6.033
COMPUTER SYSTEMS ENGINEERING
HANDS ON 8: DATABASES

JOHN WANG

1. Using SQL

(1) Completed
(2) Completed
(3) Displaying the fullname of a person with username bitdidd1:

```
SELECT fullname FROM accounts WHERE username = 'bitdidd1';
```

Average account balance of people with at least 70 in account:

```
SELECT AVG(balance) FROM accounts WHERE balance > 70;
```

(4)

```
UPDATE accounts SET balance = balance - 10 WHERE fullname = 'jones';
UPDATE accounts SET balance = balance + 10 WHERE fullname = 'mike';
```

2. Transactions

(5) The list of accounts from the two terminals are not the same. This is because the transaction of the second terminal has not yet been committed, so the first terminal does not see the insertion.
(6) The list of accounts does not include this new record. This is because the second terminal is still in a transaction which has not completed yet, so it has not been updated with the information from the committed transaction from the first terminal.
(7) Committing the transaction finally shows the newly added record in the list of accounts. This is because both transactions have finally committed so each transaction can see the new record.
(8) The second update stops and blocks forever. This is because the first update has not been committed so the transaction prevents you from updating the balance until the first transaction commits or aborts.
(9) Once I abort, the second transaction updates successfully inside of its transaction.
(10) The resulting balance is 73 and mike has lost 10 (so only the second update was successful).
(11) Nothing has changed yet because we have not committed the transaction.
(12) The changes become visible only after the commit command has been called. This is because the transaction is supposed to be an atomic object, so all records are committed at the same time when the user calls commit.

3. Transaction Isolation Levels

(13) Postgres probably implements READ COMMITTED as the default because it is simple. Usually, applications will not care about seeing things after commit or before commit (since those transactions will be fast). It takes incredibly long queries or other complex actions to make a noticable difference. Because of this, it seems excessive for the average user to have to understand SERIALIZABLE. In the normal case of usage, READ COMMITTED is simple and will suffice.