Bournemouth University

# PROJECT REPORT

Data Processing and Analytics Course (COMP7067)

**Submitted to:** Dr. Ismail Alarab

**Submitted by Group 18:**
Yasemin Karaca (s5619032)
Gozde Sarsar (s5634417)
Djameleddine Derias (s5087829)
Hakam Farid (s5629628)
Moustafa Amin (s5553760)

**May 2024**

## Table of Contents

# Abstract

The project is split into two main parts: the first part involves developing a database from an Entity-Relationship model to implementation in SQL, MongoDB, and Neo4J, alongside generating and evaluating test cases. The second part includes machine learning, selecting a dataset for a classification issue, applying neural networks, random forest, and other methods like PCA for dimensionality reduction and feature selection to assess performance impacts. The project demonstrates a detailed exploration of both theoretical and practical elements in database management and machine learning, supported by multiple software tools to achieve robust data handling and analysis.

# 1. Introduction

## Assignment Description

This document has been created and is being submitted as the main deliverable for the Data Processing and Analytics Course. The assignment comprises two main parts focused on database technologies and data analysis with machine learning. In Part A, choosing one of three use-case scenarios—Art Galleries, Academic Publishing, or Soccer Teams—and developing an Entity-Relationship model, translating it into a relational schema, and implementing key entities in SQL, MongoDB, and Neo4J, along with generating and showcasing sample data. In addition, justifying the technology choices and executing five test cases for each database technology. Part B requires the choice of one out of three data sets for a specified classification issue, the execution of a neural network and an additional classification method, the use of Principal Component Analysis for reducing dimensionality, the performance of feature selection to assess performance effects, and the investigation of model interpretability. The task is structured to evaluate both theoretical comprehension and practical implementation in the fields of database administration and machine learning methodologies.

## Tools/Software Employed

The following tools were utilised in the development of this report and its background:

- **ERD-Plus:** for ER Model and Relational Model development
- **MySQL:** for SQL database development
- **Neo4j:** for developing graph databases.
- **MongoDB:** for developing of non-relational document database

- **Python:** for analysing the data

### Document Organisation

Task A is the subject of Section II of this report, while Task B will be discussed in Section III. Each section’s outline is based on a set of instructions from the assignment brief.

For Part A, we first go over our entity-relationship model. Next, we proceed with the creation of a relational schema. Subsections 3–4 and 5 are repeated three times (designated as a, b, and c) for SQL, Neo4J, and MongoDB, respectively.

In Part B, we start by preprocessing and cleaning the chosen dataset. After that, we implement two classification models (Neural Network & Random Forest). The next step is to use PCA (Principal Component Analysis) to reduce dimensionality, and we analyze and compare the results of the two models. Next, we analyze the influence of feature selection on classification metrics and repeat one of the previous classifications using the obtained Principal Components. Finally, we apply 2 Explainable AI methods to understand the why the machine learning model made certain decisions

We have carefully considered the ILOs and Marking Criteria for each part and have tried to include extra material to support our understanding mg and methodology.

# 2. Part A

**Use Case Selection**

Use case 1 was selected for this project because it aligns with our passion for art and allows us to explore the complex relationships between artworks, artists, and their audiences within a database framework. This use case offers a multifaceted approach to handling data by including diverse entities such as artists, artworks, and customers, along with their interconnected attributes like art styles, group classifications, and purchase preferences. Implementing a database for ArtBase will enable us to design a system that not only stores but also effectively manages and retrieves information critical for art galleries. This practical application helps in understanding both the technical aspects of database management systems and the real-world dynamics of the art industry, making it an ideal scenario for our academic and professional growth in the field of database technologies.

## 2.1. Entity Relationship Model

**Model Development Step 1: Use-case review**

Analysing the supplied use-case statement was the initial stage in developing the model. To further elaborate on this step, we replicate the use-case statement below and highlight its crucial elements:

Blue text - indicative of entities

Green text - indicative of attributes

Red text – indicative of relationships and in some cases their cardinality.

Consider your passion for art leading to the creation of ArtBase, a company that builds artwork items for art galleries. The core of this company is a database with a schema that captures all the information that galleries need to maintain. Galleries keep information about artists, their names (which are unique), birthplaces, age, and style of art. For each piece of artwork, the database should include the artist's name, the year it was made, its unique title, its type of art (e.g., painting, lithograph, sculpture, photograph), and its price. Pieces of artwork are also classified into groups of various kinds, for example, portraits, still lives, works of the 19th century, etc. A given piece may belong to more than one group. Each group is identified by a name (like those just given) that describes the group. Finally, galleries keep information about customers. For each customer, galleries keep that person's unique name, address, total amount of dollars spent in the gallery, the artists and groups of art that the customer tends to like.

The entities and their respective attributes:

1- Artist

- Artist_name (primary key)
- Artist_birthplace
- Age
- Style

2- Artwork

- Title (primary key)
- Artist_name (foreign key from Artist)
- Year_made
- Art_Type (e.g., painting, lithograph, sculpture, photograph)
- Price

3- ArtworkGroup

- Group_name (primary key)

4- Customer

- Cust_name (primary key)
- Cust_address
- Total_spent ($)

We used this information as our base to build the Entity Relationship Model in the next step.

## Model Development Step 2: ER Version 1

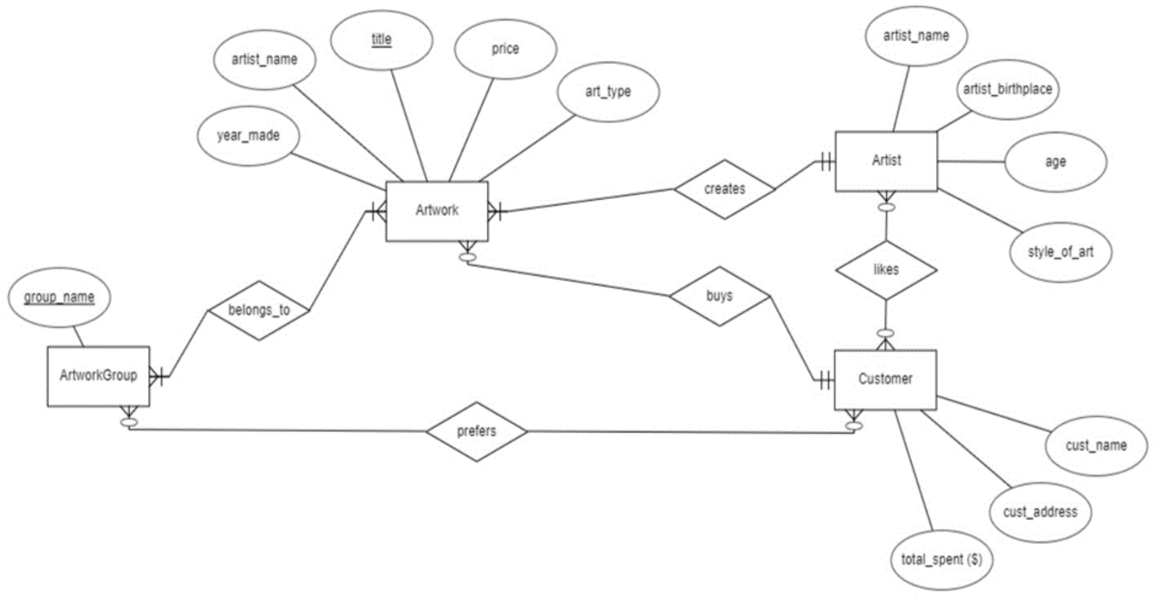


**Figure 1. ER diagram version 1**

The ER v1 meets the use-case requirements in the following manner:

| Use-case Requirement Statement | How ER v1 satisfies the requirement |
|---|---|
| Galleries keep information about artists, their names (which are unique), birthplaces, age, and style of art | ▪ Artist is an entity.<br>▪ Attributes: artist_id (we assumed this attribute to be used as primary key instead of artist_name), artist_name , artist_birthplace, age, style_of_art. |

| | |
|---|---|
| For each piece of artwork, the database should include the artist's name, the year it was made, its unique title, its type of art (e.g., painting, lithograph, sculpture, photograph), and its price. | ▪ Artwork is an entity . Attributes: artist_id (instead of artist_name to be in consistence with artist entity), year_made |
| Pieces of artwork are also classified into groups of various kinds, for example, portraits, still lifes, works of the 19th century, etc. A given piece may belong to more than one group. Each group is identified by a name (like those just given) that describes the group | ▪ ArtworkGroup is an entity. ▪ Attributes: group_name. ▪ Relationship between Artwork and ArtworkGroup (belongs_to). ▪ Cardinality of relationship is many-to-many |
| Finally, galleries keep information about customers. For each customer, galleries keep that person's unique name, address, total amount of dollars spent in the gallery. | ▪ Customer is an entity. ▪ Attributes: cust_id (we assumed this attribute to be used as primary key instead of cust_name), cust_name, cust_address, total_spent. |
| the artists and groups of art that the customer tends to like. | ▪ Relationship between Customer and Artist (likes). ▪ Cardinality of relationship is many-to-many. ▪ Relationship between Customer and ArtworkGroup (prefers) ▪ Cardinality of relationship is many-to-many. |

After reviewing the first ER model, we decided we can make the following modifications to improve it:

1- Considering 'Gallery' as an entity and we assumed its attributes to be the following:
   ▪ Gallery_name (Primary Key)
   ▪ Manager
   ▪ City
   ▪ Total_sale ($)
2- Use ID as an alternative PK for Artist and Customer (this will make our ER better and easier for querying), which led to the change of the attribute 'artist_name' in entity 'Artwork' to 'artist_id' (this is a foreign key from Artist).
3- We have noticed that age is not a static attribute and should be dynamic (its value should change depending on the current date), so we have used birthdate instead and derived the age from it.

4- For the ‘Customer’ entity we added the additional attribute ‘gender’ as it will allow us to use the ‘ENUM’ datatype in our model.
5- Add attribute ‘purchase_date’ to relationship ‘busy’ to make it more efficient in tracking the different customers purchase orders.

### Model Development Step 3: ER Version 2 (Final)

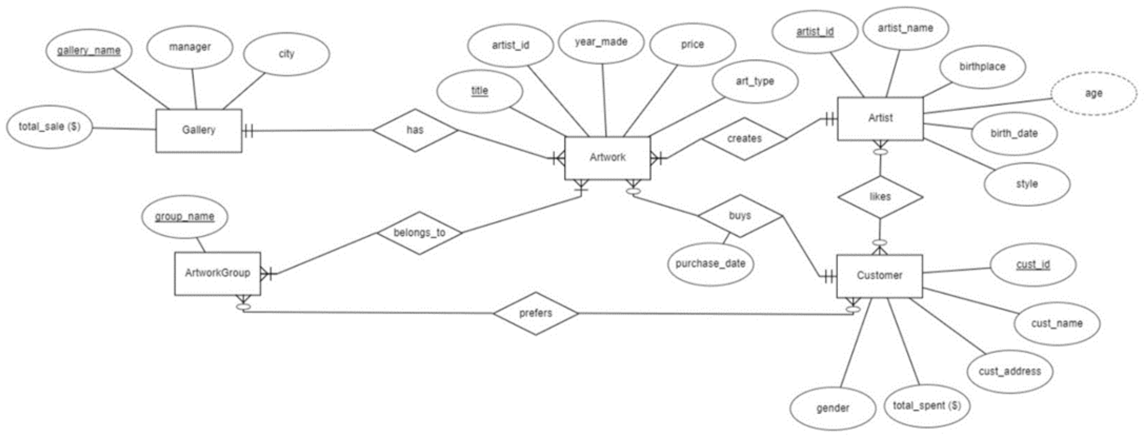


**Figure 2. ER diagram version 2**

### Description of entities

| Entity | Attribute | Definition and Purpose |
|---|---|---|
| **Gallery**<br>Regular Entity \| Represents galleries in different cities | ▪ gallery_name<br>▪ manager<br>▪ city<br>▪ total_sale($) | ▪ Unique Gallery name (Primary Key)<br>▪ Name of each gallery manager<br>▪ The city which the gallery is located<br>▪ Total amount of revenue of each gallery in USD |

| | | |
|---|---|---|
| **Artwork**<br>Regular Entity \| Represent artworks made by different artists | ▪ title<br>▪ artist_id<br>▪ year_made<br>▪ price<br>▪ type_of_art | ▪ Unique artwork title (Primary Key)<br>▪ Artist_id (which is a foreign key for artist entity)<br>▪ The year which the artwork was created<br>▪ The price of the artwork in USD<br>▪ Art type (e.g., painting, lithograph, sculpture, etc.) |
| **Artist**<br>Regular Entity \| Represents all the artists who created at least one artwork | ▪ artist_id<br>▪ artist_name<br>▪ artist_birthplace<br>▪ birth_date<br>▪ age<br>▪ style | ▪ Unique Artist ID (Primary Key)<br>▪ Unique Artist name<br>▪ City where artist was born<br>▪ Date of birth (YYYY-MM-DD)<br>▪ Age of each artist (Derived from birth_date)<br>Art Style (e.g., Symbolist, Landscape Photography, Symbolist sculpture, etc.) |
| **Customer**<br>Regular Entity \|<br>Represents all customers who bought artworks | ▪ Cust_id<br>▪ Cust_name<br>▪ Cust_address<br>▪ Total_spent<br>▪ gender | ▪ Unique customer ID (Primary Key)<br>▪ Unique Artist name<br>▪ Detailed address of customers<br>▪ Total money spent purchasing artwork in USD<br>▪ Male or Female (M/F) |
| **ArtworkGroup**<br>Regular Entity \| Represents the different art Groups | ▪ group_name | ▪ Unique group name [e.g., Minimalism, Modern Art, Surrealism, Portraits, etc.] (Primary Key) |

## Relationships (Cardinalities & Participations)

*(Note: Mand = Mandatory, Opt = Optional)*

| | **Between** | **Cardinality** | **Participation** | **Attributes** | **Explanation** |
|---|---|---|---|---|---|
| Has | Gallery -Artwork | 1-M | Mand-Mand | | Relates galleries and the displayed artworks.<br>A gallery can have more than one artwork.<br>Artwork can be displayed in only one gallery at a time. |

| | | | | | |
|---|---|---|---|---|---|
| Creates | Artist- Artwork | 1-M | Mand-Mand | | Relates artists with their created artworks.<br>An artist can create many artworks.<br>A single artwork is created by only one artist. |
| Buys | Customer-Artwork | 1-M | Mand-Opt | purchase_date | Links customers with their purchased artworks<br>A customer may buy more than one piece of artwork.<br>A single piece of artwork is owned by only one customer. |
| Belongs_to | Artwork-ArtworkGroup | M-M | Mand-Mand | | Links each artwork with the groups it belongs to.<br>A single artwork can belong to many artgroups.<br>A single artgroup can have many artworks. |
| Likes | Customer-Artist | M-M | Opt- Opt | | Relates Customers with their favorite artists.<br>A customer may like many artists.<br>An artist can be the favorite for many customers. |
| Prefers | Customer - ArtworkGroup | M-M | Opt- Opt | | Relates Customers and their preferred artgroups.<br>A customer may prefer many artwork groups.<br>An Artgroup may be preferred by many customers. |

## 2.2. Conversion into Relational Schema (Model)

In this section, we will explain in detail the steps taken to transform the ER model to the Relational Schema utilizing the 8-step conversion rules which consist of the following:

1. Convert each regular(strong) entity to a relation.
2. Convert each weak entity into a relation with foreign keys to its identifying relations/entities (A weak entity cannot exist alone).

3. Convert one-to-one relationships into a UNIQUE foreign key reference from one relation to the other.
4. Convert one-to-many relationships into a foreign key reference from the N-side relation to the1-side relation.
5. Convert many-to-many relationships into a new relation with foreign keys to the two participating entities.
6. Convert a multi-valued attribute into a relation with a composite primary key consisting of the attribute value plus the primary key of the attribute's entity.
7. Convert n-ary relationships by creating a new relation to represent the relationship and creating foreign keys that reference the related entities.
8. Convert subclasses and super-classes by creating a relation for each subclass and superclass and place foreign keys in the relations corresponding to subclasses.

## Step 1: Convert regular(strong) entities to relations.

This step involves the following:

a. For each regular entity creates a relation that includes all its simple attributes. Include only the simple component attributes of a composite attribute.
b. Choose one of the key attributes of the Entity as a primary key for relation.

As all the entities in our ER Model are regular, the outcome of Step 1 is given below:

- Gallery (gallery_name, manager, city, total_sale
- Artwork (title, artist_id, year_made, price, art_type
- Artist (artist_id, artist_name, birthplace, birth_date, age, style_of_art
- Customer (cust_id, cust_name, cust_address, total_spent, gender
- ArtworkGroup (group_name

## Step 2: Convert Weak Entities in Relations

As we don’t have any weak entities in our ER Model, we will not apply this step and proceed to step 3.

## Step 3: Convert One-to-one relationships

There are no one-to-one relationships in our ER Model. Therefore, we can proceed directly to step 4.

## Step 4: Convert One-to-Many relationships

This step involves the following:

a. Identify each one-to-many relation.
b. Add the primary key of the one relation as a foreign key into the many-side relation.
c. Add any relationship attributes as columns to the many-side relation.

In our ER model we have 3 one-to-many relationships (has, creates, buys), so the outcome after applying the above-mentioned conversion steps will be as follows:

- Artwork (title, artist_id, year_made, price, art_type, galler_name*, cust_id*, purchase_date

So far, our conversion outcome is:

- Gallery (gallery_name, manager, city, total_sale
- Artwork (title, artist_id, year_made, price, art_type, galler_name*, cust_id*, purchase_date
- Artist(artist_id, artist_name, birthplace, birth_date, age, style_of_art
- Customer (cust_id, cust_name, cust_address, total_spent, gender
- ArtworkGroup (group_name)

## Step 5: Convert Many-to-Many relationships.

This step comprises:

- Identify each many-to-many relationship.
- Create a relation for this relationship
- Add primary keys of participating relations as foreign keys to the new relation.
- Add simple attributes of the relationship as columns in the new relation.

There are 3 many-to many relationships in our ER Model (likes, prefers, belongs_to), so the outcome after applying the above-mentioned conversion steps will be as follows:

- likes (artist_id*, cust_id*)
- prefers (group_name*, cust_id*)
- belongs_to (group_name*, title*)

## Step 6: Convert Multi-valued attributes.

As we don’t have any Multi-valued attribute in our ER Model, we will proceed to step 7.

## Step 7: N-ary Relationships

This step relates to converting N-ary relationships. This step does not apply to our ER Model as we do not have any relationship with N > 2.

## Step 8: Subclasses and Super-classes

This step relates to the inheritance concept where two or more entities with a single superclass need to generalize generic attributes and identity to a parent relation, which implements child/ varying attributes into child relations.

This step does not apply to our ER Model as our case does not require a super/sub-class structure.

After completing the 8-step conversion process, the outcome is as shown below:

- Gallery (gallery_name, manager, city, total_sale)
- Artwork (title, artist_id, year_made, price, art_type, galler_name*, cust_id*, purchase_date)
- Artist(artist_id, artist_name, birthplace, birth_date, age, style_of_art)
- Customer (cust_id, cust_name, cust_address, total_spent, gender)
- ArtworkGroup (group_name)
- likes (artist_id*, cust_id*)
- prefers (group_name*, cust_id*)
- belongs_to (group_name*, title*)

This can be presented as a Relational Schema below:

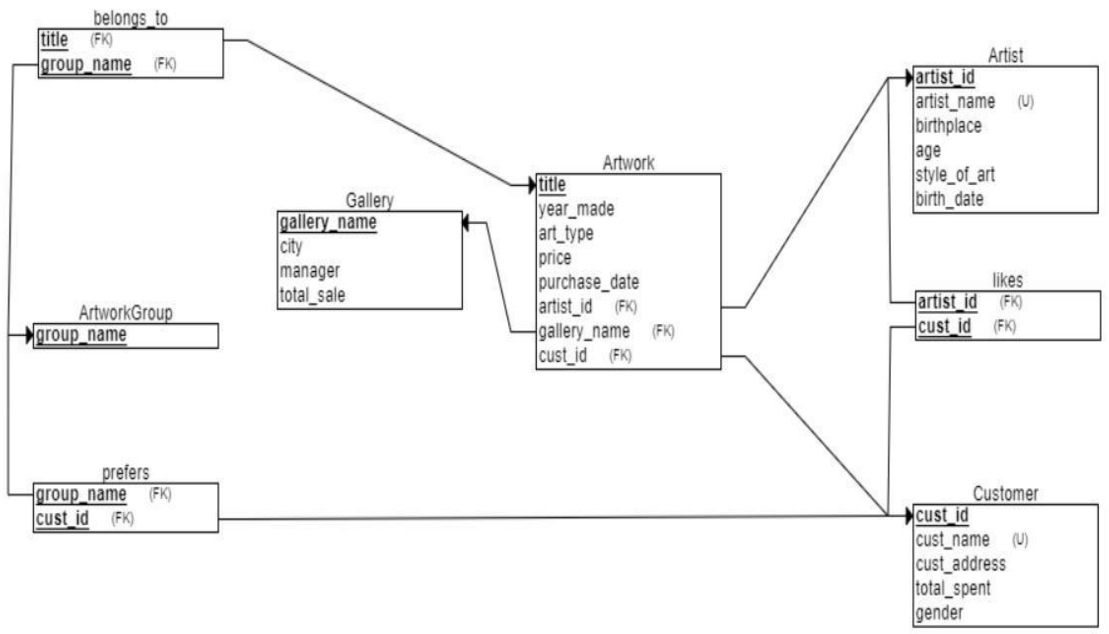


**Figure 2. ERD Schema**

Now we can identify each attribute's (column) datatype in the table below:

| Relation (Table) | Attributes (Columns) | Datatype | Comments |
|---|---|---|---|
| **Gallery** | gallery_name | VARCHAR | Primary Key |
| | city | VARCHAR | |
| | manager | VARCHAR | |
| | total_sale | NUMERIC | |
| **Artwork** | title | VARCHAR | Primary Key |
| | year_made | INT | |
| | art_type | VARCHAR | |
| | price | NUMERIC | |
| | purchase_date | DATE | |
| | artist_id | INT | Foreign Key for artist table |
| | gallery_name | VARCHAR | Foreign Key for galley table |
| | cust_id | INT | Foreign Key for customer table |
| **Artist** | artist_id | INT | Primary Key |
| | artist_name | VARCHAR | Unique |

| | | | |
|---|---|---|---|
| | birthplace | VARCHAR | |
| | birth_date | DATE | |
| | age | INT | |
| | style_of_art | VARCHAR | |
| **Customer** | cust_id | INT | Primary Key |
| | cust_name | VARCHAR | Unique |
| | cust_address | VARCHAR | |
| | total_spent | NUMERIC | |
| | gender | ENUM | ('M','F') |
| **ArtworkGroup** | group_name | VARCHAR | Primary Key |
| **Likes** | artist_id | INT | Foreign Key for artist table<br>Foreign Key for customer table<br>The combination of the two foreign keys is the table Primary key |
| | cust_id | INT | |
| **Prefers** | group_name | VARCHAR | Foreign Key for artworkgroup table<br>Foreign Key for customer table<br>The combination of the two foreign keys is the table Primary key |
| | cust_id | INT | |
| **Belongs_to** | title | VARCHAR | Foreign Key for artwork table<br>Foreign Key for artworkgroup table<br>The combination of the two foreign keys is the table Primary key |
| | group_name | VARCHAR | |

## 2.4. Structured Query Language (SQL) Implementation

SQL (Structured Query Language) is a common computer language used to administrate and work with relational databases. It is employed for database administration as well as functions which include data management, updating, and querying. Its simple and user-friendly syntax allows even non-technical users to interact with databases and retrieve data without having to write lengthy lines of code.

We have decided to implement the whole Relational Model in the database using MySQL for its Easy To Use, security and High Performance.

### Data Definition Language Queries

DDL queries were used to create all the relations in the database.

```
-- 1) creating a new database called "artbase"

CREATE DATABASE IF NOT EXISTS artbase;

-- 2) select "galleries" Database to work on

USE artbase;

-- 3) creating tables

-- 3.1) we start with tables that don't have any Foreign Keys (artist, gallery, artgroup,
customer)

-- Artist Relation

CREATE TABLE IF NOT EXISTS artist(

            artist_id INT NOT NULL AUTO_INCREMENT ,

  artist_name VARCHAR(100) UNIQUE,

  birthplace VARCHAR(50),

  birth_date DATE,

  age INT,

  style_of_art VARCHAR(50),

  primary key (artist_id)

);

-- Gallery Relation

CREATE TABLE IF NOT EXISTS gallery (

  gallery_name VARCHAR(40) NOT NULL,

  manager VARCHAR(40),

  city VARCHAR(55),

  total_sale NUMERIC(15,2),

  PRIMARY KEY (gallery_name)

);

-- Artgroup Relation
```

```
CREATE TABLE IF NOT EXISTS artgroup(
                         group_name VARCHAR(50) PRIMARY KEY
);
-- Customer Relation
CREATE TABLE IF NOT EXISTS customer (
  cust_id INT PRIMARY KEY AUTO_INCREMENT,
  cust_name VARCHAR(50) UNIQUE,
  cust_address VARCHAR(100),
  total_amount_spent DECIMAL(10, 2),
  gender ENUM('M','F')
);
-- alter the table to start Auto_Increment from 10001
ALTER TABLE customer AUTO_INCREMENT =10001;
-- 3.2) now, we create tables that have foreign keys (artwork, belongs_to, prefers, likes)
-- Artwork Relation
CREATE TABLE IF NOT EXISTS artwork (
  title VARCHAR(50) PRIMARY KEY ,
  year_made INT,
  art_type VARCHAR(50),
  price DECIMAL(10, 2),
  purchase_date DATE,
  artist_id INT,
  gallery_name VARCHAR(40),
  cust_id INT,
  FOREIGN KEY (artist_id) REFERENCES artist (artist_id) ON DELETE CASCADE,
  FOREIGN KEY (gallery_name) REFERENCES gallery (gallery_name) ON DELETE CASCADE,
  FOREIGN KEY (cust_id) REFERENCES customer (cust_id) ON DELETE CASCADE
```

```
);

-- Belongs_to Relation
CREATE TABLE IF NOT EXISTS belongs_to(
        title VARCHAR(50),
  group_name VARCHAR(50),
  FOREIGN KEY (title) REFERENCES artwork(title) ON DELETE CASCADE,
  FOREIGN KEY(group_name) REFERENCES artgroup(group_name) ON DELETE CASCADE
);

-- Prefers Relation
CREATE TABLE IF NOT EXISTS prefers(
        group_name VARCHAR(50),
  cust_id INT,
  FOREIGN KEY (group_name) REFERENCES artgroup(group_name) ON DELETE CASCADE,
  FOREIGN KEY(cust_id) REFERENCES customer(cust_id) ON DELETE CASCADE
);
-- Likes Relation
CREATE TABLE IF NOT EXISTS likes(
        artist_id INT,
  cust_id INT,
  FOREIGN KEY(artist_id) REFERENCES artist(artist_id) ON DELETE CASCADE,
  FOREIGN KEY(cust_id) REFERENCES customer(cust_id) ON DELETE CASCADE
);
```

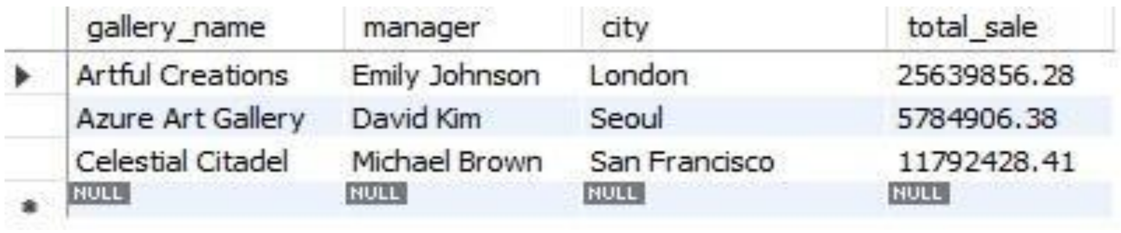
## Data Insertion Queries (part of DML)

The following queries were used to insert sample data into the database.

```
-- 4) Inserting Data into the above created Tables (Relations)
```

```
INSERT INTO gallery(gallery_name, manager, city, total_sale)
VALUES
('Gallery of Splendor', 'Hiroshi Tanaka', 'Tokyo', 10256320.62),
('Artful Creations', 'Emily Johnson', 'London', 25639856.28),
('Serenity Gallery','John Smith', 'New York', 12563475.36),
('Celestial Citadel', 'Michael Brown', 'San Francisco', 11792428.41),
('Sunset Gallery', 'Robert Williams', 'New York', 1730897.22),
('Urban Edge Gallery', 'Jessica Martinez','Chicago', 3706043.09),
('Spectrum Gallery', 'William Jones','New York', 14731834.01),
('Azure Art Gallery', 'David Kim', 'Seoul', 5784906.38),
('Tranquil Gallery', 'Ahmed Mahmoud' , 'Cairo', 1601705.32),
('Metropolitan Gallery', 'Isabella Rossi' , 'Paris', 12420081.76),
('Cityscape Art Gallery', 'Natasha Patel' , 'New Delhi', 4920438.18),
('Riverside Art Museum', 'Pablo Fernandez' , 'Sao Paulo', 11905084.52),
('Harmony Art Center', 'Elena Petrova' ,'Moscow', 19685478.7),
('Renaissance Gallery', 'Ingrid Johansson' , 'Istanbul', 9100296.74),
('Vivid Visions Gallery', 'Isabella Costa', 'Mexico City', 9645745.23),
('Solstice Gallery', 'James Anderson', 'New York', 11347987.5),
('Unity Art Studio', 'Sophie Dupont' , 'Paris', 7435281.48),
('Twilight Gallery', 'Maria Gonzale','Paris', 17418466.21),
('Dreamspace Art Gallery', 'Abdul Rahman', 'Cairo', 12638987.4),
('Elysium Art Gallery', 'Sarah Taylor', 'Manila', 8267463.13);
```

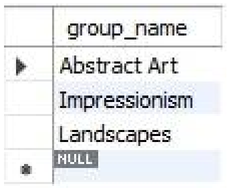
Sample:



| gallery_name | manager | city | total_sale |
|---|---|---|---|
| Artful Creations | Emily Johnson | London | 25639856.28 |
| Azure Art Gallery | David Kim | Seoul | 5784906.38 |
| Celestial Citadel | Michael Brown | San Francisco | 11792428.41 |
| NULL | NULL | NULL | NULL |

```
INSERT INTO artgroup

VALUES

('Portraits'),

('Landscapes'),

('Abstract Art'),

('Modern Art'),

('Street Art'),

('Impressionism'),

('Surrealism'),

('Minimalism'),

('Still Life');
```

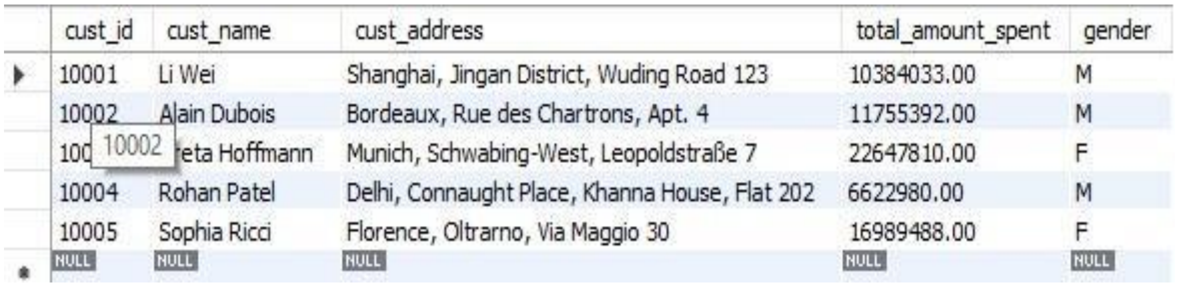
Sample:



| | group_name |
|---|---|
| ▶ | Abstract Art |
| | Impressionism |
| | Landscapes |
| * | NULL |

```
INSERT INTO customer(cust_name, cust_address, total_amount_spent, gender)

VALUES

('Li Wei', 'Shanghai, Jingan District, Wuding Road 123', 10384033,'M'),

('Alain Dubois', 'Bordeaux, Rue des Chartrons, Apt. 4', 11755392,'M'),

('Greta Hoffmann', 'Munich, Schwabing-West, Leopoldstraße 7', 22647810 , 'F'),

('Rohan Patel', 'Delhi, Connaught Place, Khanna House, Flat 202', 6622980, 'M'),

('Sophia Ricci', 'Florence, Oltrarno, Via Maggio 30', 16989488, 'F'),

('Haruka Tanaka ', 'Kyoto, Gion District, Hanami-koji Street', 854930, 'M'),

('Diego Hernandez', 'Oaxaca City, Centro Historico, Calle 20 de Noviembre 18', 21992792,
'M'),

('Aisha Abubakar', 'Abuja, Central Business District, Shehu Shagari Way', 21158076, 'F'),
('Ivan Petrov', 'St. Petersburg, Vasilievsky Island, Universitetskaya Embankment 15',
16259245, 'M'),
('Minseo Park', 'Busan, Haeundae District, Marine Drive 100', 11578684, 'M'),
('Carmen Garcia', 'Seville, Santa Cruz neighborhood, Calle Mateos Gago 5', 554097, 'F'),
('Chaiyo Wong', 'Chiang Mai, Old City, Soi Wat Chiang Man 2', 21438573, 'M'),
('Elif Demir', 'Ankara, Kavaklıdere, Adnan Menderes Boulevard 145', 16068919, 'M'),
('William Davies', 'Manchester, Northern Quarter, Tib Street 8', 20456277, 'M'),
('Sarah Miller', 'Los Angeles, Hollywood Hills, Sunset Boulevard 777', 19805663, 'F'),
('Mateo Torres', 'Mendoza City, Centro District, Calle Chile 1125', 24783806, 'M'),
('Chloe Jackson', 'Melbourne, Fitzroy, Brunswick Street 34', 23630422, 'F'),
('Felipe Santos', 'Salvador, Barra district, Avenida Oceânica 4000', 10110662, 'M'),
('Isabelle Tremblay', 'Quebec City, Petit Champlain district, Rue du Petit-Champlain 67',
14200263, 'F'),
('Omar Khalid ', 'Luxor, West Bank, Karnak Temple Road', 9864718, 'M'),
('Nikos Georgiou ', 'Mykonos Town, Chora, Petros the Pelican statue', 4298331, 'M'),
("Liam O'Connor", "Galway City, Quay Street, The King's Head Pub", 3547922, 'M')
;
```

## Sample:



| cust_id | cust_name | cust_address | total_amount_spent | gender |
|---|---|---|---|---|
| 10001 | Li Wei | Shanghai, Jingan District, Wuding Road 123 | 10384033.00 | M |
| 10002 | Alain Dubois | Bordeaux, Rue des Chartrons, Apt. 4 | 11755392.00 | M |
| 100 | eta Hoffmann | Munich, Schwabing-West, Leopoldstraße 7 | 22647810.00 | F |
| 10004 | Rohan Patel | Delhi, Connaught Place, Khanna House, Flat 202 | 6622980.00 | M |
| 10005 | Sophia Ricci | Florence, Oltrarno, Via Maggio 30 | 16989488.00 | F |
| NULL | NULL | NULL | NULL | NULL |

As we can't directly insert a calculated age using CURDATE() in a generated column due to its non-deterministic nature, we are going to use a trigger: This method involves creating a trigger that automatically calculates and inserts the age into a separate age column whenever a new row is inserted with a birth date.

```
DELIMITER //

CREATE TRIGGER update_age

BEFORE INSERT ON artist

FOR EACH ROW

SET NEW.age = FLOOR(DATEDIFF(CURDATE(), NEW.birth_date) / 365.25);

// DELIMITER ;
```

DELIMITER //: Sets a temporary delimiter for the trigger definition.

CREATE TRIGGER update_age: Defines a trigger named update_age.

BEFORE INSERT: Specifies the trigger fires before any insert operation on the table.

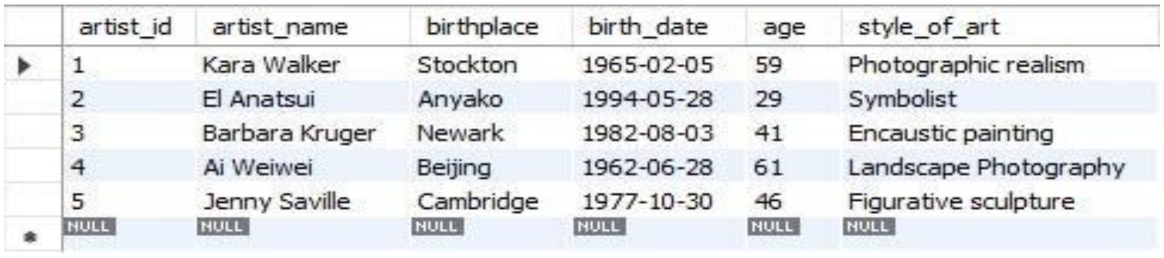
FOR EACH ROW: Indicates the trigger logic applies to each inserted row.

SET NEW.age = FLOOR(DATEDIFF(CURDATE(), NEW.birth_date) / 365.25): Calculates age using the birth_date in the new row (NEW) and sets the value in the age column.

// DELIMITER ;: Resets the delimiter back to the semicolon.

```
INSERT INTO artist(artist_name, birthplace, birth_date, style_of_art)

VALUES

('Kara Walker', 'Stockton', '1965-02-05', 'Photographic realism'),

('El Anatsui', 'Anyako', '1994-05-28', 'Symbolist'),

('Barbara Kruger', 'Newark', '1982-08-03', 'Encaustic painting'),

('Ai Weiwei', 'Beijing', '1962-06-28', 'Landscape Photography'),

('Jenny Saville', 'Cambridge', '1977-10-30', 'Figurative sculpture'),

('Theaster Gates Jr.', 'Chicago', '1973-01-18', 'Symbolist sculpture'),

('Beatriz Milhazes', 'Rio de Janeiro', '1961-02-28', 'Impressionism'),

('Shirin Neshat', 'Qazvin', '1957-04-18', 'Conceptual art'),

('Do Ho Suhi', 'Seoul', '1963-02-27', 'Modernism'),
('William Kentridge', 'Johannesburg', '1951-11-01', 'Expressionism'),
('Julie Mehretu', 'Addis Ababa', '2000-03-20', 'Photorealism'),
('Takashi Murakami', 'Tokyo', '1989-10-15', 'Post-Impressionism'),
('Lorna Simpson', 'Stockholm', '1987-09-30', 'Drip painting'),
('Jeff Koons', 'Mumbai', '1982-05-17', 'Photographic self-portraits'),
('Marina Abramovic', 'Belgrade', '1977-02-09', 'Realism'),
('Anish Kapoori', 'Bombay', '1954-03-27', 'Surrealism'),
('Glenn Ligoni', 'New York ', '1960-12-05', 'Figurative art'),
('Isadora Bloom', 'Florence', '1978-07-08', 'Conceptual art'),
('Milo Montague', 'Paris', '1985-06-16', 'Realism'),
('Luna Everhart', 'New York ', '1969-01-25', 'Modernism'),
('Jasper Wilde', 'Barcelona', '1973-05-20', 'Figurative art'),
('Esme Nightingale', 'Venice', '1988-04-12', 'Drip painting'),
('Felix Sterling', 'London', '1962-10-08', 'Realism'),
('Seraphina Rivers', 'Berlin', '1970-06-29', 'Figurative art'),
('Orion Grey', 'Amsterdam', '1976-08-04', 'Conceptual art'),
('Aurelia Frosti', 'Rome', '1982-12-07', 'Realism'),
('Dante Celestino', 'Kyoto', '1990-06-17', 'Landscape Photography'),
('Marcella Monroe', 'Prague', '1975-05-15', 'Photographic realism'),
('Atticus Stone', 'Moscow', '1981-09-01', 'Figurative art'),
('Serenity Sky', 'Buenos Aires', '1965-04-04', 'Realism'),
('Raphael Hawthorne', 'Los Angeles', '1992-08-09', 'Conceptual art'),
('Aurora Davenport', 'Vienna', '1968-12-30', 'Realism'),
('Silas Raines', 'Copenhagen', '1979-01-27', 'Landscape Photography')
;
```

## Sample:



| | artist_id | artist_name | birthplace | birth_date | age | style_of_art |
|---|---|---|---|---|---|---|
| ▶ | 1 | Kara Walker | Stockton | 1965-02-05 | 59 | Photographic realism |
| | 2 | El Anatsui | Anyako | 1994-05-28 | 29 | Symbolist |
| | 3 | Barbara Kruger | Newark | 1982-08-03 | 41 | Encaustic painting |
| | 4 | Ai Weiwei | Beijing | 1962-06-28 | 61 | Landscape Photography |
| | 5 | Jenny Saville | Cambridge | 1977-10-30 | 46 | Figurative sculpture |
| * | NULL | NULL | NULL | NULL | NULL | NULL |

INSERT INTO artwork (title, year_made, art_type, price, purchase_date, artist_id, gallery_name, cust_id)

VALUES

('Le Violon d''Ingres', 1986, 'Photograph', 1487000, '2009-05-19', 12, 'Vivid Visions Gallery', 10019),

('The Flatiron', 2007, 'Photograph', 31251000, '2015-11-23', 3, 'Serenity Gallery', 10002),

('Rhein II', 1978, 'Photograph', 8823000, '1988-02-09', 29, 'Cityscape Art Gallery', 10017),

('Spiritual America', 1993, 'Photograph', 25498000, '1999-09-07', 8, 'Harmony Art Center', 10019),

('Untitled #96', 1982, 'Photograph', 17662000, '2011-04-28', 20, 'Tranquil Gallery', 10022),

('Untitled #93', 2017, 'Photograph', 453900, '2019-12-27', 33, 'Unity Art Studio', 10013),

('To Her Majesty', 1967, 'Photograph', 1291700, '1997-07-14', 14, 'Celestial Citadel', 10007),

('Untitled (Cowboy)', 2001, 'Photograph', 29734000, '2007-01-02', 6, 'Twilight Gallery', 10013),

('Dead Troops Talk', 1991, 'Photograph', 6270000, '2005-10-21', 30, 'Artful Creations', 10012),

('Salvator Mundi', 1970, 'Painting', 20156000, '1978-06-05', 11, 'Gallery of Splendor', 10016),

('Les Demoiselles d''Avignon', 2014, 'Painting', 17982000, '2022-03-08', 25, 'Celestial Citadel', 10017),

('Guernica', 2004, 'Painting', 14503000, '2009-02-11', 9, 'Tranquil Gallery', 10022), ('Nafea Faa Ipoipo', 1973, 'Painting', 28765000, '1991-08-30', 33, 'Renaissance Gallery', 10021),

('Number 17A', 1988, 'Painting', 4791000, '2020-09-15', 19, 'Harmony Art Center', 10004),

('Interchange', 2016, 'Painting', 33510000, '2018-05-04', 4, 'Twilight Gallery', 10019),

('Three Studies for Lucian Freud II', 1994, 'Painting', 8244000, '2013-02-22', 21, 'Sunset Gallery', 10019),

('Nu couché (Reclining Nude)', 1984, 'Painting', 11425000, '1995-10-17', 10, 'Twilight Gallery', 10013),

('Head of a Woman (Dora Maar)', 1975, 'Sculpture (bronze)', 2749000, '1999-03-01', 31, 'Metropolitan Gallery', 10016),

('The Thinker', 1998, 'Sculpture (bronze)', 5179000, '2007-11-10', 16, 'Urban Edge Gallery', 10008),

('Balloon Dog (Orange)', 2011, 'Sculpture (stainless steel)', 2223300, '2017-08-24', 2, 'Renaissance Gallery', 10005),

('The Card Players (various works from the series)', 1969, 'Painting', 9407000, '2000-12-29', 23, 'Elysium Art Gallery', 10008),

('Schriempfer Rocks', 1985, 'Painting', 16896000, '2001-07-06', 26, 'Riverside Art Museum', 10017),

('The Kiss', 2002, 'Painting', 5921000, '2014-04-13', 5, 'Elysium Art Gallery', 10015),

('Water Lilies (various works from the series)', 1966, 'Painting', 30975000, '1994-01-12', 18, 'Serenity Gallery', 10020),

('False Start', 1972, 'Painting', 19685000, '1979-08-22', 22, 'Azure Art Gallery', 10004),

('Turquoise Marilyn', 2012, 'Painting', 23829000, '2013-09-20', 17, 'Serenity Gallery', 10021),

('Les Chats (The Cats)', 1996, 'Lithograph', 14716000, '2017-06-01', 19, 'Elysium Art Gallery', 10008),

('Drella (Andy Warhol)', 2006, 'Lithograph', 16842000, '2019-02-14', 1, 'Unity Art Studio', 10016), ('Marilyn (Campbell''s Soup Cans)', 1987, 'Lithograph', 8720000, '1994-12-31', 21, 'Renaissance Gallery', 10009),

('Bullfight No. 1', 1977, 'Lithograph', 27753000, '2010-07-16', 3, 'Sunset Gallery', 10014),

('Ballerina Turning in First Position', 2010, 'Lithograph', 32504000, '2020-04-27', 13, 'Metropolitan Gallery', 10006),

('Ferns (Series O)', 1971, 'Lithograph', 1982000, '1982-11-18', 6, 'Vivid Visions Gallery', 10018),

('Supplément au Voyage de Cook', 1999, 'Lithograph', 15738000, '2017-03-25', 15, 'Unity Art Studio', 10012),

('Landscape with Trees', 1980, 'Lithograph', 23697000, '2004-12-07', 22, 'Serenity Gallery', 10006),

('Popeye', 1968, 'Lithograph', 413000, '1988-09-21', 1, 'Sunset Gallery', 10001),

('Nu assis (Seated Nude)', 2019, 'Lithograph', 30120000, '2022-10-26', 7, 'Renaissance Gallery', 10010),

('Tree', 1992, 'Installation', 12255000, '2005-05-04', 22, 'Elysium Art Gallery', 10016),

('The Great Wave off Kanagawa', 1989, 'Woodblock print', 4161000, '1995-02-03', 2, 'Dreamspace Art Gallery', 10002),

('Serenity''s Embrace', 1976, 'Photograph', 28105000, '2007-08-17', 21, 'Riverside Art Museum', 10010),

('Midnight Sonata', 2009, 'Photograph', 7978000, '2011-03-04', 14, 'Riverside Art Museum', 10016),

('Whispering Winds', 2013, 'Painting', 2904000, '2019-11-09', 5, 'Spectrum Gallery', 10018),

('Enchanted Garden', 2000, 'Painting', 23167000, '2008-12-20', 11, 'Vivid Visions Gallery', 10014),

('Celestial Symphony', 1987, 'Painting', 18839000, '1997-04-08', 16, 'Twilight Gallery', 10008),

('Mystic Moonlight', 1999, 'Painting', 19192000, '2015-01-27', 2, 'Urban Edge Gallery', 10013), ('Radiant Reverie', 1974, 'Sculpture (bronze)', 8189000, '1996-10-05', 23, 'Elysium Art Gallery', 10005),

('Ethereal Elegance', 2005, 'Sculpture (stainless steel)', 1628000, '2017-07-18', 12, 'Serenity Gallery', 10021),

('Tranquil Oasis', 1981, 'Lithograph', 27207000, '1996-04-22', 3, 'Tranquil Gallery', 10005),

('Luminescent Lullaby', 2018, 'Painting', 6719000, '2021-09-01', 24, 'Unity Art Studio', 10003),

('Seraphic Serenade', 1979, 'Installation', 8935000, '1998-06-11', 8, 'Cityscape Art Gallery', 10006),

('Twilight Tapestry', 1990, 'Woodblock print', 9508000, '2015-02-18', 20, 'Cityscape Art Gallery', 10003),

('Aurora''s Dance', 2003, 'Painting', 13963000, '2007-09-28', 5, 'Gallery of Splendor', 10008),

('Harmonic Horizon', 1972, 'Painting', 11386000, '1999-12-14', 14, 'Unity Art Studio', 10022),

('Dreamweaver''s Delight', 1989, 'Lithograph', 15971000, '2003-08-31', 27, 'Dreamspace Art Gallery', 10006),

('Elysian Echoes', 1971, 'Photograph', 26829000, '1979-05-26', 28, 'Unity Art Studio', 10008),

('Enigmatic Enchantment', 2000, 'Painting', 10467000, '2021-08-10', 30, 'Tranquil Gallery', 10014),

('Solstice Sonata', 1983, 'Painting', 23484000, '2011-10-21', 32, 'Sunset Gallery', 10005),

('Whispering Willow', 2010, 'Sculpture (bronze)', 13297000, '2013-07-12', 33, 'Azure Art Gallery', 10021),

('Celestial Cascade', 1995, 'Lithograph', 23426000, '2000-09-09', 24, 'Metropolitan Gallery', 10019),

('Tranquil Tranquility', 1976, 'Sculpture (stainless steel)', 5731000, '1987-04-14', 32, 'Gallery of Splendor', 10011), ('Echoes of Elysium', 1992, 'Installation', 6675000, '1998-01-17', 1, 'Solstice Gallery', 10015),

('Midnight Melody', 1985, 'Woodblock print', 4842000, '2001-10-24', 3, 'Tranquil Gallery', 10007),

('Enchanted Evening', 2015, 'Lithograph', 19813000, '2023-11-15', 5, 'Serenity Gallery', 10004),

('Symphony of Serenity', 1968, 'Photograph', 3692000, '2000-03-22', 8, 'Metropolitan Gallery', 10019),

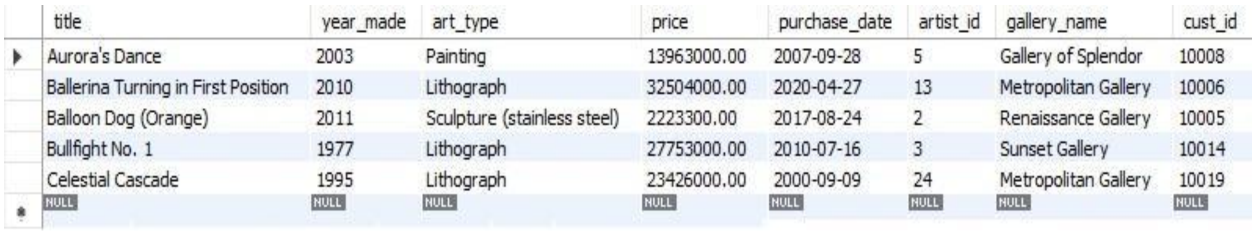
('Ethereal Dreamscape', 1980, 'Sculpture (bronze)', 9156000, '1999-10-08', 10, 'Twilight Gallery', 10012),

('Radiant Rhapsody', 1997, 'Sculpture (stainless steel)', 26478000, '2018-06-23', 20, 'Harmony Art Center', 10011),

('Cosmic Canvas', 1977, 'Installation', 10352000, '2012-01-30', 13, 'Spectrum Gallery', 10013),

('Echoes of Eden', 2019, 'Woodblock print', 1780000, '2021-05-12', 18, 'Dreamspace Art Gallery', 10011);

## Sample:



| title | year_made | art_type | price | purchase_date | artist_id | gallery_name | cust_id |
|---|---|---|---|---|---|---|---|
| Aurora's Dance | 2003 | Painting | 13963000.00 | 2007-09-28 | 5 | Gallery of Splendor | 10008 |
| Ballerina Turning in First Position | 2010 | Lithograph | 32504000.00 | 2020-04-27 | 13 | Metropolitan Gallery | 10006 |
| Balloon Dog (Orange) | 2011 | Sculpture (stainless steel) | 2223300.00 | 2017-08-24 | 2 | Renaissance Gallery | 10005 |
| Bullfight No. 1 | 1977 | Lithograph | 27753000.00 | 2010-07-16 | 3 | Sunset Gallery | 10014 |
| Celestial Cascade | 1995 | Lithograph | 23426000.00 | 2000-09-09 | 24 | Metropolitan Gallery | 10019 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

INSERT INTO belongs_to (title, group_name)

VALUES

('Solstice Sonata', 'Landscapes'),

('Interchange', 'Surrealism'),

('Serenity''s Embrace', 'Portraits'),

('Ballerina Turning in First Position', 'Portraits'),

```
('Rhein II', 'Surrealism'),
('Symphony of Serenity', 'Impressionism'),
('Le Violon d''Ingres', 'Minimalism'),
('Solstice Sonata', 'Impressionism'),
('Ethereal Elegance', 'Impressionism'),
('Head of a Woman (Dora Maar)', 'Impressionism'),
('Landscape with Trees', 'Abstract Art'),
('The Card Players (various works from the series)', 'Surrealism'),
('Dreamweaver''s Delight', 'Impressionism'),
('The Thinker', 'Portraits'),
('Midnight Melody', 'Street Art'),
('Tranquil Oasis', 'Surrealism'),
('Tree', 'Abstract Art'),
('Nu couché (Reclining Nude)', 'Minimalism'),
('Marilyn (Campbell''s Soup Cans)', 'Portraits'),
('Midnight Melody', 'Modern Art'),
('Whispering Winds', 'Street Art'),
('Luminescent Lullaby', 'Street Art'),
('Luminescent Lullaby', 'Impressionism'),
('Dreamweaver''s Delight', 'Minimalism'),
('Echoes of Elysium', 'Portraits'),
('Twilight Tapestry', 'Modern Art'),
('Tranquil Tranquility', 'Abstract Art'),
('Cosmic Canvas', 'Abstract Art'),
('Untitled #96', 'Minimalism'),
('Whispering Willow', 'Surrealism'),
('The Flatiron', 'Modern Art'),
('Dreamweaver''s Delight', 'Minimalism'),
('Symphony of Serenity', 'Abstract Art'),
('Seraphic Serenade', 'Portraits'),
('The Flatiron', 'Surrealism'),
('Aurora''s Dance', 'Abstract Art'),
('Schriempfer Rocks', 'Street Art'),
('Bullfight No. 1', 'Impressionism'),
('Spiritual America', 'Portraits'),
('Ethereal Dreamscape', 'Still Life'),
('Les Demoiselles d''Avignon', 'Portraits'),
('Radiant Rhapsody', 'Street Art'),
('Elysian Echoes', 'Portraits'),
('Untitled #93', 'Abstract Art'),
('Dead Troops Talk', 'Impressionism'),
('The Thinker', 'Impressionism'),
('Enchanted Evening', 'Surrealism'),
('Bullfight No. 1', 'Abstract Art'),
('Mystic Moonlight', 'Street Art'),
('Head of a Woman (Dora Maar)', 'Abstract Art'),
('Popeye', 'Landscapes'),
('Water Lilies (various works from the series)', 'Abstract Art'),
('Landscape with Trees', 'Portraits'),
('Number 17A', 'Abstract Art'),
('Untitled #96', 'Minimalism'),
('Ballerina Turning in First Position', 'Abstract Art'),
('Midnight Sonata', 'Portraits'),
('Harmonic Horizon', 'Modern Art'),
('Aurora''s Dance', 'Street Art'),
('Midnight Melody', 'Impressionism'),
('Enigmatic Enchantment', 'Abstract Art'),
('To Her Majesty', 'Abstract Art'),
('Turquoise Marilyn', 'Minimalism'),
('Salvator Mundi', 'Still Life'),
('Tree', 'Surrealism'),
('Water Lilies (various works from the series)', 'Still Life'),
('Enigmatic Enchantment', 'Street Art'),
('Radiant Reverie', 'Surrealism'),
('Dreamweaver''s Delight', 'Street Art'),
('Celestial Symphony', 'Street Art'),
('Aurora''s Dance', 'Portraits'),
('Untitled #93', 'Abstract Art'),
('The Flatiron', 'Still Life'),
('Twilight Tapestry', 'Landscapes'),
('The Flatiron', 'Abstract Art'),
('Untitled (Cowboy)', 'Still Life'),
('Guernica', 'Impressionism'),
('Echoes of Eden', 'Surrealism'),
('The Kiss', 'Abstract Art'),
```

('Ferns (Series O)', 'Street Art'),

('Les Chats (The Cats)', 'Minimalism'),

('Celestial Cascade', 'Impressionism'),

('Drella (Andy Warhol)', 'Minimalism'),

('Supplément au Voyage de Cook', 'Still Life'),

('Balloon Dog (Orange)', 'Modern Art'),

('The Great Wave off Kanagawa', 'Abstract Art'),

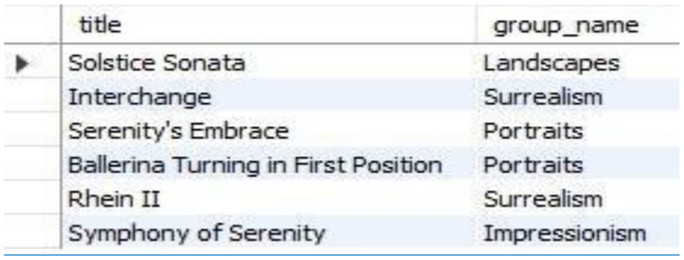
('Three Studies for Lucian Freud II', 'Minimalism'),

('Enchanted Garden', 'Abstract Art'),

('Nu assis (Seated Nude)', 'Minimalism'),

('False Start', 'Modern Art');

## Sample:



| title | group_name |
|---|---|
| Solstice Sonata | Landscapes |
| Interchange | Surrealism |
| Serenity's Embrace | Portraits |
| Ballerina Turning in First Position | Portraits |
| Rhein II | Surrealism |
| Symphony of Serenity | Impressionism |

INSERT INTO Likes (artist_id, cust_id)

VALUES

 (28, 10005),

 (33, 10001),

 (33, 10003),

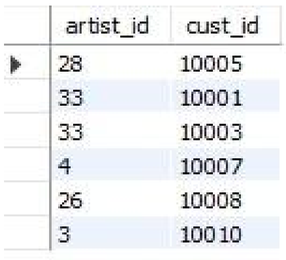
 (4, 10007),

 (26, 10008),

 (3, 10010),

```
(15, 10014),
(13, 10001),
(23, 10015),
(19, 10003),
(8, 10015),
(23, 10016),
(16, 10021),
(9, 10011),
(5, 10020),
(17, 10011),
(28, 10003),
(30, 10002),
(2, 10014),
(4, 10001),
(7, 10002),
(15, 10022),
(7, 10003),
(16, 10002),
(10, 10020),
(4, 10004),
(2, 10013),
(16, 10013),
(2, 10005),
(21, 10007),
(29, 10009);
```

## Sample:



| artist_id | cust_id |
|---|---|
| 28 | 10005 |
| 33 | 10001 |
| 33 | 10003 |
| 4 | 10007 |
| 26 | 10008 |
| 3 | 10010 |

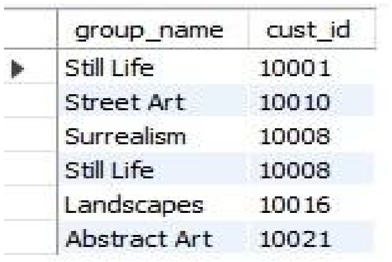
```
INSERT INTO Prefers (group_name, cust_id)

VALUES

 ("Still Life", 10001),

 ("Street Art", 10010),

 ("Surrealism", 10008),

 ("Still Life", 10008),

 ("Landscapes", 10016),

 ("Abstract Art", 10021),

 ("Impressionism", 10022),

 ("Street Art", 10004),

 ("Impressionism", 10004),

 ("Impressionism", 10020),

 ("Minimalism", 10014),

 ("Portraits", 10011),

 ("Surrealism", 10019),

 ("Minimalism", 10012),

 ("Landscapes", 10009),

 ("Impressionism", 10001),
```

("Abstract Art", 10003),

("Modern Art", 10017),

("Minimalism", 10002);

Sample:



| group_name | cust_id |
|---|---|
| Still Life | 10001 |
| Street Art | 10010 |
| Surrealism | 10008 |
| Still Life | 10008 |
| Landscapes | 10016 |
| Abstract Art | 10021 |

### 2.4.1. Database Normalisation

As stated in the previous section, for SQL, we decided to implement the whole Relational Model rather than a subset of it. This was done to help us create more complex DDL and DML queries and to benefit from the powerful querying capabilities of SQL that allow for complex and precise data retrieval. This is particularly valuable for operations that involve multi-table joins and transactions that need to reflect multiple related data items simultaneously (such as retrieving data from two tables that are not directly connected by joining with another table that have relationship with both).

### 2.4.2. SQL Test Cases

We tried to select our test cases from a business perspective such that each query answers some important business question or reveals an insight that shareholders or upper management of the ArtBase company may request.

#### *2.4.2.1 Test Case 1: Customer favourite artists*

Business Benefit

This is a useful piece of information for the galley in order to target the customers who like a particular artist, hence increasing the probability of selling the artwork and maximizing their profit by offering them early access to purchase the artwork.

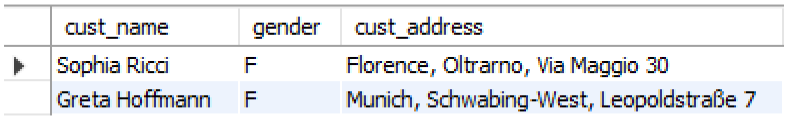
QUERY

1) A new artwork titled 'Elysian Echoes' is available for sale, it was made by artist "Marcella Monroe" and the gallery wants to know the names and address of the

customers who like this artist's style so they can contact them to check if they are interested in buying it.

```
SELECT  c.cust_name, c.gender, c.cust_address
 FROM artwork aw
 JOIN artist a
 ON aw.artist_id = a.artist_id
 JOIN likes l
 On A.artist_id = l.artist_id
 JOIN customer c
 ON l.cust_id =c.cust_id
 WHERE a.artist_name = "Marcella Monroe";
```

Output



| cust_name | gender | cust_address |
|---|---|---|
| Sophia Ricci | F | Florence, Oltrarno, Via Maggio 30 |
| Greta Hoffmann | F | Munich, Schwabing-West, Leopoldstraße 7 |

The output shows that there are 2 customers that most likely will be interested in buying the artwork.

#### *2.4.2.2 Test Case 2: Customers favorite artgroups*

Business Benefit

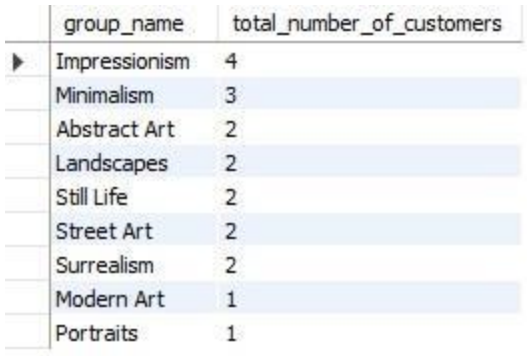
This reveals a useful insight into the most loved artgroup among the existing customers. Galleries then  can use this insight to increase the number of artworks that belong to these artgroup.

QUERY

2) The company wants to check which art group is the most preferred by customers

```
SELECT group_name , COUNT(cust_id) AS total_ number_of_customers,
 FROM prefers
 GROUP BY group_name
 ORDER BY COUNT(cust_id) DESC;
```

Output



| group_name | total_number_of_customers |
|---|---|
| Impressionism | 4 |
| Minimalism | 3 |
| Abstract Art | 2 |
| Landscapes | 2 |
| Still Life | 2 |
| Street Art | 2 |
| Surrealism | 2 |
| Modern Art | 1 |
| Portraits | 1 |

The output is ordered in a descending matter to make it easier to interpret the result. The results show that the most purchased artwork belong to 'Impressionism' artgroup with total of 4 purchase followed by 'Minimalism' total of 3 purchase, so the galleries can focus on obtaining more artwork from these 2 art groups.

#### *2.4.2.3 Test Case 3: View to show Artworks in each gallery*

Business Benefit

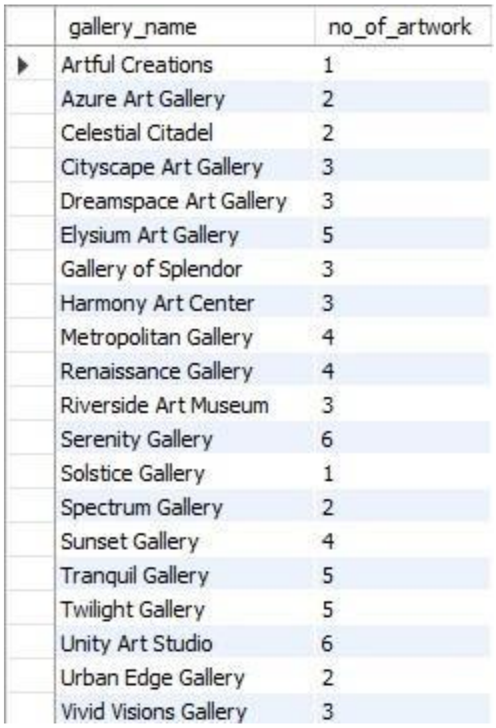
By creating a view for showing the total number of artworks each gallery in the database has, it makes accessing this information easy for the shareholders or the galleries managers to keep track of each gallery inventory without the need to have deep knowledge in SQL.

QUERY

3) Create a View to get the number of artworks in each gallery. (A view is essentially a stored query accessible as a virtual table in a relational database that does not require any storage of its own)

```
CREATE OR REPLACE VIEW galley_artworks(gallery_name, no_of_artwork) as select
g.gallery_name, COUNT(aw.title)
from gallery g, artwork aw
WHERE g.gallery_name = aw.gallery_name
group by gallery_name
ORDER BY gallery_name ASC;
```

Output



| gallery_name | no_of_artwork |
|---|---|
| Artful Creations | 1 |
| Azure Art Gallery | 2 |
| Celestial Citadel | 2 |
| Cityscape Art Gallery | 3 |
| Dreamspace Art Gallery | 3 |
| Elysium Art Gallery | 5 |
| Gallery of Splendor | 3 |
| Harmony Art Center | 3 |
| Metropolitan Gallery | 4 |
| Renaissance Gallery | 4 |
| Riverside Art Museum | 3 |
| Serenity Gallery | 6 |
| Solstice Gallery | 1 |
| Spectrum Gallery | 2 |
| Sunset Gallery | 4 |
| Tranquil Gallery | 5 |
| Twilight Gallery | 5 |
| Unity Art Studio | 6 |
| Urban Edge Gallery | 2 |
| Vivid Visions Gallery | 3 |

As the output is generated from a VIEW, it can be accessed directly in the DBMS without the need to write any query. The output shows a list of all galleries in ASCENDING order (ordered alphabetically) and the total number of artworks each gallery has.

#### *2.4.2.4 Test Case 4: Customers spending more money*

##### Business Benefit

Getting insight into the total number of customers who spend more than 20,000,000 (USD) purchasing any artworks and one of those artworks was created by artist ‘Barbara Kruger’.

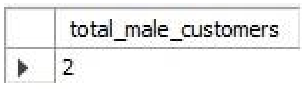
##### QUERY

5) find total number of Male customers who bought artwork of 'Barbara Kruger' and their total spend is more than 20,000,000

```
SELECT
  COUNT(DISTINCT c.cust_id) AS total_male_customers
FROM
  customer c
  JOIN artwork aw ON c.cust_id = aw.cust_id
  JOIN artist a ON a.artist_id = aw.artist_id
WHERE
```

```
 c.gender = 'M'
AND c.total_amount_spent > 20000000
 AND aw.artist_id = (
  SELECT
    artist_id
  FROM
    artist
  WHERE
    artist_name = 'Barbara Kruger'
 );
```

### Output



```
total_male_customers
2
```

In the query above , we utilize subquery which greatly enhances the flexibility and functionality of SQL operations.

The output shows that there are a total of 2 customers who spend more than 20,000,000 (USD) purchasing any artworks and one of those artworks was created by artist 'Barbara Kruger'.

### 2.4.2.5 Test Case *5: Purchases analysis based on gender*

#### Business Benefit

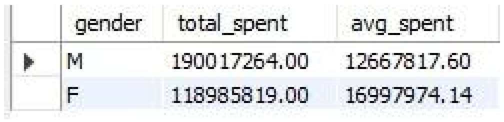
Getting insight into the total and average spending of all existing customers based on their gender. This information will help shareholders and upper management better understand the distribution of the male and female customers who bought any artwork from any gallery and the average each gender is spending.

#### QUERY

6) Based on gender, who spent more money in total and on average?

```
SELECT gender, ROUND(SUM(total_amount_spent),2) AS total_spent,
ROUND(AVG(total_amount_spent),2) AS avg_spent
FROM customer
GROUP BY gender
ORDER BY total_spent DESC;
```

Output

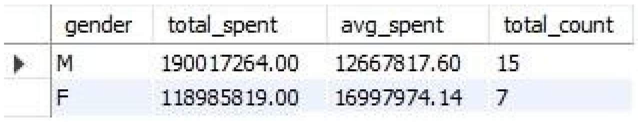


| | gender | total_spent | avg_spent |
|---|---|---|---|
| ▶ | M | 190017264.00 | 12667817.60 |
| | F | 118985819.00 | 16997974.14 |

The output shows that the total amount spent by male customers is higher than the female customers. On the other hand, the average female customers spending is slightly higher than the male customers. If we want to dig deeper to understand this results , then we have to get the total number of Male Customers and compare it to female ones. We can get this information by retrieving the total number of customers based on their gender by modifying the above query :

```
SELECT gender, ROUND(SUM(total_amount_spent),2) AS total_spent, ROUND(AVG(total_amount_spent),2) AS avg_spent, COUNT('gender') AS total_count

FROM customer

GROUP BY gender

ORDER BY total_spent DESC;
```

Output



| | gender | total_spent | avg_spent | total_count |
|---|---|---|---|---|
| ▶ | M | 190017264.00 | 12667817.60 | 15 |
| | F | 118985819.00 | 16997974.14 | 7 |

From the above results we can see that the number of male customers is more than double the number of female customers. This clearly shows that female customers are spending more money on purchasing artwork than male customers do.

## 2.5. MongoDB Implementation

MongoDB is a highly versatile and scalable NoSQL database designed for modern applications that manage extensive and varied datasets. MongoDB stores data in flexible, JSON-like documents, unlike traditional relational databases that use structured tables. This approach supports complex, hierarchical data models and allows different data schemas within the same collection, offering significant flexibility and ease of use. MongoDB's dynamic schema and robust querying capabilities enable rapid development and adaptation, accommodating evolving data needs with minimal effort. This makes it a popular choice across various industries where scalability and efficient data handling are essential, allowing organizations to fully harness their data's power.

To implement a selection of important entities from your relational schema in MongoDB, we need to take a different approach than what you'd use for a relational database. MongoDB is a document-oriented NoSQL database, so the implementation will involve designing documents and collections that represent the relationships and entities effectively.

**Entities to Implement**

Let's choose three important entities:

**Artist**

**Artwork**

**Customer**

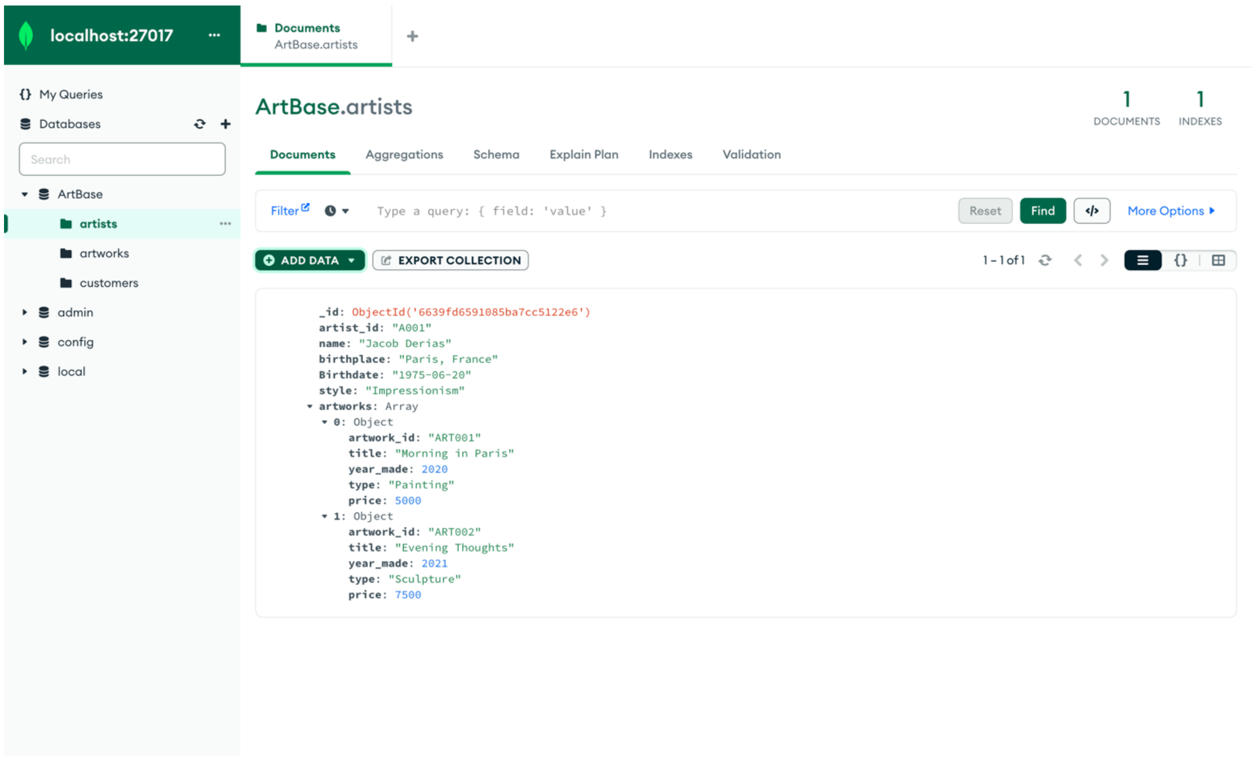
These three entities are central to the use case, covering the core aspects of the art management system.

**1. Artist Collection**

Artist documents will store information about the artists, including a nested array for artworks to directly associate artists with their creations.

Sample Schema:

```
{
 "_id": ObjectId(),
 "artist_id": "A001",
 "name": "Jacob Derias",
 "birthplace": "Paris, France",
 "birthdate": "1975-06-20",
 "style": "Impressionism",
 "artworks": [
  {
   "artwork_id": "ART001",
   "title": "Beautiful morning in bournemouth",
   "year_made": 2020,
   "type": "Painting",
   "price": 5000
  },
  {
   "artwork_id": "ART002",
   "title": "Evening Thoughts",
   "year_made": 2021,
   "type": "Sculpture",
   "price": 7500
  }
 ]
}
```



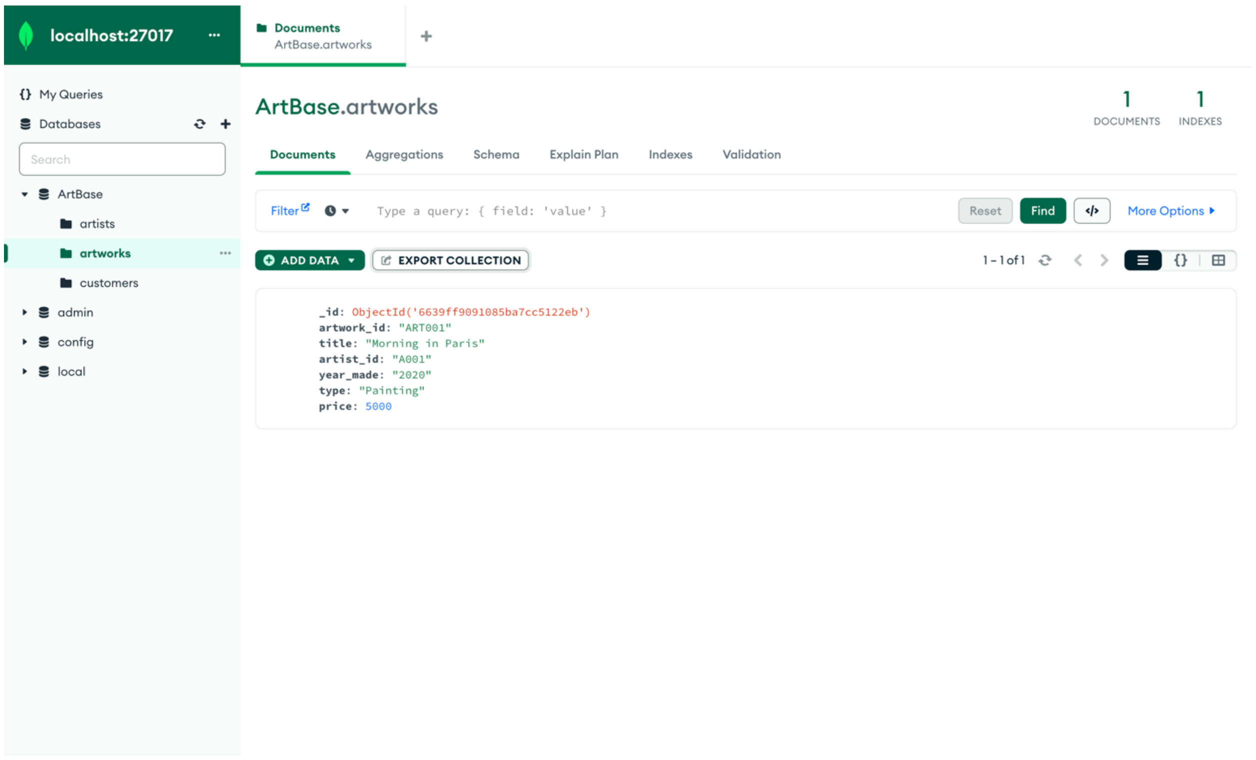
```
_id: ObjectId('6639fd6591085ba7cc5122e6')
artist_id: "A001"
name: "Jacob Derias"
birthplace: "Paris, France"
Birthdate: "1975-06-20"
style: "Impressionism"
artworks: Array
  0: Object
    artwork_id: "ART001"
    title: "Morning in Paris"
    year_made: 2020
    type: "Painting"
    price: 5000
  1: Object
    artwork_id: "ART002"
    title: "Evening Thoughts"
    year_made: 2021
    type: "Sculpture"
    price: 7500
```

## 2. Artwork Collection

Artwork documents will store information about artworks, linked to artists through artist_id.

Sample Schema:

```
{
 "_id": ObjectId(),
 "artwork_id": "ART001",
 "title": " Beautiful Morning in Bournemouth ",
 "artist_id": "A001",
 "year_made": 2020,
 "type": "Painting",
 "price": 5000,
```



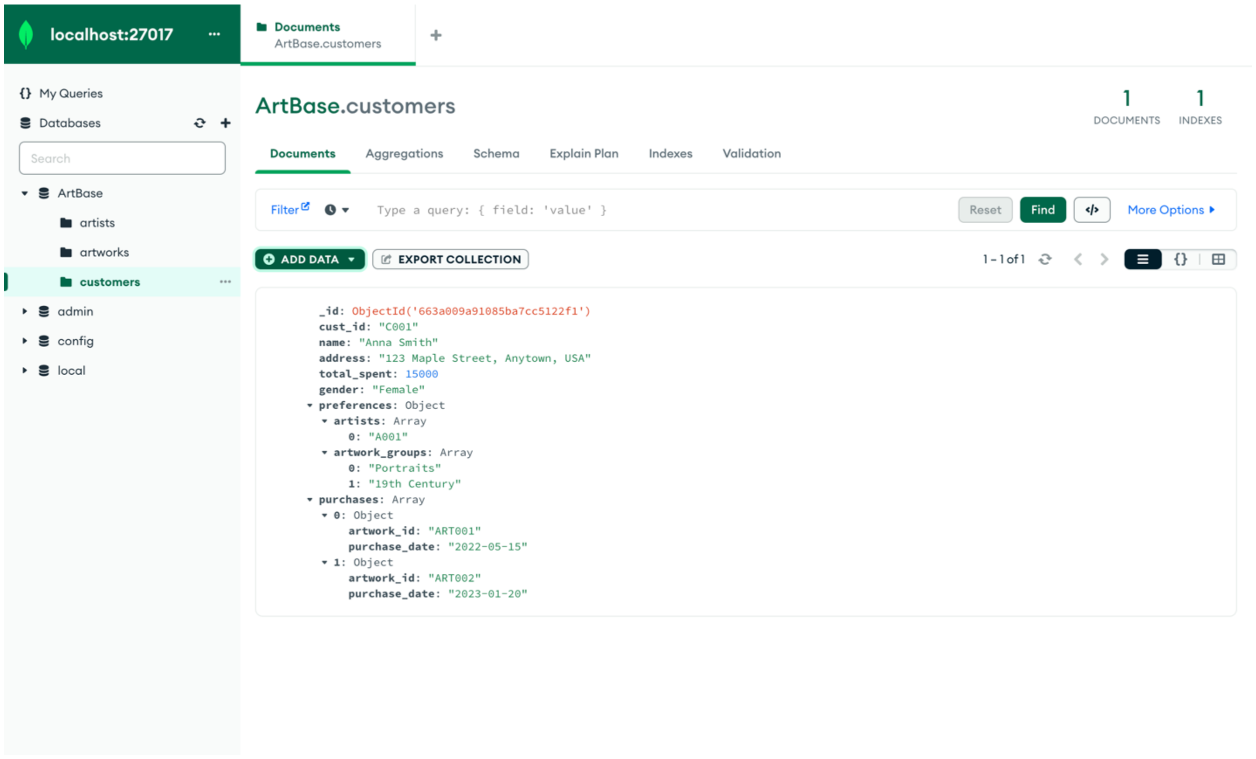
```
_id: ObjectId('6639ff9091085ba7cc5122eb')
artwork_id: "ART001"
title: "Morning in Paris"
artist_id: "A001"
year_made: "2020"
type: "Painting"
price: 5000
```

**Customer Collection**

Customer documents will include personal information as well as a list of purchased artworks, utilising references to artwork IDs, and preferences towards certain artists or artwork groups.

Sample Schema:

```
{
 "_id": ObjectId(),
 "cust_id": "C001",
 "name": "Anna Smith",
 "address": "93 wall Street, New York, USA ",
 "total_spent": 15000,
 "gender": "Female",
  "preferences": {
    "artists": ["A001"],
    "artwork_groups": ["Portraits", "19th Century"]
  },
  "purchases": [
    {
      "artwork_id": "ART001",
      "purchase_date": "2022-05-15"
    },
    {
      "artwork_id": "ART002",
      "purchase_date": "2023-01-20"
    }
  ]
}
```



```
_id: ObjectId('663a009a91085ba7cc5122f1')
cust_id: "C001"
name: "Anna Smith"
address: "123 Maple Street, Anytown, USA"
total_spent: 15000
gender: "Female"
preferences: Object
  artists: Array
    0: "A001"
  artwork_groups: Array
    0: "Portraits"
    1: "19th Century"
purchases: Array
  0: Object
    artwork_id: "ART001"
    purchase_date: "2022-05-15"
  1: Object
    artwork_id: "ART002"
    purchase_date: "2023-01-20"
```

```
# Connect to the MongoDB instance

mongo


use ArtBase DB


# Insert an artist document with embedded artworks

db.artists.insertOne({

  "artist_id": "A001",

  "name": "Jacob Derias",

  "birthplace": "Paris, France",

  "birthdate": "1975-06-20",

  "style": "Impressionism",
 "artworks": [
  {
   "artwork_id": "ART001",
   "title": "Beautiful Morning in Bournemouth",
   "year_made": 2020,
   "type": "Painting",
   "price": 5000
  },
  {
   "artwork_id": "ART002",
   "title": "Evening Thoughts",
   "year_made": 2021,
   "type": "Sculpture",
   "price": 7500
  }
 ]
});

# Insert a customer document with references to purchases
db.customers.insertOne({
 "cust_id": "C001",
 "name": "Anna Smith",
 "address": "93 wall Street, New York, USA",
 "total_spent": 15000,
 "gender": "Female",
 "preferences": {
  "artists": ["A001"],
  "artwork_groups": ["Portraits", "19th Century"]
 },
 "purchases": [
  {
   "artwork_id": "ART001",
   "purchase_date": "2022-05-15"
  },
  {
   "artwork_id": "ART002",
   "purchase_date": "2023-01-20"
  }
 ]
});
```

The selection of entities for MongoDB implementation—Artist, Artwork, and Customer—was influenced by critical factors that match the database's strengths and the particular requirements of the ArtBase management system. Here's a detailed rationale for selecting these entities:

## 1. Core Functional Requirements

These three entities are central to the fundamental operations and purposes of an art gallery management system:

**Artist:** Central to any art gallery, artists are the creators of artworks. Storing detailed information about artists, including a direct association with their artworks, supports functionalities such as cataloging, artist profiles, and direct attribution in sales and exhibits.

**Artwork:** As the primary product of an art gallery, managing artwork details efficiently is crucial. This includes tracking each piece's creation, categorization, and sales. Effective management of artworks directly impacts inventory management, sales tracking, and customer recommendations.

**Customer:** Customers are key to the business operations of a gallery. Managing customer profiles, their purchase history, and preferences allows for tailored marketing, customer engagement, and sales strategy development.

## 2. Using MongoDB's Strengths in Handling Relationships

The nature of MongoDB's document model is particularly well-suited to handling the complex and often nested relationships that these entities exhibit:

**Embedded Documents:** MongoDB excels at handling embedded documents, which is ideal for associating artists with their artworks. This not only simplifies queries (retrieving all artworks for an artist in a single query) but also enhances performance by reducing the need for joins.

**Reference Links:** For entities where relationships might become too cumbersome to manage through embedding—like customers who might have extensive purchase histories or varied preferences—MongoDB's reference pattern is used. This allows efficient management without overloading single documents.

## 3. Scalability and Flexibility

MongoDB offers scalability and flexibility which is crucial for dynamically changing data:

**Schema Flexibility:** Art galleries might expand their data collection over time (e.g., adding virtual tours, online sales channels). MongoDB's schema-less nature allows for easy modifications and additions to entity attributes without significant backend overhaul or downtime.

**Scalable Reads and Writes:** Given that galleries might experience high read and write loads during events, sales, or exhibits, MongoDB can handle these with its efficient indexing and shading capabilities.

## 4. Real-World Application and Query Efficiency

Implementing these entities in MongoDB also considers the real-world usage and query patterns:

**Complex Queries**: MongoDB's aggregation framework allows for complex queries across these entities, making it possible to generate sophisticated reports and insights (e.g., total sales per artist, most popular art styles among customers).

**Operational Efficiency:** Day-to-day operations in a gallery, such as quickly updating artwork records post-sale or updating artist profiles, are more streamlined with MongoDB's document model.

Below are five test cases designed to demonstrate the practical usage of the MongoDB collections we've implemented for the art management system. These test cases cover a range of typical operations an art gallery might perform using the database. For each test case, I'll provide the MongoDB query code, explain the use case, and describe the expected output.

## 2.5.1. MongoDB Test Cases

### *2.5.1.1. Test Case 1: Find All Artworks by a Specific Artist*

**Purpose:** Retrieve all artworks created by an artist named "Jacob Derias".

**Query:**

```
db.artists.find(
 { name: "Jacob Derias" },
 { artworks: 1, _id: 0 }
)
```

**Expected Output:**

This query will return the list of all artworks created by "Jacob Derias", including details like title, year made, type, and price.

### *2.5.1.2. Test Case 2: Update Artwork Price*

**Purpose:** Update the price of an artwork titled " Beautiful Morning in Bournemouth " to $5500.

**Query:**

```
db.artists.updateOne(
 { "artworks.title": " Beautiful Morning in Bournemouth " },
 { $set: { "artworks.$.price": 5500 } }

)
```

**Expected Output:**

This query updates the price of the specific artwork. The output would be a success message indicating that the update was made successfully (e.g., acknowledged: true, modifiedCount: 1).

#### *2.5.1.3. Test Case 3: Find Customers Interested in a Specific Artwork Group*

**Purpose:** Retrieve all customers who have a preference for "Portraits".

**Query:**

```
db.customers.find(

 { "preferences.artwork_groups": "Portraits" },

 { name: 1, _id: 0 }

)
```

**Expected Output:**

This query will list all customers who prefer artworks classified under "Portraits". It will return their names.

#### *2.5.1.4. Test Case 4: Add a New Artwork to an Artist's Profile*

**Purpose:** Add a new artwork titled "Sunset Boulevard" to artist "Jacob Derias".

**Query:**

```
db.artists.updateOne(

 { name: "Jacob Derias" },

 { $push: { artworks: { artwork_id: "ART003", title: "Sunset Boulevard", year_made: 2023,
type: "Painting", price: 7000 } } }

)
```

**Expected Output:**

This query will add a new artwork under Jacob Derias's list of artworks. The output will confirm that the document was updated successfully.

#### *2.5.1.5. Test Case 5: List All Purchases Made by a Customer*

**Purpose:** List all artworks purchased by a customer named "Anna Smith".

**Query:**

```
db.customers.aggregate([
 { $match: { name: "Anna Smith" } },
 { $unwind: "$purchases" },
 { $lookup: {
   from: "artists",
   localField: "purchases.artwork_id",
   foreignField: "artworks.artwork_id",
   as: "artwork_details"
 }},
 { $unwind: "$artwork_details" },
 { $unwind: "$artwork_details.artworks" },
 { $match: { "artwork_details.artworks.artwork_id": { $eq: "$purchases.artwork_id" } } },
 { $project: { _id: 0, title: "$artwork_details.artworks.title", purchase_date:
"$purchases.purchase_date", price: "$artwork_details.artworks.price" } }
])
```

**Expected Output:**

This query provides a detailed list of all artworks purchased by "Anna Smith", including the title, purchase date, and price of each artwork. This involves joining data across documents, which is done here using the $lookup stage in the aggregation pipeline.

These queries should help demonstrate the functionality of MongoDB setup for managing an art gallery's data. They include operations like reading, updating, and complex joins, which are common requirements for database systems in similar applications.

## 2.6. NEO4J Implementation

After implementing our use case in the table-based databases above, we will now use Neo4j to do the same.

Unlike SQL and MongoDB, Neo4j is a graph-based database. The main difference being in the way that entities and relationships are represented. SQL and MongoDB represent them using lines in a table while Neo4j uses nodes to represent entities. The relationships between the nodes are represented using lines, this way of viewing makes it easier to visualise complex relationships and interactions in the data.

To represent the data in Neo4j, csv files were used. The 'LOAD CSV WITH HEADERS' command was used to load the data into Neo4j. This command takes every row in a csv file and turns it into a node. The use of csv files to load the data allows for multiple nodes of the same type to be loaded in at the same time. The alternative would be to write a CREATE command for every node. This would not only be time consuming, but also is an error prawn process. The following nodes were selected for implementation in Neo4j:

- Artist
- Artwork
- Gallery
- Customers

The rationale behind choosing these entities is the fact that any art gallery revolves around these four entities. They are interconnected and form meaningful relationships. This means that we can make use of the graph-based nature of Neo4j to get a sound understanding of different artists, the artwork they create and the gallery to which they belong.

The following code shows how the nodes were created in Neo4j:

```
LOAD CSV WITH HEADERS FROM 'file:///artist.csv' AS row
MERGE (a:Artist {artist_id: toInteger(row.artist_id),
        artist_name: row.artist_name,
        birthplace: row.birthplace,
        birth_date: row.birth_date,
        age: row.age, style: row.style})
RETURN count(a);
```

```
LOAD CSV WITH HEADERS FROM 'file:///artwork.csv' AS row
MERGE (a:Artwork {title: row.title, artist_name: row.artist_name,
        yaer_made: row.yaer_made,
        art_type: row.art_type,
        price: row.price,
        purchase_date: row.purchase_date,
        artist_id: toInteger(row.artist_id),
        gallery_name: row.gallery_name,
        cust_id :toInteger(row.cust_id) })
RETURN count(a);
```

```
LOAD CSV WITH HEADERS FROM 'file:///gallery.csv' AS row
MERGE (g:Gallery {gallery_name: row.gallery_name,
        manager: row.manager,
        city: row.city,
        total_sale: row.total_sale})
RETURN count(g);
```

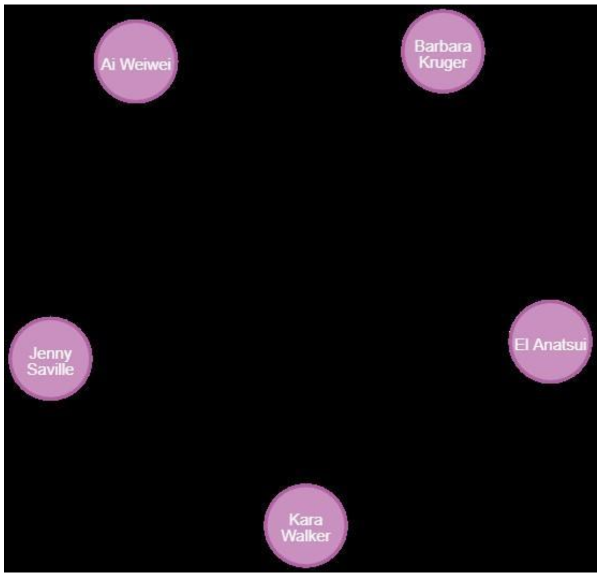
```
LOAD CSV WITH HEADERS FROM 'file:///customer.csv' AS row
MERGE (c:Customer {cust_id: toInteger(row.cust_id),
                    cust_name: row.cust_name,
                    cust_address: row.cust_address,
                    total_amount_spent: row.total_amount_spent,
                    gender: row.gender})
RETURN count(c);
```

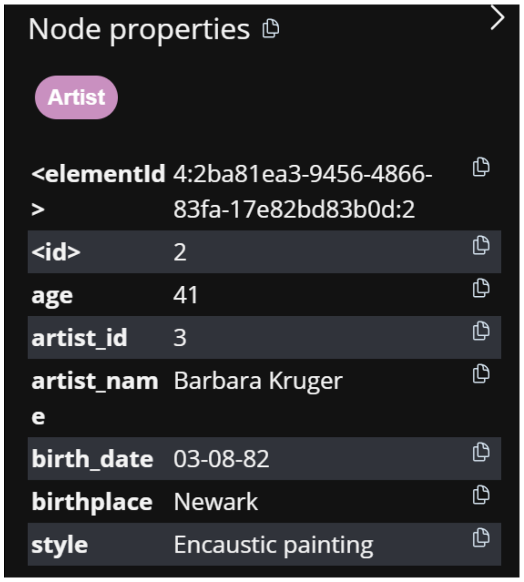
**Data Visualisation**

Each entity has many rows, so we will visualise five of each for the sake of simplicity. The following will showcase the query used to visualise the entity as well as a table representation showing the different properties for each entity.

To visualise artists:

```
MATCH (a:Artist)
RETURN a
LIMIT 5
```



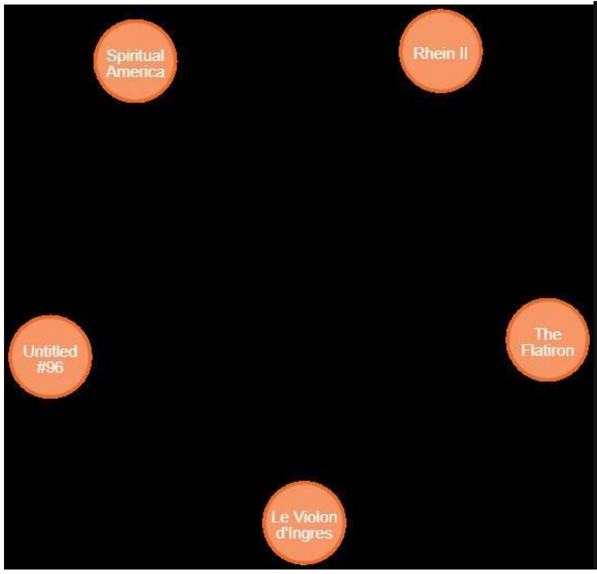

Node properties

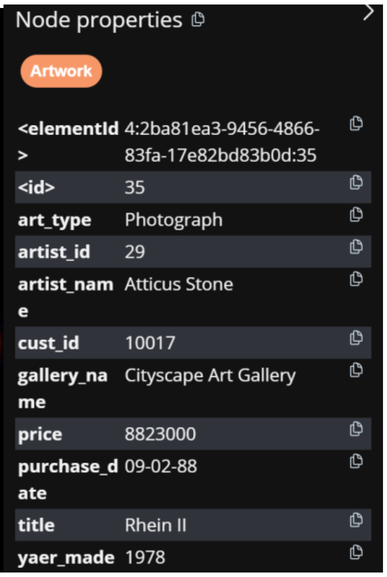
Artist

|  |  |
|---|---|
| <elementId> | 4:2ba81ea3-9456-4866-83fa-17e82bd83b0d:2 |
| <id> | 2 |
| age | 41 |
| artist_id | 3 |
| artist_name | Barbara Kruger |
| birth_date | 03-08-82 |
| birthplace | Newark |
| style | Encaustic painting |

To visualise artwork:

```
MATCH (a:Artwork)
RETURN a
LIMIT 5
```



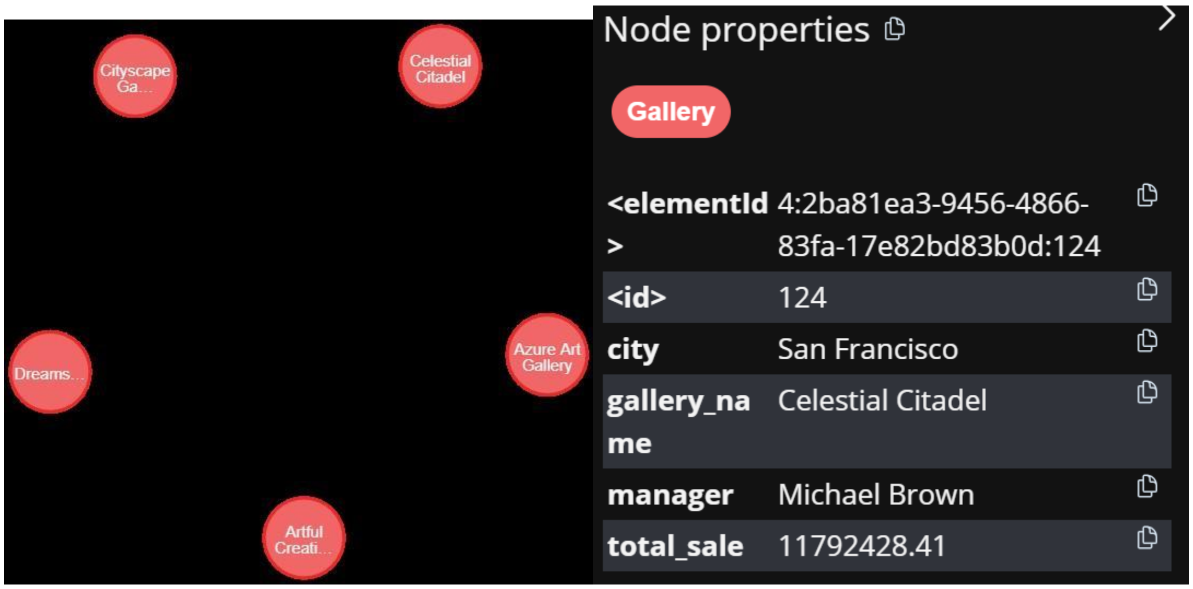

Node properties

Artwork

|  |  |
|---|---|
| <elementId> | 4:2ba81ea3-9456-4866-83fa-17e82bd83b0d:35 |
| <id> | 35 |
| art_type | Photograph |
| artist_id | 29 |
| artist_name | Atticus Stone |
| cust_id | 10017 |
| gallery_name | Cityscape Art Gallery |
| price | 8823000 |
| purchase_date | 09-02-88 |
| title | Rhein II |
| yaer_made | 1978 |

To visualise gallery:

```
MATCH (g:Gallery)
RETURN g
LIMIT 5
```



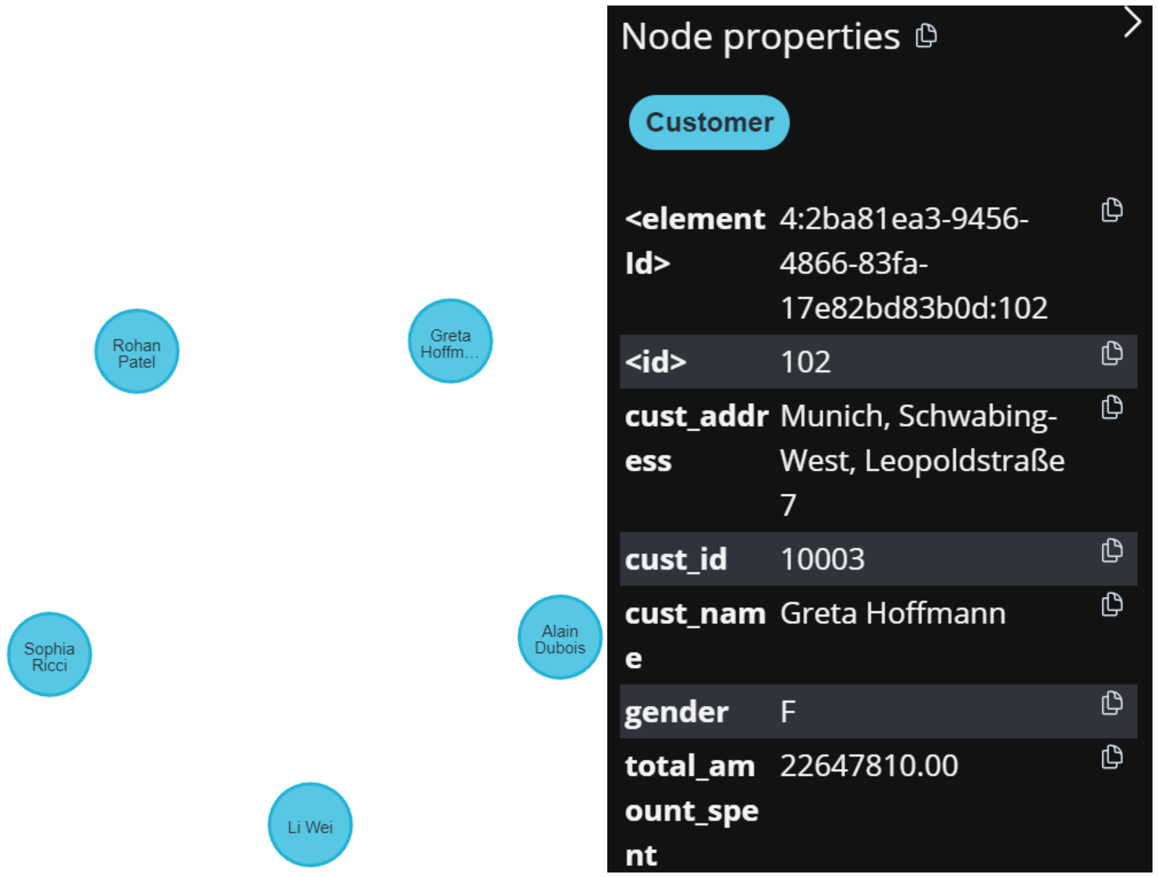
Node properties

Gallery

| | |
|---|---|
| <elementId> | 4:2ba81ea3-9456-4866-83fa-17e82bd83b0d:124 |
| <id> | 124 |
| city | San Francisco |
| gallery_name | Celestial Citadel |
| manager | Michael Brown |
| total_sale | 11792428.41 |

To visualise Customers:

```
MATCH (c:Customer)
RETURN c
LIMIT 5
```



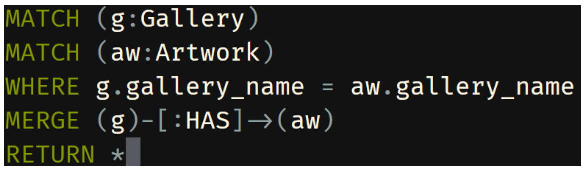
Node properties

Customer

| | |
|---|---|
| <element Id> | 4:2ba81ea3-9456-4866-83fa-17e82bd83b0d:102 |
| <id> | 102 |
| cust_address | Munich, Schwabing-West, Leopoldstraße 7 |
| cust_id | 10003 |
| cust_name | Greta Hoffmann |
| gender | F |
| total_amount_spent | 22647810.00 |

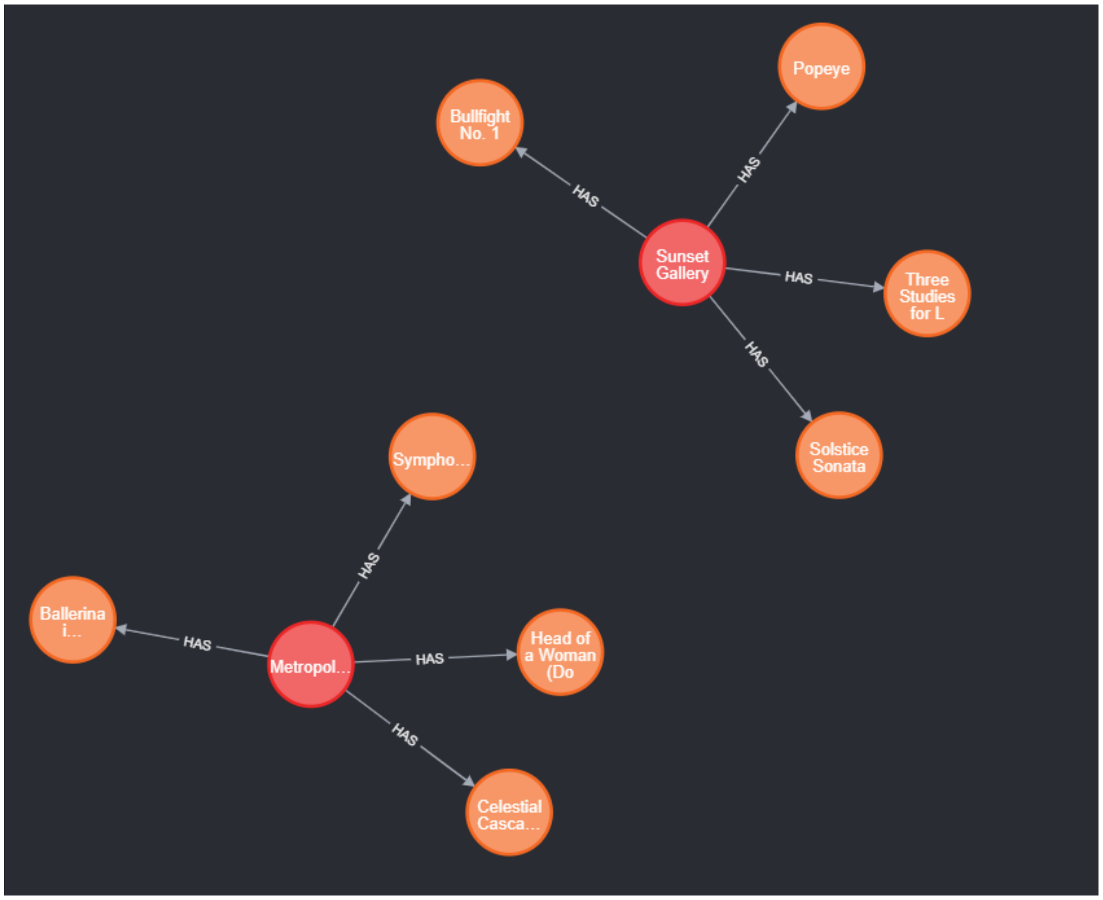
Adding the relationships:

1. **Gallery HAS artwork:**

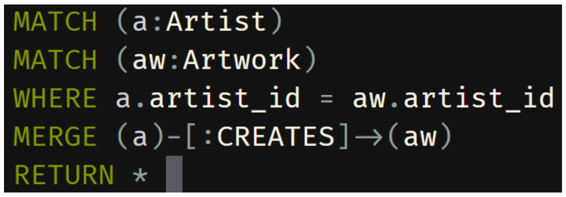


```
MATCH (g:Gallery)
MATCH (aw:Artwork)
WHERE g.gallery_name = aw.gallery_name
MERGE (g)-[:HAS]→(aw)
RETURN *
```

The first relationship was added using the above query. It is a Gallery ‘has’ artwork relationship, showing which galleries have all the different pieces of artwork. The two nodes are joined on the gallery_name column in both entities. This is the equivalent of a one-to-many relationship in traditional table databases, where one gallery ‘has’ many artworks. Below is an example of this relationship:

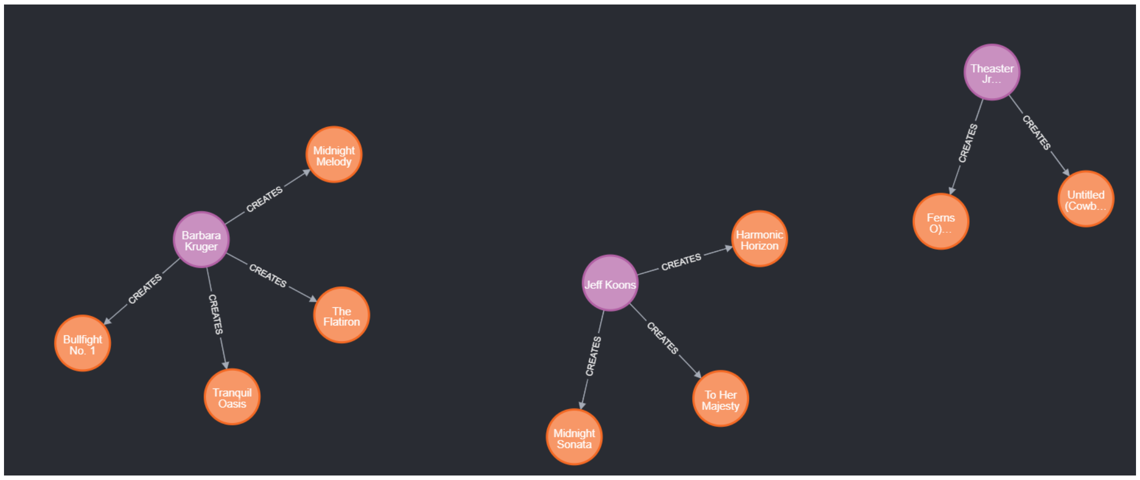


## 2. Artist CREATES Artwork

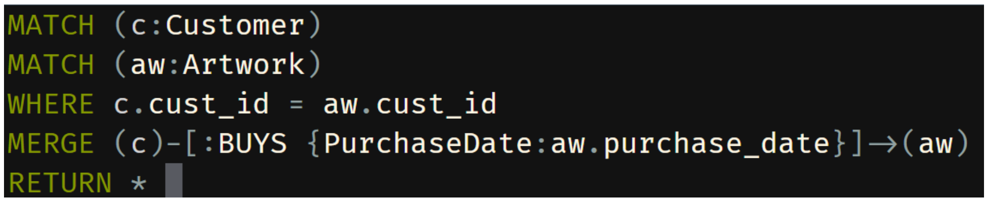


```
MATCH (a:Artist)
MATCH (aw:Artwork)
WHERE a.artist_id = aw.artist_id
MERGE (a)-[:CREATES]→(aw)
RETURN *
```

The artist creates artwok relationship is probably the most important relationship for this graph database. This will allow us to see which artists create which pieces of art. Below is an illustration of this relationship showing that each art piece is created by one artist and that each artist can create mulitple pieces of art.

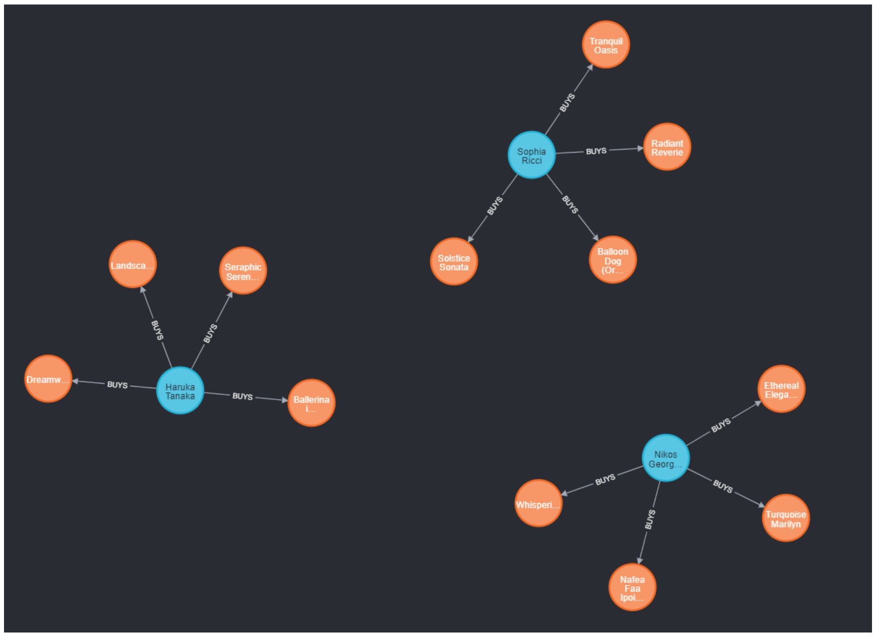


### 3.Customer BUYS Artwork



```
MATCH (c:Customer)
MATCH (aw:Artwork)
WHERE c.cust_id = aw.cust_id
MERGE (c)-[:BUYS {PurchaseDate:aw.purchase_date}]->(aw)
RETURN *
```

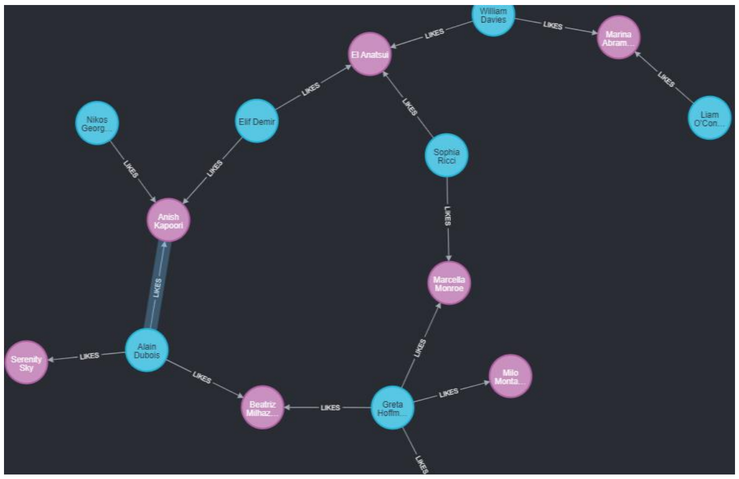
Another important relationship for any art gallery. This allows us to see which pieces of art are bought by which customers. As we will see later, the lifetime spend of customers can be deduced amongst other things. Here is an illustration of this relationship:



### 4.Customer LIKES ARTIST

```
LOAD CSV WITH HEADERS FROM 'file:///likes.csv' AS row
MATCH (c:Customer {cust_id: toInteger(row.cust_id)})
MATCH (a:Artist {artist_id:toInteger(row.artist_id)})
MERGE (c)-[:LIKES]-(a)
RETURN *;
```

This relationship will allow for customer preferences to be inferred. Knowing which artist a customer likes can be useful for targeted ad campaigns for example. This is a many-to-many relationship where each customer can like multiple artists and each artist can be liked by multiple customers. Here is a representation of this relationship:



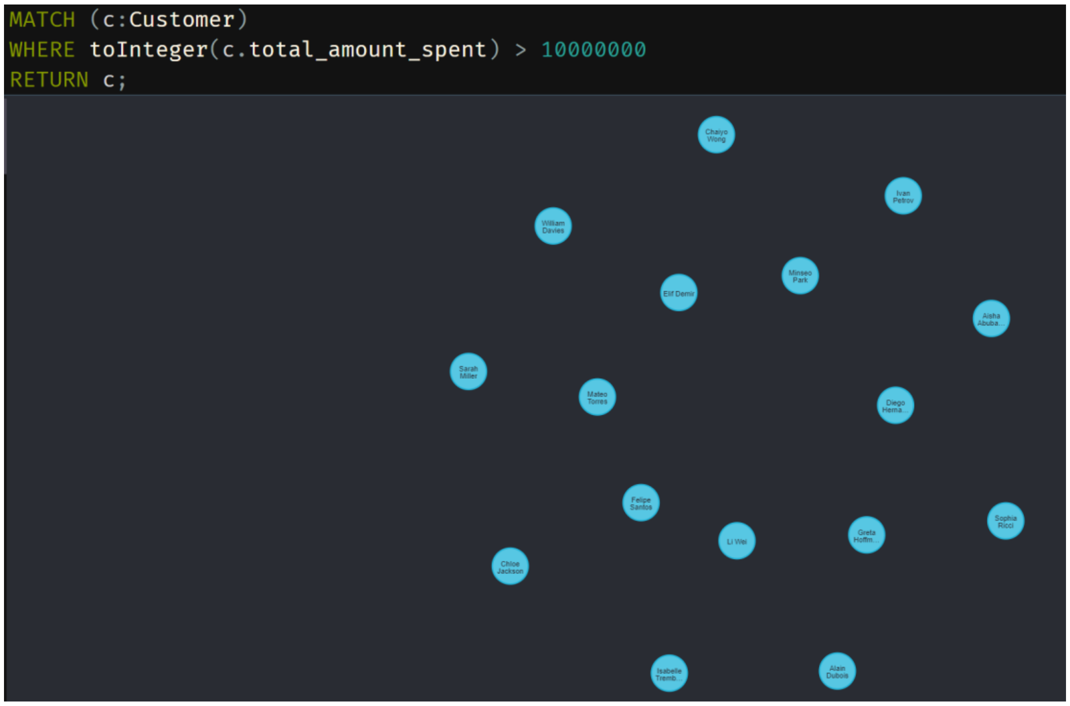
**Explaining Choice of Entities:**

As mentioned earlier, the entities have been carefully chosen to fit the Neo4j database management system while providing the best usability in a practical use case. Seeing as this project is for an art gallery, the entities chosen were the ones central to any art gallery business. Artist, Artwork and Gallery allow decision makers to get a broad understanding of the different pieces they have on offer in each gallery and which artists are providing them.

## 2.6.1. NEO4J Test Cases

### *2.6.1.1. Test Case 1. Customers who've spent over £1 million.*

**Query and Result:**



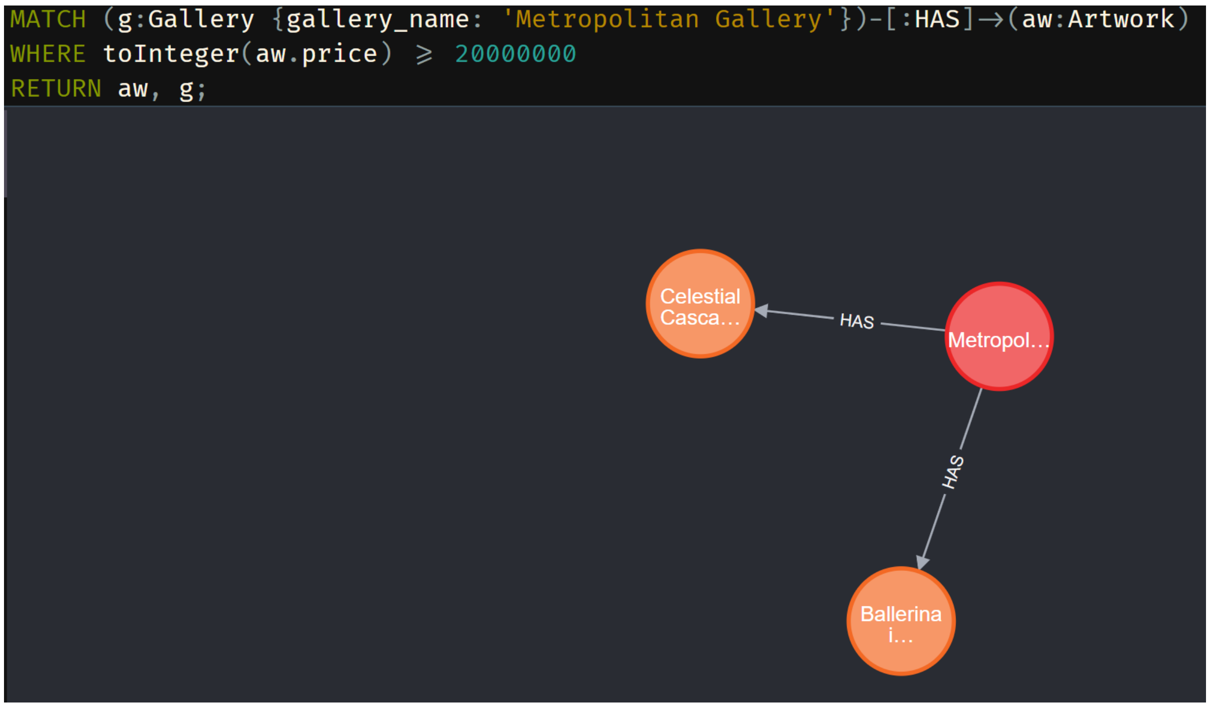
```
MATCH (c:Customer)
WHERE toInteger(c.total_amount_spent) > 10000000
RETURN c;
```

**Explanation:**

Match is used to return the node labelled Customer. The Where clause adds the condition to only return those with a total amount spent attribute over £1 million.

### *2.6.1.2. Test Case 2: Return artwork in the metropolitan gallery costing more than £2 million.*

**Query and Result:**



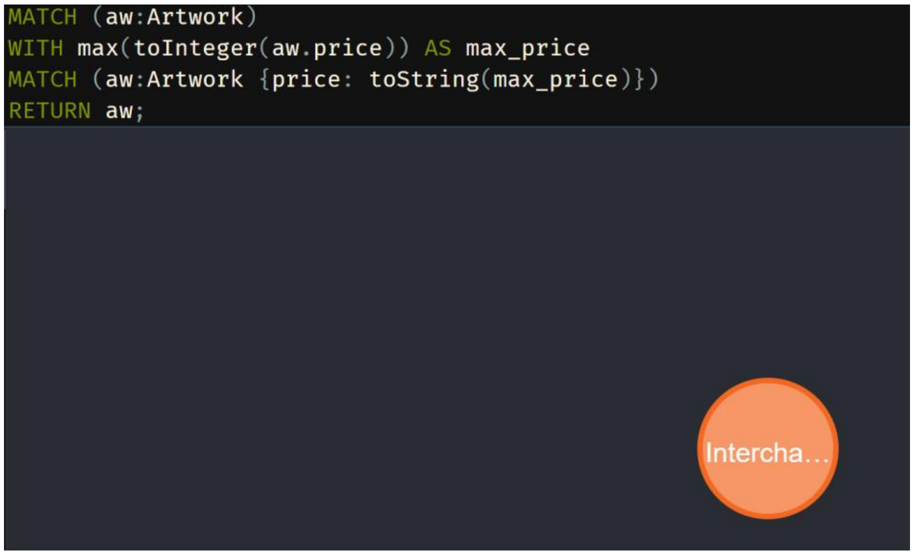
```
MATCH (g:Gallery {gallery_name: 'Metropolitan Gallery'})-[:HAS]→(aw:Artwork)
WHERE toInteger(aw.price) ≥ 20000000
RETURN aw, g;
```

**Query Explanation:**

The Match Clause Specifies the Gallery and Artwork entities based on the HAS relationship. The direction of the relationship is denoted by the arrow. The WHERE clause filters out relationships to only those with a price over £2 million.

### *2.6.1.3. Test Case 3: Return the most expensive art piece in the database.*

**Query and Result**



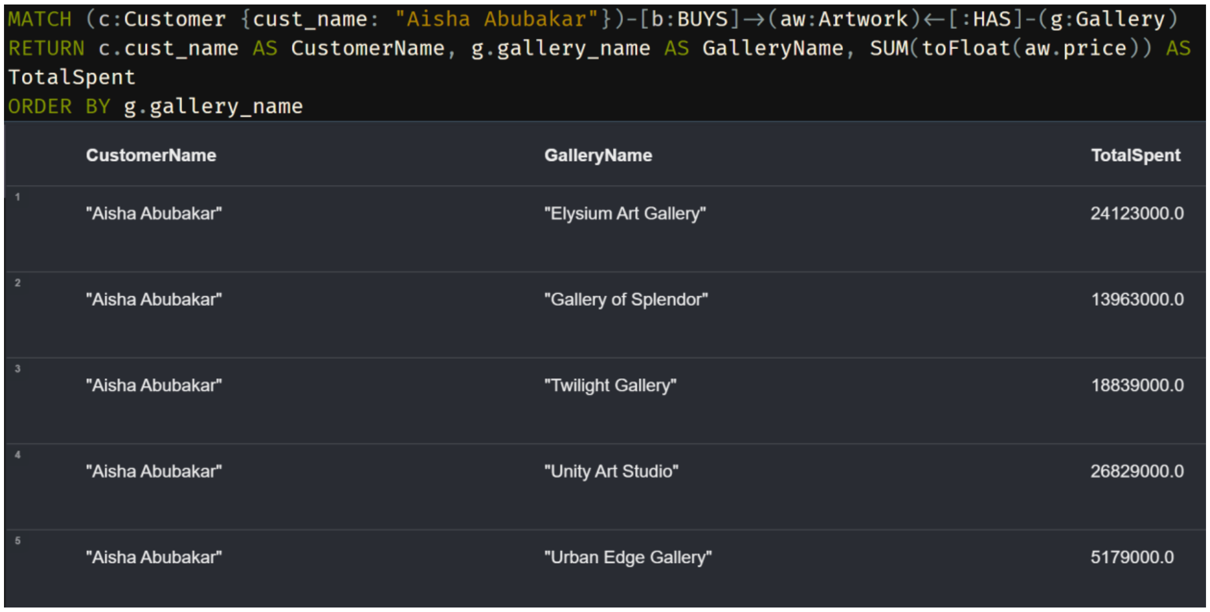
```
MATCH (aw:Artwork)
WITH max(toInteger(aw.price)) AS max_price
MATCH (aw:Artwork {price: toString(max_price)})
RETURN aw;
```

**Query Explanation:**

The Match clause specifies the artwork entity and the WITH clause specifies a maximum price variable. The maximum price is obtained using the in built max function on the price column in the artwork entity.

### *2.6.1.4. Test Case 4: Get total amount spent by Aisha Abubakr at each gallery.*

**Query and Result:**



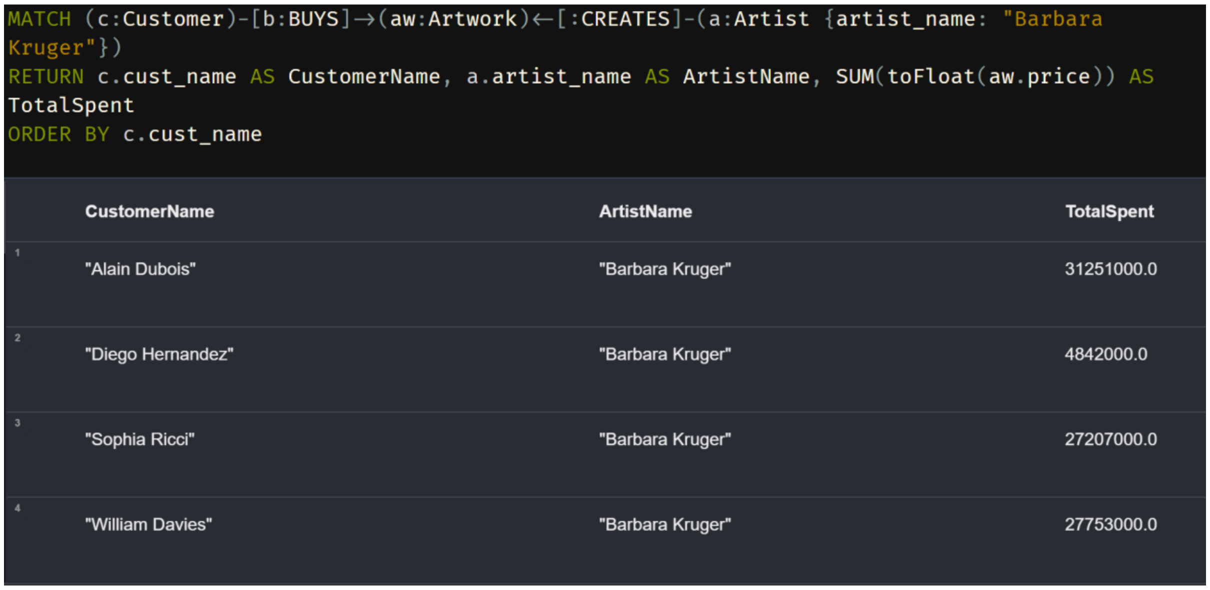
```
MATCH (c:Customer {cust_name: "Aisha Abubakar"})-[b:BUYS]→(aw:Artwork)←[:HAS]-(g:Gallery)
RETURN c.cust_name AS CustomerName, g.gallery_name AS GalleryName, SUM(toFloat(aw.price)) AS
TotalSpent
ORDER BY g.gallery_name
```

| | CustomerName | GalleryName | TotalSpent |
|---|---|---|---|
| 1 | "Aisha Abubakar" | "Elysium Art Gallery" | 24123000.0 |
| 2 | "Aisha Abubakar" | "Gallery of Splendor" | 13963000.0 |
| 3 | "Aisha Abubakar" | "Twilight Gallery" | 18839000.0 |
| 4 | "Aisha Abubakar" | "Unity Art Studio" | 26829000.0 |
| 5 | "Aisha Abubakar" | "Urban Edge Gallery" | 5179000.0 |

**Query Explanation:**

The Match clause specifies 3 entities and 2 relationships to achieve this result. The cust_name attribute is used to filter the results by the name of the customer. The return clause specifies the column headers as aliases and the result is ordered by the descending order of the amount spent. This gives us a good understanding of which galleries this customer likes to shop at.

#### *2.6.1.5. Test Case 5: Get total amount spent by each customer or the artist Barbra Kruger*



```
MATCH (c:Customer)-[b:BUYS]→(aw:Artwork)←[:CREATES]-(a:Artist {artist_name: "Barbara
Kruger"})
RETURN c.cust_name AS CustomerName, a.artist_name AS ArtistName, SUM(toFloat(aw.price)) AS
TotalSpent
ORDER BY c.cust_name
```

| | CustomerName | ArtistName | TotalSpent |
|---|---|---|---|
| 1 | "Alain Dubois" | "Barbara Kruger" | 31251000.0 |
| 2 | "Diego Hernandez" | "Barbara Kruger" | 4842000.0 |
| 3 | "Sophia Ricci" | "Barbara Kruger" | 27207000.0 |
| 4 | "William Davies" | "Barbara Kruger" | 27753000.0 |

**Query Explanation:**

Again, the MATCH clause specifies three entities and 2 relationships, with the artist's name attribute limited to Barbra Kruger. This query gives an understanding of which customers like Barbra Kruger's work the most depending on how much they've spent.

# 3. Part B

## 3.1. Background

This part focuses on the application of machine-learning techniques to the Room Occupancy Estimation dataset from the UCI Machine Learning Repository. We will begin by defining the training and testing sets for our dataset, followed by the implementation of a neural network and SVM algorithm of our choice. The performance of these algorithms will be compared to provide insights into their effectiveness.

Next, we will apply Principal Component Analysis (PCA) to the dataset and discuss its outcome, particularly how the number of principal components affects the percentage of variance covered. We will also apply a feature selection method of our choice and compare the performance of one of the previously used algorithms before and after this application.

In the spirit of Explainable AI, we will delve into the understanding of our machine learning model decisions. We will investigate the explainability of our models by answering why the machine learning model made specific decisions on three samples.

The tests will be performed on the test data set to evaluate the results, using the AUC (area under curve) and accuracy as metrics for comparing the performance. This report will provide comprehensive answers to each of the five points above, offering a detailed exploration of machine learning techniques applied to the Room Occupancy Estimation dataset.

## 3.2. Choice of dataset

We chose Room Occupancy Estimation dataset as it provides a rich ground for applying various machine learning techniques to predict future trends, recognize patterns, and even automate systems based on the predictive outputs. This includes not just occupancy but also the prediction of environmental conditions based on historical data.

## 3.3. Dataset Exploration and Description

The dataset contains 10,129 entries and 19 columns related to temperature, light, sound, CO2 levels, and PIR (passive infrared) sensors across various sensors, alongside date and time entries. The data was collected over 4 days in a 6m x 4.6m room with people ranging from 0 to 3 people. Readings from 7 sensor nodes arranged in a star configuration are transmitted to edge nodes every 30 seconds via wireless transceivers. Manual calibration of CO2, sound and PIR sensors was performed. The sound sensor gain was set to maximum when calibrating the CO2 sensor zero point. PIR sensor sensitivity and motion detection time are set to the highest values. The sensor deployment includes temperature, light and sound sensors at nodes S1-S4 for motion detection, CO2 sensor at S5, and PIR sensors at S6 and S7 on the roof overhangs for motion detection to locate the assist charge.

## 3.4. Data Training and Data Testing

We'll split the dataset into training and testing sets. A common split ratio is 80% for training and 20% for testing. Before splitting, we'll remove the 'Date' and 'Time' columns as they require preprocessing to be used effectively for model training.

Code

```
data = pd.read_csv('/content/Occupancy_Estimation.csv')

data.drop(['Date', 'Time'], axis=1, inplace=True)

X = data.drop('Room_Occupancy_Count', axis=1)

y = data['Room_Occupancy_Count']

# Encoding target variable

encoder = LabelEncoder()

y_encoded = encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2,
random_state=42)

# Scaling features

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

## 3.5. Implementing Models

We will implement two models:

1) **A simple Neural Network**

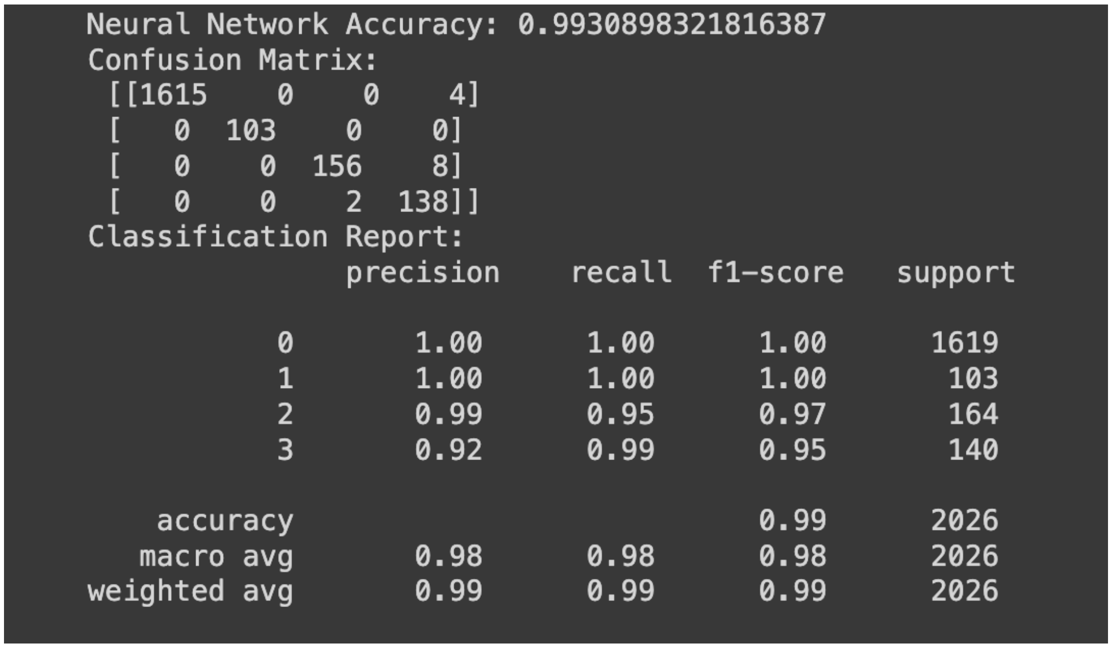
   Code

   ```
   import numpy as np
   from tensorflow.keras.models import Sequential
   from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
   from sklearn.preprocessing import StandardScaler
   from sklearn.model_selection import train_test_split
   ```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import pandas as pd
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.utils.class_weight import compute_class_weight

data = pd.read_csv('Occupancy_Estimation.csv')

# Dropping the 'Date' and 'Time' columns for model training
feature_columns = data.columns.drop(['Date', 'Time', 'Room_Occupancy_Count'])
X = data[feature_columns]
y = data['Room_Occupancy_Count']

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and test sets
X_train_scaled, X_test_scaled, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Calculate class weights
classes = np.unique(y_train)
class_weights = compute_class_weight('balanced', classes=classes, y=y_train)
class_weights_dict = {c: w for c, w in zip(classes, class_weights)}

# Define the neural network model
model = Sequential([
   Dense(256, activation='relu', input_shape=(X_train_scaled.shape[1],)),
   BatchNormalization(),
   Dropout(0.3),
   Dense(128, activation='relu'),
   BatchNormalization(),
   Dropout(0.3),
   Dense(64, activation='relu'),
   Dropout(0.2),
   Dense(32, activation='relu'),
   Dropout(0.1),
   Dense(len(np.unique(y)), activation='softmax')  # Assuming a multi-class
classification
])

# Compile the model with a potentially different learning rate
from tensorflow.keras.optimizers import Adam
optimizer = Adam(learning_rate=0.0001)
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Setup early stopping and model checkpoint
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss',
save_best_only=True, save_weights_only=True)

# Train the model with class weights
history = model.fit(X_train_scaled, y_train, epochs=100, batch_size=32,
validation_split=0.2, verbose=1,
                class_weight=class_weights_dict, callbacks=[early_stopping,
model_checkpoint])

# Load the best weights
model.load_weights('best_model.h5')


# Evaluate the model
predictions = model.predict(X_test_scaled)
predicted_classes = np.argmax(predictions, axis=1)
accuracy = accuracy_score(y_test, predicted_classes)
confusion = confusion_matrix(y_test, predicted_classes)
report = classification_report(y_test, predicted_classes)

print("Neural Network Accuracy:", accuracy)
```

print("Confusion Matrix:\n", confusion)
print("Classification Report:\n", report)

Output



```
Neural Network Accuracy: 0.9930898321816387
Confusion Matrix:
 [[1615    0    0    4]
 [   0  103    0    0]
 [   0    0  156    8]
 [   0    0    2  138]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00      1619
           1       1.00      1.00      1.00       103
           2       0.99      0.95      0.97       164
           3       0.92      0.99      0.95       140

    accuracy                           0.99      2026
   macro avg       0.98      0.98      0.98      2026
weighted avg       0.99      0.99      0.99      2026
```

A classification report and a confusion matrix for a four-class prediction model are included in the results. Analysing these results will enable us to better understand the models' functionality.

### Confusion Matrix Analysis

The confusion matrix indicates the model's accuracy in predicting outcomes compared to the actual results.

**Class 0:** The model predicted 1615 true positives, indicating that this class was properly identified. There is one false positive for class 1, none for class 2 and 4 for class 3. This class is classified nearly precisely.

**Class 1:** Because there were no false positives and all 103 samples were correctly identified as class 1 (true positives), this class was excellent.

**Class 2:** 156 out of the 164 examples had an accurate classification, while 8 cases were incorrectly classified to class 3. This suggests an amount of confusion between classes 2 and 3.

**Class 3:** Just 2 examples in this class was inaccurately classified as class 2, out of 140 examples that were correctly identified. This indicates a very high accuracy for class 3, despite having a little chance of being mistaken for class 2.

### Classification Report Analysis

Key metrics such as precision, recall and F-1 score were provided for each class, along with overall accuracy.

**Precision (Positive Predictive Value):** Shows how accurate the positive predictions are.

**Class 0:** 100% - Every item with a class 0 label was accurate.

**Class 1:** 100% - All predictions of class 1 was correct.

**Class 2:** 99% - Even though some class 2 items were misclassified as class 3, nearly all predictions of class 2 were correct.

**Class 3:** 92% - Compared to other classes, this class is lower because 12 class 2 items were misclassified as class 3.

**Recall (Sensitivity or True Positive Rate):** This shows the accuracy with which the model can identify every positive sample.

**Class 0:** 100% - Every actual class 0 items were identified.

**Class 1:** 100% - Every actual class 1 items were identified.

**Class 2:** 95% - Some class 8 items were missed and misclassified as class 3.

**Class 3:** 99% - Almost all class 3 items were correctly identified, except 2.

**F1-Score:** The average of precision and recall.

**Class 0, 1:** An outstanding balance between recall and precision can be seen in F1 scores that are perfect or almost perfect.

**Class 2, 3:** Compared to classes 0 and 1, it has lower F1 scores, indicating the problems in misclassification between these two classes.

**Overall Accuracy:** 99%, indicating the model's outstanding overall performance across all classes.

### Insights and Recommendations

**Overall Performance:** With strong metrics in every class and an overall accuracy of 99%, the model works excellently. Anyhow, there is a clear area of confusion between classes 2 and 3.
**Focus on Class 2 and 3:** Even though the accuracy is high, the confusion between classes 2 and 3 needs to be addressed. Investigating the features that are causing misclassifications between these classes could be beneficial. This could be addressed by additional feature engineering, gathering more data, or even adjusting class weights in the model training process.
**Model Refinement:** Modifying the model architecture or hyperparameters can help in increasing class 2 recall and class 3 precision.

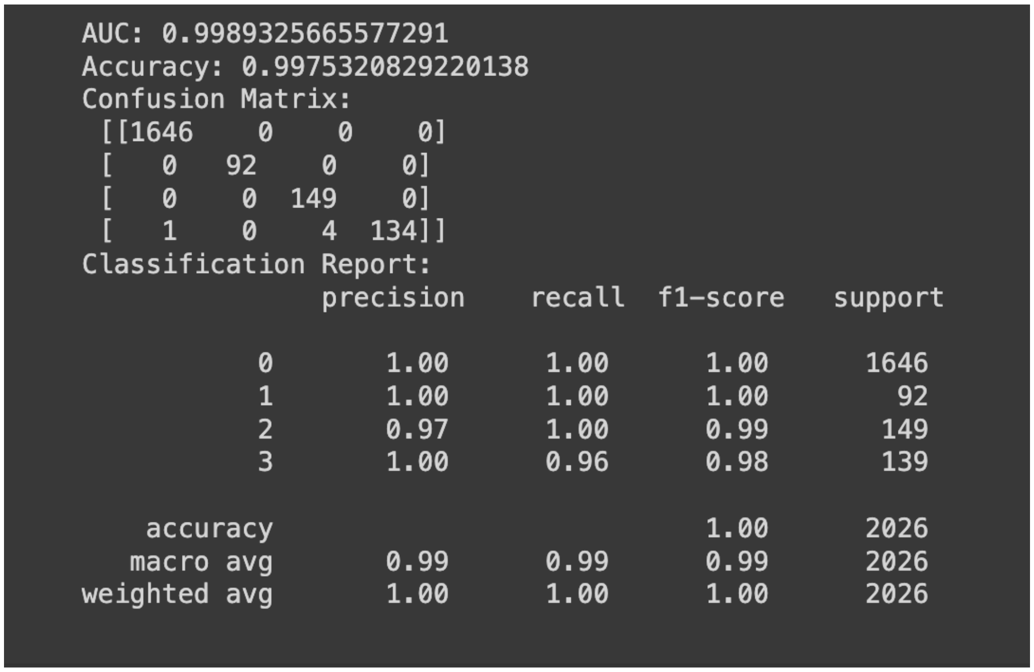
Although the model is robust, this analysis indicates that there is room for improvement in the ability to distinguish between classes 2 and 3, which could result in even higher overall performance.

## 2) Random Forest Model

### Code

```
X = data[feature_columns]
y = data['Room_Occupancy_Count']

# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Splitting the data into training and test sets based on a sequential index
split_index = int(len(data) * 0.8)
X_train_scaled = X_scaled[:split_index]
X_test_scaled = X_scaled[split_index:]
y_train = y[:split_index]
y_test = y[split_index:]

# Define the Random Forest model
rf_model = RandomForestClassifier()
```

```
# Train the Random Forest model
rf_model.fit(X_train_scaled, y_train)

# Make predictions with the Random Forest model
rf_predictions = rf_model.predict(X_test_scaled)

# Evaluate the Random Forest model
accuracy = accuracy_score(y_test, rf_predictions)
confusion = confusion_matrix(y_test, rf_predictions)
classification_rep = classification_report(y_test, rf_predictions)

# Print results
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", confusion)
print("Classification Report:\n", classification_rep)
```



```
AUC: 0.9989325665577291
Accuracy: 0.9975320829220138
Confusion Matrix:
 [[1646    0    0    0]
 [   0   92    0    0]
 [   0    0  149    0]
 [   1    0    4  134]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00      1646
           1       1.00      1.00      1.00        92
           2       0.97      1.00      0.99       149
           3       1.00      0.96      0.98       139

    accuracy                           1.00      2026
   macro avg       0.99      0.99      0.99      2026
weighted avg       1.00      1.00      1.00      2026
```

Once again, a confusion matrix was used to analyse the performance of the neural network. When it comes to class 0, the model performed exceptionally well with 1646 correct classifications and 0 instances being misclassified. This shows that the class of 0 people in the room was one that the model captured very well. The same can largely be said for all the other classes, however, classes 2 and 3 had slightly worse performance in terms of classification. Class 2 has a precision of 0.97 where most of the errors were misclassification of class 2 as class 3. This can tell us that the model is finding it slightly harder to distinguish between sensor data with 2 people in the room and sensor data for 3 people in the room. Another possible reason may be the imbalance in classes, where the original dataset contained a lot more instances with 0 people in the room compared to 1,2 and 3 people in the room. Overall, the accuracy of the model is excellent at 0.99 percent, and it can be deduced that the model has a lot of predictive power when it comes to the dataset.

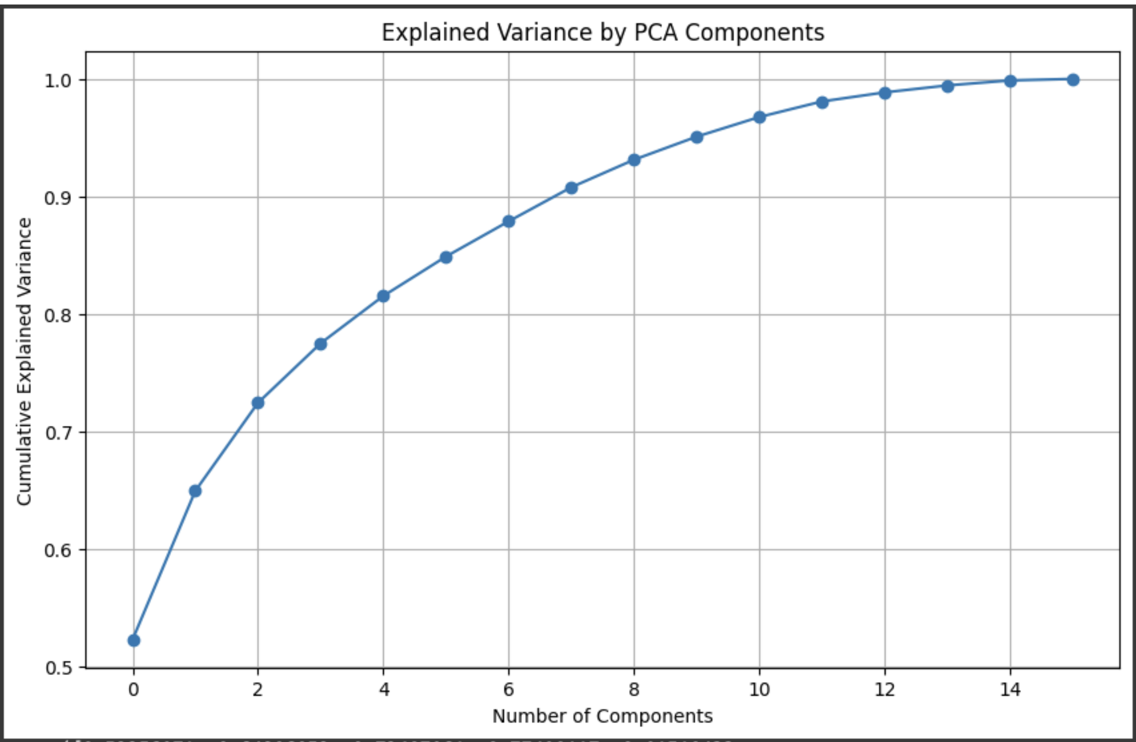
## 3.6. Applying Principal Component Analysis (PCA)

We will apply PCA to the dataset to reduce dimensionality and analyse the variance explained by the principal components.

### Code

```
# Standardise the features

pca = PCA().fit(X_scaled)

# Calculate cumulative explained variance

cumulative_variance = np.cumsum(pca.explained_variance_ratio_)

# Plotting the cumulative explained variance

plt.figure(figsize=(10, 6))

plt.plot(cumulative_variance, marker='o')

plt.xlabel('Number of Components')

plt.ylabel('Cumulative Explained Variance')

plt.title('Explained Variance by PCA Components')

plt.grid(True)

plt.show()

cumulative_variance
```

## Output



```
array([0.52256871, 0.64906052, 0.72407861, 0.77480117, 0.81510433,
       0.84867922, 0.87868205, 0.90769944, 0.93112683, 0.95079098,
       0.96751547, 0.98079039, 0.98850436, 0.99442713, 0.99867489,
       1.        ])
```

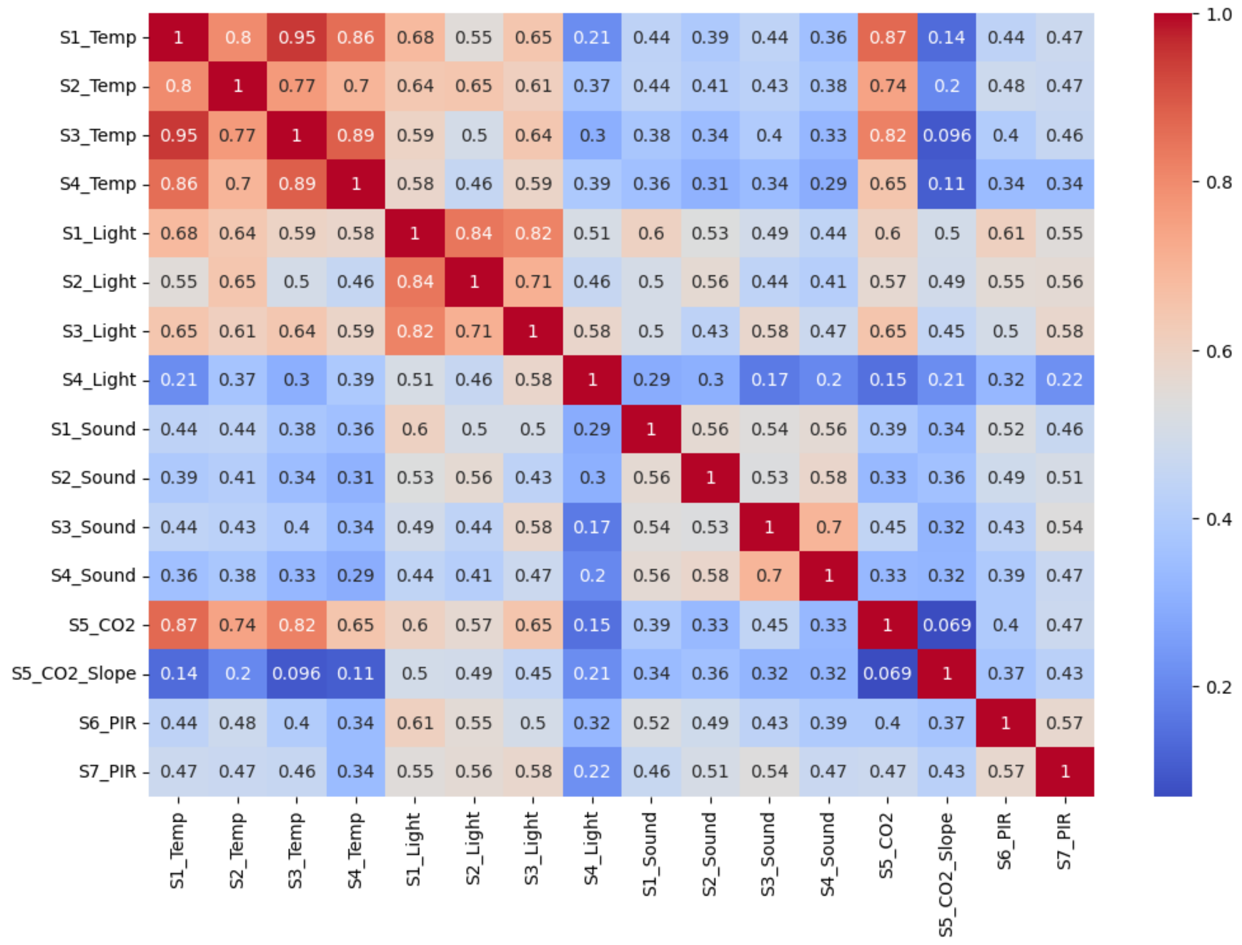
The graph above shows a principal component analysis (PCA) for our sensor dataset. A few important conclusions can be drawn regarding the effect of dimensionality reduction on the dataset. PCA allows attempts to represent the variation in the data using fewer features than the original dataset. From the plot we can make the important observation that the first component alone represents 52.39% of the variation in the data. While including the first two components captures 65.10% of the variation in the data. These results show us that the first few components alone capture a lot of the variation in the dataset.

After the first few components, we can see that there are diminishing returns regarding the variation in the data. This can most clearly be seen by the small increase in variation captured between the 9th and the 10th component. Adding the 10th component results in an increase of just 2%, showing that these components are not as important as the first few components.

From the plot, we can see that the PCA plot plateaus at around the 10th component. This simply means that there is no extra information gained beyond the 10th component. This information can help us capture as much variation as possible with the fewest number of components. Doing so can help simplify the model and avoid overfitting by including too many components. From the plot, we see that 93.8% of the variation in the data is captured by the first 8 components which may be enough to build a machine learning model. Knowing this we can deploy a machine learning model using the first 8 components, this gives a happy medium between capturing variation in the data and not overfitting.

## 3.7. Feature Selection

A correlation matrix explains how much the variation in one variable is explained by the variation in another. A value closer to one indicates a higher correlation.



| | S1_Temp | S2_Temp | S3_Temp | S4_Temp | S1_Light | S2_Light | S3_Light | S4_Light | S1_Sound | S2_Sound | S3_Sound | S4_Sound | S5_CO2 | S5_CO2_Slope | S6_PIR | S7_PIR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S1_Temp | 1 | 0.8 | 0.95 | 0.86 | 0.68 | 0.55 | 0.65 | 0.21 | 0.44 | 0.39 | 0.44 | 0.36 | 0.87 | 0.14 | 0.44 | 0.47 |
| S2_Temp | 0.8 | 1 | 0.77 | 0.7 | 0.64 | 0.65 | 0.61 | 0.37 | 0.44 | 0.41 | 0.43 | 0.38 | 0.74 | 0.2 | 0.48 | 0.47 |
| S3_Temp | 0.95 | 0.77 | 1 | 0.89 | 0.59 | 0.5 | 0.64 | 0.3 | 0.38 | 0.34 | 0.4 | 0.33 | 0.82 | 0.096 | 0.4 | 0.46 |
| S4_Temp | 0.86 | 0.7 | 0.89 | 1 | 0.58 | 0.46 | 0.59 | 0.39 | 0.36 | 0.31 | 0.34 | 0.29 | 0.65 | 0.11 | 0.34 | 0.34 |
| S1_Light | 0.68 | 0.64 | 0.59 | 0.58 | 1 | 0.84 | 0.82 | 0.51 | 0.6 | 0.53 | 0.49 | 0.44 | 0.6 | 0.5 | 0.61 | 0.55 |
| S2_Light | 0.55 | 0.65 | 0.5 | 0.46 | 0.84 | 1 | 0.71 | 0.46 | 0.5 | 0.56 | 0.44 | 0.41 | 0.57 | 0.49 | 0.55 | 0.56 |
| S3_Light | 0.65 | 0.61 | 0.64 | 0.59 | 0.82 | 0.71 | 1 | 0.58 | 0.5 | 0.43 | 0.58 | 0.47 | 0.65 | 0.45 | 0.5 | 0.58 |
| S4_Light | 0.21 | 0.37 | 0.3 | 0.39 | 0.51 | 0.46 | 0.58 | 1 | 0.29 | 0.3 | 0.17 | 0.2 | 0.15 | 0.21 | 0.32 | 0.22 |
| S1_Sound | 0.44 | 0.44 | 0.38 | 0.36 | 0.6 | 0.5 | 0.5 | 0.29 | 1 | 0.56 | 0.54 | 0.56 | 0.39 | 0.34 | 0.52 | 0.46 |
| S2_Sound | 0.39 | 0.41 | 0.34 | 0.31 | 0.53 | 0.56 | 0.43 | 0.3 | 0.56 | 1 | 0.53 | 0.58 | 0.33 | 0.36 | 0.49 | 0.51 |
| S3_Sound | 0.44 | 0.43 | 0.4 | 0.34 | 0.49 | 0.44 | 0.58 | 0.17 | 0.54 | 0.53 | 1 | 0.7 | 0.45 | 0.32 | 0.43 | 0.54 |
| S4_Sound | 0.36 | 0.38 | 0.33 | 0.29 | 0.44 | 0.41 | 0.47 | 0.2 | 0.56 | 0.58 | 0.7 | 1 | 0.33 | 0.32 | 0.39 | 0.47 |
| S5_CO2 | 0.87 | 0.74 | 0.82 | 0.65 | 0.6 | 0.57 | 0.65 | 0.15 | 0.39 | 0.33 | 0.45 | 0.33 | 1 | 0.069 | 0.4 | 0.47 |
| S5_CO2_Slope | 0.14 | 0.2 | 0.096 | 0.11 | 0.5 | 0.49 | 0.45 | 0.21 | 0.34 | 0.36 | 0.32 | 0.32 | 0.069 | 1 | 0.37 | 0.43 |
| S6_PIR | 0.44 | 0.48 | 0.4 | 0.34 | 0.61 | 0.55 | 0.5 | 0.32 | 0.52 | 0.49 | 0.43 | 0.39 | 0.4 | 0.37 | 1 | 0.57 |
| S7_PIR | 0.47 | 0.47 | 0.46 | 0.34 | 0.55 | 0.56 | 0.58 | 0.22 | 0.46 | 0.51 | 0.54 | 0.47 | 0.47 | 0.43 | 0.57 | 1 |

We have chosen to implement a correlation matrix as our feature selection method as we have found a high correlation between numerical features and target variables as illustrated in the figure above. Using the correlation feature selection method allows us to pick out the most impacted features and use them in our AI model.

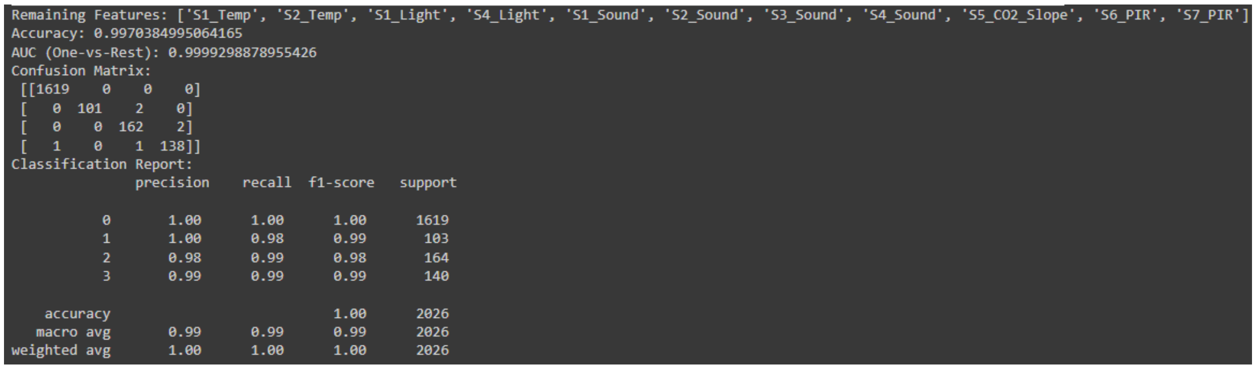
When implemented in Python, using the corr() function, it calculates a correlation matrix for the features and uses a heatmap to display it. This stage helps in the identification of strongly correlated features that may lead to issues with multicollinearity in machine learning models. Afterwards, in order to reduce redundancy and possible overfitting, the script selects features by identifying those whose correlations are higher than a predetermined threshold (0.8) and eliminating one from each highly correlated pair. The dataset is then split into training and testing sets for performance evaluation, and the remaining features are used to train a Random Forest Classifier. Lastly, accuracy and AUC measures are used to assess the model's efficacy, with the One-vs-Rest strategy—which works well in multi-class classification scenarios—being employed. A confusion matrix and a classification report also offer insights into the model's performance in several classes, providing a thorough assessment of its overall effectiveness.

## Code

```
# Calculate the correlation matrix

corr_matrix = X.corr()

# Plot the correlation matrix for visualization (optional)

plt.figure(figsize=(12, 8))

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')

plt.show()


# Set the threshold

threshold = 0.8

# Identify pairs of highly correlated features

to_drop = set()

for i in range(len(corr_matrix.columns)):

  for j in range(i):
        if abs(corr_matrix.iloc[i, j]) > threshold:
            to_drop.add(corr_matrix.columns[i])

# Drop the highly correlated features

X_filtered = X.drop(columns=to_drop)

# Print the features that are retained after removing the highly correlated ones

print("Remaining Features:", X_filtered.columns.tolist())

# Split the data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X_filtered, y, test_size=0.2,
random_state=42)

# Train a Random Forest Classifier

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)

# Evaluate the model

# Predict probabilities for AUC

probabilities = model.predict_proba(X_test)  # Get the probabilities for all classes

predicted_classes = model.predict(X_test)

accuracy = accuracy_score(y_test, predicted_classes)

# Compute AUC using One-vs-Rest strategy

auc_ovr = roc_auc_score(y_test, probabilities, multi_class="ovr", average='macro')  #
Multi-class scenario

confusion = confusion_matrix(y_test, predicted_classes)

report = classification_report(y_test, predicted_classes)

print("Accuracy:", accuracy)

print("AUC (One-vs-Rest):", auc_ovr)

print("Confusion Matrix:\n", confusion)
```

print("Classification Report:\n", report)

### Output



```
Remaining Features: ['S1_Temp', 'S2_Temp', 'S1_Light', 'S4_Light', 'S1_Sound', 'S2_Sound', 'S3_Sound', 'S4_Sound', 'S5_CO2_Slope', 'S6_PIR', 'S7_PIR']
Accuracy: 0.9970384995064165
AUC (One-vs-Rest): 0.9999298878955426
Confusion Matrix:
 [[1619    0    0    0]
 [   0  101    2    0]
 [   0    0  162    2]
 [   1    0    1  138]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00      1619
           1       1.00      0.98      0.99       103
           2       0.98      0.99      0.98       164
           3       0.99      0.99      0.99       140

    accuracy                           1.00      2026
   macro avg       0.99      0.99      0.99      2026
weighted avg       1.00      1.00      1.00      2026
```

### 3.7.1 Comparison Between Before and After Future Selection

The Random Forest model shows outstanding performance on this dataset, evidenced by an AUC score of 0.999 and an accuracy rate of 0.997 after strategic feature selection and comprehensive training. Initially, the model's accuracy stood at 0.925 and AUC at 0.988. These metrics saw significant improvements following feature selection, which minimised noise and reduced the risk of overfitting. The confusion matrix reveals that the model excels across all classes with minimal misclassifications, and the alignment of predictions with actual labels results in a high count of true positives and few false positives. Additionally, the classification report shows high recall, precision, and F1 scores for each category, underscoring the model's adeptness at maintaining a strong balance between precision and recall. This enhancement in both accuracy and AUC underscores the successful optimisation of the Random Forest model for this dataset, improving its efficiency and solidifying its classification strength.

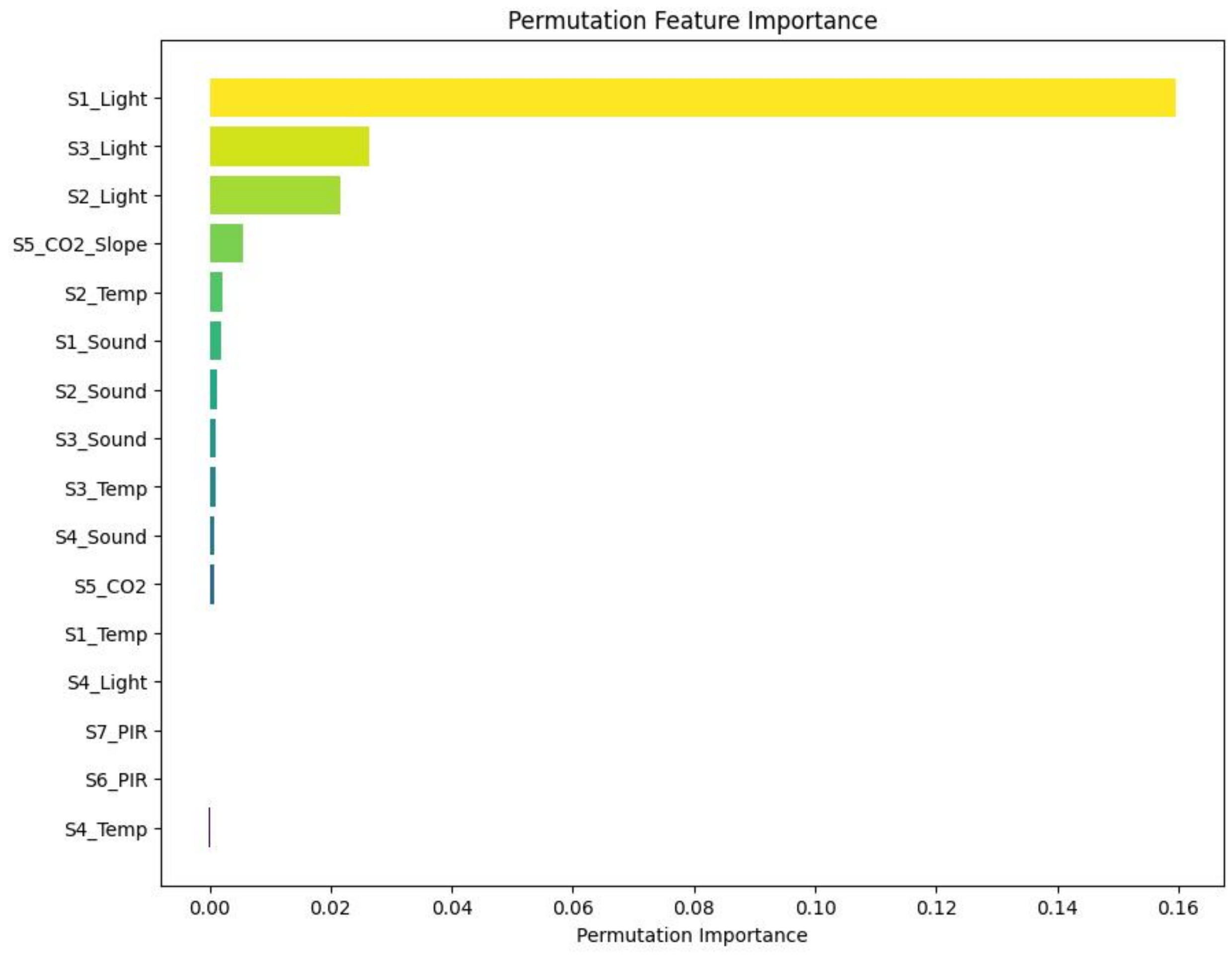
## 3.8. Explainable AI

We are going to apply the following two Explainable AI methods to our Random Forest model, Permutation Feature Importance and SHAP (SHapley Additive exPlanations).

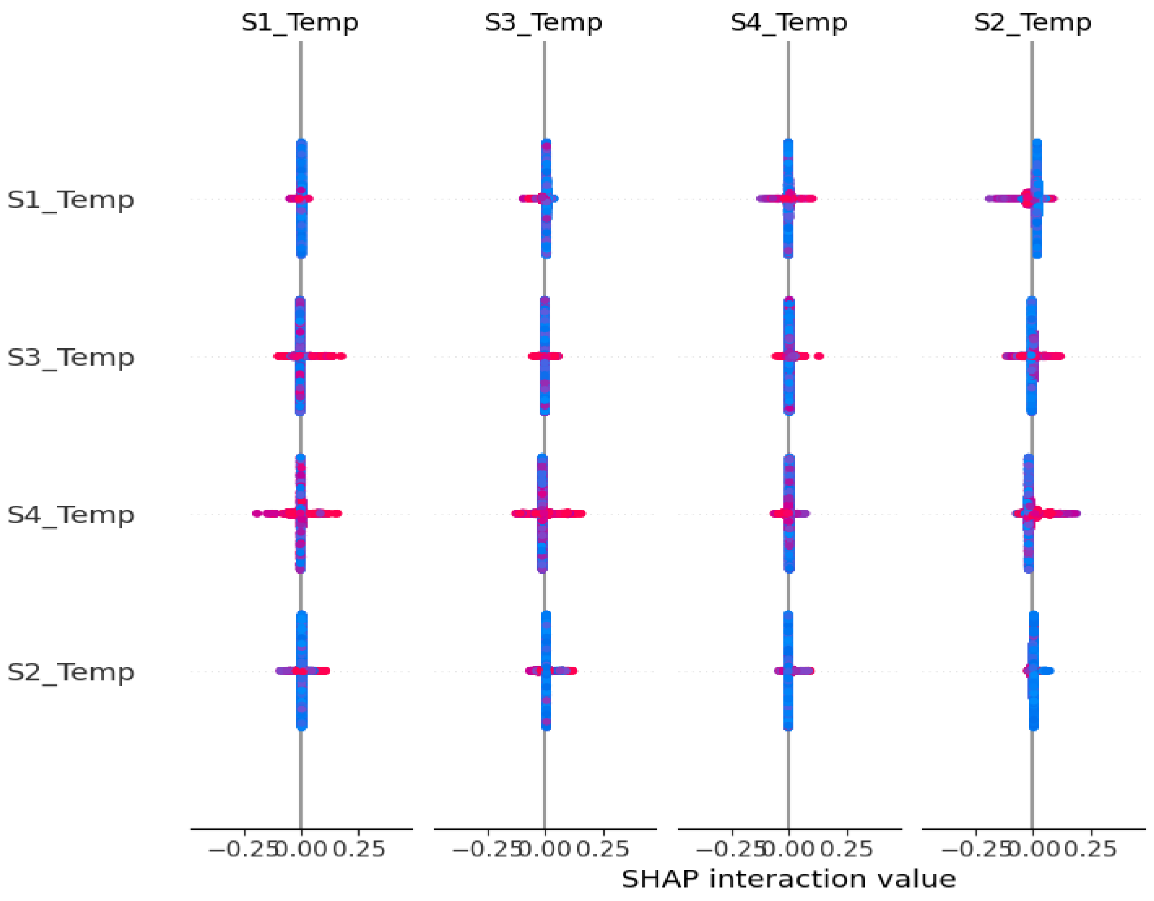
Permutation feature importance is a feature importance method that assesses the impact of each feature on the model's accuracy. This is done by sequentially removing features from the model and assessing the impact that the removed feature has on the accuracy of the model. Permutation is a straightforward way of assessing the impact of a feature on the accuracy of a model.

SHAP Is another explainable AI method that shows us the impact that each feature has on the prediction being made. The SHAP is again a straightforward way to assess how much each feature is impacting the prediction made by the model. A strong point of the SHAP method the fact that it can be applied to any type of model and that allows us to compare the same parameters across different models.

## Output



By permuting the values of each feature, the association between the feature and the target is broken, and the increase in the prediction error of the model is measured. This provides a more accurate representation of the feature's predictive ability within the model. Based on their higher importance values, features such as S1_Light and S3_Light seem to have the biggest effects on model performance.



Each matrix cell in this plot shows the interaction effect between two features, and the diagonal cells show the primary impact of each feature on the predictions made by the model. Pink crosses represent the interaction values, which are mainly centered around zero. This indicates that there aren't any significant interactions between the various temperature sensors. This implies that although the predictions of the model may be affected by individual temperatures, the overall effect of changing two temperatures at the same time does not significantly differ from what would be predicted by considering each temperature's individual contributions alone. Understanding the fundamental dynamics of the model requires this kind of insight, particularly in complex systems where interactions between variables may have a substantial impact on results. Such details can direct future feature engineering, data collection, and model improvement to improve interpretability and predictive performance.

## Discussion and Conclusion

The report discussed the data processing and analytics project from start to finish including the Gallery database implementation in multiple database management systems as well as different query test cases for the database.

For part A the gallery project was chosen due to the breadth of data available and possibilities when it comes to outlining the different entities and the real-world use cases available to us. MongoDB, SQL and Neo4j were used for these purposes with the advantages of each being maximized and highlighted throughout the project. Before implementing in any database management system though, the process of drawing a relational schema was outlined in detail. The process included identifying primary keys, foreign keys and the different relationships present between the entities. The relational schema acts as a good starting point for the project and allowed for the gallery database to be outlined clearly.

All the entities were implemented in SQL and the strengths of it were highlighted. The main strength of SQL being the ability to use join tables to illustrate many to many relationships in a straightforward manner. This was particularly helpful for the many to many relationships including the ones between galleries and artworks.

Neo4j was also used as another database management system. Being a graph-based database management system means that Neo4j represents the data as nodes with lines being the relationship between them. This helps with visualizing data that is complex in nature. Four of the entities were chosen for Neo4j based on real world use cases. The four entities were the ones considered to be at the heart of any art gallery database including the customers, artworks and galleries.

The same steps were followed for MongoDB where it is a powerful database management system capable of performing complex queries to the dataset. For all three DBMSs a set of queries was showcased, and the results are outlined in each section. Overall, Part A can be considered a major success where multiple technologies were used and analysed based on performance and suitability for the task at hand.

Part B consisted of an artificial intelligence project on a dataset of choice. The dataset chosen was the sensor dataset with a target variable of number of room occupants. This dataset was chosen by the group due to the fact that it’s a real-world problem and a potential solution can be applied in various use cases including energy saving projects.

After briefly inspecting the dataset and ensuring there are no missing values and outliers, a selection of models was implemented to predict the target variable, occupancy count. The first model is a simple neural network. The rationale behind selecting a simple neural network was to act as a baseline for results. This baseline resulted in a very accurate model with accuracies reaching 99%. The second model was a random forest classifier. This is an ensemble method with strong predictive power and a great ability to pick up on complex relationships between the data. Again, the random forest model performed exceptionally well with accuracies reaching 99%.

PCA was used to try and represent the data in a simpler method with fewer features. The main finding from PCA was the fact that over 93% of the variation in the dataset can be represented by the first 9 components. That much variation is almost always enough for a model to be built with the added benefit of reducing features to avoid overfitting.

After implementing the models, a correlation-based feature selection algorithm was used to simplify the models. The correlation-based method was used due to the high correlations seen between the numerical variables in the dataset. The feature selection methods resulted in slight improvements to what was an already accurate model.

Explainable AI methods like permutation feature importance and SHAP in efforts to explain the results of the artificial intelligence model. Permutation feature importance shows us the effect of removing features on the accuracy of the model whereas SHAP gives insight into how each feature is contributing to the prediction of the model.

Overall, for part B many of the objectives have been met including building an artificial intelligence for the dataset, PCA analysis and applying explainable AI methods.