**Tutorial 9 Assessment**

**Question 1**

Investigate how you begin and end a transaction in your favourite programming language. (For the case that you don't have one yet, you could just use PHP or JavaScript in Xampp.) Insert a new country with a new language and at least one city, the capital, into the database.

Ensure that either all three INSERT statements succeed, or all fail.

Test your code by letting one of the statements fail (e.g. clash with a primary key or wrong foreign key) and document what happens. How does the programming language deal with the problem? What does the programmer have to do?

**Insert a new country with a new language and at least one city, the capital, into the database.**

-- Begin the transaction

START TRANSACTION;

-- Insert new country into the country table

INSERT INTO `country` (`Code`,`Name`,`Continent`,`Region`,`SurfaceArea`,`IndepYear`,`Population`,`LifeExpectancy`,`GNP`,`GNPOld`,`LocalName`,`GovernmentForm`,`HeadOfState`,’Capital`,`Code2`) VALUES ('KLD','Kingdom of Lordran','North America',‘Central America’,8120000.00,2011,24505,50.0,5695.00,2847.00,'Lordran',’Lord’,’Gywn’,1,'LR');

-- Insert new language for the new country into the countrylanguage table

INSERT INTO `countrylanguage’ (`CountryCode`,`Language`,`IsOfficial`,`Percentage`) VALUES ('KLD','Oolacilean','T',100.00);

-- Insert at least one city for the new country into the city table (designated as capital)

INSERT INTO `city’ (‘ID’,`Name`,`CountryCode`,`District`,`Population`) VALUES (4080,`Anor Londo`,`KLD`,`City of Gods`,6000);

-- Commit the transaction

COMMIT;

**Ensure that either all three INSERT statements succeed, or all fail.**

-- Begin the transaction

START TRANSACTION;

-- Use SAVEPOINT to provide a named point in the transaction to rollback to in case of error

SAVEPOINT ProcedureSave;

-- Insert new country into the country table

INSERT INTO `country` (`Code`,`Name`,`Continent`,`Region`,`SurfaceArea`,`IndepYear`,`Population`,`LifeExpectancy`,`GNP`,`GNPOld`,`LocalName`,`GovernmentForm`,`HeadOfState`,’Capital`,`Code2`) VALUES ('KLD','Kingdom of Lordran','North America',‘Central America’,8120000.00,2011,24505,50.0,5695.00,2847.00,'Lordran',’Lord’,’Gywn’,1,'LR');

-- Insert new language for the new country into the countrylanguage table

INSERT INTO `countrylanguage’ (`CountryCode`,`Language`,`IsOfficial`,`Percentage`) VALUES ('KLD','Oolacilean','T',100.00);

-- Insert at least one city for the new country into the city table (designated as capital)

INSERT INTO `city’ (‘ID’,`Name`,`CountryCode`,`District`,`Population`) VALUES (4080,`Anor Londo`,`KLD`,`City of Gods`,6000);

-- Check for errors

IF @@ERROR <> 0

BEGIN

-- Rollback to the savepoint if an error occurred

ROLLBACK TO ProcedureSave;

END

ELSE

BEGIN

-- Commit the transaction if everything succeeded

COMMIT;

END;

If the second INSERT statement tries to insert a duplicate primary key into the country table, it will trigger an error because it would violate primary key constraints and fail due to a clash with a primary key. When this happens, the error will be caught, and the transaction will be rolled back to the ProceduceSave, meaning none of the changes made within the transaction will be applied to the database, even though the INSERT statement was successful. Having errors and rollbacks ensures the database remains consistent even if an error occurs during the transaction.

The programming language (SQL) deals with the problem of errors and transactions through a error handling by having error messages, and transaction control mechanisms like using BEGIN TRANSACTION, COMMIT, and ROLLBACK statements.

The programmer has to implement these transaction control mechanisms as well by using statements like IF @@ERROR <> 0 to check for errors (it will return 0 if there are no errors), ROLLBACK to undo changes, SAVEPOINT ProcedureSave to have a savepoint to ROLLBACK to.