**Practical 3 – Relational Databases**

**Lee O’Sullivan - 2992902**

**Question 1 -**

Employee (employeeID, fname, lname, address, phone, roleID, roleName);

employeeID – varchar(5)

fname – varchar(30)

lname – varchar(30)

address – varchar(50)

phone – varchar(30)

roleID – varchar(5)

roleName – varchar(30)

Department (departmentID, departmentName, address, tel\_No);

departmentID – varchar(5)

departmentName – varchar(30)

address – varchar(50)

tel\_No – varchar(30\_

Team(teamID, teamName, teamLeaderID, projectID, teamMemberID);

teamID – varchar(5)

teamName – varchar(30)

teamLeaderID – varchar(5)

projectID – varchar(5)

teamMemberID – varchar(5)

Project(projectID, projectName, projectDescription, projectDuration, teamID);

projectID – varchar(5)

projectName – varchar(30)

projectDescription – varchar(100)

projectDuration – varchar(5)

teamID – varchar(5)

Product(productID, productName, productDescription, wholesalePrice, retailPrice, projectID);

productID – varchar(5)

productName – varchar(30)

productDescription – varchar(100)

wholesalePrice – Money

retailPrice – Money

Customer(customerID, accNum, customerName, accBalance, paymentAmount, purchaseDate, purchaseAmount, tel\_No, address, dob);

customerID – varchar(5)

accNum – varchar(5)

CustomerName – varchar(60)

accBalance – Money

PaymentAmount – Money

purchaseDate – Date

purchaseAmount – Money

tel\_No – varchar(30)

address – varchar(50)

dob – Date

The tables above are not normalised thus when making the database these tables would need to be normalised, for example the Customer table would need to be broken down into multiple tables such as accountDetails, purchases, and the Product table would need to be split into product and stock as not all the attributes depend on the primary key. The team and project tables would also need to be split

**Question 2 –**

Employee (emp-id, first-name, last-name, address, phone, role-id, role-name)

1NF –

As an employee can have multiple roles, a composite key will be needed, otherwise an employee with 2 roles would cause an error in the database or 2 lines with the same employeeID while employeeID is the sole primary key will also cause an error, so roleID and employeeID will act as a composite key, this will allow employees which have multiple roles to cause no error if the employeeID is repeated as long as the roleID and employeeID aren’t duplicated simultaneously.

Employee (employeeID, fname, lname, address, phone, roleID, roleName)

2NF –

Due to the employee table having a composite key there is some partial dependency within the table. So a new table Role is created, with roleID as a primary key and roll name as an attribute, another table with roleID and employeeID is created as if employeeID was in the Role table.

Employee(employeeID, fname, lname, address, phone);

Role(roleID, rollName);

EmployeeRoles (roleID, employeeID);

3NF –

As 3nf states there can be no transitive dependencies then the employee table must be altered as a person’s phone number depends on their address as an employee’s phone number can be different depending on where they live as different cities and countries have different phone prefix’s. So a new table EmployeePersonalDetails is made with address as the primary key which becomes a foreign key in the employee table.

Employee (employeeID, fname, lname, address);

EmployeePersonalDetails (address, phone);

Role (roleID, rollName);

EmployeeRole (roleID, employeeID);