*ANVESHA TRIPATHI*

*ROLL NO.-20221209.*

*BSC (H)STATISTICS.*

*CS ASSIGNMENT.*

*Q1) Create and use the following student-course database schema for a college to answer the given queries using the standalone SQL editor.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *STUDENT* | *ROLL NO.* | *STUDENT NAME* | *COURSE ID* | *DOB* |
|  | *Char(6)* | *Varchar(20)* | *Varchar(20)* | *Date* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *COURSE* | *CID* | *COURSE NAME* | *COURSE TYPE* | *TEACHER-IN -CHARGE* | *TOTAL SEATS* | *DURATION* |
|  | *Char(6)* | *Varchar(20)* | *Char (8)* | *Varchar (15)* | *Unsigned int* | *Unsigned int* |

|  |  |  |  |
| --- | --- | --- | --- |
| *ADMISSION* | *ROLL NO.* | *CID* | *DATE OF ADMISSION* |
|  | *Char(6)* | *Char(6)* | *Date* |

*Here, Roll No (ADMISSION) and CID (ADMISSION) are foreign keys. Note that course type may have two values viz. Fulltime and Part time and a student may enroll in any number of courses.*

*1. Retrieve names of students enrolled in any course.*

*2. Retrieve names of students enrolled in at least one part time course.*

*3. Retrieve students' names starting with letter ‘A’.*

*4. Retrieve students' details studying in courses ‘computer science’ or ‘chemistry’.*

*5. Retrieve students’ names whose roll no either starts with ‘X’ or ‘Z’ and ends with ‘9’.*

*6. Find course details with more than N students enrolled where N is to be input by the user.*

*7. Update student table for modifying a student name.*

*8. Find course names in which more than five students have enrolled.*

*9. Find the name of youngest student enrolled in course ‘BSC (P) CS’.*

*10. Find the name of most popular society (on the basis of enrolled students).*

*11. Find the name of two popular part time courses (on the basis of enrolled students).*

*12. Find the student names who are admitted to full time courses only.*

*13. Find course names in which more than 30 students took admission.*

*14. Find names of all students who took admission to any course and course names in which at least one student has enrolled.*

*15. Find course names where teacher-in-charge has ‘Gupta’ as surname and the course is full time.*

*16. Find the course names in which the number of enrolled students is only 10% of its total seats.*

*17. Display the vacant seats for each course.*

*18. Increment Total Seats of each course by 20%.*

*19. Add enrolment fees paid (‘yes’/’No’) field in the enrolment table.*

*20. Update the date of admission for all the courses by 1 year.*

*21. Create a view to keep track of course names with total number of students enrolled in it.*

*22. Count the number of courses with more than 5 students enrolled for each type of course.*

*23. Add column Mobile number in student table with default value ‘9999999999’*

*24. Find the total number of students whose age is > 18 years.*

*25. Find names of students who are born in 2001 and are admitted to at least one part time course.*

*26. Count all courses having ‘science’ in the name and starting with the word ‘B.Sc.’*

*1. To retrieve names of students enrolled in any course.*

*SELECT StudentName*

*FROM STUDENT;*

*2. Retrieve names of students enrolled in at least one part-time course.*

*SELECT DISTINCT s.StudentName*

*FROM STUDENT s*

*JOIN ADMISSION a ON s.RollNo = a.RollNo*

*JOIN COURSE c ON a.CID = c.CID*

*WHERE c.CourseType = 'Parttime';*

*3. Retrieve students' names starting with letter ‘A’.*

*SELECT StudentName*

*FROM STUDENT*

*WHERE StudentName LIKE 'A%';*

*4. Retrieve students' details studying in courses ‘computer science’ or ‘chemistry’.*

*SELECT \**

*FROM STUDENT s*

*JOIN ADMISSION a ON s.RollNo = a.RollNo*

*JOIN COURSE c ON a.CID = c.CID*

*WHERE c.CourseName IN ('Computer Science', 'Chemistry');*

*5. Retrieve students’ names whose roll no either starts with ‘X’ or ‘Z’ and ends with ‘9’.*

*SELECT StudentName*

*FROM STUDENT*

*WHERE (RollNo LIKE 'X%' OR RollNo LIKE 'Z%') AND RollNo LIKE '%9';*

*6. Find course details with more than N students enrolled where N is to be input by the user.*

*Assuming N is a parameter, replace it with the actual value when executing the query.*

*SELECT c.\*, COUNT(a.RollNo) AS EnrolledStudents*

*FROM COURSE c*

*JOIN ADMISSION a ON c.CID = a.CID*

*GROUP BY c.CID*

*HAVING EnrolledStudents > N;*

*7. Update student table for modifying a student name.*

*Assuming you want to update the name of a student with RollNo 'XYZ123'.*

*UPDATE STUDENT*

*SET StudentName = 'NewName'*

*WHERE RollNo = 'XYZ123';*

*8. Find course names in which more than five students have enrolled.*

*SELECT c.CourseName*

*FROM COURSE c*

*JOIN ADMISSION a ON c.CID = a.CID*

*GROUP BY c.CID*

*HAVING COUNT(a.RollNo) > 5;*

*9. Find the name of the youngest student enrolled in the course ‘BSc(P)CS’.*

*SELECT TOP 1 s.StudentName*

*FROM STUDENT s*

*JOIN ADMISSION a ON s.RollNo = a.RollNo*

*WHERE a.CID = 'BSc(P)CS'*

*ORDER BY s.DOB DESC;*

*10. Find the name of the most popular society (on the basis of enrolled students).*

*SELECT TOP 1 c.CourseName AS MostPopularSociety*

*FROM COURSE c*

*JOIN ADMISSION a ON c.CID = a.CID*

*GROUP BY c.CID, c.CourseName*

*ORDER BY COUNT(a.RollNo) DESC;*

*11. Find the name of two popular part-time courses (on the basis of enrolled students)*

*SELECT TOP 2 c.CourseName AS PopularPartTimeCourse*

*FROM COURSE c*

*JOIN ADMISSION a ON c.CID = a.CID*

*WHERE c.CourseType = 'Parttime'*

*GROUP BY c.CID, c.CourseName*

*ORDER BY COUNT(a.RollNo) DESC;*

*12. Find the student names who are admitted to full-time courses only.*

*SELECT s.StudentName*

*FROM STUDENT s*

*JOIN ADMISSION a ON s.RollNo = a.RollNo*

*JOIN COURSE c ON a.CID = c.CID*

*WHERE c.CourseType = 'Fulltime'*

*GROUP BY s.RollNo, s.StudentName*

*HAVING COUNT(DISTINCT c.CID) = 1;*

*13. Find course names in which more than 30 students took admission.*

*SELECT c.CourseName*

*FROM COURSE c*

*JOIN ADMISSION a ON c.CID = a.CID*

*GROUP BY c.CID, c.CourseName*

*HAVING COUNT(a.RollNo) > 30;*

*14. Find names of all students who took admission to any course and course names in which at least one student has enrolled.*

*SELECT DISTINCT s.StudentName, c.CourseName*

*FROM STUDENT s*

*JOIN ADMISSION a ON s.RollNo = a.RollNo*

*JOIN COURSE c ON a.CID = c.CID;*

*15. Find course names where the teacher-in-charge has ‘Gupta’ as a surname and the course is full time.*

*SELECT c.CourseName*

*FROM COURSE c*

*WHERE c.CourseType = 'Fulltime' AND c.TeacherInCharge LIKE '% Gupta%';*

*16. Find the course names in which the number of enrolled students is only 10% of its total seats.*

*SELECT c.CourseName*

*FROM COURSE c*

*JOIN ADMISSION a ON c.CID = a.CID*

*GROUP BY c.CID, c.CourseName, c.TotalSeats*

*HAVING COUNT(a.RollNo) <= 0.1 \* c.TotalSeats;*

*17. Display the vacant seats for each course.*

*SELECT c.CourseName, (c.TotalSeats - COUNT(a.RollNo)) AS VacantSeats*

*FROM COURSE c*

*LEFT JOIN ADMISSION a ON c.CID = a.CID*

*GROUP BY c.CID, c.CourseName, c.TotalSeats;*

*18. Increment Total Seats of each course by 20%.*

*UPDATE COURSE*

*SET TotalSeats = TotalSeats \* 1.2;*

*19. Add enrolment fees paid ('yes'/'no') field in the enrolment table.*

*ALTER TABLE ADMISSION*

*ADD COLUMN EnrolmentFeesPaid VARCHAR(3);*

*20. Update the date of admission for all the courses by 1 year.*

*UPDATE ADMISSION*

*SET DateOfAdmission = DATEADD(YEAR, 1, DateOfAdmission);*

*21. Create a view to keep track of course names with the total number of students enrolled in it.*

*CREATE VIEW CourseEnrollmentView AS*

*SELECT c.CourseName, COUNT(a.RollNo) AS TotalStudentsEnrolled*

*FROM COURSE c*

*LEFT JOIN ADMISSION a ON c.CID = a.CID*

*GROUP BY c.CID, c.CourseName;*

*22. Count the number of courses with more than 5 students enrolled for each type of course.*

*SELECT CourseType, COUNT(CASE WHEN EnrolledStudents > 5 THEN 1 END) AS CoursesWithMoreThan5Students*

*FROM (*

*SELECT c.CourseType, COUNT(a.RollNo) AS EnrolledStudents*

*FROM COURSE c*

*LEFT JOIN ADMISSION a ON c.CID = a.CID*

*GROUP BY c.CID, c.CourseType*

*) AS CourseEnrollment*

*GROUP BY CourseType;*

*23. Add a column Mobile number in the student table with a default value ‘9999999999’.*

*ALTER TABLE STUDENT*

*ADD COLUMN MobileNumber VARCHAR(10) DEFAULT '9999999999';*

*24. Find the total number of students whose age is > 18 years.*

*SELECT COUNT(\*)*

*FROM STUDENT*

*WHERE DATEDIFF(YEAR, DOB, GETDATE()) > 18;*

*25. Find names of students who are born in 2001 and are admitted to at least one part-time course.*

*SELECT DISTINCT s.StudentName*

*FROM STUDENT s*

*JOIN ADMISSION a ON s.RollNo = a.RollNo*

*JOIN COURSE c ON a.CID = c.CID*

*WHERE YEAR(s.DOB) = 2001 AND c.CourseType = 'Parttime';*

*26. Count all courses having ‘science’ in the name and starting with the word ‘B.Sc.’.*

*SELECT COUNT(\*)*

*FROM COURSE*

*WHERE CourseName LIKE 'B.Sc.%' AND CourseName LIKE '%science%';*

*Q2) Create the following tables with appropriate data type for attributes and integrity constraints on the tables. Enter at least 5 records in each table and answer the queries given below.*

*Suppliers (SNo, Sname, Status, SCity)*

*Parts (PNo, Pname, Colour, Weight, City)*

*Project (JNo, Jname, Jcity)*

*Shipment (Sno, Pno, Jno, Quantity)*

*1. Find supplier numbers for suppliers in Mandi with status less than 20.*

*2. Find supplier details for suppliers who supply part P2. Display the supplier list in decreasing order of supplier numbers.*

*3. Find suppliers names for suppliers who do not supply part P2.*

*4. For each shipment get full shipment details, including total shipment weights computed as Weight\*Quantity of corresponding parts.*

*5. Get all the shipments where the quantity is in the range 300 to 750 inclusive.*

*6. Get part numbers for parts that either weigh more than 1Kg or are supplied by suppliers S2 or both.*

*7. Get the names of cities that store more than two red parts. Change the column name in the output to “City-Parts”.*

*8. Update the city of supplier S1 to “Delhi”.*

*9. Get part numbers for parts supplied by a supplier in Allahabad to a project in Chennai.*

*10. Find the names of all parts whose color starts with the letter b.*

*11. Change the datatype of the weight attribute in the Parts table from int to float.*

*12. Find the number of parts of each color.*

*13. Find the names of all the projects which are located in the city Jaipur and in which the part is supplied by supplier S3.*

*14. Delete all the projects which are located in Madras.*

*15. Find all part-details of parts that are shipped to any project carried out in Mumbai.*

*16. Find number of unique projects supplied by supplier S1.*

*17. Add column SDate in shipment table.*

*18. For each supplier which supplies parts to a project, find the total no. of parts supplied by the supplier.*

*19. Find all supplier name, part name and project name triples such that the indicated supplier, part, and project are all located in the same city. List the name of the city along with the names of suppliers, project and parts.*

*20. Get the names of cities from where more than three yellow parts are supplied.*

*21. Find all distinct cities where either supplier is living or parts are shipped from or projects are carried out. Change the column name in the output to “All-Cities”.*

*22. Find names of cities such that atleast one supplier is living there and atleast one part is shipped and one project is carried out. Change the column name in the output to “Common-Cities”.*

*23. Modify data type of any attribute of table shipment.*

*-- Table creation with appropriate data types and constraints*

*CREATE TABLE Suppliers (*

*SNo INT PRIMARY KEY,*

*Sname VARCHAR(50),*

*Status INT,*

*SCity VARCHAR(50)*

*);*

*CREATE TABLE Parts (*

*PNo INT PRIMARY KEY,*

*Pname VARCHAR(50),*

*Colour VARCHAR(20),*

*Weight DECIMAL(10, 3),*

*City VARCHAR(50)*

*);*

*CREATE TABLE Project (*

*JNo INT PRIMARY KEY,*

*Jname VARCHAR(50),*

*JCity VARCHAR(50)*

*);*

*CREATE TABLE Shipment (*

*Sno INT,*

*Pno INT,*

*Jno INT,*

*Quantity INT,*

*PRIMARY KEY (Sno, Pno, Jno),*

*FOREIGN KEY (Sno) REFERENCES Suppliers(SNo),*

*FOREIGN KEY (Pno) REFERENCES Parts(PNo),*

*FOREIGN KEY (Jno) REFERENCES Project(JNo)*

*);*

*-- Inserting sample data into the tables*

*-- Suppliers*

*INSERT INTO Suppliers VALUES (1, 'Supplier1', 15, 'Mandi');*

*INSERT INTO Suppliers VALUES (2, 'Supplier2', 25, 'Allahabad');*

*INSERT INTO Suppliers VALUES (3, 'Supplier3', 10, 'Delhi');*

*INSERT INTO Suppliers VALUES (4, 'Supplier4', 18, 'Mandi');*

*INSERT INTO Suppliers VALUES (5, 'Supplier5', 30, 'Chennai');*

*-- Parts*

*INSERT INTO Parts VALUES (1, 'Part1', 'Red', 0.5, 'Delhi');*

*INSERT INTO Parts VALUES (2, 'Part2', 'Blue', 1.2, 'Mandi');*

*INSERT INTO Parts VALUES (3, 'Part3', 'Green', 0.8, 'Chennai');*

*INSERT INTO Parts VALUES (4, 'Part4', 'Red', 1.5, 'Mandi');*

*INSERT INTO Parts VALUES (5, 'Part5', 'Blue', 0.7, 'Allahabad');*

*-- Project*

*INSERT INTO Project VALUES (1, 'Project1', 'Chennai');*

*INSERT INTO Project VALUES (2, 'Project2', 'Mandi');*

*INSERT INTO Project VALUES (3, 'Project3', 'Delhi');*

*INSERT INTO Project VALUES (4, 'Project4', 'Allahabad');*

*INSERT INTO Project VALUES (5, 'Project5', 'Chennai');*

*-- Shipment*

*INSERT INTO Shipment VALUES (1, 1, 1, 200);*

*INSERT INTO Shipment VALUES (2, 2, 2, 300);*

*INSERT INTO Shipment VALUES (3, 3, 3, 400);*

*INSERT INTO Shipment VALUES (4, 4, 4, 500);*

*INSERT INTO Shipment VALUES (5, 5, 5, 600);*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *SUPPLIERS* | *SNo* | *Sname* | *Status* | *SCity* |
|  | *1* | *Supplier 1* | *15* | *Mandi* |
|  | *2* | *Supplier 2* | *25* | *Allahabad* |
|  | *3* | *Supplier 3* | *10* | *Delhi* |
|  | *4* | *Supplier 4* | *18* | *Mandi* |
|  | *5* | *Supplier 5* | *30* | *Chennai* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *PARTS* | *PNo* | *Pname* | *Colour* | *Weight* | *City* |
|  | *1* | *Part 1* | *Red* | *0.5* | *Delhi* |
|  | *2* | *Part 2* | *Blue* | *1.2* | *Mandi* |
|  | *3* | *Part 3* | *Green* | *0.8* | *Chennai* |
|  | *4* | *Part 4* | *Red* | *1.5* | *Mandi* |
|  | *5* | *Part 5* | *Blue* | *0.7* | *Allahabad* |

|  |  |  |  |
| --- | --- | --- | --- |
| *PROJECT* | *JNo* | *Jname* | *Jcity* |
|  | *1* | *Project 1* | *Chennai* |
|  | *2* | *Project 2* | *Mandi* |
|  | *3* | *Project 3* | *Delhi* |
|  | *4* | *Project 4* | *Allahabad* |
|  | *5* | *Project 5* | *Chennai* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *SHIPMENT* | *Sno* | *Pno* | *Jno* | *Quantity* |
|  | *1* | *1* | *1* | *200* |
|  | *2* | *2* | *2* | *300* |
|  | *3* | *3* | *3* | *400* |
|  | *4* | *4* | *4* | *500* |
|  | *5* | *5* | *5* | *600* |

*1. Find supplier numbers for suppliers in Mandi with status less than 20.*

*SELECT SNo*

*FROM Suppliers*

*WHERE SCity = 'Mandi' AND Status < 20;*

*2. Find supplier details for suppliers who supply part P2. Display the supplier list in decreasing order of supplier numbers.*

*SELECT S.\**

*FROM Suppliers S*

*JOIN Shipment SH ON S.SNo = SH.Sno*

*WHERE SH.Pno = 2*

*ORDER BY S.SNo DESC;*

*3. Find suppliers' names for suppliers who do not supply part P2.*

*SELECT Sname*

*FROM Suppliers*

*WHERE SNo NOT IN (SELECT Sno FROM Shipment WHERE Pno = 2);*

*4. For each shipment get full shipment details, including total shipment weights computed as Weight\*Quantity of corresponding parts.*

*SELECT SH.\*, P.Weight \* SH.Quantity AS TotalWeight*

*FROM Shipment SH*

*JOIN Parts P ON SH.Pno = P.PNo;*

*5. Get all the shipments where the quantity is in the range 300 to 750 inclusive.*

*SELECT \**

*FROM Shipment*

*WHERE Quantity BETWEEN 300 AND 750;*

*6. Get part numbers for parts that either weigh more than 1Kg or are supplied by suppliers S2 or both.*

*SELECT DISTINCT Pno*

*FROM Parts*

*WHERE Weight > 1 OR Pno IN (SELECT Pno FROM Shipment WHERE Sno = 2);*

*7. Get the names of cities that store more than two red parts. Change the column name in the output to “City-Parts”.*

*SELECT City AS "City-Parts"*

*FROM Parts*

*WHERE Colour = 'Red'*

*GROUP BY City*

*HAVING COUNT(\*) > 2;*

*8. Update the city of supplier S1 to “Delhi”.*

*UPDATE Suppliers*

*SET SCity = 'Delhi'*

*WHERE SNo = 1;*

*9. Get part numbers for parts supplied by a supplier in Allahabad to a project in Chennai.*

*SELECT DISTINCT Pno*

*FROM Shipment*

*WHERE Sno IN (SELECT SNo FROM Suppliers WHERE SCity = 'Allahabad')*

*AND Jno IN (SELECT JNo FROM Project WHERE JCity = 'Chennai');*

*10. Find the names of all parts whose color starts with the letter b.*

*SELECT Pname*

*FROM Parts*

*WHERE Colour LIKE 'B%';*

*11. Change the datatype of the weight attribute in the Parts table from int to float.*

*ALTER TABLE Parts*

*ALTER COLUMN Weight FLOAT;*

*12. Find the number of parts of each color.*

*SELECT Colour, COUNT(\*) AS NumberOfParts*

*FROM Parts*

*GROUP BY Colour;*

*13. Find the names of all the projects which are located in the city Jaipur and in which the part is supplied by supplier S3.*

*SELECT Jname*

*FROM Project*

*WHERE JCity = 'Jaipur'*

*AND Jno IN (SELECT Jno FROM Shipment WHERE Sno = 3);*

*14. Delete all the projects which are located in Madras.*

*DELETE FROM Project*

*WHERE JCity = 'Madras';*

*15. Find all part-details of parts that are shipped to any project carried out in Mumbai.*

*SELECT P.\**

*FROM Parts P*

*JOIN Shipment SH ON P.PNo = SH.Pno*

*JOIN Project J ON SH.Jno = J.Jno*

*WHERE J.JCity = 'Mumbai';*

*16. Find the number of unique projects supplied by supplier S1.*

*SELECT COUNT(DISTINCT Jno) AS UniqueProjectsSupplied*

*FROM Shipment*

*WHERE Sno = 1;*

*17. Add column SDate in shipment table.*

*ALTER TABLE Shipment*

*ADD COLUMN SDate DATE;*

*18. For each supplier which supplies parts to a project, find the total no. of parts supplied by the supplier.*

*SELECT Sno, Jno, COUNT(\*) AS TotalPartsSupplied*

*FROM Shipment*

*GROUP BY Sno, Jno;*

*19. Find all supplier name, part name and project name triples such that the indicated supplier, part, and project are all located in the same city. List the name of the city along with the names of suppliers, project and parts.*

*SELECT S.Sname, P.Pname, J.Jname, S.SCity AS "City"*

*FROM Shipment SH*

*JOIN Suppliers S ON SH.Sno = S.SNo*

*JOIN Parts P ON SH.Pno = P.PNo*

*JOIN Project J ON SH.Jno = J.JNo*

*WHERE S.SCity = P.City AND P.City = J.JCity;*

*20. Get the names of cities from where more than three yellow parts are supplied.*

*SELECT P.City AS "All-Cities"*

*FROM Parts P*

*WHERE P.Colour = 'Yellow'*

*GROUP BY P.City*

*HAVING COUNT(\*) > 3;*

*21. Find all distinct cities where either supplier is living or parts are shipped from or projects are carried out. Change the column name in the output to “All-Cities”.*

*SELECT DISTINCT SCity AS "All-Cities" FROM Suppliers*

*UNION*

*SELECT DISTINCT City FROM Parts*

*UNION*

*SELECT DISTINCT JCity FROM Project;*

*22. Find names of cities such that at least one supplier is living there and at least one part is shipped and one project is carried out. Change the column name in the output to “Common-Cities”.*

*SELECT SCity AS "Common-Cities" FROM Suppliers*

*INTERSECT*

*SELECT City FROM Parts*

*INTERSECT*

*SELECT JCity FROM Project;*

*23. Modify the data type of any attribute of the table shipment.*

*-- Assuming modifying the Quantity attribute to be a FLOAT.*

*ALTER TABLE Shipment*

*ALTER COLUMN Quantity FLOAT;*