Data Integration Project
Step 2: Integration

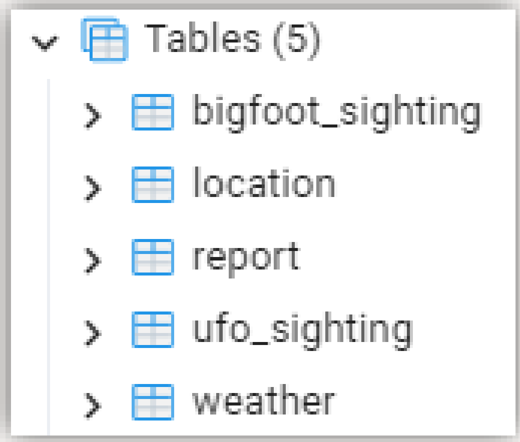
Is Bigfoot an Alien?

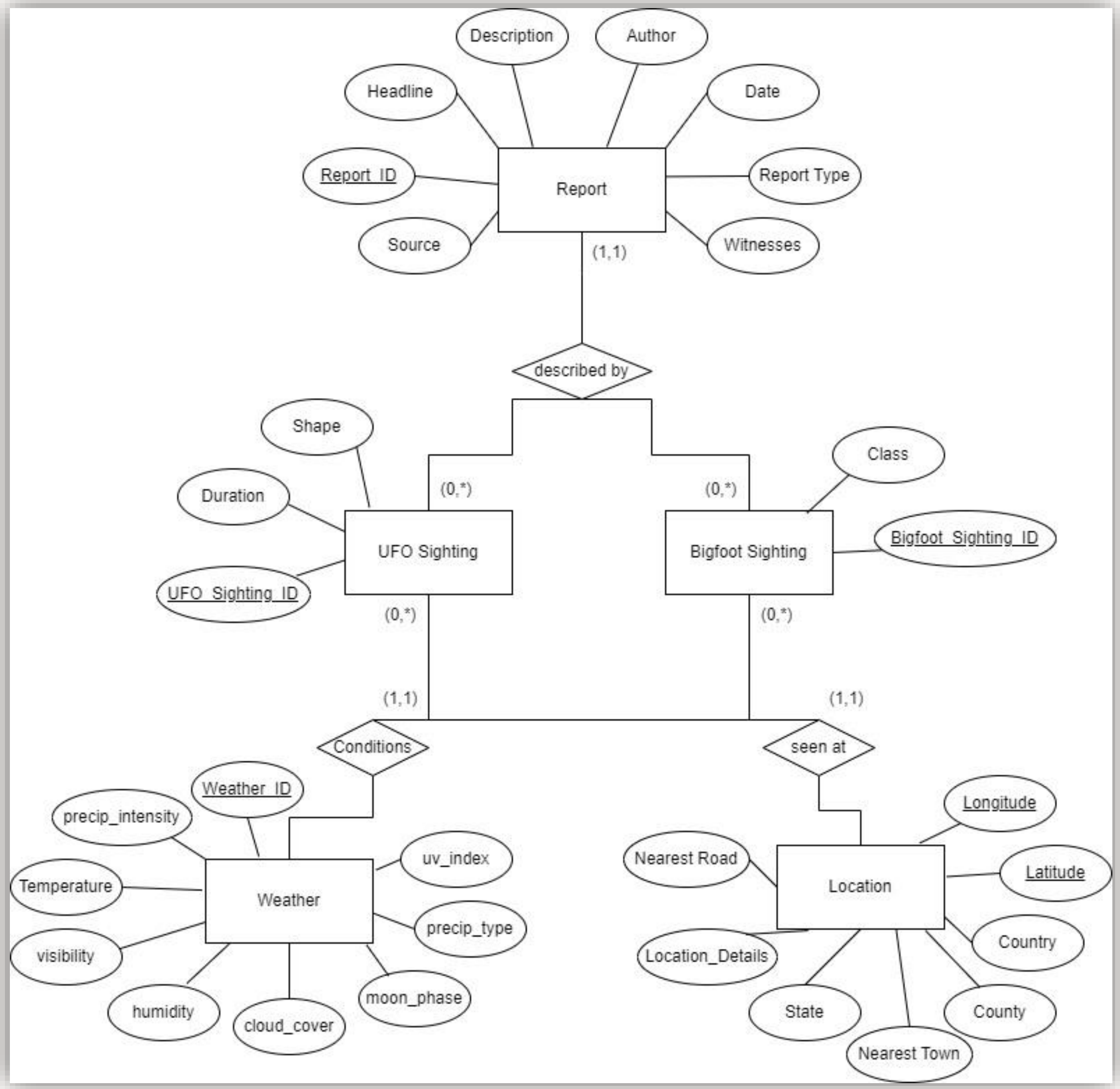
Julian Trösser

Zoe Chiying Lai

Changes to our ER-Model

- Dropped “Witnesses” in “Report”
- Dropped “Shape” in “UFO Sighting”
- No significant changes


Tables (5)
- bigfoot_sighting
- location
- report
- ufo_sighting
- weather



Manual Data Integration

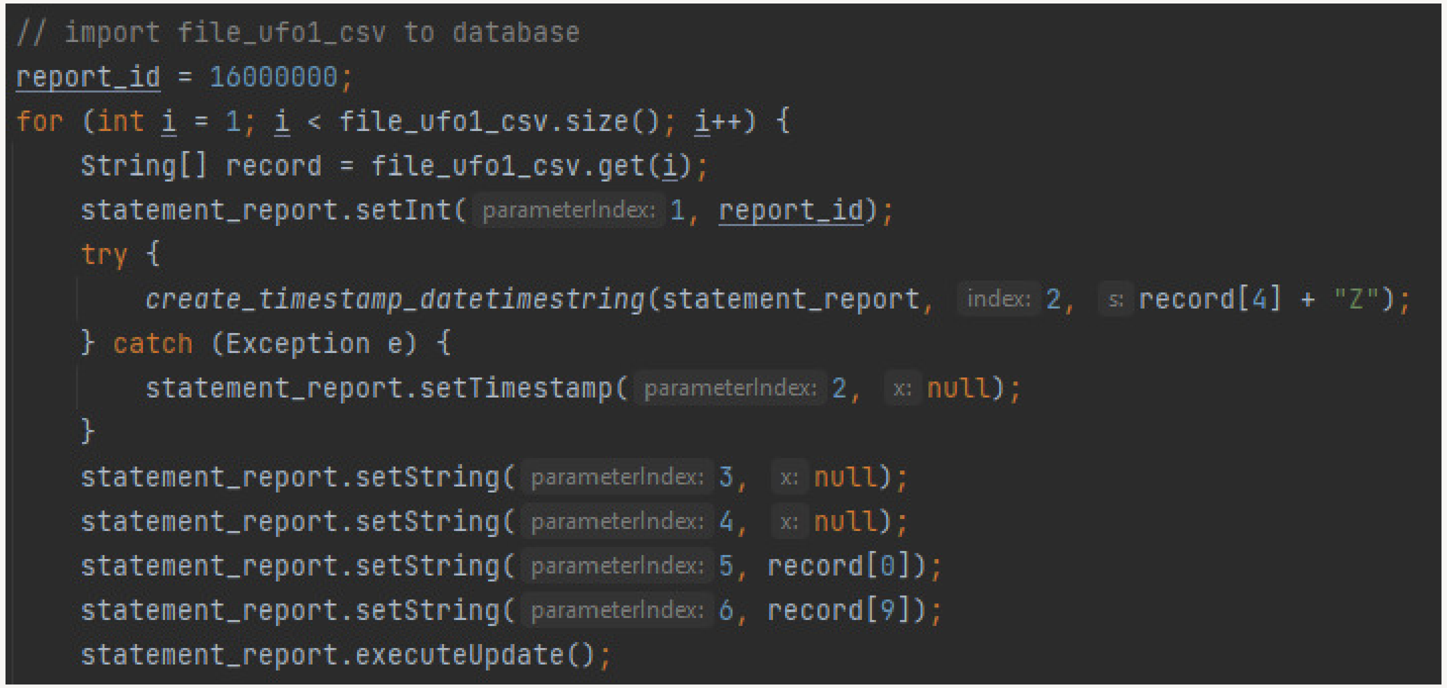
Reasons

- Datasets have different formatting
- Many referencing across datasets as well as SQL relations
- Lack of time for new tools such as Apache Camel

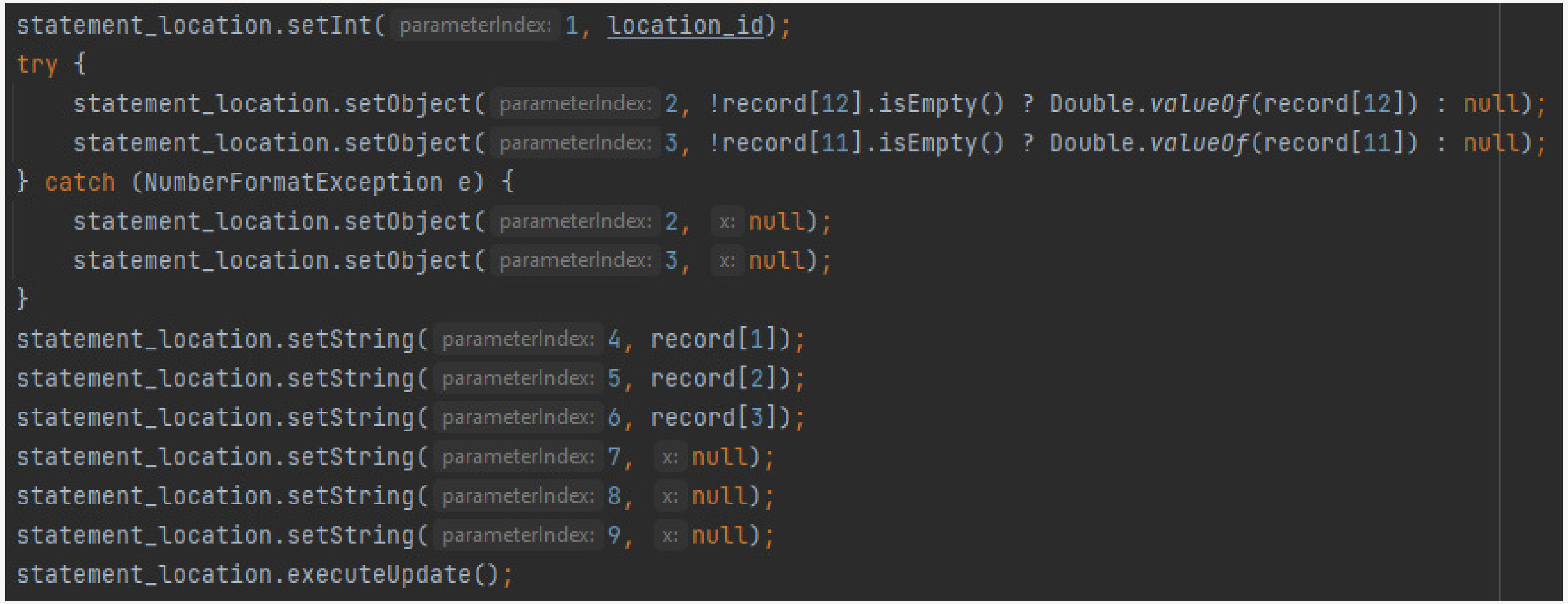
Steps

- Created our designated database in postgreSQL
- Looked through source attributes
- Looped through records for each dataset
  - Mapped appropriate attributes to designated relation in database
  - Additional functions to handle formatting, if any e.g. Timestamp, Duration

Example



```
// import file_ufo1_csv to database
report_id = 16000000;
for (int i = 1; i < file_ufo1_csv.size(); i++) {
    String[] record = file_ufo1_csv.get(i);
    statement_report.setInt( parameterIndex: 1, report_id);
    try {
        create_timestamp_datetimestring(statement_report,  index: 2,  s: record[4] + "Z");
    } catch (Exception e) {
        statement_report.setTimestamp( parameterIndex: 2,  x: null);
    }
    statement_report.setString( parameterIndex: 3,  x: null);
    statement_report.setString( parameterIndex: 4,  x: null);
    statement_report.setString( parameterIndex: 5, record[0]);
    statement_report.setString( parameterIndex: 6, record[9]);
    statement_report.executeUpdate();
```

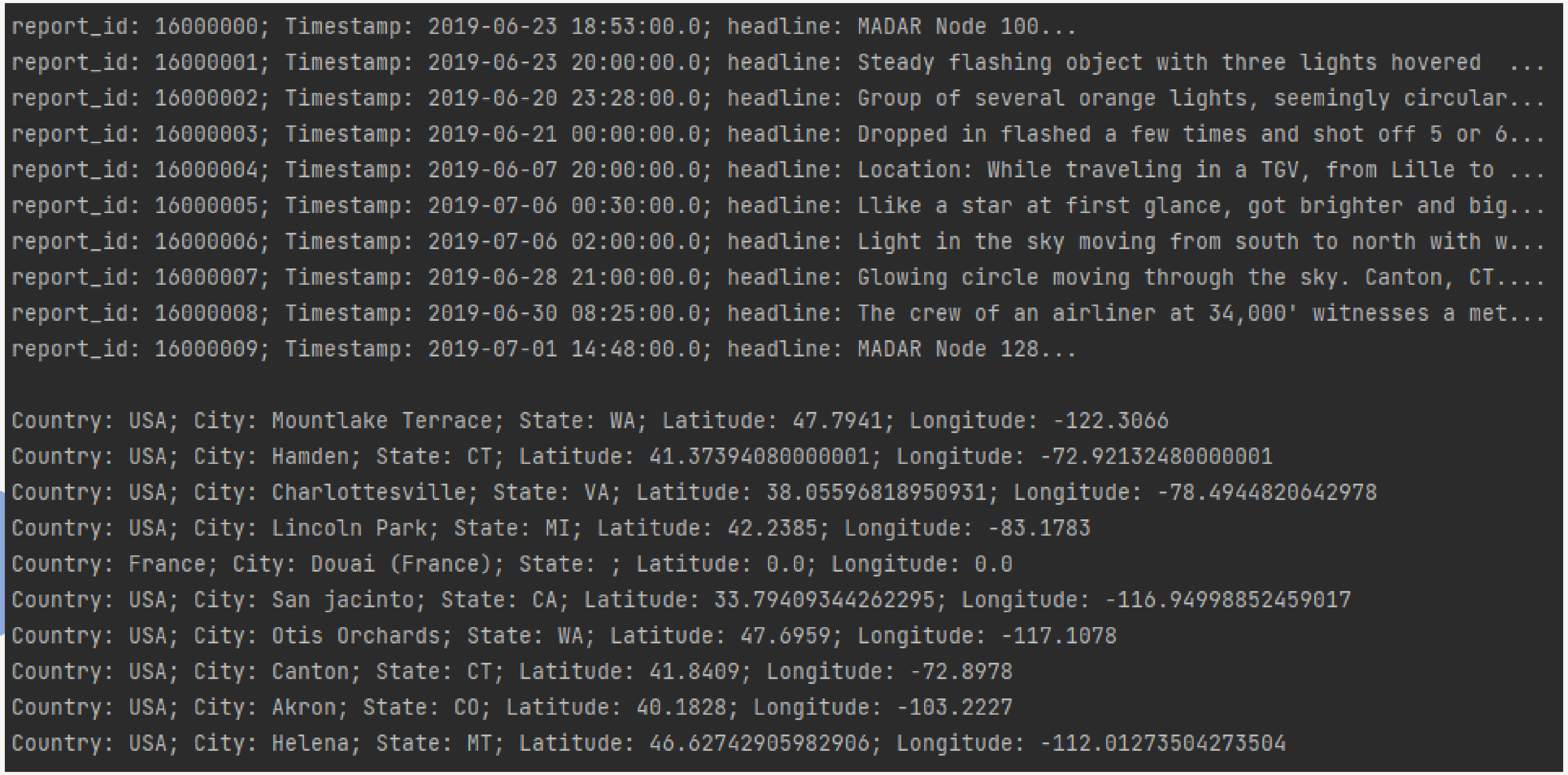


```
statement_location.setInt( parameterIndex: 1, location_id);
try {
    statement_location.setObject( parameterIndex: 2, !record[12].isEmpty() ? Double.valueOf(record[12]) : null);
    statement_location.setObject( parameterIndex: 3, !record[11].isEmpty() ? Double.valueOf(record[11]) : null);
} catch (NumberFormatException e) {
    statement_location.setObject( parameterIndex: 2,  x: null);
    statement_location.setObject( parameterIndex: 3,  x: null);
}
statement_location.setString( parameterIndex: 4, record[1]);
statement_location.setString( parameterIndex: 5, record[2]);
statement_location.setString( parameterIndex: 6, record[3]);
statement_location.setString( parameterIndex: 7,  x: null);
statement_location.setString( parameterIndex: 8,  x: null);
statement_location.setString( parameterIndex: 9,  x: null);
statement_location.executeUpdate();
```

Example Result

Queries:

- SELECT * FROM report WHERE date IS NOT NULL LIMIT 10
- SELECT * FROM location LIMIT 10

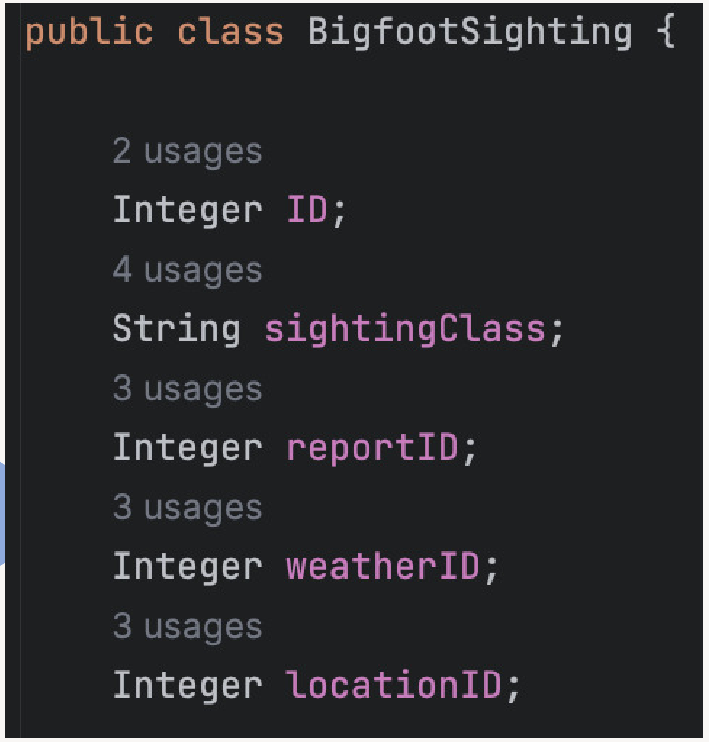


```
report_id: 16000000; Timestamp: 2019-06-23 18:53:00.0; headline: MADAR Node 100...
report_id: 16000001; Timestamp: 2019-06-23 20:00:00.0; headline: Steady flashing object with three lights hovered  ...
report_id: 16000002; Timestamp: 2019-06-20 23:28:00.0; headline: Group of several orange lights, seemingly circular...
report_id: 16000003; Timestamp: 2019-06-21 00:00:00.0; headline: Dropped in flashed a few times and shot off 5 or 6...
report_id: 16000004; Timestamp: 2019-06-07 20:00:00.0; headline: Location: While traveling in a TGV, from Lille to ...
report_id: 16000005; Timestamp: 2019-07-06 00:30:00.0; headline: Llike a star at first glance, got brighter and big...
report_id: 16000006; Timestamp: 2019-07-06 02:00:00.0; headline: Light in the sky moving from south to north with w...
report_id: 16000007; Timestamp: 2019-06-28 21:00:00.0; headline: Glowing circle moving through the sky. Canton, CT....
report_id: 16000008; Timestamp: 2019-06-30 08:25:00.0; headline: The crew of an airliner at 34,000' witnesses a met...
report_id: 16000009; Timestamp: 2019-07-01 14:48:00.0; headline: MADAR Node 128...

Country: USA; City: Mountlake Terrace; State: WA; Latitude: 47.7941; Longitude: -122.3066
Country: USA; City: Hamden; State: CT; Latitude: 41.37394080000001; Longitude: -72.92132480000001
Country: USA; City: Charlottesville; State: VA; Latitude: 38.05596818950931; Longitude: -78.4944820642978
Country: USA; City: Lincoln Park; State: MI; Latitude: 42.2385; Longitude: -83.1783
Country: France; City: Douai (France); State: ; Latitude: 0.0; Longitude: 0.0
Country: USA; City: San jacinto; State: CA; Latitude: 33.79409344262295; Longitude: -116.94998852459017
Country: USA; City: Otis Orchards; State: WA; Latitude: 47.6959; Longitude: -117.1078
Country: USA; City: Canton; State: CT; Latitude: 41.8409; Longitude: -72.8978
Country: USA; City: Akron; State: CO; Latitude: 40.1828; Longitude: -103.2227
Country: USA; City: Helena; State: MT; Latitude: 46.62742905982906; Longitude: -112.01273504273504
```

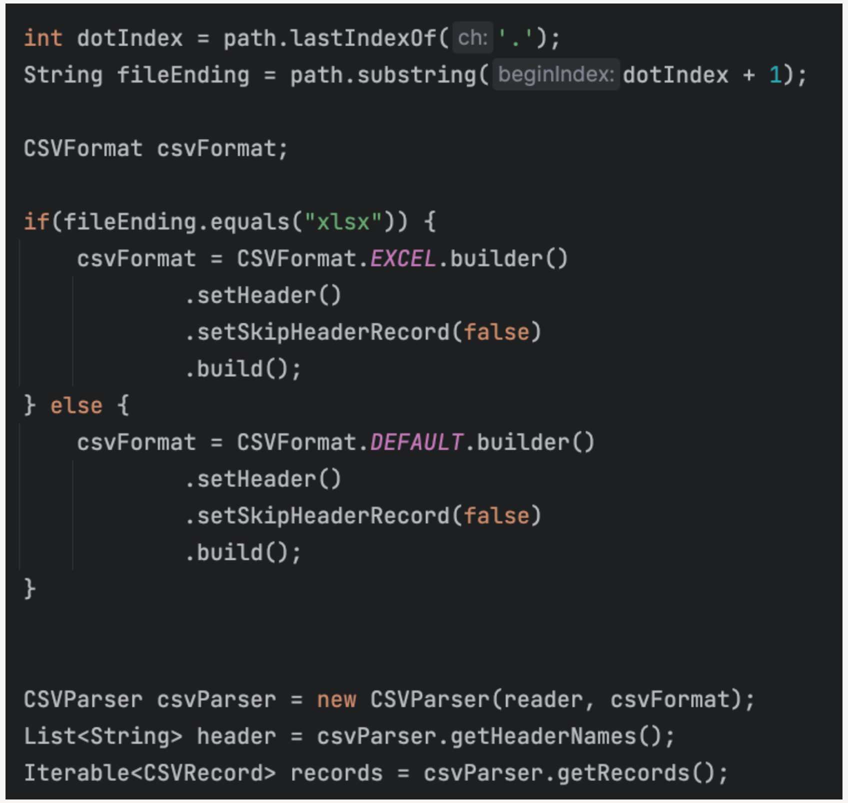
Semi-Automatic Approach

Steps

- Created our designated database in PostgreSQL
- Created a Class for every Table in the Database
- Read the file using Apache Commons CSV

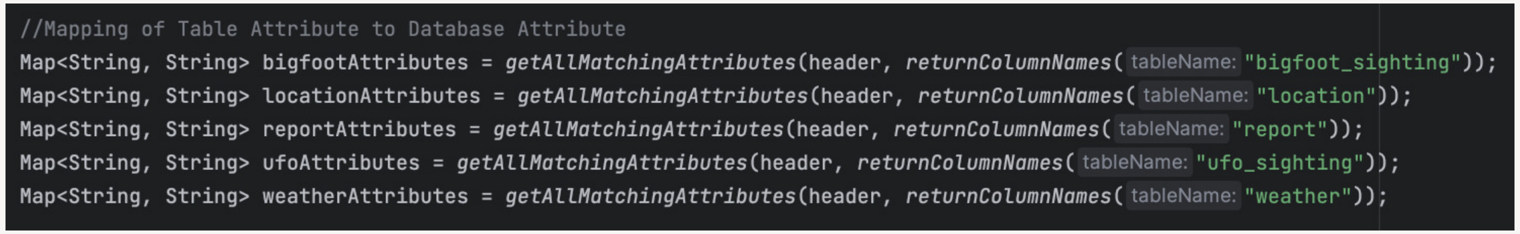


```java
public class BigfootSighting {

    2 usages
    Integer ID;
    4 usages
    String sightingClass;
    3 usages
    Integer reportID;
    3 usages
    Integer weatherID;
    3 usages
    Integer locationID;
```

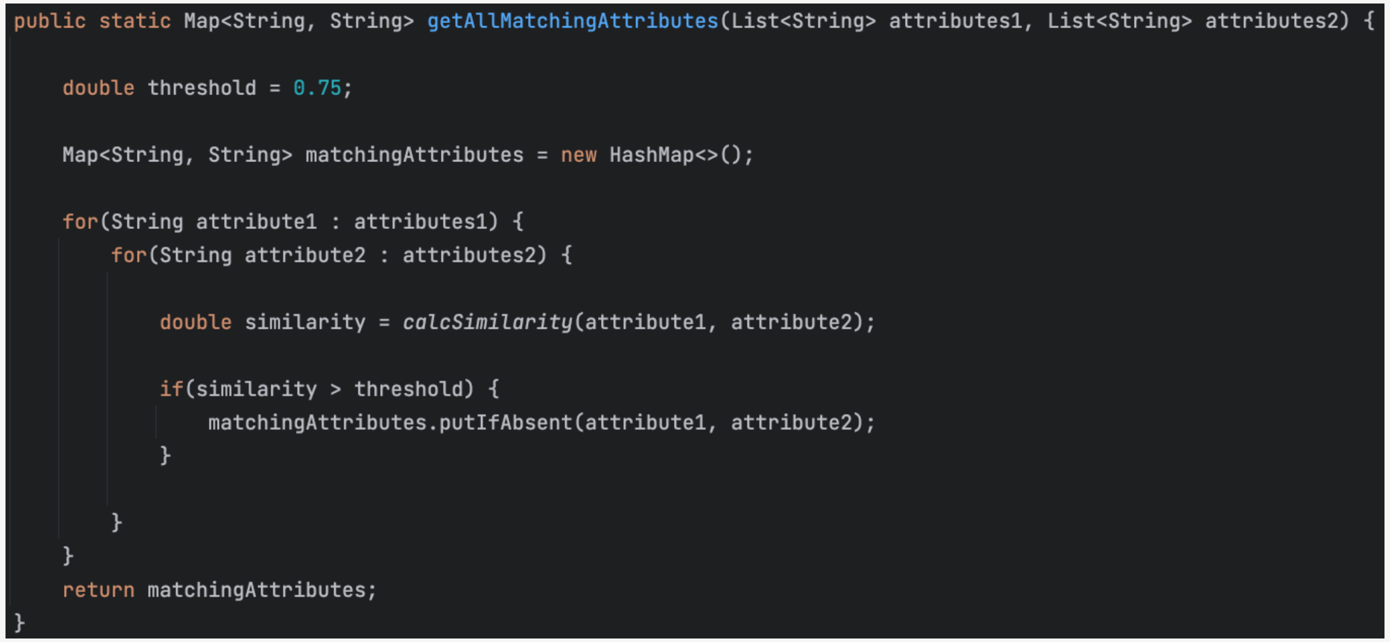


```java
int dotIndex = path.lastIndexOf( ch: '.');
String fileEnding = path.substring( beginIndex: dotIndex + 1);

CSVFormat csvFormat;

if(fileEnding.equals("xlsx")) {
    csvFormat = CSVFormat.EXCEL.builder()
            .setHeader()
            .setSkipHeaderRecord(false)
            .build();
} else {
    csvFormat = CSVFormat.DEFAULT.builder()
            .setHeader()
            .setSkipHeaderRecord(false)
            .build();
}


CSVParser csvParser = new CSVParser(reader, csvFormat);
List<String> header = csvParser.getHeaderNames();
Iterable<CSVRecord> records = csvParser.getRecords();
```

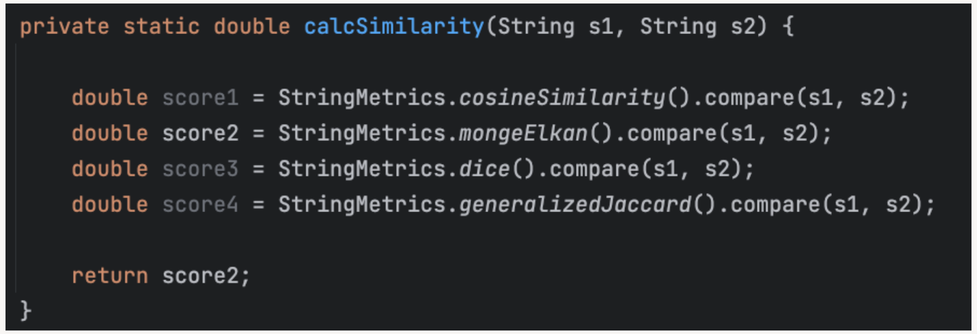
- Automatically Mapped the Attributes in the Spreadsheets to the ones in our Database
- The Mapping is based on a Similarity between those Attributes (e.g Classification (Datase) should map to Class (Database))
- Using Simmetrics to calculate the similarities
- Trying multiple Metrics we found
- Pure Monge-Elkan performs the best for our Data, however a Mix of Monge-Elkan and Cosine is also quite good.
- If the Similiarity is above a certain threshold (for us 0.75 worked well) a Mapping is made



```java
//Mapping of Table Attribute to Database Attribute
Map<String, String> bigfootAttributes = getAllMatchingAttributes(header, returnColumnNames( tableName: "bigfoot_sighting"));
Map<String, String> locationAttributes = getAllMatchingAttributes(header, returnColumnNames( tableName: "location"));
Map<String, String> reportAttributes = getAllMatchingAttributes(header, returnColumnNames( tableName: "report"));
Map<String, String> ufoAttributes = getAllMatchingAttributes(header, returnColumnNames( tableName: "ufo_sighting"));
Map<String, String> weatherAttributes = getAllMatchingAttributes(header, returnColumnNames( tableName: "weather"));
```



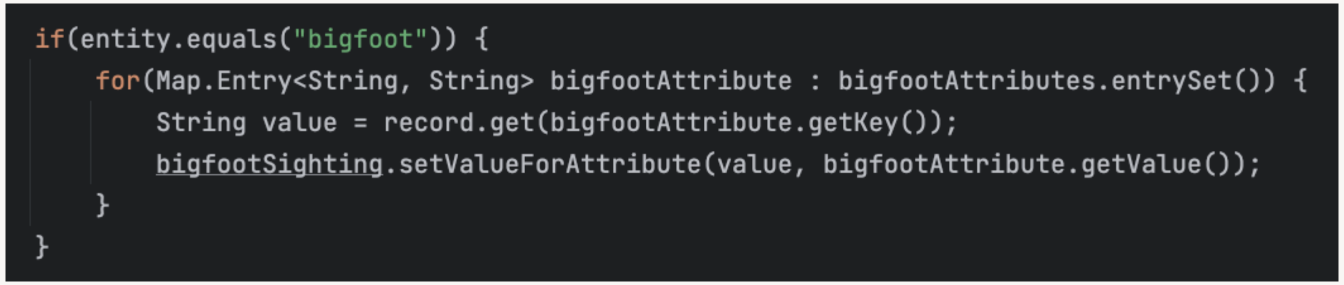
```java
public static Map<String, String> getAllMatchingAttributes(List<String> attributes1, List<String> attributes2) {

    double threshold = 0.75;

    Map<String, String> matchingAttributes = new HashMap<>();

    for(String attribute1 : attributes1) {
        for(String attribute2 : attributes2) {

            double similarity = calcSimilarity(attribute1, attribute2);

            if(similarity > threshold) {
                matchingAttributes.putIfAbsent(attribute1, attribute2);
            }

        }
    }
    return matchingAttributes;
}
```



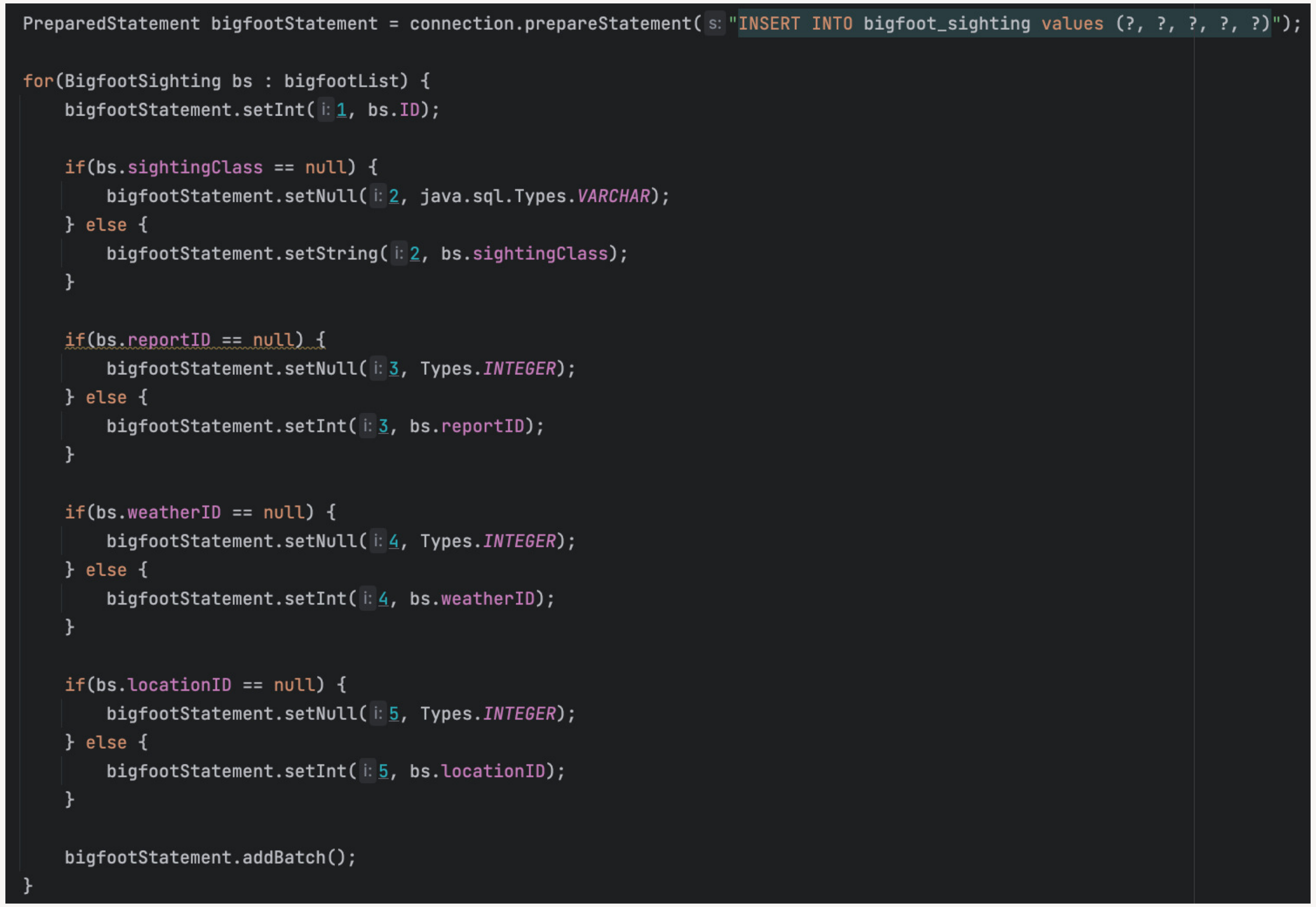
```java
private static double calcSimilarity(String s1, String s2) {

    double score1 = StringMetrics.cosineSimilarity().compare(s1, s2);
    double score2 = StringMetrics.mongeElkan().compare(s1, s2);
    double score3 = StringMetrics.dice().compare(s1, s2);
    double score4 = StringMetrics.generalizedJaccard().compare(s1, s2);

    return score2;
}
```

- We then set the values for our Objects
- And then Added them to the Database

- Benefits of this approach:
  - Don’t have to know the specific indices
  - Generally aplicalble to any Spreadsheet that contains (a subset of) our desired attributes



```java
if(entity.equals("bigfoot")) {
    for(Map.Entry<String, String> bigfootAttribute : bigfootAttributes.entrySet()) {
        String value = record.get(bigfootAttribute.getKey());
        bigfootSighting.setValueForAttribute(value, bigfootAttribute.getValue());
    }
}
```



```java
PreparedStatement bigfootStatement = connection.prepareStatement( s: "INSERT INTO bigfoot_sighting values (?, ?, ?, ?, ?)");

for(BigfootSighting bs : bigfootList) {
    bigfootStatement.setInt( i: 1, bs.ID);

    if(bs.sightingClass == null) {
        bigfootStatement.setNull( i: 2, java.sql.Types.VARCHAR);
    } else {
        bigfootStatement.setString( i: 2, bs.sightingClass);
    }

    if(bs.reportID == null) {
        bigfootStatement.setNull( i: 3, Types.INTEGER);
    } else {
        bigfootStatement.setInt( i: 3, bs.reportID);
    }

    if(bs.weatherID == null) {
        bigfootStatement.setNull( i: 4, Types.INTEGER);
    } else {
        bigfootStatement.setInt( i: 4, bs.weatherID);
    }

    if(bs.locationID == null) {
        bigfootStatement.setNull( i: 5, Types.INTEGER);
    } else {
        bigfootStatement.setInt( i: 5, bs.locationID);
    }

    bigfootStatement.addBatch();
}
```

Problems Encountered

Formatting

- Not consistent within dataset
- Some values might not be valid
- Example: “Timestamp: 2090-02-23 00:00:00.0”

Long texts

- Headlines, descriptions, weather conditions
- Difficult to interpret in large-scale

Similarity

- Quite difficult to find the best Metric or Metric Combination for our Data
- Exact Threshold value took some trial and error
- Might not be as effective on other Data because we fine tuned the Metrics to our desire

Impedance Mismatch

- Not an elegant solution to create an Object for every Table

Thank you