1. Consider the following schema for a Library Database:

BOOK (Book\_id, Title, Publisher\_Name, Pub\_Year, No\_of\_copies) BOOK\_AUTHORS (Book\_id, Author\_Name)

PUBLISHER (Name, Address, Phone)

* 1. Retrieve details of all books in the Book\_id, title, name of publisher, authors
  2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun2017.
  3. Delete a book in BOOK table. Update the contents of other tables to reflect this data Manipulation operations.
  4. Create a view of all books and its number of copies that are currently available in the Library.
  5. Write a Pl/SQL procedure to display the book details of particular author.

1. Create a table employee (S.No, Name, Designation, branch),
   1. Alter the table by adding a column salary
   2. Copy the table employee as Emp
   3. Delete 2nd row from the table
   4. Drop the table
   5. Demonstrate the triggers for automatic updation.
2. Create a table called Employee (Emp\_no Emp\_name, Emp\_dept,Job ,Mgr ,Sal)
   1. By using the group by clause, display the Emp\_name who belongs to Emp\_dept=”xxx” along with salary
   2. Display lowest paid employee details under each department
   3. List the employee names in descending order.
   4. Rename the column of Employee table using alter command
   5. Insert row in employee table using Triggers.
3. Consider the following tables namely “DEPARTMENTS” and “EMPLOYEES” Departments (dept\_no , dept\_ name , dept\_location ),

Employees ( emp\_id , emp\_name , emp\_salary,dept\_no).

* 1. Develop a query to grant some privileges of employees table into departments table
  2. Develop a query to revoke all privileges of employees table from departments table
  3. Develop a query to revoke some privileges of employees table from departments table
  4. Write a query to implement the save point.
  5. Demonstrate the user defined procedure for the above employee database

1. Create the following tables,

Event (eventid, name, description,city) Participant (playerid, name, eventid, gender, year) Prizes (prizeid, prize-money, eventid, rank,year) Winners (prizeid, playerid)

* 1. Choose appropriate primary keys and foreign keys for the tables.
  2. Playerid should contain at least one digit character.
  3. Retrieve the name of events where all prize winners are females
  4. Create a non-updatable view to retrieve the names of all participants who won 1st prizes along with their event names
  5. Write a trigger to make sure that for every new event created, 3 prizes are created in prizes table. (1st prize - 1500, 2nd - 1000, 3rd 500)

1. Consider the schema for Movie Database: ACTOR (Act\_id, Act\_Name, Act\_Gender) DIRECTOR (Dir\_id, Dir\_Name, Dir\_Phone)

MOVIES (Mov\_id, Mov\_Title, Mov\_Year, Mov\_Lang, Dir\_id) MOVIE\_CAST (Act\_id, Mov\_id, Role)

* 1. List the titles of all movies directed by „XXXX‟.
  2. Find the movie names where one or more actors acted in two or more movies.
  3. List all actors who acted in a movie before 2010 and also in a movie after 2015 (use JOIN operation).
  4. Create a view of movies with a particular actor with director.
  5. Demonstrate the User defined function for the movie database.

1. Consider the schema for College Database:

STUDENT (*RegNo, StudName, Address, Phone, Gender*) SUBJECT (*Subcode, Title, Sem, Credits*)

MARKS (*RegNo, Subcode, Test1, Test2, Test3, Finalmark*)

* 1. Compute the total number of male and female students in each semester and in each section.
  2. Calculate the *Finalmark* (average of best two test marks) and update the corresponding table for all students.
  3. Categorize students based on the following criterion: If *Finalmark* = 81 to 100 then CAT = „Outstanding‟ If *Finalmark* = 51 to 80 then CAT = „Average‟

If *Finalmark* < 51 then CAT = „Weak

* 1. Create a view of Test3 marks of particular student in all subjects.
  2. Demonstrate the procedure for the above Database.

1. Create table as Bank ( S.No,Cust\_Name, Acc\_No, Balance, Branch),
   1. Select with where clause.
   2. Select with comparison operator.
   3. Update the balance in the second row.
   4. Select with between in the field balance.
   5. Write a trigger when balance is below 1000.
2. Create a table Account (Account\_No, Cust\_Name, Branch\_Name, Account\_Balance, Account\_Type) Select an appropriate primary key.
   1. Display the Cust\_Name and Account\_No of the customers of "Branch = XXXXX".
   2. Display the names and account types of all the customers whose account balance is more than 10,000.
   3. Add column Cust\_Date\_of Birth in the ACCOUNT table.
   4. Display Account\_No, Cust\_Name and Branch of all the customers whose account balance is less than 1,000.
   5. Write a procedure for the above Database.
3. Create the tables CUSTOMER (C\_ID, Name, Address, City, Mobile\_No) and ORDER (C\_ID, P\_ID, P\_Name, P\_COST),
   1. List the names and addresses of all the customers who have ordered products of customers than 500.
   2. List the names of all the products ordered whose cost is 1,000 or more.
   3. List the product names which are ordered by customers of "City = Delhi".
   4. Add column "Email\_id" in the CUSTOMER table.
   5. Demonstrate the user defined function for the above tables.
4. Create the tables SALESMAN (Salesman\_id, Name, City, Commission), CUSTOMER (Customer\_id, Cust\_Name, City, Grade,Salesman\_id), ORDERS (Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id)
   1. Find the name and numbers of all salesmen who had more than one customer.
   2. List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)
   3. Create a view that finds the salesman who has the customer with the highest order of a day.
   4. Perform the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.
   5. Demonstrate the Triggers for the above table.
5. Create Document, column and graph based data using NOSQL database tools.
6. Create an XML database for the student profile and validate it using XML schema.
7. Write user defined functions and stored procedures in SQL.
8. Create a Employee database and write SQL Triggers for insert, delete, and update operations in a database table.
9. Create a table Supplier (Sup\_No, Sup\_Name ,Item\_Supplied , Item\_Price , City)
   1. Write sql query to display Supplier numbers and Supplier names whose name starts with „S‟
   2. Write sql query to add a new column called CONTACTNO.
   3. Write sql query to display supplier numbers, Supplier names and item price for suppliers in Chennai in the ascending order of item price.
   4. Create a view on the table which displays only supplier numbers and supplier names.
   5. Demonstrate the procedure for the supplier table.