



## Design Document

Student Name: Aaron O' Brien

Student ID: W20093613

Course: Multimedia Applications Development

Year: 2

Date: 03 / 02 / 2022

Module: Database Systems

Lecturer: Mary Fitzgerald

# Contents

<b>Business Description .....</b>	<b>3</b>
<b>Conceptual Data Model for GameStop .....</b>	<b>4</b>
<b>Logical Data Model for GameStop .....</b>	<b>7</b>
<b>Storage Representation.....</b>	<b>14</b>
<b>Table Design .....</b>	<b>24</b>
<b>Queries.....</b>	<b>32</b>
<b>Security .....</b>	<b>35</b>
<b>Conclusion .....</b>	<b>41</b>

## Business Description

GameStop is an electronic shop that focuses on gaming. Selling products such as games, consoles to run those games and additional accessories (i.e.: headsets). It was first founded in the United States of America, going as far back as to 1984. James McCurry and Gary M. Kusin created the first vision of GameStop named “Babbage” in Dallas, Texas. Selling video games for the popular Atari 2600 console.

EB Games, another video-game retail company was bought by GameStop in 2005. Making GameStop expand to Europe, Canada, Australia and New Zealand. As GameStop become more global, more stores were opened worldwide. Ireland used to have its own game store named Gamesworld, which would then be purchase by GameStop in 2003. Meaning GameStop took over all the Gamesworld’s stores and then opening new stores within Ireland during the 2000s and to this modern day.

GameStop Ireland has one of their stores located in Waterford City, co. Waterford. As mentioned before, it’s a retail store with a primary focus on selling gaming products. Products such as physical game discs to run software on powerful consoles such as the Sony PlayStation 5 and Microsoft Xbox Series X.

Accessories can be purchased to support customer’s consoles. They come in a variety of products such as headsets and keyboards. GameStop Ireland also offers a free repair service for broken gaming consoles.

For my design document, I will focus on a scenario where a customer wants to make an order inside a GameStop store, mainly Christmas gifts. Since it will be during Christmas season, it will be the busiest time for retail stores such as GameStop.

GameStop Ireland employs both part- and full-time workers. There will be more suppliers due the high demand at the Christmas season. The range of products can be the game’s software, the console or any merchandise of a popular franchise (i.e. Pokémon).

# Conceptual Data Model for GameStop

Boundary (In): What is related to the customer in the store. Such as the staff, customers, products, orders.

Boundary (Out): What is non-related to the customer in the store. Such as maintenance, timetable, property billing, finance.

The following classes chosen:

- Customer
- Order
- gameProduct (Super Class)
- Employee (Super Class)
- Supplier
- Shipment

The following sub-classes chosen:

- FullTimeEmployees (Employee)
- PartTimeEmployees (Employee)
- game\_software (gameProduct)
- game\_console (gameProduct)
- game\_Merchadise (gameProduct)

Customers are ordering inside the store. Meaning the attributes customerId, name (fName, IName), address (street, town, and country) and contact number (Multi-attribute) are needed.

Each order will need an orderNum, orderDate, and totalCost.

Employees are needed to record the customer's orders. I need their employeeId, name (fName, IName), contact number (Multi-attribute).

Each order will contain the attributes productNum, productName, unitInStock.

Each supplier will send the orders in a shipment. Details needed for suppliers are the supplier number, supplier name (fName, lName), address (street, town and country), contact number and email address.

The employee receives the shipment which contains the orders, this will include details such as the orderDescription and time.

### **How it works:**

Each customer places 0 or more orders. Each order involves one and one customer only. Customer 1 **contains** 0..\* Order.

Each Order contains at least one gameProduct and each gameProduct is assigned to 0 or more orders. Order 0..\* **contains** 1..\* gameProduct.

Each employee handles 0 or more orders, each Order will be process by at least one employee. Employee 1..\* **processes** 0..\* Order.

Each Supplier sends the orders to one and only one Shipment. Orders will be sent by one or more employee. Supplier 1..\* **sends** 1 Shipment.

Each Employee receives the orders from one and only one shipment. Each shipment is taken by at least 1 or more employees. Shipment 1 **receives** 1..\* Employee.

### **Association Class / Relationship Attributes:**

OrderDetail: Contains the attributes quantity and unitPrice.

These are not attributes for Order because it would have multiple values for quantity and unitPrice, with each order containing gameProduct

### **Super Class & Sub-Classes:**

- 1) Each employee must be either a FullTimeEmployee or PartTimeEmployee. The attributes for a full-time worker would be Salary and Bonus. The attributes for a part time worker would HourlyRate and NoOfHoursWorked.

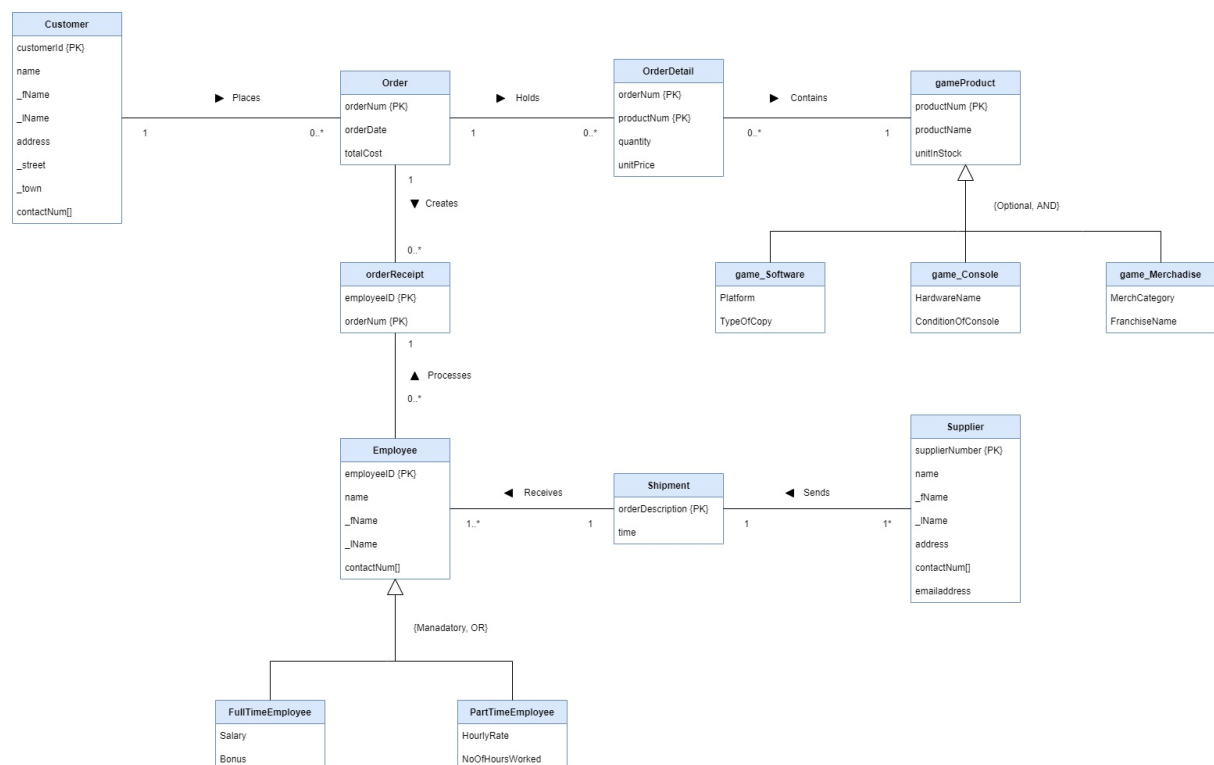
When I use the entity Employee, one of the sub classes must be chosen due to the {Mandatory, OR} constraint. The attributes of a subclass are then joining with the Employee (Superclass).

- 2) Each gameProduct (Superclass) can be optional, giving them a choice between these subclasses. The attributes of game\_Software are Platform, what gaming hardware does the game run. TypeOfCopy is whether the software is a physical disc or is it digital (downloading online).

The game\_Console attributes are HardwareName and ConditionOfConsole, is the console brand new or used. The game\_Merchadise attributes are MerchCategory or FranchiseName.

The entity gameProduct is a Superclass, meaning the attributes of a subclass are joined with gameProduct.

## EER Model



# Logical Data Model for GameStop

Step 1: Mapping all the entity types to a set of relations.

- Customer (customerId, fname, lname, street, town, contactNum)

**PRIMARY KEY:** customerId

- Order (orderNum, orderDate, totalCost)

**PRIMARY KEY:** orderNum

- gameProduct (productNum, productName, unitInStock)

**PRIMARY KEY:** productNum

- Employee (employeeID, fname, lname, contactNum)

**PRIMARY KEY:** employeeID

- Supplier (supplierNumber, fname, lname, street, town, county, contactNum, emailaddress)

**PRIMARY KEY:** supplierNumber

- Shipment (orderDescription, time)

**PRIMARY KEY:** orderDescription

The superclasses along with its subclasses can be implemented in three ways:

### Employee

#### a) One Table

- Employee (employeeID, fname, lname, contactNum, Salary, Bonus, HourlyRate, NoOfHoursWorked)

PRIMARY KEY: employeeID

#### b) Two Tables

- FullTimeEmployee (employeeID, fname, lname, contactNum, Salary, Bonus)

PRIMARY KEY: employeeID

- PartTimeEmployee (employeeID, fname, lname, contactNum, HourlyRate, NoOfHoursWorked)

PRIMARY KEY: employeeID

#### c) Three Tables

- Employee (employeeID, fname, lname, contactNum)

PRIMARY KEY: employeeID

- FullTimeEmployee (employeeID, Salary, Bonus)

PRIMARY KEY: employeeID

FOREIGN KEY: employeeID references Employee (employeeID)



- PartTimeEmployee (employeeID, HourlyRate, NoOfHoursWorked)

**PRIMARY KEY:** employeeID

**FOREIGN KEY:** employeeID references Employee (employeeID)

### gameProduct

#### a) One Table

- gameProduct (productNum, productName, unitInStock, Platform, TypeOfCopy, HardwareName, ConditionOfConsole, MerchCategory, FranchiseName)

**PRIMARY KEY:** productNum

#### b) Two Tables

- game\_Software (productNum, productName, unitInStock, Platform, TypeOfCopy)

**PRIMARY KEY:** productNum

- game\_Console (productNum, productName, unitInStock, HardwareName, ConditionOfConsole)

**PRIMARY KEY:** productNum

- game\_Merchandise (productNum, productName, unitInStock, MerchCategory, FranchiseName)

**PRIMARY KEY:** productNum

### c) Three Tables

- gameProduct (productNum, productName, unitInStock)

**PRIMARY KEY:** productNum

- game\_Console (productNum, HardwareName, ConditionOfConsole)

**PRIMARY KEY:** productNum

**FOREIGN KEY:** productNum references gameProduct (productNum)

- game\_Merchandise (productNum, MerchCategory, FranchiseName)

**PRIMARY KEY:** productNum

**FOREIGN KEY:** productNum references gameProduct (productNum)

Step 2: Mapping the relationships.

1) “Places” is the relationship between Customer and Order and is a 1:0..\* relationship. This means I can put the primary key of Customer onto Order since Order is the many side. Creating a foreign key inside Order.

The updated table for Order:

- Order (orderNum, orderDate, totalCost, customerId)

**PRIMARY KEY:** orderNum

**FOREIGN KEY:** customerId references Customer (customerId)

2) The “Receives” relationship between Shipment and Employee is a 1:1..\* relationship. Primary key of Shipment goes to Employee (many side) as a foreign key.

The updated table for Employee:

- Employee (employeeID, fname, lname, contactNum, orderDescription)

**PRIMARY KEY:** employeeId

**FOREIGN KEY:** orderDescription references Shipment (orderDescription)

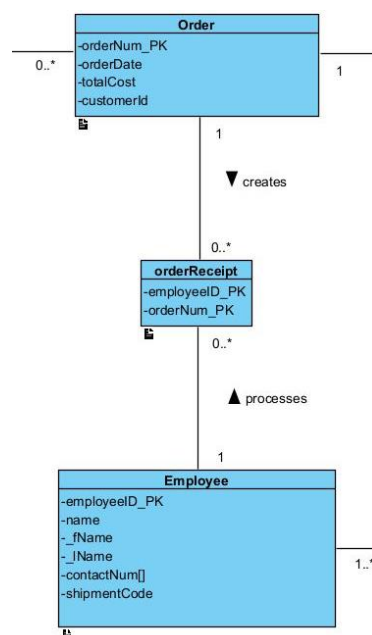
3) The “Processes” relationship between Employee and Order is a 1\*:0\* or a \*: \* (many to many) relationship. This time I need to create a new relation. Then I can insert the primary keys from both tables in the relationship of this new relation as the foreign key values. This creates a new table too.

- orderReceipt (employeeID, orderNum)

**PRIMARY KEY:** employeeID, orderNum

**FOREIGN KEY:** employeeID references Employee (employeeID)

**FOREIGN KEY:** orderNum references Order (orderNum)



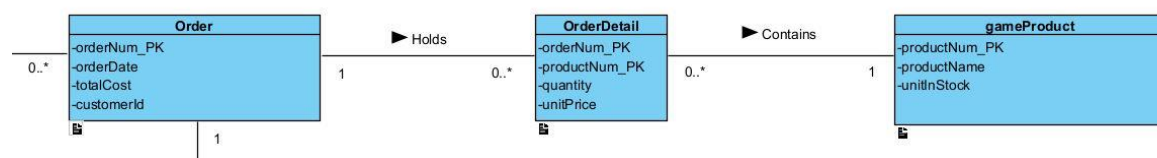
4) “Contains” relationship between Order and gameProduct is a 0..\* : 1..\* or a many to many relationship. Like the “Processes”, I need to create a new table with the Primary keys and foreign keys from Order and gameProduct.

- OrderDetail (orderNum, productNum, quantity, unitPrice)

**PRIMARY KEY:** orderNum, productNum.

**FOREIGN KEY:** orderNum references Order (orderNum)

**FOREIGN KEY:** productNum references gameProduct (productNum)



5) “Sends” relationship between Supplier and Shipment is a 1 \*:1, a one-to-many relationship.

Step 3: Full Tables (without subclasses, as it was done earlier)

- Customer (customerId, fname, lname, street, town, county, contactNum)

**PRIMARY KEY:** customerId

- Order (orderNum, orderDate, totalCost, customerId)

**PRIMARY KEY:** orderNum

**FOREIGN KEY:** customerId references Customer (customerId)

- Employee (employeeID, fname, lname, contactNum, orderDescription)

**PRIMARY KEY:** employeeID

**FOREIGN KEY:** orderDescription references Shipment  
(orderDescription)

- gameProduct (productNum, productName, unitInStock)

**PRIMARY KEY:** productNum

- Supplier (supplierNumber, fname, lname, street, town, county,  
contactNum, emailaddress)

**PRIMARY KEY:** supplierNumber

- orderReceipt (employeeID, orderNum)

**PRIMARY KEY:** employeeID, orderNum

**FOREIGN KEY:** employeeID references Employee (employeeID)

**FOREIGN KEY:** orderNum references Order (orderNum)

- OrderDetail (orderNum, productNum, quantity, unitPrice)

**PRIMARY KEY:** orderNum, productNum.

**FOREIGN KEY:** orderNum references Order (orderNum)

**FOREIGN KEY:** productNum references gameProduct  
(productNum)

- Sends (orderDescription, time)

**PRIMARY KEY:** orderDescription

# Storage Representation

## 1) Employee

We have 20 employees working in GameStop, Waterford. There are 14 full time and 6 part time employees. Due to Employee being a superclass, I will have to calculate the number of bytes in each subclasses. After that, I will compare each subclass and choose the table option with the least number of bytes for the total storage at the end.

### a) One Table

(Employee with All Attributes)

Attributes	Bytes	Field Type
employeeID	6	Char
name	20	Char
contactNum	16	Char
Salary	5	Char
Bonus	5	Decimal (5,2)
HourlyRate	6	Decimal (4,2)
NoOfHoursWorked	2	Char
<b>Total Bytes</b>	<b>60</b>	

Storage for Employee (Superclass) table:

$6 + 20 + 16 + 5 + 5 + 6 + 2 = 60$  bytes for each employee

$20 * 60 = 1200$  bytes in total

## b) Two Table

(FullTimeEmployees)

Attributes	Bytes	Field Type
employeeID	6	Char
name	20	Char
contactNum	16	Char
Salary	5	Char
Bonus	5	Decimal (5,2)
<b>Total Bytes</b>	<b>52</b>	

Storage for FullTimeEmployee (Subclass) table:

$6 + 20 + 16 + 5 + 5 = 52$  bytes for each full-time employee

$14 * 52 = 728$  bytes in total

(PartTimeEmployees)

Attributes	Bytes	Field Type
employeeID	6	Char
name	20	Char
contactNum	16	Char
HourlyRate	6	Decimal (4,2)
NoOfHoursWorked	2	Char
<b>Total Bytes</b>	<b>50</b>	

Storage for PartTimeEmployee (Subclass) table:

$6 + 20 + 16 + 6 + 2 = 50$  bytes for each part-time employee

$6 * 50 = 300$  bytes

$728 + 300 = 1028$  bytes of storage needed for this option.

### c) Three Table

(Employee)

Attributes	Bytes	Field Type
employeeID	6	Char
name	20	Char
contactNum	16	Char
<b>Total Bytes</b>	<b>42</b>	

$$6 + 20 + 16 = 42 \text{ bytes}$$

$$20 * 42 = \mathbf{840} \text{ bytes of storage}$$

(FullTimeEmployee with Foreign Key)

Attributes	Bytes	Field Type
employeeID	6	Char
Salary	5	Char
Bonus	5	Decimal (5,2)
<b>Total Bytes</b>	<b>16</b>	

$$6 + 5 + 5 = 16 \text{ bytes}$$

$$14 * 16 = \mathbf{224} \text{ bytes of storage}$$

(PartTimeEmployee with Foreign Key)

Attributes	Bytes	Field Type
employeeID	6	Char
HourlyRate	6	Decimal (4,2)
NoOfHoursWorked	2	Char
<b>Total Bytes</b>	<b>14</b>	

$$6 + 6 + 2 = 14 \text{ bytes}$$

$$6 * 14 = \mathbf{84} \text{ bytes of storage}$$



$$840 + 224 + 84 = \mathbf{1148 \text{ bytes in total}}$$

In conclusion, I would pick the Two Table option as it has the least amount of storage (1028 bytes).

## 2) gameProduct

We have 100 gaming products that can be ordered in GameStop, Waterford. There are 40 gaming software, 35 gaming consoles and 25 gaming merchandise. The entity, gameProduct is a superclass. Like my other superclass Employee. I must calculate the number of bytes in each subclasses. After that, I will compare each subclass and choose the table option with the least number of bytes for the total storage at the end. The reason is that the less amount of storage, the faster the GameStop Database responds and receives data.

### **a) One Table**

(gameProduct with All Attributes)

Attributes	Bytes	Field Type
productNum	9	Char
productName	20	Char
unitInStock	5	Char
Platform	20	Char
TypeOfCopy	8	Char
HardwareName	15	Char
ConditionOfConsole	5	Char
MerchCategory	20	Char
FranchiseName	25	Char
<b>Total Bytes</b>	<b>127</b>	

$$9 + 20 + 5 + 20 + 8 + 15 + 5 + 20 + 25 = 127 \text{ bytes}$$

$$100 * 127 = \mathbf{12700 \text{ bytes}}$$

## b) Two Table

(game\_Software)

Attributes	Bytes	Field Type
productNum	9	Char
productName	20	Char
unitInStock	5	Char
Platform	20	Char
TypeOfCopy	8	Char
<b>Total Bytes</b>	<b>62</b>	

Storage for game\_Software (Subclass) table:

$9 + 20 + 5 + 20 + 8 = 62$  bytes for each game\_Software

$40 * 62 = 2480$  bytes in total

(game\_Console)

Attributes	Bytes	Field Type
productNum	9	Char
productName	20	Char
unitInStock	5	Char
HardwareName	15	Char
ConditionOfConsole	5	Char
<b>Total Bytes</b>	<b>54</b>	

Storage for game\_Console (Subclass) table:

$9 + 20 + 5 + 15 + 5 = 54$  bytes for each game\_Software

$35 * 54 = 1890$  bytes in total

(game\_Merchandise)

Attributes	Bytes	Field Type
productNum	9	Char
productName	20	Char
unitInStock	5	Char
MerchCategory	20	Char
FranchiseName	25	Char
<b>Total Bytes</b>	<b>79</b>	

Storage for game\_Merchandise (Subclass) table:

$9 + 20 + 5 + 20 + 25 = 79$  bytes for each game\_Merchandise

$25 * 79 = 1975$  bytes in total

$2480 + 1890 + 1975 = 6345$  bytes

a) Three Table

(gameProduct)

Attributes	Bytes	Field Type
productNum	9	Char
productName	20	Char
unitInStock	5	Char
<b>Total Bytes</b>	<b>34</b>	

$9 + 20 + 5 = 34$  bytes

$100 * 34 = 3400$  bytes of storage

(gameSoftware with Foreign Key)

Attributes	Bytes	Field Type
productNum	9	Char
Platform	20	Char
TypeOfCopy	8	Char
<b>Total Bytes</b>	<b>37</b>	

$9 + 20 + 8 = 37$  bytes

$100 * 37 = 3700$  bytes of storage

(gameConsole with Foreign Key)

Attributes	Bytes	Field Type
productNum	9	Char
HardwareName	15	Char
ConditionOfConsole	5	Char
<b>Total Bytes</b>	<b>29</b>	

$9 + 15 + 5 = 29$  bytes

$100 * 29 = 2900$  bytes of storage

(gameMerchandise with Foreign Key)

Attributes	Bytes	Field Type
productNum	9	Char
MerchCategory	20	Char
FranchiseName	25	Char
<b>Total Bytes</b>	<b>54</b>	

$9 + 20 + 25 = 54$  bytes

$100 * 54 = 5400$  bytes of storage

$3400 + 3700 + 2900 + 5400 = 15400$  bytes in total

In conclusion, I would pick the Two Table option as it has the least amount of storage (6345 bytes).

### 3) Customer

Attributes	Bytes	Field Type
customerId	6	Char
name	20	Char
address	25	Char
contactNum	16	Char
<b>Total Bytes</b>	<b>67</b>	

$6 + 20 + 25 + 16 = 67$  bytes for each customer

The average number of customers that would order is 15. Therefore, the total storage for Customer is  $15 * 67 = 1005$  bytes

### 4) Order

Attributes	Bytes	Field Type
orderNum	9	Char
orderDate	8	Date
totalCost	8	Decimal (6,2)
customerId	6	Char
<b>Total Bytes</b>	<b>31</b>	

$9 + 8 + 8 + 6 = 31$  bytes for each Order

The average number of Orders would be 25. The total storage for Order is  $25 * 31 = 775$  bytes

### 5) OrderDetail

Attributes	Bytes	Field Type
orderNum	9	Char
productNum	9	Char
quantity	5	Char
unitPrice	6	Decimal (4,2)
<b>Total Bytes</b>	<b>29</b>	

$9 + 9 + 5 + 6 = 23$  bytes for each OrderDetail

The average number of OrderDetail would be 25. The total storage for OrderDetail is  $25 * 29 = 725$  bytes

#### 6) OrderReceipt

Attributes	Bytes	Field Type
employeeID	6	Char
orderNum	9	Char
<b>Total Bytes</b>	<b>15</b>	

$9 + 6 = 15$  bytes for each OrderReceipt

The average number of OrderReceipt would be 25. The total storage for OrderReceipt is  $25 * 15 = 375$  bytes

#### 7) Shipment

Attributes	Bytes	Field Type
orderDescription	20	Char
time	25	Char
<b>Total Bytes</b>	<b>15</b>	

$20 + 25 = 45$  bytes for each Shipment

The average number of Shipment would be 20. The total storage for Shipment is  $20 * 15 = 300$  bytes

### 9) Supplier

Attributes	Bytes	Field Type
supplierNumber	10	Char
name	20	Char
address	25	Char
contactNum	16	Char
emailaddress	30	Char
<b>Total Bytes</b>	<b>101</b>	

$10 + 20 + 25 + 16 + 30 = 101$  bytes for each Supplier

The number of Suppliers would be 12. The total storage for Supplier is  $12 * 101 = 1212$  bytes

(Employee) + (gameProduct) + (Customer) + (Order) + (OrderDetail) + (OrderReceipt) + (Shipment) + (Supplier) = Total Number of Bytes

$1028 + 6345 + 1005 + 775 + 725 + 375 + 300 + 101 = 10654$  bytes

10654 bytes is the total storage from every table in my GameStop Database.

# Table Design

## Customer

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
customerId	varchar	6	Not null		unique	Primary key	This is a unique code for each customer.
name	varchar	20	Not Null				This is the first and last name of the customer
address	varchar	25	Not null				This is the full address of the customer.
contactNum	varchar	16	Not null				This is the mobile number of the customer.

## Order

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
orderNum	varchar	9	Not null		unique	Primary key	This is a unique code for each order.
orderDate	Date	8	Not Null				This is the date for when the order was issued.
totalCost	int	8	Not null				This is the total cost each order.
customerId	varchar	6	Not null			Foreign key	This is a unique code for each customer.



## OrderDetail

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
orderNum	varchar	9	Not null		unique	Primary key	This is a unique number for each order.
productNum	varchar	9	Not Null		unique	Primary key	This is the unique for each product.
quantity	int	5	Not null				This is the number of items that are ordered.
unitPrice	int	6	Not null				This is the price of each product.

## OrderReceipt

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
employeeID	varchar	6	Not Null		unique	Primary key	This is a unique code for each employee.
orderNum	varchar	9	Not null		unique	Primary key	This is a unique number for each order.

## Shipment

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
orderDescription	varchar	20	Not Null		unique	Primary key	This is a description of each order.
time	varchar	25	Not null				This is the duration it takes for the shipment to arrive at the store.

## Supplier

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
supplierNumber	int	10	Not Null		unique	Primary key	This is the unique code for each supplier.
name	varchar	20	Not Null				This is the first and last name of the supplier
address	varchar	25	Not null				This is the full address of the supplier.
contactNum	varchar	16	Not null				This is the mobile number of the supplier.
emailaddress	varchar	30	Not null				This is the email address of the supplier

## FullTimeEmployee

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
employeeID	varchar	6	Not Null		unique	Primary key	This is a unique code for each employee.
name	varchar	20	Not Null				This is the first and last name of the employee.
contactNum	varchar	16					This is the mobile number of the supplier.
Salary	int	5	Not Null				The total salary for each employee
Bonus	int	5					Additional money to employee but is not mandatory

## PartTimeEmployee

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
employeeID	varchar	6	Not Null		unique	Primary key	This is a unique code for each employee.
name	varchar	20	Not Null				This is the first and last name of the employee.
contactNum	varchar	16					This is the mobile number of the supplier.
hourlyRate	Smallint	6	Not Null				The hourly rate of each employee that's get paid.
NoOfHoursWorked	tinyint	2					The hours that each employee worked for the week.

## game\_Software

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
productNum	varchar	9	Not Null		unique	Primary key	This is the unique for each product.
productName	varchar	20	Not Null				This is the first and last name of the product.
unitInStock	int	5					This is the number of products in the warehouse.
Platform	varchar	20	Not Null				The product's hardware that is used to run the software.
TypeOfCopy	varchar	8	Not Null		Physical or Digital		The software can be a physical disc or be digital download.

## game\_Console

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
productNum	varchar	9	Not Null		unique	Primary key	This is the unique for each product.
productName	varchar	20	Not Null				This is the first and last name of the product.
unitInStock	int	5					This is the number of products in the warehouse.
HardwareName	varchar	15	Not Null				The name of the console.
ConditionOfConsole	varchar	5	Not Null		New or Used		The condition of the console.

## game\_Merchandise

FIELD	TYPE	SIZE	NULL/not NULL	DEFAULT	Constraints	INDEX	Description
productNum	varchar	9	Not Null		unique	Primary key	This is the unique for each product.
productName	varchar	20	Not Null				This is the first and last name of the product.
unitInStock	int	5					This is the number of products in the warehouse.
MerchCategory	varchar	20	Not Null				The category that the merch belongs to.
FranchiseName	varchar	25	Not Null				The name of the franchise that the merch belongs to.

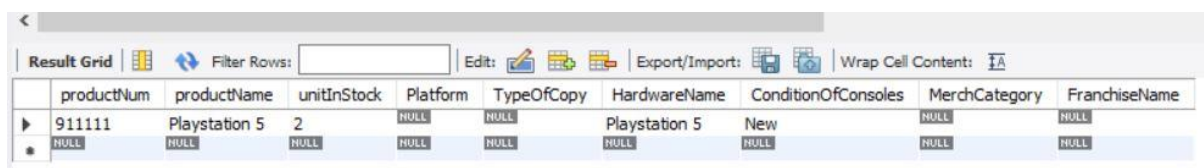
## Queries

The next section is the making the physical database for Gamestop Ireland – Waterford. Therefore, there is a lot of planning for making a database. It can be a very complex project. My assignment requires me to make two separate SQL files. Part 1 for the database itself and part 2 for executing queries.

The following text below contain eight queries I done on my database. The reason why I need to query is to check if my database is working efficiently.

**Q1: GameStop is known for selling gaming hardware consoles, check to see if they have the brand new 'PlayStation 5' console in stock?**

Answer: `SELECT * FROM gameProduct WHERE productName IN ('Playstation 5');`



	productNum	productName	unitInStock	Platform	TypeOfCopy	HardwareName	ConditionOfConsoles	MerchCategory	FranchiseName
▶	911111	Playstation 5	2	NULL	NULL	Playstation 5	New	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**Q2: What day, month and year does Gamestop expects to recieve the "LEGO PS4 GAME" delivery?**

Answer: `SELECT * FROM Shipment WHERE orderDescription IN ('ONE LEGO PS4 GAME');` (Image One)

`SELECT DATE_FORMAT(date('2021-11-20'), '%W %D %M %Y') as 'LEGO SX Delivery Date';` (Image Two)



	orderDescription	EstArrival
▶	ONE LEGO PS4 GAME	2021-11-10
*	NULL	NULL

	LEGO Delivery Date
▶	Saturday 20th November 2021



**Q3: How much stock in total, does Gamestop Waterford currently hold?**

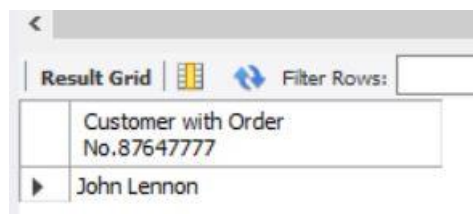
**Answer:** `SELECT SUM(unitInStock) As 'Total number of stock in Gamestop Waterford' FROM gameProduct;`



Result Grid	
	Total number of stock in Gamestop Waterford
▶	47

**Q4: What customer has the order number 87647777?\***

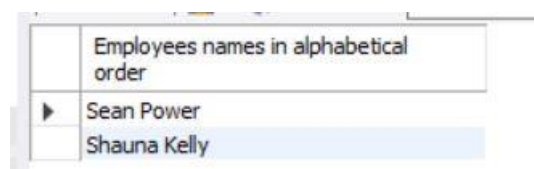
**Answer:** `SELECT name AS 'Customer with Order No. 87647777' FROM Customer WHERE customerId IN (SELECT customerId FROM Reservation WHERE orderNum = 87647777);`



Result Grid	
	Customer with Order No.87647777
▶	John Lennon

**Q5: List all employees names in alphabetical order?**

**Answer:** `SELECT name as 'Employees names in alphabetical order' FROM Employee ORDER by Name;`



Result Grid	
	Employees names in alphabetical order
▶	Sean Power
	Shauna Kelly

**Q6: What is the average number of stock does GameStop Waterford hold in store?**

**Answer:** `SELECT AVG(unitInStock) As 'Average number of stock in total in Gamestop Waterford' FROM gameProduct;`

Result Grid		Filter Rows:
	Average number of stock in total in Gamestop Waterford	
▶	11.75	

**Q7: Each Employee wants to know what date their assigned order made from customer will arrive in GameStop?**

Answer: `SELECT employeeId, name, EstArrival AS 'Assigned Order Delivery Date' FROM Shipment join Employee ON Shipment.orderDescription = Employee.orderDescription;`

	employeeId	name	Assigned Order Delivery Date
▶	3501	Sean Power	2021-11-10
	3501	Sean Power	2021-11-10
	4000	Shauna Kelly	2021-11-10
	4000	Shauna Kelly	2021-11-10

**Q8: Checking that each customer has a unique id.**

Answer: `SELECT Name, COUNT(customerId) FROM Customer GROUP BY Name Order By Name;`

Result Grid			Filter Rows:
	Name	COUNT(customerId)	
▶	Holly Bean	1	
	John Lennon	1	

# Security

Security is always an important factor when it comes down Database design. Not all users need certain privileges in a database or read everything, especially other users' personal information. With MySQL, I can add/update/remove privileges on a specific user.

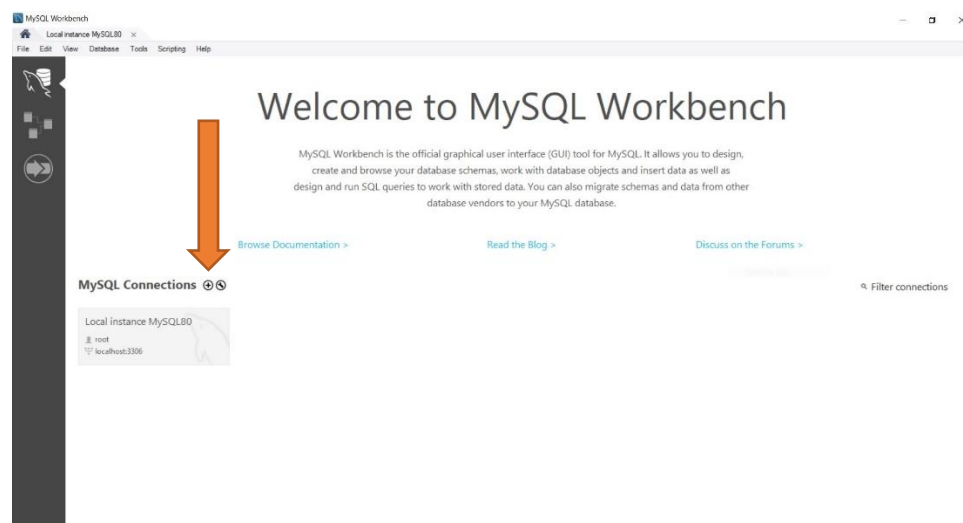
In my Gamestop database, I wish to add a user called the “Manager”. He/she will be granted all privileges, meaning they would act as a root user towards other users.

The syntax to create a new user:

**CREATE USER Username IDENTIFIED BY password**

**For creating the manager, I use this statement:**

**CREATE USER Manager IDENTIFIED BY 'BOSS';**



*Figure 1 MySQL Workspace*

In the screenshot above, I am using MySQL Workbench. I have just now created a new user named “Manager”. I want to make a new login for the manager and link my GameStop database for them to use. The first task is selecting the + button on “MySQL Connections” to create a new connection (login).

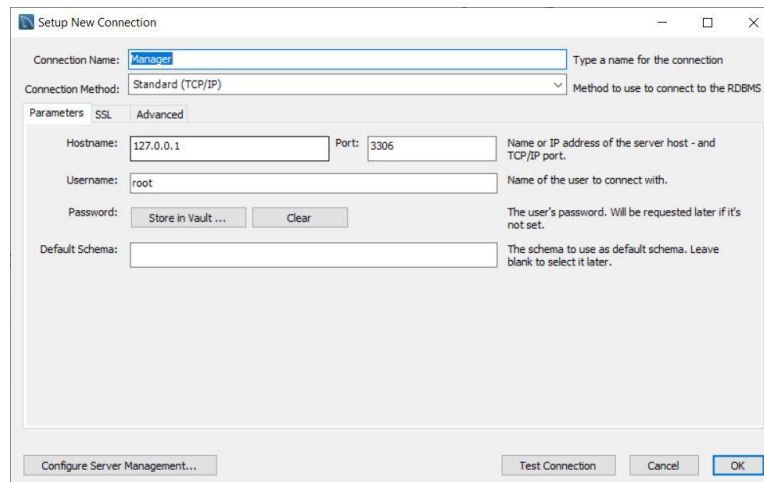


Figure 2 Setup New Connection

The new Connection Name will be “Manager”, as well is Username. The hostname remains the same. Next is the password, select Password: Store In Vault. The Manager’s password was made earlier in the CREATE USER statement, so this connection’s password is ‘BOSS’.

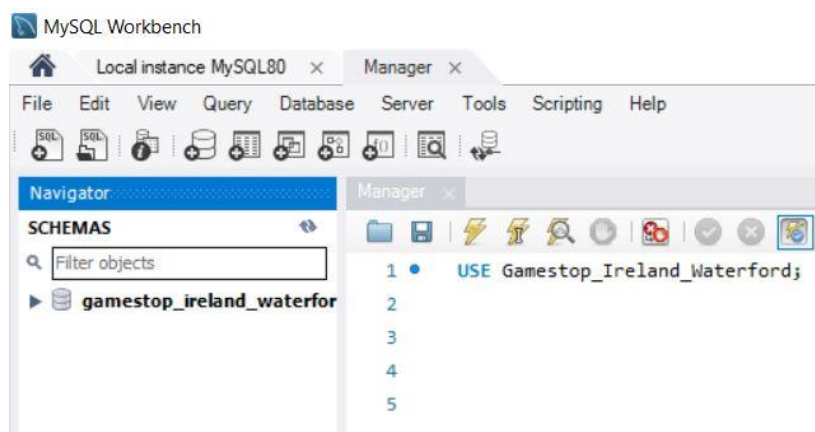


Figure 3 Manager User Using GameStop Database

Now the manager can work on their own MySQL queries on their account. However, as the root user I must grant the Manager these privileges. As the Manager, they must be given all privileges.

**To grant the Manager all privileges and grant privileges to other users:**  
**GRANT ALL ON Gamestop\_Ireland\_Waterford.\* TO Manager**  
**WITH GRANT OPTION;**

**To check if the Manager get these grants, I use the statement:**  
**SHOW GRANTS FOR Manager;**

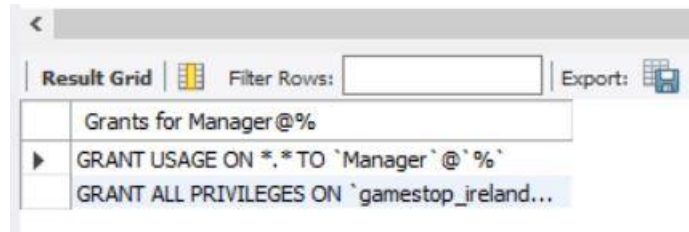


Figure 4 Results From Show Grants

I also want to make another user, named “Employee”. To do so, I repeat the same process as the manager. The only difference is that they won’t have the same privileges as the Manager.

```

84
85 • CREATE USER Employee IDENTIFIED BY 'WORKER';
86

```

Figure 5 Using CREATE USER statement for Employee

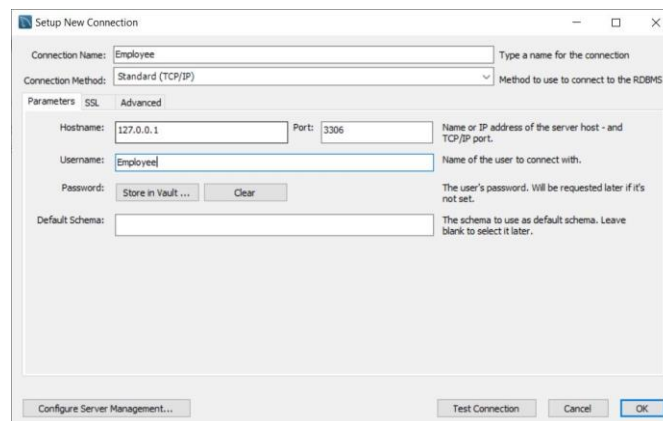


Figure 6 Setup Connection for Employee

The final step for the Employee user is giving them privileges to certain tables. This will be different to the Manager as they have higher power in the company.

1: **GRANT SELECT ON Employee TO Employee;**

This allows employees to only read their table and personal information in Employees. I did this because I don’t want the Employees to change their details, such as changing their Salary or No of Hours Worked.

2: **Grant SELECT ON** orderReceipt **TO** Employee;

**GRANT UPDATE** (orderNum) **ON** orderReceipt **TO** Employee;

Allows employees to read only the orderReceipt table, which contains the customer's order number. However, I also updated the orderNum attribute. In the case that a customer's order gets cancel, the employee can change it in the orderReceipt table.

3: **Grant SELECT ON** OrderDetail **TO** Employee;

**GRANT UPDATE** (orderNum) **ON** OrderDetail **TO** Employee;

This is the same method as the previous grant except it's the orderNum attribute in the OrderDetail table.

4: **GRANT SELECT ON** gameProduct **TO** Employee;

**GRANT UPDATE ON** gameProduct **TO** Employee;

Allows Employee to read gameProduct table and update it as the gameProduct's stock will always change. Such as the amount sold and new stock coming in.

5: **GRANT SELECT ON** Customer **TO** Employee;

**GRANT SELECT ON** Shipment **TO** Employee;

**GRANT SELECT ON** Supplier **TO** Employee;

I want the employee to read the Customer, Shipment and Supplier tables. They are not in the position to make any changes to these tables.

Views are a virtual representation of a table within a database. It allows only certain users to see specific fields (keep the rest hidden) in a table. Even when changes are made, they will reflect on the virtual table.

**Syntax =**

**CREATE VIEW** VIEW\_NAME

[(VFIELD1, VFIELD2 ...VFIELD)] **AS SELECT** STATEMENT;

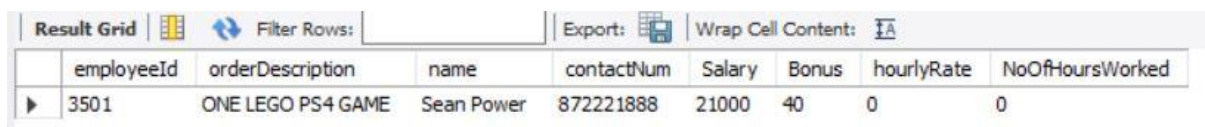
1: **CREATE VIEW** Sean\_Power\_Gamestop\_Waterford **AS SELECT**  
\* **FROM** Employee **WHERE** employeeId = 3501;

This view was made by using the **CREATE VIEW** statement. The name of view is called Sean\_Power\_Gamestop\_Waterford, as this view is only for that employee: Sean Power. The **WHERE** clause allows me to filter the search to only get values link to Sean.

I made a view for Sean because I don't believe it is right to see his co-workers information such as their Salary, Bonus, NoOfHoursWorked, and HourlyRate.

Runs the view:

**SELECT \* FROM** Sean\_Power\_Gamestop\_Waterford;



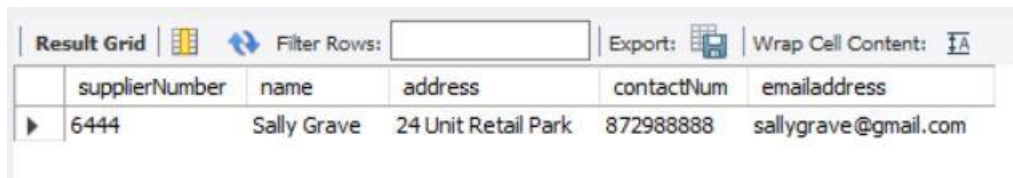
employeeId	orderDescription	name	contactNum	Salary	Bonus	hourlyRate	NoOfHoursWorked
3501	ONE LEGO PS4 GAME	Sean Power	872221888	21000	40	0	0

Figure 7 Results from Sean\_Power\_Gamestop\_Waterford

2: **CREATE VIEW** Sally\_Grave\_Dublin\_Supplier **AS SELECT** \*  
**FROM** Supplier **WHERE** supplierNumber = 6444;

I made this view because I wanted each supplier to have their own information separated. When Gamestop gets more orders, more suppliers may be needed. Meaning the list of suppliers will get bigger. This view allows to filter their search if they only want to find Sally Grave.

`SELECT * FROM Sally_Grave_Dublin_Supplier;`



The screenshot shows a SQL query result grid with a toolbar at the top. The toolbar includes a 'Result Grid' button, a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' button. The grid contains one row of data with the following columns: supplierNumber, name, address, contactNum, and emailaddress.

	supplierNumber	name	address	contactNum	emailaddress
▶	6444	Sally Grave	24 Unit Retail Park	872988888	sallygrave@gmail.com

Figure 8 Results from Sally\_Grave\_Dublin\_Supplier

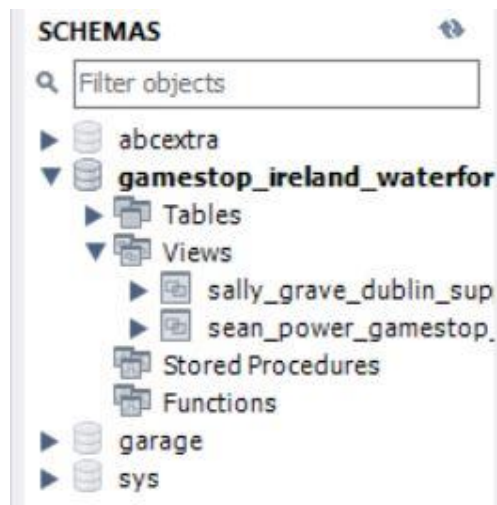


Figure 9 Updated SCHEMAS with the Views for GameStop Database



## Conclusion

I was relieved that we got Database again for the second semester as it was fresh on my mind from semester 1. Although it wasn't my favourite module, I did get over a 90% grade during New Year which I was very surprised. It was very nice I could use the same database from the last semester in this project. This is because I remember it took a long time to make up one, so I'm glad I didn't need to make a new one.

The module itself was fairly alright, material was easy to understand after reading it over a few times on Moodle. The lab practicals were alright personally, I did find it weird that we needed to log in Zoom for attendance even though we were in the room already. Then again, Covid was quite high around that time and masks were required.

As for the multi-choice exams, I found them good, and I think it's better personally because its not too intimating unlike an actual written exam. As for the project, it wasn't too hard but there was a lot of workload. Unless that is just me because this document is 41 pages long.

I enjoy the material but the MySQL section I found requires a lot of work and can be very stressful if a small error occurs and then going back to change the database again. I don't think there is much to change about the module, I found it grand and was able to manage it with my other modules. The only change I would put in is maybe if there was some pre-made videos on the SQL section. Sometimes I find that copying/paste a code is grand for labs but trying to understand it is where I find it difficult. In a video, if I see the lecturer putting the code in and explaining it in the process what the code does, really helps me out..