ask	SQL Syntax
Create `STUDENT` Table	`CREATE TABLE STUDENT (student_id INT PRIMARY KEY, Name VARCHAR(50), department VARCHAR(10), mark1 INT, mark2 INT, cgpa FLOAT);`
Insert Data into `STUDENT` Table	`INSERT INTO STUDENT VALUES (101, 'Anu', 'CS', 85, 90, 9.0), (102, 'Rahul', 'EC', 78, 82, 8.5), ...;`
Rename Table `HOSTEL_DETAILS` to `HOSTEL`	`ALTER TABLE HOSTEL_DETAILS RENAME TO HOSTEL;`
Update CGPA for Student with ID 103	`UPDATE STUDENT SET cgpa = 8.0 WHERE student_id = 103;`
Display Students with CGPA > 8	`SELECT Name FROM STUDENT WHERE cgpa > 8.0;`
Delete Student with Room Number 108	`DELETE FROM HOSTEL WHERE roomno = 108;`
Update Department for 'Anu'	`UPDATE STUDENT SET department = 'EC' WHERE Name = 'Anu' AND department = 'ES';`

Q2

Task	SQL Syntax
Create `works` Table with Constraints	`CREATE TABLE works (emp_id VARCHAR(10), company_name VARCHAR(50), salary FLOAT, FOREIGN KEY (emp_id) REFERENCES emp(emp_id), PRIMARY KEY (emp_id, company_name));`
Create `manages` Table with Constraints	`CREATE TABLE manages (emp_id VARCHAR(10), manager_id VARCHAR(10), FOREIGN KEY (emp_id) REFERENCES emp(emp_id), FOREIGN KEY (manager_id) REFERENCES emp(emp_id), PRIMARY KEY (emp_id, manager_id));`

Alter `emp` Table to Add Constraint on `emp_name`	`ALTER TABLE emp ADD CONSTRAINT emp_name_not_null CHECK (emp_name IS NOT NULL);`
Find Names of Employees Who Work for SBI	`SELECT emp_name FROM emp INNER JOIN works ON emp.emp_id = works.emp_id WHERE works.company_name = 'SBI';`
Find Employees Who Live in Same City as Their Company	`SELECT emp_name FROM emp INNER JOIN works ON emp.emp_id = works.emp_id INNER JOIN company ON works.company_name = company.company_name WHERE emp.city = company.city;`
Find Employees and Their Managers Who Live in Same City and Street	`SELECT e.emp_name AS employee_name, m.emp_name AS manager_name FROM emp e INNER JOIN manages mg ON e.emp_id = mg.emp_id INNER JOIN emp m ON mg.manager_id = m.emp_id WHERE e.city = m.city AND e.street_no = m.street_no;`
Find Employees Earning More Than Average Salary in Company	`SELECT emp_name FROM works w INNER JOIN emp e ON w.emp_id = e.emp_id WHERE w.salary > (SELECT AVG(salary) FROM works WHERE company_name = w.company_name);`
Find Company That Pays Least Total Salary Along with Salary Paid	`SELECT company_name, SUM(salary) AS total_salary FROM works GROUP BY company_name ORDER BY total_salary ASC LIMIT 1;`
Give Managers of SBI a 10% Raise	`UPDATE works SET salary = salary * 1.10 WHERE emp_id IN (SELECT emp_id FROM manages INNER JOIN works ON manages.emp_id = works.emp_id WHERE works.company_name = 'SBI');`
Find Company with Most Employees	`SELECT company_name FROM works GROUP BY company_name ORDER BY COUNT(emp_id) DESC LIMIT 1;`

Find Companies Whose Employees Earn More Than Average Salary at Indian Bank	`SELECT company_name FROM works w1 WHERE (SELECT AVG(salary) FROM works WHERE company_name = 'Indian Bank') < (SELECT AVG(salary) FROM works WHERE company_name = w1.company_name) GROUP BY company_name;`
Find Employees Who Earn More Than Each Employee of Indian Bank	`SELECT emp_name, salary FROM works WHERE salary > ALL (SELECT salary FROM works WHERE company_name = 'Indian Bank');`

Q3

Task	SQL Syntax
Implement `UPPER` function on `Bank-name`	`SELECT UPPER(bank_name) FROM account;`
Implement `LOWER` function on `Bank-name`	`SELECT LOWER(bank_name) FROM account;`
Implement `LENGTH` function on `Bank-name`	`SELECT LENGTH(bank_name) FROM account;`
Implement `REPLACE` function on `Bank-name`	`SELECT REPLACE(bank_name, 'Bank', 'Banking') FROM account;`
Implement `ROUND` function on `Account balance`	`SELECT ROUND(account_balance, 2) FROM account;`
Implement `CEIL` function on `Account balance`	`SELECT CEIL(account_balance) FROM account;`
Implement `FLOOR` function on `Account balance`	`SELECT FLOOR(account_balance) FROM account;`
Implement `SIGN` function on `Account balance`	`SELECT SIGN(account_balance) FROM account;`
Implement `CURRENT_DATE` function	`SELECT CURRENT_DATE FROM account;`

Implement `SYSDATE` function	`SELECT SYSDATE() FROM account;`
Extract Month from Date (Assuming a date column `account_date`)	`SELECT EXTRACT(MONTH FROM account_date) FROM account;`
Extract Year from Date (Assuming a date column `account_date`)	`SELECT EXTRACT(YEAR FROM account_date) FROM account;`
Implement `ASCII` function for characters	`SELECT ASCII('A'), ASCII('B'), ASCII('C'), ASCII('D'), ASCII('E');`

Q4

Here's a quick revision table with SQL syntax for the tasks:

Task	**SQL Syntax**
Create `Subject` Table	`CREATE TABLE Subject (subject_code INT PRIMARY KEY, subject_name VARCHAR(50), max_marks INT, faculty_code INT, FOREIGN KEY (faculty_code) REFERENCES Faculty(faculty_code));`
Display the Number of Faculties	`SELECT COUNT(*) AS num_faculties FROM Faculty;`
Display Details of Students with Name Starting with 'A'	`SELECT * FROM Student WHERE student_name LIKE 'A%';`
Display Total Number of Records in `Student` Table	`SELECT COUNT(*) AS total_students FROM Student;`

Find Number of Branches in `Student` Table	`SELECT COUNT(DISTINCT student_branch) AS num_branches FROM Student;`
Display Faculties and Their Allotted Subjects	`SELECT Faculty.faculty_name, Subject.subject_name FROM Faculty INNER JOIN Subject ON Faculty.faculty_code = Subject.faculty_code;`
Display Names of Faculties Teaching More Than One Subject	`SELECT Faculty.faculty_name FROM Faculty INNER JOIN Subject ON Faculty.faculty_code = Subject.faculty_code GROUP BY Faculty.faculty_name HAVING COUNT(Subject.subject_name) > 1;`
Display Subject Names and Marks in Ascending Order of Marks	`SELECT subject_name, max_marks FROM Subject ORDER BY max_marks ASC;`
Rename `max_marks` Column to `Maximum`	`ALTER TABLE Subject CHANGE max_marks Maximum INT;`

Q5

Task	SQL Syntax
Display name and salary for employees whose salary is not in the range of 5000 and 35000	`SELECT name, salary FROM Emp WHERE salary NOT BETWEEN 5000 AND 35000;`

Display employee name, job, and start date for employees hired between Feb 20, 1990, and May 1, 1998, ordered by start date	`SELECT name, job, hiredate FROM Emp WHERE hiredate BETWEEN '1990-02-20' AND '1998-05-01' ORDER BY hiredate ASC;`
List name and salary of employees earning between 5000 and 12000 and in department 2 or 4	`SELECT name AS Employee, salary AS "Monthly Salary" FROM Emp WHERE salary BETWEEN 5000 AND 12000 AND department_no IN (2, 4);`
Display names and hire dates of employees hired in 1994	`SELECT name, hiredate FROM Emp WHERE YEAR(hiredate) = 1994;`
Display name, salary, and commission for employees earning commissions, sorted by salary and commission	`SELECT name, salary, commission FROM Emp WHERE commission IS NOT NULL ORDER BY salary DESC, commission DESC;`
Display name and job title of employees who do not have a manager	`SELECT name, job FROM Emp WHERE manager_id IS NULL;`
Display name of employees where the third letter of the name is 'a'	`SELECT name FROM Emp WHERE SUBSTRING(name, 3, 1) = 'a';`
Display name of employees who have both 'a' and 'e' in their name	`SELECT name FROM Emp WHERE name LIKE '%a%' AND name LIKE '%e%';`

Display name, job, and salary for employees with job 'Sales Representative' or 'Stock Clerk' and salary not 20000, 4000, or 7000	`SELECT name, job, salary FROM Emp WHERE job IN ('Sales Representative', 'Stock Clerk') AND salary NOT IN (20000, 4000, 7000);`
Display name, department number, and department name for all employees	`SELECT Emp.name, Emp.department_no, Depart.department_name FROM Emp INNER JOIN Depart ON Emp.department_no = Depart.department_id;`
Display employee numbers and names of employees who work in a department with any employee whose name contains a 'u'	`SELECT DISTINCT e1.emp_no, e1.name FROM Emp e1 INNER JOIN Emp e2 ON e1.department_no = e2.department_no WHERE e2.name LIKE '%u%';`
Display name and hire date of any employee in the same department as 'Amit', excluding 'John'	`SELECT e1.name, e1.hiredate FROM Emp e1 INNER JOIN Emp e2 ON e1.department_no = e2.department_no WHERE e2.name = 'Amit' AND e1.name != 'John';`

Q6

Here's a quick revision table for the SQL queries related to the **Movie Database**:

Task	SQL Syntax
Select Movies Directed by Alfred Hitchcock	`SELECT Mov_Title FROM Movies JOIN Director ON Movies.Dir_id = Director.Dir_id WHERE Director.Dir_Name = 'Alfred Hitchcock';`
Select Movies with Actors who Acted in More Than One Movie	`SELECT DISTINCT m.Mov_Title FROM Movies m JOIN Movie_Cast mc ON m.Mov_id = mc.Mov_id WHERE mc.Act_id IN (SELECT Act_id FROM Movie_Cast GROUP BY Act_id HAVING COUNT(Mov_id) >= 2);`

Select Actors from Movies Released Before 2000 or After 2015	`SELECT DISTINCT a.Act_Name FROM Actor a JOIN Movie_Cast mc ON a.Act_id = mc.Act_id JOIN Movies m ON mc.Mov_id = m.Mov_id WHERE m.Mov_year < 2000 OR m.Mov_year > 2015;`
Update Movie Rating for Movies Directed by Steven Spielberg	`UPDATE Rating r JOIN Movies m ON r.Mov_id = m.Mov_id JOIN Director d ON m.Dir_id = d.Dir_id SET r.Rev_stars = 5 WHERE d.Dir_Name = 'Steven Spielberg';`
Select Movies with Maximum Rating Stars	`SELECT m.Mov_Title, MAX(r.Rev_stars) AS Highest_Stars FROM Movies m JOIN Rating r ON m.Mov_id = r.Mov_id GROUP BY m.Mov_Title HAVING COUNT(r.Rev_stars) > 0 ORDER BY m.Mov_Title;`

Q7

Here's a quick revision table for tasks involving **Views and Assertions**:

Task	SQL Syntax
Create a View for the Employee Table	`CREATE VIEW EmployeeView AS SELECT * FROM Employee;`
Vertical Partitioning: Display Specific Columns from the View	`SELECT Emp_no, Emp_name, Dept_no, Job FROM EmployeeView;`
Horizontal Partitioning: Display Employees with Job Title 'ASP'	`SELECT * FROM EmployeeView WHERE Job = 'ASP';`

Update the View and Reflect Changes in the Original Table	`UPDATE EmployeeView SET Dept_no = 105 WHERE Emp_no = 2;`
Drop the View	`DROP VIEW EmployeeView;`

This summary provides concise SQL commands for each task to help with quick revision.

Q8

Task	SQL Syntax
Create Employee Table	`CREATE TABLE Employee (Emp_id INT PRIMARY KEY, Emp_name VARCHAR(100), Designation VARCHAR(50), Dept_no INT, Salary INT);`

Insert 5 Initial Rows	`START TRANSACTION;`
	
	`INSERT INTO Employee (Emp_id, Emp_name, Designation, Dept_no, Salary) VALUES (1, 'Alice', 'Manager', 101, 60000), ...;`

Create Savepoint 's'	`SAVEPOINT s;`
Add One Extra Row	`INSERT INTO Employee (Emp_id, Emp_name, Designation, Dept_no, Salary) VALUES (6, 'Frank', 'Technician', 106, 35000);`
Display the Table	`SELECT * FROM Employee;`
Rollback to Savepoint 's'	`ROLLBACK TO s;`
Display Table After Rollback	`SELECT * FROM Employee;`

Commit the Transaction	`COMMIT;`
Final Table Display After Commit	`SELECT * FROM Employee;`

This table helps quickly revise key TCL commands like **SAVEPOINT**, **ROLLBACK**, and **COMMIT** along with sample queries for practical use.

Q10

Task	SQL Syntax

Create Trigger for Auto Prizes	`DELIMITER $$`
	
	`CREATE TRIGGER AddPrizesAfterEvent`
	
	`AFTER INSERT ON Event`
	
	`FOR EACH ROW BEGIN`
	
	`INSERT INTO Prizes (Money, Event_id, Rank, Year) VALUES (1500, NEW.Event_id, 1, YEAR(CURDATE()));`
	
	`INSERT INTO Prizes (Money, Event_id, Rank, Year) VALUES (1000, NEW.Event_id, 2, YEAR(CURDATE()));`
	
	`INSERT INTO Prizes (Money, Event_id, Rank, Year) VALUES (500, NEW.Event_id, 3, YEAR(CURDATE())); END$$ DELIMITER ;`

Q10

Here’s a quick reference table for MongoDB CRUD operations:

Operation	Command	Description	Example

Create	`insertOne()`	Add a single document to a collection.	`db.myCollection.insertOne({ name: "Amal", age: 20 })`
	`insertMany()`	Add multiple documents.	`db.myCollection.insertMany([{ name: "Rida", age: 21 }, { name: "Ashitha", age: 22 }])`
Read	`find()`	Retrieve all documents.	`db.myCollection.find()`
	`find(query)`	Retrieve specific documents.	`db.myCollection.find({ name: "Amal" })`
Update	`updateOne()`	Update the first matching document.	`db.myCollection.updateOne({ name: "Amal" }, { $set: { age: 21 } })`
	`updateMany()`	Update all matching documents.	`db.myCollection.updateMany({ age: { $lt: 25 } }, { $set: { status: "active" } })`
Delete	`deleteOne()`	Delete the first matching document.	`db.myCollection.deleteOne({ name: "Amal" })`
	`deleteMany()`	Delete all matching documents.	`db.myCollection.deleteMany({ age: { $lt: 20 } })`