**qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

|  |
| --- |
| Database Documentation  Assignment 1: Linda’s Lighting  D14123580  Donncha Cassidy-Hand |

## Table of Contents

## 

[**Entity Choice** 1-2](#_Toc405726769)

[**Datatype Choice** 3](#_Toc405726769)

[**Relationship Choice** 4-6](#_Toc405726769)

[**Tables and Inserts** 7-12](#_Toc405726769)

[**Queries** 13-17](#_Toc405726769)

[**Alterations and Updates** 18-19](#_Toc405726769)

## Entity Choice

## 

The following is a list of the entities used for the database including an explanation of what the purpose of each one is.

* **Customer** 
  + Contains necessary customer information, Name, Phone Number and Address.
* **LightingSystem**
  + Information on the date on which the contract was undertaken, the return date for the customer and the PaymentNo, CustomerID are stored here.
* **LightingSpec**
  + Has information on the specification of the lighting system that is to be installed.
* **Customer\_Payment**
  + Contains entities of the date of when the customer paid and the amount that they pied.
* **Employee**
  + The department of which the employee belongs to, the contractID which the employee has been assigned to and the name of the employee.
* **StockAllocated**
  + The stock which has been allocated to the contract, includes the amount of stock allocated to it, the stock code and the price of the stock.
* **Department**
  + Department table contains descriptions of the four departments with the appropriate department associated with each employee.
* **StockCategory**
  + This table gives each stock item an assigned category.
* **Stock**
  + Contains attributes for the Stock Description for each item type with a stockcode and the quantity of the stock.
* **StockOrderItem**
  + The StockOrderItem table gives the amount of stockordered.
* **StockOrder**
  + Provides the orderdate and delivery date of the items been delivered.
* **Supplier**
  + Has information on the supplier of the items, including the name of the company and its address.
* **Supplier\_Payment**
  + Keeps the date the customer paid for their Lighting System and how much they paid.

## Datatype Choice

## 

The datatypes that I used were VARCHAR2, DATE and NUMBER.

This allowed for providing IDs using characters and numbers, dates in the context of eg. 06 Jan 2001 and Number allowed for the use of pricing such as 50.20.

VARCHAR2 is used for all the Primary keys as it allows for numbers and characters to be associated together ie. CO1001, E1002 etc.

Date is used for all the dates as it allows for the standard context of dating.

Number is used for pricing as it allows for an integer with the addition of decimal points. It is particularly useful for pricing with the use of allow a decimal point to two numbers ie. NUMBER(5,2) will allow for numbers such as €52500.99

## Relationship Choice

## 

**Customer to LightingSystem:**

Is an non-identifying one to many relationship. This means each Customer can have many LightingSystem.

**Customer\_Payment to LightingSystem:**

Is an non-identifying one to many relationship, which allows one customer payment to be able to pay for multiple Lighting Systems.

**StockAllocated to LightingSystem:**

Is a Many to Many relationship which allows for the allocation of many stock items to many lighting systems.

**LightingSystem to LightingSpec:**

Is a One to One relationship, allowing each LightingSystem to be given a LightingSpec, providing details on what the LightingSystem is going to be.

**LightingSystem to Employee:**

Is a non-identifying One to Many relationship which allows for many Employees to be assigned to a single LightingSystem.

**Department to Employee:**

Is a non-Identifying One to Many relationship, which allows for the allocation of each Employee to a single department.

**Department to StockCategory:**

Is a non-Identifying One to Many relationship which allows for departments to be assigned to Stock Categories.

**StockCategory to Stock:**

Is a non-Identifying One to Many relationship which allows each stock to be associated with a Stock Category.

**Stock to StockOrderItem:**

Is a One to Many relationship which allows for each stock item to be submitted for ordered for delivery.

**StockOrder to StockOrderItem:**

Is a One to Many relationship which allows for each stock that is been

**Supplier to StockOrder:**

Is a non-Identifying One to Many relationship that assigns a single supplier to many stock orders.

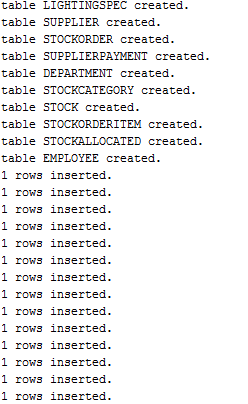
**SupplierPayment to StockOrder:**

Is a One to One realationship which allows for a single payment to each stock order.

## Tables and Inserts

## 

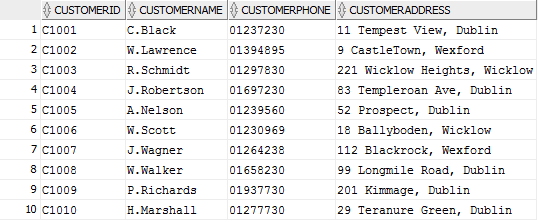
The following screen capture is taken from Oracle SQL Developer. The output in the screen captures confirms the creation of the Tables and the information been inserted into each table.



### Table Data on SQL Developer

The following screen captures are of each table in SQL Developer after the information from the provided SQL code is inserted into the tables after the creation of the tables. The output is functional Database with foreign key constraints established and primary keys.

**Customer**



**Customer\_Payment**

## C:\Users\donncha\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Customer_Payment.png

## 

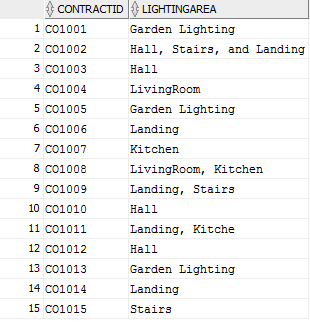
**Department**

## C:\Users\donncha\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Department.png

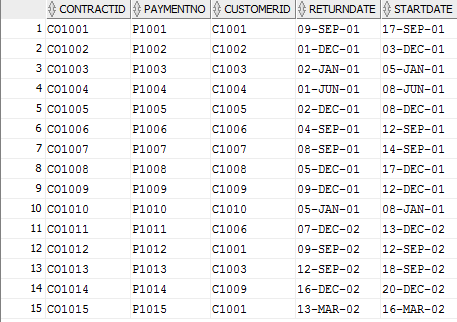
**Employee**

## C:\Users\donncha\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Employee.png

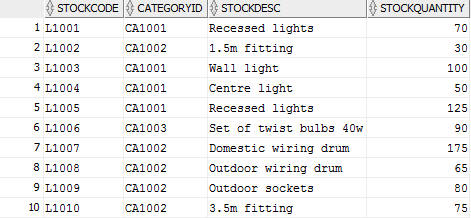
**LightingSpec**



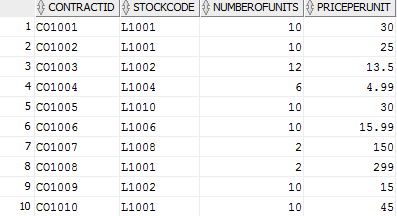
**LightingSystem**



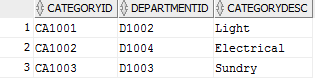
**Stock**



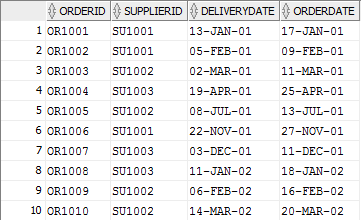
**StockAllocated**



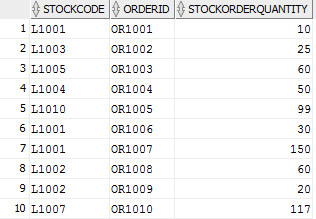
**StockCategory**



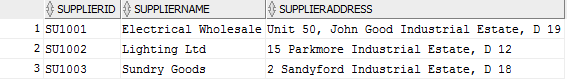
**StockOrder**



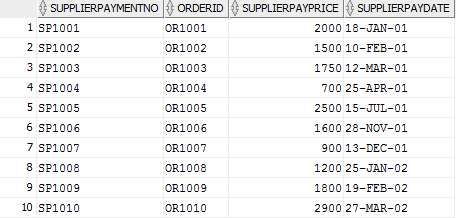
**StockOrderItem**



**Supplier**



**Supplier\_Payment**



## Queries

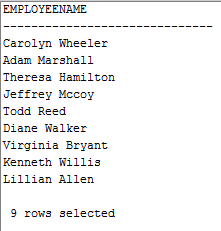
## 

The following are explanations of the queries included in the SQL code. These are to demonstrate my understanding of each of the following terms including Single row function, aggregate functions and various Joins.

The following code outputs an example of the use of a single row function.

**select EmployeeName from employee;**

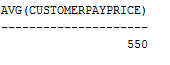
**SQL Developer Output:**



The following code outputs an example of the use of an AGGREGATE function by using AVG to find the average of a set of numbers.

**select AVG(CustomerPayPrice) from Customer\_Payment;**

**SQL Developer Output:**



The following code outputs an example of the use of a two table inner join, it joins the tables StockOrderItem and StockOrder and outputs the Orderdate which is common to both of them.

**select DeliveryDate**

**from StockOrderItem o**

**inner join StockOrder a**

**on o.OrderID=a.OrderID**

**ORDER BY a.OrderDate;**

**SQL Developer Output:**



The following code outputs an example of the use of inner joining three tables, it joins the tables LightingSystme, Employee and Department. It then outputs the ContractID associated with each Customer.

**select l.ContractID, e.EmployeeID, d.DepartmentDesc**

**from LightingSystem l**

**inner join Employee e**

**on l.ContractID=e.ContractID**

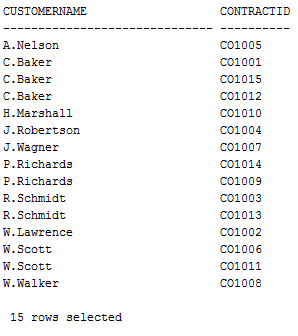
**inner join Department d**

**on e.DepartmentID=d.DepartmentID**

**ORDER BY l.ContractID**

**;**

**SQL Developer Output:**



The following code outputs an example of the use of a Left Outer Join which joins the attributes specified in the select and outputs the information which is only common to the first table specified.

**SELECT Customer.customerName, LightingSystem.ContractID**

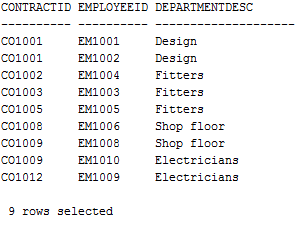
**FROM Customer**

**LEFT JOIN LightingSystem**

**ON Customer.CustomerID=LightingSystem.CustomerID**

**ORDER BY Customer.customerName;**

**SQL Developer Output:**



The following code outputs an example of the use of a Right Outer Join which joins the attributes specified in the select and outputs the information which is only common to the second table specified.

**SELECT Department.DepartmentID, Employee.EmployeeName**

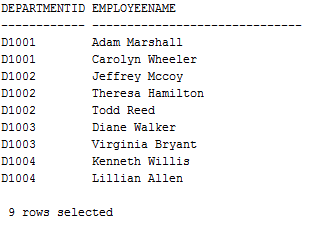
**FROM Department**

**RIGHT JOIN Employee**

**ON Department.DepartmentID=Employee.DepartmentID**

**ORDER BY Employee.DepartmentID;**

**SQL Developer Output:**



## Alternations and Updates

## 

The following is a demonstration of my understanding of using Alterations and Updates in SQL. The examples below show adjustments and changes been made in the SQL Developer Database and the output to prove confirmation of each Alteration or Update been successful.

The following code updates the Customer table to change the customerName from C.Black to C.Baker.

**UPDATE Customer**

**SET customerName='C.Black'**

**WHERE CustomerName='C.Baker';**

**SQL Developer Output:**



The following code adds a new column to the table Customer and setting a default constraint as Dublin if there is no value entered into the row of the newly created column.

**-- Add column to the table and a value constraint --**

**ALTER TABLE Customer**

**ADD customerCity VARCHAR2(30) default 'Dublin';**

**SQL Developer Output:**



The following code modifies the column customerCity in the table Customer by changing the maximum length of the datatype VARCHAR to 20 from the its original length of 30.

**-- Modify a column on a table --**

**ALTER TABLE Customer**

**MODIFY customerCity VARCHAR2(20) default 'Dublin';**

**SQL Developer Output:**

**C:\Users\donncha\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png**

The following code drops the column customerCity in the Customer table.

**-- Drop column in a table --**

**ALTER TABLE Customer**

**DROP COLUMN customerCity;**

**SQL Developer Output:**

**C:\Users\donncha\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png**

The following code drops the foreign key constraint PayNo\_CusP\_LightS\_FK in the Lighting System table.

**-- Drop constraint on a table --**

**ALTER TABLE LightingSystem**

**DROP CONSTRAINT PayNo\_CusP\_LightS\_FK;**

**SQL Developer Output:**

