DBMS_CODES

Assignment 2a

```
CREATE TABLE student_info(
      rollno INT PRIMARY KEY,
      name VARCHAR(255) NOT NULL,
      email VARCHAR(255)
);

desc student_info;

CREATE VIEW view_student_info as (
    select rollno from student_info
);

desc view_student_info;

-- like this index is created
CREATE INDEX ind_name ON student_info(name);

-- like this we can delete index
DROP INDEX ind_name;


 CREATE TABLE info(
     id INT PRIMARY KEY,
     rollno INT NOT NULL,
     name VARCHAR(255)
);

---- sequence
CREATE TABLE info_main(
     id_seq INT PRIMARY KEY,
     rollno INT NOT NULL,
     name VARCHAR(255)
);


CREATE SEQUENCE id_seq
    INCREMENT BY 1
    START WITH 1
    MINVALUE 1
    MAXVALUE 100
    CYCLE
    CACHE 2;



INSERT INTO info_main (id_seq,rollno, name) VALUES (id_seq.NEXTVAL,'081','mihir');
INSERT INTO info_main (id_seq,rollno, name) VALUES (id_seq.NEXTVAL,'082','megha');
INSERT INTO info_main (id_seq,rollno, name) VALUES (id_seq.NEXTVAL,'083','sanket');

select * from info_main;


-- synonym is replica of the StudentInfo table
CREATE synonym student for student_info;

desc student;


SELECT * FROM info;
```

Assignment 2b

```
CREATE TABLE Stu_info (
id INT PRIMARY KEY,
rollno INT NOT NULL,
name varchar(255),
```

```
email varchar(255)
);

INSERT INTO Stu_info (id,rollno,name,email) VALUES ('1','81','mihir','mihir@gmail.com');
INSERT INTO Stu_info (id,rollno,name,email) VALUES ('2','82','megha','megha@gmail.com');
INSERT INTO Stu_info (id,rollno,name,email) VALUES ('3','22','Garry','garry@gmail.com');
INSERT INTO Stu_info (id,rollno,name,email) VALUES ('4','98','madhur','madhur@gmail.com');
INSERT INTO Stu_info (id,rollno,name,email) VALUES ('5','95','mohit','mohit@gmail.com');

SELECT * FROM Stu_info ORDER BY id ASC;
SELECT name,email FROM Stu_info WHERE rollno='82';
SELECT * FROM Stu_info WHERE email='garry@gmail.com' AND  rollno='22'


desc Stu_info;


UPDATE Stu_info
  SET name = 'Lokesh',email = 'lokesh@gmail.com'
    WHERE id = 4;

DELETE FROM Stu_info WHERE name= 'megha' OR rollno='82';
DELETE FROM Stu_info WHERE name='Garry' AND rollno='22';

UPDATE Stu_info
     SET name = 'sanket',email = 'sanket@gmail.com'
     WHERE id = 1 AND rollno= '81';
```

Assignment 3

```
-- SQL Queries – all types of Join, Sub-Query and
-- View:
-- Design at least 10 SQL queries for suitable database
-- application using SQL DML statements: all types of
-- Join,
-- Sub-Query and View.


CREATE TABLE Customer_table(
id INT PRIMARY KEY,
name VARCHAR(25) NOT NULL,
salary float NOT NULL
);

INSERT INTO Customer_table (id,name,salary) VALUES (1,'Mohit',500000);
INSERT INTO Customer_table (id,name,salary) VALUES (2,'Ajay',250000);
INSERT INTO Customer_table (id,name,salary) VALUES (3,'Rutuja',350000);
INSERT INTO Customer_table (id,name,salary) VALUES (4,'Hemanshu',353250);

select * from Customer_table ORDER BY id ASC;

CREATE TABLE Orders_table(
     order_id INT PRIMARY KEY,
     Customer_table_id INT REFERENCES Customer_table,
     amount INT NOT NULL
);

desc Orders_table;

INSERT INTO Orders_table VALUES(1,2,200);
INSERT INTO Orders_table VALUES (2,2,1200);
INSERT INTO Orders_table VALUES (3,3,2300);
INSERT INTO Orders_table VALUES (4,4,2100);
INSERT INTO Orders_table VALUES(5,1,100);

SELECT * FROM Orders_table ORDER BY order_id ASC;

-- INNER JOIN
SELECT name,salary,amount
FROM Customer_table
INNER JOIN Orders_table
ON Customer_table.id = Orders_table.Customer_table_id;

-- LEFT JOIN
SELECT name,salary,amount
FROM Customer_table
LEFT JOIN Orders_table
ON Customer_table.id = Orders_table.Customer_table_id;

-- RIGHT JOIN
```

```
SELECT name,salary,amount
FROM Customer_table
RIGHT JOIN Orders_table
ON Customer_table.id = Orders_table.Customer_table_id;


-- FULL JOIN
SELECT name,salary,amount
FROM Customer_table
FULL JOIN Orders_table
ON id = Orders_table.Customer_table_id;

--  This are sub querys

SELECT name
FROM Customer_table
WHERE id
IN (SELECT Customer_table_id FROM Orders_table);

UPDATE Customer_table
SET salary=salary+1000
WHERE id
IN (SELECT Customer_table_id FROM Orders_table);

SELECT * FROM Customer_table;
```

Assignment 4

```
--  exception added
CREATE TABLE  BORROWER
(
roll_no NUMBER,
name VARCHAR2(25),
dateofissue DATE,
name_of_book VARCHAR2(25),
status VARCHAR2(20)
);

CREATE TABLE FINE
(
roll_no NUMBER,
date_of_return DATE,
amt NUMBER
);

INSERT INTO borrower VALUES(54,'SUDARSHAN',TO_DATE('01-10-2022','DD-MM-YYYY'),'HARRY POTTER','I');

INSERT INTO borrower VALUES(56,'SUMIT',TO_DATE('15-10-2022','DD-MM-YYYY'),'DARK MATTER','I');

INSERT INTO borrower VALUES(68,'MANDAR',TO_DATE('24-09-2022','DD-MM-YYYY'),'SILENT HILL','I');

INSERT INTO borrower VALUES(66,'SIDDHAM',TO_DATE('26-08-2022','DD-MM-YYYY'),'GOD OF WAR','I');

INSERT INTO borrower VALUES(50,'SHREYAS',TO_DATE('09-09-2022','DD-MM-YYYY'),'SPIDER-MAN','I');

INSERT INTO borrower VALUES(51,'SHREYA',TO_DATE('09-12-2022','DD-MM-YYYY'),'SPIDER','I');


DECLARE
  i_roll_no NUMBER;
  name_of_book VARCHAR2(25);
  no_of_days NUMBER;
  return_date DATE := TO_DATE(SYSDATE,'DD-MM-YYYY');
  temp NUMBER;
  doi DATE;   -- this is date of issue
  fine NUMBER:=0;
    NEG_DAYS exception;

BEGIN

  i_roll_no := :i_roll_no;  --  this is user input user will put roll no and the name of book
  name_of_book := :name_of_book;

    -- this is query to find date of issue of certain book
  SELECT to_date(borrower.dateofissue,'DD-MM-YYYY') INTO doi
        FROM borrower
        WHERE borrower.roll_no = i_roll_no
        AND borrower.name_of_book = name_of_book;

-- this is to find no of days between return date and date of issue
```

```
        no_of_days := return_date-doi;

-- this is to check if no of days is negative
  IF (no_of_days<0) THEN
        raise NEG_DAYS;
        END IF;
        dbms_output.put_line(no_of_days);
  IF (no_of_days >15 AND no_of_days <=30) THEN
    fine := 5*(no_of_days-15);

  ELSIF (no_of_days>30 ) THEN
    temp := no_of_days-30; -- first 30 will charge only 5 rs per day then  50 rupess per day start
    fine := 75 + temp*50;
  END IF;
  dbms_output.put_line(fine);
  IF (fine>0) THEN
        INSERT INTO fine VALUES(i_roll_no,return_date,fine);
        END IF;

        UPDATE borrower SET status = 'R' WHERE borrower.roll_no = i_roll_no;
    -- this is to update the status of book to returned R means RETURN I means ISSUED
EXCEPTION
        WHEN NEG_DAYS THEN
  DBMS_OUTPUT.PUT_LINE('NEGATIVE DAYS NOT EXCEPTED');
        when NO_DATA_FOUND then
             dbms_output.put_line('no_data_found');
        when OTHERS then
             dbms_output.put_line('some_error_found');
END;

SELECT * FROM FINE
SELECT * FROM BORROWER

DROP TABLE FINE
DROP TABLE BORROWER
```

Assignment 5

```
CREATE TABLE Circle(
  radius NUMBER,
  area NUMBER
); -- table to store radius & area

DECLARE
  radius_var NUMBER;
  area_var NUMBER;
  pi NUMBER := 3.14;
BEGIN
  FOR radius_var IN 5 .. 9 LOOP
    area_var := pi*radius_var*radius_var;
    dbms_output.put_line(area_var);
    INSERT INTO Circle VALUES (radius_var,area_var);
  END LOOP;

END;
/

SELECT * FROM Circle;
```

Assignment 6

```
CREATE TABLE STUDENT_MARKS_FINAL
(
FullName VARCHAR2(25),
total_marks NUMBER
);

CREATE TABLE STUDENT_RESULTS
(
roll_number NUMBER ,
FullName VARCHAR2(25),
class VARCHAR2(30)
);
```

```
CREATE OR REPLACE PROCEDURE procedure_1
(roll_no IN NUMBER, FullName IN VARCHAR2 ,marks IN NUMBER)
AS
BEGIN
  IF (marks<=1500 and marks>=990) THEN
    DBMS_OUTPUT.PUT_LINE (roll_no||' - '||FullName||' : DISTINCTION');
        INSERT INTO STUDENT_RESULTS VALUES (roll_no,FullName,'DISTINCTION');
  ELSIF (marks<=989 and marks>=900) THEN
    DBMS_OUTPUT.PUT_LINE (roll_no||' - '||FullName||' : FIRST CLASS');
        INSERT INTO STUDENT_RESULTS VALUES (roll_no,FullName,'FIRST CLASS');
  ELSIF (marks<=899 and marks>825) THEN
    DBMS_OUTPUT.PUT_LINE(roll_no||' - '||FullName||' : HIGHER SECOND CLASS');
        INSERT INTO STUDENT_RESULTS VALUES (roll_no,FullName,'HIGHER SECOND CLASS');
  ELSE
    DBMS_OUTPUT.PUT_LINE (roll_no||' - '||FullName||' : FAIL');
        INSERT INTO STUDENT_RESULTS VALUES (roll_no,FullName,'FAIL');
  END IF;
      INSERT INTO STUDENT_MARKS_FINAL VALUES (FullName,marks);
END procedure_1;

BEGIN
  procedure_1(1,'Garry',80);
  procedure_1(2,'Abbas ',20);
  procedure_1(3,'Sohum ',50);
  procedure_1(4,'Itachi ',70);
END;
```

Assignment 7

```
CREATE TABLE Cust_New
(
Name VARCHAR2(15)
);

INSERT INTO Cust_New VALUES ('ABC');

CREATE TABLE Cust_Old
(
Name VARCHAR2(15)
);

INSERT INTO Cust_Old VALUES ('ABC');
INSERT INTO Cust_Old VALUES ('PQR');
INSERT INTO Cust_Old VALUES ('XYZ');

DECLARE
CURSOR cur1 IS
SELECT Name from Cust_Old;

CURSOR cur2 IS
SELECT Name from Cust_New;

R VARCHAR(15);
C_Name VARCHAR(15);

BEGIN
OPEN cur1;
OPEN cur2;

LOOP
Fetch cur1 into C_Name;
Fetch cur2 into R;

EXIT WHEN cur1%FOUND = FALSE;
IF R <> C_Name THEN
INSERT INTO Cust_New VALUES (C_Name);
END IF;
END LOOP;
CLOSE cur1;
END;


SELECT * FROM Cust_New;
```

Assignment 8

```
-- the values in backup table are inserted only when the update or delete operation happen on Library
CREATE TABLE Library
(
Book_Id NUMBER(5),
Book_Name VARCHAR2(20),
Book_Type VARCHAR2(20),
Issued_By VARCHAR2(20)
);

INSERT INTO Library VALUES (1,'Harry Potter','Fiction','Garry');
INSERT INTO Library VALUES (2,'The Alchemist','Fiction','Abbas');
INSERT INTO Library VALUES (3,'The Monk Who Sold His Ferrari','Fiction','Sohum');
INSERT INTO Library VALUES (4,'The Secret','Fiction','Itachi');


CREATE TABLE Back_UP
(
Book_Id NUMBER(5),
Book_Name VARCHAR2(20),
Book_Type VARCHAR2(20),
Issued_By VARCHAR2(20)
);


CREATE TRIGGER Update_Rec
AFTER UPDATE OR DELETE ON Library
FOR EACH ROW
BEGIN
INSERT INTO Back_UP
(Book_Id, Book_Name, Book_Type, Issued_By)
VALUES
(:old.Book_Id, :old.Book_Name, :old.Book_Type, :old.Issued_By);
END;

-- this will trigger the trigger method
UPDATE Library
SET Issued_By = 'Eleven'
WHERE Issued_By = 'Itachi';

SELECT * FROM Back_UP;
```

Bank PL_SQL

```
CREATE TABLE Bank_Account(
    Account_No NUMBER PRIMARY KEY ,
    Balance NUMBER
);

INSERT INTO Bank_Account(Account_No,Balance) VALUES (101,5000);
INSERT INTO Bank_Account(Account_No,Balance) VALUES (2,6000);
INSERT INTO Bank_Account(Account_No,Balance) VALUES (1,200);
INSERT INTO Bank_Account(Account_No,Balance) VALUES (3,800);
INSERT INTO Bank_Account(Account_No,Balance) VALUES (11,1000);


DECLARE
temp_account_no NUMBER(5);
temp_balance NUMBER(10);
withdraw_amount NUMBER(10):=0;
insufficient_balance EXCEPTION;

BEGIN
temp_account_no:=:temp_account_no;
withdraw_amount:=:withdraw_amount;
SELECT Balance INTO temp_balance FROM Bank_Account WHERE Account_No = temp_account_no;
DBMS_OUTPUT.put_line('Account No = ' || temp_account_no || 'Balance = ' ||temp_balance);

IF(temp_balance<withdraw_amount) THEN
RAISE insufficient_balance;
END IF;
temp_balance :=temp_balance-withdraw_amount;
DBMS_OUTPUT.put_line('Current Balance is: '||temp_balance);

EXCEPTION WHEN insufficient_balance THEN
DBMS_OUTPUT.put_line('Insufficient Balance!');
```

```
END;
/
```

```
Mongo-DB 1st Assignment (CRUD)

use AssgFirst;
db.dropDatabase();
db.createCollection("Information");
db.Information.insertOne({name:"Garry",marks:97});
db.Information.insertMany([{name:"Garry",marks:97},{name:"Devi",marks:87},{name:"Ruby",marks:69},{name:"Ram",marks:99}]);
db.Information.find().pretty();
db.Information.update({name:"Ram"},{$set:{name:"Krishna"}});
db.Information.replaceOne({"_id":ObjectId("63749ed1f19c503e449d9348")},{name:"CarryBhai"});
db.Information.deleteOne({name:"Garry"});
db.Information.deleteMany({name:"Garry",name:"Sam"});
db.Information.find({$and:[{name:"Krishna"},{marks:99}]}).pretty();
db.Information.find({$nor:[{name:"Krishna"},{marks:99}]}).pretty();
db.Information.find({$or:[{name:"Krishna"},{name:"Devi"}]}).pretty();
```

```
Mongo-DB 2nd Assignment (Aggregation & Indexing)

use AssgnSecond;
db.createCollection("customers");
db.customers.insertMany([{name:"Harry",type:"Developer"},{name:"Carry",type:"Tester"},{name:"Garry",type:"Developer"},{name:"Sohim",ty
db.customers.find();
db.customers.aggregate([{$group:{_id:"$type",category:{$sum:1}}}]);
db.customers.createIndex({name:1},{name:"Developer's Names"});
db.customers.getIndexes()
db.customers.dropIndex("Developer's Names")
```

```
Mongo-DB 3rd Assignment (map & reduce)

• use AssgnThird;

• db.createCollection("student_info");


• db.student_info.insertMany([{name:"Lokesh",marks:99},{name:"Ruturaj",marks:89},{name:"Lokesh",marks:59},{name:"Ruturaj",marks:76},{n

• db.student_info.find().pretty();


• var map = function() {emit(this.name,this.marks);};

• var reduce = function(name,marks){return Array.sum(marks);};


• db.student_info.mapReduce( map,reduce,{out:"Result"} );

• db.Result.find().pretty();
```