# Introduction

## Case Study Background

Everblue Ocean Express (ECE) is a container and shipping company from Britain who transport goods around the world for their customers. They require a new database system to create schedules for the vessels to transport containers which have been newly acquired from an existing shipping company. In this report, I will showcase the process of planning, designing and creating said database, where the requirements set by ECE will be considered.

## Aims of The Report

The aim of this report is to document the planning, design and creation process of the database system.

The report will be used to show the thought process when creating the database to justify how the requirements set by ECE were met.

The report will also be used to show to stakeholders the creation process of the database so that it can be maintained after creation by ECE.

## Objectives of The Report

1. Within the report, show the designed ER diagram created for the database system and go over the constraints and assumptions made.
2. Provide evidence of database creation via SQL table definitions.
3. Create a discussion within the report about the dummy data, which was entered into the database, with screenshots.
4. To use SQL queries to show that the database meets the requirements asked by the stakeholders.

# Chapter 1: The Entity Relationship Diagram

## The Final ERD

Before the creation of the database, it is imperative that the design phase must take place where we will create an entity relationship diagram to represent the complex relationships between the tables of the database, such as identifying the constraints and the multiplicities of the tables. We will make sure to list the primary key and foreign key to enforce data integrity. By creating this ERD, this allows for us to plan efficiently and effectively so that if changes need to be made, we can make them on the ERD prior to creating the database. Shown below is the final ERD, which was created via StarUML, for the ECE system. Diagram, schematic

Description automatically generated

## Assumptions

Assumptions were made when creating the ERD to ensure that the final database design would match the stakeholder requirements. Below I will be going over the relationships and cardinalities within the diagram and state the thought process behind each assumption.

Diagram, schematic

Description automatically generated



Diagram

Description automatically generated





Diagram

Description automatically generated



Diagram, schematic

Description automatically generated

Diagram

Description automatically generated



Diagram

Description automatically generated



Diagram

Description automatically generated





Diagram

Description automatically generated



Diagram

Description automatically generated

A picture containing text, receipt

Description automatically generated



Diagram

Description automatically generated



Diagram

Description automatically generated

Diagram

Description automatically generated



# Chapter 2: Implementation – SQL Table Definitions

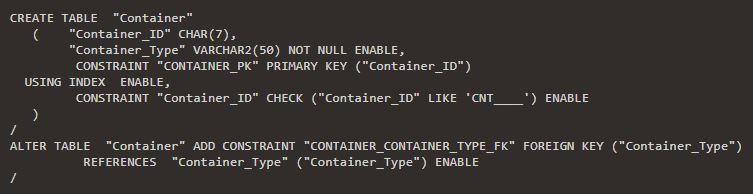
In this chapter, the different SQL statements used in the creation of the different tables within Apex Oracle will be shown. The database will follow the structure provided in the ERD created in StarUML. Evidence of the tables being created in Apex Oracle will also be provided alongside the SQL CREATE statements.

## Data Integrity

Throughout the table creation process, keeping data integrity was very important, therefore, certain constraints were put in place to ensure this…

* For all IDs, a specific constraint was set where the ID must start with certain characters, for example, with Booking\_ID, it must start with ‘BKG’, followed by four characters. This was done so it is easy to identify what a Booking\_ID is on other tables. By also making it four characters other than just four numbers, it gives the database longevity in case there is the low chance of all number combinations being used, therefore letters can be added.
* All IDs have a global length of 7 for consistency.
* All tables are not nullable, meaning the database will not have any blank spots, ensuring data integrity is kept and is consistent.

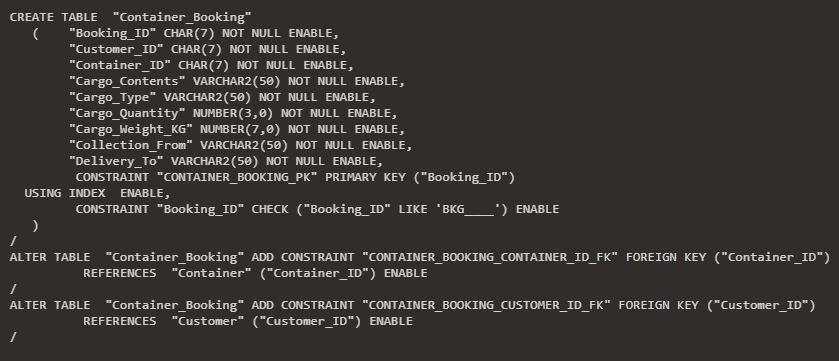
## Container



A screenshot of a computer screen

Description automatically generated

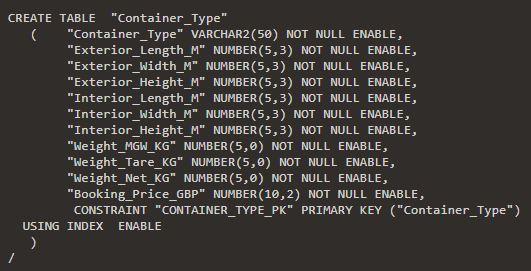
## Container\_Booking



A screenshot of a computer

Description automatically generated with medium confidence

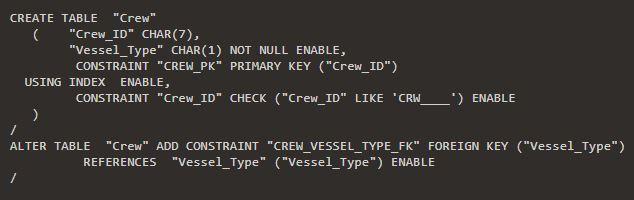
## Container\_Type



Graphical user interface

Description automatically generated

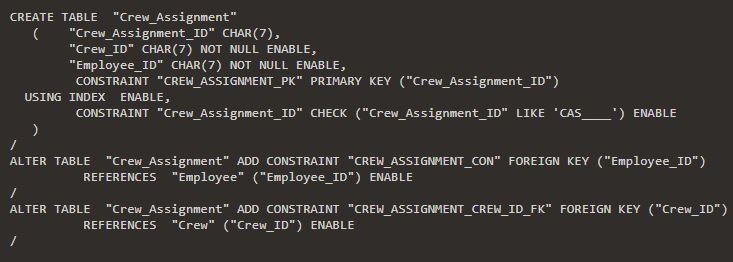
## Crew



A screenshot of a computer

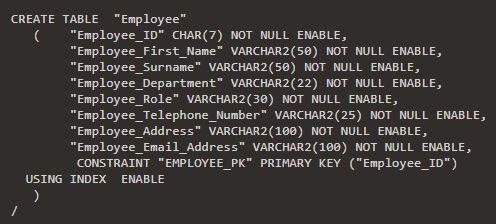
Description automatically generated with medium confidence

## Crew\_AssignmentA screenshot of a computer Description automatically generated with medium confidence



## CustomerA screenshot of a computer Description automatically generated with medium confidence

## Employee



Graphical user interface, application

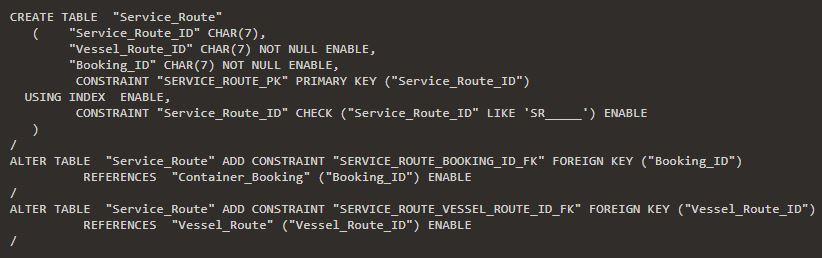
Description automatically generated

## Port

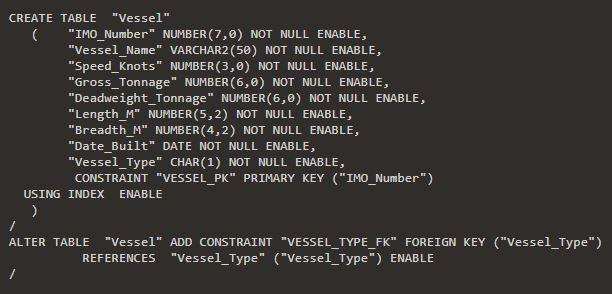
Graphical user interface, application

Description automatically generated

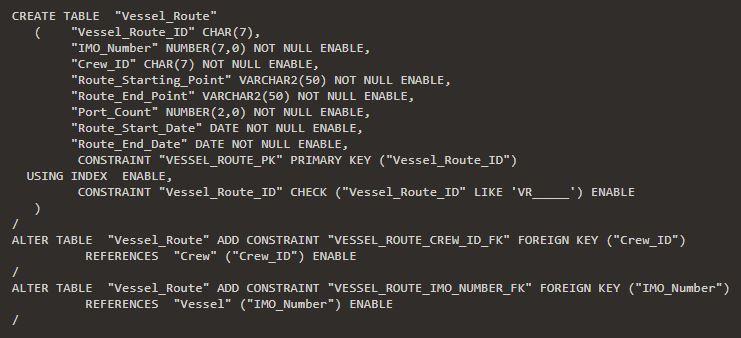
## Service\_RouteA screenshot of a computer Description automatically generated with medium confidence



## VesselGraphical user interface, application Description automatically generated



## Vessel\_Route

A screenshot of a computer

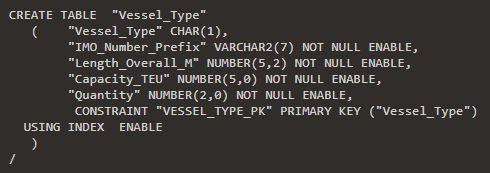
Description automatically generated with medium confidence

## Vessel\_Schedule

A screenshot of a computer

Description automatically generated with medium confidence

## Vessel\_Type



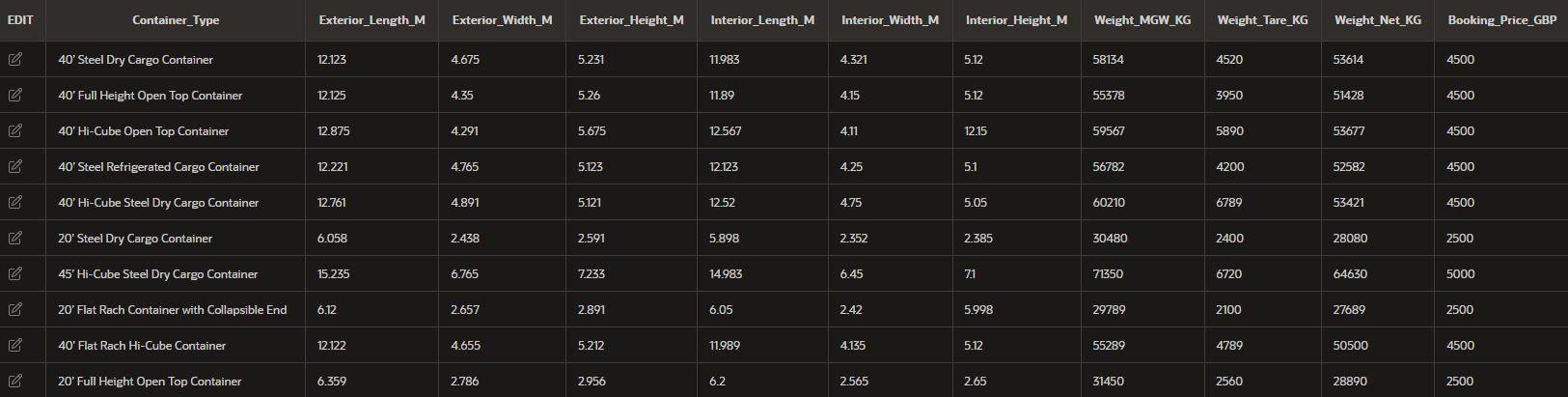
A screenshot of a computer

Description automatically generated with medium confidence

# Chapter 3: Discussion of The Data

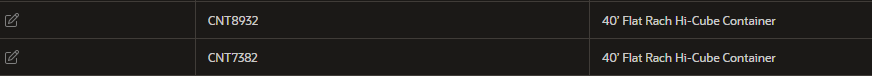
In this chapter of the report, we will be discussing the data, which was inputted into the database, and showing that the multiplicity of the relationships that were shown in the panning stage when creating the ERD are present.

## Container\_Type

Here we have the different container types which can be booked out by customers to transport their cargo. As shown in the ERD, we have included all the different measurement features and weight features, as well as the price to book said container. This will allow the customer to see if their cargo can fit inside a specific container, as well as be able to see the price to book said container. 

## ContainerText Description automatically generated

This is the Container table data, where we can see that there are multiple containers that are available for a customer to book out.

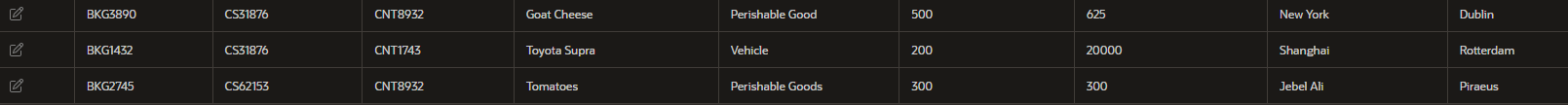
As stated in the ERD, the relationship between Container and Container\_Type is a one-to-many relationship where many containers can belong to a singular Container\_Type

From the focused screenshot above of the data in the Container table, we can see that there are two unique containers that have the same Container\_Type ‘40’ Flat Rach Hi-Cube Container’, showing that the multiplicity holds true in this table relationship.

## Container Booking

Above, we can see the data that has been inputted into the Container\_Booking table, where the customer can make a booking of a specific container and list the details of the cargo which they wish to be transported as well as the port location from which the cargo will go from and be delivered to. 

When creating the ERD diagram, it was mentioned that the relationship between Container\_Booking and Container would be that of a one-to-many where a Container could be booked multiple times.



In this focused screenshot, we can see that the Container with ID ‘CNT8932’ has been booked twice, showing that the multiplicity remains true in the final database design.

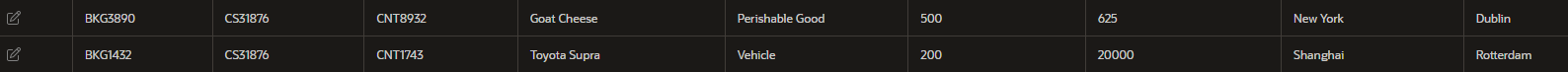
Special conditions had to be met within the “Container\_Booking” table for the database to show it works as intended, for example…

* A customer should be able to make multiple bookings for the same Vessel\_Route, which is shown in conjunction later with the “Service\_Route” table, where the customer with ID “CS53492” booked two containers for the same Vessel\_Route.
* A customer should also be able to make bookings for another Vessel\_Route too, where we can see that the customer with ID “CS31876” made two different bookings for two different routes (can later be seen in the “Service\_Route” table).

## Customer

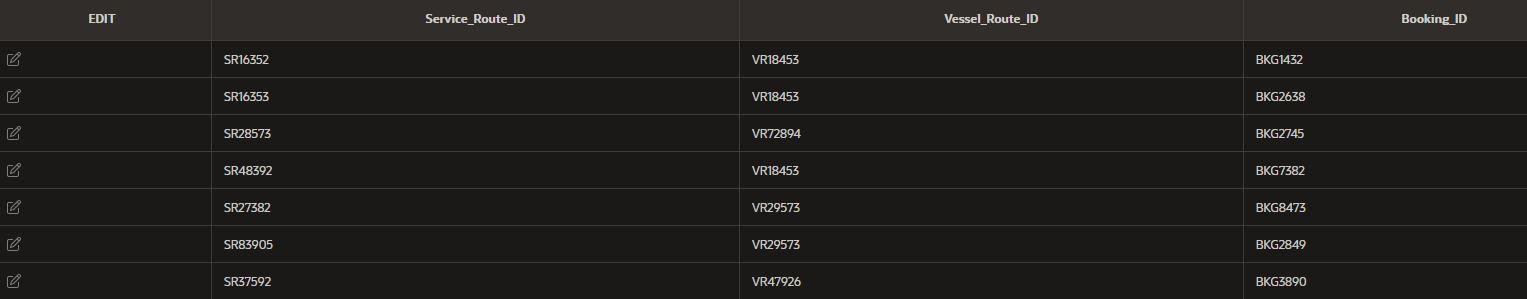
This is the data which has been inputted into the Customer table, where we can see that the Customer has been assigned a unique ID, as well as all their information being stored, in case they need to be contacted regarding their booking. 

The relationship between Customer and Container\_Booking was listed as a one-to-many, where a customer would have the capability to book out multiple containers, whereas a singular booking must belong to a singular customer.



The focused screenshot of the Container\_Booking table data shows this to be true, as we can see that the Customer with the ID of CS31876 has created two container bookings, showing that the multiplicity planned in the ERD is true.

## Service Route

The following data was inputted into the Service\_Route table so that we could assign a booking that was made with the journey in which a vessel is going on. This is done by using both the foreign keys of Vessel\_Route\_ID and Booking\_ID.

As stated in the ERD, the relationship between Service\_Route and Container\_Booking is a one to one where a booking must have a Service\_Route. From the data above we can see that all Service\_Route\_ID entries are unique and are assigned with a unique Booking\_ID every time.

## Vessel\_Route

This data shown above is the dummy data inputted into the Vessel\_Route table, where the IMO\_Number identifying the Vessel on the journey is shown, as well as the Crew assigned to said vessel, with the start and ending ports shown with the dates at which the journey will start and conclude. 

Within the ERD, we state that the relationship between the Vessel table and Vessel\_Route is a one-to-many, since a Vessel can be assigned to many routes, which can be seen in this screenshot since we can see that the Vessel with the IMO\_Number ‘707453’ has been assigned on two vessel routes.

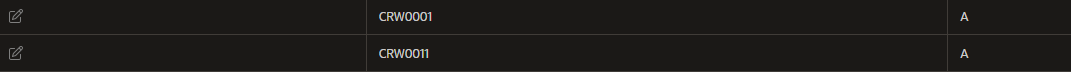
Also, within the ERD, we state that the relationship between Crew and Vessel\_Route is that of a one-to-many since a Crew can be assigned to multiple routes. From the screenshot, we can also see that this is true since the Crew with the Crew\_ID ‘CRW0001’ has been assigned to many vessel routes.

As well as this, there were special conditions that had to be met with the database, within the “Vessel\_Route” table to show that the database can meet intricate requirements, for example…

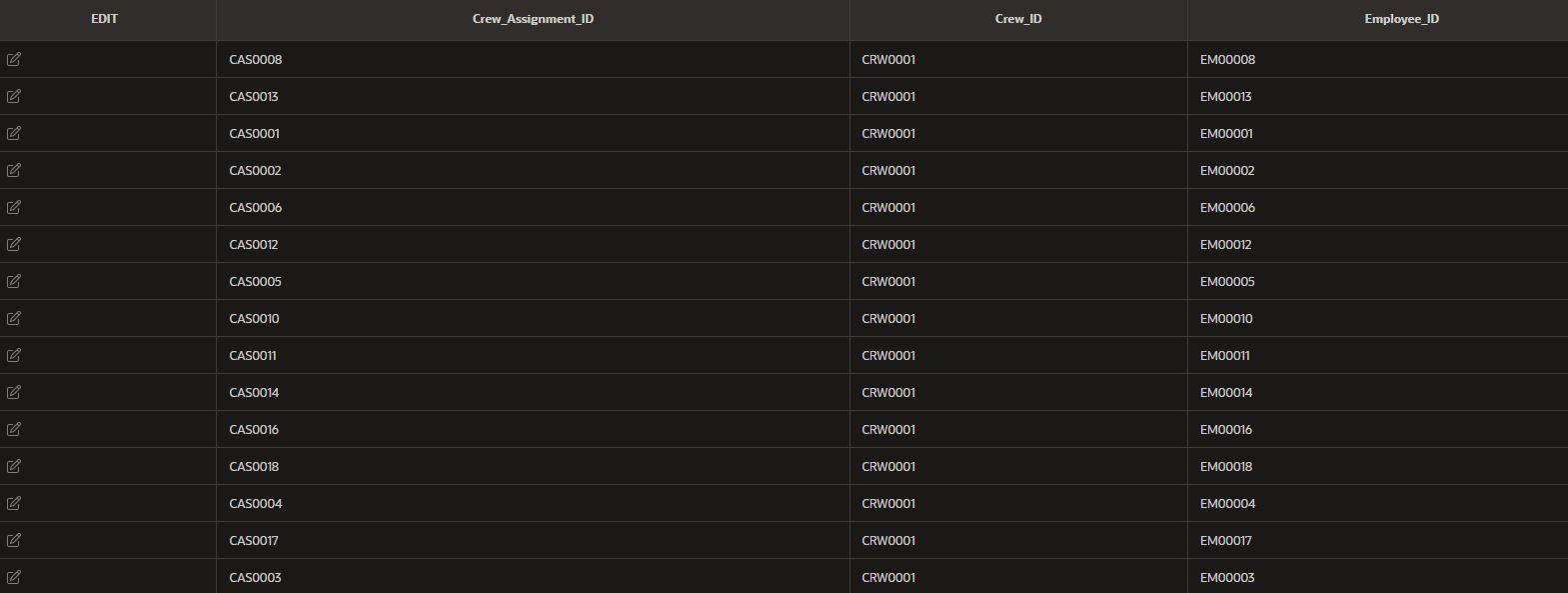
* Show at least two different vessels that share the same route, which was done with the vessels ‘707453’ and ‘7332329’, who both take the route from Shanghai to Antwerp, showing that multiple vessels can be assigned to the same route.
* A singular vessel that can be assigned on multiple routes on different dates, which is shown by vessel ‘707453’ which can be seen taking the route from Shanghai to Antwerp, finishing that journey in March, then embarking on the journey from Busan to Genoa, starting in April.
* Vessels can have different routes, but stop at the same port within the routes, for example, later on when we see the “Vessel\_Schedule” table and link it with the “Vessel\_Route” table, we can see that the vessels on the Vessel\_Route of ‘VR18453’ and ‘VR71843’ both stop at Rotterdam despite being on different routes, as well as the vessels on Vessel\_Route ‘VR71834 and ‘VR29573’ both stop at Algeciras.

## CrewA screenshot of a computer Description automatically generated with medium confidence

This is the data inputted for the Crew table which shows the different crews, represented through their ID, as well as the vessel type which said Crew can operate.

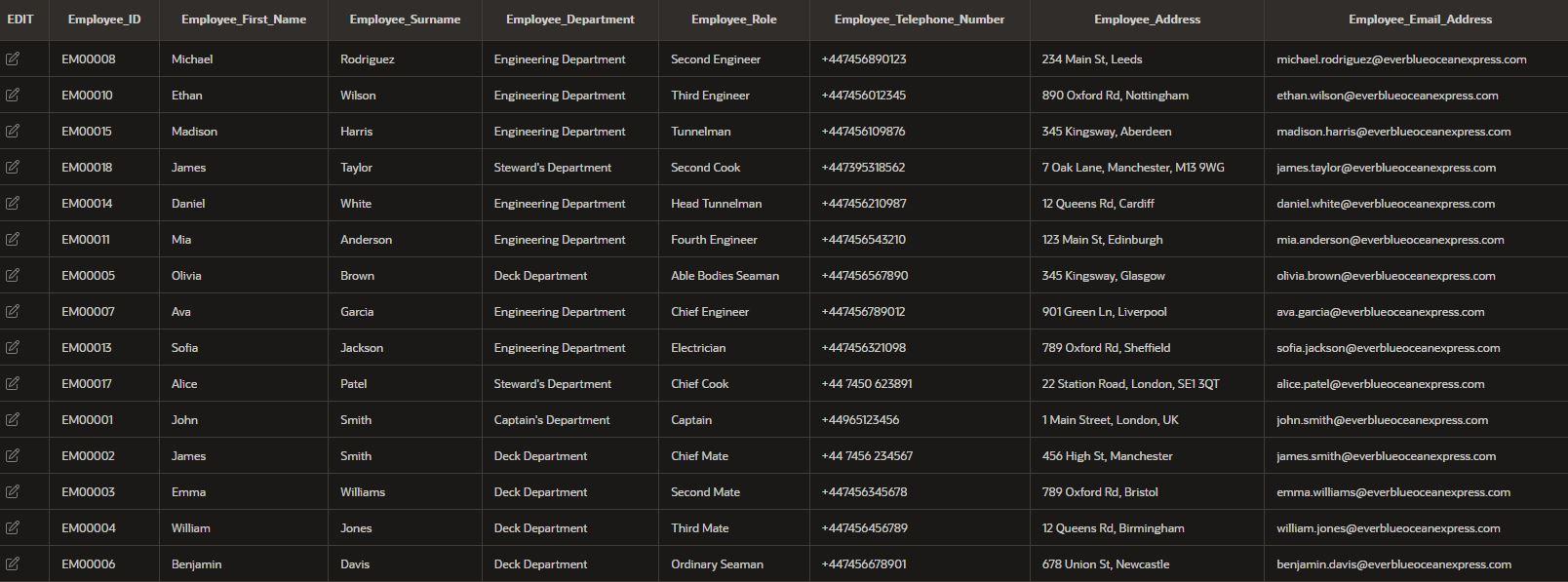
In the ERD, it was stated that the relationship between Crew and Vessel\_Type is a one-to-many relationship, where a Vessel\_Type may have many different crews. This is shown in the focused screenshot below where we can see that both ‘CRW0001’ and ‘CRW0011’ can both be used on vessels of type ‘A’.

## Crew\_Assignment

This is the data inputted into the Crew\_Assignment table, which is used to assign a specific Employee (identified through their ID) to a Crew. 

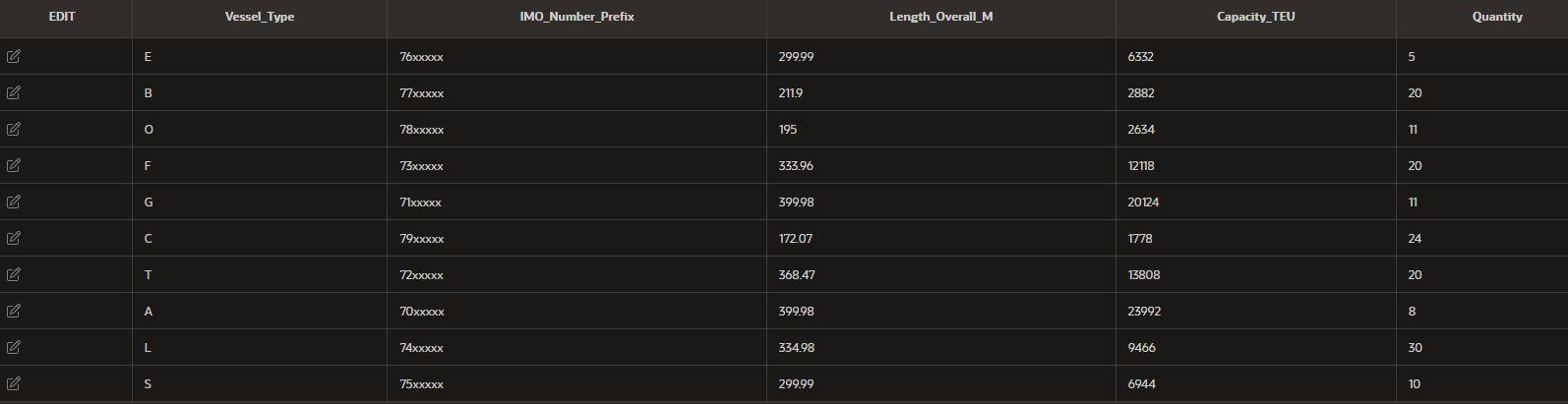
Within the ERD, we stated that the relationship between Crew and Crew\_Assignment is a one-to-many, where a Crew can have multiple employees assigned to it. This can be seen in the screenshot above, as we can see that many different employees have been assigned to the Crew with the Crew\_ID of ‘CRW0001’.

## Employee

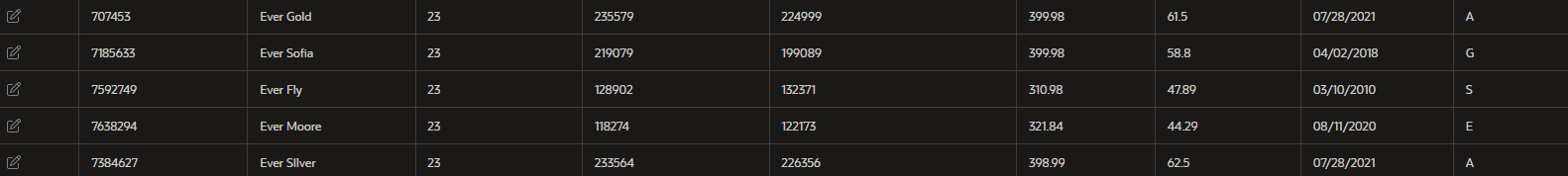
Listed above, is some dummy data which belongs to the Employee table, where an Employee is given a unique identifier, as well as having all their details stored, such as their name, address, email address and telephone number. Most importantly, their job role is listed so that assigning an Employee to a Crew is made easier.

The relationship between Crew\_Assignment and Employee, as stated in the ERD is that of a one-to-many where an Employee may be assigned to multiple different crews. From the focused screenshot below of the Crew\_Assignment table, we can see that the Employee with the ID ‘EM00001’ has been assigned on to two different crews, Crew ‘CRW0001’ and Crew ‘CRW0002’. 

## Vessel\_Type

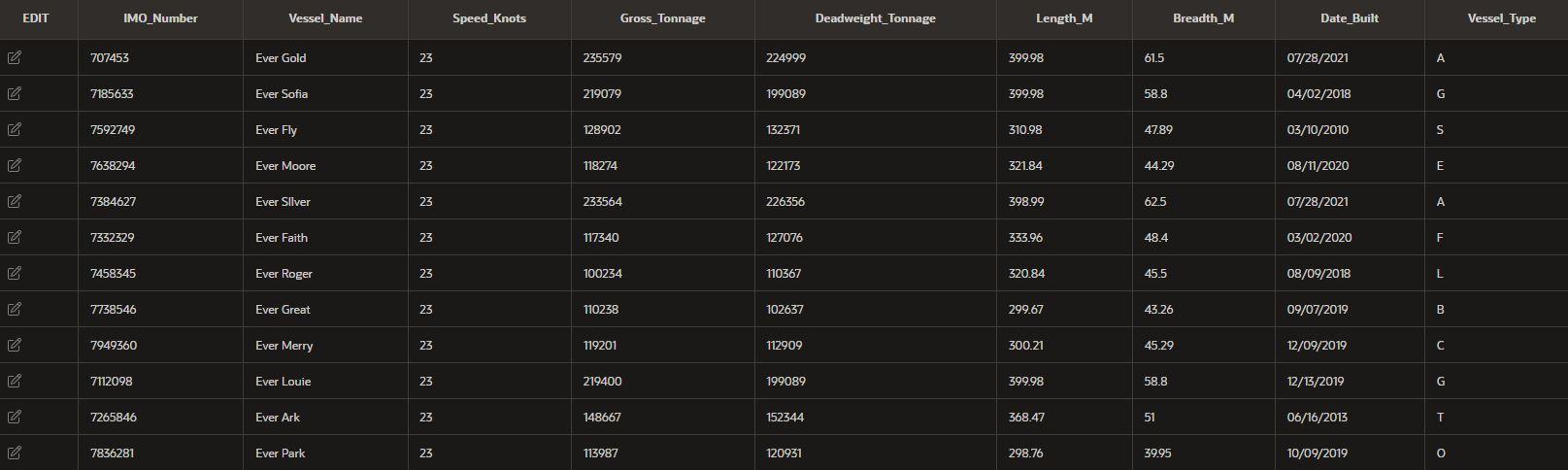
This is the dummy data which has been inputted into the Vessel\_Type table, which shows the measurements, capacity and number of vessels of a specific type, which is identified by a singular letter.

The relationship between Vessel\_Type and Vessel, as stated by the ERD is a one-to-many where a Vessel\_Type can belong to multiple vessels.



From this focused screenshot of the Vessel table, we can see that both the ‘Ever Gold’ and ‘Ever Silver’ both belong to Vessel\_Type A, showing that the multiplicity remained true during the implementation phase.

## Vessel

Above is the dummy data entered for the Vessel table, showing the different vessels, the name assigned to them, and their different stats, as well as the type they belong to.

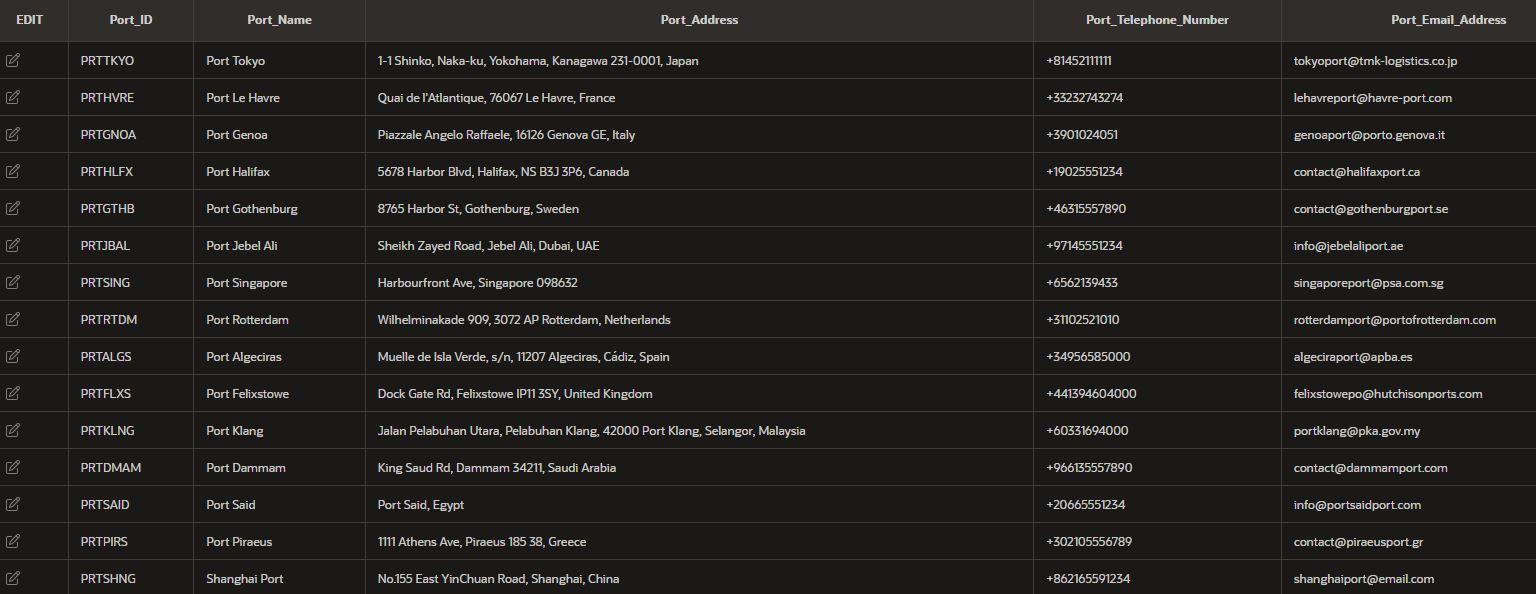
In the ERD, it stated that the relationship between Vessel and Vessel\_Schedule is a one to many, since a Vessel can have many schedules.

This can be shown in this focused screenshot of the Vessel\_Schedule table, where we can see that the Vessel with the IMO\_Number ‘707453’ has been assigned to multiple schedules to show when it is arriving and departing from the different ports.

## Vessel\_Schedule

This is the dummy data entered for the Vessel\_Schedule table, which shows the date of which a specific Vessel will arrive at a specific Port. This in conjunction with the Vessel\_Route table will notify the customer of the dates in which their cargo would be at each Port whilst on the route.

## Port

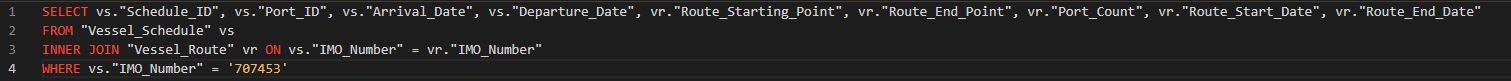
The data above is within the Port table, showing the details of the offices at each Port, as mentioned within the stakeholder criteria.

The relationship between Port and Vessel\_Schedule, as stated in the ERD is a one-to-one where a Vessel\_Schedule must have a singular Port assigned to it, which can be seen by the screenshot above.

# Chapter 4: Queries

In this chapter, we will be going over six important queries that were created using the database system. These six queries should prove to the stakeholders that the system meets the requirements. More queries are available for view on the Apex Oracle ‘Saved SQL’ tab; however, the queries below should showcase the functionality of the system.

## Query 1 – Finding Vessel Schedule Based off IMO\_Number

Below is the SQL Query which can be used by a customer or an employee to find the schedule of a specific vessel based off its IMO\_Number. In this query, we are selecting the columns “Schedule\_ID”, “Port\_ID”, “Arrival\_Date” and “Departure\_Date” from the “Vessel\_Schedule” table, and using INNER JOIN to join the “Route\_Starting\_Point”, “Route\_End\_Point”, “Port\_Count”, “Route\_Start\_Date” and “Route\_End\_Date” from the “Vessel\_Route column, based off of the IMO\_Number of ‘707453’.

The purpose of this query is to provide the customer and an employee all the available scheduled information of the vessel, where they can check the dates of when it will arrive at a specific port, as well as the routes the vessel has been designated on. A customer can use this query to track where their cargo is currently.

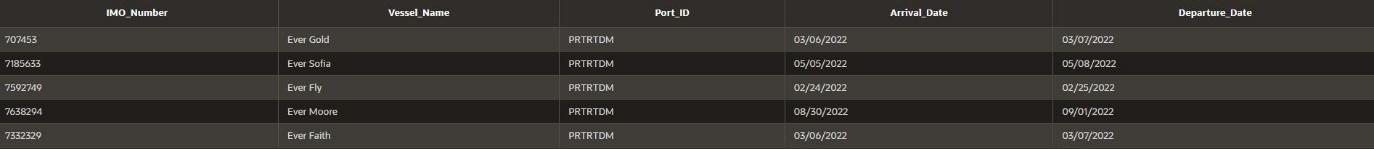
We can see the results of the query above, where we can now see the full schedule for the vessel with the IMO\_Number ‘707453’.

## Query 2 – Finding When a Vessel Arrives at a Port

This query is specifically designated for the customer, so that they can check when any vessel is arriving at the port, they plan on using to deliver their cargo. Once again, we will be joining two tables together, the “Vessel” and “Vessel\_Schedule” tables, where the columns “IMO\_Number”, “Vessel\_Name”, “Port\_ID”, “Arrival\_Date” and “Departure\_Date” will be displayed based off the “Port\_ID” of “PRTRTDM”.

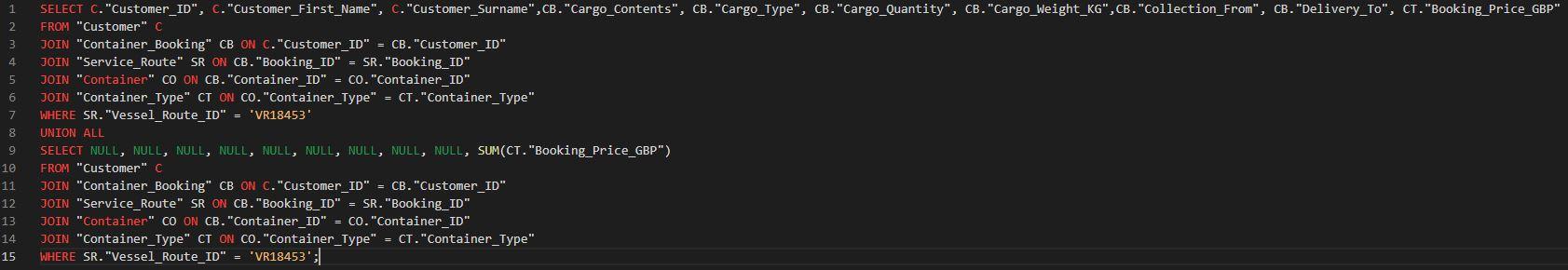
Text

Description automatically generated

The purpose of this query is to list the names of all vessels, as well as the dates they arrive and depart at a specific port in which the customer would want to use, for example, we are using the port located in Rotterdam. So, if a customer would like to use this port, this query will find them all the vessels, as well as the date at which they would arrive in Rotterdam so that they can see if booking a container for that date is good for them. Since this query is for the customers specifically, no ”Vessel\_Route” information is required since a customer would only care about the port the cargo will be picked up from and dropped off at, instead of needing information off all ports across a journey which may not concern them. 

From these results, we can see that there are five different vessels that will pass by port Rotterdam, therefore, the customer can see which date may work out for them and pick the vessel accordingly.

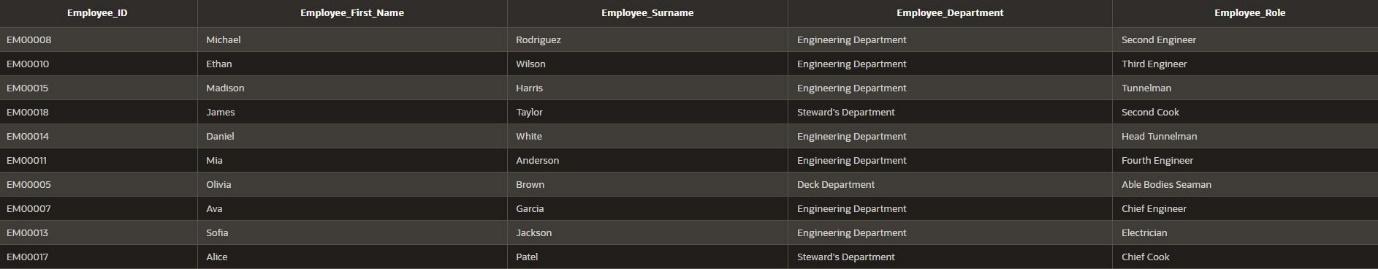
## Query 3 – Show all Customers & Cargo on a Specific Vessel Route and Profit Made

This query will be used by employees to track all the cargo on a specific journey, where the customer’s name will be listed alongside said cargo, as well as the price of all the containers on the journey, showing the amount of money a specific Vessel\_Route has made. This is done by combining columns from the “Customer”, “Customer\_Booking” and “Container\_Type” under the specific “Vessel\_Route\_ID” of ‘VR18453’. The “Customer\_ID”, “Customer\_First\_Name”, “Customer\_Surname”, “Cargo\_Contents”, “Cargo\_Type”, “Cargo\_Quantity”, “Cargo\_Weight\_KG”, “Collection\_From”, “Delivery\_To” and “Booking\_Price\_GBP”. 

To include the total booking price without repeating data, we created another SELECT statement which will retrieve NULL values for all columns except for the last one, where it will total the sum of all the Booking\_Price\_GBP that is listed, and then by using a UNION ALL operator, we can now combine the two SELECT statements together.

From these results, we can see that there are currently three customers that have booked cargo for the journey with “Vessel\_Route\_ID” of ‘VR18453’, where the total of all bookings result to “11500”.

## Query 4 – Show all Crew Members That Belong to a Specific Vessel\_Route

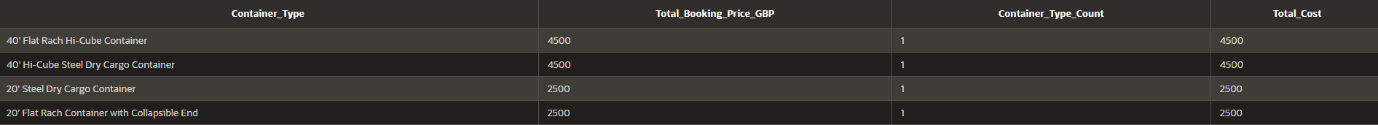
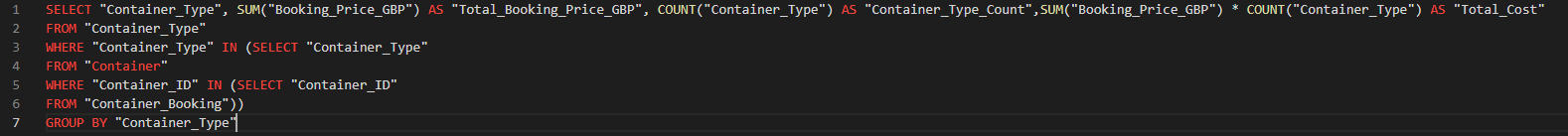
This query will be used by ECE to find all the employees that have been assigned to a Vessel\_Route. This will be done by selecting the “Employee\_ID”, “Employee\_First\_Name”, “Employee\_Surname”, “Employee\_Department” and “Employee\_Role” from the “Employee” table where we will link it with the “Vessel\_Route\_ID” all the way in the “Vessel\_Route” table, for this example, we will use the “Vessel\_Route\_ID” of ‘VR18453’. 

Graphical user interface, text

Description automatically generated

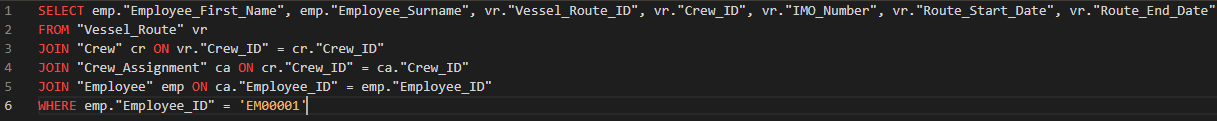
From these results, we can now see each employee, with their roles shown that have been assigned to this “Vessel\_Route”. This is a very helpful query in making sure each vessel has the correct number of crew, as well as having a member of each role that is needed.

## Query 5 – Profit Made by Each Container Type

This query is used to find out how profitable each vessel container is. This could be useful for the ECE, since if a specific container type is popular, they can order more for their inventory to potentially make more profit.

From the results we can see all the different container types that have been booked by customers, as well as the total cost of all the container types booked out, which is a valuable piece of information for ECE to have since it allows them to see which containers are popular.

## Query 6 – Finding How Many Vessel Routes Which a Specific Employee is Assigned

This query will be helpful to the employees of ECE since they can check their schedule to see which crew they will be working on for a vessel route, as well as the different start and end dates of said routes, so they know when they will be working.

For this example, we will be using the Employee with ID ‘EM00001’ to find out the details on the routes which they have been assigned to with their crew. This will be done by joining the tables “Crew”, “Vessel\_Route”, “Crew\_Assignment” and “Employee”. The “Employee\_First\_Name” and “Employee\_Surname” will be printed from the “Employee” table, whereas the “Vessel\_Route\_ID”, “Crew\_ID”, “IMO\_Number”, “Route\_Start\_Date” and “Route\_End\_Date” will be printed from the “Vessel\_Route” table.

The results above show all the different routes which the employee “John Smith” will be assigned on, as well as which crew he would be a part of during the journey. A screenshot of a computer

Description automatically generated with medium confidence