Module – 3 Practice Question

Set – 1 (Constraints, ALTER, Tricky Updates)

1. You have a table employees with columns:
   employees(id INT PRIMARY KEY, first_name VARCHAR(50), last_name VARCHAR(50), salary DECIMAL(10,2), department_id INT).
   - a) Write a query to add a new column bonus DECIMAL(10,2) with default value 0 for all existing and future rows.
   - b) Now modify bonus so that it cannot be negative (use an appropriate constraint).
   - c) Later, the company decides to remove the bonus feature completely. Write the query to safely drop the bonus column only if it exists.
2. Consider two tables:
   departments(dept_id INT PRIMARY KEY, dept_name VARCHAR(50) UNIQUE)
   employees(emp_id INT PRIMARY KEY, emp_name VARCHAR(50), dept_id INT, FOREIGN KEY (dept_id) REFERENCES departments(dept_id)).
   - a) Insert 1 department and 3 employees for that department.
   - b) Try to delete that department and ensure all its employees are also deleted automatically by modifying the schema (do not use a separate DELETE on employees).
   - c) Write the final CREATE TABLE employees statement that supports this behavior.
3. A table salaries has:
   salaries(emp_id INT, pay_month DATE, amount DECIMAL(10,2), PRIMARY KEY(emp_id, pay_month)).
   - a) Write a query to increase salary by 10% only for those employees whose current month's amount is less than the average salary of all employees for that same month.
   - b) Write a query to list employees whose salary has never decreased compared to any previous month.

Set – 2 (Joins, Aggregation, Subqueries, Views)

1. You have the tables:
   students(student_id INT PRIMARY KEY, first_name VARCHAR(50), last_name VARCHAR(50))
   courses(course_id INT PRIMARY KEY, course_name VARCHAR(100), credits INT)
   enrollments(enrollment_id INT PRIMARY KEY, student_id INT, course_id INT, enrollment_date DATE).
   - a) Write a query to list all students who are enrolled in more than 2 courses, along with the total number of courses they are enrolled in.
   - b) Write a query to list the students who are not enrolled in any course.
   - c) Create a view popular_courses that shows course_id, course_name, and total_enrollments for courses having more than 5 enrollments.
2. You have:
   employees(emp_id INT PRIMARY KEY, emp_name VARCHAR(50), department_id INT, salary DECIMAL(10,2))
   departments(department_id INT PRIMARY KEY, department_name VARCHAR(50)).
   - a) Write a query to find the department(s) having the highest average salary (if multiple departments tie, show all of them).
   - b) Write a query to list employees whose salary is greater than the average salary of their own department.
   - c) Create a view high_earners that shows emp_id, emp_name, department_name, salary only for such employees.

3. A table orders has:
   orders(order_id INT PRIMARY KEY, customer_id INT, order_date DATE, total_amount DECIMAL(10,2)).
   - a) Write a query to find the customer(s) who spent the maximum total amount across all their orders.
   - b) Write a query to show, for each month, the total sales and the percentage contribution of that month to the total annual sales (assume one year's data).

Set – 3 (DCL, Security, Window Functions, Complex Conditions)

1. A database LibraryDB has table Books(book_id INT PRIMARY KEY, title VARCHAR(100), author VARCHAR(100), publication_year INT, genre VARCHAR(50), available_copies INT).
   - a) Create a role librarian_role that can SELECT, INSERT, UPDATE, DELETE on Books.
   - b) Create a role member_role that can only SELECT on Books.
   - c) Create two users, assign them to these roles, and then write the commands to revoke DELETE permission from librarian_role without affecting other permissions.
2. You have table transactions:
   transactions(txn_id INT PRIMARY KEY, account_id INT, txn_date DATE, amount DECIMAL(10,2)).
   - a) Write a query to list each transaction along with the running total (cumulative sum) of amount for that account_id, ordered by txn_date.
   - b) Write a query to detect any account where the total amount of transactions in a single day exceeds 1,00,000.
   - c) Using a window function, list the top 3 highest transactions per account_id.

3. Consider table attendance:
   attendance(id INT PRIMARY KEY, student_id INT, attendance_date DATE, status ENUM('P','A')).
   - a) Write a query to calculate, for each student, the attendance percentage (P out of total records) and list only those students whose attendance is below 75%.
   - b) Write a query to find any continuous streak of 3 or more absences ('A') for any student.
   - c) Suggest and write a constraint or mechanism (CHECK or trigger, depending on DB support) to prevent inserting attendance records with a future date.

Set – 4 (DDL, Constraints, Indexing, Schemas)

1. You are designing a database edunetDB for an online course platform.
   - a) Create a schema training for grouping all course-related tables.
   - b) Inside this schema, create a table training.courses with:
     course_id INT PRIMARY KEY,
     course_name VARCHAR(100) NOT NULL,
     level ENUM('Beginner','Intermediate','Advanced'),
     created_at DATETIME DEFAULT CURRENT_TIMESTAMP.
   - c) Add a UNIQUE constraint on course_name and explain what will happen if you try to insert two rows with the same course_name.
2. The table edunet_employees is frequently searched by email and department.
   - a) Write SQL to create a composite index on (department, email).
   - b) Explain using a query example when this composite index will be used efficiently and when it may be ignored by the optimizer.
   - c) Write a query that forces the optimizer to use this index (assume your SQL dialect supports index hints).

3. You have a table logs(id INT PRIMARY KEY, log_time DATETIME, message TEXT).
    - a) Create a partitioning or logical design (using extra columns or check constraints) so that old data (older than 1 year) can be quickly deleted.
    - b) Write the SQL statement to remove all logs older than 1 year while keeping the table structure.
    - c) Explain why TRUNCATE is not suitable in this scenario.

Set – 5 (DML, Joins, Subqueries, Functions)

1. Consider:
   students(student_id INT PK, first_name VARCHAR(50), last_name VARCHAR(50), dob DATE)
   courses(course_id INT PK, course_name VARCHAR(100), credits INT)
   enrollments(enrollment_id INT PK, student_id FK, course_id FK, enrollment_date DATE).
    - a) Write a query to list each student with age (in years, based on dob) and total credits of all enrolled courses.
    - b) List only those students whose total credits are greater than the average credits of all students.
    - c) Use a subquery or CTE to find students who have never enrolled in any course with credits < 3.
2. You have:
   edunet_employees(employee_id INT PK, first_name VARCHAR(50),

last_name VARCHAR(50), salary DECIMAL(10,2), department_id INT) edunet_departments(department_id INT PK, department_name VARCHAR(50)).

- a) Write a single query to show, for each department, the highest salary and the employee(s) who earn that salary. Use aggregate + join or window functions.
- b) Write a query to list departments that have fewer than 3 employees using a subquery.
- c) Using a self-join on edunet_employees, list each employee along with their manager name (manager_id column references employee_id in same table).

3. A table orders(order_id INT PK, customer_id INT, order_date DATE, total_amount DECIMAL(10,2)).
    - a) Write a query to calculate for each customer: total_amount_spent, number_of_orders, average_order_value.
    - b) Using a window function, list all orders along with the running total of total_amount per customer ordered by order_date.
    - c) Find customers whose maximum single order value is greater than twice their average order value.

Set – 6 (DCL, Authentication/Authorization, Encryption, Backup)

1. In a HospitalDB system, you have table patients(patient_id INT PK, name VARCHAR(100), diagnosis VARCHAR(255), bill_amount DECIMAL(10,2)).
    - privileges on patients using GRANT (doctors can SELECT/UPDATE diagnosis, nurses can SELECT, receptionists only SELECT bill_amount).
    - b) Create a user doctor_rahul and assign doctor_role; show SQL to verify his permissions.
    - c) Later, restrict doctors from updating bill_amount while keeping their existing permissions. Write the necessary REVOKE or GRANT changes.

2. You are asked to secure sensitive user passwords in a table app_users(user_id INT PK, username VARCHAR(50), password_plain VARCHAR(100)).
    - a) Redesign the table to store encrypted passwords using suitable column(s) and remove plain-text storage.
    - b) Write sample SQL showing how to insert a new user with an encrypted password using a built-in encryption function (name the function generically, no vendor-specific syntax required).
    - c) Write SQL to retrieve and decrypt the password for verification (only for demonstration; mention why this is a bad practice in real systems).
3. A college uses LibraryDB with table issue_log(issue_id INT PK, book_id INT, student_id INT, issue_date DATE, return_date DATE, fine_amount DECIMAL(10,2)).
    - a) Propose a backup strategy using FULL, DIFFERENTIAL, and TRANSACTION LOG backups for this database with justification of frequency.
    - b) Write generic SQL or pseudo-SQL statements to perform a full backup and then a differential backup.
    - c) Given an accidental DELETE of all rows from issue_log at 3:15 PM, describe (stepwise) how to restore the database to 3:14 PM using backup + transaction log.

Set – 7 (SQL Injection, Prevention, Advanced Queries)

1. A web application builds this dynamic query from user input:
   "SELECT * FROM users WHERE username = '" + user_input + "' AND password = '" + pass_input + "';"
    - a) Explain how an attacker could log in as admin without knowing the password using SQL injection.
    - b) Rewrite the same logic using a parameterized query / prepared statement so that injection is not possible.

- c) State two additional precautions (other than prepared statements) that should be implemented at SQL or application level.

2. Consider:
   accounts(account_id INT PK, customer_id INT, balance DECIMAL(12,2))
   transactions(txn_id INT PK, account_id INT, txn_date DATETIME, amount DECIMAL(10,2)).
   - a) Write a query to list any account where the total withdrawals (negative amounts) in a single day exceed 50,000.
   - b) Using a window function, show each transaction with a 7-day rolling sum of amount per account.
   - c) Suggest an SQL-level check (constraint, trigger, or both) that helps reduce the risk of fraudulent large withdrawals and write a sample definition.
3. For a university system:
   users(user_id INT PK, username VARCHAR(50), role ENUM('Admin','Faculty','Student'))
   audit_log(log_id INT PK, user_id INT, action VARCHAR(100), action_time DATETIME).
   - a) Write a trigger that automatically inserts a row into audit_log whenever a user with role='Admin' performs an UPDATE on any row in users.
   - b) Explain how this trigger supports accountability and auditing in database security.
   - c) Extend the design by adding a column ip_address in audit_log and update the trigger to store it (assume it is passed via a session variable).

Set – 8: InventoryDB (INNER JOIN focus)

You are given a database InventoryDB.

Tables (DDL)

- products(product_id INT PK, product_name VARCHAR(100) NOT NULL, category_id INT, price DECIMAL(10,2), stock INT)
- categories(category_id INT PK, category_name VARCHAR(50) UNIQUE)
- suppliers(supplier_id INT PK, supplier_name VARCHAR(100), city VARCHAR(50))
- product_suppliers(product_id INT FK, supplier_id INT FK, PRIMARY KEY(product_id, supplier_id))

Tasks:

1. Write DDL to create all four tables with appropriate PRIMARY KEY and FOREIGN KEY constraints.
2. Insert at least 5 products, 3 categories, 3 suppliers and 1–2 mappings per product in product_suppliers (DML).
3. Increase price by 8% for all products in category "Electronics" and decrease stock by 1 for every sale of that product (write one UPDATE and one DELETE/INSERT style example).
4. Using an INNER JOIN, list each product with its category_name and all supplier_name values (one row per product–supplier pair).
5. Using an INNER JOIN + aggregation, list each supplier with the total number of distinct products supplied.
6. Create a role inventory_manager that can SELECT, INSERT, UPDATE on products and product_suppliers, but cannot DELETE. Grant this role to user inv_user. (DCL).
7. Revoke UPDATE on price from inventory_manager without affecting other privileges.

Set – 9: CollegeDB (LEFT / RIGHT / SELF JOIN focus)

You are given a database CollegeDB.

Tables (DDL)

- departments(dept_id INT PK, dept_name VARCHAR(100) NOT NULL)

- faculty(faculty_id INT PK, faculty_name VARCHAR(100), dept_id INT FK, salary DECIMAL(10,2))
- courses(course_id INT PK, course_title VARCHAR(100), dept_id INT FK, credits INT)
- faculty_courses(faculty_id INT FK, course_id INT FK, PRIMARY KEY(faculty_id, course_id))

Tasks:

1. Write DDL to create all four tables with FK from faculty.dept_id and courses.dept_id to departments, and from faculty_courses to faculty and courses.
2. Insert at least 4 departments, 6 faculty, 6 courses and 1–2 teaching assignments per course (DML).
3. Using a LEFT JOIN, list all departments and the total number of faculty in each; departments with zero faculty must also appear.
4. Using a RIGHT JOIN, list all courses and the name of the faculty teaching them; if any course is not assigned, show NULL for faculty_name.
5. Using a SELF JOIN on faculty, assume column manager_id (FK to faculty_id) is added; write the ALTER TABLE to add this column and then query each faculty with their manager's name.
6. Create a view dept_load that shows dept_name, total_courses, total_faculty using joins and GROUP BY.
7. Create role hr_role with SELECT on faculty and departments, and UPDATE only on faculty.salary. Grant this role to user hr_analyst and show one statement to verify their grants.

Set – 10: OnlineBankDB (FULL/OUTER JOIN logic + DCL)

Assume your SQL dialect supports FULL OUTER JOIN (if not, simulate with UNION of LEFT and RIGHT joins).

Tables (DDL)

- customers(customer_id INT PK, full_name VARCHAR(100), city VARCHAR(50))
- accounts(account_id INT PK, customer_id INT FK, account_type VARCHAR(20), balance DECIMAL(12,2))
- transactions(txn_id INT PK, account_id INT FK, txn_date DATE, amount DECIMAL(10,2), txn_type ENUM('DEPOSIT','WITHDRAW'))
- login_users(user_id INT PK, username VARCHAR(50) UNIQUE, customer_id INT FK, password_hash VARBINARY(256))

Tasks:

1. Write DDL to create all four tables with appropriate PK–FK relationships and a CHECK that amount > 0 in transactions.
2. Insert at least 5 customers, 6 accounts (some customers with more than 1 account), and 10–12 transactions of both types (DML).
3. Using a FULL OUTER JOIN (or LEFT + RIGHT + UNION), list all customers and their accounts; include customers without accounts and accounts with missing customer records (test by inserting one orphan row).
4. Using an INNER JOIN, list each transaction with customer name, account_type and running balance per account, using a window function if allowed.
5. Using a LEFT JOIN, find customers who have no transactions in any of their accounts.
6. Create role teller_role that can SELECT on customers, accounts, transactions and INSERT into transactions only (no UPDATE/DELETE). Create role admin_role that has ALL privileges on OnlineBankDB. (DCL)
7. Create user teller_01 and grant teller_role; later, revoke INSERT on transactions from teller_role and give teller_01 only SELECT privileges. Show the GRANT and REVOKE statements.