Sr.	Practical
Lab-1	**Perform SQL Queries for Select with Operators** Create Database with Name: **Student_Info** **Create following table under Student_Info database. (Using Design Mode)**

Student	
Column_Name	**DataType**
StuID	Int
Name	Varchar (100)
EnrollmentNo	Varchar (12)
Division	Varchar (50)
Sem	Int
BirthDate	Datetime
Email	Varchar (100)
ContactNo	Varchar (50)

StuID	Name	EnrollmentNo	Division	Sem.	BirthDate	Email	ContactNo
101	Naimish Patel	090200107051	BCX-3	3	1992-12-06	naimishp49@gmail.com	8866205253
102	Firoz A. S.	090200107090	BCY-3	3	1994-05-03	Firoz.me@gmail.com	8885999922
103	Krunal Vyas	090243107101	BCZ-5	5	1984-03-01	Krunal.vyas@gmail.com	9990888877
104	Vijay Patel	090200107102	BCX-5	5	1985-02-15	Vijay.patel123@gmail.com	8787878787
105	Maulik Trivedi	090200107103	BCY-3	3	1988-01-20	Maulik123@gmail.com	8789564512

From the above given table perform the following queries:

1. Display all the records of Student table.
2. Display Name, Enrollment number & Division of 3rd semester student only.
3. Display the Name & ID of student and label the columns as “Student Name” and “Student ID”.
4. Display Name of Student who belongs to Semester 5 and belong to BCX-5 division.
5. Find Student Name & Enrollment number in which Student Id between 102 to 105. (Use AND & BETWEEN).
6. Find Student Name, Enrollment number with their Email who belongs to 5th or 3rd Semester. (Use IN & OR).
7. Display student Name & Id who does not belongs to BCY-3 and BCX-3 division.
8. Display all the students whose name starts with “v”.
9. Display All the Details of first two students.
10. Display all the student details order by Birth Date.
11. Display student ID, Name, Enrollment number and Email ID whose semester is either 3 or 5 and division in BCZ-5 and BCY-3.
12. Display Name & Enrollment no of first 30% Students.
13. Display Unique Semesters.
14. Find the student details who born between year 1984 and 1990.
15. Retrieve all the Students who have no Enrollment.
16. Retrieve all the students name and semester whose email id contain “123”.
17. Find Students who born before date 01-01-1986 & belongs to 4th semester.
18. Display all the students order by name in descending order.
19. Write an SQL query to clone a new table Student_New from Student table with all data.
20. Insert a new row in Student_New table: (106, “Chirag Patel”, 090200107104, “BCY-5”, 5, 1987-03-23, “chirag_s@gmail.com”, 7788998893)

Lab-2 **Perform SQL Queries on Math, String, Date, Aggregate Functions and Like Operator**

Create Database with Name: **Employee_Info**

Create following table under Employee_Info database. (Using Query)

Employee	
Column_Name	**DataType**
EID	Int
EName	Varchar (100)
Gender	Varchar (10)
JoiningDate	Datetime
Salary	Decimal (8,2)
City	Varchar (100)

EID	EName	Gender	JoiningDate	Salary	City
1	Nick	Male	01-JAN-13	4000	London
2	Julian	Female	01-OCT-14	3000	New York
3	Roy	Male	01-JUN-16	3500	London
4	Tom	Male	*NULL*	4500	London
5	Jerry	Male	01-FEB-13	2800	Sydney
6	Philip	Male	01-JAN-15	7000	New York
7	Sara	Female	01-AUG-17	4800	Sydney
8	Emily	Female	01-JAN-15	5500	New York
9	Michael	Male	*NULL*	6500	London
10	John	Male	01-JAN-15	8800	London

From the above given table perform the following queries:

1. Display all the employees whose name starts with “m” and 4th character is “h”.
2. Find the value of 3 raised to 5. Label the column as output.
3. Write a query to subtract 20 days from the current date.
4. Produce output like <Ename> having <salary> salary (i.e. Nick having 4000 salary).
5. Write a query to display name of employees whose name starts with “j” and contains “n” in their name.
6. Display 2nd to 9th character of the given string “SQL Programming”.
7. Display name of the employees whose city name ends with “ney” & contains six characters.
8. Convert all employee names to Uppercase whose name starts with “j” and having salary more than 3000.
9. Write a query to convert value 15 to string.
10. Count the number of employees in each city.
11. Concatenate the city and employee name and generate the output like: <EName> belongs to <City> city (i.e. Tom belongs to London city).
12. Get the sum of salaries for employees who joined after 2014.
13. Display all the employees whose name ends with either “n” or “y”.
14. Find smallest integer value that is greater than or equal to 63.1, 63.8 and -63.2.
15. Calculate the Total, Average, Minimum, and Maximum Salaries by City.
16. Display all employees whose joining date is not specified.
17. Display name of the employees in capital letters and city in small letters.
18. Display gender wise maximum salary.
19. Calculate the Number of Days Each Employee Has Worked (If JoiningDate is Not Null).
20. Display name of the employees and their experience in years.

Lab-3 **Perform SQL Update, Delete, Alter and Rename command**

Consider the same Employee table of Lab-2 and perform the following queries:

1. Update the salary of an employee to 4500 whose EID is 1.
2. Increase the salary of all the employee by 7% who belongs to “London” city.
3. Add a new column department Varchar (50) to the employee table.
4. Delete Employee Record Where EID = 5.
5. Delete employees who have 'NULL' in their JoiningDate.
6. Modify the salary column to allow for 10 digits and 2 decimal places.
7. Add new columns, Email and PhoneNumber, to store employee emails and phone numbers.
8. Set JoiningDate to NULL for employees in “Sydney”.
9. Reduce salary by 200 for all employees earning above 5000.
10. Rename the Salary column to MonthlySalary.
11. Remove column PhoneNumber from employee table.
12. Rename a column from Ename to FirstName.
13. Rename a table from Employee to EmpMaster.
14. Remove City column form the EmpMaster table.
15. Drop EmpMaster table from the database.

Lab-4 **Implement SQL Joins**

Create Database with Name: **Person_Info**

Create following tables under Person_Info database. (Using Design Mode)

Person

Column_Name	DataType	Constraints
PersonID	Int	Primary Key
PersonName	Varchar (100)	Not Null
DepartmentID	Int	Foreign Key, Null
Salary	Decimal (8,2)	Not Null
JoiningDate	Datetime	Not Null
City	Varchar (100)	Not Null

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

Department

Column_Name	DataType	Constraints
DepartmentID	Int	Primary Key
DepartmentName	Varchar (100)	Not Null, Unique
DepartmentCode	Varchar (50)	Not Null, Unique
Location	Varchar (50)	Not Null

DepartmentID	DepartmentName	DepartmentCode	Location
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

From the above given table perform the following queries:

1. Find all persons with their department name & code.
2. Give department wise maximum & minimum salary with department name.
3. Find all departments whose total salary is exceeding 100000.
4. Retrieve person name, salary & department name who belongs to Jamnagar city.
5. Find all persons who does not belongs to any department.
6. Find department wise person counts.
7. Find average salary of person who belongs to Ahmedabad city.
8. Produce Output Like: <PersonName> earns <Salary> from department <DepartmentName> monthly (In Single Column).
9. List all departments who have no persons.
10. Find city & department wise total, average & maximum salaries.
11. Display Unique city names.
12. List out department names in which more than two persons.
13. Combine person name's first three characters with city name's last three characters in single column.
14. Give 10% increment in Computer department employee's salary.
15. Display all the person name's who's joining dates difference with current date is more than 365 days.

Lab-5

Implement SQL Views

Create Database with Name: **SQL_Views**

Create following table under SQL_View database. (Using Design Mode)

1. Simple View

Student		
Column_Name	**DataType**	**Constraints**
Rno	Int	Primary Key
Name	Varchar (50)	Null
Branch	Varchar (50)	Null
SPI	Decimal (4,2)	Null
Bklog	Int	Null

RNo	Name	Branch	SPI	Bklog
101	Raju	CE	8.80	0
102	Amit	CE	2.20	3
103	Sanjay	ME	1.50	6
104	Neha	EC	7.65	1
105	Meera	EE	5.52	2
106	Mahesh	EC	4.50	3

From the above given table perform the following queries:

Part – A

1. Create a view Personal with all columns.

2. Create a view Student_Details having columns Name, Branch & SPI.
3. Create a view Academic having columns RNo, Name, Branch.
4. Create a view Student_Data having all columns but students whose bklogs are more than 2.
5. Create a view Student_Pattern having RNo, Name & Branch columns in which Name consists of four letters.

Part – B

6. Insert a new record to Academic view. (107, Meet, ME). Remaining all columns must be null.
7. Update the branch of Amit from CE to ME in Student_Details view.
8. Delete a student whose roll number is 104 from Academic view.
9. Create a view that displays information of all students whose spi is above 8.5.
10. Create a view that displays 0 backlog students.

Part – C

11. Create a view Computer that displays CE branch data only.
12. Create a view Result_EC that displays the name and SPI of students with SPI less than 5 of branch EC.
13. Update the result of student Sanjay to 4.90 in Result_EC view.
14. Create a view Stu_Bklog with RNo, Name and Bklog columns in which name starts with 'M' and having bklogs more than 5.
15. Drop Computer view form the database.

2. Complex View

Create following tables under SQL_View database. (Using Design Mode)

Customer		
Column_Name	**DataType**	**Constraints**
CustomerID	Int	Primary Key
FirstName	Varchar (50)	Not Null
LastName	Varchar (50)	Not Null

CustomerID	**FirstName**	**LastName**
1	John	Doe
2	Jane	Smith
3	Michael	Johnson
4	Mark	Wood
5	Moin	Khan

Account		
Column_Name	**DataType**	**Constraints**
AccountID	Int	Primary Key
CustomerID	Int	Foreign Key, Null
Balance	Decimal (10,2)	Not Null
AccountType	Varchar (50)	Not Null
CreatedDate	Date	Not Null

AccountID	**CustomerID**	**Balance**	**AccountType**	**CreatedDate**
101	1	5000	Current	2023-01-01
102	1	8000	Saving	2023-02-25
103	2	10000	Saving	2023-03-30
104	4	15000	Current	2020-06-15
105	3	7500	Saving	2021-11-27
106	5	13450	Current	2019-10-13

From the above given tables perform the following queries:

Part – A

1. Create view that displays all the customers along with their corresponding account balances.
2. Create view that displays total balance for each customer.
3. Create view that displays customers who have multiple accounts.

Part – B

4. Create a view that displays customer details who have an account created in the last month.
5. Create a view that displays customers who have the highest account balance.

Part – C

6. Create a view that displays name of the customers whose account balance is between 5000 to 10000 and account type is Saving.
7. Create a view that displays minimum and maximum balance for each customer.

Lab-6 **Implement advanced queries on SQL Views**

Create following tables under SQL_VIEW database and solve given queries:

Country		
Column_Name	**DataType**	**Constraints**
Country_ID	Int	Primary Key
Country_Name	Varchar (100)	Not Null, Unique
Population	Int	Not Null
Area_sq_km	Float	Not Null
Capital	Varchar (100)	Not Null
Currency	Varchar (50)	Not Null

State		
Column_Name	**DataType**	**Constraints**
State_ID	Int	Primary Key
State_Name	Varchar (100)	Not Null, Unique
Population	Int	Not Null
Area_sq_km	Float	Not Null
Capital	Varchar (100)	Not Null
Country_ID	Varchar (50)	Foreign Key, Null

Customer_ID	Customer_Name	Population	Area_sq_km	Capital	Currency
1	USA	331002651	9833520	Washington, D.C.	USD
2	CANADA	38005238	9984670	Ottawa	CAD
3	BRAZIL	212559417	8515767	Brasília	BRL
4	INDIA	1380004385	3287263	New Delhi	INR
5	RUSSIA	145934462	17098246	Moscow	RUB
6	CHINA	1439323776	9706961	Beijing	CNY
7	AUSTRALIA	25499881	7692024	Canberra	AUD
8	ARGENTINA	45195777	2780400	Buenos Aires	ARS
9	GERMANY	83783942	357022	Berlin	EUR
10	SOUTH AFRICA	59308690	1221037	Pretoria	ZAR

State_ID	State_Name	Population	Area_sq_km	Capital	Customer_ID
1	California	39538223	423972	Sacramento	1
2	Texas	28995881	695662	Austin	1
3	Ontario	14734014	917741	Toronto	2

4	São Paulo	46289333	248209	São Paulo	3
5	Maharashtra	114063427	307713	Mumbai	4
6	Moscow Oblast	7694989	443562	Moscow	5
7	Beijing	21542000	16410	Beijing	6
8	New South Wales	8160062	800642	Sydney	7
9	Buenos Aires Province	17700000	307571	La Plata	8
10	Bavaria	13076721	70550	Munich	9

Part – A

1. Create a view that displays the top 5 countries with the highest population, along with their population figures.
2. Create a view that lists countries that do not have any states.
3. Create a view that displays the state with the highest population for each country, along with its population figure.
4. Create a view that lists states that do not have a designated capital.
5. Create a view that displays countries with more than one capital city.

Customer		
Column_Name	**DataType**	**Constraints**
CustomerID	Int	Primary Key
FirstName	Varchar (50)	Not Null
LastName	Varchar (50)	Not Null
Email	Varchar (50)	Not Null
Phone	Nvarchar (20)	Not Null

CustomerID	FirstName	LastName	Email	Phone
1	John	Doe	john.doe@example.com	1234567890
2	Jane	Smith	jane.smith@example.com	9876543210
3	Mike	Johnson	mike.johnson@example.com	1112223333
4	Emily	Williams	emily.williams@example.com	4445556666
5	David	Brown	david.brown@example.com	7778889999

Orders		
Column_Name	**DataType**	**Constraints**
OrderID	Int	Primary Key
CustomerID	Int	Foreign Key, Null
OrderDate	Date	Not Null
TotalAmount	Decimal (10,2)	Not Null

OrderID	CustomerID	OrderDate	TotalAmount
1001	1	2023-07-01	100.50
1002	1	2023-07-02	75.20
1003	3	2023-07-03	250.75
1004	4	2023-07-04	50.00
1005	5	2023-07-05	300.00

Create following tables and solve given queries:

Part – B

1. Create a view AllOrdersView to Get All Orders with customer name.
2. Create a view to Get Customers with No Email Addresses.
3. Create a view to return sum of total amount of order as total_amount.
4. Create a view to Get Customers with Their Total Order Amount.

5. Create a view to Get Customers with Their Latest Order Date.

Part – C

6. Create a view to Get Customers with No Orders.
7. Create a view to Get Customers with Their Total Number of Orders.
8. Create a view to Get Customers with High-Value Orders.
9. Getting Customers with more than 1 order Placed.
10. Getting Customers with Orders in Date Range 2023-07-01 to 2023-07-04.

Lab-7 **Perform PL/SQL Programs**

Part – A

1. Write a PL/ SQL program to print a welcome message on a screen.
2. Write a PL/SQL program to addition of two numbers.
3. Write a PL/SQL program to print maximum number out of three numbers.
4. Write a PL/ SQL program to print number from 1 to 10. (Using while loop)
5. Write a PL/ SQL program to check where given number is ODD or EVEN.

Part – B

6. Write a PL/ SQL program to print ODD numbers between 1 and 10.
7. Write a PL/ SQL program to print Sum of 1 to 50 numbers.
8. Write a PL/ SQL program to print Sum of even numbers between 1 to 20.
9. Write a PL/ SQL program to inserting even numbers into even table & odd numbers into odd table between 1 to 50.

Part – C

10. Write a PL/ SQL program to calculate the factorial of N number and display the result.
11. Write a PL/ SQL program to check weather given number is prime or not.
12. Write a PL/ SQL program to reverse a string and display the reversed string.
13. Write a PL/ SQL program to generate the Fibonacci series up to N number and display the series.
14. Write a PL/ SQL program to check given year is leap year or not.

Lab-8 **Implement SQL Stored Procedures: CRUD Operation**

Create tables under SQL_SP database as per following data.

Student		
Column_Name	**DataType**	**Constraints**
RNo	Int	Primary Key
Name	Varchar (50)	Not Null
Branch	Varchar (50)	Not Null

RNo	**Name**	**Branch**
101	Raju	CE
102	Amit	CE
103	Sanjay	ME
104	Neha	EC
105	Meera	EE
106	Mahesh	ME

Result		
Column_Name	**DataType**	**Constraints**
RNo	Int	Foreign Key, Null
SPI	Decimal (4,2)	Not Null

RNo	**SPI**
101	8.8

102	9.2
103	7.6
104	8.2
105	7.0
107	8.9

From the above given tables perform the following queries:

Part – A

1. Both tables Insert, Update and Delete.
2. Both tables SelectPK.
3. Both tables SelectAll.

Part – B

4. Create a stored procedure that takes branch as input and returns a table with all the students studying in that department.
5. Create a stored procedure to display Rno, Name and SPI of first 2 students only.

Part – C

6. Create a stored procedure which displays top 5 students based on SPI in descending order.
7. Create a stored procedure which displays branch wise maximum and minimum SPI.

Lab-9 **Implement SQL Stored Procedures with parameters.**

Create tables under SQL_SP database as per following data.

Department		
Column_Name	**DataType**	**Constraints**
DepartmentID	Int	Primary Key
DepartmentName	Varchar (100)	Not Null, Unique

Designation		
Column_Name	**DataType**	**Constraints**
DesignationID	Int	Primary Key
DesignationName	Varchar (100)	Not Null, Unique

Person		
Column_Name	**DataType**	**Constraints**
WorkerID	Int	Primary Key, Auto Increment
FirstName	Varchar (100)	Not Null
LastName	Varchar (100)	Not Null
Salary	Decimal (8,2)	Not Null
JoiningDate	Datetime	Not Null
DepartmentID	Int	Foreign Key, Null
DesignationID	Int	Foreign Key, Null

DepartmentID	DepartmentName
1	Admin
2	IT
3	HR
4	Account

DesignationID	DesignationName
11	Jobber
12	Welder
13	Clerk
14	Manager
15	CEO

WorkerID	FirstName	LastName	Salary	JoiningDate	DepartmentID	DesignationID
101	Rahul	Anshu	56000	01-01-1990	1	12
102	Hardik	Hinsu	18000	25-09-1990	2	11
103	Bhavin	Kamani	25000	14-05-1991	*NULL*	11
104	Bhoomi	Patel	39000	20-02-2014	1	13
105	Rohit	Rajgor	17000	23-07-1990	2	15
106	Priya	Mehta	25000	18-10-1990	2	*NULL*
107	Neha	Trivedi	18000	20-02-2014	3	15

Consider the following tables and solve the given queries:

Part – A

1. Create a stored procedure that takes department name as an input and returns a table with all workers working in that department.
2. Create procedure that takes department name & designation name as input and returns a table with worker's first name, salary, joining date & department name.
3. Create a Procedure that takes the first name as an input parameter and display all the details of the worker with their department & designation name.
4. Create Procedure which displays department wise maximum, minimum & total salaries.
5. Create Procedure which displays designation wise maximum, minimum & total salaries.

Consider the following tables and solve the given queries:

Employees		
Column_Name	**DataType**	**Constraints**
Emp_ID	Int	Primary Key
Emp_Name	Varchar (50)	Not Null
Emp_Salary	Decimal (8,2)	Not Null
Department	Varchar (50)	Not Null
Hire_Date	Date	Not Null

Emp_ID	Emp_Name	Emp_Salary	Department	Hire_Date
1	John	50000.00	Sales	2022-01-15
2	Jane	60000.00	Marketing	2021-05-10
3	Mike	75000.00	IT	2020-09-20
4	Emily	45000.00	Finance	2023-02-28
5	David	80000.00	IT	2021-11-05

Part – B

1. Create a Stored Procedure to Calculate Total Salary Expense.
2. Create a Stored Procedure to Get Employees with the Longest Tenure.
3. Create a Stored Procedure to Calculate the Total Number of Employees in Each Department.
4. Create a Stored Procedure to Calculate the Average Salary for Each Department.

Part – C

5. Create a Stored Procedure to Calculate Average Salary in a Department.
6. Create a Stored Procedure to Generate Monthly Salary Report.
7. Create a Stored Procedure to Get Highest Paid Employee.
8. Create a Stored Procedure to Get Employees Hired in a Specific Year.

Lab-10 **Implement SQL User Defined Functions (UDFs)**

Create following table under SQL_UDF database as per following data. (Using Query)

Employee	
Column_Name	**DataType**
EID	Int
EName	Varchar (100)
Gender	Varchar (10)
JoiningDate	Datetime
Salary	Decimal (8,2)
City	Varchar (100)

EID	EName	Gender	JoiningDate	Salary	City
1	Nick	Male	01-JAN-13	4000	London
2	Julian	Female	01-OCT-14	3000	New York
3	Roy	Male	01-JUN-16	3500	London
4	Tom	Male	*NULL*	4500	London
5	Jerry	Male	01-FEB-13	2800	Sydney
6	Philip	Male	01-JAN-15	7000	New York
7	Sara	Female	01-AUG-17	4800	Sydney
8	Emily	Female	01-JAN-15	5500	New York
9	Michael	Male	*NULL*	6500	London
10	John	Male	01-JAN-15	8800	London

From the above given table perform the following queries:

Part – A

Scalar Valued Functions

1. Create a function which displays total number of employees.
2. Create a function which returns highest salary from Employee table.
3. Create a function to get the experience of the employee based on their joining date.
4. Create a function that calculates the factorial of a given number.
5. Create a function which returns minimum salary of female employee.
6. Create a function which count unique city from employee table.
7. Create a Scalar-valued function that returns the name combined with salary of an employee based on their employee id and displayed output like 'Roy having 3500 salaries'.

Table Valued Functions

1. Create a function which retrieve the data of Employee table.
2. Create a function which returns an employee table with city wise total salary.
3. Create a function which returns an employee table with gender wise maximum, minimum, total and average salaries.
4. Create a function which return an employee table with details of employee whose name starts with **J**.
5. Create a function to get all the male employees.
6. Create a function to get employees from a given city.
7. Create a function that displays employees with a salary greater than a specified amount.
8. Create a function to get employees who joined after a given specified date.

Create following table under SQL_UDF database as per following data. (Using Design Mode)

Employee	
Column_Name	**DataType**
Employee_ID	Int, Primary Key
First_Name	Varchar (100)
Last_Name	Varchar (50)

Age	Int
Departemt	Varchar (50)

Employee_ID	First_Name	Last_Name	Age	Department
1	John	Doe	30	HR
2	Jane	Smith	25	Finance
3	Michael	Johnson	35	IT
4	Emily	Williams	28	Marketing
5	Robert	Brown	22	IT

From the above given table perform the following queries:

Part – B

1. Create UDF to get the full name and department of an employee.
2. Create UDF to calculate the age of an employee based on the birth year.
3. Create UDF to get the number of employees in a specific department.
4. Create UDF to concatenate the first name and last name with a custom separator.
5. Create UDF to check if an employee is part of the IT department.
6. Create UDF to convert age into a friendly message.
7. Create UDF to find the average age of employees in a department.
8. Create UDF to check if an employee exists in the table.
9. Create UDF to get the last name in uppercase.

Part – C

10. Create UDF to check if an employee is older than a specific age.
11. Create UDF to get the first initial of an employee's first name.
12. Create UDF to get the number of employees older than a specific age.
13. Create UDF to check if an employee's first name starts with a specific letter.
14. Create UDF to calculate the years of experience based on the current year and an employee's starting year.

Lab-11 **Implement SQL Trigger**

Create following tables under TRIGGER_SQL database as per following data. (Using Design Mode)

Person		
Column_Name	**DataType**	**Constraints**
PersonID	Int	Primary Key
PersonName	Varchar (100)	Not Null
Salary	Decimal (8,2)	Not Null
JoiningDate	Datetime	Not Null
City	Varchar (100)	Not Null
Age	Int	Null
BirthDate	Datetime	Not Null

PersonLog		
Column_Name	**DataType**	**Constraints**
PLogID	Int	Primary Key, Auto Increment
PersonID	Int	Not Null
PersonName	Varchar (250)	Not Null
Operation	Varchar (50)	Not Null
UpdateDate	Datetime	Not Null

From the above given tables perform the following queries:

Part – A

1. Create a trigger that fires on INSERT, UPDATE and DELETE operation on the Person table to display a message "Record is Affected."
2. Create a trigger that fires on INSERT operation on the Person table to convert person name into uppercase whenever the record is inserted.
3. Create an INSERT trigger on person table, which calculates the age and update that age in Person table.

Part – B

4. Create a trigger that fires on INSERT, UPDATE and DELETE operation on the Person table. For that, create a new table PersonLog to log (enter) all operations performed on the person table.
5. 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.

Part – C

6. Create DELETE trigger on PersonLog table, when we delete any record of PersonLog table it prints 'Record deleted successfully from PersonLog'.

Lab-12 **Implement SQL Cursor**

Create following table under CURSOR_SQL database as per following data. (Using Design Mode)

Products		
Column_Name	**DataType**	**Constraints**
Product_id	Int	Primary Key
Product_Name	Varchar (250)	Not Null
Price	Decimal (10,2)	Not Null

Products		
Product_id	**Product_Name**	**Price**
1	Smatphone	35000
2	Laptop	65000
3	Headphones	5500
4	Television	85000
5	Gaming Console	32000

From the above given tables perform the following queries:

Part – A

1. Create a cursor Product_Cursor to fetch all the rows from a products table.
2. Create a cursor Product_Cursor_Fetch to fetch the records in form of ProductID_ProductName (Example: 1_Smartphone).
3. Create a cursor that displays the products price above 50000.
4. Create a cursor Product_CursorUpdate that retrieves all the data from the products table and increases the price by 10%.

Part – B

5. Create a cursor that finds product with maximum price.
6. Create a cursor to insert details of Products table into the NewProducts table if the product is "Laptop".

Part – C

7. Create a cursor that increase price of products by 5000 if price is below 40000.
8. Create a cursor that displays products with prices below the average price.
9. Create a cursor Product_CursorDelete that deletes all the data from the Products table.

Lab-13

Implement SQL Exception Handling

Create following tables under EXCEP_SQL database as per following data. (Using Design Mode)

Customers		
Column_Name	**DataType**	**Constraints**
Customer_id	Int	Primary Key
Customer _Name	Varchar (250)	Not Null
Email	Varchar (50)	Unique

Orders		
Column_Name	**DataType**	**Constraints**
Order_id	Int	Primary Key
Customer_id	Int	Foreign Key
Order_date	date	Not Null

From the above given tables perform the following queries:

Part – A

1. Handle Divide by Zero Error and Print message like: Error occurs that is - Divide by zero error.
2. Try to convert string to integer and handle the error using try…catch block.
3. Handle a Primary Key Violation while inserting data into customers table and print the error details such as the error message, error number, severity, and state.

Part – B

4. Handle a Foreign Key Violation while inserting data into Orders table and print appropriate error message.
5. Create a procedure that prints the sum of two numbers: take both numbers as integer & handle exception with all error functions if any one enters string value in numbers otherwise print result.

Part – C

6. Validate Age and raise the exception if Age is below 18 (Use RAISERROR() function).
7. Throw custom exception that throws error if the data is invalid.
8. Throw custom exception using stored procedure which accepts Customer_id as input & that throws Error like no Customer_id is available in database.