Task 1

How did you use connection pooling?

We follow the instruction on the github pooling example, and use the pooling on the movie list search (both keyword search and advanced search), with the prepared statements to get a better performance.

- **File name, line numbers as in Github**

  The commits related to the pooling change are 74ba053 and 6e8d247.
  project2/src/MovieList/MovieListServlet.java: line 225-230
  project2/WebContent/META-INF/context.xml: line 16-22

- **Snapshots showing use in your code**

  project2/src/MovieList/MovieListServlet.java line 225-230

```
224                         // Get a connection from dataSource
225                         Context initCtx = new InitialContext();
226
227                         Context envCtx = (Context) initCtx.lookup("java:comp/env");
228
229                         DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
230                         Connection dbcon = ds.getConnection();
```

project2/WebContent/META-INF/context.xml: line 16-22
for single instance version -- when we testing we actually write original/instance 1's IP, but we don't have it on github anymore

```
16      <Resource name="jdbc/moviedb"
17                auth="Container"
18                driverClassName="com.mysql.jdbc.Driver"
19                type="javax.sql.DataSource"
20                username="jenny"
21                password="cs122b"
22                url="jdbc:mysql://localhost:3306/moviedb?useSSL=false&amp;cachePrepStmts=true"/>
```

How did you use Prepared Statements?

- **File name, line numbers as in Github**

project2/src/MovieList/MovieListServlet.java line 30-171. we have different prepared statements for different kinds (title, year, director, etc.) of search.

- **Snapshots showing use in your code**

project2/src/MovieList/MovieListServlet.java line 73-110
this is how we search and return a list of results if the user input a title keyword.

```
73          private PreparedStatement getStatementByTitle(HttpServletRequest request, Connection dbcon, String titleQuery) throws j
74          {
75
76                  String query = "SELECT M.id AS id, M.title AS title, M.year AS year, M.director AS director,\n" +
77                                 "       R.rating AS rating\n" +
78                                 "FROM ratings R RIGHT JOIN movies M ON M.Id=R.movieId\n" +
79                                 "WHERE\n" +
80                                 "  (('%%'=?) OR M.id IN (\n" +
81                                 "  SELECT movieId FROM stars_in_movies T INNER JOIN stars S ON T.starId=S.id AND S.name LIKE ?
82                                 "        ) )\n" +
83                                 "      AND M.director LIKE ? AND ( MATCH(M.title) AGAINST(? IN BOOLEAN MODE) OR  ed(M.title, ?)
84                                 "AND (-1=? OR M.year=?) GROUP BY M.id, R.rating \n ";
85
86                  query = paginationQuery(request, query);
87                  PreparedStatement preparedStatement = dbcon.prepareStatement(query);
88
89                  String starName = request.getParameter("StarName");
90                  if (starName==null) starName="";
91                  starName = "%"+ starName.trim()+"%";
92                  preparedStatement.setString(1, starName);
93                  preparedStatement.setString(2, starName);
94                  String director = request.getParameter("Director");
95                  if (director==null) director="";
96                  preparedStatement.setString(3, "%"+director.trim()+"%");
97                  String trim_title = titleQuery.trim();
98                  String titleString = "("+ String.join( "* ",trim_title.split("\\s+"))+"*)";
99                  preparedStatement.setString(4, titleString);
100
101                 preparedStatement.setString(5, trim_title);
102                 Integer threshold = (trim_title.length() /3);
103                 preparedStatement.setString(6,  threshold.toString() );
104                 Integer year = NumberUtil.parseIntWithDefault(request.getParameter("MovieYear"),-1);
105
106                 preparedStatement.setInt(7, year);
107                 preparedStatement.setInt(8, year);
108                 return preparedStatement;
109
110         }
```

Task 2

Address of AWS and Google instances

Instance 1 (origin): 52.53.40.250
Instance 2 (master): 52.9.187.8
Instance 3 (slave): 52.9.214.59
Google cloud: 35.233.214.141

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

We tested on Safari, and we can access the site on 52.53.40.250:80 (AWS instance 1), and 35.233.214.141:80(Google cloud). for the 8080, we can access on the AWS' 8080 ports.

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

In META-INF/context.xml, we use the resource jdbc/moviedb, connect to localhost:3306, which can be either master or slave, and in the keyword search (project2/src/MovieList/MovieListServlet.java), we have the pooling setup. Therefore, when it is called, it will look up the context from resources, and get the data source and connection accordingly.

project2/src/MovieList/MovieListServlet.java line 225-230

```
224                         // Get a connection from dataSource
225                         Context initCtx = new InitialContext();
226
227                         Context envCtx = (Context) initCtx.lookup("java:comp/env");
228
229                         DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
230                         Connection dbcon = ds.getConnection();
```

project2/WebContent/META-INF/context.xml: line 16-22

```
16    <Resource name="jdbc/moviedb"
17              auth="Container"
18              driverClassName="com.mysql.jdbc.Driver"
19              type="javax.sql.DataSource"
20              username="jenny"
21              password="cs122b"
22              url="jdbc:mysql://localhost:3306/moviedb?useSSL=false&amp;cachePrepStmts=true"/>
```

How read/write requests were routed?

We have two resources, one is jdbc/movedb, connect to localhost:3306 this one is for reading. Another one is jdbc/write, connecting to master's private ip:3306, for write only. So in general we connect to jdbc/moviedb, but for add movie, star, update shopping cart, we connect to jdbc/write which will direct to master instance.

Note!!! Somehow even though we inserted into master instance, but log shows up in the slave instance. The reason we are sure the query is executed in master instances is, first we did check, if we only insert in the salve, the new information won't show up in master instance. So we manually inserted some test information using employee's add star, the information did show up in master instance. So it means our redirect worked → for writing to database, it uses master's instance.

project2/WebContent/META-INF/context.xml -- line 1-24

```
1  <Context>
2
3      <!--master 172.31.23.152  52.9.187.8-->
4      <!--slave 172.31.16.84    52.9.214.59-->
5      <Resource name="jdbc/write"
6                auth="Container"
7                driverClassName="com.mysql.jdbc.Driver"
8                type="javax.sql.DataSource"
9                maxTotal="100" maxIdle="30" maxWaitMillis="10000"
10               username="jenny"
11               password="cs122b"
12               url="jdbc:mysql://172.31.23.152:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
13
14     <!--slave <172.31.16.84:3306>-->
15     <!-- Defines a Data Source Connecting to localhost moviedb-->
16     <Resource name="jdbc/moviedb"
17               auth="Container"
18               driverClassName="com.mysql.jdbc.Driver"
19               type="javax.sql.DataSource"
20               username="jenny"
21               password="cs122b"
22               url="jdbc:mysql://localhost:3306/moviedb?useSSL=false&amp;cachePrepStmts=true"/>
23
24 </Context>
```

project2/WebContent/WEB-INF/web.xml --line 19-30

```
19      <resource-ref>
20          <description>MySQL DataSource</description>
21          <res-ref-name>jdbc/write</res-ref-name>
22          <res-type>javax.sql.DataSource</res-type>
23          <res-auth>Container</res-auth>
24      </resource-ref>
25      <resource-ref>
26          <description>MySQL DataSource</description>
27          <res-ref-name>jdbc/moviedb</res-ref-name>
28          <res-type>javax.sql.DataSource</res-type>
29          <res-auth>Container</res-auth>
30      </resource-ref>
```

project2/src/Session/CheckoutServlet.java: line 27

```
25  public class CheckoutServlet extends HttpServlet {
26          private static final long serialVersionUID = 6L;
27          @Resource(name = "jdbc/write")
28          private DataSource dataSource;
```

project2/src/MovieList/AddMovieServlet.java: line 20

```
19  public class AddMovieServlet extends HttpServlet{
20      @Resource(name = "jdbc/write")
21      private DataSource dataSource;
```

project2/src/MovieList/AddStarServlet.java: line 19

```
18  @WebServlet(name = "employee.AddStarServlet ", urlPatterns = "/api/_dashboard/add-star")
19  public class AddStarServlet extends HttpServlet {
20      @Resource(name = "jdbc/write")
21      private DataSource dataSource;
```

Task 3

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

They are in the testplan folder. For the single instance cases, log files have prefix “test_single_”, and for the scaled instance cases, log files have prefix “test_scale_”.
The number associated with the file name is the case number noted on the project website: https://grape.ics.uci.edu/wiki/public/wiki/cs122b-2018-spring-project5

- **Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located?**
  testplan/jmeter_report.html
- **Have you uploaded the script to Github? Where is it located?**
  testplan/login.jmx, testplan/analysis.ipynb

- **Have you uploaded the WAR file and README to Github? Where is it located?**
  yes. WAR file: project2/target/ROOT.war
  README: /