**GitHub Copilot**

Used 1 reference

To write a SQL parser for validating Databricks Delta tables and test it on your local machine, you can use Python with the sqlparse library for parsing SQL and the databricks-cli for interacting with Databricks. Here is a step-by-step guide:

**Step 1: Install Required Libraries**

First, install the required libraries using pip:

pip install sqlparse databricks-cli

**Step 2: Write the SQL Parser**

Create a Python script to parse and validate SQL queries for Databricks Delta tables.

import sqlparse

from databricks\_cli.sdk import ApiClient

from databricks\_cli.clusters.api import ClusterApi

from databricks\_cli.dbfs.api import DbfsApi

from databricks\_cli.workspace.api import WorkspaceApi

# Initialize Databricks API client

def init\_databricks\_client(host, token):

    client = ApiClient(host=host, token=token)

    return client

# Validate SQL query

def validate\_sql\_query(query):

    parsed = sqlparse.parse(query)

    for statement in parsed:

        if statement.get\_type() != 'UNKNOWN':

            print(f"Valid SQL statement: {statement}")

        else:

            print(f"Invalid SQL statement: {statement}")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    # Replace with your Databricks host and token

    DATABRICKS\_HOST = 'https://<databricks-instance>.<region>.databricks.com'

    DATABRICKS\_TOKEN = 'your-databricks-token'

    client = init\_databricks\_client(DATABRICKS\_HOST, DATABRICKS\_TOKEN)

    # Example SQL query

    sql\_query = """

    SELECT \* FROM delta.`/path/to/delta/table`

    """

    validate\_sql\_query(sql\_query)

1 vulnerability

**Step 3: Test the Script Locally**

Run the script on your local machine to test the SQL parser and validation.

python sql\_parser.py

**Step 4: Write Unit Tests**

Create a separate file for unit tests using the unittest framework.

import unittest

from sql\_parser import validate\_sql\_query

class TestSQLParser(unittest.TestCase):

    def test\_valid\_sql(self):

        query = "SELECT \* FROM delta.`/path/to/delta/table`"

        self.assertIsNone(validate\_sql\_query(query))

    def test\_invalid\_sql(self):

        query = "INVALID SQL"

        self.assertIsNone(validate\_sql\_query(query))

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

**Step 5: Run Unit Tests**

Run the unit tests to ensure your SQL parser works correctly.

python -m unittest test\_sql\_parser.py

**Explanation:**

1. **Install Libraries**: Install sqlparse for SQL parsing and databricks-cli for interacting with Databricks.
2. **SQL Parser Script**: Write a script to parse and validate SQL queries.
3. **Test Locally**: Run the script locally to test the SQL parser.
4. **Unit Tests**: Write unit tests to validate the functionality of the SQL parser.
5. **Run Tests**: Execute the unit tests to ensure everything works as expected.

By following these steps, you can create a SQL parser for validating Databricks Delta tables and test it on your local machine.

To extend the script to validate SQL queries using PySpark commands on Databricks, you can use the Databricks REST API to submit a job that runs the validation on a Databricks cluster. Here is a step-by-step guide:

**Step 1: Install Required Libraries**

First, install the required libraries using pip:

pip install sqlparse databricks-cli requests

**Step 2: Write the SQL Parser and Validator**

Create a Python script to parse and validate SQL queries for Databricks Delta tables using PySpark.

import sqlparse

import requests

import json

from databricks\_cli.sdk import ApiClient

# Initialize Databricks API client

def init\_databricks\_client(host, token):

    client = ApiClient(host=host, token=token)

    return client

# Validate SQL query using sqlparse

def validate\_sql\_query(query):

    parsed = sqlparse.parse(query)

    for statement in parsed:

        if statement.get\_type() != 'UNKNOWN':

            print(f"Valid SQL statement: {statement}")

        else:

            print(f"Invalid SQL statement: {statement}")

# Submit a job to Databricks to validate SQL using PySpark

def validate\_sql\_with\_pyspark(client, cluster\_id, query):

    job\_payload = {

        "run\_name": "SQL Validation Job",

        "existing\_cluster\_id": cluster\_id,

        "libraries": [],

        "spark\_python\_task": {

            "python\_file": "dbfs:/path/to/validate\_sql.py",

            "parameters": [query]

        }

    }

    response = client.perform\_query("POST", "/jobs/runs/submit", data=json.dumps(job\_payload))

    return response

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    # Replace with your Databricks host and token

    DATABRICKS\_HOST = 'https://<databricks-instance>.<region>.databricks.com'

    DATABRICKS\_TOKEN = 'your-databricks-token'

    CLUSTER\_ID = 'your-cluster-id'

    client = init\_databricks\_client(DATABRICKS\_HOST, DATABRICKS\_TOKEN)

    # Example SQL query

    sql\_query = """

    SELECT \* FROM delta.`/path/to/delta/table`

    """

    # Validate SQL locally

    validate\_sql\_query(sql\_query)

    # Validate SQL using PySpark on Databricks

    response = validate\_sql\_with\_pyspark(client, CLUSTER\_ID, sql\_query)

    print(response)

1 vulnerability

**Step 3: Create the PySpark Validation Script**

Create a PySpark script (validate\_sql.py) that will be uploaded to DBFS and executed on the Databricks cluster.

from pyspark.sql import SparkSession

def validate\_sql(query):

    spark = SparkSession.builder.appName("SQL Validator").getOrCreate()

    try:

        spark.sql(query)

        print("SQL query is valid.")

    except Exception as e:

        print(f"SQL query is invalid: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

    import sys

    query = sys.argv[1]

    validate\_sql(query)

**Step 4: Upload the PySpark Script to DBFS**

Use the Databricks CLI to upload the PySpark script to DBFS.

databricks fs cp validate\_sql.py dbfs:/path/to/validate\_sql.py

**Step 5: Test the Script Locally**

Run the main script on your local machine to test the SQL parser and validation.

python sql\_parser.py

**Explanation:**

1. **Install Libraries**: Install sqlparse, databricks-cli, and requests.
2. **SQL Parser Script**: Write a script to parse and validate SQL queries locally and using PySpark on Databricks.
3. **PySpark Validation Script**: Create a PySpark script to validate SQL queries on a Databricks cluster.
4. **Upload Script to DBFS**: Use the Databricks CLI to upload the PySpark script to DBFS.
5. **Test Locally**: Run the main script locally to test the SQL parser and validation.

By following these steps, you can create a script to validate SQL queries for Databricks Delta tables using both local parsing and PySpark commands on Databricks, and test it on your local machine.