Structured Query Language (SQL)

Seth Temple
sethtem@umich.edu

This handout provides knowledge of relational databases and specifically useful SQL terms.

1 The very basics

Relational databases organize data into tables with well-defined relationships between them. They enforce data consistency and accuracy through constraints, such as **primary keys** and **foreign keys**. Relational databases offer powerful queries, via the structured query language (SQL), for retrieving and manipulating data. Relational databases are *widely used in enterprise applications.*

SQL is a *declarative* programming language. You specify what you want to do, and the program gets the outcome without you specifying how to achieve the outcome. Many are more familiar with imperative programming like Python, where the focus is on how to perform the task. Some important SQL basics are:

- Always finish queries with **semicolon ;**.
- **Asterisk *** will show all data in the table.
- Standard naming conventions:
  - The language is **case-insensitive, but**
  - All caps for key words
  - Camel case for column names
  - Call tables as plurals
- The language is not syntax sensitive like Python. Use new lines to improve code readability.
- Use the keyword **AS** to provide shorthand for columns
- Use the keyword **LIMIT** followed by integers to see the start of a table.

Seth Temple
sethtem@umich.edu

Different databases may have different commands and data types implemented. Some example database management systems (DBMS) are:

- MySQL : open-source
- SQLite : lightweight and self-contained, often used in mobile
- Microsoft SQL Server : commercially available
- Cloud-based SQL services (Amazon, Azure, Google)
- You can *interface with SQL via Python*, SAS, etc.
- *NoSQL* databases are designed to handle large amounts of unstructured or semi-structured data and provide a flexible schema or no schema at all. An example is MongoDB. These DMBS are more appropriate for complex data like graph data or JSON-formatted data.

References for this document come from this LinkedIn Learning collection, including some practice exercises. I also used Llama from Meta AI to draft this handout.

2 Data manipulation language (DML)

Most data scientists use SQL as a DML. The acronym **CRUD**, which stands for Create, Read, Update, and Delete, describes the functions of a DML. The main SQL key word that we use is **SELECT**. The following is a typical command that we use:

SELECT ColumnOne, ColumnTwo AS Column2
FROM table
WHERE some_condition
ORDER BY Column2 ASC
LIMIT 10;

2.1 Filtering

Filter with the keyword **WHERE** followed by conditions.

- Use = operator, in contrast to the Python == operator
- AND plus OR for logic
- Use parentheses for complex logic
- Use IS NULL plus IS NOT NULL, in contrast to = NULL
- Use **LIKE** '%somestring%' for regular expressions. There is syntax for more complicated regular expressions as well.
- Use **HAVING** instead when filtering on grouped data.

2.2 Math

The following functions are useful for math aggregate operations.

- +, -, *, /, %

- COUNT
- MIN
- MAX
- SUM
- AVG
- ROUND
- VAR_POP
- STDDEV_POP
- TRUNC
- CEIL
- FLOOR

You use **GROUP BY** followed by one or more columns (comma separated).

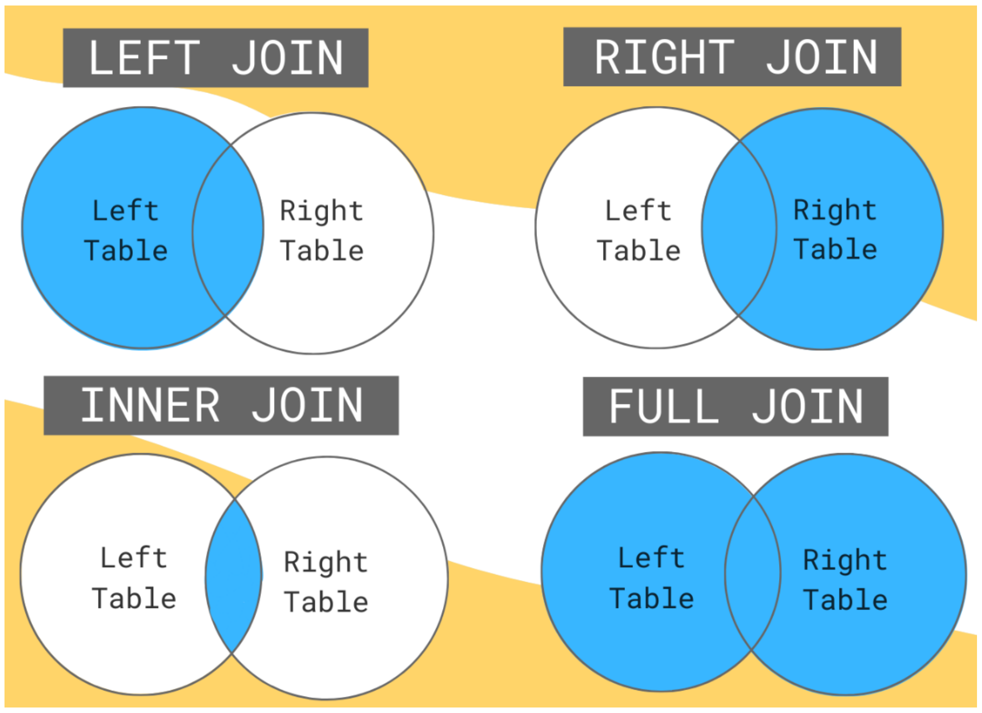
2.3 Text

The following functions are useful for text operations.

- LOWER
- UPPER
- SUBSTR(phrase, start_index, string_length)
- CONCAT(phrase1, phrase2)
- REPLACE(phrase, substr1, substr2)
- INITCAP
- LENGTH
- TRIM

2.4 Joins

Joining tables, via their relations with common keys, is a main advantage of SQL compared to Excel, Python, and other software. The main syntax is **JOIN** table1.Column **ON** table2.Column. Below is a visual of the join types.



2.5 Insert, update, and delete

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

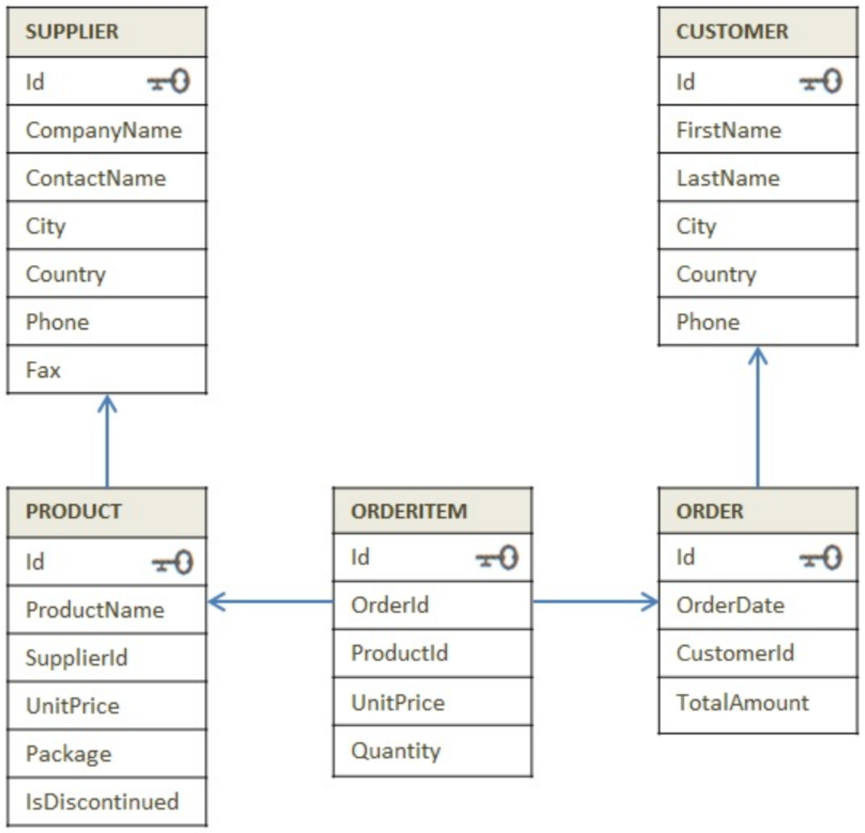
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;

DELETE FROM table_name
WHERE condition;

Seth Temple
sethtem@umich.edu

3 Data definition language (DDL)

Tables are connected via keys. Below is an example of a database solution for a consumer business. Data scientists may be less involved in creating the database in an enterprise.



3.1 Tables

The command to make a table is:

CREATE TABLE table (
ColumnOne type,
ColumnTwo type,
PRIMARY KEY (ColumnOne) AUTO INCREMENT
)

Seth Temple
sethtem@umich.edu

The options to define columns in the table are the following. You append these to your column definitions.

- AUTO INCREMENT : good to use for the primary key
- **FOREIGN KEY** : references primary key in another table, e.g., FOREIGN KEY (KeyColumn) REFERENCES othertable(OtherColumn)
- UNIQUE : requires that all values are unique
- NOT NULL : requires that all values are specified
- CHECK : enforces a specific condition,
  e.g., (Country IN ('USA', 'Canada'))
- DEFAULT : fills value when unspecified,
  e.g., DEFAULT CURRENT_TIMESTAMP

3.2 Data types

The data types are the following.

- INT (faster and smaller storage up to 32 bits) or BIGINT (64 bits for high-volume). Use UNSIGNED for non-negative values.
- VARCHAR(###) or CHAR(###)
  - CHAR is fixed length but can be faster
  - VARCHAR can be slower but saves memory
- Floating point numbers
  - FLOAT : 32 bit precision
  - DOUBLE : 64 bit precision
  - DECIMAL(num_digits, num_places)
- DATE or TIME or DATETIME or TIMESTAMP
- BOOLEAN

3.3 Indices

Indices help optimize for performance in huge tables:

- Use for columns that we frequently filter on
- Use for columns with many unique values
- Slows down write operations
- Commands:
  - CREATE INDEX IndexName ON table (IndexedColumn);
  - SHOW INDEX FROM table;
  - DROP INDEX IndexName FROM table;

3.4 Views

These are virtual tables based on a SELECT query. It presents a simplified table without storing new data. The syntax is the following.

```
CREATE VIEW view_name AS
ColumnOne, ColumnTwo, ...
FROM table
WHERE Condition;
```

You can use views to restrict data access to some roles with

```
GRANT SELECT ON view_name TO job_role;
REVOKE SELECT ON view_name TO user;
```

You can use GRANT and REVOKE for other key words and on tables.

Seth Temple
sethtem@umich.edu

3.5 Schemas

A schema is the blueprint of a database, including the relationships between different tables, columns, and data types. It defines how data is stored and organized in a database. You can set one up by the following.

```
CREATE DATABASE mydatabase;
USE mydatabase;
CREATE SCHEMA myschema;
CREATE TABLE myschema.customers (
customer_id INT PRIMARY KEY,
name VARCHAR(255),
email VARCHAR(255)
);

CREATE TABLE myschema.orders (
order_id INT PRIMARY KEY,
customer_id INT,
order_date DATE,
FOREIGN KEY (customer_id) REFERENCES myschema.customers(customer_id)
);
```

To use tables shorthand within the schema, you may use USE myschema;

4 More examples

4.1 Apply discount price to items based on switch logic

This example shows how to use switch logic with **CASE**, **WHEN**, **THEN**, and **ELSE**.

```
SELECT
Name,
Price AS OriginalPrice,

CASE
WHEN Price < 7. THEN '5%'
WHEN Price BETWEEN 7. AND 10. THEN '10%'
ELSE '15%'
END DiscountPercent,

ROUND(
Price * (
1 - CASE
WHEN Price < 7. THEN 0.05
WHEN Price BETWEEN 7. AND 10. THEN 0.10
ELSE 0.15
END
),
2
) DiscountPrice

FROM Dishes
ORDER BY Name ASC;
```

4.2 How many pizzas were ordered each day?

This example shows how to cast from a datetime column. Moreover, the filter is on the individual item, not a grouped column.

```
SELECT
COUNT(Orders.OrderID) NumSold,
CAST(Orders.OrderDate AS DATE) Day
FROM Orders
JOIN OrdersDishes ON OrdersDishes.OrderID = Orders.OrderID
JOIN Dishes ON Dishes.DishID = OrdersDishes.DishID
WHERE Dishes.Name = 'Handcrafted Pizza'
GROUP BY Day
ORDER BY Day ASC;
```

4.3 Find your top customers

This example shows how to filter on a grouped column. Note how we use the many-to-many table OrdersDishes to combine data from the Orders and Dishes tables.

```
SELECT
Customers.CustomerID,
Customers.FirstName,
Customers.LastName,
SUM(Dishes.Price) AS TotalSpend
FROM Orders
JOIN Customers ON Customers.CustomerID = Orders.CustomerID
JOIN OrdersDishes ON Orders.OrderID = OrdersDishes.OrderID
JOIN Dishes ON OrdersDishes.DishID = Dishes.DishID
GROUP BY Customers.CustomerID
HAVING TotalSpend > 450.
ORDER BY TotalSpend DESC;
```

Seth Temple
sethtem@umich.edu

5 SQL in Python

Words

Seth Temple
sethtem@umich.edu

6 Advanced

Words