**PRACTICAL NO.: 01**

**Title**:

Design and Create 3 Real-World system database using Entities, Attributes, and Relationship

**Objective:**

1. To create database for ecommerce
2. To create database for college attendance
3. To create database for college canteen

**Theory:**

A database is information that is set up for easy access, management and updating. Computer databases typically store aggregations of [data](https://www.techtarget.com/searchdatamanagement/definition/data) records or files that contain information, such as sales transactions, customer data, financials and product information. Databases are used for storing, maintaining and accessing any sort of data. They collect information on people, places or things. That information is gathered in one place so that it can be observed and analyzed. Databases can be thought of as an organized collection of information.

1. The E-commerce Database, named ecommerce\_db, serves as the backbone for managing various aspects of an online shopping platform. This report provides an in-depth overview of the database structure, including its entities, relationships, and functionalities.

**Entity and Attribute Descriptions:**

* buyer: Stores information about buyers who make purchases on the platform.

Attributes: buyer\_id, user\_id (foreign key referencing user table), full\_name, shipping\_address, payment\_info, created\_at, updated\_at.

* category: Contains categories to which products belong.

Attributes: category\_id, category\_name.

* order: Records details of orders placed by buyers.

Attributes: order\_id, buyer\_id (foreign key referencing buyer table), product\_id (foreign key referencing product table), quantity, total\_price, order\_date, order\_status, created\_at, updated\_at.

* payment: Stores payment information for orders.

Attributes: payment\_id, order\_id (foreign key referencing order table), payment\_date, amount, payment\_method, payment\_status, created\_at, updated\_at.

* product: Contains details about products available for sale.

Attributes: product\_id, seller\_id (foreign key referencing seller table), category\_id (foreign key referencing category table), subcategory\_id (foreign key referencing subcategory table), product\_name, description, price, quantity\_available, created\_at, updated\_at.

* role: Defines user roles.

Attributes: role\_id, role\_name.

* seller: Stores information about sellers who list products on the platform.

Attributes: seller\_id, user\_id (foreign key referencing user table), company\_name, contact\_info, address, created\_at, updated\_at.

* subcategory: Contains subcategories of products.

Attributes: subcategory\_id, category\_id (foreign key referencing category table), subcategory\_name, created\_at, updated\_at.

* user: Stores user account information.

Attributes: user\_id, username, password, email, role\_id (foreign key referencing role table), created\_at, updated\_at.

**Relationships**:

* Buyer - User: One-to-one relationship: Each buyer is associated with one user account.
* Order - Buyer: Many-to-one relationship: Many orders can belong to one buyer.
* Order - Product: Many-to-one relationship: Many orders can contain one product.
* Payment - Order: One-to-one relationship: Each payment is associated with one order.
* Product - Seller: Many-to-one relationship: Many products can belong to one seller.
* Product - Category: Many-to-one relationship: Many products can belong to one category.
* Product - Subcategory: Many-to-one relationship: Many products can belong to one subcategory.
* Seller - User: One-to-one relationship: Each seller is associated with one user account.
* Subcategory - Category: Many-to-one relationship: Many subcategories can belong to one category.
* User - Role: Many-to-one relationship: Many users can have one role.

1. A "College Canteen Database" is designed to manage the operations of a canteen or cafeteria within a college or educational institution. It facilitates the tracking of food items, orders, payments, and other relevant information.

**Entity and Attribute Descriptions:**

* Customers:
  + Attributes:
  + customer\_id (Primary Key)
  + name
  + contact\_number
  + email
* Feedback:
  + Attributes:
  + feedback\_id (Primary Key)
  + customer\_id (Foreign Key referencing Customers)
  + feedback\_text
  + feedback\_date
* Menu Items:
  + Attributes:
  + item\_id (Primary Key)
  + item\_name
  + price
* Orders:
  + Attributes:
  + order\_id (Primary Key)
  + customer\_id (Foreign Key referencing Customers)
  + item\_id (Foreign Key referencing Menu Items)
  + quantity
  + order\_date
* Payments:
  + Attributes:
  + payment\_id (Primary Key)
  + order\_id (Foreign Key referencing Orders)
  + amount
  + payment\_date
* Staff:
  + Attributes:
  + staff\_id (Primary Key)
  + name
  + position
  + contact\_number
  + email

**Relationships**:

* + Customers - Feedback: One-to-many relationship: One customer can provide multiple feedbacks.
  + Customers - Orders: One-to-many relationship: One customer can place multiple orders.
  + Customers - Payments: One-to-many relationship: One customer can make multiple payments.
  + Menu Items - Orders: One-to-many relationship: One menu item can be included in multiple orders.
  + Orders - Payments: One-to-one relationship: Each order can have only one payment associated with it.
  + Customers - Staff: Many-to-many relationship: One customer can interact with multiple staff members, and one staff member can interact with multiple customers.

1. The "Student Attendance Database" is designed to manage and track attendance records for students in an educational institution. It typically includes tables for storing information about students, courses, batches, departments, faculties, universities, and attendance records.

**Entity and Attribute Descriptions:**

* Attendance:
  + Attributes:
  + Attendance\_ID (Primary Key)
  + Student\_ID (Foreign Key referencing Student)
  + Course\_ID (Foreign Key referencing Course)
  + AttendanceDate
  + AttendanceStatus
* Batch:
  + Attributes:
  + Batch\_ID (Primary Key)
  + Batch\_Name
  + Department\_ID (Foreign Key referencing Department)
* Course:
  + Attributes:
  + Course\_ID (Primary Key)
  + CourseName
  + CourseCode
  + Department\_ID (Foreign Key referencing Department)
  + Batch\_ID (Foreign Key referencing Batch)
* Department:
  + Attributes:
  + Department\_ID (Primary Key)
  + Department\_Name
  + Faculty\_ID (Foreign Key referencing Faculty)
* Faculty:
  + Attributes:
  + Faculty\_ID (Primary Key)
  + Faculty\_Name
  + University\_ID (Foreign Key referencing University)
* Student:
  + Attributes:
  + Student\_ID (Primary Key)
  + FirstName
  + LastName
  + DOB
  + Department\_ID (Foreign Key referencing Department)
* University:
  + Attributes:
  + University\_ID (Primary Key)
  + University\_Name
  + Location
  + Phone\_No

**Relationships**:

* Attendance - Student: Many-to-one relationship: Many attendances can be associated with one student.
* Attendance - Course: Many-to-one relationship: Many attendances can be associated with one course.
* Batch - Department: Many-to-one relationship: Many batches can belong to one department.
* Course - Department: Many-to-one relationship: Many courses can belong to one department.
* Course - Batch: Many-to-one relationship: Many courses can belong to one batch.
* Department - Faculty: Many-to-one relationship: Many departments can belong to one faculty.
* Faculty - University: Many-to-one relationship: Many faculties can belong to one university.
* Student - Department: Many-to-one relationship: Many students can belong to one department.

**Database Schema Script:**

1. **Database schema of ‘ecommerce\_db’**

CREATE TABLE `buyer` (

`buyer\_id` int(11) NOT NULL,

`user\_id` int(11) NOT NULL,

`full\_name` varchar(255) NOT NULL,

`shipping\_address` varchar(255) NOT NULL,

`payment\_info` varchar(255) NOT NULL,

`created\_at` timestamp NOT NULL DEFAULT current\_timestamp(),

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `category` (

`category\_id` int(11) NOT NULL,

`category\_name` varchar(255) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `order` (

`order\_id` int(11) NOT NULL,

`buyer\_id` int(11) NOT NULL,

`product\_id` int(11) NOT NULL,

`quantity` int(11) NOT NULL,

`total\_price` decimal(10,2) NOT NULL,

`order\_date` timestamp NOT NULL DEFAULT current\_timestamp(),

`order\_status` varchar(255) NOT NULL,

`created\_at` timestamp NOT NULL DEFAULT current\_timestamp(),

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `payment` (

`payment\_id` int(11) NOT NULL,

`order\_id` int(11) NOT NULL,

`payment\_date` timestamp NOT NULL DEFAULT current\_timestamp(),

`amount` decimal(10,2) NOT NULL,

`payment\_method` varchar(255) NOT NULL,

`payment\_status` varchar(255) NOT NULL,

`created\_at` timestamp NOT NULL DEFAULT current\_timestamp(),

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `product` (

`product\_id` int(11) NOT NULL,

`seller\_id` int(11) NOT NULL,

`category\_id` int(11) NOT NULL,

`subcategory\_id` int(11) NOT NULL,

`product\_name` varchar(255) NOT NULL,

`description` text DEFAULT NULL,

`price` decimal(10,2) NOT NULL,

`quantity\_available` int(11) NOT NULL,

`created\_at` timestamp NOT NULL DEFAULT current\_timestamp(),

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `role` (

`role\_id` int(11) NOT NULL,

`role\_name` varchar(255) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `seller` (

`seller\_id` int(11) NOT NULL,

`user\_id` int(11) NOT NULL,

`company\_name` varchar(255) NOT NULL,

`contact\_info` varchar(255) DEFAULT NULL,

`address` varchar(255) DEFAULT NULL,

`created\_at` timestamp NOT NULL DEFAULT current\_timestamp(),

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `subcategory` (

`subcategory\_id` int(11) NOT NULL,

`category\_id` int(11) NOT NULL,

`subcategory\_name` varchar(255) NOT NULL,

`created\_at` timestamp NOT NULL DEFAULT current\_timestamp(),

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `user` (

`user\_id` int(11) NOT NULL,

`username` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

`role\_id` int(11) NOT NULL,

`created\_at` timestamp NOT NULL DEFAULT current\_timestamp(),

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

ALTER TABLE `buyer`

ADD PRIMARY KEY (`buyer\_id`),

ADD KEY `user\_id` (`user\_id`);

ALTER TABLE `category`

ADD PRIMARY KEY (`category\_id`);

ALTER TABLE `order`

ADD PRIMARY KEY (`order\_id`),

ADD KEY `buyer\_id` (`buyer\_id`),

ADD KEY `product\_id` (`product\_id`);

ALTER TABLE `payment`

ADD PRIMARY KEY (`payment\_id`),

ADD KEY `order\_id` (`order\_id`);

ALTER TABLE `product`

ADD PRIMARY KEY (`product\_id`),

ADD KEY `seller\_id` (`seller\_id`),

ADD KEY `category\_id` (`category\_id`),

ADD KEY `subcategory\_id` (`subcategory\_id`);

ALTER TABLE `role`

ADD PRIMARY KEY (`role\_id`);

ALTER TABLE `seller`

ADD PRIMARY KEY (`seller\_id`),

ADD KEY `user\_id` (`user\_id`);

ALTER TABLE `subcategory`

ADD PRIMARY KEY (`subcategory\_id`),

ADD KEY `category\_id` (`category\_id`);

ALTER TABLE `user`

ADD PRIMARY KEY (`user\_id`),

ADD UNIQUE KEY `username` (`username`),

ADD UNIQUE KEY `email` (`email`),

ADD KEY `role\_id` (`role\_id`);

ALTER TABLE `buyer`

ADD CONSTRAINT `buyer\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`);

ALTER TABLE `order`

ADD CONSTRAINT `order\_ibfk\_1` FOREIGN KEY (`buyer\_id`) REFERENCES `buyer` (`buyer\_id`),

ADD CONSTRAINT `order\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `product` (`product\_id`);

ALTER TABLE `payment`

ADD CONSTRAINT `payment\_ibfk\_1` FOREIGN KEY (`order\_id`) REFERENCES `order` (`order\_id`);

ALTER TABLE `product`

ADD CONSTRAINT `product\_ibfk\_1` FOREIGN KEY (`seller\_id`) REFERENCES `seller` (`seller\_id`),

ADD CONSTRAINT `product\_ibfk\_2` FOREIGN KEY (`category\_id`) REFERENCES `category` (`category\_id`),

ADD CONSTRAINT `product\_ibfk\_3` FOREIGN KEY (`subcategory\_id`) REFERENCES `subcategory` (`subcategory\_id`);

ALTER TABLE `seller`

ADD CONSTRAINT `seller\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`);

ALTER TABLE `subcategory`

ADD CONSTRAINT `subcategory\_ibfk\_1` FOREIGN KEY (`category\_id`) REFERENCES `category` (`category\_id`);

ALTER TABLE `user`

ADD CONSTRAINT `user\_ibfk\_1` FOREIGN KEY (`role\_id`) REFERENCES `role` (`role\_id`);

COMMIT;

1. **Database schema of ‘college\_canteen\_db’**

CREATE TABLE `customers` (

`customer\_id` int(11) NOT NULL,

`name` varchar(100) NOT NULL,

`contact\_number` bigint(20) DEFAULT NULL,

`email` varchar(50) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `feedback` (

`feedback\_id` int(11) NOT NULL,

`customer\_id` int(11) DEFAULT NULL,

`feedback\_text` text DEFAULT NULL,

`feedback\_date` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `menu\_items` (

`item\_id` int(11) NOT NULL,

`item\_name` varchar(100) NOT NULL,

`price` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `orders` (

`order\_id` int(11) NOT NULL,

`customer\_id` int(11) NOT NULL,

`item\_id` int(11) NOT NULL,

`quantity` int(11) NOT NULL,

`order\_date` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `payments` (

`payment\_id` int(11) NOT NULL,

`order\_id` int(11) NOT NULL,

`amount` int(11) NOT NULL,

`payment\_date` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `staff` (

`staff\_id` int(11) NOT NULL,

`name` varchar(100) NOT NULL,

`position` varchar(100) NOT NULL,

`contact\_number` bigint(20) NOT NULL,

`email` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

ALTER TABLE `customers`

ADD PRIMARY KEY (`customer\_id`);

ALTER TABLE `feedback`

ADD PRIMARY KEY (`feedback\_id`),

ADD KEY `fk\_feedback\_customers` (`customer\_id`);

ALTER TABLE `menu\_items`

ADD PRIMARY KEY (`item\_id`);

ALTER TABLE `orders`

ADD PRIMARY KEY (`order\_id`),

ADD KEY `fk\_orders\_customers` (`customer\_id`),

ADD KEY `fk\_ordera\_items` (`item\_id`);

ALTER TABLE `payments`

ADD PRIMARY KEY (`payment\_id`),

ADD KEY `fk\_payments\_orders` (`order\_id`);

ALTER TABLE `staff`

ADD PRIMARY KEY (`staff\_id`);

ALTER TABLE `customers`

MODIFY `customer\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;

ALTER TABLE `feedback`

MODIFY `feedback\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

ALTER TABLE `menu\_items`

MODIFY `item\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

ALTER TABLE `orders`

MODIFY `order\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

ALTER TABLE `payments`

MODIFY `payment\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

ALTER TABLE `staff`

MODIFY `staff\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

ALTER TABLE `feedback`

ADD CONSTRAINT `fk\_feedback\_customers` FOREIGN KEY (`customer\_id`) REFERENCES `customers` (`customer\_id`);

ALTER TABLE `orders`

ADD CONSTRAINT `fk\_ordera\_items` FOREIGN KEY (`item\_id`) REFERENCES `menu\_items` (`item\_id`),

ADD CONSTRAINT `fk\_orders\_customers` FOREIGN KEY (`customer\_id`) REFERENCES `customers` (`customer\_id`);

ALTER TABLE `payments`

ADD CONSTRAINT `fk\_payments\_orders` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`order\_id`);

COMMIT;

1. **Database schema of ‘student\_attendance\_db’**

CREATE TABLE `attendance` (

`Attendance\_ID` int(11) NOT NULL,

`Student\_ID` int(11) DEFAULT NULL,

`Course\_ID` int(11) DEFAULT NULL,

`AttendanceDate` date DEFAULT NULL,

`AttendanceStatus` varchar(20) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `batch` (

`Batch\_ID` int(11) NOT NULL,

`Batch\_Name` varchar(20) DEFAULT NULL,

`Department\_ID` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `course` (

`Course\_ID` int(11) NOT NULL,

`CourseName` varchar(255) DEFAULT NULL,

`CourseCode` varchar(20) DEFAULT NULL,

`Department\_ID` int(11) DEFAULT NULL,

`Batch\_ID` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `department` (

`Department\_ID` int(11) NOT NULL,

`Department\_Name` varchar(255) DEFAULT NULL,

`Faculty\_ID` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `faculty` (

`Faculty\_ID` int(11) NOT NULL,

`Faculty\_Name` varchar(255) DEFAULT NULL,

`University\_ID` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `student` (

`Student\_ID` int(11) NOT NULL,

`FirstName` varchar(255) DEFAULT NULL,

`LastName` varchar(255) DEFAULT NULL,

`DOB` date DEFAULT NULL,

`Department\_ID` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

CREATE TABLE `university` (

`University\_ID` int(11) NOT NULL,

`University\_Name` varchar(255) DEFAULT NULL,

`Location` varchar(255) DEFAULT NULL,

`Phone\_No` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

ALTER TABLE `attendance`

ADD PRIMARY KEY (`Attendance\_ID`),

ADD KEY `Student\_ID` (`Student\_ID`),

ADD KEY `Course\_ID` (`Course\_ID`);

ALTER TABLE `batch`

ADD PRIMARY KEY (`Batch\_ID`),

ADD KEY `Department\_ID` (`Department\_ID`);

ALTER TABLE `course`

ADD PRIMARY KEY (`Course\_ID`),

ADD KEY `Department\_ID` (`Department\_ID`),

ADD KEY `Batch\_ID` (`Batch\_ID`);

ALTER TABLE `department`

ADD PRIMARY KEY (`Department\_ID`),

ADD KEY `Faculty\_ID` (`Faculty\_ID`);

ALTER TABLE `faculty`

ADD PRIMARY KEY (`Faculty\_ID`),

ADD KEY `University\_ID` (`University\_ID`);

ALTER TABLE `student`

ADD PRIMARY KEY (`Student\_ID`),

ADD KEY `Department\_ID` (`Department\_ID`);

ALTER TABLE `university`

ADD PRIMARY KEY (`University\_ID`);

ALTER TABLE `attendance`

ADD CONSTRAINT `attendance\_ibfk\_1` FOREIGN KEY (`Student\_ID`) REFERENCES `student` (`Student\_ID`),

ADD CONSTRAINT `attendance\_ibfk\_2` FOREIGN KEY (`Course\_ID`) REFERENCES `course` (`Course\_ID`);

ALTER TABLE `batch`

ADD CONSTRAINT `batch\_ibfk\_1` FOREIGN KEY (`Department\_ID`) REFERENCES `department` (`Department\_ID`);

ALTER TABLE `course`

ADD CONSTRAINT `course\_ibfk\_1` FOREIGN KEY (`Department\_ID`) REFERENCES `department` (`Department\_ID`),

ADD CONSTRAINT `course\_ibfk\_2` FOREIGN KEY (`Batch\_ID`) REFERENCES `batch` (`Batch\_ID`);

ALTER TABLE `department`

ADD CONSTRAINT `department\_ibfk\_1` FOREIGN KEY (`Faculty\_ID`) REFERENCES `faculty` (`Faculty\_ID`);

ALTER TABLE `faculty`

ADD CONSTRAINT `faculty\_ibfk\_1` FOREIGN KEY (`University\_ID`) REFERENCES `university` (`University\_ID`);

ALTER TABLE `student`

ADD CONSTRAINT `student\_ibfk\_1` FOREIGN KEY (`Department\_ID`) REFERENCES `department` (`Department\_ID`);

COMMIT;

**Conclusion:**

Hence, we create 3 database ‘ecommerce\_db’, ‘college\_canteen\_db’, and ‘student-attendance\_db’ which are used in real world system using SQL command in Lab.

**ASSIGNMENT 02:**

**Title:**

ER (Entity -Relationship Diagram) of LAB 1

**Objective**:

To Create Er Diagrams of E-Commerce, Social Media & Stock-Market

**Theory**:

An Entity-Relationship (ER) diagram is a visual representation of the entities and relationships within a database schema. It helps to illustrate the structure of a database, including the entities (such as tables), attributes (columns), and relationships (links between entities).

1. **Entity**:

An entity may be any object, class, person, or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- a manager, product, employee, department, etc. can be taken as an entity.



a. Weak Entity

An entity that depends on another entity is called a weak entity. The weak entity doesn't contain any key attribute

of its own. The weak entity is represented by a double rectangle.



**2. Attribute**

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, ID, age, contact number, name, etc. can be attributes of a student.



a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key.

The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

An attribute that is composed of many other attributes is known as a composite attribute. The composite

attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attributes. The double

oval is used to represent multivalued attributes.

For example, a student can have more than one phone number.



d. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented

by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



**3. Relationship**

A relationship is used to describe the relation between entities. A diamond or rhombus is used to represent

the relationship.



Types of relationships are as follows:

a. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

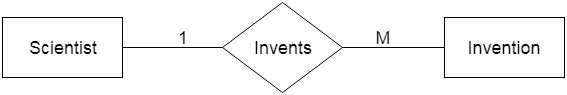


b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates

with the relationship then this is known as a one-to-many relationship.

For example, a Scientist can invent many inventions, but the invention is done by the only a specific scientist.

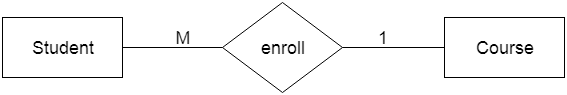


c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates

with the relationship then it is known as a many-to-one relationship.

For example, a Student enrolls for only one course, but a course can have many students.

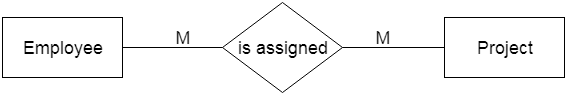


d. Many-to-many relationship

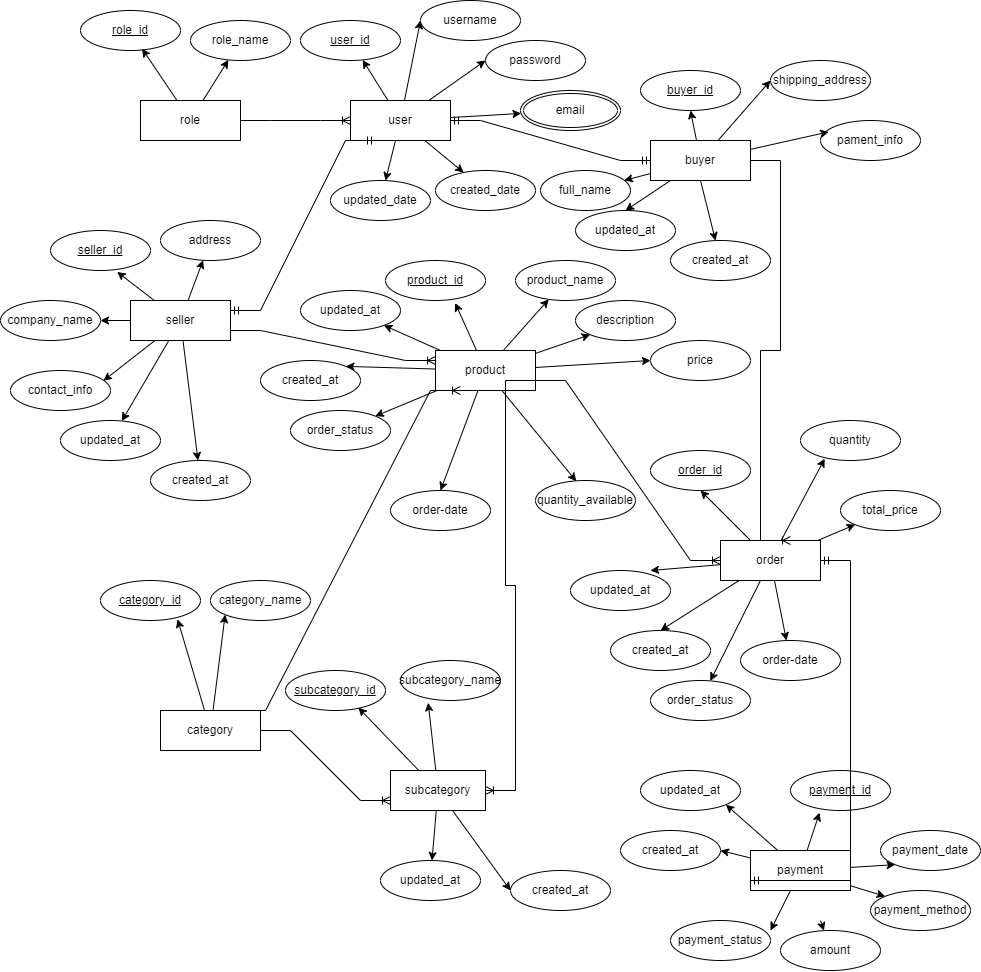
When more than one instance of the entity on the left, and more than one instance of an entity on the right

associates with the relationship then it is known as a many-to-many relationship.

For example, an Employee can be assigned to many projects and a project can have many employees.



**ER Diagram for e-commerce:**



**Conclusion:**

Hence, we created the entity-relation diagram of ecommerce database in Lab.

**PRACTICAL NO.: 03**

**Title**:

Design the Relational database

**Objective:**

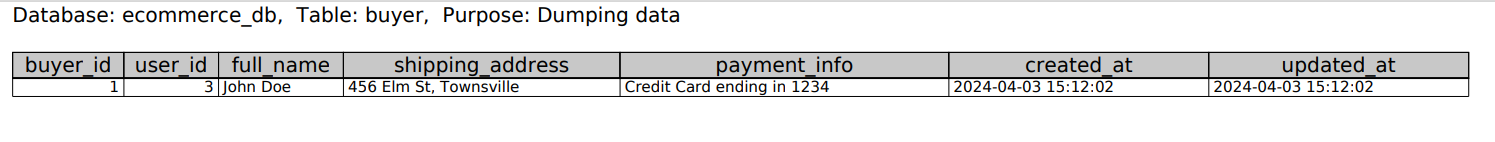
* To create relational database for ecommerce
* To create relational database for college canteen

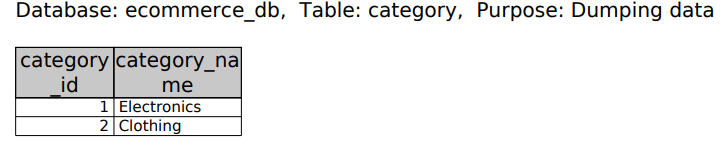
**Theory:**

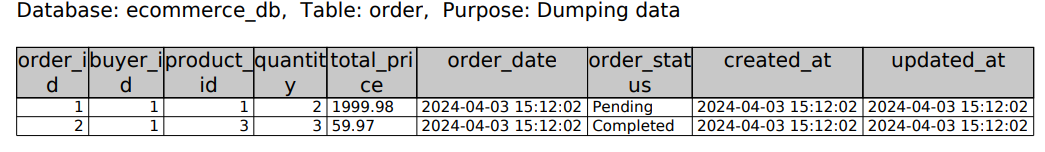
The relational database represents both data and the relationships among those data using relation. A relation is used to represent information about any entity and its relationship with other entities in the form of attributes and tuples. Relational data model represents the logical view of how data is stored in the relational database. A relation schema depicts the attributes of the table and a relation instance is a two-dimensional table with a time-varying set of tuples.

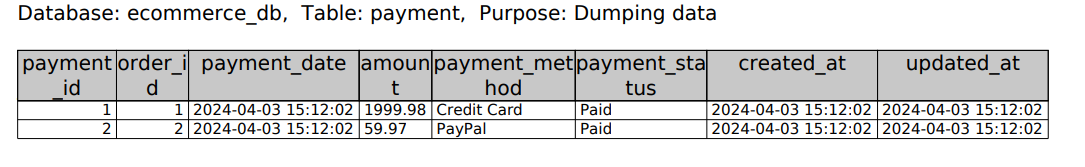
**Structure of relational Database:**

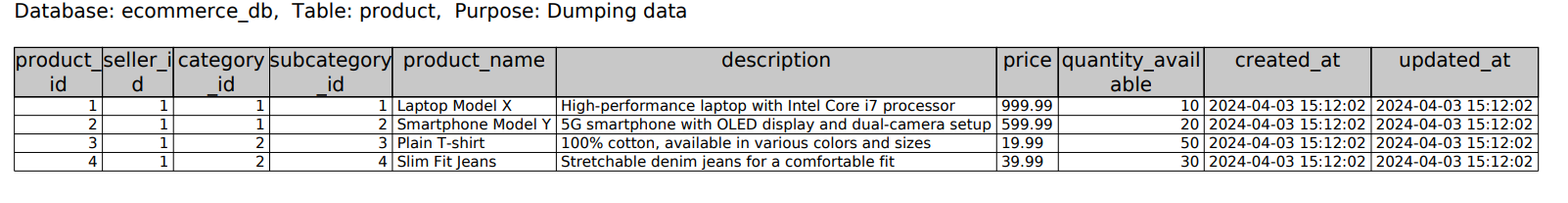
1. **Relational DB of ‘ecommerce’**

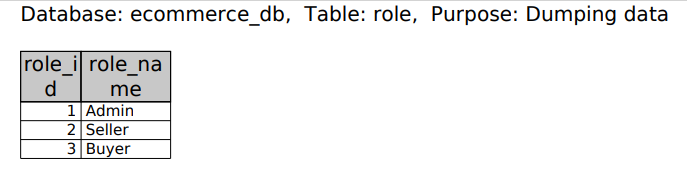


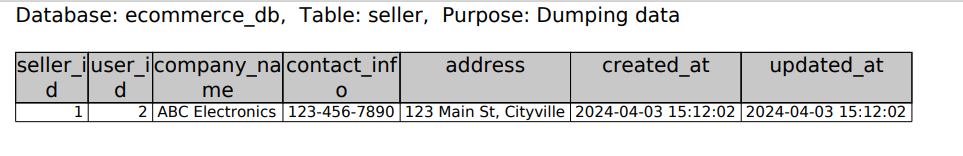


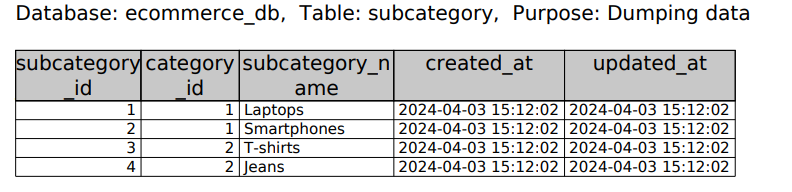


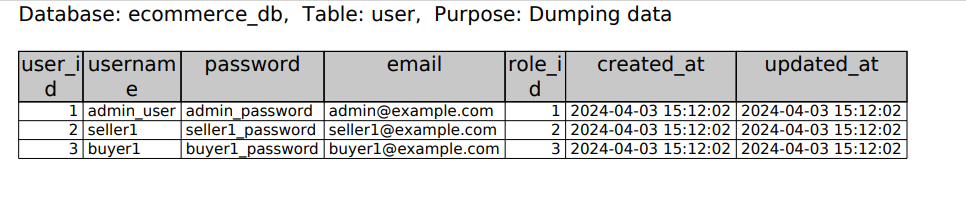




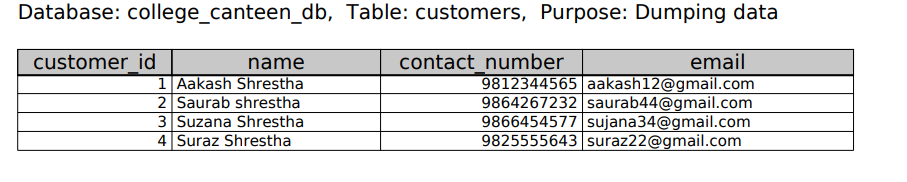


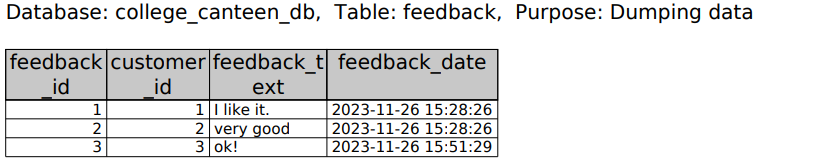


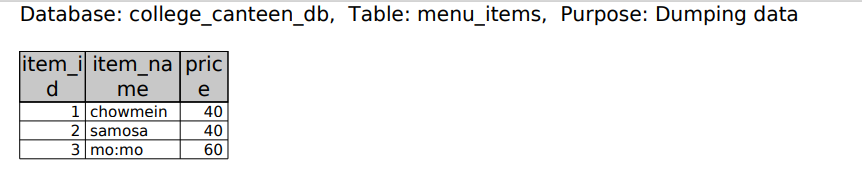


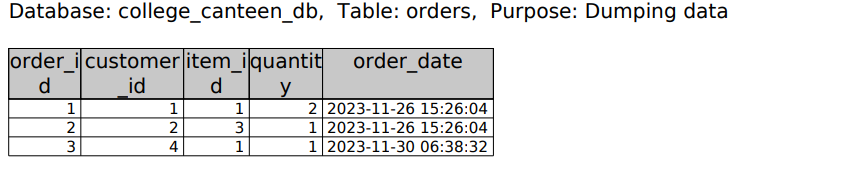


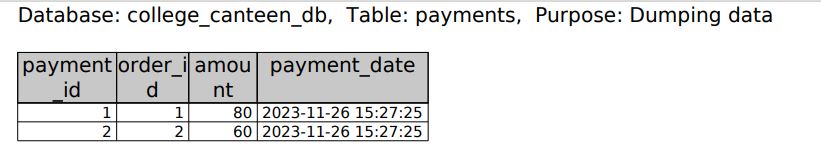
1. **Relational DB of ‘college\_canteen’**

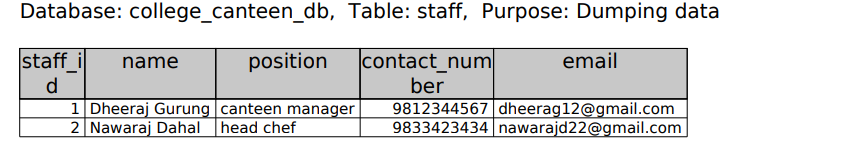




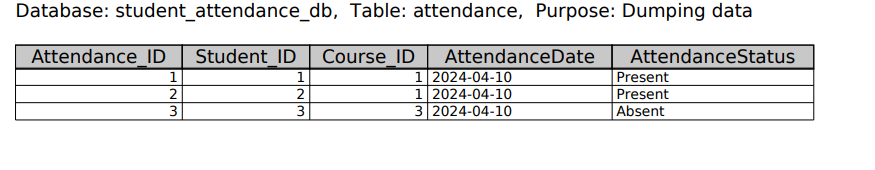


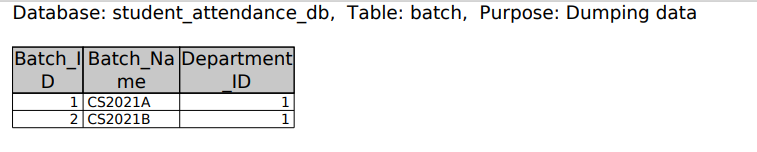


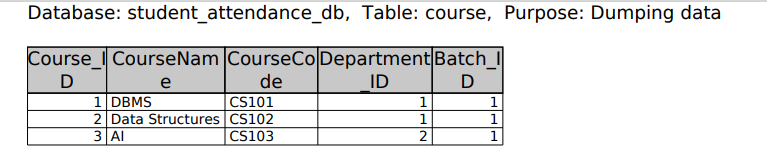


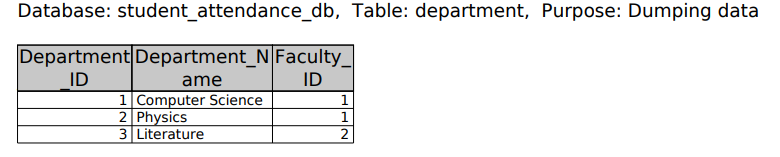


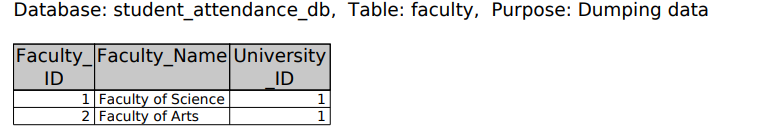
1. **Relational DB of ‘student\_attendance’**

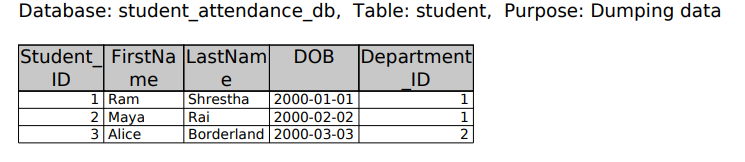


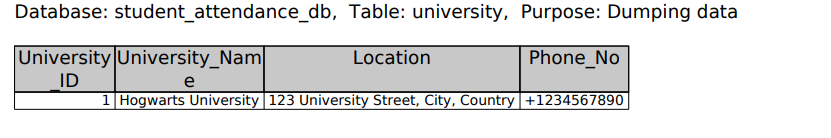












**Conclusion:**

Hence, we designed the relational database using phpmy admin in Lab.

**ASSIGNMENT 04:**

**Title:**

Basic SQL syntax

**Objective**:

To practice the basic SQL syntax

**Theory**:

* + 1. **CREATE DATABASE**

The **CREATE DATABASE** statement is used to create a new SQL database. The basic syntax of CREATE DATABASE statement is as follows:

CREATE DATABASE databasename;

**Code:**

CREATE DATABASE DBMS\_Lab;

* + 1. **DROP DATABASE**

The **DROP DATABASE** statement is used to drop an existing SQL database. The basic syntax of DROP DATABASE statement is as follows:

DROP DATABASE databasename;

**Code:**

DROP DATABASE TestDB;

* + 1. **USE DATABASE**

When we have multiple databases in our SQL Schema, then before starting our operation, we need to

select a database where all the operations would be performed. The **USE** statement is used to select an existing SQL database. The basic syntax of USE statement is as follows:

USE databasename;

**Code:**

USE DBMS\_Lab;

* + 1. **CREATE TABLE**

Creating a basic table involves naming and defining its columns and each column’s data type. The **CREATE TABLE** statement is used to create a new table in a database. The basic syntax of CREATE TABLE statement is as follows:

CREATE TABLE table\_name(

column1 datatype,

column2 datatype,

column3 datatype,

column4 datatype,

.......

columnN datatype,

PRIMARY KEY (one or more columns)

);

**Code:**

CREATE TABLE Customers (

cID int,

LastName varchar(50),

FirstName varchar(50),

Customers\_address varchar(100),

City varchar(40),

PRIMARY KEY(cID)

);

* + 1. **DROP TABLE**

The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table. The basic syntax of DROP TABLE statement is as follows:

DROP TABLE table\_name;

**Code:**

DROP TABLE Customers;

* + 1. **ALTER TABLE**

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. The ALTER TABLE statement is also used to add and drop various constraints on an existing table. The basic syntax of ALTER TABLE statement is as follows:

a) ALTER TABLE table\_name

ADD column\_name datatype;

b) ALTER TABLE table\_name

DROP COLUMN column\_name;

c) ALTER TABLE table\_name

ALTER COLUMN column\_name datatype;

**Code:**

ALTER TABLE Customers

ADD Sex varchar(20);

ALTER TABLE Customers

DROP COLUMN Customers\_address;

ALTER TABLE Customers

ALTER COLUMN City varchar(30);

* + 1. **INSERT INTO**

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database. The basic

syntax of INSERT INTO statement is as follows:

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

OR,

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

**Code:**

INSERT INTO Customers(cID,LastName,FirstName,City,Sex)

VALUES (1,'Shrestha','Aasish','Itahari','Male');

* + 1. **SELECT**

The **SELECT** statement is used to select data from a database. The data returned is stored in a result table, called the result-set. The basic syntax of SELECT statement is as follows:

SELECT column1, column2, ...  
FROM table\_name;

OR,

SELECT \* FROM table\_name;

**Code:**

SELECT \* FROM Customers;

* + 1. **UPDATE**

The **UPDATE** statement is used to modify the existing records in a table. You can use the WHERE clause

with the UPDATE query to update the selected rows, otherwise all the rows would be affected. The basic syntax of UPDATE statement is as follows:

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

**Code:**

UPDATE Customers SET

FirstName = 'Lazio', City = 'Dharan'

WHERE cID = 2;

* + 1. **DELETE**

The **DELETE** statement is used to delete existing records in a table. The basic syntax of DELETE statement is

as follows:

DELETE FROM table\_name WHERE condition;

**Code:**

DELETE FROM Customers

WHERE cID = 3;

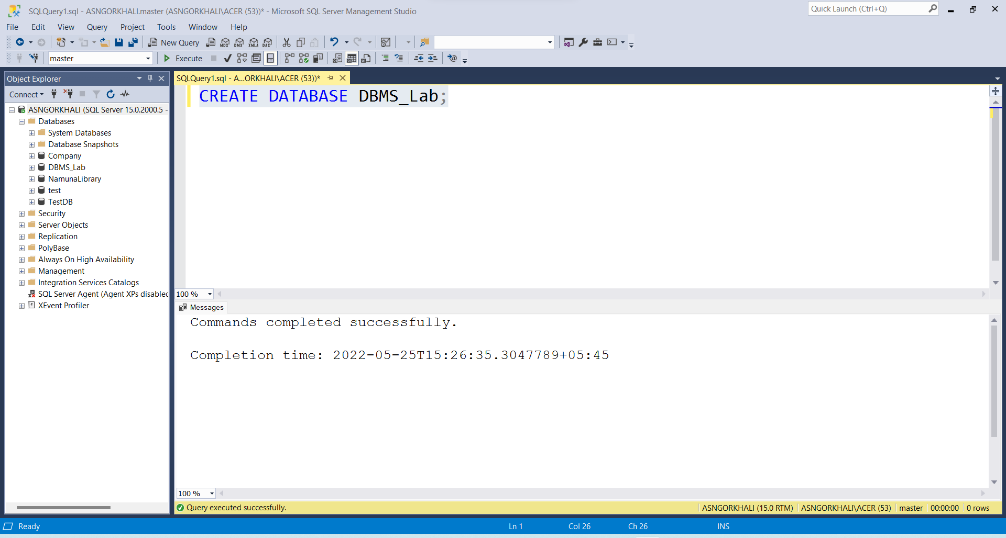
* + 1. **WHERE CLAUSE**

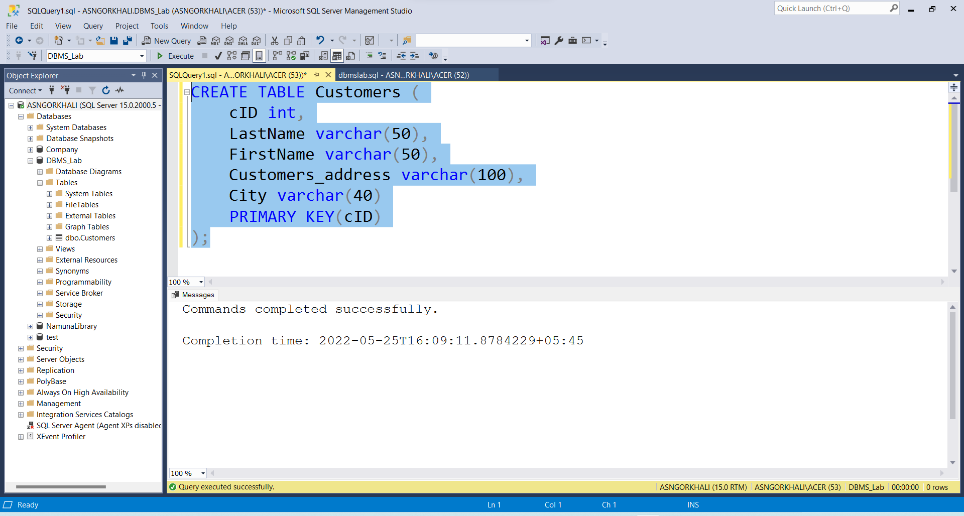
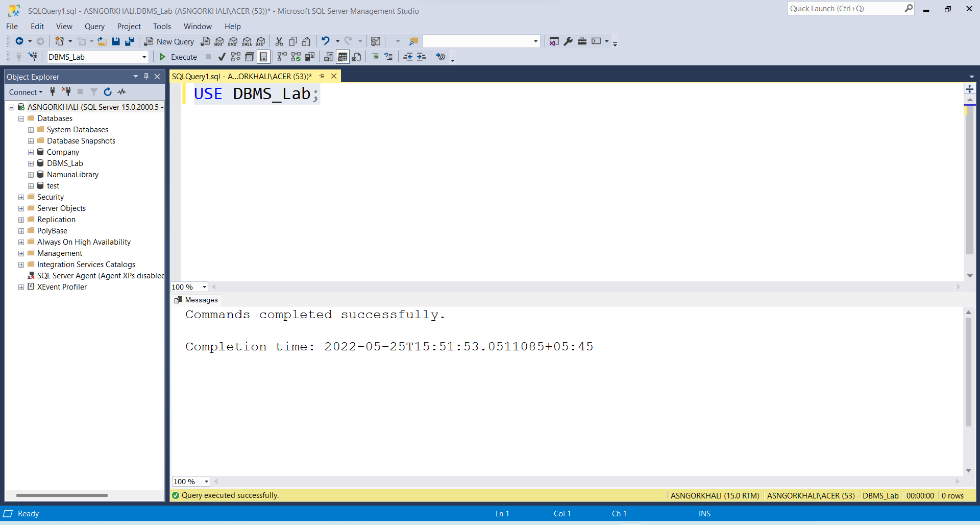
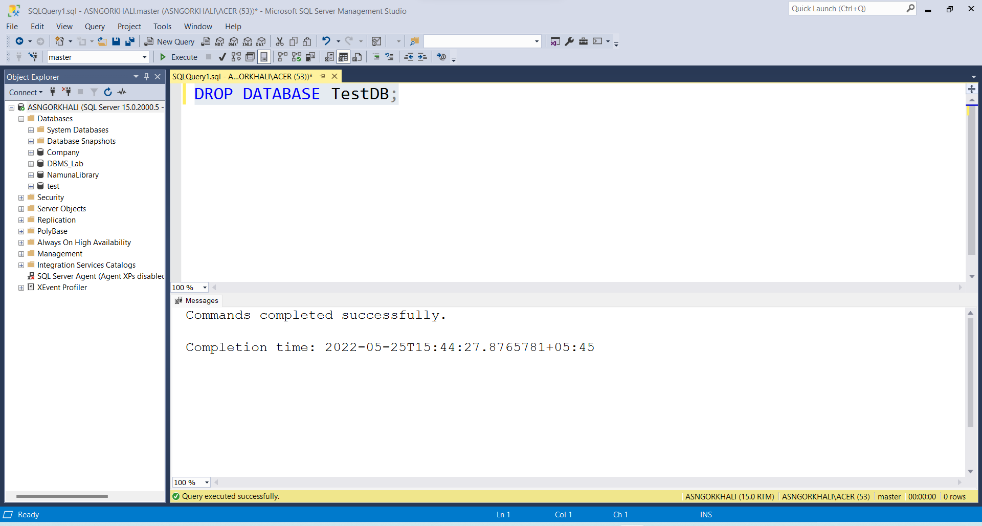
The **WHERE** clause is used to filter records. It is used to extract only those records that fulfill a specified condition. The WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE etc. You can specify a condition using the [comparison or logical operators](https://www.tutorialspoint.com/sql/sql-operators.htm) like >, <, =, **LIKE, NOT**, etc. The basic syntax of WHERE Clause is as follows:

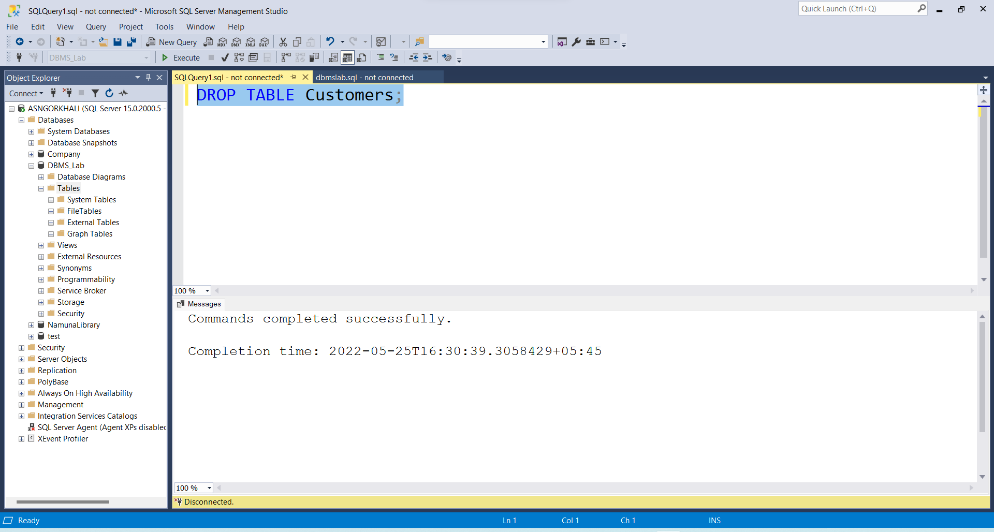
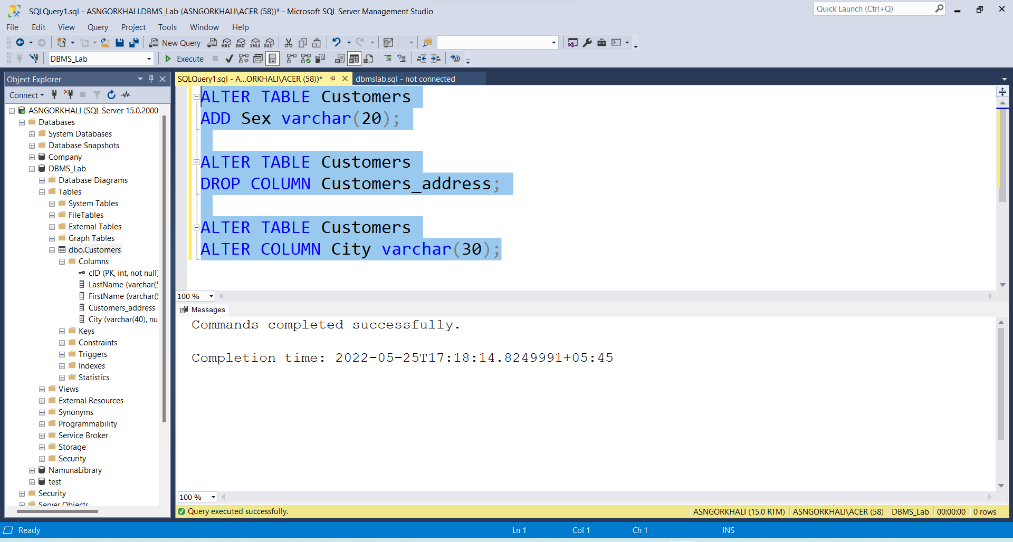
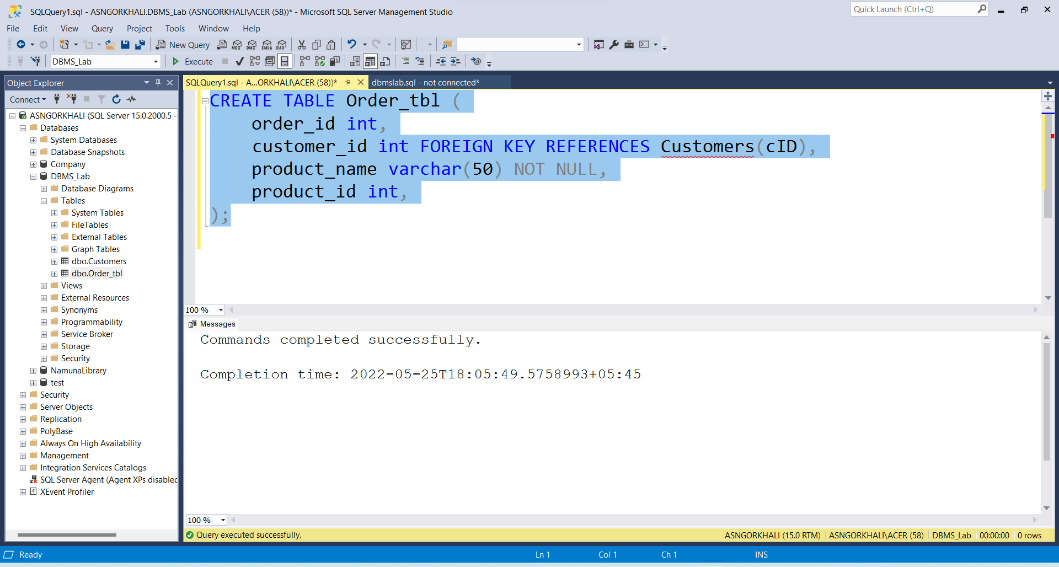
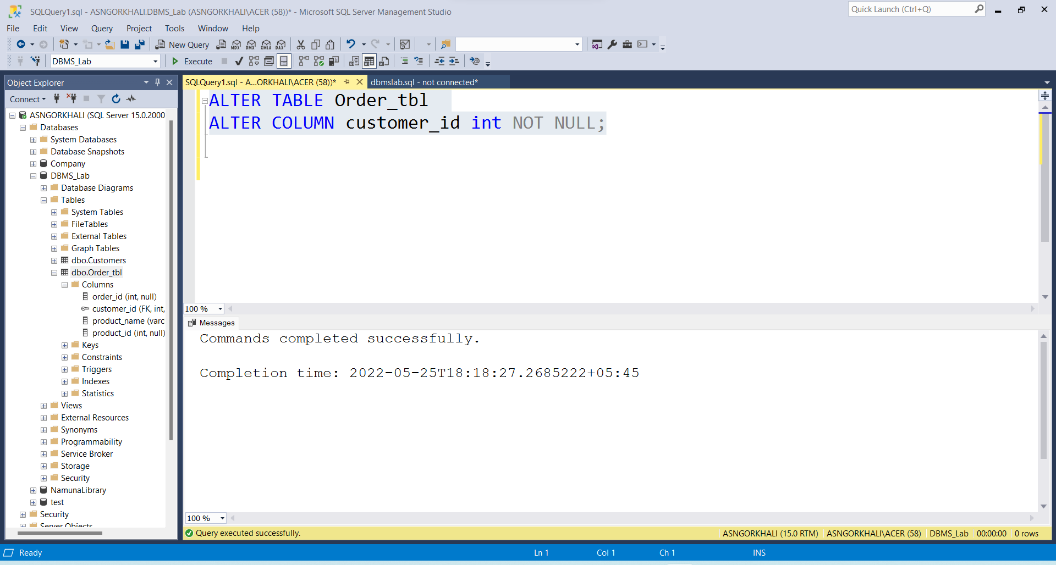
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

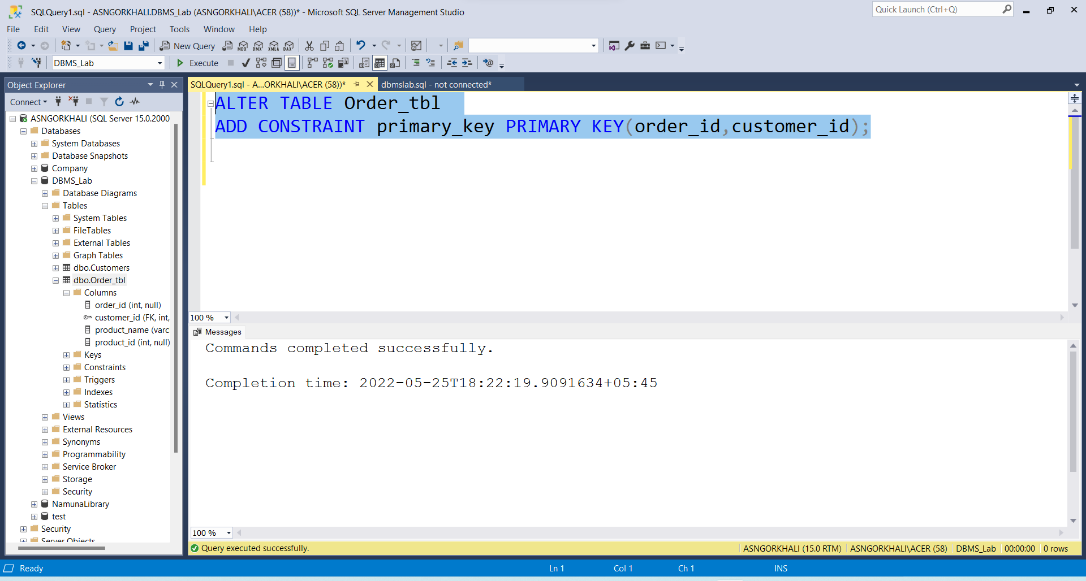
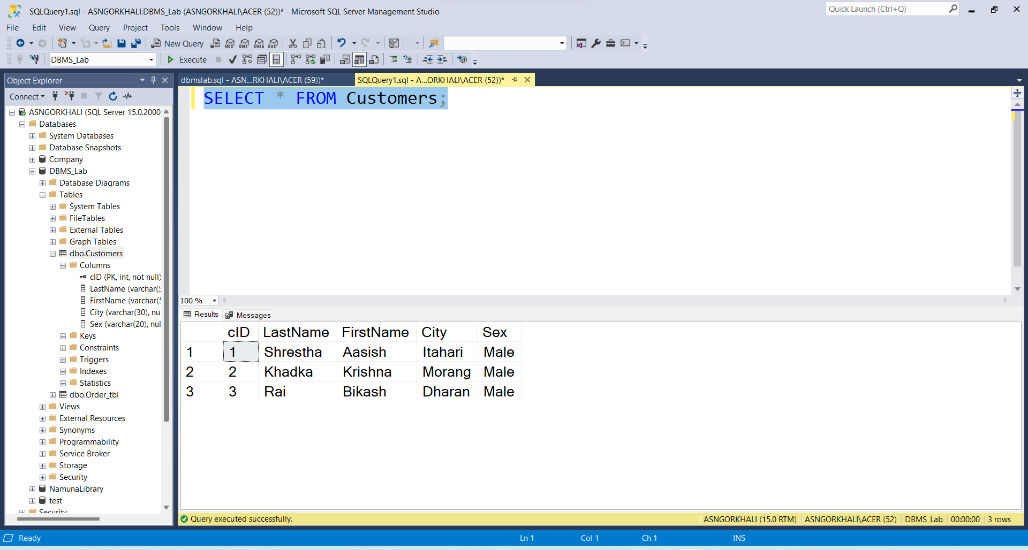
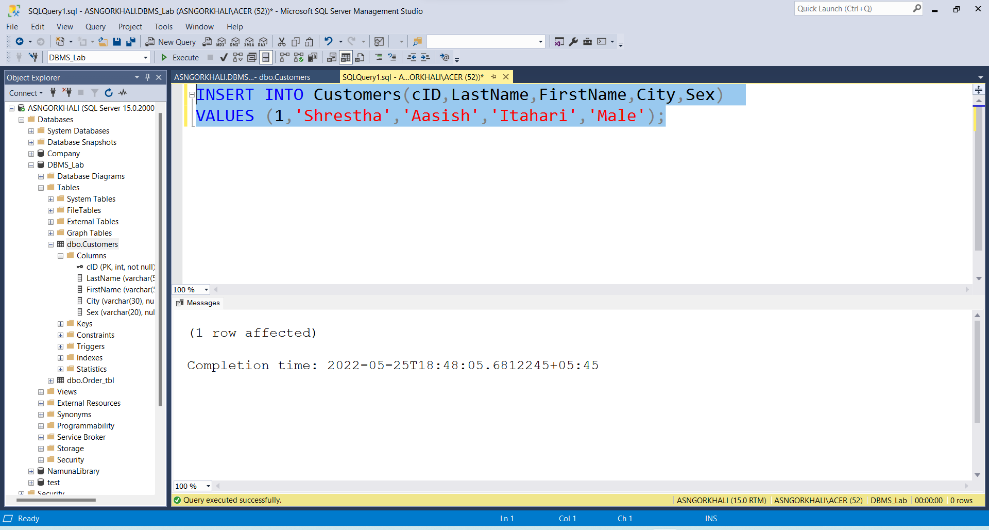
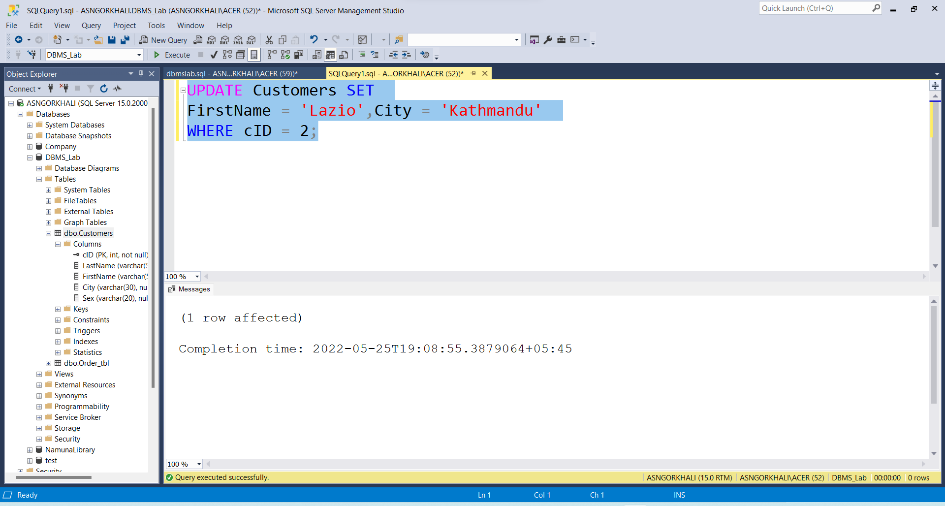
**Code:**

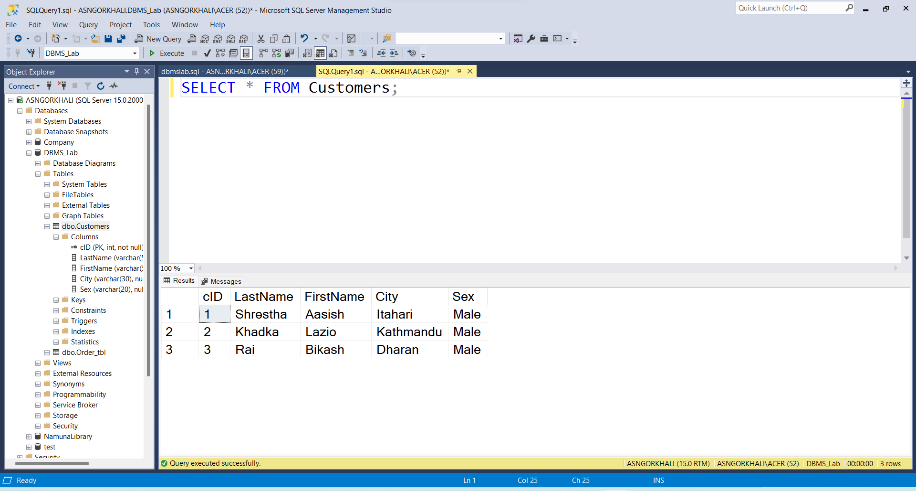
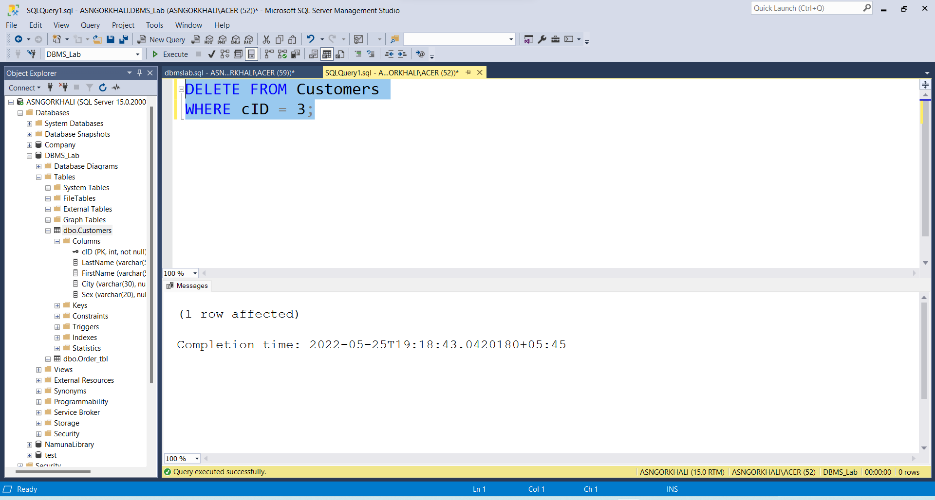
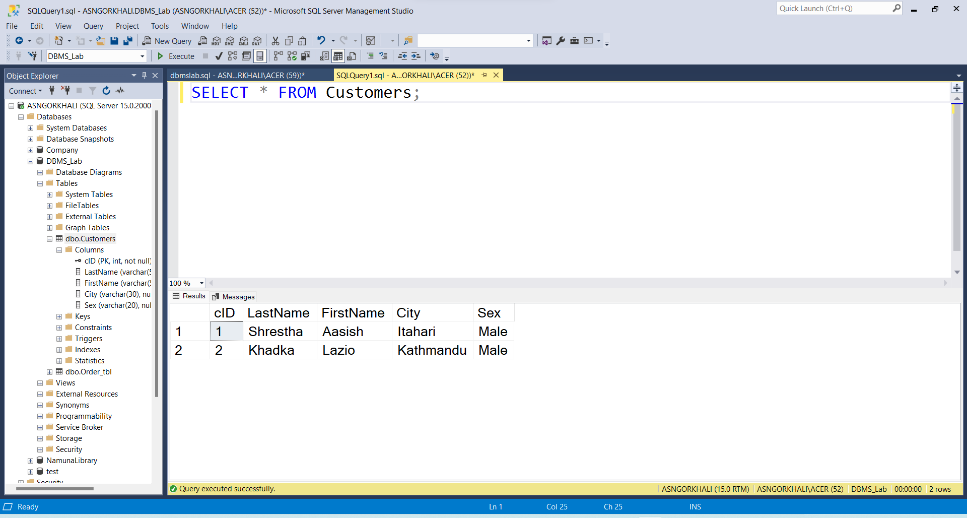
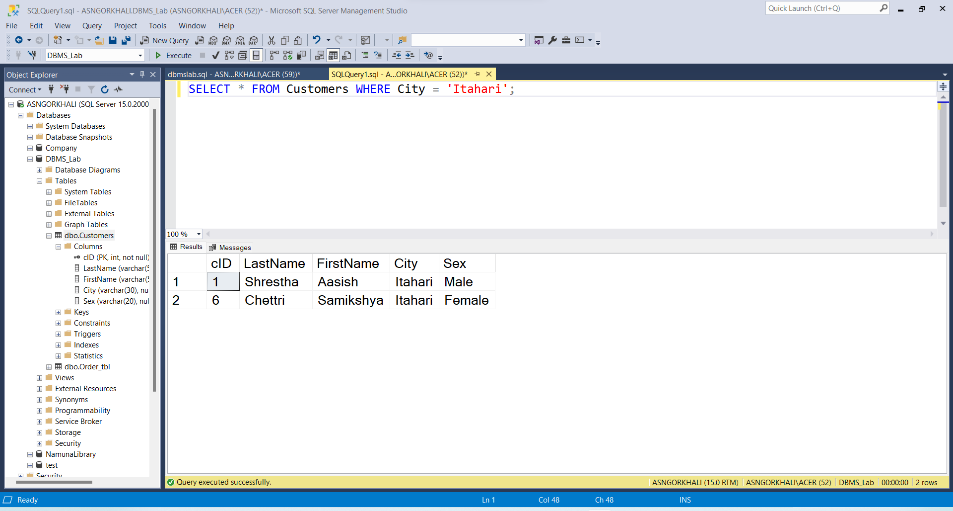
SELECT \* FROM Customers WHERE City = 'Itahari';

**Output:**









**Conclusion**:

Hence, we practiced and implemented the basic SQL syntax/queries in Lab.