**Ex No: 7 **simulate how Hadoop components HDFS, Sqoop, and Hive work together** for data movement and analysis.**

## AIM:

To implement Pig Latin scripts to load, filter, project, group, sort, and join datasets using Apache Pig.

## Algorithm :

**Establish Connection** – Connect Sqoop with the MySQL database using JDBC URL, username, and password.

**Specify Table** – Use --table option to select the target table for import.

**Define Target Directory** – Provide HDFS destination path using --target-dir to store imported data.

**Select Columns** – Optionally specify required columns using --columns.

**Enable Parallelism** – Use --num-mappers to define parallel map tasks for faster data transfer.

**Handle Primary Key** – Specify --split-by column to ensure balanced data distribution across mappers.

**Import Data** – Execute the Sqoop import command to transfer data from MySQL to HDFS.

**Verify Transfer** – Use Hadoop fs -ls or fs -cat commands to confirm data import in HDFS.

## Python Implementation

import pandas as pd

import sqlite3

import random

from contextlib import contextmanager

# Step 1: Generate sample weather data (simulating HDFS CSV file)

def generate\_sample\_data(num\_records=1000):

years = list(range(1900, 2021))

data = {

'record\_id': range(1, num\_records + 1),

'year': [random.choice(years) for \_ in range(num\_records)],

'temperature\_c': [random.uniform(-50, 50) for \_ in range(num\_records)]

}

df = pd.DataFrame(data)

# Save to CSV (simulating HDFS file)

csv\_path = '/content/weather\_data.csv'

df.to\_csv(csv\_path, index=False)

print(f"Sample data generated and saved to {csv\_path} (simulating HDFS file).")

return csv\_path

# Step 2: Simulate SQLite connection for Hive-like table

@contextmanager

def sqlite\_connection(db\_name):

conn = sqlite3.connect(db\_name)

try:

yield conn

finally:

conn.close()

# Step 3: Simulate Sqoop export/import (CSV to SQLite)

def sqoop\_like\_import(csv\_path, db\_name, table\_name):

# Read CSV (simulating Sqoop export from HDFS)

df = pd.read\_csv(csv\_path)

print(f"Sqoop-like export: Read {len(df)} records from {csv\_path} (HDFS).")

# Import to SQLite (simulating Hive table)

with sqlite\_connection(db\_name) as conn:

df.to\_sql(table\_name, conn, if\_exists='replace', index=False)

print(f"Sqoop-like import: Loaded data into {db\_name}.{table\_name} (Hive table).")

# Create index (optional, for performance, mimicking Hive index)

conn.execute(f'CREATE INDEX idx\_year ON {table\_name}(year)')

print(f"Index 'idx\_year' created on {table\_name}.year.")

# Step 4: Generate weather report from SQLite (Hive-like query)

def generate\_weather\_report(db\_name, table\_name):

with sqlite\_connection(db\_name) as conn:

query = f'''

SELECT year,

MIN(temperature\_c) AS min\_temp\_c,

MAX(temperature\_c) AS max\_temp\_c

FROM {table\_name}

GROUP BY year

ORDER BY year

'''

report\_df = pd.read\_sql\_query(query, conn)

report\_df['min\_temp\_c'] = report\_df['min\_temp\_c'].round(1)

report\_df['max\_temp\_c'] = report\_df['max\_temp\_c'].round(1)

return report\_df

# Step 5: Run the POC

if \_\_name\_\_ == "\_\_main\_\_":

print("=== Simulating Sqoop Export/Import to Hive ===")

# Generate sample data (HDFS-like CSV)

csv\_path = generate\_sample\_data(1000)

# Simulate Sqoop import to Hive

db\_name = 'weather\_hive.db'

table\_name = 'weather\_data'

sqoop\_like\_import(csv\_path, db\_name, table\_name)

# Generate report to verify data

print("\nGenerating Weather Temperature Statistics Report...")

report = generate\_weather\_report(db\_name, table\_name)

print("\n=== Weather Report ===")

print("Year\tMin Temp (°C)\tMax Temp (°C)")

print("-" \* 35)

for \_, row in report.iterrows():

print(f"{int(row['year'])}\t{row['min\_temp\_c']}\t\t{row['max\_temp\_c']}")

print(f"\nSample data from {table\_name} (first 5 rows):")

with sqlite\_connection(db\_name) as conn:

sample\_data = pd.read\_sql\_query(f'SELECT \* FROM {table\_name} LIMIT 5', conn)

print(sample\_data)

# Expected Output:

=== Weather Report ===

Year Min Temp (°C) Max Temp (°C)

-----------------------------------

1900 -42.2 32.2

1901 -18.5 48.6

1902 -46.4 26.2

1903 -38.9 34.3

.

.

.

2017 -40.8 37.1

2018 -24.1 46.8

2019 -43.1 46.8

2020 -47.4 47.7

Sample data from weather\_data (first 5 rows):

record\_id year temperature\_c

0 1 1936 40.403910

1 2 1937 -16.953221

2 3 1907 21.564613

3 4 1966 -23.531338

4 5 1935 -23.215985

# Result:

Thus, Python program is used to **simulate how Hadoop components — HDFS, Sqoop, and Hive — work together** for data movement and analysis, but **entirely within a local Python environment** using **Pandas and SQLite**.