----------lab 1

```
INSERT INTO PERSON
VALUES
(101, 'Rahul Tripathi', 2, 56000, '2000-01-01', 'Rajkot'),
(102, 'Hardik Pandya', 3, 18000, '2001-09-25', 'Ahmedabad'),
(103, 'Bhavin Kanani', 4, 25000, '2000-05-14', 'Baroda'),
(104, 'Bhoomi Vaishnav', 1, 39000, '2005-02-08', 'Rajkot'),
(105, 'Rohit Topiya', 2, 17000, '2001-07-23', 'Jamnagar'),
(106, 'Priya Menpara', NULL, 9000, '2000-10-18', 'Ahmedabad'),
(107, 'Neha Sharma', 2, 34000, '2002-12-25', 'Rajkot'),
(108, 'Nayan Goswami', 3, 25000, '2001-07-01', 'Rajkot'),
(109, 'Mehul Bhundiya', 4, 13500, '2005-01-09', 'Baroda'),
(110, 'Mohit Maru', 5, 14000, '2000-05-25', 'Jamnagar')


INSERT INTO DEPARTMENT
VALUES
(1, 'Admin', 'Adm', 'A-Block'),
(2, 'Computer', 'CE', 'C-Block'),
(3, 'Civil', 'CI', 'G-Block'),
(4, 'Electrical', 'EE', 'E-Block'),
(5, 'Mechanical', 'ME', 'B-Block');

--1.	Find all persons with their department name & code.
SELECT
Person.PersonName,
Department.DepartmentName,
Department.DepartmentCode
```

```
FROM Person
INNER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
--2.	Find person's name whose department located in C-Block.
SELECT
Person.PersonName,
Department.DepartmentName,
Department.Location
FROM Person
INNER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
WHERE Department.Location='C-Block'
--3.	Retrieve person name, salary & department name who belongs to Jamnagar city.
SELECT
Person.PersonName,
Person.City,
Person.Salary,
Department.DepartmentName
FROM Person
LEFT OUTER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
WHERE Person.City='Jamnagar'
--4.	Retrieve person name, salary & department name who does not belongs to Rajkot city.
SELECT
Person.PersonName,
Person.City,
Person.Salary,
Department.DepartmentName
FROM Person
```

```
LEFT OUTER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
WHERE Person.City<>'Rajkot'
-- 5.	Find detail of all persons who belongs Computer department.
SELECT
Person.PersonName,
Person.City,
Person.Salary,
Person.JoiningDate,
Department.DepartmentName,
Department.Location
FROM Person
INNER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
WHERE Department.DepartmentName='Computer'
--6.	Find all persons who does not belongs to any department.
SELECT
Person.PersonName
FROM Person
WHERE Person.DepartmentID is NULL
--7.	Retrieve person’s name who joined Civil department after 1-Aug-2001.
SELECT
Person.PersonName,
Department.DepartmentName,
Person.JoiningDate
FROM Person
LEFT OUTER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
WHERE Person.JoiningDate>'1-Aug-2001' and Department.DepartmentName='Civil'
```

--8. Display all the person's name with department whose joining dates difference with current date is more than 365 days.

```
SELECT
Person.PersonName,
Department.DepartmentName,
Person.JoiningDate
FROM Person
INNER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
WHERE DATEDIFF(DAY,Person.JoiningDate,GETDATE())>365
```

--9. Find department wise person counts.

```
SELECT
Department.DepartmentName,
COUNT(Department.Departmentid) "Person count"
FROM Person
INNER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
GROUP BY Department.DepartmentName
```

--10. Give department wise maximum & minimum salary with department name.

```
SELECT
Department.DepartmentName,
MAX(Person.Salary) "Max Salary",
MIN(Person.Salary) "Min Salary"
FROM Person
INNER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
GROUP BY Department.DepartmentName
```

--11. Find city wise total, average, maximum and minimum salary.

```
SELECT
```

```
Person.City,
Max(Person.Salary) as MaxSalary,
MIN(Person.Salary) as MinSalary,
AVG(Person.Salary) as AvgSalary,
SUM(Person.Salary) as TotalSalary
FROM Person GROUP BY Person.City
--12.	Find all departments whose total salary is exceeding 100000.
SELECT
Department.DepartmentName,
SUM(Person.Salary) "Total Dept Salary"
FROM Person
INNER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
GROUP BY Department.DepartmentName
HAVING SUM(Person.Salary)>100000
--13.	Find average salary of person who belongs to Ahmedabad city.
SELECT AVG(Person.Salary) as AvgSalary, Person.City
FROM Person
GROUP BY Person.City
HAVING Person.City='Ahmedabad'
--14.	List all departments who have no person.
SELECT
Department.DepartmentName
FROM Person
FULL OUTER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
GROUP BY Department.DepartmentName
HAVING COUNT (Person.DepartmentID) =0
--15.	List out department names in which more than two persons are working.
```

```
SELECT

Department.DepartmentName,

COUNT(*) as PersonCount

FROM Person

INNER JOIN Department

ON Person.DepartmentID=Department.DepartmentID

GROUP BY Department.DepartmentName

HAVING COUNT(Person.DepartmentID)>2

--16.	Produce Output Like: <PersonName> lives in <City> and works in <DepartmentName> Department. (In single column)

SELECT

Person.PersonName + ' lives in ' + Person.City + ' and works in '

+ Department.DepartmentName + ' Department'

FROM Person

INNER JOIN Department

ON Person.DepartmentID=Department.DepartmentID

--17.	Produce Output Like: <PersonName> earns <Salary> from department <DepartmentName> monthly. (In single column)

SELECT

Person.PersonName + ' earns ' + CAST(Salary as varchar) +

' from Department ' + Department.DepartmentName + ' monthly '

FROM Person

INNER JOIN Department

ON Person.DepartmentID=Department.DepartmentID

--18.	Find city & department wise total, average & maximum salaries.

SELECT

Person.City,

Department.DepartmentName,

Max(Person.Salary) as MaxSalary,
```

```
MIN(Person.Salary) as MinSalary,
AVG(Person.Salary) as AvgSalary,
SUM(Person.Salary) as TotalSalary
FROM Person
LEFT OUTER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
GROUP BY Person.City, Department.DepartmentName
--19.	Give 10% increment in Computer department employee’s salary. (Use Update)
UPDATE  Person
SET Person.Salary=(Person.Salary+(Person.Salary*10)/100)
FROM Person
INNER JOIN Department
ON Person.DepartmentID=Department.DepartmentID
WHERE Department.DepartmentName='Computer'



----------LAB2&LAB3
---lab2

--1
--insert all data in all table
--insert person
create procedure per_insert
@FirstName Varchar (100),
@LastName Varchar (100),
@Salary Decimal (8,2),
@JoiningDate Datetime,
@DepartmentID Int,
@DesignationID Int
```

```
as insert into Person
values
(@FirstName,@LastName,@Salary,@JoiningDate,@DepartmentID,@DesignationID)


exec per_insert 'Rahul' ,'Anshu', 56000,'01-01-1990' ,1, 12
exec per_insert 'Hardik' ,'Hinsu', 18000,'1990-09-25' ,2, 11
exec per_insert 'Bhavin' ,'Kamani', 25000,'1991-05-14' ,NULL, 11
exec per_insert 'Bhoomi' ,'Patel', 39000,'2014-02-20' ,1, 13
exec per_insert 'Rohit' ,'Rajgor', 17000,'1990-07-23' ,2, 15
exec per_insert 'Priya' ,'Mehta', 25000,'1990-10-18' ,2, NULL
exec per_insert 'Neha' ,'Trivedi', 18000,'2014-02-20' ,3, 15

--insert department
create procedure dep_insert

@DepartmentName Varchar (100)

as insert into Department
values
(@DepartmentName)


 exec dep_insert 'Admin'
 exec dep_insert 'IT'
exec dep_insert 'HR'
exec dep_insert 'Account'
```

```
--insert designation

create procedure desi_insert

@DesignationName Varchar (100)

as insert into Designation
values
(@DesignationName)


 exec desi_insert 'Jobber'
 exec desi_insert 'Welder'
 exec desi_insert 'Clerk'
 exec desi_insert 'Manager'
 exec desi_insert 'Ceo'



--2
--update person
create procedure per_update
@WorkerID int,
@FirstName Varchar (100),
@LastName Varchar (100),
@Salary Decimal (8,2),
@JoiningDate Datetime,
@DepartmentID Int,
@DesignationID Int
```

```
as
update Person
set
FirstName=@FirstName,
LastName=@LastName,
Salary=@Salary,
JoiningDate=@JoiningDate,
DepartmentID=@DepartmentID,
DesignationID=@DesignationID

where
WorkerID=@WorkerID

--update department
create procedure dep_upadte

@DepartmentName Varchar (100),
@DepartmentID int
as
update Department
set
DepartmentName=@DepartmentName
where
DepartmentID=@DepartmentID

--update designation
create procedure desi_upadte
```

```
@DesignationName Varchar (100),

@DesignationID int
as
update Designation
set
DesignationName=@DesignationName
where
DesignationID=@DesignationID



--3
--delete person
create procedure per_delete
 @WorkerID int
as
delete from Person
where WorkerID=@WorkerID

--delete departmant
create procedure dep_delete
@DepartmentID int
as
delete from Department
where DepartmentID=@DepartmentID
end

--delete designation
create procedure desi_delete
```

```
@DesignationID int
as
delete from Designation
	where designationid = @designationid
end
--
-----------------------------------Stored Procedures (Lab – 2)
--1.	All tables Insert
--a.	Insert into Department table
CREATE PROCEDURE PR_Department_Insert
	@DepartmentID		int,
	@DepartmentName	varchar(100)
AS
BEGIN
INSERT INTO Department
	(
		DepartmentID,
		DepartmentName
	)
VALUES
	(
		@DepartmentID,
		@DepartmentName
	)
END
--b.	Insert into Designation table
CREATE PROCEDURE PR_Designation_Insert
	@DesignationID		int,
	@DesignationName	varchar(100)
```

```
AS
BEGIN
INSERT INTO Designation
	(
		DesignationID,
		DesignationName
	)
VALUES
	(
		@DesignationID,
		@DesignationName
	)
END
--c.	Insert into Person table
CREATE PROCEDURE PR_PERSON_Insert
	@FirstName	varchar(50),
	@LastName	varchar(50),
	@Salary		decimal(8,2),
	@JoiningDate		datetime,
	@DepartmentID		int,
	@DesignationID		int
AS
BEGIN
INSERT INTO Person
	(
		FirstName,
		LastName,
		Salary,
		JoiningDate,
```

```
                DepartmentID,
                DesignationID
        )
VALUES
        (
                @FirstName,
                @LastName,
                @Salary ,
                @JoiningDate ,
                @DepartmentID ,
                @DesignationID
        )
END
--2.     All tables Update
--a.     Update Department table
CREATE PROCEDURE PR_Department_Update
        @DepartmentID           int,
        @DepartmentName     varchar(100)
AS
BEGIN
UPDATE Department
SET
                DepartmentName = @DepartmentName
WHERE DepartmentID = @DepartmentID
END
--b.     Update Designation table
CREATE PROCEDURE PR_Designation_Update
        @DesignationID          int,
        @DesignationName    varchar(100)
```

```
AS
BEGIN
UPDATE Designation
SET
		DesignationName = @DesignationName
WHERE DesignationID = @DesignationID
END
--c.	Update Person table
CREATE PROCEDURE PR_Person_Update
	@WorkerID		int,
	@FirstName		varchar(100),
	@LastName		varchar(100),
	@Salary			decimal(8,2),
	@JoiningDate		datetime,
	@DepartmentID		int,
	@DesignationID		int
AS
BEGIN
UPDATE Person
SET
		FirstName = @FirstName,
		LastName = @LastName,
		Salary = @Salary,
		JoiningDate = @JoiningDate,
		DepartmentID = @DepartmentID,
		DesignationID = @DesignationID
WHERE WorkerID = @WorkerID
END
--3.	All tables Delete
```

```
--a.	Delete from Department table
CREATE PROCEDURE PR_Department_Delete
		@DepartmentID		int
AS
BEGIN
DELETE FROM Department
		WHERE DepartmentID = @DepartmentID
END
--b.	Delete from Designation table
CREATE PROCEDURE PR_Designation_Delete
		@DesignationID		int
AS
BEGIN
		DELETE FROM Designation
		WHERE DesignationID = @DesignationID
END
c.	Delete from Person table
CREATE PROCEDURE PR_Person_Delete
@WorkerID	int
AS
BEGIN
		DELETE FROM Person
		WHERE WorkerID = @WorkerID
END
--4.	All tables SelectPK
--a.	Select from Department table by Primary key
CREATE PROCEDURE PR_Department_SelectPK
@DepartmentID		int
AS
```

```
BEGIN
SELECT
		DepartmentID,
		DepartmentName
FROM	Department
WHERE DepartmentID = @DepartmentID
END
--b.	Select from Designation table by Primary key
CREATE PROCEDURE PR_Designation_SelectPK
	@DesignationID		int
AS
BEGIN
	SELECT
		DesignationID,
		DesignationName
	FROM	Designation
	WHERE DesignationID = @DesignationID
END
--c.	Select from Person table by Primary key
CREATE PROCEDURE PR_Person_SelectPK
	@WorkerID	int
AS
BEGIN
SELECT
		WorkerID,
		FirstName,
		LastName,
		Salary,
		JoiningDate,
```

```
                DepartmentID,
                DesignationID
FROM   Person
WHERE WorkerID = @WorkerID
END
--5.    All tables SelectAll (If foreign key is available than do join and take columns on select list)
--a.    Select All from Department table
CREATE PROCEDURE PR_Department_SelectAll
AS
BEGIN
        SELECT DepartmentID, DepartmentName
        FROM Department
END
--b.    Select All from Designation table
CREATE PROCEDURE PR_Designation_SelectAll
AS
BEGIN
        SELECT DesignationID, DesignationName
        FROM Designation
END
--c.    Select All from Person table
CREATE PROCEDURE PR_Person_SelectAll
AS
BEGIN
SELECT
                Person.WorkerID,
                Person.FirstName,
                Person.LastName,
                Person.Salary,
```

```
		Person.JoiningDate,
		Department.DepartmentName,
		Designation.DesignationName
	FROM Person
	LEFT OUTER JOIN Department
	ON Person.DepartmentID=Department.DepartmentID
	LEFT OUTER JOIN Designation
	ON Person.DesignationID=Designation.DesignationID
END
--Stored Procedures (Lab – 3)
--1.	Create Procedure that show detail of first 3 persons.
CREATE PROCEDURE PR_SelectFirstThree_Person
	AS
	BEGIN
		SELECT TOP 3
			Person.WorkerID,
			Person.FirstName,
			Person.LastName,
			Person.Salary,
			Person.JoiningDate,
			Department.DepartmentName,
			Designation.DesignationName
		FROM Person
		LEFT OUTER JOIN Department
		ON Person.DepartmentID=Department.DepartmentID
		LEFT OUTER JOIN Designation
		ON Person.DesignationID=Designation.DesignationID
END
```

--2. Create Procedure that takes department dame as input and returns a table with all workers working in that department.

```
CREATE PROCEDURE PR_Person_SelectByDepartmentName

@DepartmentName      varchar(50)

AS

BEGIN

        SELECT

                Person.FirstName,

                Department.DepartmentName

        FROM Person

        LEFT OUTER JOIN Department

        ON Person.DepartmentID=Department.DepartmentID

        WHERE Department.DepartmentName=@DepartmentName

END
```

--3. Create Procedure that takes department name & designation name as input and returns a table with worker's first name, salary, joining date & department name.

```
CREATE PROCEDURE PR_Person_SelectByDesignationNameDepartmentName

@DepartmentName      varchar(200),

        @DesignationName      varchar(250)

AS

BEGIN

SELECT

                Person.FirstName,

                Person.Salary,

                Person.JoiningDate,

                Department.DepartmentName

FROM Person

LEFT OUTER JOIN Department

ON Department.DepartmentID = Person.DepartmentID
```

```
LEFT OUTER JOIN Designation
ON Designation.DesignationID = Person.DesignationID
WHERE DesignationName = @DesignationName
AND DepartmentName = @DepartmentName
END
```

--4. Create Procedure that takes first name as an input parameter and display all the details of the worker with their department & designation name.

```
CREATE PROCEDURE PR_Person_SelectByFirstName
@FirstName      varchar(200)
AS
BEGIN
SELECT
                Person.WorkerID,
                Person.FirstName,
                Person.LastName,
                Person.Salary,
                Person.JoiningDate,
                Department.DepartmentName,
                Designation.DesignationName
FROM Person
LEFT OUTER JOIN Department
ON Department.DepartmentID = Person.DepartmentID
LEFT OUTER JOIN Designation
ON Designation.DesignationID = Person.DesignationID
WHERE FirstName = @FirstName
END
```

--5. Create Procedure which displays department wise maximum, minimum & total salaries.

```
CREATE PROCEDURE PR_Person_MaxMinTotalSalary_DepartmentWise
AS
```

```
BEGIN
SELECT
		Department.DepartmentName,
		MAX(Person.salary) as MaxSalary,
		MIN(Person.salary) as MinSalary,
SUM(Person.salary) as TotalSalary
FROM Person
INNER JOIN Department
ON Department.DepartmentID = Person.DepartmentID
GROUP BY Department.DepartmentName
END
--6.	Create Procedure which displays designation wise maximum, minimum & total salaries.
CREATE PROCEDURE PR_Person_MaxMinTotalSalary_DesignationWise
AS
BEGIN
SELECT
		Designation.DesignationName,
		MAX(Person.salary) as MaxSalary,
		MIN(Person.salary) as MinSalary,
		SUM(Person.salary) as TotalSalary
FROM Person
INNER JOIN Designation
ON Designation.DesignationID = Person.DesignationID
GROUP BY Designation.DesignationName
END


----------------------------------------------LAB-4
--1
```

```
create function h()
returns varchar(100)
as
begin
 declare @h1 varchar(100);
 set @h1='hello world';
 return @h1;
 end;

 select dbo.h()

 --2
 alter function addi(@num1 int,@num2 int)
 returns int
 as
 begin
  declare @num3 int;
            set @num3=@num1+@num2;
            return @num3;
            end;

select dbo.addi(4,5)

--3
create function cube(@num int)
returns int
as
begin
 declare @ans int
```

```
set @ans=@num*@num*@num;
return @ans
end

 select dbo.cube(3)

 --4
 create function oddeven(@num int)
 returns varchar(100)
 as
 begin
 declare @ans varchar(100)

        if(@num%2=0)
                set @ans='num is even'
        else
                set @ans='num is even'
        return @ans
        end

select dbo.oddeven(4)

--5
create function compare(@num1 int ,@num2 int)
returns varchar(100)
as
begin
declare @ans varchar(100)
 set @ans =case
```

```
                        when @num1>@num2 then cast(@num1 as varchar) + 'is grater than '
+convert(varchar ,@num2)
                        when @num1<@num2 then cast(@num1 as varchar) + 'is less than '+
convert(varchar ,@num2)
                else 'both are same '
                        end
                        return @ans
end
select dbo.compare(2,3)


--6
alter function one_to(@num int)
returns nvarchar(1000)
as
begin
 declare @i int;
 set @i=1;
 declare @ans nvarchar(1000)
 set @ans=''

 while @i<=@num
        begin
        set @ans=@ans+cast(@i as varchar(100))+','
        set @i=@i+1
        end
        return @ans
        END

select dbo.one_to(50)
```

```
--7
-- SUM OF EVEN FROM 1 TO 20

alter function ADD_(@n int)
returns int

as begin
	declare @SUM int = 0;
	declare @i int;
	set @i = 1
while(@i<=@n)
	begin

	if(@i%2 = 0)
		set @sum = @sum+@i
	set @i = @i + 1;
	end
	return @sum;
end

select dbo.add_(20)


-- prime number

create function primeNUmber(@n int)
returns varchar(20)
```

```
as begin
	declare @ans varchar(20);
	declare @count int = 0;
	declare @i int;
	set @i = 1
while(@i<=@n)
	begin
	if(@n%@i = 0)
	begin
		set @count = @count+1;
	end
	set @i = @i + 1;
	end
	if(@count = 2)
		set @ans = 'IS Prime';
	else
		set @ans = 'IS not Prime';

	return @ans
end

select dbo. primeNUmber(7)

-- date diff

alter function dateNuDiff(@start datetime, @end datetime)
returns int
```

```
as begin

return DateDiff(day, @start, @end);

end

select dbo.dateNuDiff(2023-12-30, 2023-12-15)



-- year and month diff

create Function fun_countdays(@month int , @year int)
returns int
as
begin
declare @diff int
declare @x date
declare @end date
set @x=DATEFROMPARTS(@year,@month,'01')
set @end=EOMONTH(@x)
set @diff=DATEDIFF(day,@x,@end)
return @diff+1
end

select dbo.fun_countdays(11,2023)



-- Table Valued Functions
```

```
--1

create function get_personByB()

returns table
as
        return (select * from Person
                        where FirstName like 'B%')

select * from get_personByB()



--2

create function get_personByUnique()

returns table
as
        return (select distinct FirstName from person)

select *  from get_personByUnique()

--3

create function get_personDetailByDeptID(@DeptID int)

returns table
as
```

```
        return(select * from person
                        where DepartmentID = @deptID)

select *  from get_personDetailByDeptID(2)
--------------------

--Scalar valued functions
--1.     Write a function to print "Hello World".
CREATE FUNCTION fn_PrintHello()
RETURNS VARCHAR(50)
AS
BEGIN
        DECLARE @Str AS VARCHAR(50)
        SET @Str='Hello World'
        RETURN @Str
END
--2.     Write a function which returns addition of two numbers.
CREATE FUNCTION fn_Addition(@No1 AS INT,@No2 AS INT)
RETURNS INT
AS
BEGIN
        RETURN(@No1+@No2)
END
--3.     Write a function to print cube of given number.
CREATE FUNCTION fn_Cube(@No AS INT)
RETURNS INT
AS
BEGIN
        RETURN(@No*@No*@No)
```

```
END
```

4. Write a function to check where given number is ODD or EVEN.

```
CREATE FUNCTION fn_CheckEvenOdd(@No AS INT)
RETURNS VARCHAR(50)
AS
BEGIN
	DECLARE @Str AS VARCHAR(50)
	IF (@No%2=0)
		SET @Str='NO IS EVEN'
	ELSE
		SET @Str='NO IS ODD'
	RETURN @Str
END
```

--5. Write a function to compare two integers and returns the comparison result. (Using Case statement)

```
CREATE FUNCTION fn_Compare(@a AS INT,@b AS INT)
RETURNS VARCHAR(50)
AS
BEGIN
	DECLARE @Str AS VARCHAR(50)
	SET @Str=
	CASE
		WHEN @a>@b THEN 'a is greater then b'
		WHEN @a<@b THEN 'a is less then b'
		ELSE 'a is equal to b'
	END
	RETURN @Str
END
```

--6. Write a function to print number from 1 to N. (Using while loop)

```
CREATE FUNCTION fn_Print1toN(@No AS INT)
RETURNS VARCHAR(MAX)
AS
BEGIN
	DECLARE @Str AS VARCHAR(MAX)
	SET @Str=' '
	DECLARE @i AS INT
	SET @i=1
	WHILE @i<=@No
	BEGIN
		SET @Str=@Str+CAST(@i AS VARCHAR)+' '
		SET @i=@i+1
	END
	RETURN @Str
END
--7.	Write a function to print sum of even numbers between 1 to 20.
CREATE FUNCTION fn_SumOf1to20()
RETURNS INT
AS
BEGIN
	DECLARE @i AS INT SET @i=1
	DECLARE @Sum AS INT SET @Sum=0
	WHILE (@i<=20)
	BEGIN
		IF (@i%2=0)
			SET @Sum=@Sum+@i
	SET @i=@i+1
	END
	RETURN @Sum
```

```
END
```

--8. Write a function to check weather given number is prime or not.

```
CREATE FUNCTION fn_IsPrime(@No AS INT)
RETURNS VARCHAR(50)
AS
BEGIN
  DECLARE @flag AS BIT
        SET @flag=1
  DECLARE @i AS INT
        SET @i=2
        DECLARE @Str as VARCHAR(50)
        WHILE (@i<@No)
  BEGIN
    IF (@No % @i = 0)
    BEGIN
      SET @flag = 0
      BREAK
    END
    SET @i = @i + 1
  END
        IF (@flag=0)
                SET @Str='No is not Prime'
        ELSE
                SET @Str='No is Prime'
  RETURN @Str
END
```

--9. Write a function which accepts two parameters start date & end date, and returns a difference in days.

```
CREATE FUNCTION fn_DayDiff(@StartDate AS DATE,@EndDate AS DATE)
```

```
RETURNS INT
AS
BEGIN
	DECLARE @Day AS INT
	SET @Day=DATEDIFF(DAY,@StartDate,@EndDate)
	RETURN @Day
END
```

--10. Write a function which accepts year & month in integer and returns total days in given month & year.

```
CREATE FUNCTION fn_NoOfDaysInMonthYear(@Year AS INT,@Month AS INT)
RETURNS INT
AS
BEGIN
	DECLARE  @Convert_To_FirstDay AS DATE
	DECLARE @LastDay_Of_Month AS DATE
	DECLARE @Day_Diff AS INT
	SET @Convert_To_FirstDay=DATEFROMPARTS(@Year,@Month,1)
	SET @LastDay_Of_Month=EOMONTH(@Convert_To_FirstDay)
	SET @Day_Diff=DATEDIFF(DAY,@Convert_To_FirstDay,@LastDay_Of_Month)+1
	RETURN @Day_Diff
END
```

----------Table valued functions (Use tables of lab-2)

--1. Write a function which returns a table with detail of person whose first name starts with B.

```
CREATE FUNCTION fn_FirstNameWithB()
RETURNS TABLE
AS
	RETURN(SELECT * FROM Person WHERE FirstName LIKE 'B%')
```

--2. Write a function which returns a table with unique first names from person table.

```
CREATE FUNCTION fn_UniqueName()
RETURNS TABLE
AS
        RETURN(SELECT DISTINCT FirstName FROM Person)
--3.    Write a function which accepts department ID as a parameter & returns a detail of the persons.
CREATE FUNCTION fn_GetPersonsByDepartmentID (@DepartmentId INT)
RETURNS @personsTable TABLE (
        FirstName VARCHAR(100),
LastName VARCHAR(100),
        Salary Decimal(8,2),
        DepartmentID INT)
AS
BEGIN
   INSERT INTO @personsTable (FirstName, LastName, Salary,DepartmentID)
   SELECT FirstName, LastName, Salary, DepartmentID
   FROM Person
   WHERE DepartmentID = @departmentId

   RETURN
END



--------lab 5
/* ([{ACTNO:"102",CNAME:"SUNIL",BNAME:"AJNI",AMOUNT:5000.00,ADATE:"4-1-96"},
 {ACTNO:"103",CN
AME:"MEHUL",BNAME:"KAROLBAGH",AMOUNT:3500.00,ADATE:"17-11-95"},
 {ACTNO:"104",CNAME:"MADHURI",BNAME:"CHANDI",AMOUNT:1200.00,ADATE:"17-12-95"},
 {ACTNO:"105",CNAME:"PRMOD",BNAME:"M.G. ROAD",AMOUNT:3000.00,ADATE:"27-3-96"},
 {ACTNO:"106",CNAME:"SANDIP",BNAME:"ANDHERI",AMOUNT:2000.00,ADATE:"31-3-96"},
```

{ACTNO:"107",CNAME:"SHIVANI",BNAME:"VIRAR",AMOUNT:1000.00,ADATE:"5-9-95"},

{ACTNO:"108",CNAME:"KRANTI",BNAME:"NEHRU PLACE",AMOUNT:5000.00,ADATE:"2-7-95"},

])

--1

db.Deposit.find()

--2

db.Deposit.find().preety()

--3

db.Deposit.findone()

--4

109 KIRTI VIRAR 3000.00 3-5-97

db.Deposit.indsertone({ACTNO:"109",CNAME:"KIRIT",BNAME:"VIRAR",AMOUNT:3000.00,ADATE:"3-5-97"})

--5

110 MITALI ANDHERI 4500.00 4-9-95

111 RAJIV NEHRU PLACE 7000.00 2-10-98

db.Deposit.insertmany([

{ACTNO:"110",CNAME:"MITALI",BNAME:"ANDHGERI",AMOUNT:4500.00,ADATE:"4-9-95"},

{ACTNO:"111",CNAME:"RAJIV",BNAME:"NEHRU PLACE",AMOUNT:7000.00,ADATE:"2-10-98"}

])

--6

db.deposite.find({},{CNAME:1,BNAME:1,AMOUNT:1})

--7

db.deposite.find().sort({CNAME:1})

--8

db.deposite.find().sort({ACTNO:1,AMOUNT:-1})

--9

db.deposite.find().limit(2)

--10

db.deposite.find().skip(2)

--11

db.deposite.find().skip(2).limit(1)

--12

db.deposite.find().skip(5).limit(2)

--13

db.deposite.find().count()

--14

db.Deposit.drop()

--15

db.dropdatabase()

*/

----------labb 6

create database Person_LogInfo

```
create table Person
(PersonID Int ,
PersonName Varchar (100),
Salary Decimal (8,2) ,
JoiningDate Datetime,
City Varchar (100),
Age Int,
BirthDate Datetime,)


create table PersonLog(
PLogID Int ,
PersonID Int ,
PersonName Varchar (250),
Operation Varchar (50),
UpdateDate Datetime ,)

---
/* 1-Create a trigger that fires on INSERT, UPDATE and DELETE operation on the Person table to display
a
message “Record is Affected.” */

create trigger In_Del_Upd_Table
on person
after insert ,delete ,update
as
begin
```

```
print 'record is Affected'
end

/*2-Create a trigger that fires on INSERT, UPDATE and DELETE operation on the Person table. For that, log
all operations performed on the person table into PersonLog.*/


-- insert
create trigger insert_person
on
person
after insert
as
begin
        declare @pid int ,@pname varchar(100)
        select @pid=PersonID from inserted
        select    @pname=PersonName from inserted
        insert into PersonLog
        values (@pid,@pname,'inserted',GETDATE())
        end


--delete
alter trigger delete_person
on
person
after delete
```

```
as
begin
	declare @pid int ,@pname varchar(100)
	select @pid=PersonID from deleted
	select    @pname=PersonName from deleted
	insert into PersonLog
	values (@pid,@pname,'deleted',GETDATE())
	end

--update

create trigger update_person
on
person
after update
as
begin
	declare @pid int ,@pname varchar(100)
	select @pid=PersonID from inserted
	select    @pname=PersonName from inserted
	insert into PersonLog
	values (@pid,@pname,'update',GETDATE())
	end
```

/*3. Create an INSTEAD OF trigger that fires on INSERT, UPDATE and DELETE operation on the Person table.

For that, log all operations performed on the person table into PersonLog*/

```
--insert


create trigger insert_insteadof_person
on
person
instead of insert
as
begin
        declare @pid int ,@pname varchar(100)
        select @pid=PersonID from inserted
        select    @pname=PersonName from inserted
        insert into PersonLog
        values (@pid,@pname,'inserted',GETDATE())
        end




--delete
create trigger delete_insteadof_person
on
person
instead of delete
as
begin
        declare @pid int ,@pname varchar(100)
        select @pid=PersonID from deleted
        select    @pname=PersonName from deleted
        insert into PersonLog
        values (@pid,@pname,'deleted',GETDATE())
```

```
        end

--update

create trigger update_insteadof_person
on
person
instead of update
as
begin
        declare @pid int ,@pname varchar(100)
        select @pid=PersonID from inserted
        select    @pname=PersonName from inserted
        insert into PersonLog
        values (@pid,@pname,'update',GETDATE())
        end



        select*from Person
        select*from PersonLog

        insert into Person
        values(103,'nirav',40000,'23-july-06','rajkot',23,'23-may-2005')
/*4. Create a trigger that fires on INSERT operation on the Person table to convert person name into
uppercase whenever the record is inserted.*/

create trigger uppercase
on Person
for INSERT
```

```
as
begin
        declare @pid int, @pname varchar(100)
        select @pid = Personid from inserted
        select @pname = personname from inserted
        update Person
        set Personname = upper(@pname)
        where personid = @pid
        end

/*5. Create a trigger that fires on INSERT operation on person table, which calculates the age and
update
that age in Person table.*/
create trigger ageupdate
on Person
AFTER update
as
begin
        declare @pid int, @dob varchar(100)
        select @pid = Personid from inserted
        select @dob = birthdate from inserted
        update Person
        set age = datediff(year,@dob, getdate())
        end

--6
create trigger deletemessage
on PersonLog
after delete
```

```
as begin
print 'record delete sucssefully form pewrsonlog'
end
```

```
-----------------------------------lab 7
create database  Product_Info

create tabkle
```

```
CREATE DATABASE Product_Info_450

CREATE TABLE Product_Info
(
Product_id        Int        Primary Key,
Product_Name Varchar(250) Not Null,
Price Decimal(10,2)       Not Null
)
CREATE TABLE NewProducts
(
Product_id        Int        Primary Key,
Product_Name Varchar(250) Not Null,
Price Decimal(10,2)       Not Null
)
INSERT INTO Product_Info VALUES(1,'Smatphone',35000);
```

```
INSERT INTO Product_Info VALUES(2,'Laptop',65000);
INSERT INTO Product_Info VALUES(3,'Headphones',5500);
INSERT INTO Product_Info VALUES(4,'Television',85000);
INSERT INTO Product_Info VALUES(5,'Gaming Console',32000);

--
declare
    @product_id as int,

        @product_name as varchar(250),

        @price as decimal(10,2)

        declare cursor_product1 cursor
        for select
         product_id,
              product_name,
              price

               from
                  product_info
                       open cursor_product1
                       fetch next from cursor_product1 into

                        @product_id,
                  @product_name,
                  @price

                              while @@fetch_status=0
```

```
                                    begin
                                    select @product_id,@product_name,@price
                                    fetch next from cursor_product1 into

                                     @product_id,
                              @product_name,
                              @price

                                    end
                                    close  cursor_product1
                                    deallocate  cursor_product1
--

declare
     @product_id as int,

          @product_name as varchar(250)



          declare cursor_product2 cursor
          for select
            product_id,
                    product_name


                    from
                       product_info
                             open cursor_product2
```

```
                fetch next from cursor_product2 into

                 @product_id,
            @product_name


                while @@fetch_status=0
                begin
            print cast (@product_id as varchar(250))+'_'+@product_name
                fetch next from cursor_product2  into
                 @product_id,
            @product_name


                end
                close  cursor_product2
                deallocate  cursor_product2
--

declare
    @product_id as int


        declare cursor_product3 cursor
        for select
          product_id
```

```
 from
     product_info
          open cursor_product3
          fetch next from cursor_product3 into

           @product_id




                    while @@fetch_status=0
                    begin
delete from Product_Info
where product_id=@product_id
                    fetch next from cursor_product3  into
                     @product_id




                    end
                    close  cursor_product3
          deallocate  cursor_product3

                    select*from nEWProducts

                    --
```

```
declare
    @product_id as int,

        @product_name as varchar(250),

        @price as decimal(10,2)

        declare cursor_product4 cursor
        for select
         product_id,
              product_name,
              price

              from
                 product_info
                      open cursor_product4
                      fetch next from cursor_product4 into

                       @product_id,
                 @product_name,
                 @price

                          while @@fetch_status=0
                          begin

                          update product_info set price = 1.1*@price
                          where product_id = @product_id
                          fetch next from cursor_product4 into
```

```
                                @product_id,
                        @product_name,
                        @price


                                end
                                close  cursor_product4
                                deallocate  cursor_product4
--5

declare
    @product_id as int,

        @product_name as varchar(250),

        @price as decimal(10,2)

        declare cursor_product5 cursor
        for select
          product_id,
              product_name,
              price

               from
                  product_info
                      open cursor_product5
                      fetch next from cursor_product5 into

                       @product_id,
                  @product_name,
```

```
                @price

                        while @@fetch_status=0
                        begin
                        if(@product_name ='Laptop')
                        begin
                        insert into newproducts
                        values (@product_id,@product_name,@price)
                        end
                        fetch next from cursor_product5 into

                         @product_id,
        @product_name,
        @price

                        end
                        close  cursor_product5
                        deallocate  cursor_product5

----------lab 8
CREATE DATABASE Customers_info
CREATE TABLE Customers(
Customer_id     Int     Primary Key,
Customer_Name           Varchar(250) Not Null,
Email Varchar(50) Unique)

CREATE TABLE Orders
(
Order_id        Int     Primary Key,
```

```
Customer_id     Int         REFERENCES Customers(Customer_id) ,
Order_date      date        Not Null
)



---1
        declare @val1 int ;
        declare @val2 int;
        set @val1=10;
        set @val2=0;

        begin try
                select @val1/@val2
        end try

        begin catch
                select 'Error occur that is'+ERROR_MESSAGE() as Error
        end catch




--2

declare @Str varchar(50) ;

begin try
        set @Str='hello'
        set @Str=cast(@Str as int)
end try
```

```
begin catch
		select 'Error occur that is'+ERROR_MESSAGE() as Error
end catch
```

---3

```
create proc Pr_sum
@num int,
@num2 varchar(50)

as
begin
begin try
	Print @num+@num2
end try


begin catch
	select
		ERROR_LINE() as ErrorLine,
		ERROR_NUMBER() as ErrorNumber,
		ERROR_MESSAGE() as ErrorMess,
		ERROR_STATE() as ErrorState,
```

```
                ERROR_PROCEDURE() as ErrorProce,
                ERROR_SEVERITY() as ErrorSeverity
end catch
end


exec Pr_sum 10,'pratham'


---4
create proc Pr_Primer
@Customer_id int,
@Customer_Name varchar(250),
@Email varchar(50)
as
begin
begin try
        insert into Customers values(@Customer_id,@Customer_Name,@Email)
end try


begin catch
        select
                ERROR_LINE() as ErrorLine,
                ERROR_NUMBER() as ErrorNumber,
                ERROR_MESSAGE() as ErrorMess,
                ERROR_STATE() as ErrorState,
                ERROR_PROCEDURE() as ErrorProce,
                ERROR_SEVERITY() as ErrorSeverity
```

```
end catch
end

exec Pr_Primer 1,'pratham','pratham@maile.com'
exec Pr_Primer 1,'preyarsh','pratham@maile.com'



--5
create proc Pr_custom
@Customer_id int

as
begin
	if exists (select * from Customers where Customer_id=@Customer_id)
		print('Error like  Customer_id is available in database')
	else
		throw 50001,'Error like no Customer_id is available in database',1
end

exec Pr_custom 8



---7

create proc Pr_invalid
@Customer_id int,
@Customer_Name varchar(250),
@Email varchar(50)
```

```
as
begin
	if @Customer_id>0
	insert into Customers values(@Customer_id,@Customer_Name,@Email)
	else
		throw 50002,'Error like no Customer_id is invalid ',1
end
exec Pr_invalid -1,'Preyarsh','Pretyfdsghf@maile.com'



-------6
create proc Pr_Foreign
@Order_id int,
@Customer_id int,
@Order_date date

as
begin
begin try
	insert into Orders values(@Order_id,@Customer_id,@Order_date)
end try



begin catch
	Print 'Foreign key Violation: No Customer_id is not available in database'
end catch
end
exec  Pr_Foreign 1,1,'1-May-2023'
exec  Pr_Foreign 1,10,'1-May-2023'
```

--lab 9 all normal

/*part a

1.

db.employee.find({GENDER:"Male"})

2.

db.employee.find({CITY:"London"})

3.

db.employee.find({SALARY:{$gt:3500}})

4.

db.employee.find({SALARY:{JOININGDATE:{$lt:ISODATE("15-01-01")}}})

or

db.employee.find({JOININGDATE:{$lt:"2015-01-01"}})

5.

db.employee.find({EID:{$gte:7}})

6.

db.employee.find({CITY:{$in:["London","New York"]}})

7.

db.employee.find({CITY:{$nin:["London","New York"]}})

8.

db.employee.find({CITY:"London"},{EID:1})

9.

db.employee.find({CITY:"New York"},{ENAME:1}).limit(2)

10.

db.employee.find({CITY:"New York",{ENAME:1}).limit(2).skip(2)

11.

```
db.employee.find({GENDER:"Male"},{CITY:"Sydney"})
```

12

```
db.employee.find({
  $or: [
    { CITY: "London" },
    { CITY: "Sydney" }
  ]
}, {
  EID: 1,
  ENAME: 1,
  CITY: 1,
  SALARY: 1,
  _id: 0,
})
```

13

```
db.employee.find({
  SALARY: { $gt: 7000 }
}, {
  ENAME: 1,
  SALARY: 1,
  CITY: 1,
  _id: 0,
})
```

14) `db.employee.find({ENAME:/^E/})`

15) `db.employee.find({ENAME:/^[S,M]/})`

16) db.employee.find({CITY:/^[A-M]/})

17) db.employee.find({CITY:/ney$/})

18) db.employee.find({ENAME:/[N,n]/})

19) db.employee.find({ENAME:/^E.{4}/})

20) db.employee.find({ENAME:/^S.*a$/})

21) db.employee.find({ENAME:/^Phi/},{EID:1,ENAME:1,CITY:1,SALARY:1})

22) db.employee.find({CITY:/dne/},{ENAME:1,JOININGDATE:1,CITY:1})

23

```
db.employee.find({
  CITY: { $nin: ["London", "Sydney"] }
}, {
  ENAME: 1,
  JOININGDATE: 1,
  CITY: 1,
  _id: 0
})
```

24

```
db.employee.deleteMany({
  CITY: "New York"
```

})

25

```
db.employee.updateMany(
 { ENAME: "Nick" },
 {
   $set: {
     ENAME: "Naysa",
     GENDER: "Female",
   }
 }
)
```

part b

collaction name student

26.

```
db.student.find({GENDER :'Female'})
```

27.

```
db.student.find({CITY: 'Rajkot'})
```

28.

```
db.student.find({SEM: 7})
```

29.

```
db.student.find({SEM: {$nin:[3]}})
```

30.

```
db.student.find({ROLLNO: {$gt:107}})
```

31.

```
db.student.find({CITY: {$in:['Jamnagar','Baroda']}})
```

32.

```
db.student.find({FEES: {$lt: 9000}})
```

33.

```
db.student.find({DEPARTMENT: 'Mechanical'})
```

34.

```
db.student.find({CITY: 'Baroda'},{SNAME: 1,_id: 0})
```

35.

```
db.student.find({$and:[{SEM: 3},{GENDER: 'Male'}]},{SNAME: 1,_id: 0})
```

36.

```
db.student.find({ROLLNO :{$lt: 105}},{SNAME: 1,CITY:1,FEES:1,_id: 0})
```

37.

```
db.student.find({SNAME : /^k/i})
```

38.

```
db.student.find({SNAME : /^[z,d]/i})
```

39.

```
db.student.find({CITY : /^[a-r]/i})
```

40.

```
db.student.find({$and:[{SNAME : /^P/},{SNAME : /i$/}]})
```

```
db.student.find({SNAME : /^P.*i$/i})
```

41.

```
db.student.find({DEPARTMENT : /^C/i})
```

42.

```
db.student.find({CITY : /med/i},{SNAME: 1,SEM: 1,FEES: 1,DEPARTMENT: 1,_id:0})
```

43.

```
db.student.find({CITY : {$nin:['Rajkot','Baroda']}},{SNAME: 1,SEM: 1,FEES: 1,DEPARTMENT: 1,_id:0})
```

44.

```
db.student.deleteMany({CITY: 'Jamnagar'})
```

45.

```
db.student.updateOne({SNAME:'Krish'},{$set:{SNAME: 'Fenny',GENDER: 'Female'}})
```

lab c

46.

```
db.student.find({CITY: 'Ahmedabad'},{SNAME:1,_id:0}).limit(2).skip(2)
```

47.

```
db.student.find({$or:[{CITY: 'Baroda'},{DEPARTMENT:
'CE'}]},{SNAME:1,ROLLNO:1,FEES:1,DEPARTMENT:1,_id:0})
```

48.

```
db.student.find({CITY: /oda$/i})
```

49.

```
db.student.find({$and:[{SNAME: /v/},{SNAME: /V/}]})
```

50.

```
db.student.find({SNAME: /^v.{3}$/i})
```

```
*/
```

--lab9 regex

```
/*--lab   9
```

regular exp

--part a 14 to 22

--part b 37 to 42

--part c 48,49,50

part a

14

```
db.employee.find({
  ENAME: { $regex: "^E", $options: "i" }
})
```

15

```
db.employee.find({
  $or: [
    { ENAME: { $regex: "^S", $options: "i" } }, // Names starting with S
    { ENAME: { $regex: "^M", $options: "i" } }  // Names starting with M
  ]
})
```

16

```
db.employee.find({
  CITY: { $regex: "^[A-M]", $options: "i" }
})
```

17

```
db.employee.find({
  CITY: { $regex: "ney$", $options: "i" }
})
```

18

```
db.employee.find({
  ENAME: { $regex: "n", $options: "i" }
})
```

19

```
db.employee.find({
  ENAME: { $regex: "^E.{4}$" }
})
```

20

```
db.employee.find({
  ENAME: { $regex: "^S.*a$", $options: "i" }
})
```

21

```
db.employee.find({
  ENAME: { $regex: "^Phi", $options: "i" }
}, {
  EID: 1,
  ENAME: 1,
  CITY: 1,
  SALARY: 1,
  _id: 0,
})
```

22

```
db.employee.find({
  CITY: { $regex: "dne", $options: "i" }
}, {
  ENAME: 1,
  JOININGDATE: 1,
  CITY: 1,
  _id: 0
})
```

part b

37.

```
db.student.find({SNAME:{$regex:"^k", $options: "i"}})
```

38.

```
db.student.find({SNAME:{$regex:"^[Z,D]"}})
```

39.

```
db.student.find({CITY :{$regex:"^[A-R]"$options: "i"})
```

40.

```
db.student.find({CITY:{$regex:"^[P,R$]"}})
```

41.

```
db.student.find({SNAME:{$regex:"^[P,i$]"}})
```

42.

```
db.student.find({DEPARTMENT:{$regex:"^C"}})
```

C-part

48.

```
db.student.find({CITY:{$regex:"[med]"}},{NAME:1,SEM:1,FEES:1,DEPARTMENT:1})
```

49.

```
db.student.find({CITY:{$regex:"oda$"}})
```

50.

```
db.student.find({SNAME:{$regex:"[V,v]"}})
```

51.

db.student.find({SNAME:{$regex:"^[V...]"}})

*/

----lab10

/*1.

db.employee.aggregate([{'$group':{_id:"$CITY"}}])

db.employee.aggregate([{$group:{_id:"$CITY"}}])

2.

db.employee.aggregate([{$group:{_id:"$CITY",EMP:{$sum:1}}}])

3.

db.employee.aggregate([{$group:{_id:null, SAlary:{$sum:'$SALARY'}}}])

4.

db.employee.aggregate([{$group:{_id:null,AVG:{$avg:'$SALARY'}}}])

5.

db.employee.aggregate([{$group:{_id:null,MAX:{$max:'$SALARY'},MIN:{$min:'$SALARY'}}}])

6.

db.employee.aggregate([{$group:{_id:'$CITY',TOTALSALARY:{$sum:'$SALARY'}}}])

7.

db.employee.aggregate([{$group:{_id:"$GENDER",MAX:{$max:'$SALARY'},MIN:{$min:'$SALARY'}}}])

8.

db.employee.aggregate([{$group:{_id:'$CITY',MAX:{$max:'$SALARY'},MIN:{$min:'$SALARY'}}}])

9.

db.employee.aggregate([{$match: {CITY: 'Sydney'}},{$group:{_id:"$CITY",PERSON:{$sum:1}}}])

10.

db.employee.aggregate([{$match: {CITY: 'New York'}},{$group:{_id:"$CITY",AVG:{$avg:'$SALARY'}}}])

11.

```
db.student.aggregate([{$group:{_id:'$DEPARTMENT'}}])
```

12.

```
db.student.aggregate({$group:{_id :'SCITY',STUno:{$sum: 1}}})
```

13.

```
db.student.aggregate([{$group: {_id: '$CITY',NO:{$sum: 1}}}])
```

14.

```
db.student.aggregate([{$group: {_id:null,avgFEE:{$avg:'$FEES'}}}])
```

15.

*/