Task 1

- How did you use connection pooling?

We first placed a resource in our context.xml file, which declared a resource with the name balanced/moviedb. This resource connects to the localhost's mysql and acts as a DataSource. We then placed a reference to this resource in our web.xml. In our codebase createDatabaseConnection(), is our way of connecting to the database. We changed this function, to pull from the DataSource, balanced/moviedb, instead of the regular resource we used in past projects.

- File name, line numbers as in Github

  cs122b-spring18-team-6/proj1/WebContent/META-INF/**context.xml, 1-21**
  cs122b-spring18-team-6/proj1/WebContent/WEB-INF/**web.xml, 12-21**

  cs122b-spring18-team-6/proj1/src/com/fablix/servlets/:
  _dashboard, 138
  Browse, 68
  Cart, 50
  Checkout, 83
  Movie, 53
  MovieList, 62
  SearchJs, 48
  Stars, 53

- Snapshots showing use in your code

```
12      <Resource name="balanced/moviedb" auth="Container" type="javax.sql.DataSource"
13                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
14                password="admin" driverClassName="com.mysql.jdbc.Driver"
15                url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
16
```

```
<resource-ref>
  <res-ref-name>balanced/moviedb</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

```
    public Connection createDatabaseConnection() throws IOException {
        try {
//            Class.forName("com.mysql.jdbc.Driver").newInstance();
//            Connection dbcon = (Connection) DriverManager.getConnection(ec2_DB_loginUrl, loginUser, loginPassword);

            Context initCtx = new InitialContext();

            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            if (envCtx == null)
                return null;

            DataSource ds = (DataSource) envCtx.lookup("balanced/moviedb");
            if (ds == null)
                return null;

            Connection dbcon = ds.getConnection();
```

- How did you use Prepared Statements?

We have prepared statements about everywhere that we have a query to be sent to the database, both search queries from the user and support queries created for use inside the code base. We have also placed a property inside the url for our connections that caches prepared statements.

- File name, line numbers as in Github

cs122b-spring18-team-6/proj1/src/com/database/access/**Queries.java,** This entire file contains all queries we make, all containing prepared statements, 47, 63, 101, 111, 116,131, 181, 186, 214, 231, 242, 278, 286, 309, 319, 334, 350, 362, 368, 378, 407, 458, 471, 486, 506.

- Snapshots showing use in your code

```
12      <Resource name="balanced/moviedb" auth="Container" type="javax.sql.DataSource"
13                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
14                password="admin" driverClassName="com.mysql.jdbc.Driver"
15                url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
16
```

```
String query_for_movies_stars = "select s.name, s.birthYear, group_concat(m.title separator ', ') as movieTitles from movies m " +
"join stars s on s.name=? and s.id=? join stars_in_movies sim on s.id=sim.starId  AND sim.movieId=m.id group by s.birthYear";

String query_for_stars = "select * from stars where name=?";
PreparedStatement prep = dbcon.prepareStatement(query_for_stars);
prep.setString(1, star);
ResultSet results = prep.executeQuery();
```

```
prep = dbcon.prepareStatement(query_genre);
prep.setString(1, genre);
results = prep.executeQuery();
```

```
public ResultSet FullTextSearchQuery(Connection db, Statement statement, String title, boolean offset,
    String ft_query;
    if (offset) {
        ft_query = "select * from movieList2 where match(title) against %s limit 10";
    } else {
        ft_query = "select * from movieList2 where match(title) against %s " + android;
    }

    PreparedStatement prep;
    ResultSet results;

    if (title.isEmpty()) {
        return null;
    } else {
        String entries[] = title.split(" ");
        String query_title = "('";
        for (int i = 0; i < entries.length; i++) {
            if (entries[i].length() <= 3) {
                query_title += "'?'" + "*";
            } else {
                query_title += "+" + "'?'" + "*";
            }
            if (i + 1 != entries.length)
                query_title += " ";
        }

        query_title += "' in boolean mode)";
        prep = db.prepareStatement(String.format(ft_query, query_title));
        System.out.println(prep.toString());
        for (int i = 0; i < entries.length; i++) {
            prep.setString(i+1, entries[i]);
        }
```

```
prep = db.prepareStatement(String.format(ft_query, query_title));
for (int i = 0; i < entries.length; i++) {
    prep.setString(i+1, entries[i]);
}
```

```
try {
    results = statement.executeQuery(String.format(ft_query, query_title));
    results = prep.executeQuery();
    return results;
```

Task 2

- Address of AWS and Google instances

  AWS:
  - Instance 1: 54.219.171.102 (Original)
  - Instance 2: 54.215.233.110 (Master)
  - Instance 3: 13.56.188.251 (Slave)

  Google:
  - Instance: 35.230.19.222

- Have you verified that they are accessible? Does Fablix site get opened both on Google's 80 port and AWS' 8080 port?

Yes Fabflix can be opened by navigating to (Google) 35.230.19.222:80/Fabflix/ as well as the scaled version through (AWS) 54.219.171.102/Fabflix/. The AWS connection will use the load balancer on port 80 to redirect to the master/slave on port 8080.

- Explain how connection pooling works with two backend SQL (in your code)?

For connection pooling to enable both our master/slave we created two resources in our context.xml and two references to these resources in web.xml. One of these resources connects to localhost's mysql database while the other resource connects to the master instance. The first is called balanced/moviedb and the second is called master/moviedb. The code uses balanced/moviedb for regular reads.

- File name, line numbers as in Github

  cs122b-spring18-team-6/proj1/WebContent/META-INF/**context.xml, 1-21**
  cs122b-spring18-team-6/proj1/WebContent/WEB-INF/**web.xml, 12-21**

- Snapshots

```
12      <Resource name="balanced/moviedb" auth="Container" type="javax.sql.DataSource"
13                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
14                password="admin" driverClassName="com.mysql.jdbc.Driver"
15                url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
16
17      <Resource name="master/moviedb" auth="Container" type="javax.sql.DataSource"
18                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
19                password="admin" driverClassName="com.mysql.jdbc.Driver"
20                url="jdbc:mysql://172.31.20.28:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
21  </Context>
```

```
  <resource-ref>
    <res-ref-name>balanced/moviedb</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
<resource-ref>
    <res-ref-name>master/moviedb</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
```

- How read/write requests were routed?

Read and writes were routed using the two resources created in the context.xml. We use the balanced/moviedb, which connects to the localhost's mysql database for reads. These can be sent to either master/slave. For writes we switch to the master/moviedb resource which points to the master's mysql database. This way when we want to do any writes we use the master instance instead of the slave, so that the databases can stay in proper sync. Our createDatabaseConnection() creates a connection to the balanced/moviedb, which will propagate read requests to either master/slave. Our createMasterConnection() will create a connection to the master and will send all requests to that instance. Anywhere that a write is meant to occur, like the dashboard, or the cart, a connection is made to the Master through the call createMasterConnection().

- File name, line numbers as in Github

  cs122b-spring18-team-6/proj1/src/com/database/access/**Basic.java, 45-92**
  cs122b-spring18-team-6/proj1/src/com/fablix/servlets/**_dashboard.java, 138**
  cs122b-spring18-team-6/proj1/src/com/fablix/servlets/**Cart.java, 50**
  cs122b-spring18-team-6/proj1/src/com/fablix/servlets/**Checkout.java, 83**

  cs122b-spring18-team-6/proj1/src/com/fablix/servlets/:
    Browse, 68
    Movie, 53
    MovieList, 62
    SearchJs, 48
    Stars, 53

- Snapshots

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

    <Resource name="master/moviedb" auth="Container" type="javax.sql.DataSource"
              maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
              password="admin" driverClassName="com.mysql.jdbc.Driver"
              url="jdbc:mysql://172.31.20.28:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
</Context>
```

```
    public Connection createDatabaseConnection() throws IOException {
        try {
//            Class.forName("com.mysql.jdbc.Driver").newInstance();
//            Connection dbcon = (Connection) DriverManager.getConnection(ec2_DB_loginUrl, loginUser, loginPassword);

            Context initCtx = new InitialContext();

            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            if (envCtx == null)
                return null;

            DataSource ds = (DataSource) envCtx.lookup("balanced/moviedb");
            if (ds == null)
                return null;

            Connection dbcon = ds.getConnection();
            return dbcon;
        } catch (java.lang.Exception ex) {
                System.out.println("Exception:  " + ex.getMessage());
                return null;
        } // end catch SQLException

    }
    public Connection createMasterConnection() throws IOException {
        try {
//            Class.forName("com.mysql.jdbc.Driver").newInstance();
//            Connection dbcon = (Connection) DriverManager.getConnection(ec2_DB_loginUrl, loginUser, loginPassword);

            Context initCtx = new InitialContext();

            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            if (envCtx == null)
                return null;

            DataSource ds = (DataSource) envCtx.lookup("master/moviedb");
            if (ds == null) {
                System.out.println("ERORR IN POOL");
                return null;
            }

            Connection dbcon = ds.getConnection();
```

```
            Basic DBObj = new Basic();
            Connection dbcon = DBObj.createMasterConnection();
            String id = DBObj.checkSuperLoginCookie(request, response);
            if (id == null && !logged) {
                RequestDispatcher view = request.getRequestDispatcher("dashboardlogin.jsp");
                view.forward(request, response);
                return;
            } else {
```

Task 3

- Have you uploaded the log file to Github? Where is it located?

  Yes. They are found in the base cs122b-spring18-team-6/**new_times.** They are the files without the files that are either scaled_# or single_#.

- Have you uploaded the HTML file to Github? Where is it located?

  Yes. It is in the base cs122b-spring18-team-6/**jmeter_report.html**.

- Have you uploaded the script to Github? Where is it located?

  Yes. It is in the base cs122b-spring18-team-6/proj1/WebContent/**parser.py**.

- Have you uploaded the WAR file and README to Github? Where is it located?
  Yes. It is in the base folder under cs122b-spring18-team-6/**Fabflix.war** and cs122b-spring18-team-6/**parserREADME.txt**.