

ASSIGNMENT (III)

**Art Gallery Management System**

**Question 1 :**

**RELATIONAL SCHEMA**

**CREATE TABLE Users** (

UserID INT **PRIMARY KEY,**

UserEmail **VARCHAR(255) UNIQUE,**

Username **VARCHAR(100) UNIQUE,**

Password **VARCHAR(255),**

Role **VARCHAR(100),**

FirstName **VARCHAR(100),**

LastName **VARCHAR(100),**

Address **VARCHAR(255),**

PhoneNo **VARCHAR(20) UNIQUE,**

)

**CREATE TABLE Artist** (

AID **INT,**

UserID **INT,**

SocialMediaProfile **VARCHAR(255),**

SignatureStyle **VARCHAR(255),**

CONSTRAINT PKARTIST **PRIMARY KEY** (AID),

CONSTRAINT FKARTUSER **FOREIGN KEY** (UserID) **REFERENCES** Users(UserID)

)

**CREATE TABLE Artwork (**

ArtID INT **NOT NULL**,

AID INT **NOT NULL**,

Title **VARCHAR(255),**

Medium **VARCHAR(100),**

Dimensions **VARCHAR(100),**

DateCompleted **DATE,**

Availability **VARCHAR(20),**

Price **DECIMAL(10, 2),**

CONSTRAINT PKARTWORK **PRIMARY KEY** (ArtID),

CONSTRAINT FKART\_ARTIST **FOREIGN KEY** (AID) **REFERENCES** Artist (AID)

)

**CREATE TABLE** **Gallery** (

GID **INT,**

GalleryName **VARCHAR(255),**

Status **VARCHAR(50),**

Location **VARCHAR(255),**

CONSTRAINT PKGALLERY **PRIMARY KEY** (GID)

)

**CREATE TABLE Exhibition** (

EID **INT,**

GID **INT,**

StartDate **DATE,**

EndDate **DATE,**

Location **VARCHAR(255),**

EName **VARCHAR(255),**

Theme **VARCHAR(255),**

ArtID **INT,**

CONSTRAINT PKEXHIBITION **PRIMARY KEY** (EID),

CONSTRAINT FKEXHI\_GALLERY **FOREIGN KEY** (GID) **REFERENCES** Gallery(GID),

CONSTRAINT FKEXHI\_ART **FOREIGN KEY** (ArtID) **REFERENCES** Artwork(ArtID)

)

**CREATE TABLE Sales** (

SaleID **INT,**

UserID **INT,**

ArtID **INT,**

Price **DECIMAL(10, 2),**

SaleDate **DATE,**

ArtistCommission **DECIMAL(10, 2),**

CONSTRAINT PKSALES **PRIMARY KEY** (SaleID),

CONSTRAINT FKSALESUSER **FOREIGN KEY** (UserID) **REFERENCES** Users(UserID),

CONSTRAINT FKSALESART **FOREIGN KEY** (ArtID) **REFERENCES** Artwork(ArtID),

**CHECK** (Price > 0)

)

**CREATE TABLE Loan** (

LoanID **INT,**

UserID **INT**,

ArtID **INT,**

LoanAmount **DECIMAL(10, 2),**

DurationInDays **INT,**

ExpiryDate **DATE,**

TransportTracking **VARCHAR(100),**

MonitoringStatus **VARCHAR(50),**

CONSTRAINT PKLOAN **PRIMARY KEY** (LOANID),

CONSTRAINT FKLOANUSER **FOREIGN KEY** (UserID) **REFERENCES** Users(UserID),

CONSTRAINT FKLOANART **FOREIGN KEY** (ArtID) **REFERENCES** Artwork(ArtID),

)

**Question 2:**

**PYTHON CODE FOR DATA GENERATION**

**USERS:**

import random  
import pyodbc as odbc  
  
# Define customized list of user attributes  
user\_ids = list(range(1, 102)) # 100 User IDs from 1 to 100  
emails = [  
 f"user{i}@example.com" for i in range(1, 101)  
]  
  
usernames = [  
 f"user{i}" for i in range(1, 101)  
]  
  
passwords = [  
 f"Password{i}!" for i in range(1, 101)  
]  
  
roles = ["Admin", "Staff", "Client", "Artist"]  
  
first\_names = [  
 "John", "Alice", "Bob", "Carol", "David", "Emma", "Frank", "Grace", "Henry", "Ivy",  
 "Jack", "Linda", "Michael", "Susan", "Peter", "Sophia", "Robert", "Jennifer", "William", "Jessica",  
 "Daniel", "Mary", "Christopher", "Karen", "Matthew", "Nancy", "Joshua", "Lisa", "Andrew", "Samantha",  
 "James", "Karen", "Joseph", "Emily", "Ryan", "Nicole", "Richard", "Michelle", "David", "Angela",  
 "Charles", "Kimberly", "Thomas", "Melissa", "Steven", "Patricia", "Donald", "Amy", "Mark", "Laura",  
 "Paul", "Rebecca", "Kevin", "Stephanie", "George", "Elizabeth", "Brian", "Julie", "Edward", "Heather",  
 "Matthew", "Sarah", "Adam", "Nathan", "Chris", "Lauren", "Brandon", "Amber", "Patrick", "Danielle",  
 "Olivia", "Sophia", "Emma", "Isabella", "Ava", "Mia", "Evelyn", "Harper", "Camila", "Gianna",  
 "Abigail", "Luna", "Ella", "Elizabeth", "Sofia", "Emily", "Avery", "Mila", "Scarlett", "Aria",  
 "Chloe", "Layla", "Amelia", "Hannah", "Lily", "Zoey", "Riley", "Nora", "Liam", "Mason"  
]  
  
last\_names = [  
 "Smith", "Johnson", "Williams", "Jones", "Brown", "Davis", "Miller", "Wilson", "Moore", "Taylor",  
 "Anderson", "Thomas", "Jackson", "White", "Harris", "Martin", "Thompson", "Garcia", "Martinez", "Robinson",  
 "Clark", "Rodriguez", "Lewis", "Lee", "Walker", "Hall", "Allen", "Young", "Hernandez", "King",  
 "Wright", "Lopez", "Hill", "Scott", "Green", "Adams", "Baker", "Gonzalez", "Nelson", "Carter",  
 "Mitchell", "Perez", "Roberts", "Turner", "Phillips", "Campbell", "Parker", "Evans", "Edwards", "Collins",  
 "Stewart", "Sanchez", "Morris", "Rogers", "Reed", "Cook", "Morgan", "Bell", "Murphy", "Bailey",  
 "Sullivan", "Price", "Perry", "Powell", "Russell", "Reed", "Watson", "Brooks", "Sanders", "Wood",  
 "Ward", "Torres", "Peterson", "Gray", "Ramirez", "James", "Watson", "Brooks", "Kelly", "Sanders",  
 "Price", "Bennett", "Wood", "Barnes", "Ross", "Henderson", "Coleman", "Jenkins", "Perry", "Powell",  
 "Long", "Patterson", "Hughes", "Flores", "Washington", "Butler", "Simmons", "Foster", "Gonzales", "Bryant"  
]  
  
addresses = [  
 f"{i} Maple Street" for i in range(1, 101)  
]  
  
phone\_numbers = [  
 f"123-456-78{str(i).zfill(2)}" for i in range(1, 101)  
]  
  
# Randomly select 30 unique users from the list  
selected\_indices = random.sample(range(100), 100)  
selected\_users = [(user\_ids[i], emails[i], usernames[i], passwords[i], first\_names[i], last\_names[i], addresses[i], phone\_numbers[i]) for i in selected\_indices]  
  
# Ensure there are exactly 2 Admin roles, 8 Staff, 10 Clients, and 10 Artists  
roles\_list = ["Admin"] \* 10 + ["Staff"] \* 20 + ["Client"] \* 45 + ["Artist"] \* 25  
random.shuffle(roles\_list)  
  
# Combine selected\_users with roles  
selected\_users\_with\_roles = [(user[0], user[1], user[2], user[3], roles\_list[i], user[4], user[5], user[6], user[7]) for i, user in enumerate(selected\_users)]  
  
# Sort users by user\_id for deterministic order  
selected\_users\_with\_roles.sort(key=lambda user: user[0])  
  
# Database connection details  
SERVER\_NAME = r'DESKTOP-8J774MH\SQLEXPRESS'  
DATABASE\_NAME = 'ArtGalleryManagementSystem'  
  
# Connection string  
conn\_str = (  
 f'DRIVER={{ODBC Driver 17 for SQL Server}};'  
 f'SERVER={SERVER\_NAME};'  
 f'DATABASE={DATABASE\_NAME};'  
 r'Trusted\_Connection=yes;'  
)  
  
# Establish connection  
conn = odbc.connect(conn\_str)  
cursor = conn.cursor()  
  
# Create the Users table if it doesn't exist  
cursor.execute('''  
 IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Users')  
 CREATE TABLE Users  
 ( UserID INT PRIMARY KEY,  
 UserEmail VARCHAR(255) UNIQUE,  
 Username VARCHAR(100) UNIQUE,  
 Password VARCHAR(255),  
 Role VARCHAR(100),  
 FirstName VARCHAR(100),  
 LastName VARCHAR(100),  
 Address VARCHAR(255),  
 PhoneNo VARCHAR(20) UNIQUE)  
 ''')  
  
# Insert records into the Users table  
insert\_query = '''  
INSERT INTO Users (UserID, UserEmail, Username, Password, Role, FirstName, LastName, Address, PhoneNo)  
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)  
'''  
  
for user in selected\_users\_with\_roles:  
 cursor.execute(insert\_query, user)  
 print("Inserted Successfully")  
  
# Commit the transaction  
conn.commit()  
  
# Close the cursor and connection  
cursor.close()  
conn.close()

**ARTIST:**

import random  
import pyodbc as odbc  
  
# Define customized lists for Artist attributes  
  
social\_media\_profiles = [  
 "@theartones", "@creativebrush", "@artbyali", "@masterpieces", "@urbanartist", "@canvascreations",  
 "@abstractvisions", "@portraitpioneer", "@modernmuse", "@classiccreations", "@studioexpressions",  
 "@dynamicdesigns", "@visualvoyager", "@artisthub", "@creativecorner", "@artisticadventures",  
 "@inspiredillustrations", "@timelessart", "@artisticjourney", "@fineartfocus", "@artsyvibes",  
 "@colorfulcreations", "@artfusion", "@avantgardeartist", "@modernartmovement", "@artisticendeavors",  
 "@vibrantvisions", "@expressiveart", "@urbanexpressions", "@creativecanvas", "@artisticflair",  
 "@soulfulsketches", "@artisticimpressions", "@uniqueartistry", "@visualmastery", "@artfulconcepts",  
 "@boldbrushstrokes", "@artisticnarratives", "@artisticfusion", "@modernartist", "@artgal"  
 "@modernartist", "@inspiredcreations", "@abstractexpressions", "@creativevisions", "@colorfulcreations",  
 "@urbanartistry", "@timelesscreations", "@modernmasterpieces", "@dynamicart", "@classiccreations",  
 "@studioexpressions", "@fineartfocus", "@artisthub", "@artisticadventures", "@artisticjourney",  
 "@artisticnarratives", "@creativecanvas", "@visualvoyager", "@artisticflair", "@uniqueartistry"  
]  
  
signature\_styles = [  
 "impressionist", "classical", "abstract", "realism", "cubism", "surrealism", "expressionism", "pop art",  
 "minimalism", "modernism", "post-impressionism", "art nouveau", "baroque", "rococo", "fauvism", "dada",  
 "constructivism", "futurism", "conceptual art", "contemporary", "graffiti", "street art", "photorealism",  
 "neo-expressionism", "hyperrealism", "symbolism", "mannerism", "romanticism", "abstract expressionism",  
 "art deco", "renaissance", "op art", "kinetic art", "outsider art", "naïve art", "land art", "installation art",  
 "performance art", "digital art", "video art", "sound art", "interactive art", "cyber art", "media art",  
 "process art", "happenings", "fluxus", "assemblage", "readymade", "collage", "mixed media", "arte povera",  
 "brutalism", "lowbrow", "psychedelic art", "urban art", "post-modernism", "neo-dada"  
]  
  
# Database connection details  
SERVER\_NAME = r'DESKTOP-8J774MH\SQLEXPRESS'  
DATABASE\_NAME = 'ArtGalleryManagementSystem'  
  
# Connection string  
conn\_str = (  
 f'DRIVER={{ODBC Driver 17 for SQL Server}};'  
 f'SERVER={SERVER\_NAME};'  
 f'DATABASE={DATABASE\_NAME};'  
 r'Trusted\_Connection=yes;'  
)  
  
# Establish connection  
conn = odbc.connect(conn\_str)  
cursor = conn.cursor()  
  
# Create Artist table if it doesn't exist  
cursor.execute('''  
 IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Artist')  
 CREATE TABLE Artist  
 ( AID INT,  
 UserID INT,  
 SocialMediaProfile VARCHAR(255),  
 SignatureStyle VARCHAR(255),  
 CONSTRAINT PKARTIST PRIMARY KEY (AID),  
 CONSTRAINT FKARTUSER FOREIGN KEY (UserID) REFERENCES Users(UserID))  
 ''')  
  
# Retrieve UserIDs where Role is 'Artist'  
cursor.execute('SELECT UserID FROM Users WHERE Role = ?', 'Artist')  
artist\_user\_ids = [row[0] for row in cursor.fetchall()]  
  
# Randomly select attributes for artists  
num\_artists = min(len(artist\_user\_ids), 100) # Ensure we don't exceed the number of user IDs  
selected\_user\_ids = random.sample(artist\_user\_ids, num\_artists)  
  
# Generate unique AID values  
existing\_aids = set()  
artist\_records = []  
for user\_id in selected\_user\_ids:  
 aid = random.randint(301, 400) # Adjust the range to avoid conflicts with existing user IDs  
 while aid in existing\_aids:  
 aid = random.randint(301, 400)  
 existing\_aids.add(aid)  
 artist\_records.append((  
 aid,  
 user\_id,  
 random.choice(social\_media\_profiles),  
 random.choice(signature\_styles)  
 ))  
  
# Insert records into the Artist table  
for record in artist\_records:  
 cursor.execute('''  
 INSERT INTO Artist (AID, UserID, SocialMediaProfile, SignatureStyle)  
 VALUES (?, ?, ?, ?)  
 ''', record)  
 print("Inserted Successfully")  
  
# Commit the transaction  
conn.commit()  
  
# Close the cursor and connection  
cursor.close()  
conn.close()  
  
print("Total artist records generated:", len(artist\_records))

**ARTWORK:**

import random  
from pyodbc import connect  
from datetime import datetime  
  
# Define lists for Artwork attributes  
art\_ids = list(range(1, 101))  
titles = [  
 "The Starry Night", "The Last Supper", "Mona Lisa", "The Persistence of Memory",  
 "Girl with a Pearl Earring", "The Scream", "Guernica", "The Night Watch",  
 "The Kiss", "The Birth of Venus", "American Gothic", "Water Lilies",  
 "The Garden of Earthly Delights", "The Arnolfini Portrait", "The Hay Wain", "The Fighting Temeraire",  
 "Liberty Leading the People", "The Raft of the Medusa", "Las Meninas", "The Son of Man",  
 "The Sleeping Gypsy", "The Ambassadors", "No. 5, 1948", "Wanderer above the Sea of Fog",  
 "The Great Wave off Kanagawa", "The Third of May 1808", "Starry Night Over the Rhone", "Olympia",  
 "Impression, Sunrise", "Bal du moulin de la Galette", "The Storm on the Sea of Galilee",  
 "The Card Players", "The Old Guitarist", "A Sunday Afternoon on the Island of La Grande Jatte",  
 "The Dance Class", "The Arnolfini Portrait", "The Sleeping Gypsy", "The Ambassadors",  
 "The Night Watch", "The Garden of Earthly Delights", "The Kiss", "American Gothic",  
 "Water Lilies", "The Persistence of Memory", "Girl with a Pearl Earring", "The Scream",  
 "Guernica", "The Last Supper", "The Starry Night", "The Birth of Venus"  
]  
mediums = [  
 "oil", "acrylic", "watercolor", "ink", "charcoal", "pastel", "tempera",  
 "gouache", "fresco", "encaustic", "digital"  
]  
dimensions = [  
 "1280x720", "1920x1080", "3840x2160", "800x600", "1024x768", "1600x900", "2560x1440", "1280x960",  
 "1366x768", "1440x900", "1680x1050"  
]  
dates\_completed = [  
 "2023-01-15", "2023-02-20", "2023-03-18", "2023-04-22", "2023-05-05", "2023-06-11", "2023-07-19",  
 "2023-08-24", "2023-09-30", "2023-10-14", "2023-11-23", "2023-12-02"  
]  
prices = [  
 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000, 130000,  
 140000, 150000, 160000, 170000, 180000, 190000, 200000, 210000, 220000, 230000, 240000,  
 250000, 260000, 270000, 280000, 290000, 300000, 310000, 320000, 330000, 340000, 350000,  
 360000, 370000, 380000, 390000, 400000, 410000, 420000, 430000, 440000, 450000, 460000,  
 470000, 480000, 490000, 500000  
]  
availability = ["Available", "Not for Sale", "Sold"]  
  
# Database connection details  
SERVER\_NAME = r'DESKTOP-8J774MH\SQLEXPRESS'  
DATABASE\_NAME = 'ArtGalleryManagementSystem'  
  
# Connection string  
conn\_str = (  
 f'DRIVER={{ODBC Driver 17 for SQL Server}};'  
 f'SERVER={SERVER\_NAME};'  
 f'DATABASE={DATABASE\_NAME};'  
 r'Trusted\_Connection=yes;'  
)  
  
# Establish connection  
conn = connect(conn\_str)  
cursor = conn.cursor()  
  
# Create Artwork table if it doesn't exist  
cursor.execute('''  
 IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Artwork')  
 CREATE TABLE Artwork  
 (  
 ArtID INT NOT NULL,  
 AID INT NOT NULL,  
 Title VARCHAR(255),  
 Medium VARCHAR(100),  
 Dimensions VARCHAR(100),  
 DateCompleted DATE,  
 Availability VARCHAR(20),  
 Price DECIMAL(10, 2),  
 CONSTRAINT PKARTWORK PRIMARY KEY (ArtID),  
 CONSTRAINT FKART\_ARTIST FOREIGN KEY (AID) REFERENCES Artist (AID)  
  
 )  
 ''')  
  
# Fetch A\_IDs from Artist table  
cursor.execute('SELECT AID FROM Artist')  
artist\_ids = [row[0] for row in cursor.fetchall()]  
  
# Randomly select 100 values from the customized lists  
artwork\_records = list(zip(  
 random.sample(art\_ids, 100),  
 random.choices(artist\_ids, k=100),  
 random.sample(titles, 50),  
 random.choices(mediums, k=100),  
 random.choices(dimensions, k=100),  
 random.choices(dates\_completed, k=100),  
 random.choices(prices, k=100),  
 random.choices(availability, k=100)  
))  
  
# Insert records into the Artwork table  
for record in artwork\_records:  
 # Ensure that the price is set to None if availability is 'Not for Sale'  
 if record[7] == 'Not for Sale':  
 price = None  
 else:  
 price = record[6]  
  
 cursor.execute('''  
 INSERT INTO Artwork (ArtID, AID, Title, Medium, Dimensions, DateCompleted, Availability, Price)  
 VALUES (?, ?, ?, ?, ?, ?, ?, ?)  
 ''',  
 record[0], record[1], record[2], record[3], record[4], record[5], record[7], price)  
 print("Inserted Successfully")  
  
# Commit the transaction  
conn.commit()  
  
# Close the cursor and connection  
cursor.close()  
conn.close()

**GALLERY:**

import random  
from pyodbc import connect  
  
# Define lists for Gallery attributes  
g\_ids = list(range(1, 101))  
gallery\_names = [  
 "Artful Ancestry", "Skarstedt Fine Art", "Modern Masterpieces", "Creative Expressions",  
 "Heritage Gallery", "Visionary Art"  
]  
statuses = ["Open", "Closed"]  
locations = [  
 "Block C, Cornellia St.", "Avenue 5, Blueberry Hill", "Sector 10, Riverside", "Lot 23, Maple Street"  
]  
  
# Database connection details  
SERVER\_NAME = r'DESKTOP-8J774MH\SQLEXPRESS'  
DATABASE\_NAME = 'ArtGalleryManagementSystem'  
  
# Connection string  
conn\_str = (  
 f'DRIVER={{ODBC Driver 17 for SQL Server}};'  
 f'SERVER={SERVER\_NAME};'  
 f'DATABASE={DATABASE\_NAME};'  
 r'Trusted\_Connection=yes;'  
)  
  
# Establish connection  
conn = connect(conn\_str)  
cursor = conn.cursor()  
  
# Create Gallery table if it doesn't exist  
cursor.execute('''  
 IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Gallery')  
 CREATE TABLE Gallery  
 (  
 GID INT,  
 GalleryName VARCHAR(255),  
 Status VARCHAR(50),  
 Location VARCHAR(255),  
 CONSTRAINT PKGALLERY PRIMARY KEY (GID)  
 )  
 ''')  
  
# Randomly select 2 values from the customized lists  
selected\_gallery\_names = random.sample(gallery\_names, 2)  
selected\_statuses = random.sample(statuses, 2)  
selected\_locations = random.sample(locations, 2)  
  
# Generate 60 records by cycling through the selected values  
gallery\_records = list(zip(  
 random.sample(g\_ids, 2),  
 [selected\_gallery\_names[i % 2] for i in range(5)],  
 [selected\_statuses[i % 2] for i in range(5)],  
 [selected\_locations[i % 2] for i in range(5)]  
))  
  
# Insert records into the Gallery table  
for record in gallery\_records:  
 cursor.execute('''  
 INSERT INTO Gallery (GID, GalleryName, Status, Location)  
 VALUES (?, ?, ?, ?)  
 ''',  
 record[0], record[1], record[2], record[3])  
 print("Inserted Successfully")  
  
# Commit the transaction  
conn.commit()  
  
# Close the cursor and connection  
cursor.close()  
conn.close()

**EXHIBITION:**

import random  
from pyodbc import connect  
from datetime import datetime, timedelta  
  
# Define lists for Exhibition attributes  
e\_ids = list(range(1, 121))  
start\_dates = [  
 "2023-01-15", "2023-02-20", "2023-03-18", "2023-04-22", "2023-05-05", "2023-06-11", "2023-07-19",  
 "2023-08-24", "2023-09-30", "2023-10-14", "2023-11-23", "2023-12-02"  
]  
end\_dates = []  
for start\_date in start\_dates:  
 start\_datetime = datetime.strptime(start\_date, "%Y-%m-%d")  
 max\_end\_date = start\_datetime + timedelta(days=5)  
 end\_datetime = start\_datetime + timedelta(days=random.randint(1, 5))  
 # Ensure that the end date is within 5 days after the start date  
 if end\_datetime > max\_end\_date:  
 end\_datetime = max\_end\_date  
 end\_dates.append(end\_datetime.strftime("%Y-%m-%d"))  
  
locations = [  
 "Hyde Park, London", "Louvre, Paris", "Museum of Modern Art, New York", "Tate Modern, London",  
 "The Met, New York", "Uffizi Gallery, Florence"  
]  
e\_names = [  
 "Solstice", "Artistry Showcase", "Creative Expression Expo", "Visual Delights Exhibition",  
 "Palette Perfection Showcase", "Imagination Unleashed Expo", "Cultural Fusion Exhibition", "Canvas Creations Showcase", "Eclectic Artistry Expo",  
 "Vibrant Visions Exhibition", "Artisanal Craftsmanship Showcase", "Gallery Galore Expo", "Modern Masterpieces Exhibition",  
 "Abstract Adventures Showcase", "Surreal Splendor Expo", "Contemporary Perspectives Exhibition",  
 "Artful Endeavors Showcase", "Fusion of Colors Expo", "Urban Artistry Exhibition", "Ephemeral Elegance Showcase",  
 "Timeless Treasures Expo", "Harmony in Hues Exhibition", "Nature's Canvas Showcase", "Captivating Contrasts Expo",  
 "Whimsical Wonders Exhibition", "Ethereal Essence Showcase", "Serene Symmetry Expo", "Dynamic Dimensions Exhibition",  
 "Radiant Realms Showcase", "Enigmatic Expressions Expo", "Transcendent Textures Exhibition",  
]  
themes = [  
 "Nature and Landscapes",  
 "Portraits and Figures",  
 "Abstract Expressionism",  
 "Surrealism",  
 "Still Life",  
 "Urban Scenes",  
 "Mythology and Folklore",  
 "Fantasy and Science Fiction",  
 "Impressionism",  
 "Realism",  
 "Wildlife and Animals",  
 "Historical Events",  
 "Religious and Spiritual",  
 "Social and Political Commentary",  
 "Romance and Love",  
 "Dreams and Imagination",  
 "Symbolism",  
 "Contemporary Issues",  
 "Pop Culture",  
 "Environmental Conservation",  
 "Cultural Diversity",  
 "Technological Advancement",  
 "Human Emotions and Psychology",  
 "Architectural Wonders",  
 "Celestial Bodies and Space Exploration",  
 "Mysticism and Esotericism",  
 "Nostalgia and Memory",  
 "Identity and Self-Discovery",  
 "Abstract Concepts and Ideas",  
 "Seasons and Weather"  
]  
  
# Database connection details  
SERVER\_NAME = r'DESKTOP-8J774MH\SQLEXPRESS'  
DATABASE\_NAME = 'ArtGalleryManagementSystem'  
  
# Connection string  
conn\_str = (  
 f'DRIVER={{ODBC Driver 17 for SQL Server}};'  
 f'SERVER={SERVER\_NAME};'  
 f'DATABASE={DATABASE\_NAME};'  
 r'Trusted\_Connection=yes;'  
)  
  
# Establish connection  
conn = connect(conn\_str)  
cursor = conn.cursor()  
  
# Create Exhibition table if it doesn't exist  
cursor.execute('''  
 IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Exhibition')  
 CREATE TABLE Exhibition  
 ( EID INT,  
 GID INT,  
 StartDate DATE,  
 EndDate DATE,  
 Location VARCHAR(255),  
 EName VARCHAR(255),  
 Theme VARCHAR(255),  
 ArtID INT,  
 CONSTRAINT PKEXHIBITION PRIMARY KEY (EID),  
 CONSTRAINT FKEXHI\_GALLERY FOREIGN KEY (GID) REFERENCES Gallery(GID),  
 CONSTRAINT FKEXHI\_ART FOREIGN KEY (ArtID) REFERENCES Artwork(ArtID)  
  
 )  
 ''')  
  
# Fetch ArtID values from Artwork table  
cursor.execute('SELECT ArtID FROM Artwork')  
art\_ids = [row[0] for row in cursor.fetchall()]  
  
# Fetch GID values from Gallery table  
cursor.execute('SELECT GID FROM Gallery')  
gallery\_ids = [row[0] for row in cursor.fetchall()]  
  
# Randomly select 120 values from the customized lists  
exhibition\_records = list(zip(  
 random.sample(e\_ids, 120),  
 random.choices(gallery\_ids, k=120),  
 random.choices(start\_dates, k=120),  
 random.choices(end\_dates, k=120),  
 random.choices(locations, k=120),  
 random.choices(e\_names, k=120),  
 random.choices(themes, k=120),  
 random.choices(art\_ids, k=120)  
))  
  
# Insert records into the Exhibition table  
for record in exhibition\_records:  
 cursor.execute('''  
 INSERT INTO Exhibition (EID, GID, StartDate, EndDate, Location, EName, Theme, ArtID)  
 VALUES (?, ?, ?, ?, ?, ?, ?, ?)  
 ''',  
 record[0], record[1], record[2], record[3], record[4], record[5], record[6], record[7])  
 print("Inserted Successfully")  
  
# Commit the transaction  
conn.commit()  
  
# Close the cursor and connection  
cursor.close()  
conn.close()

**SALES:**

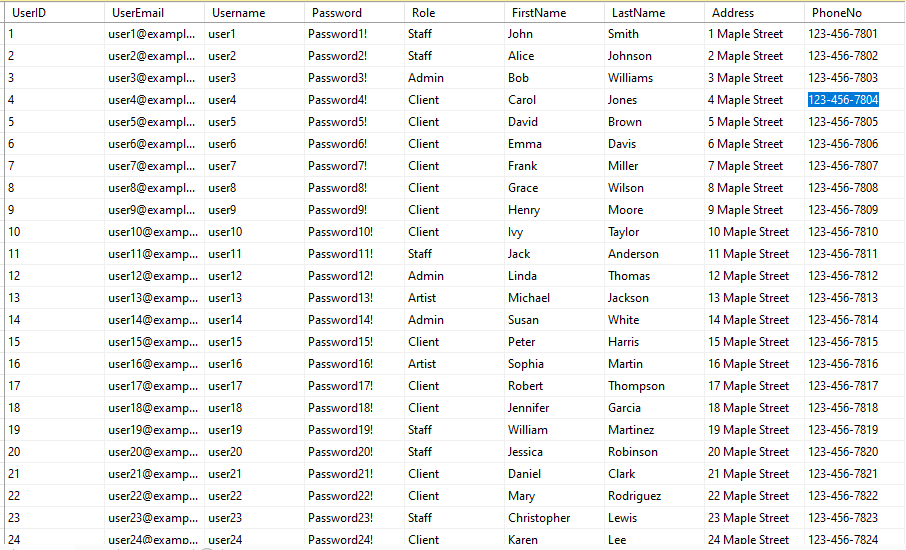
import random  
import pyodbc as odbc  
  
# Database connection details  
SERVER\_NAME = r'DESKTOP-8J774MH\SQLEXPRESS'  
DATABASE\_NAME = 'ArtGalleryManagementSystem'  
  
# Connection string  
conn\_str = (  
 f'DRIVER={{ODBC Driver 17 for SQL Server}};'  
 f'SERVER={SERVER\_NAME};'  
 f'DATABASE={DATABASE\_NAME};'  
 r'Trusted\_Connection=yes;'  
)  
  
# Establish connection  
conn = odbc.connect(conn\_str)  
cursor = conn.cursor()  
  
# Create Sales table if it doesn't exist  
cursor.execute('''  
 IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Sales')  
 CREATE TABLE Sales  
 (  
 SaleID INT,  
 UserID INT,   
 ArtID INT,  
 Price DECIMAL(10, 2),  
 SaleDate DATE,  
 ArtistCommission DECIMAL(10, 2),  
 CONSTRAINT PKSALES PRIMARY KEY (SaleID),  
 CONSTRAINT FKSALESUSER FOREIGN KEY (UserID) REFERENCES Users(UserID),  
 CONSTRAINT FKSALESART FOREIGN KEY (ArtID) REFERENCES Artwork(ArtID),  
 CHECK (Price > 0)  
 )  
 ''')  
  
# Fetch ArtID values and their corresponding prices from Artwork table for artworks that are available  
cursor.execute('SELECT ArtID, Price FROM Artwork WHERE Availability = \'Available\'')  
artworks = cursor.fetchall()  
art\_ids = [row[0] for row in artworks]  
art\_prices = {row[0]: row[1] for row in artworks}  
  
# Fetch UserID values from Users table where the role is 'client'  
cursor.execute('SELECT UserID FROM Users WHERE Role = \'client\'')  
user\_ids = [row[0] for row in cursor.fetchall()]  
  
# Define lists for Sales attributes  
sale\_ids = list(range(1, 121))  
sale\_dates = [  
 "2022-01-15", "2022-02-20", "2022-03-18", "2022-04-22", "2022-05-05", "2022-06-11", "2022-07-19",  
 "2022-08-24", "2022-09-30", "2022-10-14", "2022-11-23", "2022-12-02", "2023-01-15", "2023-02-20",  
 "2023-03-18", "2023-04-22", "2023-05-05", "2023-06-11", "2023-07-19", "2023-08-24", "2023-09-30",  
 "2023-10-14", "2023-11-23", "2023-12-02", "2024-01-15", "2024-02-20", "2024-03-18", "2024-04-22",  
 "2024-05-05", "2024-06-11"  
]  
artist\_commissions = [  
 1500, 2500, 350, 4500, 5500, 65000, 7500, 8500, 95000, 10500,  
 11500, 12500, 13500, 14500, 1550, 16500, 17500, 18500, 19500, 20500,  
 21500, 22500, 23500, 24500, 25500, 26500, 27500, 28500, 29500, 30500,  
 31500, 32500, 33500, 34500, 35500, 36500, 37500, 38500, 39500, 40500,  
 41500, 42500, 43500, 44500, 45500, 46500, 47500, 48500, 49500, 50000  
]  
  
# Randomly select 120 values from the customized lists  
sales\_records = list(zip(  
 random.sample(sale\_ids, 120),  
 random.choices(user\_ids, k=120),  
 random.choices(art\_ids, k=120),  
 random.choices(sale\_dates, k=120),  
 random.choices(artist\_commissions, k=120)  
))  
  
# Insert records into the Sales table  
for record in sales\_records:  
 art\_id = record[2]  
 price = art\_prices[art\_id] # Use the price from the Artwork table  
 cursor.execute('''  
 INSERT INTO Sales (SaleID, UserID, ArtID, Price, SaleDate, ArtistCommission)  
 VALUES (?, ?, ?, ?, ?, ?)  
 ''',  
 record[0], record[1], art\_id, price, record[3], record[4])  
 print("Inserted Successfully")  
  
# Commit the transaction  
conn.commit()  
  
# Close the cursor and connection  
cursor.close()  
conn.close()

**LOAN:**

import random  
import pyodbc as odbc  
from datetime import datetime, timedelta  
  
# Define lists for Loan attributes  
loan\_ids = list(range(1, 101))  
durations = [30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360]  
transport\_options = ["TCS Service", "FedEx", "DHL", "UPS", "BlueDart"]  
monitoring\_statuses = ["On track", "Delayed", "Completed"]  
  
# Generate random expiry dates within the next two years  
def random\_date():  
 start\_date = datetime(2023, 1, 1)  
 end\_date = datetime(2025, 1, 1)  
 delta = end\_date - start\_date  
 random\_days = random.randint(0, delta.days)  
 return start\_date + timedelta(days=random\_days)  
  
# Database connection details  
SERVER\_NAME = r'DESKTOP-8J774MH\SQLEXPRESS'  
DATABASE\_NAME = 'ArtGalleryManagementSystem'  
  
# Connection string  
conn\_str = (  
 f'DRIVER={{ODBC Driver 17 for SQL Server}};'  
 f'SERVER={SERVER\_NAME};'  
 f'DATABASE={DATABASE\_NAME};'  
 r'Trusted\_Connection=yes;'  
)  
  
# Establish connection  
conn = odbc.connect(conn\_str)  
cursor = conn.cursor()  
  
# Create Loan table if it doesn't exist  
cursor.execute('''  
 IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Loan')  
 CREATE TABLE Loan  
 (  
 LoanID INT,  
 UserID INT,   
 ArtID INT,  
 LoanAmount DECIMAL(10, 2),  
 DurationInDays INT,  
 ExpiryDate DATE,  
 TransportTracking VARCHAR(100),  
 MonitoringStatus VARCHAR(50),  
 CONSTRAINT PKLOAN PRIMARY KEY (LOANID),  
 CONSTRAINT FKLOANUSER FOREIGN KEY (UserID) REFERENCES Users(UserID),  
 CONSTRAINT FKLOANART FOREIGN KEY (ArtID) REFERENCES Artwork(ArtID)  
  
 )  
 ''')  
  
# Fetch Art\_IDs and their corresponding prices from Artwork table for available artworks  
cursor.execute('SELECT ArtID, Price FROM Artwork WHERE Availability = \'Available\'')  
artworks = cursor.fetchall()  
art\_ids = [row[0] for row in artworks]  
art\_prices = {row[0]: row[1] for row in artworks}  
  
# Fetch UserID values from Users table where the role is 'client'  
cursor.execute('SELECT UserID FROM Users WHERE Role = \'client\'')  
user\_ids = [row[0] for row in cursor.fetchall()]  
  
# Ensure unique Loan\_ID values by shuffling and slicing the list  
random.shuffle(loan\_ids)  
unique\_loan\_ids = loan\_ids[:100] # Adjusted to generate 100 records  
  
# Randomly select 100 values from the customized lists  
loan\_records = []  
for loan\_id in unique\_loan\_ids:  
 art\_id = random.choice(art\_ids)  
 user\_id = random.choice(user\_ids)  
 loan\_amount = art\_prices[art\_id] / 2  
 duration\_in\_days = random.choice(durations)  
 expiry\_date = random\_date().date()  
 transport\_tracking = random.choice(transport\_options)  
 monitoring\_status = random.choice(monitoring\_statuses)  
 loan\_records.append((loan\_id, user\_id, art\_id, loan\_amount, duration\_in\_days, expiry\_date, transport\_tracking, monitoring\_status))  
  
# Insert records into the Loan table  
for record in loan\_records:  
 cursor.execute('''  
 INSERT INTO Loan (LoanID, UserID, ArtID, LoanAmount, DurationInDays, ExpiryDate, TransportTracking, MonitoringStatus)  
 VALUES (?, ?, ?, ?, ?, ?, ?, ?)  
 ''',  
 record[0], record[1], record[2], record[3], record[4], record[5], record[6], record[7])  
 print("Inserted Successfully")  
  
# Commit the transaction  
conn.commit()  
  
# Close the cursor and connection  
cursor.close()  
conn.close()

**FAKER GENERATED TABLE**

**USERS:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

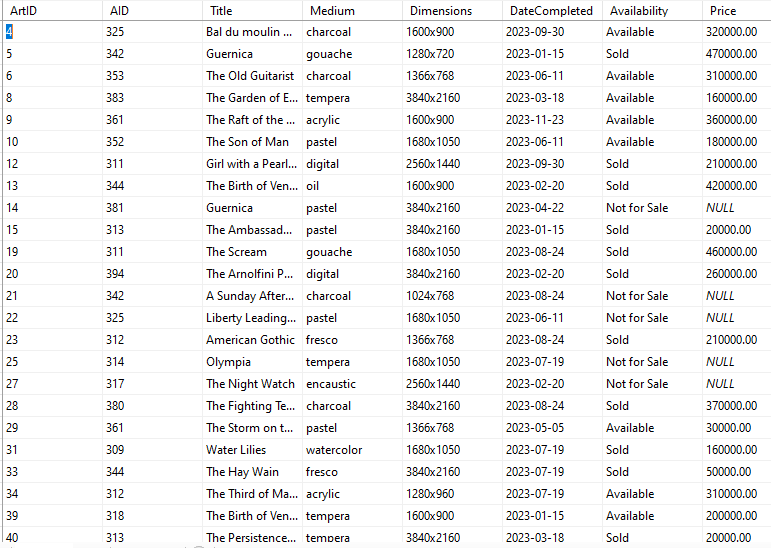
**ARTIST:**

**A screenshot of a computer

Description automatically generated**

****

**ARTWORK:**

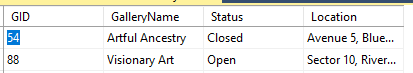
****

**A screenshot of a computer

Description automatically generated**

****

**GALLERY:**

****

**EXHIBITION:**

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**SALES:**

**A screenshot of a table

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a table

Description automatically generated**

**A screenshot of a table

Description automatically generated**

**A table of numbers and numbers

Description automatically generated**

**LOAN:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a data table

Description automatically generated**

**A screenshot of a computer

Description automatically generated**