Lecture 9 Part 2: Automate SQLs

Introduction

This lecture demonstrates how to use **Azure Data Studio's Notebook feature** with Python to:

- Automatically create a SQL table from a dataset.
- Load a CSV file directly into SQL Server.
- Perform queries on the imported data.

We will use the **Air Quality Dataset (AirQualityUCI.csv)** for this example.

Setting Up the Notebook

To create a Python Notebook in **Azure Data Studio**:

1. Open Azure Data Studio.
2. Click "File" → "New Notebook".
3. In the first cell, select "Python" as the kernel.

Installing Required Libraries

In the first cell of the notebook, install the necessary Python libraries:

```
!pip install pandas pyodbc
```

Explanation:

- `pandas` is used to read the CSV file.
- `pyodbc` is used to connect to SQL Server.

Connecting to SQL Server

In the next cell, establish a connection to SQL Server:

```
import pandas as pd
import pyodbc

# Establish connection to SQL Server
conn = pyodbc.connect(
    'DRIVER={ODBC Driver 17 for SQL Server};'
    'SERVER=localhost;'  # Change this if using a different
       server
    'DATABASE=FRE521D;'  # Use your actual database
    'Trusted_Connection=yes;'
)

cursor = conn.cursor()
```

Explanation:

- Establishes a connection to SQL Server.
- Uses `Trusted_Connection=yes;` to authenticate with Windows credentials.

Loading the CSV File

Next, we load the dataset into Pandas:

```
# Load CSV file
df = pd.read_csv("AirQualityUCI.csv", delimiter=";")

# Display first few rows
df.head()
```

Explanation:

- Reads the CSV file into a Pandas DataFrame.
- Uses `delimiter=";"` since the dataset is semicolon-separated.

Creating a Table Automatically

Now, we generate the SQL table dynamically based on the CSV column names and data types:

```
# Function to map Pandas data types to SQL Server data types
def map_dtype(dtype):
    if "int" in str(dtype):
        return "INT"
    elif "float" in str(dtype):
        return "FLOAT"
    else:
        return "VARCHAR(255)"

# Generate CREATE TABLE query dynamically
table_name = "AirQuality"
columns = ", ".join([f"{col} {map_dtype(dtype)}" for col, dtype
   in zip(df.columns, df.dtypes)])

create_table_query = f"CREATE TABLE {table_name} ({columns});"

# Execute the CREATE TABLE statement
cursor.execute(create_table_query)
conn.commit()
```

Explanation:

- Dynamically generates a SQL table using column names and data types from the dataset.
- Uses `VARCHAR(255)` for text, `INT` for integers, and `FLOAT` for decimals.

Inserting Data into SQL Server

Now, we insert the data from Pandas into the SQL table:

```
# Insert data into SQL table
for index, row in df.iterrows():
    placeholders = ", ".join(["?"] * len(row))
    insert_query = f"INSERT INTO {table_name} VALUES ({
       placeholders})"

    cursor.execute(insert_query, tuple(row))

conn.commit()
```

Explanation:

- Iterates through the DataFrame rows and inserts them into SQL Server.
- Uses parameterized queries (‘¿) to prevent SQL injection.

Querying the Imported Data

After inserting data, verify it using:

```
# Query the table to check inserted data
query = f"SELECT TOP 5 * FROM {table_name};"
df_sql = pd.read_sql(query, conn)
df_sql.head()
```

Explanation:

- Fetches the top 5 rows from the SQL table to confirm successful import.

Conclusion

In this lecture, we:

- Created an **Azure Data Studio Notebook** with Python.
- Connected to **SQL Server** using `pyodbc`.
- Loaded **AirQualityUCI.csv** into Pandas.
- **Automatically generated** a SQL table based on dataset columns.
- Inserted data from Pandas into SQL Server.
- Queried the table to verify successful data import.