Assignment No 3

Q . Consider Following Schema
Employee (Employee_id, First_name, last_name , hire_date, salary, Job_title, manager_id, department_id)
Departments(Department_id, Department_name, Manager_id, Location_id)
Locations(location_id ,street_address ,postal_code, city, state, country_id)

1. Write a query to find the names (first_name, last_name) and the salaries of the employees who have a higher salary than the employee whose last_name=''Singh".

```
SELECT first_name, last_name, salary
FROM Employee
WHERE salary > (SELECT salary FROM Employee WHERE last_name = 'Singh');
```

2. Write a query to find the names (first_name, last_name) of the employees who have a manager and work for a department based in the United States.

```
SELECT e.first_name, e.last_name
FROM Employee e
INNER JOIN Departments d ON e.department_id = d.department_id
INNER JOIN Locations l ON d.location_id = l.location_id
WHERE e.manager_id IS NOT NULL AND l.country_id = 'US';
```

3. Find the names of all employees who works in the IT department.

```
SELECT first_name, last_name
FROM Employee
WHERE department_id IN (SELECT department_id FROM Departments WHERE department_name = 'IT');
```

4. Write a query to find the names (first_name, last_name), the salary of the employees whose salary is greater than the average salary.

```
SELECT first_name, last_name, salary
FROM Employee
WHERE salary > (SELECT AVG(salary) FROM Employee);
```

5. Write a query to find the names (first_name, last_name), the salary of the employees who earn more than the average salary and who works in any of the IT departments.

```
SELECT e.first_name, e.last_name, e.salary
FROM Employee e
INNER JOIN Departments d ON e.department_id = d.department_id
WHERE e.salary > (SELECT AVG(salary) FROM Employee) AND d.department_name = 'IT';
```

6. Write a query to find the names (first_name, last_name), the salary of the employees who earn the same salary as the minimum salary for all departments.

```
SELECT e.first_name, e.last_name, e.salary
FROM Employee e
WHERE e.salary = (SELECT MIN(salary) FROM Employee WHERE department_id = e.department_id);
```

7. Write a query to display the employee ID, first name, last names, salary of all employees whose salary is above average for their departments.

```
SELECT e.employee_id, e.first_name, e.last_name, e.salary
FROM Employee e
WHERE e.salary > (SELECT AVG(salary) FROM Employee WHERE department_id = e.department_id);
```

8. Write a query to find the employee id, name (last_name) along with their manager_id, manager name (last_name).

```
SELECT e.Employee_id, e.last_name AS Employee_last_name, e.manager_id, m.last_name AS
Manager_last_name
FROM Employee e
LEFT JOIN Employee m ON e.manager_id = m.Employee_id;
```

9. Find the names and hire date of the employees who were hired after 'Jones'.

```
SELECT first_name, last_name, hire_date
FROM Employee
WHERE hire_date > (SELECT hire_date FROM Employee WHERE last_name = 'Jones');
```

10. Write a query to get the department n

```
SELECT d.Department_name, COUNT(e.Employee_id) AS Number_of_Employees
FROM Departments d
LEFT JOIN Employee e ON d.Department_id = e.department_id
GROUP BY d.Department_name;
```

Assignment No 4

Q . PROBLEM STATEMENT: Write a PL/SQL block of code for the following requirements:-
Schema:
l. Borrower(RolIin, Name, Dateoflssue, NameofBook, Status)
2. Fine(Roll_no,Date,Amt)

Accept roll_no & name of book from user. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5 per day. If no. of days>30, per day fine will be Rs 5() per day & for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table.

Solution

```
CREATE TABLE Borrower (
        roll INT
        ,NAME varchar2(20)
        ,doi DATE
        ,nob varchar2(20)
        ,STATUS CHAR(1) DEFAULT 'I'
        );
```

```
CREATE TABLE Fine (
        roll INT
        ,Dt DATE
        ,amount INT
        );
```

Creating Tables

PL/SQL Block to issue book:

```
DECLARE roll1 INT;
name1 varchar2(20);
doi1 DATE;
nob1 varchar2(20);
BEGIN
        roll1 : = :roll_No;
        name1 : = :Name_of_student;
        doi1 : = :Date_of_Issue_mm_dd_yyyy;
        nob1 : = :Name_of_Book;
        INSERT INTO Borrower (
                roll
                ,NAME
                ,doi
                ,nob
                )
        VALUES (
                roll1
                ,name1
                ,doi1
                ,nob1
                );
        dbms_output.put_line('Book isuued');
END
```

PL/SQL Block to return code:

```
DECLARE roll1 INT;

doi1 DATE;

nob1 varchar2(20);

days_passed INT;

fine INT;

BEGIN

	roll1 : = :rollNo;

	SELECT doi , nob ,  doi1 ,nob1
		FROM Borrower
		WHERE roll = roll1;

	days_passed : = SYSDATE - doi1;

	dbms_output.put_line(days_passed);

	IF (days_passed > 30) THEN dbms_output.put_line('per day fine 50');
		fine : = days_passed * 50;

		ELSIF(days_passed > 15) THEN dbms_output.put_line('per day fine 5');

		fine : = days_passed * 5;ELSE

		dbms_output.put_line('No fine');

		fine : = days_passed * 0;
	ENDIF ;
	INSERT INTO fine
	VALUES (
		roll1
		,SYSDATE
		,fine
		);

	UPDATE borrower  SET STATUS = 'R' WHERE roll = roll1 AND STATUS = 'I'
	dbms_output.put_line(days_passed || nob1);
END
```

Assignment No 2 (a)

1. Create following tables using given schema and insert appropriate data into these tables.
Student(StudID, Name, Address, Marks)
Employee(EmplyeeID, Name, Address, Salary, DateOfJoining ,Department)
Weather(CityID, CityName, MinTemp, MaxTemp)

```
CREATE TABLE Student ( StudID INT,     Name VARCHAR(50), Address VARCHAR(100), Marks FLOAT );

CREATE TABLE Employee (    EmployeeID INT, Name VARCHAR(50),    Address VARCHAR(100),    Salary
DECIMAL(10, 2),    DateOfJoining DATE,   Department VARCHAR(50) );

CREATE TABLE WeatherData (    CityID INT,    MinTemp FLOAT,    MaxTemp FLOAT );

-- Insert data into Student table
INSERT INTO Student (StudID, Name, Address, Marks)
VALUES (1, 'John Doe', '123 Main St', 95.5),       (2, 'Jane Smith', '456 Elm St', 88.0);

-- Insert data into Employee table
INSERT INTO Employee (EmployeeID, Name, Address, Salary, DateOfJoining, Department)
VALUES (101, 'Alice Johnson', '789 Oak St', 60000.00, '2022-01-15', 'HR'),
      (102, 'Bob Williams', '101 Pine St', 75000.00, '2021-05-10', 'IT');

-- Insert data into WeatherData table
INSERT INTO WeatherData (CityID, MinTemp, MaxTemp)
VALUES (1, 15.5, 28.0),   (2, 20.0, 32.5);
```

2. Alter Student and Employee table to add Not Null constraint on all columns.

```
-- Alter Student table
ALTER TABLE StudentALTER COLUMN StudID INT NOT NULL,ALTER COLUMN Name VARCHAR(50) NOT
NULL,ALTER COLUMN Address VARCHAR(100) NOT NULL,ALTER COLUMN Marks FLOAT NOT NULL;

-- Alter Employee table
ALTER TABLE EmployeeALTER COLUMN EmployeeID INT NOT NULL,ALTER COLUMN Name VARCHAR(50)
NOT NULL,ALTER COLUMN Address VARCHAR(100) NOT NULL,ALTER COLUMN Salary DECIMAL(10, 2)
NOT NULL,ALTER COLUMN DateOfJoining DATE NOT NULL,ALTER COLUMN Department VARCHAR(50)
NOT NULL:
```

3. Alter the Student table to add Primary key constraint on StudID column.

```
-- Add primary key constraint to Student table
ALTER TABLE Student
ADD CONSTRAINT PK  Student PRIMARY KEY (StudID);
```

4. Create a view JoiningInfo on Employee table displaying Employee ID, Name and DateOfJoining of employees.

```
-- Create the view
CREATE VIEW JoiningInfo AS
SELECT EmployeeID, Name, DateOfJoining
FROM Employee;
```

5. Crete index on primary key columns of all the tables.

```
-- Create indexes on primary key columns
CREATE INDEX IDX_Student_StudID ON Student (StudID);
CREATE INDEX IDX_Employee_EmployeeID ON Employee (EmployeeID);
CREATE INDEX IDX_WeatherData_CityID ON WeatherData (CityID);
```

6. Crate view MarksInfo on Student table displaying StuID and Marks.

```
-- Create the view
CREATE VIEW MarksInfo AS
SELECT StudID, Marks
FROM Student;
```

7. Change the name of Weather table to WeatherData.

```
-- Rename the table
EXEC sp_rename 'Weather', 'WeatherData';
```

8. Drop column CityName from WeatherData table.

```
-- Drop the column
ALTER TABLE WeatherData
DROP COLUMN CityName;
```

9. Add column Grade to Student table.

```
-- Add the column
ALTER TABLE Student
ADD Grade VARCHAR(2);
```

10. Crate a view “DistinctionStudents” on student table displaying data of students having Distinction as Grade.

```
-- Create the view
CREATE VIEW DistinctionStudents AS
SELECT *
FROM Student
WHERE Grade = 'Distinction';
```

11. Create a sequence on StudID in student table.

```
-- Create a sequence
CREATE SEQUENCE Student_StudID_Seq AS INT START WITH 1 INCREMENT BY 1;
```

12. Create a synonym ‘Emp_Info’ for Employee table.

```
-- Create the synonym
CREATE SYNONYM Emp_Info FOR Employee;
```

Assignment No 2 (b)

Create the Employee table using following schema

Employee (Employee_id, First_name, Last_name, Salary, Joining_date, Department)

```
CREATE TABLE Employee (
   Employee_id INT PRIMARY KEY,    First_name VARCHAR(50),    Last_name VARCHAR(50),
   Salary DECIMAL(10, 2),    Joining_date DATE,    Department VARCHAR(50)
);
```

1. Insert 10 to 15 appropriate records in the Employee table.

```
INSERT INTO Employee (Employee_id, First_name, Last_name, Salary, Joining_date, Department)
VALUES
   (001, 'Abhi', 'Fadake', 25000, '2022-01-15', 'HR'),    (002, 'Priya', 'Sharma', 28000, '2021-03-20', 'IT'),
   (003, 'Amit', 'Singh', 22000, '2023-05-10', 'Finance'),   (004, 'Sneha', 'Verma', 19000, '2022-09-05', 'HR'),
   (005, 'Neha', 'Gupta', 30000, '2022-11-08', 'IT'), (006, 'Sanjay', 'Yadav', 26000, '2023-02-18', 'Finance'),
   (007, 'Preeti', 'Patel', 24000, '2021-07-30', 'HR'),(008, 'Vikas', 'Mishra', 28000, '2022-03-14', 'IT'),
   (009, 'Anita', 'Jain', 21000, '2021-12-25', 'Finance'), (010, 'Rahul', 'Shukla', 32000, '2023-04-02', 'IT'),
   (011, 'Shreya', 'Sharma', 29000, '2022-08-12', 'HR'),  (012, 'Om', 'Kale', 23000, '2021-10-09', 'Finance');
```

2. Get First_Name,Last_Name from employee table

```
SELECT First_name, Last_name FROM Employee;
```

3. Get unique DEPARTMENT from employee table

```
SELECT DISTINCT Department FROM Employee;
```

4. Get FIRST_NAME ,Joining year,Joining Month and Joining Date from employee table
Select FIRST_NAME, year(joining_date),month(joining_date), DAY(joining_date) from EMPLOYEE

```
SELECT First_name, YEAR(Joining_date) AS Joining_Year, MONTH(Joining_date) AS Joining_Month,
DAY(Joining_date) AS Joining_Date
FROM Employee;
```

5. Get all employee details from the employee table order by Salary Ascending

```
SELECT * FROM Employee ORDER BY Salary ASC;
```

6. Get all employee details from the employee table whose First_Name starts with A.

```
SELECT * FROM Employee WHERE First_name LIKE 'A%';
```

7. Update the Salary column by incrementing salary of all employees having salary less than 20000 by 5000.

```
UPDATE Employee  SET Salary = Salary + 5000  WHERE Salary < 20000;
```

8. Delete the department of employee no 004.

```
DELETE FROM Employee WHERE Employee_id = 004;
```

9. Find department wise minimum salary.

```
SELECT Department, MIN(Salary) AS Minimum_Salary
FROM Employee
GROUP BY Department;
```

10. Find department wise Average salary in ascending order.

```
SELECT Department, AVG(Salary) AS Average_Salary
FROM Employee
GROUP BY Department
ORDER BY Average_Salary ASC;
```

Consider Following Schema

Employee(employee_id, employee_name, City, Company_Name, Salary)

```
CREATE TABLE Employee (
   employee_id INT PRIMARY KEY,    employee_name VARCHAR(255),    City VARCHAR(255),
   Company_Name VARCHAR(255),    Salary DECIMAL(10, 2)  );
```

11. Find details of all employees who work for company “IBM” and live in city “Pune”.

```
SELECT *
FROM Employee
WHERE Company_Name = 'IBM' AND City = 'Pune';
```

12. Find names, and cities of all employees who work for “Infosys” or earn more than 30000.

```
SELECT employee_name, City
FROM Employee
WHERE Company_Name = 'Infosys' OR Salary > 30000;
```

13. Find all employees who are employees of “IBM” and not living in city “Mumbai”.

```
SELECT *
FROM Employee
WHERE Company_Name = 'IBM' AND City <> 'Mumbai';
```

14. Find company wise maximum salary.

```
SELECT Company_Name, MAX(Salary) AS Max_Salary
FROM Employee
GROUP BY Company_Name;
```

15. Find those companies whose employees earn higher salary, than average salary at “IBM”.

```
SELECT DISTINCT E1.Company_Name
FROM Employee E1
WHERE E1.Salary > (
   SELECT AVG(Salary)
   FROM Employee
   WHERE Company_Name = 'IBM'
);
```