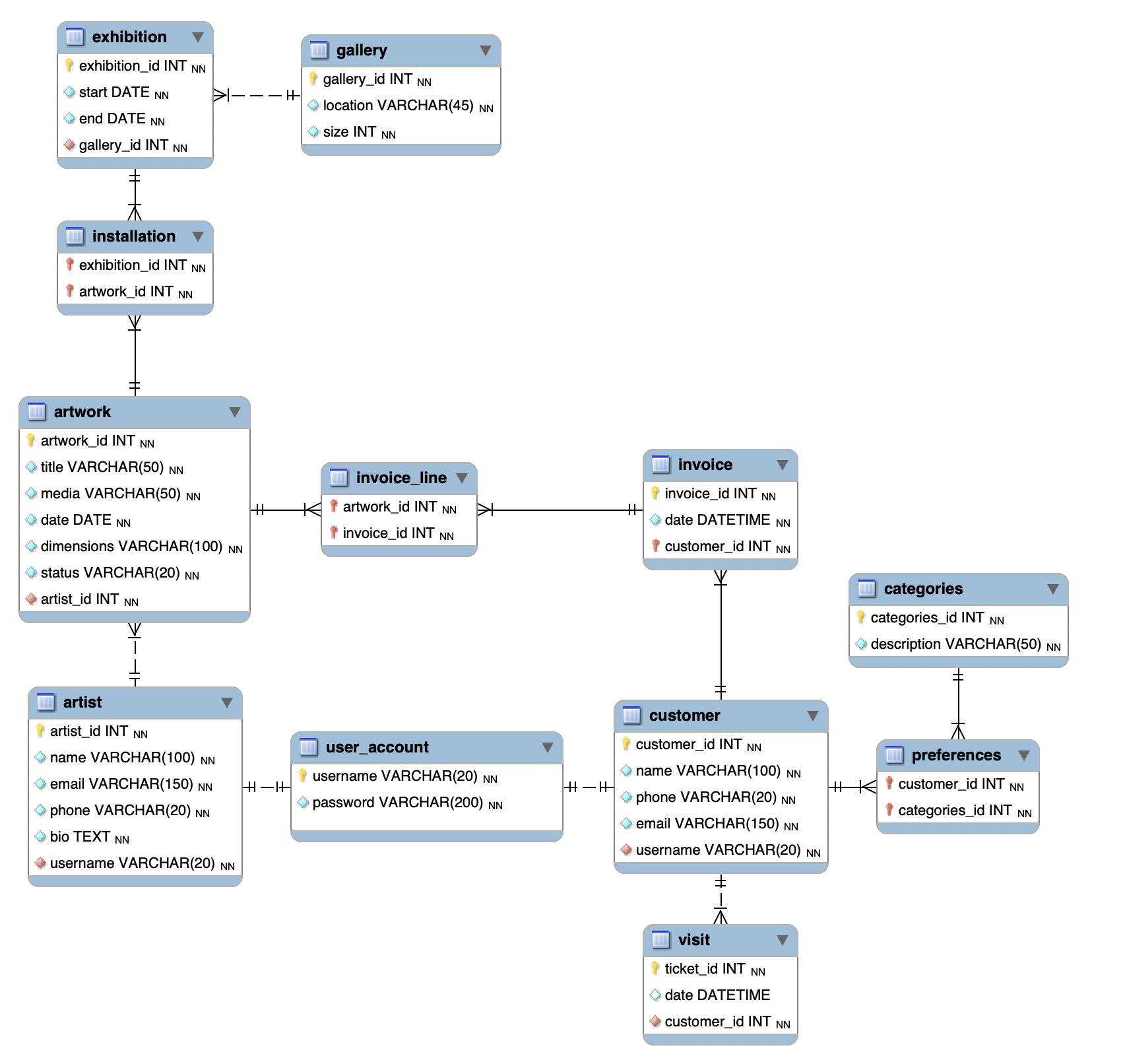
Assignment 2 – Implementing the Model

In this assignment you will take an ERD that was developed for the art gallery from Assignment 1 and implement this in MySQL along with some same data.

Regardless of your solution for assignment 1, you must use the following ERD for assignment 2.



# Data Definition Language Statements

You will need to write the DDL statements that can be used to create all the tables in the physical model. The creation will of course require the correct constraints including primary and foreign keys.

You will also need to write a “clean up” script that can be used to remove the tables for your project.

You will need to submit the SQL files used to create and remove the database scripts.

# Sample Data

You need to load sample data for the 12 tables. The more real the data looks, the better you can understand how the database is structured. The amount of sample data does depend on the table itself; some tables will need more rows than others. These are the minimum requirements, but data will make more sense if you were to double each of these.

* Artists: 3 records
* Customer: 3 records
* Artwork: 20 records
* User Accounts: one for each artist and each customer
* Customer Visit: 3 records
* Categories: 3 records
* Preferences: 2 records
* Invoice: 2 records
* Invoice Lines: 5 records
* Galleries: 4 records
* Exhibition: 5 records
* Installations: 10 records

To create the sample data, you may utilize any one of the following sources:

* manually – Just type data yourself. I sometimes use Excel to help create it.
* mockaroo.com – A very useful website that will generate random data. This is very good at some types of data.
* ChatGPT – **Yes**, you are permitted to use this tool for generating the data; it is a good use of ChatGPT! Please remember that you may need to ask it several times to get data that is compatible with the schema. I do not recommend using the tool to write the queries however because you will need to know how to do these yourself on the tests.
* Write a Python Program! Sometimes I’ve found that writing a program can generate data more easily than any other method.
* You may use a mixture of these for the different tables.

Whatever method you use, you must cite how you generated the contents for the table.

# Queries

Write queries for each of the following things that would likely be of interest to the gallery system. Please note that your data will be completely different because we all have random data.

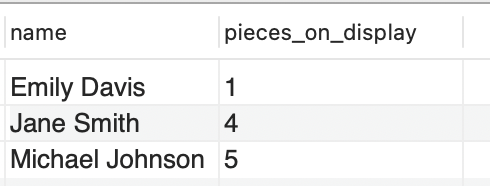
## Query 1

The art gallery wants to be able to display information on their website page about the total amount of square meters they have in all of their galleries. The “size” parameter in the gallery table is the number of square meters for each gallery.

## Query 2

Produce a report indicating how many pieces each artist has on display. The status of ‘DISPLAY’ could be used but you might be using a different value.

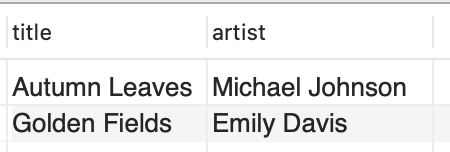
The report would look like this:



## Query 3

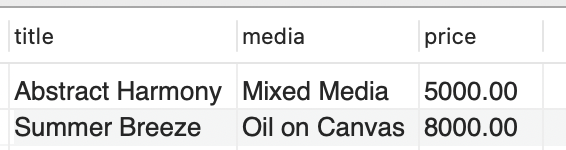
The gallery requires a query that will return a list of art along with the artist’s name that is being shown in a specific gallery.

In the example below, I asked for all the pieces with the word ‘canvas’ in the media for the gallery located in ‘Al Khor’. You may ask for different locations and different media based on your own sample data.



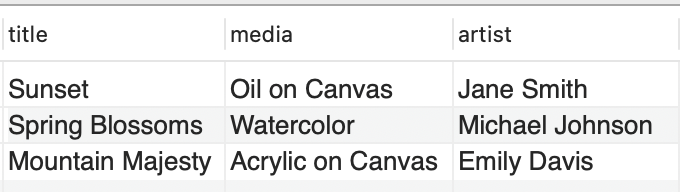
## Query 4

When art pieces are sold, an invoice needs to be created. The invoice must detail the name of the piece, the media on which is was created, and the price that was paid. Write the query that summarizes this information and produces an output like this. This query is for one specific invoice.



## Query 5

Write the query/queries required to create a new exhibition that runs from the start of the current semester until the end of the semester. The exhibition will consist of 10 paintings (you pick). Run a query after you have created the exhibition showing the title of the artwork, the media, and the artists’ names in the exhibition.



# Due Date

This assignment is due on June 13, 2024 at 11:59pm. Late submissions will be accepted until June 15, 2024 at 11:59pm with a 50% penalty.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Exemplary** | **Satisfactory** | **Developing** | **Unsatisfactory** |
| **DDL Statement Creation** (20 Points) | DDL statements are perfectly crafted to reflect the physical ERD. Every table, primary key, foreign key, and constraint is correctly defined without errors. This level demonstrates a deep understanding of how to translate an ERD into a functional database schema. (20) | DDL statements are mostly correct with only minor mistakes or omissions, such as slight errors in data types or missing a non-critical constraint. The student shows a good understanding of schema translation but may overlook finer details. (14) | DDL statements are present but contain several inaccuracies or omissions that could affect database integrity or performance, such as incorrect primary/foreign key relationships or missing several constraints. There is room for improvement in understanding or attention to detail. (8) | DDL statements are largely incorrect or missing, showing a significant misunderstanding of how to implement the ERD physically. Critical components like tables or keys are incorrectly defined or not included, severely impacting the schema's functionality. (0) |
| **Data Generation and Insertion** (10 Points) | Data inserted into the database is comprehensive, covering all tables and accurately representing relationships and constraints within the domain. The dataset includes a variety of cases, from typical to edge cases, effectively testing the schema's robustness and flexibility. (10) | Data covers most scenarios and tables, with minor gaps. The inserted data is realistic and respects table relationships, but some edge cases or constraints are not fully explored, offering a solid but not exhaustive test of the database schema. (7) | Data insertion covers only the basic scenarios, leaving out complex relationships or edge cases. Some tables may have sparse or unrealistic data, indicating a need for a more thorough approach to testing the database schema. (4) | Data insertion is minimal or incorrect, with many tables left empty or filled with data that doesn't respect defined relationships or constraints, indicating a lack of effort or understanding in how to populate the database schema realistically. (0) |
| **SQL Query Writing** (20 Points) | Queries are perfectly written, correctly utilizing joins, subqueries, and aggregation functions to retrieve data as specified. Queries are optimized for performance and clarity, demonstrating advanced SQL knowledge and the ability to handle complex data retrieval tasks. (20) | Queries meet most requirements with minor logical errors or inefficiencies. Joins and aggregation are used correctly, but there might be simpler or more effective ways to achieve the same results. Demonstrates a good grasp of SQL with minor areas for improvement. (14) | Queries show an attempt to meet requirements but contain several errors or inefficiencies. Basic joins may be used, but aggregation or more complex SQL features are incorrectly implemented or missing, indicating a developing understanding of SQL. (8) | Queries are incorrect, missing, or fail to meet the specified requirements, indicating a significant misunderstanding of how to retrieve data from the database or use SQL effectively. (0) |