**Database Management Systems**

First we install the xampp for php file.

Title page->[**ArtBase**](http://localhost:8080/ahritha/index.html)

1. **Solution:- Project Title: ArtBase Database Design**
2. **Conceptual Design:**

* **Entity Types:**

**Artist**: Represents individuals who create artwork.

**Artwork**: Represents individual pieces of art created by artists.

**Group**: Represents classifications or categories that artworks can belong to.

**Customer**: Represents individuals who purchase artwork from galleries.

* **Relationship Types:**

**Artists to Artworks:** A one-to-many relationship where one artist can create multiple artworks, but each artwork is created by only one artist.

**Artworks to Groups:** A many-to-many relationship since a single artwork can belong to multiple groups, and each group can include multiple artworks.

**Customers to Artists:** A many-to-many relationship if customers have preferences for multiple artists.

**Customers to Groups:** A many-to-many relationship if customers have preferences for multiple groups of art.

* **Data Dictionary:**

Artist: Contains details about artists.

Attributes: Name, Birthplace, Age, Style of Art.

Artwork: Contains details about artworks.

Attributes: Artist (FK), Year Made, Title, Type of Art, Price.

Group: Contains details about groups.

Attributes: Name.

Customer: Contains details about customers.

Attributes: Name, Address, Total Spent, Favored Artists (FK), Favored Groups (FK).

**Constraints and Assumptions:**

* Each artist has a unique name.
* Each artwork has a unique title.
* An artwork can belong to multiple groups.
* Customers can favor multiple artists and groups.

The total amount spent by a customer is a cumulative figure.

Conceptual Model:

Your conceptual model will include entities and relationships with cardinality and participation constraints. For example, an artist can have zero or many artworks, but each artwork must have one and only one artist.

**3)**  **Logical design**

Creating a data dictionary in T-SQL involves documenting the structure of your database, including tables, columns, data types, and any constraints or relationships between tables. Here’s a conceptual outline of how you can develop and update a data dictionary using T-SQL:

-- Data Dictionary for ArtBase Database

-- Table: Artists

CREATE TABLE Artists (

    ArtistID *INT* *PRIMARY KEY* IDENTITY(1,1),

    Name NVARCHAR(255) UNIQUE NOT NULL,

    Birthplace NVARCHAR(255) NOT NULL,

    Age *INT* NOT NULL,

    StyleOfArt NVARCHAR(255) NOT NULL

);

-- Table: Artworks

CREATE TABLE Artworks (

    ArtworkID *INT* *PRIMARY KEY* IDENTITY(1,1),

    ArtistID *INT* *FOREIGN KEY* *REFERENCES* Artists(ArtistID),

    YearMade *INT* NOT NULL,

    Title NVARCHAR(255) UNIQUE NOT NULL,

    TypeOfArt NVARCHAR(255) NOT NULL,

    Price *DECIMAL*(10, 2) NOT NULL

);

-- Table: Groups

CREATE TABLE Groups (

    GroupID *INT* *PRIMARY KEY* IDENTITY(1,1),

    Name NVARCHAR(255) UNIQUE NOT NULL

);

-- Table: Customers

CREATE TABLE Customers (

    CustomerID *INT* *PRIMARY KEY* IDENTITY(1,1),

    Name NVARCHAR(255) UNIQUE NOT NULL,

    Address NVARCHAR(255) NOT NULL,

    TotalSpent *DECIMAL*(10, 2) NOT NULL

);

-- Table: CustomerPreferences (Many-to-Many Relationship)

CREATE TABLE CustomerPreferences (

    CustomerID *INT* *FOREIGN KEY* *REFERENCES* Customers(CustomerID),

    ArtistID *INT* *FOREIGN KEY* *REFERENCES* Artists(ArtistID),

    GroupID *INT* *FOREIGN KEY* *REFERENCES* Groups(GroupID),

*PRIMARY KEY* (CustomerID, ArtistID, GroupID)

);

**List of Assumptions:**

* **Artists** are uniquely identified by their name.
* **Artworks** have a unique title and are associated with one artist.
* **Groups** are uniquely identified by their name and can include many artworks.
* **Customers** are uniquely identified by their name and can have preferences for multiple artists and groups.

**Updates:**

To update the data dictionary, you would add new tables or modify existing ones as needed.

To document changes, you could use T-SQL comments or extended properties.

For example, to add a description to the Artists table, you could use:

EXEC sp\_addextendedproperty

    @name = N'MS\_Description',

    @value = 'Table containing artist information.',

    @level0type = N'Schema', @level0name = 'dbo',

    @level1type = N'Table',  @level1name = 'Artists';

1. **Physical Database Design:**

Artists Table:

**Attributes:**

ArtistID: Integer (Primary Key, Auto-Increment)

Name: NVARCHAR(255) (Unique, Not Null)

Birthplace: NVARCHAR(255) (Not Null)

Age: Integer (Not Null)

StyleOfArt: NVARCHAR(255) (Not Null)

**Functional Dependencies:** ArtistID → {Name, Birthplace, Age, StyleOfArt}

**Constraints**: PRIMARY KEY (ArtistID), UNIQUE (Name)

**Normalization**: The table is in 3rd Normal Form as each non-key attribute is only dependent on the primary key.

Artworks Table:

**Attributes**:

ArtworkID: Integer (Primary Key, Auto-Increment)

ArtistID: Integer (Foreign Key)

YearMade: Integer (Not Null)

Title: NVARCHAR(255) (Unique, Not Null)

TypeOfArt: NVARCHAR(255) (Not Null)

Price: DECIMAL(10, 2) (Not Null)

**Functional Dependencies**: ArtworkID → {ArtistID, YearMade, Title, TypeOfArt, Price}

**Constraints**: PRIMARY KEY (ArtworkID), FOREIGN KEY (ArtistID) REFERENCES Artists(ArtistID), UNIQUE (Title)

**Normalization**: The table is in 3rd Normal Form as each non-key attribute is only dependent on the primary key.

Groups Table:

**Attributes**:

GroupID: Integer (Primary Key, Auto-Increment)

Name: NVARCHAR(255) (Unique, Not Null)

Functional Dependencies: GroupID → {Name}

**Constraints**: PRIMARY KEY (GroupID), UNIQUE (Name)

**Normalization**: The table is in 3rd Normal Form as each non-key attribute is only dependent on the primary key.

Customers Table:

**Attributes**:

CustomerID: Integer (Primary Key, Auto-Increment)

Name: NVARCHAR(255) (Unique, Not Null)

Address: NVARCHAR(255) (Not Null)

TotalSpent: DECIMAL(10, 2) (Not Null)

**Functional Dependencie**s: CustomerID → {Name, Address, TotalSpent}

**Constraints**: PRIMARY KEY (CustomerID), UNIQUE (Name)

**Normalization**: The table is in 3rd Normal Form as each non-key attribute is only dependent on the primary key.

CustomerPreferences Table:

**Attributes**:

CustomerID: Integer (Foreign Key)

ArtistID: Integer (Foreign Key)

GroupID: Integer (Foreign Key)

**Functional Dependencies**: {CustomerID, ArtistID, GroupID} → {}

**Constraints**: PRIMARY KEY (CustomerID, ArtistID, GroupID), FOREIGN KEY (CustomerID) REFERENCES **Customers**(CustomerID), FOREIGN KEY (ArtistID) REFERENCES Artists(ArtistID), FOREIGN KEY (GroupID) REFERENCES Groups(GroupID)

**Normalization**: This table is used to resolve the many-to-many relationships and is in 3rd Normal Form.

Normalization Decisions and Issues:

**Normalization**: All tables are designed to be in the 3rd Normal Form to eliminate redundant data and ensure data integrity.

**De-normalization**: Depending on query performance requirements, you might consider de-normalizing some of the data. For example, if there are frequent queries joining the Artists and Artworks tables, you might create a view or a denormalized table that pre-joins these tables for faster access.

Indexes: Consider adding indexes on foreign keys and columns that are frequently used in search criteria to improve query performance.

CREATE TABLE Artists (

    ArtistID *INT* *PRIMARY KEY* IDENTITY(1,1),

    Name NVARCHAR(255) UNIQUE NOT NULL,

    Birthplace NVARCHAR(255) NOT NULL,

    Age *INT* NOT NULL,

    StyleOfArt NVARCHAR(255) NOT NULL

);

**5) Basic SQL Operations:**

*<?php* *include* "connection.php";

$id = $\_GET['ArtistID'];

$stmt=$conn->prepare("DELETE from artists where ArtistID = ?");

$stmt->bind\_param("i",$id);

*if*($stmt->execute()){

    header("Location: ../notification\_list.php");

    $stmt->close();

    $conn->close();

}

Select:

SELECT a.Name, aw.Title, aw.TypeOfArt, aw.Price

FROM Artists a

JOIN Artworks aw ON a.ArtistID = aw.ArtistID

**Insert:-**

-- Add a new artist

INSERT INTO Artists (Name, Birthplace, Age, StyleOfArt)

VALUES ('Jane Doe', 'New York', 40, 'Contemporary');

**Update:-**

 UPDATE Artworks

SET Price = 1500.00

WHERE ArtworkID = 1;

**Delete:-**

DELETE FROM Customers

WHERE CustomerID = 1;

**Joining Tables :-**

 SELECT c.Name AS CustomerName, a.Name AS ArtistName

FROM Customers c

JOIN CustomerPreferences cp ON c.CustomerID = cp.CustomerID

JOIN Artists a ON cp.ArtistID = a.ArtistID;

**Stored Procedure:-**

CREATE PROCEDURE AddNewArtwork

    @ArtistID *INT*,

    @YearMade *INT*,

    @Title NVARCHAR(255),

    @TypeOfArt NVARCHAR(255),

    @Price *DECIMAL*(10, 2)

AS

BEGIN

    INSERT INTO Artworks (ArtistID, YearMade, Title, TypeOfArt, Price)

    VALUES (@ArtistID, @YearMade, @Title, @TypeOfArt, @Price);

END;

**To execute the stored procedure:**

EXEC AddNewArtwork 1, 2024, 'Sunset Over Mountains', 'Painting', 2000.00;

**Trigger:**

**Trigger to Update TotalSpent for a Customer After a Purchase:**

CREATE TRIGGER UpdateTotalSpent

ON Purchases

AFTER INSERT

AS

BEGIN

    DECLARE @CustomerID *INT*, @Price *DECIMAL*(10, 2);

    SELECT @CustomerID = i.CustomerID, @Price = i.Price

    FROM inserted i;

    UPDATE Customers

    SET TotalSpent = TotalSpent + @Price

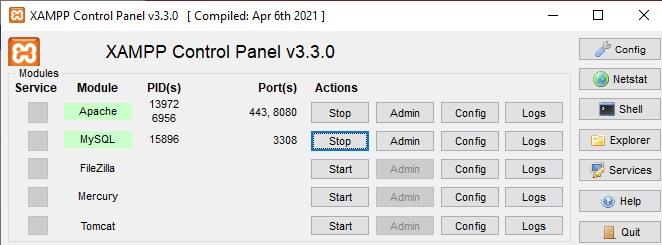
    WHERE CustomerID = @CustomerID;

END;

**6. Application (optional) –**

**Firstly, We download and install the Xampp. We have Installed php to use it.**

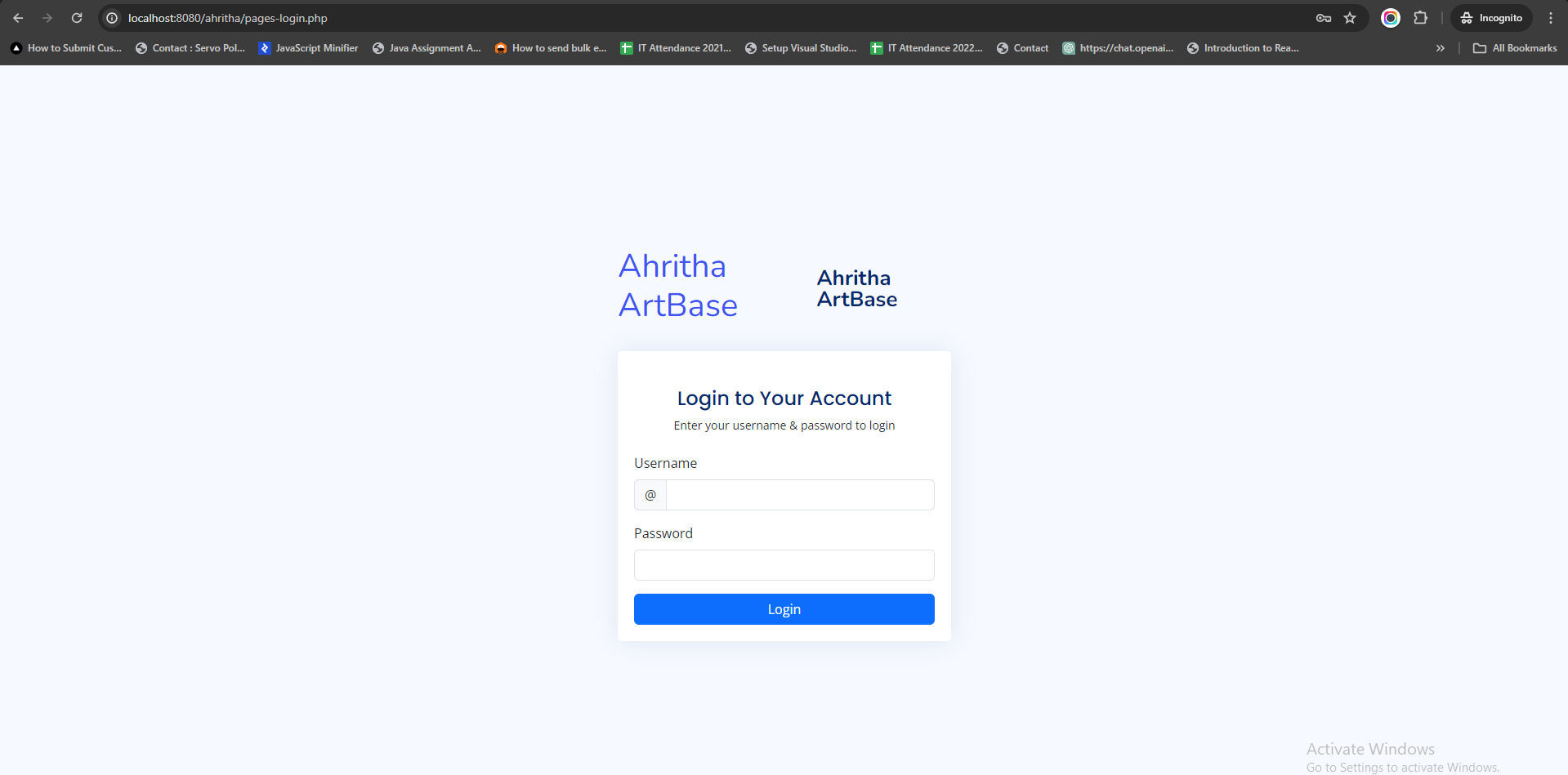
**We start the xampp**



**Enter the Localhost in the the browser.**

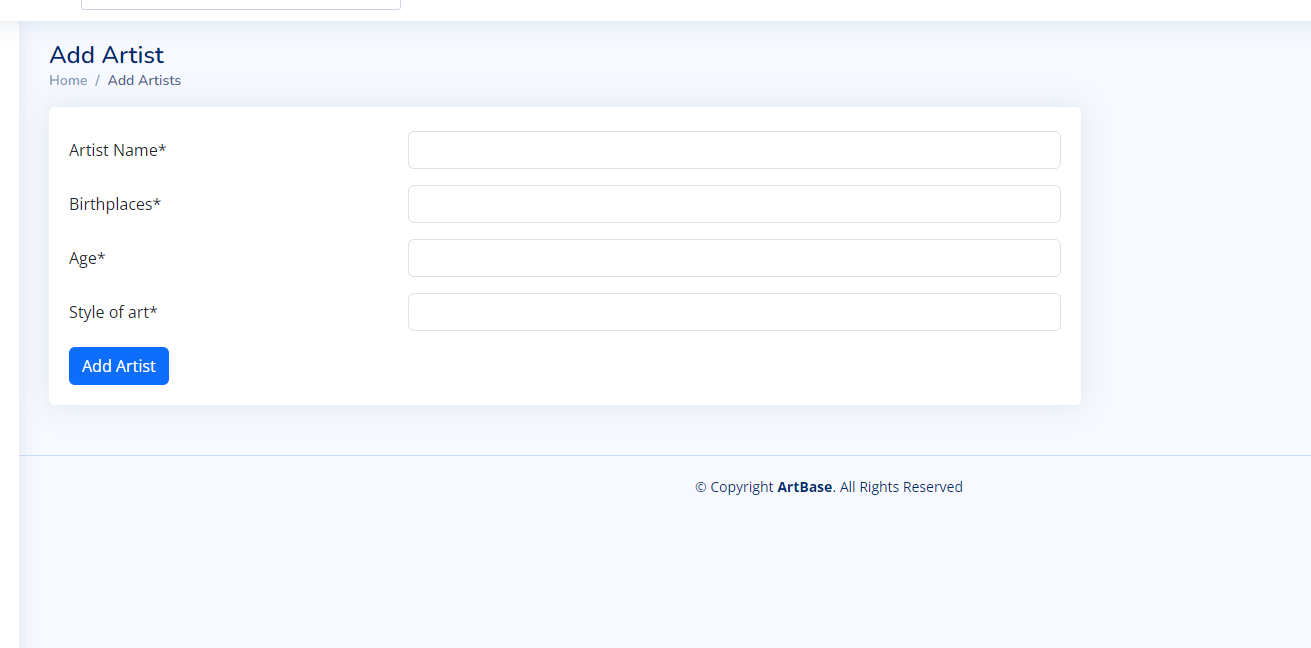
[**http://localhost/ahritha/**](http://localhost/ahritha/)





**Username🡪 art@123**

**Password🡪123456**

**We Make the Artist Form**

When we click on the **add Artist** button

It will be insert in the table artists .

Firstly we create the table -- Table: Artists

CREATE TABLE artists (

    ArtistID *INT* *PRIMARY KEY* IDENTITY(1,1),

    Name NVARCHAR(255) UNIQUE NOT NULL,

    Birthplace NVARCHAR(255) NOT NULL,

    Age *INT* NOT NULL,

    StyleOfArt NVARCHAR(255) NOT NULL

);

This is my form

 <form *method*="post" *class*="mt-4" *enctype*="multipart/form-data">

                                    <div *class*="row">

                                        <div *class*="col-sm-12">

                                            <div *class*="row mb-3"> <label *for*="inputNumber" *class*="col-sm-4 col-form-label">Artist Name<span *class*="red">\*</span></label>

                                                <div *class*="col-sm-8"> <input *type*="text" *name*="a\_name" *class*="form-control" *required*></div>

                                            </div>

                                        </div>

                                    </div>

                                    <div *class*="row">

                                        <div *class*="col-sm-12">

                                            <div *class*="row mb-3"> <label *for*="inputNumber" *class*="col-sm-4 col-form-label">Birthplaces<span *class*="red">\*</span></label>

                                                <div *class*="col-sm-8"> <input *type*="text" *name*="Birthplaces" *class*="form-control" *required*></div>

                                            </div>

                                        </div>

                                    </div>

                                    <div *class*="row">

                                        <div *class*="col-sm-12">

                                            <div *class*="row mb-3"> <label *for*="inputNumber" *class*="col-sm-4 col-form-label">Age<span *class*="red">\*</span></label>

                                                <div *class*="col-sm-8"> <input *type*="number" *name*="Age" *class*="form-control" *required*></div>

                                            </div>

                                        </div>

                                    </div>

                                    <div *class*="row">

                                        <div *class*="col-sm-12">

                                            <div *class*="row mb-3"> <label *for*="inputNumber" *class*="col-sm-4 col-form-label">Style of art<span *class*="red">\*</span></label>

                                                <div *class*="col-sm-8"> <input *type*="text" *name*="style\_art" *class*="form-control" *required*></div>

                                            </div>

                                        </div>

                                    </div>

                                    <!-- <div class="row">

                                        <div class="col-sm-12">

                                            <div class="row mb-3"> <label for="inputText"

                                                    class="col-sm-4 col-form-label">Product Image<span

                                                        class="red">\*</span></label>

                                                <div class="col-sm-8"> <input type="file" name="ssi\_image"

                                                        class="form-control" required></div>

                                            </div>

                                        </div>

                                    </div> -->

                                    <div>

                                        <button *type*="submit" *name*="saveuser" *class*="btn btn-primary">Add

                                            Artist</button>

                                    </div>

                                </form>

**We create the table and insert the data**

*if* ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    $Name = $\_POST["a\_name"];

    $Birthplace = $\_POST["Birthplaces"];

    $Age = $\_POST["Age"];

    $StyleOfArt= $\_POST["style\_art"];

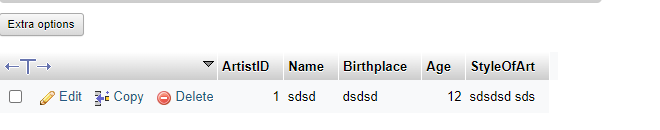
    $stmt = $conn->prepare("INSERT INTO artists(Name,Birthplace,Age,StyleOfArt) Values(?,?,?,?)");

    $stmt->bind\_param("ssis", $Name, $Birthplace,$Age,$StyleOfArt);

    $stmt->execute();

    $stmt->close();

}



**We can show the data from the artists table shown in the screen shot.**

**We select the table artist with using the selct query.**

*<?php* *include* "includes/header.php";

*include* "includes/connection.php";

*include* "includes/session.php";

$stmt = $conn->prepare("SELECT \* from artists");

$stmt->execute();

$result = $stmt->get\_result();

*?>*

 <tbody>

*<?php* $count = 1;

*while* ($getData = $result->fetch\_assoc()) { *?>*

            <tr>

              <td>

*<?=* $count++; *?>*

              </td>

              <td>

*<?=* $getData['Name']; *?>*

              </td>

              <td>

*<?=* $getData['Birthplace']; *?>*

              </td>

              <td>

*<?=* $getData['Age']; *?>*

              </td>

              <td>

*<?=* $getData['StyleOfArt']; *?>*

              </td>

              <td><a *href*="editNotification.php?id=*<?=* $getData['id']; *?>*"><i

*class*="bi bi-pen btn btn-outline-primary"></i></a>

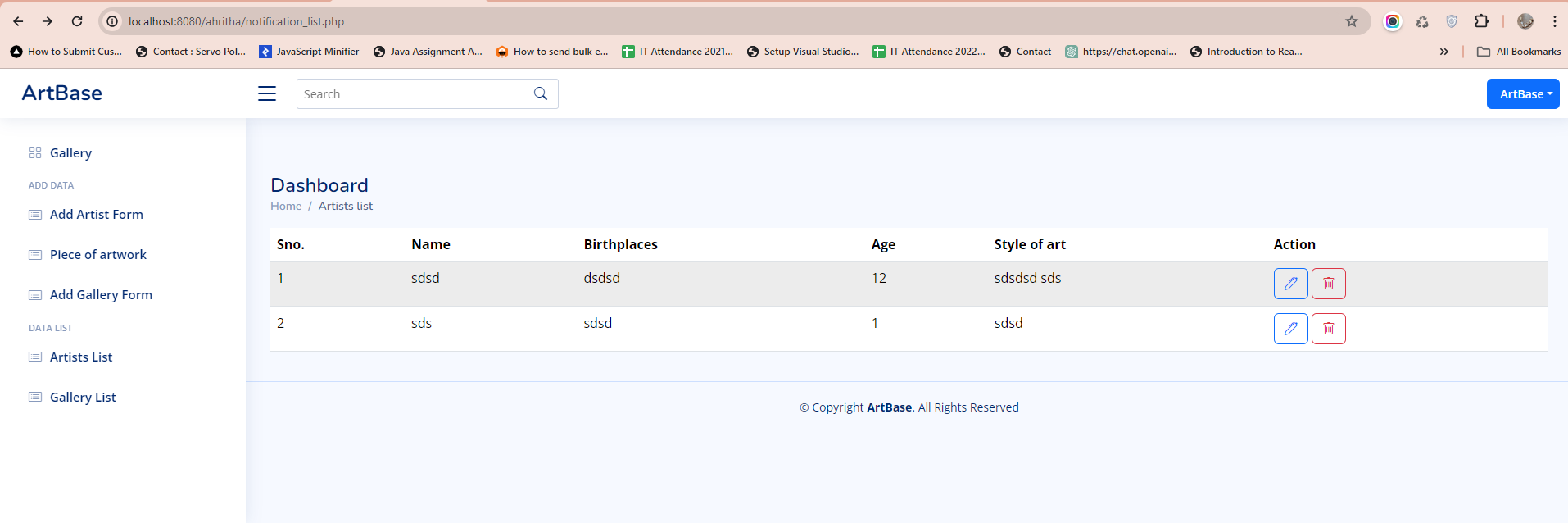
                <a *href*="includes/deleteNotification.php?id=*<?=* $getData['id']; *?>*"><i

*class*="bi bi-trash btn btn-outline-danger"></i></a>

              </td>

            </tr>

*<?php* } *?>*



**We use the Update query**

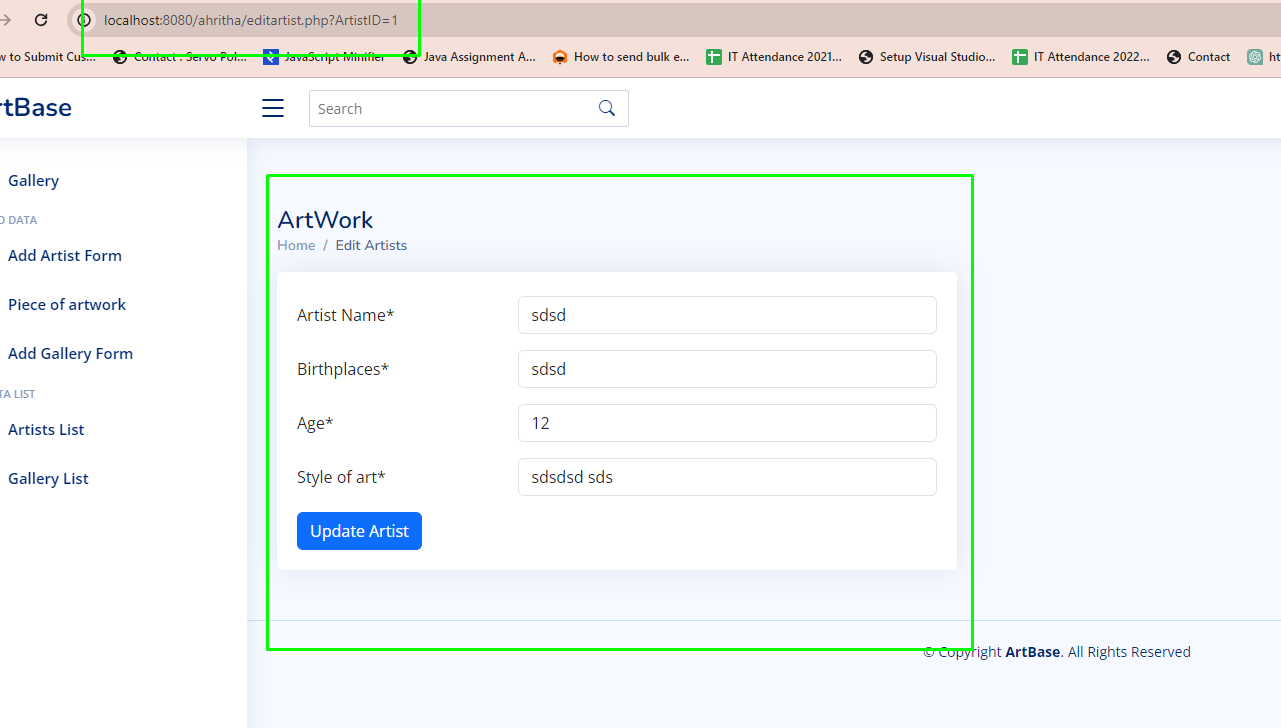
When we click on the **Edit** button

a *href*="editartist.php?ArtistID=*<?=* $getData['ArtistID']; *?>*">

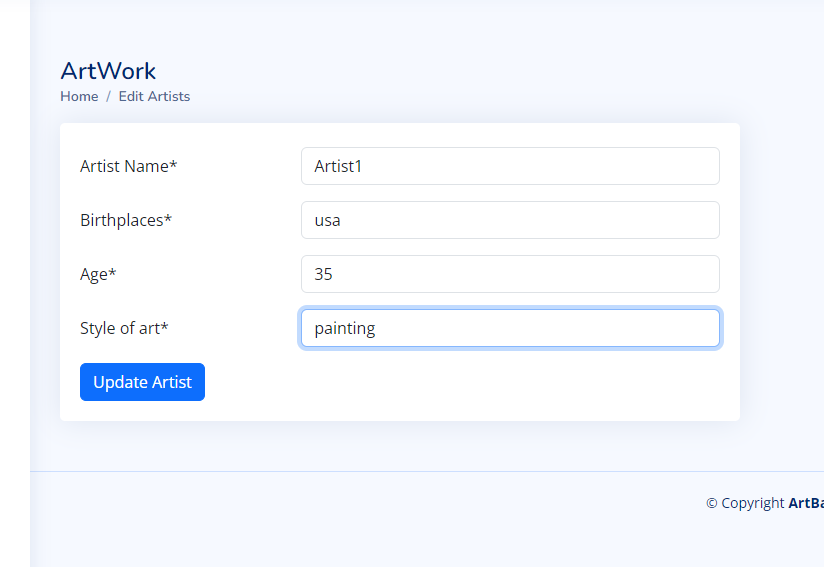
              <i

*class*="bi bi-pen btn btn-outline-primary"></i>

                  </a>



Update form will be Open.

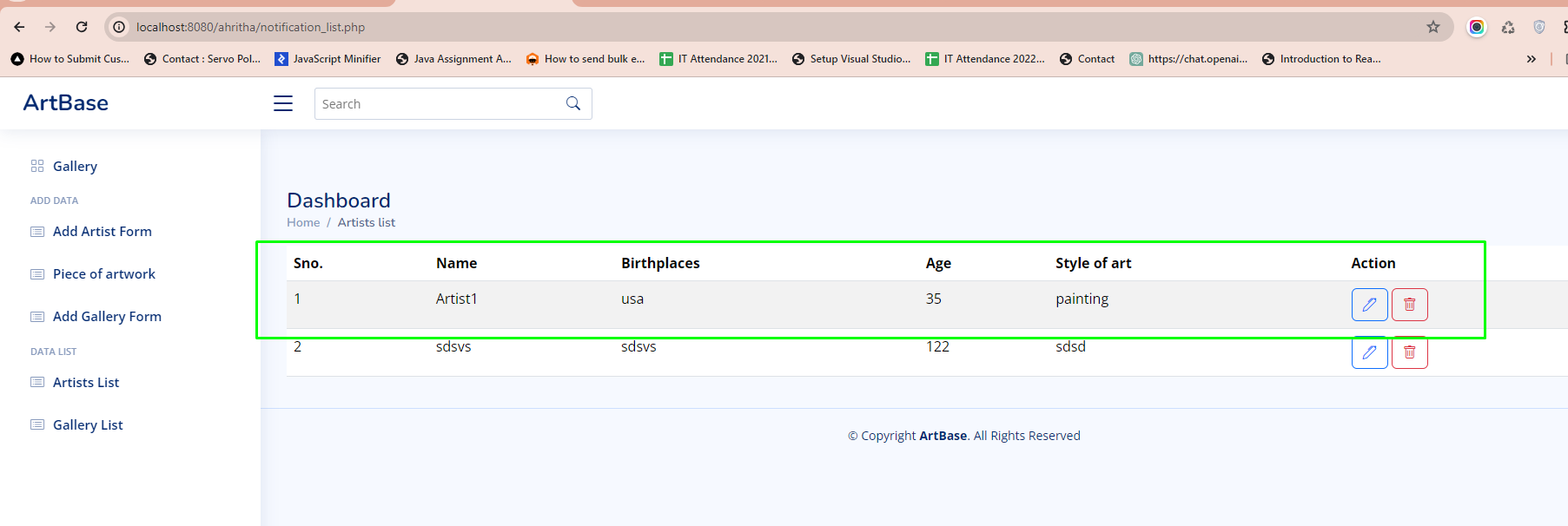
We Enter the new data 

SQL We can see in the database.

SQL statements correctly establish the structure for the ArtBase Database, including the many-to-many relationship in the CustomerPreferences table.

We use the update command

UPDATE artists SET Name = ?, Birthplace = ?, Age = ?, StyleOfArt = ? WHERE ArtistID = ?

We can see 

When we click on the Delete Button

That time delete query

DELETE from artists where id = ?

Here’s a brief explanation of the key components you’ve used:

* **PRIMARY KEY**: This constraint uniquely identifies each record in a database table. Primary keys must contain unique values, and they cannot contain NULL values. A table can have only one primary key, which may consist of single or multiple columns.
* **FOREIGN KEY:** A foreign key is a column or a group of columns in a table that links to the primary key of another table. The foreign key ensures referential integrity by enforcing a link between the data in two tables.
* **IDENTITY(1,1):** This clause is used for auto-incrementing the value of the column. The first 1 indicates that the numbering starts at 1, and the second 1 indicates that the increment is by 1.
* **UNIQUE:** This constraint ensures that all values in a column are different.
* **NOT NULL:** This constraint dictates that a column cannot store NULL values.
* **NVARCHAR(255**): This data type is used for variable-length Unicode character data. The maximum length is defined in parentheses.
* **DECIMAL(10, 2):** This data type stores fixed precision and scale numbers. For example, DECIMAL(10, 2) can store numbers with up to 10 digits, two of which can be after the decimal point.

6). 6. Application (optional) –

Retrieve all artists’ names and their styles of art.

SQL

SELECT Name, StyleOfArt FROM Artists;

SELECT Name, StyleOfArt FROM Artists;

SQL

SELECT \* FROM Artworks WHERE YearMade = 2020;

List all customers with a total spent greater than $10,000.

SQL

SELECT Name FROM Customers WHERE TotalSpent > 10000;

Find all artworks with a price between $500 and $2000.

SQL

SELECT Title, Price FROM Artworks WHERE Price BETWEEN 500 AND 2000;

Show the names of groups that have ‘Modern’ in their name.

SQL

SELECT Name FROM Groups WHERE Name LIKE '%Modern%';

Advanced SQL Queries

Retrieve all customers over 30 years old (assuming you have a BirthYear column in your Customers table).

SQL

SELECT Name FROM Customers WHERE YEAR(GETDATE()) - BirthYear > 30;

List all artists and the number of artworks they have made.

SQL

SELECT a.Name, COUNT(aw.ArtworkID) AS NumberOfArtworks

FROM Artists a

JOIN Artworks aw ON a.ArtistID = aw.ArtistID

GROUP BY a.Name;

Find the total value of artworks sold by each artist.

SQL

SELECT a.Name, SUM(aw.Price) AS TotalValue

FROM Artists a

JOIN Artworks aw ON a.ArtistID = aw.ArtistID

GROUP BY a.Name;

Show customers and their preferred styles of art (assuming there’s a StyleOfArt column in CustomerPreferences).

SQL

SELECT c.Name, cp.StyleOfArt

FROM Customers c

JOIN CustomerPreferences cp ON c.CustomerID = cp.CustomerID;

Retrieve the most expensive artwork sold for each type of art.

SQL

SELECT TypeOfArt, MAX(Price) AS HighestPrice

FROM Artworks

GROUP BY TypeOfArt;