**Chit no 1 :Design and Develop SQLDDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc. CREATE TABLE EMPLOYEE ( EMP\_ID NUMBER(5) PRIMARY KEY, EMP\_NAME VARCHAR(10) NOT NULL, JOB\_NAME VARCHAR(10) NOT NULL, MANAGER\_ID NUMBER(5) DEFAULT '00000', HIRE\_DATE DATE, SALARY DECIMAL(6,2), COMMISSION DECIMAL(6,2) NULL, DEPT\_ID NUMBER(4));**

-- Create TABLE

CREATE TABLE EMPLOYEE (

EMP\_ID NUMBER(5) PRIMARY KEY,

EMP\_NAME VARCHAR(10) NOT NULL,

JOB\_NAME VARCHAR(10) NOT NULL,

MANAGER\_ID NUMBER(5) DEFAULT '00000',

HIRE\_DATE DATE,

SALARY DECIMAL(6,2),

COMMISSION DECIMAL(6,2) NULL,

DEPT\_ID NUMBER(4)

);

-- Create VIEW

CREATE VIEW EMPLOYEE\_VIEW AS

SELECT EMP\_ID, EMP\_NAME, JOB\_NAME, SALARY

FROM EMPLOYEE

WHERE SALARY > 50000;

-- Create INDEX

CREATE INDEX IDX\_EMPLOYEE\_DEPT\_ID ON EMPLOYEE(DEPT\_ID);

-- Create SEQUENCE

CREATE SEQUENCE EMPLOYEE\_SEQ

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

-- Create SYNONYM

CREATE SYNONYM EMPLOYEE\_SYNONYM FOR EMPLOYEE;

-- Add FOREIGN KEY CONSTRAINT

ALTER TABLE EMPLOYEE

ADD CONSTRAINT FK\_MANAGER\_ID

FOREIGN KEY (MANAGER\_ID)

REFERENCES EMPLOYEE(EMP\_ID);

-- Add CHECK CONSTRAINT

ALTER TABLE EMPLOYEE

ADD CONSTRAINT CHK\_SALARY\_RANGE

CHECK (SALARY BETWEEN 1000 AND 100000);

-- Add UNIQUE CONSTRAINT

ALTER TABLE EMPLOYEE

ADD CONSTRAINT UQ\_EMP\_NAME

UNIQUE (EMP\_NAME);

-- Add NOT NULL CONSTRAINT

ALTER TABLE EMPLOYEE

MODIFY HIRE\_DATE DATE NOT NULL;

**chit no 2 :Write at least 10 SQL queries on the suitable database application using SQL DML statements. Emp(Id,Name,Joining date,salary,dept)**

**1. Write a query in SQL to insert single record into table.**

**2. Write a query in SQL to multiple rows into table.**

**3. Display all the records from table.**

**4. Display all the records from table where salary is >8000.**

**5. Write a query in SQL to display records in ascending order of joining date .**

**6. Write a query in SQL to update salary of employee id=100 as 50000.**

**7. Write a query to remove record of employee id=106 from table;**

**8. Use DISTINCT keyword to display salary of an employee.**

**9. Write a query in SQL to display minimum / maximum salary of an employee.**

**10. Write a query in SQL to display total number of employees.**

**11. Display sum of salaries of dept=10.**

-- 1. Insert a single record into the table

INSERT INTO Emp (Id, Name, Joining\_date, Salary, Dept)

VALUES (101, 'John Doe', '2023-01-01', 9000, 20);

-- 2. Insert multiple rows into the table

INSERT INTO Emp (Id, Name, Joining\_date, Salary, Dept)

VALUES (102, 'Jane Smith', '2023-02-15', 10000, 30),

(103, 'Bob Johnson', '2023-03-10', 8500, 20),

(104, 'Alice Brown', '2023-04-20', 12000, 10);

-- 3. Display all records from the table

SELECT \* FROM Emp;

-- 4. Display records where salary is > 8000

SELECT \* FROM Emp WHERE Salary > 8000;

-- 5. Display records in ascending order of joining date

SELECT \* FROM Emp ORDER BY Joining\_date ASC;

-- 6. Update the salary of employee with ID=100 to 50000

UPDATE Emp SET Salary = 50000 WHERE Id = 100;

-- 7. Remove the record of employee with ID=106 from the table

DELETE FROM Emp WHERE Id = 106;

-- 8. Use DISTINCT to display unique salary values

SELECT DISTINCT Salary FROM Emp;

-- 9. Display the minimum and maximum salary of an employee

SELECT MIN(Salary) AS MinSalary, MAX(Salary) AS MaxSalary FROM Emp;

-- 10. Display the total number of employees

SELECT COUNT(\*) AS TotalEmployees FROM Emp;

-- 11. Display the sum of salaries of employees in department 10

SELECT SUM(Salary) AS TotalSalaryDept10 FROM Emp WHERE Dept = 10;

**chit no 21 :Consider the following Relational Database. student (roll\_no, name,city,marks,c\_no) Course (c\_no,cname,fees) Construct Queries into Relational algebra.**

**1. Write a query in SQL to insert single record into table.**

**2. Write a query in SQL to multiple rows into table.**

**3. Display all the records from table.**

**4. List Student Details enrolled for ‘BBA (C.A)’ Course.**

**5. List the Course having fees < 20000**

**6. Display all students living in either ‘Nasik’ or ‘Pune’ city.**

**7. Display Course detail for student ‘Gaurav Sharma’.**

INSERT INTO student VALUES (101, 'John Doe', 'Pune', 85, 1);

INSERT INTO student VALUES (102, 'Jane Smith', 'Nasik', 90, 2),

(103, 'Bob Johnson', 'Mumbai', 78, 1),

(104, 'Alice Brown', 'Pune', 95, 2);

SELECT \* FROM student;

SELECT \* FROM student WHERE c\_no = (SELECT c\_no FROM Course WHERE cname = 'BBA (C.A)');

SELECT \* FROM Course WHERE fees < 20000;

SELECT \* FROM student WHERE city IN ('Nasik', 'Pune');

SELECT \* FROM Course WHERE c\_no = (SELECT c\_no FROM student WHERE name = 'Gaurav Sharma');

**Chit 15 : Write at least 10 SQL queries on the suitable database application using SQL DML statements. Insert a new employee with the following details: EmployeeID, FirstName, LastName,age, DepartmentID, and Salary.**

**1. Update the salary of all employees in the Sales department by 10%.**

**2. Delete all orders placed before a certain date.**

**3. Retrieve the names and ages of all customers who have made a purchase in the last month.**

**4. Increase the price of all products in the "Electronics" category by 5%.**

**5. Find the total number of products in each category.**

**6. Assign a new manager to a specific department.**

**7. Transfer an employee from one department to another.**

**8. Mark an order as shipped and update the shipped date.**

**9. Retrieve the highest salary among all employees.**

INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, DepartmentID, Salary)

VALUES (101, 'John', 'Doe', 30, 2, 60000);

UPDATE Employees

SET Salary = Salary \* 1.10

WHERE DepartmentID = (SELECT DepartmentID FROM Departments WHERE DepartmentName = 'Sales');

DELETE FROM Orders

WHERE OrderDate < '2023-01-01';

SELECT CustomerName, Age

FROM Customers

WHERE CustomerID IN (SELECT CustomerID FROM Orders WHERE OrderDate >= DATEADD(MONTH, -1, GETDATE()));

UPDATE Products

SET Price = Price \* 1.05

WHERE Category = 'Electronics';

SELECT Category, COUNT(\*) AS TotalProducts

FROM Products

GROUP BY Category;

UPDATE Departments

SET ManagerID = 105

WHERE DepartmentID = 3;

UPDATE Employees

SET DepartmentID = 4

WHERE EmployeeID = 102;

UPDATE Orders

SET Shipped = 1, ShippedDate = GETDATE()

WHERE OrderID = 1001;

SELECT MAX(Salary) AS HighestSalary

FROM Employees;

**Chit No. 14 : Design and Develop SQLDDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc. Employee (empno,name,office,age) Books(isbn,title,authors,publisher) Loan(empno, isbn,date) Write the following queries in relational algebra.**

**Q. Create a view of table employee with same structure and all the records.**

**Q. Create a new view having specific record but all the fields from existing table.**

**Q. Create Index for a table book on column name title.**

**Q, Create Index for a table book on column name isbn, publisher.**

**Q. Create a synonym named ‘EMP\_INFO for the ‘EMPLOYEE’ table.**

CREATE VIEW Employee\_View AS

SELECT \* FROM Employee;

CREATE VIEW Specific\_Employee\_View AS

SELECT empno, name

FROM Employee

WHERE age > 25;

CREATE INDEX idx\_books\_title ON Books(title);

CREATE INDEX idx\_books\_isbn\_publisher ON Books(isbn, publisher);

CREATE SYNONYM EMP\_INFO FOR Employee;

**Chit No. 6 : Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.**

**Write a Stored Procedure namely proc\_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks899and 825 category is Higher Second Class.**

**Write a PL/SQLblock to use procedure created with above requirement. Stud\_Marks(name, total\_marks) Result(Roll,Name, Class)**

create table stud\_marks(name varchar(20),total\_marks number(5));

create table results(roll number(3),name varchar(20),class varchar(20));

SQL>create or replace procedure proc\_grades(rno number,name varchar,marks number)is

2 class varchar(20);

3

4 begin

5 if(marks<=1500 and marks>=990)then

6 class:='distinction';

7 elsif(marks<=989 and marks>=900)then

8 class:='first';

9 elsif(marks<=899 and marks>=825)then

10 class:='higher second';

11 end if;

12 insert into stud\_marks values(name,marks);

13 insert into results values(roll,name,class);

14 end;

15 /

SQL> exec proc\_grades(1,'ram',1100);

SQL> exec proc\_grades(2,'shyam',967);

SQL> exec proc\_grades(3,'rohan',865);

SQL> select \* from stud\_marks;

SQL>select \* from results;

**Chit No. 5 : Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.**

SQL>create table areas ( r number(2), area number (14,2));

SQL>declare

r number(5);

area number(14,2);

pi constant number (4,2):=3.14;

begin

r:=3;  
 while r<=7

loop

area:=pi\*power(r,2);

insert into areas values(r,area );

r:=r+1;

end loop;

end;

/

SQL>select \* from areas;

**Chit No. 8 : Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).**

**Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library\_Audit table.**

create table library(bno number(5),bname varchar(20),author varchar(20),

allowed\_days number(5));

insert into library values (1,'java','Mr.Patil',10);

insert into library values (2,'UB','Mr.Sharama',15);

insert into library values (3,'cpp','Mr.Surve',15);

create table library\_audit(bno number(5),o\_days number(5),n\_days number(5));

create or replace trigger tr1

before update or delete on library

for each row

begin

insert into library\_audit values(:new.bno,:old.allowed\_days,:new.allowed\_days);

end;

/

select \* from library;

update library set allowed\_days=15 where bno=1;

select \* from library;

select \* from library\_audit;

delete from library where bno=1;

select \* from library\_audit;

**Chit No. 7 : Cursors:(All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)**

**Write a PL/SQL block of code using parameterized Cursor that will merge the data available in the newly created table N\_Roll Call with the data available in the table O\_RollCall. If the data in the first table already exist in the second table then that data should be skipped.**