1. Remote Attacker

1.1 How many accounts does Nigel Farage have?
He has 2 accounts.

1.2 Explain how you found the answer to the previous question.
As the article said: this application is quite rough, is implemented by an inexperienced java developer, he has no formal training in security system development.
So I guess the account password in the user query statement is simply spliced with the SQL statement and I tried SQL injection, that is, by using string *" xxx' OR 1=1 -- " as the user name. After login I found that* Nigel Farage has two accounts.

2. Attacker with Local Access

2.1 How has the developer tried to protect the database?
He encrypted the database by using AES. And decrypt the database when the service is running.

2.2 What is wrong with this approach to protecting the database?
In terms of performance, this method is only suitable for small databases. When the amount of data is large, the decryption time will be too long.
In the aspect of security, the key decryption method is used in the code. Once the key is leaked, the data in the whole database will be leaked.
The database will be decrypted during starting up, and all subsequent queries will be directed against the database so that the attacker can access through the interface.
In addition, temporary files are used to store the decrypted data, which may lead to data leakage.

2.3 Who has deposited the most money? Why has Mary Jones deposited money?
Jacob Rees-Mogg deposited the most money. Because Mary Jones wants to store the retirement fund.

2.4 Explain brieflfly how you found the answers to the previous question.
By modifying the decryptData() function in file Decryptor.java, the decrypted database file is saved locally and thus I can open it using SQLite expert.

2.5 Outline how the handling of passwords could be improved.
The password encrypted by hash is stored in the database. When logging in, the password is converted into ciphertext by using MD5 algorithm and compared with the ciphertext stored in the database.

Besides, using ? as a placeholder can effectively avoid SQL injection.

3. Fixing the Issues

3.1.

In order to avoid the problem of SQL injection I have mentioned in 3.1, I used ? as a place holder to receive input instead of simply splicing the password into SQL.

```
private static final String AUTH_QUERY = "select * from account_owner where username=? and password=?";
private static final String ACCOUNT_QUERY = "select * from account where owner_id='%d'";
```

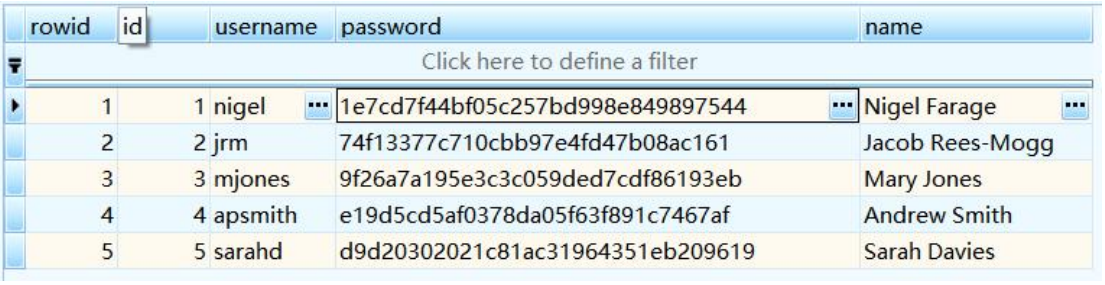
```
try (PreparedStatement stmt = database.prepareStatement(AUTH_QUERY)) {
  stmt.setString( parameterIndex: 1, username);
  stmt.setString( parameterIndex: 2, password);
  ResultSet results = stmt.executeQuery();
```

The PreparedStatement object will first prepare the SQL statement, and then use the setString() method to set the parameters in the statement, which will avoid the problem of SQL injection.

3.2

```
  String MD5password = MD5(password);
//   System.out.println(MD5password);
  try (PreparedStatement stmt = database.prepareStatement(AUTH_QUERY)) {
    stmt.setString( parameterIndex: 1, username);
    stmt.setString( parameterIndex: 2, MD5password);
    ResultSet results = stmt.executeQuery();
```

```
public static String MD5(String str) {
  byte[] digest = null;
  try {
    MessageDigest md5 = MessageDigest.getInstance("md5");
    digest  = md5.digest(str.getBytes( charsetName: "utf-8"));
  } catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
  } catch (UnsupportedEncodingException e) {
    e.printStackTrace();
  }
  String md5Str = new BigInteger( signum: 1, digest).toString( radix: 16);
  return md5Str;
}
```



rowid	id	username	password	name
Click here to define a filter				
1	1	nigel	1e7cd7f44bf05c257bd998e849897544	Nigel Farage
2	2	jrm	74f13377c710cbb97e4fd47b08ac161	Jacob Rees-Mogg
3	3	mjones	9f26a7a195e3c3c059ded7cdf86193eb	Mary Jones
4	4	apsmith	e19d5cd5af0378da05f63f891c7467af	Andrew Smith
5	5	sarahd	d9d20302021c81ac31964351eb209619	Sarah Davies

The password is encrypted by MD5 algorithm when it is stored. When logging in, password will be converted by MD5 algorithm first and then compared with the ciphertext stored in the database. This method can avoid the password in the database being directly obtained by the attacker. And this method can effectively prevent password leakage.