Module name and code: COMP3391Practical Database Applications 2020-21

Student number: 19012712

Submission date: 8th January 2021

Assignment number 1

Contents Page

1. Design
   1. Introduction .............................................................................................. 3
   2. Functional and non-functional requirements …......................................... 3
   3. Entity Relationship Diagram ...................................................................... 4
   4. Data dictionary ………………………………………………………………........................ 5
   5. Assumptions .............................................................................................. 9
   6. Wireframe design for user interfaces ........................................................ 10
2. Documentation
   1. Database connection ................................................................................. 12
   2. DDL for creation of tables and key constraints .......................................... 13
   3. DML for inserting records ........................................................................... 14
   4. DML for deletion of records ....................................................................... 15
   5. DML for aggregations ................................................................................. 16
   6. DML for joins .............................................................................................. 17
   7. DML for subqueries .................................................................................... 18
   8. DDL for views ............................................................................................. 19
   9. DCL for user access .................................................................................... 20

1. Introduction

Blue Horizons is an art gallery which hosts exhibitions or events throughout the year. Clients, who may be individuals or a company/collector, can visit the events and then purchase the artworks either online via the website or at the event.

Blue Horizons will advertise its events on its web site and social media. It will also mailshot previous customers and those who have visited the gallery or website and requested to be put on the mailing list.

Blue Horizons would like to keep a record of details for all artists and their artworks who have shown their work at the gallery previously or who have been approached to show their work at an upcoming event.

Blue Horizons would also like to keep a record of which events artists have been shown at and record any sales.

Blue Horizons also employs several staff and would like to record which staff are deployed to an exhibition.

Blue Horizons would also like to keep a record of those expressing an interest in receiving a newsletter or updates on planned events.

Blue Horizons would use the database for queries on artworks and which would genres be suitable for a planned event. The database could also be used for queries on Sales to find out which artworks have been sold and how many were sold during a specified event.

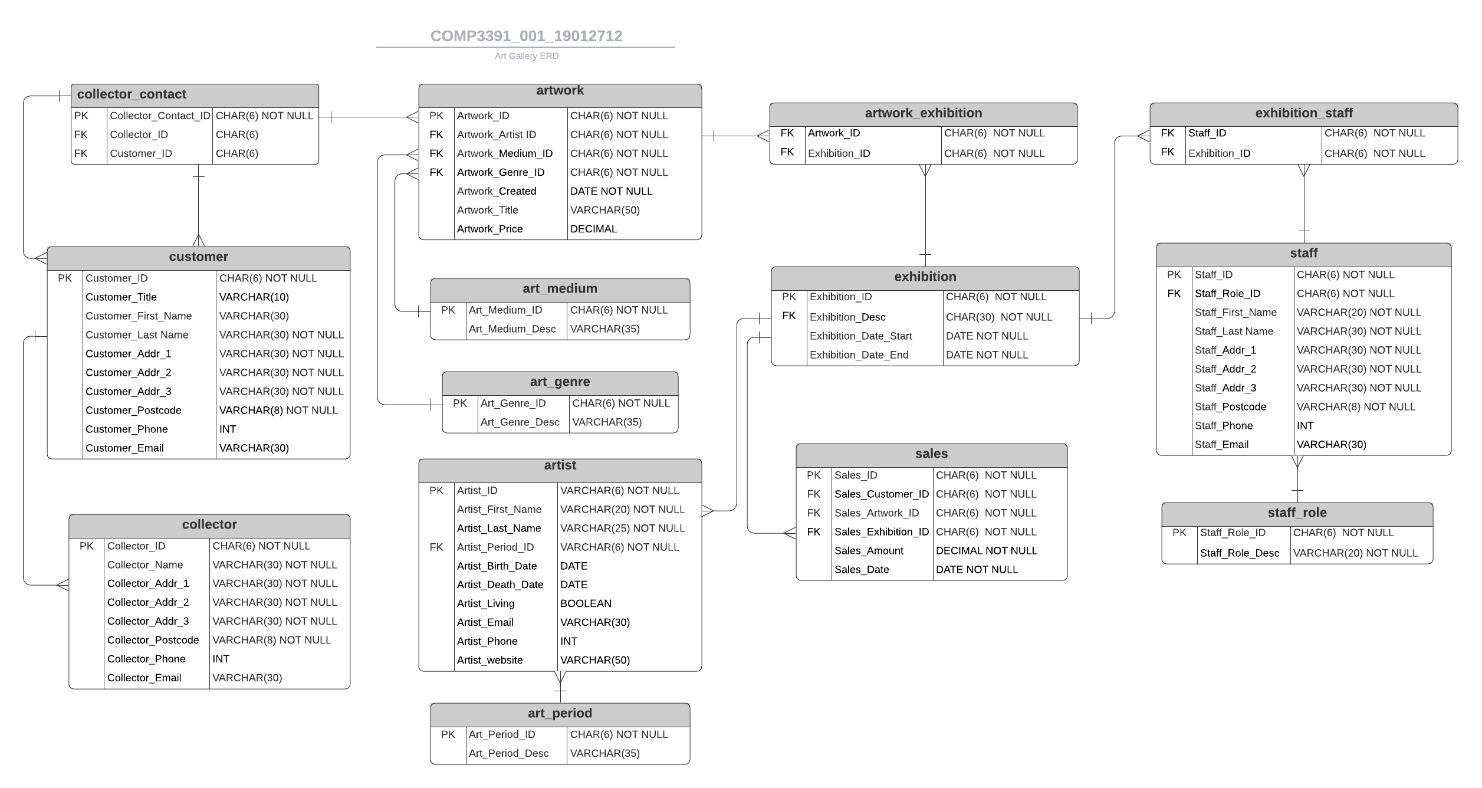
2. Functional and non-functional requirements

|  |  |  |
| --- | --- | --- |
|  | *Functional Requirements* | *MoSCoW* |
| 1 | Admin to log in securely | Must |
| 2 | Schedule events | Must |
| 3 | Record sales | Must |
| 4 | Output sales report | Must |

Non-functional requirements

|  |  |  |
| --- | --- | --- |
|  | *Non-functional requirements* | *Type* |
| 1 | Only authorised admin to access database | Security (data protection act compliance) |
| 2 | Database to be backed up regularly | Operational |
| 3 | Events to be well organised |  |

3. Entity Relationship Diagram for Blue Horizons Art Gallery



4. Data dictionary - populated with appropriate constraints

**artist**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Artist\_ID | CHAR | 6 | PK NOT NULL | Unique ID |
| Artist\_Period\_ID | VARCHAR | 35 | FK - NOT NULL | e.g Renaissance |
| Artist\_First\_Name | VARCHAR | 20 | NOT NULL | First name of Artist |
| Artist\_Last\_Name | VARCHAR | 25 | NOT NULL | Surname of Artist |
| Artist\_Birth\_Date | DATE |  |  | e.g 01/01/1919 |
| Artist\_Death\_Date | DATE |  | IF Not Null CHECK\_Death\_Date (Artist\_Death\_Date > Artist\_Birth\_Date) | e.g 01/01/1967 |
| Artist\_Living | BOOLEAN |  | Set to True if Artist\_Death\_Date is NULL | If the artist is living this is set to True |
| Artist\_Email | VARCHAR | 30 |  | Email contact eg artist@mail.com |
| Artist\_Phone | INT |  |  | Phone contact for the artist |
| Artist\_Website | VARCHAR | 50 |  | Artists website address. |

**artwork**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Artwork\_ID | CHAR | 6 | PK | Unique ID e.g. AW1000 |
| Artwork\_Artist\_ID | CHAR | 6 | FK - NOT NULL | Artist ID e.g AR1000 |
| Artwork\_Medium\_ID | CHAR | 6 | FK - NOT NULL | Medium ID e.g AM1000 |
| Artwork\_Genre\_ID | CHAR | 6 | FK - NOT NULL | Genre ID e.g AG1000 |
| Artwork\_Created | DATE |  | FK - NOT NULL | Date Artwork was created e.g 01/01/1919 |
| Artwork\_Title | VARCHAR | 50 |  | Title/description of the artwork e.g. Mona Lisa |
| Artwork\_Price\_GBP | DECIMAL |  |  | Price of the Artwork in GBP e.g. £1500 |
| Artwork\_Sold | BOOLEAN |  |  | Set to ‘true’ if a sales record exists for this artwork |

**artwork\_exhibition**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Exhibition\_ID | CHAR | 6 | FK - NOT NULL | Exhibition ID e.g.EX1000 |
| Artwork\_ID | CHAR | 6 | FK - NOT NULL | ArtworkID e.g.AW1000 |

**art\_genre**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Art\_Genre\_ID | CHAR | 6 | PK | Unique Genre ID e.g.AG1000 |
| Art\_Genre\_Desc | VARCHAR | 35 |  | Description of the genre e.g. Landscapes |

**art\_medium**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Art\_ Medium\_ID | CHAR | 6 | PK | Unique Genre ID e.g.AG1000 |
| Art\_ Medium\_Desc | VARCHAR | 35 |  | Description of medium used e.g. oil pastel |

**collector**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Collector\_ID | CHAR | 6 | PK | Unique Collector ID e.g.CO1000 |
| Collector\_Name | VARCHAR | 30 | NOT NULL | Collectors name |
| Collector\_Addr1 | VARCHAR | 30 | NOT NULL | 1st line of address e.g street number and name |
| Collector\_Addr2 | VARCHAR | 30 | NOT NULL | 2nd line of address e.g City |
| Collector\_Addr3 | VARCHAR | 30 | NOT NULL | 3rd line of address e.g County |
| Collector\_Postcode | CHAR | 8 | NOT NULL | Valid Post code for address |
| Collector\_Phone | INT |  |  | Phone contact number |
| Collector\_Email | VARCHAR | 30 |  | Email contact eg collector@mail.com |

**collector\_contact**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Collector\_Contact\_ID |  | 6 | PK | Unique collector contact ID e.g.CC1000 |
| Collector\_ID | CHAR | 6 | FK - NOT NULL | collector ID e.g.CO1000 |
| Customer\_ID | CHAR | 6 | FK - NOT NULL | customer ID e.g.CU1000 |
| Mailing\_List | BOOLEAN |  |  | True if customer is on the mailing list |

**customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Customer\_ID | CHAR | 6 | PK | Unique customer ID e.g.CO1000 |
| Customer\_Title | CHAR | 10 | CHECK\_Title (‘Mr’,’Mrs’,’Miss’,’Mx’,’Dr’,’Sir’) | Customer’s title e.g. Mx |
| Customer\_First\_Name | VARCHAR | 30 |  | First Name of the customer |
| Customer\_Last\_Name | VARCHAR | 30 | NOT NULL | Surname of the customer |
| Customer\_Addr1 | VARCHAR | 30 | NOT NULL | 1st line of address e.g street number and name |
| Customer\_Addr2 | VARCHAR | 30 | NOT NULL | 2nd line of address e.g City |
| Customer\_Addr3 | VARCHAR | 30 | NOT NULL | 3rd line of address e.g County |
| Customer\_Postcode | CHAR | 8 | NOT NULL | Valid Post code for address |
| Customer\_Phone | INT |  |  | Phone contact |
| Customer\_Email | VARCHAR | 30 |  | Email contact eg customer@mail.com |

**exhibition\_staff**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Exhibition\_ID | CHAR | 6 | FK - NOT NULL | Exhibition ID e.g.EX1000 |
| Staff\_ID | CHAR | 6 | FK - NOT NULL | Staff ID e.g.ST1000 |

**exhibition**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Exhibition\_ID | CHAR | 6 | PK | Unique Exhibition ID e.g.EX1000 |
| Exhibition\_Desc | CHAR | 30 | NOT NULL | Short description of the event e.g. |
| Exhibition\_Date\_Start | DATE |  | NOT NULL | Date when the event begins |
| Exhibition\_Date\_End | DATE |  | NOT NULL  CHK\_Exhibition\_Dates CHECK(Exhibition\_Date\_End > Exhibition\_Date\_Start | Date when the event ends. |

**sales**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Sales\_ID | CHAR | 6 | PK | Unique sales ID e.g.SA1000 |
| Sales\_Customer\_ID | CHAR | 6 | FK - NOT NULL | customer ID e.g.CU1000 |
| Sales\_Artwork\_ID | CHAR | 6 | FK - NOT NULL | Artwork ID e.g.AW1000 |
| Sales\_Exhibition\_ID | CHAR | 6 | FK - NOT NULL | Exhibition ID e.g.EX1000 |
| Sales\_Amount\_GBP | DECIMAL | 10,2 | NOT NULL | Amount of the sale in GBP e.g. £1500 |
| Sales\_Date | DATE |  | NOT NULL | Date of sale e.g 01/01/2019 |

**staff**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Staff\_ID | CHAR | 6 | PK | Unique staff ID e.g.ST1000 |
| Staff\_Role\_ID | CHAR | 6 | FK - NOT NULL | Staff Role ID e.g.SR1000 |
| Staff\_First\_Name | VARCHAR | 20 | NOT NULL | First Name of the member of staff |
| Staff\_Last\_Name | VARCHAR | 30 | NOT NULL | Surname of the member of staff |
| Staff\_Addr1 | VARCHAR | 30 | NOT NULL | 1st line of address e.g street number and name |
| Staff\_Addr2 | VARCHAR | 30 | NOT NULL | 2nd line of address e.g City |
| Staff\_Addr3 | VARCHAR | 30 | NOT NULL | 3rd line of address e.g County |
| Staff\_Postcode | CHAR | 8 | NOT NULL | Valid Post code for address |
| Staff\_Phone | INT |  |  | Phone contact number |
| Staff\_Email | VARCHAR | 30 |  | Email contact |

**staff\_role**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **datatype** | **Length** | **Constraint** | **Example** |
| Staff\_role\_ID | CHAR | 6 | PK | Unique staff role ID e.g.SR1000 |
| Staff\_Role\_Desc | VARCHAR | 20 | NOT NULL | Description of the staff role e.g. Curator |

5. Assumptions

Admin will login using a valid and secure username and password

The gallery has only one venue which can host several events simultaneously.

An artist can show many artworks at many events.

An individual artwork can only appear once at an event.

Potential visitors are informed of events via the website, through a mailing list and advertising on social media

An artist can be responsible for many artworks

A customer can buy many artworks

A client can either be an individual or a company (Collector)

6. **Wireframe for the user login interface**

Graphical user interface, text, application

Description automatically generated

Login screen for existing users.

Enter Username {

(Validate\_username) check if username = existing username

If

true

then

Error\_message ‘ username exists, try again’

else

go to (Enter\_Password)}

(Enter\_Password) check if valid password using regex (e.g. \b[A-Z0-9.\_%+-][+@[A-Z0-9.-]+\.[A-Z]{2,}\b](mailto:+@[A-Z0-9.-]+\.%5bA-Z%5d%7b2,%7d\b)

If

false

then

Error message ‘invalid email, please try again’

else

check username/password = existing username/password

If

true

then

go to admin page

else Error\_message ‘password is incorrect, try again’

If

user does not exist

Then

load Create New User page

**Wireframe for the admin interface**

Below shows an interface for a user to search for upcoming events by month

User chooses a month from a dropdown menu and a query is performed to show all events occurring within that month.

An improvement would be to add two date fields to enable the user to search between two datesTable

Description automatically generated

Get month from dropdown menu

Get event details where start\_date\_month = event\_month

Get user input from dropdown menu ( get distinct event\_month from all event start dates)

Get event details where start\_date\_month = event\_month

If

false

then

display message ‘no records found’

Else

display event details

From exhibition table where event\_month = exhibition\_start\_date\_month

**Wireframe for the curator interface**

Below shows an interface for a curator to search for all artworks currently registered with the gallery by artist.

The user enters part or all of an artists surname and a query is performed to show all artworks with price and sold status ordered by artist.Table

Description automatically generatedGet user\_input

Get artist\_ID where artist surname like user\_input

Check if artist\_ID = Artwork\_Artist\_ID

If

false

then

display message ‘no records found’

Else

display artist and artwork details

From artwork table where artist\_ID = Artwork\_Artist\_ID

**Documentation**

* 1. **Database Connection**

Database connection using a file named connect.php.

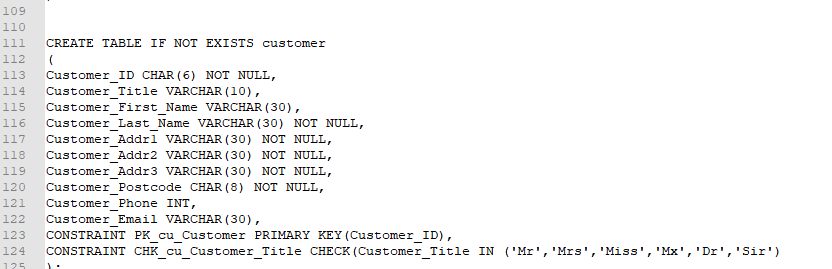
The following code creates a connection to a named database (in this case – comp3391\_19012712) , with a user name of root and a blank password. This is fine for test purposes, but users will need to be created and privileges assigned to protect data and prevent accidental deletion or updates.



* 1. **DDL for creation of tables and constraints**

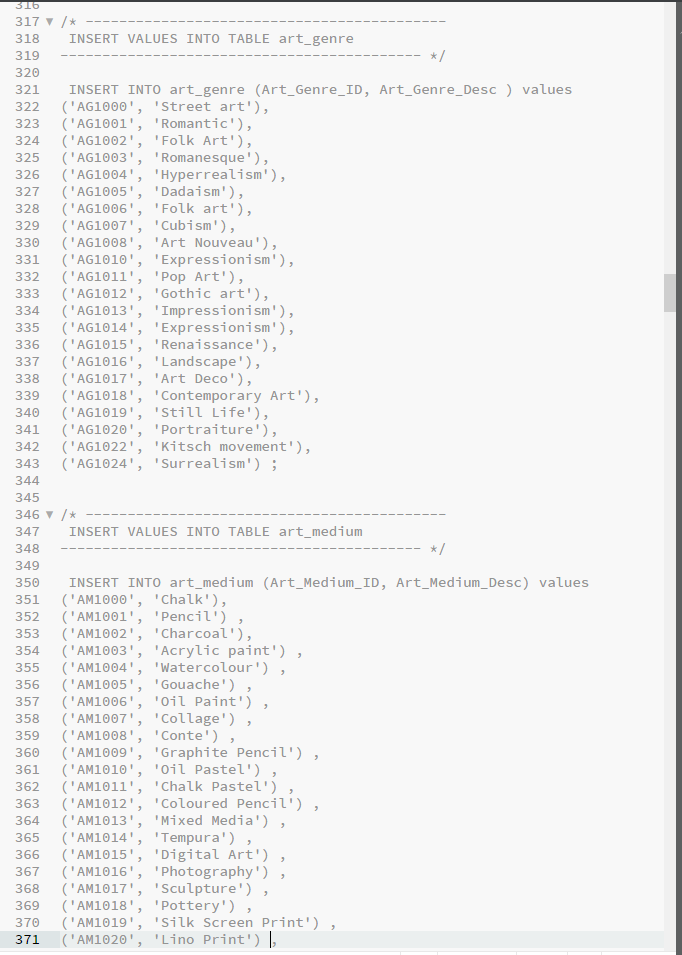
Create a table from the data dictionary. First the table to be created should be dropped in case of conflicts in name.

The table is then created by giving it a name then defining each field and its properties. Constraints are included to define primary and foreign keys or to restrict input to specific values. For example the constraint below checks that the customer title is a valid title.



* 1. **DML for inserting records**

Insert data into records by specifying the fields and the values to be entered into each field.



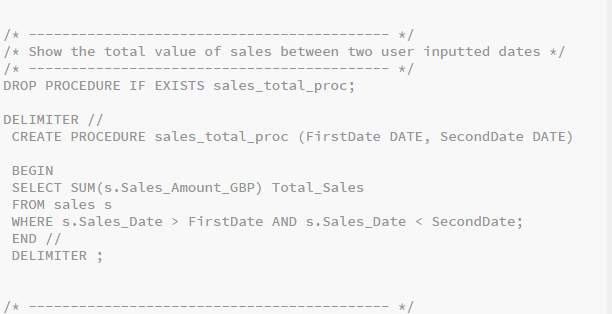
* 1. **DML for deletion of records**

Delete a record from a database by using the syntax DELETE FROM, the WHERE clause should be used to prevent deleting all records from a table. In the example below one record is deleted where a field is equal to ’Fauvism’ a wild card (%) could be used with LIKE to delete a broader range of records (e.g. LIKE ‘F%’ would have deleted records with a genres of Fundamentalism and Fun Art as well as Fauvism).



* 1. **DML for aggregations**

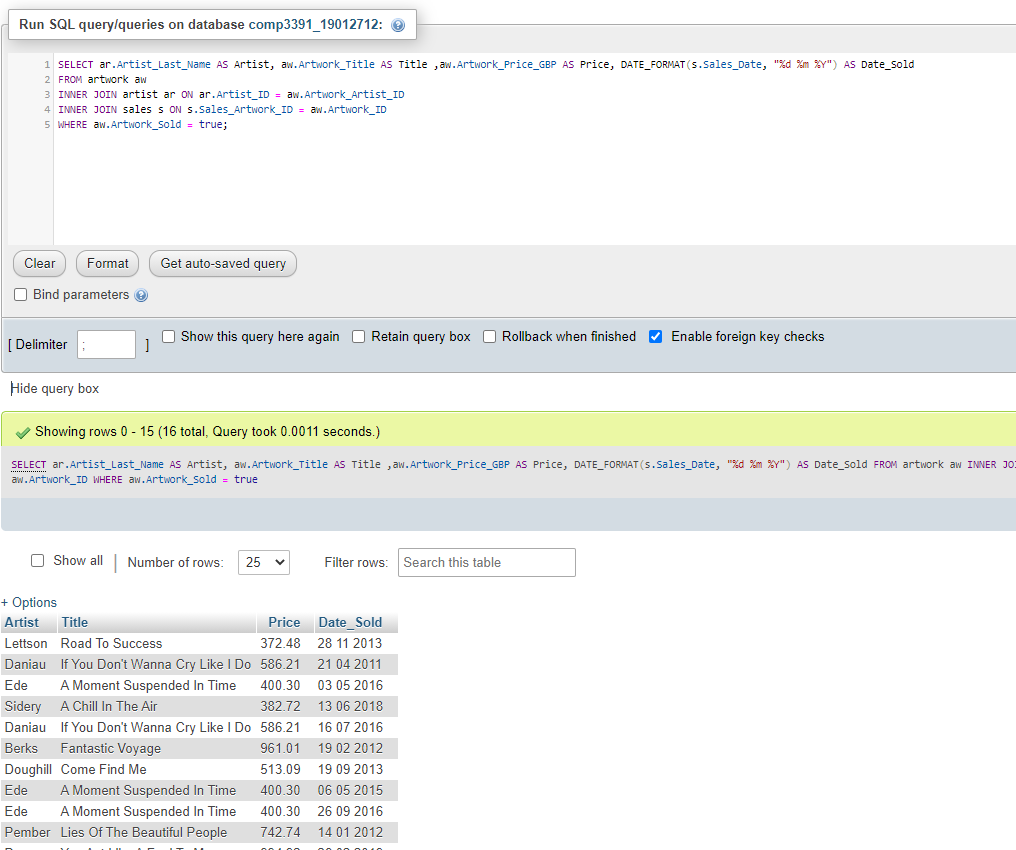
Below shows a procedure that uses aggregation. In this example the SUM() function is used to calculate the total value of artworks sold between two given dates.



* 1. **DML for joins**

A JOIN statement will allow a complicated query from several tables.

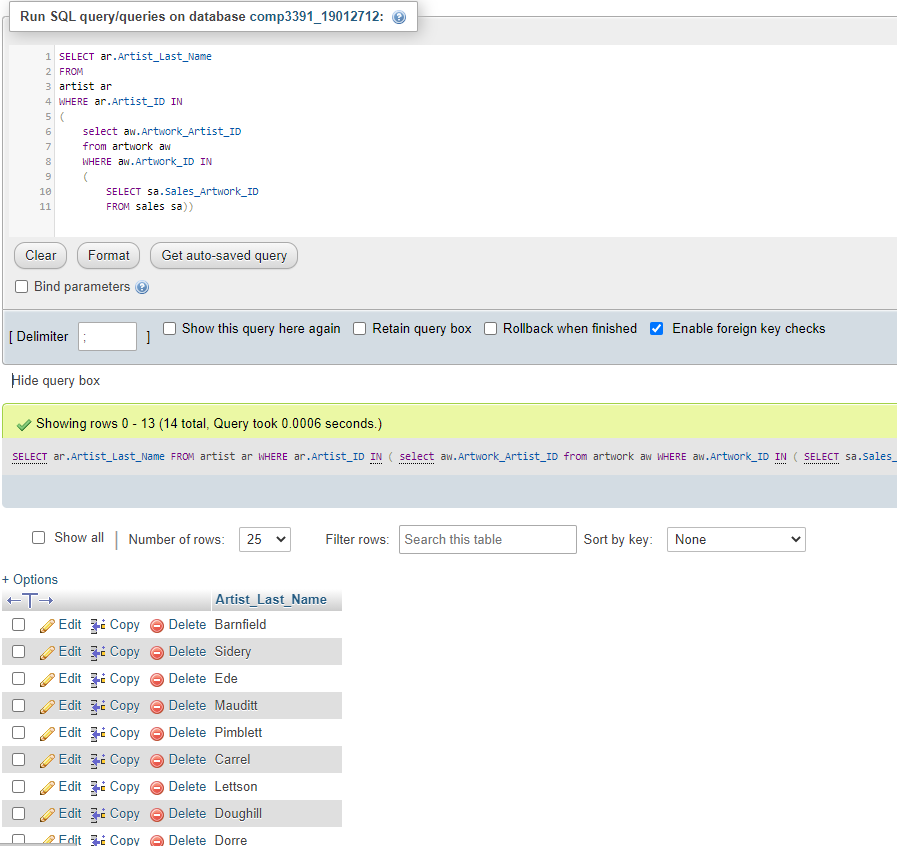
Below shows a query on three tables to find the all sold artworks and display the title of the work, the artist who created it, when it was sold and how much it sold for.



* 1. **DML for subqueries**

A subquery is similar to a JOIN in that it will allow a complicated query from several tables. However it will only return one field.

Below shows a subquery on three tables to find the all artists who sold artworks.



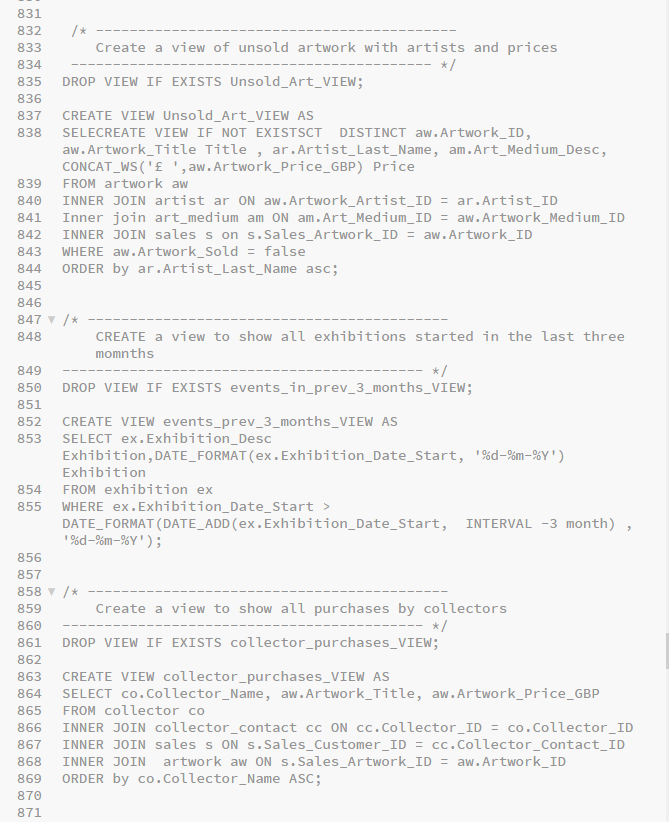
* 1. **DDL for views**

Views are created for common queries on a database.

This prevents the need for creating a query every time information is required, this speeds up the database and reduce the amount of code needed

Specific views can be created for roles which only require access to certain tables.

Below shows a query that may be useful for the sales department to find out which artwork remains unsold. Concatenated fields are given an alias to give a more readable title. Other views are created to show all events in the last three months and to show which artworks have been bought by collectors



* 1. **DCL for user access**

Below is shown code to create a user (Billy Russell) with a password of ‘snowlimp’.

Access is given to this user to manipulate the ‘staff’ table.

A role of ‘Personnel’ is created so that all privileges given to the role need not be duplicated for every user requiring access. All privileges given toi the role of Personnel and then a user can easily be

