.98	9/2/23

Retrieve specific information

To return a specific column, use the SELECT column_name statement.

```
SELECT column_name FROM table_name;
```

Example of the SELECT statement:

```
SELECT department FROM orders;
```

Table Orders

department
electronics
home
food
clothing
food

Filter data

To filter rows in a table, apply the WHERE clause. WHERE locates all results that match your criteria.

```
SELECT column_name FROM table_name WHERE condition;
```

To specify the condition, use the AND, OR, NOT, and BETWEEN operators.

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

Example of the SELECT statement:

```
SELECT QTY, orderDate FROM orders WHERE orderDate BETWEEN '2023/09/01' AND
'2023/09/30';
```

Table Orders

Qty	orderDate
1	9/12/23
1	9/4/23
2	9/2/23

Aggregate data

To combine multiple rows of data into a single summary result, use the following statements:

SUM: Calculate the sum of values in a column.
MAX: Retrieve the maximum value in a column.
COUNT: Count the number of rows in a specified column or table.
AVG: Calculate the average of values in a column.
MIN: Find the minimum value in a column.
GROUP BY: Aggregate data based on a specific condition.
ORDER BY: Sort data based on a specific order.

Example of the SELECT statement with an aggregate data function:

```
SELECT SUM(column_name) FROM table_name;
```

Example of the SUM aggregate data function:

```
SELECT SUM(price) FROM orders;
```

Table Orders

SUM(price)
107.9

DELETE statement

The DELETE statement removes records from the table.

```
DELETE FROM table_name WHERE condition;
```

Example of the DELETE statement:

```
DELETE FROM orders WHERE customerName = 'Martha Rivera';
```

Table Orders

orderID	itemID	department	customerName	Qty	price	orderDate
13293	1234	electronics	Nikki Wolf	1	19.99	9/12/23
64345	2789	home	John Stiles	1	2.99	10/25/23
23590	3334	clothing	Diego Ramirez	1	39.95	10/18/23
41502	9251	food	Zhang Wei	2	29.98	9/2/23

DROP statement

The DROP statement deletes the database or the database table.

```
DROP TABLE table_name;
```

Example of the DROP statement:

```
DROP TABLE orders;
```