Chapter 7: Pandasql – Run SQL Queries in Python

Introduction to SQL:

SQL stands for Structured Query Language. This is the database language designed for maintaining the data in relational database management systems.

It is a special tool used by data professionals for handling structured data. We can easily create, manipulate, access and modify the tables.

SQL Commands:

The SQL commands help in creating and managing the database. In SQL we have commands like:

- DDL Commands
- DML Commands
- DCL Commands
- DQL Commands
- TCL Commands

1. DDL Commands – DDL Commands stands for Data Definition language. These commands are responsible for creating, altering and deleting the data.
   a. Create – This command is used to create a table in sql database.
      Syntax: CREATE TABLE TABLENAME;
   b. Drop – This is used to remove the complete data of a table along with the whole structure or definition permanently from the database.
      Syntax: DROP TABLE TABLENAME;
   c. Alter - MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.
      Syntax: ALTER TABLE tablename ADD newcolumn column_definition [FIRST | AFTER column_name];
   d. Rename: This is used to rename an object existing in the database.
   e. Truncate: This is used to remove all records from a table, including all spaces allocated for the records are removed.

2. DQL Command – It stands for Data Query Language. This command allows getting the data out of the database to perform operations with it.

   SELECT – It is used to retrieve data from database.

   Syntax – SELECT COL1, COL2, COL2…. FROM TABLENAME;

3. DML Commands – It stands for Data Manipulation Language. The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database.

The commands are:
INSERT: It is used to insert data into a table.
UPDATE: It is used to update existing data within a table.
DELETE: It is used to delete records from a database table.

4. DCL Commands – It is knowns as Data Control Language. This includes commands such as GRANT, REVOKE which mainly deal with the rights, permissions and other controls of the database systems.

5. TCL Commands – TCL (Transaction Control Language) commands are used to control the execution of a transaction.

   COMMIT: Commits a Transaction.

   ROLLBACK: Rollbacks a transaction in case of any error occurs.

DATA Types:

Many data types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, FLOAT, DOUBLE, CHAR, VARCHAR, BINARY, VARBINARY, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM.

- Fixed-length and variable-length string types.

FUNCTIONS: Full operator and function support in the SELECT list and WHERE clause of queries.

```mysql
SELECT CONCAT(first_name, ' ', last_name)
FROM citizen
WHERE income/dependents > 10000 AND age > 30;
```

- Full support for SQL GROUP BY and ORDER BY clauses. Support for group functions (COUNT(), AVG(), STD(), SUM(), MAX(), MIN(), and GROUP_CONCAT()).

- Support for LEFT OUTER JOIN and RIGHT OUTER JOIN with both standard SQL and ODBC syntax.

- Support for aliases on tables and columns as required by standard SQL.

- Support for DELETE, INSERT, REPLACE, and UPDATE to return the number of rows that were changed (affected), or to return the number of rows matched instead by setting a flag when connecting to the server.

- Support for MySQL-specific SHOW statements that retrieve information about databases, storage engines, tables, and indexes. Support for the INFORMATION_SCHEMA database, implemented according to standard SQL.

- An EXPLAIN statement to show how the optimizer resolves a query.

Introduction to Pandasql:

Pandasql is an open source library in python. Pandasql allows you to query pandas DataFrames using SQL syntax. It works similarly to sqldf in R. Pandasql seeks to provide a more familiar way of manipulating and cleaning data for people new to Python or pandas.

Installing Pandasql:

If you are already using python use, !pip install pandasql/ %pip install pandasql. As this is the easiest way to get started.

The main function used in pandasql is sqldf.

Importing sqldf: from pandasql import sqldf.

sqldf takes 2 parameters, out of which one is completely optional (in fact I never used it). So, the important and only parameter is a SQL query string.

With its syntax sqldf (SQL query) sqldf gives a pandas Data Frame as output. We can query any pandas DataFrame using SQL in the same way as we extract data from any table using SQL.

pandasql uses SQLite syntax!

Querying the Pandas dataframe using Pandasql:

let's start by creating the datasets StudentTable and TeachingAssistantTable that will be used for hands-on practice.

1.First we have to import Pandas library.

```
import pandas as pd
```

2. Create the Student Dataframe:

```
students= {
  'Students':["Sira","Ibrahim","Moussa","Mamadou","Nabintou"],
  'Gender':['Female','Male','Male', "Male", "Female"],
  'Age':[18, 27, 19, 22, 21],
  'Email': ["sira@info.com", "ib@info.com", "mouss@info.com",
        "mam@info.com", "nab@info.com"]
```

```
    }
students_df = pd.DataFrame(students)
students_df
```

OUTPUT:

	Students	Gender	Age	Email
0	Sira	Female	18	sira@info.com
1	Ibrahim	Male	27	ib@info.com
2	Moussa	Male	19	mouss@info.com
3	Mamadou	Male	22	mam@info.com
4	Nabintou	Female	21	nab@info.com

3. Creating the Teaching Assistant Data Frame:

```
teaching_assistant= {
'Teacher':["Ibrahim","Nabintou","Mamadou","Fatim","Aziz"],
'Email':['ib@info.com','nab@info.com','mam@info.com',
        "fat@info.com", "aziz@info.com"],
'Degree':["M.S in Data Science", "B.S in Statistics",
         "B. Comp Sc", "M.S. Architecture", "B.S in Accounting"],
'Department': ["Business", "Statistics", "Comp Sc",
        "Engineering", "Business"]
    }
teaching_assistant_df = pd.DataFrame(teaching_assistant)
teaching_assistant_df
```

OUTPUT:

Teacher	Email	Degree	Department	
0	Ibrahim	ib@info.com	M.S in Data Science	Business

1	Nabintou	nab@info.com	B.S in Statistics	Statistics
2	Mamadou	mam@info.com	B. Comp Sc	Comp Sc
3	Fatim	fat@info.com	M.S. Architecture	Engineering
4	Aziz	aziz@info.com	B.S in Accounting	Business

After creating above 2 Dataframes we are going to learn the main concepts that will be covered in here:

a. Column Selection
b. Data Filtering
c. Data Aggregation
d. Data Joining

Column Selection:

his corresponds to selecting part or all the columns of your database/data frame. It is performed with the keyword

SELECT col_1, col_2, ... col_X FROM tableName

→ 1, 2, ...X correspond to the columns you are interested in having in the final result.

→ tableName is the name of the dataframe/table.

QUERY:

```
all_students = sqldf("SELECT * FROM students_df")

print(all_students)
```

OUTPUT:

	Students	Gender	Age	Email
0	Sira	Female	18	sira@info.com
1	Ibrahim	Male	27	ib@info.com
2	Moussa	Male	19	mouss@info.com
3	Mamadou	Male	22	mam@info.com
4	Nabintou	Female	21	nab@info.com

Scenario: If we want to fetch out only Students and Gender column below is the query given

```
all_students = sqldf("SELECT Students, Gender FROM students_df")

print(all_students)
```

OUTPUT:

```
  Students    Gender
0      Sira       Female
1   Ibrahim     Male
2    Moussa     Male
3   Mamadou   Male
4   Nabintou   Female
```

```
# Check the type of all_students

print(type(all_students))

# Run Pandas Statement to show the type of the columns

print("---"*10)

print(all_students.dtypes)
```

OUTPUT:

```
<class
'pandas.core.frame.DataFrame'>
------------------------------
Students    object
Gender      object
Age         int64
Email       object
dtype:      object
```

Define the query as a string. When doing so, make sure to use the triple quote sign”””so that you can write the query on multiple lines.

Apply the sqldf function to the query to get the result.

Let's say we want the student's name, their email and limit the result to the first 3.

Defining Query:

```
query = """ SELECT Students, Email
        FROM students_df
        LIMIT 3
    """
```

Query execution

```
name_email = sqldf(query)
name_email
```

OUTPUT:

	Students	Email
0	Sira	sira@info.com
1	Ibrahim	ib@info.com
2	Moussa	mouss@info.com

Data Filtering:

Data filtering is when the WHERE statement comes into the equation to perform custom filtering on the result of the SELECT statement.

Let's do the query for fetching out all the female students

Define the query

```
query = """SELECT *
        FROM students_df
        WHERE Gender = 'Female'
    """

# Execute the query
female_student = sqldf(query)
```

female_student

Let's have a look at this query that aims to retrieve the Name, Email, and Degree of all the Teaching Assistants with a Master's Degree

```
query = """ SELECT Teacher, Email, Degree
        FROM teaching_assistant_df
        WHERE Degree LIKE 'M.S%'
        """
ms_students = sqldf(query)
ms_students
```

OUTPUT:

	Teacher	Email	Degree
0	Ibrahim	ib@info.com	M.S in Data Science
1	Fatim	fat@info.com	M.S. Architecture

Let's perform the same request using Pandas Statement:

```
cols_to_get = ['Teacher', 'Email', 'Degree']
teaching_assistant_df[teaching_assistant_df.Degree.str.startswith('M.S')][cols_to_get]
```

OUTPUT:

Teacher	Email	Degree
0 Ibrahim	ib@info.com	M.S in Data Science
3 Fatim	fat@info.com	M.S. Architecture

Data Aggregation:

Aggregation in SQL is performed with the help of aggregation functions, and these are some of the most commonly used: COUNT, SUM, MAX & MIN, and AVG. For instance, you can get the age of students, based on their gender.

QUERY:

```
query = """ SELECT AVG(Age) as Average_Age, Gender
        FROM students_df
```

```
    GROUP BY Gender
"""
avg_age = sqldf(query)
avg_age
```

OUTPUT:

	Average_Age	Gender
0	19.500000	Female
1	22.666667	Male

Data Joining:

The join concept becomes useful when we want to combine data from at least two tables. This section goes straight to the point with only one type of join.

Imagine you want to know who is both a student and also a teaching assistant. The answer to this requires joining our two tables as follows, using an INNER JOIN and the final result contains the following information:

Student Name, Gender, Email, and Age from the students_df table.

Department from the teaching_assistant_df dataframe.

```
query = """ SELECT st.Students, st.Gender, st.Email, st.Age, tat.Department
    FROM students_df st INNER JOIN teaching_assistant_df tat
    ON st.Email = tat.Email;
    """
result = sqldf(query)
result
```

OUTPUT:

	Students	Gender	Email	Age	Department
0	Ibrahim	Male	ib@info.com	27	Business
1	Mamadou	Male	mam@info.com	22	Comp Sc
2	Nabintou	Female	nab@info.com	21	Statistics

Limitations of Pandasql:

As Pandasql uses SQLite, it is subjected to all the limitations of SQLite. For example, SQLite does not implement right outer join or full outer join.

Pandasql performs query only, it cannot perform SQL operations such as update, insert or alter tables.