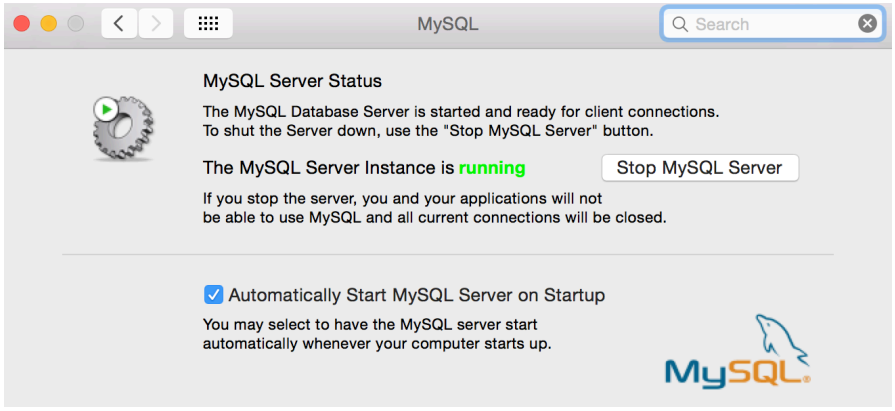
Assignment 05

Problem 1

1. Install the MySQL server and start the service.



Download the MySQL Community Edition package and store the path into "~/.bashrc".

```
export PATH=/usr/local/bin:${PATH}
export PATH=/Users/hqiu/Documents/mysql-5.6.27-osx10.9-x86_64/bin:${PATH}
```

2. Create a new local database within the MySQL server and a new user that has all privileges on that database.

```
hqiu@bos-mpdei>> mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.6.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> select user();
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)
```

```
mysql> CREATE DATABASE hqiuDB;
Query OK, 1 row affected (0.00 sec)

mysql> use hqiuDB;
Database changed
mysql> CREATE USER hqiu IDENTIFIED by 'hqiu';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT USAGE ON *.* TO hqiu@localhost IDENTIFIED BY 'hqiu';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON hqiuDB.* TO hqiu@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

3. Login the new database we just created with the new user.

```
hqiu@bos-mpdei>> mysql -u hqiu -p hqiuDB
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.6.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select user();
+----------------+
| user()         |
+----------------+
| hqiu@localhost |
+----------------+
1 row in set (0.00 sec)
```

4. In the database ‘hqiuDB’, create table FLOWERS that contains the information of 3 flowers. The table contains the name, typical height and a brief description of every flower.

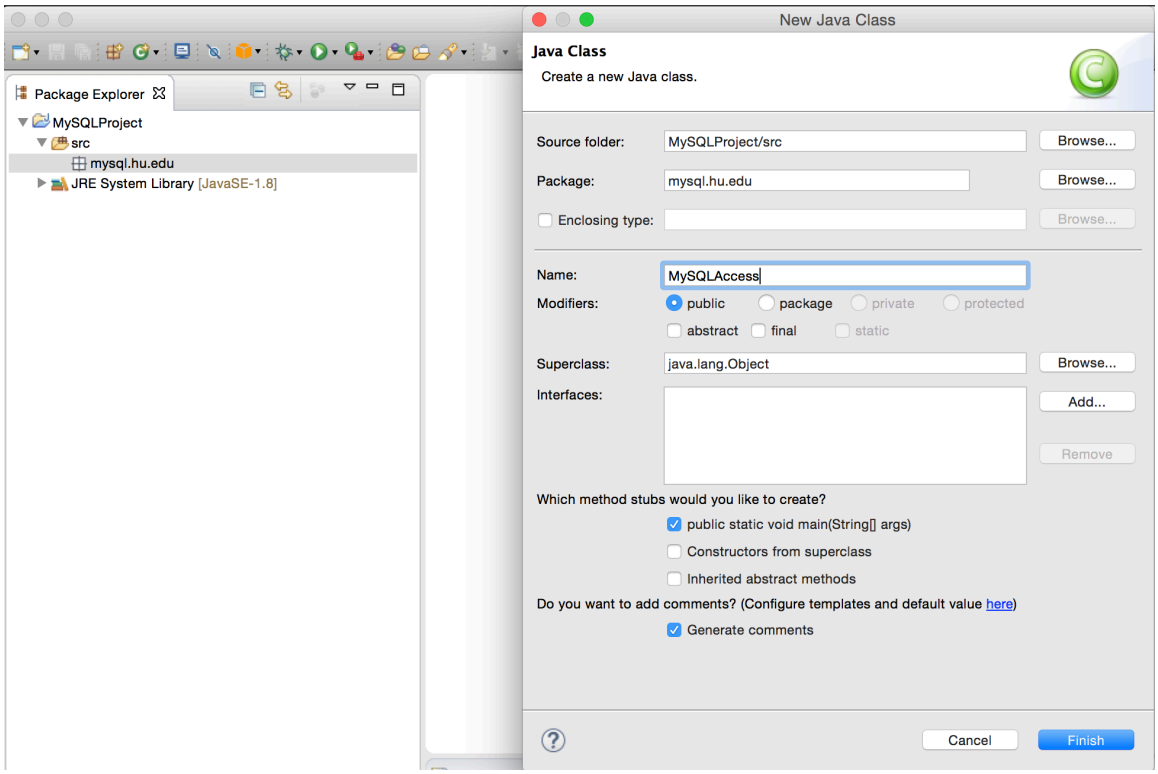
```
mysql> CREATE TABLE FLOWERS (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> name VARCHAR(30) NOT NULL,
    -> height DOUBLE NOT NULL,
    -> description VARCHAR(128) NOT NULL,
    -> PRIMARY KEY (ID));
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO FLOWERS (name, height, description) VALUES
    -> ('Rose', 14.5,'A woody perennial of the genus Rosa.'),
    -> ('Lilium', 16.7, 'A genus of herbaceous flowering plants.'),
    -> ('Tulip', 18.6, 'A genus of perennial, bulbous plants.');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT count(*) FROM FLOWERS;
+----------+
| count(*) |
+----------+
|        3 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT * FROM FLOWERS;
+----+--------+--------+----------------------------------------+
| id | name   | height | description                            |
+----+--------+--------+----------------------------------------+
|  1 | Rose   |   14.5 | A woody perennial of the genus Rosa.   |
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.  |
+----+--------+--------+----------------------------------------+
3 rows in set (0.00 sec)
```

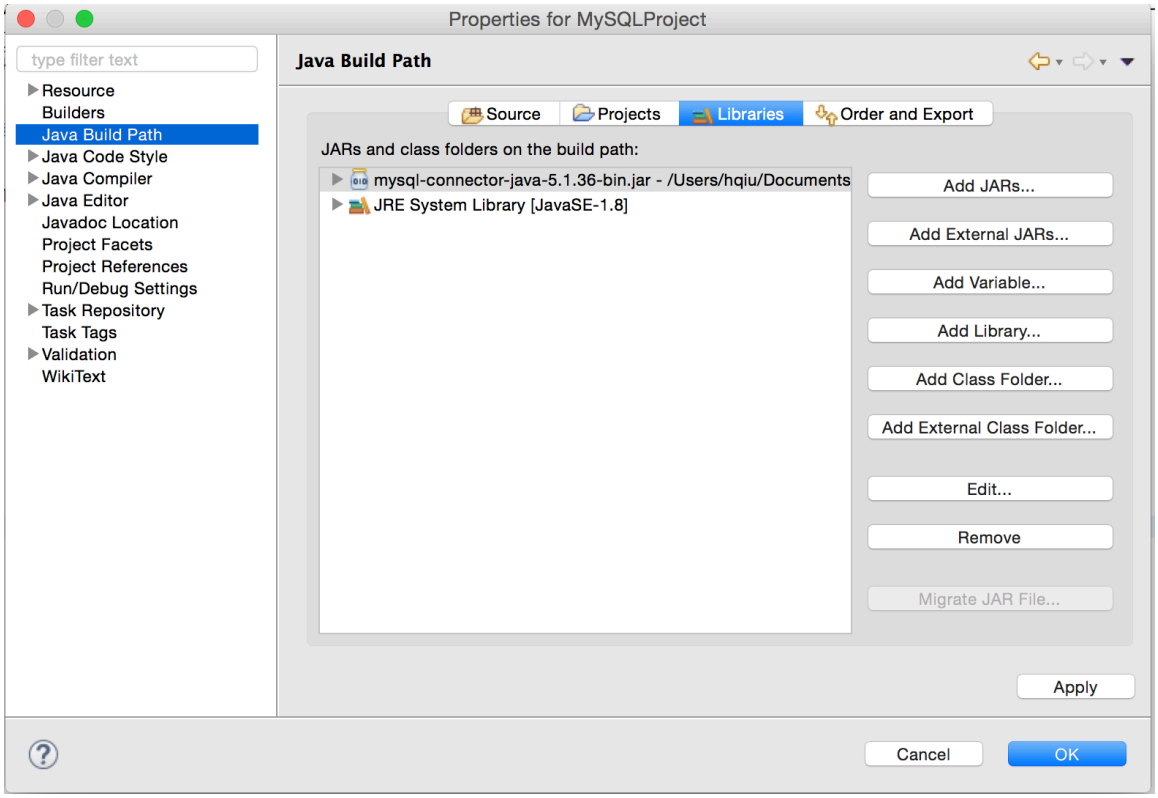
5. Practice with the new database. Query the data. Delete a record and populate some more records.

```
mysql> SELECT * FROM FLOWERS;
+----+--------+--------+----------------------------------------+
| id | name   | height | description                            |
+----+--------+--------+----------------------------------------+
|  1 | Rose   |   14.5 | A woody perennial of the genus Rosa.   |
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.  |
+----+--------+--------+----------------------------------------+
3 rows in set (0.00 sec)

mysql> DELETE FROM FLOWERS WHERE name='Rose';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM FLOWERS;
+----+--------+--------+----------------------------------------+
| id | name   | height | description                            |
+----+--------+--------+----------------------------------------+
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.  |
+----+--------+--------+----------------------------------------+
2 rows in set (0.00 sec)

mysql> INSERT INTO FLOWERS (name, height, description) VALUES ('Rose', 14.5,'A woody perennial of the
genus Rosa.');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM FLOWERS;
+----+--------+--------+----------------------------------------+
| id | name   | height | description                            |
+----+--------+--------+----------------------------------------+
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.  |
|  4 | Rose   |   14.5 | A woody perennial of the genus Rosa.   |
+----+--------+--------+----------------------------------------+
3 rows in set (0.00 sec)

mysql> SELECT height FROM FLOWERS WHERE name='Tulip';
+--------+
| height |
+--------+
|   18.6 |
+--------+
1 row in set (0.00 sec)
```

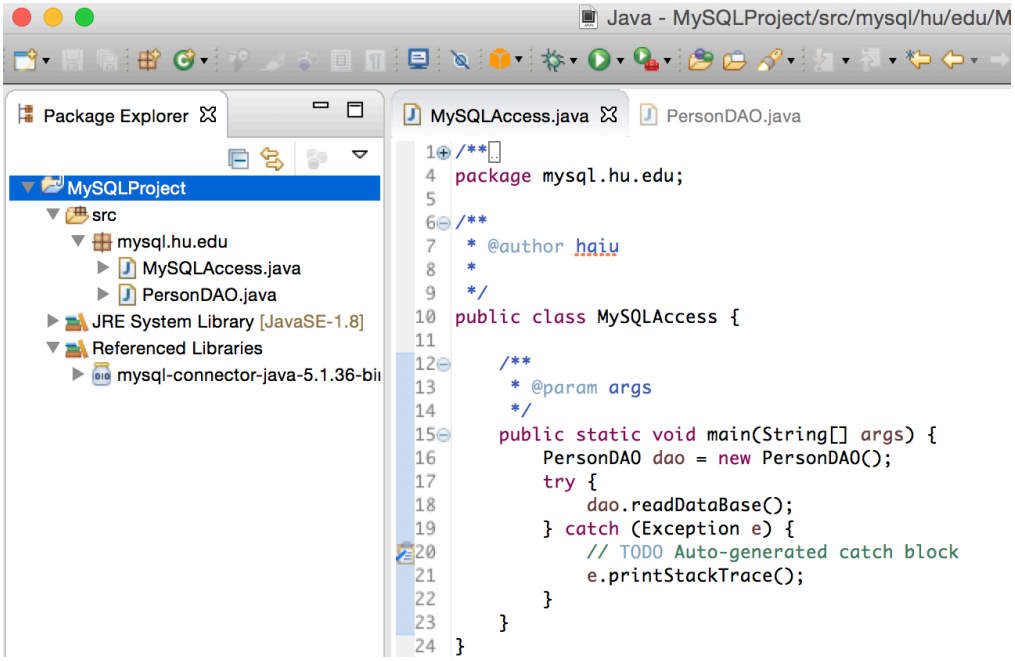
Problem 2

1. Use Java Programming Language and JDBC to operate the MySQL database. Download the MySQL Connector for Java. Create a new project using Eclipse. Configure the library path, include the JAR file for the connector.


Package Explorer
MySQLProject
src
mysql.hu.edu
JRE System Library [JavaSE-1.8]
New Java Class
Java Class
Create a new Java class.
Source folder:
MySQLProject/src
Browse...
Package:
mysql.hu.edu
Enclosing type:
Name:
MySQLAccess
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Cancel
Finish


Properties for MySQLProject
type filter text
Resource
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project Facets
Project References
Run/Debug Settings
Task Repository
Task Tags
Validation
WikiText
Java Build Path
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
mysql-connector-java-5.1.36-bin.jar - /Users/hqiu/Documents
JRE System Library [JavaSE-1.8]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
Apply
Cancel
OK

2. Create package “mysql.hu.edu” and two classes “MySQLAccess” and “PersonDAO” in the new project.



Configure the JDBC URL. Populate and query existing table FLOWERS in my local MySQL DB.

```
public void readDataBase() throws Exception {
    try {
        // This will load the MySQL driver, each DB has its own driver
        Class.forName("com.mysql.jdbc.Driver");

        // Setup the connection with the DB
        connect = DriverManager.getConnection("jdbc:mysql://localhost/hqiuDB?" + "user=hqiu&password=hqiu");
        // connect = DriverManager.getConnection("jdbc:mysql://jelenainst.c5cxb8gzb9wo.us-east-1.rds.amazonaws.co
        //   "user=hqiu&password=hqiu");

        // Statements allow to issue SQL queries to the database
        statement = connect.createStatement();

        // Result set get the result of the SQL query
        resultSet = statement.executeQuery("SELECT * FROM hqiuDB.FLOWERS");
        writeResultSet(resultSet);

        // PreparedStatements can use variables and are more efficient
        preparedStatement = connect.prepareStatement("INSERT INTO hqiuDB.FLOWERS VALUES (DEFAULT, ?, ?, ?)");
        // ("name, height, description");

        // Parameters start with 1
        preparedStatement.setString(1, "Orchid");
        preparedStatement.setDouble(2, 7.6);
        preparedStatement.setString(3, "A diverse and widespread family of flowering plants.");
        preparedStatement.executeUpdate();

        preparedStatement = connect.prepareStatement("INSERT INTO FLOWERS (name, height, description) VALUES "
                + "('Daisy', 5.2,'An herbaceous perennial plant.')");
        preparedStatement.executeUpdate();

        preparedStatement = connect
                .prepareStatement("SELECT name, height, description FROM hqiuDB.FLOWERS");
        resultSet = preparedStatement.executeQuery();
        writeResultSet(resultSet);
```

Run the application. Here are the results from the Eclipse console. I first query all the available rows in the database. Then I populate two new records into the table and query the table again.

```
<terminated> MySQLAccess [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk
Name: Rose
Height: 14.5
Description: A woody perennial of the genus Rosa.
Name: Lilium
Height: 16.7
Description: A genus of herbaceous flowering plants.
Name: Tulip
Height: 18.6
Description: A genus of perennial, bulbous plants.

Name: Rose
Height: 14.5
Description: A woody perennial of the genus Rosa.
Name: Lilium
Height: 16.7
Description: A genus of herbaceous flowering plants.
Name: Tulip
Height: 18.6
Description: A genus of perennial, bulbous plants.
Name: Orchid
Height: 7.6
Description: A diverse and widespread family of flowering plants.
Name: Daisy
Height: 5.2
Description: An herbaceous perennial plant.

The columns in the table are:
Table: flowers
Column 1 name
Column 2 height
Column 3 description
```

Check from the MySQL database.

```
mysql> SELECT * FROM FLOWERS;
+----+--------+--------+-----------------------------------------+
| id | name   | height | description                             |
+----+--------+--------+-----------------------------------------+
|  1 | Rose   |   14.5 | A woody perennial of the genus Rosa.    |
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.   |
+----+--------+--------+-----------------------------------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM FLOWERS;
+----+--------+--------+-----------------------------------------------------------+
| id | name   | height | description                                               |
+----+--------+--------+-----------------------------------------------------------+
|  1 | Rose   |   14.5 | A woody perennial of the genus Rosa.                      |
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants.                   |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.                     |
|  4 | Orchid |    7.6 | A diverse and widespread family of flowering plants.      |
|  5 | Daisy  |    5.2 | An herbaceous perennial plant.                            |
+----+--------+--------+-----------------------------------------------------------+
5 rows in set (0.00 sec)
```

Problem 3

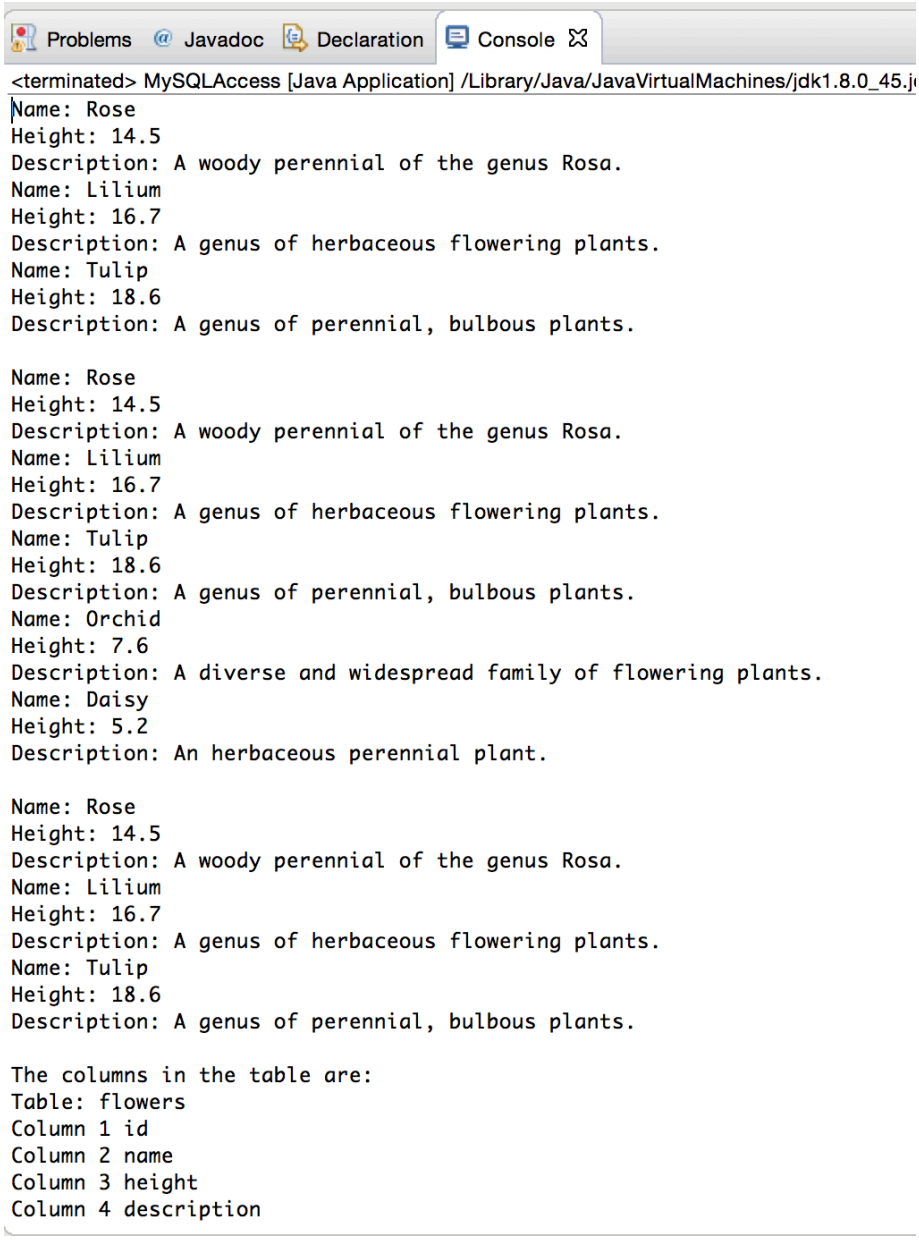
1. Manually delete the table FLOWERS. Create the table through the program. Add a new “prepareStatement” to create tables here. Populate some records into the table. The statement string is the same as we operate through the MySQL command line tool.

```
public void readDataBase() throws Exception {
    try {
        // This will load the MySQL driver, each DB has its own driver
        Class.forName("com.mysql.jdbc.Driver");

        // Setup the connection with the DB
        connect = DriverManager.getConnection("jdbc:mysql://localhost/hqiuDB?" + "user=hqiu&password=hqiu");
        // connect = DriverManager.getConnection("jdbc:mysql://jelenainst.c5cxb8gzb9wo.us-east-1.rds.amazonaws.com:33
        //   "user=hqiu&password=hqiu");

        // Statements allow to issue SQL queries to the database
        statement = connect.createStatement();

        // Problem 3: Create the table
        preparedStatement = connect.prepareStatement("CREATE TABLE FLOWERS (id INT NOT NULL AUTO_INCREMENT, "
                + "name VARCHAR(30) NOT NULL, height DOUBLE NOT NULL, description VARCHAR(128) NOT NULL, "
                + "PRIMARY KEY (ID))");
        preparedStatement.executeUpdate();

        // Problem 3: Insert a few records
        preparedStatement = connect.prepareStatement("INSERT INTO FLOWERS (name, height, description) VALUES "
                + "('Rose', 14.5,'A woody perennial of the genus Rosa.'), "
                + "('Lilium', 16.7, 'A genus of herbaceous flowering plants.'), "
                + "('Tulip', 18.6, 'A genus of perennial, bulbous plants.')");
        preparedStatement.executeUpdate();
```

2. Query the data. Insert another two records. Check if we have successfully inserted the records and then delete them. Method “executeQuery” of class Statement is used for SQL Select statements. Method “executeUpdate” of class Statement is used for SQL Insert, Update and Delete statements.

```
        // Problem 2 & 3: Result set get the result of the SQL query
        resultSet = statement.executeQuery("SELECT * FROM hqiuDB.FLOWERS");
        writeResultSet(resultSet);

        // PreparedStatements can use variables and are more efficient
        // Add more records.
        preparedStatement = connect.prepareStatement("INSERT INTO hqiuDB.FLOWERS VALUES (DEFAULT, ?, ?, ?)");
        // ("name, height, description");

        // Parameters start with 1
        preparedStatement.setString(1, "Orchid");
        preparedStatement.setDouble(2, 7.6);
        preparedStatement.setString(3, "A diverse and widespread family of flowering plants.");
        preparedStatement.executeUpdate();

        preparedStatement = connect.prepareStatement("INSERT INTO FLOWERS (name, height, description) VALUES "
                + "('Daisy', 5.2,'An herbaceous perennial plant.')");
        preparedStatement.executeUpdate();

        preparedStatement = connect.prepareStatement("SELECT name, height, description FROM hqiuDB.FLOWERS");
        resultSet = preparedStatement.executeQuery();
        writeResultSet(resultSet);

        // Remove some records.
        preparedStatement = connect.prepareStatement("DELETE FROM hqiuDB.FLOWERS WHERE name= ?");
        preparedStatement.setString(1, "Orchid");
        preparedStatement.executeUpdate();

        preparedStatement = connect.prepareStatement("DELETE FROM hqiuDB.FLOWERS WHERE name='Daisy'");
        preparedStatement.executeUpdate();

        resultSet = statement.executeQuery("SELECT * FROM hqiuDB.FLOWERS");
        writeResultSet(resultSet);

        writeMetaData(resultSet);

    } catch (Exception e) {
        throw e;
    } finally {
        close();
    }
```

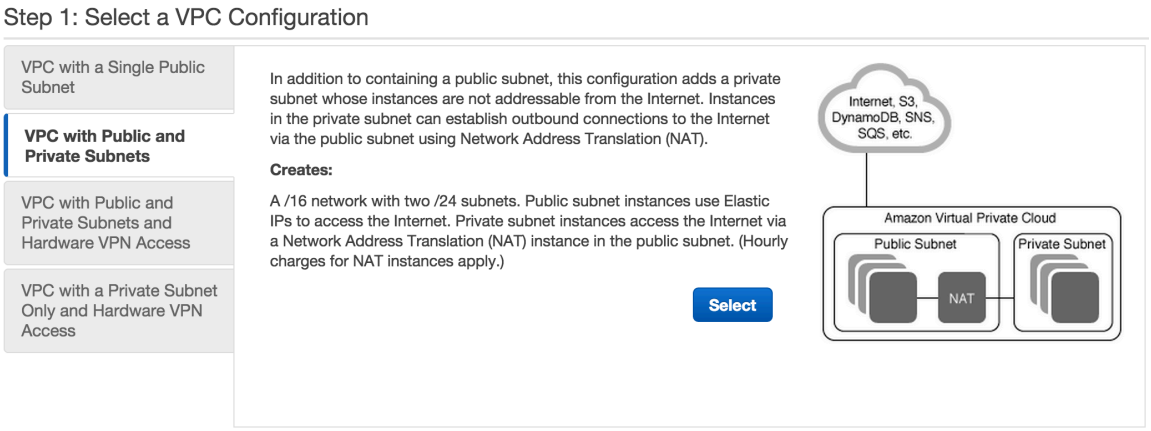
Run the application. Here are the results from the Eclipse console. Double check the results from the MySQL database.

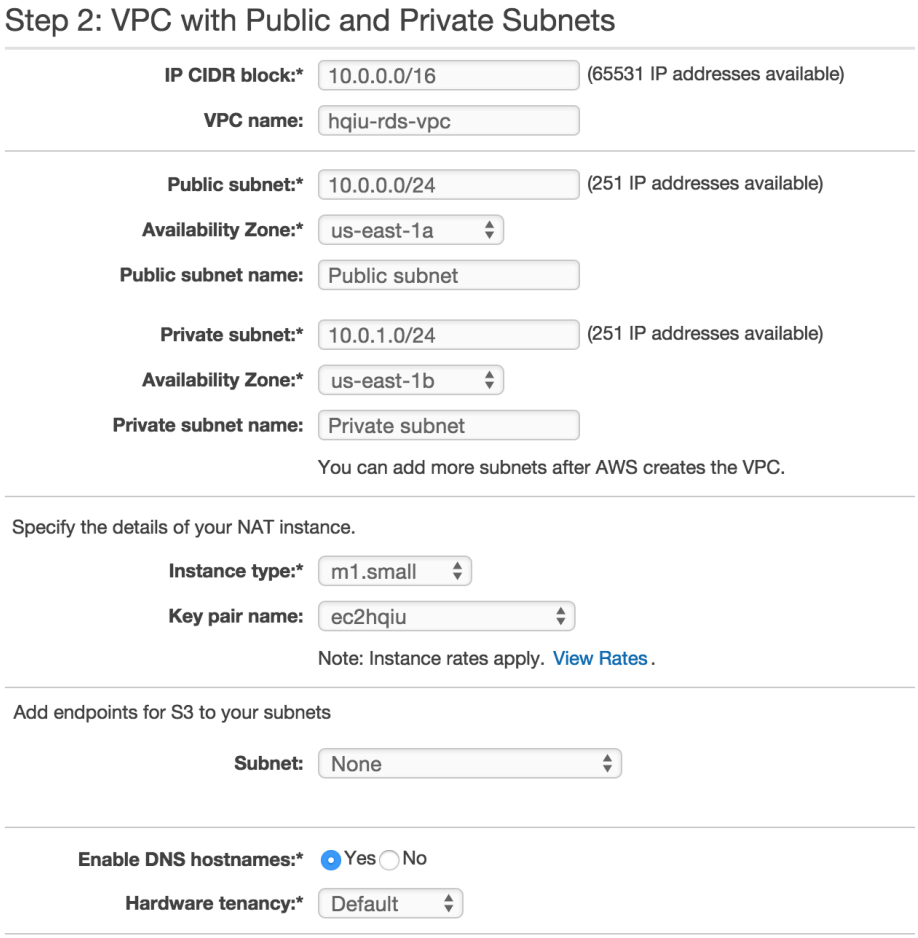


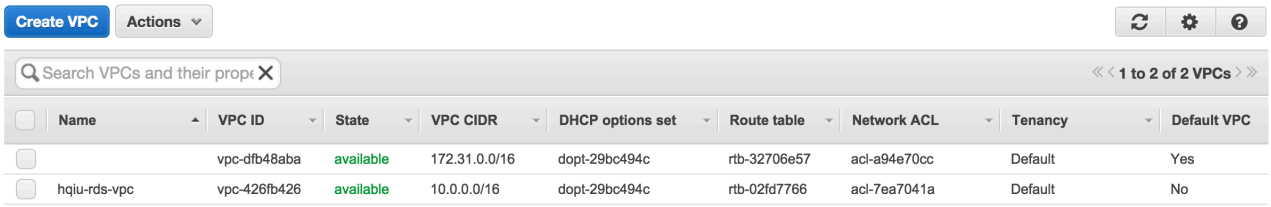

```
mysql> DROP TABLE FLOWERS;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM FLOWERS;
+----+--------+--------+-----------------------------------------+
| id | name   | height | description                             |
+----+--------+--------+-----------------------------------------+
|  1 | Rose   |   14.5 | A woody perennial of the genus Rosa.    |
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.   |
+----+--------+--------+-----------------------------------------+
3 rows in set (0.00 sec)
```

Problem 4

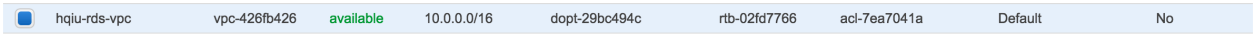
1. Create a new VPC with two subnets (one public and one private). Go to AWS console and navigate to VPC service. Select “Start VPC Wizard”. Select “VPC with Public and Private Subnets”. Select different availability zones for two subnets. Select my key pair and enable DNS hostnames.

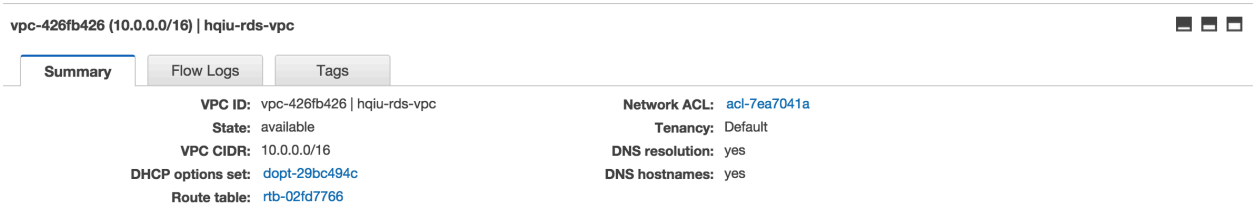




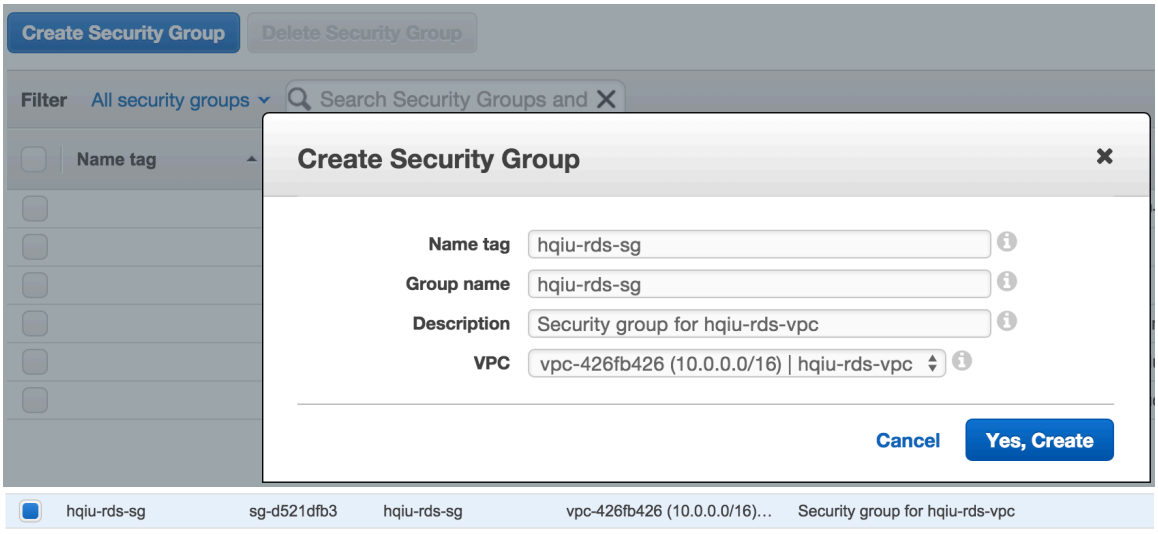


VPC-426fb426 is available.

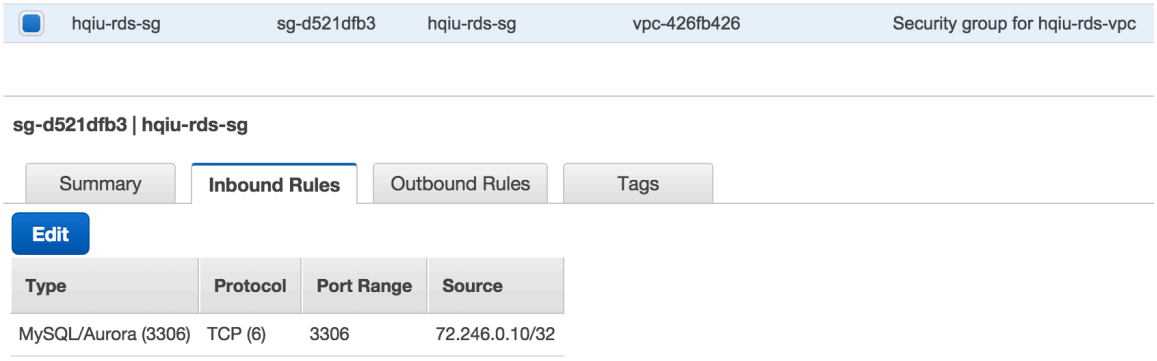




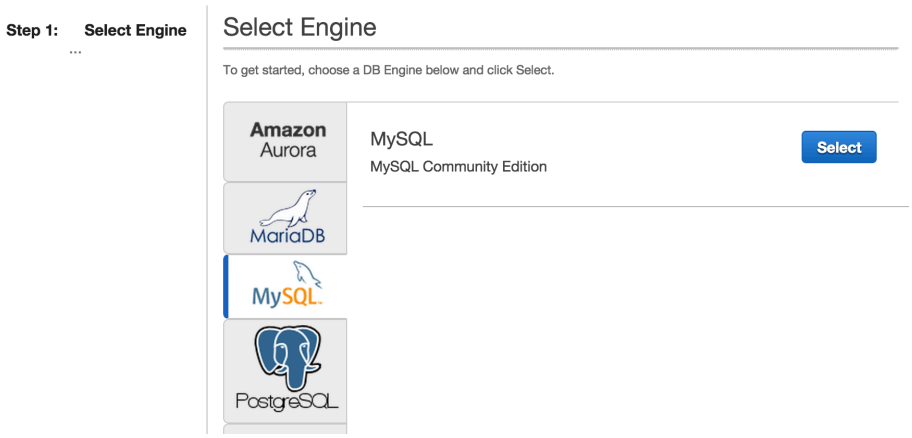
2. Create a new security group associated with my new VPC. Select “Create Security Group”. Select the VPC-426fb426 I’ve just created.

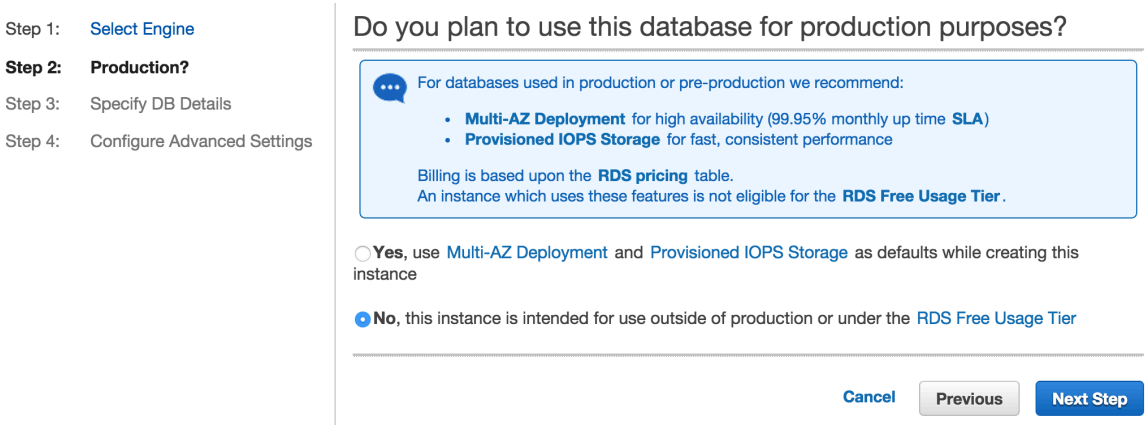


3. Set inbound rules for the security group. Select new group “hqiu-rds-sg” and then add MySQL/Aurora(3306) port. Add the IP address of my machine. “CIDR /32” means an exact match.



4. Create a new micro instance of MySQL database in Amazon RDS service using AWS console. “DB Instance Identifier” is the server name. Select the new VPC and Security Group we just created. Set the “Public Accessible” to “Yes”.





Step 1: Select Engine

Step 2: Production?

Step 3: Specify DB Details

Step 4: Configure Advanced Settings

Your current selection is eligible for the free tier.

Learn More.

Specify DB Details

Instance Specifications

DB Engine mysql

License Model general-public-license

DB Engine Version 5.6.23

Review the **Known Issues/Limitations** to learn about potential compatibility issues with specific database versions.

DB Instance Class db.t2.micro — 1 vCPU, 1 GiB RAM

Multi-AZ Deployment No

Storage Type General Purpose (SSD)

Allocated Storage* 5 GB

Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. **Click here** for more details.

Settings

DB Instance Identifier* hqiuDBIns

Master Username* hqiu

Master Password* ••••••••••

Confirm Password* ••••••••••

Specify a name that is unique for all DB instances owned by your AWS account in the current region. DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance". Learn More.

This instance will be created with the new Certificate Authority rds-ca-2015. If you are using SSL to connect to this instance, you should use the new certificate bundle. Learn more here

VPC* hqiu-rds-vpc (vpc-426fb426)

Subnet Group Create new DB Subnet Group

Publicly Accessible Yes

Availability Zone us-east-1a

VPC Security Group(s)

- Create new Security Group
- default (VPC)
- hqiu-rds-sg (VPC)

Select the security group or groups that have rules authorizing connections from all of the EC2 instances and devices that need to access the data stored in the DB instance. By default, security groups do not authorize any connections; you must specify rules for all instances and devices that will connect to the DB instance. Learn More.

Database Options

Database Name hqiuDB

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Database Port 3306

DB Parameter Group default.mysql5.6

Option Group default:mysql-5-6

Copy Tags To Snapshots

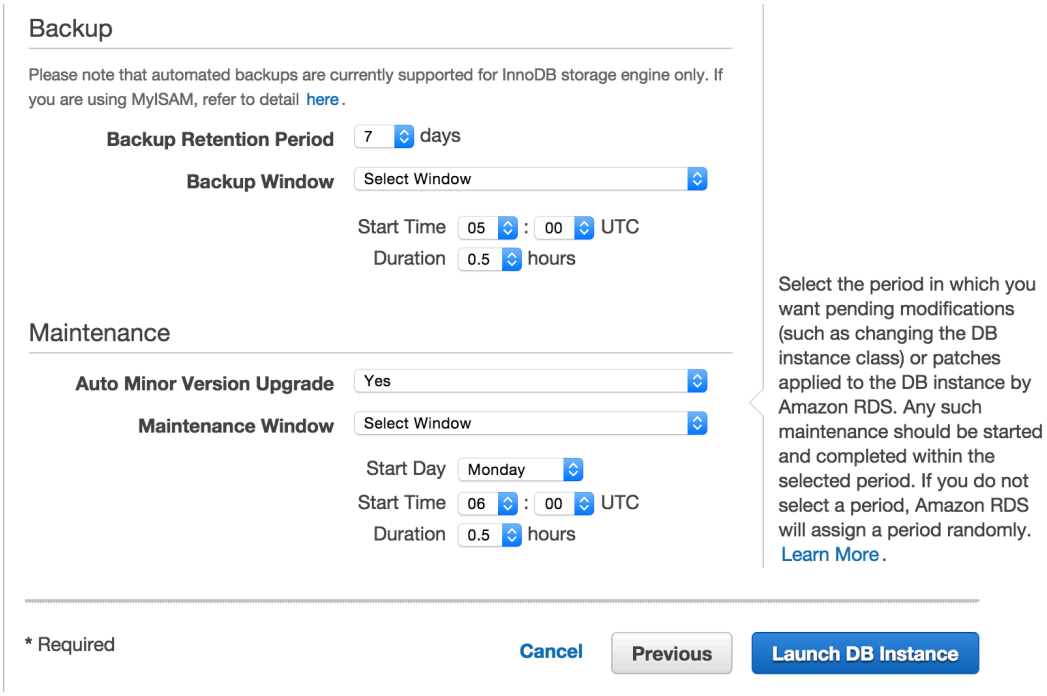
Enable Encryption No

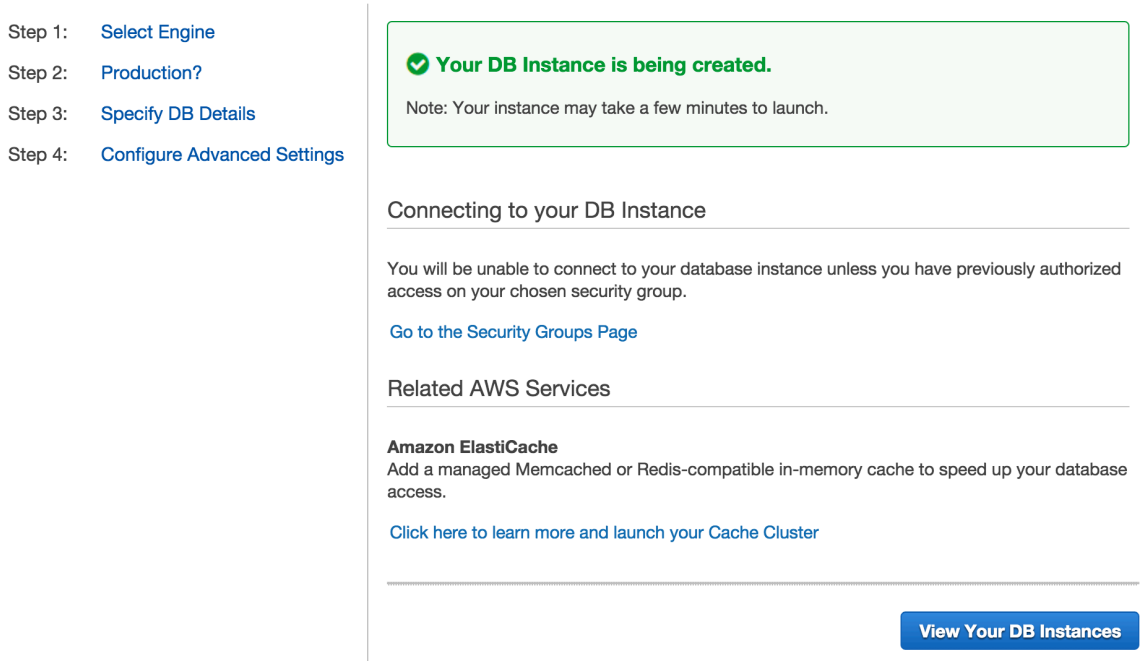
The selected Engine or DB Instance Class does not support storage encryption.

Connection Information

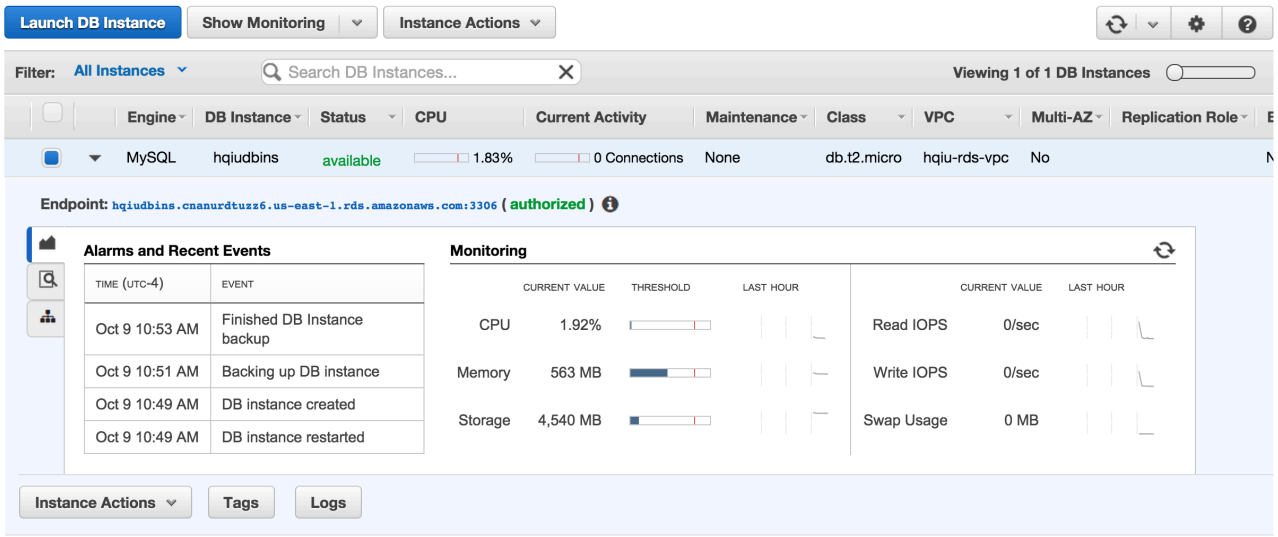
Security Group Rules:

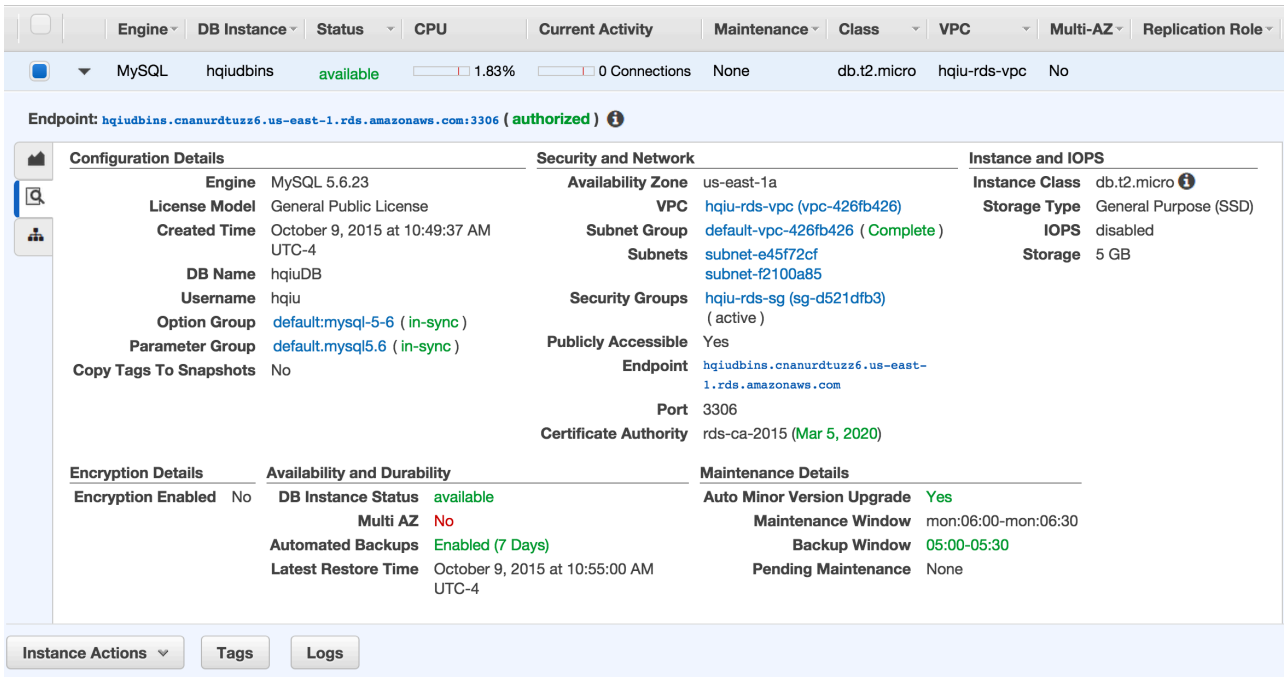
Security Group	Type	Rule
hqiu-rds-sg	CIDR/IP - Inbound	172.28.8.38/32





5. Check the details of the database. The end point is “hqiudbins.cnanurdtuzz6.us-east-1.rds.amazonaws.com”.





6. Connect to the remote RDS database using my local MySQL client. Connect through the end point: hqiudbins.cnanurdtuzz6.us-east-1.rds.amazonaws.com.

```
hqiu@bos-mpdei>> mysql -h hqiudbins.cnanurdtuzz6.us-east-1.rds.amazonaws.com -u hqiu -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 5.6.23-log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| hqiuDB             |
| innodb             |
| mysql              |
| performance_schema |
+--------------------+
5 rows in set (0.02 sec)

mysql> use hqiuDB;
Database changed
mysql> select database();
+------------+
| database() |
+------------+
| hqiuDB     |
+------------+
1 row in set (0.02 sec)

mysql> select user();
+-------------------------------------------------+
| user()                                          |
+-------------------------------------------------+
| hqiu@a72-246-0-10.deploy.akamaitechnologies.com |
+-------------------------------------------------+
1 row in set (0.01 sec)
```

7. Repeat the same operations we've done in Problem 2 & 3.

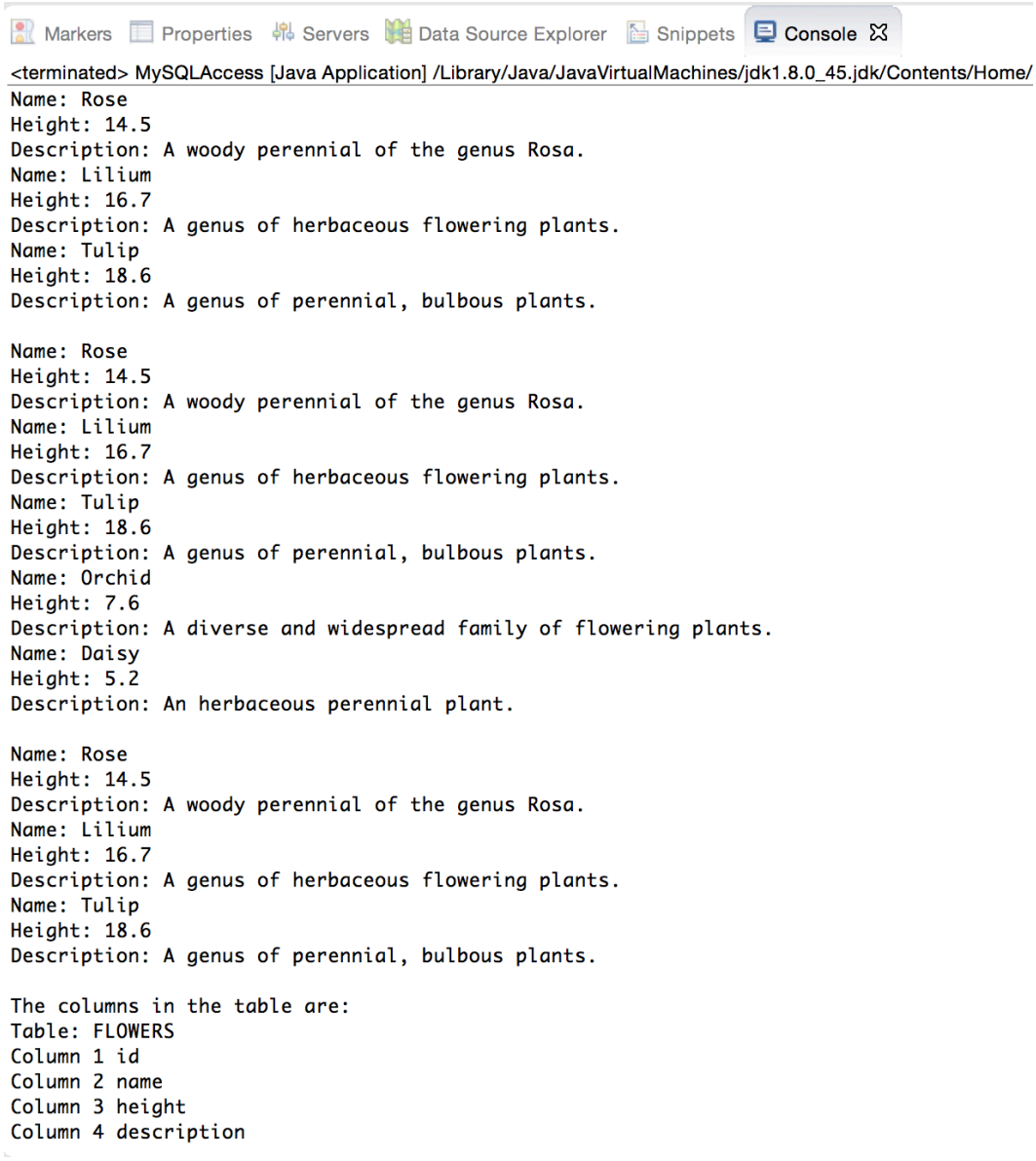
```
hqiu@bos-mpdei>> mysql -h hqiudbins.cnanurdtuzz6.us-east-1.rds.amazonaws.com -u hqiu -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 70
Server version: 5.6.23-log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use hqiuDB;
Database changed
mysql> CREATE TABLE FLOWERS (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> name VARCHAR(30) NOT NULL,
    -> height DOUBLE NOT NULL,
    -> description VARCHAR(128) NOT NULL,
    -> PRIMARY KEY (ID));
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO FLOWERS (name, height, description) VALUES
    -> ('Rose', 14.5,'A woody perennial of the genus Rosa.'),
    -> ('Lilium', 16.7, 'A genus of herbaceous flowering plants.'),
    -> ('Tulip', 18.6, 'A genus of perennial, bulbous plants.');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM FLOWERS;
+----+--------+--------+----------------------------------------+
| id | name   | height | description                            |
+----+--------+--------+----------------------------------------+
|  1 | Rose   |   14.5 | A woody perennial of the genus Rosa.   |
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.  |
+----+--------+--------+----------------------------------------+
3 rows in set (0.03 sec)

mysql> SELECT height FROM FLOWERS WHERE name='Tulip';
+--------+
| height |
+--------+
|   18.6 |
+--------+
1 row in set (0.02 sec)

mysql> DELETE FROM FLOWERS WHERE name='Rose';
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM FLOWERS;
+----+--------+--------+----------------------------------------+
| id | name   | height | description                            |
+----+--------+--------+----------------------------------------+
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.  |
+----+--------+--------+----------------------------------------+
2 rows in set (0.03 sec)
```

Problem 5

1. Modify the code from Problem 3 and connect to the Amazon's RDS database. Change the field "getConnection". Connect to the end point "hqiudbins.cnanurdtuzz6.us-east-1.rds.amazonaws.com". Change to the corresponding username and password. Do the same operations to create a table called "FLOWERS" in the remote RDS database, populate some records, do some query and delete some records.

```
public void readDataBase() throws Exception {
    try {
        // This will load the MySQL driver, each DB has its own driver
        Class.forName("com.mysql.jdbc.Driver");

        // Setup the connection with the DB
        // connect = DriverManager.getConnection("jdbc:mysql://localhost/hqiuDB?" + "user=hqiu&password=hqiu");
        connect = DriverManager.getConnection("jdbc:mysql://hqiudbins.cnanurdtuzz6.us-east-1.rds.amazonaws.com:"
                + "3306/hqiuDB?" + "user=hqiu&password=hqiu890908");

        // Statements allow to issue SQL queries to the database
        statement = connect.createStatement();

        // Problem 3: Create the table
        preparedStatement = connect.prepareStatement("CREATE TABLE FLOWERS (id INT NOT NULL AUTO_INCREMENT, "
                + "name VARCHAR(30) NOT NULL, height DOUBLE NOT NULL, description VARCHAR(128) NOT NULL, "
                + "PRIMARY KEY (ID))");
        preparedStatement.executeUpdate();

        // Problem 3: Insert a few records
        preparedStatement = connect.prepareStatement("INSERT INTO FLOWERS (name, height, description) VALUES "
                + "('Rose', 14.5,'A woody perennial of the genus Rosa.'), "
                + "('Lilium', 16.7, 'A genus of herbaceous flowering plants.'), "
                + "('Tulip', 18.6, 'A genus of perennial, bulbous plants.')");
        preparedStatement.executeUpdate();

        // Problem 2 & 3: Result set get the result of the SQL query
        resultSet = statement.executeQuery("SELECT * FROM hqiuDB.FLOWERS");
        writeResultSet(resultSet);
```

2. Run the application. Here are the results from the Eclipse console. We can successfully connect to the remote AWS RDS database and get the same right results.

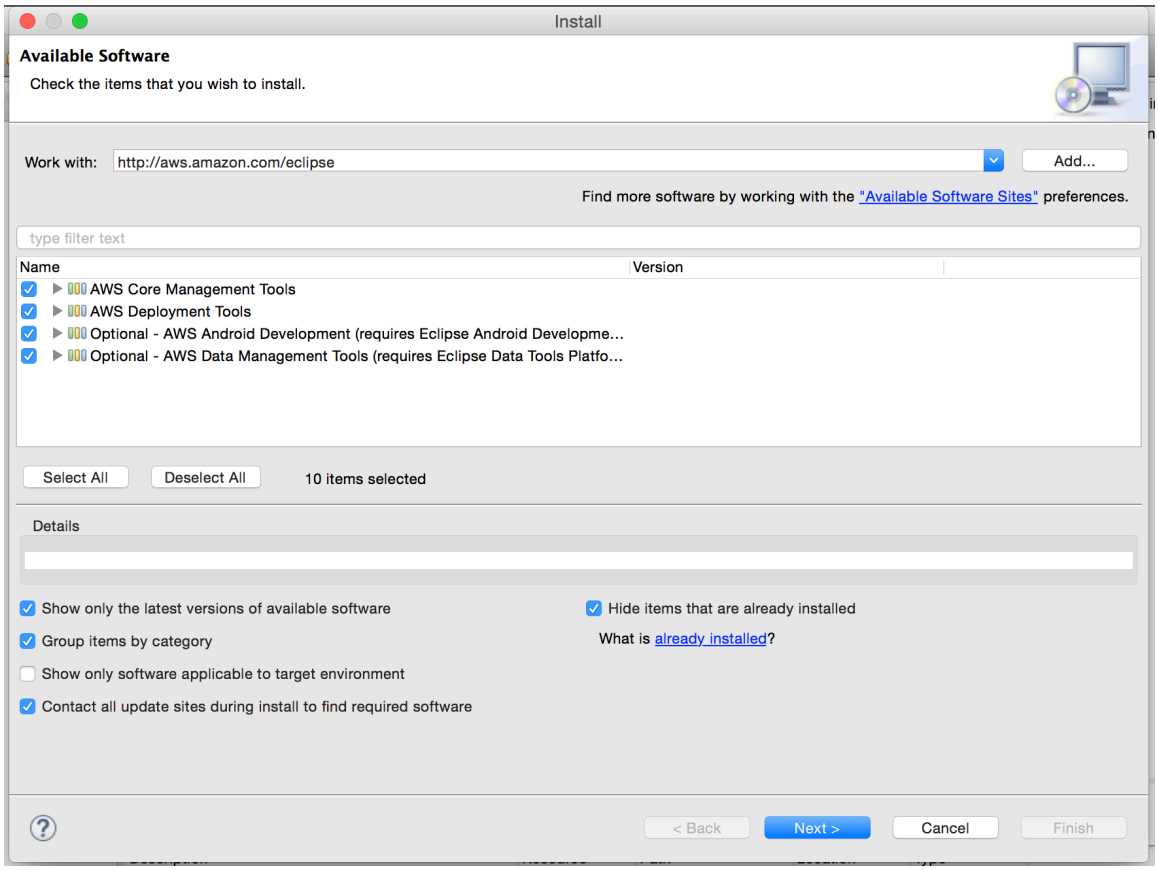


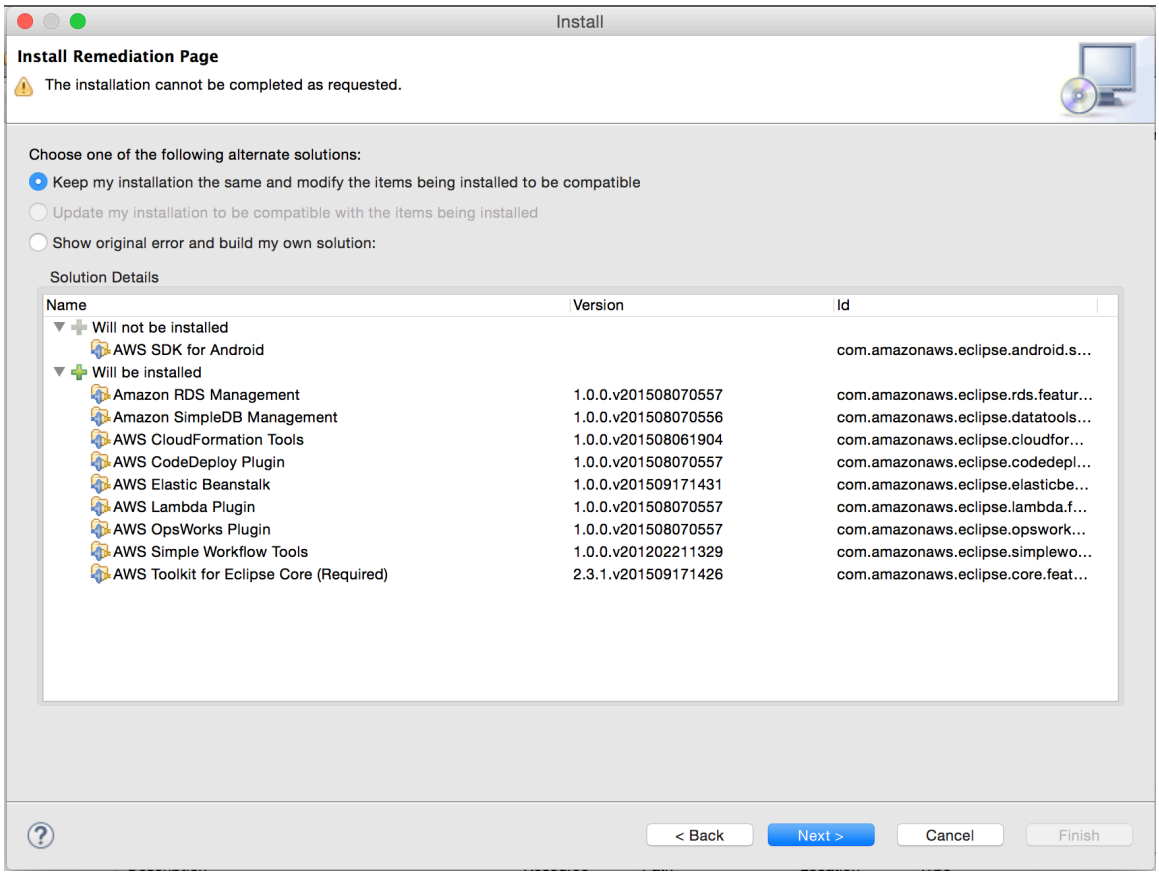
3. Double check from the MySQL command line tool.

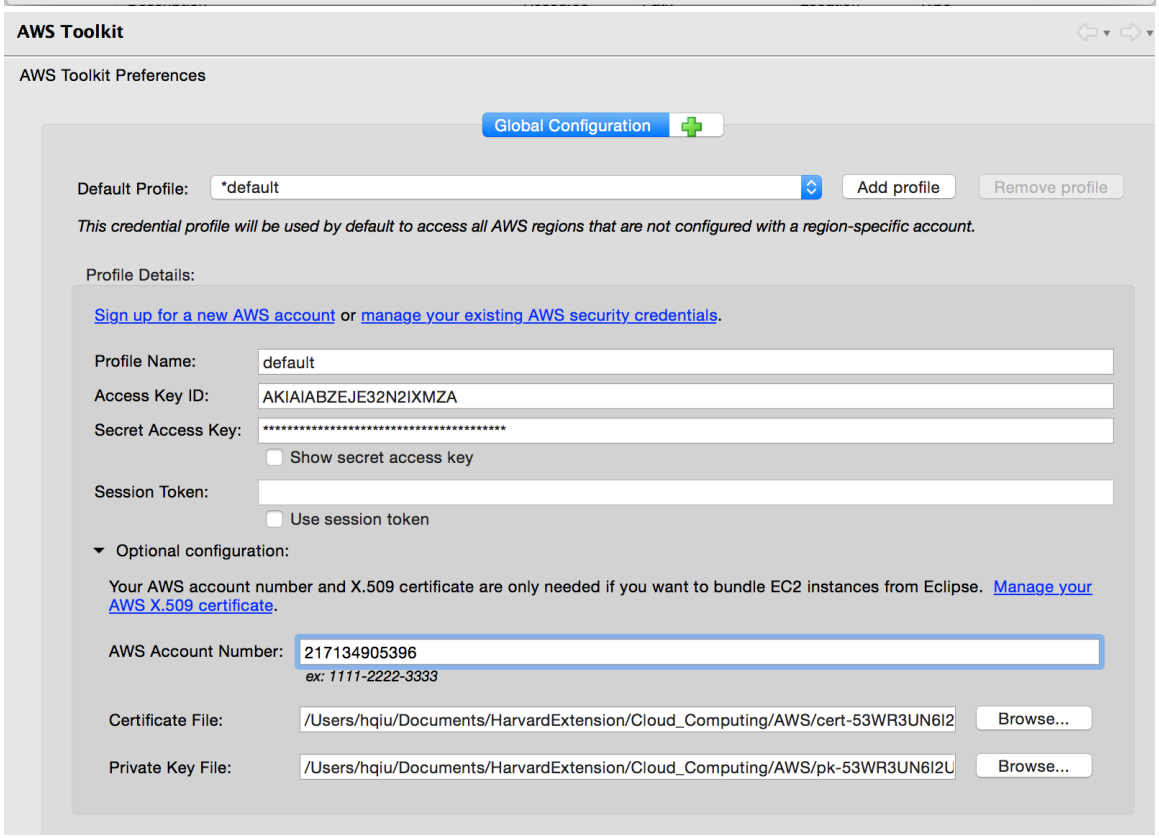
```
hqiu@bos-mpdei>> mysql -h hqiudbins.cnanurdtuzz6.us-east-1.rds.amazonaws.com -u hqiu -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.6.23-log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use hqiuDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------+
| Tables_in_hqiuDB |
+------------------+
| FLOWERS          |
+------------------+
1 row in set (0.02 sec)

mysql> SELECT * FROM FLOWERS;
+----+--------+--------+----------------------------------------+
| id | name   | height | description                            |
+----+--------+--------+----------------------------------------+
|  1 | Rose   |   14.5 | A woody perennial of the genus Rosa.   |
|  2 | Lilium |   16.7 | A genus of herbaceous flowering plants. |
|  3 | Tulip  |   18.6 | A genus of perennial, bulbous plants.  |
+----+--------+--------+----------------------------------------+
3 rows in set (0.02 sec)
```

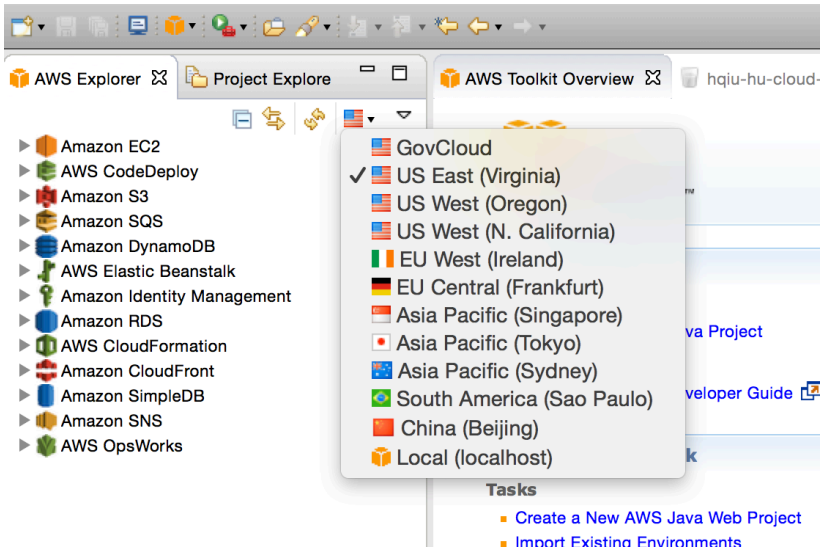
Problem 6

1. First install the AWS toolkit into Eclipse. Set up the security information. It is important to set my default region to “US East (Virginia)”.

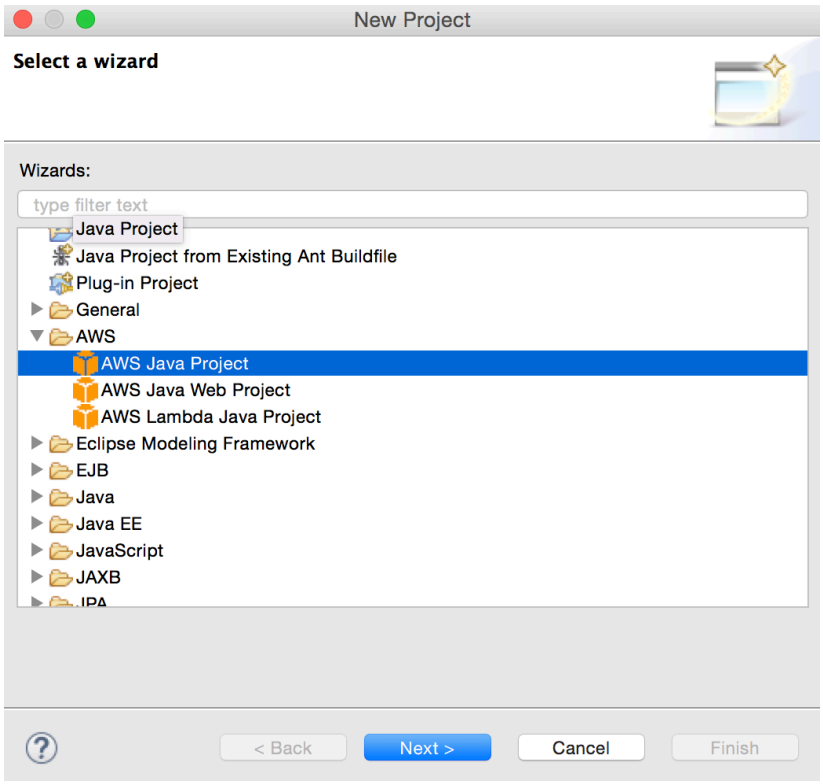

Install
Available Software
Check the items that you wish to install.
Work with:
http://aws.amazon.com/eclipse
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
AWS Core Management Tools
AWS Deployment Tools
Optional - AWS Android Development (requires Eclipse Android Developme...
Optional - AWS Data Management Tools (requires Eclipse Data Tools Platfo...
Select All
Deselect All
10 items selected
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Cancel
Finish


Install
Install Remediation Page
The installation cannot be completed as requested.
Choose one of the following alternate solutions:
Keep my installation the same and modify the items being installed to be compatible
Update my installation to be compatible with the items being installed
Show original error and build my own solution:
Solution Details
Name
Version
Id
Will not be installed
AWS SDK for Android
com.amazonaws.eclipse.android.s...
Will be installed
Amazon RDS Management
1.0.0.v201508070557
com.amazonaws.eclipse.rds.featur...
Amazon SimpleDB Management
1.0.0.v201508070556
com.amazonaws.eclipse.datatools...
AWS CloudFormation Tools
1.0.0.v201508061904
com.amazonaws.eclipse.cloudfor...
AWS CodeDeploy Plugin
1.0.0.v201508070557
com.amazonaws.eclipse.codedepl...
AWS Elastic Beanstalk
1.0.0.v201509171431
com.amazonaws.eclipse.elasticbe...
AWS Lambda Plugin
1.0.0.v201508070557
com.amazonaws.eclipse.lambda.f...
AWS OpsWorks Plugin
1.0.0.v201508070557
com.amazonaws.eclipse.opswork...
AWS Simple Workflow Tools
1.0.0.v201202211329
com.amazonaws.eclipse.simplewo...
AWS Toolkit for Eclipse Core (Required)
2.3.1.v201509171426
com.amazonaws.eclipse.core.feat...
< Back
Next >
Cancel
Finish


AWS Toolkit
AWS Toolkit Preferences
Global Configuration
Default Profile:
*default
Add profile
Remove profile
This credential profile will be used by default to access all AWS regions that are not configured with a region-specific account.
Profile Details:
Sign up for a new AWS account or manage your existing AWS security credentials.
Profile Name:
default
Access Key ID:
AKIAIABZEJE32N2IXMZA
Secret Access Key:
Show secret access key
Session Token:
Use session token
Optional configuration:
Your AWS account number and X.509 certificate are only needed if you want to bundle EC2 instances from Eclipse. Manage your AWS X.509 certificate.
AWS Account Number:
217134905396
ex: 1111-2222-3333
Certificate File:
/Users/hqiu/Documents/HarvardExtension/Cloud_Computing/AWS/cert-53WR3UN6I2
Browse...
Private Key File:
/Users/hqiu/Documents/HarvardExtension/Cloud_Computing/AWS/pk-53WR3UN6I2U
Browse...

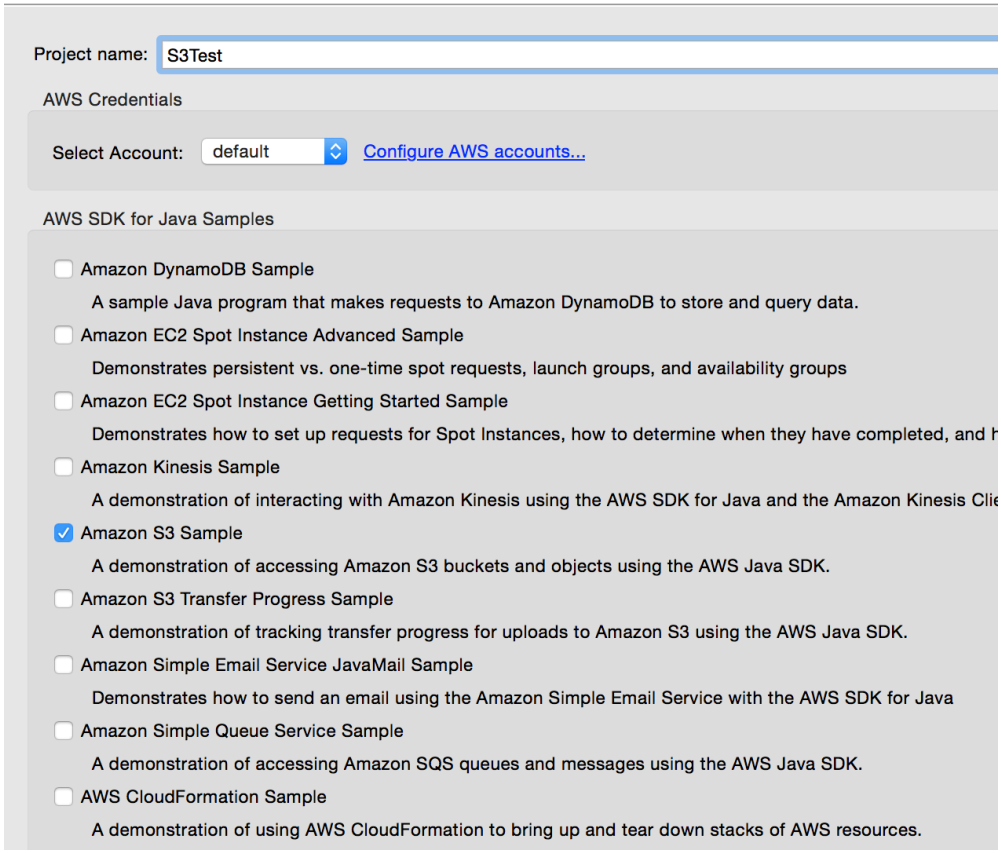


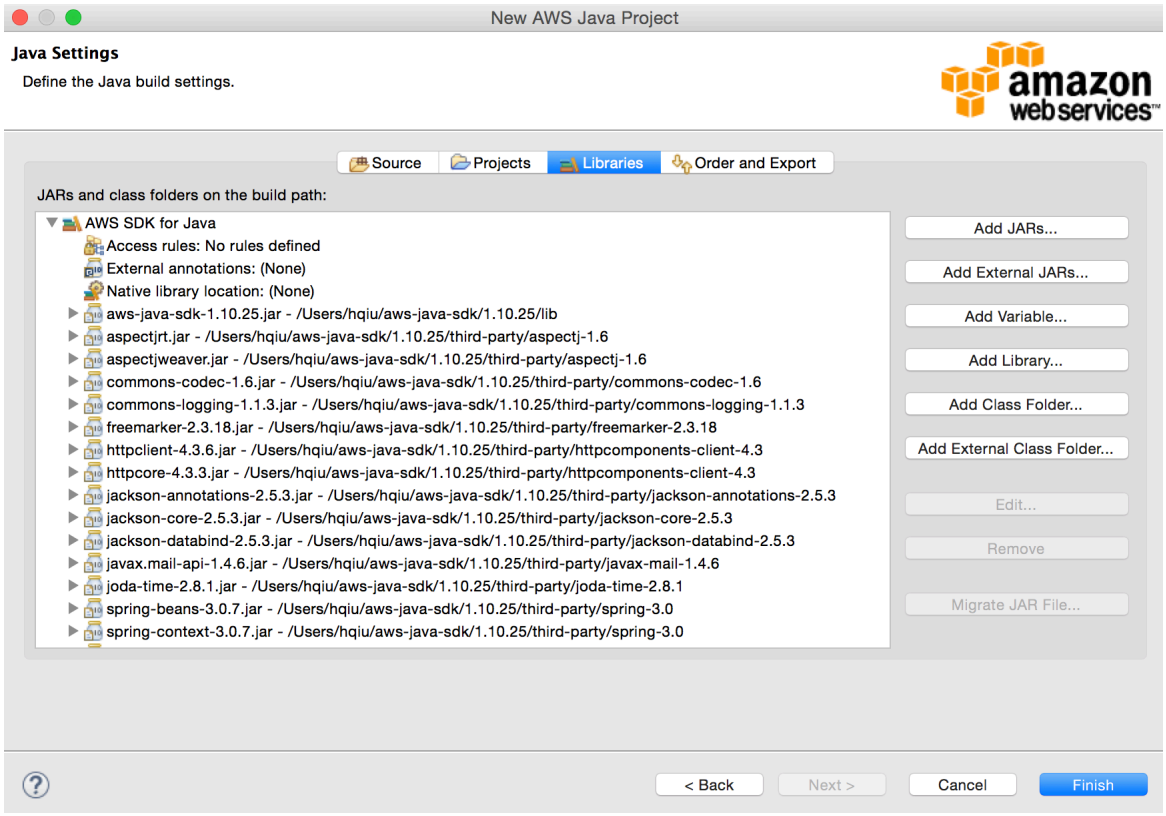
2. Create a new AWS Java Project. Give a project name, select S3 Sample Template. Examine the JRE libraries.



Create an AWS Java project

Create a new AWS Java project in the workspace





3. Open the sample code. The template code tries to create a new bucket, upload some new objects, lists the object in the buckets, delete the objects and the bucket. Modify this code. Try with a single S3 bucket folder with several object files. First change the region to “US_EAST_1”.

```
AmazonS3 s3 = new AmazonS3Client(credentials);
Region usEast1 = Region.getRegion(Regions.US_EAST_1);
s3.setRegion(usEast1);
```

Upload a batch of files using method “PutObjectRequest()” like the example. The “key” is the “key” field of the object. The key is basically just the previous key following by some number.

```
/*
 * Upload an object to your bucket - You can easily upload a file to
 * S3, or upload directly an InputStream if you know the length of
 * the data in the stream. You can also specify your own metadata
 * when uploading to S3, which allows you set a variety of options
 * like content-type and content-encoding, plus additional metadata
 * specific to your applications.
 */
System.out.println("Uploading a new object to S3 from a file\n");
s3.putObject(new PutObjectRequest(bucketName, key, createSampleFile()));

System.out.println("Uploading some other objects to S3 from a file\n");
for (int i = 2; i <= 5; i++) {
    String key2 = key + i;
    s3.putObject(new PutObjectRequest(bucketName, key2, createSampleFile()));
}
```

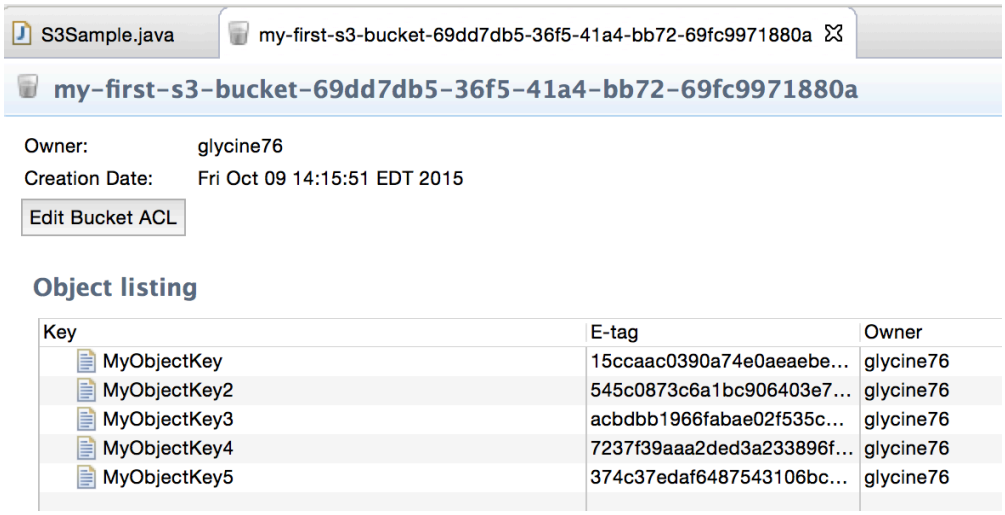
Download and show the content of the uploaded files.

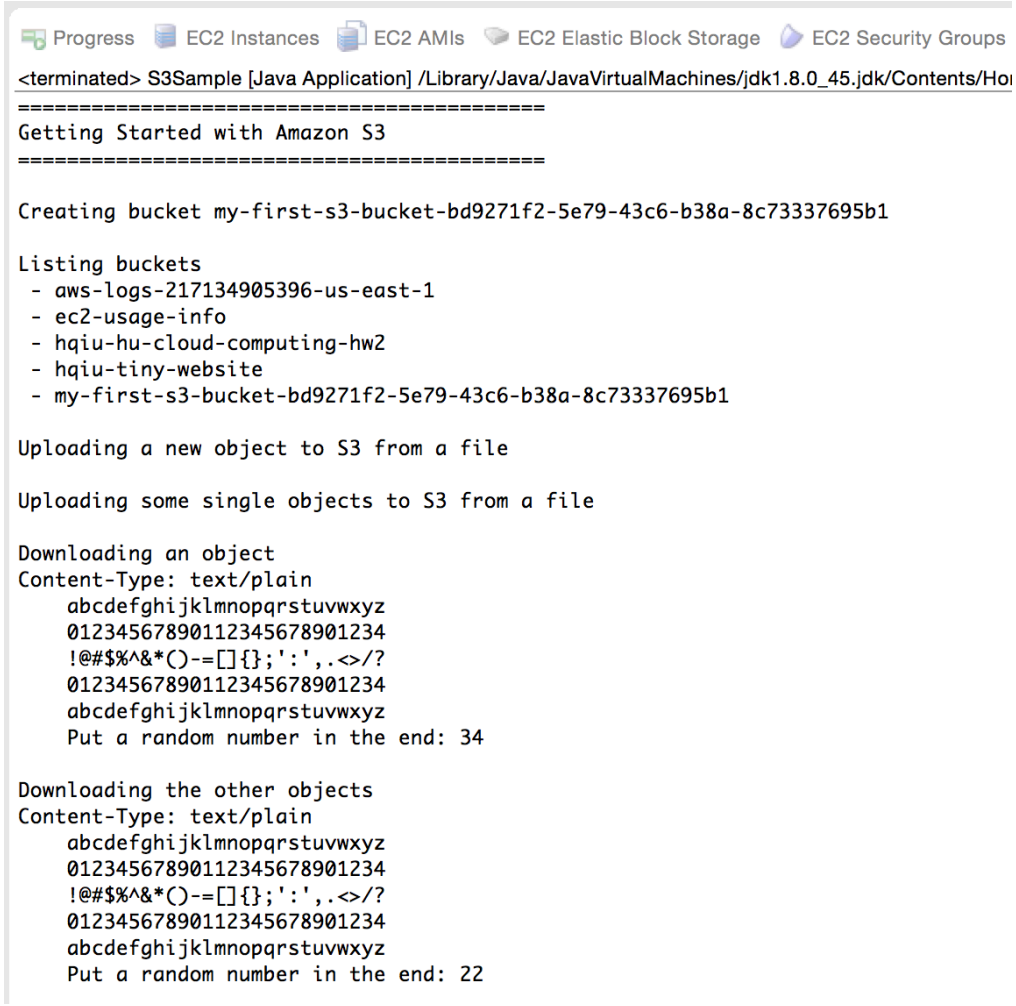
```
/*
 * Download an object - When you download an object, you get all of
 * the object's metadata and a stream from which to read the contents.
 * It's important to read the contents of the stream as quickly as
 * possibly since the data is streamed directly from Amazon S3 and your
 * network connection will remain open until you read all the data or
 * close the input stream.
 *
 * GetObjectRequest also supports several other options, including
 * conditional downloading of objects based on modification times,
 * ETags, and selectively downloading a range of an object.
 */
System.out.println("Downloading an object");
S3Object object = s3.getObject(new GetObjectRequest(bucketName, key));
System.out.println("Content-Type: "  + object.getObjectMetadata().getContentType());
displayTextInputStream(object.getObjectContent());

System.out.println("Downloading the other objects");
for (int i = 2; i <= 5; i++) {
    String key2 = key + i;
    S3Object object2 = s3.getObject(new GetObjectRequest(bucketName, key2));
    System.out.println("Content-Type: "  + object2.getObjectMetadata().getContentType());
    displayTextInputStream(object2.getObjectContent());
}
```

Delete the uploaded objects based on the bucket name and key.

```
/*
 * Delete an object - Unless versioning has been turned on for your bucket,
 * there is no way to undelete an object, so use caution when deleting objects.
 */
System.out.println("Deleting the objects\n");
s3.deleteObject(bucketName, key);
for (int i = 2; i <= 5; i++) {
    String key2 = key + i;
    s3.deleteObject(bucketName, key2);
}
```

4. Check the output from the AWS console. To validate the results, I also temporarily commented out the “delete” operation and check from the “Amazon S3” to make sure that I did upload the files as I expected.

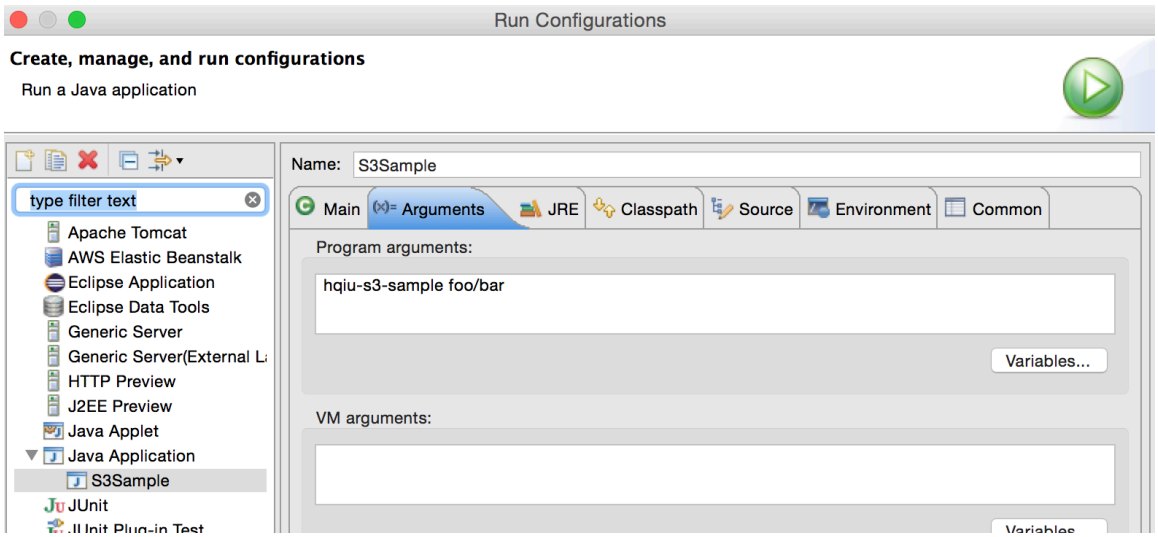

S3Sample.java
my-first-s3-bucket-69dd7db5-36f5-41a4-bb72-69fc9971880a
my-first-s3-bucket-69dd7db5-36f5-41a4-bb72-69fc9971880a
Owner: glycine76
Creation Date: Fri Oct 09 14:15:51 EDT 2015
Edit Bucket ACL
Object listing
Key
E-tag
Owner
MyObjectKey 15ccaac0390a74e0aeaebe... glycine76
MyObjectKey2 545c0873c6a1bc906403e7... glycine76
MyObjectKey3 acbdbb1966fabae02f535c... glycine76
MyObjectKey4 7237f39aaa2ded3a233896f... glycine76
MyObjectKey5 374c37edaf6487543106bc... glycine76


Progress
EC2 Instances
EC2 AMIs
EC2 Elastic Block Storage
EC2 Security Groups
<terminated> S3Sample [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Ho
==========
Getting Started with Amazon S3
==========
Creating bucket my-first-s3-bucket-bd9271f2-5e79-43c6-b38a-8c73337695b1
Listing buckets
- aws-logs-217134905396-us-east-1
- ec2-usage-info
- hqiu-hu-cloud-computing-hw2
- hqiu-tiny-website
- my-first-s3-bucket-bd9271f2-5e79-43c6-b38a-8c73337695b1
Uploading a new object to S3 from a file
Uploading some single objects to S3 from a file
Downloading an object
Content-Type: text/plain
abcdefghijklmnopqrstuvwxyz
01234567890112345678901234
!@#$%^&*()-=[]{};':',.<>/?
01234567890112345678901234
abcdefghijklmnopqrstuvwxyz
Put a random number in the end: 34
Downloading the other objects
Content-Type: text/plain
abcdefghijklmnopqrstuvwxyz
01234567890112345678901234
!@#$%^&*()-=[]{};':',.<>/?
01234567890112345678901234
abcdefghijklmnopqrstuvwxyz
Put a random number in the end: 22

```
Content-Type: text/plain
    abcdefghijklmnopqrstuvwxyz
    01234567890112345678901234
    !@#$%^&*()-=[]{};':',.<>/?
    01234567890112345678901234
    abcdefghijklmnopqrstuvwxyz
    Put a random number in the end: 92

Content-Type: text/plain
    abcdefghijklmnopqrstuvwxyz
    01234567890112345678901234
    !@#$%^&*()-=[]{};':',.<>/?
    01234567890112345678901234
    abcdefghijklmnopqrstuvwxyz
    Put a random number in the end: 77

Content-Type: text/plain
    abcdefghijklmnopqrstuvwxyz
    01234567890112345678901234
    !@#$%^&*()-=[]{};':',.<>/?
    01234567890112345678901234
    abcdefghijklmnopqrstuvwxyz
    Put a random number in the end: 39

Listing objects with prefix "My"
 - MyObjectKey  (size = 170)
 - MyObjectKey2  (size = 170)
 - MyObjectKey3  (size = 170)
 - MyObjectKey4  (size = 170)
 - MyObjectKey5  (size = 170)

Deleting the objects

Deleting bucket my-first-s3-bucket-bd9271f2-5e79-43c6-b38a-8c73337695b1
```

5. Modify the code to make it handle any arbitrary nested folder structure with objects of any S3 bucket. I set the “bucketName” and the nested “folder” as the input arguments. Here I give “bucketName” to “hqiu-s3-example” and set “folder” to “foo/bar”.

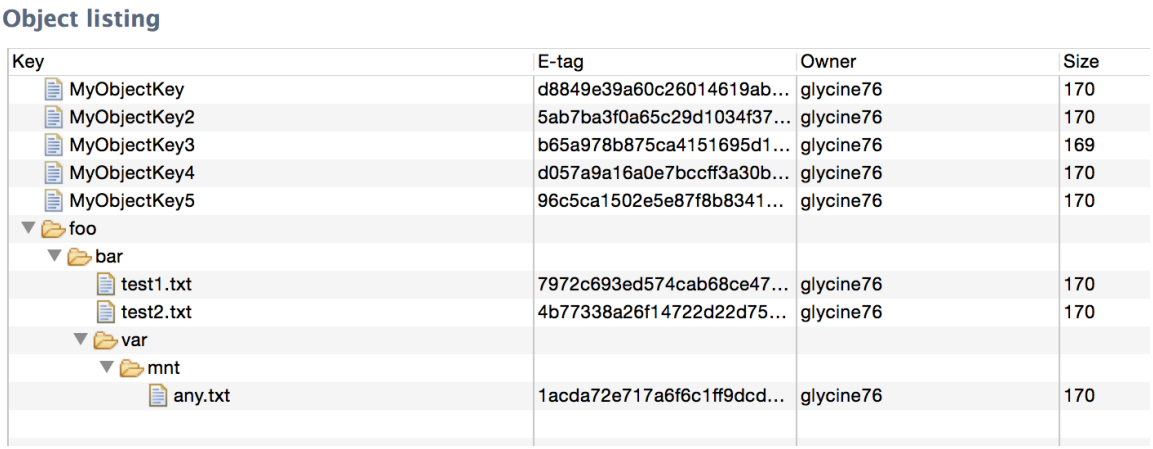
```java
public class S3Sample {

    public static void main(String[] args) throws IOException {

        String bucketName = null;
        String folder = null;

        if (args.length < 2) {
            System.out.println("Usage: S3Sample <bucketName> <folder>");
            System.exit(1);
        } else {
            // Get the input bucketName from outside.
            bucketName = args[0];
            folder = args[1];
        }
```



6. Upload any arbitrary file under any nested folder structure. We just need to give the full path to the "key". The API "putObject()" will create the intermediate path for us. Here I created two files under "foo/bar" and another file under "foo/bar/var/mnt".

```
/*
 * Upload an object to your bucket - You can easily upload a file to
 * S3, or upload directly an InputStream if you know the length of
 * the data in the stream. You can also specify your own metadata
 * when uploading to S3, which allows you set a variety of options
 * like content-type and content-encoding, plus additional metadata
 * specific to your applications.
 */
System.out.println("Uploading a new object to S3 from a file\n");
s3.putObject(new PutObjectRequest(bucketName, key, createSampleFile()));

/*
 * Problem 6 Part 1:
 */
System.out.println("Uploading some single objects to S3 from a file\n");
for (int i = 2; i <= 5; i++) {
    String key2 = key + i;
    s3.putObject(new PutObjectRequest(bucketName, key2, createSampleFile()));
}

/*
 * Problem 6 Part 2:
 */
System.out.println("Uploading some nested objects/folder structures to S3\n");
for (int i = 1; i <= 2; i++) {
    String key3 = folder + "/test" + i + ".txt";
    s3.putObject(new PutObjectRequest(bucketName, key3, createSampleFile()));
}

String key4 = folder + "/var/mnt/any.txt";
s3.putObject(new PutObjectRequest(bucketName, key4, createSampleFile()));
```

Check the file structure from the “Amazon S3”.



Object listing

Key	E-tag	Owner	Size
MyObjectKey	d8849e39a60c26014619ab...	glycine76	170
MyObjectKey2	5ab7ba3f0a65c29d1034f37...	glycine76	170
MyObjectKey3	b65a978b875ca4151695d1...	glycine76	169
MyObjectKey4	d057a9a16a0e7bccff3a30b...	glycine76	170
MyObjectKey5	96c5ca1502e5e87f8b8341...	glycine76	170
foo			
bar			
test1.txt	7972c693ed574cab68ce47...	glycine76	170
test2.txt	4b77338a26f14722d22d75...	glycine76	170
var			
mnt			
any.txt	1acda72e717a6f6c1ff9dcd...	glycine76	170

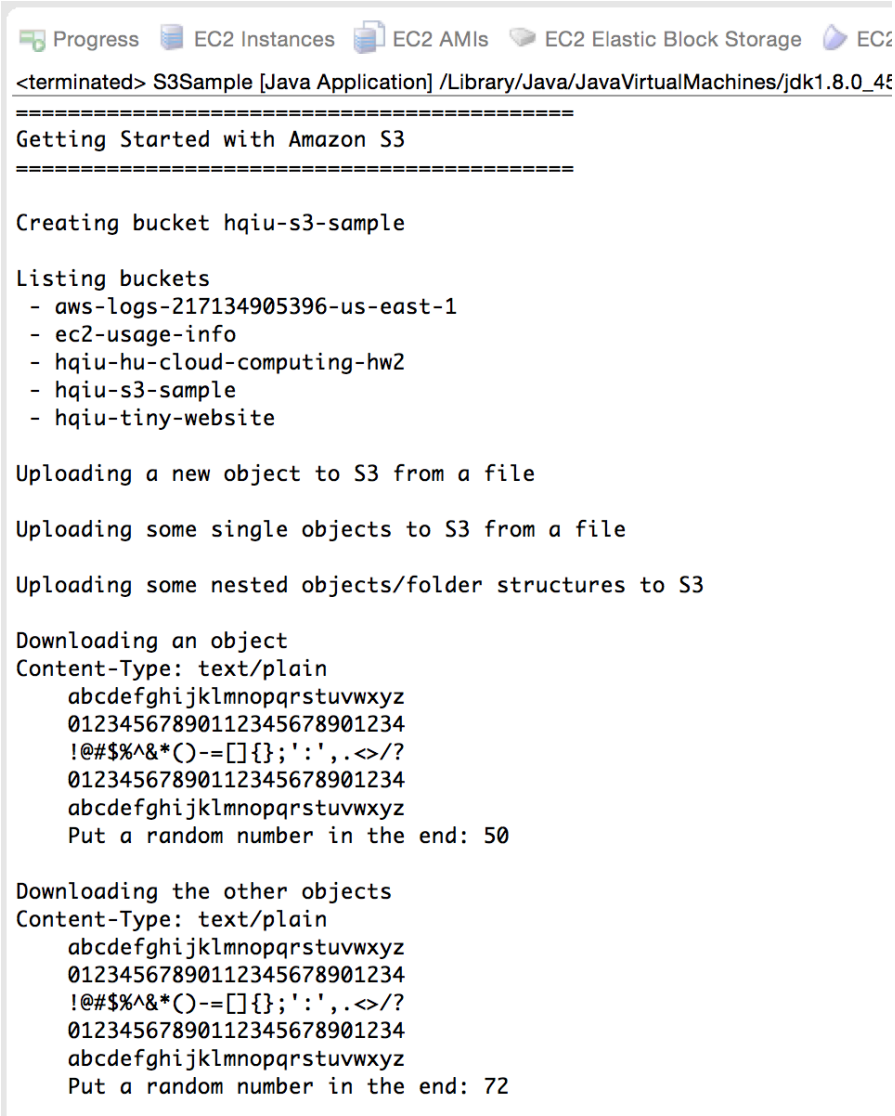
7. List the object under any arbitrary path/structure. Use the method “withPrefix()” to select the objects in the Amazon S3 bucket where the key begins with the given prefix.

```
/*
 * Problem 6 Part 2:
 */
System.out.println("Listing objects with prefix \"" + folder + "\"");
objectListing = s3.listObjects(new ListObjectsRequest()
        .withBucketName(bucketName)
        .withPrefix(folder));
for (S3ObjectSummary objectSummary : objectListing.getObjectSummaries()) {
    System.out.println(" - " + objectSummary.getKey() + "  " +
                       "(size = " + objectSummary.getSize() + ")");
}
System.out.println();

System.out.println("Listing all objects");
objectListing = s3.listObjects(new ListObjectsRequest().withBucketName(bucketName));
for (S3ObjectSummary objectSummary : objectListing.getObjectSummaries()) {
    System.out.println(" - " + objectSummary.getKey() + "  " +
                       "(size = " + objectSummary.getSize() + ")");
}
System.out.println();
```

Delete the whole nested folder structure. First query the object list and get the summary and key of the object. Then delete the object based on the bucketName and object key.

```
/*
 * Problem 6 Part 2: Delete nested folder structure.
 */
objectListing = s3.listObjects(new ListObjectsRequest()
        .withBucketName(bucketName)
        .withPrefix(folder));
for (S3ObjectSummary objectSummary : objectListing.getObjectSummaries()) {
    s3.deleteObject(bucketName, objectSummary.getKey());
}
```

8. Check the output from the Eclipse console.



```
Downloading the other objects
Content-Type: text/plain
    abcdefghijklmnopqrstuvwxyz
    01234567890112345678901234
    !@#$%^&*()-=[]{};':',.<>/?
    01234567890112345678901234
    abcdefghijklmnopqrstuvwxyz
    Put a random number in the end: 72

Content-Type: text/plain
    abcdefghijklmnopqrstuvwxyz
    01234567890112345678901234
    !@#$%^&*()-=[]{};':',.<>/?
    01234567890112345678901234
    abcdefghijklmnopqrstuvwxyz
    Put a random number in the end: 44

Content-Type: text/plain
    abcdefghijklmnopqrstuvwxyz
    01234567890112345678901234
    !@#$%^&*()-=[]{};':',.<>/?
    01234567890112345678901234
    abcdefghijklmnopqrstuvwxyz
    Put a random number in the end: 82

Content-Type: text/plain
    abcdefghijklmnopqrstuvwxyz
    01234567890112345678901234
    !@#$%^&*()-=[]{};':',.<>/?
    01234567890112345678901234
    abcdefghijklmnopqrstuvwxyz
    Put a random number in the end: 48
```

```
Listing objects with prefix "My"
 - MyObjectKey  (size = 170)
 - MyObjectKey2  (size = 170)
 - MyObjectKey3  (size = 170)
 - MyObjectKey4  (size = 170)
 - MyObjectKey5  (size = 170)

Listing objects with prefix "foo/bar"
 - foo/bar/test1.txt  (size = 170)
 - foo/bar/test2.txt  (size = 170)
 - foo/bar/var/mnt/any.txt  (size = 170)

Listing all objects
 - MyObjectKey  (size = 170)
 - MyObjectKey2  (size = 170)
 - MyObjectKey3  (size = 170)
 - MyObjectKey4  (size = 170)
 - MyObjectKey5  (size = 170)
 - foo/bar/test1.txt  (size = 170)
 - foo/bar/test2.txt  (size = 170)
 - foo/bar/var/mnt/any.txt  (size = 170)

Deleting the objects

Deleting bucket hqiu-s3-sample
```