DataMiningandDataWarehousing Laboratory (CSPC-328)

B.TechVIthSemester
(January–June2024)

Submittedby

Kirtan Gohil(21103073) Group-G3

Submittedto

Dr.SamayveerSingh

DepartmentofComputerScience&Engineering
Dr.B.R.AmbedkarNationalInstituteofTechnology Jalandhar
-144008,Punjab,India

TableofContent

Practical1

Aim:-DesigningDatabaseUsingERModelling

Que1CreatedatabasedesignforHospitalManagementSystemusingER Modelling

The patient, physician, department, room, and appointment are the entities that make up the hospital administration system.
The following is a relationship between these entities areas:

An appointment is for one patient and one doctor. A patient may have one or more appointments.
A doctor may schedule many appointments with various patients.
One department is assigned to a doctor.
A department may employ several physicians.
One patient can be assigned to one room, and one or more patients can be housed in a room.
A doctor is in charge of each room, however they can oversee more than one.
These relationships allow us to develop the subsequent ER model:

1.Entities:

- Patient with attributes (Name, Age, Room Number, and Patient ID).
- Physician with the following attributes: DepartmentID, Name, Specialty, DoctorID.
- Department including features like DepartmentName, DepartmentID.
  Room has the following attributes: bed count, supervising doctor ID, room number.
- Appointment with the following attributes: PatientID, DoctorID, Date, Time, Appointment ID.

2. Relationships:
A patient's relationship with an appointment is symbolized by a "has" relationship.
A doctor-patient connection is based on a "conducts" relationship.
A department and a doctor are associated, represented by a "assignedto" relationship.
Multiple doctors are associated with a department through the "employs" relationship.
A patient and a room are connected through a "assignedto" relationship.
A room can have a relationship with numerous patients, represented by a "houses" relationship.
A room has a relationship with a doctor, which is represented by a "supervisedby" relationship.
An diagram representing things as boxes and relationships as lines linking these boxes—often with additional symbols to signify the kind and cardiacality of the interactions—would be the visual representation of the ER model.
The relationships and entities within the hospital management system are shown in Fig. 1.1.

The patient, doctor, department, room, and appointment are the five main entities that are included. Patients may schedule many appointments, with a doctor and a single patient at each visit.Physicians are assigned to departments, and each department may have more than one physician on staff.Patients are assigned to rooms, and each room can accommodate several patients under a single doctor's care.The ER graphic also shows how a doctor is able to oversee many rooms.The entities are linked together by a number of links, including "has," "conducts," "assigned to," "employees," "houses," and "supervisedby," which illustrate the many relationships and interactions that exist in a medical setting.The diagram shows the relationships between the various components of the system and acts as a visual representation of the data model.

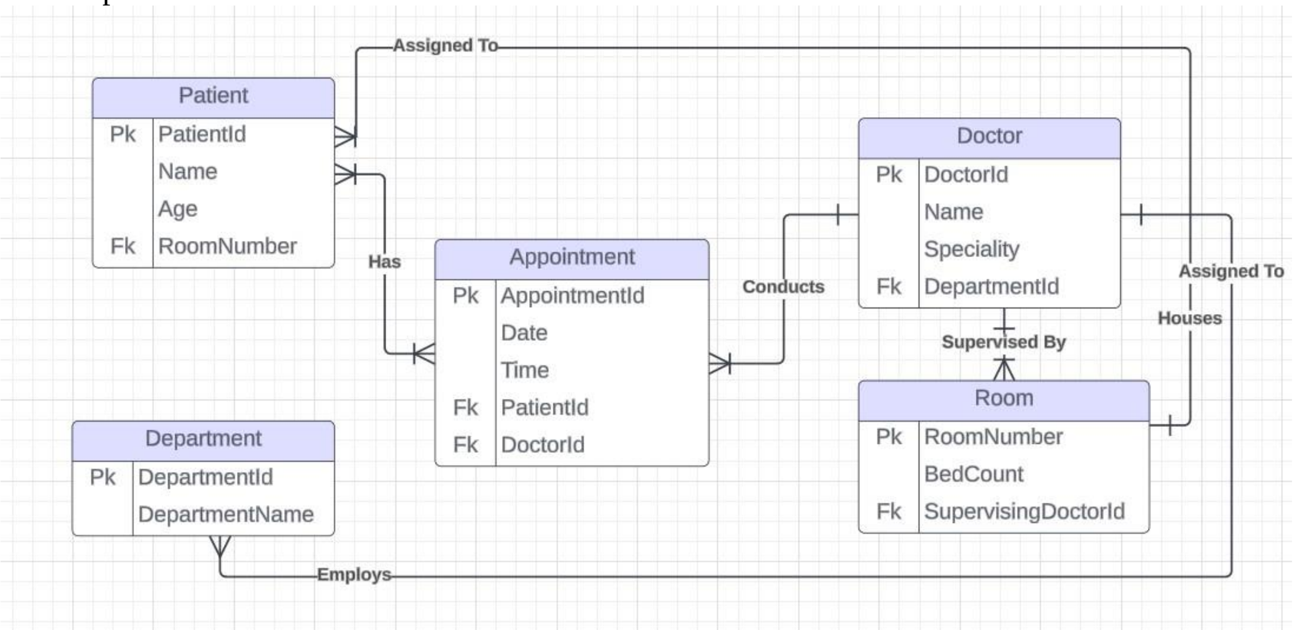


Fig.1.1:ERdiagramforHospitalManagementSystem

Que2CreatedatabasedesignforLibraryManagementSystemusingER Modelling

The following entities are included in the library management system: book, author, borrower, genre, and loan.The following is a relationship between these entities areas:

A book is authored by one or more writers.

- A writer can pen one or more books.
- A borrower may check out many books, but a book may be checked out by just one borrower. •in real time.
- A book falls into a specific genre.
- A genre can be connected to more than one book.
- The loan specifies when a book was checked out and when it must be returned.

These relationships led us to derive the subsequent ER model:

1. Entities

- Book with attributes: Title, ISBN, BookID, GenreID.
- Author with attributes: Name, BirthDate, and AuthorID.
- Borrower with properties: Name, Address, Phone, and Borrower ID.
- Genre with attributes (GenreName, GenreID).
- Loan with attributes: BookID, BorrowerID, Borrow Date, Due Date, Loan ID.

2. Relationships:

- A book is linked to its author(s) by means of a "writtenby" relationship.

One or more books are associated with an author via a "writes" relationship.

- A "borrows" relationship connects a borrower with books.

- A book and borrower have a relationship thanks to the "isborrowedby" connection.
- A book and a genre are connected by a "belongsto" relationship.
- Aloani is related to a borrower and a book through a "issued for" relationship. • A genre is connected to many books through a "encompasses" relationship.

To visualize the ER model, entities would be shown as boxes with relationships between them shown as lines or arrows. The types and cardinality of each link would be represented by annotations or symbols.

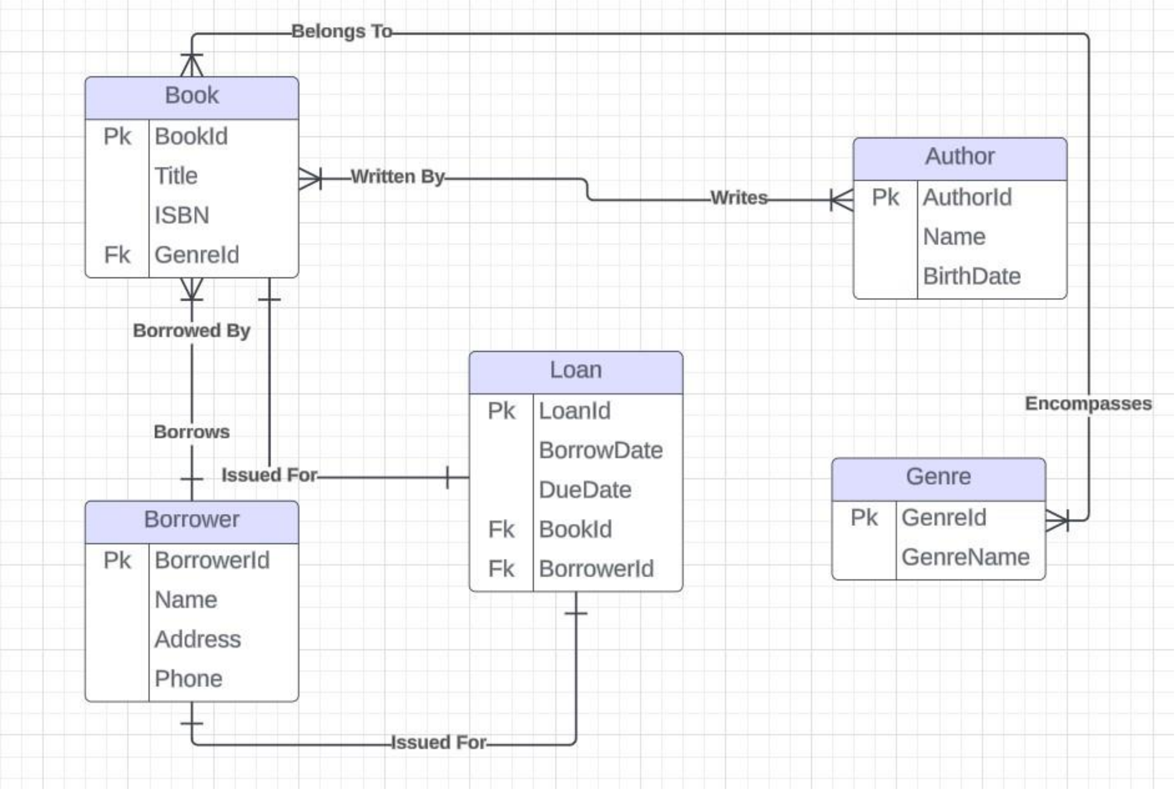


Fig.1.2:ERdiagramforLibraryManagementSystem

Figure 1.2 illustrates the connections and entities in the Library Management System.There are five main components to it: Book, Author, Borrower, Genre, and Loan.The graphic shows how a book is linked to one or more writers by a "written by" relationship, enabling numerous authors to contribute to a single work.Books are linked to authors by a "writes" relationship, meaning that an author is able to write more than one book.The relationship "borrows" links borrowers to books; this means that one borrower may check out numerous books at once, but only one borrower may check out a book at a time.Books are grouped by genres using a "belongsto" relationship, which indicates that a given book is part of a particular genre. Genres might include more than one book.The "issuedfor" relationships bind loans to both borrowers and books, indicating the date a book was borrowed and the return deadline.

Practical2

Aim:-NormalisingaDatabaseUsingGriffithNormalisation Tool

Que1Understandthefunctionaldependenciesandnormalizeeach functional dependencyupto2NF,3NF,andBCNFusingnormalizationtoolfrom GriffithUniversity. Foreachquestion:

- Findtheminimalcover.
- Identifythecandidatekey(s)orprimarykey.
- Checkforpartialdependenciestodetermineiftherelationisin2NF.
- Checkfortransitivedependenciestoassessiftherelationisin3NF.
- CheckfortransitivedependenciestoassessiftherelationisinBCNF.

A.StudentDatabase:

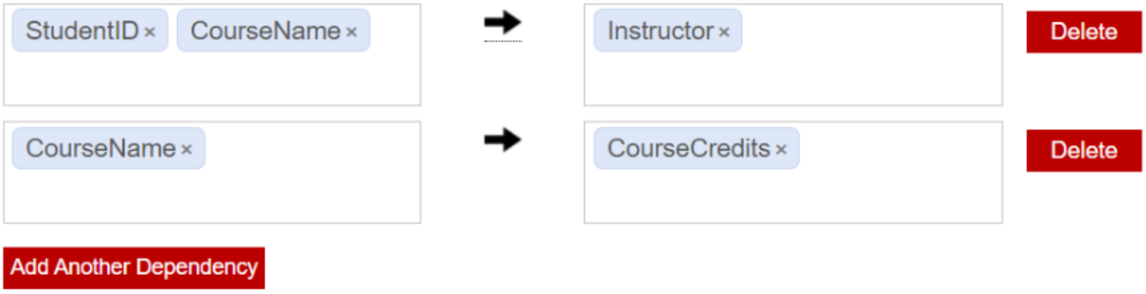
Giventherelation:

StudentCourses(StudentID,CourseName,Instructor,CourseCredits)

andthefunctionaldependencies: StudentID,CourseName→Instructor

CourseName→CourseCredits

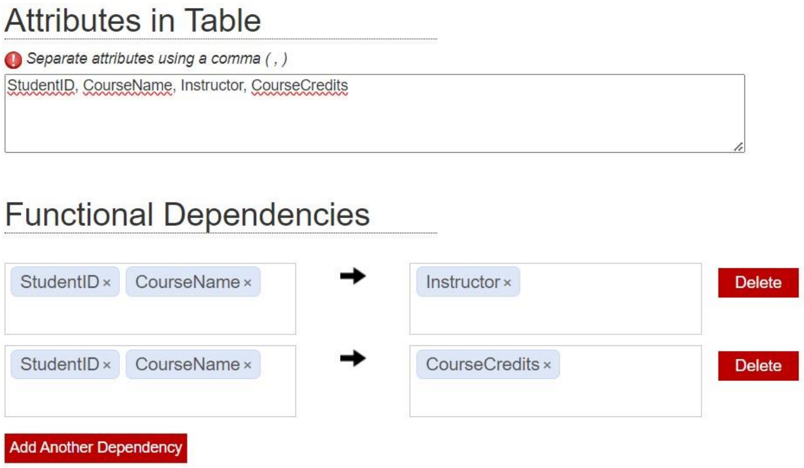
PreviousFunctionalDependencies



StudentID × CourseName × → Instructor ×

CourseName × → CourseCredits ×

Fig1.A.1 ModifiedFunctionalDependencies



Attributes in Table

Separate attributes using a comma (,)

StudentID, CourseName, Instructor, CourseCredits

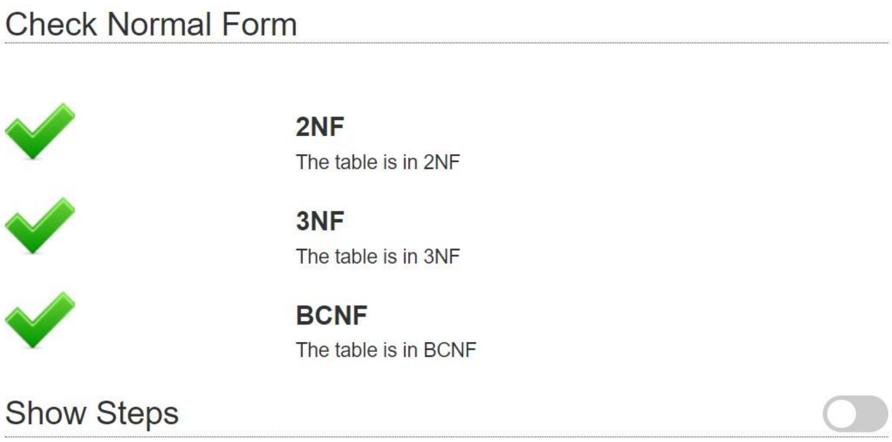
Functional Dependencies

StudentID × CourseName × → Instructor ×

StudentID × CourseName × → CourseCredits ×

Fig1.A.2

Result



Check Normal Form

✓ **2NF**
The table is in 2NF

✓ **3NF**
The table is in 3NF

✓ **BCNF**
The table is in BCNF

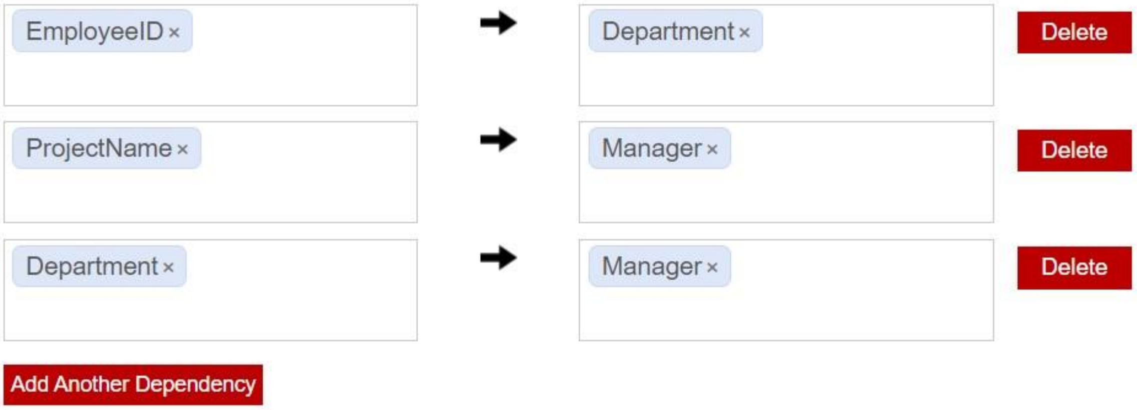
Show Steps ○

Fig1.A.3

Fig1.A.1showspreviousFunctionalDependencieswhicharenotinBCNF.Fig1.A.2 showsnewFunctionalDependencieswhichshowsIfyouknowastudent'sIDand thenameofthecoursethey'retaking,youcandeterminetheinstructorwhoteaches thatcourseandhowmanycreditsthatcoursecarries.Fig1.A.3showstheresultthat newFDsareinBCNF.

B.EmployeeManagement:
Giventherelation:
EmployeeProjects(EmployeeID,ProjectName,Manager,Department)
withthefunctionaldependencies:
EmployeeID→Department
ProjectName→Manager Department→Manager

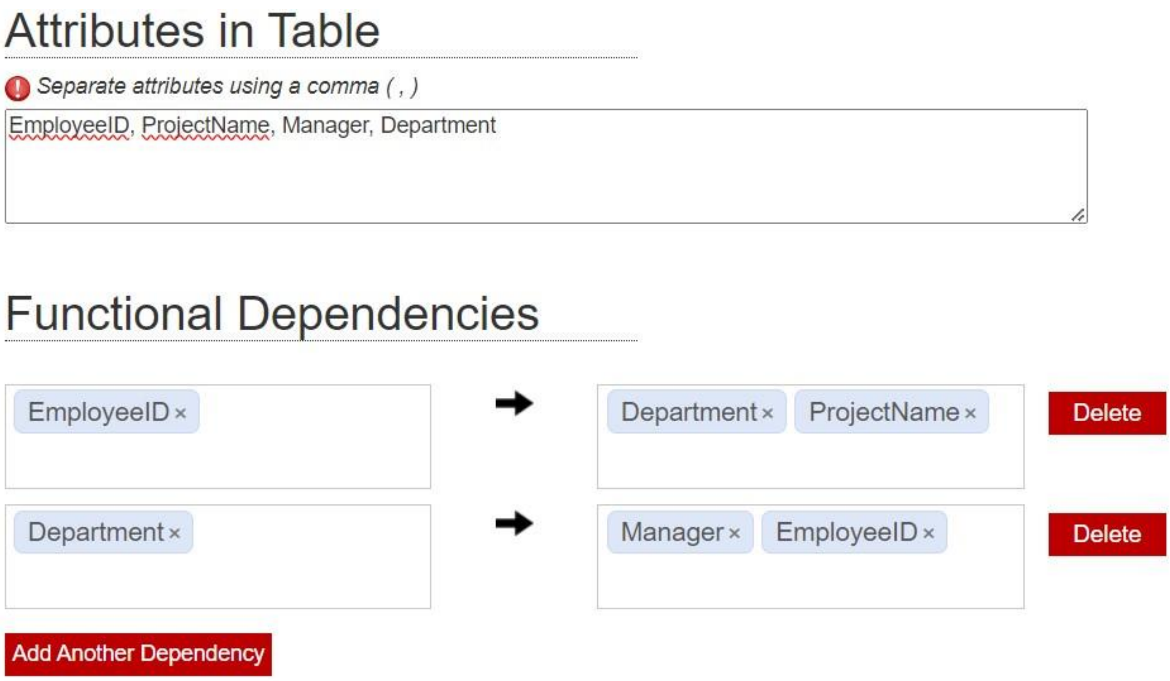
PreviousFunctionalDependencies



EmployeeID ×	→	Department ×	Delete
ProjectName ×	→	Manager ×	Delete
Department ×	→	Manager ×	Delete

Add Another Dependency

Fig1.B.1 ModifiedDependencies



Attributes in Table

Separate attributes using a comma (,)

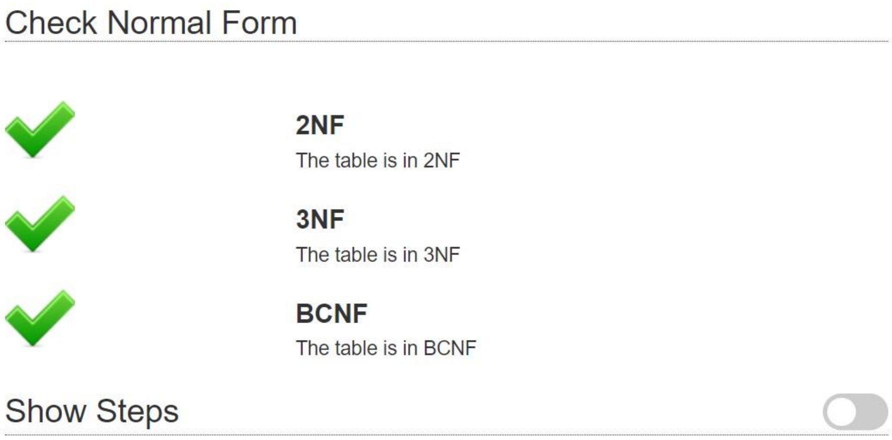
EmployeeID, ProjectName, Manager, Department

Functional Dependencies

EmployeeID ×	➜	Department × ProjectName ×	Delete
Department ×	➜	Manager × EmployeeID ×	Delete

Fig1.B.2

Result



Check Normal Form

✓ **2NF** The table is in 2NF

✓ **3NF** The table is in 3NF

✓ **BCNF** The table is in BCNF

Fig1.B.3

Fig1.B.1showspreviousFunctionalDependencieswhicharenotinBCNF.Fig1.B.2 showsnewFunctionalDependencieswhichshowsGivenanEmployeeID,wecan determinetheProjectNameandDepartmentassociatedwiththatemployee. GivenaDepartment,wecandeterminetheManagerandEmployeeIDassociated withthatdepartment.Fig1.B.3showstheresultthatnewFDsareinBCNF.

C.LibrarySystem:

Considertherelation:

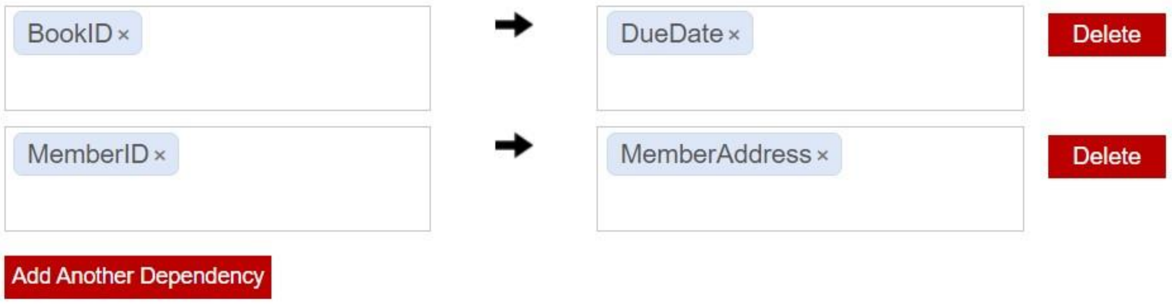
BookLending(BookID,MemberID,BorrowDate,DueDate,MemberAddress)

andthefunctionaldependencies:

BookID→DueDate

MemberID→MemberAddress

PreviousFunctionalDependencies

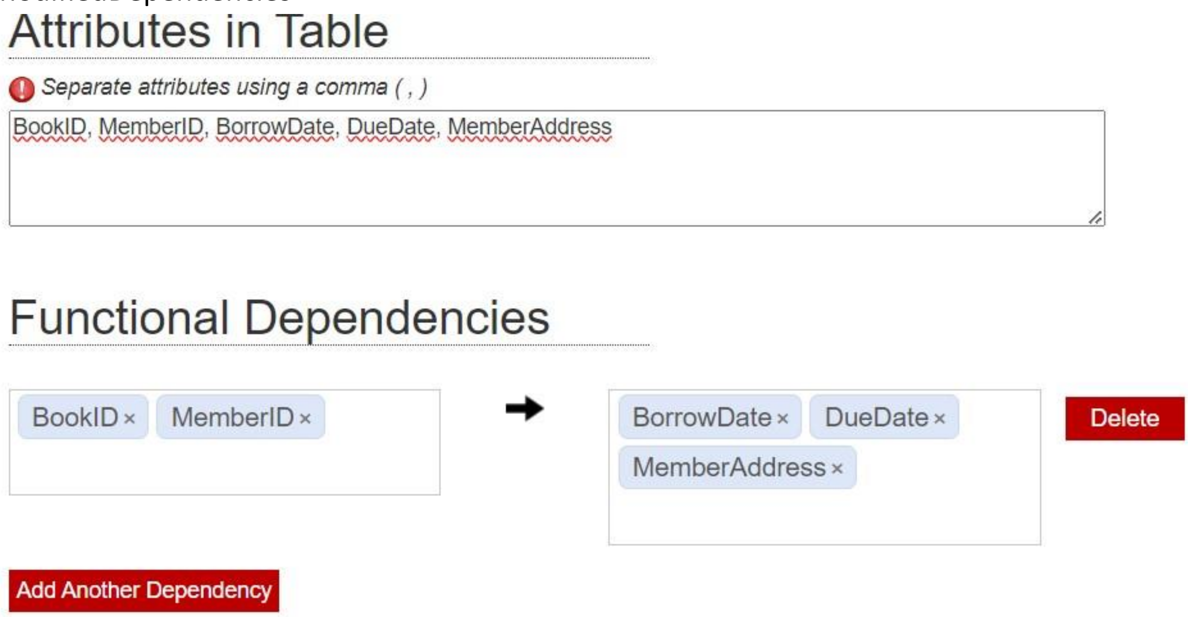


BookID × → DueDate ×

MemberID × → MemberAddress ×

Fig1.C.1

ModifiedDependencies



Attributes in Table

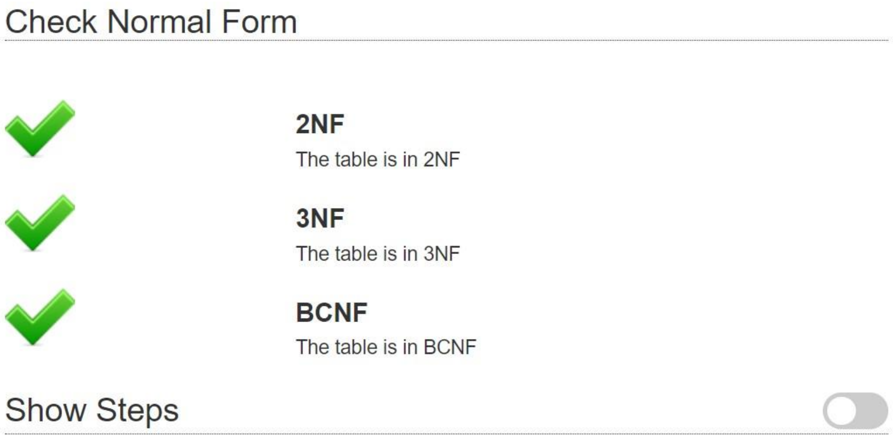
Separate attributes using a comma (,)

BookID, MemberID, BorrowDate, DueDate, MemberAddress

Functional Dependencies

BookID × MemberID × → BorrowDate × DueDate × MemberAddress ×

Fig1.C.2 Result



Check Normal Form

✓ **2NF**
The table is in 2NF

✓ **3NF**
The table is in 3NF

✓ **BCNF**
The table is in BCNF

Show Steps

Fig1.C.3

Fig1.C.1showspreviousFunctionalDependencieswhicharenotinBCNF.Fig1.C.2 showsnewFunctionalDependencieswhichshowsIfyouknowwhichbookis borrowedbywhichmember,youcandeterminethemember'saddress,theduedate ofthebook,andthedateitwasborrowed.Fig1.C.3showstheresultthatnewFDs areinBCNF.

D.HospitalManagement:

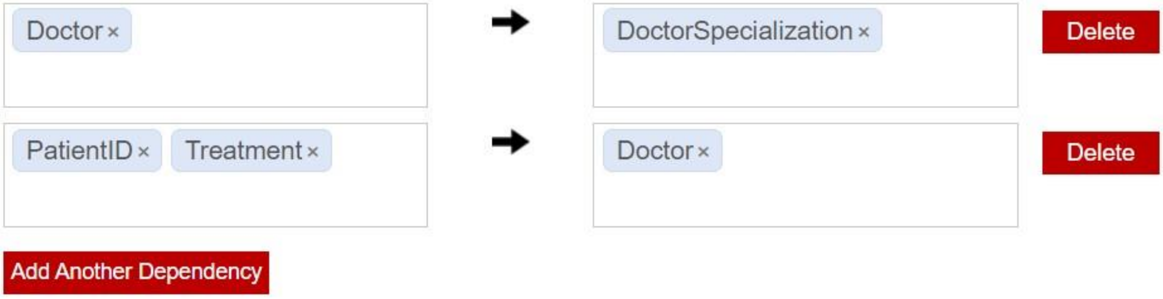
-Fortherelation:

PatientTreatment(PatientID,Treatment,Doctor,DoctorSpecialization)

withthefunctionaldependencies: Doctor→DoctorSpecialization

PatientID,Treatment→Doctor

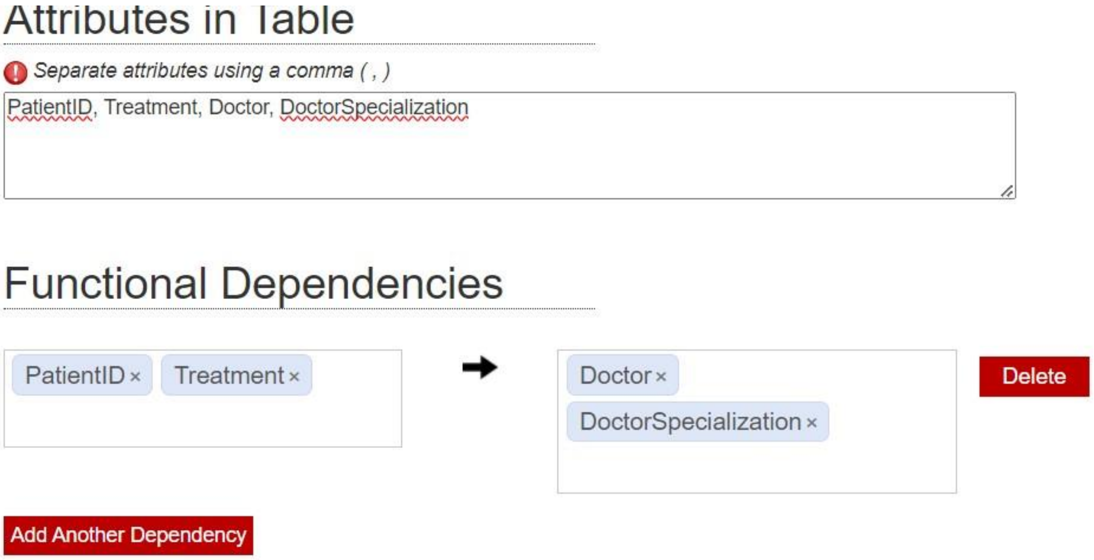
PreviousFunctionalDependencies



Doctor × → DoctorSpecialization ×

PatientID × Treatment × → Doctor ×

Fig1.D.1 ModifiedDependencies



Attributes in Table

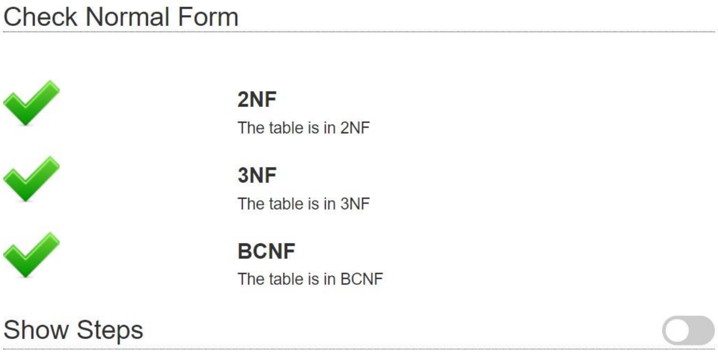
Separate attributes using a comma (,)

PatientID, Treatment, Doctor, DoctorSpecialization

Functional Dependencies

PatientID × Treatment × → Doctor × DoctorSpecialization ×

Fig1.D.2 Result



Check Normal Form

✓ **2NF** The table is in 2NF

✓ **3NF** The table is in 3NF

✓ **BCNF** The table is in BCNF

Show Steps

Fig1.D.3

Fig1.D.1showspreviousFunctionalDependencieswhicharenotinBCNF.Fig1.D.2 showsnewFunctionalDependencieswhichshowsIfyouknowthePatientIDandthe treatmenttheyareundergoing,youcandeterminewhichdoctorisresponsiblefor providingthattreatment,alongwiththedoctor'sspecialization.Fig1.D.3showsthe resultthatnewFDsareinBCNF.

E.AirlineReservationSystem:

-Giventherelation:

FlightReservations(FlightNumber,Date,PassengerID,SeatNumber,ClassType, Price,DepartureTime,ArrivalTime,DepartureCity,ArrivalCity) -Functionaldependenciesare:

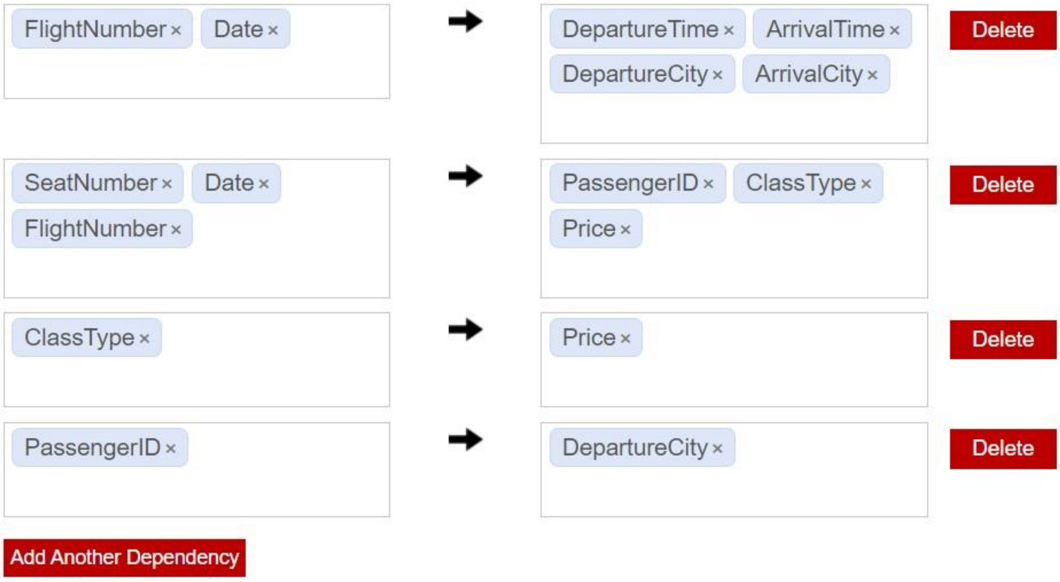
FlightNumber,Date→DepartureTime,ArrivalTime,DepartureCity,ArrivalCity

SeatNumber,Date,FlightNumber→PassengerID,ClassType,Price

ClassType→Price

PassengerID→DepartureCity

PreviousFunctionalDependencies



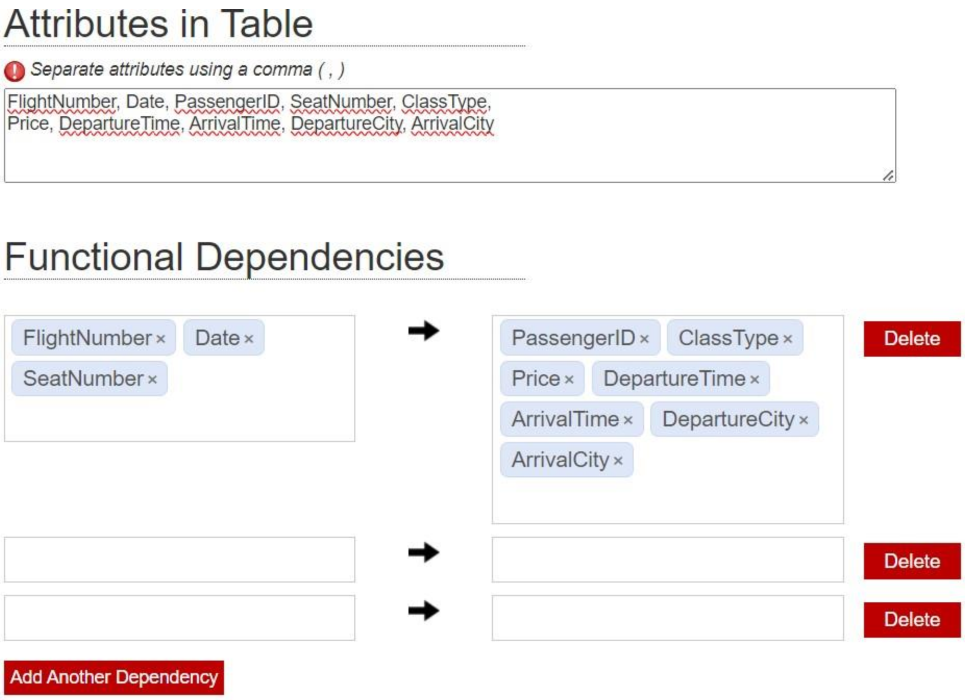
FlightNumber × Date × → DepartureTime × ArrivalTime × DepartureCity × ArrivalCity ×

SeatNumber × Date × FlightNumber × → PassengerID × ClassType × Price ×

ClassType × → Price ×

PassengerID × → DepartureCity ×

Fig1.E.1 ModifiedDependencies



Attributes in Table

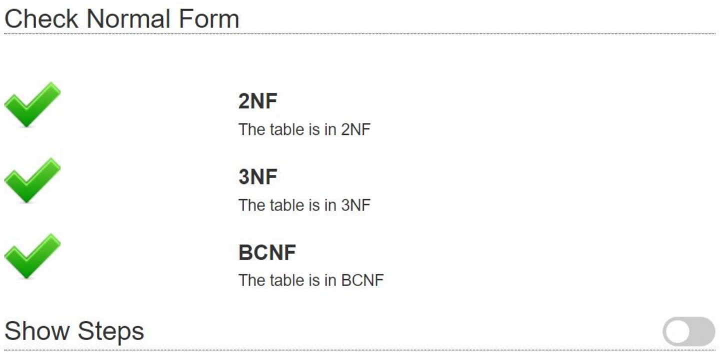
Separate attributes using a comma (,)

FlightNumber, Date, PassengerID, SeatNumber, ClassType, Price, DepartureTime, ArrivalTime, DepartureCity, ArrivalCity

Functional Dependencies

FlightNumber, Date, SeatNumber → PassengerID, ClassType, Price, DepartureTime, ArrivalTime, DepartureCity, ArrivalCity

Fig1.E.2 Result



Check Normal Form

✓ **2NF** The table is in 2NF

✓ **3NF** The table is in 3NF

✓ **BCNF** The table is in BCNF

Show Steps ○

Fig1.E.3

Fig1.E.1showspreviousFunctionalDependencieswhicharenotinBCNF.Fig1.E.2 showsnewFunctionalDependencieswhichshowsthatifyouhaveinformation abouttheflightnumber,date,andseatnumber,youcandeterminethedetailsrelated tothatspecificbooking,includingthedepartureandarrivaltimes,cities,passenger ID,classtype,andpriceassociatedwiththatbooking.Fig1.E.3showstheresultthat newFDsareinBCNF.

F.6.UniversityEnrolmentSystem:

-Giventherelation:

Enrollments(StudentID,CourseCode,Semester,Grade,InstructorID,CourseName, CourseCredits,Department)

-Functionaldependenciesare:

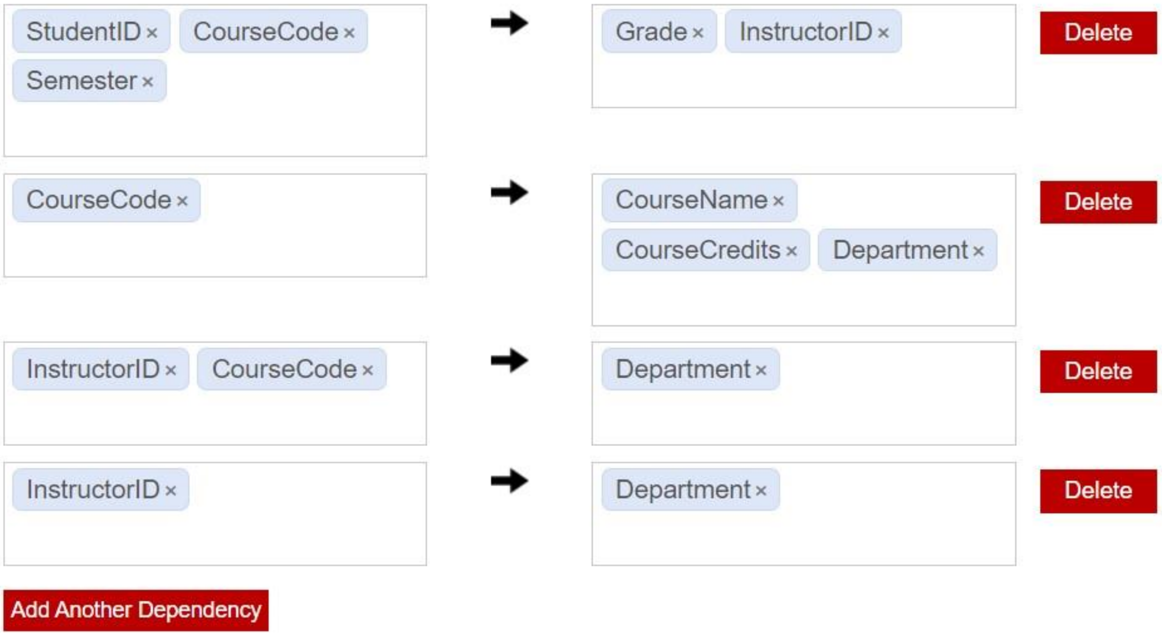
StudentID,CourseCode,Semester→Grade,InstructorID

CourseCode→CourseName,CourseCredits,Department

InstructorID,CourseCode→Department

InstructorID→Department

PreviousFunctionalDependencies



StudentID × CourseCode × Semester × → Grade × InstructorID ×

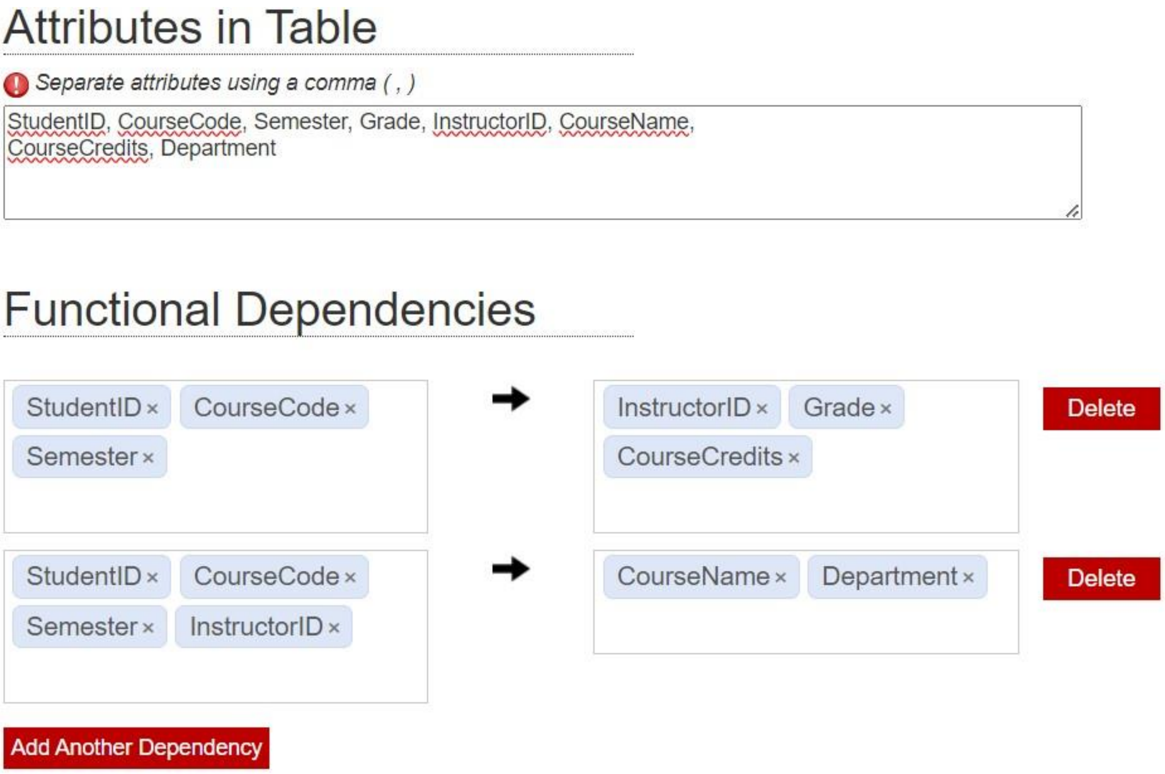
CourseCode × → CourseName × CourseCredits × Department ×

InstructorID × CourseCode × → Department ×

InstructorID × → Department ×

Fig1.F.1

ModifiedDependencies



Attributes in Table

Separate attributes using a comma (,)

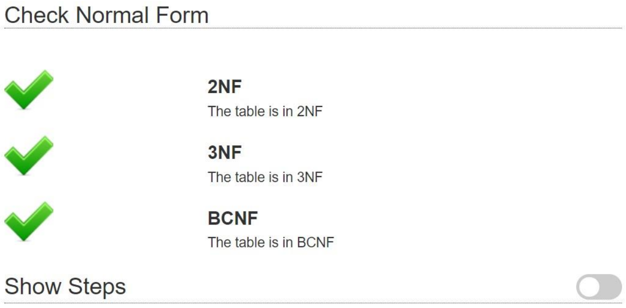
StudentID, CourseCode, Semester, Grade, InstructorID, CourseName, CourseCredits, Department

Functional Dependencies

StudentID × CourseCode × Semester × → InstructorID × Grade × CourseCredits ×

StudentID × CourseCode × Semester × InstructorID × → CourseName × Department ×

Fig1.F.2 Result



Check Normal Form

✓ **2NF**
The table is in 2NF

✓ **3NF**
The table is in 3NF

✓ **BCNF**
The table is in BCNF

Show Steps

Fig1.F.3

Fig1.F.1showspreviousFunctionalDependencieswhicharenotinBCNF.Fig1.F.2 showsnewFunctionalDependencieswhichshowsthatforagivenstudent,a specificcourseinaparticularsemesteruniquelydeterminesthegradereceivedby thestudent,theinstructorteachingthecourse,andthenumberofcreditsassociated withthecourse. Itmeansthatforagivenstudenttakingaspecificcourseinaparticularsemester withaparticularinstructor,thereisonlyonedepartmenttowhichthecoursebelongs andonespecificnameforthecourse.Fig1.F.3showstheresultthatnewFDsarein BCNF.

G.MusicStreamingPlatform:

-Fortherelation:

UserPlays(UserID,SongID,Date,ArtistName,Album,Genre,PlayCount, SubscriptionType)

-Functionaldependenciesare:

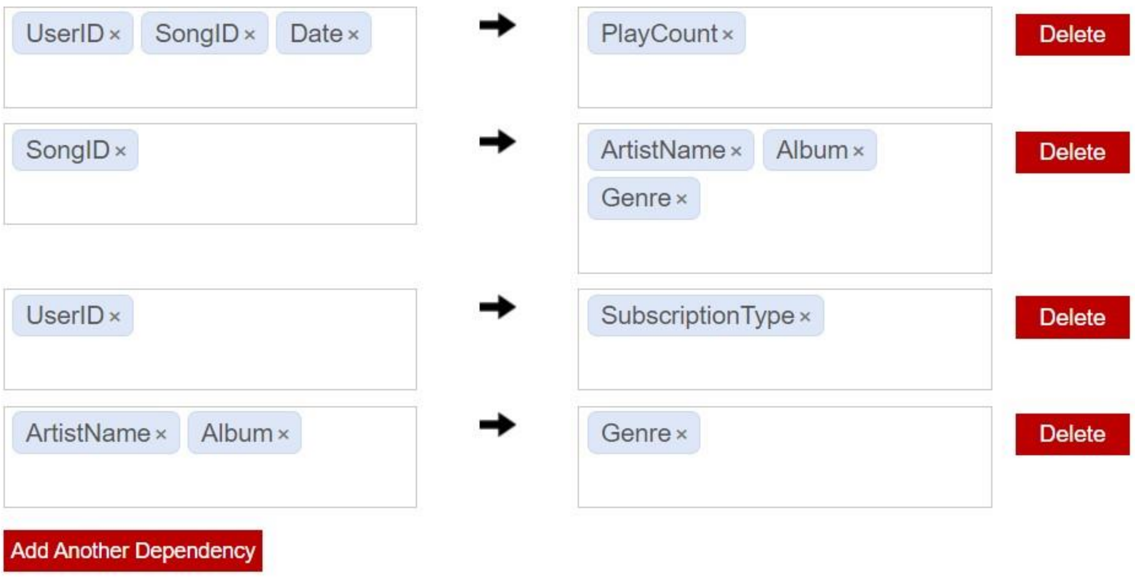
UserID,SongID,Date→PlayCount

SongID→ArtistName,Album,Genre

UserID→SubscriptionType

ArtistName,Album→Genre

PreviousFunctionalDependencies



UserID × SongID × Date × → PlayCount × Delete

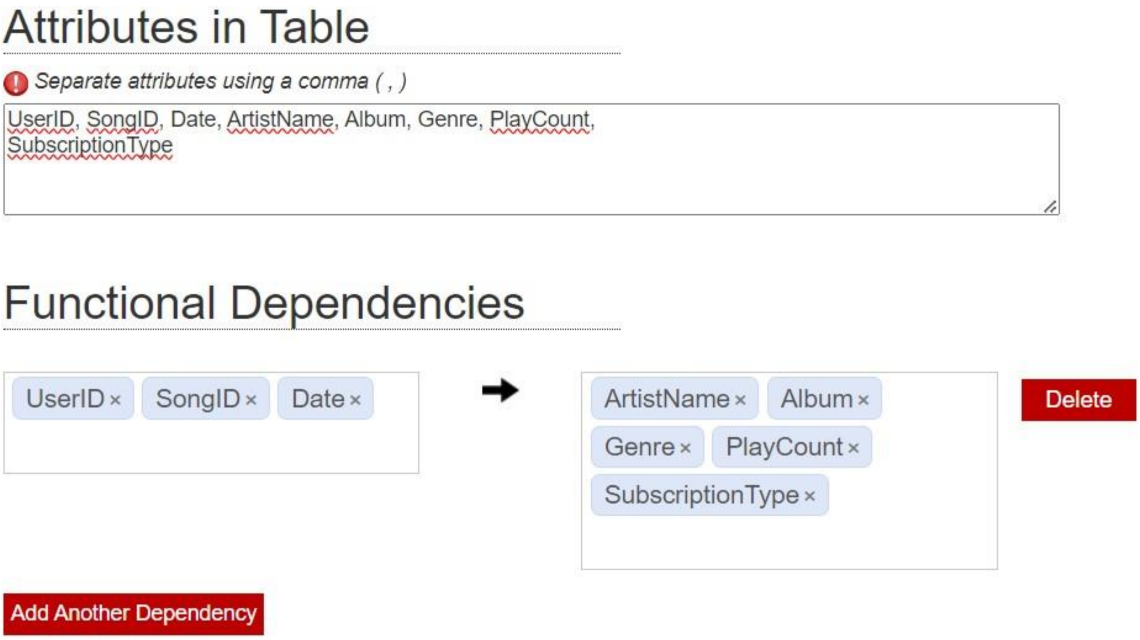
SongID × → ArtistName × Album × Genre × Delete

UserID × → SubscriptionType × Delete

ArtistName × Album × → Genre × Delete

Add Another Dependency

Fig1.G.1 ModifiedDependencies



Attributes in Table

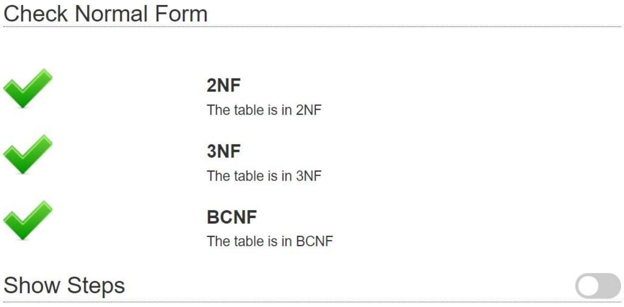
Separate attributes using a comma (,)

UserID, SongID, Date, ArtistName, Album, Genre, PlayCount, SubscriptionType

Functional Dependencies

UserID × SongID × Date × → ArtistName × Album × Genre × PlayCount × SubscriptionType ×

Fig1.G.2 Result



Check Normal Form

✓ **2NF** The table is in 2NF

✓ **3NF** The table is in 3NF

✓ **BCNF** The table is in BCNF

Fig1.G.3

Fig1.G.1showspreviousFunctionalDependencieswhicharenotinBCNF.Fig1.G.2 showsnewFunctionalDependencieswhichshowsthatforagivenuser,listeningto aspecificsongonaparticulardateuniquelydeterminesvariousattributesrelatedto thatlisteningevent,suchashowmanytimesthesongwasplayed(PlayCount),the typeofsubscriptiontheuserhas(SubscriptionType),thenameoftheartist,the album,andthegenreofthesong.Fig1.G.3showstheresultthatnewFDsarein BCNF.

H.RealEstateSystem:

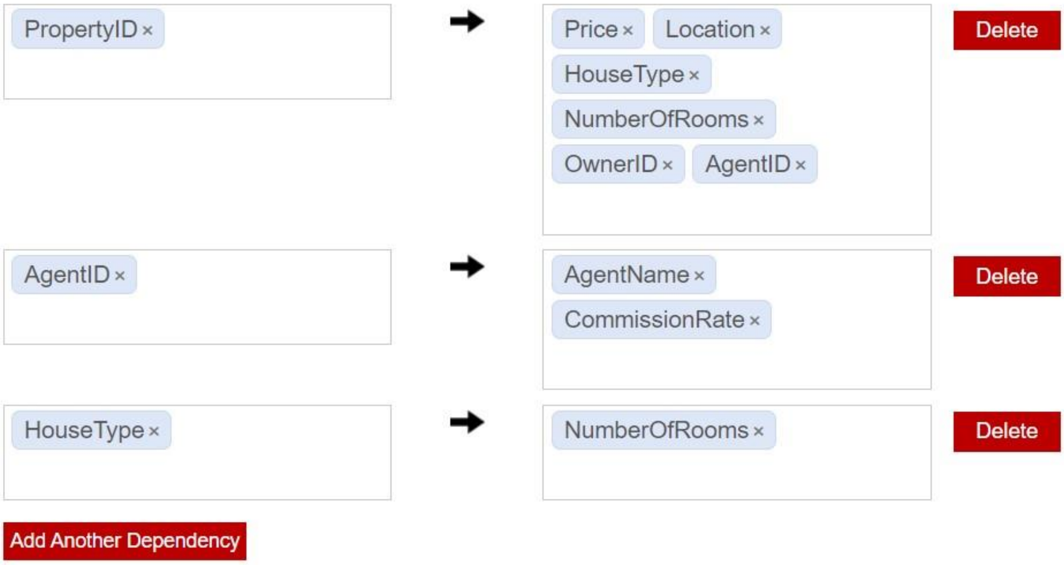
-Fortherelation:

PropertyListings(PropertyID,OwnerID,AgentID,Price,Location,HouseType, NumberOfRooms,AgentName,CommissionRate) -Functionaldependenciesare:

PropertyID→Price,Location,HouseType,NumberOfRooms,OwnerID,AgentID

AgentID→AgentName,CommissionRate HouseType→NumberOfRooms

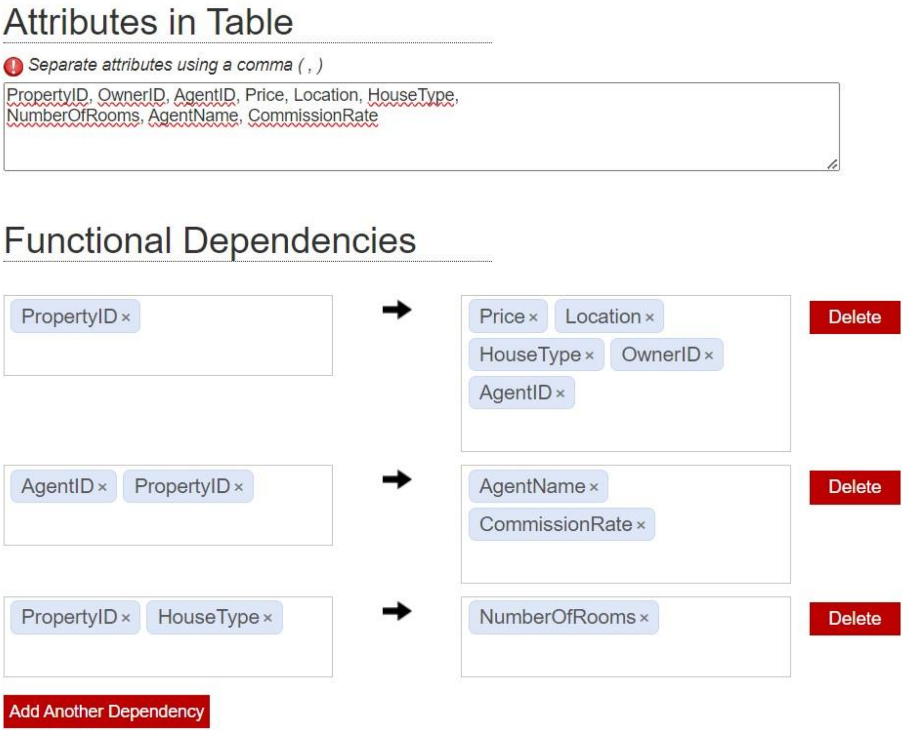
PreviousFunctionalDependencies


PropertyID × → Price × Location × HouseType × NumberOfRooms × OwnerID × AgentID ×

AgentID × → AgentName × CommissionRate ×

HouseType × → NumberOfRooms ×

Fig1.H.1 ModifiedDependencies


Attributes in Table

Separate attributes using a comma (,)

PropertyID, OwnerID, AgentID, Price, Location, HouseType, NumberOfRooms, AgentName, CommissionRate

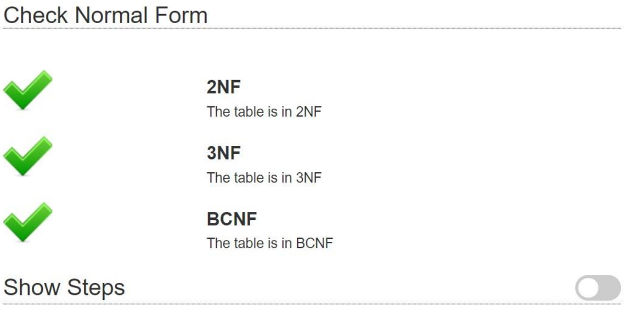
Functional Dependencies

PropertyID × → Price × Location × HouseType × OwnerID × AgentID ×

AgentID × PropertyID × → AgentName × CommissionRate ×

PropertyID × HouseType × → NumberOfRooms ×

Fig1.H.2 Result


Check Normal Form

✓ **2NF**
The table is in 2NF

✓ **3NF**
The table is in 3NF

✓ **BCNF**
The table is in BCNF

Fig1.H.3

Fig1.H.1showspreviousFunctionalDependencieswhic

harenotinBCNF.Fig1.H.2 showsnewFunctionalDependencieswhichshowsthat eachpropertyinthetableis uniquelyidentifiedbyitsPropertyID,andforeachPropertyID,thereisafixedprice, location,housetype,ownerID,andagentIDassociated withit.
ItmeansthateachagentassignedtoaspecificpropertyisuniquelyidentifiedbytheirAgentID,andforeachcombinationofAgentIDandPropertyID,thereisafixednamefortheagentandafixedcommissionrateassociatedwiththatagent's involvementinthatpropertytransaction.
ItmeansthatthenumberofroomsinapropertyisuniquelydeterminedbythecombinationofitsPropertyIDandHouseType.Fig1.H.3showstheresultthatnew FDsareinBCNF.

Que2DesignaBCNFNormalizedDatabaseandverifyusingGriffithTool.

AnsDatabaseisFlightReservationSystem.

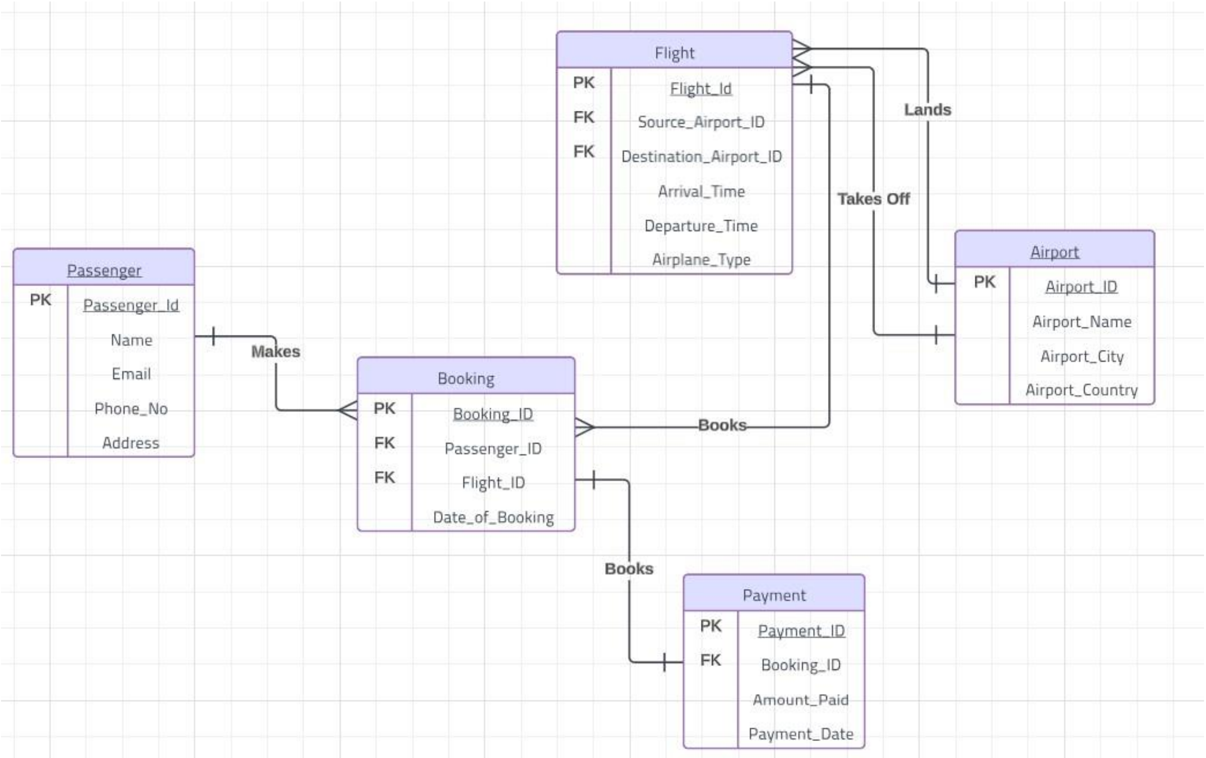


Fig2.1

Fig2.1showsthedesignofairlinereservationsystemdatabase.

FunctionalDependenciesare:

FlightsTable:

- Flight_ID->Source_Airport_ID
- Flight_ID->Destination_Airport_ID
- Flight_ID->Departure_Time
- Flight_ID->Arrival_Time
- Flight_ID->Airplane_Type AirportsTable: Airport_Code->Airport_Name

Airport_Code->Airport_City

Airport_Code->Airport_Country PassengerTable:

- Customer_ID->Name
- Customer_ID->Email
- Customer_ID->Phone_No
- Customer_ID->Address BookingsTable:

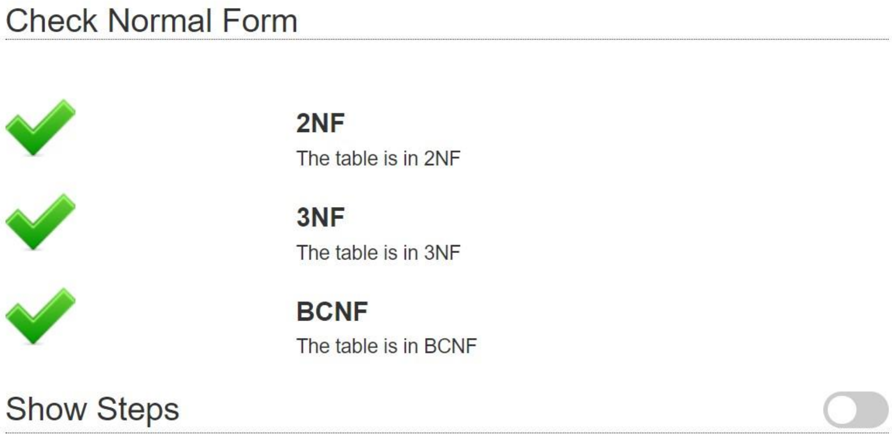
Booking_ID->Flight_ID

Booking_ID->Passenger_ID

Booking_ID->Date_of_Booking PaymentsTable:

- Payment_ID->Booking_ID
- Payment_ID->Amount_Paid
- Payment_ID->Payment_Date

VerificationUsingGriffithTool



Check Normal Form

✓ **2NF**
The table is in 2NF

✓ **3NF**
The table is in 3NF

✓ **BCNF**
The table is in BCNF

Show Steps

Fig2.2

Result

Fig2.2showsthatEachTableisinBCNF.

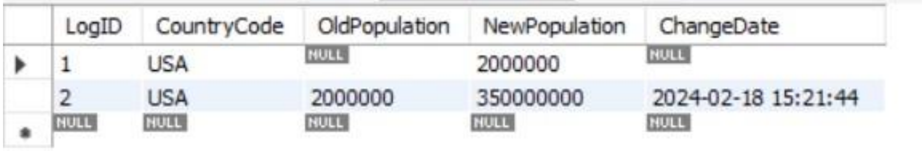
Practical-3

Aim:-CreateProcedures,TriggersandCursors

Que1WriteastoredprocedurenamedUpdateCountryPopulationthat updatesthepopulationofagivencountrybasedonaprovidedcountry codeandnewpopulationvalue.Additionally,theprocedureshouldlog theoldandnewpopulationvaluestoapopulation_change_logtable.

Ans

```
DELIMITER//
CREATEPROCEDUREUpdateCountryPopulation(INCountryCodeCHAR(3),IN
NewPopulationINT)
BEGIN
   DECLAREOldPopulationINT;

   --Gettheoldpopulation
   SELECTPopulationINTOOldPopulation
   FROMcountry
   WHERECode=CountryCode;

   --Updatethepopulation
   UPDATEcountry
   SETPopulation=NewPopulation WHERECode=CountryCode;

   --Logthepopulationchange
   INSERTINTOpopulation_change_log(CountryCode,OldPopulation,
NewPopulation,ChangeDate)
   VALUES(CountryCode,OldPopulation,NewPopulation,NOW());--NOW()isused
forthecurrenttimestampinMySQL
END//
DELIMITER;

CALLUpdateCountryPopulation('USA',350000000);
```



	LogID	CountryCode	OldPopulation	NewPopulation	ChangeDate
▶	1	USA	NULL	2000000	NULL
	2	USA	2000000	350000000	2024-02-18 15:21:44
*	NULL	NULL	NULL	NULL	NULL

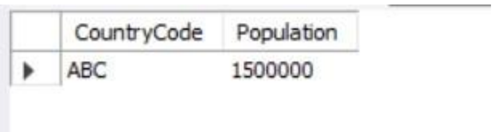
Fig3.1

Fig3.1showspopulation_change_logtablewhichhasoldpopulation,new populationanddateofchange.

Que2Developatriggernamedafter_country_insertthatchecksifthe insertedcountry'spopulationexceeds1million.Ifitdoes,inserta recordintoahigh_population_countriestable.

Ans

```
CREATETRIGGERafter_country_insert
AFTERINSERTONcountry
FOREACHROW
BEGIN
   DECLARECountryPopulationINT;

   --Getthepopulationoftheinsertedcountry
   SELECTPopulationINTOCountryPopulation
   FROMcountry
   WHERECode=NEW.Code;

   --Checkifpopulationexceeds1million
   IFCountryPopulation>1000000THEN
      --Insertintohigh_population_countriestable
      INSERTINTOhigh_population_countries(CountryCode,Population)
      VALUES(NEW.Code,CountryPopulation);
ENDIF; END//
DELIMITER;

INSERTINTOcountry(Code,Population)VALUES('ABC',1500000);

select*fromhigh_population_countries;
```



	CountryCode	Population
▶	ABC	1500000

Fig3.2

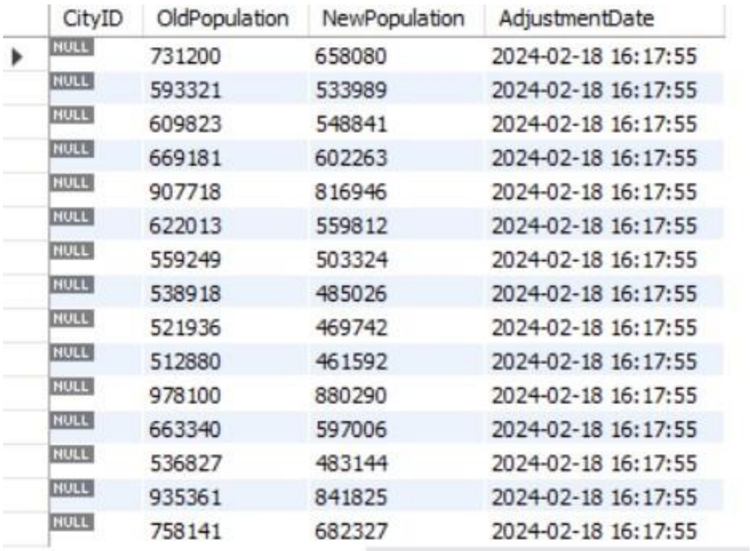
Fig3.2showshigh_population_countriestablewithcountrycodeandpopulation.

Que3DevelopaprocedureAdjustCityPopulationsusingacursorthat decreasesthepopulationby10%forallcitiesinagivencountrycode, providedthecurrentpopulationisbetween500,000and1million. Additionally,logthesechangestoacity_population_adjustmentstable withcityID,oldpopulation,andnewpopulation.

Ans

```
      DELIMITER//
                 CREATEPROCEDUREAdjustCityPopulations(INCountryCodeCHAR(3))
      BEGIN
         DECLAREdoneINTDEFAULTFALSE;
         DECLARECityIDINT;
         DECLAREOldPopulationINT;
         DECLARENewPopulationINT;
         --Declarecursor
         DECLAREcity_cursorCURSORFOR
```

```
    SELECTCityID,Population
    FROMcity
    WHERECountryCode=CountryCode
    ANDPopulationBETWEEN500000AND1000000;

    --Declarehandlerfornomorerows
    DECLARECONTINUEHANDLERFORNOTFOUNDSETdone=TRUE;

    --Openthecursor
    OPENcity_cursor;

    --Startloopingthroughthecursor
    adjust_loop:LOOP --Fetchtherow
        FETCHcity_cursorINTOCityID,OldPopulation;

        --Checkifnomorerows IFdoneTHEN
            LEAVEadjust_loop;
        ENDIF;

        --Calculatenewpopulation(decreaseby10%)
        SETNewPopulation=ROUND(OldPopulation*0.9,0);

        --Updatecitypopulation
        UPDATEcity
        SETPopulation=NewPopulation WHERECityID=CityID;

        --Logpopulationadjustment
        INSERTINTOcity_population_adjustment(CityID,OldPopulation,
NewPopulation,AdjustmentDate)
        VALUES(CityID,OldPopulation,NewPopulation,NOW()); ENDLOOPadjust_loop;

    --Closethecursor
    CLOSEcity_cursor;
END//
DELIMITER;
CALLAdjustCityPopulations('USA'); select*fromcity_population_adjustment;
```



CityID	OldPopulation	NewPopulation	AdjustmentDate
NULL	731200	658080	2024-02-18 16:17:55
NULL	593321	533989	2024-02-18 16:17:55
NULL	609823	548841	2024-02-18 16:17:55
NULL	669181	602263	2024-02-18 16:17:55
NULL	907718	816946	2024-02-18 16:17:55
NULL	622013	559812	2024-02-18 16:17:55
NULL	559249	503324	2024-02-18 16:17:55
NULL	538918	485026	2024-02-18 16:17:55
NULL	521936	469742	2024-02-18 16:17:55
NULL	512880	461592	2024-02-18 16:17:55
NULL	978100	880290	2024-02-18 16:17:55
NULL	663340	597006	2024-02-18 16:17:55
NULL	536827	483144	2024-02-18 16:17:55
NULL	935361	841825	2024-02-18 16:17:55
NULL	758141	682327	2024-02-18 16:17:55

Fig3.3

Fig3.3showscity_population_adjustmenttablewhichrecordthepopulation statisticsanddateofchange.

Practical-4

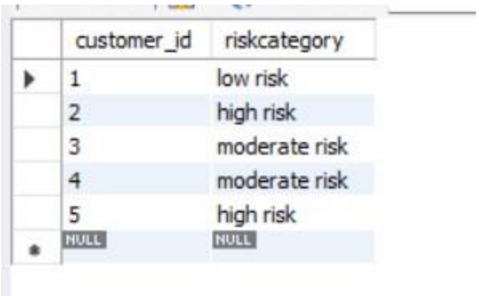
Aim:-Writeprogramstoimplementandunderstandusageof Datamarts.

Question1:Designadatamartforabanktostorethecredithistoryof customersinabank.Usethiscreditprofilingtoprocessfutureloan applications.(Suggestivetables:CustomerProfile,accounts,loans, creditcards,paymenthistorytable,inquiries,Collections,CreditScore History).

Ans

```
createdatabasebank;

createtablecustomer_profile(customer_idintprimarykey,first_name
varchar(25),last_namevarchar(25),d_o_bdate,addressvarchar(50),phone_no
int,emailvarchar(25),incomeint);

createtableaccounts(account_idintprimarykey,customer_idint,accounttype
varchar(25),dateofopendate,accountstatusvarchar(25),foreignkey(customer_id)
referencescustomer_profile(customer_id),balanceint);

createtableloans(loan_idintprimarykey,customer_idint,loantype
varchar(25),loanamountint,termint,interest_ratedecimal(4,2),loanstatus
varchar(25),foreignkey(customer_id)referencescustomer_profile(customer_id));

createtablecreditcards(card_idintprimarykey,customer_idint,cardtype
varchar(25),creditlimitdecimal(10,2),cardissuedatedate,foreignkey(customer_id)
referencescustomer_profile(customer_id),currentbalancedecimal(10,2));

createtablepaymenthistory(payment_idintprimarykey,customer_idint,account_id
int,paymentamountdecimal(10,2),paymentdatedate,foreignkey(customer_id)
referencescustomer_profile(customer_id),foreignkey(account_id)references
accounts(account_id));

createtableinquiries(inquiry_idintprimarykey,customer_idint,inquirydate
date,inquirytypevarchar(25),foreignkey(customer_id)references
customer_profile(customer_id));

createtablecollections(collection_idintprimarykey,customer_idint,collectiondate
date,collectiontypevarchar(25),amountint,foreignkey(customer_id)references
customer_profile(customer_id));

createtablecredit_score_history(creditscore_idintprimarykey,customer_id
int,creditscoreint,scoredatedate,foreignkey(customer_id)references
customer_profile(customer_id));
--DATAMART:
```

```
createtablecustomerrisk(customer_idintprimarykey,riskcategoryvarchar(25));
insertintocustomerrisk(customer_id,riskcategory)selectc.customer_id,case
whenc.income>75000andsum(a.balance)>100000then'lowrisk'
whenc.income>50000andsum(a.balance)>60000then'moderaterisk' else'highrisk'
endasriskcategory
fromcustomer_profilecjoinaccountsaonc.customer_id=a.customer_idgroupby c.customer_id;
```

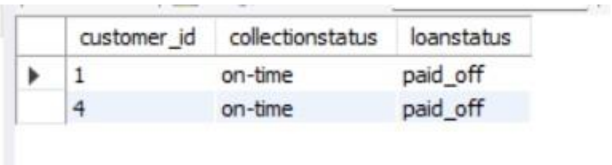


customer_id	riskcategory
1	low risk
2	high risk
3	moderate risk
4	moderate risk
5	high risk
NULL	NULL

Fig4.1

InFig4.1,itshowsthatitdividesthecustomersintodifferentriskcategorybaseon incomeandbalanceofcustomers.

```
createtableloanassessmentasselectc.customer_idas
customer_id,c.collectionstatusascollectionstatus,l.loanstatusasloanstatusfrom
collectionscjoinloanslonc.customer_id=l.customer_idwherecollectionstatus='ontime'andloanst
atus='paid_off';
```

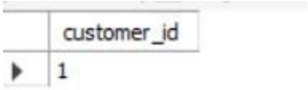


customer_id	collectionstatus	loanstatus
1	on-time	paid_off
4	on-time	paid_off

Fig4.2

InFig4.2itshowstheresultofcustomerswhoseloanstatusispaidoffand collectionstatusisontime.

```
createtableloanpassasselectl.customer_idfromloanassessmentljoin
customerriskconl.customer_id=c.customer_idjoincredit_score_historychon
ch.customer_id=c.customer_idwherec.riskcategory='lowrisk'and ch.creditscore>750;
```



customer_id
1

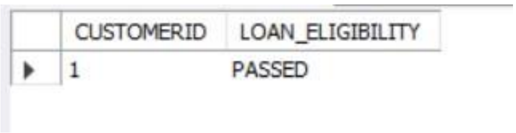
Fig4.3

InFig4.3itshowsthecustomerswhichhaslowriskcategoryhasloanstatusas paidoffandontimeandcreditscoregreaterthan750.

```
CREATEPROCEDURELOAN_PASS_RESULT(INCUSTOMERIDINT) BEGIN
DECLAREMESSAGE_TEXTVARCHAR(50);
IFEXISTS(
        SELECT1FROMloanpass
        WHEREcustomer_id=CUSTOMERID
```

```
  )THEN
      SELECTCUSTOMERID,'PASSED'ASLOAN_ELIGIBILITY;
ELSE
      SELECTCUSTOMERID,'REJECTED'ASLOAN_ELIGIBILITY;
ENDIF;
END//
DELIMITER; callLOAN_PASS_RESULT(1);
```

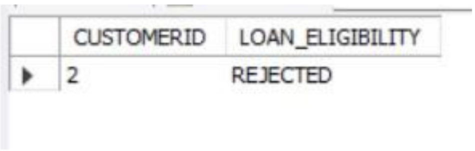
Output1



	CUSTOMERID	LOAN_ELIGIBILITY
▶	1	PASSED

Fig4.4

callLOAN_PASS_RESULT(2); Output2



	CUSTOMERID	LOAN_ELIGIBILITY
▶	2	REJECTED

Fig4.5

RESULT:SuccessfullyimplementedandlearnttheusageofDatamarts.