-- Database Schema

CREATE TABLE STUDENT (

RollNo Char(6) PRIMARY KEY,

StudentName Varchar(20),

Course Varchar(10),

DOB Date

);

CREATE TABLE SOCIETY (

SocID Char(6) PRIMARY KEY,

SocName Varchar(20),

MentorName Varchar(15),

Date Date,

TotalSeats INT UNSIGNED

);

CREATE TABLE ENROLLMENT (

RollNo Char(6),

SID Char(6),

DateOfEnrollment Date,

FOREIGN KEY (RollNo) REFERENCES STUDENT(RollNo),

FOREIGN KEY (SID) REFERENCES SOCIETY(SocID)

);

-- Sample Data

INSERT INTO STUDENT (RollNo, StudentName, Course, DOB) VALUES

('S001', 'Alice', 'Computer Science', '2000-05-12'),

('S002', 'Bob', 'Chemistry', '2001-08-21'),

('S003', 'Charlie', 'Physics', '1999-11-15'),

('S004', 'David', 'Computer Science', '2000-03-09');

INSERT INTO SOCIETY (SocID, SocName, MentorName, Date, TotalSeats) VALUES

('SOC001', 'Debating', 'John Doe', '2023-01-01', 50),

('SOC002', 'Dancing', 'Jane Smith', '2023-01-05', 30),

('SOC003', 'NSS', 'Michael Gupta', '2023-02-10', 40);

INSERT INTO ENROLLMENT (RollNo, SID, DateOfEnrollment) VALUES

('S001', 'SOC001', '2023-01-02'),

('S001', 'SOC002', '2023-01-03'),

('S002', 'SOC003', '2023-01-04'),

('S003', 'SOC001', '2023-01-05'),

('S004', 'SOC002', '2023-01-06'),

('S004', 'SOC003', '2023-01-07');

-- Queries

-- 1. Retrieve names of students enrolled in any society

SELECT DISTINCT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo;

-- 2. Retrieve all society names

SELECT SocName FROM SOCIETY;

-- 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 WHERE Course 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 society details with more than N TotalSeats where N is to be input by the user

-- Assuming N is 40

SELECT \* FROM SOCIETY WHERE TotalSeats > 40;

-- 7. Update society table for mentor name of a specific society

UPDATE SOCIETY SET MentorName = 'New Mentor' WHERE SocID = 'SOC001';

-- 8. Find society names in which more than five students have enrolled

SELECT s.SocName

FROM SOCIETY s

JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName

HAVING COUNT(\*) > 5;

-- 9. Find the name of youngest student enrolled in society ‘NSS’

SELECT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

WHERE e.SID = 'SOC003'

ORDER BY s.DOB ASC

LIMIT 1;

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

SELECT s.SocName

FROM SOCIETY s

JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName

ORDER BY COUNT(\*) DESC

LIMIT 1;

-- 11. Find the name of two least popular societies (on the basis of enrolled students)

SELECT s.SocName

FROM SOCIETY s

JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName

ORDER BY COUNT(\*) ASC

LIMIT 2;

-- 12. Find the student names who are not enrolled in any society

SELECT StudentName FROM STUDENT WHERE RollNo NOT IN (SELECT RollNo FROM ENROLLMENT);

-- 13. Find the student names enrolled in at least two societies

SELECT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

GROUP BY s.StudentName

HAVING COUNT(\*) >= 2;

-- 14. Find society names in which maximum students are enrolled

SELECT s.SocName

FROM SOCIETY s

JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName

ORDER BY COUNT(\*) DESC

LIMIT 1;

-- 15. Find names of all students who have enrolled in any society and society names in which at least one student has enrolled

SELECT DISTINCT s.StudentName, soc.SocName

FROM STUDENT s

LEFT JOIN ENROLLMENT e ON s.RollNo = e.RollNo

LEFT JOIN SOCIETY soc ON e.SID = soc.SocID;

-- 16. Find names of students who are enrolled in any of the three societies ‘Debating’, ‘Dancing’ and ‘Sashakt’.

SELECT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

JOIN SOCIETY soc ON e.SID = soc.SocID

WHERE soc.SocName IN ('Debating', 'Dancing', 'Sashakt');

-- 17. Find society names such that its mentor has a name with ‘Gupta’ in it.

SELECT SocName

FROM SOCIETY

WHERE MentorName LIKE '%Gupta%';

-- 18. Find the society names in which the number of enrolled students is only 10% of its capacity.

SELECT s.SocName

FROM SOCIETY s

JOIN (

SELECT SID, COUNT(\*) AS NumEnrolled

FROM ENROLLMENT

GROUP BY SID

) e ON s.SocID = e.SID

WHERE e.NumEnrolled <= 0.1 \* s.TotalSeats;

-- 19. Display the vacant seats for each society.

SELECT s.SocName, s.TotalSeats - COUNT(e.RollNo) AS VacantSeats

FROM SOCIETY s

LEFT JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName;

-- 20. Increment Total Seats of each society by 10%

UPDATE SOCIETY SET TotalSeats = TotalSeats \* 1.1;

-- 21. Add the enrollment fees paid (‘yes’/’No’) field in the enrollment table.

ALTER TABLE ENROLLMENT ADD EnrollmentFeesPaid ENUM('Yes', 'No');

-- 22. Update date of enrollment of society id ‘s1’ to ‘2018-01-15’, ‘s2’ to current date and ‘s3’ to ‘2018-01-02’.

UPDATE ENROLLMENT

SET DateOfEnrollment = CASE SID

WHEN 's1' THEN '2018-01-15'

WHEN 's2' THEN CURRENT\_DATE

WHEN 's3' THEN '2018-01-02'

ELSE DateOfEnrollment

END;

-- 23. Create a view to keep track of society names with the total number of students enrolled in it.

CREATE VIEW Society\_Enrollment\_Count AS

SELECT s.SocName, COUNT(e.RollNo) AS TotalEnrolled

FROM SOCIETY s

LEFT JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName;

-- 24. Find student names enrolled in all the societies.

SELECT s.StudentName

FROM STUDENT s

WHERE NOT EXISTS (

SELECT SocID FROM SOCIETY

EXCEPT

SELECT SID FROM ENROLLMENT WHERE RollNo = s.RollNo

);

-- 25. Count the number of societies with more than 5 students enrolled in it

SELECT COUNT(\*) AS NumSocieties

FROM (

SELECT SID, COUNT(\*) AS NumEnrolled

FROM ENROLLMENT

GROUP BY SID

HAVING NumEnrolled > 5

) AS PopularSocieties;

-- 26. Add column Mobile number in student table with default value ‘9999999999’

ALTER TABLE STUDENT ADD MobileNumber VARCHAR(15) DEFAULT '9999999999';

-- 27. Find the total number of students whose age is > 20 years.

SELECT COUNT(\*) AS NumStudents

FROM STUDENT

WHERE TIMESTAMPDIFF(YEAR, DOB, CURDATE()) > 20;

-- 28. Find names of students who are born in 2001 and are enrolled in at least one society.

SELECT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

WHERE YEAR(s.DOB) = 2001;

-- 29. Count all societies whose name starts with ‘S’ and ends with ‘t’ and at least 5 students are enrolled in the society.

SELECT COUNT(\*) AS NumSocieties

FROM SOCIETY s

JOIN (

SELECT SID, COUNT(\*) AS NumEnrolled

FROM ENROLLMENT

GROUP BY SID

HAVING NumEnrolled >= 5

) e ON s.SocID = e.SID

WHERE SocName LIKE 'S%t';

-- 30. Display the following information:

-- Society name, Mentor name, Total Capacity, Total Enrolled, Unfilled Seats

SELECT s.SocName, s.MentorName, s.TotalSeats, COUNT(e.RollNo) AS TotalEnrolled, (s.TotalSeats - COUNT(e.RollNo)) AS UnfilledSeats

FROM SOCIETY s

LEFT JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocID;

Sure, here's the SQL script for the given queries along with the database administration commands:

```sql

-- Database Schema

CREATE TABLE STUDENT (

RollNo Char(6) PRIMARY KEY,

StudentName Varchar(20),

Course Varchar(10),

DOB Date

);

CREATE TABLE SOCIETY (

SocID Char(6) PRIMARY KEY,

SocName Varchar(20),

MentorName Varchar(15),

Date Date,

TotalSeats INT UNSIGNED

);

CREATE TABLE ENROLLMENT (

RollNo Char(6),

SID Char(6),

DateOfEnrollment Date,

FOREIGN KEY (RollNo) REFERENCES STUDENT(RollNo),

FOREIGN KEY (SID) REFERENCES SOCIETY(SocID)

);

-- Sample Data

INSERT INTO STUDENT (RollNo, StudentName, Course, DOB) VALUES

('S001', 'Alice', 'Computer Science', '2000-05-12'),

('S002', 'Bob', 'Chemistry', '2001-08-21'),

('S003', 'Charlie', 'Physics', '1999-11-15'),

('S004', 'David', 'Computer Science', '2000-03-09');

INSERT INTO SOCIETY (SocID, SocName, MentorName, Date, TotalSeats) VALUES

('SOC001', 'Debating', 'John Doe', '2023-01-01', 50),

('SOC002', 'Dancing', 'Jane Smith', '2023-01-05', 30),

('SOC003', 'NSS', 'Michael Gupta', '2023-02-10', 40);

INSERT INTO ENROLLMENT (RollNo, SID, DateOfEnrollment) VALUES

('S001', 'SOC001', '2023-01-02'),

('S001', 'SOC002', '2023-01-03'),

('S002', 'SOC003', '2023-01-04'),

('S003', 'SOC001', '2023-01-05'),

('S004', 'SOC002', '2023-01-06'),

('S004', 'SOC003', '2023-01-07');

-- Queries

-- 1. Retrieve names of students enrolled in any society

SELECT DISTINCT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo;

-- 2. Retrieve all society names

SELECT SocName FROM SOCIETY;

-- 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 WHERE Course 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 society details with more than N TotalSeats where N is to be input by the user

-- Assuming N is 40

SELECT \* FROM SOCIETY WHERE TotalSeats > 40;

-- 7. Update society table for mentor name of a specific society

UPDATE SOCIETY SET MentorName = 'New Mentor' WHERE SocID = 'SOC001';

-- 8. Find society names in which more than five students have enrolled

SELECT s.SocName

FROM SOCIETY s

JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName

HAVING COUNT(\*) > 5;

-- 9. Find the name of youngest student enrolled in society ‘NSS’

SELECT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

WHERE e.SID = 'SOC003'

ORDER BY s.DOB ASC

LIMIT 1;

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

SELECT s.SocName

FROM SOCIETY s

JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName

ORDER BY COUNT(\*) DESC

LIMIT 1;

-- 11. Find the name of two least popular societies (on the basis of enrolled students)

SELECT s.SocName

FROM SOCIETY s

JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName

ORDER BY COUNT(\*) ASC

LIMIT 2;

-- 12. Find the student names who are not enrolled in any society

SELECT StudentName FROM STUDENT WHERE RollNo NOT IN (SELECT RollNo FROM ENROLLMENT);

-- 13. Find the student names enrolled in at least two societies

SELECT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

GROUP BY s.StudentName

HAVING COUNT(\*) >= 2;

-- 14. Find society names in which maximum students are enrolled

SELECT s.SocName

FROM SOCIETY s

JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName

ORDER BY COUNT(\*) DESC

LIMIT 1;

-- 15. Find names of all students who have enrolled in any society and society names in which at least one student has enrolled

SELECT DISTINCT s.StudentName, soc.SocName

FROM STUDENT s

LEFT JOIN ENROLLMENT e ON s.RollNo = e.RollNo

LEFT JOIN SOCIETY soc ON e.SID = soc.SocID;

-- 16. Find names of students who are enrolled in any of the three societies ‘Debating’, ‘Dancing’ and ‘Sashakt’.

SELECT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

JOIN SOCIETY soc ON e.SID = soc.SocID

WHERE soc.SocName IN ('Debating', 'Dancing', 'Sashakt');

-- 17. Find society names such that its mentor has a name with ‘Gupta’ in it.

SELECT SocName

FROM SOCIETY

WHERE MentorName LIKE '%Gupta%';

-- 18. Find the society names in which the number of enrolled students is only 10% of its capacity.

SELECT s.SocName

FROM SOCIETY s

JOIN (

SELECT SID, COUNT(\*) AS NumEnrolled

FROM ENROLLMENT

GROUP BY SID

) e ON s.SocID = e.SID

WHERE e.NumEnrolled <= 0.1 \* s.TotalSeats;

-- 19. Display the vacant seats for each society.

SELECT s.SocName, s.TotalSeats - COUNT(e.RollNo) AS VacantSeats

FROM SOCIETY s

LEFT JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName;

-- 20. Increment Total Seats of each society by 10%

UPDATE SOCIETY SET TotalSeats = TotalSeats \* 1.1;

-- 21. Add the enrollment fees paid (‘yes’/’No’) field in the enrollment table.

ALTER TABLE ENROLLMENT ADD EnrollmentFees

Paid ENUM('Yes', 'No');

-- 22. Update date of enrollment of society id ‘s1’ to ‘2018-01-15’, ‘s2’ to current date and ‘s3’ to ‘2018-01-02’.

UPDATE ENROLLMENT

SET DateOfEnrollment = CASE SID

WHEN 's1' THEN '2018-01-15'

WHEN 's2' THEN CURRENT\_DATE

WHEN 's3' THEN '2018-01-02'

ELSE DateOfEnrollment

END;

-- 23. Create a view to keep track of society names with the total number of students enrolled in it.

CREATE VIEW Society\_Enrollment\_Count AS

SELECT s.SocName, COUNT(e.RollNo) AS TotalEnrolled

FROM SOCIETY s

LEFT JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocName;

-- 24. Find student names enrolled in all the societies.

SELECT s.StudentName

FROM STUDENT s

WHERE NOT EXISTS (

SELECT SocID FROM SOCIETY

EXCEPT

SELECT SID FROM ENROLLMENT WHERE RollNo = s.RollNo

);

-- 25. Count the number of societies with more than 5 students enrolled in it

SELECT COUNT(\*) AS NumSocieties

FROM (

SELECT SID, COUNT(\*) AS NumEnrolled

FROM ENROLLMENT

GROUP BY SID

HAVING NumEnrolled > 5

) AS PopularSocieties;

-- 26. Add column Mobile number in student table with default value ‘9999999999’

ALTER TABLE STUDENT ADD MobileNumber VARCHAR(15) DEFAULT '9999999999';

-- 27. Find the total number of students whose age is > 20 years.

SELECT COUNT(\*) AS NumStudents

FROM STUDENT

WHERE TIMESTAMPDIFF(YEAR, DOB, CURDATE()) > 20;

-- 28. Find names of students who are born in 2001 and are enrolled in at least one society.

SELECT s.StudentName

FROM STUDENT s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

WHERE YEAR(s.DOB) = 2001;

-- 29. Count all societies whose name starts with ‘S’ and ends with ‘t’ and at least 5 students are enrolled in the society.

SELECT COUNT(\*) AS NumSocieties

FROM SOCIETY s

JOIN (

SELECT SID, COUNT(\*) AS NumEnrolled

FROM ENROLLMENT

GROUP BY SID

HAVING NumEnrolled >= 5

) e ON s.SocID = e.SID

WHERE SocName LIKE 'S%t';

-- 30. Display the following information:

-- Society name, Mentor name, Total Capacity, Total Enrolled, Unfilled Seats

SELECT s.SocName, s.MentorName, s.TotalSeats, COUNT(e.RollNo) AS TotalEnrolled, (s.TotalSeats - COUNT(e.RollNo)) AS UnfilledSeats

FROM SOCIETY s

LEFT JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY s.SocID;

```

For the database administration commands:

1. \*\*Create User\*\*:

```sql

CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';

```

2. \*\*Create Role\*\*:

```sql

CREATE ROLE 'role\_name';

```

3. \*\*Grant Privileges to a Role\*\*:

```sql

GRANT SELECT, INSERT, UPDATE ON database\_name.table\_name TO 'role\_name';

```

4. \*\*Revoke Privileges from a Role\*\*:

```sql

REVOKE SELECT, INSERT, UPDATE ON database\_name.table\_name FROM 'role\_name';

```

5. \*\*Create Index\*\*:

```sql

CREATE INDEX index\_name ON table\_name(column\_name);

```

Database administration commands are actions taken by a database administrator to manage users, roles, permissions, and optimize database performance. Let's break down each command:

**Create User**: This command creates a new user in the database. Users are entities that interact with the database by executing queries or accessing data.

**Create Role**: Roles are a way to group together privileges and assign them to users. Creating a role allows you to define a set of permissions that can be granted to multiple users.

**Grant Privileges to a Role**: This command assigns specific privileges (such as SELECT, INSERT, UPDATE) to a role. By granting privileges to a role, you can control what actions users assigned to that role can perform in the database.

**Revoke Privileges from a Role**: If you need to change or remove privileges from a role, you can use the REVOKE command. This command removes previously granted privileges from the specified role.

**Create Index**: Indexes are data structures that improve the speed of data retrieval operations on a database table. Creating an index on a column or set of columns allows the database to quickly locate rows based on the indexed values.

So, when the prompt mentions "Do the following database administration commands", it's asking you to perform actions like creating users, roles, granting or revoking privileges, and creating indexes in a database management system.

**1. Create User:**

Creating a user allows for specific individuals or applications to access the database. Let's create a user named 'student\_user' with password 'password123':

CREATE USER 'student\_user'@'localhost' IDENTIFIED BY 'password123';

**2. Create Role:**

Roles help in organizing users based on their responsibilities or access levels. Let's create a role named 'student\_role':

CREATE ROLE 'student\_role';

**3. Grant Privileges to a Role:**

Granting privileges to a role allows users assigned to that role to perform certain actions on database objects. Let's grant SELECT privilege on the STUDENT table to the 'student\_role':

GRANT SELECT ON database\_name.STUDENT TO 'student\_role';

Replace 'database\_name' with your actual database name.

**4. Revoke Privileges from a Role:**

If there's a need to change or remove privileges from a role, you can revoke them. For instance, let's revoke UPDATE privilege from the 'student\_role':

REVOKE UPDATE ON database\_name.STUDENT FROM 'student\_role';

**5. Create Index:**

Indexes improve query performance by allowing the database system to quickly retrieve rows. Let's create an index on the 'RollNo' column of the STUDENT table:

CREATE INDEX idx\_rollno ON STUDENT(RollNo);

This creates an index named 'idx\_rollno' on the 'RollNo' column of the 'STUDENT' table.

After executing these commands, 'student\_user' will be able to access the database using the provided credentials, 'student\_role' will have SELECT privilege on the 'STUDENT' table, and an index will be created on the 'RollNo' column for faster retrieval of data based on roll numbers.