MIDS W205

Exercise 1
Updated: 9/26/15

Instructors:

Jari Koister, jari@ischool.berkeley.edu

Dan McClary, dan.mcclary@ischool.berkely.edu

Karthik Ramasamy, karthik@ischool.berkeley.edu

Arash Nourian , nourian@ischool.berkeley.edu

Manos Papagelis , papaggel@ischool.berkeley.edu

Introduction

Your data science organization has been asked to conduct a study on quality of care for Medicare patients. In order to do this, you will need to apply the concepts and tools discussed in the first half of the semester. Over the course of the next weeks, you will

- Load data into an HDFS data lake
- Create an ER diagram and schema for the data you are going to analyze
- Transform raw data from the data lake to fit your data model
- Analyze your derived data to answer questions about quality of care

The motivation for this exercise is twofold. First, the exercise is designed to put into practice concepts and technologies discussed in the first half of the semester. Second, the exercise requires that you make the connection between analyzing data and how data are stored, modeled, and processed.

The Questions

Your organization wishes to understand how to change outcomes for Medicare patients. To maximize impact, your group will have to determine

- What hospitals are models of high-quality care—that is, which hospitals have the most consistently high scores for a variety of procedures?
- What states are models of high-quality care?
- Which procedures have the greatest variability between hospitals?
- Are average scores for hospital quality or procedural variability correlated with patient survey responses?

The Data

You will be working with data from the Centers for Medicare and Medicaid Services (CMS) Hospital Compare project. You can download the entire dataset as flat comma-separated files here. A description of the various files is provided in the data dictionary.

Tip

You can download data from a remote source into a Linux system using the wget command. Just type the following:

wget <url that you copied>

You can also unzip ZIP files on the command line using command (this will create a new directory): unzip <zipfile>

You do not—and should not—attempt to use every file to solve the exercise. Instead, focus on identifying those base files that will help you build quality answers to the questions.

- General Hospital information is found in **Hospital General Information.csv**.
  - Suggested renaming: **hospitals.csv**
- Procedure data can be found in
  - **Timely and Effective Care - Hospital.csv**
    - Suggested renaming: **effective_care.csv**
  - **Readmissions and Deaths - Hospital.csv**
    - Suggested renaming **readmissions.csv**
- The mapping of measures to codes can be found in **Measure Dates.csv**.
- Survey response data can be found in (your choice of)
  - **hvbp_hcahps_05_28_2015.csv**
    - Suggested renaming: **surveys_responses.csv**

Tip

You can easily strip the first line of a file and write it to a new file with a new name like this:

tail -n +2 /path/to/original > /path/to/new_file

Tasks

Each week's work builds on the previous week, starting with loading and describing data and concluding with answering the questions described above. Throughout the exercise, you will need to check your work into a Git repository and push it to your GitHub account. You will be graded both on the correctness of your code and output as well as on your analytical reasoning.

Week 5 half of week 6

Percentage of Exercise Grade: 33%

In the first week, you need to stage and model the CMS Hospital Compare. You will stage it in a HDFS-backed data lake, impose that schema in the Hive metastore, and a design a schema into which you will transform the data.

1. Create a folder in your GitHub repository for exercise_1.
   1.1. Create a folder under exercise_1 called **loading_and_modeling**.
2. Load the raw data files into HDFS under "/user/<user_name>/hospital_compare" on either your AWS or Vagrant machine.
   2.1. It is **strongly** recommended that you remove the header lines from the files before loading them into HDFS.
   2.2. You may want to consider renaming the files to eliminate spaces and better describe the contents.
   2.3. Place the commands to do any renaming and loading to HDFS in a file called **load_data_lake.sh**.
      2.3.1. Add this file to your Git repository, then commit and push the changes.
3. Build an ER diagram for the entities and relationships you need to answer the questions above.
   3.1. At minimum, your ER diagram must include entities for Hospitals, Procedures, and Survey Results.
   3.2. Save this ER diagram as a PNG file to the loading_and_modeling directory.
   3.3. Add it to your Git repository, then commit and push the changes.
4. Write Data Definition Language (DDL) SQL statements for each of the base files you have loaded into HDFS.
   4.1. DDL statements are of the following forms:
      4.1.1. DROP TABLE <table_name>;
      4.1.2. CREATE EXTERNAL TABLE <table_name> (<col1_name>, <col1_type>, …)

ROW FORMAT DELIMITED
FIELDS TERMINATED BY ‘,’
STORED AS TEXTFILE
LOCATION ‘/path/in/hdfs’;

4.1.3. Store the statements in a file called **hive_base_ddl.sql**.
4.1.3.1. Run this file in Hive using: hive –f /path/to/hive_base_ddl.sql.
4.1.3.2. Refer to the Hive Language Manual for more detail on data definition statements.
4.1.3.3. When the DDL is error free, add this file to your Git repository, then commit and push the changes.

Week half of week 6 and First Half of Week 7

Percentage of Exercise Grade: 33%

In the 10 days following the construction of your data lake, you will need to transform the raw data into a set of tables that matches your ER diagram. You may use Hive’s SQL interface, SparkSQL’s SQL interface, or Pyspark to perform the transformations.

1. Create a folder under exercise_1 called **transforming**.
2. For each of the entities (tables) in your ER diagram write either:
   2.1. A SQL query that transforms the raw data into that shape
      2.1.1. SQL queries that transform data should use CREATE TABLE AS SELECT to store transformed data.
   2.2. A Python program that uses pyspark to transform the data
      2.2.1. Pyspark programs that transform data should use sc.saveAsTextFile to store transformed data.
3. Put each transformation in a file ending in .sql (for Hive/SparkSQL) or .py (for Pyspark) under “transforming.”
   3.1. When each file runs correctly and the data are in the appropriate shape, add it to the Git repository, commit it, and push to GitHub.

Second Half Week 7 and Week 8

Percentage of Exercise Grade: 33%

In the final 10 days of the exercise, you should focus on answering the three questions described at the outset.

1. Create a folder called **investigations** under exercise_1.
2. Create folders under investigations called
   2.1. best_hospitals
   2.2. best_states
   2.3. hospital_variability
   2.4. hospitals_and_patients
3. For each question, devise either SQL queries (for Hive or SparkSQL) or Python programs (using PySpark) to produce a table of **10** entries that support your answer to the question. **For example**:
   3.1. Devise a SQL query that finds the 10 hospitals with the highest quality of care, along with their aggregate and average quality scores, as well as variability in scores.
   3.2. Place this query in a file called **best_hospitals.sql**.
   3.3. Add a text file called **best_hospitals.txt**.
      3.3.1. Write your answer to the question about hospital quality of care and provide the table of 10 results that support this.
      3.3.2. Your written answer should provide **your conclusion,** your justification for **why this approach is appropriate**, and **why these results support your conclusion**.
4. For each query/Python script and textfile, add them to your Git repository, commit them, and push the result.

Evaluation and Acceptance Criteria

Deliverables

1. All code outlined above **must be** committed and pushed to your GitHub repository.
2. All code must be runnable by your instructor in the ucb_complete AWS AMI.
3. Your GitHub repository **must be** shared with your section instructor.
4. All code is due at midnight, Hawaii time. Sunday, October 18. Check-ins to GitHub beyond that date will be ignored.

Grading Criteria

1. All code will be executed by your section instructor or TA in the AWS AMI.
2. Full points require the following:
    a. All code runs without error.
    b. All queries produce the results desired (matches the ER diagram or the table in the results files).
    c. Analysis results sufficiently support the answer the question and the reasoning behind that answer.
    d. Conclusion is consistent with the data provided.