CBYREGOWDAINSTITUTEOF TECHNOLOGY

DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING

AffiliatedtoVisvesvarayaTechnologicalUniversity "Jnana Sangama", Belgaum – 560 018.

LABORATORYMANUAL

"DBMSLABORATORYWITHMINIPROJECT–18CSL58"

Semester:V Scheme:CBCS

PreparedBy

Dr.DeepikaLokesh

Prof.vanitha
Prof.kirthika

Scrutinizedby:
VASUDEVAR
AssistantProfessor

CBYREGOWDAINSTITUTEOFTECHNOLOGY
DepartmentofComputerScienceandEngineering An ISO 9001:2015 Certified Institute
Kolar–SrinivaspurRoad, Kolar – 563101
2023-24

DBMSLABORATORYWITHMINIPROJECT
[AsperChoiceBasedCreditSystem(CBCS)scheme]
(Effective from the academic year 2018 -2019)
SEMESTER–V

SubjectCode:**18CSL58** IAMarks:**40**
NumberofContactHours/Week:**0:2:2** ExamMarks:**60**
TotalNumberofLabContactHours:**36** ExamHours:**03**

CREDITS–02

Courseobjectives: Thiscoursewillenablestudentsto

- Foundationknowledgeindatabaseconcepts,technologyandpractice togroomstudents intowell-informed databaseapplication developers.
- StrongpracticeinSQLprogrammingthroughavariety ofdatabaseproblems.
- Developdatabaseapplicationsusingfront-endtools andback-endDBMS.

Description(Ifany):

PART-A:SQLProgramming(Max.ExamMks.50)

- Design, develop, and implement the specified queries for the following problems using Oracle, MySQL, MS SQL Server, or anyother DBMS underLINUX/Windowsenvironment.
- Create Schema andinsertat least5 records for each table.Add appropriate database constraints.

PART-B:MiniProject(Max.ExamMks.30)

- Use Java, C#, PHP, Python, or any other similar front-end tool. All applications mustbedemonstratedondesktop/laptopasastand-aloneorwebbasedapplication (Mobile apps on Android/IOS are not permitted.)
- Installationprocedureoftherequiredsoftwaremustbedemonstrated,carriedoutin groups and documented in the journal.

Lab Experiments:PartA:SQLP rogramming

1. ConsiderthefollowingschemaforaLibraryDatabase:
BOOK (Book_id, Title, Publisher_Name, Pub_Year)
BOOK_AUTHORS (Book_id, Author_Name)
PUBLISHER (Name, Address, Phone)
BOOK_COPIES (Book_id, Programme_id, No-of_Copies)BOOK_LENDING(Book_id,Programme_id,Card_No,Date_Out,Due_Date)
LIBRARY_PROGRAMME(Programme_id,Programme_Name,Address)

WriteSQLqueriesto

1. Retrieve details ofall books in the library id, title, name of publisher, authors, number of copies in each Programme, etc.
2. Get the particulars of borrowers who haveborrowed more than 3 books, but fromJan 2017 to Jun 2017.
3. Delete a book inBOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. Partition the BOOK table based on yearof publication. Demonstrate its working witha simple query.
5. Create a viewof all books and its number of copies that are currently available inthe Library.

2. Consider the following schema for Order Database:
SALESMAN(Salesman_id, Name, City,
Commission)CUSTOMER(Customer_id, Cust_Name, City, Grade,
Salesman_id)ORDERS(Ord_No,Purchase_Amt,Ord_Date,Customer_id,Sa
lesman_id) Write SQL queries to

1. Count the customers with grades above Bangalore's average.
2. Findthenameandnumbersof allsalesmanwhohadmorethanonecustomer.
3. List allthe salesmanand indicatethose whohave and don't have customersintheir cities (Use UNION operation.)
4. Create a viewthat finds the salesman who has the customer with the highest order ofa day.
5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also bedeleted.

3. Consider the schema for Movie Database:
ACTOR(Act_id, Act_Name, Act_Gender)
DIRECTOR(Dir_id,Dir_Name,Dir_Phone)
MOVIES (Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)
MOVIE_CAST(Act_id, Mov_id, Role)
RATING(Mov_id,Rev_Stars)
Write SQL queries to

1. Listthetitlesof allmoviesdirectedby'Hitchcock'.
2. Findthemovienameswhereoneormoreactorsactedintwo ormoremovies.
3. List allactors who acted ina moviebefore 2000 and also ina movieafter 2015 (use JOIN operation).
4. Find the title of movies and number ofstars for each movie that has at leastone rating and findthe highest numberofstars that moviereceived. Sort the result bymovietitle.
5. Updateratingof all movies directed by 'Steven Spielberg' to5.

4. Consider the schema for College Database:
STUDENT(USN, SName, Address,Phone,Gender)
SEMSEC(SSID, Sem, Sec)
CLASS(USN,SSID)
SUBJECT(Subcode,Title,Sem,Credits)
IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)
Write SQL queries to

1. Listallthestudentdetailsstudyinginfourthsemester'C'section.
2. Compute the total number of maleand female students ineach semester and ineach section.
3. CreateaviewofTest1 marks of student USN '1BI15CS101'inall subjects.
4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.
5. Categorize students based on the followingcriterion:
   If FinalIA = 17 to 20 then CAT = 'Outstanding'
   If FinalIA = 12 to 16 then CAT = 'Average'
   If FinalIA< 12 then CAT = 'Weak'
   Givethesedetailsonly for8thsemesterA,B,andCsectionstudent

5. ConsidertheschemaforCompanyDatabase:
   EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN, DNo)
   DEPARTMENT (DNo, DName, MgrSSN,
   MgrStartDate)DLOCATION (DNo, DLoc)
   PROJECT (PNo, PName, PLocation, DNo)
   WORKS_ON(SSN, PNo, Hours)
   WriteSQLqueriesto
   1. Make a list of all project numbers for projects that involve an employee whose last nameis'Scott', either as a worker or as a manager of the department that controls the project.
   2. Showtheresultingsalariesif every employee working on the 'IoT' project isgivena 10percentraise.
   3. Find the sumofthe salaries ofall employees ofthe'Accounts'department, as wellas the maximum salary, the minimum salary, and the average salary in thisdepartment
   4. Retrievethenameofeachemployeewho worksonalltheprojectscontrolledby departmentnumber 5 (use NOT EXISTSoperator).
   5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs.6,00,000.

PartB:Miniproject

- Foranyproblemselected
- Make sure that the application should have five or more tables
- Indicative areas include; health care

Course outcomes: The students should be able to:
- Create, Update and query on the database.
- DemonstratetheworkingofdifferentconceptsofDBMS
- Implement,analyzeandevaluatetheprojectdevelopedforanapplication.

ConductionofPracticalExamination:

Experimentdistribution

1. For laboratories having onlyone part: Students are allowed to pick one experiment from the lot with equal opportunity.
2. ForlaboratorieshavingPARTAandPARTB:Studentsare allowedtopickone experimentfromPART A andone experiment fromPART B, with equal opportunity.
3. Changeofexperimentisallowedonlyonceandmarksallottedforproceduretobemade zero of the changed part only.
4. MarksDistribution(Coursedtochangeinaccordancewithuniversityregulations)

5. Forlaboratorieshavingonlyonepart–Procedure+Execution+Viva-Voce:
   15+70+15=100Marks
6. ForlaboratorieshavingPARTAandPARTB
   i. PartA–Procedure+Execution+Viva=6+28+6=40Marks
   ii. PartB– Procedure+Execution+Viva=9+42+9=60Marks

INTRODUCTION

INTRODUCTIONTODATABASE

Whatis Database?

A database is a separate application that stores a collection of data. Each database has one or moredistinctAPIsforcreating,accessing,managing,searching,andreplicatingthedataitholds. now a days we use relational database management systems (RDBMS) to store and manager huge volume of data.

A**RelationalDataBaseManagementSystem(RDBMS)**isasoftwarethat:

- Enablesyoutoimplementadatabasewithtables,columns,andindexes. ● Guarantees the Referential Integrity between rows of various tables.
- InterpretsanSQLqueryandcombinesinformation fromvarioustables.

RDBMSTerminology:

Database: Adatabaseisacollectionoftables,withrelated data.

Table: Atableisamatrixwithdata.Atableinadatabaselookslikea*simplespreadsheet.*

Column: Onecolumn(dataelement)containsdataofoneandthesamekind,forexamplethecolumn postcode. or phone numbers

Row: Arow(=tuple, entryorrecord) isagroupofrelated data, for examplethedataofone subscription.

Redundancy: Storingdatatwice,redundantlytomakethesystemfaster.

Primary Key: Aprimarykeyisunique. Akeyvaluecan notoccurtwice inonetable. Withakeyyou can find at most one row.

ForeignKey: Aforeignkeyisthelinkingpinbetweentwotables.

CompoundKey: Acompoundkey(compositekey) isakeythat consistsofmultiplecolumns,because one column is not sufficiently unique.

Index: Anindex inadatabaseresemblesanindexatthebackofabook.

ReferentialIntegrity: ReferentialIntegritymakes surethataforeignkeyvaluealwayspointstoan existing row

DDLorDataDefinitionLanguageactuallyconsistsoftheSQLcommandsthatcanbeusedto tocreateandmodifythestructureofdatabaseobjectsinadatabase.Thesedatabaseobjectsinclude views, schemas, tables, indexes, etc.

Someexamples:

- CREATE - to create objects in the database
- ALTER-altersthestructureofthedatabase
- DROP - delete objects from the database

DMLisDataManipulationLanguagestatements:whichareusedtointeractwithadatabaseby deleting, inserting, retrieving, or updating data in the database.

Someexamples:

- SELECT-retrievedatafromtheadatabase
- INSERT - insert data into a table
- UPDATE -updatesexistingdatawithinatable
- DELETE-deletes allrecordsfromatable,thespacefortherecords remain

DCL is Data Control Language statements: which includes commands such as GRANT and REVOKEwhichmainlydealswiththerights,permissionsandothercontrolsofthedatabasesystem.

Someexamples:

- **GRANT**-gives user's access privileges to database.
- **REVOKE**-withdraw user's access privileges given by usingtheGRANT command.

TCLisTransactionControlLanguagewhichdealswithatransactionwithinadatabase. Some

examples:

- COMMIT-savework done
- SAVEPOINT-identifyapoint inatransactiontowhichyoucanlaterrollback
- ROLLBACK - restore database to original since the last COMMIT
- SETTRANSACTION-Changetransactionoptionslikewhatrollbacksegmenttouse.

SQLDataTypes

Eachcolumn inadatabasetableisrequired tohaveanameandadatatype.

An SQL developer must decide what type of data that will be stored inside each column when creatingatable.Thedatatype isaguideline forSQLto understandwhattypeofdataisexpected inside of each column, and it also identifies how SQL will interact with the stored data.

MySQLusesmanydifferentdatatypesbrokenintothreecategories Numeric

- DateandTime
- String Type
-

DATATYPES

NUMERIC:

- **INT**–Anormal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to4294967295. You can specify a width of up to 11 digits.

- **TINYINT** – A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to127.If unsigned,the allowable range isfrom 0 to 255.You can specify a width of up to 4 digits.

- **SMALLINT** – A small integer that can besigned or unsigned. If signed, the allowable range is from -32768to32767.If unsigned,theallowablerangeisfrom 0to65535.Youcanspecifya width of up to 5 digits.

- **MEDIUMINT**A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.

- **BIGINT** – A large integer that can besigned orunsigned. Ifsigned, the allowable rangeis from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.

- **FLOAT (M, D)**A floating-point number that cannot be unsigned. You can define the display length (M)and thenumberof decimals (D). This is not required and will default to 10, 2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

- **DOUBLE(M, D)**A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This isnot required and will default to 16, 4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

- **DECIMAL (M, D)**–an unpacked floating-point number that cannot beunsigned.Inthe unpacked decimals, each decimal corresponds to one byte. Defining the displaylength (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

DATEANDTIMETYPES

The MySQL date and time data types are as follows –

- **DATE**– A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.

- **DATETIME** – A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00and 9999-12-31 23:59:59. Forexample, 3:30 inthe afternoononDecember 30th,1973 wouldbestoredas1973-12-3015:30:00.

- **TIMESTAMP**– A timestamp between midnight, January,1st1970andsometimein2037.This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).

- **TIME** – Stores the time in a HH:MM:SS format.

- **YEAR (M)** –Storesayearina2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR (2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

STRINGTYPES

Thislistdescribesthecommonstringdatatypes inMySQL.

- **CHAR (M)** –Afixed-length string between 1 and 255 characters in length (for example CHAR (5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.

- **VARCHAR(M)** –Avariable-length string between 1 and 255 characters in length. For example, VARCHAR (25). You must define a length when creating a VARCHAR field.

- **BLOBorTEXT** –Afield with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two isthatthesortsandcomparisonsonthestoreddataare**case sensitive** onBLOBsandare**not case sensitive** in TEXT fields. You do not specify a length with BLOB or TEXT.

- **TINYBLOBorTINYTEXT**–ABLOBorTEXTcolumnwithamaximumlengthof255 characters.YoudonotspecifyalengthwithTINYBLOBorTINYTEXT.

- **MEDIUMBLOBor MEDIUMTEXT** – A BLOB or TEXT column with a maximum length of 16777215characters.Youdonotspecifyalength withMEDIUMBLOBorMEDIUMTEXT.

- **LONGBLOBorLONGTEXT** –ABLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.

CREATETABLE

Specifiesanewbaserelationbygivingit aname, andspecifyingeachofitsattributesand their data types

Syntaxof CREATECommand:

CREATE TABLE<table name>

(<AttributeA1><DataTypeD1>[<Constraints>],

<AttributeA2><DataTypeD2>[<Constraints>],

…….

<Attribute An><Data Type Dn> [<Constraints>],

[<integrity-constraint1>,<integrity-constraintk>]);

Specifyingtheunique, primarykeyattributes, secondarykeys,andreferentialintegrityconstraints

EXAMPLEOFCREATING TABLE

CREATETABLEORDERS

(

ORDER_IDINT(6)PRIMARYKEY,

ORDER_DATE DATE

);

ALTERTABLESTATEMENT

Onceatableiscreatedinthe database, therearemanyoccasionswhereone maywishto changethe structure of the table. Typical cases include the following:

- Add a column
- Dropacolumn
- Changeacolumnname
- Changethedatatypeforacolumn
- addanddropvariousconstraintsonanexistingtable.includingprimarykeyandforeignkey The

SQL syntax for **ALTER TABLE** is

ALTERTABLE"table_name"[alterspecification]

[alterspecification] isdependentonthetypeofalterationwewishtoperform. alterspecification is already mentioned above

ADDINGCOLUMNIN TABLE

Toaddacolumn inatable,usethefollowingsyntax:

ALTERTABLETABLENAMEADDCOLUMN_NAMEDATETYPE;

Example: ALTER TABLE ORDERS ADD JOB VARCHAR (20);

MODIFYING DATATYPE FOR COLUMN IN TABLE

Tomodifyacolumndatatypeinatable,usethefollowing syntax:

ALTERTABLEORDERSMODIFYCOLUMN_NAMEVARCHAR(50);

Example:ALTERTABLEORDERSMODIFYJOBVARCHAR(50);

RENAMINGCOLUMNINTABLE

YoucanrenameacolumninMySQLusingtheALTERTABLEandCHANGEcommands together to change an existing column.

Forexample,saythecolumniscurrentlynamedJOB,but youdecidethatDESIGNATIONisa more appropriate title. The column is located on the table entitled ORDERS.

Hereisanexampleofhowtochangeit:

ALTERTABLETABLENAMECHANGEOLDNAMENEWNAMEVARCHAR(20);

Example:ALTERTABLEORDERSCHANGEJOBDESIGNATIONVARCHAR(20);

DELETINGCOLUMN

Todeleteacolumninatable, usethefollowingsyntax:

ALTERTABLETABLENAMEDROPCOLUMNCOLUMN_NAME

Example:ALTERTABLEORDERSDROPCOLUMN DESIGNATION;

RENAMINGATABLE

Torenameatable, usethefollowingsyntax:

RENAMETABLEOLDTABLENAMETONEWTABLENAME;

Example:RENAMETABLEORDERSTOORDERS_TBL

DROPTABLE

It isveryeasytodropanexistingMySQLtable, but youneedtobeverycarefulwhiledeleting anyexisting table because the data lost will not be recovered after deleting a table.

TheDROPTABLEstatement isusedtodropanexistingtableinadatabase DROP

TABLE TABLENAME;

Example:DROPTABLEORDER

TRUNCATETABLESTATEMENT

ifwewish tosimply getridofthedatabutnotthetableitself? Forthis,wecan usethe

TRUNCATETABLE command.

ThesyntaxforTRUNCATETABLEis

TRUNCATE "table_name"

So, ifwewantedtotruncatethetablecalledcustomer that wecreatedinMYSQL, wesimply type,

Example:TRUNCATEcustomer

CONSTRAINTS:

Commontypesofconstraints includethefollowing:

PrimaryKey:-

- A primarykeyis used to uniquelyidentifyeach row in a table. It can either be part of the actual record itself, or it can be an artificial field (one that has nothing to do with the actual record).
- Aprimarykeycan consist ofone or more fields on atable. When multiple fields areused as a primary key, they are called a composite key.
- Primary keys can be specified either when the table is created (using CREATE TABLE) or by changing the existing table structure (using ALTER TABLE).

Belowareexamplesforspecifyingaprimarykeywhencreatingatable:

Example:

CREATETABLEORDERS(ORDER_IDINT(6)PRIMARYKEY,ORDER_DATEDATE);

Below are examples for specifying a primary key by altering a table:

CREATETABLEORDERS(ORDER_IDINT(6),ORDER_DATEDATE);

Example:ALTERTABLEORDERSADDPRIMARYKEY(ORDER_ID);

Note:-BeforeusingtheALTERTABLEcommandtoaddaprimarykey, you'llneedtomakesure that the field is defined as 'NOT NULL' -- in other words, NULL cannot be an accepted value for that field.and column values must be unique

ALTERTABLETABLENAMEDROPPRIMARYKEYCONSTRAINT

TodropaPRIMARYKEYconstraintinTableORDERS,usethefollowing MYSQL:syntax

Example:ALTERTABLEORDERSDROPPRIMARYKEY

FOREIGNKEY

- Aforeignkeyis afield(orfields)that pointstotheprimarykeyofanother table.
- The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted
- For example, say we have two tables,a CUSTOMER table that includes all customer data, and an ORDERS table that includes all customer orders. The constraint here is that allorders must be associated with a customer that is already in the CUSTOMER table.
- In this case, we will place a foreign key on the ORDERS table and have it relate to the primary key of the CUSTOMER table. This way, we can ensure that all orders in the ORDERS table are related to a customer in the CUSTOMER table. .In other words, the ORDERS table cannot contain information on a customer that is not in the CUSTOMER table.

Thestructureofthesetwotableswillbeasfollows: Table CUSTOMER

column name	Characteristic
SID	Primary Key
Last_Name	varchar(50)
First_Name	varchar(50)

TableORDERS

column name	characteristic
Order_ID	Primary Key
Order_Date	Date
Customer_SID	ForeignKey
Amount	Decimal(10,2)

```
CREATETABLECUSTOMER (
SIDINTPRIMARYKEY,
Last_Namevarchar(50),
First_Namevarchar(50)
);
```

Inthebelowexample,theCustomer_SIDcolumntheORDERStableisaforeignkeypointing to the SID column which is primarykey in the CUSTOMER table.

Belowweshowexamplesofhowtospecifytheforeignkeywhencreating theORDERStable: CREATE TABLE ORDERS

```
(
Order_ID int,
Order_Date date,
Customer_SIDint,
Amount double,
PrimaryKey(Order_ID),
ForeignKey(Customer_SID)referencesCUSTOMER(SID)
);
```

Belowareexamples forspecifyingaforeignkeybyalteringatable. ThisassumesthattheORDERStablehasbeencreated,andtheforeignkeyhasnotyetbeenput in

```
ALTERTABLEORDERSADDFOREIGNKEY(customer_sid)REFERENCES
CUSTOMER(SID);
```

Wecandropaforeignkeybyusingbelow syntax

```
ALTERTABLEORDERSDROPFOREIGNKEYFOREIGNKEY_CONSTRAINT_NAME;
```

NOTNULL Constraint:- By default, a column can hold NULL. If you don't want to allow or storeNULLvalue inacolumn, youwillwantto placeaconstraintonthiscolumnspecifyingthat NULL is now not an allowable value.

DEFAULTConstraint:-TheDEFAULTconstraint providesadefault valuetoacolumnwhen the INSERT INTO statement does not provide a specific value.

UNIQUEConstraint:-TheUNIQUEconstraintensuresthat allvaluesinacolumnaredistinct.

CHECKConstraint:-The CHECK constraint ensures that all values in a column satisfycertain conditions.Oncedefined,thedatabasewillonlyinsert anewroworupdateanexistingrowifthe new valuesatisfiesthe CHECKconstraint.TheCHECKconstraint isused to ensure dataquality

BASICQUERIESIN SQL

- SQLhasonebasicstatementforretrievinginformationfromadatabase;theSLELECT statement
- Thisis*notthesameas*theSELECToperationoftherelationalalgebra •

  Important distinction between SQL and the formal relational model;
- SQLallowsatable(relation) to havetwoormoretuplesthatareidentical inalltheir attribute values
- Hence,anSQLrelation(table) is a*multi-set*(sometimescalledabag)oftuples; itis*not*aset oftuples
- SQLrelationscan beconstrainedto besetsbyusingtheCREATEUNIQUE INDEX command, or by using the DISTINCT option
- BasicformoftheSQLSELECTstatementiscalleda*mapping*ofa*SELECT-FROM-WHERE block*

  SELECT<attributelist>FROM<tablelist>WHERE<condition>
- <attributelist> isalistofattributenameswhosevaluesareto beretrievedbythequery •

  <table list > isa list ofthe relation names required to process thequery
- <condition>isaconditional(Boolean)expressionthatidentifiesthetuples toberetrievedby the query

SIMPLESQLQUERIES

BasicSQLqueriescorrespondtousingthefollowingoperationsoftherelationalalgebra:

SELECTPROJECTJOIN

AllsubsequentexamplesusesCOMPANYdatabaseasshownbelow:

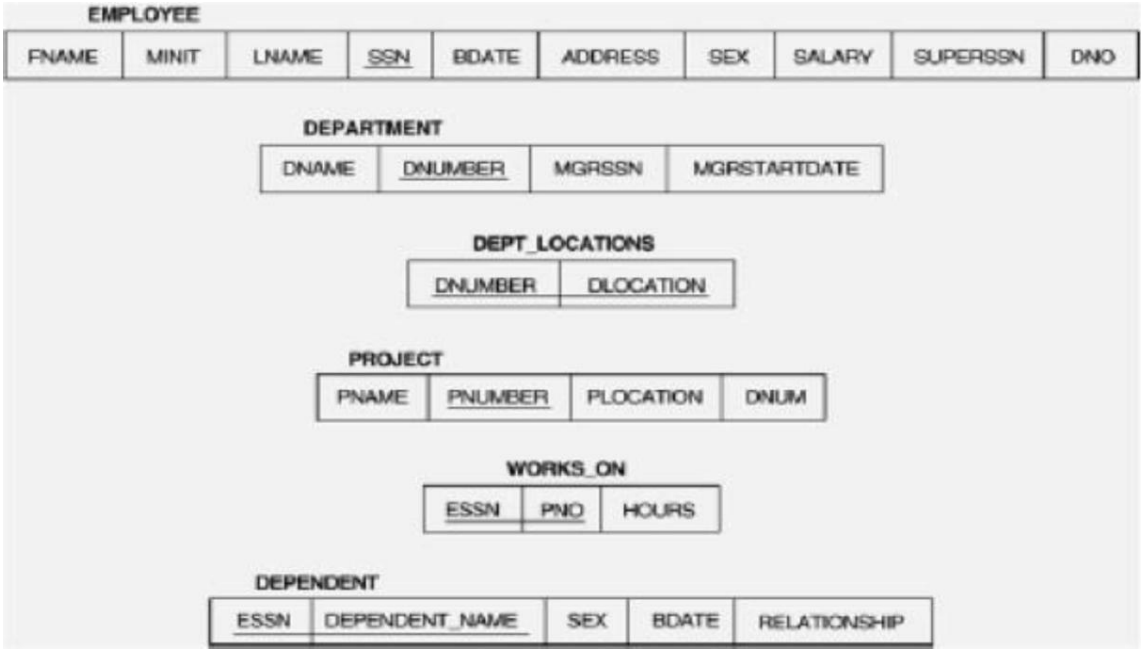
Exampleofasimplequeryononerelation

Query0: Retrievethebirthdateandaddressoftheemployeewhosenameis'JohnB. Smith'.

Q0:SELECTBDATE,ADDRESSFROMEMPLOYEE

WHERE FNAME='John' AND MINIT='B' AND LNAME='Smith'

Similar to a SELECT-PROJECT pair of relational algebra operations: TheSELECT-clause specifies the projection attributes and the WHERE-clause specifies the selectioncondition However, the result of the query may contain duplicate tuples.


EMPLOYEE
FNAME
MINIT
LNAME
SSN
BDATE
ADDRESS
SEX
SALARY
SUPERSSN
DNO
DEPARTMENT
DNAME
DNUMBER
MGRSSN
MGRSTARTDATE
DEPT_LOCATIONS
DNUMBER
DLOCATION
PROJECT
PNAME
PNUMBER
PLOCATION
DNUM
WORKS_ON
ESSN
PNO
HOURS
DEPENDENT
ESSN
DEPENDENT_NAME
SEX
BDATE
RELATIONSHIP

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

Exampleofasimplequeryontworelations

Query 1: Retrieve the name and address of all employees who work for the 'Research'department.

Q1:SELECTFNAME,LNAME,ADDRESSFROMEMPLOYEE,DEPARTMENT WHERE DNAME='Research' AND DNUMBER=DNO

Similar to a SELECT-PROJECT-JOIN sequence of relational algebra operations (DNAME='Research') is a selection condition (corresponds to a SELECT operation in relational algebra) (DNUMBER=DNO) is a join condition (corresponds to a JOIN operation in relational algebra)

Exampleofasimplequeryonthree relations

Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

Q2:SELECTPNUMBER,DNUM,LNAME,BDATE,ADDRESSFROMPROJECT, DEPARTMENT,EMPLOYEEWHERE DNUM=DNUMBERANDMGRSSN=SSN ANDPLOCATION='Stafford'

InQ2,therearetwo joinconditions.ThejoinconditionDNUM=DNUMBERrelatesaprojecttoits controllingdepartment ThejoinconditionMGRSSN=SSNrelatesthecontrollingdepartmenttothe employee who manages that department

ALIASES,*ANDDISTINCT,EMPTYWHERE-CLAUSE

- InSQL,wecanusethesamename fortwo(ormore)attributesas longastheattributesare in different relations
- Aquerythatreferstotwoormoreattributeswith thesamenamemustqualifytheattribute name with the relation name byprefixing the relation name to the attributename

  Example:EMPLOYEE.LNAME,DEPARTMENT.DNAME
- Somequeriesneedtorefertothesamerelationtwice.Inthiscase,aliasesaregiventothe relation name.

Example

Query 3: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

Q3:SELECTE.FNAME,E.LNAME,S.FNAME,S.LNAMEFROMEMPLOYEESWHERE E.SUPERSSN=S.SSN

In Q3, the alternate relation names E and S arecalled aliases or tuple variables for the EMPLOYEE relationWecanthinkof EandSastwodifferentcopiesofEMPLOYEE;Erepresentsemployeesin role of supervisees and S represents employees in role of supervisors

AliasingcanalsobeusedinanySQLqueryforconvenience.CanalsousetheAS keyword to

specifyaliases

Q3:SELECTE.FNAME,E.LNAME,S.FNAME,S.LNAMEFROMEMPLOYEEASE, EMPLOYEE AS S WHERE E.SUPERSSN=S.SSN

UNSPECIFIEDWHERE-clause

AmissingWHERE-clauseindicatesno condition;hence,alltuplesoftherelationsintheFROM- clause are selected. This is equivalent to the condition WHERE TRUE

Example:

Query4:RetrievetheSSNvaluesforallemployees.

Q4:SELECTSSNFROMEMPLOYEE

IfmorethanonerelationisspecifiedintheFROM-clauseandthereisnojoincondition,thenthe CARTESIAN PRODUCT of tuples is selected

Example:

Q5:SELECTSSN,DNAMEFROMEMPLOYEE,DEPARTMENT

Note: Itisextremelyimportantnottooverlookspecifyinganyselectionand joinconditionsinthe WHERE-clause; otherwise, incorrect and very large relations mayresult

USEOF*

Toretrieveallt heattributevaluesoftheselectedtuples,a*isused,whichstands forallthe attributes

Examples:

R**etrieveallt heattributevaluesofEMPLOYEESwhoworkin department5.**

Q1a:SELECT*FROMEMPLOYEEWHERE DNO=5

Retrieveallt heattributesofanemployeeandattributesofDEPARTMENTheworksinfor every employee of 'Research' department.

Q1b:SELECT*FROMEMPLOYEE,DEPARTMENTWHEREDNAME='Research' AND DNO=DNUMBER

USEOFDISTINCT

SQLdoesnottreat arelationasaset;duplicatetuplescanappear. Toeliminateduplicatetuples in a query result, the keyword DISTINCT is used

Example:theresultof**Q1c**mayhaveduplicateSALARYvalueswhereas **Q1d** doesnothaveany duplicate values

Q1c:SELECTSALARYFROMEMPLOYEE

Q1d:SELECT**DISTINCT**SALARY FROMEMPLOYEE

SETOPERATIONS

SQLhas directlyincorporated some set operations such as union operation(UNION), set difference (MINUS) and intersection (INTERSECT) operations. The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result. The set operations apply only to union compatible relations; the two relations must have the same attributes and the attributes must appear in the sameorder

Query 5: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

Q5:(SELECTPNAMEFROMPROJECT,DEPARTMENT,EMPLOYEEWHERE DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')

UNION

(SELECT PNAME FROM PROJECT, WORKS_ON, EMPLOYEE WHERE PNUMBER=PNO AND ESSN=SSN AND NAME='Smith')

NESTINGOFQUERIES

A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query, called the outer query. Many of the previous queries can be specified inan alternative form using nesting

Query 6: Retrieve the name and address of all employees who work for the 'Research'department.

Q6:SELECTFNAME,LNAME,ADDRESSFROMEMPLOYEEWHEREDNO**IN** (SELECTDNUMBERFROMDEPARTMENTWHEREDNAME='Research')

Note: The nested query selects the number of the 'Research' department. The outer query selects an EMPLOYEE tuple if its DNO value is in the result of either nested query. The comparison operator INcompares avaluev with aset (ormulti-set)ofvaluesV,and evaluates to TRUEifvis oneofthe elements in V

In general, we can have several levels of nested queries. A reference to an unqualified attribute refersto the relation declared in the innermost nestedquery. In this example, thenested queryis not correlated with the outer query

CORRELATEDNESTEDQUERIES

If a condition in the WHERE-clause of a nested query references an attribute of a relation declared in the outer query, the two queries are said to be correlated. The result of a correlated nested queryis different for each tuple (or combination of tuples) of the relation(s) the outer query

Query 7:Retrievethenameofeachemployeewhohasadependentwiththesamefirstname as the employee.

Q7:SELECTE.FNAME,E.LNAMEFROMEMPLOYEEASEWHEREE.SSN**IN** (SELECT ESSN FROM DEPENDENT WHERE ESSN=E.SSN AND

E.FNAME=DEPENDENT_NAME)

InQ7,thenestedqueryhasadifferentresultintheouterquery.Aquerywrittenwithnested SELECT... FROM… WHERE... blocks and using the **= orIN** comparison operators can *always* be expressed as a single block query. For example, Q7 may be written as in Q7a

Q7a:SELECTE.FNAME,E.LNAMEFROMEMPLOYEEE,DEPENDENTDWHERE E.SSN=D.ESSN AND E.FNAME=D.DEPENDENT_NAME

THEEXISTSFUNCTION

EXISTSisusedtocheckwhethertheresultofacorrelatednestedqueryisempty(containsno tuples) or not. We can formulate Query7 in an alternative form that uses EXIST.

Q7b:SELECTFNAME,LNAMEFROMEMPLOYEE

WHERE**EXISTS**(SELECT*FROMDEPENDENTWHERESSN=ESSN

AND FNAME=DEPENDENT_NAME)

Query8: Retrievethenamesofemployeeswhohavenodependents.

Q8:SELECTFNAME,LNAMEFROMEMPLOYEE WHERE

NOT EXISTS

(SELECT*FROMDEPENDENTWHERESSN=ESSN)

Note: InQ8,thecorrelatednestedqueryretrievesallDEPENDENTtuplesrelatedtoan EMPLOYEE tuple. If none exist, the EMPLOYEE tuple is selected

EXPLICITSETS

Itisalsopossibleto useanexplicit(enumerated)setofvaluesinthe WHERE-clauseratherthana nested query

Query9:Retrievethesocialsecuritynumbersofallemployeeswhoworkonprojectnumber 1, 2, or 3.

Q9:SELECTDISTINCTESSNFROMWORKS_ONWHEREPNO**IN(1,2,3)**

NULLS IN SQL QUERIES

SQL allows queries that check if a value is NULL (missing or undefined or not applicable). SQL uses IS or IS NOT to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate.

Query10:Retrievethenamesofallemployeeswhodonothavesupervisors.

Q10:SELECTFNAME,LNAMEFROMEMPLOYEE WHERE

SUPERSSN IS NULL

Note: IfajoinconditionisspecifiedtupleswithNULLvaluesforthejoinattributesarenot

includedintheresult

AGGREGATEFUNCTIONS

IncludeCOUNT,SUM,MAX,MIN,andAVG

Query11:Findthemaximumsalary,theminimumsalary,andtheaveragesalaryamongall employees.

Q11:SELECT**MAX(SALARY),MIN(SALARY),AVG(SALARY)**

FROMEMPLOYEE

Note: SomeSQLimplementations maynotallow morethanonefunctionintheSELECT-clause

Query12:Findthemaximumsalary,theminimumsalary,andtheaveragesalaryamong employees who work for the 'Research' department.

Q12: SELECT **MAX (SALARY), MIN(SALARY), AVG(SALARY)** FROM EMPLOYEE, DEPARTMENT WHERE DNO=DNUMBER AND DNAME='Research'

Queries13and14:Retrievethetotalnumberofemployeesinthecompany(Q13),andthenumber of employees in the 'Research' department (Q14).

Q13:SELECT**COUNT(*)**FROMEMPLOYEE

Q14:SELECT**COUNT(*)**FROMEMPLOYEE,DEPARTMENT

WHERE DNO=DNUMBER AND DNAME='Research'

GROUPING

- Inmanycases,wewant toapplytheaggregatefunctions to subgroupsoftuples inarelation •

  Each subgroup of tuples consists ofthe set of tuples that havethe same value for the grouping attribute(s)

- Thefunctionisappliedtoeachsubgroupindependently
- SQLhasaGROUPBY-clauseforspecifyingthegroupingattributes,whichmust also appear in the SELECT-clause

Query15:Foreachdepartment,retrievethedepartment number,thenumberofemployees in the department, and their average salary.

Q15:SELECTDNO,COUNT(*),AVG(SALARY)

FROM EMPLOYEE **GROUP BY** DNO

- InQ15,theEMPLOYEEtuplesaredividedinto groups.Eachgrouphavingthesamevalue for the grouping attribute DNO
- TheCOUNT andAVGfunctionsareappliedtoeachsuchgroupoftuples separately

- TheSELECT-clauseincludesonlythegroupingattributeandthefunctionstobeappliedon each group of tuples
- Ajoinconditioncan beusedinconjunctionwithgrouping

Query 16: For each project, retrieve the project number, project name, and the number of employees who work on that project.

```
Q16:SELECTPNUMBER,PNAME,COUNT(*)
FROM PROJECT, WORKS_ON
WHEREPNUMBER=PNOGROUP
BY PNUMBER, PNAME
```

THEHAVING-CLAUSE

Sometimes we wantto retrieve the values of these functions for only those groups that satisfy certain conditions. The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)

Query 17: For each project on which more than two employees work, retrieve the project number, project name, and the number of employees who work on that project.

```
Q17:SELECTPNUMBER,PNAME,COUNT(*)
FROM PROJECT, WORKS_ON
WHEREPNUMBER=PNOGROUP
BYPNUMBER,PNAMEHAVING
COUNT (*) > 2
```

SUBSTRINGCOMPARISON

TheLIKEcomparisonoperatorisusedtocomparepartialstrings.Tworeservedcharacters areused: **'%'**(or '*' in some implementations) replaces an arbitrary number of characters, and **'_'**replaces a single arbitrary character.

Query18:RetrieveallemployeeswhoseaddressisinHouston,Texas.Here,thevalueofthe ADDRESS attribute must contain the substring 'Houston,TX' in it.

```
Q18:SELECTFNAME, LNAME
FROMEMPLOYEEWHEREADDRESSLIKE '%Houston,TX%'
```

Query19:Retrieveallemployeeswhowereborn during the1950s.

Here,'5'mustbethe8thcharacterofthestring(according toourformatfordate),sotheBDATE value is '_____5_', with each underscore as a place holder for a single arbitrary character.

```
Q19:SELECTFNAME, LNAME
```

FROMEMPLOYEEWHEREBDATE**LIKE**'_____5_'

Note: TheLIKEoperator allowsustogetaroundthefactthateachvalueisconsideredatomicand indivisible. Hence, in SQL, character string attribute values are not atomic

ARITHMETICOPERATIONS

Thestandardarithmeticoperators'+','-'.'*',and'/'(foraddition,subtraction,multiplication,and division, respectively) can be applied to numeric values in an SQL queryresult

Query20:Showtheeffectofgivingallemployeeswhoworkonthe'ProductX'projecta10% raise.

Q20:SELECTFNAME,LNAME,1.1*SALARY

FROM EMPLOYEE, WORKS_ON, PROJECT

WHERE SSN=ESSN

AND PNO=PNUMBER AND PNAME='ProductX'

ORDER BY

TheORDERBYclauseisusedtosortthetuplesinaqueryresultbasedon thevaluesofsome attribute(s)

Query 21: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

Q21:SELECTDNAME,LNAME,FNAME,PNAME

FROMDEPARTMENT,EMPLOYEE,WORKS_ON,PROJECT

WHERE DNUMBER=DNO

ANDSSN=ESSN

ANDPNO=PNUMBERORDER

BY DNAME, LNAME

The default order is in ascending order of values. We can specify the keyword DESC if we want a descending order; the keyword ASC can be used to explicitly specify ascending order, even though it is thedefault

Ex:ORDERBYDNAME**DESC**,LNAME**ASC**,FNAME**ASC MORE**

EXAMPLE QUERIES:

Query22:Retrievethenamesofallemployeeswhohavetwoormoredependents.

Q22:SELECTLNAME,FNAME FROM

EMPLOYEEWHERE(SELECTCOUNT(*)FROMDEPENDENT WHERE

SSN=ESSN) ≥ 2):

Query23:Listthenamesofmanagerswhohaveleastonedependent.

Q23:SELECTFNAME,LNAME FROM

EMPLOYEE

WHEREEXISTS(SELECT*FROMDEPENDENTWHERESSN=ESSN)AND

EXISTS (SELECT * FROM DEPARTMENT WHERE SSN=MGRSSN);

SPECIFYINGUPDATESINSQL

TherearethreeSQLcommandstomodifythedatabase:**INSERT**,**DELETE**, and **UPDATE. INSERT**

- Initssimplestform,itisusedtoaddoneormoretuplestoarelation
- Attributevaluesshouldbelistedinthesameorderastheattributeswerespecifiedinthe **CREATETABLE**command

Example:

INSERTINTOEMPLOYEEVALUES('Richard','K','Marini','653298653','30-DEC-52','98Oak Forest,Katy,TX', 'M', 37000,'987654321', 4)

- AnalternateformofINSERTspecifiesexplicitlytheattributenamesthatcorrespondto the values in the new tuple. Attributes with NULL values can be left out.

Example: Insert atupleforanewEMPLOYEEforwhomweonlyknowtheFNAME, LNAME,and SSN attributes.

INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)VALUES ('Richard', 'Marini', '653298653')

Important Note: Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database. Another variation of INSERT allows insertion of multiple tuples resulting from a **query** into a relation

Example: Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department. A table DEPTS_INFO is created first, and is loaded with the summary information retrieved from the database by the query.

CREATETABLEDEPTS_INFO(DEPT_NAMEVARCHAR(10),NO_OF_EMPSINT,

TOTAL_SAL INT);

INSERTINTODEPTS_INFO(DEPT_NAME,NO_OF_EMPS,TOTAL_SAL)

SELECTDNAME,COUNT(*),SUM(SALARY)FROMDEPARTMENT,EMPLOYEE WHERE DNUMBER=DNO GROUP BY DNAME ;

Note: TheDEPTS_INFOtablemaynotbeup-to-dateifwechangethetuplesineitherthe

DEPARTMENTortheEMPLOYEErelations*after*issuingtheabove.Wehavetocreate aview (see later) to keep such a table up to date.

DELETE

- Removestuplesfromarelation. IncludesaWHERE-clausetoselectthetuplestobedeleted •

  Referential integrity should be enforced

- Tuplesaredeleted fromonly*onetable*at atime(unlessCASCADE isspecifiedona referential integrity constraint)
- AmissingWHERE-clausespecifiesthat *alltuples*intherelationaretobedeleted;thetable then becomes an empty table
- Thenumberoftuplesdeleteddependsonthenumberoftuples intherelationthat satisfythe WHERE-clause

Examples:

1. DELETEFROMEMPLOYEE WHERE LNAME='Brown';
2. DELETEFROMEMPLOYEE WHERESSN='123456789';
3. DELETEFROMEMPLOYEEWHEREDNOIN(SELECTDNUMBERFROM DEPARTMENT WHERE DNAME='Research');
4. DELETEFROMEMPLOYEE;

UPDATE

- Usedtomodifyattributevaluesofoneormoreselectedtuples •

  A WHERE-clause selects the tuples to be modified

- AnadditionalSET-clausespecifiestheattributestobemodifiedandtheirnew values •

  Each command modifies tuples *in the same relation*

- Referentialintegrityshouldbeenforced

Example1: Changethelocationandcontrollingdepartmentnumberofprojectnumber10to 'Bellaire' and 5, respectively.

UPDATEPROJECT

SETPLOCATION='Bellaire',DNUM=5WHEREPNUMBER=10;

Example2: Giveallemployeesinthe'Research'departmenta10%raiseinsalary.

UPDATEEMPLOYEE

SETSALARY=SALARY *1.1

WHEREDNOIN(SELECTDNUMBERFROMDEPARTMENT

WHEREDNAME='Research');

INTRODUCTIONTOJOINS

- Joinshelpretrievingdatafromtwoormoredatabasetables.
- AJOINclauseisusedtocombinerows fromtwoormoretables,basedonarelatedcolumn between them.
- Joinestablishestemporarilyrelationshipbetweentwoormoretables.Thetablesaremutually related using primary and foreign keys.

CREATETABLEMOVIES(MOVIE_IDINT(5)PRIMARYKEY,MOVIE_NAME VARCHAR(50));

CREATETABLEACTORS(ACTOR_IDINT(5)PRIMARYKEY,ACTOR_NAME VARCHAR(50), MOVIE_ID INT(5),FOREIGN KEY(MOVIE_ID) REFERENCES MOVIES(MOVIE_ID));

HEREISINSERTSCRIPTSFORBOTHTABLESMOVIES ASWELLASACTORS

INSERT INTOMOVIES VALUES(1000,'SHOLAY'); INSERT INTOMOVIES VALUES(1001,'ITTEFAQ');
INSERTINTOMOVIESVALUES(1002,'TEESRIMANZIL');
INSERT INTO MOVIES VALUES(1003,'JEWEL THIEF ');
INSERT INTO MOVIES VALUES(1004,'CARAVAN');
INSERT INTOMOVIES VALUES(1005,' GUMNAAM');

INSERTINTOACTORSVALUES(1,'AMITABHBACHCHAN',1000);
INSERT INTO ACTORS VALUES(2,'RAJESH KHANNA',1001);
INSERT INTO ACTORS VALUES(3,'SHAMI KAPOOR',1002);
INSERT INTOACTORS VALUES(4,'DEV ANAND',1003);
INSERTINTOACTORSVALUES(5,'NULL',1004);

SELECT * FROM MOVIES;

MOVIE_ID	MOVIE_NAME
1000	SHOLAY
1001	ITTEFAQ
1002	TEESRIMANZIL
1003	JEWELTHIEF
1004	CARAVAN
1005	GUMNAAM

SELECT*FROMACTORS;

ACTOR_ID	ACTOR_NAME	MOVIE_ID
1	AMITABHBACHCHAN	1000
2	RAJESHKHANNA	1001
3	SHAMIKAPOOR	1002
4	DEVANAND	1003
5	NULL	1004

TypesofJOINS

Cross JOIN

CrossJOINisasimplest formofJOINswhichmatcheseachrowfromonedatabasetableto all rows of another.

Inotherwordsit givesuscombinationsofeachrowoffirst tablewithallrecords insecond table.

Select*FROMTableACROSS JOINTableB;

//OR//

Select*FROMTable1A1,Table1A2;

SELECT*FROMMOVIESCROSSJOINACTORS;

ExecutingtheabovescriptinMySQLworkbench givesusthefollowingresults.

MOVIE_ID	MOVIE_NAME	ACTOR_ID	ACTOR_NAME	MOVIE_ID
1000	SHOLAY	1	AMITABHBACHCHAN	1000
1000	SHOLAY	2	RAJESHKHANNA	1001
1000	SHOLAY	3	SHAMIKAPOOR	1002
1000	SHOLAY	4	DEVANAND	1003
1000	SHOLAY	5	NULL	1004
1001	ITTEFAQ	1	AMITABHBACHCHAN	1000
1001	ITTEFAQ	2	RAJESHKHANNA	1001
1001	ITTEFAQ	3	SHAMIKAPOOR	1002
1001	ITTEFAQ	4	DEVANAND	1003
1001	ITTEFAQ	5	NULL	1004

1002	TEESRIMANZIL	1	AMITABHBACHCHAN	1000
1002	TEESRIMANZIL	2	RAJESHKHANNA	1001
1002	TEESRIMANZIL	3	SHAMIKAPOOR	1002
1002	TEESRIMANZIL	4	DEVANAND	1003
1002	TEESRIMANZIL	5	NULL	1004
1003	JEWELTHIEF	1	AMITABHBACHCHAN	1000
1003	JEWELTHIEF	2	RAJESHKHANNA	1001
1003	JEWELTHIEF	3	SHAMIKAPOOR	1002
1003	JEWELTHIEF	4	DEVANAND	1003
1003	JEWELTHIEF	5	NULL	1004
1004	CARAVAN	1	AMITABHBACHCHAN	1000
1004	CARAVAN	2	RAJESHKHANNA	1001
1004	CARAVAN	3	SHAMIKAPOOR	1002
1004	CARAVAN	4	DEVANAND	1003
1004	CARAVAN	5	NULL	1004
1005	GUMNAAM	1	AMITABHBACHCHAN	1000
1005	GUMNAAM	2	RAJESHKHANNA	1001
1005	GUMNAAM	3	SHAMIKAPOOR	1002
1005	GUMNAAM	4	DEVANAND	1003
1005	GUMNAAM	5	NULL	1004

INNERJOIN

Technically,Joinmade byusingequality-operator(=)tocomparevaluesofPrimaryKeyofone table and Foriegn Key values of antoher table, hence result set includes common(matched) records from both tables

The innerJOINisusedto returnrowsfrombothtablesthatsatisfythegivencondition.

SELECT*FROMTable1AINNERJOINTable2B ONA.<PrimaryKey>=B.<ForeignKey>;

SELECTMOVIE_NAME,ACTOR_NAME,ACTOR_IDFROMMOVIESMINNERJOIN ACTORS A ON M.MOVIE_ID=A.MOVIE_ID;

ExecutingtheabovescriptinMySQLworkbenchgivesusthefollowingresut

MOVIE_NAME	ACTOR_NAME	ACTOR_ID
SHOLAY	AMITABHBACHCHAN	1
ITTEFAQ	RAJESHKHANNA	2
TEESRIMANZIL	SHAMIKAPOOR	3

JEWELTHIEF	DEVANAND	4
CARAVAN	NULL	5

INNERJOINCONSISTINGOFWHERECONDITIONANDACTORNAMESHOULD NOT BE NULL

SELECTMOVIE_NAME,ACTOR_NAME,ACTOR_IDFROMMOVIES MINNERJOIN ACTORSAONM.MOVIE_ID=A.MOVIE_IDWHEREACTOR_NAME!='NULL';

OR

SELECTMOVIE_NAME,ACTOR_NAME,ACTOR_IDFROMMOVIESMINNERJOIN ACTORS A ON M.MOVIE_ID=A.MOVIE_ID WHERE ACTOR_NAME<>'NULL';

ExecutingtheabovescriptinMySQLworkbenchgivesusthefollowing results.

MOVIE_NAME	ACTOR_NAME	ACTOR_ID
SHOLAY	AMITABHBACHCHAN	1
ITTEFAQ	RAJESHKHANNA	2
TEESRIMANZIL	SHAMIKAPOOR	3
JEWELTHIEF	DEVANAND	4

INNERJOINCONSISTINGOFWHERECONDITIONANDACTORNAMEFIELDSARE HAVING NULL VALUE

SELECTMOVIE_NAME,ACTOR_NAME,ACTOR_IDFROMMOVIESMINNERJOIN ACTORS A ON M.MOVIE_ID=A.MOVIE_ID WHERE ACTOR_NAME='NULL';

Executingtheabovescript inMySQLworkbenchgivesusthefollowing results.

MOVIE_NAME	ACTOR_NAME	ACTOR_ID
CARAVAN	NULL	5

OUTER-JOIN

Afull outerjoin, orfull join, which is not supported bythepopularMySQLdatabasemanagement system,However,cancustomizedselectionofun-matchedrowse.g,selectingunmatchedrowfrom first table or second table by sub-types: LEFT OUTER JOIN and RIGHT OUTER JOIN.

It candetectrecordshavingno matchinjoinedtable.ItretunsNULLvalues forrecordsofjoined table if no match is found.

LEFTJOIN

The LEFT JOIN retuns all the rows from the table on the left even if no matching rows have beenfoundinthetableontheright. Whereno matcheshavebeenfoundinthetableontheright, NULL is returned.

Select*FROMTable1 ALEFTOUTERJOINTable2BOn A.<PrimaryKey>=B.<ForeignKey>;

SELECTMOVIE_NAME,ACTOR_NAME,ACTOR_IDFROMMOVIESMLEFT OUTER JOIN ACTORS A ON M.MOVIE_ID=A.MOVIE_ID;

Executing the above script in MySQL workbench gives beloew result.You can see that in the returnedresult whichislistedbelowthatformovieswhichdonothaveaactor,actornamefields are having NULL values. That means no matching member found actor table for that particular movie.

MOVIE_NAME	ACTOR_NAME	ACTOR_ID
SHOLAY	AMITABHBACHCHAN	1
ITTEFAQ	RAJESHKHANNA	2
TEESRIMANZIL	SHAMIKAPOOR	3
JEWELTHIEF	DEVANAND	4
CARAVAN	NULL	5
GUMNAAM	NULL	NULL

RIGHTJOIN

RIGHTJOINisobviouslytheoppositeofLEFTJOIN.TheRIGHTJOIN retunsallthecolumns fromthe table onthe right even if no matching rows have been found in the table on the left. Wherenomatcheshavebeenfoundinthetableontheleft, NULLisreturned.

Select*FROMTable1ARIGHTOUTERJOINTable2BonA.<PrimaryKey>=B.<ForeignKey>;

SELECTMOVIE_NAME,ACTOR_NAME,ACTOR_IDFROMMOVIESMRIGHT OUTER JOIN ACTORS A ON M.MOVIE_ID=A.MOVIE_ID;

ExecutingtheabovescriptinMySQLworkbenchgivesthefollowingresults.

MOVIE_NAME	ACTOR_NAME	ACTOR_ID
SHOLAY	AMITABHBACHCHAN	1
ITTEFAQ	RAJESHKHANNA	2
TEESRIMANZIL	SHAMIKAPOOR	3
JEWELTHIEF	DEVANAND	4
CARAVAN	NULL	5

INTRODUCTIONTOSUBQUERY

- Asubqueryisaquerywithinanotherquery. Theouterqueryiscalled as mainqueryand inner query is called as subquery.
- Subqueriesarenestedqueriesthatprovidedatatotheenclosingquery.
- Subquery must be enclosed in parentheses.
- Subqueriescanreturnindividual valuesor alistofrecords
- YoucanplacetheSubqueryinanumberofSQLclauses:WHEREclause,HAVINGclause, FROM clause.
- SubqueriescanbeusedwithSELECT,UPDATE,INSERT,DELETEstatementsalongwith expressionoperator. It couldbeequalityoperatororcomparisonoperatorsuchas=,>,=,<= and Like operator.
- Thesubquerygenerallyexecutes first,and itsoutput isusedto completethequerycondition for the main or outer query.
- Subqueriesareontherightsideofthecomparison operator.
- ORDERBYcommand**cannot**beusedinaSubquery. GROUPBYcommandcanbeusedto perform same function as ORDER BY command.
- Usesingle-rowoperatorswithsinglerowSubqueries.Usemultiple-rowoperatorswith multiple-row Subqueries.

Syntax:

- ThereisnotanygeneralsyntaxforSubqueries.However,Subqueriesareseentobeusedmost frequently with SELECT statement as shown below:

  SELECTcolumn_nameFROMtable_nameWHEREcolumn_nameexpressionoperator (

  SELECT COLUMN_NAMEfrom TABLE_NAME WHERE ...);

BelowissampletableStudentDetailsandStudentSectionwehavecreatedtodemonstrate working of subquesry

```
createtabStudentDetails
(
Student_IDintprimarykey
NAME          varchar(100),
ROLL_NO     int,
LOCATION varchar(100),
PHONE_NUMBERbigint
);
```

```
createtableStudentSection
(
NAMEvarchar(100),
ROLL_NO      int,
Section char(1)
);
```

Belowissampleinsert scriptsforinsertinginformationintoStudentDetails

```
insertintoStudentDetailsvalues(1000,'Hemanth',101,'Mysore',9845113337);
insert into StudentDetails values(2000,'Nitin',102,'Banglore',8877665544);
insertintoStudentDetailsvalues(3000,'Sandeep',103,'Kodagu',9538945623);
insertintoStudentDetailsvalues(4000,'SashankHegde',104,'Udupi',8989898989);
insert into StudentDetails values(5000,'Nagendra',105,'Banglore',9901478945);
```

Belowissampleinsertscriptsforinsertinginformationintostudentsection insert

```
into StudentSection values('Sashank Hegde',104,'A');
insertintoStudentSectionvalues('Nagendra',105,'B');
insert into StudentSection values('Nitin',102,'A');
insert into StudentSectionvalues('Hemanth',101,'B');
```

select*fromStudentDetails

```
mysql> select * from StudentDetails;
+------------+---------------+---------+----------+--------------+
| Student_ID | NAME          | ROLL_NO | LOCATION | PHONE_NUMBER |
+------------+---------------+---------+----------+--------------+
|       1000 | Hemanth       |     101 | Mysore   |   9845113337 |
|       2000 | Nitin         |     102 | Banglore |   8877665544 |
|       3000 | Sandeep       |     103 | Kodagu   |   9538945623 |
|       4000 | Sashank Hegde |     104 | Udupi    |   8989898989 |
|       5000 | Nagendra      |     105 | Danglore |   9901478945 |
+------------+---------------+---------+----------+--------------+
5 rows in set (0.00 sec)

mysql>
```

select *fromStudentSection

```
mysql> select * from StudentSection;
+---------------+---------+---------+
| NAME          | ROLL_NO | Section |
+---------------+---------+---------+
| Sashank Hegde |     104 | A       |
| Nagendra      |     105 | B       |
| Nitin         |     102 | A       |
| Hemanth       |     101 | B       |
+---------------+---------+---------+
4 rows in set (0.00 sec)

mysql>
```

Query1:TodisplayNAME,LOCATION,PHONE_NUMBERoftheStudentDetailstablewhose section is A

SelectNAME,LOCATION,PHONE_NUMBERfromStudentDetailsWHEREROLL_NO IN(SELECT ROLL_NO from StudentSection where SECTION='A');

Explanation : First subquery executes " SELECT ROLL_NO from STUDENT where SECTION='A'" returns ROLL_NO from STUDENT table whose SECTION is 'A'.Then outer-queryexecutesitandreturntheNAME,LOCATION,PHONE_NUMBERfromtheDATABASE table of the student whose ROLL_NO is returned from inner subquery.

Belowis snapshotofoutput ofaboveexceuted subquery

```
mysql> Select NAME,LOCATION,PHONE_NUMBER from StudentDetails
    -> WHERE ROLL_NO IN(SELECT ROLL_NO from StudentSection where SECTION='A');
+---------------+----------+--------------+
| NAME          | LOCATION | PHONE_NUMBER |
+---------------+----------+--------------+
| Nitin         | Banglore |   8877665544 |
| Sashank Hegde | Udupi    |   8989898989 |
+---------------+----------+--------------+
2 rows in set (0.00 sec)

mysql>
```

Query2:ToupdatenamefromStudentDetailstablewhoserollnoissameasthatin StudentSection table and having nameas Nitin by Using subquery

UPDATEStudentDetailsSETNAME='Nitin Jain'

WHEREROLL_NOIN(SELECTROLL_NOFROMStudentSectionwhere NAME='Nitin');

```
mysql> select * from studentdetails;
+------------+---------------+---------+----------+--------------+
| Student_ID | NAME          | ROLL_NO | LOCATION | PHONE_NUMBER |
+------------+---------------+---------+----------+--------------+
|       1000 | Hemanth       |     101 | Mysore   |   9845113337 |
|       2000 | Nitin         |     102 | Banglore |   8877665544 |
|       3000 | Sandeep       |     103 | Kodagu   |   9538945623 |
|       4000 | Sashank Hegde |     104 | Udupi    |   8989898989 |
|       5000 | Nagendra      |     105 | Banglore |   9901478945 |
+------------+---------------+---------+----------+--------------+
5 rows in set (0.00 sec)

mysql> UPDATE StudentDetails SET NAME='Nitin Jain'
    -> WHERE ROLL_NO IN(SELECT ROLL_NO FROM StudentSection where NAME='Nitin');
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from studentdetails;
+------------+---------------+---------+----------+--------------+
| Student_ID | NAME          | ROLL_NO | LOCATION | PHONE_NUMBER |
+------------+---------------+---------+----------+--------------+
|       1000 | Hemanth       |     101 | Mysore   |   9845113337 |
|       2000 | Nitin Jain    |     102 | Banglore |   8877665544 |
|       3000 | Sandeep       |     103 | Kodagu   |   9538945623 |
|       4000 | Sashank Hegde |     104 | Udupi    |   8989898989 |
|       5000 | Nagendra      |     105 | Banglore |   9901478945 |
+------------+---------------+---------+----------+--------------+
5 rows in set (0.00 sec)
```

Query3:TodeletestudentsfromStudent2tablewhoserollnoissameasthatinStudent1table and having location as chennai

DELETEFROMStudentDetailsWHEREROLL_NOIN(SELECTROLL_NO

FROMStudentSectionWHERENAME=Nagendra); ’

```
mysql> select * from StudentDetails;
+------------+---------------+---------+-----------+--------------+
| Student_ID | NAME          | ROLL_NO | LOCATION  | PHONE_NUMBER |
+------------+---------------+---------+-----------+--------------+
|       1000 | Hemanth S R   |     101 | Mysore    |   9845113337 |
|       2000 | Hemanth S R   |     102 | Banglore  |   8877665544 |
|       3000 | Sandeep       |     103 | Kodagu    |   9538945623 |
|       4000 | Sashank Hegde |     104 | Udupi     |   8989898989 |
|       5000 | Nagendra      |     105 | Banglore  |   9901478945 |
+------------+---------------+---------+-----------+--------------+
5 rows in set (0.00 sec)

mysql> DELETE FROM StudentDetails  WHERE ROLL_NO IN (SELECT ROLL_NO
    -> FROM StudentSection WHERE NAME = 'Nagendra');
Query OK, 1 row affected (0.00 sec)

mysql> select * from StudentDetails;
+------------+---------------+---------+-----------+--------------+
| Student_ID | NAME          | ROLL_NO | LOCATION  | PHONE_NUMBER |
+------------+---------------+---------+-----------+--------------+
|       1000 | Hemanth S R   |     101 | Mysore    |   9845113337 |
|       2000 | Hemanth S R   |     102 | Banglore  |   8877665544 |
|       3000 | Sandeep       |     103 | Kodagu    |   9538945623 |
|       4000 | Sashank Hegde |     104 | Udupi     |   8989898989 |
+------------+---------------+---------+-----------+--------------+
4 rows in set (0.00 sec)

mysql> _
```

Joinvs.Subquery

- JOINsarefasterthanasubqueryanditisveryrarethat theopposite.
- In JOINs the RDBMS calculates an execution plan, that can predict, what data should be loaded and how much it will take to processed and as a result this process save some times, unlikethesubquerythere isno pre-processcalculationandrunallthequeriesand loadalltheir data to do the processing.
- AJOIN ischeckedconditionsfirst andthenput it intotableanddisplays;whereasasubquery take separate temp table internally and checking condition.
- When joins are using, there should be connection between two or more than two tables and eachtablehasarelationwithotherwhilesubquerymeansqueryinsideanotherquery,hasno need to relation, it works on columns and conditions

VIEWSINSQL

- Aviewisasingle virtualtablethat isderived fromothertables.Theothertablescould be base tables or previously defined view.
- Allowsfor limitedupdateoperationsSincethetablemaynotphysicallybestored •

  Allows full query operations
- Aconvenienceforexpressingcertain operations
- Aviewdoesnotnecessarilyexistinphysicalform,whichlimitsthepossibleupdate operations that can be applied to views.

Viewssyntax

Let'snowlookatthebasicsyntaxusedtocreateaviewinMySQL. CREATE VIEW`view_name`AS SELECT statement;

WHERE

- **"CREATE VIEW"view_name"**tellsMySQLserver to createaviewobject inthe database
- **"ASSELECTstatement"**istheSQLstatementstobepacked intheviews. Itcanbea SELECT statement can contain data from one table or multiple tables.

Example1-SimpleViewconsistingofonlyonetables

```
CREATEVIEWVW_BOOKDETAILSASSELECTBOOK_NAME,BOOK_AUTHOR, PUBLISHER
 FROM BOOK_DETAILS;
```

ExampleofSimpleViewconsistingoftwotablesand usingwherecondition

```
CREATEVIEWVW_ORDERSASSELECTBOOK_NAME,BOOK_AUTHOR,
PUBLISHER FROM ORDERS A, BOOK_DETAILS B WHERE
A.ORDER_ID=B.ORDER_ID;
```

ExampleofViewconsisting ofinnerjoin

```
CREATEVIEWVW_BOOK1ASSELECTA.ORDER_ID,ORDER_DATE,BOOK_NAME,
BOOK_AUTHOR, PUBLISHER FROM ORDERS A INNER JOIN BOOK_DETAILS B ON
A.ORDER_ID=B.ORDER_ID;
```

ExampleofViewconsisting ofinnerjoinandwherecondition

```
CREATEVIEWVW_BOOK2ASSELECTA.ORDER_ID,ORDER_DATE,BOOK_NAME,
BOOK_AUTHOR, PUBLISHER FROM ORDERS A INNER JOIN BOOK_DETAILS B ON
A.ORDER_ID=B.ORDER_ID ID AND PUBLISHER='Tata McGraw-Hill'
```

```
SELECT*FROMVW_BOOKDETAILS
SELECT * FROM VW_ORDERS
SELECT * FROM VW_BOOK1
SELECT * FROM VW_BOOK2
```

SHOWTABLES

Tables_in_naveen
VW_BOOKDETAILS
VW_ORDERS
VW_BOOK1
VW_BOOK2

Example2-SimpleView

CREATEVIEWVW_PUBLICATIONASSELECTPUB_YEARFROMBOOK; SELECT *

FROM VW _PUBLICATION

```
mysql> SELECT * FROM VW_PUBLICATION;
+-----------+
| PUB_YEAR  |
+-----------+
| JAN-2017  |
| JUN-2016  |
| SEP-2016  |
| SEP-2015  |
| MAY-2016  |
+-----------+
5 rows in set (0.00 sec)
```

DROPINGVIEWS

TheDROPcommandcanbeusedtodeleteaviewfromthedatabasethatisno longerrequired. The

basic syntax to drop a view is as follows.

DROP VIEWVIEWNAME;

DROPVIEWV_PUBLICATION;

Youmaywanttouseviewsprimarilyforfollowing3reasons

- Ultimately , you will use your SQL knowledge , to create applications , which will use a database for data requirements. It's recommended that you use VIEWS of the original table structureinyourapplicationinsteadofusingthetablesthemselves.Thisensuresthatwhen you refactor your DB, your legacycode will see the orignal schema via the view without breaking the application.
- **VIEWS increase re-usability**. You will not have to create complex queries involving joins repeatedly. All the complexity is converted into a single line of query use VIEWS. Such condensedcodewillbeeasiertointegratein yourapplication.Thiswilleliminateschancesof typos and your code will be more readable.
- **VIEWShelpindatasecurity**.Youcanuseviewsto showonlyauthorized informationto users and hide sensitive data like credit card numbers, pass

INTRODUCTIONTOSTOREDPROCEDURES

- AstoredprocedureisapreparedSQLcodethat youcansave, sothecodecanbereusedoverand over again.So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- Youcanalso passparameterstoastoredprocedure,sothatthestoredprocedurecanact basedon the parameter value(s) that is passed.
- A procedure can return oneor morethan one value through parametersor maynot return atall. TheprocedurecanbeusedinSQLqueries.

Creatingaprocedure

Syntax

```
CREATEPROCEDUREprocedure_name
(
parameterdatatype,
 parameter datatype
)
 BEGIN
        Declaration_section
        Executable_section
END;
```

Parameter

procedure_name:nameofthestoredprocedure.

Parameter:numberofparameters.Itcanbeoneormorethanone.

declaration_section: all variables are declared.

executable_section: code is written here.

Avariableis anameddataobject whosevaluecan changeduringthestored procedureexecution. Wetypicallyusethevariablesin storedproceduresto holdtheimmediateresults.Thesevariables are localto the stored procedure. You must declare a variable before using it.

```
DELIMITER //
CREATEPROCEDUREsp_name
(
p_1INT
)

BEGIN
  ...codegoeshere...
 END //
 DELIMITER ;
```

- Replace procedure_name with sp_procedure_name whatever name you'd like to use for the stored procedure. The parentheses are required — theyencloseanyparameters.Ifnoparameters are required, the parentheses can be empty.

- The main body of the stored procedure goes in between the BEGIN and END keywords. These keywordsareusedforwritingcompoundstatements.Acompoundstatementcancontainmultiple statements,andthesecanbenestedifrequired.Therefore,you cannestBEGINandENDblocks.

- Inmostcases,youwillalsoneedtosurroundtheCREATEPROCEDURE statement with DELIMITER commands and change END; to END //. Like this:

AbouttheDELIMITER Command

- The first commandis DELIMITER // , which is not related tothe stored procedure syntax. TheDELIMITERstatementchangesthestandard delimiterwhich isasemicolon(;)toanother.

- In thiscase, thedelimiterischanged fromthesemicolon(;)todouble-slashes(//)Weneedto change delimiter from ; to //.Because we want to pass the stored procedure to the server as a whole rather than letting mysql tool interpret each statement at a time.

- Following theEND keyword,we use thedelimiter //toindicatetheendof thestoredprocedure. Thelastcommand(DELIMITER;)changesthedelimiterbacktothesemicolon (;).

HowtoExecuteaStoredProcedure

Callsp_procedure_name();

WritingthefirstMySQLstored procedure

Herewearecreatingsampletablenamedemployee

```
createtableemployee
(
employee_idintprimarykey,
Name varchar(50),
Designation varchar(50),
Salarydecimal(10,2)
)

insert into employeevalues(100,'vishwanath','clerk',20000.00);insert
into employeevalues(101,'shashikiran','instructor',20000.00);
insertintoemployeevalues(102,'nitin','assitiantprofessor',25000.00);
insertintoemployeevalues(103,'deepak','associateprofessor',40000.00);
insert into employeevalues(104,'sanjay','professor',80000.00);
insertintoemployeevalues(105,'yogesh','systemadmin',30000.00); insert
into employeevalues(106,'anand','clerk',20000.00);
insertintoemployeevalues(107,'Hemanth','professor',80000.00);
insert into employeevalues(108,'Robert','cashier',15000.00); insert
into employee values(109,'amit','clerk',20000.00);
insertintoemployeevalues(110,'george','HRManager',30000.00);
```

```
mysql> select * from employee;
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
+-------------+-------------+----------------------+----------+
11 rows in set (0.00 sec)
```

Example

We are going to develop a simple stored procedure named SP_getEmployeeto help you get familiarwiththesyntax.TheSP_getEmployee()storedprocedureselectsallemployeeinformation from the employee table.:

DELIMITER $$

DROPPROCEDUREIFEXISTSSP_getEmployee$$

CREATE PROCEDURE SP_getEmployee()

BEGIN

SELECT*FROMemployee;

END$$

Executethestoredprocedureaboveasfollows:

callSP_getEmployee();

```
mysql> call SP_getEmployee();
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
+-------------+-------------+----------------------+----------+
11 rows in set (0.00 sec)

Query OK, 0 rows affected (0.03 sec)

mysql> _
```

IntroductiontoMySQLstoredprocedureparameters

Almoststoredproceduresthat youdeveloprequireparameters.Theparametersmakethestored procedure more flexible and useful.

Thesyntaxofdefiningaparameterinthestoredproceduresisasfollows: MODE

param_name param_type(param_size)

TheMODEcouldbeIN,OUTorINOUT, dependingonthepurposeoftheparameterinthe stored procedure.

Theparam_name isthenameoftheparameter.Thenameoftheparametermust followthenaming rules of the column name in MySQL.

Followedtheparameternameisitsdatatypeandsize.Likeavariable,thedatatypeofthe parameter can be any valid MySQL data type.

Eachparameterisseparated byacomma(,) ifthestoredprocedurehasmorethanoneparameter

IN is the default mode. When you define an IN parameter in a stored procedure, the calling programhastopassanargumenttothestoredprocedure. Inaddition, thevalueofanINparameter is protected. It means that even the value of the IN parameter is changed inside the stored procedure, its original value is retained after the stored procedure ends. In other words, the stored procedure only works on the copy of the IN parameter.

OUTthevalue of an OUT parameter can be changed inside the stored procedure and its new valueispassedbacktothecallingprogram.Noticethatthestoredprocedurecannotaccessthe initial value of the OUT parameter when it starts.

INOUT–anINOUT parameterisacombinationofINandOUTparameters.It meansthatthe calling program may pass the argument, and the stored procedure can modify theINOUTparameter,andpassthenew valuebacktothecallingprogram.

MySQLProcedure:ParameterINexample

Stored Procedure With One Parameter

ThefollowingSQLstatement createsastoredprocedurethatselectsemployeeinformationfroma employee Table based on employee_id from the " SP_getEmployeeid .:

```
//SP_getEmployeeid
DELIMITER $$
DROPPROCEDUREIFEXISTSSP_getEmployeeid$$
CREATE PROCEDURE SP_getEmployeeid
 (
 INemp_idINT(10)
 )
 BEGIN
```

select*FROMemployeewhereemployee_id=emp_id; END$$

Executethestoredprocedurebelow .....asfollows:

```
mysql> call SP_getEmployeeid(110);
+-------------+--------+-------------+----------+
| employee_id | Name   | Designation | Salary   |
+-------------+--------+-------------+----------+
|         110 | george | HR Manager  | 30000.00 |
+-------------+--------+-------------+----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call SP_getEmployeeid(104);
+-------------+--------+-------------+----------+
| employee_id | Name   | Designation | Salary   |
+-------------+--------+-------------+----------+
|         104 | sanjay | professor   | 80000.00 |
+-------------+--------+-------------+----------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> _
```

StoredProcedureWithMultipleParameters

Settingup multipleparametersisveryeasy.Just list eachparameterandthedatatypeseparatedby a comma as shown below.

ThefollowingSQLstatementcreatesastoredprocedurethatselectsstudentdetails froma particular USN with a particular NAME from the " studentmarks" table:

Example

DELIMITER $$

DROPPROCEDUREIFEXISTSSP_getStudentdetails$$

CREATE PROCEDURE SP_getStudentdetails

(

INusn1 varchar(50), IN

name1varchar(50)

)

BEGIN

select*FROMstudentmarkswhereusn=usn1andname=name1; END$$

Executethestoredprocedureaboveasfollows:

```
mysql> select * from studentmarks;
+------------+----------+------------+--------+-------+
| USN        | NAME     | PHONE      | GENDER | Marks |
+------------+----------+------------+--------+-------+
| 4AD17CS010 | ARVIND   | 9900211201 | M      |    21 |
| 4AD17CS011 | AJAY     | 9845091341 | M      |    17 |
| 4AD17CS020 | AKSHAY   | 8877881122 | M      |    25 |
| 4AD17CS025 | AKSHATHA | 7894737377 | F      |    18 |
| 4AD17CS029 | CHANDANA | 7696772121 | F      |    16 |
| 4AD17CS032 | BHASKAR  | 9923211099 | M      |    19 |
| 4AD17CS045 | JEEVAN   | 9944850121 | M      |    15 |
| 4AD17CS062 | SANDHYA  | 7722829912 | F      |    24 |
| 4AD17CS066 | VEENA    |  877881122 | F      |    22 |
| 4AD17CS091 | TARANATH | 7712312312 | M      |    23 |
+------------+----------+------------+--------+-------+
10 rows in set (0.00 sec)

mysql> call SP_getStudentdetails('4AD17CS010','ARVIND');
+------------+--------+------------+--------+-------+
| USN        | NAME   | PHONE      | GENDER | Marks |
+------------+--------+------------+--------+-------+
| 4AD17CS010 | ARVIND | 9900211201 | M      |    21 |
+------------+--------+------------+--------+-------+
1 row in set (0.00 sec)
```

MySQLProcedure:ParameterOUTexample

The following example shows a simple stored procedure that uses an OUT parameter. Within the procedureMySQLMAX() functionretrieves maximumsalaryfromMAX_SALARYofemployee. table.

```
DELIMITER $$
DROPPROCEDUREIFEXISTSsp_getemployemaxsalary()$$
 CREATE PROCEDURE sp_getemployemaxsalary
  (
 outmax_salaryfloat(10,2)
)
 BEGIN
SELECTmax(Salary)intomax_salaryfromemployee; END$$
DELIMITER;
```

In the body of the procedure, the parameter max_salary will get the highest salary from Salary columnofEmployeeTable.AftercallingtheprocedurethewordOUTtellstheMYSQLthatthe valuegoesoutfromtheprocedure.Heremax_salaryisthenameoftheoutput parameter andwe have passed its value to a session variable named @ m, in the CALL statement.

```
mysql> select * from employee;
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
|         111 | Jaishankar  | Project Manager      | 90000.00 |
|         112 | Kiran       | Insurance Manager    | 35000.00 |
+-------------+-------------+----------------------+----------+
13 rows in set (0.00 sec)

mysql>
```

```
mysql> call sp_getemployemaxsalary(@m);
Query OK, 0 rows affected (0.02 sec)

mysql> select  @m;
+-------+
| @m    |
+-------+
| 90000 |
+-------+
1 row in set (0.00 sec)

mysql>
```

MySQLProcedure:ParameterINOUTexample

HerewearecreatingsampletableSTUDENTwithUSNasprimarykey

```
CREATE TABLE STUDENT
 (
USNVARCHAR(10)PRIMARYKEY,
SNAME VARCHAR (25),
ADDRESSVARCHAR(25),
PHONEBIGINT(10),
GENDERCHAR(1)
);
```

HereweareinsertingsampledataintoSTUDENTTable

```
INSERTINTOSTUDENTVALUES('4AD16CS020','AKSHAY','BELAGAVI',8877881122,'M');
INSERT INTO STUDENT VALUES ('4AD16CS062','SANDHYA','BENGALURU',
7722829912,'F');
INSERTINTOSTUDENTVALUES('4AD16CS091','TARANATH','BENGALURU',
7712312312,'M');
INSERTINTOSTUDENTVALUES('4AD16CS066','SUPRIYA','MANGALURU',
8877881122,'F');
INSERTINTOSTUDENTVALUES('4AD16CS010','ABHAY','BENGALURU',9900211201,'M');
```

INSERTINTOSTUDENTVALUES('4AD16CS032','BHASKAR','BENGALURU',
9923211099,'M');
INSERTINTOSTUDENTVALUES('4AD16C S025','AKSHATHA','BENGALURU',
7894737377,'F');
INSERT INTO STUDENT VALUES ('4AD16CS011','AJAY','TUMKUR', 9845091341,'M');
INSERTINTOSTUDENTVALUES('4AD16CS029','CHITRA','DAVANGERE',7696772121,'F');
INSERT INTO STUDENT VALUES ('4AD16CS045','JEEVAN','BELLARY', 9944850121,'M');

```
mysql> SELECT * FROM STUDENT;
+------------+---------+-----------+------------+--------+
| USN        | SNAME   | ADDRESS   | PHONE      | GENDER |
+------------+---------+-----------+------------+--------+
| 4AD16CS010 | ABHAY   | BENGALURU | 9900211201 | M      |
| 4AD16CS011 | AJAY    | TUMKUR    | 9845091341 | M      |
| 4AD16CS020 | AKSHAY  | BELAGAVI  | 8877881122 | M      |
| 4AD16CS025 | AKSHATHA| BENGALURU | 7894737377 | F      |
| 4AD16CS029 | CHITRA  | DAVANGERE | 7696772121 | F      |
| 4AD16CS032 | BHASKAR | BENGALURU | 9923211099 | M      |
| 4AD16CS045 | JEEVAN  | BELLARY   | 9944850121 | M      |
| 4AD16CS062 | SANDHYA | BENGALURU | 7722829912 | F      |
| 4AD16CS066 | SUPRIYA | MANGALURU | 8877881122 | F      |
| 4AD16CS091 | TARANATH| BENGALURU | 7712312312 | M      |
+------------+---------+-----------+------------+--------+
10 rows in set (0.01 sec)
```

The following example shows a simple stored procedure that uses an INOUT parameter and an IN parameter.Theuserwillsupply'M'or 'F'throughINparameterto countanumberofmaleorfemale gender from STUDENT table. The INOUT parameter (countgender) will return the result to a user. Here is the code and output of the procedure

```
DELIMITER $$
DROPPROCEDUREIFEXISTSsp_studentcountgender$$
 create procedure sp_studentcountgender
(
INstudentgenderchar(1),
OUT countgender int
 )
 begin
 selectCOUNT(GENDER)INTOcountgenderFROMSTUDENTWHERE
 GENDER=studentgender;
 END$$
DELIMITER;
```

```
mysql> call sp_studentcountgender('M',@CountGender);
Query OK, 0 rows affected (0.00 sec)

mysql> select @CountGender;
+---------------+
| @CountGender  |
+---------------+
|             6 |
+---------------+
1 row in set (0.00 sec)

mysql> call sp_studentcountgender('F',@CountGender);
Query OK, 0 rows affected (0.00 sec)

mysql> select @CountGender;
+---------------+
| @CountGender  |
+---------------+
|             4 |
+---------------+
1 row in set (0.00 sec)

mysql>
```

Other examplesofstoredprocedureareas follows

Thisanexampleofinsertinginformationintotableemployeethroughstoredprocedure

//INSERTSTOREDPROCEDURE

DELIMITER $$

DROPPROCEDUREIFEXISTSsp_InsertEmployee$$

CREATE PROCEDURE sp_InsertEmployee

(

IN employee_id INT(10),

INNameVARCHAR(255),

INDesignationVARCHAR(255),

IN Salary float(10,2)

)

BEGIN

INSERTINTOemployeevalues(employee_id,Name,Designation,Salary); END$$

```
mysql> call sp_InsertEmployee(111,'Jagdish','System Engineer',40000);
Query OK, 1 row affected (0.02 sec)

mysql> call SP_getEmployee();
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
|         111 | Jagdish     | System Engineer      | 40000.00 |
+-------------+-------------+----------------------+----------+
12 rows in set (0.01 sec)

Query OK, 0 rows affected (0.06 sec)

mysql> _
```

Thisanexampleofupdatinginformation into tableemployeethroughstoredprocedure

//UPDATESTOREDPROCEDURE

DELIMITER $$

DROPPROCEDUREIFEXISTSsp_UpdatetEmployee$$

CREATE PROCEDURE sp_UpdatetEmployee

(

INemp_idINT(10),

INNameVARCHAR(255),

INDesignationVARCHAR(255),

IN Salary float(10,2)

)

BEGIN

UPDATEEMPLOYEESETName=Name,Designation=Designation,Salary=SalaryWHERE employee_id = emp_id ;

END$$

```
mysql> call sp_getemployee;
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
|         111 | Jagdish     | System Engineer      | 40000.00 |
+-------------+-------------+----------------------+----------+
12 rows in set (0.00 sec)

Query OK, 0 rows affected (0.05 sec)

mysql> call sp_UpdatetEmployee(111,'Jaishankar','Project Manager',90000);
Query OK, 1 row affected (0.04 sec)

mysql> call sp_getemployee;
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
|         111 | Jaishankar  | Project Manager      | 90000.00 |
+-------------+-------------+----------------------+----------+
12 rows in set (0.00 sec)
```

```
CREATEPROCEDURESP_InsertUpdateemployee
(
INemp_idint,
INNameVARCHAR(255),
INDesignationVARCHAR(255),
 IN Salary float(10,2)
)
BEGIN
DECLAREcountaint;
SELECTcount(*)INTOcountaFROMemployeeWHEREemployee_id=emp_id; IF
counta=0
```

```
THEN
INSERT INTO employee values(emp_id,Name,Designation,Salary);
ELSEIF counta>0
THEN
UPDATEEMPLOYEESETName=Name,Designation=Designation,Salary=Salary
WHERE employee_id=emp_id;
ENDIF;
END$$
```

```
mysql> call SP_getEmployee;
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
|         111 | Jaishankar  | Project Manager      | 90000.00 |
+-------------+-------------+----------------------+----------+
12 rows in set (0.00 sec)

Query OK, 0 rows affected (0.07 sec)

mysql> call sp_InsertUpdateEmployee(112,'Kishore','Insurance Agent',25000);
Query OK, 1 row affected (0.03 sec)

mysql> call SP_getEmployee;
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
|         111 | Jaishankar  | Project Manager      | 90000.00 |
|         112 | Kishore     | Insurance Agent      | 25000.00 |
+-------------+-------------+----------------------+----------+
13 rows in set (0.00 sec)
```

```
mysql> call SP_getEmployee;
+-------------+-------------+---------------------+----------+
| employee_id | Name        | Designation         | Salary   |
+-------------+-------------+---------------------+----------+
|         100 | vishwanath  | clerk               | 20000.00 |
|         101 | shashikiran | instructor          | 20000.00 |
|         102 | nitin       | assitiant professor | 25000.00 |
|         103 | deepak      | associate professor | 40000.00 |
|         104 | sanjay      | professor           | 80000.00 |
|         105 | yogesh      | system admin        | 30000.00 |
|         106 | anand       | clerk               | 20000.00 |
|         107 | Hemanth     | professor           | 80000.00 |
|         108 | Robert      | cashier             | 15000.00 |
|         109 | amit        | clerk               | 20000.00 |
|         110 | george      | HR Manager          | 30000.00 |
|         111 | Jaishankar  | Project Manager     | 90000.00 |
|         112 | Kishore     | Insurance Agent     | 25000.00 |
+-------------+-------------+---------------------+----------+
13 rows in set (0.00 sec)

Query OK, 0 rows affected (0.08 sec)

mysql> call sp_InsertUpdateEmployee(112,'Kiran','Insurance Manager',35000);
Query OK, 1 row affected (0.03 sec)

mysql> call SP_getEmployee;
+-------------+-------------+---------------------+----------+
| employee_id | Name        | Designation         | Salary   |
+-------------+-------------+---------------------+----------+
|         100 | vishwanath  | clerk               | 20000.00 |
|         101 | shashikiran | instructor          | 20000.00 |
|         102 | nitin       | assitiant professor | 25000.00 |
|         103 | deepak      | associate professor | 40000.00 |
|         104 | sanjay      | professor           | 80000.00 |
|         105 | yogesh      | system admin        | 30000.00 |
|         106 | anand       | clerk               | 20000.00 |
|         107 | Hemanth     | professor           | 80000.00 |
|         108 | Robert      | cashier             | 15000.00 |
|         109 | amit        | clerk               | 20000.00 |
|         110 | george      | HR Manager          | 30000.00 |
|         111 | Jaishankar  | Project Manager     | 90000.00 |
|         112 | Kiran       | Insurance Manager   | 35000.00 |
+-------------+-------------+---------------------+----------+
13 rows in set (0.00 sec)
```

Anexampleofdeletingrecordintablenamedemployeebyusingstored procedure

//DELETESTOREDPROCEDURE

```
DELIMITER $$
DROPPROCEDUREIFEXISTSsp_DeletetEmployee$$
CREATE PROCEDURE sp_DeletetEmployee
(
INemp_idINT(10)
)
 BEGIN
Deletefromemployeewhereemployee_id=emp_id;
END$$
```

```
mysql> SELECT * FROM EMPLOYEE;
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
|         111 | Jaishankar  | project manager      | 90000.00 |
|         112 | arun        | Graphics Engineer    | 35000.00 |
|         113 | Frank       | Finance Manager      | 40000.00 |
+-------------+-------------+----------------------+----------+
14 rows in set (0.00 sec)

mysql> CALL  sp_DeletetEmployee(113);
Query OK, 1 row affected (0.13 sec)

mysql> SELECT * FROM EMPLOYEE;
+-------------+-------------+----------------------+----------+
| employee_id | Name        | Designation          | Salary   |
+-------------+-------------+----------------------+----------+
|         100 | vishwanath  | clerk                | 20000.00 |
|         101 | shashikiran | instructor           | 20000.00 |
|         102 | nitin       | assitiant professor  | 25000.00 |
|         103 | deepak      | associate professor  | 40000.00 |
|         104 | sanjay      | professor            | 80000.00 |
|         105 | yogesh      | system admin         | 30000.00 |
|         106 | anand       | clerk                | 20000.00 |
|         107 | Hemanth     | professor            | 80000.00 |
|         108 | Robert      | cashier              | 15000.00 |
|         109 | amit        | clerk                | 20000.00 |
|         110 | george      | HR Manager           | 30000.00 |
|         111 | Jaishankar  | project manager      | 90000.00 |
|         112 | arun        | Graphics Engineer    | 35000.00 |
+-------------+-------------+----------------------+----------+
13 rows in set (0.00 sec)

mysql> _
```

Variablesscope

Avariablehasitsownscopethatdefinesitslifetime.Ifyoudeclareavariableinsideastored procedure, it will be out of scope when the END statement ofstored procedure reaches.

If you declare a variable inside BEGIN END block, it will be out of scope if the END is reached. Youcandeclaretwoormorevariableswiththesame nameindifferent scopesbecauseavariableis only effective in its own scope. However, declaring variables with the same name in different scopes is not good programming practice.

Avariablewhosenamebeginswiththe@signisasessionvariable. Itisavailableandaccessible until the session ends.

DeclareaVariable:

DECLAREvar_name[,var_name]...type[DEFAULT value]

Toprovideadefault valueforavariable, includeaDEFAULTclause.Thevaluecanbespecified as an expression; it need not be constant. If the DEFAULT clause is missing, the initial value is NULL.

Example1:Localvariables

Localvariables are declared within storedprocedures and are onlyvalid within the BEGIN…END blockwheretheyaredeclared. LocalvariablescanhaveanySQLdatatype. Thefollowingexample shows the use of local variables in a stored procedure.

Herebelowisanexampleofstoredprocedurewhichisusedtoinsertaswellbothupdatecontents of employee table

```
DELIMITER $$
DROPPROCEDUREIFEXISTSSP_InsertUpdateemployee$$
 CREATE PROCEDURE SP_InsertUpdateemployee
(
INemp_idint,
INNameVARCHAR(255),
INDesignationVARCHAR(255),
IN Salary float(10,2)
)
BEGIN
DECLAREcountaint;
SELECTcount(*)INTOcountaFROMemployeeWHEREemployee_id=emp_id; IF
counta=0
THEN
INSERT INTO employee values(emp_id,Name,Designation,Salary);
ELSEIF counta>0
THEN
UPDATEEMPLOYEESETName=Name,Designation=Designation,Salary=Salary
WHERE employee_id=emp_id;
ENDIF;
END$$
```

TODROPSTORED PROCEDURE

OnceyouhavecreatedyourprocedureinMySQL,youmightfindthat youneedtoremoveit from the database. This statement is used to drop a stored procedure .

DROPprocedureIFEXISTSprocedure_name;

procedure_name: The name of the procedure that you wish to drop.'

ExampleofDropingStoredProcedure

dropprocedureIFEXISTSSP_getcustomerid; or

dropprocedureSP_getcustomerid;

MySQLstoredproceduresadvantages

- **Reduce Network Traffic** : Stored procedures help reduce the traffic between application and databaseserverbecauseinsteadofsendingmultiplelengthySQLstatements,theapplicationhas to send only the name and parameters of the stored procedure.
- **Faster Query Execution** : Since stored procedures are Parsed, Compiled at once, and the executableiscachedintheDatabase.Thereforeifsamequeryisrepeatedmultipletimesthen Database directly executes the executable and hence Time is saved in Parse,Compile etc.
- **Secure:** MySQL stored procedures are secure because the database administrator can grant appropriatepermissionstoapplicationsthataccessstoredproceduresinthedatabasewithout giving any permissions on the underlying database table

MySQLstoredproceduresdisadvantages

- Stored procedure's constructs are not designed for developing complex and flexible business logic.
- It isdifficult todebugstored procedures. Onlyafewdatabase management systemsallow you to debugstoredprocedures.Unfortunately,MySQLdoesnotprovidefacilitiesfordebuggingstored procedures.
- Memoryusage increased: Ifweuse manystoredprocedures,the memoryusageofevery connection that is using those stored procedures will increase substantially.

INTRODUCTIONTOMYSQLTRIGGER

- Atrigger isasetofSQLStatementsthatarerunautomaticallywhenaspecifiedchangeoperation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table.
- ASQLtriggerisaspecialtypeofstoredprocedure. Itisspecialbecauseit isnotcalleddirectly like a stored procedure. The main difference between a trigger and a stored procedure is that a trigger iscalledautomaticallywhenadatamodificationevent ismadeagainst atablewhereasa stored procedure must be called explicitly.
- Atriggercanbedefinedto beinvokedeither beforeorafterthedataischangedbyINSERT, UPDATE or DELETE statement
- Triggersareusefulfortaskssuchasenforcingbusinessrules,validatinginput data,andkeepingan audit trail.A trigger can be set to activate either before or after the trigger event.For example, you can have a trigger activate before each row that is inserted into a table or after each row that is updated
- It is important to understand the SQL trigger's advantages anddisadvantagessothatyoucanuseit appropriately. Inthefollowingsections,wewilldiscusstheadvantagesanddisadvantagesofusing SQL triggers.

AdvantagesofusingSQLtriggers

- SQLtriggersprovideanalternativewayto checkthe integrityofdata. •

  SQL triggers can catch errors in business logic in the database layer.
- SQLtriggersprovideanalternativewayto runscheduled tasks.Byusing SQLtri ggers, you don't havetowaittorunthescheduledtasksbecausethetriggersareinvokedautomaticallybeforeor after a change is made to the data in the tables.
- SQLtriggersareveryusefulto auditthechangesofdataintables.

DisadvantagesofusingSQLtriggers

- SQLtriggersonlycanprovideanextendedvalidationandtheycannot replaceallthevalidations. Somesimplevalidationshavetobedone intheapplication layer. Forexample, you canvalidate user's inputs in the client side by using JavaScriptorontheserver sideusingserver-sidescripting languagessuchasJSP,PHP,ASP.NET,Perl.
- SQLtriggersareinvokedandexecutedinvisiblefromtheclientapplications,therefore,it is difficult to figure out what happens in the database layer.
- SQLtriggersmayincreasetheoverhead ofthedatabase server.

InMySQL,triggercanalsobecreated.Thereare6typeoftriggersthatcanbemadetheyare:-

1. **After/Before insert**
2. **After/Beforeupdate**
3. **After/Before delete**

```
CREATETRIGGERtrigger_name
trigger_event ON table_name
FOREACHROW
BEGIN
...
END;
```

Here,

- Trigger_name is the name ofthe trigger which must be put after CREATE TRIGGER statement.
- Thenamingconventionfortrigger_namecanbelike[triggertime]_[tablename]_[triggerevent].For example, before_student_update or after_student_insert can be a name of the trigger.
- Trigger_time is the time of trigger activation and it can be BEFORE or AFTER. We must have to specifytheactivationtimewhiledefiningatrigger.WemusttouseBEFOREifwewanttoprocess action prior to the change made on the table and AFTER if we want to process action post to the change made on the table.
- Trigger_event canbeINSERT,UPDATEorDELETE.Thisevent causesthetriggertobeinvoked. A trigger onlycan be invoked byone event. To definea trigger that is invoked bymultiple events, we have to define multiple triggers, one for each event.
- Table_nameisthenameofthetable.Actually,atrigger isalwaysassociatedwithaspecifictable. Without a table, a trigger would not exist hence we have to specify the table name after the 'ON'keyword.
- BEGIN…END is the block in which we will define the logic for the trigger.

AFTER/BEFOREINSERTTRIGGER

```
CREATE TRIGGER trigger_name
   AFTER/BEFOREINSERTONtable_name
FOREACHROW
      BEGIN
     --variabledeclarations
     --triggercode
     END;
```

Parameter:

- trigger_name:nameofthetriggertobecreated.
- AFTER/BEFOREINSERT:Itpointsthetriggerafterorbeforeinsertqueryisexecuted.
- table_name: name of the table in which a trigger is created.

AFTER/BEFOREUPDATETrigger

- InMySQL,AFTER/BEFOREUPDATEtrigger canalsobecreated.
- AFTER/BEFOREUPDATEtriggermeanstriggerwillinvokeafter/beforetherecordis updated.

Syntax:

```
CREATE TRIGGER trigger_name
   AFTER/BEFORE UPDATE    ONtable_name
 FOREACHROW
       BEGIN
      --variabledeclarations
      --triggercode
      END;
```

Parameter:

- trigger_name:nameofthetriggertobecreated.
- AFTERUPDATE:Itpointsthetriggerupdatequeryisexecuted. •
  table_name: name of the table in which a trigger is created.

AFTER/BEFOREDELETETrigger

- InMySQL,AFTER/BEFOREDELETEtriggercanalsobecreated.
- AFTER/BEFOREDELETEtrigger meanstrigger willinvokeafter/beforetherecordisdeleted.

Syntax:

```
CREATE TRIGGER trigger_name
   AFTER/BEFOREDELETEONtable_name
FOREACHROW
       BEGIN
      --variabledeclarations
      --triggercode
      END;
```

Parameter:

- trigger_name: nameofthetrigger tobecreated.
- AFTER/BEFOREDELETE:Itpointsthetriggerafter/beforedeletequeryisexecuted. •
  table_name: name of the table in which a trigger is created.

LABEXPERIMENTS
PARTA:SQLPROGRAMMING

A. ConsiderthefollowingschemaforaLibraryDatabase:

BOOK (Book_id, Title, Publisher_Name, Pub_Year)

BOOK_AUTHORS (Book_id, Author_Name)

PUBLISHER (Name, Address, Phone)

BOOK_COPIES(Book_id,Programme_id,No-of_Copies)

BOOK_LENDING(Book_id,Programme_id,Card_No,Date_Out,Due_Date)

LIBRARY_PROGRAMME(Programme_id, Programme_Name,Address)

Write SQL queries to

1. **Retrievedetailsofallbooksinthelibrary –id,title,nameofpublisher,authors, number of copies in each Programme, etc.**
2. **Gettheparticularsofborrowerswhohaveborrowedmorethan3books,butfrom Jan 2017 to Jun 2017**
3. **DeleteabookinBOOKtable.Updatethecontentsofothertablestoreflectthisdata manipulation operation.**
4. **PartitiontheBOOKtablebasedonyearofpublication.Demonstrateitsworking with a simple query.**
5. **Createaviewofallbooksanditsnumberofcopiesthatarecurrentlyavailablein the Library.**

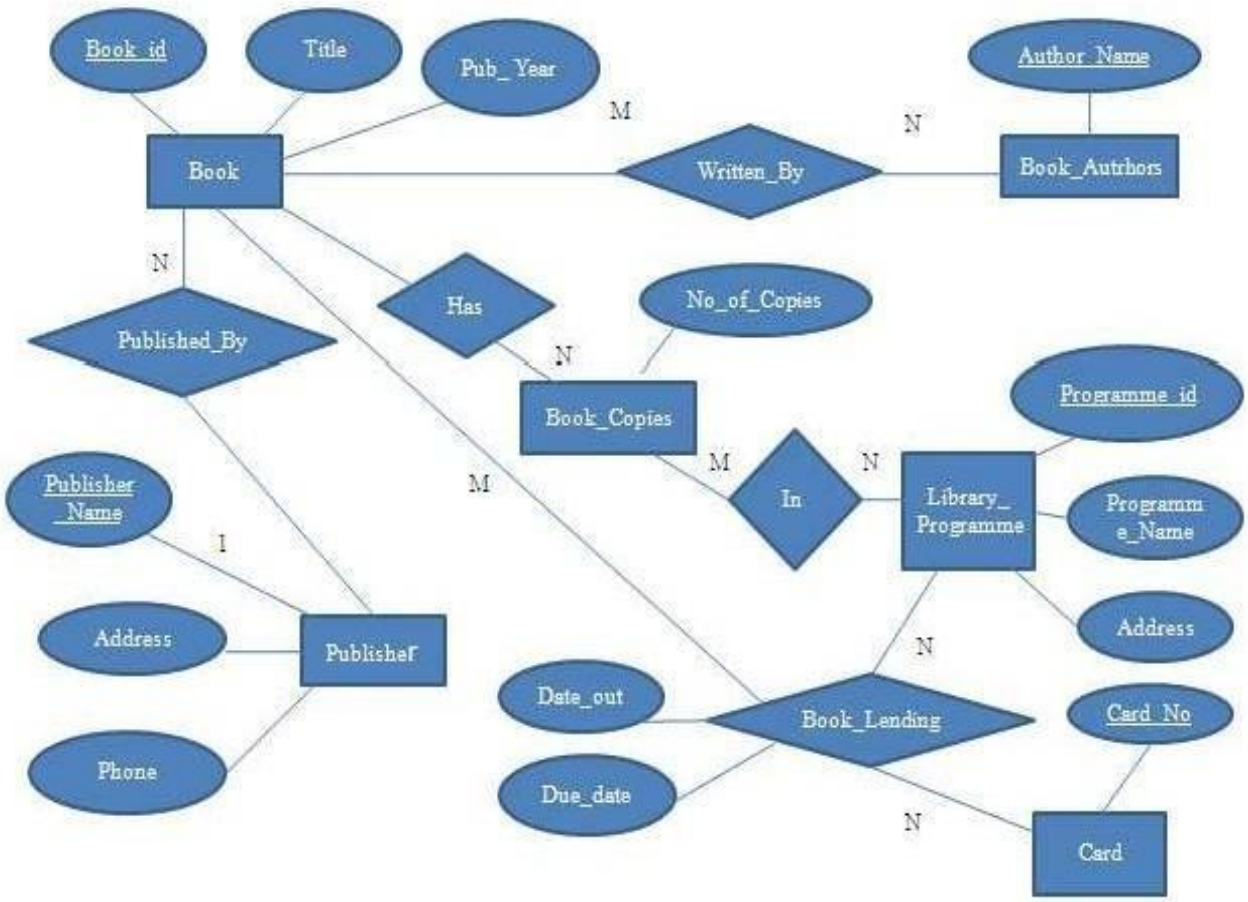
ProgramObjectives:

Thiscoursewillenablestudentsto

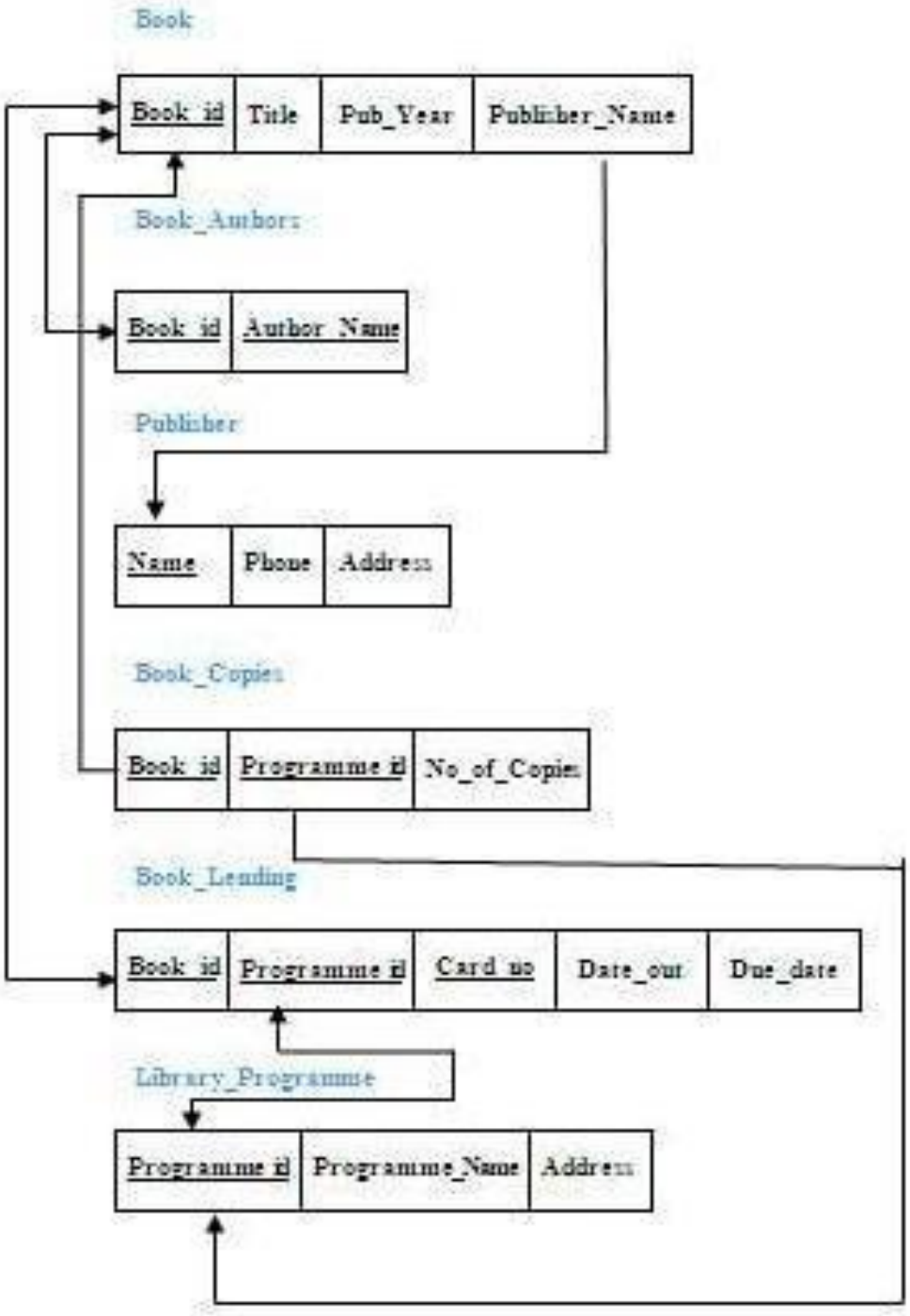
- Foundationknowledgeindatabaseconcepts,technologyandpracticetogroomstudents into well-informed database application developers.
- Strongpractice inSQLprogrammingthroughavarietyofdatabaseproblems. Develop
- database applications using front-end tools and back-end DBMS.

Solution:

Entity-RelationshipDiagram



SchemaDiagram



TableCreation

```
CREATE TABLE BOOK
(BOOK_IDINT(10)PRIMARYKEY,
TITLE VARCHAR (20),
PUB_YEARVARCHAR(20),
PUBLISHER_NAMEVARCHAR(20),
FOREIGNKEY(PUBLISHER_NAME)REFERENCESPUBLISHER(NAME)ONDELETE
CASCADE);


CREATETABLEBOOK_AUTHORS(
AUTHOR_NAME VARCHAR (20),
BOOK_IDINT(10),
PRIMARYKEY(BOOK_ID,AUTHOR_NAME),
FOREIGNKEY(BOOK_ID)REFERENCESBOOK(BOOK_ID)ONDELETECASCADE);

CREATETABLEPUBLISHER(
NAMEVARCHAR(20)PRIMARYKEY,
PHONE BIGINT (20),
ADDRESSVARCHAR(100));

CREATETABLEBOOK_COPIES(
NO_OF_COPIES INT (5),
BOOK_IDINT(10),
PROGRAMME_IDINT(10),
PRIMARYKEY(BOOK_ID,PROGRAMME_ID),
FOREIGNKEY(BOOK_ID)REFERENCESBOOK(BOOK_ID)ONDELETECASCADE,
FOREIGN KEY (PROGRAMME_ID) REFERENCES LIBRARY_PROGRAMME
(PROGRAMME_ID) ON DELETE CASCADE);

CREATETABLEBOOK_LENDING(
DATE_OUT DATE,
DUE_DATEDATE,
BOOK_IDINT(10),
PROGRAMME_IDINT(10),
CARD_NOINT (10),
PRIMARYKEY(BOOK_ID,PROGRAMME_ID,CARD_NO),
FOREIGNKEY(BOOK_ID)REFERENCESBOOK(BOOK_ID)ONDELETECASCADE,
FOREIGN KEY (PROGRAMME_ID) REFERENCES
LIBRARY_PROGRAMME(PROGRAMME_ID) ON DELETE CASCADE,
FOREIGNKEY(CARD_NO)REFERENCESCARD(CARD_NO)ONDELETECASCADE);

CREATETABLECARD(
CARD_NOINT(10)PRIMARYKEY);
```

CREATETABLELIBRARY_PROGRAMME(
PROGRAMME_ID INT (10) PRIMARY KEY,
PROGRAMME_NAME VARCHAR (50),
ADDRESSVARCHAR(100));

TableDescriptions

DESCBOOK
;

```
mysql> DESC BOOK;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| BOOK_ID        | int(10)     | NO   | PRI | NULL    |       |
| TITLE          | varchar(20) | YES  |     | NULL    |       |
| PUB_YEAR       | varchar(20) | YES  |     | NULL    |       |
| PUBLISHER_NAME | varchar(20) | YES  | MUL | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

DESCBOOK_AUTHORS;

```
mysql> DESC BOOK_AUTHORS;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| AUTHOR_NAME | varchar(20) | NO   | PRI |         |       |
| BOOK_ID     | int(10)     | NO   | PRI | 0       |       |
+-------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

DESCPUBLISHER;

```
mysql> DESC PUBLISHER;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| NAME    | varchar(20)  | NO   | PRI | NULL    |       |
| PHONE   | bigint(20)   | YES  |     | NULL    |       |
| ADDRESS | varchar(100) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESCBOOK_COPIES

```
mysql> DESC  BOOK_COPIES;
+--------------+---------+------+-----+---------+-------+
| Field        | Type    | Null | Key | Default | Extra |
+--------------+---------+------+-----+---------+-------+
| NO_OF_COPIES | int(5)  | YES  |     | NULL    |       |
| BOOK_ID      | int(10) | NO   | PRI | NULL    |       |
| PROGRAMME_ID | int(10) | NO   | PRI | NULL    |       |
+--------------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

DESCBOOK_LENDING;

```
mysql> DESC BOOK_LENDING;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| DATE_OUT    | date    | YES  |     | NULL    |       |
| DUE_DATE    | date    | YES  |     | NULL    |       |
| BOOK_ID     | int(10) | NO   | PRI | NULL    |       |
| PROGRAMME_ID| int(10) | NO   | PRI | NULL    |       |
| CARD_NO     | int(10) | NO   | PRI | NULL    |       |
+-------------+---------+------+-----+---------+-------+
5 rows in set (0.03 sec)

mysql>
```

DESCCARD;

```
mysql> DESC CARD;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| CARD_NO | int(10) | NO   | PRI | NULL    |       |
+---------+---------+------+-----+---------+-------+
1 row in set (0.00 sec)

mysql>
```

DESCLIBRARY_PROGRAMME

```
mysql> DESC LIBRARY_PROGRAMME;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| PROGRAMME_ID   | int(10)     | NO   | PRI | NULL    |       |
| PROGRAMME_NAME | varchar(50) | YES  |     | NULL    |       |
| ADDRESS        | varchar(100)| YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

InsertionofValuestoTables

```
INSERT INTO BOOK VALUES (1,'DBMS','JAN-2017', 'MCGRAW-HILL');
INSERTINTOBOOKVALUES(2,'ADBMS','JUN-2016','MCGRAW-HILL');
INSERT INTO BOOK VALUES (3, 'CD','SEP-2016','PEARSON');
INSERTINTOBOOKVALUES(4,'ALGORITHMS','SEP-2015','MIT'); INSERT
INTO BOOK VALUES (5,'OS','MAY-2016','PEARSON');

INSERTINTOBOOK_AUTHORSVALUES('NAVATHE',1);
INSERTINTOBOOK_AUTHORSVALUES('NAVATHE',2);
INSERT INTO BOOK_AUTHORS VALUES ('ULLMAN',3);
INSERT INTO BOOK_AUTHORS VALUES ('CHARLES', 4);
INSERT INTOBOOK_AUTHORS VALUES('GALVIN', 5);


INSERTINTOPUBLISHERVALUES('MCGRAW-HILL',9989076587,'BANGALORE');
INSERT INTOPUBLISHER VALUES ('PEARSON', 9889076565,'NEWDELHI');
INSERTINTOPUBLISHERVALUES('PRENTICEHALL',7455679345,'HYEDRABAD');
INSERT INTOPUBLISHER VALUES ('WILEY', 8970862340,'CHENNAI');
INSERTINTOPUBLISHERVALUES('MIT',7756120238,'BANGALORE');

INSERTINTOBOOK_COPIESVALUES(10,1,10);
INSERTINTOBOOK_COPIESVALUES(5,1,11);
INSERTINTOBOOK_COPIESVALUES(2,2,12);
INSERTINTOBOOK_COPIESVALUES(5,2,13);
INSERTINTOBOOK_COPIESVALUES(7,3,14);
INSERTINTOBOOK_COPIESVALUES(1,5,10);
INSERTINTOBOOK_COPIESVALUES(3,4,11);

INSERTINTOBOOK_LENDINGVALUES('2017-01-01','2017-06-01',1,10,101);
INSERTINTOBOOK_LENDINGVALUES('2017-01-11','2017-03-11',3,14, 101);
INSERTINTOBOOK_LENDINGVALUES('2017-02-21','2017-04-21',2,13,101);
INSERTINTOBOOK_LENDINGVALUES('2017-03-15','2017-07-15',4,11,101);
INSERTINTOBOOK_LENDINGVALUES('2017-04-12','2017-05-12',1,11,104);

INSERTINTOCARDVALUES(100);
INSERTINTOCARDVALUES(101);
INSERTINTOCARDVALUES(102);
INSERTINTOCARDVALUES(103);
INSERTINTOCARDVALUES(104);

INSERT INTO LIBRARY_PROGRAMME VALUES (10,'VIJAY NAGAR','MYSURU');
INSERT INTO LIBRARY_PROGRAMME VALUES (11,'VIDYANAGAR','HUBLI'); ;
INSERTINTOLIBRARY_PROGRAMMEVALUES(12,'KUVEMPUNAGAR','MYSURU');
INSERT INTO LIBRARY_PROGRAMME VALUE(13,'RAJAJINAGAR','BANGALORE');
INSERT INTOLIBRARY_PROGRAMME VALUES (14,'MANIPAL','UDUPI');
```

SELECT*FROMBOOK;

BOOK_ID	TITLE	PUB_YEAR	PUBLISHER_NAME
1	DBMS	Jan-2017	MCGRAW-HILL
2	ADBMS	Jun-2017	MCGRAW-HILL
3	CD	Sep-2016	PEARSON
4	ALGORITHMS	Sep-2015	MIT
5	OS	May-2016	PEARSON

SELECT*FROMBOOK_AUTHORS;

AUTHOR_NAME	BOOK_ID
NAVATHE	1
NAVATHE	2
ULLMAN	3
CHARLES	4
GALVIN	5

SELECT*FROMPUBLISHER;

NAME	PHONE	ADDRESS
MCGRAW-HILL	9989076587	BANGALORE
MIT	7756120238	BANGALORE
PEARSON	9889076565	NEWDELHI
PRENTICEHALL	7455679345	HYEDRABAD
WILEY	8970862340	CHENNAI

SELECT*FROMBOOK_COPIES;

NO_OF_COPIES	BOOK_ID	PROGRAMME_ID
10	1	10
5	1	11
2	2	12
5	2	13
7	3	14
1	5	10
3	4	11

SELECT*FROMBOOK_LENDING;

DATEOUT	DUEDATE	BOOKID	PROGRAMME_ID	CARDNO
2017-01-01	2017-06-01	1	10	
2017-01-11	2017-03-11	3	4	101
2017-02-21	2017-04-21	2	13	101
2017-03-15	2017-07-15	4	11	101
2017-04-12	2017-05-12	1	11	104

SELECT*FROM CARD;

CARDNO
101
102
103
104
105

SELECT*FROMLIBRARY_PROGRAMME;

PROGRAMME_ID	PROGRAMME_NAME	ADDRESS
10	VIJAYNAGAR	MYSURU
11	VIDYANAGAR	HUBLI
12	KUVEMPUNAGAR	MYSURU
13	RAJAJINAGAR	BANGALORE
14	MANIPAL	UDUPI

Queries:

1. **Retrievedetailsofallbooksinthelibrary –id,title,nameofpublisher,authors,number of copies in each branch, etc.**

SELECT B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME, A.AUTHOR_NAME, C.NO_OF_COPIES,L.PROGRAMME_IDFROMBOOKB,BOOK_AUTHORSA,BOOK_COPIES C, LIBRARY_PROGRAMME LWHERE B.BOOK_ID=A.BOOK_ID AND B.BOOK_ID=C.BOOK_ID AND L.PROGRAMME_ID=C.PROGRAMME_ID;

BOOK_ ID	TITLE	PUBLISHER_ NAME	AUTHOR_ NAME	NO_ OF_COPIES	PROGRAMME _ID
1	DBMS	MCGRAW-HILL	NAVATHE	10	10
1	DBMS	MCGRAW-HILL	NAVATHE	5	11
2	ADBMS	MCGRAW-HILL	NAVATHE	2	12
2	ADBMS	MCGRAW-HILL	NAVATHE	5	13
3	CD	PEARSON	ULLMAN	7	14
4	ALGORITHMS	MIT	CHARLES	1	11
5	OS	PEARSON	GALVIN	3	10

2. **Gettheparticularsofborrowerswhohaveborrowedmorethan3books,butfromJan 2017 to Jun 2017.**

SELECTCARD_NOFROMBOOK_LENDINGWHEREDATE_OUT BETWEEN'2017-01-01'AND'2017-07-01'GROUPBYCARD_NO HAVING COUNT(*)>3;

3. **DeleteabookinBOOKtable.Updatethecontentsofothertablestoreflectthisdata manipulation operation.**

DELETEFROMBOOKWHEREBOOK_ID=3;

```
mysql> SELECT * FROM BOOK;
+---------+------------+----------+-----------------+
| BOOK_ID | TITLE      | PUB_YEAR | PUBLISHER_NAME |
+---------+------------+----------+-----------------+
|       1 | DBMS       | JAN-2017 | MCGRAW-HILL     |
|       2 | ADBMS      | JUN-2016 | MCGRAW-HILL     |
|       3 | CD         | SEP-2016 | PEARSON         |
|       4 | ALGORITHMS | SEP-2015 | MIT             |
|       5 | OS         | MAY-2016 | PEARSON         |
+---------+------------+----------+-----------------+
5 rows in set (0.00 sec)

mysql> DELETE FROM BOOK WHERE BOOK_ID=3;
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM BOOK;
+---------+------------+----------+-----------------+
| BOOK_ID | TITLE      | PUB_YEAR | PUBLISHER_NAME |
+---------+------------+----------+-----------------+
|       1 | DBMS       | JAN-2017 | MCGRAW-HILL     |
|       2 | ADBMS      | JUN-2016 | MCGRAW-HILL     |
|       4 | ALGORITHMS | SEP-2015 | MIT             |
|       5 | OS         | MAY-2016 | PEARSON         |
+---------+------------+----------+-----------------+
4 rows in set (0.00 sec)
```

4. **PartitiontheBOOKtablebasedonyearofpublication.Demonstrateitsworkingwitha simple query.**

CREATEVIEWVW_PUBLICATIONASSELECTPUB_YEARFROMBOOK;

SELECT * FROM VW_PUBLICATION

```
mysql> SELECT * FROM VW_PUBLICATION;
+----------+
| PUB_YEAR |
+----------+
| JAN-2017 |
| JUN-2016 |
| SEP-2016 |
| SEP-2015 |
| MAY-2016 |
+----------+
5 rows in set (0.00 sec)
```

5. **Createaviewofallbooksanditsnumberofcopiesthatarecurrentlyavailableinthe Library.**
CREATEVIEWVW_BOOKSASSELECTB.BOOK_ID,B.TITLE,C.NO_OF_COPIES FROM BOOK B, BOOK_COPIES C, LIBRARY_PROGRAMME L WHERE B.BOOK_ID=C.BOOK_ID AND C.PROGRAMME_ID=L.PROGRAMME_ID;

SELECT*FROMVW_BOOKS;

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_library  |
+--------------------+
| book               |
| book_authors       |
| book_copies        |
| book_lending       |
| card               |
| library_programme  |
| publisher          |
| vw_books           |
+--------------------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM  VW_BOOKS;
+---------+--------------+---------------+
| BOOK_ID | TITLE        | NO_OF_COPIES  |
+---------+--------------+---------------+
|       1 | DBMS         |            10 |
|       1 | DBMS         |             5 |
|       2 | ADBMS        |             2 |
|       2 | ADBMS        |             5 |
|       3 | CD           |             7 |
|       5 | OS           |             1 |
|       4 |  ALGORITHMS  |             3 |
+---------+--------------+---------------+
7 rows in set (0.00 sec)

mysql>
```

ProgramOutcomes:

Thestudentsareableto

- Create, Updateand queryonthedatabase.
- DemonstratetheworkingofdifferentconceptsofDBMS
- Implement,analyzeandevaluatetheprojectdevelopedforanapplication.

B. Consider the following schema for Order Database:

SALESMAN(Salesman_id,Name,City,Commission)
CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)
ORDERS(Ord_No,Purchase_Amt,Ord_Date,Customer_id,Salesman_id)

WriteSQLqueriesto

1. **Count the customers with grades above Bangalore's average.**
2. **Findthenameand numbersofallsalesmenwhohadmorethanonecustomer.**
3. **List all salesmen and indicate those who have and don't have customers in their cities(UseUNIONoperation.)**
4. **Createaviewthatfindsthesalesmanwhohasthecustomerwiththehighestorder of a day.**
5. **DemonstratetheDELETEoperationbyremovingsalesmanwithid1000.Allhis orders must also be deleted.**

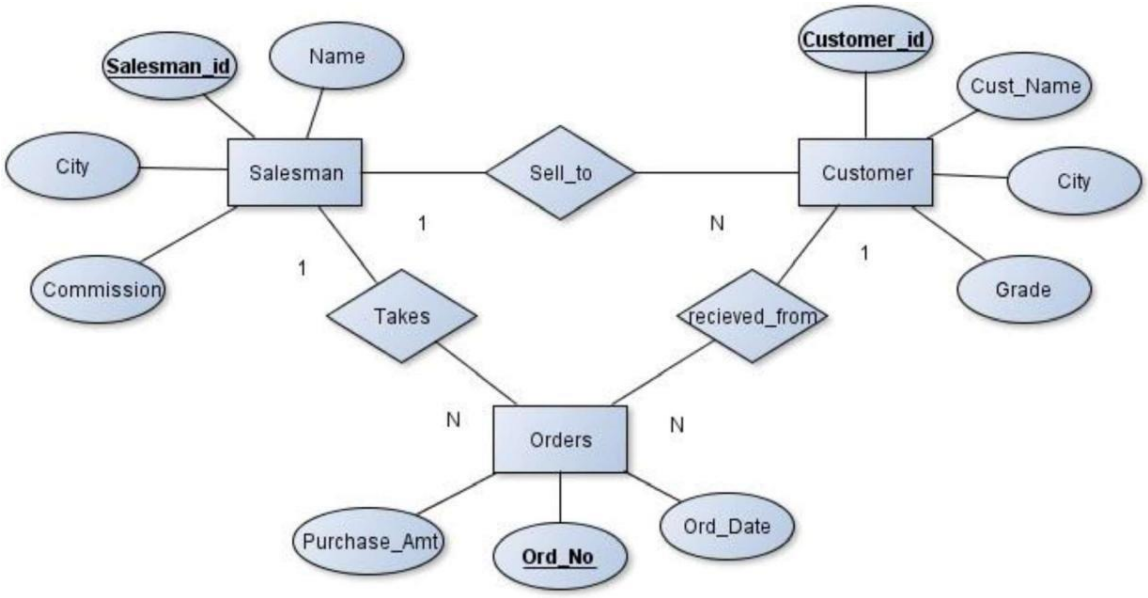
ProgramObjectives:

Thiscoursewillenablestudentsto

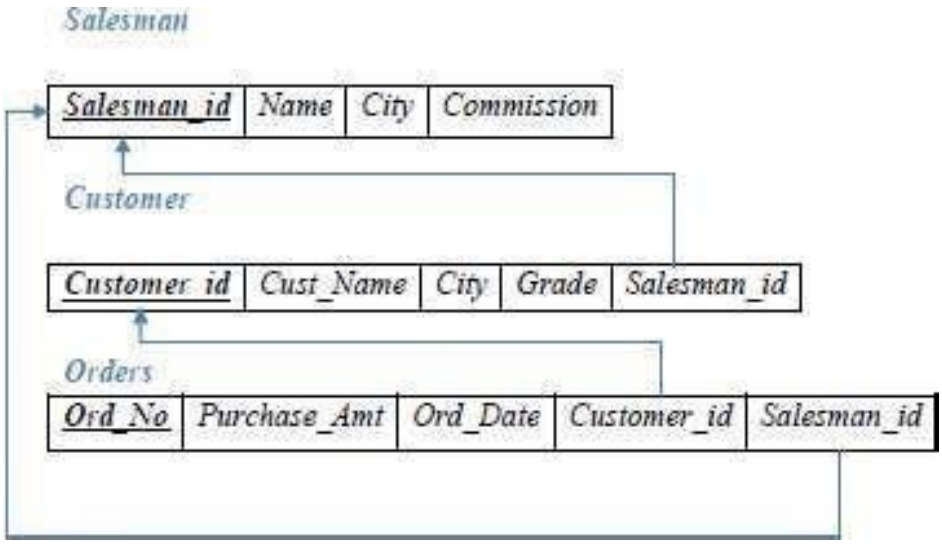
- Foundationknowledgeindatabaseconcepts,technologyandpracticetogroomstudents into well-informed database application developers.
- Strongpractice inSQLprogrammingthroughavarietyofdatabaseproblems. Develop
- database applications using front-end tools and back-end DBMS.

Solution:

Entity-RelationshipDiagram



SchemaDiagram



TableCreation

```
CREATE TABLE SALESMAN (
SALESMAN_IDINT(4)PRIMARYKEY,
NAME VARCHAR (20),
CITYVARCHAR (20),
COMMISSIONVARCHAR(20));

CREATE TABLE CUSTOMER (
CUSTOMER_IDINT(5)PRIMARYKEY,
CUST_NAME VARCHAR (20),
CITYVARCHAR (20),GRADEINT(4),
SALESMAN_IDINT(6),
FOREIGNKEY(SALESMAN_ID)REFERENCESSALESMAN(SALESMAN_ID)ONDELETE
SET NULL);

CREATE TABLE ORDERS (
ORD_NO INT (5) PRIMARY KEY,
PURCHASE_AMTDECIMAL(10,2),
ORD_DATE DATE,
CUSTOMER_IDINT(4),
SALESMAN_IDINT(4),
FOREIGNKEY(CUSTOMER_ID)REFERENCESCUSTOMER(CUSTOMER_ID)ONDELETE
CASCADE,
FOREIGNKEY(SALESMAN_ID)REFERENCESSALESMAN(SALESMAN_ID)ONDELETE
CASCADE);
```

TableDescriptions

DESCSALESMAN;

```
mysql> DESC SALESMAN;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| SALESMAN_ID | int(4)      | NO   | PRI | NULL    |       |
| NAME        | varchar(20) | YES  |     | NULL    |       |
| CITY        | varchar(20) | YES  |     | NULL    |       |
| COMMISSION  | varchar(20) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

DESCCUSTOMER;

```
mysql> DESC CUSTOMER;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| CUSTOMER_ID | int(5)      | NO   | PRI | NULL    |       |
| CUST_NAME   | varchar(20) | YES  |     | NULL    |       |
| CITY        | varchar(20) | YES  |     | NULL    |       |
| GRADE       | int(4)      | YES  |     | NULL    |       |
| SALESMAN_ID | int(6)      | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

DESC ORDERS;

```
mysql> DESC ORDERS;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| ORD_NO       | int(5)       | NO   | PRI | NULL    |       |
| PURCHASE_AMT | decimal(10,2)| YES  |     | NULL    |       |
| ORD_DATE     | date         | YES  |     | NULL    |       |
| CUSTOMER_ID  | int(4)       | YES  | MUL | NULL    |       |
| SALESMAN_ID  | int(4)       | YES  | MUL | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

INSERT INTO SALESMAN VALUES(101,'RICHARD','LOS ANGELES','18%');
INSERT INTO SALESMAN VALUES(103,'GEORGE','NEWYORK','32%');
INSERT INTO SALESMAN VALUES(110,'CHARLES','BANGALORE','54%');
INSERTINTOSALESMANVALUES(122,'ROWLING','PHILADELPHIA','46%');
INSERT INTOSALESMAN VALUES(126,'KURT','CHICAGO','52%');
INSERTINTOSALESMANVALUES(132,'EDWIN','PHOENIX','41%');

INSERT INTO CUSTOMER VALUES(501,'SMITH','LOS ANGELES',10,103);
INSERT INTO CUSTOMER VALUES(510,'BROWN','ATLANTA',14,122);
INSERT INTO CUSTOMER VALUES(522,'LEWIS','BANGALORE',10,132);
INSERT INTO CUSTOMER VALUES(534,'PHILIPS','BOSTON',17,103);
INSERTINTOCUSTOMERVALUES(543,'EDWARD','BANGALORE',14,110);
INSERT INTOCUSTOMER VALUES(550,'PARKER','ATLANTA',19,126);

INSERT INTO ORDERS VALUES(1,1000, '2017-05-04',501,103);
INSERT INTO ORDERS VALUES(2,4000,'2017-01-20',522,132);
INSERT INTO ORDERS VALUES(3,2500, '2017-02-24',550,126);
INSERT INTO ORDERS VALUES(5,6000,'2017-04-13',522,103);
INSERT INTO ORDERS VALUES(6,7000, '2017-03-09',550,126);
INSERTINTOORDERSVALUES(7,3400,'2017-01-20',501,122);

SELECT*FROMSALESMAN;

```
mysql> SELECT * FROM SALESMAN;
+-------------+---------+-------------+------------+
| SALESMAN_ID | NAME    | CITY        | COMMISSION |
+-------------+---------+-------------+------------+
|         101 | RICHARD | LOS ANGELES | 18%        |
|         103 | GEORGE  | NEWYORK     | 32%        |
|         110 | CHARLES | BANGALORE   | 54%        |
|         122 | ROWLING | PHILADELPHIA| 46%        |
|         126 | KURT    | CHICAGO     | 52%        |
|         132 | EDWIN   | PHOENIX     | 41%        |
+-------------+---------+-------------+------------+
6 rows in set (0.00 sec)
```

SELECT*FROMCUSTOMER;

```
mysql> SELECT * FROM CUSTOMER;
+-------------+-----------+-------------+-------+-------------+
| CUSTOMER_ID | CUST_NAME | CITY        | GRADE | SALESMAN_ID |
+-------------+-----------+-------------+-------+-------------+
|         501 | SMITH     | LOS ANGELES |    10 |         103 |
|         510 | BROWN     | ATLANTA     |    14 |         122 |
|         522 | LEWIS     | BANGALORE   |    10 |         132 |
|         534 | PHILIPS   | BOSTON      |    17 |         103 |
|         543 | EDWARD    | BANGALORE   |    14 |         110 |
|         550 | PARKER    | ATLANTA     |    19 |         126 |
+-------------+-----------+-------------+-------+-------------+
6 rows in set (0.00 sec)
```

SELECT*FROMORDERS;

```
mysql> select * from orders;
+--------+--------------+------------+-------------+-------------+
| ORD_NO | PURCHASE_AMT | ORD_DATE   | CUSTOMER_ID | SALESMAN_ID |
+--------+--------------+------------+-------------+-------------+
|      1 |      1000.00 | 2017-05-04 |         501 |         103 |
|      2 |      4000.00 | 2017-01-20 |         522 |         132 |
|      3 |      2500.00 | 2017-02-24 |         550 |         126 |
|      5 |      6000.00 | 2017-04-13 |         522 |         103 |
|      6 |      7000.00 | 2017-03-09 |         550 |         126 |
|      7 |      3400.00 | 2017-01-20 |         501 |         122 |
+--------+--------------+------------+-------------+-------------+
6 rows in set (0.00 sec)
```

Queries

1. **Count the customers with grades above Bangalore's average.**

```
SELECTGRADE,COUNT(CUSTOMER_ID)FROM
CUSTOMER GROUP BY GRADE
HAVINGGRADE>(SELECTAVG(GRADE)FROM CUSTOMER
WHERE CITY='BANGALORE');
```

```
+-------+-------------------------------+
| GRADE | COUNT(DISTINCT CUSTOMER_ID)   |
+-------+-------------------------------+
|    14 |                             2 |
|    17 |                             1 |
|    19 |                             1 |
+-------+-------------------------------+
3 rows in set (0.03 sec)
```

2. **Findthenameand numbersofallsalesmenwhohadmorethan onecustomer.**

```
SELECTSALESMAN_ID,NAME
FROM SALESMAN A
WHERE1<(SELECTCOUNT(*)FROMCUSTOMER
WHERE SALESMAN_ID=A.SALESMAN_ID)
```

OR

```
SELECTS.SALESMAN_ID,NAME,FROMCUSTOMER
C,SALESMAN S WHERE
S.SALESMAN_ID=C.SALESMAN_ID GROUP BY
C.SALESMAN_ID HAVING COUNT(*)>1
```

```
+-------------+--------+
| SALESMAN_ID | NAME   |
+-------------+--------+
|         103 | GEORGE |
+-------------+--------+
1 row in set (0.00 sec)
```

3. **List all salesmen and indicate those who have and don't have customers in their cities (UseUNIONoperation.)**

```
SELECT S.SALESMAN_ID,NAME,CUST_NAME,COMMISSION FROM SALESMAN
S,CUSTOMER C
WHERES.CITY=C.CITY
UNION
SELECTSALESMAN_ID,NAME,'NOMATCH',COMMISSIONFROMSALESMAN WHERE
NOT CITY = ANY (SELECT CITY
FROMCUSTOMER)ORDERBY2 DESC;
```

```
+-------------+---------+-----------+------------+
| SALESMAN_ID | NAME    | CUST_NAME | COMMISSION |
+-------------+---------+-----------+------------+
|         122 | ROWLING | NO MATCH  | 46%        |
|         101 | RICHARD | SMITH     | 18%        |
|         126 | KURT    | NO MATCH  | 52%        |
|         103 | GEORGE  | NO MATCH  | 32%        |
|         132 | EDWIN   | NO MATCH  | 41%        |
|         110 | CHARLES | LEWIS     | 54%        |
|         110 | CHARLES | EDWARD    | 54%        |
+-------------+---------+-----------+------------+
7 rows in set (0.03 sec)
```

4. **Createaviewthatfindsthesalesmanwhohasthecustomerwiththehighestorderofa day.**
   CREATEVIEWVW_ELITSALESMANAS
   SELECTB.ORD_DATE,A.SALESMAN_ID,A.NAMEFROMSALESMANA,ORDERSB
   WHERE A.SALESMAN_ID = B.SALESMAN_ID AND B.PURCHASE_AMT=(SELECT
   MAX(PURCHASE_AMT) FROM ORDERS C
   WHEREC.ORD_DATE=B.ORD_DATE);

   SELECT *FROMVW_ELITSALESMAN

```
mysql> SELECT * FROM  VW_ELITSALESMAN;
+------------+--------------+----------+
| ORD_DATE   | SALESMAN_ID  | NAME     |
+------------+--------------+----------+
| 2017-05-04 |          103 | GEORGE   |
| 2017-01-20 |          132 | EDWIN    |
| 2017-02-24 |          126 | KURT     |
| 2017-04-13 |          103 | GEORGE   |
| 2017-03-09 |          126 | KURT     |
+------------+--------------+----------+
5 rows in set (0.00 sec)
```

5. **DemonstratetheDELETEoperationbyremovingsalesmanwithid1000.Allhisorders must also be deleted.**
   UseONDELETECASCADEattheendofforeignkeydefinitionswhilecreatingchildtable orders and then execute the following:

   DELETEFROMSALESMANWHERESALESMAN_ID=101;

```
mysql> SELECT * FROM SALESMAN;
+-------------+---------+-------------+------------+
| SALESMAN_ID | NAME    | CITY        | COMMISSION |
+-------------+---------+-------------+------------+
|         101 | RICHARD | LOS ANGELES | 18%        |
|         103 | GEORGE  | NEWYORK     | 32%        |
|         110 | CHARLES | BANGALORE   | 54%        |
|         122 | ROWLING | PHILADELPHIA| 46%        |
|         126 | KURT    | CHICAGO     | 52%        |
|         132 | EDWIN   | PHOENIX     | 41%        |
+-------------+---------+-------------+------------+
6 rows in set (0.02 sec)

mysql> DELETE FROM SALESMAN WHERE SALESMAN_ID=101;
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM SALESMAN;
+-------------+---------+-------------+------------+
| SALESMAN_ID | NAME    | CITY        | COMMISSION |
+-------------+---------+-------------+------------+
|         103 | GEORGE  | NEWYORK     | 32%        |
|         110 | CHARLES | BANGALORE   | 54%        |
|         122 | ROWLING | PHILADELPHIA| 46%        |
|         126 | KURT    | CHICAGO     | 52%        |
|         132 | EDWIN   | PHOENIX     | 41%        |
+-------------+---------+-------------+------------+
5 rows in set (0.00 sec)
```

ProgramOutcomes:

Thestudentsareableto

- Create,Updateandqueryonthedatabase.
- DemonstratetheworkingofdifferentconceptsofDBMS
- Implement,analyzeandevaluatetheprojectdevelopedforanapplication.

C. Consider the schema for Movie Database:

ACTOR (Act_id, Act_Name, Act_Gender)
DIRECTOR(Dir_id,Dir_Name,Dir_Phone)
MOVIES(Mov_id,Mov_Title,Mov_Year,Mov_Lang,Dir_id)
MOVIE_CAST (Act_id, Mov_id, Role)
RATING(Mov_id,Rev_Stars)WriteSQLqueries to

1. **List the titles of all movies directed by'Hitchcock'.**
2. **Findthemovienameswhereoneormoreactorsacted intwoormoremovies.**
3. **Listallactorswhoacted inamoviebefore2000andalsoinamovieafter2015 (use JOIN operation).**
4. **Findthetitleof movies andnumberof starsforeachmoviethathasatleast one ratingandfindthehighestnumberof starsthatmoviereceived.Sorttheresultby movietitle.**
5. **Update rating of all movies directed by 'Steven Spielberg' to5.**

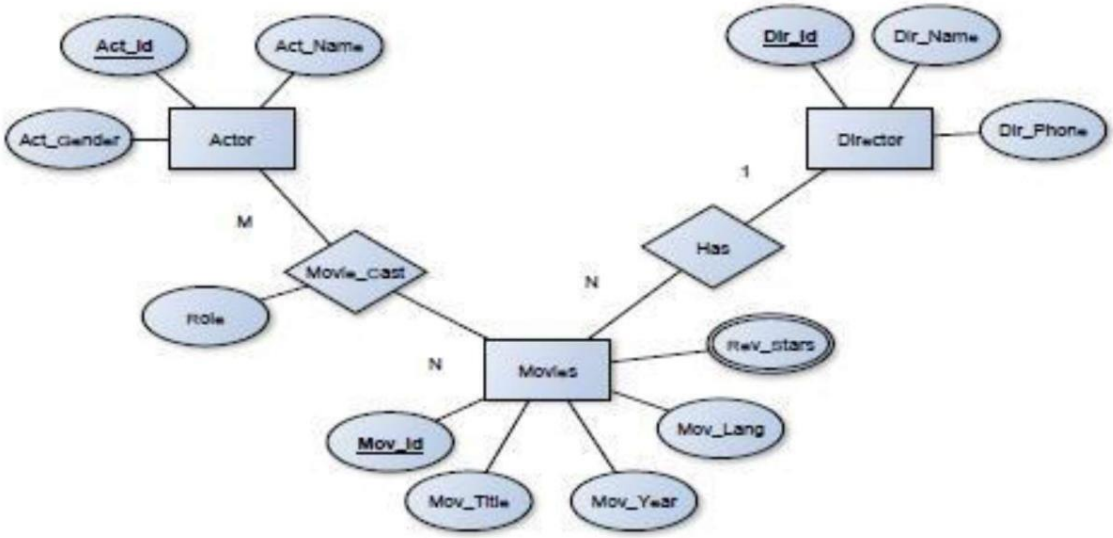
ProgramObjectives:

Thiscoursewillenablestudentsto

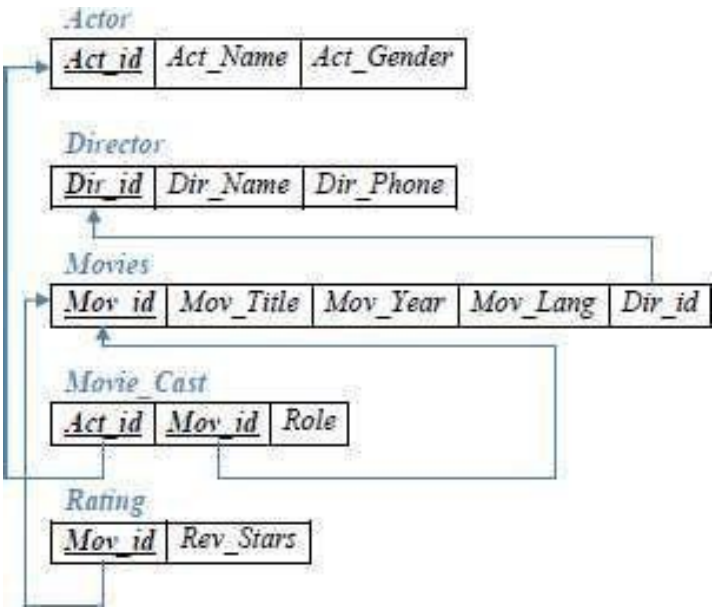
- Foundationknowledge indatabaseconcepts,technologyandpracticeto groomstudents into well-informed database application developers.
- Strongpractice inSQLprogrammingthroughavarietyofdatabaseproblems. Develop
- database applications using front-end tools and back-end DBMS.

Solution:

Entity-RelationshipDiagram



SchemaDiagram



TableCreation

```
CREATE TABLE ACTOR (
ACT_IDINT(5)PRIMARYKEY,
ACT_NAME VARCHAR (20),
ACT_GENDERCHAR(1));

CREATE TABLE DIRECTOR (
DIR_IDINT(5)PRIMARYKEY,
DIR_NAME VARCHAR (20),
DIR_PHONE BIGINT);

CREATE TABLE MOVIES
(MOV_IDINT(4)PRIMARYKEY,
MOV_TITLE VARCHAR (50),
MOV_YEARINT(4),
MOV_LANGVARCHAR(20),
DIR_IDINT(5),
FOREIGNKEY(DIR_ID)REFERENCESDIRECTOR(DIR_ID));

CREATETABLEMOVIES_CAST(
ACT_ID INT  (5),
MOV_IDINT(5),
ROLEVARCHAR(20),
PRIMARYKEY(ACT_ID,MOV_ID),
FOREIGN KEY (ACT_ID) REFERENCES ACTOR (ACT_ID),
FOREIGNKEY(MOV_ID)REFERENCESMOVIES(MOV_ID));
```

```
CREATE TABLE RATING (
MOV_IDINT(5)PRIMARYKEY,
REV_STARS VARCHAR (25),
FOREIGNKEY(MOV_ID)REFERENCESMOVIES(MOV_ID));
```

TableDescriptions

DESCACTOR;

```
mysql> DESC ACTOR;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| ACT_ID     | int(5)      | NO   | PRI | NULL    |       |
| ACT_NAME   | varchar(20) | YES  |     | NULL    |       |
| ACT_GENDER | char(1)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESCDIRECTOR;

```
mysql> DESC DIRECTOR;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| DIR_ID    | int(5)      | NO   | PRI | NULL    |       |
| DIR_NAME  | varchar(20) | YES  |     | NULL    |       |
| DIR_PHONE | bigint(20)  | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESCMOVIES;

```
mysql> DESC MOVIES;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| MOV_ID    | int(4)      | NO   | PRI | NULL    |       |
| MOV_TITLE | varchar(50) | YES  |     | NULL    |       |
| MOV_YEAR  | int(4)      | YES  |     | NULL    |       |
| MOV_LANG  | varchar(20) | YES  |     | NULL    |       |
| DIR_ID    | int(5)      | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

DESCMOVIES_CAST;

```
mysql> DESC MOVIES_CAST;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| ACT_ID | int(5)      | NO   | PRI | 0       |       |
| MOV_ID | int(5)      | NO   | PRI | 0       |       |
| ROLE   | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESCRATING;

```
mysql> DESC RATING;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| MOV_ID    | int(5)      | NO   | PRI | NULL    |       |
| REV_STARS | varchar(25) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

InsertionofValuestoTables

INSERTINTOACTORVALUES(1,'MADHURI DIXIT','F');
INSERT INTO ACTOR VALUES (2,'AAMIR KHAN','M');
INSERT INTO ACTOR VALUES (3,'JUHI CHAWLA','F');
INSERT INTO ACTOR VALUES (4,'SRIDEVI','F');

INSERT INTO DIRECTOR VALUES (100,'SUBHASH KAPOOR',9563400156);
INSERT INTO DIRECTOR VALUES(102,'ALAN TAYLOR',9971960035);
INSERT INTO DIRECTOR VALUES (103,'SANTHOSH ANANDDRAM', 9934611125);
INSERT INTO DIRECTOR VALUES (104,'IMTIAZ ALI', 8539920975);
INSERT INTO DIRECTOR VALUES (105,'HITCHCOCK',7766138911);
INSERTINTODIRECTORVALUES(106,'STEVENSPIELBERG',9966138934);

INSERTINTOMOVIESVALUES(501,'JABHARRYMETSEJAL',2017,'HINDI',104);
INSERT INTO MOVIES VALUES (502,'RAJAKUMARA',2017,'KANNADA',103);
INSERT INTO MOVIES VALUES (503,'JOLLY LLB 2', 2013,'HINDI', 100);
INSERTINTOMOVIESVALUES(504,'TERMINATORGENESYS',2015,'ENGLISH',102);
INSERT INTO MOVIES VALUES (505,'JAWS',1975,'ENGLISH',106);
INSERTINTOMOVIESVALUES(506,'BRIDGEOFSPIES',2015,'ENGLISH',106);
INSERT INTO MOVIES VALUES (507,'VERTIGO',1943,'ENGLISH',105);
INSERT INTO MOVIES VALUES (508,'SHADOW OF A DOUBT',1943,'ENGLISH',105);

INSERT INTO MOVIES_CAST VALUES (1, 501,'HEROINE');
INSERT INTO MOVIES_CAST VALUES (1, 502,'HEROINE');
INSERTINTOMOVIES_CASTVALUES(3,503,'COMEDIAN');
INSERT INTO MOVIES_CAST VALUES (4, 504,'GUEST');
INSERT INTO MOVIES_CAST VALUES (4, 501,'HERO');

INSERTINTORATINGVALUES(501,4);
INSERTINTORATINGVALUES(502,2);
INSERTINTORATINGVALUES(503,5);
INSERTINTORATINGVALUES(504,4);
INSERTINTORATINGVALUES(505,3);
INSERTINTORATINGVALUES(506,2);

SELECT*FROMACTOR;

ACT_ID	ACT_NAME	ACT
1	MADHURIDIXIT	F
2	AAMIRKHAN	M
3	JUHICHAWLA	F
4	SRIDEVI	F

SELECT*FROMDIRECTOR;

DIR_ID	DIR_NAME	DIR_PHONE
100	SUBHASHKAPOOR	56340015
102	ALANTAYLOR	719600310
103	SANTHOSHANANDDRAM	99346111
104	IMTIAZALI	85399209
105	HITCHCOCK	7766138911
106	STEVENSPIELBERG	9966138934

SELECT*FROMMOVIES;

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
501	JABHARRYMETSEJAL	2017	HINDI	104
502	RAJAKUMARA	2017	KANNADA	103
503	JOLLYLLB2	2013	HINDI	100
504	TERMINATORGENESYS	2015	ENGLISH	102
505	JAWS	1975	ENGLISH	106
506	BRIDGEOFSPIES	2015	ENGLISH	106
507	VERTIGO	1958	ENGLISH	105
508	SHADOWOFA DOUBT	1943	ENGLISH	105

SELECT*FROMMOVIE_CAST;

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
501	JABHARRYMETSEJAL	2017	HINDI	104
502	RAJAKUMARA	2017	KANNADA	103
503	JOLLYLLB2	2013	HINDI	100
504	TERMINATORGENESYS	2015	ENGLISH	102
505	JAWS	1975	ENGLISH	106
506	BRIDGEOFSPIES	2015	ENGLISH	106
507	VERTIGO	1958	ENGLISH	105
508	SHADOWOFA DOUBT	1943	ENGLISH	105

SELECT*FROMRATING;

MOV_ID	REV_STARS
501	4
502	2
503	5
504	4
505	3
506	2
507	2
508	4

Queries:

1. **List the titles of all movies directed by'Hitchcock'.**
   SELECTMOV_TITLEFROMMOVIESWHEREDIR_IDIN(SELECTDIR_IDFROM DIRECTOR WHERE DIR_NAME = 'HITCHCOCK');
   OR
   SELECTMOV_TITLEFROMMOVIESM,DIRECTORDWHEREM.DIR_ID=D.DIR_ID AND DIR_NAME='HITCHCOCK';

```
+--------------------+
| MOV_TITLE          |
+--------------------+
| VERTIGO            |
| SHADOW OF A DOUBT  |
+--------------------+
2 rows in set (0.00 sec)
```

2. **Findthemoviename whereoneormoreactorsactedintwoormoremovies.**
   SELECTMOV_TITLE FROMMOVIESM,MOVIES_CAST MV
   WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN(SELECT ACT_ID FROM MOVIES_CASTGROUPBYACT_IDHAVINGCOUNT(ACT_ID)>1)GROUPBY MOV_TITLE HAVING COUNT(*)>1;

```
+--------------------+
| MOV_TITLE          |
+--------------------+
| JAB HARRY MET SEJAL |
+--------------------+
1 row in set (0.00 sec)
```

3. **Listallactorswhoacted inamoviebefore2000andalsoinamovieafter2015(useJOIN operation).**
   SELECTACT_NAME,MOV_TITLE,MOV_YEARFROMACTORAJOIN MOVIE_CAST C ON A.ACT_ID=C.ACT_ID INNER JOIN MOVIES M ONC.MOV_ID=M.MOV_IDWHEREM.MOV_YEARNOT BETWEEN2000AND2015;

```
+---------------+--------------------+----------+
| ACT_NAME      | MOV_TITLE          | MOV_YEAR |
+---------------+--------------------+----------+
| MADHURI DIXIT | JAB HARRY MET SEJAL |     2017 |
| MADHURI DIXIT | RAJAKUMARA         |     2017 |
| SRIDEVI       | JAB HARRY MET SEJAL |     2017 |
+---------------+--------------------+----------+
3 rows in set (0.00 sec)
```

4. **Findthetitleofmoviesandnumberofstarsforeachmoviethathasatleastoneratingand find the highest number of stars that movie received. Sort the result by movietitle.** SELECT MOV_TITLE,MAX(REV_STARS) FROM MOVIES M ,RATING R WHERE M.MOV_ID=R.MOV_IDGROUPBYMOV_TITLEHAVINGMAX(REV_STARS)>0ORDER BY MOV_TITLE;

```
+---------------------+-------------------+
| MOV_TITLE           | MAX(REV_STARS)    |
+---------------------+-------------------+
| BRIDGE OF SPIES     | 2                 |
| JAB HARRY MET SEJAL | 4                 |
| JAWS                | 3                 |
| JOLLY LLB 2         | 5                 |
| RAJAKUMARA          | 2                 |
| TERMINATOR GENESYS  | 4                 |
+---------------------+-------------------+
6 rows in set (0.00 sec)
```

5. **Update rating of all movies directed by 'Steven Spielberg' to5**
   UPDATERATINGSETREV_STARS=5WHEREMOV_IDIN(SELECTMOV_IDFROM MOVIES WHERE DIR_ID IN(SELECT DIR_ID FROM DIRECTOR WHEREDIR_NAME='STEVENSPIELBERG'));
   OR
   UPDATE RATING R, MOVIES M, DIRECTOR D SET REV_STARS=5 WHERE R.MOV_ID=M.MOV_IDANDM.DIR_ID=D.DIR_IDANDDIR_NAME='STEVEN SPIELBERG';

```
mysql> SELECT * FROM RATING;
+--------+-----------+
| MOV_ID | REV_STARS |
+--------+-----------+
|    501 | 4         |
|    502 | 2         |
|    503 | 5         |
|    504 | 4         |
|    505 | 5         |
|    506 | 5         |
+--------+-----------+
6 rows in set (0.00 sec)
```

ProgramOutcomes:

Thestudentsareableto

- Create, Updateand queryonthedatabase.
- DemonstratetheworkingofdifferentconceptsofDBMS
- Implement,analyzeandevaluatetheprojectdevelopedforanapplication.

D. ConsidertheschemaforCollegeDatabase:

STUDENT(USN,SName,Address,Phone,Gender)
SEMSEC (SSID, Sem, Sec)
CLASS(USN,SSID)
SUBJECT(Subcode,Title,Sem,Credits)
IAMARKS(USN,Subcode,SSID,Test1,Test2,Test3,FinalIA)

WriteSQLqueriesto

1. **Listallthestudentdetails studying in fourth semester 'C'section.**
2. **Computethetotalnumberofmaleandfemalestudentsineachsemesterandineach section.**
3. **Create a view of Test1 marks of student USN '1BI15CS101' in allsubjects.**
4. **CalculatetheFinalIA(averageofbesttwotestmarks)andupdatethecorresponding table for all students.**
5. **Categorizestudentsbasedonthefollowingcriterion:**
   IfFinalIA=17to20then CAT= 'Outstanding'
   If FinalIA = 12 to 16 then CAT = 'Average'
   If FinalIA< 12 then CAT = 'Weak'
   Givethesedetailsonly for8thsemesterA,B,andCsection students.

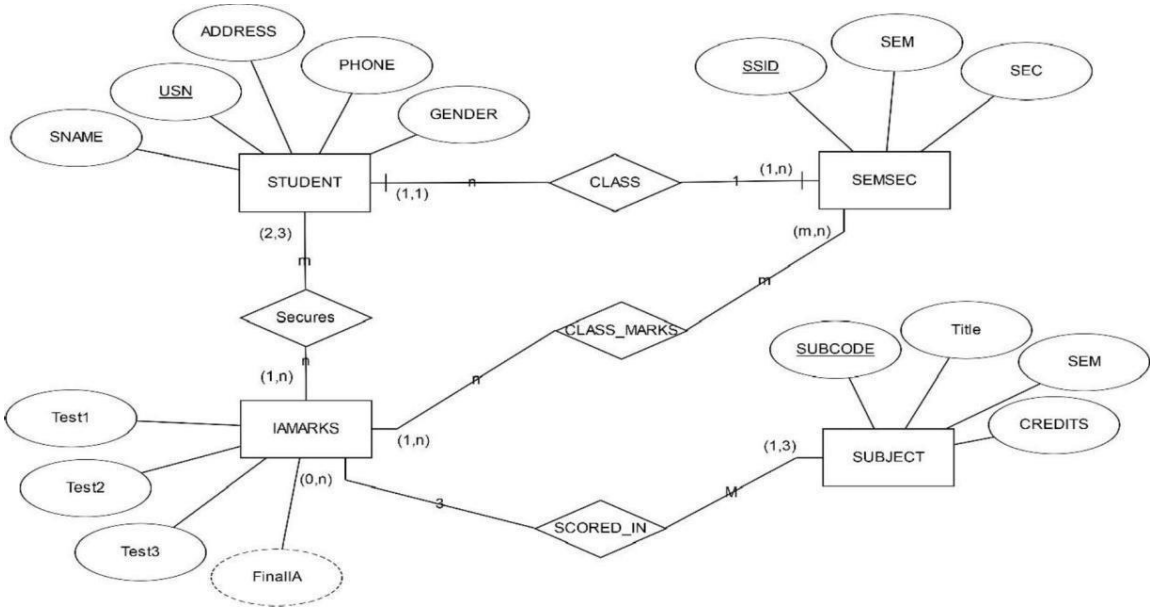
ProgramObjectives:

Thiscoursewillenablestudentsto

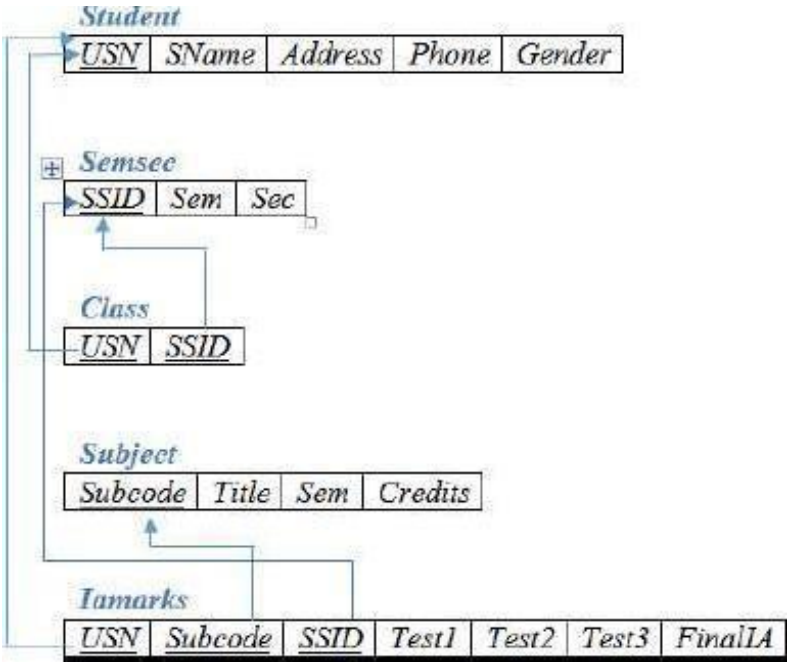
- Foundationknowledgeindatabaseconcepts,technologyandpracticetogroomstudents into well-informed database application developers.
- Strongpractice inSQLprogrammingthroughavarietyofdatabaseproblems. Develop
- database applications using front-end tools and back-end DBMS.

Solution:

Entity-RelationshipDiagram



SchemaDiagram



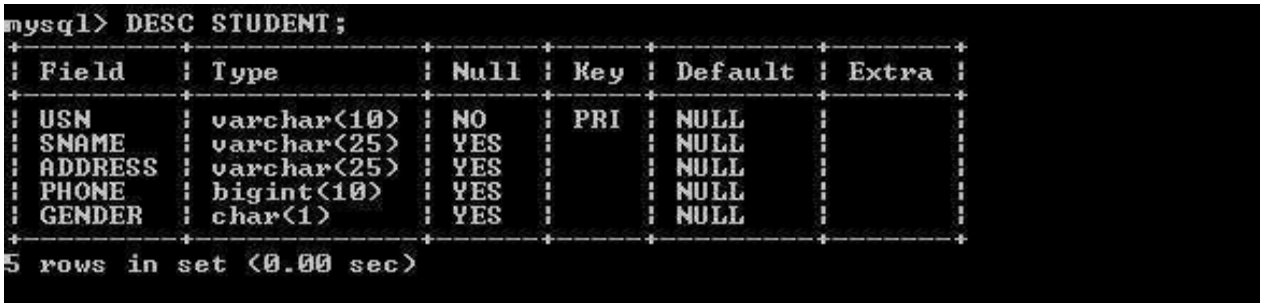
TableCreation

```
CREATETABLESTUDENT(
USNVARCHAR(10)PRIMARYKEY,
SNAME VARCHAR (25),
ADDRESSVARCHAR(25),
PHONEBIGINT(10),
GENDERCHAR(1));

CREATETABLESEMSEC(
SSIDVARCHAR(5)PRIMARYKEY,
SEM INT (5),
SECCHAR (1));

CREATETABLECLASS(
USN VARCHAR (10),
SSID VARCHAR (5),
PRIMARYKEY(USN,SSID),
FOREIGNKEY(USN)REFERENCESSTUDENT(USN),
FOREIGNKEY(SSID)REFERENCESSEMSEC(SSID));

CREATETABLESUBJECT(
SUBCODE VARCHAR(10)
PRIMARY KEY,
TITLEVARCHAR(20),
SEM INT,
CREDITSINT);
```

```
CREATETABLEIAMARKS(
USN VARCHAR (10),
SUBCODEVARCHAR(8),
SSIDVARCHAR(5),
TEST1INT(2),
TEST2INT(2),
TEST3INT(2),
FINALIAINT(2),
PRIMARYKEY(USN,SUBCODE,SSID),
FOREIGNKEY(USN)REFERENCESSTUDENT(USN),
FOREIGNKEY(SUBCODE)REFERENCESSUBJECT(SUBCODE),FOREIGN
KEY (SSID) REFERENCES SEMSEC (SSID));
```

TableDescriptions

DESC STUDENT;



```
mysql> DESC STUDENT;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| USN     | varchar(10) | NO   | PRI | NULL    |       |
| SNAME   | varchar(25) | YES  |     | NULL    |       |
| ADDRESS | varchar(25) | YES  |     | NULL    |       |
| PHONE   | bigint(10)  | YES  |     | NULL    |       |
| GENDER  | char(1)     | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

DESC SEMSEC;

```
mysql> DESC SEMSEC;
+-------+-----------+------+-----+---------+-------+
| Field | Type      | Null | Key | Default | Extra |
+-------+-----------+------+-----+---------+-------+
| SSID  | varchar(5)| NO   | PRI | NULL    |       |
| SEM   | int(5)    | YES  |     | NULL    |       |
| SEC   | char(1)   | YES  |     | NULL    |       |
+-------+-----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESC CLASS;

```
mysql> DESC CLASS;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| USN   | varchar(10) | NO   | PRI |         |       |
| SSID  | varchar(5)  | NO   | PRI |         |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

foreignkey(ssid)referencessemsec(ssid));

desciamarks;

```
+-------+-----------+----+---+-------+-----+
|Field  |Type       |Null|Key|Default|Extra|
+-------+-----------+----+---+-------+-----+
|usn    |varchar(20)|YES|MUL|NULL     |     |
|subcode|varchar(20)|YES|MUL|NULL     |     |
|ssid   |varchar(15)|YES|MUL|NULL     |     |
|test1  |int(2)     |YES|   |NULL     |     |
|test2  |int(2)     |YES|   |NULL     |     |
|test3  |int(3)     |YES|   |NULL     |     |
|finalia|int(2)     |YES|   |NULL     |     |
+-------+-----------+----+---+-------+-----+
```

Insertionofvaluesintotables

```
insert into student values ('1ck15cs045','jeeva','bellary', 9944501212212,'m');
insert into student values ('1ck15cs090','sunny','bangaluru', 882857541,'m');
insert into student values ('1ck15cs091','santosh','mangaluru', 882332201,'m');
insert into student values ('1ck16cs045','ismail','kalburgi', 990023221,'m');
insert into student values ('1ck16cs081','sameera','shimoga',94450121,'f');
insertintostudentvalues('1ck16cs122','vinayaka','chikamagalur',80880011,'m');
insert into student values ('1ck16cs088','sameera','shimoga',99042212,'f');
```

select*fromstudent;

```
+----------+--------+----------+--------+------+
|usn       |sname   |address   |phone   |gender|
+----------+--------+----------+--------+------+
|1ck15cs045|jeeva   |bellary   |994450121|m     |
|1ck15cs090|sunny   |bangaluru |882857541|m     |
|1ck15cs091|santosh|mangaluru  |882332201|m     |
|1ck16cs045|ismail  |kalburgi  |990023221|m     |
|1ck16cs081|sameera|shimoga    |990559012|f     |
|1ck16cs122|vinayaka|chikamagalur|80880011|m    |
|1ck16cs088|sameera|shimoga    |99042212|f      |
+----------+--------+----------+--------+------+
```

```
insertintosemsecvalues('cse8a',8,'a');
insertintosemsecvalues('cse8b',8,'b');
insertintosemsecvalues('cse8c',8,'c');
insertintosemsecvalues('cse7a',7,'a');
insertintosemsecvalues('cse7b',7,'a');
insertintosemsecvalues('cse7c',7,'c');
insertintosemsecvalues('cse6a',6,'a');
insertintosemsecvalues('cse6b',6,'b');
insertintosemsecvalues('cse6c',6,'c');
insertintosemsecvalues('cse5a',5,'a');
insertintosemsecvalues('cse5b',5,'b');
insertintosemsecvalues('cse5c',5,'c');
insertintosemsecvalues('cse4a',4,`a`);
insertintosemsecvalues('cse4b',4,`b`);
insertintosemsecvalues('cse4c',4,'c`);
insertintosemsecvalues('cse3a',3,'a');
insertintosemsecvalues('cse3b',3,'b');
```

```
insertintosemsecvalues('cse3c',3,‘c‘);
insertintosemsecvalues('cse2a',2,‘a‘);
insertintosemsecvalues('cse2b',2,'b‘);
insertintosemsecvalues('cse2c',2,'c');
insertintosemsecvalues('cse1a',1,'a');
insertintosemsecvalues('cse1b',1,‘b‘);
insertintosemsecvalues('cse1c',1,‘c‘); insert
into semsec values('cse4a',4,'a'); insert into
semsec values('cse4b',4,'b'); insert into
semsec values('cse4c',4,'c');
```

select*fromsemsec;

```
+·····+·····+·····+
|ssid|sem|sec|
+-----+-----+·····+
|cse4a|      4|a      |
|cse4b|      4|b      |
|cse4c|      4|c      |
|cse5a|      5|b      |
|cse5b|      5|c      |
|cse5c|      5|a      |
|cse6a|      7|a      |
|cse6b|      6|b      |
|cse6c|      6|c      |
|cse7a|      7|a      |
|cse7b|      7|b      |
|cse7c|      7|c      |
|cse8a|      8|a      |
|cse8b|      8|b      |
|cse8c|      8|c      |
+-----+----- +----- +
```

```
insertintoclassvalues('1ck15cs045','cse5a');
insertintoclassvalues('1ck15cs090','cse5b');
insertintoclassvalues('1ck15cs091','cse5c');
insertintoclassvalues('1ck16cs045','cse6a');
insertintoclassvalues('1ck16cs081','cse6b');
insertintoclassvalues('1ck16cs088','cse6c');
insertintoclassvalues('1ck16cs122','cse7c');
insertintoclassvalues('1ck16cs088','cse8a');
insertintoclassvalues('1ck16cs081','cse8b');
insertintoclassvalues('1ck16cs081','cse8b');
insertintoclassvalues('1ck15cs091','cse4c');
```

select*fromclass;

```
+·····+·····+
|usn          |ssid|
+·····+·····+
|1ck15cs091|cse4c |
|1ck15cs045|cse5a|
|1ck15cs090|cse5b|
|1ck15cs091|cse5c|
|1ck16cs045|cse6a|
|1ck16cs081|cse6b|
|1ck16cs088|cse6c|
|1ck16cs122|cse7c|
|1ck16cs088|cse8a|
```

```
|1ck16cs081|cse8b|
+……….……+……..+
```

```
insert into subject values('10cs81','aca',8,4);
insert into subject values ('10cs82','ssm', 8, 4);
insert into subject values ('10cs83','nm', 8, 4);
insert into subject values ('10cs84','cc', 8, 4);
insert into subject values ('10cs85','pw', 8, 4);
insertintosubjectvalues('10cs71','ooad',7,4);
insert into subject values ('10cs72','ecs', 7, 4);
insert into subject values ('10cs73','ptw', 7, 4);
insertintosubjectvalues('10cs74','dwdm',7,4);
insert into subject values ('10cs76','san', 7, 4);
insert into subject values ('15cs51', 'me', 5, 4);
insert into subject values ('15cs52','cn', 5, 4);
insertintosubjectvalues('15cs53','dbms',5,4);
insert into subject values ('15cs54','atc', 5, 4);
insertintosubjectvalues('15cs55','java',5,3);
insert into subject values ('15cs56','ai', 5, 3);
insert into subject values ('15cs41','m4', 4, 4);
insert into subject values ('15cs42','se', 4, 4);
insert into subject values ('15cs43','daa', 4, 4);
insertintosubjectvalues('15cs44','mpmc',4,4);
insert into subject values ('15cs45','ooc', 4, 3);
insert into subject values ('15cs46','dc', 4, 3);
insert into subject values ('15cs31','m3', 3, 4);
insert into subject values ('15cs32','ade', 3, 4);
insert into subject values ('15cs33','dsa', 3, 4);
insert into subject values ('15cs34','co', 3, 4);
insert into subject values ('15cs35','usp', 3, 3);
insert into subject values ('15cs36','dms', 3, 3);
```

select*fromsubject;

```
+……….+……..+……..+……..+
|subcode|title|sem|credits|
+……….+……..+……..+……..+
| 10cs71  | ooad  |    7 |       4 |
| 10cs72  | ecs   |    7 |       4 |
| 10cs73  | ptw   |    7 |       4 |
| 10cs74  | dwdm  |    7 |       4 |
| 10cs76  | san   |    7 |       4 |
| 10cs81  | aca   |    8 |       4 |
| 10cs82  | ssm   |    8 |       4 |
| 10cs83  | nm    |    8 |       4 |
| 10cs84  | cc    |    8 |       4 |
| 10cs85  | pw    |    8 |       4 |
| 15cs31  | m3    |    3 |       4 |
| 15cs32  | ade   |    3 |       4 |
| 15cs33  | dsa   |    3 |       4 |
| 15cs34  | co    |    3 |       4 |
| 15cs35  | usp   |    3 |       3 |
| 15cs36  | dms   |    3 |       3 |
| 15cs41  | m4    |    4 |       4 |
| 15cs42  | se    |    4 |       4 |
| 15cs43  | daa   |    4 |       4 |
| 15cs44  | mpmc  |    4 |       4 |
```

```
|15cs45|ooc       |    4 |       3|
|15cs46|dc        |    4 |       3|
|15cs51|me        |    5 |       4|
|15cs52|cn        |    5 |       4|
|15cs53|dbms|          5 |       4|
|15cs54|atc       |    5 |       4|
|15cs55|java|          5 |       3|
|15cs56|ai        |    5 |       3|
+------------+---------+---------+---------+
```

```
insertintoiamarksvalues('1ck15cs045','10cs81','cse5a',15,16,18,20);
insertintoiamarksvalues('1ck15cs090','10cs82','cse5b',15,15,18,20);
insertintoiamarksvalues('1ck15cs091','10cs83','cse5c',15,15,18,20);
insertintoiamarksvalues('1ck15cs045','10cs84','cse6a',15,15,17,19);
insertintoiamarksvalues('1ck16cs081','10cs85','cse6b',15,15,17,18);
insertintoiamarksvalues('1ck16cs081','10cs71','cse6c',15,15,17,18);
insertintoiamarksvalues('1ck16cs122','10cs72','cse7a',15,15,16,18);
insertintoiamarksvalues('1ck16cs088','10cs73','cse7b',15,15,16,19);
```

select*fromiamarks;

```
+------------+---------+-----+-----+-----+-----+-----+
|usn         |subcode|ssid|test1|test2|test3|finalia|
+------------+---------+-----+-----+-----+-----+-----+
|1ck15cs045|10cs81|cse5a|           15|     16|     18|       20|
|1ck15cs090|10cs82|cse5b |          15|     15|     18|       20|
|1ck15cs091|10cs83|cse5c|           15|     15|     18|       20|
|1ck15cs045|10cs84|cse6a|           15|     15|     17|       19|
|1ck16cs081|10cs85|cse6b|           15|     15|     17|       18|
|1ck16cs081|10cs71|cse6c|           15|     15|     17|       18|
|1ck16cs122|10cs72|cse7a|           15|     15|     16|       18|
|1ck16cs088|10cs73|cse7b|           15|     15|     16|       19|
+------------+---------+-----+-----+-----+-----+-----+
```

Queries:

1. Listallthestudentdetailsstudyinginfourthsemester‘C’section.

```
selects.*,ss.sem,ss.sec
fromstudents,semsecss,class c
wheres.usn=c.usnandss.ssid=c.ssid
      andss.sem =4 and ss.sec='c';
```

```
+------------+---------+-----------+-----------+------+---+---+
|usn         |sname    |address    |phone      |gender|sem|sec|
+------------+---------+-----------+-----------+------+---+---+
|1ck15cs091|santosh|mangaluru|882332201| m          |    4|c      |
+------------+---------+-----------+-----------+------+---+---+
```

2. Computethetotalnumberofmaleand femalestudentsineachsemesterandineachsection.

```
selectss.sem,ss.sec,s.gender,count(s.gender)ascount from
student s, semsec ss, class c
wheres.usn=c.usnandss.ssid=c.ssid group
by ss.sem, ss.sec, s.gender
orderbysem;
```

```
+---+---+------+-----+
|sem|sec|gender|count|
```

```
+----------+----------+----------------+-------------+
|     4|c       |m         |      1|
|     5|a       |m         |      1|
|     5|b       |m         |      1|
|     5|c       |m         |      1|
|     6|b       |f         |      1|
|     6|c       |f         |      1|
|     7|a       |m         |      1|
|     7|c       |m         |      1|
|     8|a       |f         |      1|
|     8|b       |f         |      1|
+----------+----------+----------------+-------------+
```

3. CreateaviewofTest1marksofstudentUSN‘1ck15CS91’inallsubjects.

```
createviewstu_test1_marks_view as
select test1, subcode
fromiamarks
whereusn='1ck15cs091';

select*fromstu_test1_marks_view;
```

```
+-----------------+-----------------+
|test1|subcode|
+-----------------+-----------------+
|     15|10cs83|
+-----------------+-----------------+
```

4. CalculatetheFinalIA(averageofbesttwotestmarks)andupdatethecorresponding table for all students.

5. Categorizestudentsbasedonthefollowingcriterion:

IfFinalIA=17to20thenCAT=‘Outstanding’ If
FinalIA = 12 to 16 then CAT = ‘Average’ If
FinalIA< 12 then CAT = ‘Weak’
Givethesedetailsonlyfor8th semesterA,B,and Csection students.

```
selects.usn,s.sname,s.address,s.phone,s.gender,
(casewhenia.finaliabetween17and20then'outstanding' when
ia.finalia between 12 and 16 then 'average'
else'weak'
end)ascat
fromstudents,semsecss,iamarksia,subjectsub where
s.usn = ia.usn and
ss.ssid=ia.ssidand
sub.subcode=ia.subcodeandsub.sem=8;
```

```
+----------+---------+---------+---------+------+-----------+
|usn       |sname    |address  |phone    |gender| cat       |
+----------+---------+---------+---------+------+-----------+
|1ck15cs045|jeeva    |bellary  |994450121|m     |outstanding|
|1ck15cs090|sunny    |bangaluru|882857541|m     |outstanding|
|1ck15cs091|santosh|mangaluru|882332201| m      |outstanding|
|1ck15cs045|jeeva    |bellary  |994450121|m     |outstanding|
|1ck16cs081|sameera|shimoga   |990559012|f     |outstanding|
-------+-------------------------------+----------------+----------------+-----------+-
```

E. ConsidertheschemaforCompanyDatabase:

EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN, DNo)DEPARTMENT(DNo,DName,MgrSSN,MgrStartDate)
DLOCATION (DNo,DLoc)
PROJECT(PNo,PName,PLocation,DNo)
WORKS_ON (SSN, PNo, Hours)

WriteSQLqueriesto

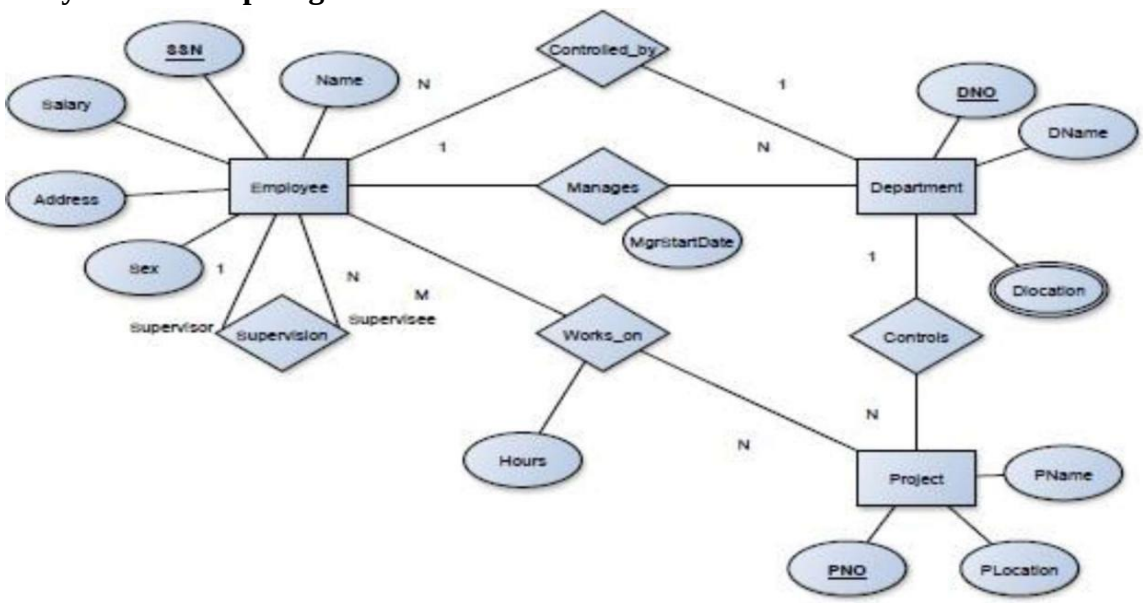
1. **Makealistofallprojectnumbersforprojectsthatinvolveanemployeewhoselast name is 'Scott', either as a worker or as a manager of the department that controls the project.**
2. **Showtheresultingsalariesifevery employee working on the 'IoT' project is given a 10 percent raise.**
3. **Find the sumof the salaries of all employees of the 'Accounts' department, as well as themaximumsalary,theminimumsalary,andtheaveragesalaryinthisdepartment**
4. **Retrieve thename of eachemployee who workson allthe projectscontrolled by department number 5 (use NOT EXISTS operator).**
5. **Foreach department thathasmorethan fiveemployees, retrievethe department numberandthenumberofitsemployeeswhoare makingmorethanRs. 6,00,000.**

ProgramObjectives:

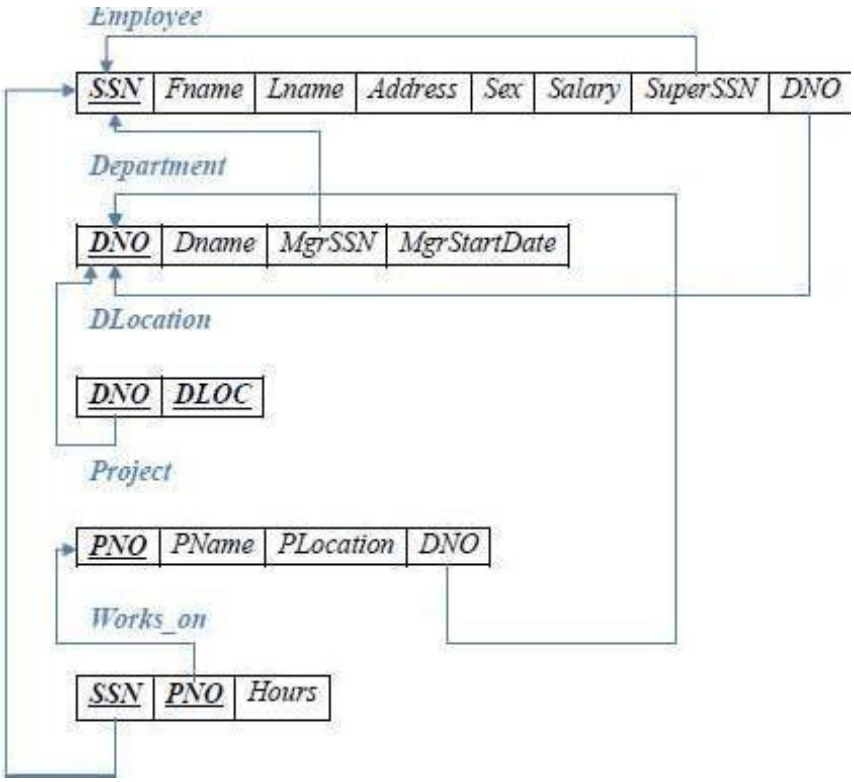
Thiscoursewillenablestudentsto

- Foundationknowledgeindatabaseconcepts,technologyandpracticetogroomstudents into well-informed database application developers.
- Strongpractice inSQLprogrammingthroughavarietyofdatabaseproblems. Develop
- database applications using front-end tools and back-end DBMS.

Solution:
Entity-RelationshipDiagram



SchemaDiagram



TableCreation

CREATE TABLE DEPARTMENT (
DNOVARCHAR(20)PRIMARYKEY,
DNAME VARCHAR (20),
MGRSTARTDATE DATE,
MGRSSNVARCHAR (20));

CREATETABLEEMPLOYEE(
SSNVARCHAR(20)PRIMARYKEY,
FNAME VARCHAR (20),
LNAMEVARCHAR(20),
ADDRESSVARCHAR(100),
SEXCHAR(1),
SALARYINT(10),
SUPERSSNVARCHAR(20),
DNOVARCHAR (20),
FOREIGNKEY(SUPERSSN)REFERENCESEMPLOYEE(SSN),
FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO));

NOTE: OnceDEPARTMENTandEMPLOYEEtablesarecreatedwemustalterdepartment table to add foreign constraint MGRSSN using sql command
ALTERTABLEDEPARTMENTADDFOREIGNKEY(MGRSSN)REFERENCES EMPLOYEE(SSN);

```
CREATETABLEDLOCATION(
DLOC VARCHAR (20),
DNO VARCHAR (20),
PRIMARYKEY(DNO,DLOC),
FOREIGNKEY(DNO)REFERENCESDEPARTMENT (DNO));

CREATE TABLE PROJECT (
PNOINT(10)PRIMARYKEY,
PNAME VARCHAR (20),
PLOCATIONVARCHAR(20),
DNOVARCHAR (20),
FOREIGNKEY(DNO)REFERENCESDEPARTMENT (DNO));

CREATETABLEWORKS_ON( HOURS
INT (4),
SSNVARCHAR(20),
PNOINT(10),
PRIMARYKEY(SSN,PNO),
FOREIGNKEY(SSN)REFERENCESEMPLOYEE(SSN),
FOREIGN KEY (PNO) REFERENCES PROJECT (PNO));
```

TableDescriptions

DESCEMPLOYEE;

```
mysql> DESC EMPLOYEE;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| SSN      | varchar(20)  | NO   | PRI | NULL    |       |
| FNAME    | varchar(20)  | YES  |     | NULL    |       |
| LNAME    | varchar(20)  | YES  |     | NULL    |       |
| ADDRESS  | varchar(100) | YES  |     | NULL    |       |
| SEX      | char(1)      | YES  |     | NULL    |       |
| SALARY   | int(10)      | YES  |     | NULL    |       |
| SUPERSSN | varchar(20)  | YES  | MUL | NULL    |       |
| DNO      | varchar(20)  | YES  | MUL | NULL    |       |
+----------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

DESC DEPARTMENT;

```
mysql> DESC DEPARTMENT;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| DNO          | varchar(20) | NO   | PRI | NULL    |       |
| DNAME        | varchar(20) | YES  |     | NULL    |       |
| MGRSTARTDATE | date        | YES  |     | NULL    |       |
| MGRSSN       | varchar(20) | YES  | MUL | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
+------------+----------------+----+---+-------+-----+
|Field       |Type            |Null|Key|Default|Extra|
+------------+----------------+----+---+-------+-----+
|hours       |int(2)          |YES|      |          |      |
|ssn         |int)            |MUL|      |NULL      |      |
|pno         |int             | MUL|     |NULL      |      |
+------------+----------------+----+---+-------+-----+
```

Insertingofvaluestotables

```
insertintodepartmentvalues('1','accounts','01-01-01');
insert into department values ('2','it','01-08-16');
insert into department values ('3','ece','01-06-08');
insert into department values ('4','ise','01-08-15');
insert into department values ('5','cse','01-06-02');
```

select*fromdepartment;

```
+-----+----------+---------------+
|dno|dname       |mgrstartdate|
+-----+----------+---------------+
|1    |accounts|2001-01-01      |
|2    |it        |2001-08-16    |
|3    |ece       |2001-06-08    |
|4    |ise       |2001-08-15    |
|5    |cse       |2001-06-02    |
+-----+----------+---------------+
```

```
insert into employee values(100,'john','scott','bangalore',1,'m',100,10000);
insertintoemployeevalues(101,'james','smith','bangalore',2,'m',100,15000);
insertintoemployeevalues(102,'hearn','baker','bangalore',2,'m',101,20000);
insert into employee values(103,'edward','scott','mysore',3,'m',103,25000);
insertintoemployeevalues(104,'pauan','hegde','mangalore',4,'m',104,30000);
insert into employee values(105,'girish','malya','mysore',5,'m',105,35000);
insert into employee values(105,'neha','sn','bangalore',5,'m',105,35000);
insert into employee values(106,'neha','sn','bangalore',5,'m',105,35000);
insert into employee values(107,'anil','bv','bangalore',6,'m',106,30000);
insert into employee values(108,'anil','bv','bangalore',6,'m',106,25000);
```

select*fromemployee;

```
+-----+-------+-------+----------+-----+-----+--------+------+
|ssn|fname|lname|address            |dno|sex|superssn|salary|
+-----+-------+-------+----------+-----+-----+--------+------+
|100| john     |scott|bangalore|       1|m       |      100|10000|
|101|james|smith|bangalore|           2|m       |      100|15000|
|102|hearn|baker|bangalore|           2|m       |      101|20000|
|103|edward|scott|mysore            |    3|m       |      103|25000|
|104|pauan|hegde|mangalore|           4|m       |      104|30000|
|105|girish|malya|mysore            |    5|m       |      105|35000|
|106| neha     |sn     |bangalore |      5|m       |      105|35000|
|107| anil     |bv     |bangalore |      6|m       |      106|30000|
|108| anil     |bv     |bangalore |      6|m       |      106|25000|
+-----+-------+-------+----------+-----+-----+--------+------+
```

```
insert into dlocation values('bangalore',1);
insert into dlocation values('mangalore',2);
insert into dlocation values('mysore',3);
insert into dlocationvalues('kolar',4); insert
into dlocation values('shimoga',5);
insertintodlocationvalues('chikkamangal',5);
```

select*fromdlocation;

```
+…….……+……+
|dloc           |dno|
+……..……+……..+
|bangalore      |    1|
|mangalore      |    2|
|mysore         |    3|
|kolar          |    4|
|chikkamangal|       5|
|shimoga        |    5|
+……..……+……..+
```

```
insert into project values(11,'iot','bangalore',1);
insert into project values(12,'cloud','bangalore',1);
insert into project values(13,'cloud','bangalore',2);
insertintoprojectvalues(14,'bigdata','bangalore',3);
insertintoprojectvalues(15,'sensors','bangalore',4);
insert into project values(16,'bank management','bangalore',5);
insertintoprojectvalues(17,'salarymanagement','bangalore',5);
```

select*fromproject;

```
+……+……..……+……..……+……+
|pno|pname                  |plocation|dno|
+……+……..……+……..……+……+
|11| iot                    |bangalore|      1|
|12|cloud                   |bangalore|      1|
|13|cloud                   |bangalore|      2|
|14|bigdata                 |bangalore|      3|
|15|sensors                 |bangalore|      4|
|16|bankmanagement          |bangalore|      5|
|17|salarymanagement|bangalore |            5|
+……+……..……+……..……+……+
```

```
insert into works_on values(4,100,11);
insert into works_on values(6,101,12);
insert into works_on values(8,102,13);
insertintoworks_onvalues(10,103,14);
insertintoworks_onvalues(03,104,15);
insertintoworks_onvalues(04,105,16);
insertintoworks_onvalues(04,106,17);
```

select*fromworks_on;

```
+……..……+……..……+……+
|hours|ssn|pno|
+----------+---------+---------- +
|      4|100     |    11|
|      6|101|         12|
|      8|102|         13|
|     10|103|         14|
|      3|104|         15|
|      4|105|         16|
|      4|106|         17|
+----------+---------- +---------- +
```

QUERIES

1. Makealistofallprojectnumbersforprojectsthatinvolveanemployeewhoselastnameis 'Scott', either as a worker or as a manager of the department that controls theproject.

```
(selectdistinctp.pno
fromprojectp,departmentd,employee e
wheree.dno=d.dnoandd.mgrssn=e.ssnande.lname='scott') union
(selectdistinctp1.pno
fromprojectp1,works_onw,employeee1
wherep1.pno=w.pnoande1.ssn=w.ssnande1.lname='scott');
```

```
+····+
|pno|
+····+
|11|
|14|
+····+
```

2. Showtheresultingsalariesifeveryemployeeworkingonthe'IoT'projectisgivena10 percent raise.

```
selecte.fname,e.lname,1.1*e.salaryasincr_sal from
  employee e, works_on w, project p
  where e.ssn=w.ssn
  and w.pno=p.pno
  andp.pname='iot';
```

```
+····+····+····+
|fname|lname|incr_sal|
+····+····+····+
|john|scott|11000.0|
+····+····+····+
```

3. Findthesumofthesalariesofallemployeesofthe'Accounts'department,aswellasthe maximum salary, the minimum salary, and the average salary in this department

```
sum(e.salary),max(e.salary),min(e.salary),avg(e.salary)
fromemployeee,departmentd
wheree.dno=d.dnoandd.dname='accounts';
```

```
+····+····+····+····+
|sum(e.salary)|max(e.salary)|min(e.salary)|avg(e.salary)|
+····+····+····+····+
|          55000|          30000|          25000|     27500.0000|
+····+····+····+····+
```

4. Retrievethenameofeachemployeewhoworksonalltheprojectscontrolledbydepartment number 5 (use NOT EXISTS operator). For each department that has more than five employees,retrievethedepartmentnumberandthenumberof itsemployeeswhoaremaking more than Rs. 6,00,000.

selecte.fname,e.lname

from employee e

```
wherenotexists((selectpno
                from project
                where dno=_5_)
                minus(selectpno
                        from works_on
                         wheree.ssn=ssn));
```

5. Foreachdepartmentthathasmorethanfiveemployees,retrievethedepartmentnumberandthe number of its employees who are making more than Rs. 6,00,000.

```
selectd.dno,count(*)
fromdepartmentd,employeee
where d.dno=e.dno
ande.salary>600000
andd.dnoin(selecte1.dno
            from employee e1
           group by e1.dno
           havingcount(*)>5)
           group by d.dno;
```

Viva Questions

1. **Whatis SQL?**

   StructuredQueryLanguage

2. **Whatis database?**

   A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

3. **Whatis DBMS?**

   It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

4. **WhatisaDatabasesystem?**

   ThedatabaseandDBMSsoftwaretogetheriscalled asDatabasesystem.

5. **AdvantagesofDBMS?**

   - Redundancyiscontrolled.
   - Unauthorized access is restricted.
   - Providingmultipleuserinterfaces. •

     Enforcing integrity constraints.
   - Providingbackupandrecovery.

6. **Disadvantagein FileProcessingSystem?**

   - Dataredundancy&inconsistency. •

     Difficult in accessing data.
   - Dataisolation.
   - Dataintegrity.
   - Concurrentaccessisnotpossible. •

     SecurityProblems.

7. **Definethe"integrityrules"**

   TherearetwoIntegrityrules.

   - EntityIntegrity: Statesthat"Primarykeycannot haveNULLvalue"
   - Referential Integrity: States that "Foreign Key canbeeither aNULLvalueorshould bePrimaryKeyvalueofother relation.

8. Whatisaview? Howitisrelatedtodata independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary. Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

9. Whatis DataModel?

A collection of conceptual tools for describing data, data relationships, data semantics and constraints.

10. Whatis E-Rmodel?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database bya set of attributes.

11. WhatisObjectOrientedmodel?

This model is based on collection of objects. An object contains values stored in instance variableswithintheobject.Anobjectalsocontainsbodiesofcodethatoperateontheobject. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

12. Whatis anEntity?

Itisan'object'intherealworldwithanindependentexistence.

13. What is anEntitytype?

Itisacollection(set)ofentitiesthathavesame attributes.

14. Whatis anattribute?

Itisaparticularproperty, whichdescribestheentity.

15. WhatisdegreeofaRelation?

Itisthenumberofattribute ofitsrelationschema.

16. Whatis Relationship?

Itisanassociationamongtwoormoreentities.

17. What isDDL(DataDefinitionLanguage)?

Adatabaseschema isspecified byasetofdefinitionsexpressed byaspeciallanguage called DDL.

18. WhatisDML(DataManipulationLanguage)?

Thislanguagethatenableusertoaccessormanipulatedataasorganized byappropriate

datamodel.

19. Whatisnormalization?

ItisaprocessofanalyzingthegivenrelationschemasbasedontheirFunctional Dependencies (FDs) and primary keyto achieve the properties

- Minimizing redundancy
- Minimizinginsertion,deletionandupdateanomalies.

20. Whatis1NF(Normal Form)?

Thedomainofattribute mustincludeonlyatomic(simple, indivisible)values.

21. Whatis 2NF?

A relation schema Risin 2NF if itisin 1NF andeverynon-prime attribute A in Risfully functionally dependent on primary key.

22. Whatis 3NF?

Arelationschema R isin3NFifit isin2NFand foreveryFD X Aeither ofthefollowing is true

- Xisa Super-keyofR.
- Aisaprime attributeofR.

Inotherwords, ifeverynonprimeattributeisnon-transitivelydependent onprimarykey.

23. WhatisBCNF(Boyce-CoddNormalForm)?

A relation schema R is in BCNF if itis in 3NF and satisfies additional constraints thatforevery FD X A, X must be a candidate key.

24. Whatis 4NF?

ArelationschemaRissaidtobein4NFifforeveryMultivalued dependencyXYthat holds over R, one of following is true

- Xissubsetorequalto(or)XY=R. X is a
- super key.

25. Whatis 5NF?

A Relation schemaRissaidtobe5NF if forevery join dependency {R1,R2,...,Rn}that holds R, one the following is true

- Ri=R forsomei.
- ThejoindependencyisimpliedbythesetofFD,overR inwhichtheleftsideiskey ofR.

26. Whatarepartial,alternate,,artificial,compoundandnaturalkey?

Partial Key:

It is a set of attributes that can uniquely identify weak entities and that are related tosameowner entity. It is sometime called as Discriminator.

AlternateKey:

AllCandidateKeysexcludingthePrimaryKeyareknownasAlternate Keys.

ArtificialKey:

If no obvious key, either standalone or compound is available, then the last resort is to simplycreateakey,byassigningauniquenumbertoeach recordoroccurrence. Then thisis known as developing an artificial key.

CompoundKey:

If no single data element uniquelyidentifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.

NaturalKey:

When one of the data elements stored within a construct is utilized as the primarykey, thenit is called the natural key.

27. Whatismeantbyqueryoptimization?

The phase that identifies an efficient execution plan for evaluating a querythat has the least estimated cost is referred to as query optimization.

28. Whatdoyoumeanbyatomicityandaggregation?

Atomicity:

Eitherallactionsarecarriedoutornoneare. Usersshouldnothavetoworryabouttheeffect of incomplete transactions. DBMS ensures this by undoing the actions of incomplete transactions.

Aggregation*:*

Aconceptwhichisused tomodelarelationshipbetweenacollectionofentitiesand relationships.Itisusedwhenweneedtoexpressarelationshipamongrelationships.

29. Whatis acheckpoint andwhendoesitoccur?

A Checkpoint is like a snapshot ofthe DBMS state. Bytaking checkpoints, the DBMS can reduce the amount of workto be done during restart in the event of subsequent crashes.

30. Whatdoyoumeanbyflatfiledatabase?

Itisadatabase inwhichtherearenoprogramsor useraccesslanguages.Ithasno cross-file

capabilitiesbutisuser-friendlyandprovidesuser-interface management.

31. BrieftheoryofNetwork,Hierarchicalschemasandtheirproperties

Networkschemausesa graphdatastructuretoorganizerecordsexampleforsuchadatabase managementsystemisCTCGwhileahierarchicalschemausesatreedata structureexample for such a system is IMS.

32. Whatis aquery?

Aquerywith respect to DBMS relates to usercommands that areusedtointeract with adata base. The query language can be classified into data definition language and data manipulation language.

33. WhatdoyoumeanbyCorrelatedsubquery?

Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent queryor itcan be executed once for each row returned by the parentquery.If the subquery is executed for each row of the parent, this is called a *correlated subquery*.

Acorrelated subquerycan beeasilyidentifiedifitcontainsanyreferencesto theparent

subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated subquery.

E.g. Select * From CUSTWhere'2019/03/05' IN (SelectODATE From ORDER Where CUST.CNUM = ORDER.CNUM)

34. Whataretheprimitiveoperationscommontoallrecordmanagementsystems?

Addition,deletionand modification

35. HowdoyoucommunicatewithanRDBMS?

YoucommunicatewithanRDBMS usingStructured QueryLanguage(SQL)

36. DefineSQLandstatethedifferencesbetweenSQLandotherconventional programming Languages

SQL is a nonprocedural language that is designed specifically for data access operations on normalized relational database structures. The primary difference between SQL and other conventional programming languages is that SQL statements specify what data operations should be performed rather than how to perform them.

37. WhatisdatabaseTrigger?

A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can e defined to execute once for the entire statement or once for everyrow that is inserted, updated, or deleted.

38. Whatarestored-procedures?Andwhataretheadvantagesofusing them.

Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and return the result to the client. Stored procedures are used to reduce networktraffic.

39. WhichisthesubsetofSQLcommandsusedtomanipulateDatabasestructures, including tables?

DataDefinitionLanguage(DDL)

40. Whatoperatorperformspatternmatching?

LIKEoperator

41. Whatoperatortestscolumn fortheabsenceofdata?

ISNULLoperator

42. Whatarethewildcardsusedforpatternmatching?

Forsinglecharactersubstitutionand%formulti-charactersubstitutio

43. WhatarethedifferencebetweenTRUNCATEandDELETEcommands?

TRUNCATE	DELETE
• TRUNCATEisaDDLcommand	• DELETEisaDMLcommand
• TRUNCATEoperationcannotbe rolled back	• DELETEoperationcanberolledback
• TRUNCATEdoesnotinvoketrigger	• DELETEdoesinvoketrigger
• TRUNCATEresetsauto_increment value to 0	• DELETE does not resets auto_incrementvalueto0

44. WhatistheuseoftheADDORDROPoptionintheALTERTABLEcommand?

Itisusedtoadd/dropcolumnsoradd/dropconstraintsspecifiedonthetable

45. What isthe useofDESC inSQL?

DESChastwopurposes.It isusedtodescribeaschemaaswellasto retrieverowsfromtablein descending order.

ThequerySELECT*FROMEMPORDERBYENAMEDESCwilldisplaytheoutputsorted on ENAME in descending order

46. WhatistheuseofONDELETECASCADE?

Whenever rows in the master (referenced) table are deleted ,the respective rows of the child (referencing)tablewitha matchingforeignkeycolumnwillgetdeletedaswell.Thisiscalleda cascade delete

ExampleTables:

```
CREATETABLECustomer
 (
customer_idINT(6)PRIMARYKEY,
 cname VARCHAR (100),
 caddressVARCHAR(100)
 );

 CREATETABLEOrder (
 order_idINT(6)PRIMARYKEY,
 products VARCHAR (100),
 payment DECIMAL(10,2),
 customer_id INT (6) ,
 FOREIGNKEY(customer_id)REFERENCESCustomer(customer_id)ONDELETECASCADE
  );
```

Customer is the master table and Order is the child table, where 'customer_id’is primary key in customertableandcustomer_id istheforeignkeyinOrdertableandrepresentsthecustomerwho placed the order. When a row of Customer is deleted, any Order row matching the deleted Customer's customer_id will also be deleted.

47. WhatistheuseofFloor()?

TheFLOOR()functionreturnsthelargestintegervaluethat issmallerthanorequaltoanumber.

EXAMPLE;

SELECTFLOOR(25.75);

OUTPUT

25

48. Whatisthe use ofTruncate()?

TheTRUNCATE() functiontruncatesanumber tothespecifiednumberofdecimalplaces.

EXAMPLE;

SELECTTRUNCATE(135.375,2);

OUTPUT

135.37

49. WhatistheuseofCEILING?

Returnthesmallestinteger valuethatisgreaterthanorequalto25.75:

EXAMPLE;

SELECTCEILING(25.75)

OUTPUT

26

50. WhatyoumeanbySQLUNIQUEConstraint?

- The UNIQUE constraint ensures that all values in a columnare different.
- BoththeUNIQUEandPRIMARYKEYconstraintsprovideaguaranteeforuniquenessfora column or set of columns.
- A PRIMARY KEY constraint automaticallyhas a UNIQUE constraint.
- However,youcanhave manyUNIQUEconstraintspertable,butonlyonePRIMARYKEY constraint per table.

51. HowtoaddanddropUNIQUEConstraintintablein mysql?

ALTERTABLEcontactsADDCONSTRAINTUNC_name_emailUNIQUE(name,email)

ALTER TABLE contacts DROP INDEX UNC_name_email;

52. WhatistheGroupbyClause?

- TheGROUPBYclauseisaSQLcommandthat is usedto grouprowsthat havethesame values.
- The GROUP BY clause is used inthe SELECT statement .Optionallyitis used inconjunction withaggregatefunctionstoproducesummaryreportsfromthedatabase.That'swhat it does, summarizing data from the database.
- ThequeriesthatcontaintheGROUPBYclausearecalledgroupedqueriesandonlyreturn single row for every grouped item.

Example:SELECTCOUNT(CustomerID),CountryFROMCustomers GROUP BY Country

53. Whatisuseofhaving clauseinmysql

- TheHAVINGclauseisusedintheSELECTstatementtospecifyfilterconditionsforagroupof rows or aggregates.
- The HAVING clause is often used with the GROUP BY clause to filter groups based on a specifiedcondition. IftheGROUPBYclauseisomitted,theHAVINGclausebehaveslike the WHERE clause.
- NoticethattheHAVINGclauseappliesafilterconditiontoeachgroupofrows,while the WHERE clause applies the filter condition to each individual row.

Example:SELECTCOUNT(CustomerID),CountryFROMCustomers GROUPBYCountryHAVINGCOUNT(CustomerID)>5;

54. Whatis distinctclausein SQL?

Whenqueryingdatafromatable, youmayget duplicaterows.Inordertoremovetheseduplicate rows, you use the DISTINCT clause in the SELECT statement.

Example:SELECTDISTINCTcolumnsFROMtable_nameWHEREwhere_conditions;

55. Whatisaunion?

Unionscombinetheresultsfrommultiple SELECT queriesintoaconsolidatedresultset.

Theonlyrequirementsforthistoworkisthatthenumberofcolumnsshouldbethesamefrom all the SELECT queries which needs to be combined

56. What isuseofMySQLAggregateFunctions?

- Thedatathat youneed isnot alwaysdirectlystoredinthetables.However, youcanget it by performing the calculations of the stored data when you select it.
- Bydefinition,anaggregatefunctionperformsacalculationonasetofvaluesandreturnsa single value.
- MySQLprovidesmanyaggregatefunctionsthatincludeAVG,COUNT,SUM,MIN,MAX,etc. AnaggregatefunctionignoresNULLvalueswhenitperformscalculationexceptfor the COUNT function.
- Often,aggregatefunctionsareaccompanied bytheGROUPBYclauseoftheSELECT statement

•Below are some of aggregate functions used in sql query

AVGfunction

TheAVG functioncalculatestheaveragevalueofasetofvalues.It ignoresNULLvaluesinthe calculation.

COUNTfunction

TheCOUNT functionreturnsthenumberoftherowsinatable.Forexample, youcanuse below query.below query return number of employees in Employee table

SELECTCOUNT(Empname)FROMEmployee

SUM function

TheSUM functionreturnsthesumofasetofvalues.The SUMfunctionignores NULLvalues. If no matching row found, the SUM function returns a NULL value.

The**SUM** functiontogetthesumofsalaryofemployees intheEmployeetable SELECT

SUM(Salary) FROM Employee

MAXfunction

TheMAXfunctionreturnsthemaximumvalueinasetofvalues. Below

query gets maximum salary of table Employee

SELECTMAX(Salary)FROMEmployee

MINfunction

TheMINfunctionreturnstheminimumvalueinasetofvalues.

Below query returns minimum salary of table Employee

SELECT MIN(Salary) FROM Employee

CREATEcommand

```
CREATETABLEEmployee
(
Empnoint(4)primarykey,
Empnamevarchar(50), job
varchar(40),
Hiredatedate,
Salarydecimal(10,2),
Deptno int(7),
Age int(10)
);
```

DESCcommand

DESCEmployee;

```
mysql> DESC Employee;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| Empno    | int(4)        | NO   | PRI | NULL    |       |
| Empname  | varchar(50)   | YES  |     | NULL    |       |
| job      | varchar(40)   | YES  |     | NULL    |       |
| Hiredate | date          | YES  |     | NULL    |       |
| Salary   | decimal(10,2) | YES  |     | NULL    |       |
| Deptno   | int(7)        | YES  |     | NULL    |       |
| Age      | int(10)       | YES  |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

INSERTcommand

Insertthevalues intothetableasspecified.

1) InsertintoEmployee values(1000,'Hemanth','Manager','2018-11-17',35000, 30, 38);
2) InsertintoEmployeevalues(1001, 'Nitin','Manager','2018-05-01',45000, 10, 42);
3) InsertintoEmployeevalues(1002,' Sachin','Salesman','2018-01-09',18000,20, 28);
4) InsertintoEmployeevalues(100 3, 'Deepak','Clerk','2018-05-15',15000,40, 34);
5) InsertintoEmployeevalues(10 04, 'Ajay','Analyst','2018-10-22',60000,50, 45);
6) InsertintoEmployeevalues(1005, 'Arun','Programmer','2018-7-24',25000, 60,25);

```
mysql> select * from Employee;
+-------+---------+------------+------------+----------+--------+------+
| Empno | Empname | job        | Hiredate   | Salary   | Deptno | Age  |
+-------+---------+------------+------------+----------+--------+------+
|  1000 | Hemanth | Manager    | 2018-11-17 | 35000.00 |     30 |   38 |
|  1001 | Nitin   | Manager    | 2018-05-01 | 45000.00 |     10 |   42 |
|  1002 | Sachin  | Salesman   | 2018-01-09 | 18000.00 |     20 |   28 |
|  1003 | Deepak  | Clerk      | 2018-05-15 | 15000.00 |     40 |   34 |
|  1004 | Ajay    | Analyst    | 2018-10-22 | 60000.00 |     50 |   45 |
|  1005 | Arun    | Programmer | 2018-07-24 | 25000.00 |     60 |   25 |
+-------+---------+------------+------------+----------+--------+------+
6 rows in set (0.00 sec)
```

Queries:Problemso
nselectcommand:

1) **Display the details of all managers of** Employee Table

SELECT*FROMEmployeeWHEREjob='Manager';

2) **Displaythedetailsofallemployeesgetting salary lessthan 30,000.**

SELECT*FROMEmployee WHEREsalary<30000;

3) **Displaythedetailsofemployeeswhoageisbetween35and 45**

SELECT*FROMEmployeeWHEREageBETWEEN35AND45;

4) **DisplaythedetailsofClerkswhohavejoinedafter01-MAR-05**.

SELECT * FROM Employee WHERE job='Clerk' AND hiredate>'2018-03-05';

5) **Sortthedetailsindescendingorderof**Empno**.**

SELECT*FROMEmployeeORDERBYEmpnoDESC;

6) **Sortthedetailsofemployeesinascendingorderof name**

SELECT*FROMEmployeeORDERBYEmpname

7) **Displaythedetailsofemployeeswhosenamescontain 'i' in them.**

SELECT*FROMEmployeeWHEREEmpnameLIKE'%i%';

8) **Display the details of employees whose namesstarts with'a' in them.**

SELECT*FROMEmployeeWHEREEmpnameLIKE 'a%';

9) **Displaythedetailsofemployeeswhosenamesdoesnotstartswith 'a'in them.**

SELECT*FROMEmployeeWHEREEmpname NOTLIKE'a%';

10) **Displaytheemployeedetailswhosenameshaveexactly4 characters.**

SELECT*FROMEmployeeWHERElength(Empname)=4

11) **Copyalltherecordsoftheirfromemployeetableandinserttherecordsintoatemptable with column names same as in Employee table**

CREATETABLETEMPSELECT*FROMEmployee;

Problemsonupdatecommand:

1. **Updatethesalaryby10%hiketoManagerworkingindepartmentnumber20and30**

   SOL:UPDATEEMPSETSAL=SAL*1.1 WHEREDeptnoIN(20,30)ANDJOB =‘ Manager’;

2. **Give5%raiseinsalaryto alltheSalesman**

   SOL1:UPDATEEMPLOYEESETSalary=Salary*1.15WHEREJOB='Salesman';

   OR

   SOL2:UPDATEEMPLOYEESETSalary=Salary+(Salary*15/100)WHERE JOB='Salesman';

3. **ChangethedepartmentnoofSachinto40**

   SOL: UPDATEEMP SETDEPTNO=40WHEREEmpname= ‘Sachin;

4. **Updateallemployeenametouppercase**

   SOL:UPDATEEMPLOYEESETEmpname=upper(Empname);

Problemsondeletecommand:

1. **Deleteallthe recordsofemployees**

   SOL:DELETEFROMEmployee;

2. **Delete the records of employee name Ajay’s only**

   SOL: DELETE FROM EMP WHERE ENAME = ‘Ajay’;

3. **Deletetherecord ofemployeetablewhoseEmpnois1005**

   SOL:DELETEFROMEMP WHEREEmpno=1005;

4. **Deletethefirstfiverecordsofemployeetable**

   SOL:DELETEFROMEMPLOYEELIMIT5;

ALTERcommand

1. **Howtocreatedatabasename** COLLEGE**?**

   CREATEDATABASECOLLEGE;

2. **HowModifydatetypeofagecolumninEmployeetable**

   ALTERTABLEEmployeeMODIFYageint(3);

3. **HowtorenamecolumnnameofjobtoDesignation inEmployeetable?**

   ALTERTABLEEmployee CHANGE jobDesignationvarchar(40);

4. **HowtoaddcolumnCommission inEmployeetable?**

   ALTERTABLEEmployee addCommissionvarchar(40);

5. **HowtodropcolumnCommissioninEmployee table?**

   ALTERTABLEEmployeeDROPcolumnCommission;

6. **Howtoaddprimarykeyto Employee table?**

   ALTERTABLEEmployeeaddprimarykey(Empno);

7. **HowtodropprimarykeytoEmployeetable?**

   ALTERTABLEEmployeeDROPprimarykey;

8. **Howtorenameemployeetable?**

   RENAMETABLEEmployeetoEmployee_Details

9. **HowtodeletecontentsofEmployee table?**

   DELETEFROMEmployee;

   OR

   TRUNCATEEmployee;

10. **HowtodropEmployeetable?**

    DROPTABLEEmployee;

11. **Howtodropdatabasename**COLLEGE**?**

    DROP DATABASE COLLEGE

VIVAQUESTIONS

1. Whatis Database?
2. Whatis DBMS?
3. Whatis RDBMS?
4. HowisitdifferentfromDBMS?
5. WhatisSQL?
6. WhatisthedifferencebetweenSQLandMySQL?
7. WhatareTablesandFields?
8. WhatareConstraintsinSQL?
9. WhatisaPrimary Key?
10. WhatisaUNIQUE constraint?
11. WhatisaForeign Key?
12. WhatisaJoin?Listitsdifferent types.
13. WhatisaSelf-Join?
14. WhatisaCross-Join?
15. WhatisanIndex?Explainitsdifferenttypes.
16. WhatisthedifferencebetweenClusteredandNon-clustered index?
17. WhatisData Integrity?
18. Whatisa Query?
19. WhatisaSubquery?Whatareits types?
19. WhatistheSELECTstatement?
20. WhataresomecommonclausesusedwithSELECTqueryinSQL?
21. WhatareUNION,MINUSandINTERSECTcommands?
22. WhatisCursor?Howtousea Cursor?
23. WhatareEntitiesandRelationships?
24. Listthedifferenttypesof relationshipsinSQL.
25. Whatisan Aliasin SQL?
26. What is a View?
27. What is Normalization?
28. What is Denormalization?
29. Whatarethe variousforms of Normalization?
30. WhataretheTRUNCATE,DELETEandDROPstatements?
31. Whatisthe differencebetweenDROP andTRUNCATEstatements?
32. WhatisthedifferencebetweenDELETE andTRUNCATEstatements?
33. WhatareAggregateand Scalarfunctions?
34. WhatisUser-defined function?Whatareitsvarioustypes?
35. What is OLTP?

36. Whatarethedifferences betweenOLTP andOLAP?
37. WhatisCollation?Whatarethedifferenttypesof Collation Sensitivity?
38. Whatis aStored Procedure?
39. WhatisaRecursiveStoredProcedure?
40. Howto create emptytables withthe same structureas another table?