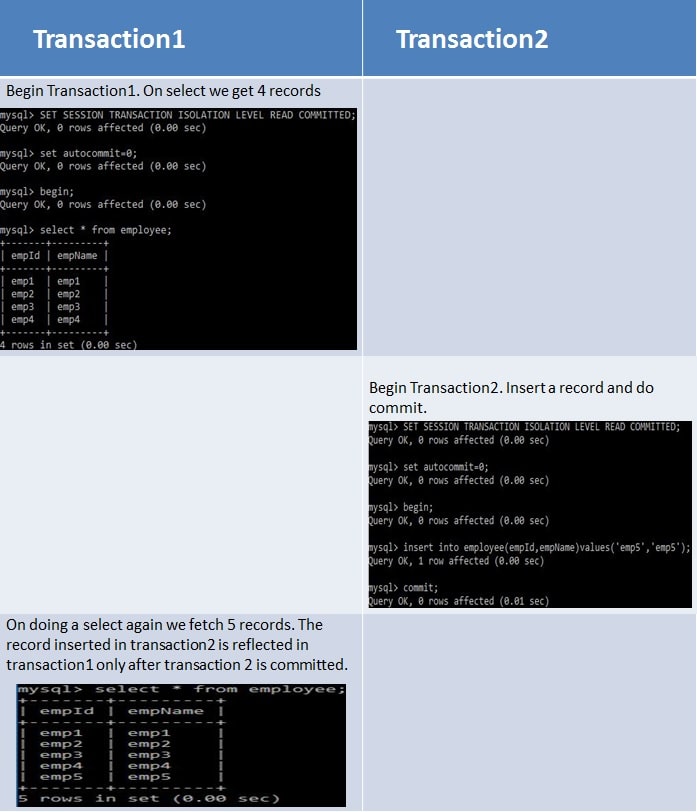
**What is Transaction Isolation?**  
Transaction Isolation defines the database state when two transactions concurrently act on the same database entity. It involves locking of database records. So it describes the **behavior or state of the database when one transaction is working on database entity and then some other concurrent transaction tries to simultaneously access/edit the same database entity.**

The following are the types of Transaction Isolation Levels-

* **SERIALIZABLE:** If two transactions are executing concurrently then it is as if the transactions get executed serially i.e the first transaction gets committed only then the second transaction gets executed. This is **total isolation**. So a running transaction is never affected by other transactions.However this may cause issues as **performance will be low and deadlock might occur**.  
  
* **REPEATABLE\_READ**  
  If two transactions are executing concurrently - **till the first transaction is committed the existing records cannot be changed by second transaction but new records can be added.** After the second transaction is committed, the new added records get reflected in first transaction which is still not committed. For MySQL the default isolation level is REPEATABLE\_READ.  
  However the REPEATABLE READ isolation level behaves differently when using mysql. When using MYSQL we are not able to see the newly added records that are committed by the second transaction.  
  
* **READ\_COMMITTED**  
  If two transactions are executing concurrently - **before the first transaction is committed the existing records can be changed as well as new records can be changed by second transaction.** After the second transaction is committed, the newly added and also updated records get reflected in first transaction which is still not committed.  
  
* **READ\_UNCOMMITTED**  
  If two transactions are executing concurrently - before the first transaction is committed the existing records can be changed as well as new records can be changed by second transaction. **Even if the second transaction is not committed the newly added and also** updated records get reflected in first transaction which is still not committed.  
  