**CO527- Advanced Database Systems**

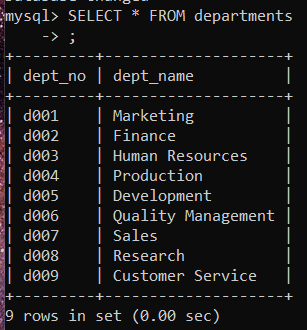
**Lab 04**

**E/17/407**

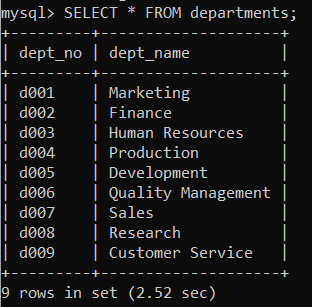
**Wijesooriya H.D**

**(Q1) I of ACID**

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



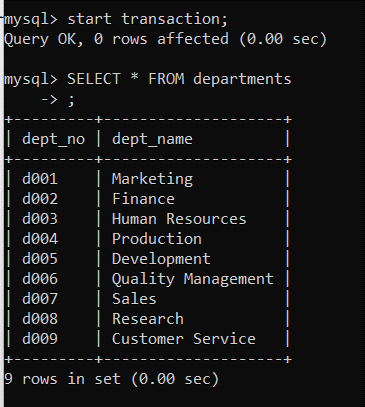
**Session 02**



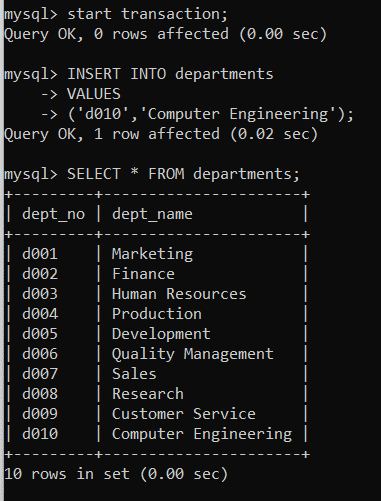
**Session 01**

**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.**



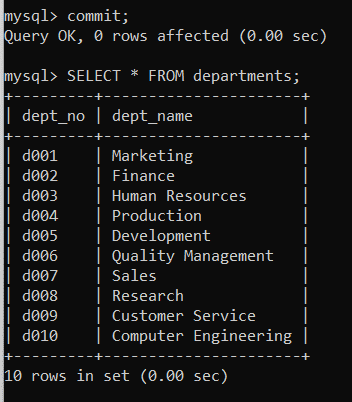
**Session 02**



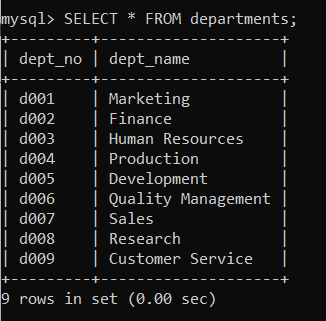
**Session 01**

Changes (‘d010’, ‘Computer Engineering’) are not visible in the session 2. (In session 1 the changes are visible because, there the auto commit mode is set to one.)

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



**Session 01**



**Session 02**

Updates are not visible in the 2nd command window as it is in the middle of the transaction. . Once we committed the ongoing transaction in session 02 and start a new transaction then we were able to see the updates.

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

Before the commit in 1st window:

In this case the updates were not visible in the 2nd window. This is because the updates done at the 1st window were not permanently saved in the database as the transaction was not committed.

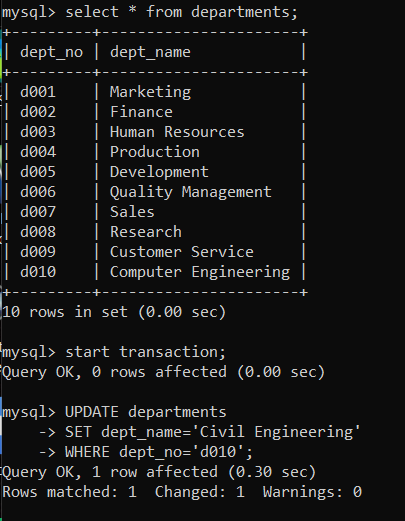
After the commit in 1st window:

In this case the updates are not visible in the 2nd command window as it is in the middle of the transaction. Once we committed the ongoing transaction in session 02 and start a new transaction then we were able to see the updates.

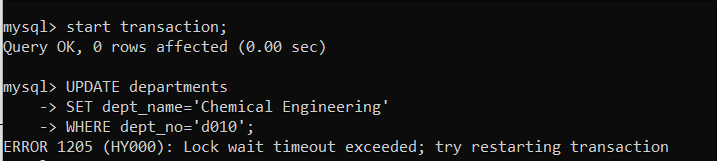
**(Q2)**

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

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



**Session 01**

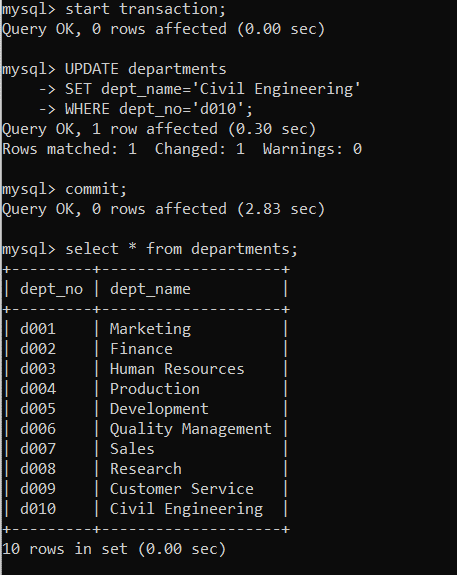


**Session 02**

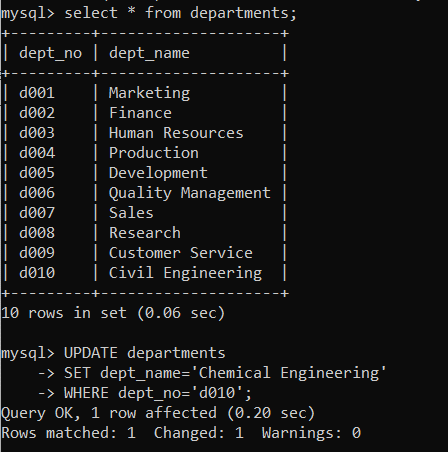
We were not allowed to updates in session 02, before committing the transaction in session 01.

(**session 01 update** : (‘d010’ , ‘Civil Engineering’) , **session 02 update** : (‘d010’ , ‘Chemical Engineering’ ) )

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



**Session 01**



**Session 02**

When we commit the changes in session 01 , then we are allowed to do the updates in session 02.(**session 01 update** : (‘d010’ , ‘Civil Engineering’) , **session 02 update** : (‘d010’ , ‘Chemical Engineering’ ) ) . Also in session 02 we were able to see the updates done in session 01.

**(Q3)**

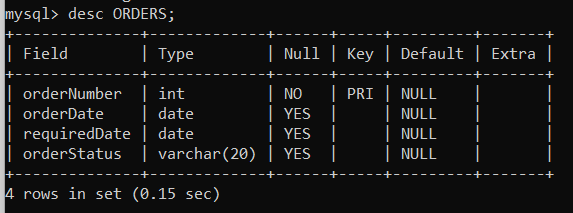
**Use your imagination and words to write a scenario where using transactions is essential and then create the required tables and test how the transaction will effect your tables,**

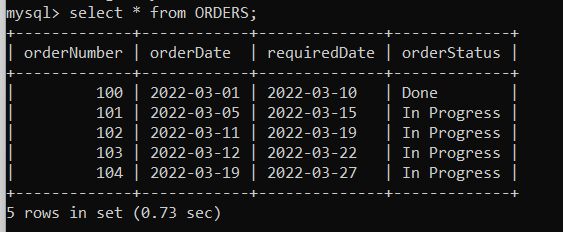
**Scenario :** Lets imagine a shop which sells some products orders online. In this scenario transaction processing is essential when we need to add new sales to the tables in our database and, when we need to do updates on tables. The steps of adding a new sale to the tables of our database are given below.

* First we need to get the latest sales order number from the **ORDERS** table and use the next sales order number as the new sales order number.
* Then we need to  [insert](https://www.mysqltutorial.org/mysql-insert-statement.aspx) a new sales order into the **ORDERS** table.
* Then we get the newly inserted sales order number.
* Next we insert the new sales order items into the**orderDetails** table with the sales order number.

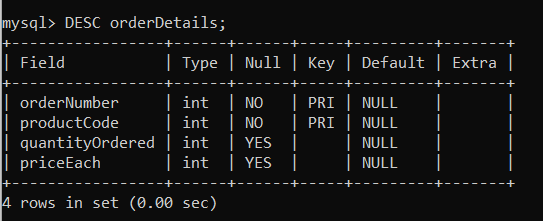
Let’s consider what would happen to the sales order data if one or more steps above fail due to some reasons such as [table locking](https://www.mysqltutorial.org/mysql-table-locking/)? As an example, if the step of adding order’s items into**orderDetails** table fails, we will have an empty sales order. That is why we need transaction processing in this scenario. MySQL transaction allows us to execute a set of MySQL operations to ensure that the database never contains the result of partial operations.

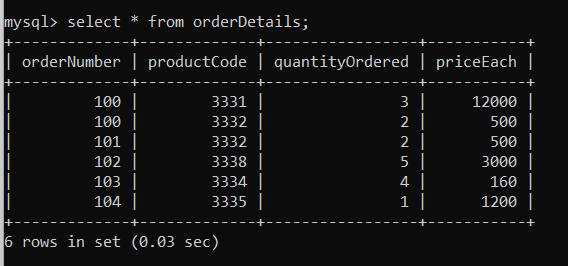
**Tables in the designed database:**

****

****

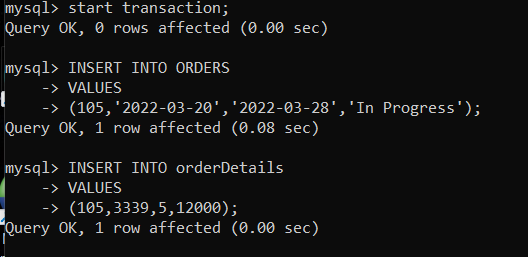
Content of the **ORDERS** table before the transaction begins

****

****

Content of the **ordeDetails** table before the transaction begins

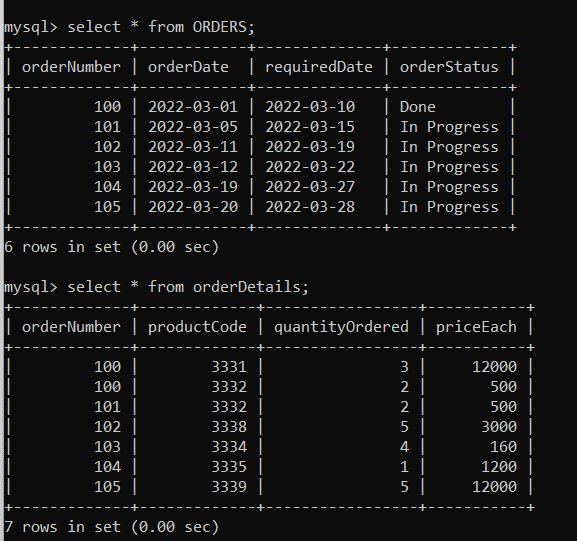
**1. during the transaction execution.**

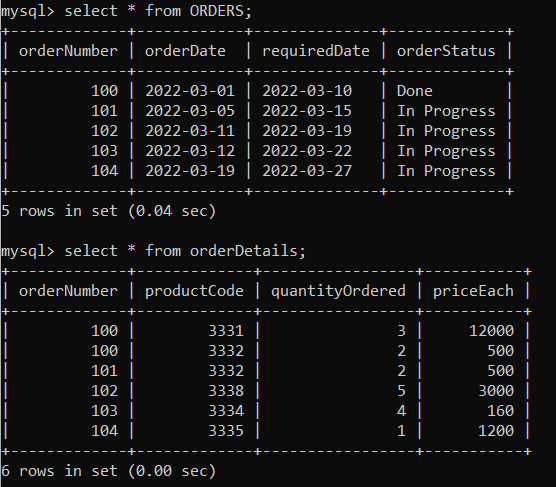
****

**Session 01** : Inserting a new query (an order with ordeNumber 105 )

**Session 01 :**

In this session we can see the newly inserted query (the order with orderNumber=105 ) even before it commits the transaction because the auto commit mode is set to one.



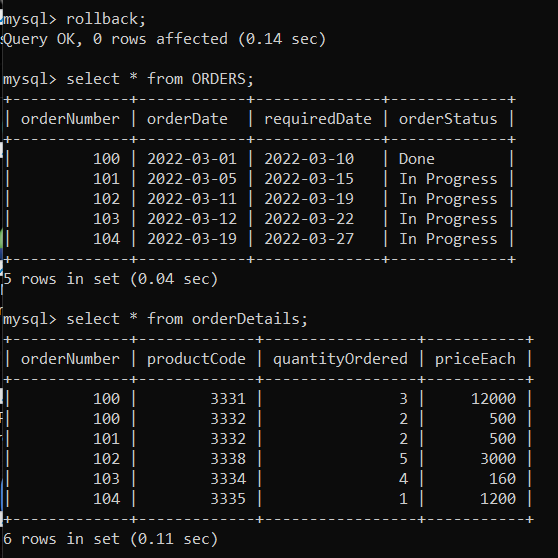


**Session 02:**

In this session we can’t see the newly inserted query (the order with orderNumber=105 ) because the transaction in the session 01 has not committed yet. (transaction in session 01 did not make the changes permanent)

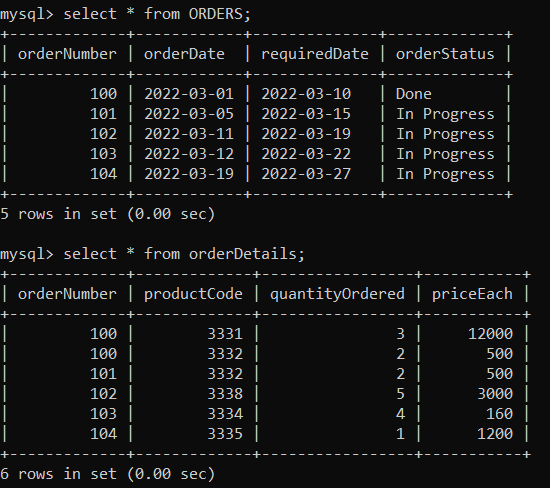
**‘**

**2. after rollback statement.**



**Session 01:**

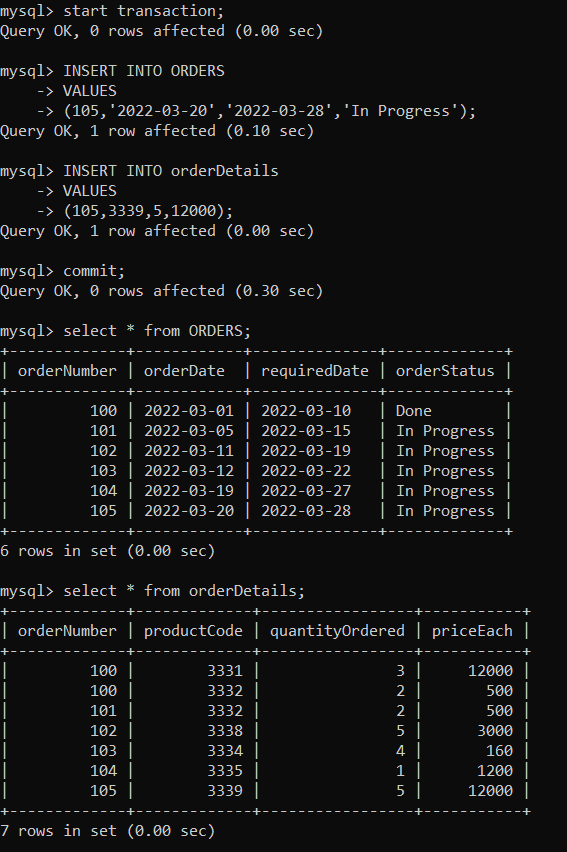
In this session we can’t see the newly inserted query (the order with orderNumber=105 ) because it rollbacks the transaction . ROLLBACK is used to roll back or undo all the changes. It erases all data modifications made from the start of the transaction.



**Session 02:**

In here also we can’t see the newly inserted query (the order with orderNumber=105 ) because the transaction in session 01 has undo the changes done by it.

**3. after the commit statement.**

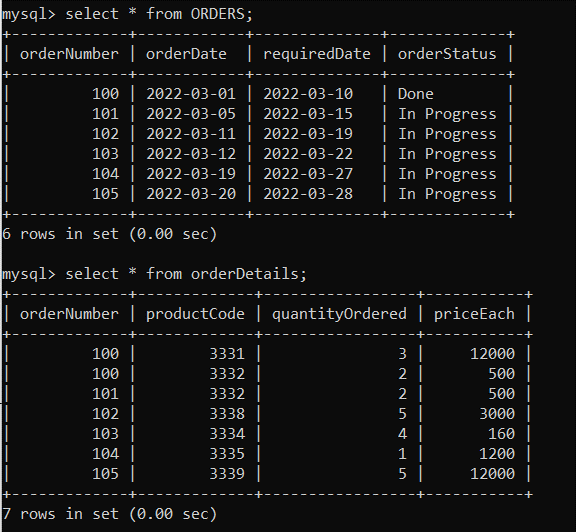


**Session 01:**

In this session we can see the newly inserted query (the order with orderNumber=105 ) because it has committed the transaction . (the changes have been made permanent.)

**Session 02:**

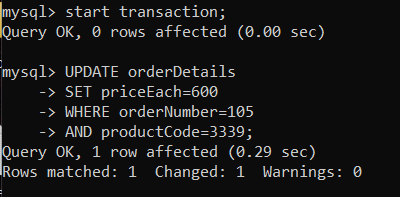
In this session we can see the newly inserted query (the order with orderNumber=105 ) because the transaction in the session 01 has committed . (session 01 has made the changes permanent.)

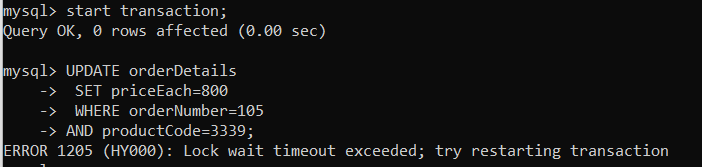


**4. during 2 concurrent transactions, both of them update a record and both of them commit it.**

**Session 01:**

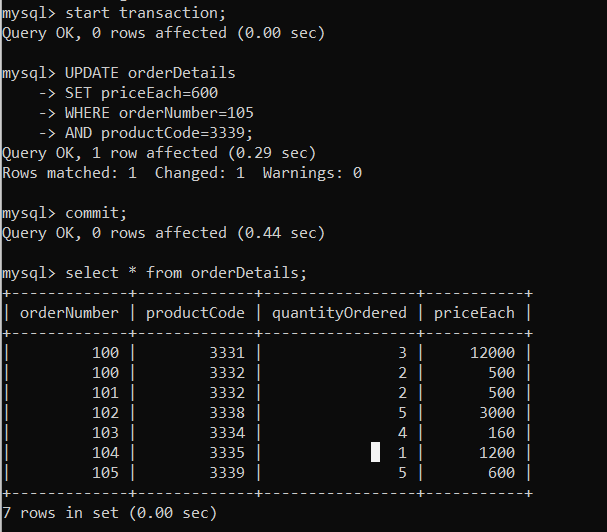
in this session we are going to update priceEach=600 for the orderNumber=105

****

****

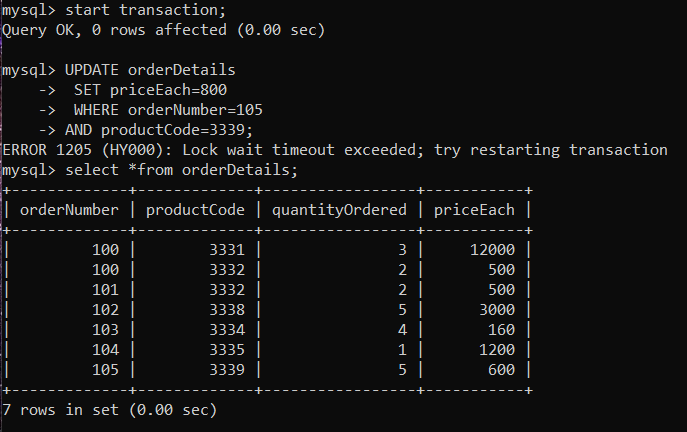
**Session 02 :**

In this session we are going to update priceEach=800 for orderNumber=105, but we are not allowed to do that as the transaction in session 01 has not committed yet.

****

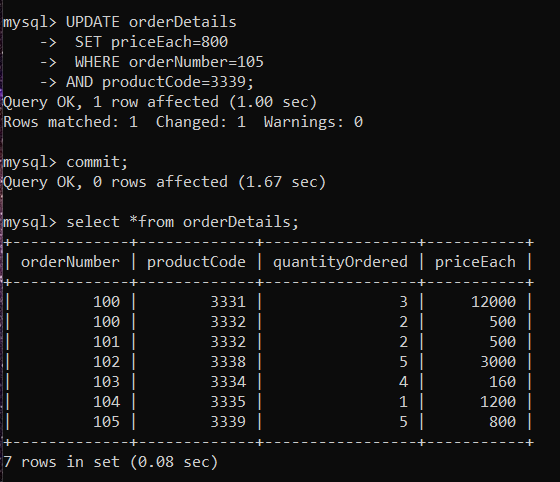
**Session 01 :**

Here we have committed the transaction (set priceEach=600 for orderNumber=105 ) therefore we can see the updates in orderDetails table.

****

**Session 02 :**

Here we are not allowed to do another update as the transaction in session 01 has not committed yet.

****

**Session 02 :**

Once the transaction in session 01 is committed we are allowed to do updates in session 02. And also we can see the new updates done in session 02 in orderDetails table. (set priceEach=800 for orderNumber=105 )