SQL Notes
Brandon J. Knight

Introduction to SQL

- A **database** is a system used to store and manage data. More specifically, it allows us to:
    - **Store data reliably** so it's preserved and protected
    - **Retreive information efficiently** when we need it
    - **Manipulate data systematically**to transform and update it

Data Organization:

- Data in a database is organized into **tables** with rows and columns.
- **Rows** represent individual entities or events.
- **Columns** represent specific attributes or properties.

When I'm extracting data from various sources for my pipelines, I'm pulling from databases that power specific source systems–whether it's a CRM, an ecommerce platform, or internal business tools.

- One of the most popular types of databases is a **relational database**. These databases store tables that represent specific entities or events.

SQL Basics:

- Structured Query Language (SQL) is the most widley used programming language for working with data in databases
- In its most common usage, you write SQL code to request data from databases, and they return the results you needs.
- SQL is much impeller than general programming languages because it's designed for one specific purpose: working with data.

Benefits of SQL tables over Excel or Google sheets:

- Designed to handle much larger amounts of data (millions or billions of rows)
- Allow thousands of users to access and update data simultaneously
- Optimized for automated access by applications

Select Statement:

Returns all the data from 2 columns

```
SELECT id,
FROM products;
```

Returns all the data from 3 columns

```
SELECT id,
       name,
       rating
FROM products;
```

SQL Notes
Brandon J. Knight

Finding top 10 highest-rated products

```
SELECT id,
       name,
       rating
FROM products
ORDER BY rating DESC
LIMIT 10;
```

Column Selection

```
SELECT column_1,
       column_2
FROM table_1;
```

- We used **SELECT** to specify which columns to retrieve and **FROM** to specify the table to retrieve from.
- **SELECT column_1**: selects one column
- **SELECT column_1, column_2**: selects multiple columns
- **SELECT ***: selects all columns

Sorting

```
SELECT column_1,
       column_2
FROM table_1
ORDER BY column_1 DESC;
```

- We use **ORDER BY** to sort rows by a specific column.
- **ORDER BY column_1**: Ascending order (lowest to highest A-z, 0-9)
- **ORDER BY rating DESC:** Descending order (highest to lowest Z-A, 9-0)

Row Limiting

```
SELECT *
FROM table_1
LIMIT 10;
```

- We use LIMIT to restrict the number of rows returned by a query
- LIMIT is placed at the end of the query and returns the first N rows products by the query
- **Limiting is an essential practice because it prevents slow, expensive queries** and protects against mistakes that might return far more data than expected.

Temporary Results

SQL Notes
Brandon J. Knight

When running a **SELECT** query, **the results is a temporary table** that matches your criteria. This is just a temporary copy of the data–it's not stored in the database. If you want to keep these results for later use, you need ot intentionally save them to your computer or the cloud.

The original database remains completely unchanged

Error Handling

- **SQL requires precise syntax**, and small mistakes can prevent queries from running.
- Read error messages carefully, check spelling, use a good development environment, and leverage AI assistance.
- The more you work with SQL, the fewer mistakes you'll make and the faster you'll spot and fix them.

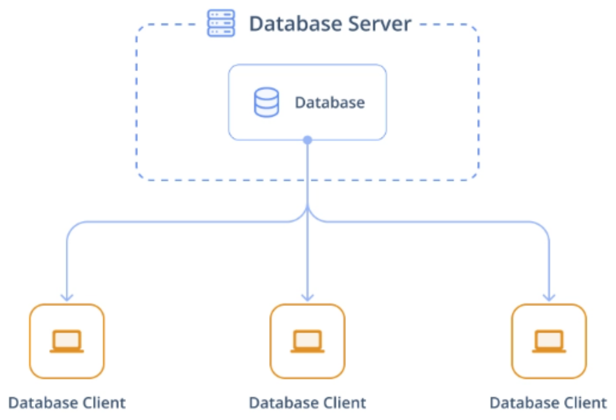
SQL Server Specifics:

- **Limiting results**:
    - In SQL Server, instead of LIMIT 10, you use: SELECT TOP(10)
    - Example: SELECT TOP(10) * FROM products;
- **Case sensitivity**:
- SQL Server is **not case-sensitive** for column names, table names, and keywords
- So **customer, Customer,** and **CUSTOMER** are all treated as the same.

When working with a database, there are **two systems involved**:

1. **Database Server**: The computer than stores and runs SQL Server
2. **Database Client**: The software on your computer (like SQL Server Management Studio) that lets you connect to the server, write SQL queries, send them to the database, and see the results.

Multiple people can connect to the **same SQL Server** simoltaneiously using their own clients.
Each person sees the same data (unless permissions differ).
This setup is called **client-server architecture**–and it's how most database systems work, whether on-permises like SQL Server or in the cloud.



SQL Notes
Brandon J. Knight

To connect SQL Server Management Studio to a database, you'll need:
Server Name: The name or address of the SQL Server
Authentication method: Either Windows Authentication (uses your Windows login) or SQL Server Authentication (requires a username and password)
Database name: Which specific database to use

When working at an organization, you'll need to **request these connection details from your database administrator or IT team**. They'll provide the server name, set up your access permissions, and tell you which authentication method to use.

The **database server** handles storage and query processing, while the **database client** (like SSMS) connects to the server, sends queries, and displays the results in a user-freindly format.

Finding Unique Values

```
SELECT DISTINCT column_name
FROM table_name;
```

- We use **DISTINCT** to retrieve only unique values from a column, eliminating duplicates.
- **DISTINCT** is placed immediately after **SELECT**.

Column Aliasing

```
SELECT column_1 AS alias_name,
       column_2
FROM table_1;
```

- The **AS** keyword assigned temporary names to the columns in query results to improve readability.
- The alias appears in column headers but doesn't change the original database column name.
- Effective aliases should be descriptive, consistent, and use snake_case convention for best practices.

SQL Style Conventions

- Write SQL keywords in **uppercase** (SELECT, FROM, ORDER BY) to distinguish them from table and column names.
- This convention improves code readability and makes queries easier to understand and maintain.

SQL Flavors

- Different database systems use **SQL flavors** with the same core syntax but minor differences in specific features.
- What you learn in one system transfers easily to others, making SQL skills boardly applicable across platforms.