PRACTICAL-4 **DATE:_____**

AIM: For a given set of relation schemes, create tables and perform the following Simple Queries, Simple Queries with Aggregate functions, Queries with Aggregate functions (group by and having clause), Queries involving- Date Functions, String Functions , Math Functions

INPUT:-

i). Relation Schema :-

- A Relation Schema is a blueprint or structure that defines a relation in a relational database. It specifies the name of the relation and the names and types of its attributes.

- Key components of a relation schema include :

i). Relation Name
ii). Attributes
iii). Domain
iv). Keys

ii). Aggregate Functions :-

i). COUNT() : Returns the number of rows.
ii). AVG() : Returns the average value.
iii). SUM() : Returns the sum of values.
iv). Min() : Returns the minimum value.
v). Max() : Returns the maximum value.

iii). Date Functions :-

i). Year() : Extracts the year from a date.
ii). CURDATE() : Returns the current date.

iv). String Functions :-

i). LIKE : Used to search for a specified pattern in column
ii). CONCAT() : Concatenates two or more strings.

v). Math Functions :-

i). ROUND() : Rounds a number to a specific number of decimal places.
ii). ABS() : Returns the absolute value of a number.
iii). CEIL() : Returns the smallest integer value greater than or equal to a number.
iv). FLOOR() : Returns the largest integer value less than or equal to a number.

SQL Queries :-

i) Create Customers, Accounts and Transactions table.

```
// Customers Table
CREATE TABLE Customers (
        CustomerID INT AUTO_INCREMENT PRIMARY KEY,
        FirstName VARCHAR(50) NOT NULL,
        LastName VARCHAR(50) NOT NULL,
        DateofBirth DATE NOT NULL,
        Gender CHAR(1),
        Mobile VARCHAR(10) NOT NULL
    );
// Accounts Table
CREATE TABLE Accounts (
        AccountID INT AUTO_INCREMENT PRIMARY KEY
        CustomerID INT,
        AccountType VARCHAR(20) NOT NULL,
        Balance DECIMAL(15,2),
```

```
        FOREIGN KEY (CustomerID) REFERENCES
                        Customers (CustomerID)
    );
// Transactions Table
CREATE TABLE Transactions (
    TransactionID INT AUTO_INCREMENT PRIMARY KEY,
    AccountID INT,
    TransactionDate DATE NOT NULL,
    Amount Decimal(15,2) NOT NULL,
    TransactionType VARCHAR(20) NOT NULL,
    FOREIGN KEY (AccountID) REFERENCES
                        Accounts(AccountID)
);
```

ii). Insert data into Customers, Accounts and Transactions.

```
// Customers
INSERT INTO Customers (FirstName, LastName, DateofBirth,
                        Gender, Mobile) VALUES
('John', 'Doe', '1980-01-05', 'M', '1234567890'),
('Jane', 'Smith', '1985-05-23', 'F', '0987654321');
// Accounts
INSERT INTO Accounts (CustomerID, AccountType, Balance)
    VALUES
(1, 'Savings', 5000.00),
(2, 'Checking', 1500.00),
(1, 'Checking', 2000.00);
// Transactions
INSERT INTO Transactions (AccountID, TransactionDate,
                Amount, TransactionType) VALUES
(1, '2024-01-10', 500.00, 'Deposit'),
(2, '2024-01-15', 200.00, 'Withdrawal'),
(3, '2024-01-20', 300.00, 'Deposit');
```

iii). Simple Queries :-

a). View all Customers :

```
SELECT * FROM Customers ;
```

CustomerID	FirstName	LastName	DateofBirth	Gender	Mobile
1	John	Doe	1980-01-15	M	1234567890
2	Jane	Smith	1985-05-23	F	0987654321

b). View all Accounts :

```
SELECT * FROM Accounts ;
```

AccountID	CustomerID	AccountType	Balance
1	1	Savings	5000.00
2	2	Checking	1500.00
3	1	Checking	2000.00

c). View all Transactions :

```
SELECT * FROM Transactions ;
```

TransactionID	AccountID	TransactionDate	Amount	TransactionType
1	1	2024-01-10	500.00	Deposit
2	2	2024-01-15	200.00	Withdrawal
3	3	2024-01-20	300.00	Deposit

iv). Simple Queries with Aggregate Functions :-

a). Total balance in all accounts :

```
SELECT SUM(Balance) AS TotalBalance FROM Accounts;
```

TotalBalance
8500.00

b). Average balance in Savings accounts :

```
SELECT AVG(Balance) As AverageSavingsBalance FROM
  ACCOUNTS WHERE AccountType = 'Savings';
```

AverageSavingsBalance
5000.00

v). Queries with Aggregate Functions (GROUP BY and Having) :-

a). Total balance per account type :

```
SELECT AccountType, SUM(Balance) AS TotalBalance
FROM ACCOUNTS
GROUP BY AccountType;
```

AccountType	TotalBalance
Savings	5000.00
Checking	3500.00

b). Total balance per customer :

```
SELECT CustomerID, SUM(Balance) As TotalBalance
FROM Accounts GROUP BY CustomerID
HAVING SUM(Balance) >3000;
```

CustomerID	TotalBalance
1	7000.00

vi). Queries Involving Date Functions :-

a). Transactions in the last month :

```
SELECT * FROM Transactions
WHERE TransactionDate >= DATEADD(MONTH, -1, GETDATE());
```

TransactionID	AccountID	TransactionDate	Amount	TransactionType
1	1	2024-01-10	500.00	Deposit
2	2	2024-01-15	200.00	Withdrawal
3	3	2024-01-20	300.00	Deposit

b). Customer's ages :

```
SELECT FirstName, LastName,
DATEDIFF(YEAR, DateOfBirth, GETDATA()) AS Age
FROM Customers;
```

FirstName	LastName	Age
John	Doe	44
Jane	Smith	39

vii). Queries Involving String Functions :-

a). Customer's full names :

```
SELECT CONCAT(FirstName,' ', LastName) As FullName
FROM Customers;
```

FullName
John Doe
Jane Smith

viii). Queries Involving Math Functions :-

a). Round the balance to the nearest whole number :

```
SELECT AccountID, ROUND(Balance, 0) AS RoundedBalance
FROM Accounts;
```

AccountID	RoundedBalance
1	5000
2	1500
3	2000

b). Find the absolute value of transactions :

```
SELECT TransactionID, ABS(Amount) AS AbsoluteAmount
FROM Transactions;
```

TransactionID	AbsoluteAmount
1	500.00
2	200.00
3	300.00