**Getting Started with SQL**

In this module, you will be introduced to databases. You will learn how to use basic SQL statements like SELECT, INSERT, UPDATE and DELETE. You will also get an understanding of how to refine your query results with the WHERE clause as well as using COUNT, LIMIT and DISTINCT.

## **Learning Objectives:**

* Define databases and SQL
* Write basic data retrieval queries using SELECT statement
* Refine query results by using COUNT, DISTINCT and LIMIT statements
* Insert data to a table using INSERT statements and alter or remove data from a table using UPDATE and DELETE statements

# **Welcome to the course**

## **Course Introduction**

* **Importance of Data Science**: Data science is a rapidly growing field, essential for analyzing and managing vast amounts of data generated daily.
* **Role of SQL**: SQL (Structured Query Language) is crucial for querying and manipulating data in databases, making it a vital skill for data professionals.
* **Python Integration**: Python is used alongside SQL to access databases and perform data analysis, leveraging its libraries for efficient data handling.
* **Course Structure**: The course covers SQL basics, data manipulation, database creation, and advanced SQL concepts, with hands-on labs and a final project.

## **Course Overview**

The course is structured into 6 weeks, each focusing on a specific topic.

**Week 1: Getting started with SQL**

This week will let you learn the basics of SQL and databases. You will also learn how to query tables in a database.

**Week 2: Introduction to relational Databases and Tables**

This week is all about relational databases, creating tables, and modifying their contents.

**Week 3: Intermediate SQL**

In this module, you will learn more about different types of SQL queries, functions, string patterns, grouping, and sorting.

**Week 4: Accessing databases with Python**

This week, you will learn the nuances of accessing databases using Python libraries and SQL magic in Jupyter Notebooks.

**Week 5: Course Assignment**

This week is designed to give you an understanding of how to deal with real-world datasets and complete an assignment which tests your skills acquired throughout the course.

**Week 6: Bonus Module: Advanced SQL for Data Engineers (Honors)**

In this additional module, you will learn how to apply advanced queries in SQL, like Views, Stored Procedures, and ACID transactions.

This course can be applied to multiple specializations or Professional Certificates programs. Completing this course will count towards your learning in any of the following programs:

* [Data Engineering Foundations Specialization](https://www.coursera.org/specializations/data-engineering-foundations)
* [IBM Data Analyst Professional Certificate](https://www.coursera.org/professional-certificates/ibm-data-analyst)
* [Data Science Fundamentals with Python and SQL Specialization](https://www.coursera.org/specializations/data-science-fundamentals-python-sql)
* [IBM Data Engineering Professional Certificate](https://www.coursera.org/professional-certificates/ibm-data-engineer)
* [Introduction to Data Science Specialization](https://www.coursera.org/specializations/introduction-data-science)
* [IBM Data Science Professional Certificate](https://www.coursera.org/professional-certificates/ibm-data-science)

# **Basic SQL**

## **Introduction to Databases**

* **What is SQL?**
  + SQL stands for **Structured Query Language**.
  + It is used for querying and managing data in relational databases.
* **What is Data?**
  + Data is a collection of facts, which can be in the form of words, numbers, or images.
  + It is a critical asset for businesses, stored securely for quick access.
* **What is a Database?**
  + A database is a program that stores data and allows for adding, modifying, and querying that data.
  + Data is often organized in **tables** (like spreadsheets) with columns and rows.
* **Relational Database:**
  + A relational database organizes data in tables and allows relationships between them.
  + **RDBMS** (Relational Database Management System) is a set of software tools for managing data.
* **Basic SQL Commands:**
  + **CREATE**: Create a table.
  + **INSERT**: Add data to a table.
  + **SELECT**: Retrieve data from a table.
  + **UPDATE**: Modify existing data in a table.
  + **DELETE**: Remove data from a table.

## **Select Statement**

* **Purpose of a Database Management System (DBMS)**:
  + The main goal is to **store and facilitate retrieval** of data.
* **SELECT Statement**:
  + Used to **retrieve data** from a relational database table.
  + It is a part of **Data Manipulation Language (DML)**, which is used to read and modify data.
  + The output of a SELECT statement is called a **result set** or **result table**.
* **Basic Syntax**:
  + The simplest form of a SELECT statement is:

SELECT \* FROM table\_name;

* + For example, to retrieve all data from a table named "book":

SELECT \* FROM book;

* **Retrieving Specific Columns**:
  + You can specify which columns to retrieve instead of selecting all:

SELECT book\_id, title FROM book;

* + The order of columns in the result set matches the order in the SELECT statement.
* **Using the WHERE Clause**:
  + The WHERE clause is used to **restrict the result set** based on a condition (predicate).
  + A predicate evaluates to **true, false, or unknown**.
  + Example of using WHERE to find a specific book:

SELECT book\_id, title FROM book WHERE book\_id = 'B1';

* **Comparison Operators**:
  + The WHERE clause can use various comparison operators:
    - **Equal to** (=)
    - **Greater than** (>)
    - **Less than** (<)
    - **Greater than or equal to** (>=)
    - **Less than or equal to** (<=)
    - **Not equal to** (<>)

By understanding these concepts, you will be able to effectively retrieve data from relational database tables using SQL.

## **Select Statement Examples**

**SELECT statement examples**

**Estimated time:** 5 min

**Objectives**

At the end of this reading, you will learn how to:

* Use various SELECT queries to retrieve data from the database.

**SELECT statement usage**

SELECT is classified as a Database Query command used to retrieve information from a database table.

There are various forms in which a SELECT statement is used.

1. The **general syntax** of a SELECT statement retrieves the data under the listed columns from Table\_1. The code is:
2. SELECT COLUMN1, COLUMN2, ... FROM TABLE\_1 ;

Copied!

1. To retrieve all columns from a table, use **" \* "** instead of specifying individual column names. The code below retrieves the entire table.
2. SELECT \* FROM TABLE\_1 ;

Copied!

1. Use the **WHERE clause** to filter the required data based on a predicate. The code below filters the response to only the entries that match the predicate.
2. SELECT <COLUMNS> FROM TABLE\_1 WHERE <predicate> ;

Copied!

**SELECT examples**

Let's look at these codes in action. Below is a database table called 'COUNTRY,' which contains the columns ID, Name, and CCode. Here, CCode is a 2 letter country code.

| **ID** | **Name** | **CCode** |
| --- | --- | --- |
| 1 | United States of America | US |
| 2 | China | CH |
| 3 | Japan | JA |
| 4 | Germany | GE |
| 5 | India | IN |
| 6 | United Kingdom | UK |
| 7 | France | FR |
| 8 | Italy | IT |
| 9 | Canada | CA |
| 10 | Brazil | BR |

**Example #1**

When we apply the SELECT code SELECT \* FROM COUNTRY ;, the query retrieves all rows and columns from the database table named COUNTRY.

* 'SELECT \*' instructs the database to select all columns from the table.
* 'FROM COUNTRY' specifies the table from which to retrieve the data. In this case, it's the "COUNTRY" table, so the entire table appears, as shown below.

Response:

| **ID** | **Name** | **CCode** |
| --- | --- | --- |
| 1 | United States of America | US |
| 2 | China | CH |
| 3 | Japan | JA |
| 4 | Germany | GE |
| 5 | India | IN |
| 6 | United Kingdom | UK |
| 7 | France | FR |
| 8 | Italy | IT |
| 9 | Canada | CA |
| 10 | Brazil | BR |

**Example #2**

The SQL query SELECT ID, Name FROM COUNTRY ; retrieves specific columns from a database table named 'COUNTRY'.

* 'SELECT ID, Name' instructs the database to select two specific columns from the table: "ID" and "Name." It will return these two columns for each row that matches the query criteria.
* 'FROM COUNTRY' specifies the table from which to retrieve the data, which is the "COUNTRY" table. The table below shows that only the "ID" and "CCode" columns were retrieved.

Response:

| **ID** | **Name** |
| --- | --- |
| 1 | United States of America |
| 2 | China |
| 3 | Japan |
| 4 | Germany |
| 5 | India |
| 6 | United Kingdom |
| 7 | France |
| 8 | Italy |
| 9 | Canada |
| 10 | Brazil |

**Example #3**

The SQL query SELECT \* FROM COUNTRY WHERE ID <= 5 ; retrieves all columns from the "COUNTRY" table where the value in the "ID" column is less than or equal to 5.

* `SELECT \* instructs the database to select all columns from the specified table.
* FROM COUNTRY specifies the table from which to retrieve the data, which is the 'COUNTRY' table.
* WHERE ID <= 5 ; is a condition that filters the rows from the table. It will only return rows where the value in the "ID" column is less than or equal to 5. In the table below, you can see that only rows 1-5 were retrieved.

Response:

| **ID** | **Name** | **CCode** |
| --- | --- | --- |
| 1 | United States of America | US |
| 2 | China | CH |
| 3 | Japan | JA |
| 4 | Germany | GE |
| 5 | India | IN |

**Example #4**

The SQL querySELECT \* FROM COUNTRY WHERE CCode = 'CA' ; retrieves all columns from the "COUNTRY" table where the value in the "CCode" column is equal to 'CA'.

* `SELECT \* instructs the database to select all columns from the specified table.
* FROM COUNTRY specifies the bale from which to retrieve the data, which is the 'COUNTRY' table.
* WHERE CCode = 'CA'; is a condition that filters the rows from the table. It will only return rows where the value in the "CCode" column is equal to 'CA.' In the table below, you will find that only the CA column was retrieved.

Response:

| **ID** | **Name** | **CCode** |
| --- | --- | --- |
| 9 | Canada | CA |

In the lab that follows later in the module, you will apply these concepts and practice more SELECT queries hands-on.

In this reading, you learned that:

* SELECT is a Database Query command that retrieves information from a database table.
* The SELECT statement has various forms depending on what action you require.
* The general syntax will retrieve the data under the listed columns from a named table.
* Use "\*" to retrieve all columns from a table without specifying individual column names.
* Use the WHERE clause to filter the data based on a predicate.

## **COUNT, DISTINCT, LIMIT**

* **COUNT**:
  + A built-in database function that retrieves the number of rows matching the query criteria.
  + Example: SELECT COUNT(\*) FROM tablename;
  + To count rows for a specific condition: SELECT COUNT(COUNTRY) FROM MEDALS WHERE COUNTRY='CANADA';
* **DISTINCT**:
  + Used to remove duplicate values from a result set.
  + Example: SELECT DISTINCT columnname FROM tablename;
  + To get unique countries that received gold medals: SELECT DISTINCT COUNTRY FROM MEDALS WHERE MEDALTYPE = 'GOLD';
* **LIMIT**:
  + Restricts the number of rows retrieved from the database.
  + Example: SELECT \* FROM tablename LIMIT 10;
  + To retrieve a few rows for a specific year: SELECT \* FROM MEDALS WHERE YEAR = 2018 LIMIT 5;

These expressions are useful for working with SELECT statements in SQL.

## **INSERT STATEMENT**

* **Purpose**: The INSERT statement is used to add new rows to a relational database table.
* **Syntax**:

INSERT INTO table\_name (column1, column2, ...) VALUES (value1, value2, ...);

* + **table\_name**: Identifies the table.
  + **column names**: Lists each column in the table.
  + **values**: Specifies the data values to be added.
* **Example**: To add a row for an author named Raul Chong:
* INSERT INTO author (author\_id, last\_name, first\_name, email, city, country)

VALUES ('A1', 'Chong', 'Raul', 'RFC@IBM.com', 'Toronto', 'CA');

* **Multiple Rows**: You can insert multiple rows at once by separating each row with a comma:
* INSERT INTO author (author\_id, last\_name, first\_name, email, city, country)
* VALUES ('A1', 'Chong', 'Raul', 'RFC@IBM.com', 'Toronto', 'CA'),

('A2', 'Ahuja', 'Rav', 'Rav@IBM.com', 'Toronto', 'CA');

* **Important Note**: The number of values must match the number of columns specified.

## **UPDATE and DELETE statements**

**UPDATE** and **DELETE** statements in SQL:

UPDATE Statement

* **Purpose**: Used to alter or modify data in a relational database table.
* **Syntax**:

UPDATE [TableName] SET [ColumnName] = [Value] WHERE [Condition]

* **Example**:
  + To change the first name and last name of the author with AUTHOR\_ID A2:

UPDATE AUTHOR SET LAST\_NAME = 'KATTA', FIRST\_NAME = 'LAKSHMI' WHERE AUTHOR\_ID = 'A2'

* **Important Note**: If the **WHERE** clause is omitted, all rows in the table will be updated.

DELETE Statement

* **Purpose**: Used to remove one or more rows from a table.
* **Syntax**:

DELETE FROM [TableName] WHERE [Condition]

* **Example**:
  + To delete rows for AUTHOR\_ID A2 and A3:

DELETE FROM AUTHOR WHERE AUTHOR\_ID IN ('A2', 'A3')

* **Important Note**: If the **WHERE** clause is omitted, all rows in the table will be removed.

Importance of the WHERE Clause

* The **WHERE** clause is crucial in both statements to specify which rows should be affected. Without it, unintended changes or deletions can occur.

## **Summary: Basic SQL**

Congratulations! You have completed this lesson. At this point in the course, you know:

* The Data Manipulation Language (DML) statements read and modify data.
* The search condition of the WHERE clause uses a predicate to refine the search.
* The SQL retrieves specific data from databases.
* The COUNT, DISTINCT, and LIMIT are expressions used with SELECT statements.
* The real-world applications of SELECT statements.
* The INSERT, UPDATE, and DELETE are DML statements for populating and changing tables.