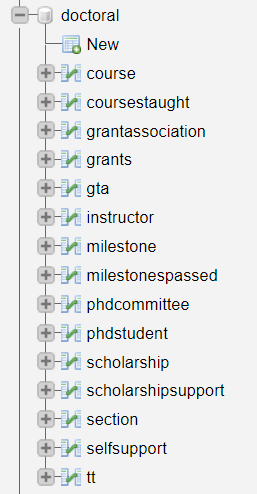
**Part 1**

**CREATE DATABASE statements:**

|  |
| --- |
| CREATE DATABASE doctoral; |

**CREATE TABLE statements:**

|  |
| --- |
| CREATE TABLE PHDSTUDENT(  StudentId VARCHAR(255) NOT NULL,  FName VARCHAR(255),  LName VARCHAR(255),  StSem VARCHAR(255),  StYear INT,  Supervisor VARCHAR(255),  PRIMARY KEY(StudentId)  ); |
| CREATE TABLE GRANTS(  GrantId VARCHAR(255),  Type VARCHAR(255),  GrantTitle VARCHAR(255),  Sourc VARCHAR(255),  StDate DATE,  EndDate DATE,  StAmount INT,  CurrBalance INT,  PRIMARY KEY(GrantId)  ); |
| CREATE TABLE SCHOLARSHIP(  ScholarshipID VARCHAR(255),  Type VARCHAR(255),  Source VARCHAR(255),  FundingCountry VARCHAR(255),  FundingOrganization VARCHAR(255),  PRIMARY KEY(ScholarshipID)  ); |
| CREATE TABLE INSTRUCTOR(  InstructorId VARCHAR(255),  FName VARCHAR(255),  LName VARCHAR(255),  StartDate DATE,  Degree VARCHAR(255),  Rank VARCHAR(255),  Type VARCHAR(255),  PRIMARY KEY(InstructorId)  ); |
| CREATE TABLE COURSE(  CourseID VARCHAR(255),  CName VARCHAR(255),  PRIMARY KEY(CourseID)  ); |
| CREATE TABLE SECTION(  SectionId VARCHAR(255),  CourseId VARCHAR(255),  InstructorId VARCHAR(255),  TA VARCHAR(255),  PRIMARY KEY(SectionId, CourseId),  FOREIGN KEY(CourseId) REFERENCES course(CourseID)  ); |
| CREATE TABLE MILESTONE(  MID VARCHAR(255),  MName VARCHAR(255),  PRIMARY KEY(MID)  ); |
| CREATE TABLE GRA(  StudentId VARCHAR(255),  GrantId VARCHAR(255),  MonthlyPay INT,  MajaorResearchArea VARCHAR(255),  PRIMARY KEY(StudentId),  FOREIGN KEY(StudentId) REFERENCES phdstudent(StudentId),  FOREIGN KEY(GrantId) REFERENCES grants(GrantId)  ); |
| CREATE TABLE GTA(  SectionID VARCHAR(255),  MonthlyPay INT,  Department VARCHAR(255),  StudentId VARCHAR(255),  PRIMARY KEY (StudentId),  FOREIGN KEY (SectionID) REFERENCES section(SectionId),  FOREIGN KEY (StudentId) REFERENCES phdstudent(StudentId)  ); |
| CREATE TABLE scholarshipsupport (  StudentId varchar(255),  ScholarshipID varchar(255),  MonthlyStipned float,  Category varchar(255),  PRIMARY KEY (StudentId),  FOREIGN KEY (ScholarshipID) REFERENCES scholarship (ScholarshipID),  FOREIGN KEY (StudentId) REFERENCES phdstudent (StudentId)  ); |
| CREATE TABLE selfsupport (  StudentId varchar(255),  TutionPaymentPlan varchar(255),  PRIMARY KEY (StudentId),  FOREIGN KEY (StudentId) REFERENCES phdstudent (StudentId)  ); |
| CREATE TABLE tt (  InstructorId varchar(255),  NoOfPhDStudents int(11),  PRIMARY KEY (InstructorId),  FOREIGN KEY (InstructorId) REFERENCES instructor (InstructorId)  ); |
| CREATE TABLE milestonespassed (  StudentId varchar(255) ,  MId varchar(255) ,  PassDate date ,  PRIMARY KEY (StudentId,MId),  FOREIGN KEY (StudentId) REFERENCES phdstudent (StudentId)  ); |
| CREATE TABLE phdcommittee (  StudentId varchar(255) ,  InstructorId varchar(255) ,  PRIMARY KEY (StudentId,InstructorId),  FOREIGN KEY (StudentId) REFERENCES phdstudent (StudentId),  FOREIGN KEY (InstructorId) REFERENCES instructor (InstructorId)  ); |
| CREATE TABLE coursestaught (  CourseID varchar(255) ,  InstructorId varchar(255) ,  PRIMARY KEY (CourseID,InstructorId),  FOREIGN KEY (CourseID) REFERENCES course (CourseID),  FOREIGN KEY (InstructorId) REFERENCES instructor (InstructorId)  ); |

****

**SELECT statements:**

|  |  |  |
| --- | --- | --- |
| **1** | SELECT \* FROM PHDSTUDENT;  A screenshot of a computer  Description automatically generated | |
| **2** | | SELECT \* FROM GRANTS;  A screenshot of a computer  Description automatically generated |
| **3** | | SELECT \* FROM SCHOLARSHIP;A screenshot of a computer  Description automatically generated |
| **4** | | SELECT \* FROM INSTRUCTOR;  A screenshot of a computer  Description automatically generated |
| **5** | | SELECT \* FROM COURSE;  A screenshot of a computer  Description automatically generated |
| **6** | | SELECT \* FROM SECTION;  A screenshot of a computer code  Description automatically generated |
| **7** | | SELECT \* FROM MILESTONE;  A screenshot of a computer  Description automatically generated |
| **8** | | SELECT \* FROM GRA;  A screenshot of a computer  Description automatically generated |
| **9** | | SELECT \* FROM GTA;  A screenshot of a computer  Description automatically generated |
| **10** | | SELECT \* FROM SCHOLARSHIPSUPPORT;  A screenshot of a computer  Description automatically generated |
| **11** | | SELECT \* FROM SELFSUPPORT;  A screenshot of a computer  Description automatically generated |
| **12** | | SELECT \* FROM TT;  A screenshot of a computer  Description automatically generated |
| **13** | | SELECT \* FROM MILESTONESPASSED;  A screenshot of a computer  Description automatically generated |
| **14** | | SELECT \* FROM PHDCOMMITTEE;  A screenshot of a computer  Description automatically generated |
| **15** | | SELECT \* FROM COURSESTAUGHT;  A screenshot of a computer  Description automatically generated |

**Part 2**

**SQL Statements:**

|  |  |
| --- | --- |
| 1 | Retrieve the full names of all instructors who have a PhD degree.  Query: SELECT CONCAT(FName,' ',LName) as FullName FROM instructor WHERE Degree='PhD';  A screenshot of a computer  Description automatically generated |
| 2 | **Find the names of all PhD students who are also working as Graduate Teaching Assistants (GTAs).**  **Query:** SELECT CONCAT(FName,' ',LName) AS Name FROM phdstudent WHERE StudentId IN (SELECT StudentId FROM gta);  A screenshot of a computer  Description automatically generated |
| 3 | **Retrieve a list of all courses taught by a specific instructor with InstructorID = BL9856.**  **Query:** SELECT CName FROM course WHERE CourseID IN (SELECT CourseID FROM coursestaught WHERE InstructorId='BL9856');  A screenshot of a computer  Description automatically generated |
| 4 | **Show the total number of PhD students supervised by each TT instructor. If a TT instructor has not supervised any students so far,**  **display a zero for them.**  **Query:** SELECT instructor.InstructorId, ifnull(tt.NoOfPhDStudents,0) as NoOfPhDStudents FROM ( instructor LEFT OUTER JOIN tt ON instructor.InstructorId = tt.InstructorId ); |
| 5 | **Retrieve a list of students who have passed a specific milestone (assume MId = “CM”).**  **Query:** SELECT \* FROM phdstudent WHERE phdstudent.StudentId in (select milestonespassed.StudentId from milestonespassed WHERE milestonespassed.MId='CM');  A screenshot of a computer  Description automatically generated |
| 6 | **Retrieve a list of details such as GrantId, GrantTitle, Source, StDate, EndDate, StAmount, and CurrBalance for all grants that started in the year 2018.**  **Query: select \* from GRANTS where YEAR(grants.StDate)=2018;**  A screenshot of a computer  Description automatically generated |
| 7 | **Retrieve a list of all PhD students who are self-supporting along with the details of their tuition payment plans.**  **Query:** select \* from phdstudent WHERE phdstudent.StudentId in ( select selfsupport.StudentId from selfsupport);  A screenshot of a computer  Description automatically generated |
| 8 | **Find the total monthly stipend received by each PhD student from scholarships.**  **SQL:** SELECT StudentId, MonthlyStipned FROM scholarshipsupport;  A screenshot of a computer  Description automatically generated |
| 9 | **Identify all instructors who are not supervising any PhD students.**  **SQL:** select Instructor.InstructorId from instructor where instructor.InstructorId not in (SELECT DISTINCT(phdcommittee.InstructorId) from phdcommittee); |
| 10 | **Retrieve a list of all courses that have never been taught (i.e., not listed in the COURSESTAUGHT table).**  **SQL:** select course.CourseID,course.CName from course where course.CourseID not in (select DISTINCT(coursestaught.CourseID) from coursestaught); |
| 11 | **Retrieve a list of the average monthly pay for Graduate Research Assistants in each major research area.**  **SQL:** select avg(gra.MonthlyPay) as AvgMonthlyPay,gra.MajaorResearchArea from gra group by gra.MajaorResearchArea; |
| 12 | **Retrieve a list of full names and the number of sections each instructor is teaching.**  **SQL:** select count(section.SectionId) as TotalSection,instructor.FName,instructor.LName from section JOIN instructor on section.InstructorId=instructor.InstructorId GROUP BY section.InstructorId;  A screenshot of a computer  Description automatically generated |
| 13 | **Retrieve a list of all students who have received a scholarship from ' National Science Foundation'.**  **SQL:** select \* from phdstudent where phdstudent.StudentId in (select scholarshipsupport.StudentId from scholarshipsupport where scholarshipsupport.ScholarshipID in (SELECT scholarship.ScholarshipID from scholarship where scholarship.FundingOrganization like '%National Science Foundation%')); |
| 14 | **Execute a command to delete a record that violates a referential integrity constraint. State the message produced by the DBMS.**  **SQL:** DELETE FROM phdstudent WHERE phdstudent.StudentId='AA1234';  **Message**: Cannot delete or update a parent row: a foreign key constraint fails (`doctoral`.`milestonespassed`, CONSTRAINT `milestonespassed\_ibfk\_1` FOREIGN KEY (`StudentId`) REFERENCES `phdstudent` (`StudentId`)) A screenshot of a computer  Description automatically generated |
| 15 | **Execute an update command for Instructor table that attempts to update a record and thereby violates the foreign key constraint.State the message produced by the DBMS.**  **SQL:** UPDATE instructor SET instructor.InstructorId='AA1111' WHERE instructor.InstructorId='AO5671'; **Message:** Cannot delete or update a parent row: a foreign key constraint fails (`doctoral`.`coursestaught`, CONSTRAINT `coursestaught\_ibfk\_2` FOREIGN KEY (`InstructorId`) REFERENCES `instructor` (`InstructorId`))  A screenshot of a computer  Description automatically generated |
| 16 | **Execute 3 insert commands for Instructor or TT tables that attempt to insert records, such that the records violate the explicit schema-based constraints (Key, Entity Integrity, Referential Integrity constraints). Make each of the 3 records violate a different types of integrity constraint. Include the insert statements and the error messages produced. a) Violates Key constraint. b) Violates Entity Integrity constraint. c) Violates Referential Integrity constraint.**  **SQL: (a) Violates Key Constraint**  INSERT INTO phdstudent(StudentId, FName, LName, StSem, StYear, Supervisor) VALUES ('AA2345','Sakshi','Patel','Fall','2017','AO5671'); **Message:** Duplicate entry 'AA2345' for key 'PRIMARY' A screenshot of a pink box  Description automatically generated  **(b) Violates Entity Integrity Constraint** INSERT INTO phdstudent(StudentId, FName, LName, StSem, StYear, Supervisor) VALUES (null,'Sakshi','Patel','Fall','2017','AO5671');  **Message:** Column 'StudentId' cannot be null    **(c)Violates Referential Integrity Constraint**  INSERT INTO scholarshipsupport(StudentId, ScholarshipID, MonthlyStipned, Category) VALUES ('AA1234','SCH1011','1200.00','local');  **Message:** Cannot add or update a child row: a foreign key constraint fails (`doctoral`.`scholarshipsupport`, CONSTRAINT `scholarshipsupport\_ibfk\_1` FOREIGN KEY (`ScholarshipID`) REFERENCES `scholarship` (`ScholarshipID`))  A screenshot of a computer error  Description automatically generated |