# **CSCU9B3 Report**

By 2721301

Contents

[**CSCU9B3 Report** 1](#_Toc24979964)

[**Task A** 3](#_Toc24979965)

[**Second Normal Form** 3](#_Toc24979966)

[**Third Normal Form** 5](#_Toc24979967)

[**Why Third Normal Form?** 6](#_Toc24979968)

[**Task B** 7](#_Toc24979969)

[**Table Creation** 7](#_Toc24979970)

[**Keys** 9](#_Toc24979971)

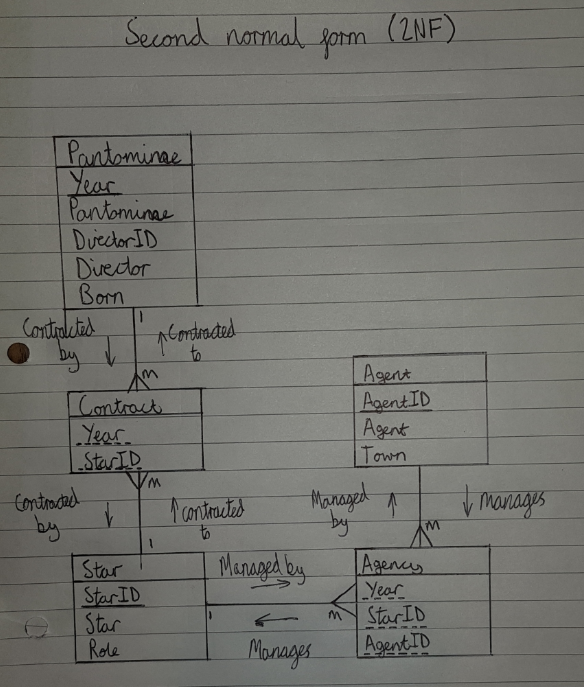
[**Task C** 11](#_Toc24979972)

[**Task D** 16](#_Toc24979973)

## **Task A**

Design your database using E-R modelling and normalisation

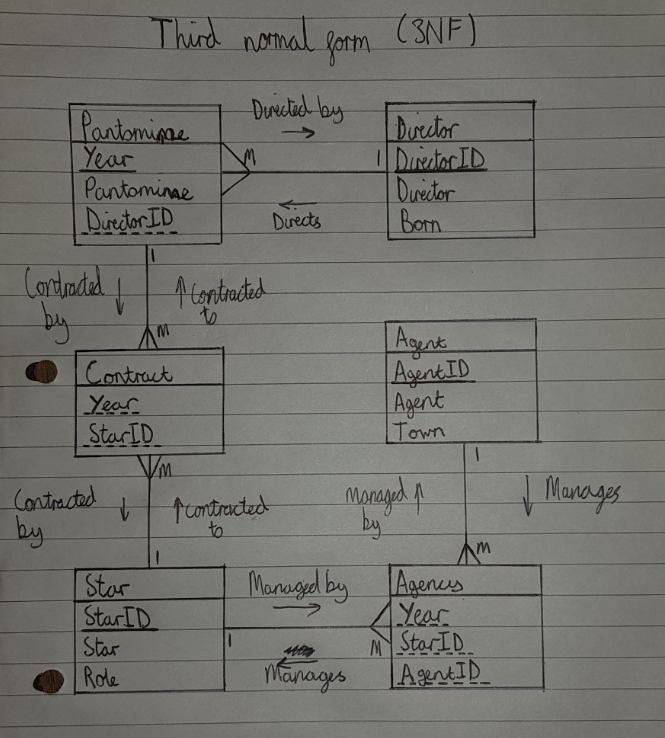
### **Second Normal Form**



I started designing this database by examining the data provided and created an entity relationship diagram achieving second normal form, as the data was already in first normal form. I created the tables Pantomime, Star and Agent in order to get rid of repeating groups in the data, this is called non-loss decomposition. I am assuming that each pantomime has a star, and each star has an agent. The relationship between Pantomime and Star was many to many so I added the contract table to get rid of the many to many relationships. One pantomime has many contracts for stars and many stars are contracted to one pantomime. Star to agent was also many to many, so I added the agency table. One star has an agency contract to many agents and one agent has an agency contract to many stars. Every relationship is mandatory as the pantomime needs stars and their contracts, stars need to be contracted to a pantomime, every star needs contracted to an agent and every agent needs contracted by a star. The primary keys have a solid underline and foreign keys have a dashed underline. Star relates to the pantomime table using the “year” and “StarID” as foreign keys in the contract table. And star is related to agent using “StarID” and “AgentID” as foreign keys in the agency table. I initially assumed that “Pantomime” should be the primary key for the pantomime table but changed it to “Year” after I realised the values in “Year” are unique for every record whereas “Pantomime” is not.

Having the data in second normal form gets rid of the update, deletion and insertion anomalies that first normal form will cause. Second normal form saves space by avoiding repetitions.

### **Third Normal Form**

This ER diagram now achieves third normal form as every non-key column is non-transitive on the primary key. I done this by separating the director and the pantomime into separate tables as the attributes “Director” and “Born” relied on the “DirectorID” and not the primary key “Year” in the pantomime table, making it transitive. The relationships in this table are one pantomime is directed by one director, one director directs many pantomimes, one pantomime is contracted by many contracts, one contract is contracted to one pantomime, one contract is contracted to one star, one star is contracted by many contracts, one star is managed many agency contracts, one agency contract manages one star, one agency contract is managed by one agent and one agent manages many agency contracts. Every relationship is mandatory as it requires a record in every field. No attributes in this table are transitive to the primary key and it ensures referential integrity, every column of each table only relies on its primary key. In the diagram each primary key is shown with a solid underline and each foreign key is shown with a dashed underline.

### **Why Third Normal Form?**

Third normal form is very desirable for the maintenance of databases, it ensures referential integrity within the database and has many benefits. These benefits include preventing data anomalies such as update anomalies and deletion anomalies which would cause problems in the database whenever data is altered, it greatly reduces the maintenance required on a database over time and makes the tables available to different types of queries. Third normal form allows non-prime attributes to only depend on their primary key. An example of the problems caused without 3NF would be the pantomime table in my second normal form ER diagram. This table included the Directors name and birthplace which relies on the “DirectorID” and not the primary key of that table. This means that if we were to update any of the attributes of the director, for example mis-information on their birthplace, the pantomime table would have to update the record of every single Pantomime they directed. If deleting this director, it would face this same problem.

## **Task B**

MySQL database implementation

### **Table Creation**

The following is the SQL code used to create every table from the final E-R diagram.

Pantomime:

CREATE TABLE Pantomime

(

Year int NOT NULL,

Pantomime varchar(255),

DirectorID int NOT NULL

);

Director

CREATE TABLE Director

(

DirectorID int NOT NULL,

Director varchar(255),

Born varchar(255)

);

Contract

CREATE TABLE Contract

(

StarID int NOT NULL,

Year int NOT NULL

);

Star

CREATE TABLE Star

(

StarID int NOT NULL,

Star varchar(255),

Role varchar(255)

);

Agency

CREATE TABLE Agency

(

Year int NOT NULL,

AgentID int NOT NULL,

StarID int NOT NULL

);

Agent

CREATE TABLE Agent

(

AgentID int NOT NULL,

Agent varchar(255),

Town varchar(255)

);

### **Keys**

The following sets the primary keys of the tables and the foreign keys to relate the tables.

**Primary**

Pantomime

ALTER TABLE pantomime

ADD PRIMARY KEY (Year);

Agent

ALTER TABLE agent

ADD PRIMARY KEY (AgentID);

Director

ALTER TABLE director

ADD PRIMARY KEY (DirectorID);

Star

ALTER TABLE star

ADD PRIMARY KEY (StarID);

**Foreign**

Pantomime to Director

ALTER TABLE pantomime

ADD FOREIGN KEY (DirectorID) REFERENCES director(DirectorID);

Contract to Pantomime and Star

ALTER TABLE contract

ADD FOREIGN KEY (Year) REFERENCES pantomime(Year),

ADD FOREIGN KEY (StarID) REFERENCES star(StarID);

Agency to Pantomime, Agent and Star

ALTER TABLE agency

ADD FOREIGN KEY (Year) REFERENCES pantomime(Year),

ADD FOREIGN KEY (AgentID) REFERENCES agent(AgentID),

ADD FOREIGN KEY (StarID) REFERENCES star(StarID);

## **Task C**

Querying your database

The following is the code I used to query my database, and a screenshot of the result of each query.

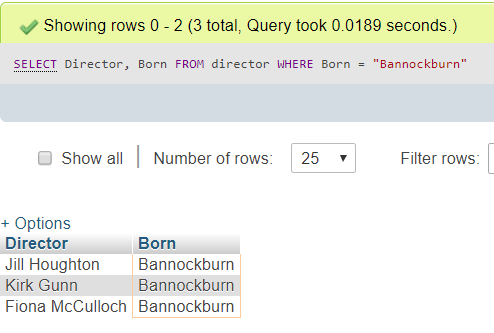
1. List the names of all directors from Bannockburn.

Code

SELECT Director, Born FROM director

WHERE Born = "Bannockburn";

Results



1. Display a list, with suitable column headings, of how often each pantomime has been performed, shown in ascending order of the number of performances.

Code

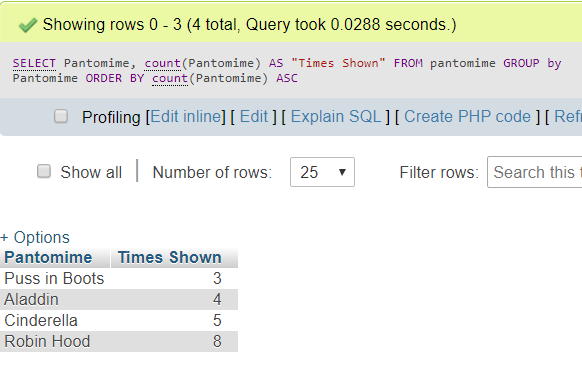
SELECT Pantomime, count(Pantomime) AS "Times Shown"

FROM pantomime

GROUP by Pantomime

ORDER BY count(Pantomime) ASC;

Results



1. Display the total number of performances ever given by each type of star role.

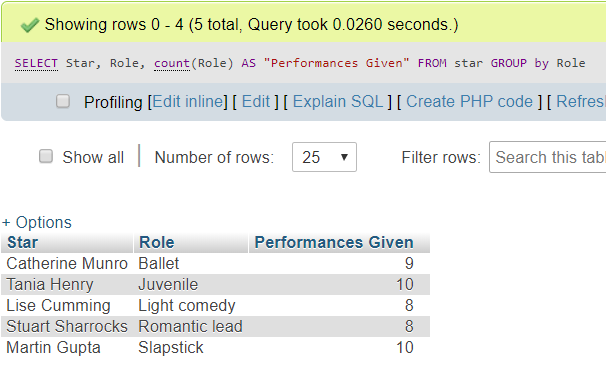
Code

SELECT Star, Role, count(Role) AS "Performances Given"

FROM star

GROUP by Role;

Results



1. Display a list of all directors who have directed the pantomime “Aladdin” and in what year.

Code

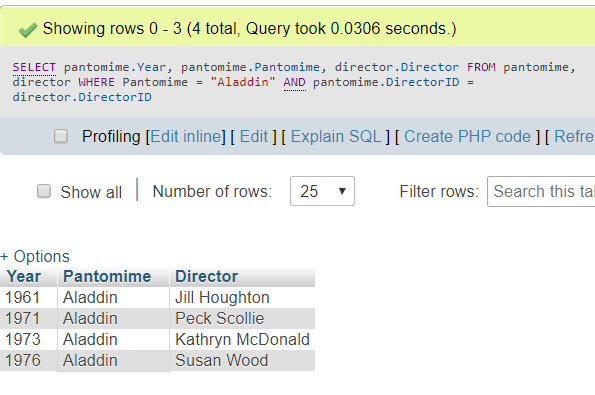
SELECT pantomime.Year, pantomime.Pantomime, director.Director

FROM pantomime, director

WHERE Pantomime = "Aladdin"

AND pantomime.DirectorID = director.DirectorID;

Results



1. List all the stars and their roles from final performance (determined as a part of this query) of the pantomime “Aladdin”.

Code

SELECT pantomime.Year, pantomime.Pantomime, star.Star, star.Role

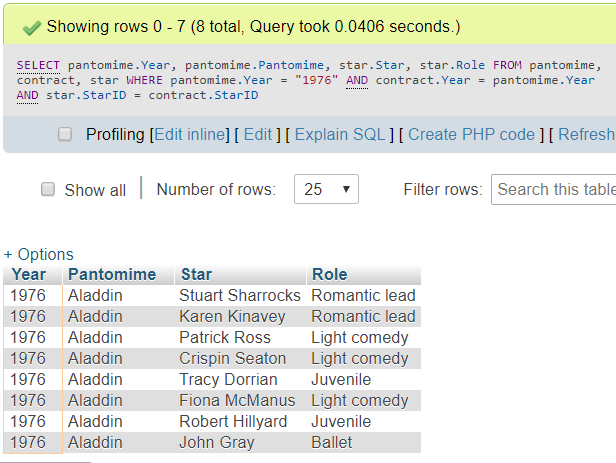
FROM pantomime, contract, star

WHERE pantomime.Year = "1976"

AND contract.Year = pantomime.Year

AND star.StarID = contract.StarID;

Results



## **Task D**

Use PHP to interface to your database

The following is code that gives the database a user interface. The purpose of this interface is if the user searches for a star, it will display the stars ID, their name, role type, their agent and shows which year they were managed by them. The code has comments explaining the lines of code.

Code

<html>

<head>

<title>Assignment Template</title>

</head>

<body>

<form action ="assignment.php" method="post">

Search star name: <input type="text" name="name">

<input type="submit">

</form>

<?php

// procedural version

$servername = "wamp0.cs.stir.ac.uk";

$username = "XXXX";

$password = "XXXX";

$database = "XXXX";

//will post if input box is not empty

if(!empty($\_POST['name'])) {

$name = $\_POST['name'];

//posts what is in the input box

// Create connection

$conn = mysqli\_connect($servername, $username, $password, $database);

// Check connection

if (!$conn) { //runs if no connection

die("Connection failed: " . mysqli\_connect\_error());

}

echo "Connected successfully to database.<br><br>"; //runs if connection

$name=mysqli\_real\_escape\_string($conn,$name);

$name=strip\_tags($name);

$sql = "SELECT star.StarID, star.Star, star.Role, agent.Agent, agency.Year FROM star, agency, agent WHERE star.Star LIKE '%".$name."%' AND star.StarID=agency.StarID AND agency.AgentID=agent.AgentID";

//select statement

$result = mysqli\_query($conn, $sql);

//assigs result to whatever query returns

if (!$result) {

echo "Search produced an error: ". mysqli\_error(); //error if no result

}

else {

echo "Results are:<br>";

if(mysqli\_num\_rows($result)==0){ //runs if no results returned

echo "No results of this star";

}else{ //runs if there are results

// output data of each row

while($row = mysqli\_fetch\_row($result)) {

echo "StarID: " . $row[0]. " - Star: " . $row[1]. " - Role: " . $row[2]. " - Agent: " . $row[3]." - Year: " . $row[4]. "<br>";

}

}

}

mysqli\_close($conn);

}

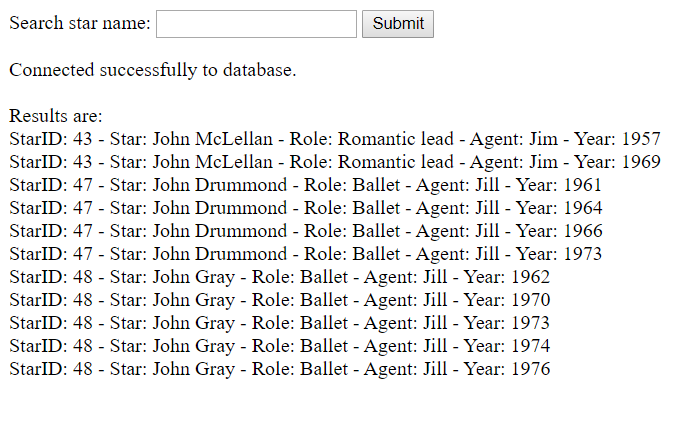
?>

</body>

</html>

Result

The following screenshot are the results shown when searching for the star “John”.



This page can be accessed with the following link:

<http://wamp0.cs.stir.ac.uk/brs00040/assignment.php>