SQL HANDS ON

Pavithra Jose Vattakavumkal
2211585
CDB22DW022

```
SHOW DATABASES;
CREATE DATABASE SQL_Hands_On;
USE SQL_Hands_On;
```

EXERCISE 1

Problem Statement:

Create tables

Command

```
CREATE TABLE Trainer_Info(
Trainer_Id VARCHAR(20) PRIMARY KEY UNIQUE,
Salutation VARCHAR(7),
Trainer_Name VARCHAR(30),
Trainer_Location VARCHAR(30),
Trainer_Track VARCHAR(15),
Trainer_Qualification VARCHAR(100),
Trainer_Experience INTEGER,
Trainer_Email VARCHAR(100) UNIQUE,
Trainer_Password VARCHAR(20));

CREATE TABLE Batch_Info(
Batch_Id VARCHAR(20) PRIMARY KEY UNIQUE,
Batch_Owner VARCHAR(30),
Batch_BU_Name VARCHAR(30));

CREATE TABLE Module_Info(
Module_Id VARCHAR(20) PRIMARY KEY UNIQUE,
Module_Name VARCHAR(40),
Module_Duration INTEGER);

CREATE TABLE Associate_Info(
Associate_Id VARCHAR(20) PRIMARY KEY UNIQUE,
Salutation VARCHAR(7),
Associate_Name VARCHAR(30),
Associate_Location VARCHAR(30),
Associate_track VARCHAR(15),
Associate_Qualification VARCHAR(200),
Associate_Email VARCHAR(100) UNIQUE,
Associate_Password VARCHAR(20));

CREATE TABLE Questions(
Question_Id VARCHAR(20) PRIMARY KEY UNIQUE,
Module_Id VARCHAR(20),
Question_Text VARCHAR(900),
FOREIGN KEY (Module_Id) REFERENCES Module_info(Module_Id));

CREATE TABLE Associate_Status(
Associate_Id VARCHAR(20),
Module_Id VARCHAR(20),
Start_Date VARCHAR(20),
End_Date VARCHAR(20),
AFeedbackGiven VARCHAR(20),
TFeedbackGiven VARCHAR(20),
FOREIGN KEY (Associate_Id) REFERENCES Associate_Info(Associate_Id),
FOREIGN KEY (Module_Id) REFERENCES Module_Info(Module_Id));



CREATE TABLE Trainer_Feedback(
Trainer_Id VARCHAR(20),
Question_Id VARCHAR(20),
Batch_Id VARCHAR(20),
Module_Id VARCHAR(20),
Trainer_Rating INTEGER,
FOREIGN KEY (Trainer_Id) REFERENCES Trainer_Info(Trainer_Id),
FOREIGN KEY (Question_Id) REFERENCES Questions(Question_Id),
FOREIGN KEY (Batch_Id) REFERENCES Batch_Info(Batch_Id),
FOREIGN KEY (Module_Id) REFERENCES Module_Info(Module_Id));

CREATE TABLE Associate_Feedback(
Associate_Id VARCHAR(20),
Question_Id VARCHAR(20),
Module_Id VARCHAR(20),
Associate_Rating INTEGER,
FOREIGN KEY (Associate_Id) REFERENCES Associate_Info(Associate_Id),
FOREIGN KEY (Question_Id) REFERENCES Questions(Question_Id),
FOREIGN KEY (Module_Id) REFERENCES Module_Info(Module_Id));

CREATE TABLE Login_Details(
User_Id VARCHAR(20) UNIQUE PRIMARY KEY NOT NULL,
User_Password VARCHAR(20) NOT NULL);

DESCRIBE Trainer_Info;
DESCRIBE Batch_Info;
DESCRIBE Module_Info;
DESCRIBE Associate_Info;
DESCRIBE Questions;
DESCRIBE Associate_Status;
DESCRIBE Trainer_Feedback;
DESCRIBE Associate_Feedback;
DESCRIBE Login_Details;
```

```
ALTER TABLE Associate_Status
Add Batch_Id VARCHAR(20) AFTER Module_ID,
ADD Trainer_Id VARCHAR(20) AFTER Batch_Id,
ADD FOREIGN KEY (Batch_Id) REFERENCES Batch_Info(Batch_Id),
ADD FOREIGN KEY (Trainer_Id) REFERENCES Trainer_Info(Trainer_Id);
```

EXERCISE 2

Problem Statement:

Insert details into table

Command

```
INSERT INTO Trainer_Info(Trainer_Id, Salutation, Trainer_Name, Trainer_Location, Trainer_Track,
Trainer_Qualification, Trainer_Experience, Trainer_Email, Trainer_Password)
VALUES ('F001', 'Mr.', 'PANKAJ GHOSH','Pune','Java','Bachelor of Technology', 12
,'Pankaj.Ghosh@alliance.com','fac1@123'),
('F002','Mr.','SANJAY RADHAKRISHNAN','Bangalore','DotNet','Bachelor of
Technology',12,'Sanjay.Radhakrishnan@alliance.com','fac2@123'),
('F003','Mr.','VIJAY MATHUR','Chennai','Mainframe','Bachelor of
Technology',10,'Vijay.Mathur@alliance.com','fac3@123'),
('F004','Mrs.','NANDINI NAIR','Kolkata','Java','Master of Computer
Applications',9,'Nandini.Nair@alliance.com','fac4@123'),
('F005','Miss.','ANITHA PAREKH','Hyderabad','Testing','Master of Computer
Applications',6,'Anitha.Parekh@alliance.com','fac5@123'),
('F006','Mr.','MANOJ AGRAWAL' ,'Mumbai','Mainframe','Bachelor of
Technology',9,'Manoj.Agrawal@alliance.com','fac6@123'),
('F007','Ms.','MEENA KULKARNI','Coimbatore','Testing','Bachelor of
Technology',5,'Meena.Kulkarni@alliance.com','fac7@123'),
('F009','Mr.','SAGAR MENON' ,'Mumbai','Java','Master of Science In Information
Technology',12,'Sagar.Menon@alliance.com','fac8@123');

INSERT INTO Batch_Info(Batch_Id, Batch_Owner, Batch_BU_Name)
VALUES('B001','MRS.SWATI ROY','MSP'),
('B002','MRS.ARURNA K','HEALTHCARE'),
('B003','MR.RAJESH KRISHNAN','LIFE SCIENCES'),
('B004','MR.SACHIN SHETTY','BFS'),
('B005','MR.RAMESH PATEL','COMMUNICATIONS'),
('B006','MRS.SUSAN CHERIAN','RETAIL & HOSPITALITY'),
('B007','MRS.SAMPADA JAIN','MSP'),
('B008','MRS.KAVITA REGE','BPO'),
('B009','MR.RAVI SEJPAL','MSP');

INSERT INTO Module_Info(Module_Id, Module_Name, Module_Duration)
VALUES ('O10SQL','Oracle 10g SQL',16),
('O10PLSQL','Oracle 10g PL/ SQL',16),
('J2SE','Core Java SE 1.6',288),
('J2EE','Advanced Java EE 1.6',80),
('JAVAFX','JavaFX 2.1',80),
('DOTNT4','.Net Framework 4.0',50),
('SQL2008','MS SQl Server 2008',120),
('MSBI08','MS BI Studio 2008',158),
('SHRPNT','MS Share Point',80),
('ANDRD4','Android 4.0',200),
('EM001','Instructor',0),
('EM002','Course Material',0),
('EM003','Learning Effectiveness',0),
('EM004','Environment',0),
('EM005','Job Impact',0),
('TM001','Attendees',0),
('TM002','Course Material',0),
('TM003','Environment',0);

INSERT INTO Associate_Info(Associate_Id, Salutation, Associate_Name, Associate_Location,
Associate_Track, Associate_Qualification, Associate_Email, Associate_Password)
VALUES ('A001','Miss.','GAYATHRI NARAYANAN','Gurgaon','Java','Bachelor of
Technology','Gayathri.Narayanan@hp.com','tne1@123'),
('A002','Mrs.','RADHIKA MOHAN','Kerala','Java','Bachelor of Engineering In Information
Technology','Radhika.Mohan@cognizant.com','tne2@123'),
('A003','Mr.','KISHORE SRINIVAS','Chennai','Java','Bachelor of Engineering In
Computers','Kishore.Srinivas@ibm.com','tne3@123'),
('A004','Mr.','ANAND RANGANATHAN','Mumbai','DotNet','Master of Computer
Applications','Anand.Ranganathan@finolex.com','tne4@123'),
('A005','Miss.','LEELA MENON','Kerala','Mainframe','Bachelor of Engineering In Information
Technology','Leela.Menon@microsoft.com','tne5@123'),
('A006','Mrs.','ARTI KRISHNAN','Pune','Testing','Master of Computer
Applications','Arti.Krishnan@cognizant.com','tne6@123'),
('A007','Mr.','PRABHAKAR SHUNMUGHAM','Mumbai','Java','Bachelor of
Technology','Prabhakar.Shunmugham@honda.com','tne7@123');

INSERT INTO Questions(Question_Id,Module_Id,Question_Text)
VALUES
('Q001','EM001','Instructor knowledgeable and able to handle all your queries'),
('Q002','EM001','All the topics in a particular course handled by the trainer without any gaps or
slippages'),
('Q003','EM002','The course materials presentation, handson,  etc. refered during the training are
relevant and useful.'),
('Q004','EM002','The Hands on session adequate enough to grasp the understanding of the topic.'),
('Q005','EM002','The reference materials suggested for each module are adequate.'),
('Q006','EM003','Knowledge and skills presented in this training are applicatible at your work'),
('Q007','EM003','This training increases my proficiency level'),
('Q008','EM004','The physical environment e.g. classroom space, air-conditioning was conducive to
learning.'),
('Q009','EM004','The software/hardware environment provided was sufficient for the purpose of the training.'),
('Q010','EM005','This training will improve your job performance.'),
('Q011','EM005','This training align with the business priorities and goals.'),
('Q012','TM001','Participants were receptive and had attitude towards learning.'),
('Q013','TM001','All participats gained the knowledge and the practical skills after this training.'),
('Q014','TM002','The course materials presentation, handson,  etc. available for the session covers the entire objectives of the course.'),
('Q015','TM002','Complexity of the course is adequate for the particpate level.'),
('Q016','TM002','Case study and practical demos helpful in understanding of the topic'),
('Q017','TM003','The physical environment e.g. classroom space, air-conditioning was conducive to learning.'),
('Q018','TM003','The software/hardware environment provided was adequate  for the purpose of the training.');

INSERT INTO Associate_Status(Associate_Id,Module_Id,Batch_Id,Trainer_Id,Start_Date,End_Date)
VALUES('A001','O10SQL','B001','F001','2000-12-15','2000-12-25'),
('A002','O10SQL','B001','F001','2000-12-15','2000-12-25'),
('A003','O10SQL','B001','F001','2000-12-15','2000-12-25'),
('A001','O10PLSQL','B002','F002','2001-2-1','2001-2-12'),
('A002','O10PLSQL','B002','F002','2001-2-1','2001-2-12'),
('A003','O10PLSQL','B002','F002','2001-2-1','2001-2-12'),
('A001','J2SE','B003','F003','2002-8-20','2002-10-25'),
('A002','J2SE','B003','F003','2002-8-20','2002-10-25'),
('A001','J2EE','B004','F004','2005-12-1','2005-12-25'),
('A002','J2EE','B004','F004','2005-12-1','2005-12-25'),
('A003','J2EE','B004','F004','2005-12-1','2005-12-25'),
('A004','J2EE','B004','F004','2005-12-1','2005-12-25'),
('A005','JAVAFX','B005','F006','2005-12-4','2005-12-20'),
('A006','JAVAFX','B005','F006','2005-12-4','2005-12-20'),
('A006','SQL2008','B006','F007','2007-6-21','2007-6-28'),
('A007','SQL2008','B006','F007','2007-6-21','2007-6-28'),
('A002','MSBI08','B007','F006','2009-6-26','2009-6-29'),
('A003','MSBI08','B007','F006','2009-6-26','2009-6-29'),
('A004','MSBI08','B007','F006','2009-6-26','2009-6-29'),
('A002','ANDRD4','B008','F005','2010-6-5','2010-6-28'),
('A005','ANDRD4','B008','F005','2010-6-5','2010-6-28'),
('A003','ANDRD4','B009','F005','2011-8-1','2011-8-20'),
('A006','ANDRD4','B009','F005','2011-8-1','2011-8-20');
```

EXERCISE 3

Problem Statement:
Change the password of trainer F004 from fac4@123 to nn4@123

Command

```
SELECT * FROM Trainer_Info;

UPDATE Trainer_Info
SET Trainer_Password = 'nn4@123'
WHERE Trainer_Id = 'F004';
```

EXERCISE 4

Problem Statement:
Remove following record from Associate_Status table
A003, J2EE, B004, F004, 2005-12-1, 2005-12-25

Command

```
SELECT * FROM Associate_Status;

DELETE FROM Associate_Status
WHERE Associate_Id = 'A003' AND Module_Id='J2EE'AND Batch_Id='B004' AND Trainer_Id=
'F004' AND Start_Date='2005-12-1' AND End_Date='2005-12-25';
```

EXERCISE 5

Problem Statement:
Fetch first five trainers who have maximum years of experience and display their details

Command

```
SELECT * FROM Trainer_Info;
ORDER BY Trainer_Experience DESC
LIMIT 5;
```

EXERCISE 6

Problem Statement:
Begin transaction & insert below to records into Login_Details
'U001' Admin1@123
'U002' Admin2@123
Perform rollback operation & verify whether any records are inserted in table or not.

Command

```
SELECT * FROM Login_Details;

START Transaction;

INSERT INTO Login_Details(User_Id, User_Password)
VALUES ('U001','Admin1@123'),
('U002', 'Admin2@123');

ROLLBACK;

SELECT * FROM Login_Details;
```

EXERCISE 7

Problem Statement:
Create a dummy user in database. Grant create & select table privilege to him/her. Repeat the above all queries using login credentials of newly created user. Revoke the privilege assigned to this newly created user.

Command

```
CREATE USER 'pavi'@'localhost' IDENTIFIED BY 'password';
GRANT CREATE ON SQL_Hands_On TO 'pavi'@'localhost';
GRANT SELECT ON SQL_Hands_On.Login_Details TO 'pavi'@'localhost';

SELECT *
FROM Login_Details;

START TRANSACTION;

INSERT INTO Login_Details(User_Id,User_Password)
VALUES('U003','Admin1@123'),
('U004','Admin2@123');

ROLLBACK;

SELECT *
FROM Login_Details;

REVOKE CREATE ON sql_hands_on FROM 'pavi'@'localhost';
REVOKE SELECT ON sql_hands_on.Login_Details FROM 'pavi'@'localhost';
```

EXERCISE 8

Problem Statement:
Remove table Login_Details from database.

Command

```
Drop table Login_Details;
```

EXERCISE 9

Problem Statement:
Create a table called suppliers that stores supplier ID, name, and address information

Command

```
CREATE TABLE Suppliers(
supplier_id integer PRIMARY KEY NOT NULL,
supplier_name VARCHAR(50) NOT NULL,
address VARCHAR(50),
city VARCHAR(50),
state VARCHAR(25),
zip_code VARCHAR(10));
```

EXERCISE 10

Problem Statement:
Display all the unique courses between course fees and course fees_history.

Command

```
CREATE TABLE Course_Fees(
COURSE_CODE VARCHAR(20) PRIMARY KEY UNIQUE NOT NULL,
BASE_FEES INTEGER,
SPECIAL_FEES INTEGER,
DISCOUNT INTEGER);

CREATE TABLE Course_Fees_History(
COURSE_CODE VARCHAR(20) PRIMARY KEY UNIQUE NOT NULL,
BASE_FEES INTEGER,
SPECIAL_FEES INTEGER,
CREATED_BY VARCHAR(20),
Updated_By VARCHAR(20));

INSERT INTO Course_Fees(COURSE_CODE, BASE_FEES, SPECIAL_FEES, DISCOUNT)
VALUES(1,180,100,10),
(2,150,110,10),
(3,160,170,5),
(4,150,100,10),
(6,190,100,40);

INSERT INTO
Course_Fees_History(COURSE_CODE,BASE_FEES,SPECIAL_FEES,CREATED_BY,Updated_By
)
VALUES(1,120,123,'Ram','Ramesh'),
(2,150,110,'Bala','Ram'),
(3,160,170,'Bala','Vinu'),
(4,170,235,'Ram','Ram'),
(6,190,100,'Vinod','Vinod');

SELECT Course_Code,Base_Fees,Special_fees;
FROM course_fees,
UNION
SELECT Course_Code,Base_Fees,Special_fees,
FROM Course_Fees_History;
```

EXERCISE 11

Problem Statement:
Check uniqueness of Course_Code, BASE_FEES and SPECIAL_FEES of courses in both COURSE_FEES and COURSE_FEES_HISTORY tables.

Command

```
SELECT COUNT(*) FROM COURSE_FEES;

SELECT COUNT(DISTINCT(COURSE_CODE)) FROM COURSE_FEES;
SELECT COUNT(DISTINCT(BASE_FEES)) FROM COURSE_FEES;
SELECT COUNT(DISTINCT(SPECIAL_FEES)) FROM COURSE_FEES;

SELECT COUNT(*) FROM COURSE_FEES_HISTORY;

SELECT COUNT(DISTINCT(COURSE_CODE)) FROM COURSE_FEES_HISTORY;
SELECT COUNT(DISTINCT(BASE_FEES)) FROM COURSE_FEES_HISTORY;
SELECT COUNT(DISTINCT(SPECIAL_FEES)) FROM COURSE_FEES_HISTORY;
```

EXERCISE 12

Problem Statement:
Display the minimum and maximum base fees of the courses.

Command

```
CREATE TABLE COURSE_INFO(
COURSE_CODE VARCHAR(10) PRIMARY KEY UNIQUE NOT NULL,
COURSE_NAME VARCHAR(20) NOT NULL,
COURSE_DESCRIPTION VARCHAR(25),
COURSE_START_DATE DATE,
COURSE_DURATION INTEGER,
NO_OF_PARTICIPANTS INTEGER,
COURSE_TYPE CHAR(3));

CREATE TABLE Student_Info(
Student_Id VARCHAR(10) UNIQUE PRIMARY KEY NOT NULL,
First_name VARCHAR(20),
Last_name VARCHAR(25),
Address VARCHAR(150));

INSERT INTO COURSE_FEES(COURSE_CODE,BASE_FEES,SPECIAL_FEES,DISCOUNT)
VALUES(7,NULL,120,15),
(8,NULL,140,20),
(9,300,180,50),
(10,175,150,30);

SELECT MIN(BASE_FEES) AS MIN_BASE_FEES, MAX(BASE_FEES) AS MAX_BASE_FEES
FROM COURSE_FEES;
```

EXERCISE 13

Problem Statement:
Display the average infra fees of the courses.

Command

```
ALTER TABLE Course_Fees ADD infra_fees DECIMAL(5,3);
UPDATE Course_Fees SET infra_fees = 45.751 WHERE COURSE_CODE = 1;
UPDATE Course_Fees SET infra_fees = 43.453 WHERE COURSE_CODE = 2;
UPDATE Course_Fees SET infra_fees = 43.651 WHERE COURSE_CODE = 3;
UPDATE Course_Fees SET infra_fees = 46.553 WHERE COURSE_CODE = 4;
UPDATE Course_Fees SET infra_fees = 45.751 WHERE COURSE_CODE = 6;
UPDATE Course_Fees SET infra_fees = 35.453 WHERE COURSE_CODE = 7;
UPDATE Course_Fees SET infra_fees = 42.751 WHERE COURSE_CODE = 8;
UPDATE Course_Fees SET infra_fees = 40.654 WHERE COURSE_CODE = 9;
UPDATE Course_Fees SET infra_fees = 36.654 WHERE COURSE_CODE = 10;

SELECT AVG(infra_fees) FROM Course_Fees;
```

EXERCISE 14

Problem Statement:

Develop a query which will display the course name and the number of days between the current date and course start date in Course_Info table.

Command

```
INSERT INTO
Course_Info(Course_Code,Course_Name,Course_Description,Course_Start_Date,Course_Duration,N
o_of_Participants,Course_Type)
VALUES (1,'SQL1','Introduction_SQL','2022-10-02',2,5,'CLR'),
(2,'SQL2','Intermediate_SQL','2022-10-20',2,10,'EL'),
(3,'SQL3','Advanced_SQL','2022-10-25',2,25,'OF'),
(4,'Python1','Introduction_Python','2022-11-01',5,5,'CLR'),
(6,'Python2','Intermediate_Python','2022-11-10',5,10,'EL'),
(7,'Python3','Advanced_Python','2022-11-15',5,25,'OF'),
(8,'SPSS1','INtroduction_SPSS','2022-10-01',3,5,'CLR'),
(9,'SPSS2','Intermediate_SPSS','2022-10-10',3,10,'EL'),
(10,'SPSS3','Advanced_SPSS','2022-10-15',3,25,'OF'),
(165,'R1','Introduction_R','2022-10-01',7,5,'CLR'),
(166,'R2','Intermediate_R','2022-10-10',7,10,'EL'),
(167,'R3','Advanced_R','2022-10-15',7,25,'OF');

SELECT Course_Name,DATEDIFF(Course_Start_Date,CURDATE())
FROM Course_Info;
```

EXERCISE 15

Problem Statement:

Develop a query which will concatenate the Course Name and Course Code in the following format and display all the courses in the course_info table.
“< Course Name><Course Code>”

Command

```
SELECT CONCAT('<',Course_Name,'><',Course_code,'>') FROM COURSE_INFO;
```

EXERCISE 16

Problem Statement:

Develop a query to calculate average of all the base fees, any records whose base fee is null needs to be considered as zero.

Command

```
SELECT AVG(IFNULL(BASE_FEES,0)) FROM COURSE_FEES;
```

EXERCISE 17

Problem Statement:
Query to display course type and appropriate message.

Command

```
SELECT Course_Type,
CASE WHEN Course_Type = 'CLR' THEN 'Class Room'
          WHEN Course_Type = 'EL' THEN 'ELearning'
ELSE 'Offline Reading'
END AS Message
FROM Course_Info;
```

EXERCISE 18

Problem Statement:
Develop a query which would retrieve the total number of students enrolled for courses on a specific date grouped by course start date and display course start date and total number of students.

Command

```
SELECT COURSE_START_DATE,SUM(NO_OF_PARTICIPANTS) FROM COURSE_INFO
GROUP BY COURSE_START_DATE
HAVING COURSE_START_DATE ='2022-10-25';
```

EXERCISE 19

Problem Statement:
Develop a query which would retrieve the total number of students enrolled for courses where course_type="CLR" grouped by course start date and display course start date and total number of students.

Command

```
SELECT COURSE_START_DATE, SUM(NO_OF_PARTICIPANTS) FROM COURSE_INFO
WHERE COURSE_TYPE ='CLR'
GROUP BY COURSE_START_DATE;
```

EXERCISE 20

Problem Statement:
Develop a query which would retrieve the total number of students enrolled for courses where course_type="CLR" grouped by course start date and display course start date and total number of students where the total number of students > 10.

Command

```
SELECT COURSE_START_DATE, SUM(NO_OF_PARTICIPANTS) FROM
COURSE_INFO
WHERE COURSE_TYPE = 'CLR'
GROUP BY COURSE_START_DATE
HAVING SUM(NO_OF_PARTICIPANTS >10);
```

EXERCISE 21

Problem Statement:
Develop a query which displays all the courses in increasing order of course duration.

Command

```
SELECT COURSE_NAME, COURSE_DURATION FROM COURSE_INFO
ORDER BY COURSE_DURATION ASC;
```

EXERCISE 22

Problem Statement:
Write a query to fetch student ID, first name, last name, and course code for students who have enrolled for course having course_code as 167. Student_Info and student_courses to be queried.

Command

```
INSERT INTO Student_Info(Student_Id, Course_code,First_Name,Last_Name,Address)
VALUES('ST01',1,'Shyam','Roy','123AD'),
('ST02',2,'Isabel','Alex','1578DF'),
('ST03',3,'Babu','Rosa','784TR'),
('ST04',4,'Roy','Betti','456GE'),
('ST05',6,'Abin','Paul','784YF'),
('ST06',167,'Bigil','Roy','528YGR'),
('ST07',167,'Pavan','Jose','123TR'),
('ST08',8,'Rey','Amaldev','754KU'),
('ST09',9,'Tony','Theo','963LIY'),
('ST10',10,'Ted','Frank','963PLI'),
('ST11',165,'Yugan','Sundar','741RTY'),
('ST12',8,'Jose','Sebastian','951WDV'),
('ST13',10,'Jai','Rahul','753MKJ'),
('ST14',9,'Tittu','Jose','578TGB'),
('ST15',166,'Ashly','Rebin','657ZDC');

SELECT Student_Id,First_Name,Last_Name,Student_info.Course_Code
FROM Student_Info
```

```
INNER JOIN Course_Info
ON Student_Info.Course_Code = Course_Info.Course_Code
WHERE Student_Info.Course_Code = 167;
```

EXERCISE 23

Problem Statement:
Write a query to display the discount offered on the courses along with course descriptions.

Command

```
SELECT * FROM COURSE_FEES;
INSERT INTO Course_Fees(Course_Code,Base_fees,Special_fees,Infra_Fees,Discount)
VALUES(165,560, 1000, 90.01, 5),
(166,220,1600,99.050,10),
(167,680,1250,97.86,5);

SELECT Course_Info.Course_Code,Discount,Course_Description
FROM Course_Info
INNER JOIN Course_Fees
ON Course_Fees.Course_Code = Course_Info.Course_Code;
```

EXERCISE 24

Problem Statement:
Write a query to fetch first names of the students along with the course codes of the courses they have enrolled in.

Command

```
SELECT CONCAT(First_Name,'_<',Course_Info.Course_Code,'>') AS Student_Record
FROM Student_Info
LEFT JOIN Course_Info
ON Student_Info.Course_Code = Course_Info.Course_Code;

SELECT CONCAT(First_Name,'_<',Course_Info.Course_Code,'>') AS Student_Record
FROM Student_Info
RIGHT JOIN Course_Info
ON Student_Info.Course_Code = Course_Info.Course_Code;
```

EXERCISE 25

Problem Statement:
Write a query to fetch student id for students who have enrolled for atleast one course whose fees is less than 1500

Command

```
SELECT *FROM Course_Info;

SELECT *FROM Student_Info;

INSERT INTO
Course_Info(Course_Code,Course_Name,Course_Description,Course_Start_Date,Course_Duration,N
o_of_Participants,Course_Type)
VALUES (168,'R1','Introduction_R','2022-11-01',3,5,'CLR'),
(169,'R2','Intermediate_R','2022-11-10',2,10,'EL');

INSERT INTO Student_Info(Student_Id, Course_code,First_Name,Last_Name,Address)
VALUES('ST02',2,'Isabel','Alex','1578DF'),
('ST07',167,'Pavan','Jose','123TR');

INSERT INTO Course_Fees(Course_Code,Base_fees,Special_fees,Infra_Fees,Discount)
VALUES(8,2000, 1580, 99.500, 5),
(9,220,160,99.050,10);

SELECT Student_Id
FROM Student_Info
WHERE Course_Code IN (SELECT Course_Code
FROM Course_Fees
WHERE ((Base_Fees+Special_Fees+Infra_Fees)*(1-Discount/100)) < 1500);
```

EXERCISE 26

Problem Statement:
Write a query to fetch student id and student name for students who have enrolled for atleast one course whose fees is less than 1500

Command

```
SELECT Student_Id, CONCAT(First_Name,' ',Last_Name) AS Student_Name
FROM Student_Info
WHERE Course_Code IN (SELECT Course_Code
FROM Course_Fees
WHERE ((Base_Fees+Special_Fees+Infra_Fees)*(1-Discount/100)) < 1500);
```