Big Data Healthcare Analytics Project Documentation

This project implements a big data pipeline for batch analytics on the MIMIC-III Clinical Database. It processes ICU-related healthcare data using Docker-based Hadoop and Hive, with MapReduce for basic analytics.

Repository Structure

```
docker-hadoop-spark/
├── data/
│   └── mimiciii/
│       ├── csv/                    # Original uncleaned MIMIC-III CSVs
│       ├── parquet/                # Cleaned and converted Parquet files
│       └── clean_csv/              # Cleaned CSV used for MapReduce (patients)
├── scripts/
│   ├── convert_patients_to_parquet.py
│   ├── convert_admissions_to_parquet.py
│   └── convert_icustays_to_parquet.py
├── mapreduce/
│   ├── mapper.py
│   └── reducer.py
├── hive/
│   └── create_tables.sql           # Hive DDL for external tables
├── docs/
│   ├── DataModel.md
│   ├── HiveQueries.md
│   └── MapReduce.md
├── README.md
└── docker-compose.yml
```

Setup Instructions

1. **Start Docker Hadoop Environment**

```
cd docker-hadoop-spark
docker-compose up -d
```

2. **Convert CSVs to Parquet** Run these Python scripts from Git Bash:

```
python scripts/convert_patients_to_parquet.py
python scripts/convert_admissions_to_parquet.py
python scripts/convert_icustays_to_parquet.py
```

3. **Copy Files to HDFS**

```
docker cp data/mimiciii/parquet/*.parquet namenode:/tmp/
docker exec -it namenode bash
hdfs dfs -mkdir -p /user/root/mimiciii/{patients,admissions,icustays}
hdfs dfs -put /tmp/patients.parquet /user/root/mimiciii/patients/
hdfs dfs -put /tmp/admissions.parquet /user/root/mimiciii/admissions/
hdfs dfs -put /tmp/icustays.parquet /user/root/mimiciii/icustays/
```

4. Create Hive Database and Tables

```
-- Inside Beeline
CREATE DATABASE IF NOT EXISTS mimiciii;
USE mimiciii;

-- See docs/DataModel.md for DDL
```

5. Run Hive Queries

```
USE mimiciii;
SELECT * FROM icustays LIMIT 5;
```

6. Run MapReduce

```
docker cp mapreduce/mapper.py namenode:/tmp/
docker cp mapreduce/reducer.py namenode:/tmp/
docker cp data/mimiciii/clean_csv/PATIENTS.csv namenode:/tmp/

# Inside namenode
hdfs dfs -mkdir -p /user/root/mimiciii/patients_csv
hdfs dfs -put /tmp/PATIENTS.csv /user/root/mimiciii/patients_csv

hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming*.jar \
    -input /user/root/mimiciii/patients_csv \
    -output /user/root/mimiciii/output_avg_age \
    -mapper /tmp/mapper.py \
    -reducer /tmp/reducer.py \
    -file /tmp/mapper.py \
    -file /tmp/reducer.py

hdfs dfs -cat /user/root/mimiciii/output_avg_age/part-00000
```

Data Model

1. patients (Dimension Table)

Column	Type	Description
subject_id	BIGINT	Unique patient ID
gender	STRING	Patient gender (M/F)
dob	TIMESTAMP	Date of birth
dod	TIMESTAMP	Date of death (if applicable)
expire_flag	TINYINT	1 if patient is known deceased

2. admissions (Fact Table)

Column	Type	Description
subject_id	BIGINT	Patient ID (FK)
hadm_id	BIGINT	Admission ID
admittime	TIMESTAMP	Admission timestamp
dischtime	TIMESTAMP	Discharge timestamp
deathtime	TIMESTAMP	Death timestamp (if any)
diagnosis	STRING	Primary diagnosis
hospital_expire_flag	TINYINT	1 if patient died during admission

3. icustays (Fact Table)

Column	Type	Description
icustay_id	BIGINT	ICU stay ID
hadm_id	BIGINT	Admission ID (FK)
subject_id	BIGINT	Patient ID (FK)
intime	TIMESTAMP	ICU admission time
outtime	TIMESTAMP	ICU discharge time
los	DOUBLE	Length of ICU stay (in days)

Relationships

- `patients.subject_id = admissions.subject_id`
- `admissions.hadm_id = icustays.hadm_id`
- `patients.subject_id = icustays.subject_id`

Measures

- `los` — length of ICU stay (from icustays)
- `hospital_expire_flag` — hospital mortality indicator (admissions)
- `expire_flag` — mortality indicator (patients)
- `diagnosis` — diagnosis at admission (admissions)

Use in Analytics

These dimensions and relationships allow analysis such as:

- Average length of stay by diagnosis
- ICU readmissions by patient
- Mortality rates by demographic groups

Hive Queries

1. Average Length of Stay per Diagnosis

```
SELECT
  a.diagnosis,
  ROUND(AVG(i.los), 2) AS avg_length_of_stay
FROM
  admissions a
JOIN
  icustays i
ON
  a.hadm_id = i.hadm_id
GROUP BY
  a.diagnosis
ORDER BY
  avg_length_of_stay DESC
LIMIT 20;
```

2. Distribution of ICU Readmissions

```
SELECT
  subject_id,
  COUNT(icustay_id) AS icu_admissions
FROM
  icustays
GROUP BY
  subject_id
HAVING
  COUNT(icustay_id) > 1
ORDER BY
  icu_admissions DESC
LIMIT 20;
```

3. Mortality Rates by Ethnicity

```
SELECT
  ethnicity,
  COUNT(*) AS total_admissions,
  SUM(hospital_expire_flag) AS deaths,
  ROUND(SUM(hospital_expire_flag) * 100.0 / COUNT(*), 2) AS mortality_rate_percent
FROM
  admissions
GROUP BY
  ethnicity
ORDER BY
  mortality_rate_percent DESC;
```

4. Mortality Rates by Gender

```
SELECT
  gender,
  COUNT(*) AS total_patients,
  SUM(expire_flag) AS deaths,
  ROUND(SUM(expire_flag) * 100.0 / COUNT(*), 2) AS mortality_rate_percent
FROM
  patients
GROUP BY
  gender;
```

MapReduce Task: Average Patient Age

Objective: To compute the average age of patients in the `PATIENTS_CLEAN` dataset using a MapReduce job written in Java, executed via Hadoop Streaming on a Docker-based Hadoop cluster.

- **File name:** `PATIENTS_CLEAN`
- **HDFS path:** `/user/root/clean_csv/PATIENTS_CLEAN`
- **Format:** CSV (comma-separated values)

Task Logic

Calculate **age = dod - dob**

- If `dod` is empty (patient is alive), use `2200-01-01` as reference
- Skip rows with malformed or missing dates
- Skip the header row
- Only include patients aged 0–150 years (sanity check)

Language & Frameworks

- Java 8

- Hadoop MapReduce API (YARN)

- JAR compiled inside the Docker container

Build & Execution

1. Create and compile Java class:

```
mkdir -p /root/avg_classes

export HADOOP_CLASSPATH=$(hadoop classpath)

javac -classpath $HADOOP_CLASSPATH -d /root/avg_classes /root/AverageAge.java
```

2. Package into a JAR:

```
jar -cvf /root/avg.jar -C /root/avg_classes/ .
```

3. Upload data to HDFS:

```
hdfs dfs -mkdir -p /user/root/clean_csv

hdfs dfs -put /root/PATIENTS_CLEAN /user/root/clean_csv/
```

4. Run MapReduce job:

```
hdfs dfs -rm -r /user/root/output_avg

hadoop jar /root/avg.jar AverageAge \
  /user/root/clean_csv/PATIENTS_CLEAN \
  /user/root/output_avg
```

Output

```
Average Age >>> 70.68
```