(Project Title Internet Service Provider)

Semester Project

G-5	Name	Roll Number
	Darak Ali	Fa-2021/BSCS/130-D
	Muhammad Ismail	Fa-2021/BSCS/131-D

Session: Spring 2023
Semester: 4th
Section: D
Submitted to: Mr. Abdul Rehman
Date: 26-May-2023

Department of Computer Science
Lahore Garrison University

Introduction:

This database is designed to store and manage the information of an Internet Service Provider (ISP) that offers various services to its customers. The database consists of 13 tables that capture different aspects of the ISP's business. The Customers table stores the personal and contact details of the customers, as well as their address information, which is linked to the Addresses table. The Servicess table lists the different services that the ISP provides, such as broadband, cable TV, or phone. The Subscriptions table records the subscriptions that each customer has for each service, along with the start and end dates. The Billing table tracks the billing information for each subscription, such as the billing date, amount due, and due date. The Payments table records the payments made by the customers for their bills. The Usage table records the data usage of each subscription on a daily basis. The Technicians table stores the details of the technicians who work for the ISP. The Issues table records the issues reported by the customers, such as network outage, slow speed, or equipment malfunction. The Issues table also links to the Technicians table to indicate which technician is assigned to resolve each issue. The Equipment table lists the different types of equipment that the ISP uses or provides to its customers, such as routers, modems, or set-top boxes. The ServiceOffices table stores the information of the service offices that the ISP

operates in different cities and zip codes. The Promotions table lists the promotions that the ISP offers to its customers, such as discounts, free trials, or loyalty rewards. The Promotions table also specifies the start and end dates of each promotion. The Contracts table records the contracts that each customer signs with the ISP for each service, along with the start and end dates.

Step 1- Entity Identification

Addresses, Customers, Servicess, Subscriptions, Billing, Payments, Usage, Technicians, Issues, Equipment, ServiceOffices, Promotions, Contracts

Step 2- Relationship Identification

- The Customers table has a foreign key AddressID that references the Addresses table's primary key AddressID.
- The Customers table has a foreign key EquipmentID that references the Equipment table's primary key EquipmentID.
- The Subscriptions table has two foreign keys: CustomerID and ServiceID, which reference the primary keys of the Customers and Servicess tables respectively.
- The Billing table has a foreign key SubscriptionID that references the primary key of the Subscriptions table.
- The Payments table has a foreign key BillingID that references the primary key of the Billing table.
- The Usage table has a foreign key SubscriptionID that references the primary key of the Subscriptions table.
- The Technicians table has two foreign key: EquipmentID and OfficeID,w hich references the primary key of the Equipment table and ServiceOffices table respectively.
- The Issues table has two foreign keys: CustomerID and TechnicianID, which reference the primary keys of the Customers and Technicians tables respectively.
- The Promotions table has a foreign key ServiceID that references the primary key of the Servicess table.
- The Contracts table has two foreign keys: CustomerID and ServiceID, which reference the primary keys of the Customers and Servicess tables respectively.

Step 3- Cardinality Identification

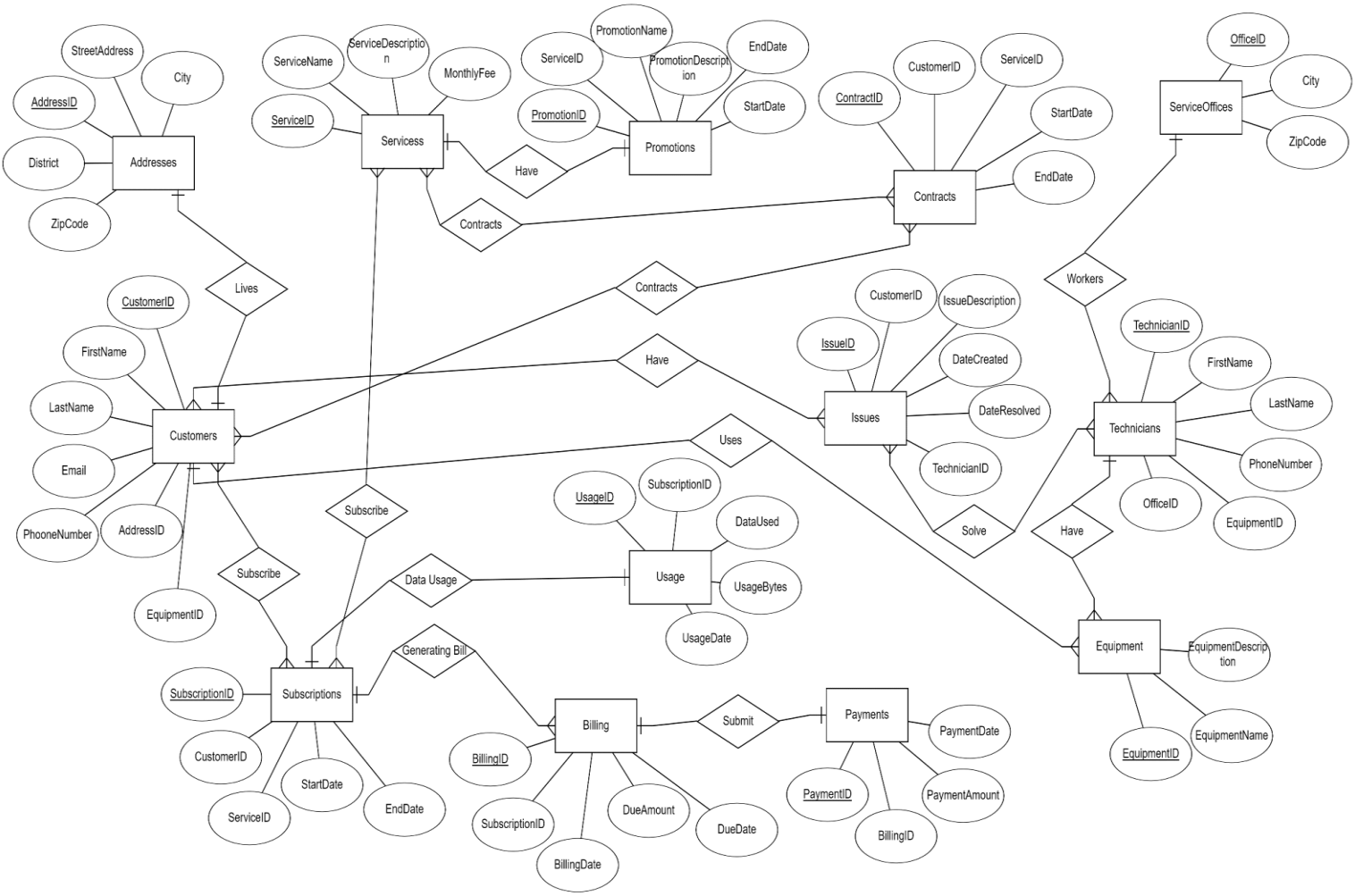
- The relationship between the Customers and Addresses tables is one-to-one
- The relationship between the Customers and Equipment tables is one-to-many
- The relationship between the Subscriptions and Customers tables is many-to-many
- The relationship between the Subscriptions and Servicess tables is many-to-many
- The relationship between the Billing and Subscriptions tables is many-to-one
- The relationship between the Payments and Billing tables is one-to-one
- The relationship between the Usage and Subscriptions tables is one-to-one
- The relationship between the Technicians and Equipment tables is one-to-many

- The relationship between the Issues and Customers tables is many-to-many
- The relationship between the Issues and Technicians tables is many-to-many
- The relationship between the Promotions and Servicess tables is one-to-one
- The relationship between the Contracts and Customers tables is many-to-many
- The relationship between the Contracts and Servicess tables is many-to-many
- The relationship between the ServicesOffices and Technicians tables is one-to-many

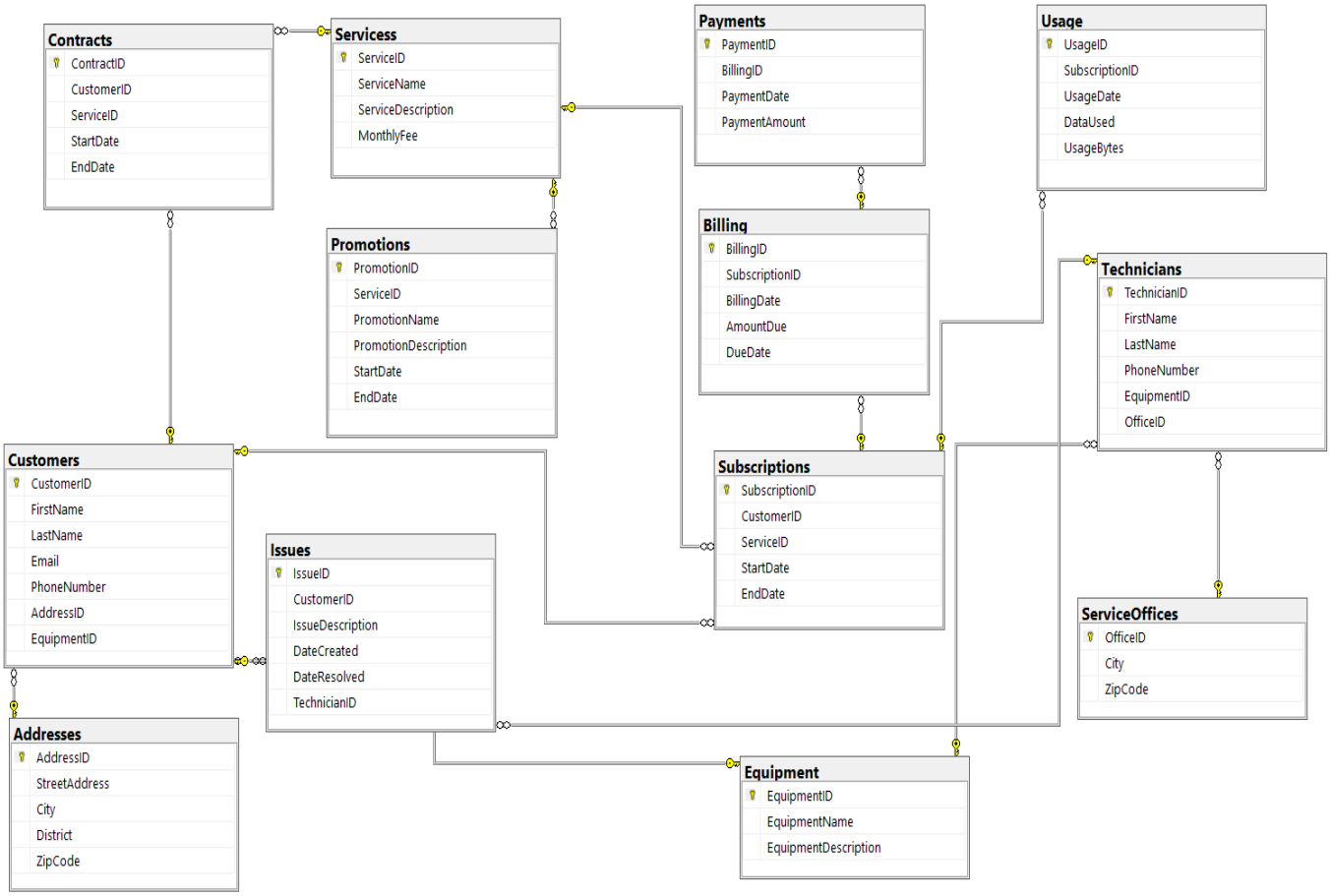
Step 4- Identify Attributes

Addresses: AddressID, StreetAddress, City, District, and ZipCode.
Customers: CustomerID, FirstName, LastName, Email, PhoneNumber, and AddressID
Servicess: ServiceID, ServiceName, ServiceDescription, and MonthlyFee**.**
Subscriptions: SubscriptionID, CustomerID, ServiceID, StartDate, and EndDate.
Billing: BillingID, SubscriptionID, BillingDate, AmountDue, and DueDate.
Payments: PaymentID, BillingID, PaymentDate, and PaymentAmount.
Usage: UsageID, SubscriptionID, UsageDate, and DataUsed.
Technicians: TechnicianID, FirstName, LastName, PhoneNumber, ServiceID
Issues: IssueID, CustomerID, IssueDescription, DateCreated, DateResolved
Equipment: EquipmentID, EquipmentName, EquipmentDescription
ServiceOffices: OfficeID, City, ZipCode
Promotions: PromotionID, ServiceID, PromotionName, PromotionDescriptio, StartDate, EndDate
Contracts: ContractID, CustomerID, ServiceID, StartDate, EndDate

Step 5- Create the ERD Diagram:



Step 6- Convert ERD to Tables in DBMS:



Sr No	Topic	Queries
1.	CREATE TABLE Statement	**10**
2.	PRIMARY KEY and FOREIGN KEY	**10**
3.	AUTO INCREMENT	**10**
4.	ALTER TABLE Statement (ADD Column, MODIFY DATATYPE, RENAME COLUMN, DROP COLUMN)	**50**
5.	INSERT INTO Statement	**10**
6.	SELECT and DISTINCT Statement	**20**
7.	WHERE Clause using AND, OR and NOT Operators	**50**
8.	ORDER BY Statement	**25**
9.	ORDER BY using AND, OR and NOT Operators	**25**
10.	GROUP BY Statement	**25**
11.	GROUP BY using AND, OR, NOT Operators and Group by	**25**
12.	Subqueries	**30**
13.	Subqueries using logical Operators	**30**
14.	Aggregate functions MAX, MIN, SUM, COUNT, and AVG.	**20**
15.	Aggregate functions using logical Operators and Group by	**30**
16.	INNER Joins	**20**
17.	INNER Joins using logical Operators, Group by and Order by	**30**
18.	LEFT JOIN	**20**
19.	RIGHT JOIN	**20**
20.	FULL OUTER JOIN	**20**
21.	Stored Procedures without parameter	**25**
22.	Stored Procedures with parameter	**25**
23.	Stored Procedures with parameter using logical Operators and Group by	**30**
24.	DML Triggers INSERT	**20**
25.	DML Triggers UPDATE	**20**
26.	DML Triggers DELETE	**20**
27.	Single-Row Functions UPPER, LOWER, LENGTH, SUBSTR using logical operators	**50**
28.	Single-Row Functions TRIM, REPLACE, ROUND, TRUNC using logical operators	**50**
29.	Transaction COMMIT and ROLLBACK	**20**
30.	Exception handling - Try Catch	**20**

1. CREATE TABLE Statement – 10 Queries

1	Create Addresses table	CREATE TABLE Addresses (AddressID INT PRIMARY KEY IDENTITY(1,1), StreetAddress VARCHAR(255), City VARCHAR(255), District VARCHAR(255), ZipCode VARCHAR(255));
2	Create Customers table	CREATE TABLE Customers (CustomerID INT PRIMARY KEY IDENTITY(1,1), FirstName VARCHAR(255), LastName VARCHAR(255), Email VARCHAR(255), PhoneNumber VARCHAR(255), AddressID INT FOREIGN KEY REFERENCES Addresses(AddressID), EquipmentID INT Foreign KEY REFERENCES Equipment(EquipmentID));
3	Create Servicess table	CREATE TABLE Servicess (ServiceID INT PRIMARY KEY IDENTITY(1,1), ServiceName VARCHAR(255), ServiceDescription VARCHAR(255), MonthlyFee float);
4	Create Subscriptions table	CREATE TABLE Subscriptions (SubscriptionID INT PRIMARY KEY IDENTITY(1,1), CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID), ServiceID INT FOREIGN KEY REFERENCES Servicess(ServiceID), StartDate DATE, EndDate DATE);
5	Create Billing table	CREATE TABLE Billing (BillingID INT PRIMARY KEY IDENTITY(1,1), SubscriptionID INT FOREIGN KEY REFERENCES Subscriptions(SubscriptionID), BillingDate DATE, AmountDue float, DueDate DATE);
6	Create Payments table	CREATE TABLE Payments (PaymentID INT PRIMARY KEY IDENTITY(1,1), BillingID INT FOREIGN KEY REFERENCES Billing(BillingID), PaymentDate DATE, PaymentAmount float);
7	Create Usage table	CREATE TABLE Usage (UsageID INT PRIMARY KEY IDENTITY(1,1),

		SubscriptionID INT FOREIGN KEY REFERENCES Subscriptions(SubscriptionID), UsageDate DATE, DataUsed float, UsageBytes varchar(255));
8	Create Technicians table	CREATE TABLE Technicians (TechnicianID INT PRIMARY KEY IDENTITY(1,1), FirstName VARCHAR(255), LastName VARCHAR(255), PhoneNumber VARCHAR(255), EquipmentID INT Foreign KEY REFERENCES Equipment(EquipmentID), OfficeID INT FOREIGN KEY REFERENCES ServiceOffices(OfficeID));
9	Create ServiceOffices table	CREATE TABLE ServiceOffices (OfficeID INT PRIMARY KEY IDENTITY (1,1), City VARCHAR(255), ZipCode VARCHAR(255));
10	Create Promotions table	CREATE TABLE Promotions (PromotionID INT PRIMARY KEY IDENTITY(1,1), ServiceID INT FOREIGN KEY REFERENCES Servicess(ServiceID), PromotionName VARCHAR(255), PromotionDescription VARCHAR(255), StartDate DATE, EndDate DATE);

2. PRIMARY KEY and FOREIGN KEY – 10 Queries

1	CustomerID is Primary Key in Customers Table	CustomerID INT PRIMARY KEY
2	ServiceID is Primary Key in Servicess Table	ServiceID INT PRIMARY KEY
3	ContractID is Primary Key in Contracts Table	ContractID INT PRIMARY KEY
4	AddressID is Primary Key in Addresses Table	AddressID INT PRIMARY KEY
5	TechnicianID is Primary Key in Technicians Table	TechnicianID INT PRIMARY KEY
6	AddressID is Foreign Key in Customers Table	AddressID INT FOREIGN KEY REFERENCES Addresses(AddressID),
7	CustomerID is Foreign Key in Subscriptions Table	CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID),
8	BillingID is Foreign Key in Payments Table	BillingID INT FOREIGN KEY REFERENCES Billing(BillingID),
9	TechnicianID is Foreign Key in Issues Table	TechnicianID INT FOREIGN KEY REFERENCES Technicians(TechnicianID)

10	ServiceID is Foreign Key in Contracts Table	`ServiceID INT FOREIGN KEY REFERENCES Servicess(ServiceID),`

3. AUTO INCREMENT – 10 Queries

1	CustomerID has auto increment value	`CustomerID INT PRIMARY KEY IDENTITY(1,1),`
2	ContractID has auto increment value	`ContractID INT PRIMARY KEY IDENTITY(1,1),`
3	PromotionID has auto increment value	`PromotionID INT PRIMARY KEY IDENTITY(1,1),`
4	OfficeID has auto increment value	`OfficeID INT PRIMARY KEY IDENTITY (1,1),`
5	AddressID has auto increment value	`AddressID INT PRIMARY KEY IDENTITY(1,1),`
6	TechnicianID has auto increment value	`TechnicianID INT PRIMARY KEY IDENTITY(1,1),`
7	IssueID has auto increment value	`IssueID INT PRIMARY KEY IDENTITY(1,1),`
8	UsageID has auto increment value	`UsageID INT PRIMARY KEY IDENTITY(1,1),`
9	PaymentID has auto increment value	`PaymentID INT PRIMARY KEY IDENTITY(1,1),`
10	SubscriptionID has auto increment value	`SubscriptionID INT PRIMARY KEY IDENTITY(1,1),`

4. ALTER TABLE Statement (ADD Column, MODIFY DATATYPE, RENAME COLUMN, DROP COLUMN) – 50 Queries

1	Drop PhoneNumber Column from Customers table	`ALTER TABLE Customers DROP COLUMN PhoneNumber;`
2	Drop MonthlyFee Column from Services table	`ALTER TABLE Servicess DROP COLUMN MonthlyFee;`
3	Drop PaymentAmount Column from Payments table	`ALTER TABLE Payments DROP COLUMN PaymentAmount;`
4	Drop AmountDue Column from Billing table	`ALTER TABLE Billing DROP COLUMN AmountDue;`
5	Drop District Column from Addresses table	`ALTER TABLE Addresses DROP COLUMN District;`
6	Drop Email Column from Customers table	`ALTER TABLE Customers DROP COLUMN Email;`
7	Drop IssueDescription Column from Issue table	`ALTER TABLE Issues DROP COLUMN IssueDescription;`
8	Drop UsageBytes Column from Usage table	`ALTER TABLE Usage DROP COLUMN UsageBytes;`

9	Drop DataUsed Column from Usage table	`ALTER TABLE Usage` `DROP COLUMN DataUsed;`
10	Drop EquipmentDescription Column from Equipment table	`ALTER TABLE Equipment` `DROP COLUMN EquipmentDescription;`
11	Drop StartDate Column from Promotions table	`ALTER TABLE Promotions` `DROP COLUMN StartDate;`
12	Drop ZipCode Column from ServiceOffices table	`ALTER TABLE ServiceOffices` `DROP COLUMN ZipCode;`
13	Drop ServiceName Column from Servicess table	`ALTER TABLE Servicess` `DROP COLUMN ServiceName;`
14	Drop City Column from Addresses table	`ALTER TABLE Addresses` `DROP COLUMN City;`
15	Drop StartDate Column from Contracts	`ALTER TABLE Contracts` `DROP COLUMN StartDate;`
16	Add PhoneNumber Column in Customers tables	`ALTER TABLE Customers` `ADD PhoneNumber INT;`
17	Add MonthlyFee Column in Services tables	`ALTER TABLE Servicess` `ADD MonthlyFee float;`
18	Add PaymentAmount Column in Payments tables	`ALTER TABLE Payments` `ADD PaymentAmount float;`
19	Add AmountDue Column in Billing tables	`ALTER TABLE Billing` `ADD AmountDue float;`
20	Add District Column in Addresses tables	`ALTER TABLE Addresses` `ADD District VARCHAR(255);`
21	Add Email Column in Customers table	`ALTER TABLE Customers` `ADD Email VARCHAR(255);`
22	Add IssueDescription Column in Issues table	`ALTER TABLE Issues` `ADD IssueDescription VARCHAR(255);`
23	Add UsageBytes Column in Usage table	`ALTER TABLE Usage` `ADD UsageBytes varchar(255);`
24	Add DataUsed Column in Usage table	`ALTER TABLE Usage` `ADD DataUsed FLOAT;`
25	Add EquipmentDescription Column in Equipment table	`ALTER TABLE Equipment` `ADD EquipmentDescription VARCHAR(255);`
26	Add StartDate Column in Promotions table	`ALTER TABLE Promotions` `ADD StartDate DATE;`
27	Add ZipCode Column in ServiceOffices table	`ALTER TABLE ServiceOffices` `ADD ZipCode VARCHAR(255);`
28	Add ServiceName Column in Servicess table	`ALTER TABLE Servicess` `ADD ServiceName VARCHAR(255);`
29	Add City Column in Addresses table	`ALTER TABLE Addresses` `ADD City VARCHAR(255);`
30	Add StartDate Column in Contracts table	`ALTER TABLE Contracts` `ADD StartDate DATE;`

31	Change the datatype of column ZipCode from ServiceOffices table	`ALTER TABLE ServiceOffices` `ALTER COLUMN ZipCode INT;`
32	Change the datatype of column PhoneNumber from Customers table	`ALTER TABLE Customers` `ALTER COLUMN PhoneNumber INT;`
33	Change the datatype of column PaymentAmount from Payments table	`ALTER TABLE Payments` `ALTER COLUMN PaymnetAmount INT;`
34	Change the datatype of column StartDate from Promotions table	`ALTER TABLE Promotions` `ALTER COLUMN StartDate DATETIME;`
35	Change the datatype of column AmountDue from Billing table	`ALTER TABLE  Billing` `ALTER COLUMN AmountDue INT;`
36	Change the datatype of column DataUsed from Usage table	`ALTER TABLE Usage` `ALTER COLUMN DataUsed Decimal;`
37	Change the datatype of column MonthlyFee from Servicess table	`ALTER TABLE  Servicess` `ALTER COLUMN MonthlyFee Decimal;`
38	Change the datatype of column IssueDescription from Issues table	`ALTER TABLE Issues` `ALTER COLUMN IssueDescription VARCHAR(MAX);`
39	Change the datatype of column UsageBytesfrom Usage table	`ALTER TABLE Usage` `ALTER COLUMN UsageBytes FLOAT;`
40	Change the datatype of column PhoneNumber from Technicians table	`ALTER TABLE Technicians` `ALTER COLUMN PhoneNumber INT;`
41	Rename the column PhoneNumber to CellNumber from Customers Table	`exec sp_rename 'Customers.PhoneNumber', 'CellNumber', 'COLUMN';`
42	Rename the column ServiceID to S_IDfrom Servicess Table	`exec sp_rename 'Servicess.ServiceID', 'S_ID', 'COLUMN';`
43	Rename the column StartDate to DateStarted from Subscriptions Table	`exec sp_rename 'Subscriptions.StartDate', 'DateStarted', 'COLUMN';`
44	Rename the column BillingDate Date_Of_billing from Billing Table	`exec sp_rename 'Billing.BillingDate', 'Date_Of_Billing', 'COLUMN';`
45	Rename the column PaymentAmount to	`exec sp_rename 'Payments.PaymentAmount', 'P_Amount', 'COLUMN';`

	P_Amount from Payments Table	
46	Rename the column UsageBytes to TotalBytesUsed from Usage Table	`exec sp_rename 'Usage.UsageBytes', 'TotalBytesUsed', 'COLUMN';`
47	Rename the column DateResolved to DateOfIssueResolved from Issues Table	`exec sp_rename 'Issues.DateResolved', 'DateOfIssueResolved', 'COLUMN';`
48	Rename the column FirstName to TechnicianFirstName from Technicians Table	`exec sp_rename 'Technicians.FirstName', 'TechnicianFirstName', 'COLUMN';`
49	Rename the column ZipCode to PostalCode from ServiceOffices Table	`exec sp_rename 'ServiceOffices.ZipCode', 'PostalCode', 'COLUMN';`
50	Rename the column EndDate to EndDateOfPromotion from Promotions Table	`exec sp_rename 'Promotions.EndDate', 'EndDateOfPromotion', 'COLUMN';`

5. INSERT INTO Statement – 10 Queries

1	Inserting values in Customers Table	`INSERT INTO Customers(FirstName,LastName,Email,PhoneNumber,AddressID,EquipmentID) VALUES('Ali','Khan','Ali@gmail.com',0300965, 1,1)`
2	Inserting values in Servicess Table	`INSERT INTO Servicess(ServiceName,ServiceDescription,MonthlyFee) VALUES('XYZ','4Mb Network',2000)`
3	Inserting values in Subscriptions Table	`INSERT INTO Subscriptions(CustomerID,ServiceID,StartDate,EndDate) VALUES(1,1,GETDATE(),GETDATE())`
4	Inserting values in Billing Table	`INSERT INTO Billing(SubscriptionID,BillingDate,AmountDue,DueDate) VALUES(1,GETDATE(),2000,GETDATE())`
5	Inserting values in Payments Table	`INSERT INTO Payments(BillingID,PaymentDate,PaymentAmount) VALUES(1,GETDATE(),2000)`
6	Inserting values in Usage Table	`INSERT INTO Usage(SubscriptionID,UsageDate,DataUsed,UsageBytes) VALUES(1,GETDATE(),24,'GB')`
7	Inserting values in Technicians Table	`INSERT INTO Technicians(FirstName,LastName,PhoneNumber,EquipmentID,OfficeID) VALUES('Kamran','Ali',0300213213,1,1)`
8	Inserting values in Addresses Table	`INSERT INTO Addresses(StreetAddress,City,District,ZipCode ) VALUES('Street ABC','Bhalwal','Sargodha',40404)`
9	Inserting values in ServiceOffices Table	`INSERT INTO ServiceOffices(City,ZipCode) VALUES('Lahore',40410)`

10	Inserting values in Contracts Table	INSERT INTO Contracts(CustomerID,ServiceID,StartDate,EndDate) VALUES(1,1,GETDATE(),GETDATE())

6. SELECT and DISTINCT Statement – 20 Queries

1	displays all the information about Customers	Select * from Customers
2	Selects only the FirstName and LastName columns from the Customers table.	SELECT FirstName, LastName FROM Customers;
3	Selects all columns and rows from the Servicess table where the MonthlyFee is greater than 2000.	SELECT * FROM Servicess WHERE MonthlyFee > 2000;
4	Returns the total number of rows in the Subscriptions table.	SELECT COUNT(*) FROM Subscriptions;
5	Returns the average value of the MonthlyFee column in the Servicess table.	SELECT AVG(MonthlyFee) FROM Servicess;
6	Selects all columns and rows from the ServiceOffices table where the City is equal to 'Lahore'.	SELECT * FROM ServiceOffices WHERE City = 'Lahore';
7	Selects all columns and rows from the Customers table where the PhoneNumber ends with '5'.	SELECT * FROM Customers WHERE PhoneNumber LIKE '%5';
8	Selects all columns and rows from the Issues table where the DateResolved column is null	SELECT * FROM Issues WHERE DateResolved IS NULL;
9	Selects all columns and rows from the Issues table where the TechnicianID is equal to 2.	SELECT * FROM Issues WHERE TechnicianID = 2;
10	Returns the total sum of the PaymentAmount column in the Payments table for payments made between January 1st, 2023 and December 31st, 2023.	SELECT SUM(PaymentAmount) FROM Payments WHERE PaymentDate BETWEEN '2023-01-01' AND '2023-12-31';

11	Selects distinct values of the LastName column from the Technicians table.	`SELECT DISTINCT LastName FROM Technicians;`
12	Selects distinct values of the City column from the Addresses table.	`SELECT DISTINCT City FROM Addresses;`
13	Returns the count of distinct values in the City column of the Addresses table.	`SELECT AVG(DISTINCT PaymentAmount) FROM Payments;`
14	Returns the sum of distinct values in the MonthlyFee column of the Servicess table	`SELECT SUM(DISTINCT MonthlyFee) FROM Servicess;`
15	Returns the count of distinct values in the City column of the Addresses table.	`SELECT COUNT(DISTINCT City) FROM Addresses;`
16	Selects distinct values of the City column from the Addresses table where the ZipCode is equal to '40410' or '40404'	`SELECT DISTINCT City FROM Addresses WHERE ZipCode = '40410' OR ZipCode = '40404';`
17	Selects distinct values of the StartDate column from the Subscriptions table where the EndDate column is null	`SELECT DISTINCT StartDate FROM Subscriptions WHERE EndDate IS NULL;`
18	Selects distinct values of the ZipCode column from the ServiceOffices table where the City is equal to 'Lahore' or 'Karachi'	`SELECT DISTINCT ZipCode FROM ServiceOffices WHERE City = 'Lahore' OR City = 'Karachi';`
19	Selects distinct values of the PhoneNumber column from the Customers table where the FirstName is equal to 'Ali' and the LastName is equal to 'Khan'	`SELECT DISTINCT PhoneNumber FROM Customers WHERE FirstName = 'Ali' AND LastName = 'Khan';`
20	Selects distinct values of the BillingDate column from the Billing table where the AmountDue is greater than 4000	`SELECT DISTINCT BillingDate FROM Billing WHERE AmountDue > 4000;`

7. WHERE Clause using AND, OR and NOT Operators– 50 Queries

1	Find the names of customers who live in a city that starts with the letter 'A'	`SELECT FirstName, LastName FROM Customers WHERE City LIKE 'A%';`

2	Find the names of services that have a monthly fee greater than 4000	SELECT ServiceName FROM Servicess WHERE MonthlyFee > 4000;
3	Find the subscription IDs that have a start date after '2022-01-01'	SELECT SubscriptionID FROM Subscriptions WHERE StartDate > '2022-01-01';
4	Find the billing IDs that have a billing date in the year 2022	SELECT BillingID FROM Billing WHERE YEAR(BillingDate) = 2022;
5	Find the subscription IDs that have an end date before '2024-01-01'	SELECT SubscriptionID FROM Subscriptions WHERE EndDate < '2024-01-01';
6	Find the names of customers who live in a city that does not start with the letter 'A'	SELECT FirstName, LastName FROM Customers WHERE NOT City LIKE 'A%';
7	Find the names of services that have a monthly fee less than or equal to 3000	SELECT ServiceName FROM Servicess WHERE MonthlyFee <= 3000;
8	Find the subscription IDs that have a start date on or before '2022-01-01'	SELECT SubscriptionID FROM Subscriptions WHERE StartDate <= '2022-01-01';
9	Find the billing IDs that have a billing date not in the year 2022	SELECT BillingID FROM Billing WHERE NOT YEAR(BillingDate) = 2022;
10	Find the subscription IDs that have an end date after '2024-01-01'	SELECT SubscriptionID FROM Subscriptions WHERE EndDate > '2024-01-01';
11	Find the names of customers who have a subscription and live in a city that starts with the letter 'A'	SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions) AND City LIKE 'A%';
12	Find the names of services that have a subscription and a monthly fee greater than 3000	SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions) AND MonthlyFee > 50;
13	Find the subscription IDs that have a billing record and a start date after '2022-01-01'	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Billing) AND StartDate > '2022-01-01';
14	Find the billing IDs that have a payment record and a billing date in the year 2022	SELECT BillingID FROM Billing WHERE BillingID IN (SELECT BillingID FROM Payments) AND YEAR(BillingDate) = 2022;
15	Find the subscription IDs that have a usage record and	SELECT SubscriptionID FROM Subscriptions

	an end date before ‘2023-01-01’	WHERE SubscriptionID IN (SELECT SubscriptionID FROM Usage) AND EndDate < '2023-01-01';
16	Find the customer IDs that have an issue record and live in a city that starts with the letter ‘A’	SELECT CustomerID FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Issues) AND City LIKE 'A%';
17	Find the names of customers who live in a city that starts with the letter ‘A’ and have a phone number starting with ‘123’	SELECT FirstName, LastName FROM Customers WHERE City LIKE 'A%' AND PhoneNumber LIKE '123%';
18	Find the names of services that have a monthly fee greater than 4000 and service description containing the word ‘fast’	SELECT ServiceName FROM Servicess WHERE MonthlyFee > 4000 AND ServiceDescription LIKE '%fast%';
19	Find the subscription IDs that have a start date after ‘2022-01-01’ and an end date before ‘2023-01-01’	SELECT SubscriptionID FROM Subscriptions WHERE StartDate > '2022-01-01' AND EndDate < '2023-01-01';
20	Find the billing IDs that have a billing date in the year 2022 and an amount due greater than 3000	SELECT BillingID FROM Billing WHERE YEAR(BillingDate) = 2022 AND AmountDue > 3000;
21	Find the names of customers who have a subscription or live in a city that starts with the letter ‘A’	SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions) OR City LIKE 'A%';
22	Find the names of services that have a subscription or have a monthly fee greater than 2000	SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions) OR MonthlyFee > 2000;
23	Find the subscription IDs that have a billing record or have a start date after ‘2022-01-01’	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Billing) OR StartDate > '2022-01-01';
24	Find the billing IDs that have a payment record or have a billing date in the year 2022	SELECT BillingID FROM Billing WHERE BillingID IN (SELECT BillingID FROM Payments) OR YEAR(BillingDate) = 2022;
25	Find the subscription IDs that have a usage record or have an end date before ‘2023-01-01’	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Usage) OR EndDate < '2023-01-01';

26	Find the customer IDs that have an issue record or live in a city that starts with the letter ‘A’	SELECT CustomerID FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Issues) OR City LIKE 'A%';
27	Find the names of customers who live in a city that starts with the letter ‘A’ or have a phone number starting with ‘123’	SELECT FirstName, LastName FROM Customers WHERE City LIKE 'A%' OR PhoneNumber LIKE '123%';
28	Find the names of services that have a monthly fee greater than 3000 or service description containing the word ‘fast’	SELECT ServiceName FROM Servicess WHERE MonthlyFee > 3000 OR ServiceDescription LIKE '%fast%';
29	Find the subscription IDs that have a start date after ‘2022-01-01’ or an end date before ‘2023-01-01’	SELECT SubscriptionID FROM Subscriptions WHERE StartDate > '2022-01-01' OR EndDate < '2023-01-01';
30	Find the billing IDs that have a billing date in the year 2022 or an amount due greater than 2000	SELECT BillingID FROM Billing WHERE YEAR(BillingDate) = 2022 OR AmountDue > 2000;
31	Find the names of customers who do not have a subscription	SELECT FirstName, LastName FROM Customers WHERE CustomerID NOT IN (SELECT CustomerID FROM Subscriptions);
32	Find the names of services that do not have a subscription	SELECT ServiceName FROM Servicess WHERE ServiceID NOT IN (SELECT ServiceID FROM Subscriptions);
33	Find the subscription IDs that do not have a billing record	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID NOT IN (SELECT SubscriptionID FROM Billing);
34	Find the billing IDs that do not have a payment record	SELECT BillingID FROM Billing WHERE BillingID NOT IN (SELECT BillingID FROM Payments);
35	Find the subscription IDs that do not have a usage record	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID NOT IN (SELECT SubscriptionID FROM Usage);
36	Find the customer IDs that do not have an issue record	SELECT CustomerID FROM Customers WHERE CustomerID NOT IN (SELECT CustomerID FROM Issues);
37	Find the names of customers who do not live in a city that starts with the letter ‘A’	SELECT FirstName, LastName FROM Customers WHERE NOT City LIKE 'A%';

38	Find the names of services that do not have a monthly fee greater than 2000	`SELECT ServiceName FROM Servicess WHERE NOT MonthlyFee > 2000;`
39	Find the subscription IDs that do not have a start date after '2022-01-01'	`SELECT SubscriptionID FROM Subscriptions WHERE NOT StartDate > '2022-01-01';`
40	Find the billing IDs that do not have a billing date in the year 2022	`SELECT BillingID FROM Billing WHERE NOT YEAR(BillingDate) = 2022;`
41	Find the names of services that have a monthly fee between 2000 and 4000	`SELECT ServiceName FROM Servicess WHERE MonthlyFee BETWEEN 2000 AND 4000;`
42	Find the names of customers who live in a city that contains the word 'Lahore'	`SELECT FirstName, LastName FROM Customers WHERE City LIKE '%Lahore%';`
43	Find the customer IDs that live in a city that ends with the letter 'n'	`SELECT CustomerID FROM Customers WHERE City LIKE '%n';`
44	Find the billing IDs that have a billing date in December	`SELECT BillingID FROM Billing WHERE MONTH(BillingDate) = 12;`
45	Find the subscription IDs that have a start date before '2021-12-31'	`SELECT SubscriptionID FROM Subscriptions WHERE StartDate < '2021-12-31'`
46	Find the names of services that have a monthly fee less than 2000	`SELECT ServiceName FROM Servicess WHERE MonthlyFee < 2000;`
47	Find the names of customers who live in a city that ends with the letter 'e'	`SELECT FirstName, LastName FROM Customers WHERE City LIKE '%e';`
48	Find the customer IDs that do not live in a city that starts with the letter 'A'	`SELECT CustomerID FROM Customers WHERE NOT City LIKE 'A%';`
49	Find the subscription IDs that have an end date on or after '2023-01-01'	`SELECT SubscriptionID FROM Subscriptions WHERE EndDate >= '2023-01-01';`
50	Find the customer IDs that live in a city that starts with the letter 'A'	`SELECT CustomerID FROM Customers WHERE City LIKE 'A%';`

8. ORDER BY Statement– 25 Queries

1	Find the names of customers in ascending order by last name	`SELECT FirstName, LastName FROM Customers ORDER BY LastName ASC;`

2	Find the names of services in descending order by monthly fee	`SELECT ServiceName`
		`FROM Servicess`
		`ORDER BY MonthlyFee DESC;`

3	Find the subscription IDs in ascending order by start date	`SELECT SubscriptionID`
		`FROM Subscriptions`
		`ORDER BY StartDate ASC;`

4	Find the billing IDs in descending order by billing date	`SELECT BillingID`
		`FROM Billing`
		`ORDER BY BillingDate DESC;`

5	Find the subscription IDs in ascending order by end date	`SELECT SubscriptionID`
		`FROM Subscriptions`
		`ORDER BY EndDate ASC;`

6	Find the customer IDs in descending order by city	`SELECT CustomerID`
		`FROM Customers`
		`ORDER BY City DESC;`

7	Find the names of customers in ascending order by first name and last name	`SELECT FirstName, LastName`
		`FROM Customers`
		`ORDER BY FirstName ASC, LastName ASC;`

8	Find the names of services in descending order by service name and monthly fee	`SELECT ServiceName`
		`FROM Servicess`
		`ORDER BY ServiceName DESC, MonthlyFee DESC;`

9	Find the subscription IDs in ascending order by start date and end date	`SELECT SubscriptionID`
		`FROM Subscriptions`
		`ORDER BY StartDate ASC, EndDate ASC;`

10	Find the billing IDs in descending order by billing date and amount due	`SELECT BillingID`
		`FROM Billing`
		`ORDER BY BillingDate DESC, AmountDue DESC;`

11	Find the subscription IDs in ascending order by end date and monthly fee	`SELECT SubscriptionID`
		`FROM Subscriptions`
		`ORDER BY EndDate ASC, MonthlyFee ASC;`

12	Find the customer IDs in descending order by city and last name	`SELECT CustomerID`
		`FROM Customers`
		`ORDER BY City DESC, LastName DESC;`

13	Find the names of customers in ascending order by phone number	`SELECT FirstName, LastName`
		`FROM Customers`
		`ORDER BY PhoneNumber ASC;`

14	-Find the names of services in descending order by service description	`SELECT ServiceName`
		`FROM Servicess`
		`ORDER BY ServiceDescription DESC;`

15	Find the subscription IDs in ascending order by monthly fee and start date	`SELECT SubscriptionID`
		`FROM Subscriptions`
		`ORDER BY MonthlyFee ASC, StartDate ASC;`

16	Find the billing IDs in descending order by amount due and billing date	`SELECT BillingID`
		`FROM Billing`
		`ORDER BY AmountDue DESC, BillingDate DESC;`

17	Find the subscription IDs in ascending order by monthly fee and end date	`SELECT SubscriptionID FROM Subscriptions ORDER BY MonthlyFee ASC, EndDate ASC;`
18	Find the customer IDs in descending order by last name and city	`SELECT CustomerID FROM Customers ORDER BY LastName DESC, City DESC;`
19	Find the names of customers in ascending order by city and phone number	`SELECT FirstName, LastName FROM Customers ORDER BY City ASC, PhoneNumber ASC;`
20	Find the names of services in descending order by monthly fee and service description	`SELECT ServiceName FROM Servicess ORDER BY MonthlyFee DESC, ServiceDescription DESC;`
21	Find the subscription IDs in ascending order by start date and monthly fee	`SELECT SubscriptionID FROM Subscriptions ORDER BY StartDate ASC, MonthlyFee ASC;`
22	Find the billing IDs in descending order by billing date and amount due	`SELECT BillingID FROM Billing ORDER BY BillingDate DESC, AmountDue DESC;`
23	Find the subscription IDs in ascending order by end date and start date	`SELECT SubscriptionID FROM Subscriptions ORDER BY EndDate ASC, StartDate ASC;`
24	Find the customer IDs in descending order by city and first name	`SELECT CustomerID FROM Customers ORDER BY City DESC, FirstName DESC;`
25	Find the names of customers in ascending order by last name and phone number	`SELECT FirstName, LastName FROM Customers ORDER BY LastName ASC, PhoneNumber ASC;`

9. ORDER BY using AND, OR and NOT Operators– 25 Queries

1	Find the names of customers who have a subscription and live in a city that starts with the letter 'A', in ascending order by last name	`SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions) AND City LIKE 'A%' ORDER BY LastName ASC;`
2	Find the names of services that have a subscription and a monthly fee greater than 50, in descending order by monthly fee	`SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions) AND MonthlyFee > 50 ORDER BY MonthlyFee DESC;`
3	Find the subscription IDs that have a billing record and a start date after '2022-01-01', in ascending order by start date	`SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Billing) AND StartDate > '2022-01-01'`

		`ORDER BY StartDate ASC;`

4	Find the billing IDs that have a payment record and a billing date in the year 2022, in descending order by billing date	`SELECT BillingID`
		`FROM Billing`
		`WHERE BillingID IN (SELECT BillingID FROM Payments)`
		`AND YEAR(BillingDate) = 2022`
		`ORDER BY BillingDate DESC;`

5	Find the subscription IDs that have a usage record and an end date before '2023-01-01', in ascending order by end date	`SELECT SubscriptionID`
		`FROM Subscriptions`
		`WHERE SubscriptionID IN (SELECT SubscriptionID FROM Usage)`
		`AND EndDate < '2023-01-01'`
		`ORDER BY EndDate ASC;`

6	Find the customer IDs that have an issue record and live in a city that starts with the letter 'A', in descending order by city	`SELECT CustomerID`
		`FROM Customers`
		`WHERE CustomerID IN (SELECT CustomerID FROM Issues)`
		`AND City LIKE 'A%'`
		`ORDER BY City DESC;`

7	Find the names of customers who live in a city that starts with the letter 'A' and have a phone number starting with '123', in ascending order by first name and last name	`SELECT FirstName, LastName`
		`FROM Customers`
		`WHERE City LIKE 'A%'`
		`AND PhoneNumber LIKE '123%'`
		`ORDER BY FirstName ASC, LastName ASC;`

8	Find the names of services that have a monthly fee greater than 3000 and service description containing the word 'fast', in descending order by service name and monthly fee	`SELECT ServiceName`
		`FROM Servicess`
		`WHERE MonthlyFee > 3000`
		`AND ServiceDescription LIKE '%fast%'`
		`ORDER BY ServiceName DESC, MonthlyFee DESC;`

9	Find the subscription IDs that have a start date after '2022-01-01' and an end date before '2023-01-01', in ascending order by start date and end date	`SELECT SubscriptionID`
		`FROM Subscriptions`
		`WHERE StartDate > '2022-01-01'`
		`AND EndDate < '2023-01-01'`
		`ORDER BY StartDate ASC, EndDate ASC;`

10	Find the billing IDs that have a billing date in the year 2022 and an amount due greater than 2000, in descending order by billing date and amount due	`SELECT BillingID`
		`FROM Billing`
		`WHERE YEAR(BillingDate) = 2022`
		`AND AmountDue > 2000`
		`ORDER BY BillingDate DESC, AmountDue DESC;`

11	This query selects all customers from the Customers table where the City is either 'New York' or 'Los Angeles' and orders the results by the LastName column.	`SELECT * FROM Customers WHERE City = 'New York'` `OR City = 'Los Angeles' ORDER BY LastName;`
12	This query selects all subscriptions from the Subscriptions table where the StartDate is on or after '2022-	`SELECT * FROM Subscriptions WHERE StartDate >= '2022-01-01' OR EndDate <= '2022-12-31' ORDER BY StartDate;`

	01-01’ or the EndDate is on or before ‘2022-12-31’ and orders the results by the StartDate column.	
13	This query selects all billing records from the Billing table where the AmountDue is greater than 100 or the DueDate is before ‘2022-12-31’ and orders the results by the AmountDue column.	`SELECT * FROM Billing WHERE AmountDue > 100 OR DueDate < '2022-12-31' ORDER BY AmountDue;`
14	This query selects all payments from the Payments table where the PaymentAmount is greater than 100 or the PaymentDate is before ‘2022-12-31’ and orders the results by the PaymentAmount column.	`SELECT * FROM Payments WHERE PaymentAmount > 100 OR PaymentDate < '2022-12-31' ORDER BY PaymentAmount;`
15	This query selects all usage records from the Usage table where the DataUsed is greater than 1000 or the UsageDate is before ‘2022-12-31’ and orders the results by the DataUsed column.	`SELECT * FROM Usage WHERE DataUsed > 1000 OR UsageDate < '2022-12-31' ORDER BY DataUsed;`
16	This query selects all technicians from the Technicians table where the FirstName is ‘John’ or the LastName is ‘Doe’ and orders the results by the LastName column.	`SELECT * FROM Technicians WHERE FirstName = 'John' OR LastName = 'Doe' ORDER BY LastName;`
17	This query selects all issues from the Issues table where the DateResolved is null (i.e., unresolved) or the DateCreated is on or after ‘2022-01-01’ and orders the results by the DateCreated column.	`SELECT * FROM Issues WHERE DateResolved IS NULL OR DateCreated >= '2022-01-01' ORDER BY DateCreated;`
18	This query selects all equipment from the Equipment table where the EquipmentName is ‘Router’ or the EquipmentDescription contains	`SELECT * FROM Equipment WHERE EquipmentName = 'Router' OR EquipmentDescription LIKE '%wireless%' ORDER BY EquipmentName;`

	‘wireless’ and orders the results by the EquipmentName column.	
19	This query selects all service offices from the ServiceOffices table where the City is ‘New York’ or the ZipCode is ‘10001’ and orders the results by the City column.	`SELECT * FROM ServiceOffices WHERE City = 'New York' OR ZipCode = '10001' ORDER BY City;`
20	This query selects all promotions from the Promotions table where the StartDate is on or after ‘2022-01-01’ or the EndDate is on or before ‘2022-12-31’ and orders the results by the StartDate column.	`SELECT * FROM Promotions WHERE StartDate >= '2022-01-01' OR EndDate <= '2022-12-31' ORDER BY StartDate;`
21	This query selects all customers from the Customers table where the City is not ‘New York’ and orders the results by the LastName column.	`SELECT * FROM Customers WHERE NOT City = 'New York' ORDER BY LastName;`
22	This query selects all subscriptions from the Subscriptions table where the StartDate is before ‘2022-01-01’ and orders the results by the StartDate column.	`SELECT * FROM Subscriptions WHERE NOT StartDate >= '2022-01-01' ORDER BY StartDate;`
23	This query selects all billing records from the Billing table where the AmountDue is less than or equal to 100 and orders the results by the AmountDue column.	`SELECT * FROM Billing WHERE NOT AmountDue > 100 ORDER BY AmountDue;`
24	This query selects all payments from the Payments table where the PaymentAmount is less than or equal to 100 and orders the results by the PaymentAmount column.	`SELECT * FROM Payments WHERE NOT PaymentAmount > 100 ORDER BY PaymentAmount;`
25	This query selects all usage records from the Usage table where the DataUsed is less than or equal to 1000 and orders the results by the DataUsed column.	`SELECT * FROM Usage WHERE NOT DataUsed > 1000 ORDER BY DataUsed;`

10. GROUP BY Statement– 25 Queries

1	To count the number of service offices in each city	`SELECT ServiceOffices.City, COUNT(ServiceOffices.OfficeID) AS OfficeCount FROM ServiceOffices GROUP BY ServiceOffices.City`
2	This query counts the number of customers in the Customers table for each City and groups the results by the City column.	`SELECT City, COUNT(*) FROM Customers GROUP BY City;`
3	This query counts the number of subscriptions in the Subscriptions table for each ServiceID and groups the results by the ServiceID column.	`SELECT ServiceID, COUNT(*) FROM Subscriptions GROUP BY ServiceID;`
4	To calculate the total data used for each usage date	`SELECT Usage.UsageDate, SUM(Usage.DataUsed) AS TotalData FROM Usage GROUP BY Usage.UsageDate`
5	To calculate the total payment amount for each payment date	`SELECT Payments.PaymentDate, SUM(Payments.PaymentAmount) AS TotalPaid FROM Payments GROUP BY Payments.PaymentDate`
6	To calculate the total amount due for each billing date	`SELECT Billing.BillingDate, SUM(Billing.AmountDue) AS TotalDue FROM Billing GROUP BY Billing.BillingDate`
7	This query calculates the total amount due in the Billing table for each BillingDate and groups the results by the BillingDate column.	`SELECT BillingDate, SUM(AmountDue) FROM Billing GROUP BY BillingDate;`
8	This query calculates the total payment amount in the Payments table for each PaymentDate and groups the results by the PaymentDate column	`SELECT PaymentDate, SUM(PaymentAmount) FROM Payments GROUP BY PaymentDate;`
9	This query calculates the total data used in the Usage table for each UsageDate and groups	`SELECT UsageDate, SUM(DataUsed) FROM Usage GROUP BY UsageDate;`

	the results by the UsageDate column.	
10	This query counts the number of issues in the Issues table for each TechnicianID and groups the results by the TechnicianID column.	`SELECT TechnicianID, COUNT(*) FROM Issues GROUP BY TechnicianID;`
11	This query counts the number of equipment in the Equipment table for each EquipmentName and groups the results by the EquipmentName column.	`SELECT EquipmentName, COUNT(*) FROM Equipment GROUP BY EquipmentName;`
12	This query counts the number of service offices in the ServiceOffices table for each City and groups the results by the City column.	`SELECT City, COUNT(*) FROM ServiceOffices GROUP BY City;`
13	This query counts the number of promotions in the Promotions table for each StartDate and groups the results by the StartDate column.	`SELECT StartDate, COUNT(*) FROM Promotions GROUP BY StartDate;`
14	This query counts the number of contracts in the Contracts table for each ServiceID and groups the results by the ServiceID column.	`SELECT ServiceID, COUNT(*) FROM Contracts GROUP BY ServiceID;`
15	This query counts the number of addresses in the Addresses table for each District and groups the results by the District column.	`SELECT District, COUNT(*) FROM Addresses GROUP BY District;`
16	This query counts the number of addresses in	`SELECT ZipCode, COUNT(*) FROM Addresses GROUP BY ZipCode;`

	the Addresses table for each ZipCode and groups the results by the ZipCode column.	
17	This query counts the number of services in the Servicess table for each ServiceName and groups the results by the ServiceName column.	`SELECT ServiceName, COUNT(*) FROM Servicess GROUP BY ServiceName;`
18	This query counts the number of services in the Servicess table for each monthly fee (MonthlyFee) and groups the results by that column.	`SELECT MonthlyFee, COUNT(*) FROM Servicess GROUP BY MonthlyFee;`
19	This query counts all subscriptions from Subscriptions table grouped by their start date (StartDate)	`SELECT StartDate, COUNT(*) FROM Subscriptions GROUP BY StartDate;`
20	This query counts all subscriptions from Subscriptions table grouped by their end date (EndDate)	`SELECT EndDate, COUNT(*) FROM Subscriptions GROUP BY EndDate;`
21	This query counts all billing records from Billing table grouped by their billing date (BillingDate)	`SELECT BillingDate, COUNT(*) FROM Billing GROUP BY BillingDate;`
22	This query counts all billing records from Billing table grouped by their due date (DueDate)	`SELECT DueDate, COUNT(*) FROM Billing GROUP BY DueDate;`
23	This query counts all payments from Payments table grouped by their payment date (PaymentDate)	`SELECT PaymentDate, COUNT(*) FROM Payments GROUP BY PaymentDate;`
24	This query calculates total data used from Usage table	`SELECT UsageBytes, SUM(DataUsed) FROM Usage GROUP BY UsageBytes;`

	grouped by usage bytes (UsageBytes)	
25	The query counts the number of subscriptions in the Subscriptions table for each CustomerID and groups the results by the CustomerID column.	`SELECT CustomerID, COUNT(*) FROM Subscriptions GROUP BY CustomerID;`

11. GROUP BY using AND, OR , NOT Operators– 25 Queries

1	This query counts the number of rows in the Addresses table for each City where the District is ‘Lahore’ and the ZipCode starts with ‘404’.	`SELECT City, COUNT(*) FROM Addresses WHERE District = 'Lahore' AND ZipCode LIKE '404%' GROUP BY City;`
2	This query counts the number of rows in the Customers table for each FirstName where the LastName is ‘Khan’ and the PhoneNumber starts with ‘0300’.	`SELECT FirstName, COUNT(*) FROM Customers WHERE LastName = 'Khan' AND PhoneNumber LIKE '0300%' GROUP BY FirstName;`
3	This query counts the number of rows in the Servicess table for each ServiceName where the MonthlyFee is greater than 1000 and the ServiceDescription contains ‘Internet’.	`SELECT ServiceName, COUNT(*) FROM Servicess WHERE MonthlyFee > 1000 AND ServiceDescription LIKE '%Internet%' GROUP BY ServiceName;`
4	This query counts the number of rows in the Subscriptions table for each CustomerID where the StartDate is on or after ‘2022-01-01’ and the EndDate is on or before ‘2022-12-31’.	`SELECT CustomerID, COUNT(*) FROM Subscriptions WHERE StartDate >= '2022-01-01' AND EndDate <= '2022-12-31' GROUP BY CustomerID;`
5	This query counts the number of rows in the Billing table for each SubscriptionID where the AmountDue is greater than	`SELECT SubscriptionID, COUNT(*) FROM Billing WHERE AmountDue > 3000 AND DueDate < GETDATE() GROUP BY SubscriptionID;`

	3000 and the DueDate is earlier than the current date.	
6	This query counts the number of rows in the Payments table for each BillingID where the PaymentAmount is greater than 3000 and the PaymentDate is earlier than the current date.	`SELECT BillingID, COUNT(*) FROM Payments WHERE PaymentAmount > 3000 AND PaymentDate < GETDATE() GROUP BY BillingID;`
7	This query calculates the total data used (SUM(DataUsed)) in the Usage table for each SubscriptionID where UsageDate is between 2022-01-01 and 2022-12-31.	`SELECT SubscriptionID, SUM(DataUsed) FROM Usage WHERE UsageDate >= '2022-01-01' AND UsageDate <= '2022-12-31' GROUP BY SubscriptionID;`
8	This query counts the number of rows in Technicians table for each FirstName where LastName is Ali and PhoneNumber starts with 0300.	`SELECT FirstName, COUNT(*) FROM Technicians WHERE LastName = 'Ali' AND PhoneNumber LIKE '0300%' GROUP BY FirstName;`
9	This query counts number of rows in Issues table for each CustomerID where DateResolved is not null and DateCreated is after 2022-01-01.	`SELECT CustomerID, COUNT(*) FROM Issues WHERE DateResolved IS NOT NULL AND DateCreated > '2022-01-01' GROUP BY CustomerID;`
10	This query counts number of rows in Equipment table for each EquipmentName where EquipmentDescription contains Router and EquipmentID is greater than 100.	`SELECT EquipmentName, COUNT(*) FROM Equipment WHERE EquipmentDescription LIKE '%Router%' AND EquipmentID > 100 GROUP BY EquipmentName;`
11	This query counts number of rows in Addresses table for each City where District is Lahore or ZipCode starts with 404.	`SELECT City, COUNT(*) FROM Addresses WHERE District = 'Lahore' OR ZipCode LIKE '404%'GROUP BY City;`
12	This query counts number of rows in Customers table for	`SELECT FirstName, COUNT(*) FROM Customers WHERE LastName = 'Khan' OR PhoneNumber LIKE '0300%' GROUP BY FirstName;`

	each FirstName where LastName is Khan or PhoneNumber starts with 0300.	
13	This query counts number of rows in Servicess table for each ServiceName where MonthlyFee is greater than 1000 or ServiceDescription contains Internet.	`SELECT ServiceName, COUNT(*) FROM Servicess WHERE MonthlyFee > 1000 OR ServiceDescription LIKE '%Internet%' GROUP BY ServiceName;`
14	This query counts number of rows in Subscriptions table for each CustomerID where StartDate is on or after 2022-01-01 or EndDate is on or before 2022-12-31.	`SELECT CustomerID, COUNT(*) FROM Subscriptions WHERE StartDate >= '2022-01-01' OR EndDate <= '2022-12-31' GROUP BY CustomerID;`
15	This query counts number of rows in Billing table for each SubscriptionID where AmountDue is greater than 3000 or DueDate is earlier than current date.	`SELECT SubscriptionID, COUNT(*) FROM Billing WHERE AmountDue > 3000 OR DueDate < GETDATE() GROUP BY SubscriptionID;`
16	This query counts number of rows in Payments table for each BillingID where PaymentAmount is greater than 3000 or PaymentDate is earlier than current date.	`SELECT BillingID, COUNT(*) FROM Payments WHERE PaymentAmount > 3000 OR PaymentDate < GETDATE() GROUP BY BillingID;`
17	This query calculates total data used (SUM(DataUsed)) in Usage table for each SubscriptionID where UsageDate is on or after 2022-01-01 or on or before 2022-12-31.	`SELECT SubscriptionID, SUM(DataUsed) FROM Usage WHERE UsageDate >= '2022-01-01' OR UsageDate <= '2022-12-31' GROUP BY SubscriptionID;`
18	This query counts number of rows in Technicians table for each FirstName where LastName is Ali or PhoneNumber starts with 0300.	`SELECT FirstName, COUNT(*) FROM Technicians WHERE LastName = 'Ali' OR PhoneNumber LIKE '0300%' GROUP BY FirstName;`

19	This query counts number of rows in Issues table for each CustomerID where DateResolved is not null or DateCreated is after 2022-01-01.	`SELECT CustomerID, COUNT(*) FROM Issues WHERE DateResolved IS NOT NULL OR DateCreated > '2022-01-01' GROUP BY CustomerID;`
20	This query counts number of rows in Equipment table for each EquipmentName where EquipmentDescription contains Router or EquipmentID is greater than 100.	`SELECT EquipmentName, COUNT(*) FROM Equipment WHERE EquipmentDescription LIKE '%Router%' OR EquipmentID > 100 GROUP BY EquipmentName;`
21	This query counts number of rows in Addresses table for each City where District isn't Lahore.	`SELECT City, COUNT(*) FROM Addresses WHERE NOT District = 'Lahore' GROUP BY City;`
22	This query counts number of rows in Customers table for each FirstName where LastName isn't Khan.	`SELECT FirstName, COUNT(*) FROM Customers WHERE NOT LastName = 'Khan' GROUP BY FirstName;`
23	This query counts number of rows in Servicess table for each ServiceName where MonthlyFee isn't greater than 1000.	`SELECT ServiceName, COUNT(*) FROM Servicess WHERE NOT MonthlyFee > 1000 GROUP BY ServiceName;`
24	This query counts number of rows in Subscriptions table for each CustomerID where StartDate isn't on or after 2022-01-01.	`SELECT CustomerID, COUNT(*) FROM Subscriptions WHERE NOT StartDate >= '2022-01-01' GROUP BY CustomerID;`
25	This query counts number of rows in Billing table for each SubscriptionID where AmountDue isn't greater than 3000	`SELECT SubscriptionID, COUNT(*) FROM Billing WHERE NOT AmountDue > 3000 GROUP BY SubscriptionID.`

12. Subqueries– 30 Queries

1	Find the names of customers who have a subscription	`SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions);`

2	Find the names of services that have a subscription	SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions);
3	Find the subscription IDs that have a billing record	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Billing);
4	Find the billing IDs that have a payment record	SELECT BillingID FROM Billing WHERE BillingID IN (SELECT BillingID FROM Payments);
5	Find the subscription IDs that have a usage record	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Usage);
6	Find the customer IDs that have an issue record	SELECT CustomerID FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Issues);
7	Find the names of customers who do not have a subscription	SELECT FirstName, LastName FROM Customers WHERE CustomerID NOT IN (SELECT CustomerID FROM Subscriptions);
8	Find the names of services that do not have a subscription	SELECT ServiceName FROM Servicess WHERE ServiceID NOT IN (SELECT ServiceID FROM Subscriptions);
9	Find the subscription IDs that do not have a billing record	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID NOT IN (SELECT SubscriptionID FROM Billing);
10	Find the billing IDs that do not have a payment record	SELECT BillingID FROM Billing WHERE BillingID NOT IN (SELECT BillingID FROM Payments);
11	Find the subscription IDs that do not have a usage record	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID NOT IN (SELECT SubscriptionID FROM Usage);
12	Find the customer IDs that do not have an issue record	SELECT CustomerID FROM Customers WHERE CustomerID NOT IN (SELECT CustomerID FROM Issues);
13	Find the names of customers who have a subscription with a monthly fee greater than 4000	SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions WHERE ServiceID IN (SELECT ServiceID FROM Servicess WHERE MonthlyFee > 50));
14	Find the names of services that have a subscription with a start date after '2022-01-01'	SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions

		WHERE StartDate > '2022-01-01');
15	Find the subscription IDs that have a billing record with an amount due greater than 3000	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Billing WHERE AmountDue > 3000);
16	Find the billing IDs that have a payment record with a payment amount greater than 2000	SELECT BillingID FROM Billing WHERE BillingID IN (SELECT BillingID FROM Payments WHERE PaymentAmount > 2000);
17	Find the subscription IDs that have a usage record with data used greater than 15	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Usage WHERE DataUsed > 15);
18	Find the customer IDs that have an issue record with an issue description containing the word ‘internet’	SELECT CustomerID FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Issues WHERE IssueDescription LIKE '%internet%');
19	Find the names of customers who have a subscription with a service name starting with the letter ‘A’	SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions WHERE ServiceID IN (SELECT ServiceID FROM Servicess WHERE ServiceName LIKE 'A%'));
20	Find the names of services that have a subscription with a customer first name starting with the letter ‘A’	SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE FirstName LIKE 'A%'));
21	Find the names of services that have a subscription with a customer last name containing the word ‘Ali’,	SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions

	start date after ‘2022-01-01’, and end date before ‘2023-01-01’	WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE LastName LIKE '%Ali%') AND StartDate > '2022-01-01' AND EndDate < '2023-01-01');
22	Find the names of services that have a subscription with a customer last name containing the word ‘Khan’	SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE LastName LIKE '%Khan%'));
23	Find the subscription IDs that have a billing record with a customer last name containing the word ‘Kamran’	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Billing WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE LastName LIKE '%Kamran%'));
24	Find the names of customers who have a subscription with a service name starting with the letter ‘A’ and a monthly fee greater than 4000:	SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions WHERE ServiceID IN (SELECT ServiceID FROM Servicess WHERE ServiceName LIKE 'A%' AND MonthlyFee > 4000)
25	Find the customer IDs that have an issue record with an issue description containing the word ‘internet’ and a first name starting with the letter ‘A’	SELECT CustomerID FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Issues WHERE IssueDescription LIKE '%internet%' AND FirstName LIKE 'A%');
26	Find the names of customers who have a subscription with a service name containing the word ‘internet’	SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions WHERE ServiceID IN (SELECT ServiceID FROM Servicess WHERE ServiceName LIKE '%Internet%'));

27	Find the subscription IDs that have a billing record with a customer last name containing the word ‘Ali’ and an amount due greater than 2000	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Billing WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE LastName LIKE '%Ali%') AND AmountDue > 2000);
28	Find the names of customers who have a subscription with a service name containing the word ‘internet’ and a monthly fee greater than 4000	SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Subscriptions WHERE ServiceID IN (SELECT ServiceID FROM Servicess WHERE ServiceName LIKE '%internet%' AND MonthlyFee > 4000));
29	Find the names of services that have a subscription with a customer last name containing the word ‘Khan’, start date after ‘2022-01-01’, end date before ‘2023-01-01’, and monthly fee greater than 2000	SELECT ServiceName FROM Servicess WHERE ServiceID IN (SELECT ServiceID FROM Subscriptions WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE LastName LIKE '%Khan%') AND StartDate > '2022-01-01' AND EndDate < '2023-01-01' AND MonthlyFee > 2000);
30	Find the subscription IDs that have a billing record with a customer last name containing the word ‘Kamran’, amount due greater than 3000, and billing date in the year 2022	SELECT SubscriptionID FROM Subscriptions WHERE SubscriptionID IN (SELECT SubscriptionID FROM Billing WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE LastName LIKE '%Kamran%') AND AmountDue > 3000 AND YEAR(BillingDate) = 2022);

13. Subqueries using Logical Operators– 30 Queries

1	This query selects all columns from the Customers table where the AddressID matches an AddressID in the Addresses table with the City ‘Lahore’ and ZipCode ‘40410’.	SELECT * FROM Customers WHERE AddressID IN (SELECT AddressID FROM Addresses WHERE City = 'Lahore' AND ZipCode = '40410')

2	This query selects all columns from the Subscriptions table where the CustomerID matches a CustomerID in the Customers table with the FirstName ‘Ali’ and LastName ‘Khan’.	`SELECT * FROM Subscriptions WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE FirstName = 'Ali' AND LastName = 'Khan')`
3	This query selects all columns from the Billing table where the SubscriptionID matches a SubscriptionID in the Subscriptions table with a StartDate on or after ‘2022-01-01’ and an EndDate on or before ‘2022-12-31’.	`SELECT * FROM Billing WHERE SubscriptionID IN (SELECT SubscriptionID FROM Subscriptions WHERE StartDate >= '2022-01-01' AND EndDate <= '2022-12-31')`
4	This query selects all columns from the Payments table where the BillingID matches a BillingID in the Billing table with an AmountDue greater than 3000 and a DueDate earlier than the current date.	`SELECT * FROM Payments WHERE BillingID IN (SELECT BillingID FROM Billing WHERE AmountDue > 3000 AND DueDate < GETDATE())`
5	This query selects all columns from the Usage table where the SubscriptionID matches a SubscriptionID in the Subscriptions table with a ServiceID that matches a ServiceID in the Servicess table with a ServiceName ‘Internet’ and a MonthlyFee less than 3000.	`SELECT * FROM Usage WHERE SubscriptionID IN (SELECT SubscriptionID FROM Subscriptions WHERE ServiceID IN (SELECT ServiceID FROM Servicess WHERE ServiceName = 'Internet' AND MonthlyFee < 3000))`

6	This query selects all columns from the Issues table where the TechnicianID matches a TechnicianID in the Technicians table with the FirstName ‘Muhammad’ and the LastName ‘Ali’.	`SELECT * FROM Issues WHERE TechnicianID IN (SELECT TechnicianID FROM Technicians WHERE FirstName = 'Muhammad' AND LastName = 'Ali')`
7	This query selects all columns from the Equipment table where the EquipmentID matches an EquipmentID in the Issues table with a non-null value for the column ‘DateResolved’ and a value for ‘DateCreated’ greater than ‘2022-01-01’.	`SELECT * FROM Equipment WHERE EquipmentID IN (SELECT EquipmentID FROM Issues WHERE DateResolved IS NOT NULL AND DateCreated > '2022-01-01')`
8	This query selects all columns from the ServiceOffices table where City is present in Addresses table with District as Lahore and ZipCode starting with 404.	`SELECT * FROM ServiceOffices WHERE City IN (SELECT City FROM Addresses WHERE District = 'Lahore' AND ZipCode LIKE '404%')`
9	This query selects all columns from Promotions table where StartDate is on or after 2022-01-01 and EndDate is on or before 2022-12-31 and PromotionName is present in Servicess table as ServiceName.	`SELECT * FROM Promotions WHERE StartDate >= '2022-01-01' AND EndDate <= '2022-12-31' AND PromotionName IN (SELECT ServiceName FROM Servicess)`
10	This query selects all columns from Contracts table where CustomerID is present in Customers table with AddressID present in Addresses table with City as Bhalwal and District as Sargodha.	`SELECT * FROM Contracts WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE AddressID IN (SELECT AddressID FROM Addresses WHERE City = 'Bhalwal' AND District = 'Sargodha'))`

11	This query selects all columns from Customers table where AddressID is present in Addresses table with City as Sargodha or ZipCode as 40404.	`SELECT * FROM Customers WHERE AddressID IN (SELECT AddressID FROM Addresses WHERE City = 'Sargodha' OR ZipCode = '40404')`
12	This query selects all columns from Subscriptions table where CustomerID is present in Customers table with FirstName as Shahid or LastName as Afridi.	`SELECT * FROM Subscriptions WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE FirstName = 'Shahid' OR LastName = 'Afridi')`
13	This query selects all columns from Billing table where SubscriptionID is present in Subscriptions table with StartDate on or after 2022-01-01 or EndDate on or before 2022-12-31.	`SELECT * FROM Billing WHERE SubscriptionID IN (SELECT SubscriptionID FROM Subscriptions WHERE StartDate >= '2022-01-01' OR EndDate <= '2022-12-31')`
14	This query selects all columns from Payments table where BillingID is present in Billing table with AmountDue greater than 4000 or DueDate earlier than current date.	`SELECT * FROM Payments WHERE BillingID IN (SELECT BillingID FROM Billing WHERE AmountDue > 4000 OR DueDate < GETDATE())`
15	This query selects all columns from Usage table where SubscriptionID is present in Subscriptions table with ServiceID present in Servicess table with ServiceName as Internet or MonthlyFee less than 3000.	`SELECT * FROM Usage WHERE SubscriptionID IN (SELECT SubscriptionID FROM Subscriptions WHERE ServiceID IN (SELECT ServiceID FROM Servicess WHERE ServiceName = 'Internet' OR MonthlyFee < 3000))`
16	This query selects all columns from the Issues table where the TechnicianID matches a TechnicianID in	`SELECT * FROM Issues WHERE TechnicianID IN (SELECT TechnicianID FROM Technicians WHERE FirstName = 'Umer' OR LastName = 'Ali')`

	the Technicians table with the FirstName ‘Umer’ or the LastName ‘Ali’.	
17	This query selects all columns from the Equipment table where the EquipmentID matches an EquipmentID in the Issues table with a non-null value for the column ‘DateResolved’ or a value for ‘DateCreated’ greater than ‘2022-01-01’.	`SELECT * FROM Equipment WHERE EquipmentID IN (SELECT EquipmentID FROM Issues WHERE DateResolved IS NOT NULL OR DateCreated > '2022-01-01')`
18	This query selects all columns from the ServiceOffices table where City is present in Addresses table with District as Karachi or ZipCode starting with 410.	`SELECT * FROM ServiceOffices WHERE City IN (SELECT City FROM Addresses WHERE District = 'Karachi' OR ZipCode LIKE '410%')`
19	This query selects all columns from Promotions table where StartDate is on or after 2022-01-01 or EndDate is on or before 2022-12-31 or PromotionName is present in Servicess table as ServiceName.	`SELECT * FROM Promotions WHERE StartDate >= '2022-01-01' OR EndDate <= '2022-12-31' OR PromotionName IN (SELECT ServiceName FROM Servicess)`
20	This query selects all columns from Contracts table where CustomerID is present in Customers table with AddressID present in Addresses table with City as Lahore or District as Lahore.	`SELECT * FROM Contracts WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE AddressID IN (SELECT AddressID FROM Addresses WHERE City = 'Lahore' OR District = 'Lahore'))`
21	This query selects all columns from Customers table where AddressID is not present in Addresses	`SELECT * FROM Customers WHERE NOT AddressID IN (SELECT AddressID FROM Addresses WHERE City = 'Bhalwal' AND ZipCode = '40404')`

	table with City as Bhalwal and ZipCode as 40404.	
22	This query selects all columns from Subscriptions table where CustomerID is not present in Customers table with FirstName as Umer and LastName as Gul.	`SELECT * FROM Subscriptions WHERE NOT CustomerID IN (SELECT CustomerID FROM Customers WHERE FirstName = 'Umer' AND LastName = 'Gul')`
23	This query selects all columns from Billing table where SubscriptionID is not present in Subscriptions table with StartDate on or after 2022-01-01 and EndDate on or before 2022-12-31.	`SELECT * FROM Billing WHERE NOT SubscriptionID IN (SELECT SubscriptionID FROM Subscriptions WHERE StartDate >= '2022-01-01' AND EndDate <= '2022-12-31')`
24	This query selects all columns from Payments table where BillingID is not present in Billing table with AmountDue greater than 2000 and DueDate earlier than current date.	`SELECT * FROM Payments WHERE NOT BillingID IN (SELECT BillingID FROM Billing WHERE AmountDue > 2000 AND DueDate < GETDATE())`
25	This query selects all columns from Usage table where SubscriptionID is not present in Subscriptions table with ServiceID present in Servicess table with ServiceName as Internet and MonthlyFee less than 1000.	`SELECT * FROM Usage WHERE NOT SubscriptionID IN (SELECT SubscriptionID FROM Subscriptions WHERE ServiceID IN (SELECT ServiceID FROM Servicess WHERE ServiceName = 'Internet' AND MonthlyFee < 1000))`
26	This query selects all columns from Issues table where TechnicianID is not present in Technicians table with FirstName as Ali and LastName as Umer.	`SELECT * FROM Issues WHERE NOT TechnicianID IN (SELECT TechnicianID FROM Technicians WHERE FirstName = 'Ali' AND LastName = 'Umer')`
27	This query selects all columns from Equipment	`SELECT * FROM Equipment`

	table where EquipmentID is not present in Issues table with DateResolved non-null and DateCreated greater than 2022-01-01.	`WHERE NOT EquipmentID IN (SELECT EquipmentID FROM Issues WHERE DateResolved IS NOT NULL AND DateCreated > '2022-01-01')`
28	This query selects all columns from ServiceOffices table where City is not present in Addresses table with District as Multan and ZipCode starting with 100.	`SELECT * FROM ServiceOffices WHERE NOT City IN (SELECT City FROM Addresses WHERE District = 'Multan' AND ZipCode LIKE '100%')`
29	This query selects all columns from Promotions table where StartDate is not on or after 2022-01-01 and EndDate is not on or before 2022-12-31 and PromotionName is not present in Servicess table as ServiceName.	`SELECT * FROM Promotions WHERE NOT StartDate >= '2022-01-01' AND NOT EndDate <= '2022-12-31' AND NOT PromotionName IN (SELECT ServiceName FROM Servicess)`
30	This query selects all columns from Contracts table where CustomerID is not present in Customers table with AddressID present in Addresses table with City as Bhalwal and District as Sargodha.	`SELECT * FROM Contracts WHERE NOT CustomerID IN (SELECT CustomerID FROM Customers WHERE AddressID IN (SELECT AddressID FROM Addresses WHERE City = 'Bhalwal' AND District = 'Sargodha'))`

14. Aggregate functions MAX, MIN, SUM, COUNT, and AVG– 20 Queries

1	Find the maximum monthly fee for all services	`SELECT MAX(MonthlyFee) AS MaxMonthlyFee FROM Servicess;`
2	Find the maximum amount due for all billings	`SELECT MAX(AmountDue) AS MaxAmountDue FROM Billing;`
3	Find the maximum payment amount for all payments	`SELECT MAX(PaymentAmount) AS MaxPaymentAmount FROM Payments;`
4	Find the maximum data used for all usage records	`SELECT MAX(DataUsed) AS MaxDataUsed FROM Usage;`
5	Find the minimum monthly fee for all services	`SELECT MIN(MonthlyFee) AS MinMonthlyFee FROM Servicess;`

6	Find the minimum amount due for all billings	`SELECT MIN(AmountDue) AS MinAmountDue FROM Billing;`
7	Find the minimum payment amount for all payments	`SELECT MIN(PaymentAmount) AS MinPaymentAmount FROM Payments;`
8	Find the minimum data used for all usage records	`SELECT MIN(DataUsed) AS MinDataUsed FROM Usage;`
9	Find the total monthly fee for all services	`SELECT SUM(MonthlyFee) AS TotalMonthlyFee FROM Servicess;`
10	Find the total amount due for all billings	`SELECT SUM(AmountDue) AS TotalAmountDue FROM Billing;`
11	Find the total payment amount for all payments	`SELECT SUM(PaymentAmount) AS TotalPaymentAmount FROM Payments;`
12	Find the total data used for all usage records	`SELECT SUM(DataUsed) AS TotalDataUsed FROM Usage;`
13	Find the total number of services	`SELECT COUNT(*) AS TotalServices FROM Servicess;`
14	Find the total number of billings	`SELECT COUNT(*) AS TotalBillings FROM Billing;`
15	Find the total number of payments	`SELECT COUNT(*) AS TotalPayments FROM Payments;`
16	Find the total number of usage records	`SELECT COUNT(*) AS TotalUsageRecords FROM Usage;`
17	Find the average monthly fee for all services	`SELECT AVG(MonthlyFee) AS AvgMonthlyFee FROM Servicess;`
18	Find the average amount due for all billings	`SELECT AVG(AmountDue) AS AvgAmountDue FROM Billing;`
19	Find the average payment amount for all payments	`SELECT AVG(PaymentAmount) AS AvgPaymentAmount FROM Payments;`
20	Find the average data used for all usage records	`SELECT AVG(DataUsed) AS AvgDataUsed FROM Usage;`

15. Aggregate functions using Logical Operators and Group by– 30 Queries

1	Find the total monthly fee for services where the monthly fee is greater than 2000	`SELECT SUM(MonthlyFee) AS TotalMonthlyFee FROM Servicess WHERE MonthlyFee > 2000;`
2	Find the average amount due for billings where the amount due is less than 3000	`SELECT AVG(AmountDue) AS AvgAmountDue FROM Billing WHERE AmountDue < 3000;`
3	Find the maximum payment amount for payments where the payment date is after '2022-01-01'	`SELECT MAX(PaymentAmount) AS MaxPaymentAmount FROM Payments WHERE PaymentDate > '2022-01-01';`
4	Find the minimum data used for usage records where the	`SELECT MIN(DataUsed) AS MinDataUsed FROM Usage`

	usage date is before ‘2022-01-01’	`WHERE UsageDate < '2022-01-01';`

5	Find the count of issues where the issue description contains the word ‘internet’	`SELECT COUNT(*) AS TotalInternetIssues`
		`FROM Issues`
		`WHERE IssueDescription LIKE '%internet%';`

6	Find the total monthly fee for each service	`SELECT ServiceName, SUM(MonthlyFee) AS TotalMonthlyFee`
		`FROM Servicess`
		`GROUP BY ServiceName;`

7	Find the average amount due for each billing date	`SELECT BillingDate, AVG(AmountDue) AS AvgAmountDue`
		`FROM Billing`
		`GROUP BY BillingDate;`

8	Find the maximum payment amount for each billing ID	`SELECT BillingID, MAX(PaymentAmount) AS MaxPaymentAmount`
		`FROM Payments`
		`GROUP BY BillingID;`

9	Find the minimum data used for each subscription ID	`SELECT SubscriptionID, MIN(DataUsed) AS MinDataUsed`
		`FROM Usage`
		`GROUP BY SubscriptionID;`

10	Find the count of issues for each customer ID	`SELECT CustomerID, COUNT(*) AS TotalIssues`
		`FROM Issues`
		`GROUP BY CustomerID;`

11	Find the total monthly fee for services where the monthly fee is greater than 3000, grouped by service name	`SELECT ServiceName, SUM(MonthlyFee) AS TotalMonthlyFee`
		`FROM Servicess`
		`WHERE MonthlyFee > 3000`
		`GROUP BY ServiceName;`

12	Find the average amount due for billings where the amount due is less than 4000, grouped by billing date	`SELECT BillingDate, AVG(AmountDue) AS AvgAmountDue`
		`FROM Billing`
		`WHERE AmountDue < 4000`
		`GROUP BY BillingDate;`

13	Find the maximum payment amount for payments where the payment date is after ‘2022-01-01’, grouped by billing ID	`SELECT BillingID, MAX(PaymentAmount) AS MaxPaymentAmount`
		`FROM Payments`
		`WHERE PaymentDate > '2022-01-01'`
		`GROUP BY BillingID;`

14	Find the minimum data used for usage records where the usage date is before ‘2022-01-01’, grouped by subscription ID	`SELECT SubscriptionID, MIN(DataUsed) AS MinDataUsed`
		`FROM Usage`
		`WHERE UsageDate < '2022-01-01'`
		`GROUP BY SubscriptionID;`

15	Find the count of issues where the issue description contains the word ‘internet’, grouped by customer ID	`SELECT CustomerID, COUNT(*) AS TotalInternetIssues`
		`FROM Issues`
		`WHERE IssueDescription LIKE '%internet%'`
		`GROUP BY CustomerID;`

16	Find the total monthly fee for services where the	`SELECT SUM(MonthlyFee) AS TotalMonthlyFee`
		`FROM Servicess`
		`WHERE MonthlyFee > 2000 AND MonthlyFee < 4000;`

	monthly fee is greater than 2000 and less than 4000	
17	Find the average amount due for billings where the amount due is less than 3000 or greater than 4000	`SELECT AVG(AmountDue) AS AvgAmountDue FROM Billing WHERE AmountDue < 3000 OR AmountDue > 4000;`
18	Find the maximum payment amount for payments where payment date is after '2022-01-01' and payment amount is less than 3000:	`SELECT MAX(PaymentAmount) AS MaxPaymentAmount FROM Payments WHERE PaymentDate > '2022-01-01' AND PaymentAmount < 3000;`
19	Find minimum data used for usage records where usage date is before '2022-01-01' and data used is greater than 10	`SELECT MIN(DataUsed) AS MinDataUsed FROM Usage WHERE UsageDate < '2022-01-01' AND DataUsed > 10;`
20	Find the count of issues where issue description contains the word 'internet' or 'cable'	`SELECT COUNT(*) AS TotalInternetCableIssues FROM Issues WHERE IssueDescription LIKE '%internet%' OR IssueDescription LIKE '%cable%';`
21	Find the total monthly fee for services where the monthly fee is greater than 2000 and less than 3000, grouped by service name	`SELECT ServiceName, SUM(MonthlyFee) AS TotalMonthlyFee FROM Servicess WHERE MonthlyFee > 2000 AND MonthlyFee < 3000 GROUP BY ServiceName;`
22	Find the average amount due for billings where the amount due is less than 3000 or greater than 4000, grouped by billing date	`SELECT BillingDate, AVG(AmountDue) AS AvgAmountDue FROM Billing WHERE AmountDue < 3000 OR AmountDue > 4000 GROUP BY BillingDate;`
23	Find the maximum payment amount for payments where payment date is after '2022-01-01' and payment amount is less than 3000, grouped by billing ID	`SELECT BillingID, MAX(PaymentAmount) AS MaxPaymentAmount FROM Payments WHERE PaymentDate > '2022-01-01' AND PaymentAmount < 3000 GROUP BY BillingID;`
24	Find minimum data used for usage records where usage date is before '2022-01-01' and data used is greater than 1, grouped by subscription ID	`SELECT SubscriptionID, MIN(DataUsed) AS MinDataUsed FROM Usage WHERE UsageDate < '2022-01-01' AND DataUsed > 1 GROUP BY SubscriptionID;`
25	Find the count of issues where issue description contains the word 'internet'	`SELECT CustomerID, COUNT(*) AS TotalInternetCableIssues FROM Issues WHERE IssueDescription LIKE '%internet%' OR IssueDescription LIKE '%cable%'`

	or ‘cable’, grouped by customer ID	`GROUP BY CustomerID;`

26	Find the total monthly fee for services where the service name starts with the letter ‘A’	`SELECT SUM(MonthlyFee) AS TotalMonthlyFee`
		`FROM Servicess`
		`WHERE ServiceName LIKE 'A%';`

27	Find the average amount due for billings where the billing date is in the year 2022	`SELECT AVG(AmountDue) AS AvgAmountDue`
		`FROM Billing`
		`WHERE YEAR(BillingDate) = 2022;`

28	Find the maximum payment amount for payments where payment date is in January	`SELECT MAX(PaymentAmount) AS MaxPaymentAmount`
		`FROM Payments`
		`WHERE MONTH(PaymentDate) = 1;`

29	Find minimum data used for usage records where usage date is on a Monday	`SELECT MIN(DataUsed) AS MinDataUsed`
		`FROM Usage`
		`WHERE DATENAME(WEEKDAY, UsageDate) = 'Monday';`

30	Find the count of issues where issue description contains the word ‘internet’ and issue was created in December	`SELECT COUNT(*) AS TotalDecemberInternetIssues`
		`FROM Issues`
		`WHERE IssueDescription LIKE '%internet%' AND`
		`MONTH(DateCreated) = 12;`

16. INNER joins– 20 Queries

1	Retrieves the first name and last name of customers along with their street address by joining the Customers and Addresses tables on the AddressID column.	`SELECT Customers.FirstName, Customers.LastName,`
		`Addresses.StreetAddress`
		`FROM Customers INNER JOIN Addresses`
		`ON Customers.AddressID = Addresses.AddressID`

2	Retrieves the subscription ID, first name and last name of customers by joining the Subscriptions and Customers tables on the CustomerID column.	`SELECT Subscriptions.SubscriptionID,`
		`Customers.FirstName, Customers.LastName`
		`FROM Subscriptions INNER JOIN Customers`
		`ON Subscriptions.CustomerID = Customers.CustomerID`

3	Retrieves the billing ID and subscription ID by joining the Billing and Subscriptions tables on the SubscriptionID column.	`SELECT Billing.BillingID,`
		`Subscriptions.SubscriptionID`
		`FROM Billing INNER JOIN Subscriptions`
		`ON Billing.SubscriptionID =`
		`Subscriptions.SubscriptionID`

4	Retrieves the payment ID and billing ID by joining the Payments and Billing tables on the BillingID column.	`SELECT Payments.PaymentID, Billing.BillingID`
		`FROM Payments INNER JOIN Billing`
		`ON Payments.BillingID = Billing.BillingID`

5	Retrieves the usage ID and subscription ID by joining the Usage and Subscriptions tables on the SubscriptionID column.	`SELECT Usage.UsageID, Subscriptions.SubscriptionID`
		`FROM Usage INNER JOIN Subscriptions`
		`ON Usage.SubscriptionID =`
		`Subscriptions.SubscriptionID`

6	Retrieves the issue ID, first name and last name of technicians by joining the Issues and Technicians tables on the TechnicianID column.	`SELECT Issues.IssueID, Technicians.FirstName,`
		`Technicians.LastName`
		`FROM Issues INNER JOIN Technicians`
		`ON Issues.TechnicianID = Technicians.TechnicianID`

7	Retrieves the service name, service description and monthly fee by joining the Servicess and Subscriptions tables on the ServiceID column.	`SELECT Servicess.ServiceName,`
		`Servicess.ServiceDescription, Servicess.MonthlyFee`
		`FROM Servicess INNER JOIN Subscriptions`
		`ON Servicess.ServiceID = Subscriptions.ServiceID`

8	Retrieves the city and zip code of service offices by joining the ServiceOffices and Addresses tables on the City column.	`SELECT ServiceOffices.City, ServiceOffices.ZipCode`
		`FROM ServiceOffices INNER JOIN Addresses`
		`ON ServiceOffices.City = Addresses.City`

9	Retrieves the promotion name and service name by joining the Promotions and Servicess tables on the PromotionName and ServiceName columns respectively.	`SELECT Promotions.PromotionName,`
		`Servicess.ServiceName`
		`FROM Promotions INNER JOIN Servicess`
		`ON Promotions.PromotionName = Servicess.ServiceName`

10	This query selects the ContractID, FirstName, and LastName columns from the Contracts and Customers tables by using INNER JOIN .	`SELECT Contracts.ContractID, Customers.FirstName,`
		`Customers.LastName`
		`FROM Contracts INNER JOIN Customers`
		`ON Contracts.CustomerID = Customers.CustomerID`

11	This query retrieves the first and last names of customers along with the name of the service they are subscribed to by joining the Customers, Subscriptions, and Servicess tables.	`SELECT Customers.FirstName, Customers.LastName,`
		`Servicess.ServiceName`
		`FROM Customers INNER JOIN Subscriptions`
		`ON Customers.CustomerID = Subscriptions.CustomerID`
		`INNER JOIN Servicess`
		`ON Subscriptions.ServiceID = Servicess.ServiceID`

12	This query retrieves the billing date, amount due, and payment amount by	`SELECT Billing.BillingDate, Billing.AmountDue,`
		`Payments.PaymentAmount`
		`FROM Billing INNER JOIN Payments`
		`ON Billing.BillingID = Payments.BillingID`

	joining the Billing and Payments tables on the BillingID column.	
13	This query retrieves the usage date, data used, and subscription start date by joining the Usage and Subscriptions tables on the SubscriptionID column.	`SELECT Usage.UsageDate, Usage.DataUsed, Subscriptions.StartDate FROM Usage INNER JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID`
14	This query retrieves the issue description and the first and last names of the technician assigned to the issue by joining the Issues and Technicians tables on the TechnicianID column.	`SELECT Issues.IssueDescription, Technicians.FirstName, Technicians.LastName FROM Issues INNER JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID`
15	This query retrieves the city and zip code of service offices and the street addresses of customers in the same city by joining the ServiceOffices and Addresses tables on the City column.	`SELECT Customers.FirstName, Customers.LastName, Issues.IssueDescription FROM Customers INNER JOIN Issues ON Customers.CustomerID = Issues.CustomerID`
16	This query retrieves the city and zip code of service offices and the street addresses of customers in the same city by joining the ServiceOffices and Addresses tables on the City column.	`SELECT ServiceOffices.City, ServiceOffices.ZipCode, Addresses.StreetAddress FROM ServiceOffices INNER JOIN Addresses ON ServiceOffices.City = Addresses.City`
17	This query retrieves the name and description of promotions and the name of services with the same name by joining the Promotions and Servicess tables on the PromotionName and ServiceName columns.	`SELECT Promotions.PromotionName, Promotions.PromotionDescription, Servicess.ServiceName FROM Promotions INNER JOIN Servicess ON Promotions.PromotionName = Servicess.ServiceName`
18	This query retrieves the contract ID, start date, and first name of the customer associated with the contract by joining the Contracts and	`SELECT Contracts.ContractID, Contracts.StartDate, Customers.FirstName FROM Contracts INNER JOIN Customers ON Contracts.CustomerID = Customers.CustomerID`

	Customers tables on the CustomerID column.	
19	This query retrieves the billing date, amount due, and payment amount for payments greater than 4000 by joining the Billing and Payments tables on the BillingID column and filtering the results with a WHERE clause.	`SELECT Billing.BillingDate, Billing.AmountDue, Payments.PaymentAmount FROM Billing INNER JOIN Payments ON Billing.BillingID = Payments.BillingID WHERE Payments.PaymentAmount > 4000`
20	This query retrieves the usage date, data used, and subscription start date for usage records with data used greater than 50 by joining the Usage and Subscriptions tables on the SubscriptionID column and filtering the results with a WHERE clause.	`SELECT Usage.UsageDate, Usage.DataUsed, Subscriptions.StartDate FROM Usage INNER JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID WHERE Usage.DataUsed > 50`

17. INNER joins using logical operators, Group By and Order By– 30 Queries

1	This query retrieves the first and last names and street address of customers with the first name 'Ali' and last name 'Khan' by joining the Customers and Addresses tables on the AddressID column and filtering the results with a WHERE clause.	`SELECT Customers.FirstName, Customers.LastName, Addresses.StreetAddress FROM Customers INNER JOIN Addresses ON Customers.AddressID = Addresses.AddressID WHERE Customers.FirstName = 'Ali' AND Customers.LastName = 'Khan'`
2	This query retrieves the subscription ID and first and last names of customers with subscriptions starting on or after '2022-01-01' and ending on or before '2022-12-31' by joining the Subscriptions and Customers tables on the CustomerID column and filtering the results with a WHERE clause.	`SELECT Subscriptions.SubscriptionID, Customers.FirstName, Customers.LastName FROM Subscriptions INNER JOIN Customers ON Subscriptions.CustomerID = Customers.CustomerID WHERE Subscriptions.StartDate >= '2022-01-01' AND Subscriptions.EndDate <= '2022-12-31'`
3	This query retrieves the billing ID and subscription ID for bills with an amount due greater than 3000 and a due date before the current date by joining the Billing and Subscriptions tables on the SubscriptionID column and filtering the results with a WHERE clause.	`SELECT Billing.BillingID, Subscriptions.SubscriptionID FROM Billing INNER JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID WHERE Billing.AmountDue > 3000 AND Billing.DueDate < GETDATE()`

4	This query retrieves the City and ZipCode columns from the ServiceOffices table and joins it with the Addresses table on the City column. The results are filtered to only include rows where the ZipCode in the ServiceOffices table starts with '404' or where the District in the Addresses table is 'Lahore'.	SELECT ServiceOffices.City, ServiceOffices.ZipCode FROM ServiceOffices INNER JOIN Addresses ON ServiceOffices.City = Addresses.City WHERE ServiceOffices.ZipCode LIKE '404%' OR Addresses.District = 'Lahore
5	This query retrieves the PromotionName column from the Promotions table and the ServiceName column from the Servicess table. The two tables are joined on the PromotionName and ServiceName columns and the results are filtered based on the StartDate and EndDate in the Promotions table.	SELECT Promotions.PromotionName, Servicess.ServiceName FROM Promotions INNER JOIN Servicess ON Promotions.PromotionName = Servicess.ServiceName WHERE Promotions.StartDate >= '2022-01-01' OR Promotions.EndDate <= '2022-12-31'
6	This query returns ContractID, FirstName, and LastName for contracts starting on or after '2022-01-01' or ending on or before '2022-12-31'.	SELECT Contracts.ContractID, Customers.FirstName, Customers.LastName FROM Contracts INNER JOIN Customers ON Contracts.CustomerID = Customers.CustomerID WHERE Contracts.StartDate >= '2022-01-01' OR Contracts.EndDate <= '2022-12-31'
7	This query returns PaymentID and BillingID for payments not greater than 4000 and not earlier than the current date.	SELECT Payments.PaymentID, Billing.BillingID FROM Payments INNER JOIN Billing ON Payments.BillingID = Billing.BillingID WHERE NOT (Payments.PaymentAmount > 4000 AND Payments.PaymentDate < GETDATE())
8	This query returns UsageID and SubscriptionID for usage not greater than 50 and not earlier than the current date.	SELECT Usage.UsageID, Subscriptions.SubscriptionID FROM Usage INNER JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID WHERE NOT (Usage.DataUsed > 50 AND Usage.UsageDate < GETDATE())
9	This query returns IssueID, FirstName, and LastName for issues not resolved and not created after '2022-01-01'.	SELECT Issues.IssueID, Technicians.FirstName, Technicians.LastName FROM Issues INNER JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID WHERE NOT (Issues.DateResolved IS NOT NULL AND Issues.DateCreated > '2022-01-01')
10	This query returns customers' first and last names with their subscription count in descending order.	SELECT Customers.FirstName, Customers.LastName, COUNT(Subscriptions.SubscriptionID) AS SubscriptionCount FROM Customers INNER JOIN Subscriptions

		ON Customers.CustomerID = Subscriptions.CustomerID GROUP BY Customers.FirstName, Customers.LastName ORDER BY SubscriptionCount DESC;
11	This query returns service names with their subscription count in descending order.	SELECT Servicess.ServiceName, COUNT(Subscriptions.SubscriptionID) AS SubscriptionCount FROM Servicess INNER JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID GROUP BY Servicess.ServiceName ORDER BY SubscriptionCount DESC;
12	This query returns billing dates with their total amount due in descending order.	SELECT Billing.BillingDate, SUM(Billing.AmountDue) AS TotalAmountDue FROM Billing INNER JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID GROUP BY Billing.BillingDate ORDER BY TotalAmountDue DESC;
13	This query returns payment dates with their total payment amount in descending order.	SELECT Payments.PaymentDate, SUM(Payments.PaymentAmount) AS TotalPaymentAmount FROM Payments INNER JOIN Billing ON Payments.BillingID = Billing.BillingID GROUP BY Payments.PaymentDate ORDER BY TotalPaymentAmount DESC;
14	This query returns usage dates with their total data used in descending order.	SELECT Usage.UsageDate, SUM(Usage.DataUsed) AS TotalDataUsed FROM Usage INNER JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID GROUP BY Usage.UsageDate ORDER BY TotalDataUsed DESC;
15	This query returns issue creation dates with their issue count in descending order.	SELECT Issues.DateCreated, COUNT(Issues.IssueID) AS IssueCount FROM Issues INNER JOIN Customers ON Issues.CustomerID = Customers.CustomerID GROUP BY Issues.DateCreated ORDER BY IssueCount DESC;
16	This query returns technicians' first and last names with their issue count in descending order.	SELECT Technicians.FirstName, Technicians.LastName, COUNT(Issues.IssueID) AS IssueCount FROM Issues INNER JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID GROUP BY Technicians.FirstName, Technicians.LastName ORDER BY IssueCount DESC;
17	This query returns contract start dates with their contract count in descending order.	SELECT Contracts.StartDate, COUNT(Contracts.ContractID) AS ContractCount FROM Contracts INNER JOIN Customers ON Contracts.CustomerID = Customers.CustomerID

		GROUP BY Contracts.StartDate ORDER BY ContractCount DESC;
18	This query returns service names with their contract count in descending order.	SELECT Servicess.ServiceName, COUNT(Contracts.ContractID) AS ContractCount FROM Contracts INNER JOIN Servicess ON Contracts.ServiceID = Servicess.ServiceID GROUP BY Servicess.ServiceName ORDER BY ContractCount DESC;
19	This query returns customers' first and last names with their total amount due in descending order.	SELECT Customers.FirstName, Customers.LastName, SUM(Billing.AmountDue) AS TotalAmountDue FROM Customers INNER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID INNER JOIN Billing ON Subscriptions.SubscriptionID = Billing.SubscriptionID GROUP BY Customers.FirstName, Customers.LastName ORDER BY TotalAmountDue DESC;
20	This query returns service names with their total amount due in descending order.	SELECT Servicess.ServiceName, SUM(Billing.AmountDue) AS TotalAmountDue FROM Servicess INNER JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID INNER JOIN Billing ON Subscriptions.SubscriptionID = Billing.SubscriptionID GROUP BY Servicess.ServiceName ORDER BY TotalAmountDue DESC;
21	This query returns customers' first and last names with their total payment amount in descending order.	SELECT Customers.FirstName, Customers.LastName, SUM(Payments.PaymentAmount) AS TotalPaymentAmount FROM Customers INNER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID INNER JOIN Billing ON Subscriptions.SubscriptionID = Billing.SubscriptionID INNER JOIN Payments ON Billing.BillingID = Payments.BillingID GROUP BY Customers.FirstName, Customers.LastName ORDER BY TotalPaymentAmount DESC;
22	This query returns service names with their total payment amount in descending order.	SELECT Servicess.ServiceName, SUM(Payments.PaymentAmount) AS TotalPaymentAmount FROM Servicess INNER JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID INNER JOIN Billing ON Subscriptions.SubscriptionID = Billing.SubscriptionID

		INNER JOIN Payments ON Billing.BillingID = Payments.BillingID GROUP BY Servicess.ServiceName ORDER BY TotalPaymentAmount DESC;
23	This query returns customers' first and last names with their total data used in descending order.	SELECT Customers.FirstName, Customers.LastName, SUM(Usage.DataUsed) AS TotalDataUsed FROM Customers INNER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID INNER JOIN Usage ON Subscriptions.SubscriptionID = Usage.SubscriptionID GROUP BY Customers.FirstName, Customers.LastName ORDER BY TotalDataUsed DESC;
24	This query returns service names with their total data used in descending order.	SELECT Servicess.ServiceName, SUM(Usage.DataUsed) AS TotalDataUsed FROM Servicess INNER JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID INNER JOIN Usage ON Subscriptions.SubscriptionID = Usage.SubscriptionID GROUP BY Servicess.ServiceName ORDER BY TotalDataUsed DESC;
25	This query returns customers' first and last names with their issue count in descending order.	SELECT Customers.FirstName, Customers.LastName, COUNT(Issues.IssueID) AS IssueCount FROM Customers INNER JOIN Issues ON Customers.CustomerID = Issues.CustomerID GROUP BY Customers.FirstName, Customers.LastName ORDER BY IssueCount DESC;
26	This query returns technicians' first and last names with their issue count in descending order.	SELECT Technicians.FirstName, Technicians.LastName, COUNT(Issues.IssueID) AS IssueCount FROM Technicians INNER JOIN Issues ON Technicians.TechnicianID = Issues.TechnicianID GROUP BY Technicians.FirstName, Technicians.LastName ORDER BY IssueCount DESC;
27	This query returns customers' first and last names with their contract count in descending order.	SELECT Customers.FirstName, Customers.LastName, COUNT(Contracts.ContractID) AS ContractCount FROM Customers INNER JOIN Contracts ON Customers.CustomerID = Contracts.CustomerID GROUP BY Customers.FirstName, Customers.LastName ORDER BY ContractCount DESC;

28	This query returns service names with their contract count in descending order.	SELECT Servicess.ServiceName, COUNT(Contracts.ContractID) AS ContractCount FROM Servicess INNER JOIN Contracts ON Servicess.ServiceID = Contracts.ServiceID GROUP BY Servicess.ServiceName ORDER BY ContractCount DESC;
29	This query returns customers' first and last names with their balance (amount due minus payment amount) in descending order for balances greater than 0.	SELECT Customers.FirstName, Customers.LastName, SUM(Billing.AmountDue - Payments.PaymentAmount) AS Balance FROM Customers INNER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID INNER JOIN Billing ON Subscriptions.SubscriptionID = Billing.SubscriptionID INNER JOIN Payments ON Billing.BillingID = Payments.BillingID GROUP BY Customers.FirstName, Customers.LastName HAVING SUM(Billing.AmountDue - Payments.PaymentAmount) > 0 ORDER BY Balance DESC;
30	This query returns service names with their balance (amount due minus payment amount) in descending order for balances greater than 0.	SELECT Servicess.ServiceName, SUM(Billing.AmountDue - Payments.PaymentAmount) AS Balance FROM Servicess INNER JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID INNER JOIN Billing ON Subscriptions.SubscriptionID = Billing.SubscriptionID INNER JOIN Payments ON Billing.BillingID = Payments.BillingID GROUP BY Servicess.ServiceName HAVING SUM(Billing.AmountDue - Payments.PaymentAmount) > 0 ORDER BY Balance DESC;

18. LEFT joins– 20 Queries

1	This query returns all columns from the Customers and Addresses tables by performing a LEFT JOIN on the AddressID column.	SELECT * FROM Customers LEFT JOIN Addresses ON Customers.AddressID = Addresses.AddressID;
2	This query returns all columns from the Customers and Subscriptions tables by performing a LEFT JOIN on the CustomerID column.	SELECT * FROM Customers LEFT JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID;

3	This query returns all columns from the Servicess and Subscriptions tables by performing a LEFT JOIN on the ServiceID column.	`SELECT * FROM Servicess LEFT JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID;`
4	This query returns all columns from the Billing and Subscriptions tables by performing a LEFT JOIN on the SubscriptionID column.	`SELECT * FROM Billing LEFT JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID;`
5	This query returns all columns from the Payments and Billing tables by performing a LEFT JOIN on the BillingID column.	`SELECT * FROM Payments LEFT JOIN Billing ON Payments.BillingID = Billing.BillingID;`
6	This query returns all columns from the Usage and Subscriptions tables by performing a LEFT JOIN on the SubscriptionID column.	`SELECT * FROM Usage LEFT JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID;`
7	This query returns all columns from the Issues and Customers tables by performing a LEFT JOIN on the CustomerID column.	`SELECT * FROM Issues LEFT JOIN Customers ON Issues.CustomerID = Customers.CustomerID;`
8	This query returns all columns from the Issues and Technicians tables by performing a LEFT JOIN on the TechnicianID column.	`SELECT * FROM Issues LEFT JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID;`
9	This query returns all columns from the Contracts and Customers tables by performing a LEFT JOIN on the CustomerID column.	`SELECT * FROM Contracts LEFT JOIN Customers ON Contracts.CustomerID = Customers.CustomerID;`
10	This query returns all columns from Contracts and Servicess tables by performing a LEFT JOIN on ServiceID column.	`SELECT * FROM Contracts LEFT JOIN Servicess ON Contracts.ServiceID = Servicess.ServiceID;`
11	This query returns customers' first and last names with their street address and city.	`SELECT Customers.FirstName, Customers.LastName, Addresses.StreetAddress, Addresses.City FROM Customers LEFT JOIN Addresses ON Customers.AddressID = Addresses.AddressID;`

12	This query returns customers' first and last names with their subscription start and end dates.	SELECT Customers.FirstName, Customers.LastName, Subscriptions.StartDate, Subscriptions.EndDate FROM Customers LEFT JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID;
13	This query returns service names with their monthly fee, subscription start date, and end date.	SELECT Servicess.ServiceName, Servicess.MonthlyFee, Subscriptions.StartDate, Subscriptions.EndDate FROM Servicess LEFT JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID;
14	This query returns billing dates with their amount due, due date, subscription start date, and end date.	SELECT Billing.BillingDate, Billing.AmountDue, Billing.DueDate, Subscriptions.StartDate, Subscriptions.EndDate FROM Billing LEFT JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID;
15	This query returns payment dates with their payment amount, billing date, and amount due.	SELECT Payments.PaymentDate, Payments.PaymentAmount, Billing.BillingDate, Billing.AmountDue FROM Payments LEFT JOIN Billing ON Payments.BillingID = Billing.BillingID;
16	This query returns usage dates with their data used, usage bytes, subscription start date, and end date.	SELECT Usage.UsageDate, Usage.DataUsed, Usage.UsageBytes, Subscriptions.StartDate, Subscriptions.EndDate FROM Usage LEFT JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID;
17	This query returns issue descriptions with their creation date, resolution date, customer first name, and last name.	SELECT Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved, Customers.FirstName, Customers.LastName FROM Issues LEFT JOIN Customers ON Issues.CustomerID = Customers.CustomerID;
18	This query returns issue descriptions with their creation date, resolution date, technician first name, and last name.	SELECT Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved, Technicians.FirstName, Technicians.LastName FROM Issues LEFT JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID;
19	This query returns contract start dates with their end date, customer first name, and last name.	SELECT Contracts.StartDate, Contracts.EndDate, Customers.FirstName, Customers.LastName FROM Contracts LEFT JOIN Customers ON Contracts.CustomerID = Customers.CustomerID;

20	This query returns contract start dates with their end date, service name, and monthly fee.	SELECT Contracts.StartDate, Contracts.EndDate, Servicess.ServiceName, Servicess.MonthlyFee FROM Contracts LEFT JOIN Servicess ON Contracts.ServiceID = Servicess.ServiceID;

19. RIGHT joins– 20 Queries

1	This query returns all columns from the Customers and Addresses tables by performing a RIGHT JOIN on the AddressID column.	SELECT * FROM Customers RIGHT JOIN Addresses ON Customers.AddressID = Addresses.AddressID;
2	This query returns all columns from the Customers and Subscriptions tables by performing a RIGHT JOIN on the CustomerID column.	SELECT * FROM Customers RIGHT JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID;
3	This query returns all columns from the Servicess and Subscriptions tables by performing a RIGHT JOIN on the ServiceID column.	SELECT * FROM Servicess RIGHT JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID;
4	This query returns all columns from the Billing and Subscriptions tables by performing a RIGHT JOIN on the SubscriptionID column.	SELECT * FROM Billing RIGHT JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID;
5	This query returns all columns from the Payments and Billing tables by performing a RIGHT JOIN on the BillingID column.	SELECT * FROM Payments RIGHT JOIN Billing ON Payments.BillingID = Billing.BillingID;
6	This query returns all columns from the Usage and Subscriptions tables by performing a RIGHT JOIN on the SubscriptionID column.	SELECT * FROM Usage RIGHT JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID;
7	This query returns all columns from the Issues and Customers tables by performing a RIGHT JOIN on the CustomerID column.	SELECT * FROM Issues RIGHT JOIN Customers ON Issues.CustomerID = Customers.CustomerID;
8	This query returns all columns from the Issues and Technicians tables by performing a RIGHT JOIN on TechnicianID column.	SELECT * FROM Issues RIGHT JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID;

9	This query returns all columns from Contracts and Customers tables by performing a RIGHT JOIN on CustomerID column.	`SELECT * FROM Contracts RIGHT JOIN Customers ON Contracts.CustomerID = Customers.CustomerID;`
10	This query returns all columns from Contracts and Servicess tables by performing a RIGHT JOIN on ServiceID column.	`SELECT * FROM Contracts RIGHT JOIN Servicess ON Contracts.ServiceID = Servicess.ServiceID;`
11	This query returns customers' first and last names with their street address and city.	`SELECT Customers.FirstName, Customers.LastName, Addresses.StreetAddress, Addresses.City FROM Customers RIGHT JOIN Addresses ON Customers.AddressID = Addresses.AddressID;`
12	This query returns customers' first and last names with their subscription start and end dates.	`SELECT Customers.FirstName, Customers.LastName, Subscriptions.StartDate, Subscriptions.EndDate FROM Customers RIGHT JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID;`
13	This query returns service names with their monthly fee, subscription start date, and end date.	`SELECT Servicess.ServiceName, Servicess.MonthlyFee, Subscriptions.StartDate, Subscriptions.EndDate FROM Servicess RIGHT JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID;`
14	This query returns billing dates with their amount due, due date, subscription start date, and end date.	`SELECT Billing.BillingDate, Billing.AmountDue, Billing.DueDate, Subscriptions.StartDate, Subscriptions.EndDate FROM Billing RIGHT JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID;`
15	This query returns payment dates with their payment amount, billing date, and amount due.	`SELECT Payments.PaymentDate, Payments.PaymentAmount, Billing.BillingDate, Billing.AmountDue FROM Payments RIGHT JOIN Billing ON Payments.BillingID = Billing.BillingID;`
16	This query returns usage dates with their data used, usage bytes, subscription start date, and end date.	`SELECT Usage.UsageDate, Usage.DataUsed, Usage.UsageBytes, Subscriptions.StartDate, Subscriptions.EndDate FROM Usage RIGHT JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID;`
17	This query returns issue descriptions with their creation date, resolution	`SELECT Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved,`

	date, customer first name, and last name.	`Customers.FirstName, Customers.LastName FROM Issues RIGHT JOIN Customers ON Issues.CustomerID = Customers.CustomerID;`
18	This query returns issue descriptions with their creation date, resolution date, technician first name, and last name.	`SELECT Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved, Technicians.FirstName, Technicians.LastName FROM Issues RIGHT JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID;`
19	This query returns contract start dates with their end date, customer first name, and last name.	`SELECT Contracts.StartDate, Contracts.EndDate, Customers.FirstName, Customers.LastName FROM Contracts RIGHT JOIN Customers ON Contracts.CustomerID = Customers.CustomerID;`
20	This query returns contract start dates with their end date, service name, and monthly fee.	`SELECT Contracts.StartDate, Contracts.EndDate, Servicess.ServiceName, Servicess.MonthlyFee FROM Contracts RIGHT JOIN Servicess ON Contracts.ServiceID = Servicess.ServiceID;`

20. FULL OUTER joins– 20 Queries

1	This query returns all columns from the Customers and Addresses tables by performing a FULL OUTER JOIN on the AddressID column.	`SELECT * FROM Customers FULL OUTER JOIN Addresses ON Customers.AddressID = Addresses.AddressID;`
2	This query returns all columns from the Customers and Subscriptions tables by performing a FULL OUTER JOIN on the CustomerID column.	`SELECT * FROM Customers FULL OUTER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID;`
3	This query returns all columns from the Servicess and Subscriptions tables by performing a FULL OUTER JOIN on the ServiceID column.	`SELECT * FROM Servicess FULL OUTER JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID;`
4	This query returns all columns from the Billing and Subscriptions tables by performing a FULL OUTER JOIN on the SubscriptionID column.	`SELECT * FROM Billing FULL OUTER JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID;`
5	This query returns all columns from the Payments and Billing tables by performing a FULL OUTER JOIN on the BillingID column.	`SELECT * FROM Payments FULL OUTER JOIN Billing ON Payments.BillingID = Billing.BillingID;`

6	This query returns all columns from the Usage and Subscriptions tables by performing a FULL OUTER JOIN on the SubscriptionID column.	`SELECT * FROM Usage`
		`FULL OUTER JOIN Subscriptions`
		`ON Usage.SubscriptionID =`
		`Subscriptions.SubscriptionID;`

7	This query returns all columns from the Issues and Customers tables by performing a FULL OUTER JOIN on CustomerID column.	`SELECT * FROM Issues`
		`FULL OUTER JOIN Customers`
		`ON Issues.CustomerID =`
		`Customers.CustomerID;`

8	This query returns all columns from Issues and Technicians tables by performing a FULL OUTER JOIN on TechnicianID column.	`SELECT * FROM Issues`
		`FULL OUTER JOIN Technicians`
		`ON Issues.TechnicianID =`
		`Technicians.TechnicianID;`

9	This query returns all columns from Contracts and Customers tables by performing a FULL OUTER JOIN on CustomerID column.	`SELECT * FROM Contracts`
		`FULL OUTER JOIN Customers`
		`ON Contracts.CustomerID =`
		`Customers.CustomerID;`

10	This query returns all columns from Contracts and Servicess tables by performing a FULL OUTER JOIN on ServiceID column.	`SELECT * FROM Contracts`
		`FULL OUTER JOIN Servicess`
		`ON Contracts.ServiceID =`
		`Servicess.ServiceID;`

11	This query returns customers' first and last names with their street address and city.	`SELECT Customers.FirstName,`
		`Customers.LastName,`
		`Addresses.StreetAddress, Addresses.City`
		`FROM Customers`
		`FULL OUTER JOIN Addresses`
		`ON Customers.AddressID =`
		`Addresses.AddressID;`

12	This query returns customers' first and last names with their subscription start and end dates.	`SELECT Customers.FirstName,`
		`Customers.LastName,`
		`Subscriptions.StartDate,`
		`Subscriptions.EndDate FROM Customers`
		`FULL OUTER JOIN Subscriptions`
		`ON Customers.CustomerID =`
		`Subscriptions.CustomerID;`

13	This query returns service names with their monthly fee, subscription start date, and end date.	`SELECT Servicess.ServiceName,`
		`Servicess.MonthlyFee,`
		`Subscriptions.StartDate,`
		`Subscriptions.EndDate FROM Servicess`
		`FULL OUTER JOIN Subscriptions`
		`ON Servicess.ServiceID =`
		`Subscriptions.ServiceID;`

14	This query returns billing dates with their amount due, due date, subscription start date, and end date.	`SELECT Billing.BillingDate,`
		`Billing.AmountDue, Billing.DueDate,`
		`Subscriptions.StartDate,`
		`Subscriptions.EndDate FROM Billing`
		`FULL OUTER JOIN Subscriptions`

		ON Billing.SubscriptionID = Subscriptions.SubscriptionID;
15	This query returns payment dates with their payment amount, billing date, and amount due.	SELECT Payments.PaymentDate, Payments.PaymentAmount, Billing.BillingDate, Billing.AmountDue FROM Payments FULL OUTER JOIN Billing ON Payments.BillingID = Billing.BillingID;
16	This query returns usage dates with their data used, usage bytes, subscription start date, and end date.	SELECT Usage.UsageDate, Usage.DataUsed, Usage.UsageBytes, Subscriptions.StartDate, Subscriptions.EndDate FROM Usage FULL OUTER JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID;
17	This query returns issue descriptions with their creation date, resolution date, customer first name, and last name.	SELECT Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved, Customers.FirstName, Customers.LastName FROM Issues FULL OUTER JOIN Customers ON Issues.CustomerID = Customers.CustomerID;
18	This query returns issue descriptions with their creation date, resolution date, technician first name, and last name.	SELECT Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved, Technicians.FirstName, Technicians.LastName FROM Issues FULL OUTER JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID;
19	This query returns contract start dates with their end date, customer first name, and last name.	SELECT Contracts.StartDate, Contracts.EndDate, Customers.FirstName, Customers.LastName FROM Contracts FULL OUTER JOIN Customers ON Contracts.CustomerID = Customers.CustomerID;
20	This query returns contract start dates with their end date, service name, and monthly fee.	SELECT Contracts.StartDate, Contracts.EndDate, Servicess.ServiceName, Servicess.MonthlyFee FROM Contracts FULL OUTER JOIN Servicess ON Contracts.ServiceID = Servicess.ServiceID;

21. STORED Procedures without Parameter– 25 Queries

1	GetAllCustomers returns all columns from the Customers table	CREATE procedure GetAllCustomers
		AS
		BEGIN
		  SELECT * FROM Customers;
		END;

2	GetAllSubscriptions returns all columns from the Subscriptions table.	CREATE PROCEDURE GetAllSubscriptions
		AS
		BEGIN
		  SELECT * FROM Subscriptions;
		END;

3	GetAllServices returns all columns from the Servicess table.	CREATE PROCEDURE GetAllServices
		AS
		BEGIN
		  SELECT * FROM Servicess;
		END;

4	GetAllBilling returns all columns from the Billing table.	CREATE PROCEDURE GetAllBilling
		AS
		BEGIN
		    SELECT * FROM Billing;
		END;

5	GetAllPayments returns all columns from the Payments table.	CREATE PROCEDURE GetAllPayments
		AS
		BEGIN
		    SELECT * FROM Payments;
		END;

6	GetAllUsage returns all columns from the Usage table.	CREATE PROCEDURE GetAllUsage
		AS
		BEGIN
		    SELECT * FROM Usage;
		END;

7	GetAllIssues returns all columns from the Issues table.	CREATE PROCEDURE GetAllIssues
		AS
		BEGIN
		    SELECT * FROM Issues;
		END;

8	GetAllTechnicians returns all columns from the Technicians table.	CREATE PROCEDURE GetAllTechnicians
		AS
		BEGIN
		    SELECT * FROM Technicians;
		END;

9	GetAllEquipment returns all columns from the Equipment table.	CREATE PROCEDURE GetAllEquipment
		AS
		BEGIN
		    SELECT * FROM Equipment;
		END;

10	GetCustomersWithSubscriptionsCountByServiceName takes a service name as a parameter and returns customers' first and last names with their subscription count for that service in descending order.	CREATE PROCEDURE GetCustomersWithSubscriptionsCountByServiceName @ServiceName VARCHAR(255)
		AS
		BEGIN
		    SELECT Customers.FirstName, Customers.LastName, COUNT(Subscriptions.SubscriptionID) AS SubscriptionCount FROM Customers INNER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID INNER JOIN Servicess ON Subscriptions.ServiceID = Servicess.ServiceID WHERE Servicess.ServiceName = @ServiceName GROUP BY Customers.FirstName, Customers.LastName ORDER BY SubscriptionCount DESC;
		END;

11	GetCustomersWithSubscriptionsCount returns customers' first and last names with their subscription count in descending order.	CREATE PROCEDURE GetCustomersWithSubscriptionsCount AS BEGIN SELECT Customers.FirstName, Customers.LastName, COUNT(Subscriptions.SubscriptionID) AS SubscriptionCount FROM Customers

		INNER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID GROUP BY Customers.FirstName, Customers.LastName ORDER BY SubscriptionCount DESC; END;

12	GetServicesWithSubscriptionsCount returns service names with their subscription count in descending order.	CREATE PROCEDURE GetServicesWithSubscriptionsCount
		AS
		BEGIN
		SELECT Servicess.ServiceName, COUNT(Subscriptions.SubscriptionID) AS SubscriptionCount FROM Servicess INNER JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID GROUP BY Servicess.ServiceName ORDER BY SubscriptionCount DESC;
		END;

13	GetBillingTotalAmountDueByDate returns billing dates with their total amount due in descending order.	CREATE PROCEDURE GetBillingTotalAmountDueByDate
		AS
		BEGIN
		SELECT Billing.BillingDate, SUM(Billing.AmountDue) AS TotalAmountDue FROM Billing INNER JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID GROUP BY Billing.BillingDate ORDER BY TotalAmountDue DESC;
		END;

14	GetPaymentsTotalAmountByDate returns payment dates with their total payment amount in descending order.	CREATE PROCEDURE GetPaymentsTotalAmountByDate
		AS
		BEGIN
		SELECT Payments.PaymentDate, SUM(Payments.PaymentAmount) AS TotalPaymentAmount FROM Payments INNER JOIN Billing ON Payments.BillingID = Billing.BillingID GROUP BY Payments.PaymentDate ORDER BY TotalPaymentAmount DESC;
		END;

15	GetUsageTotalDataUsedByDate returns usage dates with their total data used in descending order.	CREATE PROCEDURE GetUsageTotalDataUsedByDate AS BEGIN SELECT Usage.UsageDate, SUM(Usage.DataUsed) AS TotalDataUsed FROM Usage INNER JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID GROUP BY Usage.UsageDate ORDER BY TotalDataUsed DESC;

		END;
16	GetIssuesCountByDateCreated returns issue creation dates with their issue count in descending order.	CREATE PROCEDURE GetIssuesCountByDateCreated AS BEGIN SELECT Issues.DateCreated, COUNT(Issues.IssueID) AS IssueCount FROM Issues INNER JOIN Customers ON Issues.CustomerID = Customers.CustomerID GROUP BY Issues.DateCreated ORDER BY IssueCount DESC; END;
17	GetTechniciansWithIssueCount returns technicians' first and last names with their issue count in descending order.	CREATE PROCEDURE GetTechniciansWithIssueCount AS BEGIN SELECT Technicians.FirstName, Technicians.LastName, COUNT(Issues.IssueID) AS IssueCount FROM Issues INNER JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID GROUP BY Technicians.FirstName, Technicians.LastName ORDER BY IssueCount DESC; END;
18	GetCustomersWithContractCount returns customers' first and last names with their contract count in descending order.	CREATE PROCEDURE GetCustomersWithContractCount AS BEGIN SELECT Customers.FirstName, Customers.LastName, COUNT(Contracts.ContractID) AS ContractCount FROM Customers INNER JOIN Contracts ON Customers.CustomerID = Contracts.CustomerID GROUP BY Customers.FirstName, Customers.LastName ORDER BY ContractCount DESC; END;
19	GetServicesWithContractCount returns service names with their contract count in descending order.	CREATE PROCEDURE GetServicesWithContractCount AS BEGIN SELECT Servicess.ServiceName, COUNT(Contracts.ContractID) AS ContractCount FROM Servicess INNER JOIN Contracts ON Servicess.ServiceID = Contracts.ServiceID GROUP BY Servicess.ServiceName ORDER BY ContractCount DESC; END;
20	GetCustomersWithActiveSubscriptionsCount returns customers' first and last names with their active service count in descending order.	CREATE PROCEDURE GetCustomersWithActiveSubscriptionsCount

		AS
		BEGIN
		SELECT Customers.FirstName, Customers.LastName, COUNT(DISTINCT Servicess.ServiceName) AS ServiceCount FROM Customers INNER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID INNER JOIN Servicess ON Subscriptions.ServiceID = Servicess.ServiceID WHERE Subscriptions.EndDate IS NULL GROUP BY Customers.FirstName, Customers.LastName ORDER BY ServiceCount DESC;
		END;

21	GetServicesWithActiveSubscriptionsCount returns service names with their active customer count in descending order.	CREATE PROCEDURE GetServicesWithActiveSubscriptionsCount
		AS
		BEGIN
		SELECT Servicess.ServiceName, COUNT(DISTINCT Customers.CustomerID) AS CustomerCount FROM Servicess INNER JOIN Subscriptions ON Servicess.ServiceID = Subscriptions.ServiceID INNER JOIN Customers ON Subscriptions.CustomerID = Customers.CustomerID WHERE Subscriptions.EndDate IS NULL GROUP BY Servicess.ServiceName ORDER BY CustomerCount DESC;
		END;

22	GetBillingUnpaidBillCountByBillingDate returns billing dates with their unpaid bill count in descending order.	CREATE PROCEDURE GetBillingUnpaidBillCountByBillingDate
		AS
		BEGIN
		SELECT Billing.BillingDate, SUM(CASE WHEN Payments.PaymentDate IS NULL THEN 1 ELSE 0 END) AS UnpaidBillCount FROM Billing LEFT JOIN Payments ON Billing.BillingID = Payments.BillingID GROUP BY Billing.BillingDate ORDER BY UnpaidBillCount DESC;
		END;

23	GetBillingUnpaidBillCountByDueDate returns due dates with their unpaid bill count in descending order for due dates earlier than the current date.	CREATE PROCEDURE GetBillingUnpaidBillCountByDueDate AS BEGIN SELECT Billing.DueDate, SUM(CASE WHEN Payments.PaymentDate IS NULL THEN 1 ELSE 0 END) AS UnpaidBillCount FROM Billing LEFT JOIN Payments ON Billing.BillingID = Payments.BillingID WHERE Billing.DueDate < GETDATE() GROUP BY

		Billing.DueDate ORDER BY UnpaidBillCount DESC; END;

24	GetIssuesOpenIssueCountByDateCreated returns issue creation dates with their open issue count in descending order.	CREATE PROCEDURE GetIssuesOpenIssueCountByDateCreated
		AS
		BEGIN
		SELECT Issues.DateCreated, COUNT(CASE WHEN Issues.DateResolved IS NULL THEN 1 ELSE 0 END) AS OpenIssueCount FROM Issues GROUP BY Issues.DateCreated ORDER BY OpenIssueCount DESC;
		END;

25	GetTechniciansWithOpenIssueCount returns technicians' first and last names with their open issue count in descending order.	CREATE PROCEDURE GetTechniciansWithOpenIssueCount
		AS
		BEGIN
		SELECT Technicians.FirstName, Technicians.LastName, COUNT(CASE WHEN Issues.DateResolved IS NULL THEN 1 ELSE 0 END) AS OpenIssueCount FROM Technicians LEFT JOIN Issues ON Technicians.TechnicianID = Issues.TechnicianID GROUP BY Technicians.FirstName, Technicians.LastName ORDER BY OpenIssueCount DESC;
		END;

22.STORED Procedures with Parameter– 25 Queries

1	GetCustomerByID takes a customer ID as a parameter and returns all columns from the Customers table for that customer.	CREATE PROCEDURE GetCustomerByID
		@CustomerID INT
		AS
		BEGIN
		SELECT * FROM Customers WHERE CustomerID = @CustomerID;
		END;

2	GetAddressByID takes an address ID as a parameter and returns all columns from the Addresses table for that address.	CREATE PROCEDURE GetAddressByID
		@AddressID INT
		AS
		BEGIN
		SELECT * FROM Addresses WHERE AddressID = @AddressID;
		END;

3	GetSubscriptionByID takes a subscription ID as a parameter and returns all columns from the Subscriptions table for that subscription.	CREATE PROCEDURE GetSubscriptionByID
		@SubscriptionID INT
		AS
		BEGIN
		SELECT * FROM Subscriptions WHERE SubscriptionID = @SubscriptionID;
		END;

4	GetBillingByID takes a billing ID as a parameter and returns all columns from the Billing table for that billing.	CREATE PROCEDURE GetBillingByID
		@BillingID INT
		AS
		BEGIN
		SELECT * FROM Billing WHERE BillingID = @BillingID;
		END;

5	GetPaymentByID takes a payment ID as a parameter and returns all columns from the Payments table for that payment.	CREATE PROCEDURE GetPaymentByID
		@PaymentID INT
		AS
		BEGIN
		SELECT * FROM Payments WHERE PaymentID = @PaymentID;
		END;

6	GetUsageByID takes a usage ID as a parameter and returns all columns from the Usage table for that usage.	CREATE PROCEDURE GetUsageByID @UsageID INT
		AS
		BEGIN
		SELECT * FROM Usage WHERE UsageID = @UsageID;
		END;

7	GetIssueByID takes an issue ID as a parameter and returns all columns from the Issues table for that issue.	CREATE PROCEDURE GetIssueByID @IssueID INT
		AS
		BEGIN
		SELECT * FROM Issues WHERE IssueID = @IssueID;
		END;

8	GetTechnicianByID takes a technician ID as a parameter and returns all columns from the Technicians table for that technician.	CREATE PROCEDURE GetTechnicianByID
		@TechnicianID INT
		AS
		BEGIN
		SELECT * FROM Technicians WHERE TechnicianID = @TechnicianID;
		END;

9	GetEquipmentByID takes an equipment ID as a parameter and returns all columns from the Equipment table for that equipment.	CREATE PROCEDURE GetEquipmentByID
		@EquipmentID INT
		AS
		BEGIN
		SELECT * FROM Equipment WHERE EquipmentID = @EquipmentID;
		END;

10	GetCustomersWithSubscriptionsCountByServiceNamee takes a service name as a parameter and returns customers' first and last names with their subscription count for that service in descending order.	CREATE PROCEDURE GetCustomersWithSubscriptionsCountByServiceNamee @ServiceName VARCHAR(255) AS BEGIN SELECT Customers.FirstName, Customers.LastName, COUNT(Subscriptions.SubscriptionID) AS SubscriptionCount FROM Customers INNER JOIN Subscriptions ON Customers.CustomerID = Subscriptions.CustomerID INNER JOIN Servicess

		ON Subscriptions.ServiceID = Servicess.ServiceID WHERE Servicess.ServiceName = @ServiceName GROUP BY Customers.FirstName, Customers.LastName ORDER BY SubscriptionCount DESC; END;

11	spInsertAddress takes street address, city, district, and zip code as parameters and inserts them into the Addresses table.	CREATE PROCEDURE spInsertAddress
		@StreetAddress VARCHAR(255),
		@City VARCHAR(255),
		@District VARCHAR(255),
		@ZipCode VARCHAR(255)
		AS
		BEGIN
		INSERT INTO Addresses (StreetAddress, City, District, ZipCode)
		VALUES (@StreetAddress, @City, @District, @ZipCode)
		END;

12	spInsertCustomer takes first name, last name, email, phone number, and address ID as parameters and inserts them into the Customers table.	CREATE PROCEDURE spInsertCustomer
		@FirstName VARCHAR(255),
		@LastName VARCHAR(255),
		@Email VARCHAR(255),
		@PhoneNumber VARCHAR(255),
		@AddressID INT
		AS
		BEGIN
		INSERT INTO Customers (FirstName, LastName, Email, PhoneNumber, AddressID)
		VALUES (@FirstName, @LastName, @Email, @PhoneNumber, @AddressID)
		END;

13	spInsertService takes service name, service description, and monthly fee as parameters and inserts them into the Servicess table.	CREATE PROCEDURE spInsertService
		@ServiceName VARCHAR(255),
		@ServiceDescription VARCHAR(255),
		@MonthlyFee float
		AS
		BEGIN
		INSERT INTO Servicess (ServiceName, ServiceDescription, MonthlyFee)
		VALUES (@ServiceName, @ServiceDescription, @MonthlyFee)
		END;

14	spInsertSubscription takes customer ID, service ID, start date, and end date as parameters and inserts them into the Subscriptions table.	CREATE PROCEDURE spInsertSubscription
		@CustomerID INT,
		@ServiceID INT,
		@StartDate DATE,
		@EndDate DATE
		AS
		BEGIN
		INSERT INTO Subscriptions (CustomerID, ServiceID, StartDate, EndDate)
		VALUES (@CustomerID, @ServiceID, @StartDate, @EndDate)
		END;

15	spInsertBilling takes subscription ID, billing date, amount due, and due date as parameters and inserts them into the Billing table.	CREATE PROCEDURE spInsertBilling
		@SubscriptionID INT,
		@BillingDate DATE,
		@AmountDue float,
		@DueDate DATE
		AS
		BEGIN
		INSERT INTO Billing (SubscriptionID,
		BillingDate, AmountDue, DueDate)
		VALUES
		(@SubscriptionID,@BillingDate,@AmountDue,
		@DueDate)
		END;

16	spInsertPayment takes billing ID, payment date, and payment amount as parameters and inserts them into the Payments table.	CREATE PROCEDURE spInsertPayment
		@BillingID INT,
		@PaymentDate DATE,
		@PaymentAmount float
		AS
		BEGIN
		INSERT INTO Payments (BillingID
		,PaymentDate ,PaymentAmount)
		VALUES (@BillingID ,@PaymentDate
		,@PaymentAmount)
		END;

17	spInsertUsage takes subscription ID, usage date, data used, and usage bytes as parameters and inserts them into the Usage table.	CREATE PROCEDURE spInsertUsage
		@SubscriptionID INT,
		@UsageDate DATE,
		@DataUsed float,
		@UsageBytes varchar(255)
		AS
		BEGIN
		INSERT INTO Usage (SubscriptionID
		,UsageDate ,DataUsed ,UsageBytes)
		VALUES (@SubscriptionID ,@UsageDate
		,@DataUsed ,@UsageBytes)
		END;

18	spInsertTechnician takes first name, last name, and phone number as parameters and inserts them into the Technicians table.	CREATE PROCEDURE spInsertTechnician
		@FirstName VARCHAR(255),
		@LastName VARCHAR(255),
		@PhoneNumber VARCHAR(255)
		AS
		BEGIN
		INSERT INTO Technicians (FirstName
		,LastName ,PhoneNumber)
		VALUES (@FirstName ,@LastName
		,@PhoneNumber)
		END;

19	spInsertIssue takes customer ID, issue description, date created, date resolved, and technician ID as parameters and inserts them into the Issues table.	CREATE PROCEDURE spInsertIssue @CustomerID INT, @IssueDescription VARCHAR(255), @DateCreated DATE, @DateResolved DATE, @TechnicianID INT AS BEGIN

		INSERT INTO Issues (CustomerID ,IssueDescription ,DateCreated ,DateResolved ,TechnicianID) VALUES (@CustomerID ,@IssueDescription ,@DateCreated ,@DateResolved ,@TechnicianID) END;

20	spInsertEquipment takes equipment name and equipment description as parameters and inserts them into the Equipment table.	CREATE PROCEDURE spInsertEquipment
		@EquipmentName VARCHAR(255),
		@EquipmentDescription VARCHAR(255)
		AS
		BEGIN
		INSERT INTO Equipment (EquipmentName
		,EquipmentDescription)
		VALUES (@EquipmentName
		,@EquipmentDescription)
		END;

21	spInsertServiceOffice takes city and zip code as parameters and inserts them into the ServiceOffices table.	CREATE PROCEDURE spInsertServiceOffice
		@City VARCHAR(255),
		@ZipCode VARCHAR(255)
		AS
		BEGIN
		INSERT INTO ServiceOffices (City ,ZipCode
		)
		VALUES (@City ,@ZipCode)
		END;

22	spInsertPromotion takes promotion name, promotion description, start date, and end date as parameters and inserts them into the Promotions table.	CREATE PROCEDURE spInsertPromotion
		@PromotionName VARCHAR(255),
		@PromotionDescription VARCHAR(255),
		@StartDate DATE,
		@EndDate DATE
		AS
		BEGIN
		INSERT INTO Promotions
		(PromotionName,PromotionDescription
		,StartDate ,EndDate)
		VALUES
		(@PromotionName,@PromotionDescription,@St
		artDate,@EndDate)
		END;

23	spInsertContract takes customer ID, service ID, start date, and end date as parameters and inserts them into the Contracts table.	CREATE PROCEDURE spInsertContract
		@CustomerID INT,
		@ServiceID INT,
		@StartDate DATE,
		@EndDate DATE
		AS
		BEGIN
		INSERT INTO Contracts (CustomerID
		,ServiceID ,StartDate ,EndDate)
		VALUES
		(@CustomerID,@ServiceID,@StartDate,@EndDa
		te)
		END;

24	spUpdateAddress takes address ID (required), street address (optional), city (optional), district (optional), zip code	CREATE PROCEDURE spUpdateAddress @AddressId INT, @StreetAddress VARCHAR(255), @City VARCHAR(255),

	(optional) as parameters to update an existing address in the Addresses table.	@District VARCHAR(255),
		@ZipCode VARCHAR(255)
		AS
		BEGIN
		UPDATE Addresses
		SET StreetAddress =
		COALESCE(@StreetAddress, StreetAddress),
		City = COALESCE(@City,City), District =
		COALESCE(@District,District), ZipCode =
		COALESCE(@ZipCode,ZipCode)
		WHERE AddressId =
		COALESCE(@AddressId,NULL);
		END;

25	spUpdateCustomer takes customer ID (required), first name (optional), last name (optional), email (optional), phone number (optional), address ID (optional) as parameters to update an existing customer in the Customers table.	CREATE PROCEDURE spUpdateCustomer
		@CustomerId INT,
		@FirstName VARCHAR(255),
		@LastName VARCHAR(255),
		@Email VARCHAR(255),
		@PhoneNumber VARCHAR(255),
		@AddressId INT
		AS
		BEGIN
		UPDATE Customers
		SET FirstName = COALESCE(@FirstName,
		FirstName), LastName =
		COALESCE(@LastName,LastName), Email =
		COALESCE(@Email,Email), PhoneNumber =
		COALESCE(@PhoneNumber,PhoneNumber),
		AddressId =
		COALESCE(@AddressId,AddressId)
		WHERE CustomerId =
		COALESCE(@CustomerId,NULL);
		END;

23. STORED Procedures with Parameter using Logical Operators and Group By– 30 Queries

1	Get Customers By City Or Last Name	CREATE PROCEDURE GetCustomersByCityOrLastName
		@City VARCHAR(255),
		@LastName VARCHAR(255)
		AS
		BEGIN
		SELECT Customers.FirstName, Customers.LastName, Addresses.City
		FROM Customers
		INNER JOIN Addresses ON Customers.AddressID =
		Addresses.AddressID
		WHERE Addresses.City = @City OR Customers.LastName = @LastName
		END;

2	Get customers by first name and last name	CREATE PROCEDURE GetCustomersByFirstAndLastName @FirstName VARCHAR(255), @LastName VARCHAR(255) AS BEGIN SELECT Customers.FirstName, Customers.LastName FROM Customers WHERE Customers.FirstName = @FirstName AND Customers.LastName = @LastName

		END;

3	Get customers by city or zip code	CREATE PROCEDURE GetCustomersByCityOrZipCode
		@City VARCHAR(255),
		@ZipCode VARCHAR(255)
		AS
		BEGIN
		SELECT Customers.FirstName, Customers.LastName, Addresses.City, Addresses.ZipCode
		FROM Customers
		INNER JOIN Addresses ON Customers.AddressID = Addresses.AddressID
		WHERE Addresses.City = @City OR Addresses.ZipCode = @ZipCode
		END;

4	Get customers by service name:	CREATE PROCEDURE GetCustomersByServiceName
		@ServiceName VARCHAR(255)
		AS
		BEGIN
		SELECT Customers.FirstName, Customers.LastName, Servicess.ServiceName
		FROM Subscriptions
		INNER JOIN Customers ON Subscriptions.CustomerID = Customers.CustomerID
		INNER JOIN Servicess ON Subscriptions.ServiceID = Servicess.ServiceID
		WHERE Servicess.ServiceName = @ServiceName
		END;

5	Get subscriptions by start date and end date:	CREATE PROCEDURE GetSubscriptionsByStartDateAndEndDate
		@StartDate DATE,
		@EndDate DATE
		AS
		BEGIN
		SELECT Subscriptions.SubscriptionID, Subscriptions.StartDate, Subscriptions.EndDate, Customers.FirstName, Customers.LastName, Servicess.ServiceName
		FROM Subscriptions
		INNER JOIN Customers ON Subscriptions.CustomerID = Customers.CustomerID
		INNER JOIN Servicess ON Subscriptions.ServiceID = Servicess.ServiceID
		WHERE Subscriptions.StartDate >= @StartDate AND Subscriptions.EndDate <= @EndDate
		END;

6	Get technicians by first name or last name:	CREATE PROCEDURE GetTechniciansByFirstOrLastName
		@FirstName VARCHAR(255),
		@LastName VARCHAR(255)
		AS
		BEGIN
		SELECT Technicians.FirstName, Technicians.LastName
		FROM Technicians
		WHERE Technicians.FirstName = @FirstName OR Technicians.LastName = @LastName
		END;

7	Get issues by customer	CREATE PROCEDURE GetIssuesByCustomerIDAndIssueDescription @CustomerID INT, @IssueDescription VARCHAR(255)

	ID and issue description:	AS
		BEGIN
		    SELECT Issues.IssueID, Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved, Customers.FirstName, Customers.LastName
		    FROM Issues
		    INNER JOIN Customers ON Issues.CustomerID = Customers.CustomerID
		    WHERE Issues.CustomerID = @CustomerID AND Issues.IssueDescription LIKE '%' + @IssueDescription + '%'
		END;

8	Get equipment by equipment name:	CREATE PROCEDURE GetEquipmentByEquipmentName
		   @EquipmentName VARCHAR(255)
		AS
		BEGIN
		   SELECT Equipment.EquipmentName, Equipment.EquipmentDescription
		   FROM Equipment
		   WHERE Equipment.EquipmentName = @EquipmentName
		END;

9	Get service offices by city or zip code:	CREATE PROCEDURE GetServiceOfficesByCityOrZipCode
		   @City VARCHAR(255),
		   @ZipCode VARCHAR(255)
		AS
		BEGIN
		   SELECT ServiceOffices.OfficeID, ServiceOffices.City, ServiceOffices.ZipCode
		   FROM ServiceOffices
		   WHERE ServiceOffices.City = @City OR ServiceOffices.ZipCode = @ZipCode
		END;

10	Get promotions by promotion name or start date:	CREATE PROCEDURE GetPromotionsByPromotionNameOrStartDate
		   @PromotionName VARCHAR(255),
		   @StartDate DATE
		AS
		BEGIN
		   SELECT Promotions.PromotionID, Promotions.PromotionName, Promotions.PromotionDescription, Promotions.StartDate, Promotions.EndDate
		   FROM Promotions
		   WHERE Promotions.PromotionName = @PromotionName OR Promotions.StartDate >= @StartDate
		END;

11	Get contracts by customer ID or service ID:	CREATE PROCEDURE GetContractsByCustomerIDOrServiceID
		   @CustomerID INT,
		   @ServiceID INT
		AS
		BEGIN
		   SELECT Contracts.ContractID, Contracts.StartDate, Contracts.EndDate, Customers.FirstName, Customers.LastName, Servicess.ServiceName
		   FROM Contracts
		   INNER JOIN Customers ON Contracts.CustomerID = Customers.CustomerID
		   INNER JOIN Servicess ON Contracts.ServiceID = Servicess.ServiceID
		   WHERE Contracts.CustomerID = @CustomerID OR Contracts.ServiceID = @ServiceID
		END;

12	Get billing by subscription ID and due date:	CREATE PROCEDURE GetBillingBySubscriptionIDAndDueDate
		    @SubscriptionID INT,
		    @DueDate DATE
		AS
		BEGIN
		    SELECT Billing.BillingID, Billing.BillingDate, Billing.AmountDue, Billing.DueDate, Subscriptions.SubscriptionID
		    FROM Billing
		    INNER JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID
		    WHERE Billing.SubscriptionID = @SubscriptionID AND Billing.DueDate <= @DueDate
		END;

13	Get payments by billing ID and payment date:	CREATE PROCEDURE GetPaymentsByBillingIDAndPaymentDate
		    @BillingID INT,
		    @PaymentDate DATE
		AS
		BEGIN
		    SELECT Payments.PaymentID, Payments.PaymentDate, Payments.PaymentAmount, Billing.BillingID
		    FROM Payments
		    INNER JOIN Billing ON Payments.BillingID = Billing.BillingID
		    WHERE Payments.BillingID = @BillingID AND Payments.PaymentDate <= @PaymentDate
		END;

14	Get usage by subscription ID and usage date:	CREATE PROCEDURE GetUsageBySubscriptionIDAndUsageDate
		    @SubscriptionID INT,
		    @UsageDate DATE
		AS
		BEGIN
		    SELECT Usage.UsageID, Usage.UsageDate, Usage.DataUsed, Subscriptions.SubscriptionID
		    FROM Usage
		    INNER JOIN Subscriptions ON Usage.SubscriptionID = Subscriptions.SubscriptionID
		    WHERE Usage.SubscriptionID = @SubscriptionID AND Usage.UsageDate <= @UsageDate
		END;

15	Get issues by technician ID and date resolved:	CREATE PROCEDURE GetIssuesByTechnicianIDAndDateResolved
		    @TechnicianID INT,
		    @DateResolved DATE
		AS
		BEGIN
		    SELECT Issues.IssueID, Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved, Technicians.FirstName, Technicians.LastName
		    FROM Issues
		    INNER JOIN Technicians ON Issues.TechnicianID = Technicians.TechnicianID
		    WHERE Issues.TechnicianID = @TechnicianID AND (Issues.DateResolved <= @DateResolved OR Issues.DateResolved IS NULL)
		END;

16	Get customers by address ID or email:	CREATE PROCEDURE GetCustomersByAddressIDOrEmail
		   @AddressID INT,
		   @Email VARCHAR(255)
		AS

		BEGIN
		SELECT Customers.CustomerID, Customers.FirstName, Customers.LastName, Customers.Email, Addresses.StreetAddress
		FROM Customers
		INNER JOIN Addresses ON Customers.AddressID = Addresses.AddressID
		WHERE Customers.AddressID = @AddressID OR Customers.Email = @Email
		END;

17	Get customers by phone number or email:	CREATE PROCEDURE GetCustomersByPhoneNumberOrEmail
		@PhoneNumber VARCHAR(255),
		@Email VARCHAR(255)
		AS
		BEGIN
		SELECT Customers.FirstName, Customers.LastName, Customers.PhoneNumber, Customers.Email
		FROM Customers
		WHERE Customers.PhoneNumber = @PhoneNumber OR Customers.Email = @Email
		END;

18	Get subscriptions by service ID and start date:	CREATE PROCEDURE GetSubscriptionsByServiceIDAndStartDate
		@ServiceID INT,
		@StartDate DATE
		AS
		BEGIN
		SELECT Subscriptions.SubscriptionID, Subscriptions.StartDate, Subscriptions.EndDate, Servicess.ServiceName
		FROM Subscriptions
		INNER JOIN Servicess ON Subscriptions.ServiceID = Servicess.ServiceID
		WHERE Subscriptions.ServiceID = @ServiceID AND Subscriptions.StartDate >= @StartDate
		END;

19	Get customers by first name and last name, ordered by first name	CREATE PROCEDURE GetCustomersByFirstAndLastNameOrderByFirstName
		@FirstName VARCHAR(255),
		@LastName VARCHAR(255)
		AS
		BEGIN
		SELECT Customers.FirstName, Customers.LastName
		FROM Customers
		WHERE Customers.FirstName = @FirstName AND Customers.LastName = @LastName
		ORDER BY Customers.FirstName
		END;

20	Get customers by city or zip code, ordered by last name:	CREATE PROCEDURE GetCustomersByCityOrZipCodeOrderByLastName
		@City VARCHAR(255),
		@ZipCode VARCHAR(255)
		AS
		BEGIN
		SELECT Customers.FirstName, Customers.LastName, Addresses.City, Addresses.ZipCode
		FROM Customers
		INNER JOIN Addresses ON Customers.AddressID = Addresses.AddressID
		WHERE Addresses.City = @City OR Addresses.ZipCode = @ZipCode
		ORDER BY Customers.LastName
		END;

21	Get customers by service name, ordered by last name:	CREATE PROCEDURE GetCustomersByServiceNameOrderByLastName
		    @ServiceName VARCHAR(255)
		AS
		BEGIN
		    SELECT Customers.FirstName, Customers.LastName, Servicess.ServiceName
		    FROM Subscriptions
		    INNER JOIN Customers ON Subscriptions.CustomerID = Customers.CustomerID
		    INNER JOIN Servicess ON Subscriptions.ServiceID = Servicess.ServiceID
		    WHERE Servicess.ServiceName = @ServiceName
		    ORDER BY Customers.LastName
		END;

22	Get subscriptions by start date and end date, ordered by start date:	CREATE PROCEDURE GetSubscriptionsByStartDateAndEndDateOrderByStartDate
		    @StartDate DATE,
		    @EndDate DATE
		AS
		BEGIN
		    SELECT Subscriptions.SubscriptionID, Subscriptions.StartDate, Subscriptions.EndDate, Customers.FirstName, Customers.LastName, Servicess.ServiceName
		    FROM Subscriptions
		    INNER JOIN Customers ON Subscriptions.CustomerID = Customers.CustomerID
		    INNER JOIN Servicess ON Subscriptions.ServiceID = Servicess.ServiceID
		    WHERE Subscriptions.StartDate >= @StartDate AND Subscriptions.EndDate <= @EndDate
		    ORDER BY Subscriptions.StartDate
		END;

23	Get technicians by first name or last name, ordered by first name:	CREATE PROCEDURE GetTechniciansByFirstOrLastNameOrderByFirstName
		   @FirstName VARCHAR(255),
		   @LastName VARCHAR(255)
		AS
		BEGIN
		   SELECT Technicians.FirstName, Technicians.LastName
		   FROM Technicians
		   WHERE Technicians.FirstName = @FirstName OR Technicians.LastName = @LastName
		   ORDER BY Technicians.FirstName
		END;

24	Get issues by customer ID and issue description, ordered by date created:	CREATE PROCEDURE GetIssuesByCustomerIDAndIssueDescriptionOrderByDateCreated
		   @CustomerID INT,
		   @IssueDescription VARCHAR(255)
		AS
		BEGIN
		   SELECT Issues.IssueID, Issues.IssueDescription, Issues.DateCreated, Issues.DateResolved, Customers.FirstName, Customers.LastName
		   FROM Issues
		   INNER JOIN Customers ON Issues.CustomerID = Customers.CustomerID
		   WHERE Issues.CustomerID = @CustomerID AND Issues.IssueDescription LIKE '%' + @IssueDescription + '%'
		   ORDER BY Issues.DateCreated
		END;

25	Get equipment by equipment name or equipment description, ordered by equipment name	CREATE PROCEDURE GetEquipmentByEquipmentNameOrEquipmentDescriptionOrderByEquipmentName
		@EquipmentName VARCHAR(255),
		@EquipmentDescription VARCHAR(255)
		AS
		BEGIN
		SELECT Equipment.EquipmentName, Equipment.EquipmentDescription
		FROM Equipment
		WHERE Equipment.EquipmentName = @EquipmentName OR Equipment.EquipmentDescription LIKE '%' + @EquipmentDescription + '%'
		ORDER BY Equipment.EquipmentName
		END;

26	Get service offices by city or office ID, ordered by city:	CREATE PROCEDURE GetServiceOfficesByCityOrOfficeIDOrderByCity
		@City VARCHAR(255),
		@OfficeID INT
		AS
		BEGIN
		SELECT ServiceOffices.OfficeID, ServiceOffices.City, ServiceOffices.ZipCode
		FROM ServiceOffices
		WHERE ServiceOffices.City = @City OR ServiceOffices.OfficeID = @OfficeID
		ORDER BY ServiceOffices.City
		END;

27	Get promotions by promotion name or start date, ordered by start date:	--Get promotions by promotion name or start date, ordered by start date:
		CREATE PROCEDURE GetPromotionsByPromotionNameOrStartDateOrderByStartDate
		@PromotionName VARCHAR(255),
		@StartDate DATE
		AS
		BEGIN
		SELECT Promotions.PromotionID, Promotions.PromotionName, Promotions.PromotionDescription, Promotions.StartDate, Promotions.EndDate
		FROM Promotions
		WHERE Promotions.PromotionName = @PromotionName OR Promotions.StartDate >= @StartDate
		ORDER BY Promotions.StartDate
		END;

28	Get contracts by customer ID or service ID, ordered by start date:	CREATE PROCEDURE GetContractsByCustomerIDOrServiceIDOrderByStartDate @CustomerID INT, @ServiceID INT AS BEGIN SELECT Contracts.ContractID, Contracts.StartDate, Contracts.EndDate, Customers.FirstName, Customers.LastName, Servicess.ServiceName FROM Contracts INNER JOIN Customers ON Contracts.CustomerID = Customers.CustomerID INNER JOIN Servicess ON Contracts.ServiceID = Servicess.ServiceID WHERE Contracts.CustomerID = @CustomerID OR Contracts.ServiceID = @ServiceID ORDER BY Contracts.StartDate

		END;

29	Get billing by subscription ID and due date, ordered by amount due:	CREATE PROCEDURE GetBillingBySubscriptionIDAndDueDateOrderByAmountDue
		@SubscriptionID INT,
		@DueDate DATE
		AS
		BEGIN
		SELECT Billing.BillingID, Billing.BillingDate, Billing.AmountDue, Billing.DueDate, Subscriptions.SubscriptionID
		FROM Billing
		INNER JOIN Subscriptions ON Billing.SubscriptionID = Subscriptions.SubscriptionID
		WHERE Billing.SubscriptionID = @SubscriptionID AND Billing.DueDate <= @DueDate
		ORDER BY Billing.AmountDue
		END;

30	Get payments by billing ID and payment date, ordered by payment amount:	CREATE PROCEDURE GetPaymentsByBillingIDAndPaymentDateOrderByPaymentAmount
		@BillingID INT,
		@PaymentDate DATE
		AS
		BEGIN
		SELECT Payments.PaymentID, Payments.PaymentDate, Payments.PaymentAmount, Billing.BillingID
		FROM Payments
		INNER JOIN Billing ON Payments.BillingID = Billing.BillingID
		WHERE Payments.BillingID = @BillingID AND Payments.PaymentDate <= @PaymentDate
		ORDER BY Payments.PaymentAmount
		END;

24. DML Triggers INSERT– 20 Queries

1	This is an AFTER INSERT trigger on the Customers table . It inserts the details of newly inserted customers into the AuditCustomers table for tracking changes.	CREATE TRIGGER trgAfterInsert
		ON Customers
		AFTER INSERT
		AS
		BEGIN
		
		DECLARE @FirstName VARCHAR(255);
		DECLARE @LastName VARCHAR(255);
		DECLARE @Email VARCHAR(255);
		DECLARE @PhoneNumber VARCHAR(255);
		DECLARE @CustomerID INT;
		SELECT @CustomerID = i.CustomerID FROM inserted i;
		SELECT @FirstName = i.FirstName FROM inserted i;
		SELECT @LastName = i.LastName FROM inserted i;
		SELECT @Email = i.Email FROM inserted i;
		SELECT @PhoneNumber = i.PhoneNumber FROM inserted i;
		INSERT INTO AuditCustomers (CustomerID ,
		FirstName ,
		LastName ,
		Email ,
		PhoneNumber)
		VALUES (@CustomerID, @FirstName,@LastName,@Email,@PhoneNumber);
		END;

2	This is an AFTER INSERT trigger on the Addresses table. It inserts the details of newly inserted addresses into the AuditAddresses table for tracking changes.	CREATE TRIGGER trgAfterInsertAddresses
		ON Addresses
		AFTER INSERT
		AS
		BEGIN
		DECLARE @AddressID INT;
		
		DECLARE @StreetAddress VARCHAR(255);
		DECLARE @City VARCHAR(255);
		DECLARE @District VARCHAR(255)
		DECLARE @ZipCode VARCHAR(255)
		SELECT @AddressID = i.AddressID,@StreetAddress=i.StreetAddress
		,@City=i.City,@District=i.District,@ZipCode=i.ZipCode FROM inserted i;
		INSERT INTO AuditAddresses(AddressID ,
		StreetAddress ,
		City ,
		District ,
		ZipCode)
		VALUES (@AddressID,@StreetAddress ,@City,@District,@ZipCode);
		END;

3	This is an AFTER INSERT trigger on the Servicess table. It inserts the details of newly inserted services into the AuditServicess table for tracking changes.	CREATE TRIGGER trgAfterInsertServicess
		ON Servicess
		AFTER INSERT
		AS
		BEGIN
		DECLARE @ServiceID INT;
		DECLARE @ServiceName VARCHAR(255);
		DECLARE @ServiceDescription VARCHAR(255);
		DECLARE @MonthlyFee float
		SELECT @ServiceID = i.ServiceID
		,@ServiceName=i.ServiceName,@ServiceDescription=i.ServiceDescription,
		@MonthlyFee=i.MonthlyFee from inserted i;
		INSERT INTO AuditServicess(ServiceID ,
		ServiceName ,
		ServiceDescription ,
		MonthlyFee)
		VALUES (@ServiceID, @ServiceName,@ServiceDescription,@MonthlyFee);
		END;

4	This is an AFTER INSERT trigger on the Subscriptions table. It inserts the details of newly inserted subscriptions into the AuditSubscriptions table for tracking changes.	CREATE TRIGGER trgAfterInsertSubscriptions
		ON Subscriptions
		AFTER INSERT
		AS
		BEGIN
		DECLARE @SubscriptionID INT;
		DECLARE @StartDate DATE;
		DECLARE @EndDate DATE
		SELECT @SubscriptionID = i.SubscriptionID ,
		@StartDate=i.StartDate,@EndDate=i.EndDate from inserted i;
		INSERT INTO AuditSubscriptions (SubscriptionID ,
		StartDate ,
		EndDate)
		VALUES (@SubscriptionID,@StartDate,@EndDate);
		END;

5	This is an AFTER INSERT trigger on	CREATE TRIGGER trgAfterInsertBilling ON Billing

	the Billing table. It inserts the details of newly inserted billing records into the AuditBilling tab le for tracking changes.	AFTER INSERT
		AS
		BEGIN
		DECLARE @BillingID INT;
		DECLARE @BillingDate DATE;
		DECLARE @AmountDue float;
		DECLARE @DueDate DATE
		SELECT @BillingID = i.BillingID ,@BillingDate=i.BillingDate
		,@AmountDue=i.AmountDue,@DueDate=i.DueDate FROM inserted i;
		INSERT INTO AuditBilling (BillingID ,
		BillingDate ,
		AmountDue ,
		DueDate)
		VALUES (@BillingID, @BillingDate,@AmountDue,@DueDate);
		END;

6	This is an AFTER INSERT trigger on the Payments table. It inserts the details of newly inserted payment records into
	the AuditPayments table for tracking changes.
		CREATE TRIGGER trgAfterInsertPayments
		ON Payments
		AFTER INSERT
		AS
		BEGIN
		DECLARE @PaymentID INT;
		DECLARE @PaymentDate DATE;
		DECLARE @PaymentAmount float
		SELECT @PaymentID = i.PaymentID,@PaymentDate=i.PaymentDate ,
		@PaymentAmount=i.PaymentAmount from inserted i;
		INSERT INTO AuditPayments (PaymentID ,
		PaymentDate ,
		PaymentAmount)
		VALUES (@PaymentID,@PaymentDate,@PaymentAmount);
		END;

7	This is an AFTER INSERT trigger on the Usage table. It inserts the details of newly inserted usage records into the AuditUsage tabl e for tracking changes.	CREATE TRIGGER trgAfterInsertUsage
		ON Usage
		AFTER INSERT
		AS
		BEGIN
		DECLARE @UsageID INT;
		DECLARE @UsageDate DATE;
		DECLARE @DataUsed float;
		DECLARE @UsageBytes varchar(255)
		SELECT @UsageID = i.UsageID,@UsageDate=i.UsageDate,
		@DataUsed=i.DataUsed,@UsageBytes=i.UsageBytes FROM inserted i;
		INSERT INTO AuditUsage (UsageID ,
		UsageDate ,
		DataUsed ,
		UsageBytes)
		VALUES (@UsageID, @UsageDate,@DataUsed,@UsageBytes);
		END;

8	This is an AFTER INSERT trigger on the Technicians tabl e. It inserts the details of newly inserted technician records into	CREATE TRIGGER trgAfterInsertTechnicians ON Technicians AFTER INSERT AS BEGIN DECLARE @TechnicianID INT,@FirstName VARCHAR(255), @LastName VARCHAR(255), @PhoneNumber VARCHAR(255);

	the AuditTechnicians table for tracking changes.	SELECT @TechnicianID = i.TechnicianID ,@FirstName=i.FirstName ,@LastName=i.LastName,@PhoneNumber=i.PhoneNumber from inserted i; INSERT INTO AuditTechnicians VALUES (@TechnicianID, @FirstName,@LastName,@PhoneNumber); END;

9	This is an AFTER INSERT trigger on the Issues table. It inserts the details of newly inserted issue records into the AuditIssues table for tracking changes.	CREATE TRIGGER trgAfterInsertIssues
		ON Issues
		AFTER INSERT
		AS
		BEGIN
		DECLARE @IssueID INT ,@IssueDescription VARCHAR(255),
		@DateCreated DATE,
		@DateResolved DATE;
		SELECT @IssueID = i.IssueID,@IssueDescription=i.IssueDescription, @DateCreated=i.DateCreated,@DateResolved=i.DateResolved FROM inserted i;
		INSERT INTO AuditIssues
		VALUES (@IssueID, @IssueDescription,@DateCreated,@DateResolved);
		END;

10	This is an AFTER INSERT trigger on the ServiceOffices table. It inserts the details of newly inserted service office records into the AuditServiceOffices table for tracking changes.	CREATE TRIGGER trgAfterInsertServiceOffices
		ON ServiceOffices
		AFTER INSERT
		AS
		BEGIN
		DECLARE @OfficeID INT,
		@City VARCHAR(255),
		@ZipCode VARCHAR(255);
		SELECT @OfficeID = i.OfficeID,@City=i.City,@ZipCode=i.ZipCode FROM inserted i;
		INSERT INTO AuditServiceOffices
		VALUES (@OfficeID, @City,@ZipCode);
		END;

11	This is an AFTER INSERT trigger on the Promotions table. It inserts the details of newly inserted promotion records into the AuditPromotions table for tracking changes.	CREATE TRIGGER trgAfterInsertPromotions
		ON Promotions
		AFTER INSERT
		AS
		BEGIN
		DECLARE @PromotionID INT, @PromotionName VARCHAR(255),
		@PromotionDescription VARCHAR(255),
		@StartDate DATE,
		@EndDate DATE;SELECT @PromotionID = i.PromotionID,@PromotionName= i.PromotionName,@PromotionDescription= i.PromotionDescription, @StartDate= i.StartDate,@EndDate= i.EndDate FROM inserted i;
		INSERT INTO AuditPromotions
		VALUES (@PromotionID, @PromotionName,@PromotionDescription,@StartDate,@EndDate);
		END;

12	This is an AFTER INSERT trigger on the Contracts table. It inserts the details of newly inserted contract records into the AuditContracts	CREATE TRIGGER trgAfterInsertContracts
		ON Contracts
		AFTER INSERT
		AS
		BEGIN
		DECLARE @ContractID INT,@EndDate DATE;
		SELECT @ContractID = i.ContractID,@EndDate=i.EndDate FROM inserted i;
		INSERT INTO AuditContracts
		VALUES (@ContractID, @EndDate);
		END;

	table for tracking changes.	

13	This is an AFTER INSERT trigger on the Customers table . It inserts a record into the AuditCustomers table with the CustomerID of the newly inserted customer and an auditaction value of 'Inserted new customer' for tracking changes.	`CREATE TRIGGER trgAfterInsertCustomersAudit`
		`ON Customers`
		`AFTER INSERT`
		`AS`
		`BEGIN`
		`  DECLARE @CustomerID INT;`
		`  SELECT @CustomerID = i.CustomerID FROM inserted i;`
		`  INSERT INTO AuditCustomers (CustomerID, auditaction)`
		`  VALUES (@CustomerID, 'Inserted new customer');`
		`END;`

14	This is an AFTER INSERT trigger on the Addresses table. It inserts a record into the AuditAddresses table with the AddressID of the newly inserted address and an AuditAction val ue of 'Inserted new address' for tracking changes.	`CREATE TRIGGER trgAfterInsertAddressesAudit`
		`ON Addresses`
		`AFTER INSERT`
		`AS`
		`BEGIN`
		`  DECLARE @AddressID INT;`
		`  SELECT @AddressID = i.AddressID FROM inserted i;`
		`  INSERT INTO AuditAddresses (AddressID, AuditAction)`
		`  VALUES (@AddressID, 'Inserted new address');`
		`END;`

15	This is an AFTER INSERT trigger on the Servicess table. It inserts a record into the AuditServicess t able with the ServiceID of the newly inserted service and an AuditAction val ue of 'Inserted new service' for tracking changes.	`CREATE TRIGGER trgAfterInsertServicessAudit`
		`ON Servicess`
		`AFTER INSERT`
		`AS`
		`BEGIN`
		`  DECLARE @ServiceID INT;`
		`  SELECT @ServiceID = i.ServiceID FROM inserted i;`
		`  INSERT INTO AuditServicess (ServiceID, AuditAction)`
		`  VALUES (@ServiceID, 'Inserted new service');`
		`END;`

16	This is an AFTER INSERT trigger	`CREATE TRIGGER trgAfterInsertSubscriptionsAudit`
		`ON Subscriptions`
		`AFTER INSERT`

	named trgAfterInsertSubscriptionsAudit on the Subscriptions table. When a new row is inserted into the Subscriptions table, this trigger will insert a new row into the AuditSubscriptions table with the SubscriptionID of the newly inserted row and an AuditAction value of 'Inserted new subscription'.	AS
		BEGIN
		DECLARE @SubscriptionID INT;
		SELECT @SubscriptionID = i.SubscriptionID FROM inserted i;
		INSERT INTO AuditSubscriptions (SubscriptionID, AuditAction)
		VALUES (@SubscriptionID, 'Inserted new subscription');
		END;

17	This is an AFTER INSERT trigger named trgAfterInsertBillingAudit on the Billing table. When a new row is inserted into the Billing table, this trigger will insert a new row into the AuditBilling table with the BillingID of the newly inserted row and an AuditAction value of 'Inserted new billing'.	CREATE TRIGGER trgAfterInsertBillingAudit
		ON Billing
		AFTER INSERT
		AS
		BEGIN
		DECLARE @BillingID INT;
		SELECT @BillingID = i.BillingID FROM inserted i;
		INSERT INTO AuditBilling(BillingID, AuditAction)
		VALUES (@BillingID, 'Inserted new billing');
		END;

18	This is an AFTER INSERT trigger named trgAfterInsertPaymentsAudit on the Payments table. When a new row is inserted into the Payments table, this trigger will	CREATE TRIGGER trgAfterInsertPaymentsAudit
		ON Payments
		AFTER INSERT
		AS
		BEGIN
		DECLARE @PaymentID INT;
		SELECT @PaymentID = i.PaymentID FROM inserted i;
		INSERT INTO AuditPayments(PaymentID, AuditAction)
		VALUES (@PaymentID, 'Inserted new payment');
		END;

	insert a new row into the AuditPayments table with the PaymentID of the newly inserted row and an AuditAction val ue of 'Inserted new payment'.	

19	This is an AFTER INSERT trigger named trgAfterInse rtUsageAudit on the Usage table. When a new row is inserted into the Usage table, this trigger will insert a new row into the AuditUsage tabl e with the UsageID of the newly inserted row and an AuditAction val ue of 'Inserted new usage'.	`CREATE TRIGGER trgAfterInsertUsageAudit`
		`ON Usage`
		`AFTER INSERT`
		`AS`
		`BEGIN`
		`  DECLARE @UsageID INT;`
		`  SELECT @UsageID = i.UsageID FROM inserted i;`
		`  INSERT into AuditUsage(UsageID, AuditAction)`
		`  VALUES (@UsageID, 'Inserted new usage');`
		`END;`

20	This is an `AFTER INSERT` trigger named `trgrInsertServiceO ffices` on the `ServiceOffices` table. When a new row is inserted into the `ServiceOffices` table, this trigger will insert a new row into the `AuditServiceOffic es` table with the `OfficeID`, `City`, and `ZipCode` of the newly inserted	`CREATE TRIGGER trgrInsertServiceOffices`
		`ON ServiceOffices`
		`AFTER INSERT`
		`AS`
		`BEGIN`
		`  DECLARE @OfficeID INT,`
		`  @City VARCHAR(255),`
		`  @ZipCode VARCHAR(255);`
		`  SELECT @OfficeID = i.OfficeID,@City=i.City,@ZipCode=i.ZipCode FROM inserted i;`
		`  INSERT INTO AuditServiceOffices`
		`  VALUES (@OfficeID, @City,@ZipCode,'New office added');`
		`END;`

	row and an `AuditAction` value of `'New office added'`.	

25. DML Triggers UPDATE– 20 Queries

1	trgAfterupdate is an AFTER UPDATE trigger on the Customers table. When a row is updated in the Customers table, this trigger will insert a new row into the AuditCustomers t able with the updated values and an AuditAction value of 'Customer updated'.	CREATE TRIGGER trgAfterupdate
		ON Customers
		AFTER update
		AS
		BEGIN
		
		DECLARE @FirstName VARCHAR(255);
		DECLARE @LastName VARCHAR(255);
		DECLARE @Email VARCHAR(255);
		DECLARE @PhoneNumber VARCHAR(255);
		DECLARE @CustomerID INT;
		SELECT @CustomerID = i.CustomerID FROM inserted i;
		SELECT @FirstName = i.FirstName FROM inserted i;
		SELECT @LastName = i.LastName FROM inserted i;
		SELECT @Email = i.Email FROM inserted i;
		SELECT @PhoneNumber = i.PhoneNumber FROM inserted i;
		
		INSERT INTO AuditCustomers (CustomerID ,
		FirstName ,
		LastName ,
		Email ,
		PhoneNumber,AuditAction)
		VALUES (@CustomerID,
		@FirstName,@LastName,@Email,@PhoneNumber,'Customer updated');
		END;

2	trgAfterupdateAddresses is an AFTER UPDATE trigger on the Addresses table. When a row is updated in the Addresses table, this trigger will insert a new row into the AuditAddresses t able with the updated values and an AuditAction value of 'Address updated'.	CREATE TRIGGER trgAfterupdateAddresses
		ON Addresses
		AFTER update
		AS
		BEGIN
		DECLARE @AddressID INT;
		
		DECLARE @StreetAddress VARCHAR(255);
		DECLARE @City VARCHAR(255);
		DECLARE @District VARCHAR(255)
		DECLARE @ZipCode VARCHAR(255)
		SELECT @AddressID =
		i.AddressID,@StreetAddress=i.StreetAddress
		,@City=i.City,@District=i.District,@ZipCode=i.ZipCode FROM
		inserted i;
		INSERT INTO AuditAddresses(AddressID ,
		StreetAddress ,
		City ,
		District ,
		ZipCode ,AuditAction)
		VALUES (@AddressID,@StreetAddress
		,@City,@District,@ZipCode,'Address updated');
		END;

3	trgAfterupdateServicess is an AFTER UPDATE trigger on the Servicess table. When a row is updated in the Servicess table, this trigger will insert a new row into the AuditServicess table with the updated values and an AuditAction value of 'Service Updated'.	CREATE TRIGGER trgAfterupdateServicess
		ON Servicess
		AFTER update
		AS
		BEGIN
		DECLARE @ServiceID INT;
		DECLARE @ServiceName VARCHAR(255);
		DECLARE @ServiceDescription VARCHAR(255);
		DECLARE @MonthlyFee float
		SELECT @ServiceID = i.ServiceID
		,@ServiceName=i.ServiceName,@ServiceDescription=i.ServiceDescription,@MonthlyFee=i.MonthlyFee from inserted i;
		INSERT INTO AuditServicess(ServiceID ,
		ServiceName ,
		ServiceDescription ,
		MonthlyFee, AuditAction)
		VALUES (@ServiceID,
		@ServiceName,@ServiceDescription,@MonthlyFee,'Service Updated');
		END;

4	trgAfterupdateSubscriptions is an AFTER UPDATE trigger on the Subscriptions table. When a row is updated in the Subscriptions table, this trigger will insert a new row into the AuditSubscriptions table with the updated values.	CREATE TRIGGER trgAfterupdateSubscriptions
		ON Subscriptions
		AFTER update
		AS
		BEGIN
		DECLARE @SubscriptionID INT;
		DECLARE @StartDate DATE;
		DECLARE @EndDate DATE
		SELECT @SubscriptionID = i.SubscriptionID
		,@StartDate=i.StartDate,@EndDate=i.EndDate from inserted i;
		INSERT INTO AuditSubscriptions (SubscriptionID ,
		StartDate ,
		EndDate ,AuditAction)
		VALUES (@SubscriptionID,@StartDate,@EndDate);
		END;

5	trgAfterupdateBilling is an AFTER UPDATE trigger on the Billing table. When a row is updated in the Billing table, this trigger will insert a new row into the AuditBilling table with the updated values and an AuditAction value of 'Billing Updated'.	CREATE TRIGGER trgAfterupdateBilling
		ON Billing
		AFTER update
		AS
		BEGIN
		DECLARE @BillingID INT;
		DECLARE @BillingDate DATE;
		DECLARE @AmountDue float;
		DECLARE @DueDate DATE
		SELECT @BillingID = i.BillingID
		,@BillingDate=i.BillingDate
		,@AmountDue=i.AmountDue,@DueDate=i.DueDate FROM inserted i;
		INSERT INTO AuditBilling (BillingID ,
		BillingDate ,
		AmountDue ,
		DueDate,AuditAction)
		VALUES (@BillingID,
		@BillingDate,@AmountDue,@DueDate,'Billing Updated');
		END;

6	trgAfterupdatePayments is an AFTER UPDATE trigger on the Payments table. When a row is updated in the Payments table, this trigger will insert a new row into the AuditPayments table with the updated values and an AuditAction value of 'Payment updated'.	CREATE TRIGGER trgAfterupdatePayments
		ON Payments
		AFTER update
		AS
		BEGIN
		DECLARE @PaymentID INT;
		DECLARE @PaymentDate DATE;
		DECLARE @PaymentAmount float
		SELECT @PaymentID = i.PaymentID,@PaymentDate=i.PaymentDate ,@PaymentAmount=i.PaymentAmount from inserted i;
		INSERT INTO AuditPayments (PaymentID ,
		PaymentDate ,
		PaymentAmount,AuditAction)
		VALUES (@PaymentID,@PaymentDate,@PaymentAmount,'Payment updated');
		END;

7	trgAfterupdateUsage is an AFTER UPDATE trigger on the Usage table. When a row is updated in the Usage table, this trigger will insert a new row into the AuditUsage table with the updated values and an AuditAction value of 'Usage Updated'.	CREATE TRIGGER trgAfterupdateUsage
		ON Usage
		AFTER update
		AS
		BEGIN
		DECLARE @UsageID INT;
		DECLARE @UsageDate DATE;
		DECLARE @DataUsed float;
		DECLARE @UsageBytes varchar(255)
		SELECT @UsageID = i.UsageID,@UsageDate=i.UsageDate,@DataUsed=i.DataUsed,@UsageBytes=i.UsageBytes FROM inserted i;
		INSERT INTO AuditUsage (UsageID ,
		UsageDate ,
		DataUsed ,
		UsageBytes,AuditAction)
		VALUES (@UsageID, @UsageDate,@DataUsed,@UsageBytes,'Usage Updated');
		END;

8	trgAfterupdateTechnicians is an AFTER UPDATE trigger on the Technicians table. When a row is updated in the Technicians table, this trigger will insert a new row into the AuditTechnicians table with the updated values and an AuditAction value of 'technician updated'.	CREATE TRIGGER trgAfterupdateTechnicians
		ON Technicians
		AFTER update
		AS
		BEGIN
		DECLARE @TechnicianID INT,@FirstName VARCHAR(255),
		@LastName VARCHAR(255),
		@PhoneNumber VARCHAR(255);
		SELECT @TechnicianID = i.TechnicianID ,@FirstName=i.FirstName ,@LastName=i.LastName,@PhoneNumber=i.PhoneNumber from inserted i;
		INSERT INTO AuditTechnicians
		VALUES (@TechnicianID, @FirstName,@LastName,@PhoneNumber,'technician updated');
		END;

9	trgAfterupdateIssues is an AFTER	CREATE TRIGGER trgAfterupdateIssues ON Issues AFTER update

	UPDATE trigger on the Issues table. When a row is updated in the Issues table, this trigger will insert a new row into the AuditIssues table with the updated values and an AuditAction value of 'issue updated'.	AS
		BEGIN
		DECLARE @IssueID INT ,@IssueDescription VARCHAR(255),
		@DateCreated DATE,
		@DateResolved DATE;
		SELECT @IssueID =
		i.IssueID,@IssueDescription=i.IssueDescription,@DateCreated=i
		.DateCreated,@DateResolved=i.DateResolved FROM inserted i;
		INSERT INTO AuditIssues
		VALUES (@IssueID,
		@IssueDescription,@DateCreated,@DateResolved,'issue
		updated');
		END;

10	trgAfterupdateServiceOffices is an AFTER UPDATE trigger on the ServiceOffices table. When a row is updated in the ServiceOffices table, this trigger will insert a new row into the AuditServiceOffices table with the updated values and an AuditAction value of 'issue updated'.	CREATE TRIGGER trgAfterupdateServiceOffices
		ON ServiceOffices
		AFTER update
		AS
		BEGIN
		DECLARE @OfficeID INT,
		@City VARCHAR(255),
		@ZipCode VARCHAR(255);
		SELECT @OfficeID =
		i.OfficeID,@City=i.City,@ZipCode=i.ZipCode FROM inserted i;
		INSERT INTO AuditServiceOffices
		VALUES (@OfficeID, @City,@ZipCode,'issue updated');
		END;

11	trgAfterupdatePromotions is an AFTER UPDATE trigger on the Promotions table. When a row is updated in the Promotions table, this trigger will insert a new row into the AuditPromotions table with the updated values and an AuditAction value of 'Promotion updated'.	CREATE TRIGGER trgAfterupdatePromotions
		ON Promotions
		AFTER update
		AS
		BEGIN
		DECLARE @PromotionID INT, @PromotionName VARCHAR(255),
		@PromotionDescription VARCHAR(255),
		@StartDate DATE,
		@EndDate DATE;
		SELECT @PromotionID = i.PromotionID,@PromotionName=
		i.PromotionName,@PromotionDescription=
		i.PromotionDescription,@StartDate= i.StartDate,@EndDate=
		i.EndDate FROM inserted i;
		INSERT INTO AuditPromotions
		VALUES (@PromotionID,
		@PromotionName,@PromotionDescription,@StartDate,@EndDate,'Pro
		motion updated');
		END;

12	trgAfterupdateContracts is an AFTER UPDATE trigger on the Contracts table.	CREATE TRIGGER trgAfterupdateContracts ON Contracts AFTER update AS BEGIN

	When a row is updated in the Contracts table, this trigger will insert a new row into the AuditContracts table with the updated values and an AuditAction value of 'contract updated'	`    DECLARE @ContractID INT,@EndDate DATE;`
		`    SELECT @ContractID = i.ContractID,@EndDate=i.EndDate FROM inserted i;`
		`    INSERT INTO AuditContracts`
		`    VALUES (@ContractID, @EndDate,'contract updated');`
		`END;`

13	trgAfterupdateCustomersAudit is an AFTER UPDATE trigger on the Customers table. When a row is updated in the Customers table, this trigger will insert a new row into the AuditCustomers table with the CustomerID of the updated row and an AuditAction value of 'updated new customer'.	`CREATE TRIGGER trgAfterupdateCustomersAudit`
		`ON Customers`
		`AFTER update`
		`AS`
		`BEGIN`
		`    DECLARE @CustomerID INT;`
		`    SELECT @CustomerID = i.CustomerID FROM inserted i;`
		`    INSERT INTO AuditCustomers (CustomerID, auditaction)`
		`    VALUES (@CustomerID, 'updated new customer');`
		`END;`

14	trgAfterupdateAddressesAudit is an AFTER UPDATE trigger on the Addresses table. When a row is updated in the Addresses table, this trigger will insert a new row into the AuditAddresses table with the AddressID of the updated row and an AuditAction value of 'updated new address'.	`CREATE TRIGGER trgAfterupdateAddressesAudit`
		`ON Addresses`
		`AFTER update`
		`AS`
		`BEGIN`
		`    DECLARE @AddressID INT;`
		`    SELECT @AddressID = i.AddressID FROM inserted i;`
		`    INSERT INTO AuditAddresses (AddressID, AuditAction)`
		`    VALUES (@AddressID, 'updated new address');`
		`END;`

15	trgAfterupdateServicessAudit is	`CREATE TRIGGER trgAfterupdateServicessAudit`
		`ON Servicess`
		`AFTER update`

	an AFTER UPDATE trigger on the Servicess table. When a row is updated in the Servicess table, this trigger will insert a new row into the AuditServicess table with the ServiceID of the updated row and an AuditAction value of 'updated new service'.	AS
		BEGIN
		DECLARE @ServiceID INT;
		SELECT @ServiceID = i.ServiceID FROM inserted i;
		INSERT INTO AuditServicess (ServiceID, AuditAction)
		VALUES (@ServiceID, 'updated new service');
		END;

16	trgAfterupdateSubscriptionsAudit is an AFTER UPDATE trigger on the Subscriptions table. When a row is updated in the Subscriptions table, this trigger will insert a new row into the AuditSubscriptions table with the SubscriptionID of the updated row and an AuditAction value of 'updated new subscription'.	CREATE TRIGGER trgAfterupdateSubscriptionsAudit
		ON Subscriptions
		AFTER update
		AS
		BEGIN
		DECLARE @SubscriptionID INT;
		SELECT @SubscriptionID = i.SubscriptionID FROM inserted i;
		INSERT INTO AuditSubscriptions (SubscriptionID, AuditAction)
		VALUES (@SubscriptionID, 'updated new subscription');
		END;

17	trgAfterupdateBillingAudit, is an AFTER UPDATE trigger on the Billing table. When a row is updated in the Billing, this trigger will insert a new row into the AuditBilling, with the BillingID of the updated billing and an AuditAction	CREATE TRIGGER trgAfterupdateBillingAudit
		ON Billing
		AFTER update
		AS
		BEGIN
		DECLARE @BillingID INT;
		SELECT @BillingID = i.BillingID FROM inserted i;
		INSERT INTO AuditBilling(BillingID, AuditAction)
		VALUES (@BillingID, 'updated new billing');
		END;

	value of ‘updated new billing’.	

18	trgAfterupdatePaymentsAudit, is an AFTER UPDATE trigger on Payments. When a row is updated in Payments, this trigger will insert a new row into AuditPayments with PaymentID of updated payment and AuditAction value of ‘updated new payment’.	`CREATE TRIGGER trgAfterupdatePaymentsAudit`
		`ON Payments`
		`AFTER update`
		`AS`
		`BEGIN`
		`DECLARE @PaymentID INT;`
		`SELECT @PaymentID = i.PaymentID FROM inserted i;`
		`INSERT INTO AuditPayments(PaymentID, AuditAction)`
		`VALUES (@PaymentID, 'updated new payment');`
		`END;`

19	trgAfterupdateUsageAudit is an AFTER UPDATE trigger on the Usage table. When a row is updated in the Usage table, this trigger will insert a new row into the AuditUsage table with the updated values and an AuditAction value of 'update new usage'.	`CREATE TRIGGER trgAfterupdateUsageAudit`
		`ON Usage`
		`AFTER update`
		`AS`
		`BEGIN`
		`DECLARE @UsageID INT;`
		`SELECT @UsageID = i.UsageID FROM inserted i;`
		`INSERT into AuditUsage(UsageID, AuditAction)`
		`VALUES (@UsageID, 'update new usage');`
		`END;`

20	trgrupdateServiceOffices is an AFTER UPDATE trigger on the ServiceOffices table. When a row is updated in the ServiceOffices table, this trigger will insert a new row into the AuditServiceOffices table with the updated values and an AuditAction value of 'updated office status'	`CREATE TRIGGER trgrupdateServiceOffices`
		`ON ServiceOffices`
		`AFTER update`
		`AS`
		`BEGIN`
		`DECLARE @OfficeID INT,`
		`@City VARCHAR(255),`
		`@ZipCode VARCHAR(255);`
		`SELECT @OfficeID = i.OfficeID,@City=i.City,@ZipCode=i.ZipCode FROM inserted i;`
		`INSERT INTO AuditServiceOffices`
		`VALUES (@OfficeID, @City,@ZipCode,'updated office status');`
		`END;`

26. DML Triggers Delete– 20 Queries

1	trgAfterdelete is an AFTER DELETE trigger on the Customers table. When a row is deleted from the Customers table, this trigger will insert a new row into the AuditCustomers t able with the values of the deleted row and an AuditAction value of 'Customer deleted'.	CREATE TRIGGER trgAfterdelete
		ON Customers
		AFTER delete
		AS
		BEGIN
		
		DECLARE @FirstName VARCHAR(255);
		DECLARE @LastName VARCHAR(255);
		DECLARE @Email VARCHAR(255);
		DECLARE @PhoneNumber VARCHAR(255);
		DECLARE @CustomerID INT;
		SELECT @CustomerID = i.CustomerID FROM inserted i;
		SELECT @FirstName = i.FirstName FROM inserted i;
		SELECT @LastName = i.LastName FROM inserted i;
		SELECT @Email = i.Email FROM inserted i;
		SELECT @PhoneNumber = i.PhoneNumber FROM inserted i;
		
		INSERT INTO AuditCustomers (CustomerID ,
		FirstName ,
		LastName ,
		Email ,
		PhoneNumber,AuditAction)
		VALUES (@CustomerID,
		@FirstName,@LastName,@Email,@PhoneNumber,'Customer deleted');
		END;

2	trgAfterdeleteAddres ses is an AFTER DELETE trigger on the Addresses table. When a row is deleted from the Addresses table, this trigger will insert a new row into the AuditAddresses t able with the values of the deleted row and an AuditAction value of 'Address deleted'.	CREATE TRIGGER trgAfterdeleteAddresses
		ON Addresses
		AFTER delete
		AS
		BEGIN
		DECLARE @AddressID INT;
		
		DECLARE @StreetAddress VARCHAR(255);
		DECLARE @City VARCHAR(255);
		DECLARE @District VARCHAR(255)
		DECLARE @ZipCode VARCHAR(255)
		SELECT @AddressID =
		i.AddressID,@StreetAddress=i.StreetAddress
		,@City=i.City,@District=i.District,@ZipCode=i.ZipCode FROM
		inserted i;
		INSERT INTO AuditAddresses(AddressID ,
		StreetAddress ,
		City ,
		District ,
		ZipCode ,AuditAction)
		VALUES (@AddressID,@StreetAddress
		,@City,@District,@ZipCode,'Address deleted');
		END;

3	trgAfterdeleteService ss is an AFTER DELETE trigger on the Servicess table.	CREATE TRIGGER trgAfterdeleteServicess ON Servicess AFTER delete AS BEGIN DECLARE @ServiceID INT;

	When a row is deleted from the Servicess table, this trigger will insert a new row into the AuditServicess table with the values of the deleted row and an AuditAction value of 'Service deleted'.	DECLARE @ServiceName VARCHAR(255); DECLARE @ServiceDescription VARCHAR(255); DECLARE @MonthlyFee float SELECT @ServiceID = i.ServiceID ,@ServiceName=i.ServiceName,@ServiceDescription=i.ServiceDescription,@MonthlyFee=i.MonthlyFee from inserted i; INSERT INTO AuditServicess(ServiceID , ServiceName , ServiceDescription , MonthlyFee, AuditAction) VALUES (@ServiceID, @ServiceName,@ServiceDescription,@MonthlyFee,'Service deleted'); END;

4	trgAfterdeleteSubscriptions is an AFTER DELETE trigger on the Subscriptions table. When a row is deleted from the Subscriptions table, this trigger will insert a new row into the AuditSubscriptions table with the values of the deleted row and an AuditAction value of 'Subscription deleted'.	CREATE TRIGGER trgAfterdeleteSubscriptions
		ON Subscriptions
		AFTER delete
		AS
		BEGIN
		DECLARE @SubscriptionID INT;
		DECLARE @StartDate DATE;
		DECLARE @EndDate DATE
		SELECT @SubscriptionID = i.SubscriptionID
		,@StartDate=i.StartDate,@EndDate=i.EndDate from inserted i;
		INSERT INTO AuditSubscriptions (SubscriptionID ,
		StartDate ,
		EndDate ,AuditAction)
		VALUES (@SubscriptionID,@StartDate,@EndDate,'Subscription deleted');
		END;

5	trgAfterdeleteBilling, is an AFTER DELETE trigger on Billing. When a row is deleted from Billing, this trigger will insert a new row into AuditBilling with values of deleted billing and AuditAction value of 'Billing deleted'.	CREATE TRIGGER trgAfterdeleteBilling
		ON Billing
		AFTER delete
		AS
		BEGIN
		DECLARE @BillingID INT;
		DECLARE @BillingDate DATE;
		DECLARE @AmountDue float;
		DECLARE @DueDate DATE
		SELECT @BillingID = i.BillingID
		,@BillingDate=i.BillingDate
		,@AmountDue=i.AmountDue,@DueDate=i.DueDate FROM inserted i;
		INSERT INTO AuditBilling (BillingID ,
		BillingDate ,
		AmountDue ,
		DueDate,AuditAction)
		VALUES (@BillingID,
		@BillingDate,@AmountDue,@DueDate,'Billing deleted');
		END;

6	trgAfterdeletePayments, is an AFTER DELETE trigger on	CREATE TRIGGER trgAfterdeletePayments ON Payments AFTER delete

	Payments. When a row is deleted from Payments, this trigger will insert a new row into AuditPayments with PaymentID of deleted payment and AuditAction value of 'Payment deleted'.	AS
		BEGIN
		DECLARE @PaymentID INT;
		DECLARE @PaymentDate DATE;
		DECLARE @PaymentAmount float
		SELECT @PaymentID = i.PaymentID,@PaymentDate=i.PaymentDate
		,@PaymentAmount=i.PaymentAmount from inserted i;
		INSERT INTO AuditPayments (PaymentID ,
		PaymentDate ,
		PaymentAmount,AuditAction)
		VALUES (@PaymentID,@PaymentDate,@PaymentAmount,'Payment
		deleted');
		END;

7	trgAfterdeleteeUsage , is an AFTER DELETE trigger on Usage. When a row is deleted from Usage, this trigger will insert a new row into AuditUsage with UsageID and AuditAction value of 'Usage Deleted'.	CREATE TRIGGER trgAfterdeleteeUsage
		ON Usage
		AFTER delete
		AS
		BEGIN
		DECLARE @UsageID INT;
		DECLARE @UsageDate DATE;
		DECLARE @DataUsed float;
		DECLARE @UsageBytes varchar(255)
		SELECT @UsageID =
		i.UsageID,@UsageDate=i.UsageDate,@DataUsed=i.DataUsed,@UsageBy
		tes=i.UsageBytes FROM inserted i;
		INSERT INTO AuditUsage (UsageID ,
		UsageDate ,
		DataUsed ,
		UsageBytes,AuditAction)
		VALUES (@UsageID, @UsageDate,@DataUsed,@UsageBytes,'Usage
		deleted');
		END;

8	trgAftedeleteTechnicians is an AFTER DELETE trigger on the Technicians table . When a row is deleted from the Technicians table , this trigger will insert a new row into the AuditTechnicians table with the values of the deleted row and an AuditAction value of 'technician deleted'.	CREATE TRIGGER trgAftedeleteTechnicians
		ON Technicians
		AFTER delete
		AS
		BEGIN
		DECLARE @TechnicianID INT,@FirstName VARCHAR(255),
		@LastName VARCHAR(255),
		@PhoneNumber VARCHAR(255);
		SELECT @TechnicianID = i.TechnicianID
		,@FirstName=i.FirstName
		,@LastName=i.LastName,@PhoneNumber=i.PhoneNumber from
		inserted i;
		INSERT INTO AuditTechnicians
		VALUES (@TechnicianID,
		@FirstName,@LastName,@PhoneNumber,'technician deleted');
		END;

9	trgAfterdeleteIssues is an AFTER DELETE trigger on the Issues table.	CREATE TRIGGER trgAfterdeleteIssues ON Issues AFTER delete AS BEGIN DECLARE @IssueID INT ,@IssueDescription VARCHAR(255),

	When a row is deleted from the Issues table, this trigger will insert a new row into the AuditIssues table with the values of the deleted row and an AuditAction value of 'issue deleted'.	@DateCreated DATE, @DateResolved DATE; SELECT @IssueID = i.IssueID,@IssueDescription=i.IssueDescription,@DateCreated=i.DateCreated,@DateResolved=i.DateResolved FROM inserted i; INSERT INTO AuditIssues VALUES (@IssueID, @IssueDescription,@DateCreated,@DateResolved,'issue deleted'); END;

10	trgAfterdeleteServiceOffices is an AFTER DELETE trigger on the ServiceOffices table. When a row is deleted from the ServiceOffices table, this trigger will insert a new row into the AuditServiceOffices table with the values of the deleted row and an AuditAction value of 'issue deleted'.	CREATE TRIGGER trgAfterdeleteServiceOffices
		ON ServiceOffices
		AFTER delete
		AS
		BEGIN
		DECLARE @OfficeID INT,
		@City VARCHAR(255),
		@ZipCode VARCHAR(255);
		SELECT @OfficeID = i.OfficeID,@City=i.City,@ZipCode=i.ZipCode FROM inserted i;
		INSERT INTO AuditServiceOffices
		VALUES (@OfficeID, @City,@ZipCode,'issue deleted');
		END;

11	trgAfterdeletePromotions, is an AFTER DELETE trigger on Promotions. When a row is deleted from Promotions, this trigger will insert a new row into AuditPromotions with values of deleted promotion and AuditAction value of 'Promotion deleted'.	CREATE TRIGGER trgAfterdeletePromotions
		ON Promotions
		AFTER delete
		AS
		BEGIN
		DECLARE @PromotionID INT, @PromotionName VARCHAR(255),
		@PromotionDescription VARCHAR(255),
		@StartDate DATE,
		@EndDate DATE;
		SELECT @PromotionID = i.PromotionID,@PromotionName= i.PromotionName,@PromotionDescription= i.PromotionDescription,@StartDate= i.StartDate,@EndDate= i.EndDate FROM inserted i;
		INSERT INTO AuditPromotions
		VALUES (@PromotionID, @PromotionName,@PromotionDescription,@StartDate,@EndDate,'Promotion deleted');
		END;

12	trgAfterdeleteContracts is an AFTER DELETE trigger on the Contracts table. When a row is deleted from	CREATE TRIGGER trgAfterdeleteContracts ON Contracts AFTER delete AS BEGIN DECLARE @ContractID INT,@EndDate DATE;

	the Contracts table, this trigger will insert a new row into the AuditContracts ta ble with the values of the deleted row and an AuditAction value of 'contract deleted'.	`    SELECT @ContractID = i.ContractID,@EndDate=i.EndDate FROM inserted i;`
		`    INSERT INTO AuditContracts`
		`    VALUES (@ContractID, @EndDate,'contract deleted');`
		`END;`

13	trgAfterdeleteCusto mersAudit is an AFTER DELETE trigger on the Customers table. When a row is deleted from the Customers table, this trigger will insert a new row into the AuditCustomers t able with the CustomerID of the deleted row and an AuditAction value of 'deleted customer'.	`CREATE TRIGGER trgAfterdeleteCustomersAudit`
		`ON Customers`
		`AFTER delete`
		`AS`
		`BEGIN`
		`    DECLARE @CustomerID INT;`
		`    SELECT @CustomerID = i.CustomerID FROM inserted i;`
		`    INSERT INTO AuditCustomers (CustomerID, auditaction)`
		`    VALUES (@CustomerID, 'deleted customer');`
		`END;`

14	trgAfterdeleteAddres sesAudit is an AFTER DELETE trigger on the Addresses table. When a row is deleted from the Addresses table, this trigger will insert a new row into the AuditAddresses t able with the AddressID of the deleted row and an AuditAction value of 'deleted address'.	`CREATE TRIGGER trgAfterdeleteAddressesAudit`
		`ON Addresses`
		`AFTER delete`
		`AS`
		`BEGIN`
		`    DECLARE @AddressID INT;`
		`    SELECT @AddressID = i.AddressID FROM inserted i;`
		`    INSERT INTO AuditAddresses (AddressID, AuditAction)`
		`    VALUES (@AddressID, 'deleted address');`
		`END;`

15	trgAfterdeleteService ssAudit is an AFTER DELETE trigger on the Servicess table. When a row is deleted from	`CREATE TRIGGER trgAfterdeleteServicessAudit`
		`ON Servicess`
		`AFTER delete`
		`AS`
		`BEGIN`
		`   DECLARE @ServiceID INT;`
		`   SELECT @ServiceID = i.ServiceID FROM inserted i;`
		`   INSERT INTO AuditServicess (ServiceID, AuditAction)`

	the Servicess table, this trigger will insert a new row into the AuditServicess table with the ServiceID of the deleted row and an AuditAction value of 'service deleted'.	`VALUES (@ServiceID, 'service deleted');` `END;`

16	trgAfterdeleteSubscriptionsAudit is an AFTER DELETE trigger on the Subscriptions table. When a row is deleted from the Subscriptions table, this trigger will insert a new row into the AuditSubscriptions table with the SubscriptionID of the deleted row and an AuditAction value of 'subscription delete'	`CREATE TRIGGER trgAfterdeleteSubscriptionsAudit`
		`ON Subscriptions`
		`AFTER delete`
		`AS`
		`BEGIN`
		`DECLARE @SubscriptionID INT;`
		`SELECT @SubscriptionID = i.SubscriptionID FROM inserted i;`
		`INSERT INTO AuditSubscriptions (SubscriptionID, AuditAction)`
		`VALUES (@SubscriptionID, 'subscription delete');`
		`END;`

17	trgAfterdeleteBillingAudit, is an AFTER DELETE trigger on Billing. When a row is deleted from Billing, this trigger will insert a new row into AuditBilling with BillingID of deleted billing and AuditAction value of 'billing deleted'.	`CREATE TRIGGER trgAfterdeleteBillingAudit`
		`ON Billing`
		`AFTER delete`
		`AS`
		`BEGIN`
		`DECLARE @BillingID INT;`
		`SELECT @BillingID = i.BillingID FROM inserted i;`
		`INSERT INTO AuditBilling(BillingID, AuditAction)`
		`VALUES (@BillingID, 'billing deleted');`
		`END;`

18	trgAfterdeletePaymentsAudit is an AFTER DELETE trigger on the Payments table. When a row is deleted from the Payments table,	`CREATE TRIGGER trgAfterdeletePaymentsAudit`
		`ON Payments`
		`AFTER delete`
		`AS`
		`BEGIN`
		`DECLARE @PaymentID INT;`
		`SELECT @PaymentID = i.PaymentID FROM inserted i;`
		`INSERT INTO AuditPayments(PaymentID, AuditAction)`
		`VALUES (@PaymentID, 'payment delete');`
		`END;`

	this trigger will insert a new row into the AuditPayments table with the PaymentID of the deleted row and an AuditAction value of 'payment delete'.	

19	trgAfterdeleteUsageAudit is an AFTER DELETE trigger on the Usage table. When a row is deleted from the Usage table, this trigger will insert a new row into the AuditUsage table with the UsageID of the deleted row and an AuditAction value of 'usage deleted'.	`CREATE TRIGGER trgAfterdeleteUsageAudit`
		`ON Usage`
		`AFTER delete`
		`AS`
		`BEGIN`
		`DECLARE @UsageID INT;`
		`SELECT @UsageID = i.UsageID FROM inserted i;`
		`INSERT into AuditUsage(UsageID, AuditAction)`
		`VALUES (@UsageID, 'usage deleted');`
		`END;`

20	trgrdeleteServiceOffices is an AFTER DELETE trigger on the ServiceOffices table. When a row is deleted from the ServiceOffices table, this trigger will insert a new row into the AuditServiceOffices table with the values of the deleted row and an AuditAction value of 'office deleted'.	`CREATE TRIGGER trgrdeleteServiceOffices`
		`ON ServiceOffices`
		`AFTER delete`
		`AS`
		`BEGIN`
		`DECLARE @OfficeID INT,`
		`@City VARCHAR(255),`
		`@ZipCode VARCHAR(255);`
		`SELECT @OfficeID = i.OfficeID,@City=i.City,@ZipCode=i.ZipCode FROM inserted i;`
		`INSERT INTO AuditServiceOffices`
		`VALUES (@OfficeID, @City,@ZipCode,'office deleted');`
		`END;`

27. Single-Row Functions UPPER, LOWER, LENGTH, SUBSTR using logical operators – 50 Queries

1	This query selects the first name from the Customers table for customers with the first name 'John' and last name	`SELECT UPPER(FirstName) FROM Customers WHERE FirstName = 'John' AND LastName = 'Doe';`

	‘Doe’, and converts it to uppercase.	
2	This query selects the last name from the Customers table for customers with the first name ‘Jane’ or last name ‘Smith’, and converts it to uppercase.	`SELECT UPPER(LastName) FROM Customers WHERE FirstName = 'Jane' OR LastName = 'Smith';`
3	This query selects the email from the Customers table for customers with a first name not equal to ‘Bob’, and converts it to uppercase.	`SELECT UPPER(Email) FROM Customers WHERE NOT FirstName = 'Bob';`
4	This query selects first name from the Customers table for customers with the first name ‘Alice’ and last name ‘Johnson’, and converts it to uppercase.	`SELECT UPPER(FirstName) FROM Customers WHERE FirstName = 'Alice' AND LastName = 'Johnson';`
5	This query selects the city from the Addresses table for addresses with a zip code of ‘12345’ or a street address of ‘123 Main St’, and converts it to uppercase.	`SELECT UPPER(City) FROM Addresses WHERE ZipCode = '12345' OR StreetAddress = '123 Main St';`
6	This query selects the district from the Addresses table for addresses with a city not equal to ‘New York’, and converts it to uppercase.	`SELECT UPPER(District) FROM Addresses WHERE NOT City = 'New York';`
7	This query selects the service name from the Servicess table for services with a monthly fee greater than 50 and a description containing ‘internet’, and converts it to uppercase.	`SELECT UPPER(ServiceName) FROM Servicess WHERE MonthlyFee > 50 AND ServiceDescription LIKE '%internet%';`
8	This query selects the service description from the Servicess table for services with a monthly fee less than 30 or a service	`SELECT UPPER(ServiceDescription) FROM Servicess WHERE MonthlyFee < 30 OR ServiceName = 'Cable TV';`

	name of ‘Cable TV’, and converts it to uppercase.	
9	This query selects the start date from the Subscriptions table for subscriptions with a customer ID of 1 and a service ID of 2, converts it to a varchar data type in the format ‘mm/dd/yyyy’ using the CONVERT function, and then converts it to uppercase.	`SELECT UPPER(CONVERT(varchar, StartDate, 101)) FROM Subscriptions WHERE CustomerID = 1 AND ServiceID = 2;`
10	This query selects the end date from the Subscriptions table for subscriptions with a customer ID not equal to 3, converts it to a varchar data type in the format ‘mm/dd/yyyy’ using the CONVERT function, and then converts it to uppercase.	`SELECT UPPER(CONVERT(varchar, EndDate, 101)) FROM Subscriptions WHERE NOT CustomerID = 3;`
11	This query selects the first name from the Customers table for customers with the first name ‘John’ and last name ‘Doe’, and converts it to lowercase.	`SELECT LOWER(FirstName) FROM Customers WHERE FirstName = 'John' AND LastName = 'Doe';`
12	This query selects the last name from the Customers table for customers with the first name ‘Jane’ or last name ‘Smith’, and converts it to lowercase.	`SELECT LOWER(LastName) FROM Customers WHERE FirstName = 'Jane' OR LastName = 'Smith';`
13	This query selects the email from the Customers table for customers with a first name not equal to ‘Bob’, and converts it to lowercase.	`SELECT LOWER(Email) FROM Customers WHERE NOT FirstName = 'Bob';`
14	This query selects the first name from the Customers table for	`SELECT LOWER(FirstName) FROM Customers WHERE FirstName = 'Alice' AND LastName = 'Johnson';`

	customers with the first name ‘Alice’ and last name ‘Johnson’, and converts it to lowercase.	
15	This query selects the city from the Addresses table for addresses with a zip code of ‘12345’ or a street address of ‘123 Main St’, and converts it to lowercase.	`SELECT LOWER(City) FROM Addresses WHERE ZipCode = '12345' OR StreetAddress = '123 Main St';`
16	This query selects the district from the Addresses table for addresses with a city not equal to ‘New York’, and converts it to lowercase.	`SELECT LOWER(District) FROM Addresses WHERE NOT City = 'New York';`
17	This query selects the service name from the Servicess table for services with a monthly fee greater than 50 and a description containing ‘internet’, and converts it to lowercase.	`SELECT LOWER(ServiceName) FROM Servicess WHERE MonthlyFee > 50 AND ServiceDescription LIKE '%internet%';`
18	This query selects the service description from the Servicess table for services with a monthly fee less than 30 or a service name of ‘Cable TV’, and converts it to lowercase.	`SELECT LOWER(ServiceDescription) FROM Servicess WHERE MonthlyFee < 30 OR ServiceName = 'Cable TV';`
19	This query selects the start date from the Subscriptions table for subscriptions with a customer ID of 1 and a service ID of 2, converts it to a varchar data type in the format ‘mm/dd/yyyy’ using the CONVERT function, and then converts it to lowercase.	`SELECT LOWER(CONVERT(varchar, StartDate, 101)) FROM Subscriptions WHERE CustomerID = 1 AND ServiceID = 2;`

20	This query selects the end date from the Subscriptions table for subscriptions with a customer ID not equal to 3, converts it to a varchar data type in the format 'mm/dd/yyyy' using the CONVERT function, and then converts it to lowercase.	`SELECT LOWER(CONVERT(varchar, EndDate, 101)) FROM Subscriptions WHERE NOT CustomerID = 3;`
21	This query selects the length of the first name from the Customers table for customers with the first name 'John' and last name 'Doe'.	`SELECT LEN(FirstName) FROM Customers WHERE FirstName = 'John' AND LastName = 'Doe';`
22	This query selects the length of the last name from the Customers table for customers with the first name 'Jane' or last name 'Smith'.	`SELECT LEN(LastName) FROM Customers WHERE FirstName = 'Jane' OR LastName = 'Smith';`
23	This query selects the length of the email from the Customers table for customers with a first name not equal to 'Bob'.	`SELECT LEN(Email) FROM Customers WHERE NOT FirstName = 'Bob';`
24	This query selects the length of the phone number from the Customers table for customers with the first name 'Alice' and last name 'Johnson'.	`SELECT LEN(PhoneNumber) FROM Customers WHERE FirstName = 'Alice' AND LastName = 'Johnson';`
25	This query selects the length of the city from the Addresses table for addresses with a zip code of '12345' or a street address of '123 Main St'.	`SELECT LEN(City) FROM Addresses WHERE ZipCode = '12345' OR StreetAddress = '123 Main St';`
26	This query selects the length of the district from the Addresses table for addresses with a city not equal to 'New York'.	`SELECT LEN(District) FROM Addresses WHERE NOT City = 'New York';`

27	This query selects the length of the service name from the Servicess table for services with a monthly fee greater than 50 and a description containing 'internet'.	`SELECT LEN(ServiceName) FROM Servicess WHERE MonthlyFee > 50 AND ServiceDescription LIKE '%internet%';`
28	This query selects the length of the service description from the Servicess table for services with a monthly fee less than 30 or a service name of 'Cable TV'.	`SELECT LEN(ServiceDescription) FROM Servicess WHERE MonthlyFee < 30 OR ServiceName = 'Cable TV';`
29	This query selects the start date from the Subscriptions table for subscriptions with a customer ID of 1 and a service ID of 2, converts it to a varchar data type in the format 'mm/dd/yyyy' using the CONVERT function, and then calculates its length.	`SELECT LEN(CONVERT(varchar, StartDate, 101)) FROM Subscriptions WHERE CustomerID = 1 AND ServiceID = 2;`
30	This query selects the end date from the Subscriptions table for subscriptions with a customer ID not equal to 3, converts it to a varchar data type in the format 'mm/dd/yyyy' using the CONVERT function, and then calculates its length.	`SELECT LEN(CONVERT(varchar, EndDate, 101)) FROM Subscriptions WHERE NOT CustomerID = 3;`
31	This query selects the first 3 characters of the first name from the Customers table for customers with the first name 'John' and last name 'Doe'.	`SELECT SUBSTRING(FirstName, 1, 3) FROM Customers WHERE FirstName = 'John' AND LastName = 'Doe';`
32	This query selects 4 characters of the last name starting from the second	`SELECT SUBSTRING(LastName, 2, 4) FROM Customers WHERE FirstName = 'Jane' OR LastName = 'Smith';`

	character from the Customers table for customers with the first name ‘Jane’ or last name ‘Smith’.	
33	This query selects the first 5 characters of the email from the Customers table for customers with a first name not equal to ‘Bob’.	`SELECT SUBSTRING(Email, 1, 5) FROM Customers WHERE NOT FirstName = 'Bob';`
34	This query selects 3 characters of the phone number starting from the fourth character from the Customers table for customers with the first name ‘Alice’ and last name ‘Johnson’.	`SELECT SUBSTRING(PhoneNumber, 4, 3) FROM Customers WHERE FirstName = 'Alice' AND LastName = 'Johnson';`
35	This query selects the first 3 characters of the city from the Addresses table for addresses with a zip code of ‘12345’ or a street address of ‘123 Main St’.	`SELECT SUBSTRING(City, 1, 3) FROM Addresses WHERE ZipCode = '12345' OR StreetAddress = '123 Main St';`
36	This query selects 4 characters of the district starting from the second character from the Addresses table for addresses with a city not equal to ‘New York’.	`SELECT SUBSTRING(District, 2, 4) FROM Addresses WHERE NOT City = 'New York';`
37	This query selects the first 5 characters of the service name from the Servicess table for services with a monthly fee greater than 50 and a description containing ‘internet’.	`SELECT SUBSTRING(ServiceName, 1, 5) FROM Servicess WHERE MonthlyFee > 50 AND ServiceDescription LIKE '%internet%';`
38	This query selects 10 characters of the service description starting from the sixth character from the Servicess table for services with a monthly fee	`SELECT SUBSTRING(ServiceDescription, 6, 10) FROM Servicess WHERE MonthlyFee < 30 OR ServiceName = 'Cable TV';`

	less than 30 or a service name of 'Cable TV'.	
39	This query selects the start date from the Subscriptions table for subscriptions with a customer ID of 1 and a service ID of 2, converts it to a varchar data type in the format 'mm/dd/yyyy' using the CONVERT function, and then extracts its first five characters.	`SELECT SUBSTRING(CONVERT(varchar, StartDate, 101), 1, 5) FROM Subscriptions WHERE CustomerID = 1 AND ServiceID = 2;`
40	This query selects the end date from the Subscriptions table for subscriptions with a customer ID not equal to 3, converts it to a varchar data type in the format 'mm/dd/yyyy' using the CONVERT function and then extracts four characters starting from its sixth character.	`SELECT SUBSTRING(CONVERT(varchar, EndDate, 101), 6, 4) FROM Subscriptions WHERE NOT CustomerID = 3;`
41	This query selects the first 3 characters of the first name from the Customers table and assigns it an alias of ShortName.	`SELECT SUBSTRING(FirstName, 1, 3) AS ShortName FROM Customers;`
42	This query selects the first 3 characters of the last name from the Customers table and assigns it an alias of ShortName.	`SELECT SUBSTRING(LastName, 1, 3) AS ShortName FROM Customers;`
43	This query selects the first 5 characters of the email from the Customers table and assigns it an alias of ShortEmail.	`SELECT SUBSTRING(Email, 1, 5) AS ShortEmail FROM Customers;`
44	This query selects the first 3 characters of the phone number from the Customers table and	`SELECT SUBSTRING(PhoneNumber, 1, 3) AS AreaCode FROM Customers;`

	assigns it an alias of AreaCode.	
45	This query selects the first 10 characters of the street address from the Addresses table and assigns it an alias of ShortAddress.	`SELECT SUBSTRING(StreetAddress, 1, 10) AS ShortAddress FROM Addresses;`
46	This query selects the first 3 characters of the city from the Addresses table and assigns it an alias of ShortCity.	`SELECT SUBSTRING(City, 1, 3) AS ShortCity FROM Addresses;`
47	This query selects the first 3 characters of the district from the Addresses table and assigns it an alias of ShortDistrict.	`SELECT SUBSTRING(District, 1, 3) AS ShortDistrict FROM Addresses;`
48	This query selects the first 5 characters of the zip code from the Addresses table and assigns it an alias of ShortZip.	`SELECT SUBSTRING(ZipCode, 1, 5) AS ShortZip FROM Addresses;`
49	This query selects the first 3 characters of the service name from the Servicess table and assigns it an alias of ShortService	`SELECT SUBSTRING(ServiceName, 1, 3) AS ShortService FROM Servicess;`
50	This query selects the first 10 characters of the service description from the Servicess table and assigns it an alias of ShortDescription.	`SELECT SUBSTRING(ServiceDescription, 1, 10) AS ShortDescription FROM Servicess;`

28. Single-Row Functions TRIM, REPLACE, ROUND, TRUNC using logical operators – 50 Queries

1	This query selects the first name from the Customers table for customers with the first	`SELECT TRIM(FirstName) FROM Customers WHERE FirstName = 'John' AND LastName = 'Doe';`

	name ‘John’ and last name ‘Doe’, and removes any leading or trailing spaces from the first name	
2	This query selects the last name from the Customers table for customers with the first name ‘Jane’ or last name ‘Smith’, and removes any leading or trailing spaces from the last name	`SELECT TRIM(LastName) FROM Customers WHERE FirstName = 'Jane' OR LastName = 'Smith';`
3	This query selects the email from the Customers table for customers with a first name not equal to ‘Bob’, and removes any leading or trailing spaces from the email.	`SELECT TRIM(Email) FROM Customers WHERE NOT FirstName = 'Bob';`
4	This query selects the phone number from the Customers table for customers with the first name ‘Alice’ and last name ‘Johnson’, and removes any leading or trailing spaces from the phone number.	`SELECT TRIM(PhoneNumber) FROM Customers WHERE FirstName = 'Alice' AND LastName = 'Johnson';`
5	This query selects the city from the Addresses table for addresses with a zip code of ‘12345’ or a street address of ‘123 Main St’, and removes any leading or trailing spaces from the city.	`SELECT TRIM(City) FROM Addresses WHERE ZipCode = '12345' OR StreetAddress = '123 Main St';`
6	This query selects the district from the Addresses table for addresses with a city not equal to ‘New York’, and removes any leading or trailing spaces from the district.	`SELECT TRIM(District) FROM Addresses WHERE NOT City = 'New York';`
7	This query selects the service name from the Servicess table for services with a monthly fee	`SELECT TRIM(ServiceName) FROM Servicess WHERE MonthlyFee > 50 AND ServiceDescription LIKE '%internet%';`

	greater than 50 and a description containing 'internet', and removes any leading or trailing spaces from the service name.	
8	This query selects the service description from the Servicess table for services with a monthly fee less than 30 or a service name of 'Cable TV', and removes any leading or trailing spaces from the service description.	`SELECT TRIM(ServiceDescription) FROM Servicess WHERE MonthlyFee < 30 OR ServiceName = 'Cable TV';`
9	This query selects the start date from the Subscriptions table for subscriptions with a customer ID of 1 and a service ID of 2, and removes any leading or trailing spaces from the start date.	`SELECT TRIM(StartDate) FROM Subscriptions WHERE CustomerID = 1 AND ServiceID = 2;`
10	This query selects the end date from the Subscriptions table for subscriptions with a customer ID not equal to 3, and removes any leading or trailing spaces from the end date	`SELECT TRIM(EndDate) FROM Subscriptions WHERE NOT CustomerID = 3;`
11	This query rounds the number 123.456 to 2 decimal places, resulting in 123.46.	`SELECT ROUND(123.456, 2);`
12	This query calculates the average monthly fee from the Servicess table and rounds it to the nearest whole number.	`SELECT ROUND(AVG(MonthlyFee), 0) FROM Servicess;`
13	This query selects the service name and monthly fee from the Servicess table and rounds the monthly fee to the nearest multiple of 10.	`SELECT ServiceName, ROUND(MonthlyFee, -1) FROM Servicess;`

14	This query calculates the total amount due from the Billing table for bills that are past due and rounds it to 2 decimal places.	`SELECT ROUND(SUM(AmountDue), 2) FROM Billing WHERE DueDate < GETDATE();`
15	This query calculates the average data usage per customer from the Usage table and rounds it to 1 decimal place.	`SELECT CustomerID, ROUND(AVG(DataUsed), 1) FROM Usage GROUP BY CustomerID;`
16	This query selects the monthly fee from the Servicess table for services with the name 'Internet' and a description containing 'high speed', and rounds it to the nearest whole number.	`SELECT ROUND(MonthlyFee, 0) FROM Servicess WHERE ServiceName = 'Internet' AND ServiceDescription LIKE '%high speed%';`
17	This query selects the amount due from the Billing table for bills with a subscription ID of 1 or a billing date in the future, and rounds it to 1 decimal place.	`SELECT ROUND(AmountDue, 1) FROM Billing WHERE SubscriptionID = 1 OR BillingDate > GETDATE();`
18	This query selects the payment amount from the Payments table for payments with a billing ID not equal to 2, and rounds it to 2 decimal places.	`SELECT ROUND(PaymentAmount, 2) FROM Payments WHERE NOT BillingID = 2;`
19	This query selects the data used from the Usage table for usage records with a subscription ID of 3 and a usage date in the year 2022, and rounds it to the nearest whole number.	`SELECT ROUND(DataUsed, 0) FROM Usage WHERE SubscriptionID = 3 AND UsageDate BETWEEN '2022-01-01' AND '2022-12-31';`
20	This query selects the usage bytes from the Usage table for usage records with a subscription ID not equal to 4, converts it to a float data type, and rounds it to 1 decimal place.	`SELECT ROUND(CAST(UsageBytes AS float), 1) FROM Usage WHERE NOT SubscriptionID = 4;`

21	This query calculates the total monthly fee from the Servicess table if there exists a service with the name ‘Cable TV’, and rounds it to the nearest whole number.	`SELECT ROUND((SELECT SUM(MonthlyFee) FROM Servicess), 0) WHERE EXISTS (SELECT ServiceID FROM Servicess WHERE ServiceName = 'Cable TV');`
22	This query calculates the average monthly fee from the Servicess table for services with a description containing ‘unlimited’, and rounds it to 1 decimal place.	`SELECT ROUND(AVG(MonthlyFee), 1) FROM Servicess WHERE ServiceDescription LIKE '%unlimited%';`
23	This query selects the maximum monthly fee from the Servicess table for services with the name ‘Phone’ or a description containing ‘unlimited calls’, and rounds it to 2 decimal places.	`SELECT ROUND(MAX(MonthlyFee), 2) FROM Servicess WHERE ServiceName = 'Phone' OR ServiceDescription LIKE '%unlimited calls%';`
24	This query selects the minimum monthly fee from the Servicess table for services with a name not equal to ‘Internet’, and rounds it to the nearest whole number.	`SELECT ROUND(MIN(MonthlyFee), 0) FROM Servicess WHERE NOT ServiceName = 'Internet';`
25	This query calculates the average monthly fee for each service in the Servicess table, rounded to 1 decimal place. The results are grouped by service name and the service name is also included in the output.	`SELECT ServiceName, ROUND(AVG(MonthlyFee), 1) FROM Servicess GROUP BY ServiceName;`
26	The query replaces all occurrences of the letter ‘a’ with the letter ‘o’ in the FirstName column of rows from the Customers table where FirstName = ‘Jane’ AND LastName = ‘Doe’.	`SELECT REPLACE(FirstName, 'a', 'o') FROM Customers WHERE FirstName = 'Jane' AND LastName = 'Doe';`
27	The query replaces all occurrences of the letter ‘s’	`SELECT REPLACE(LastName, 's', 'z') FROM Customers WHERE FirstName = 'John' OR LastName = 'Smith';`

	with the letter ‘z’ in the LastName column of rows from the Customers table where FirstName = ‘John’ OR LastName = ‘Smith’.	
28	The query replaces all occurrences of the ‘@’ symbol with the ‘#’ symbol in the Email column of rows from the Customers table where NOT FirstName = ‘Bob’.	`SELECT REPLACE(Email, '@', '#') FROM Customers WHERE NOT FirstName = 'Bob';`
29	The query replaces all occurrences of the ‘-’ character with an empty string in the PhoneNumber column of rows from the Customers table where FirstName = ‘Alice’ AND LastName = ‘Johnson’.	`SELECT REPLACE(PhoneNumber, '-', '') FROM Customers WHERE FirstName = 'Alice' AND LastName = 'Johnson';`
30	The query replaces all occurrences of the word ‘New’ with the word ‘Old’ in the City column of rows from the Addresses table where ZipCode = ‘12345’ OR StreetAddress = ‘123 Main St’.	`SELECT REPLACE(City, 'New', 'Old') FROM Addresses WHERE ZipCode = '12345' OR StreetAddress = '123 Main St';`
31	The query replaces all occurrences of the word ‘East’ with the word ‘West’ in the District column of rows from the Addresses table where NOT City = ‘New York’.	`SELECT REPLACE(District, 'East', 'West') FROM Addresses WHERE NOT City = 'New York';`
32	The query replaces all occurrences of the word ‘Internet’ with the word ‘Web’ in the ServiceName column of rows from the Servicess table where MonthlyFee > 50 AND ServiceDescription LIKE ‘%high speed%’.	`SELECT REPLACE(ServiceName, 'Internet', 'Web') FROM Servicess WHERE MonthlyFee > 50 AND ServiceDescription LIKE '%high speed%';`
33	The query replaces all occurrences of the word	`SELECT REPLACE(ServiceDescription, 'unlimited', 'limited') FROM Servicess WHERE MonthlyFee < 30 OR ServiceName = 'Cable TV';`

	‘unlimited’ with the word ‘limited’ in the ServiceDescription column of rows from the Servicess table where MonthlyFee < 30 OR ServiceName = ‘Cable TV’.	
34	The query replaces all occurrences of the ‘/’ character with the ‘-’ character in the StartDate column (converted to a varchar using the CONVERT function) of rows from the Subscriptions table where CustomerID = 1 AND ServiceID = 2.	`SELECT REPLACE(CONVERT(varchar, StartDate, 101), '/', '-') FROM Subscriptions WHERE CustomerID = 1 AND ServiceID = 2;`
35	The query replaces all occurrences of the ‘/’ character with the ‘.’ character in the EndDate column (converted to a varchar using the CONVERT function) of rows from the Subscriptions table where NOT CustomerID = 3.	`SELECT REPLACE(CONVERT(varchar, EndDate, 101), '/', '.') FROM Subscriptions WHERE NOT CustomerID = 3;`
36	The query replaces all occurrences of the ‘/’ character with the ‘,’ character in the BillingDate column (converted to a varchar using the CONVERT function) of rows from the Billing table where SubscriptionID = 4 AND AmountDue > 100.	`SELECT REPLACE(CONVERT(varchar, BillingDate, 101), '/', ',') FROM Billing WHERE SubscriptionID = 4 AND AmountDue > 100;`
37	The query replaces all occurrences of the ‘.’ character with the ‘,’ character in the AmountDue column (converted to a varchar using the CONVERT function) of rows from the Billing table where DueDate <	`SELECT REPLACE(CONVERT(varchar, AmountDue), '.', ',') FROM Billing WHERE DueDate < GETDATE() OR BillingID = 5;`

	GETDATE() OR BillingID = 5.	
38	The query replaces all occurrences of the '/' character with the '|' character in the PaymentDate column (converted to a varchar using the CONVERT function) of rows from the Payments table where BillingID = 6 AND PaymentAmount < 50.	`SELECT REPLACE(CONVERT(varchar, PaymentDate, 101), '/', '|') FROM Payments WHERE BillingID = 6 AND PaymentAmount < 50;`
39	The query replaces all occurrences of the '.' character with the ':' character in the PaymentAmount column (converted to a varchar using the CONVERT function) of rows from the Payments table where NOT PaymentDate > GETDATE().	`SELECT REPLACE(CONVERT(varchar, PaymentAmount), '.', ':') FROM Payments WHERE NOT PaymentDate > GETDATE();`
40	The query replaces all occurrences of the '.' character with the '_' character in the DataUsed column (converted to a varchar using the CONVERT function) of rows from the Usage table where SubscriptionID = 7 AND UsageDate BETWEEN '2022-01-01' AND '2022-12-31'.	`SELECT REPLACE(CONVERT(varchar, DataUsed), '.', '_') FROM Usage WHERE SubscriptionID = 7 AND UsageDate BETWEEN '2022-01-01' AND '2022-12-31';`
41	The query deletes all data from the Addresses table.	`TRUNCATE TABLE Addresses;`
42	The query deletes all data from the Customers table.	`TRUNCATE TABLE Customers;`
43	The query deletes all data from the Servicess table	`TRUNCATE TABLE Servicess;`
44	The query deletes all data from the Subscriptions table.	`TRUNCATE TABLE Subscriptions;`

45	The query deletes all data from the Billing table.	`TRUNCATE TABLE Billing;`
46	The query deletes all data from the Payments table.	`TRUNCATE TABLE Payments;`
47	The query deletes all data from the Usage table.	`TRUNCATE TABLE Usage;`
48	The query deletes all data from the Technicians table.	`TRUNCATE TABLE Technicians;`
49	The query deletes all data from the Issues table.	`TRUNCATE TABLE Issues;`
50	The query deletes all data from the Equipment table.	`TRUNCATE TABLE Equipment;`

29. Transaction COMMIT and ROLLBACK– 20 Queries

1	The query starts a transaction, inserts a new row into the Customers table with the specified values for FirstName, LastName, Email, and PhoneNumber, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`INSERT INTO Customers (FirstName, LastName, Email, PhoneNumber) VALUES ('John', 'Doe', 'john.doe@email.com', '123-456-7890');`
		`COMMIT;`
		`ROLLBACK;`

2	The query starts a transaction, updates the PhoneNumber of the customer with CustomerID = 1 to '098-765-4321', commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`UPDATE Customers SET PhoneNumber = '098-765-4321' WHERE CustomerID = 1;`
		`COMMIT;`
		`ROLLBACK;`

3	The query starts a transaction, deletes the customer with CustomerID = 2 from the Customers table, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`DELETE FROM Customers WHERE CustomerID = 2;`
		`COMMIT;`
		`ROLLBACK;`

4	The query starts a transaction, inserts a new row into the Servicess table with the specified values for ServiceName, ServiceDescription, and MonthlyFee, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`INSERT INTO Servicess (ServiceName, ServiceDescription, MonthlyFee) VALUES ('Internet', 'High speed internet', 50);`
		`COMMIT;`
		`ROLLBACK;`

5	The query starts a transaction, updates the MonthlyFee of the service with ServiceID = 1 to 60, commits the transaction, and then rolls it back.	BEGIN TRANSACTION; UPDATE Servicess SET MonthlyFee = 60 WHERE ServiceID = 1; COMMIT; ROLLBACK;
6	The query starts a transaction, deletes the service with ServiceID = 2 from the Servicess table, commits the transaction, and then rolls it back.	BEGIN TRANSACTION; DELETE FROM Servicess WHERE ServiceID = 2; COMMIT; ROLLBACK;
7	The query starts a transaction, inserts a new row into the Subscriptions table with the specified values for CustomerID, ServiceID, StartDate, and EndDate, commits the transaction, and then rolls it back.	BEGIN TRANSACTION; INSERT INTO Subscriptions (CustomerID, ServiceID, StartDate, EndDate) VALUES (1, 1, '2022-01-01', '2022-12-31'); COMMIT; ROLLBACK;
8	The query starts a transaction, updates the EndDate of the subscription with SubscriptionID = 1 to '2023-12-31', commits the transaction, and then rolls it back.	BEGIN TRANSACTION; UPDATE Subscriptions SET EndDate = '2023-12-31' WHERE SubscriptionID = 1; COMMIT; ROLLBACK;
9	The query starts a transaction, deletes the subscription with SubscriptionID = 2 from the Subscriptions table, commits the transaction, and then rolls it back.	BEGIN TRANSACTION; DELETE FROM Subscriptions WHERE SubscriptionID = 2; COMMIT; ROLLBACK;
10	The query starts a transaction, inserts a new row into the Billing table with the specified values for SubscriptionID, BillingDate, AmountDue, and DueDate, commits the transaction, and then rolls it back.	BEGIN TRANSACTION; INSERT INTO Billing (SubscriptionID, BillingDate, AmountDue, DueDate) VALUES (1, '2022-01-01', 50, '2022-02-01'); COMMIT; ROLLBACK;
11	The query starts a transaction, updates the AmountDue of the billing record with BillingID = 1 to 60, commits the transaction, and then rolls it back.	BEGIN TRANSACTION; UPDATE billing SET AmountDue = 60 WHERE BillingID = 1; COMMIT; ROLLBACK;
12	The query starts a transaction, deletes the billing record with BillingID = 2 from the Billing table, commits the transaction, and then rolls it back	BEGIN TRANSACTION; delete Billing (SubscriptionID, BillingDate, AmountDue, DueDate) VALUES (1, '2022-01-01', 50, '2022-02-01'); COMMIT;

		`ROLLBACK;`

13	The query starts a transaction, inserts a new row into the Payments table with the specified values for BillingID, PaymentDate, and PaymentAmount, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`INSERT INTO Payments (BillingID, PaymentDate, PaymentAmount) VALUES (1, '2022-02-01', 50);`
		`COMMIT;`
		`ROLLBACK;`

14	The query starts a transaction, updates the PaymentAmount of the payment with PaymentID = 1 to 60, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`UPDATE Payments SET PaymentAmount = 60 WHERE PaymentID = 1;`
		`COMMIT;`
		`ROLLBACK;`

15	The query starts a transaction, deletes the payment with PaymentID = 2 from the Payments table, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`DELETE FROM Payments WHERE PaymentID = 2;`
		`COMMIT;`
		`ROLLBACK;`

16	The query starts a transaction, inserts a new row into the Usage table with the specified values for SubscriptionID, UsageDate, and DataUsed, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`INSERT INTO Usage (SubscriptionID, UsageDate, DataUsed) VALUES (1, '2022-01-01', 1024);`
		`COMMIT;`
		`ROLLBACK;`

17	The query starts a transaction, updates the DataUsed of the usage record with UsageID = 1 to 2048, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`UPDATE Usage SET DataUsed = 2048 WHERE UsageID = 1;`
		`COMMIT;`
		`ROLLBACK;`

18	The query starts a transaction, deletes the usage record with UsageID = 2 from the Usage table, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`DELETE FROM Usage WHERE UsageID = 2;`
		`COMMIT;`
		`ROLLBACK;`

19	The query starts a transaction, inserts a new row into the Technicians table with the specified values for FirstName, LastName, and PhoneNumber, commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`INSERT INTO Technicians (FirstName, LastName, PhoneNumber) VALUES ('John', 'Doe', '123-456-7890');`
		`COMMIT;`
		`ROLLBACK;`

20	The query starts a transaction, updates the PhoneNumber of the technician with TechnicianID = 1 to '098-765-4321', commits the transaction, and then rolls it back.	`BEGIN TRANSACTION;`
		`UPDATE Technicians SET PhoneNumber = '098-765-4321' WHERE TechnicianID = 1;`
		`COMMIT;`
		`ROLLBACK;`

30. Exception Handling- Try Catch– 20 Queries

1	The first query attempts to insert a new row into the Customers table with the specified values for FirstName, LastName, Email, and PhoneNumber. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    INSERT INTO Customers (FirstName, LastName, Email, PhoneNumber) VALUES ('John', 'Doe', 'john.doe@email.com', '123-456-7890');`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

2	The second query attempts to update the PhoneNumber of the customer with CustomerID = 1 to '098-765-4321'. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    UPDATE Customers SET PhoneNumber = '098-765-4321' WHERE CustomerID = 1;`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

3	The third query attempts to delete the customer with CustomerID = 2 from the Customers table. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    DELETE FROM Customers WHERE CustomerID = 2;`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

4	The fourth query attempts to insert a new row into the Servicess table with the specified values for ServiceName, ServiceDescription, and MonthlyFee. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    INSERT INTO Servicess (ServiceName, ServiceDescription, MonthlyFee) VALUES ('Internet', 'High speed internet', 50);`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

5	The fifth query attempts to update the MonthlyFee of the service with ServiceID = 1 to 60. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    UPDATE Servicess SET MonthlyFee = 60 WHERE ServiceID = 1;`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

6	The sixth query attempts to delete the service with ServiceID = 2 from the Servicess table. If an error	`BEGIN TRY`
		`    DELETE FROM Servicess WHERE ServiceID = 2;`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`

	occurs, the error message is printed.	`END CATCH;`

7	The seventh query attempts to insert a new row into the Subscriptions table with the specified values for CustomerID, ServiceID, StartDate, and EndDate. If an error occurs, the error message is printed.	`BEGIN TRY`
		`  INSERT INTO Subscriptions (CustomerID, ServiceID, StartDate, EndDate) VALUES (1, 1, '2022-01-01', '2022-12-31');`
		`END TRY`
		`BEGIN CATCH`
		`  PRINT ERROR_MESSAGE();`
		`END CATCH;`

8	The eighth query attempts to update the EndDate of the subscription with SubscriptionID = 1 to '2023-12-31'. If an error occurs, the error message is printed.	`BEGIN TRY`
		`  UPDATE Subscriptions SET EndDate = '2023-12-31' WHERE SubscriptionID = 1;`
		`END TRY`
		`BEGIN CATCH`
		`  PRINT ERROR_MESSAGE();`
		`END CATCH;`

9	The ninth query attempts to delete the subscription with SubscriptionID = 2 from the Subscriptions table. If an error occurs, the error message is printed.	`BEGIN TRY`
		`  DELETE FROM Subscriptions WHERE SubscriptionID = 2;`
		`END TRY`
		`BEGIN CATCH`
		`  PRINT ERROR_MESSAGE();`
		`END CATCH;`

10	The query attempts to insert a new row into the Billing table with the specified values for SubscriptionID, BillingDate, AmountDue, and DueDate. If an error occurs, the error message is printed.	`BEGIN TRY`
		`  INSERT INTO Billing (SubscriptionID, BillingDate, AmountDue, DueDate) VALUES (1, '2022-01-01', 50, '2022-02-01');`
		`END TRY`
		`BEGIN CATCH`
		`  PRINT ERROR_MESSAGE();`
		`END CATCH;`

11	The query attempts to update the AmountDue of the billing record with BillingID = 1 to 60. If an error occurs, the error message is printed.	`BEGIN TRY`
		`  UPDATE Billing SET AmountDue = 60 WHERE BillingID = 1;`
		`END TRY`
		`BEGIN CATCH`
		`  PRINT ERROR_MESSAGE();`
		`END CATCH;`

12	The query attempts to delete the billing record with BillingID = 2 from the	`BEGIN TRY`
		`  DELETE FROM Billing WHERE BillingID = 2;`
		`END TRY`
		`BEGIN CATCH`

	Billing table. If an error occurs, the error message is printed.	`    PRINT ERROR_MESSAGE();` `END CATCH;`

13	The query attempts to insert a new row into the Payments table with the specified values for BillingID, PaymentDate, and PaymentAmount. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    INSERT INTO Payments (BillingID, PaymentDate, PaymentAmount) VALUES (1, '2022-02-01', 50);`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

14	The query attempts to update the PaymentAmount of the payment with PaymentID = 1 to 60. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    UPDATE Payments SET PaymentAmount = 60 WHERE PaymentID = 1;`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

15	The query attempts to delete the payment with PaymentID = 2 from the Payments table. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    DELETE FROM Payments WHERE PaymentID = 2;`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

16	The query attempts to insert a new row into the Usage table with the specified values for SubscriptionID, UsageDate, and DataUsed. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    INSERT INTO Usage (SubscriptionID, UsageDate, DataUsed) VALUES (1, '2022-01-01', 1024);`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

17	The query attempts to update the DataUsed of the usage record with UsageID = 1 to 2048. If an error occurs, the error message is printed.	`BEGIN TRY`
		`    UPDATE Usage SET DataUsed = 2048 WHERE UsageID = 1;`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`
		`END CATCH;`

18	The query attempts to delete the usage record with UsageID = 2 from the Usage table. If an error	`BEGIN TRY`
		`    DELETE FROM Usage WHERE UsageID = 2;`
		`END TRY`
		`BEGIN CATCH`
		`    PRINT ERROR_MESSAGE();`

	occurs, the error message is printed.	`END CATCH;`

19	The query attempts to insert a new row into the Technicians table with the specified values for FirstName, LastName, and PhoneNumber. If an error occurs, the error message is printed.	`BEGIN TRY`
		`  INSERT INTO Technicians (FirstName, LastName, PhoneNumber) VALUES ('John', 'Doe', '123-456-7890');`
		`END TRY`
		`BEGIN CATCH`
		`  PRINT ERROR_MESSAGE();`
		`END CATCH;`

20	The query attempts to update the PhoneNumber of the technician with TechnicianID = 1 to '098-765-4321'. If an error occurs, the error message is printed.	`BEGIN TRY`
		`  UPDATE Technicians SET PhoneNumber = '098-765-4321' WHERE TechnicianID = 1;`
		`END TRY`
		`BEGIN CATCH`
		`  PRINT ERROR_MESSAGE();`
		`END CATCH;`