### UHI_Logo_CMYK_Sm

**Relational Database Management Systems – Report Template  
HND Computer Science**

|  |  |  |  |
| --- | --- | --- | --- |
| **Module Code/title:** | H16W 35 Relational Database Management Systems | | |
| **Module code**: | H16W 35 | **Word-count:** | N/A |
| **Date due/time:** | Friday 19th March 2021 11.59pm | **Date submitted:** | 25/01/2022 |
| **Student candidate Name:** | Calum Lindsay | | |
| **Student ID:** | 21010093 | | |

**Completing your assessment:**

You must submit all documentation in the appropriate Network & Information Security assessment folder on the VLE. The name the file should be **studentid\_web2.docx**. So for example, if your Student Id is: 0123456 then the file name will be 0123456\_web2.docx

**Declaration of originality and authorisation to hold this assessment electronically and verify that it is original:**

UHI recognises that plagiarism, where deliberately engaged in, is unacceptable and is considered serious academic malpractice. **Students are responsible for ensuring the work they submit is their own**. If you have any queries you should contact your Tutor or Student Adviser before submitting your assessment.

**By submitting this assessment I declare that the attached piece of work is my own**. I have acknowledged all the sources I have consulted and where I have used words which are not my own, I have clearly indicated this in the references.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student candidate number:** | 21010093 | **Date:** | 25/01/2022 |

**Assessment Evidence Required for Outcome 2 and Outcome 3:**

1. a. List / note any questions and the answers given by the client (lecturer)

**Question:** Can staff have multiple roles at the same time?

**Answer:** Staff can only have one role at a time.

**Question:** Does the database need to update the stock levels as products are sold?

**Answer:** No, the database is updated regularly with this information by a member of staff.

**Question:** Should the system provide an alert if stock falls below a certain point?

**Answer:** The stock of menu items is manually checked, however a simple way to check if an item is below a certain minimum would be useful.

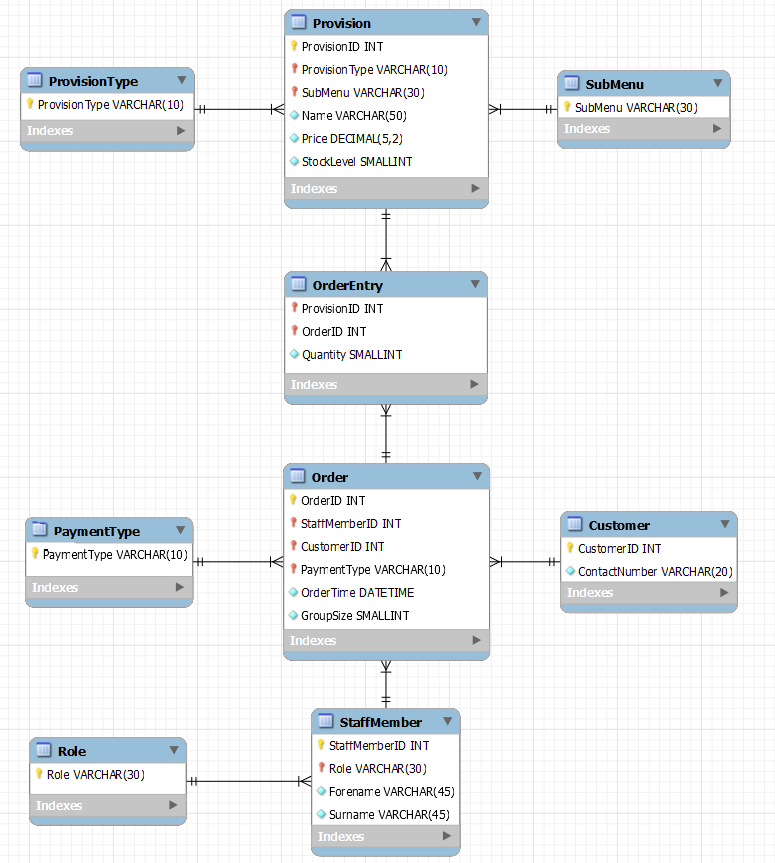
**Question:** Do menu items need to exist in the database even if they have not yet been ordered?

**Answer:** Yes as the company wishes to be able to query the database and use the result to display their menu to customers.

b. List all **assumptions** you make. Make sure to highlight:

* 1. any **structural** rules e.g. can items be on more than one menu? Are there specific staff roles?  
     + Items can only be in one sub menu (Special, Veggie, Standard, etc) and that there are no empty sub menus.
     + Staff can only have one role at a time.
     + The provision type and payment type were put in their own tables to avoid easy mistakes being made such as misspellings.
     + It is assumed that no individual item will cost more than £999.99
     + Customer details are attached to their order as they only need to be stored for track and trace purposes.
  2. any **functional** rules i.e. does/should the system provide any alerts?  
     + The system does not provide any alerts for stock management as this is not handled by the system and is rather updated manually by a member of staff.

1. **ERD Model** (Electronic Workbench Model / or similar diagram)



1. **Create table structures** corresponding to your design. Show field names, data types etc.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Field Name** | **Data Type** | **Constraint** | **Description** |
| ProvisionType | ProvisionType | VARCHAR(10) | Primary key | A type of provision (Food or Drink) |
|  |  |  |  |  |
| PaymentType | PaymentType | VARCHAR(10) | Primary key | A type of payment (Cash or Card) |
|  |  |  |  |  |
| Role | Role | VARCHAR(30) | Primary key | The work role of a member of staff |
|  |  |  |  |  |
| SubMenu | SubMenu | VARCHAR(30) | Primary key | The submenu a provision belongs to |
|  |  |  |  |  |
| Provision | ProvisionID | INT | Primary key | Unique identifier for a single Provision |
| Provision | SubMenu | VARCHAR(30) | Foreign key | The submenu a provision belongs to |
| Provision | ProvisionType | VARCHAR(10) | Foreign key | The type of the provision (Food or Drink) |
| Provision | StockLevel | SMALLINT | Mandatory field | The number of a provision in stock |
| Provision | Name | VARCHAR(50) | Mandatory field | The name of a provision |
| Provision | Price | DECIMAL(5,2) | Mandatory field | The price a provision is sold for |
|  |  |  |  |  |
| OrderEntry | ProvisionID | INT | Foreign key | A type of provision (Food or Drink) |
| OrderEntry | OrderID | INT | Foreign key | Unique identifier for a single Order |
| OrderEntry | Quantity | SMALLINT | Mandatory field | Number of a single provision ordered |
|  |  |  |  |  |
| StaffMember | StaffMemberID | INT | Primary key | Unique identifier for a member of staff |
| StaffMember | Role | VARCHAR(30) | Foreign key | The work role of a member of staff |
| StaffMember | Forename | VARCHAR(45) | Mandatory field | The forename of a member of staff |
| StaffMember | Surname | VARCHAR(45) | Mandatory field | The surname of a member of staff |
|  |  |  |  |  |
| Customer | CustomerID | INT | Primary Key | Unique identifier for a single Customer |
| Customer | ContactNumber | VARCHAR(20) | Mandatory field | The contact number for a customer |
|  |  |  |  |  |
| Order | OrderID | INT | Primary key | Unique identifier for a single Order |
| Order | PaymentType | VARCHAR(10) | Foreign key | A type of payment (Cash or Card) |
| Order | StaffMemberID | INT | Foreign key | Unique identifier for a member of staff |
| Order | CustomerID | INT | Foreign key | Unique identifier for a single Customer |
| Order | OrderTime | DATETIME | Mandatory field | The date and time an order was made |
| Order | GroupSize | SMALLINT | Mandatory field | The size of the group ordering |

1. Add appropriate test data to each of your tables.

I used the provided “fastburger” test data and added 3 Orders to October so that query 6 returns a result other than 0. I used the following query to show all the data in the database in a screenshot:

SELECT \* FROM `Order` INNER JOIN `StaffMember` ON

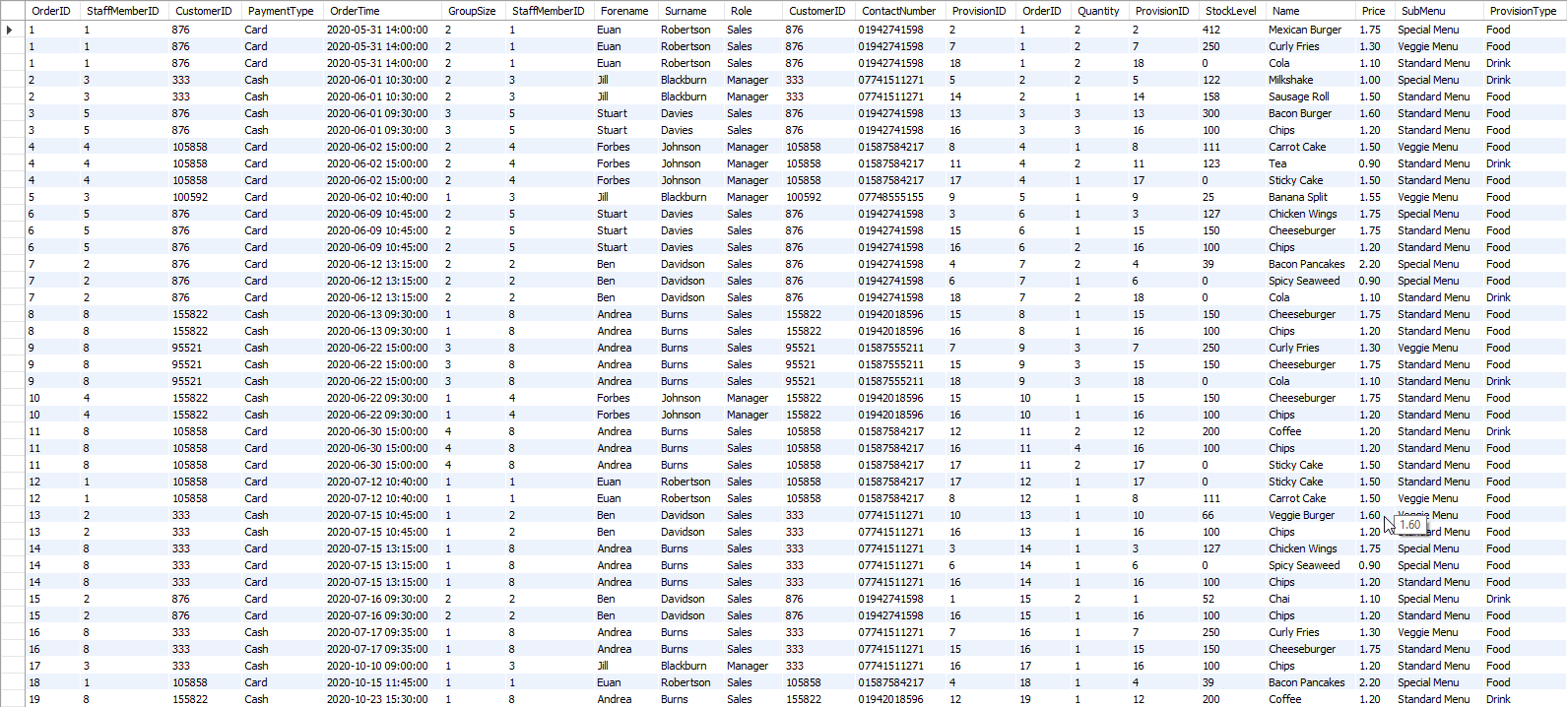
`Order`.`StaffMemberID` = `StaffMember`.`StaffMemberID` INNER JOIN

`Customer` ON `Order`.`CustomerID` = `Customer`.`CustomerID` INNER JOIN

`OrderEntry` ON `OrderEntry`.`OrderID` = `Order`.`OrderID` INNER JOIN

`Provision` ON `OrderEntry`.`ProvisionID` = `Provision`.`ProvisionID`

ORDER BY `Order`.`OrderID`



1. Document your **SQL script.** This includes creating tables and inserting data.

-- -----------------------------------------------------

-- Create Tables

-- -----------------------------------------------------

-- PaymentType Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `PaymentType` (

-- Define field

`PaymentType` VARCHAR(10) NOT NULL,

-- Define primary key

PRIMARY KEY (`PaymentType`));

-- -----------------------------------------------------

-- ProvisionType Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `ProvisionType` (

-- Define field

`ProvisionType` VARCHAR(10) NOT NULL,

-- Define primary key

PRIMARY KEY (`ProvisionType`));

-- -----------------------------------------------------

-- SubMenu Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `SubMenu` (

-- Define field

`SubMenu` VARCHAR(30) NOT NULL,

-- Define primary key

PRIMARY KEY (`SubMenu`));

-- -----------------------------------------------------

-- Role Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Role` (

-- Define field

`Role` VARCHAR(30) NOT NULL,

-- Define primary key

PRIMARY KEY (`Role`));

-- -----------------------------------------------------

-- Customer Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Customer` (

-- Define fields

`CustomerID` INT NOT NULL,

`ContactNumber` VARCHAR(20) NOT NULL,

-- Define primary key

PRIMARY KEY (`CustomerID`));

-- -----------------------------------------------------

-- StaffMember Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `StaffMember` (

-- Define fields

`StaffMemberID` INT NOT NULL AUTO\_INCREMENT,

`Forename` VARCHAR(45) NOT NULL,

`Surname` VARCHAR(45) NOT NULL,

`Role` VARCHAR(30) NOT NULL,

-- Define primary key

PRIMARY KEY (`StaffMemberID`),

CONSTRAINT `fk\_StaffMember\_Role`

FOREIGN KEY (`Role`)

REFERENCES `Role` (`Role`));

-- -----------------------------------------------------

-- Provision Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Provision` (

-- Define fields

`ProvisionID` INT NOT NULL AUTO\_INCREMENT,

`StockLevel` SMALLINT NOT NULL,

`Name` VARCHAR(50) NOT NULL,

`Price` DECIMAL(5,2) NOT NULL,

`SubMenu` VARCHAR(30) NOT NULL,

`ProvisionType` VARCHAR(10) NOT NULL,

-- Define primary and foreign keys

PRIMARY KEY (`ProvisionID`, `ProvisionType`),

CONSTRAINT `fk\_Provision\_ProvisionType`

FOREIGN KEY (`ProvisionType`)

REFERENCES `ProvisionType` (`ProvisionType`),

CONSTRAINT `fk\_Provision\_SubMenu`

FOREIGN KEY (`SubMenu`)

REFERENCES `SubMenu` (`SubMenu`));

-- -----------------------------------------------------

-- Order Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Order` (

-- Define fields

`OrderID` INT NOT NULL AUTO\_INCREMENT,

`StaffMemberID` INT NOT NULL,

`CustomerID` INT NOT NULL,

`PaymentType` VARCHAR(10) NOT NULL,

`OrderTime` DATETIME NOT NULL,

`GroupSize` SMALLINT NOT NULL,

-- Define primary and foreign keys

PRIMARY KEY (`OrderID`, `PaymentType`, `CustomerID`, `StaffMemberID`),

CONSTRAINT `fk\_Order\_PaymentType`

FOREIGN KEY (`PaymentType`)

REFERENCES `PaymentType` (`PaymentType`),

CONSTRAINT `fk\_Order\_Customer`

FOREIGN KEY (`CustomerID`)

REFERENCES `Customer` (`CustomerID`),

CONSTRAINT `fk\_Order\_StaffMember`

FOREIGN KEY (`StaffMemberID`)

REFERENCES `StaffMember` (`StaffMemberID`));

-- -----------------------------------------------------

-- OrderEntry Table

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `OrderEntry` (

-- Define fields

`ProvisionID` INT NOT NULL,

`OrderID` INT NOT NULL,

`Quantity` SMALLINT NOT NULL,

-- Define primary and foreign keys

PRIMARY KEY (`ProvisionID`, `OrderID`),

CONSTRAINT `fk\_OrderEntry\_Provision`

FOREIGN KEY (`ProvisionID`)

REFERENCES `Provision` (`ProvisionID`),

CONSTRAINT `fk\_OrderEntry\_Order`

FOREIGN KEY (`OrderID`)

REFERENCES `Order` (`OrderID`));

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Insert Data

-- -----------------------------------------------------

-- Payment Type Data

-- -----------------------------------------------------

INSERT INTO `PaymentType` VALUES ("Cash");

INSERT INTO `PaymentType` VALUES ("Card");

-- -----------------------------------------------------

-- ProvisionType Data

-- -----------------------------------------------------

INSERT INTO `ProvisionType` VALUES ("Food");

INSERT INTO `ProvisionType` VALUES ("Drink");

-- -----------------------------------------------------

-- SubMenu Data

-- -----------------------------------------------------

INSERT INTO `SubMenu` VALUES ("Special Menu");

INSERT INTO `SubMenu` VALUES ("Standard Menu");

INSERT INTO `SubMenu` VALUES ("Veggie Menu");

-- -----------------------------------------------------

-- Role Data

-- -----------------------------------------------------

INSERT INTO `Role` VALUES ("Cook");

INSERT INTO `Role` VALUES ("Manager");

INSERT INTO `Role` VALUES ("Sales");

-- -----------------------------------------------------

-- Customer Data

-- -----------------------------------------------------

INSERT INTO `Customer` VALUES (333,"07741511271");

INSERT INTO `Customer` VALUES (876,"01942741598");

INSERT INTO `Customer` VALUES (95521,"01587555211");

INSERT INTO `Customer` VALUES (100592,"07748555155");

INSERT INTO `Customer` VALUES (105858,"01587584217");

INSERT INTO `Customer` VALUES (155822,"01942018596");

-- -----------------------------------------------------

-- StaffMember Data

-- -----------------------------------------------------

INSERT INTO `StaffMember` (`Forename`,`Surname`,`Role`) VALUES ("Euan","Robertson","Sales");

INSERT INTO `StaffMember` (`Forename`,`Surname`,`Role`) VALUES ("Ben","Davidson","Sales");

INSERT INTO `StaffMember` (`Forename`,`Surname`,`Role`) VALUES ("Jill","Blackburn","Manager");

INSERT INTO `StaffMember` (`Forename`,`Surname`,`Role`) VALUES ("Forbes","Johnson","Manager");

INSERT INTO `StaffMember` (`Forename`,`Surname`,`Role`) VALUES ("Stuart","Davies","Sales");

INSERT INTO `StaffMember` (`Forename`,`Surname`,`Role`) VALUES ("Jenny","Irvine","Cook");

INSERT INTO `StaffMember` (`Forename`,`Surname`,`Role`) VALUES ("Adrian","Summerfield","Cook");

INSERT INTO `StaffMember` (`Forename`,`Surname`,`Role`) VALUES ("Andrea","Burns","Sales");

-- -----------------------------------------------------

-- Provision Data

-- -----------------------------------------------------

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (52,"Chai",1.1,"Special Menu","Drink");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (412,"Mexican Burger",1.75,"Special Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (127,"Chicken Wings",1.75,"Special Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (39,"Bacon Pancakes",2.2,"Special Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (122,"Milkshake",1.0,"Special Menu","Drink");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (0,"Spicy Seaweed",0.9,"Special Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (250,"Curly Fries",1.3,"Veggie Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (111,"Carrot Cake",1.5,"Veggie Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (25,"Banana Split",1.55,"Veggie Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (66,"Veggie Burger",1.6,"Veggie Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (123,"Tea",0.9,"Standard Menu","Drink");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (200,"Coffee",1.2,"Standard Menu","Drink");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (300,"Bacon Burger",1.6,"Standard Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (158,"Sausage Roll",1.5,"Standard Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (150,"Cheeseburger",1.75,"Standard Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (100,"Chips",1.2,"Standard Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (0,"Sticky Cake",1.5,"Standard Menu","Food");

INSERT INTO `Provision` (`StockLevel`,`Name`,`Price`,`SubMenu`,`ProvisionType`) VALUES (0,"Cola",1.1,"Standard Menu","Drink");

-- -----------------------------------------------------

-- Order Data

-- -----------------------------------------------------

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (1,876,"Card","2020-05-31 14:00:00",2);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (3,333,"Cash","2020-06-01 10:30:00",2);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (5,876,"Cash","2020-06-01 09:30:00",3);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (4,105858,"Card","2020-06-02 15:00:00",2);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (3,100592,"Card","2020-06-02 10:40:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (5,876,"Card","2020-06-09 10:45:00",2);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (2,876,"Card","2020-06-12 13:15:00",2);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (8,155822,"Cash","2020-06-13 09:30:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (8,95521,"Cash","2020-06-22 15:00:00",3);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (4,155822,"Cash","2020-06-22 09:30:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (8,105858,"Card","2020-06-30 15:00:00",4);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (1,105858,"Card","2020-07-12 10:40:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (2,333,"Cash","2020-07-15 10:45:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (8,333,"Card","2020-07-15 13:15:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (2,876,"Card","2020-07-16 09:30:00",2);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (8,333,"Cash","2020-07-17 09:35:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (3,333,"Cash","2020-10-10 09:00:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (1,105858,"Card","2020-10-15 11:45:00",1);

INSERT INTO `Order` (`StaffMemberID`,`CustomerID`,`PaymentType`,`OrderTime`,`GroupSize`) VALUES (8,155822,"Cash","2020-10-23 15:30:00",1);

-- -----------------------------------------------------

-- Order Entry Data

-- -----------------------------------------------------

INSERT INTO `OrderEntry` VALUES (2,1,2); -- Order 1

INSERT INTO `OrderEntry` VALUES (7,1,2);

INSERT INTO `OrderEntry` VALUES (18,1,2);

INSERT INTO `OrderEntry` VALUES (5,2,2); -- Order 2

INSERT INTO `OrderEntry` VALUES (14,2,1);

INSERT INTO `OrderEntry` VALUES (16,3,3); -- Order 3

INSERT INTO `OrderEntry` VALUES (13,3,3);

INSERT INTO `OrderEntry` VALUES (8,4,1); -- Order 4

INSERT INTO `OrderEntry` VALUES (11,4,2);

INSERT INTO `OrderEntry` VALUES (17,4,1);

INSERT INTO `OrderEntry` VALUES (9,5,1); -- Order 5

INSERT INTO `OrderEntry` VALUES (16,6,2); -- Order 6

INSERT INTO `OrderEntry` VALUES (3,6,1);

INSERT INTO `OrderEntry` VALUES (15,6,1);

INSERT INTO `OrderEntry` VALUES (4,7,2); -- Order 7

INSERT INTO `OrderEntry` VALUES (6,7,1);

INSERT INTO `OrderEntry` VALUES (18,7,2);

INSERT INTO `OrderEntry` VALUES (15,8,1); -- Order 8

INSERT INTO `OrderEntry` VALUES (16,8,1);

INSERT INTO `OrderEntry` VALUES (7,9,3); -- Order 9

INSERT INTO `OrderEntry` VALUES (15,9,3);

INSERT INTO `OrderEntry` VALUES (18,9,3);

INSERT INTO `OrderEntry` VALUES (15,10,1);-- Order 10

INSERT INTO `OrderEntry` VALUES (16,10,1);

INSERT INTO `OrderEntry` VALUES (17,11,2);-- Order 11

INSERT INTO `OrderEntry` VALUES (12,11,2);

INSERT INTO `OrderEntry` VALUES (16,11,4);

INSERT INTO `OrderEntry` VALUES (17,12,1);-- Order 12

INSERT INTO `OrderEntry` VALUES (8,12,1);

INSERT INTO `OrderEntry` VALUES (16,13,1);-- Order 13

INSERT INTO `OrderEntry` VALUES (10,13,1);

INSERT INTO `OrderEntry` VALUES (3,14,1); -- Order 14

INSERT INTO `OrderEntry` VALUES (6,14,1);

INSERT INTO `OrderEntry` VALUES (16,14,1);

INSERT INTO `OrderEntry` VALUES (1,15,2); -- Order 15

INSERT INTO `OrderEntry` VALUES (16,15,1);

INSERT INTO `OrderEntry` VALUES (15,16,1);-- Order 16

INSERT INTO `OrderEntry` VALUES (7,16,1);

INSERT INTO `OrderEntry` VALUES (16,17,1);-- Order 17

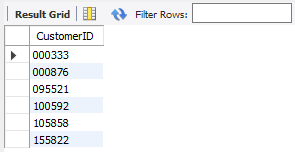
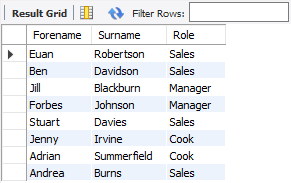
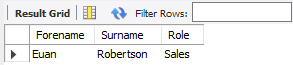
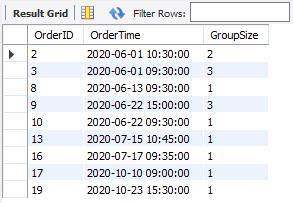
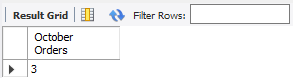
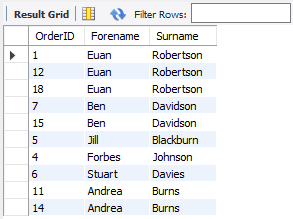
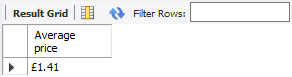
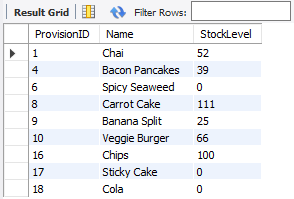
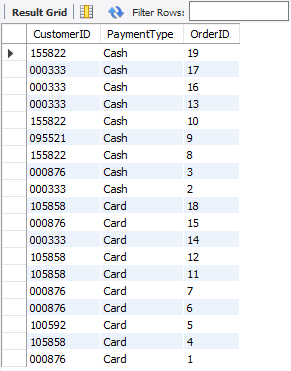
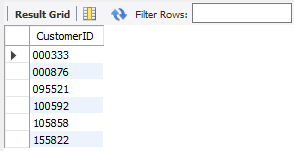
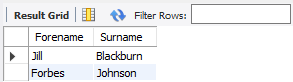
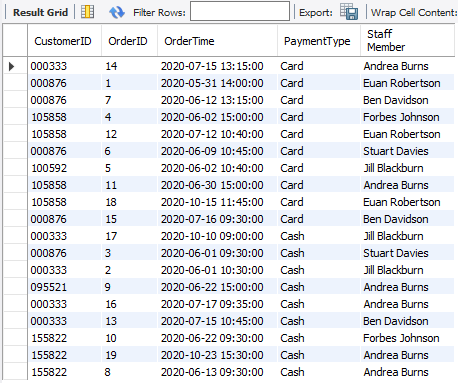
INSERT INTO `OrderEntry` VALUES (4,18,1); -- Order 18

INSERT INTO `OrderEntry` VALUES (12,19,1);-- Order 19

-- -----------------------------------------------------

**Assessment Evidence Required for Outcome 4:**

**Create and show appropriate SQL scripts and show results for the following queries**:

1. List **all** customers  
   SELECT LPAD(`CustomerID`,6,0) AS "CustomerID" FROM `Customer`  
   
2. List **staff** **forenames**, **surnames** and their **role**  
   SELECT `Forename`, `Surname`, `Role` FROM `StaffMember`  
   
3. List the **description** and **price** of Items that cost less than £**1.60**  
   SELECT `Name`, CONCAT("£",`Price`) AS "Price"   
   FROM `Provision` WHERE `Price` < 1.6  
   
4. List **staff** members whose surname is **Robertson**  
   SELECT `Forename`, `Surname`, `Role` FROM `StaffMember`  
   WHERE `Surname` = "Robertson"  
   
5. List **all orders** that are paid for by **cash**  
   SELECT `OrderID`, `OrderTime`, `GroupSize`  
   FROM `Order` WHERE `PaymentType` = "Cash"  
   
6. **Count** how many orders were taken in total in October  
   SELECT COUNT(`OrderID`) AS "October Orders" FROM `Order` WHERE   
   `OrderTime` > "2020-09-30 23:59:59" AND `OrderTime` < "2020-11-01 00:00:00"  
   
7. List the **Order ID**, and **staff first name** and **surname** where the order payment type was **card**  
   SELECT DISTINCT `OrderID`,`Forename`,`Surname` FROM `Order` INNER JOIN `StaffMember`  
   ON `Order`.`StaffMemberID` = `StaffMember`.`StaffMemberID`  
   WHERE `PaymentType` = "Card"  
   
8. List the **average price** of all items  
   SELECT CONCAT("£",FORMAT(AVG(`Price`),2)) AS "Average price" FROM `Provision`  
   
9. List all **stock items** where stock quantity is **less than** **120**  
   SELECT `ProvisionID`, `Name`, `StockLevel` FROM `Provision` WHERE `StockLevel` < 120  
   
10. List all **customers** and indicate what *payment type* they have used to place an order. **List these by cash, then by card**  
    SELECT DISTINCT LPAD(`CustomerID`,6,0) AS "CustomerID",  
    `PaymentType`, `OrderID` FROM `Order`  
    ORDER BY `PaymentType` DESC  
    
11. Using an appropriate **JOIN**, list the customers who have placed an order  
    SELECT DISTINCT LPAD(`Order`.`CustomerID`,6,0) AS “CustomerID”  
    FROM `Order` INNER JOIN `Customer`  
    ON `Order`.`CustomerID` = `Customer`.`CustomerID`  
    
12. Using an appropriate **JOIN**, list the staff managers who have taken cash orders   
    SELECT DISTINCT `Forename`, `Surname`  
    FROM `Order` INNER JOIN `StaffMember`  
    ON `Order`.`StaffMemberID` = `StaffMember`.`StaffMemberID`  
    WHERE `PaymentType` = "Cash" AND `StaffMember`.`Role` = "Manager"  
    
13. List the **customers, orders, payment type** and the **staff** who have taken their orders. **Order by payment type**. Demonstrate the use of an **ALIAS**  
    SELECT DISTINCT LPAD(`CustomerID`,6,0) AS "CustomerID",  
    `OrderID`, `OrderTime`, `PaymentType`,   
    CONCAT(`Forename`," ",`Surname`) AS "Staff Member"   
    FROM `Order` INNER JOIN `StaffMember`   
    ON `StaffMember`.`StaffMemberID` = `Order`.`StaffMemberID`   
    ORDER BY `PaymentType`  
    
14. Create a **View** called [Cash Orders] that lists all the **orders** paid by **cash**.  
    CREATE OR REPLACE VIEW `Cash Orders` AS  
    SELECT `OrderID`, `OrderTime`, `GroupSize`  
    FROM `Order` WHERE `PaymentType` = "Cash";  
    SELECT \* FROM `Cash Orders`  
    