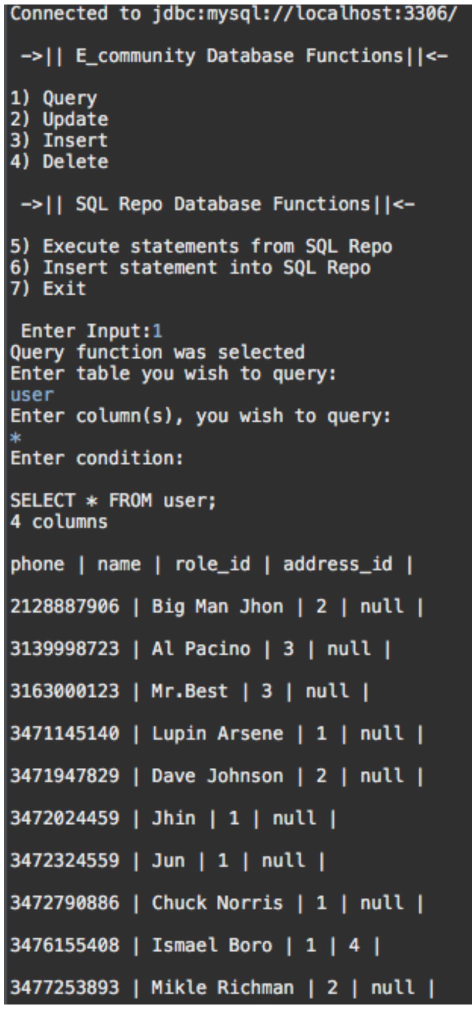
Michael Meneses
CS331 - Database Systems
Professor Bon Sy

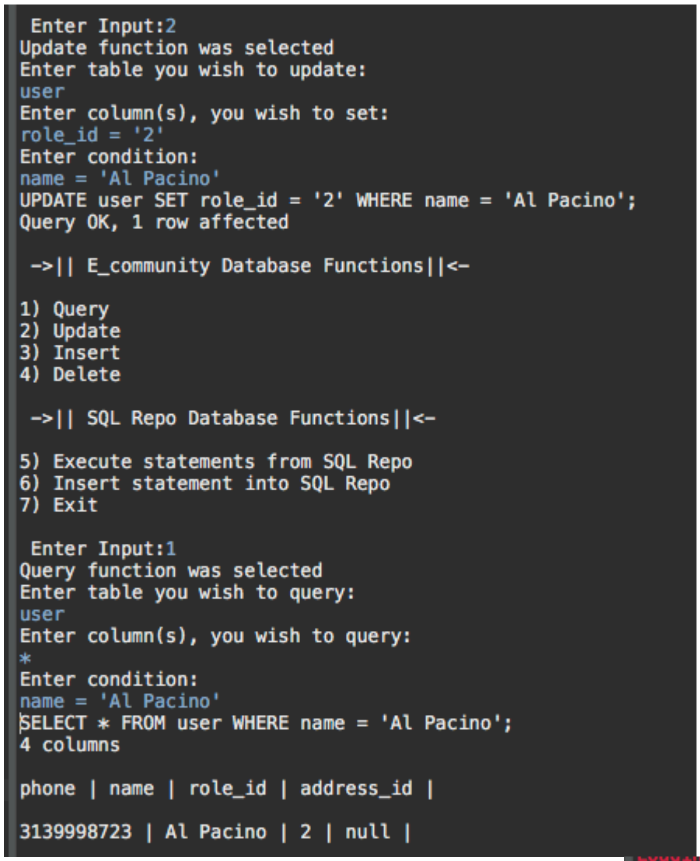
<u>Testing Results on Project</u>

Query

After testing the application, a limitation I found here was type errors. In order to display the SQL table correctly, I had to abstract the types when getting the results of the query in the ResultSet object that is returned. After some research, I also learned about and used the ResultSetMetaData object which allowed me to implement the visualization of all the columns from the query results. Instead of using getString or getInt from the results, I used getObject in order to not have to worry about any table's specific type. Then using the metadata object, we can just loop over the results with a while loop and then add a for loop to dynamically get all the columns in a SQL table. Furthermore, the metadata class allows you to get the Column Name's so I was able to format this in order to get a SQL-esque view of the data in the terminal using Java. Another thing I noticed as well was with the conditions. Each word must be spaced out accordingly in order to get all the conditions to work, especially when you are working with AND and OR operators.



```
Connected to jdbc:mysql://localhost:3306/

 ->|| E_community Database Functions||<-

1) Query
2) Update
3) Insert
4) Delete

 ->|| SQL Repo Database Functions||<-

5) Execute statements from SQL Repo
6) Insert statement into SQL Repo
7) Exit

 Enter Input:1
Query function was selected
Enter table you wish to query:
user
Enter column(s), you wish to query:
*
Enter condition:

SELECT * FROM user;
4 columns

phone | name | role_id | address_id |

2128887906 | Big Man Jhon | 2 | null |

3139998723 | Al Pacino | 3 | null |

3163000123 | Mr.Best | 3 | null |

3471145140 | Lupin Arsene | 1 | null |

3471947829 | Dave Johnson | 2 | null |

3472024459 | Jhin | 1 | null |

3472324559 | Jun | 1 | null |

3472790886 | Chuck Norris | 1 | null |

3476155408 | Ismael Boro | 1 | 4 |

3477253893 | Mikle Richman | 2 | null |
```



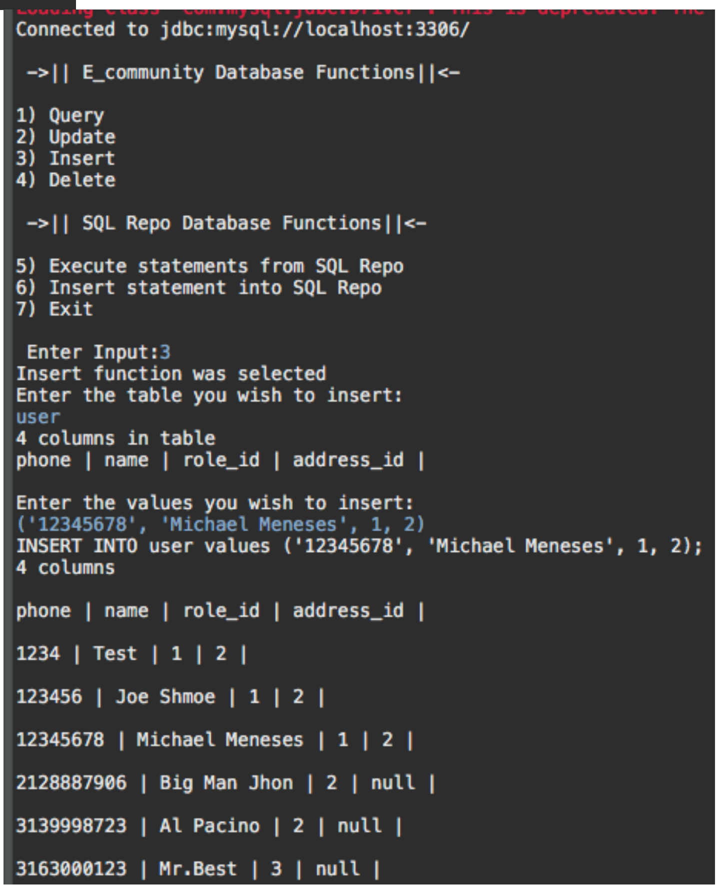
```
 Enter Input:2
Update function was selected
Enter table you wish to update:
user
Enter column(s), you wish to set:
role_id = '2'
Enter condition:
name = 'Al Pacino'
UPDATE user SET role_id = '2' WHERE name = 'Al Pacino';
Query OK, 1 row affected

 ->|| E_community Database Functions||<-

1) Query
2) Update
3) Insert
4) Delete

 ->|| SQL Repo Database Functions||<-

5) Execute statements from SQL Repo
6) Insert statement into SQL Repo
7) Exit

 Enter Input:1
Query function was selected
Enter table you wish to query:
user
Enter column(s), you wish to query:
*
Enter condition:
name = 'Al Pacino'
SELECT * FROM user WHERE name = 'Al Pacino';
4 columns

phone | name | role_id | address_id |

3139998723 | Al Pacino | 2 | null |
```

Update

I tested the Update function with different input and it worked for the most part. There were some limitations when it came to update because of the references between other tables.

Insert

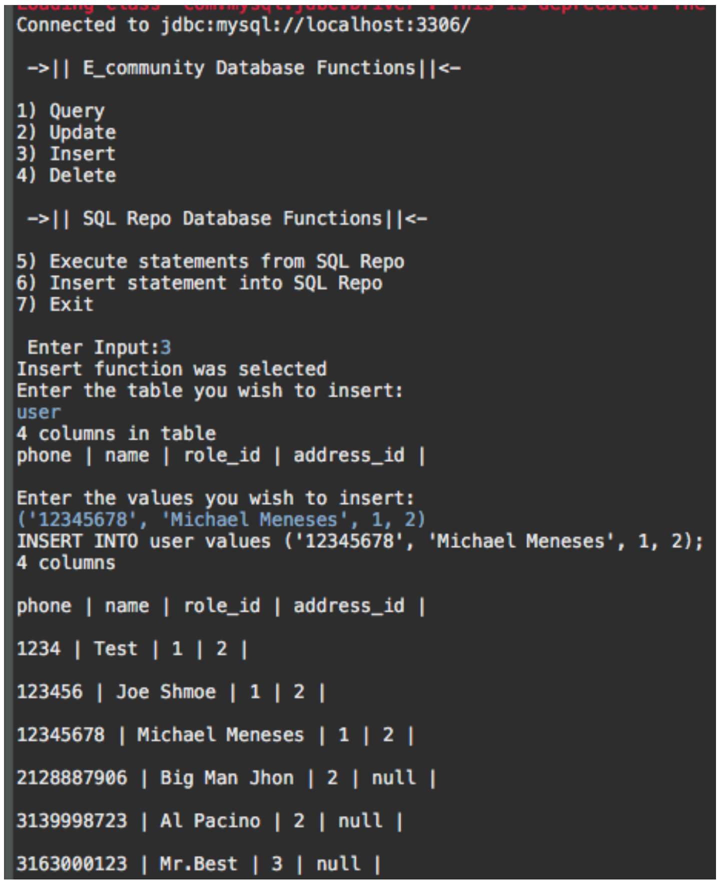
The insert function also worked well. The issue I faced was with the formatting. In order to avoid runtime errors with the format, I wrote the application so that it will show all column names beforehand and you can enter the values in correctly. Furthermore, I also made it so the input is entered directly into the parenthesis.



```
Connected to jdbc:mysql://localhost:3306/

 ->|| E_community Database Functions||<-

1) Query
2) Update
3) Insert
4) Delete

 ->|| SQL Repo Database Functions||<-

5) Execute statements from SQL Repo
6) Insert statement into SQL Repo
7) Exit

 Enter Input:3
Insert function was selected
Enter the table you wish to insert:
user
4 columns in table
phone | name | role_id | address_id |

Enter the values you wish to insert:
('12345678', 'Michael Meneses', 1, 2)
INSERT INTO user values ('12345678', 'Michael Meneses', 1, 2);
4 columns

phone | name | role_id | address_id |

1234 | Test | 1 | 2 |

123456 | Joe Shmoe | 1 | 2 |

12345678 | Michael Meneses | 1 | 2 |

2128887906 | Big Man Jhon | 2 | null |

3139998723 | Al Pacino | 2 | null |

3163000123 | Mr.Best | 3 | null |
```

Delete

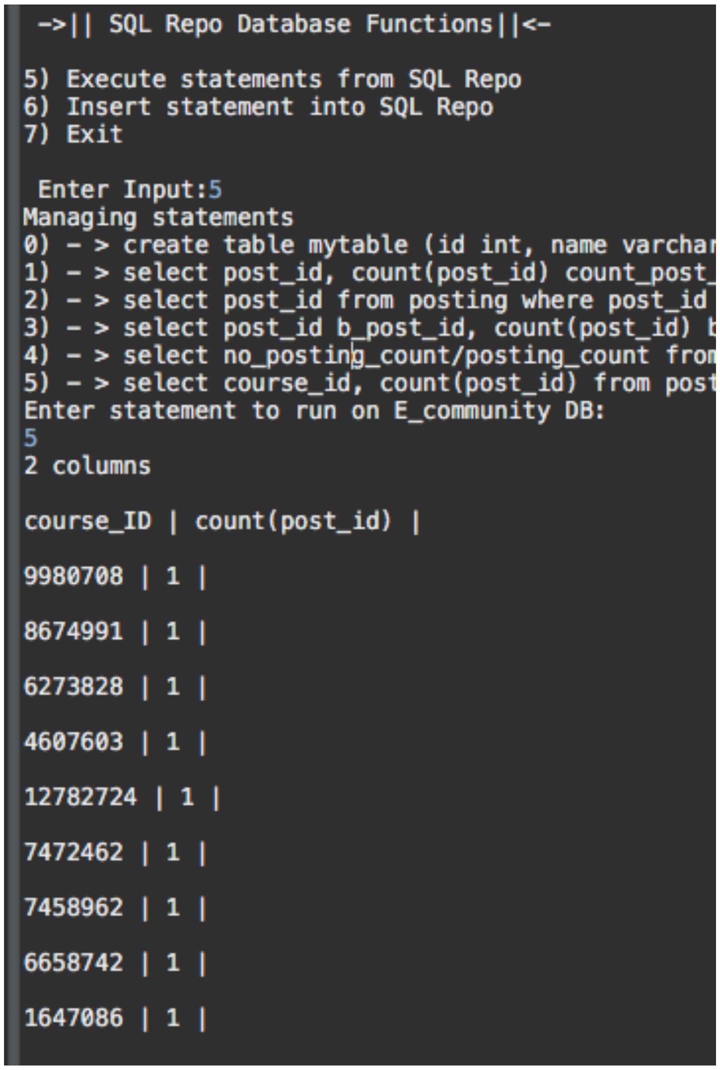
The insert function also worked well. The issue I faced was with the formatting. In order to avoid runtime errors with the format, I wrote the application so that it will show all column names beforehand and you can enter the values in correctly. Furthermore, I also made it so the input is entered directly into the parenthesis. This makes it so you must enter all the values at creation making for more data integrity.



```
Connected to jdbc:mysql://localhost:3306/

 ->|| E_community Database Functions||<-

1) Query
2) Update
3) Insert
4) Delete

 ->|| SQL Repo Database Functions||<-

5) Execute statements from SQL Repo
6) Insert statement into SQL Repo
7) Exit

 Enter Input:3
Insert function was selected
Enter the table you wish to insert:
user
4 columns in table
phone | name | role_id | address_id |

Enter the values you wish to insert:
('12345678', 'Michael Meneses', 1, 2)
INSERT INTO user values ('12345678', 'Michael Meneses', 1, 2);
4 columns

phone | name | role_id | address_id |

1234 | Test | 1 | 2 |

123456 | Joe Shmoe | 1 | 2 |

12345678 | Michael Meneses | 1 | 2 |

2128887906 | Big Man Jhon | 2 | null |

3139998723 | Al Pacino | 2 | null |

3163000123 | Mr.Best | 3 | null |
```

SQL Repo

Most commands were able to run smoothly. There were some that returned errors different SQL mode configurations.



```
 ->|| SQL Repo Database Functions||<-

5) Execute statements from SQL Repo
6) Insert statement into SQL Repo
7) Exit

 Enter Input:5
Managing statements
0) - > create table mytable (id int, name varchar
1) - > select post_id, count(post_id) count_post_
2) - > select post_id from posting where post_id
3) - > select post_id b_post_id, count(post_id) b
4) - > select no_posting_count/posting_count from
5) - > select course_id, count(post_id) from post
Enter statement to run on E_community DB:
5
2 columns

course_ID | count(post_id) |

9980708 | 1 |

8674991 | 1 |

6273828 | 1 |

4607603 | 1 |

12782724 | 1 |

7472462 | 1 |

7458962 | 1 |

6658742 | 1 |

1647086 | 1 |
```