**Course Assignment**

In this module, you will be working with multiple real-world datasets for the city of Chicago. You will be asked questions that will help you understand the data just as you would in the real world. You will be assessed on the correctness of your SQL queries and results.

**Learning Objectives**

* Discuss real-world datasets
* Analyze table details and properties
* Interpret and translate data analysis questions into SQL queries
* Run SQL queries on real world datasets

# **Assignment Preparation: Working with real-world data sets and built-in SQL functions**

## **Working with Real World Datasets**

1. **Data Formats in Databases**:
   * Real-world datasets are often available as **CSV files** (Comma-Separated Values).
   * CSV files contain data values separated by commas, and sometimes other separators like semicolons.
2. **Example Dataset**:
   * The video uses a fictional dataset called **DOGS.csv**.
   * Sample contents:
     + **ID**: 1, 2, 3
     + **Name**: Wolfie, Fluffy, Huggy
     + **Breed**: German, Shepherd, Pomeranian, Labrador
3. **Loading Data into Databases**:
   * When using **phpMyAdmin** to import a CSV file:
     + Select **CSV** from the format dropdown.
     + Check the box indicating that the first line contains column names.
   * MySQL will auto-detect header names as column names.
4. **Querying Columns**:
   * Use **backticks** around column names if they contain spaces or special characters.
   * Example:

SELECT `column name` FROM `table name`;

1. **Long Queries**:
   * For improved readability, split long queries into multiple lines.
   * In Python notebooks, use a backslash (\) to indicate continuation:
   * query = "SELECT \* FROM table\_name " \

"WHERE condition;"

1. **Using Pandas to Query**:
   * Use the read\_sql method from the Pandas library to query tables:
   * import pandas as pd

df = pd.read\_sql("SELECT \* FROM table\_name", connection)

1. **Handling Quotes in Queries**:
   * Use single quotes for the Python variable and double quotes for SQL column names:

query\_statement = 'SELECT "column name" FROM table\_name WHERE "column" = \'value\';'

1. **Limiting Results**:
   * Use the **LIMIT** clause to restrict the number of rows retrieved:

SELECT \* FROM census\_data LIMIT 3;

Summary:

* **CSV files** are a common format for datasets.
* Use **phpMyAdmin** to load data into databases.
* Use **backticks** for querying columns with special characters.
* Split long queries for readability and use **Pandas** for querying in Python.
* The **LIMIT** clause helps in retrieving a specific number of rows.

## **Getting Tables and Column Details**

1. **Getting a List of Tables**:
   * Databases may contain multiple tables, and you might not remember their names.
   * You can query system or catalog tables to get a list of tables.
2. **Catalog Tables in Different Database Systems**:
   * **DB2**: SYSCAT.TABLES
   * **SQL Server**: information\_schema.tables
   * **SQLite3**: sqlite\_master
   * **MySQL**: SHOW TABLES
3. **Query to Get Tables in SQLite3**:

SELECT name FROM sqlite\_master WHERE type='table';

1. **Query to Get Tables in MySQL**:

SHOW TABLES;

1. **Extracting Column Information**:
   * To get the attributes or column headers of a table, you can use specific commands based on the database system.
2. **Commands for Different Database Systems**:
   * **SQLite3**:

PRAGMA table\_info(table\_name);

* + **MySQL**:

DESCRIBE table\_name;

Summary of Commands:

* **List Tables in SQLite3**:

SELECT name FROM sqlite\_master WHERE type='table';

* **Get Column Information in SQLite3**:

PRAGMA table\_info(table\_name);

* **List Tables in MySQL**:

SHOW TABLES;

* **Get Column Information in MySQL**:

DESCRIBE table\_name;