SQL Documentation

For execute this project run all the SQL command in oracle 10g or other version's

TABLE OF CONTENT:

System Summary:

Our project is Diagnostic Center Management System. Here, we create a desktop-based application using C# (.NET framework) and ORACLE SQL.

Employee

1.Log in/Registration

2.Printout Bill

3.Add, search, remove and update data of patients

Admin

1.Log in/Registration

2.Check Data of Employees

3. Add, update, search and remove Employees

4.Check all the records of patients

Table Creation and Inserting Data:

Admin Table :

```
CREATE TABLE admin (
   userID INT NOT NULL,
   password VARCHAR2(4) NOT NULL,
   PRIMARY KEY (userID)
);

INSERT INTO admin VALUES (101, 'adm1');
INSERT INTO admin VALUES (102, 'adm2');
INSERT INTO admin VALUES (103, 'adm3');
INSERT INTO admin VALUES (104, 'adm4');
INSERT INTO admin VALUES (105, 'adm5');
```

employee Table:

```
CREATE TABLE employee (
   EMP_ID INT PRIMARY KEY,
   EMP_NAME VARCHAR2(25) NOT NULL,
   PASSWORD VARCHAR2(4) NOT NULL,
   MOB_NO INT NOT NULL,
   EMAIL VARCHAR2(30) NOT NULL,
   ADDRESS VARCHAR2(35) NOT NULL,
   JOIN_DATE DATE NOT NULL,
   EMP_SALARY INT NOT NULL,
   EMP_TYPE VARCHAR2(20) NOT NULL
);



INSERT INTO employee VALUES(100, 'John Doe', 'emp1', 01813775261,
'john.doe@example.com',
'123 Main Street', TO_DATE('2023-01-01', 'YYYY-MM-DD'), 5000, 'Physician');

INSERT INTO employee VALUES(101, 'Jane Smith', 'emp2', 01965458978,
'jane.smith@example.com',
'456 Oak Avenue', TO_DATE('2023-01-15', 'YYYY-MM-DD'), 6000, 'Laboratorist');
```

Patient Table:

```
Create table patient(
    PATIENTID INT PRIMARY KEY,
    PATIENTNAME VARCHAR2(25) NOT NULL,
    REFERENCE VARCHAR2(4) NOT NULL,
    GENDER INT NOT NULL,
   ADDRESS VARCHAR2(30) NOT NULL,
   AGE VARCHAR2(6) NOT NULL,
   PHONENO INT NOT NULL,
   TEST VARCHAR2(15) NOT NULL,
   TOTAL INT NOT NULL,
   DISCOUNT INT NOT NULL,
   NETTOTAL INT NOT NULL,
   TESTDATE DATE NOT NULL
);
```

1.Procedure: insert_admin

```
CREATE OR REPLACE PROCEDURE insert_admin(
  id   admin.userid%type,
  pass admin.password%type
) AS
BEGIN
  INSERT INTO admin (userid, password) VALUES (id, pass);
END insert_admin;
```

```
BEGIN
  insert_admin(106, 'adm6');
END;
```

2.Procedure: insert_employee

```
CREATE OR REPLACE PROCEDURE insert_employee(
  id   employee.EMP_ID%type,
  name employee.EMP_NAME%type,
  pass employee.PASSWORD%type,
  mobNo employee.MOB_NO%type,
  email employee.EMAIL%type,
  address employee.ADDRESS%type,
  joindate employee.JOIN_DATE%type,
  salary   employee.EMP_SALARY%type,
  type     employee.EMP_TYPE%type

) AS
BEGIN
  INSERT INTO employee
(EMP_ID,EMP_NAME,PASSWORD,MOB_NO,EMAIL,ADDRESS,JOIN_DATE,EMP_SAL
ARY,EMP_TYPE) VALUES (id,name,pass,mobNo,email,address,joindate,salary,type);
END insert_employee;

BEGIN insert_employee(103, 'Nazmul Islam', 'emp3', 01813775261, 'helloaiub@gmail.com',
'Kuril Dhaka', TO_DATE('15-08-22', 'DD-MM-YY'), 7000,'Physician');END;

BEGIN insert_employee(102, 'Ayon Ghosh','emp3',01785451230,'hellobd@gmail.com','Uttara-
12, Dhaka',TO_DATE('12-10-22', 'DD-MM-YY'),7000,'Laboratorist');END;
```

3.Procedure: insert_patient

```
CREATE OR REPLACE PROCEDURE insert_patient(
  p_PatientID     patient.PatientID%type,
  p_PatientName   patient.PatientName%type,
  p_Reference     patient.Reference%type,
  p_Gender        patient.Gender%type,
  p_Address       patient.Address%type,
  p_Age           patient.Age%type,
  p_PhoneNo       patient.PhoneNo%type,
  p_Test          patient.Test%type,
  p_Total         patient.Total%type,
  p_Discount      patient.Discount%type,
  p_NetTotal      patient.NetTotal%type,
  p_TestDate      patient.TestDate%type
```

```
) AS
BEGIN
  INSERT INTO patient (
    PatientID, PatientName, Reference, Gender, Address, Age, PhoneNo, Test, Total, Discount,
NetTotal, TestDate
  ) VALUES (
    p_PatientID, p_PatientName, p_Reference, p_Gender, p_Address, p_Age, p_PhoneNo,
p_Test, p_Total, p_Discount, p_NetTotal, TO_DATE(p_TestDate, 'DD-MM-YY')
  );
END insert_patient;
```

4.Procedure: update_employee

```
CREATE OR REPLACE PROCEDURE update_employee(
  id   employee.EMP_ID%type,
  name employee.EMP_NAME%type,
  pass employee.PASSWORD%type,
  mobNo employee.MOB_NO%type,
  email employee.EMAIL%type,
  address employee.ADDRESS%type,
  joindate employee.JOIN_DATE%type,
  salary   employee.EMP_SALARY%type,
  type     employee.EMP_TYPE%type
) AS
BEGIN
  UPDATE employee
    SET
       EMP_NAME = name,
       PASSWORD = pass,
       MOB_NO = mobNo,
       EMAIL = email,
       ADDRESS = address,
       JOIN_DATE = joindate,
       EMP_SALARY = salary,
       EMP_TYPE = type
    WHERE EMP_ID = id;
END update_employee;

BEGIN
    update_employee(id => 101, name => 'Nazmul Islam', pass => 'emp3',mobNo =>
1813775261,email => 'helloaiub@gmail.com',address => 'Kuril Dhaka',joindate =>
TO_DATE('15-08-22', 'DD-MM-YY'),salary => 7000,
       type => 'Physician');
END;
```

5.Procedure: update_patient

```
CREATE OR REPLACE PROCEDURE update_patient(
  id   patient.PATIENTID%type,

 name    patient.PATIENTNAME%type,
  ref     patient.REFERENCE%type,
  gender  patient.GENDER%type,
  address patient.ADDRESS%type,
  age     patient.AGE%type,
  phone   patient.PHONENO%type,
  tst    patient.TEST%type,
  tot    patient.TOTAL%type,
  dis    patient.DISCOUNT%type,
  nt     patient.NETTOTAL%type,
  td    patient.TESTDATE%type
) AS
BEGIN
  UPDATE patient
    SET
        PATIENTID = id,
        PATIENTNAME = name,
        REFERENCE = ref,
        GENDER = gender,
        ADDRESS =address ,
        AGE = age,
        PHONENO = phone,
        TEST = tst,
        TOTAL=tot,
        DISCOUNT=dis,
        NETTOTAL=nt,
        TESTDATE=td
    WHERE PATIENTID = id;
END update_patient;
BEGIN update_patient(
    id => 1, name => 'Nazmul Islam',ref => 'some_reference',gender => 'Male',address => 'Kuril
Dhaka',age => 30,phone => 1813775441,tst => 'MRI',tot => 4500,dis => 100,nt => 900,td =>
TO_DATE('15-08-22', 'DD-MM-YY'));END;
```

6.Procedure: delete_employee

```
CREATE OR REPLACE PROCEDURE delete_employee(
  id  employee.EMP_ID%TYPE
) AS
BEGIN
  DELETE FROM employee
  WHERE EMP_ID = id;

END delete_employee;

BEGIN delete_employee(id => 100);  END;
```

7.Procedure: delete_patient

```
CREATE OR REPLACE PROCEDURE delete_patient (
  id  patient . patientid%TYPE
) AS
BEGIN
  DELETE FROM patient
  WHERE patientid= id;

END delete_patient;

BEGIN delete_employee(id => 1);  END;
```

View

1.View: max_sal_view

```
CREATE OR REPLACE VIEW max_sal_view AS select * from employee where
EMP_SALARY=(select max(EMP_SALARY) from employee) ;

SELECT * FROM max_sal_view
```

2.View: min_sal_view

```
CREATE OR REPLACE VIEW min_sal_view AS select * from employee where
EMP_SALARY=(select min(EMP_SALARY) from employee) ;

SELECT * FROM min_sal_view
```

3.View: view_total_avg_emp

```
CREATE OR REPLACE VIEW view_total_avg_emp AS select sum(EMP_SALARY) as total_salary ,avg(EMP_SALARY) as avarage_salary,count(EMP_ID) as Total_employee from employee
SELECT * FROM view_total_avg_emp
```

4.View: senior_view

```
 CREATE OR REPLACE VIEW senior_view AS select * from employee where JOIN_DATE=(select min(JOIN_DATE) from employee)

SELECT * FROM senior_view
```

5.View: junior_view

```
CREATE OR REPLACE VIEW junior_view AS select * from employee where JOIN_DATE=(select max(JOIN_DATE) from employee)

SELECT * FROM junior_view
```

Triggers

1.Trigger: check_null_insert_employee

```
CREATE OR REPLACE TRIGGER check_null_insert_employee
BEFORE INSERT ON employee
FOR EACH ROW
BEGIN
  IF (
    :NEW.EMP_ID IS NULL OR
    :NEW.EMP_NAME IS NULL OR
    :NEW.PASSWORD IS NULL OR
    :NEW.MOB_NO IS NULL OR
    :NEW.EMAIL IS NULL OR
    :NEW.ADDRESS IS NULL OR
    :NEW.JOIN_DATE IS NULL OR
    :NEW.EMP_SALARY IS NULL OR
    :NEW.EMP_TYPE IS NULL
  ) THEN
    RAISE_APPLICATION_ERROR(-20002, 'Inserting NULL values is not allowed for any
```

```
column.');
  END IF;
END check_null_insert_employee;

select * from employee
```

2.Trigger: check_null_insert_patient

```
CREATE OR REPLACE TRIGGER check_null_insert_patient
BEFORE INSERT ON patient
FOR EACH ROW
BEGIN
  IF :NEW.PatientID IS NULL OR
     :NEW.PatientName IS NULL OR
     :NEW.Reference IS NULL OR
     :NEW.Gender IS NULL OR
     :NEW.Address IS NULL OR
     :NEW.Age IS NULL OR
     :NEW.PhoneNo IS NULL OR
     :NEW.Test IS NULL OR
     :NEW.Total IS NULL OR
     :NEW.Discount IS NULL OR
     :NEW.NetTotal IS NULL OR
     :NEW.TestDate IS NULL
  THEN
    -- Raise an exception if any NULL value is found
    RAISE_APPLICATION_ERROR(-20002, 'Inserting NULL values is not allowed for any
column.');
  END IF;
END check_null_insert_patient;

select * from patient
```

3.Trigger:

```
CREATE OR REPLACE TRIGGER count_check_after_update
AFTER UPDATE ON patient
FOR EACH ROW
DECLARE
  v_row_count_previous NUMBER;
  v_row_count_current NUMBER;
BEGIN
```

```
  SELECT COUNT(*) INTO v_row_count_previous FROM employee;

  SELECT COUNT(*) INTO v_row_count_current FROM employee;

  IF v_row_count_current = v_row_count_previous THEN
    DBMS_OUTPUT.PUT_LINE('Current count plus one is greater than the previous count after
update.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Current count plus one is not greater than the previous count
after update.');
  END IF;
END count_check_after_update;

BEGIN
count_check_after_update;
END;
```

Functions

1.Function: calculate_total_income

```
CREATE OR REPLACE FUNCTION calculate_total_income RETURN NUMBER AS
  total_income NUMBER:= 0;
BEGIN
  SELECT SUM(NETTOTAL) INTO total_income
  FROM patient;

  RETURN total_income;
END calculate_total_income;

calculate_total_income

BEGIN

  DBMS_OUTPUT.PUT_LINE(calculate_total_income());
END;
```

1.Function: calculate_initial_total

```
CREATE OR REPLACE FUNCTION calculate_initial_total RETURN NUMBER AS
  initial_total NUMBER := 0;
BEGIN
  SELECT SUM(TOTAL) INTO initial_total
  FROM patient;

  RETURN  initial_total;
END calculate_initial_total;


BEGIN
  DBMS_OUTPUT.PUT_LINE(calculate_initial_total());
END;
SELECT calculate_initial_total() as initial_total FROM DUAL
```

2.Function: calculate_ total_income

```
CREATE OR REPLACE FUNCTION calculate_total_income RETURN NUMBER AS
  total_income NUMBER := 0;
BEGIN
  SELECT SUM(NETTOTAL) INTO total_income
  FROM patient;

  RETURN total_income;
END calculate_total_income;

calculate_total_income

BEGIN
  DBMS_OUTPUT.PUT_LINE(calculate_total_income());
END;
SELECT calculate_total_income() as Total_income  FROM DUAL
```

3.Function: calculate_total_patientNo

```
CREATE OR REPLACE FUNCTION calculate_total_patientNo RETURN NUMBER AS
  total_patient NUMBER := 0;
BEGIN
  SELECT COUNT(*) INTO total_patient
  FROM patient;

  RETURN total_patient;
END calculate_total_patientNo;


BEGIN
  DBMS_OUTPUT.PUT_LINE(calculate_total_patientNo());
END;
SELECT calculate_total_patientNo() as Total_Patient_No  FROM DUAL
```

4.Function: calculate_total_discount

```
CREATE OR REPLACE FUNCTION calculate_total_discount RETURN NUMBER AS
  total_discount NUMBER := 0;
BEGIN
  SELECT SUM(DISCOUNT) INTO  total_discount
  FROM patient;

  RETURN  total_discount;
END calculate_total_discount;


BEGIN
  DBMS_OUTPUT.PUT_LINE(calculate_total_discount());
END;
SELECT calculate_total_discount() as Total_Discount FROM DUAL
```

5.Function: avg_initial_total

```
CREATE OR REPLACE FUNCTION avg_initial_total RETURN NUMBER AS
  total NUMBER := 0;
  totalNo NUMBER := 0;
BEGIN
  SELECT SUM(TOTAL) INTO total
  FROM patient;
```

```
  SELECT COUNT(*) INTO totalNo
  FROM patient;

  RETURN  ROUND(total / totalNo, 2);
END avg_initial_total;


BEGIN
  DBMS_OUTPUT.PUT_LINE(avg_initial_total());
END;
SELECT avg_initial_total() as Avarage_initial_total FROM DUAL
```

6.Function: avg_net_total

```
CREATE OR REPLACE FUNCTION avg_net_total RETURN NUMBER AS
  totalNet NUMBER := 0;
  totalNo NUMBER := 0;
BEGIN
  SELECT SUM(NETTOTAL) INTO totalNet
  FROM patient;

  SELECT COUNT(*) INTO totalNo
  FROM patient;

  RETURN  ROUND(totalNet / totalNo, 2);
END avg_net_total;
BEGIN
  DBMS_OUTPUT.PUT_LINE(avg_net_total);
END;
SELECT avg_net_total() as avg_net_total FROM DUAL
```

----------End----------