Submitted by: Karen Ngugi - karenwngugi@gmail.com

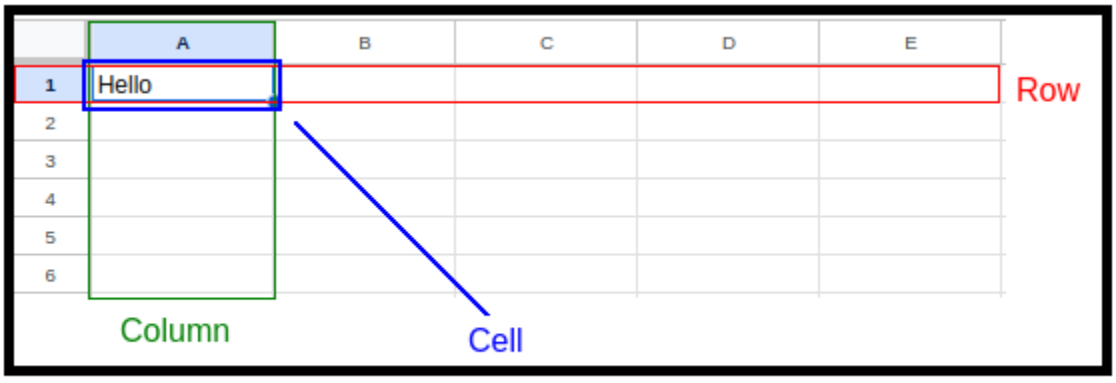
Introduction to SQL and Relational Databases

Introduction

Data is so important and prevalent in today's world that you would be hard pressed to find an organization or business that does not deal with it. Organizations collect all sorts of data from personal information to survey feedback to sales records to digital media. Such data would be useless if there is nowhere to store it, which is why there are various storage types such as databases, data lakes, and data warehouses. Databases are the most common data storage type and will be the focus of this lesson.

Databases

In order to understand databases, we need to look at tables. A **table** is a simple rectangular structure that is divided into horizontal sections called **rows** and vertical sections called **columns**. Rows and columns intersect to create **cells** which is where a single piece of data is stored.



Table

We have used tables virtually all of our lives and in many forms such as school timetables, duty rosters, etc. When you put together two or more tables, you get a **database** which is basically a container that stores data in an organized

manner. There are many types of databases such as object-oriented databases, NoSQL databases, hierarchical databases, etc.

The most common database in usage is a **relational database**. In relational databases, the tables are related to each other. An example would be a database that contains the following three tables:

- Customer information e.g. Customer ID, first and last names, telephone number, etc.
- Product information e.g. product ID, product name, quantity
- Order information e.g. Order ID, ID of customer who made the order, ID of the product ordered

Suppose you have a table like this: click to view table

Date	Time	Order No	Product ID	Product Category	Product Name	Quantity	Price	Customer ID	Customer First Name	Customer Last Name	Customer Phone Number
Wednesday March 22, 2023	11:01 AM	2474	5442	Skincare	Sunscreen	1	1200	3001	Maggie	Smith	123-456-7890
Wednesday March 22, 2023	11:01 AM	2474	7341	Skincare	Shea butter	1	800	3001	Maggie	Smith	123-456-7890
Wednesday March 22, 2023	11:01 AM	2474	9970	Food	Large chips	1	275	3001	Maggie	Smith	123-456-7890
Wednesday March 22, 2023	11:01 AM	2474	2023	Skincare	Body scrub	1	1200	3001	Maggie	Smith	123-456-7890
Monday February 13, 2023	3:51 PM	1237	8934	Toiletries	Scented candles	2	750	4119	Tim	Brown	987-654-3210
Monday February 13, 2023	3:51 PM	1237	3054	Toiletries	Incense sticks	1	90	4119	Tim	Brown	987-654-3210
Friday January 27, 2023	9:15 AM	982	6126	Skincare	Goatmilk soap	4	1000	2222	Joanna	West	777-666-5555
Friday January 27, 2023	9:15 AM	982	1234	Food	Chocolate bar	1	120	2222	Joanna	West	777-666-5555
Friday January 27, 2023	9:15 AM	982	9832	Drinks	Red wine	1	4100	2222	Joanna	West	777-666-5555
Friday January 27, 2023	9:15 AM	982	5567	Toiletries	Jumbo toilet paper roll	1	435	2222	Joanna	West	777-666-5555

You can see that there is a lot of repeated data, which we call **redundancy**. It might not be problematic on a small table like this but imagine if you had to deal with tens or hundreds of thousands of records like this. Not only will such a database be dreary to go through, it will boggle down your system because it is consuming more resources than necessary. You could filter to find the information you want, such as information regarding a specific customer or product, but that's just dreary. The solution would be to categorize the data into separate tables and link them using primary and foreign keys.

For example, you can have this table containing only customer information:

Customer ID	Customer First Name	Customer Last Name	Customer Phone Number
2222	Joanna	West	777-666-5555
3001	Maggie	Smith	123-456-7890
4119	Tim	Brown	987-654-3210

And a table containing only product information

Product ID	Product Category	Product Name	Price per item
1234	Food	Chocolate bar	120
2023	Skincare	Body scrub	1200
3054	Toiletries	Incense sticks	90
5442	Skincare	Sunscreen	1200
5567	Toiletries	Jumbo toilet paper roll	435
6126	Skincare	Goatmilk soap	250
7341	Skincare	Shea butter	800
8934	Toiletries	Scented candles	375
9832	Drinks	Red wine	4100
9970	Food	Large chips	275

You can then combine the Customers data and the Products data in the Orders data:

Date	Time	Order No	Customer ID	Product ID
Wednesday March 22, 2023	11:01 AM	2474	3001	5442, 7341. 9970, 2023
Monday February 13, 2023	3:51 PM	1237	4119	8934, 3054
Friday January 27, 2023	9:15 AM	982	2222	6126, 1234, 9832, 5567

Now let's talk about primary and foreign keys. **Primary keys** are the values of a specific column in a table that contains only unique values. These are used to keep the identities of the records separate and unique so that there can be no duplicates. In the Customers table, **Customer ID** is the primary key for the Customers table and **Product ID** is the primary key for the Products table. **Foreign keys** are primary keys of a table that are used in another table. In our example, **Customer ID** and **Product ID** are foreign keys in the Order table.

Relational Database Management Systems (RDBMS)

Relational database management systems are special software systems that allow you to manage and manipulate relational databases. There are many RDBMS software you can get such as PostgreSQL, MySQL, Microsoft SQL Server, and Oracle Database among others. Relational database management systems offer the following advantages:

- **Ease of access** - RDBMS have functionalities built into them that make it fast and easy to locate and retrieve specific data
- **Data integrity** - RDBMS check data for errors and inconsistencies before the data is stored in order to prevent data corruption and ensure data accuracy

- **Multi-person access** - RDBMS allows multiple users to access the same data at the same time
- **Accuracy and non-redundancy** - RDBMS makes use of primary and foreign keys to ensure that data is spread across multiple tables which are connected to one another thereby eliminating the need to repeat data unnecessarily
- **Scalability/flexibility** - RDBMS allow you to store and manage large amounts of data
- **Security** - RDBMS allow users to control access to data so that they are able to grant access only to those authorized to do so
- **Backup and recovery** - RDBMS provides built-in data backup and recovery features that allow you to recover your data in case of a disaster

SQL

Structured Query Language (SQL) is a programming language that is used to manage and manipulate data in tables and databases. A **SQL statement** is a sentence or command that is made up of special SQL keywords called **clauses**. Examples of popular SQL clauses include SELECT, CREATE, FROM, WHERE, etc. All RDBMS use SQL, though they differ slightly in syntax, For example, one requires that all SQL statements end with a semicolon while another can run without issues if a semicolon isn't included. One allows clauses to not be case-sensitive (i.e., it doesn't matter if the clause is in uppercase or lowercase) while another requires that the clauses are in uppercase. However, the differences are not critical and, once you get the hang of basic SQL, you will be good to go.

Below are examples of SQL statements:

- **CREATE DATABASE** my_database
  Creates a database called "my_database"
- **CREATE TABLE** my_table
  Creates a table called "my_table"
- **SELECT** * **FROM** my_table
  Displays all the records that are in the "my_table" table
- **SELECT** column_a, column_b **FROM** my_table
  Displays only "column_a" and "column_b" from the "my_table" table
- **SELECT COUNT**(*) **FROM** my_table
  Returns the number of records in the table

- **SELECT DISTINCT**(column) **FROM** my_table
  Returns distinct/unique values from that column in the "my_table" table
- **INSERT INTO** my_table (column_a, column_b, column_c) **VALUES** value_a, value_b, value_c
  Adds "value_a", "value_b", and "value_c" into "column_a", "column_b", and "column_c" respectively of the "my_table" table
- **SELECT** * **FROM** my_table **WHERE** column_a **IS NULL**
  Returns all records from "my_table" where "column_a" has null values
- **SELECT** * **FROM** my_table **WHERE** column_a **IS NOT NULL**
  Returns all records from "my_table" where "column_a" does not null values
- **SELECT** * **FROM** my_table **WHERE** column_a **BETWEEN** value_a **AND** value_b
  Returns all records from "my_table" where "column_a" is between "value_a" and "value_b" inclusive
- **SELECT** * **FROM** my_table **LIMIT** 5
  Returns exactly 5 records from the "my_table" table
- **SELECT** * **FROM** my_table **ORDER BY** column_a
  Returns all records from "my_table" ordered by "column_a"
- **SELECT** * **FROM** my_table **WHERE** column_a **LIKE** 'A%'
  Returns all records from "my_table" where the value of column_a is anything that starts with 'A'
- **SELECT** * **FROM** my_table **WHERE** column_a **LIKE** 'B_G'
  Select all records from "my_table" where the values in "column_a" start with 'B' and end with 'G' and the middle letter can be any character or number
- **UPDATE** my_table **SET** column_a = value_a **WHERE** column_a <condition>
  Updates "column_a" with "value_a" where it meets a certain condition e.g., column_a IS NULL or column_a = some_value
- **DELETE FROM** my_table **WHERE** <condition>
  Deletes records from "my_table" that meet the criteria in "condition"
- **DROP TABLE** my_table
  Deletes the "my_table" table
- **SELECT** table_1.column_1, table_1.column_2, table_2.column_1 **FROM** table_1, table_2 **WHERE** table_1.column_1 = table_2.column_1
  Selects "column_1" and "column_2" from "table_1" and "column_1" from "table_2" where the values of "column_1" of "table_1" and "column_1" of "table_2" are equal

There are many, many more statements and clauses to learn. Once you're familiar with these SQL commands, head over to https://www.sql-practice.com/ to put your knowledge to the test. You can read more on SQL commands on https://www.w3schools.com/sql/