**Chapter 6:**

**1) Write a database description for each of the relations shown, using SQL DDL. (shorten, abbreviate, or change any data names as need for your SQL version). Assume the following attribute data types:**

STUDENT\_ID (integer, primary key)

STUDENT\_NAME (25 characters)

FACULTY\_ID (integer, primary key)

FACULTY\_NAME (25 characters)

COURSE\_ID (8 characters, primary key)

COURSE\_NAME (15 characters)

DATE\_QUALIFIED (date)

SECTION\_ID (integer, primary key)

SEMESTER (7 characters)

Sol) The following are the Queries for each of the relations given,

**Query 1:**

Below is the query for creation of Student Table with fields Student’s Id & Name, here Student ID is the primary Key.

CREATE TABLE Student

(StudentID NUMBER NOT NULL,

StudentName VARCHAR2(25),

CONSTRAINT Student\_PK PRIMARY KEY (StudentID));

**Query 2:**

The query gives the creation of the table faculty with fields containing Name & Id of the Faculty’s with ID being the primary key.

CREATE TABLE Faculty

(FacultyID INTEGER NOT NULL,

FacultyNameVARCHAR2(25),

CONSTRAINT Faculty\_PK PRIMARY KEY (FacultyID));

**Query 3:**

Below Query creates the table Course containing Course Id & Name where ID acts as a primary key.

CREATE TABLE Course

(CourseID CHAR (8) NOT NULL,

CourseNameVARCHAR2(15),

CONSTRAINT Course\_PK PRIMARY KEY (CourseID));

**Query 4:**

CREATE TABLE Qualified

(FacultyID INTEGER NOT NULL,

CourseID CHAR (8) NOT NULL,

DateQualified DATE,

CONSTRAINT Qualified\_PK PRIMARY KEY (FacultyID, CourseID),

CONSTRAINT Qualified\_FK FOREIGN KEY (FacultyID) REFERENCES Faculty(FacultyID),

CONSTRAINT Qualified\_FK FOREIGN KEY (CourseID) REFERENCES Course(CourseID));

The above query explains the Qualified table with fields containing the Id’s of faculty &Course, qualified date. Here the primary keys facultyId & CourseId are used a foreign keys and the Reference clause is used in order to prevent the foreign value of the Id’s so that the relation of the qualified is not ruined with the Course & faculty tables.

**Query 5:**

CREATE TABLE Section

(SectionNo INTEGER NOT NULL,

Semester CHAR (7) NOT NULL,

CourseID CHAR (8) NOT NULL,

CONSTRAINT Section\_PK PRIMARY KEY (SectionNo, Semester, CourseID),

CONSTRAINT Section\_FK FOREIGN KEY (CourseID) REFERENCES Course(CourseID));

Section table is created using the above query with section no, semester & Id’s of the course fields. The primary key course Id is used as foreign key with the use of Reference clause to prevent the foreign value so that the relation between section and course is not effected.

**Query 6:**

CREATE TABLE Registration

(StudentID INTEGER NOT NULL,

SectionNo INTEGER NOT NULL,

Semester CHAR (7) NOT NULL,

CONSTRAINT RegistrationPK PRIMARY KEY (StudentID, SectionNo, Semester),

CONSTRAINT RegistrationFK FOREIGN KEY (StudentID) REFERENCES Student(StudentID),

CONSTRAINT RegistrationFK FOREIGN KEY (SectionNo) REFERENCES Section(SectionNo),

CONSTRAINT RegistrationFK FOREIGN KEY (Semester) REFERENCES Section(Semester));

The query above is created with the Id’s of the student, section number and semester fields with registration as primary key & the use of student ID & section no primary keys are used as foreign keys with reference clause in order to prevent the foreign value to ruin the relation between the tables section, student & registration tables.

**5) Write SQL commands for the following:**

**a. Create two different forms of the INSERT command to add a student with a student ID of 65798 and last name Lopez to the Student table.**

INSERT INTO Student (StudentID, StudentName) VALUES (65798,’Lopez’);

Or

INSERT INTO Student VALUES (65798,’Lopez’);

**b. Now write a command that will remove Lopez from the Student table.**

DELETE FROM Student WHERE StudentID = 65798;

**c. Create an SQL command that will modify the name of course ISM 4212 from Database to Introduction to Relational Databases.**

UPDATE Course SET CourseName = ‘Introduction to Relational Databases’ WHERE CourseID = ‘ISM 4212’;

**6. Write SQL quarries to answer the following questions:**

**a. Which students have an ID number that is less than 50000?**

SELECT StudentID, StudentName FROM Student WHERE (((StudentID)<50000));

**b. What is the name of the faculty member who’s ID is 4756?**

SELECT FacultyID, FacultyName FROM Faculty WHERE (((FacultyID)=4756));

**c. What is the smallest section number used in the first semester of 2008?**

SELECT SectionNo, Semester FROM Section WHERE (((SectionNo)<=2712));

**12. How many students were matched with someone in the first five months of the year?**

The use of Count in the query gives the value for the students who were matched in the first five months as follows:

SELECT COUNT(StudentID) AS CountOfStudent from MatchHistory WHERE StartDate BETWEEN ’01-JAN-2008’ AND ’31-MAY-2008’;

**Output:**

CountOfStudent **3**

**13. Which student has the highest Read score?**

Query to retrieve the student with highest Read score is as follows:

SELECT StudentID, Read From Student WHERE Read = (SELECT MAX(Read) FROM Student);

**Output:**

StudentID Read

3006 7.8

**Chapter 7:**

**7. Write the SQL command to add MATH SCORE to the Student table?**

The ALTER command is used to add columns, delete or modify in a table.

**Query**:

ALTER Table STUDENT ADD COLUMN Math Score Number (3,1);

**8. Write the SQL command to add SUBJECT to TUTOR. The only values allowed for SUBJECT will be Reading, Math and ESL.**

ALTER Table TUTOR ADD COLUMN Subject Varchar (7) CHECK (Subject IN (‘Reading’, ‘Math’, ‘ESL’));

**12. List all active students in June by name. (Make up names and other data if you are actually building a prototype database.) Include the number of hour’s students received tutoring and how many lessons they completed.**

**Query:**

SELECT S.StudentID, M.EndDate, P.LastName,

SUM(T.Hours) AS THrs,

SUM(T.Lessons) AS TLs

From (PERSON P INNER JOIN Student S ON P.Person\_ID = S.StudentID)

INNER JOIN (MatchHistory M LEFT JOIN TutorReport T ON M.MatchID = T.MatchID)

ON S.StudentID = M.StudentID

GROUP BY S.StudentID, M.EndDate, P.LastName

HAVING (((M.EndDate) Is Null));

**14. which tutor needs to be reminded to turn in reports? Write the SQL command. Show how you constructed this query using a Venn or the other type of diagram.**

**Query:**

SELECT MatchHistory.TutorID From MatchHistory

WHERE MatchHistory.MatchID NOT IN (SELECT DISTINCT TutorReport.MatchID

From TutorReport);

**Sub query to get unique Id’s of Match**

**Tutor Reports**

Not in present set of reports (of tutor)

Retrieving ID of tutor from the Tutor Reports by the basis of reports of Id’s in match that are missing