Task 1

- How did you use connection pooling?

I used connection pooling by first defining two resource names in context.xml under webcontent, allowing my instances to know where the master and slave databases are. After that was defined, I edited my helper class and changed the connection from DriverManager.getConnection() to using DataSource envCtx.lookup.

- File name, line numbers as in Github
  /src/main/webapp/META-INF/context.xml - lines 6-18
  /src/main/java/com/cs122b/Helper.java - lines 40-64

- Snapshots showing use in your code

```xml
<?xml version="1.0" encoding="UTF-8"?>

<Context>

    <!-- Defines a Data Source Connecting to localhost moviedb-->
    <Resource name="jdbc/moviedb"
              auth="Container"
              driverClassName="com.mysql.jdbc.Driver"
              type="javax.sql.DataSource"
              username="username"
              password="password"
              url="jdbc:mysql://localhost:3306/moviedb"/>


    <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="username"
              password="password" driverClassName="com.mysql.jdbc.Driver"
              url="jdbc:mysql://13.57.232.109:3306,52.53.213.204:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
</Context>
```

```java
static Connection connection() throws SQLException, NamingException {
//     the following few lines are for connection pooling
//     Obtain our environment naming context

    Context initCtx = new InitialContext();

    Context envCtx = (Context) initCtx.lookup( s: "java:comp/env");
    if (envCtx == null)
      out.println("envCtx is NULL");

    // Look up our data source (uncomment below line and comment following if on instance 1)
//    DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
    DataSource ds = (DataSource) envCtx.lookup( s: "jdbc/TestDB");

    // the following commented lines are direct connections without pooling
    //Class.forName("org.gjt.mm.mysql.Driver");
    //Class.forName("com.mysql.jdbc.Driver").newInstance();
    //Connection dbcon = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);

    if (ds == null)
      out.println("ds is null.");

    Connection dbcon = ds.getConnection();
    if (dbcon == null)
      out.println("dbcon is null.");

    // create database connection
//    return DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
    return dbcon;
  }
```

- How did you use Prepared Statements?

  Anywhere I had to have a parameter passed in to search, I cleaned/prepared it outside my getMovies method, then passed it in and created a PreparedStatement object from a querystring with ? everywhere a variable would be. Then I used statement.setstring() to set the corresponding ? to the correct variable. I then ran statement.executeQuery() and returned the results.

  - File name, line numbers as in Github
    /src/main/java/com/cs122b/Helper.java - lines 292-330, 364

  - Snapshots showing use in your code

```
291
292     String query = "Select distinct id, title, `year`, director, rating FROM " +
293             "(SELECT movies.id, title, `year`, director, rating, starId, `name` " +
294             "FROM stars_in_movies " +
295             "INNER JOIN movies " +
296             "ON stars_in_movies.movieId = movies.id " +
297             "INNER JOIN stars " +
298             "ON stars_in_movies.starId = stars.id " +
299             "left join ratings " +
300             "ON movies.id = ratings.movieId WHERE title LIKE ? " +
301             "AND `year` LIKE ? AND director LIKE ? " +
302             "AND `name` LIKE ?) as records";
303     String totalCount = "Select count(*) as count from (" + query + ") as distinctrecords";
304     PreparedStatement statement = con.prepareStatement(totalCount);
305     statement.setString( i: 1,  s: "%" + title + "%");
306     statement.setString( i: 2,  s: "%" + year + "%");
307     statement.setString( i: 3,  s: "%" + director + "%");
308     statement.setString( i: 4,  s: "%" + star + "%");
309     ResultSet res = statement.executeQuery();
310     res.next();
311     numRecords.num = res.getInt( s: "count");
312     if (sort.equals("rating")) {
313       query = query + " ORDER BY rating ";
314     } else {
315
316       query = query + " ORDER BY title ";
317     }
318     if (order.equals("DESC")) {
319       query = query + "DESC LIMIT ?, ?";
320     } else {
321       query = query + "ASC LIMIT ?, ?";
322     }
323     statement = con.prepareStatement(query);
324     out.println(query);
325     statement.setString( i: 1,  s: "%" + title + "%");
326     statement.setString( i: 2,  s: "%" + year + "%");
327     statement.setString( i: 3,  s: "%" + director + "%");
328     statement.setString( i: 4,  s: "%" + star + "%");
329     statement.setInt( i: 5, Integer.parseInt(offset));
330     statement.setInt( i: 6, Integer.parseInt(limit));
```

```
64          ResultSet resultSet = statement.executeQuery();
65          long endJDBC = System.nanoTime();
66          numRecords.jdbctime = endJDBC - startJDBC;
67          return resultSet;
68      }
```

Task 2

- Address of AWS and Google instances
  Google: 35.236.26.143
  Main AWS: http://ryanpadilla.us
  Master: 18.144.56.121
  Slave: 52.53.213.204

- Have you verified that they are accessible? Does Fablix site get opened both on Google's 80 port and AWS' 8080 port?
  Yes and yes with a caveat, tomcat still redirects to https when loading the frontend. I can still hit backend 8080 so testing was unaffected by this.

- Explain how connection pooling works with two backend SQL (in your code)?
  It's very similar to how pooling works with single localhost server, except you have to define the resource slightly differently in your context.xml. The format is still the same, except the jdbc:mysql://localhost gets changed to jdbc:mysql://<masterip>:3306,<slaveip>:3306/moviedb?params. You need to put the master first in order to have read/write redirection, as master is expected to come first. After that, the only things you need to add are connection.setReadOnly(false) wherever there is a write request, that way the write gets redirected to the master instance.

  - File name, line numbers as in Github
    /src/main/webapp/META-INF/context.xml - lines 6-18
    /src/main/java/com/cs122b/Helper.java - lines 40-64

  - Snapshots

```xml
<?xml version="1.0" encoding="UTF-8"?>

<Context>

    <!-- Defines a Data Source Connecting to localhost moviedb-->
    <Resource name="jdbc/moviedb"
              auth="Container"
              driverClassName="com.mysql.jdbc.Driver"
              type="javax.sql.DataSource"
              username="username"
              password="password"
              url="jdbc:mysql://localhost:3306/moviedb"/>


    <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="username"
              password="password" driverClassName="com.mysql.jdbc.Driver"
              url="jdbc:mysql://13.57.232.109:3306,52.53.213.204:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
</Context>
```

```java
    static Connection connection() throws SQLException, NamingException {
//      the following few lines are for connection pooling
//      Obtain our environment naming context

        Context initCtx = new InitialContext();

        Context envCtx = (Context) initCtx.lookup( s: "java:comp/env");
        if (envCtx == null)
            out.println("envCtx is NULL");

        // Look up our data source (uncomment below line and comment following if on instance 1)
//      DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
        DataSource ds = (DataSource) envCtx.lookup( s: "jdbc/TestDB");

        // the following commented lines are direct connections without pooling
        //Class.forName("org.gjt.mm.mysql.Driver");
        //Class.forName("com.mysql.jdbc.Driver").newInstance();
        //Connection dbcon = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);

        if (ds == null)
            out.println("ds is null.");

        Connection dbcon = ds.getConnection();
        if (dbcon == null)
            out.println("dbcon is null.");

        // create database connection
//      return DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
        return dbcon;
    }
```

- How read/write requests were routed?

As mentioned above, I set connection.ReadOnly(false) whenever making a write so that it gets sent to master instance.

- File name, line numbers as in Github

  /src/main/java/com/cs122b/Helper.java - lines 105, 134, 153, 231

- Snapshots

```
104 @   static JSONObject addGenre(Connection con, String genre_name) throws SQLException {
105         con.setReadOnly(false);
106         JSONArray message = new JSONArray();
107         String query = "{call add_genre(?)}";
108         CallableStatement statement = con.prepareCall(query);
109         statement.setString( i: 1, genre_name);
110         if (statement.execute()) {
111             ResultSet resultSet = statement.getResultSet();
112             while (resultSet.next()) {
113                 message.put(resultSet.getString( s: "message"));
114             }
115         }
116         JSONObject retval = new JSONObject();
117         retval.put("message", message);
118         return retval;
119     }
120
121 @   static JSONArray getGenres(Connection con) throws SQLException {
122         con.setReadOnly(true);
123         JSONArray genres = new JSONArray();
124         String query = "select name from genres";
125         Statement statement = con.createStatement();
126         ResultSet resultSet = statement.executeQuery(query);
127         while (resultSet.next()) {
128             genres.put(resultSet.getString( s: "name"));
129         }
130         return genres;
131     }
132
133 @   static JSONObject addStar(Connection con, String star_name, int star_dob) throws SQLE
134         con.setReadOnly(false);
```

```
152 @   static JSONObject addMovie(Connection con, String titl
153         con.setReadOnly(false);
154         JSONArray message = new JSONArray();
155         String query = "{call add_movie(?,?,?,?,?)}";
156         CallableStatement statement = con.prepareCall(query)
```

```
230 @   static JSONObject getSalesRecords(Connection
231         con.setReadOnly(false);
232         JSONObject records = new JSONObject();
233         Iterator<String> keys = sale.keys();
```

Task 3

- Have you uploaded the log files to Github? Where is it located?
    - All logs are located in the datafiles folder.

- Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located? I have uploaded it and it is located in root directory named as jmeter_report.html

- Have you uploaded the script to Github? Where is it located?
  It is located in the datafiles folder. It is also located as requested in war root.

- Have you uploaded the WAR file and README to Github? Where is it located?
  WAR is located in github root folder. Wanted to make some notes if someone was curious. I left comments in the file explaining what I disabled/enabled for the load tests, but it was just commenting out certain things like the connection pooling and the prepared statements. README is in default location as well.