**LabSheet 9**

**Sample solutions**

**Part 1:**

Exercise a)

Transaction(TransactionNo(PK), TransactionDate, CustID, Item, Qty, Amount)

**1NF**

**Repeating group:** Item, Qty, Amount (move to a new table, give a PK (itemID) and include PK from original table which also becomes a FK – new table has a composite PK)

Transaction(TransactionNo(PK), TransactionDate, CustID)

Transaction\_Item(TransactionNo(PK,**FK**),ItemID(PK),Item, Qty, Amount)

**2NF**

**Partial Dependency:** *Item* is only dependent on *ItemID(PK)* and not the composite PK of [*TransactionNo(PK),ItemID(PK)*] – move *Item* to a new table, include the part of the composite PK the attribute is dependent on. This (*itemID*) now becomes a FK in the Transaction\_Item table.

Transaction(TransactionNo(PK), TransactionDate, CustID)

Transaction\_Item(TransactionNo(PK,FK),ItemID(PK,**FK**), Qty, Amount)

Item(ItemID(PK), Item)

**3NF**

**Transitive Dependencies:** CustID is functionally dependent on another attribute that is not the PK in the Transaction table. (i.e. it is not functionally dependent on TransactionNo(PK)) – move custID to a new table, give it a PK (custID as it happens is a natural PK itself) and add this PK to the original table as a FK. Cust\_Name added to make the table more readable!

Final Tables:

Transaction(TransactionNo(PK), TransactionDate, CustID(**FK**))

Transaction\_Item(TransactionNo(PK,FK),ItemID(PK,FK), Qty, Amount)

Item(ItemID(PK), Item)

Customer(custID(PK), cust\_Name)

Exercise b)

Team(TeamID(PK), TeamName, Mgr, matchID, time, Date)

**Repeating group:** matchID, time, Date (move to a new table, give a PK (matchID) and include PK from original table which also becomes a FK – new table has a composite PK)

1NF

Team(TeamID(PK), TeamName, Mgr)

Team\_Match(TeamID(PK,**FK**),matchID(PK), time, Date)

2NF

**Partial Dependency:** *time,date* are only dependent on *matchID(PK)* and not the composite PK of [*TeamID (PK),matchID(PK)*] – move *time,date* to a new table, include the part of the composite PK the attribute is dependent on. This (*matchID*) now becomes a FK in the Team\_Match table.

Match(matchID(PK),time,date)

Team\_Match(TeamID(PK,FK),matchID(PK,**FK**))

Team(TeamID(PK), TeamName, Mgr)

3NF

**Transitive Dependencies:** mgr is functionally dependent on another attribute that is not the PK in the Team table. (i.e. it is not functionally dependent on TeamID (PK)) – move *mgr* to a new table, give it a PK (*mgrID*) and add this PK to the original table as a FK.

Manager(mgrID(PK), mgr)

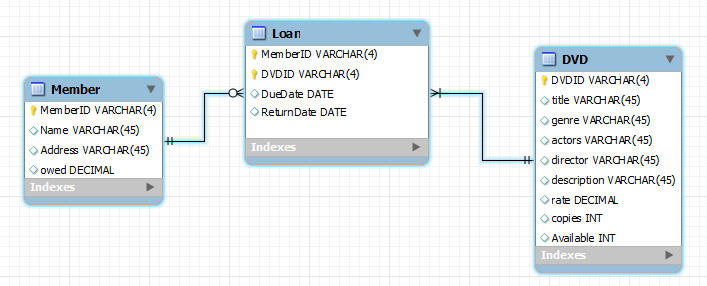
Team(TeamID(PK), TeamName, mgrID(**FK**))

Match(matchID(PK),time,date)

Team\_Match(TeamID(PK,FK),matchID(PK,FK))

**Part 2:**

**ERD:**



**Relational Model:** (composite attributes (address) + multivalued attributes (actosr) to be dealt with at this atage along with PK and FKs). Assuming here that name is not divisible but students may have Fname, Lname.

Member(MemberID(PK), name, addr1, addr2, addr3, owed)

Loan(MemberID(PK,FK), dvdID(PK,FK), dueDate, returnDate)

DVD(dvdID(PK),title, genre, director, description, rate, copies, available)

Cast(dvdID(PK,FK),ActorID(PK), Actor)

**Normalisation:**

**1NF –** no repeating group!

**2NF** – Loan is OK, Cast – Actor only functionally dependent on actorID

Cast(dvdID(PK,FK),ActorID(PK,FK))

Actor(ActorID(PK),Actor)

**3NF –** genre (dvdID doesn’t uniquely identity different genre types) + director (dvdID doesn’t uniquely identity different directors)

Genre(genreID(PK), genre)

Director(directorID(PK), director)

DVD(dvdID(PK),title, description, rate, copies, available, genreID(FK), directorID(FK))

**Final Tables:**

Member(MemberID(PK), name, addr1, addr2, addr3, owed)

Loan(MemberID(PK,FK), dvdID(PK,FK), dueDate, returnDate)

Cast(dvdID(PK,FK),ActorID(PK,FK))

Actor(ActorID(PK),Actor)

Genre(genreID(PK), genre)

Director(directorID(PK), director)

DVD(dvdID(PK),title, director, description, rate, copies, available, genreID(FK))