BIG DATA PROJECT-3
BUAN 6346.504

Group members:

Barath Kumar Dhnasekar-bxd220033

Jagadeep Nandagopal-jxn220044

Rahul Sadineni-rxs230051

Kosuri Durga Sravya-dxk220083

Nikitha Masineni-nxm230033

Navya Sahithi Surapaneni-nxs230031

Step 1: Data Analysis Using Spark Core

A. Blockchain Data Analysis – Part 1

```
import json

file_path = "hdfs://localhost/loudacre/blockchain_data.txt"
json_string = sc.textFile(file_path).reduce(lambda x, y: x + y)
data = json.loads(json_string)
```

1) How many total blocks are there in your dataset?

```
total_blocks = len(data)
print("Total number of blocks:", total_blocks)
```

```
('Total number of blocks:', 1027)
```

2) What is the largest block height among the blocks in your dataset?

```
largest_block_height = max(block['height'] for block in data)
print("Largest block height:", largest_block_height)
```

```
('Largest block height:',835509)
```

3) What is the date and time for that block

```
block_with_largest_height = next((block for block in data if
block.get('height') == largest_block_height), None)

if block_with_largest_height:
    time_largest_block = block_with_largest_height.get('time')
    print("Time for the largest block:", time_largest_block)
```

```
('Time for the largest block:', 2024-04-24T22:04:56.000-08:00)
```

4) What is the highest number of transactions in your blocks?

```
highest_transactions = 0

for block in data:
    num_transactions = block.get('n_tx', 0)
    if num_transactions > highest_transactions:
        highest_transactions = num_transactions

print("Highest number of transactions in a block:",
highest_transactions)
```

```
('Highest number of transactions in a block:', 2141853579)
```

B. Stock Market Data Analysis

```
hdfs_folder_path = "hdfs://localhost:/flume/data/project/"
data = sc.wholeTextFiles(hdfs_folder_path)
split = data.flatMap(lambda x: __import__('re').split(r'\r\n|\n',
x[1]))
header = split.first()
RDD = split.filter(lambda line: line != header and line != "")
```

1) How many records are there in the table?

```
total_records = RDD.count()
print("Total number of records:", total_records)
```

```
('Total number of records:', 9848)
```

2) How many different days are there in the table?

```
distinct_days = RDD.map(lambda x: x.split(',')[1]).distinct().count()
print("Total number of different days:", distinct_days)
```

```
('Total number of different days:', 5)
```

3) How many records per each day are there in the table?

```
records_per_day = RDD.map(lambda x: (x.split(',')[1],
1)).reduceByKey(lambda a, b: a + b)
print("Records per each day:")
for record in records_per_day.collect():
  print(record)
```

```
Records per each day:
(u'2024-03-18',  1950)
(u'2024-03-19',  1950)
(u'2024-03-20',  1950)
(u'2024-03-21',  1950)
(u'2024-03-22',  1950)
```

4) What are the symbols in the table?

```
symbols = RDD.map(lambda x: x.split(',')[0]).distinct().collect()
print("Symbols in the table:", symbols)
```

```
('Symbols in the table:', [u'MSFT', u'APPL', u'GOOGL', u'TSLA',
u'IBM'])
```

5) What is the highest price for each symbol?

```
highest_price_per_symbol = RDD.map(lambda x: (x.split(',')[0],
float(x.split(',')[4]))).reduceByKey(max)
print("Highest price for each symbol:")
for record in highest_price_per_symbol.collect():
    print(record)
```

```
Highest price for each symbol:
(u'MSFT', 430.82)
(u'APPL', 178.67)
(u'GOOGL', 152.15)
(u'TSLA', 178.18)
(u'IBM', 193.98)
```

6)What is the lowest price for each symbol?

```
lowest_price_per_symbol = RDD.map(lambda x: (x.split(',')[0],
float(x.split(',')[5]))).reduceByKey(min)
print("Lowest price for each symbol:")
for record in lowest_price_per_symbol.collect():
    print(record)
```

```
Lowest price for each symbol:
(u'MSFT', 415.571)
(u'APPL', 170.268)
(u'GOOGL', 146.17)
(u'TSLA', 166.3)
(u'IBM', 190.31)
```

7)What is the average price for each symbol?

```
avg_price_per_symbol = RDD.map(lambda line: (line.split(',')[0],
(float(line.split(',')[6]), 1))) \
    .reduceByKey(lambda a, b: (a[0] + b[0], a[1] +
b[1])).mapValues(lambda x: x[0] / x[1])
print("Average price for each symbol:")
print(avg_price_per_symbol.collect())
```

```
Average price for each symbol:
[(u'MSFT', 423.74882615), (u'APPL',174.54863282),
(u'GOOGL', 148.42665282), (u'TSLA',172.12119128),
(u'IBM', 192.19748359)]
```

8)What is the range of price for each symbol?

```
range_price_per_symbol = RDD.map(lambda line: (line.split(',')[0],
float(line.split(',')[6]))) \
    .groupByKey().mapValues(lambda x: (min(x), max(x)))
print("Range of price for each symbol:")
print(range_price_per_symbol.collect())
```

```
Range of price for each symbol:
[(u'MSFT', (17.04)), (u'APPL', (8.61)),
(u'GOOGL', (6.07)), (u'TSLA', (12.28)), (u'IBM',
(3.97))]
```

9)What is the date on which each symbol experienced the highest price?

```
symbol_date_price_rdd = RDD.map(lambda line: (line.split(',')[0],
(line.split(',')[1], float(line.split(',')[4]))))
max_price_per_symbol = symbol_date_price_rdd.reduceByKey(lambda a, b:
a if a[1] > b[1] else b)

print("Date on which each symbol experienced the highest price:")
print(max_price_per_symbol.collect())
```

```
Date on which each symbol experienced the highest price:
[(u'MSFT', (u'2024-03-21', 430.82)), (u'APPL', (u'2024-03-20',
178.67)), (u'GOOGL', (u'2024-03-18', 152.15)), (u'TSLA', (u'2024-03-
21',178.18)), (u'IBM', (u'2024-03-20', 193.98))]
```

C. Blockchain Data Analysis – Part 2

```
hdfs_path = "/user/hive/warehouse/project.db"

blocks_rdd = sc.textFile(hdfs_path + "/blocks_2024_mar_18_to_22")
blocks_info_rdd = sc.textFile(hdfs_path +
"/blocks_info_2024_mar_18_to_22")
tx_info_rdd = sc.textFile(hdfs_path + "/tx_info_2024_mar_18_to_22")
```

1) How many total blocks are there in your blocks table?

```
total_blocks = blocks_rdd.count()
print("Total number of blocks:", total_blocks)
```

```
('Total number of blocks:', 920)
```

2) What is the largest block height among the blocks in your blocks table?

```
largest_block_height = blocks_info_rdd.map(lambda x:
int(x.split('\x01')[9])).max()

print("Largest block height:", largest_block_height)
```

```
('Largest block height:', 807751)
```

3) What is the date and time for that block?

```
largest_block_row = blocks_info_rdd.filter(lambda x:
int(x.split('\x01')[9]) == largest_block_height).first()
largest_height_hash = largest_block_row.split('\x01')[1]
print("largest_height_hash:", largest_height_hash)

matching_row = blocks_rdd.filter(lambda x: x.split('\x01')[1] ==
largest_height_hash).first()
timestamp = matching_row.split('\x01')[2]

print("Timestamp for the block with the largest height:", timestamp)
```

```
('largest_height_hash:',
u'00000000000000000010862f6ef5f47744ead8de2847bf3bb64e072c1fd746c')
('Timestamp for the block with the largest height:', 2024-04-24T22:04:56.000-
08:00)
```

4) What is the largest number of transactions in your blocks?

```
block_hashes = tx_info_rdd.map(lambda row: (row.split('\x01')[2], 1))
```

```
block_transaction_counts = block_hashes.groupByKey().mapValues(len)

largest_block = block_transaction_counts.top(1, key=lambda x: x[1])

print("Block with the largest number of transactions:")
print("Block Hash:", largest_block[0][0])
print("Number of Transactions:", largest_block[0][1])
```

```
Block with the largest number of transactions:
('Block Hash:',
u'00000000000000000002dbf9d0bc1c743ac17bdb60d5c6abc8cd94f2d253621d')
('Number of Transactions:', 2141853579)
```

Step 3: Data Analysis Using Spark SQL

A. Blockchain Data Analysis – Part 1

1) How many total blocks are there in your dataset?

```
from pyspark.sql import SQLContext

sqlContext = SQLContext(sc)

blocks_df = sqlContext.createDataFrame(data)

blocks_df.registerTempTable("blocks")

total_blocks = sqlContext.sql("SELECT COUNT(*) as total_blocks FROM
blocks").first().total_blocks
print("Total Blocks: {}".format(total_blocks))
```

```
Total Blocks: 1027
```

2) What is the largest block height among the blocks in your dataset?

```
max_block_height = sqlContext.sql("SELECT MAX(height) as
max_block_height FROM blocks").first().max_block_height
print("Largest Block Height: {}".format(max_block_height))
```

```
Largest Block Height: 835509
```

3) What is the date and time for that block?

```
block_info = sqlContext.sql("SELECT time FROM blocks WHERE height =
'{}'".format(max_block_height)).first()
timestamp_value = block_info.time
print("Time for the largest block: {}".format(timestamp_value))
```

```
Time for the largest block: 2024-04-24T22:04:56.000-08:00
```

4)What is the highest number of transactions in your blocks?

```
max_transactions = sqlContext.sql("SELECT MAX(n_tx) as
max_transactions FROM blocks").first().max_transactions
print("Highest Number of Transactions: {}".format(max_transactions))
```

```
Highest Number of Transactions: 2141853579
```

B. Stock Market Data Analysis

```
from pyspark.sql import Row
def parse_line(line):
    parts = line.split(',')
    return Row(symbol=parts[0], date=parts[1], time=parts[2],
               open=float(parts[3]), high=float(parts[4]),
               low=float(parts[5]), close=float(parts[6]),
               volume=int(parts[7]))

# Convert RDD to DataFrame
df = sqlContext.createDataFrame(RDD.map(parse_line))

# Register the DataFrame as a temp table
df.registerTempTable("stock_data")

result = sqlContext.sql("SELECT * FROM stock_data")
result.show()
```

3	AAPL	2024-03-18 09:31:00.0	175.425	175.77	175.2	175.75	498142
4	AAPL	2024-03-18 09:32:00.0	175.76	175.91	175.54	175.77	566280
5	AAPL	2024-03-18 09:33:00.0	175.76	176.168	175.68	175.975	720968
6	AAPL	2024-03-18 09:34:00.0	175.972	176.05	175.64	175.86	427842
7	AAPL	2024-03-18 09:35:00.0	175.87	176.34	175.82	176.33	692365
8	AAPL	2024-03-18 09:36:00.0	176.33	176.4	175.88	176.019	684740
9	AAPL	2024-03-18 09:37:00.0	176.001	176.19	175.96	176.005	455199
10	AAPL	2024-03-18 09:38:00.0	176.01	176.29	176.01	176.24	387677

1. How many records are there in the table?

```
record_count = sqlContext.sql("SELECT COUNT(*) as record_count FROM st
print("Number of Records: {}".format(record_count))
```

```
Number of Records: 9848
```

2. How many different days are there in the table?

```
record_count = sqlContext.sql("SELECT COUNT(*) as record_count FROM
stock_data").first().record_count
print("Number of Records: {}".format(record_count))
```

```
Number of Different Days: 5
```

3. How many records per each day are there in the table?

```
distinct_days = sqlContext.sql("SELECT COUNT(DISTINCT `date`) as
distinct_days FROM stock_data").first().distinct_days
print("Number of Different Days: {}".format(distinct_days))
```

```
Records per each day:
(u'2024-03-18',  1950)
(u'2024-03-19',  1950)
(u'2024-03-20',  1950)
(u'2024-03-21',  1950)
(u'2024-03-22',  1950)
```

4. What are the symbols in the table?

```
symbols = sqlContext.sql("SELECT DISTINCT `symbol` as symbol FROM
stock_data") symbols.show()
```

```
symbol
APPL
GOOGL
IBM
MSFT
TSLA
```

5. What is the highest price for each symbol?

```
highest_price = sqlContext.sql("SELECT `symbol` as symbol, MAX(`high`) as highest_price FROM stock_data GROUP BY `symbol`")
highest_price.show()
```

```
Highest price for each symbol:
(u'MSFT', 430.82)
(u'APPL', 178.67)
(u'GOOGL', 152.15)
(u'TSLA', 178.18)
(u'IBM', 193.98)
```

6. What is the lowest price for each symbol?

```
lowest_price = sqlContext.sql("SELECT `symbol` as symbol, MIN(`low`) as lowest_price FROM stock_data GROUP BY `symbol`")
lowest_price.show()
```

```
Lowest price for each symbol:
(u'MSFT', 415.571)
(u'APPL', 170.268)
(u'GOOGL', 146.17)
(u'TSLA', 166.3)
(u'IBM', 190.31)
```

7. What is the average price for each symbol?

```
average_price = sqlContext.sql("SELECT `symbol` as symbol,
AVG(`close`) as average_price FROM stock_data GROUP BY `symbol`")
average_price.show()
```

```
Average price for each symbol:
[(u'MSFT', 423.74882615), (u'APPL',174.54863282),
(u'GOOGL', 148.42665282), (u'TSLA',172.12119128),
(u'IBM', 192.19748359)]
```

8. What is the range of price for each symbol?

```
price_range = sqlContext.sql("SELECT `symbol` as symbol, MIN(`close`)
as Minimum, MAX(`close`) as Maximum, MAX(`close`) - MIN(`close`) as
price_range FROM stock_data GROUP BY `symbol`")
price_range.show()
```

```
Range of price for each symbol:
[(u'MSFT', (17.04)), (u'APPL', (8.61)),
(u'GOOGL', (6.07)), (u'TSLA', (12.28)), (u'IBM',
(3.97))]
```

9.What is the date on which each symbol experienced the highest price?

```
date_highest_price = sqlContext.sql("""
```

```
SELECT s.`symbol` as symbol, s.`date` as transaction_date, s.`high` as
highest_price
FROM stock_data s
JOIN (
    SELECT `symbol` as symbol, MAX(`high`) as max_price
    FROM stock_data
    GROUP BY `symbol`
) t
ON s.`symbol` = t.symbol AND s.`high` = t.max_price
""")
date_highest_price.show()
```

```
Date on which each symbol experienced the highest price:
[(u'MSFT', (u'2024-03-21', 430.82)), (u'APPL', (u'2024-03-20',
178.67)), (u'GOOGL', (u'2024-03-18', 152.15)), (u'TSLA', (u'2024-03-
21',178.18)), (u'IBM', (u'2024-03-20', 193.98))]
```

C. Blockchain Data Analysis – Part 2

```
blocks = sqlCtx.load(source="jdbc",

url="jdbc:mysql://localhost/loudacre?user=training&password=training",
```

```
                          dbtable="blocks_2024_Mar_18_to_22")

blocks_info = sqlCtx.load(source="jdbc",

url="jdbc:mysql://localhost/loudacre?user=training&password=training",
                          dbtable="blocks_info_2024_Mar_18_to_22")

tx_info = sqlCtx.load(source="jdbc",

url="jdbc:mysql://localhost/loudacre?user=training&password=training",
                          dbtable="tx_info_2024_Mar_18_to_22")
```

1. **How many total blocks are there in your blocks table?**

```
total_blocks = blocks.count()
print("Total number of blocks: {}".format(total_blocks))
```

```
Total Blocks: 1027
```

2.What is the largest block height among the blocks in your blocks table?

```
largest_block_height = blocks_info.agg({"height":
"max"}).collect()[0][0]
print("Largest block height:", largest_block_height)
```

```
Largest Block Height: 835509
```

3.What is the date and time for that block?

```
largest_block_date_time = blocks.filter(blocks["block_index"] ==
largest_block_height).select("time").collect()[0][0]
formatted_time = largest_block_date_time.strftime("%Y-%m-%d %H:%M:%S")

print("Date and time for the largest block:", formatted_time)
```

```
Time for the largest block: 2024-04-24T22:04:56.000-08:00
```

4.What is the largest number of transactions in your blocks?

```
largest_num_transactions =
tx_info.groupBy("block_index").count().agg({"count":
"max"}).collect()[0][0]
print("Largest number of transactions in a block:",
largest_num_transactions)
```

```
Highest Number of Transactions: 2141853579
```