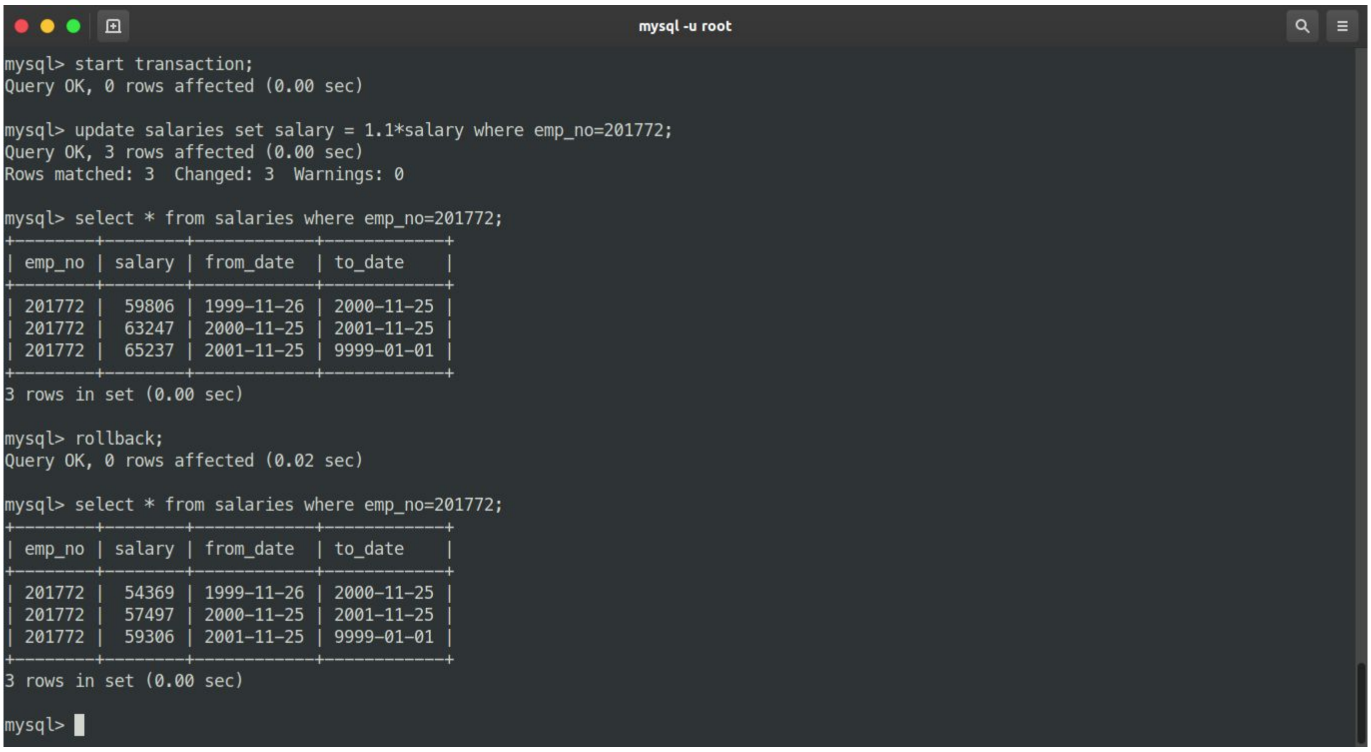
CO527 Advanced Database Systems

Lab Number: 04
Topic: Transaction Processing
Name: M.M.M Irfan
Reg No: E/15/138

Execute the following transaction for the salary table updates.

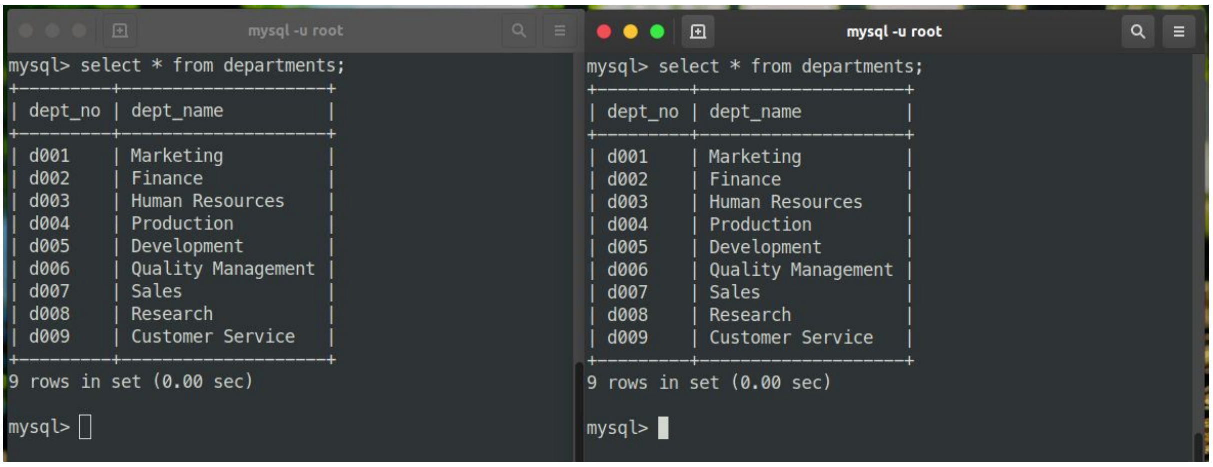
The output from the start transaction statement



```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update salaries set salary = 1.1*salary where emp_no=201772;
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> select * from salaries where emp_no=201772;
+--------+--------+------------+------------+
| emp_no | salary | from_date  | to_date    |
+--------+--------+------------+------------+
| 201772 |  59806 | 1999-11-26 | 2000-11-25 |
| 201772 |  63247 | 2000-11-25 | 2001-11-25 |
| 201772 |  65237 | 2001-11-25 | 9999-01-01 |
+--------+--------+------------+------------+
3 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from salaries where emp_no=201772;
+--------+--------+------------+------------+
| emp_no | salary | from_date  | to_date    |
+--------+--------+------------+------------+
| 201772 |  54369 | 1999-11-26 | 2000-11-25 |
| 201772 |  57497 | 2000-11-25 | 2001-11-25 |
| 201772 |  59306 | 2001-11-25 | 9999-01-01 |
+--------+--------+------------+------------+
3 rows in set (0.00 sec)

mysql>
```

1. I of ACID

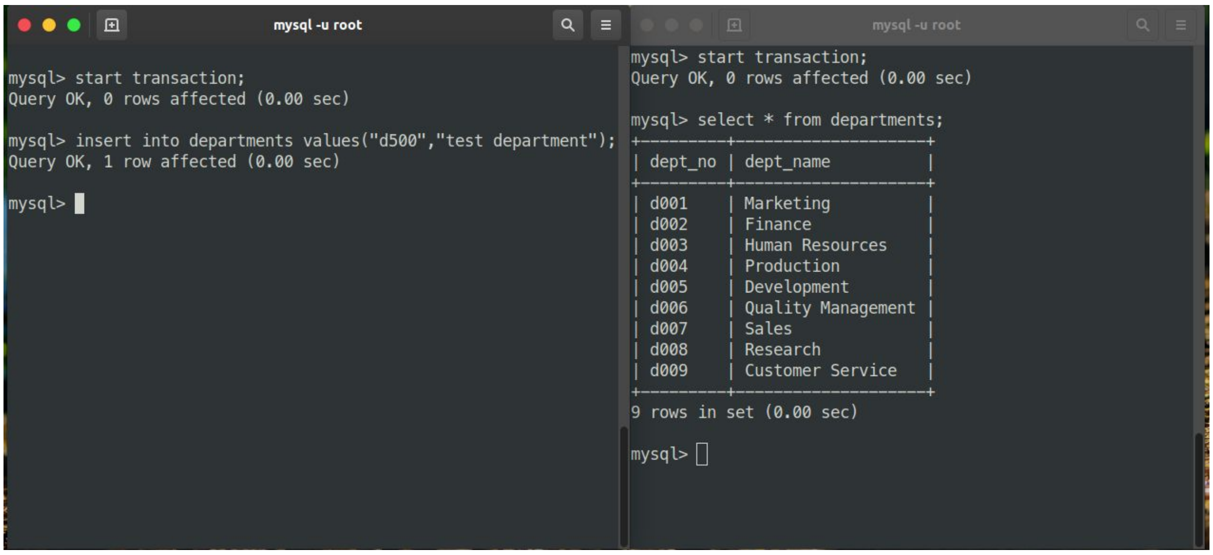
I. Issue a select query to view the current status of the department table in both sessions.



```
mysql> select * from departments;
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d001    | Marketing          |
| d002    | Finance            |
| d003    | Human Resources    |
| d004    | Production         |
| d005    | Development        |
| d006    | Quality Management |
| d007    | Sales              |
| d008    | Research           |
| d009    | Customer Service   |
+---------+--------------------+
9 rows in set (0.00 sec)

mysql>
```

```
mysql> select * from departments;
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d001    | Marketing          |
| d002    | Finance            |
| d003    | Human Resources    |
| d004    | Production         |
| d005    | Development        |
| d006    | Quality Management |
| d007    | Sales              |
| d008    | Research           |
| d009    | Customer Service   |
+---------+--------------------+
9 rows in set (0.00 sec)

mysql>
```

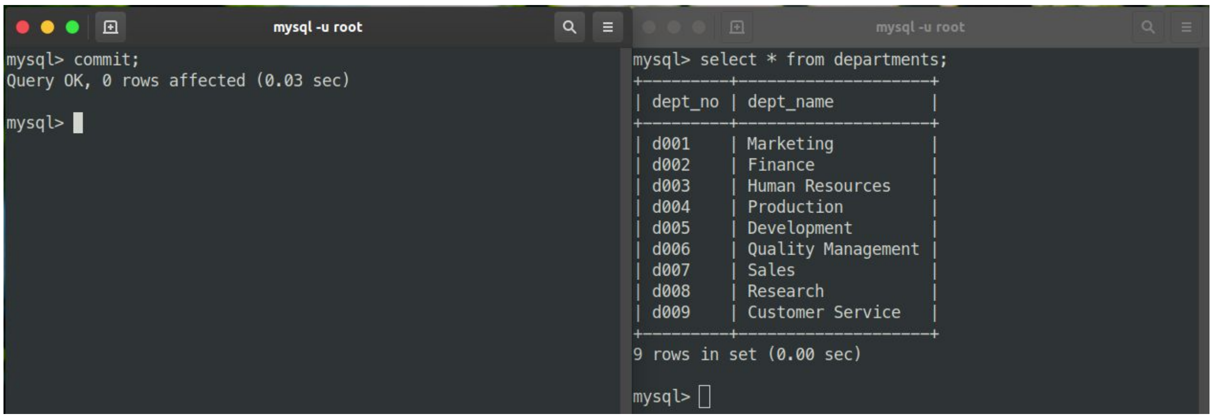
II. Now, start transaction running start transaction in both sessions.
III. Insert a new row into the departments table from the 1st session and check if the changes are visible in the second session.



```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into departments values("d500","test department");
Query OK, 1 row affected (0.00 sec)

mysql>
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from departments;
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d001    | Marketing          |
| d002    | Finance            |
| d003    | Human Resources    |
| d004    | Production         |
| d005    | Development        |
| d006    | Quality Management |
| d007    | Sales              |
| d008    | Research           |
| d009    | Customer Service   |
+---------+--------------------+
9 rows in set (0.00 sec)

mysql>
```

IV. Commit changes in the 1st command window and check if you can see the updates done at 1st window in 2nd command window.

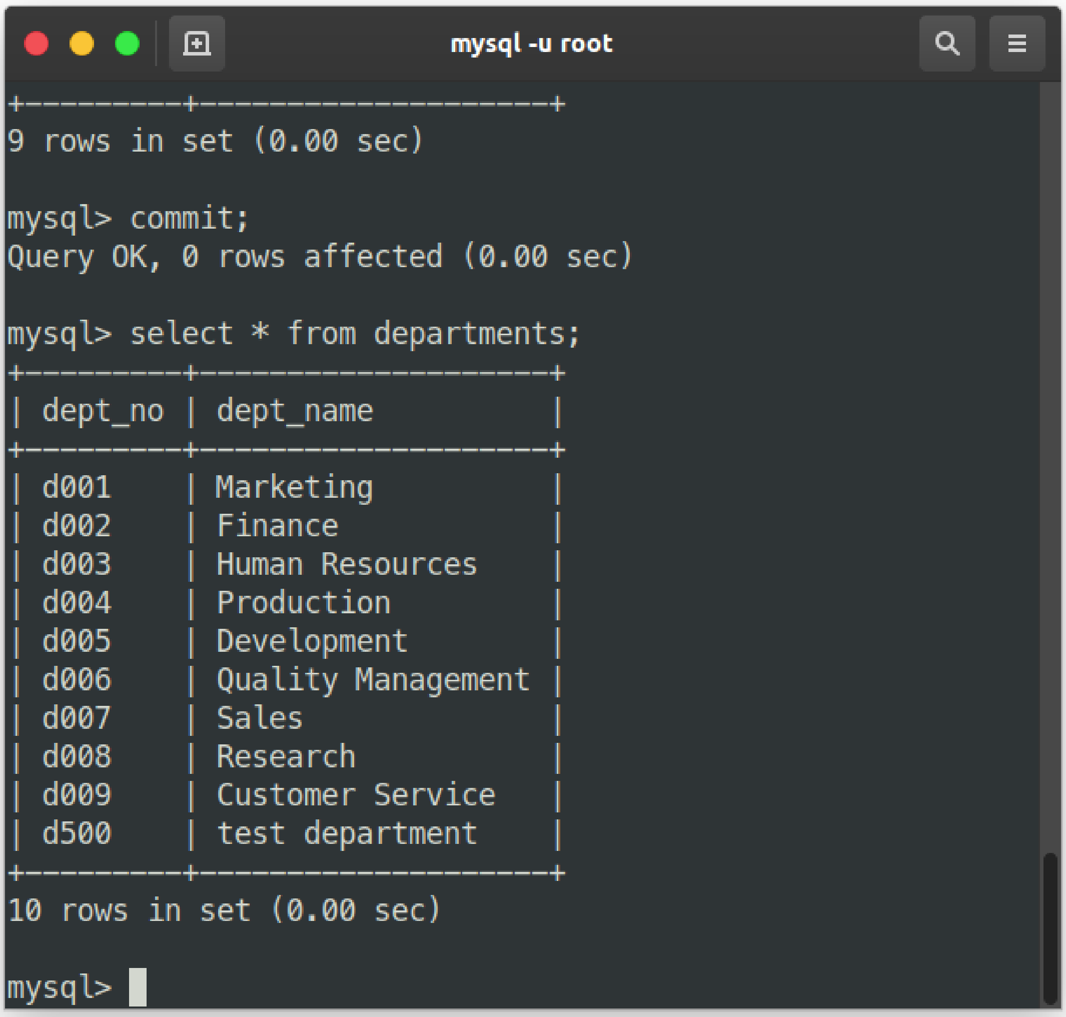


```
mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

```
mysql> select * from departments;
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d001    | Marketing          |
| d002    | Finance            |
| d003    | Human Resources    |
| d004    | Production         |
| d005    | Development        |
| d006    | Quality Management |
| d007    | Sales              |
| d008    | Research           |
| d009    | Customer Service   |
+---------+--------------------+
9 rows in set (0.00 sec)

mysql>
```

V. Explain your observations before and after running the commit in the 1st window.

Before the transaction is committed it will not reflect in any other new transaction until it gets committed.in other words, they are temporary if those changes are not committed. So when we look for entries before committing we will not see the newly inserted department. But after committing the 1st transaction it will be flush to the DB so the other new transactions can see the updated results. But in this case, the 2nd session has already a transaction started so it should go to the terminated state by either committing or aborting so that it can take the new changes.so let's see what happens when we commit the 2nd transaction

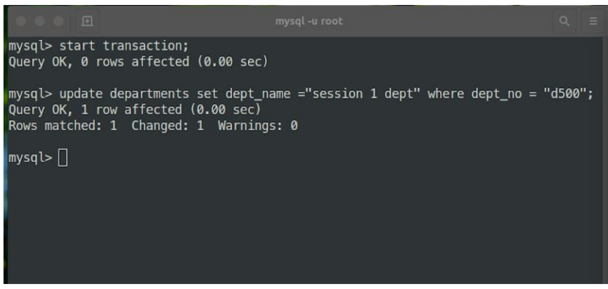


```
+----------+--------------------+
9 rows in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from departments;
+----------+--------------------+
| dept_no  | dept_name          |
+----------+--------------------+
| d001     | Marketing          |
| d002     | Finance            |
| d003     | Human Resources    |
| d004     | Production         |
| d005     | Development        |
| d006     | Quality Management |
| d007     | Sales              |
| d008     | Research           |
| d009     | Customer Service   |
| d500     | test department    |
+----------+--------------------+
10 rows in set (0.00 sec)

mysql> 
```

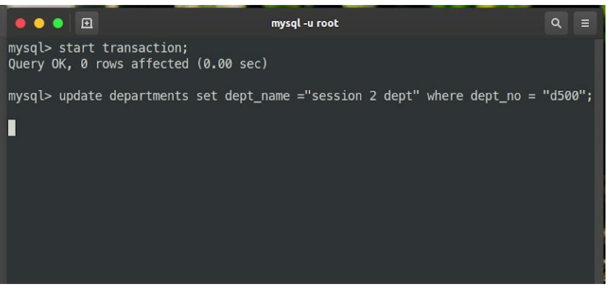
So if the transaction is not committed no other external changes will affect the view of the current transaction

2. Concurrent Updates

I. Try to do a concurrent update to the same row in departments table during two transactions



```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update departments set dept_name ="session 1 dept" where dept_no = "d500";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> 
```



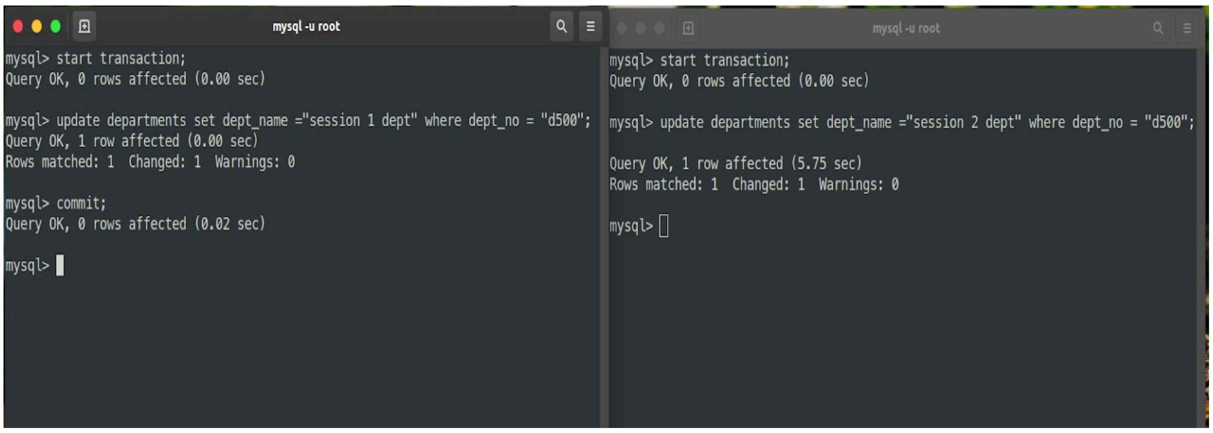
```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update departments set dept_name ="session 2 dept" where dept_no = "d500";
```

Observation: Once the 1st transaction did an update in the row, the other transaction is hanging without executing the update statement since its locked.

II. Explain what happens before ending any of the transactions.

A transaction ends when it is committed or rolled back, either explicitly with a COMMIT or ROLLBACK statement or implicitly when a DDL statement is issued. When the commit statement is used to end the transaction, the changes made by the SQL statement(s) of a transaction become permanent and visible to other users. Queries that are issued after the transaction commits will see the committed changes.
Similarly, when the transaction ends with a rollback statement all the changes made by that transaction will be discarded.

III. What happens when you commit your changes in the 1st session.

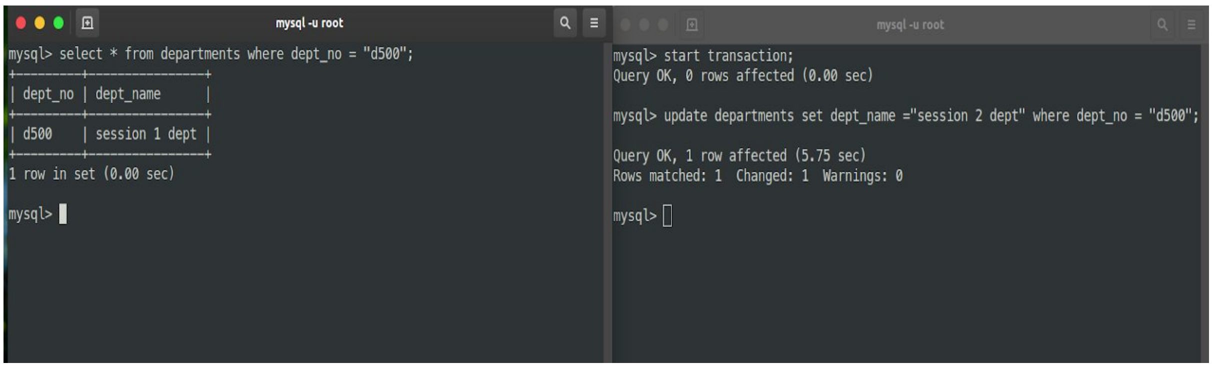


```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update departments set dept_name ="session 1 dept" where dept_no = "d500";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.02 sec)

mysql>
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update departments set dept_name ="session 2 dept" where dept_no = "d500";

Query OK, 1 row affected (5.75 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

When the 1st transaction commits the changes the lock is released and the 2nd transaction executes the update statement.
Now if we rollback the 2nd transaction the changes made by the 1st transaction will be there permanently. But if we commit the 2nd transaction, then the row will show the values updated by the 2nd transaction.

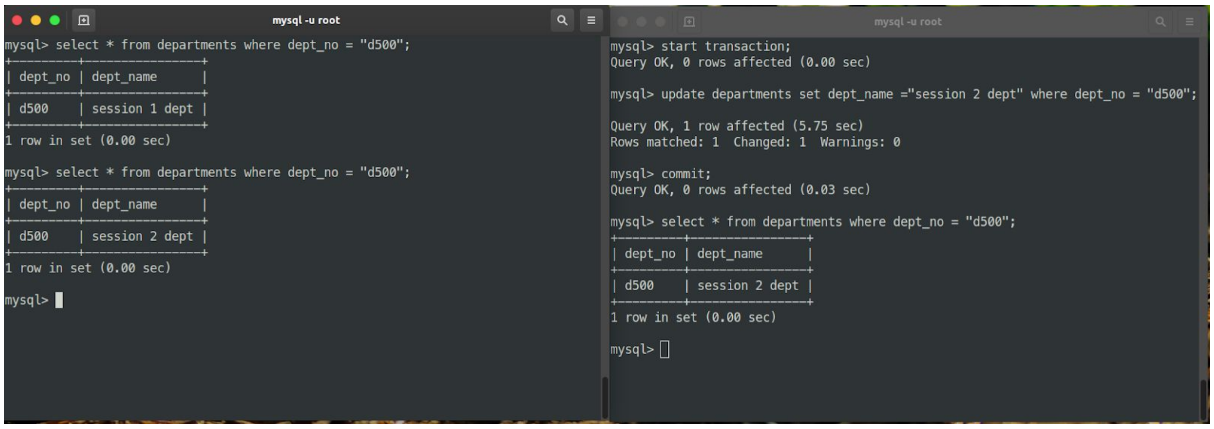
So let’s see the row after the 1st transaction is committed.



```
mysql> select * from departments where dept_no = "d500";
+---------+----------------+
| dept_no | dept_name      |
+---------+----------------+
| d500    | session 1 dept |
+---------+----------------+
1 row in set (0.00 sec)

mysql>
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update departments set dept_name ="session 2 dept" where dept_no = "d500";

Query OK, 1 row affected (5.75 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

Now when we issue a commit statement in the 2nd transaction, both will have the values updated by the 2nd transaction



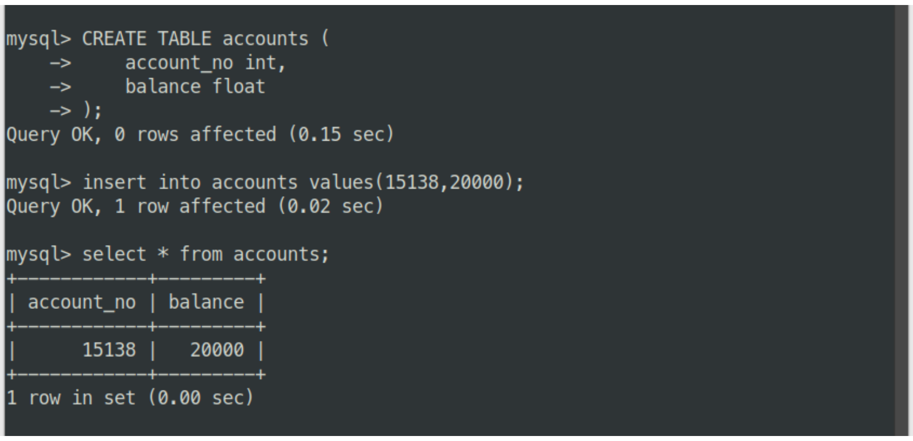
```
mysql> select * from departments where dept_no = "d500";
+---------+----------------+
| dept_no | dept_name      |
+---------+----------------+
| d500    | session 1 dept |
+---------+----------------+
1 row in set (0.00 sec)

mysql> select * from departments where dept_no = "d500";
+---------+----------------+
| dept_no | dept_name      |
+---------+----------------+
| d500    | session 2 dept |
+---------+----------------+
1 row in set (0.00 sec)

mysql> 
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update departments set dept_name ="session 2 dept" where dept_no = "d500";

Query OK, 1 row affected (5.75 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql> select * from departments where dept_no = "d500";
+---------+----------------+
| dept_no | dept_name      |
+---------+----------------+
| d500    | session 2 dept |
+---------+----------------+
1 row in set (0.00 sec)

mysql> 
```

Use your imagination and words to write a scenario where using transactions is essential and then create the required tables and test how the transaction will affect your tables,

Scenario: Consider a banking database and a situation of withdrawing the money from an account. So when two people withdraw the money from two ATMs at the same time, there can be an inconsistency in the account balance.in order to avoid that we can use transactions.

Let’s create a simple table where it has an account_no and a balance.

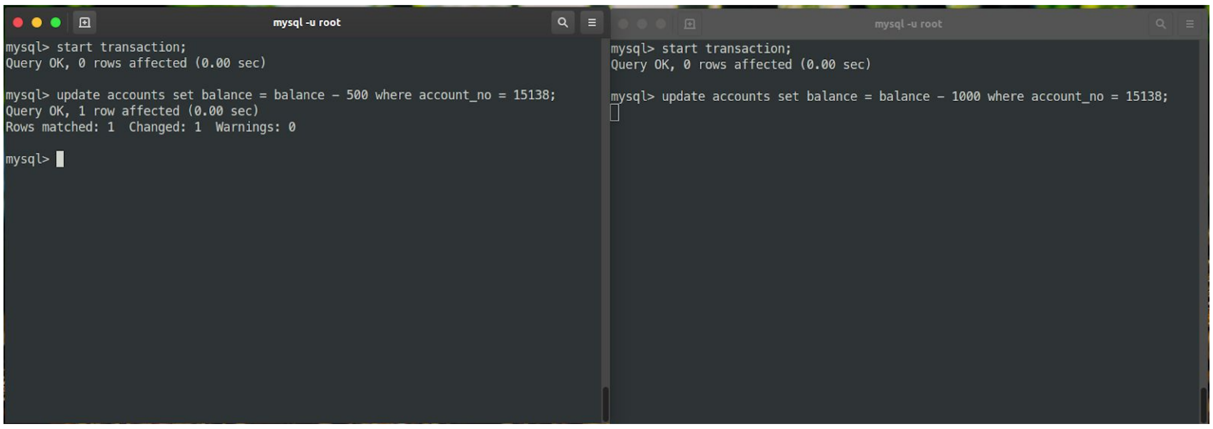


```
mysql> CREATE TABLE accounts (
    ->     account_no int,
    ->     balance float
    -> );
Query OK, 0 rows affected (0.15 sec)

mysql> insert into accounts values(15138,20000);
Query OK, 1 row affected (0.02 sec)

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   20000 |
+------------+---------+
1 row in set (0.00 sec)
```

Now let’s try to update the balance (withdraw) in two different transactions

1. during the transaction execution.

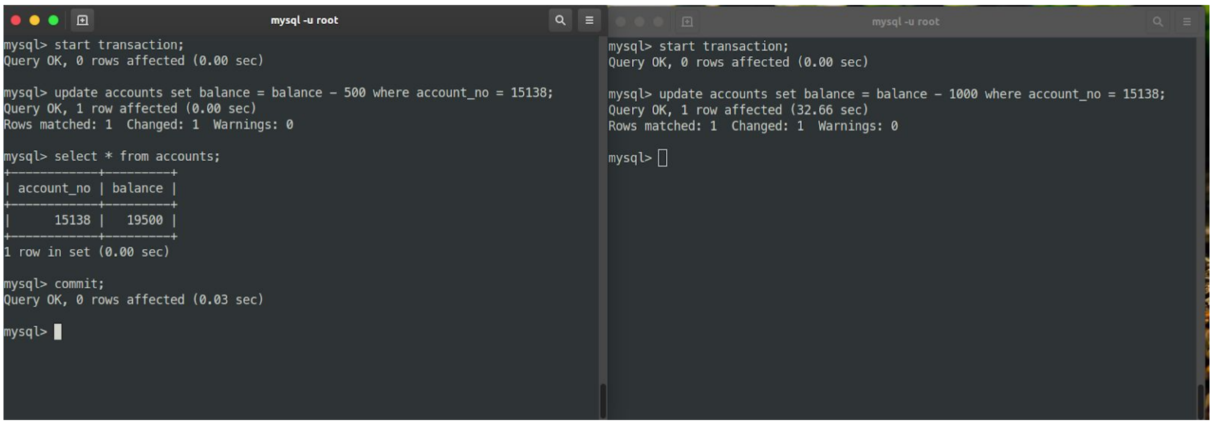
Assume transaction 1 withdraws 500 meanwhile transaction 2 withdraws 1000.
So when we do the update in the database, if transaction 1 did the update first then transaction 2 is waiting until it commits or aborts



```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update accounts set balance = balance - 500 where account_no = 15138;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update accounts set balance = balance - 1000 where account_no = 15138;
```

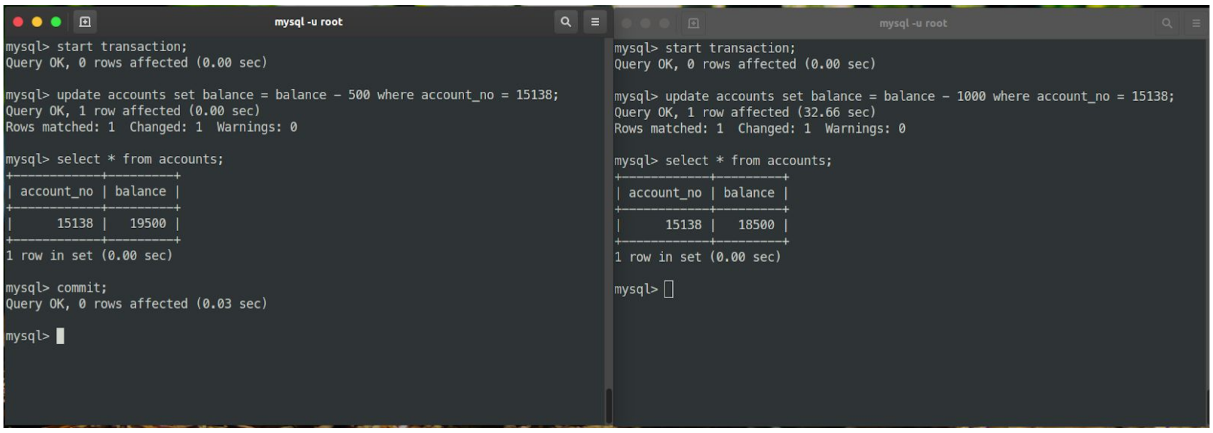
Now if we check the balance in transaction 1, it will be 19500. When we commit the changes in transaction 1, transaction 2 will execute the update statement



```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update accounts set balance = balance - 500 where account_no = 15138;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   19500 |
+------------+---------+
1 row in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update accounts set balance = balance - 1000 where account_no = 15138;
Query OK, 1 row affected (32.66 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

So in the new transaction, the initial balance would be 19500 so when we withdraw 1000 then it will become 18500 which is what the transaction 2 says as the balance

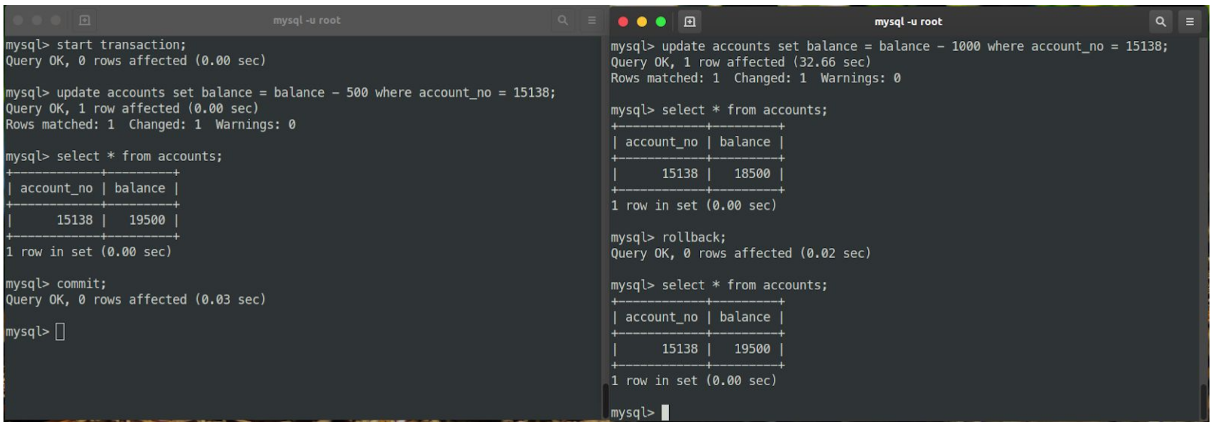


```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update accounts set balance = balance - 500 where account_no = 15138;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   19500 |
+------------+---------+
1 row in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update accounts set balance = balance - 1000 where account_no = 15138;
Query OK, 1 row affected (32.66 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   18500 |
+------------+---------+
1 row in set (0.00 sec)

mysql>
```

2. after rollback statement.

Once the transaction 2 is rolled back, the balance would change to 19500 which is the balance after the transaction 1 is completed. Changes done by the rolled-back transaction 2 will be discarded



```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update accounts set balance = balance - 500 where account_no = 15138;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   19500 |
+------------+---------+
1 row in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

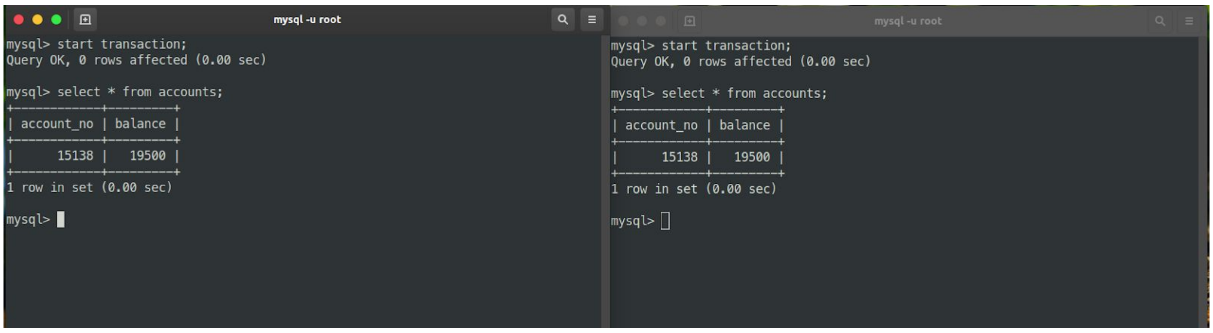
```
mysql> update accounts set balance = balance - 1000 where account_no = 15138;
Query OK, 1 row affected (32.66 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   18500 |
+------------+---------+
1 row in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   19500 |
+------------+---------+
1 row in set (0.00 sec)

mysql>
```

3. after the commit statement.

After commit statement is executed in transaction 1, the changes will appear in all other transaction which is trying to update that entry. And the update statement which was waiting to execute will be executed after the previous transaction commits it.

4. during 2 concurrent transactions, both of them update a record and both of them commit it.
Now the initial balance would be 19500 after all these operations.
now let’s try to withdraw the same amount as before, but this time both the transactions are going to commit it.

Check the balance

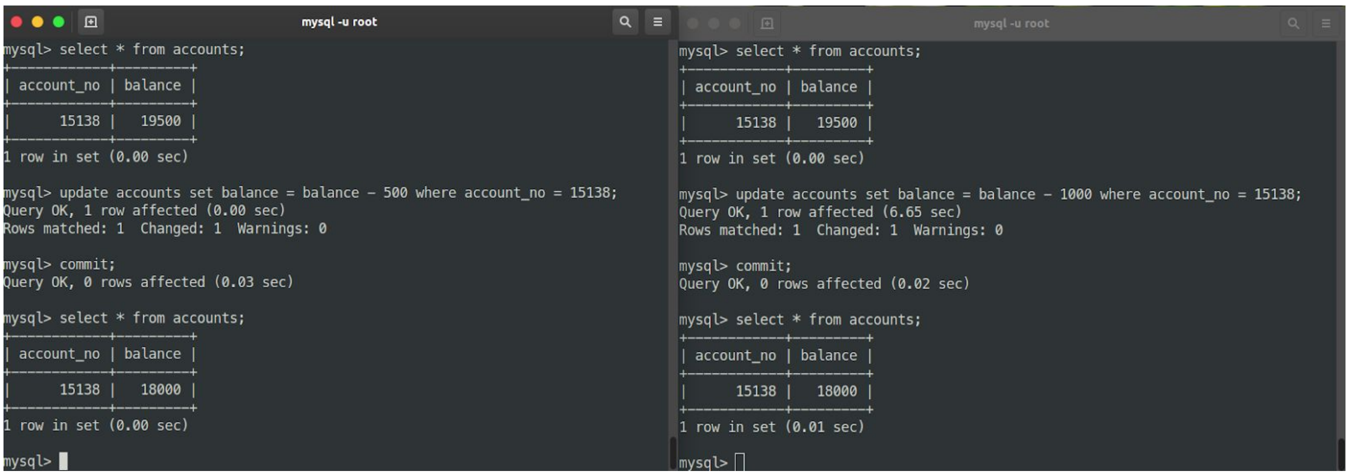


```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   19500 |
+------------+---------+
1 row in set (0.00 sec)

mysql>
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   19500 |
+------------+---------+
1 row in set (0.00 sec)

mysql>
```

Commit the changes in both transaction and check the balance
So now both the update will take place but one after the other, the 2nd update will take place only after 1st transaction commits.

Initial balance = 19500
So the final balance = initial balance - 500 - 1000 = 18000



```
mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   19500 |
+------------+---------+
1 row in set (0.00 sec)

mysql> update accounts set balance = balance - 500 where account_no = 15138;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   18000 |
+------------+---------+
1 row in set (0.00 sec)

mysql>
```

```
mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   19500 |
+------------+---------+
1 row in set (0.00 sec)

mysql> update accounts set balance = balance - 1000 where account_no = 15138;
Query OK, 1 row affected (6.65 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from accounts;
+------------+---------+
| account_no | balance |
+------------+---------+
|      15138 |   18000 |
+------------+---------+
1 row in set (0.01 sec)

mysql>
```

RESOURCES

Please find the images and the resources used in this report in the following link
https://github.com/irfanm96/CO326/tree/master/lab04/resources

REFERENCES

- https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch12.htm