Assignment 1: Database Design

Module: CI7320

Due Date: 19th April 2023 (06:00)

Student ID: K2283306

Oracle Login Details:

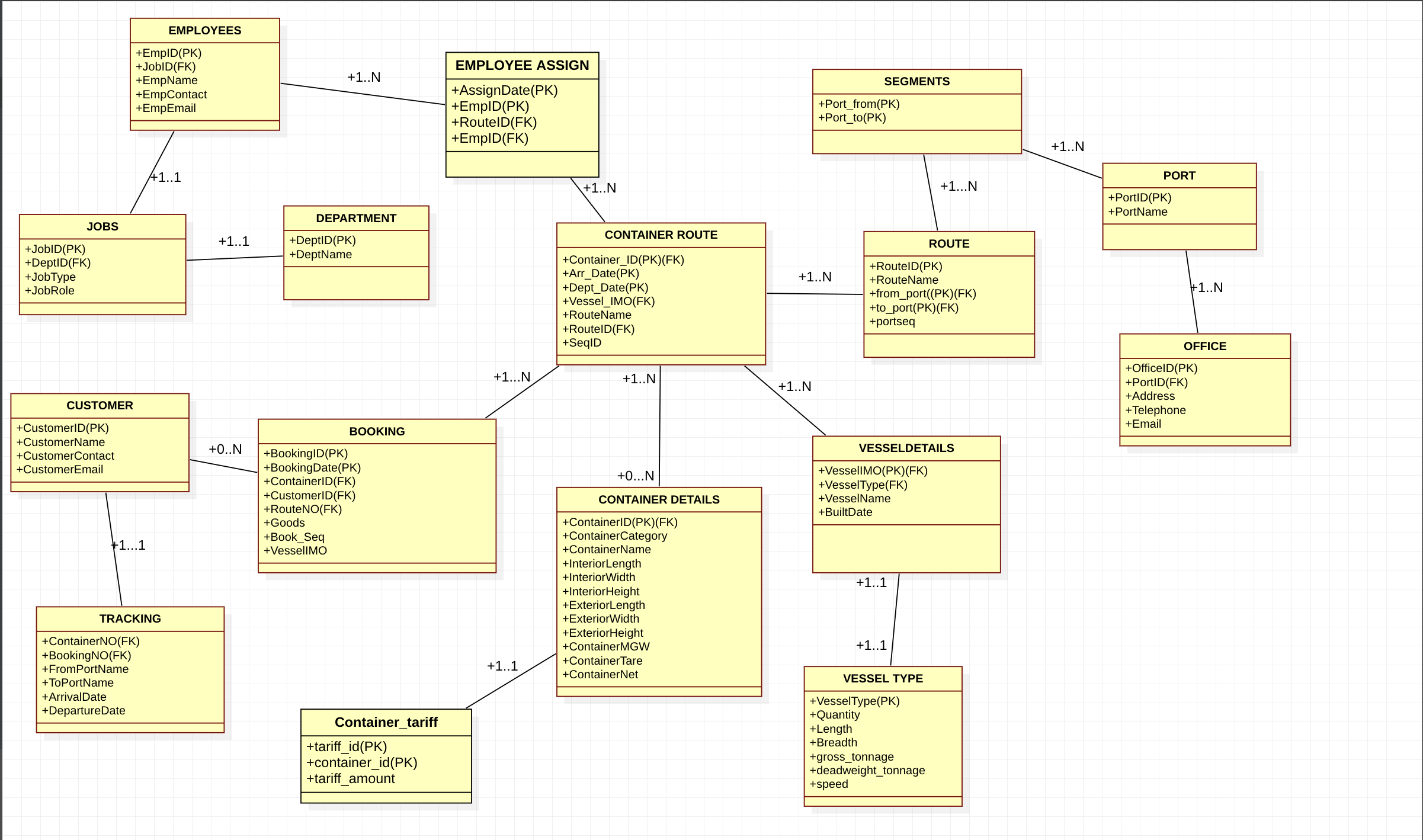
Workspace: SHAHESTA

Email: [K2283306@kingston.ac.uk](mailto:K2283306@kingston.ac.uk)

Password: K2283306

Student Name: Shahesta Nazir Ahmed

**Section 1: Entity relationship diagram with constraints and assumptions**

To have a clearer view of the diagram, kindly click on the link below:

<https://drive.google.com/file/d/1SO78igJ-r3ZYVY1-MuzGrBOdcmdZBpGE/view?usp=share_link>

Let us understand the constraints and assumptions by back tracking from the BOOKING table.

BOOKING has a COMPOSITE PRIMARY KEY as (BookingID,BookingDate). This table references a few others.

A) FOREIGN KEY ContainerID which references Container\_ID in the CONTAINER\_ROUTE table

B) Foreign RouteID which references a RouteID in the CONTAINER\_ROUTE table

C) FOREIGN KEY CustomerID which references CustomerID in the CUSTOMER table

Assumptions:

1. CONTAINER\_ROUTE has a one-many relationship with BOOKING. A booking can have multiple container routes. This is done so that a customer wishing to transport goods or collect goods from any segment of the route can do so. For a container to be assigned to a route a booking must be made.
2. CUSTOMER has a one-many relationship with BOOKING. A customer who is not registered can also make a booking, and his details will be recorded once the container has been successfully booked. This ensures that the company does not lose out on any potential customers. Just like how an e-commerce shopping platform would work.
3. A customer can make multiple BOOKINGS. This is taken care of by making a composite PRIMARY KEY (BookingID, BookingDate). Every BookingDate is linked to the dates a container has been locked. Every BookingID is linked to a unique customer for that booking.

We can observe here that the CUSTOMER table and TRACKING have a 1-1 relationship. TRACKING here is a VIEW and NOT a TABLE. This ensures that every time a new record is entered it can be reflected during tracking. Also

The CONTAINER\_ROUTE table has a COMPOSITE PRIMARY KEY (Container\_ID,Arr\_Date,Dept\_Date)

And its FOREIGN KEY constraints, namely:

FOREIGN KEY Container\_ID which references ContainerID in CONTAINERDETAILS table

FOREIGN KEY Vessel\_IMO references VesselIMO in the VESSELDETAILS table

FOREIGN KEY RouteID references RouteID in the ROUTE table.

Assumptions:

1. A container can be assigned to multiple routes. However, these are unique with respect to the dates that container is travelling that route. To elaborate, the same container can be on the same route but on a different date. On the contrary, a different container can be on the same route on the same date. A container must be associated with a booking for it to be assigned to a route.
2. A route is made up of multiple segments. These segments are ordered by a sequence ID, which helps us identify the ports the container is going to reach and when.
3. For a scheduled booking, a container is assigned to a vessel. Ideally this would be done with respect to which containers are available at the ports at the time of shipping. However, for this application, we have pre-existing data, that describes details of containers assigned to vessels.

The ROUTE table references SEGMENTS.

SEGMENTS contains all the possible pathways that Ever Blue Oceans Express owns.

ROUTE has one to many relationship with SEGMENTS.

SEGMENTS have a COMPOSITE PRIMARY KEY(Port\_from,Port\_to) and stores information from a port to every other port.

ROUTE (from\_port,to\_port) references SEGMENTS(Port\_from,Port\_to). Such a setting would result in duplicate data. To avoid that, ROUTE has a COMPOSITE PRIMARY KEY (route\_id,from\_port,to\_port). This can guarantee that different routes can have the same ports/port combinations

SEGMENTS references PORT to fetch the names of the ports. Every Port can have multiple offices, each having their own details. OFFICE details are stored separately. In a real-world setting, there are chances a port could be extremely busy.

Assumptions:

1. A segment can belong to multiple routes. But for a route to have a segment the route should exist.
2. A route can have multiple segments. Every segment is composed as a pair of two ports. Every port has a unique name.
3. (Port1,Port2) is different from (Port2,Port1).
4. Every Container belongs to a type. And every type of container will have the same information stored against it. For example, if 712222 and 713333 are both 20’ Dry Steel Cargo containers, they both will have similar dimensions which is stored in CONTAINERDETAILS.
5. Every CONTAINER is associated with a TARIFF(one to one). This information is stored on a different table CONTAINER\_TARIFF. Tariff amounts are associated with a unique tariff ID and container ID. PRIMARY KEY (tariffID,containerID).
6. In the event the tariff needs to be updated, having this structure would make that easy. (More on this in the Section 5- Critical Evaluation)
7. Information about vessels that Ever Blue owns is separated into two tables. VESSELDETAILS and VESSELTYPE. VESSELDETAILS has information pertaining to the VesselIMO, VesselName, VesselType (FOREIGN KEY references VESSELTYPE(vesseltype)) and BuiltDate.
8. Irrespective of how many distinct types of vessels ECE holds, it makes sense to store information on Vessel Specifics with the Vessel Type. Every vessel can be of any one of the types and would thus have exactly similar details associated with them. (except for the name of course).
9. Every Employee can be assigned to a route and this information is stored in EMPLOYEE\_ASSIGN, which associates an EMPLOYEE with a CONTAINER\_ROUTE which in turn is associated with a BOOKING.
10. Every employee can have exactly one job title. This is a one-one relationship between EMPLOYEE and JOB. A job type can have a DEPARTMENT. A Department can have multiple job titles.

\*\* NOTE: All tables are at least up to Third Normalization Form.

**Section2: Implementation. SQL Table definitions**

CREATE TABLE DEPARTMENT

(

DEPTID NUMBER(2) PRIMARY KEY,

DEPTNAME VARCHAR(40)

)

CREATE TABLE JOBS

(

JOBID NUMBER(10) PRIMARY KEY,

DEPTID NUMBER(2) NOT NULL,

FOREIGN KEY (DEPTID) REFERENCES DEPARTMENT(DEPTID),

JOBTYPE VARCHAR2(30),

JOBROLE VARCHAR2(30)

)

CREATE TABLE EMPLOYEE

(

EMPID NUMBER(4) PRIMARY KEY,

JOBID NUMBER(2) NOT NULL,

FOREIGN KEY(JOBID) REFERENCES JOBS(JOBID),

EMPNAME VARCHAR2(60),

EMPCONTACT VARCHAR2(15),

EMPEMAIL VARCHAR2(50)

)

CREATE TABLE OFFICE

(

OFFICEID NUMBER(10) PRIMARY KEY,

PORTID NUMBER(5),

ADDRESS VARCHAR2(100),

TELEPHONE VARCHAR2(15),

EMAIL VARCHAR2(100),

FOREIGN KEY(PORTID) REFERENCES PORT(PORTID)

)

CREATE TABLE PORTS

(

PORT\_ID NUMBER(3) PRIMARY KEY,

PORT\_NAME VARCHAR2(100) UNIQUE

)

CREATE TABLE SEGMENTS AS

SELECT SRC.PORT\_ID ID1,SRC.PORT\_NAME PORT1,TGT.PORT\_ID ID2,TGT.PORT\_NAME PORT2

FROM PORTS SRC, PORTS TGT

WHERE SRC.PORT\_NAME<>TGT.PORT\_NAME

ALTER TABLE SEGMENTS DROP CONSTRAINT PK\_SEGMENTS

ALTER TABLE SEGMENTS ADD CONSTRAINT PK\_SEGMENTS PRIMARY KEY(ID1,ID2)

CREATE TABLE CUSTOMER

(

CUSTOMERID NUMBER(10) PRIMARY KEY,

CUSTNAME VARCHAR2(100),

CUSTCONTACT VARCHAR(15),

CUSTEMAIL VARCHAR(60)

)

CREATE TABLE ROUTES

(

ROUTE\_ID NUMBER(3),

ROUTE\_NAME VARCHAR2(100),

FROM\_PORT NUMBER(3),

TO\_PORT NUMBER(3),

PORT\_SEQUENCE NUMBER(3)

)

ALTER TABLE ROUTES ADD CONSTRAINT FK\_ROUTES FOREIGN KEY (FROM\_PORT,TO\_PORT) REFERENCES SEGMENTS(ID1,ID2)

ALTER TABLE ROUTES ADD CONSTRAINT PK\_ROUTES PRIMARY KEY(ROUTE\_ID,FROM\_PORT,TO\_PORT)

CREATE TABLE VESSELDETAILS

(

VESSEL\_IMO NUMBER(10),

VESSEL\_TYPE VARCHAR2(1),

VESSEL\_NAME VARCHAR2(100),

BUILT\_DATE DATE DEFAULT SYSDATE

)

CREATE TABLE EMPLOYEE\_ROUTE

(

EMP\_ID NUMBER(10),

FROM\_PORT NUMBER(10),

TO\_PORT NUMBER(10),

ASSIGN\_DATE DATE DEFAULT SYSDATE

)

ALTER TABLE EMPLOYEE\_ROUTE ADD CONSTRAINT PK\_EMPID PRIMARY KEY(EMP\_ID,ASSIGN\_DATE)

ALTER TABLE EMPLOYEE\_ROUTE ADD CONSTRAINT FK\_EMPID FOREIGN KEY(EMP\_ID) REFERENCES EMPLOYEE(EMPID)

ALTER TABLE EMPLOYEE\_ROUTE ADD CONSTRAINT FK\_SEGMENTS FOREIGN KEY(FROM\_PORT,TO\_PORT) REFERENCES SEGMENTS(ID1,ID2)

CREATE TABLE CONTAINER\_TARIFF   
 ( TARIFF\_ID NUMBER(5,0) PRIMARY KEY,   
 TARIFF\_AMOUNT NUMBER(10,0),   
 CONTAINER\_ID NUMBER(10,0)  
 )

CREATE TABLE VESSELTYPE   
 ( VESSELTYPE VARCHAR2(1),   
 QUANTITY NUMBER(3,0),   
 LENGTH FLOAT(126),   
 BREADTH FLOAT(126),   
 GROSS\_TONNAGE FLOAT(126),   
 DEADWEIGHT\_TONNAGE FLOAT(126),   
 SPEED NUMBER(2,0),   
 PRIMARY KEY (VESSELTYPE)  
 )

CREATE TABLE CONTAINERDETAILS   
 ( CONTAINERID NUMBER(10,0),   
 CONTAINERCATEGORY VARCHAR2(20),   
 CONTAINERNAME VARCHAR2(200),   
 EXTERIORLENGTH FLOAT(126),   
 EXTERIORWIDTH FLOAT(126),   
 EXTERIORHEIGHT FLOAT(126),   
 INTERIORLENGTH FLOAT(126),   
 INTERIORWIDTH FLOAT(126),   
 INTERIORHEIGHT FLOAT(126),   
 CONTAINERMGW NUMBER(6,0),   
 CONTAINERTARE NUMBER(6,0),   
 CONTAINERNET NUMBER(6,0),   
 PRIMARY KEY (CONTAINERID)  
 )

CREATE TABLE CONTAINER\_ROUTE   
 ( ARR\_DATE DATE DEFAULT SYSDATE,   
 VESSEL\_IMO NUMBER(10,0),   
 CONTAINER\_ID NUMBER(10,0),   
 ROUTE\_NAME VARCHAR2(10),   
 SEQ\_ID NUMBER(5,0),   
 DEPT\_DATE DATE,   
 ROUTE\_ID NUMBER(10,0),   
 )  
ALTER TABLE ADD CONSTRAINT PK\_CD PRIMARY KEY (CONTAINER\_ID, ARR\_DATE, DEPT\_DATE)  
   
ALTER TABLE CONTAINER\_ROUTE ADD CONSTRAINT FK\_CONTAINER FOREIGN KEY (CONTAINER\_ID)  
 REFERENCES CONTAINERDETAILS (CONTAINERID)  
  
ALTER TABLE CONTAINER\_ROUTE ADD CONSTRAINT FK\_VESSEL FOREIGN KEY (VESSEL\_IMO)  
 REFERENCES VESSELDETAILS (VESSEL\_IMO)

CREATE TABLE BOOKING   
 ( BOOKING\_ID NUMBER(10,0) FOREIGN KEY (CONTAINER\_NO)REFERENCES CONTAINERDETAILS(CONTAINERID) ,   
 CUSTOMER\_ID NUMBER(10,0) FOREIGN KEY REFERENCES CUSTOMER(CUSTOMER\_ID),   
 BOOKING\_DATE DATE DEFAULT sysdate,   
 CONTAINER\_NO NUMBER(10,0),   
 VESSELIMO NUMBER(10,0),   
 BOOK\_SEQ NUMBER,   
 GOODS\_DESCR VARCHAR2(100),   
 ROUTE\_ID NUMBER(10,0),   
 )

ALTER TABLE BOOKING ADD CONSTRAINT PK\_BOOK PRIMARY KEY (BOOKING\_DATE, BOOKING\_ID)

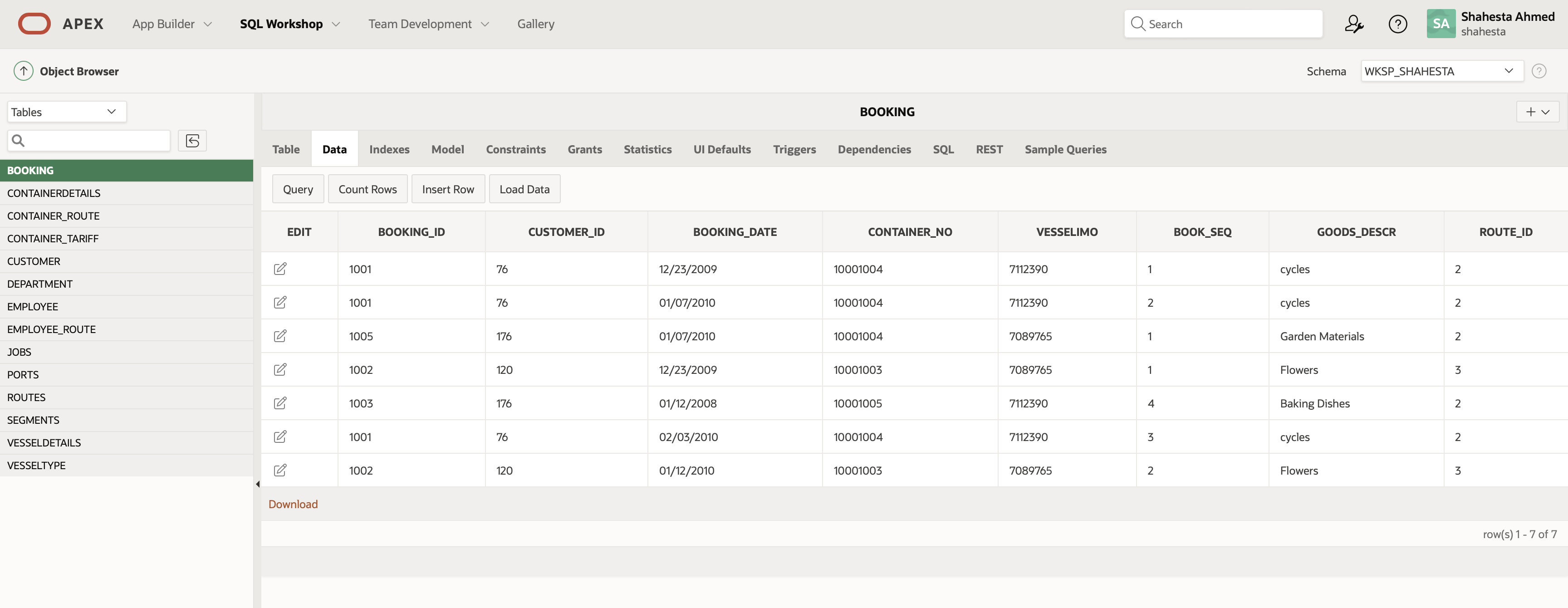
CREATE TABLE OFFICE

(

OFFICE\_ID NUMBER(5) PRIMARY KEY ,

OFFICE\_CONTACT VARCHAR2(15)

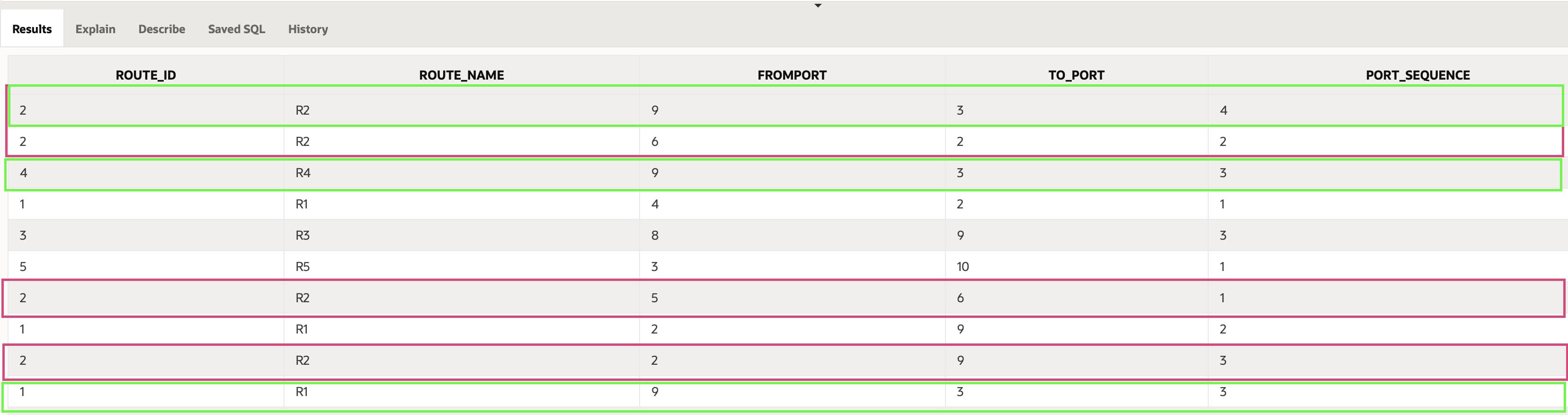
)

**A snapshot of Oracle Apex showing creation of tables**  


**Section 3: Discussion on the multiplicity of data that was entered.**



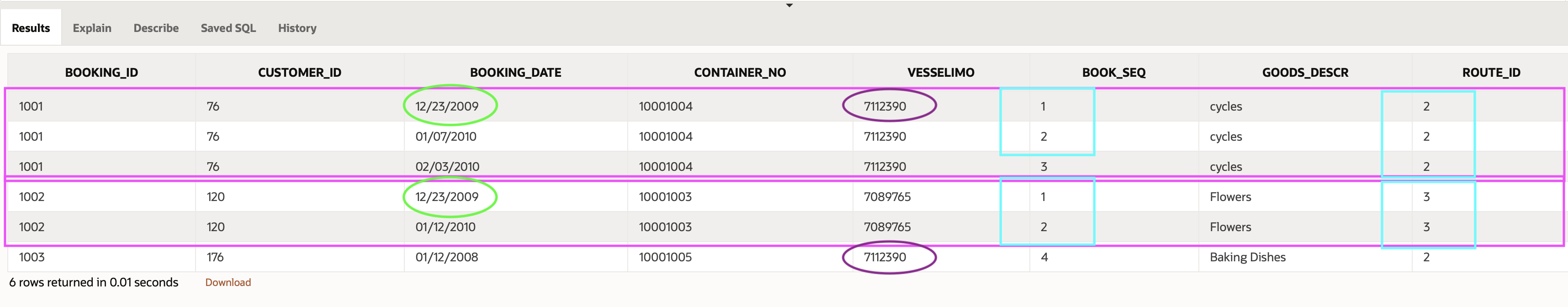
Two vessels are of the same type but with a unique VesselIMO, VesselName and BuiltDate.



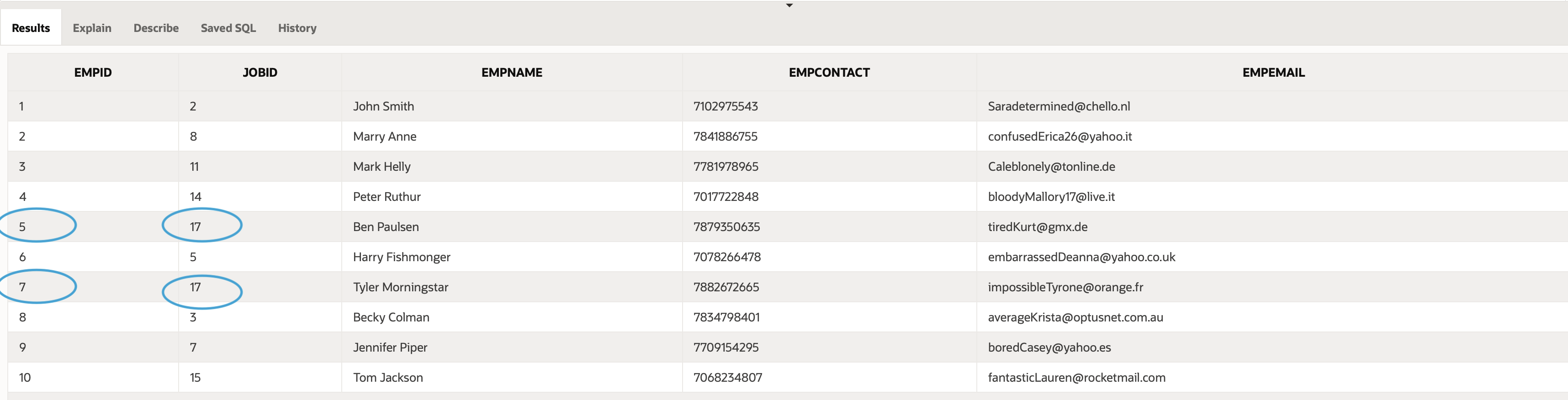
A Route can have multiple segments. Observe the red entries. Route R2 is made up of segments:

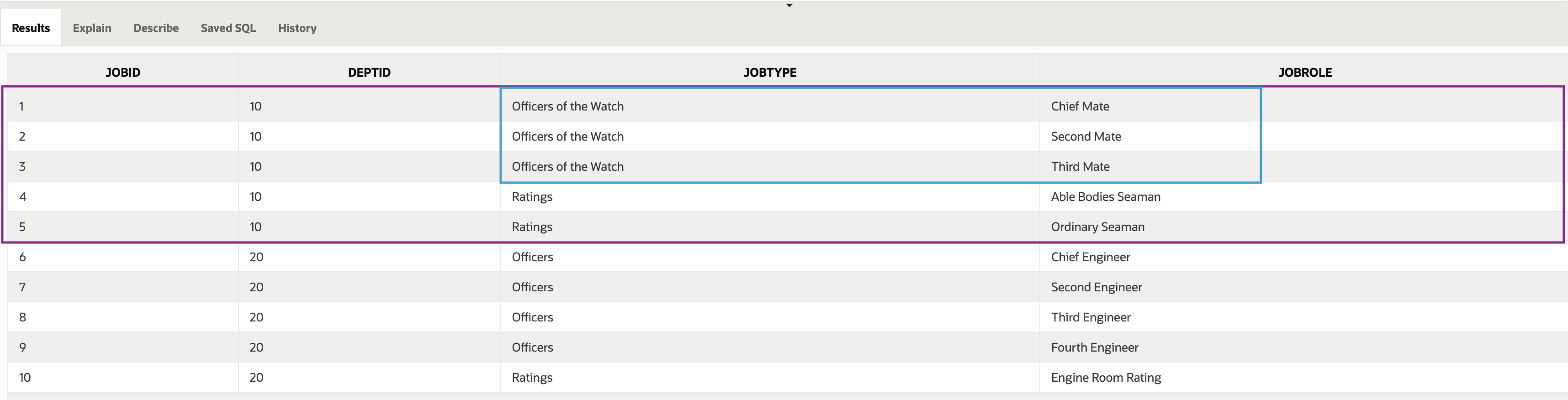
5-6-2-9-3.

A segment can belong on multiple routes. Observe the green entries. Route 1, Route 2 and Route 4 all have a common segment 9-3.

Observe the pink segments. We can observe that on the 23/12/2009 (green circles) there are two containers on 10001004 and 10001003 on route 2 on vessel 7112390 (purple circles). Container 10001004 will be dropped off at segment 3 of route 2 and container 10001003 will be dropped off at segment 2 of route 2. These are two different bookings by two different customers.

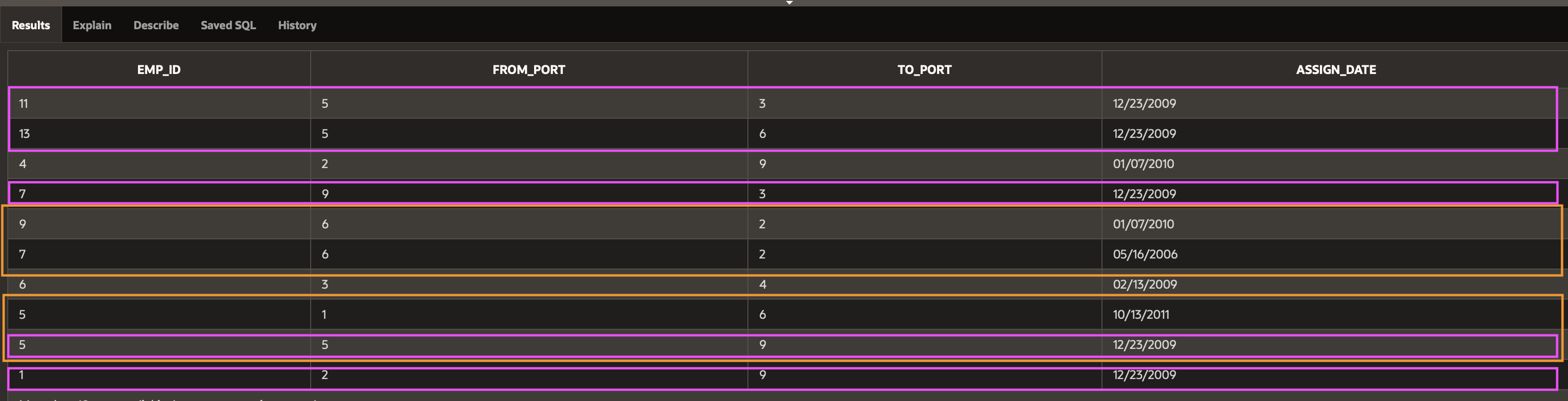
We can also see that the two routes share common segments 1,2 as is observed by the blue rectangles.



  
Two employees can have the same job id. However, every employee has just one job detail associated with them.

Every department can have multiple job titles and descriptions.

We have employee 9 assigned to two segment ports and employee 9 and 7 assigned to the same segment ports 6 and 2



Upon inspection, we can observe that employee 5 is assigned on two different dates. Employees 11 and 13 are both assigned on 23-12-2009 from port 9. And the pinks highlight all employees assigned for 23-12-2009.

**Section 4: SQL queries that meet system requirements**

--**SQL QUERY 1**

-- here is a small snapshot to show the variance in data.

-- we can observe from the following output that customer 76 has a booking 1001 of a container that is assigned route 2 and is going to be dropped off at segment 2 of route 2

-- we can also observe another customer who has a booking 1003 on the same date on the same vessel but the container is going to be dropped off at segment 4 of route 2

-- it also shows that two routes can have the same segments as in the case of booking 1001 and 1002, where the segment 6(Mundra) to 2(Santos) is common.

create view customer\_tracks\_booking as

select distinct b.customer\_id CustomerID,

b.booking\_id BookingID,r.route\_id RouteID,

c.route\_name RouteIdentity,

b.container\_no ContainerNO,c.seq\_id RouteSegment,

r.from\_port FromPortID,s.port1 FromPort,r.to\_port ToPortID,s.port2 ToPort,

c.arr\_date DateOfArrival,c.dept\_date DateofDeparture

from booking b join container\_route c

on b.booking\_date=c.arr\_date

join routes r on r.route\_id=b.route\_id

join segments s on s.id1=r.from\_port and s.id2=r.to\_port

where

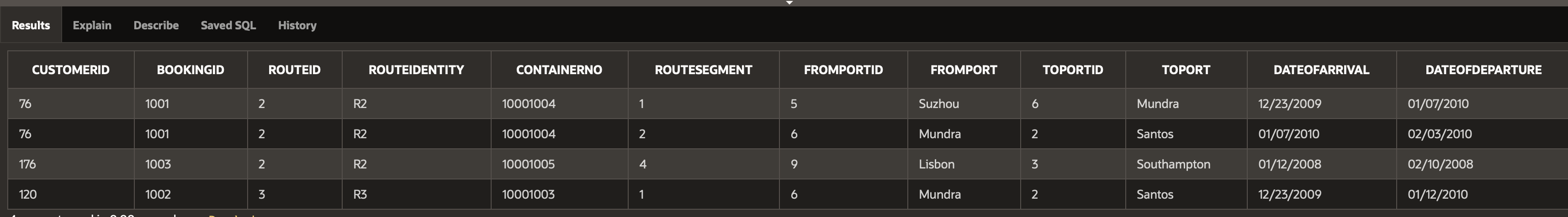
b.container\_no=c.container\_id

and b.route\_id=r.route\_id

and b.book\_seq=r.port\_sequence

and b.book\_seq=c.seq\_id

order by r.route\_id,b.customer\_id,c.seq\_id



select \* from customer\_tracks\_booking

-- track using booking\_id

select \* from customer\_tracks\_booking where bookingid=1001

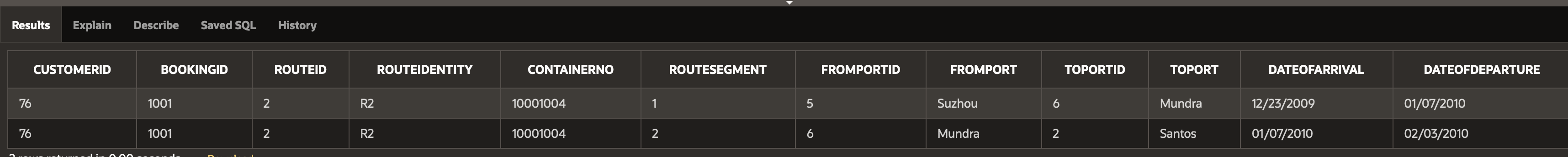
-- track using container\_id with respect to customer

select \* from customer\_tracks\_booking where containerno=10001004

-- we can further narrow it down with respect to the customer

--ex. container\_id=10001004 is booked by two customers 76,176 however on different booking dates.

select \* from customer\_tracks\_booking where containerno=10001004 and customerid=76



-- **SQL QUERY 2**

-- find the name of all employees who are assigned for booking no. 1001

-- let's explore further

-- find out how many of these assigned employees are captains and who?

select distinct er.assign\_date,b.vesselimo,e.empname,b.route\_id,b.book\_seq,s.port1, s.port2

from booking b

join employee\_route er

on b.booking\_date=er.assign\_date

join employee e

on e.empid=er.emp\_id

join segments s

on s.id1=er.from\_port and s.id2=er.to\_port

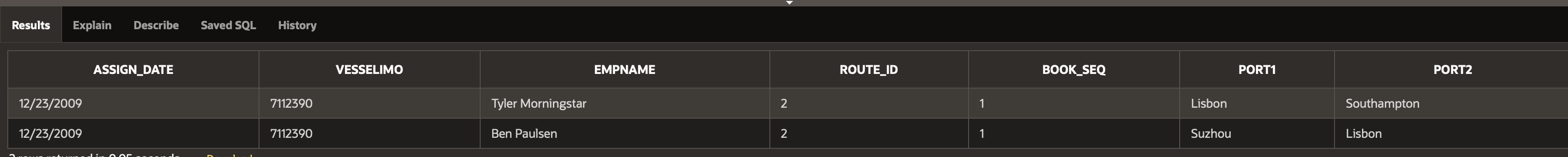
join jobs j

on e.jobid=j.jobid

where j.jobrole='Captain'

and

booking\_id=1001



-- **SQL QUERY 3**

-- find name, speed, gross tonnage, dead weight tonnage of vessel, starting with 71 and the tariff associated with the container(/s) on that vessel.

select distinct booking\_id,vessel\_imo,vessel\_name,b.container\_no,ct.tariff\_amount,vt.speed,vt.gross\_tonnage,vt.deadweight\_tonnage

from vesseldetails vd

join booking b on vd.vessel\_imo=b.vesselimo

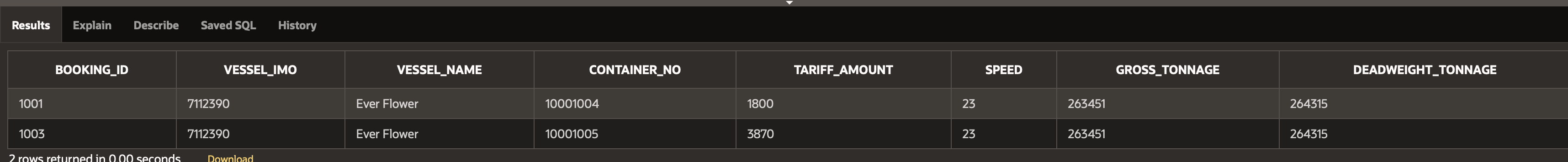
join vesseltype vt

on vt.vesseltype=vd.vessel\_type

join container\_tariff ct

on ct.container\_id=b.container\_no

where vd.vessel\_imo like '71%'



-- **SQL QUERY 4**

-- re-iterating to query 1 where we created a view customer\_tracks\_booking.

--Let's assume a customer wants to know the contact details of the office at a specific port, to enquire about his goods.

select distinct customerid,bookingid,containerno,fromport,toport,office\_contact from office o

join customer\_tracks\_booking b on

o.office\_id=b.toportid

where bookingid=1001



-- **SQL QUERY 5**

-- ECE wants to know how many vessels it currently has

create view vessel\_count as

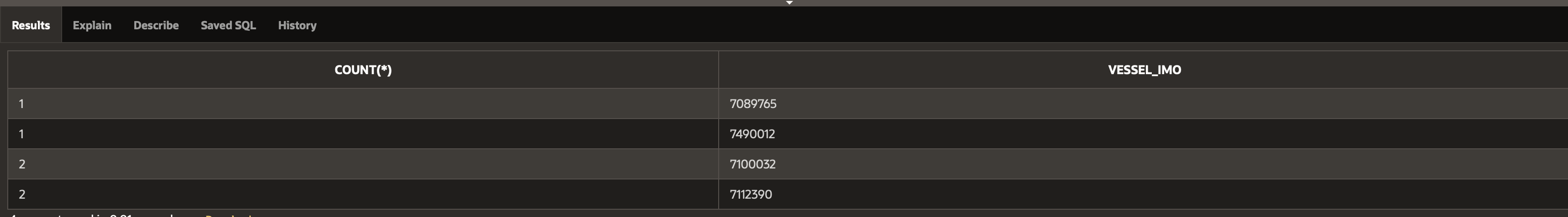
select vessel\_type,vessel\_imo from vesseldetails where vessel\_type=(

select vesseltype from vesseltype where vesseltype=vessel\_type and speed=23

)

select count(\*),vc.vessel\_imo from vessel\_count vc join vesseldetails vd on vc.vessel\_type=vd.vessel\_type

group by vc.vessel\_imo



-- **SQL QUERY 6**

-- ECE wants to know the names of ports that are common among all routes and which occur more than twice

create view unique\_ports as

select from\_port port,count(\*) count from routes

group by (from\_port)

having count(from\_port)>2

union

select to\_port port,count(\*) count from routes

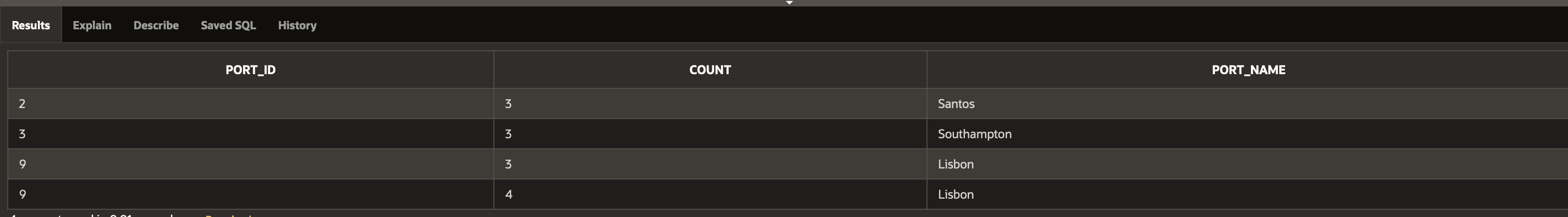
group by (to\_port)

having count(to\_port)>2

select port\_id,count,port\_name from ports p

join unique\_ports up

on up.port=p.port\_id



**Chapter 5: Critical evaluation**

This project was very well aligned to a real-world business problem. The journey from understanding client requirements and making sure the database design left room for updates, in unlikely scenarios, is what was most challenging. It took me about 9 days to agree with the design I wanted to go with (the 5th design). I wanted to achieve a design that could at least be a prototype of the real application. This led to numerous trial and error sessions about which tables should go where and do we really need those tables.

However, this project took an interesting turn of events for me. It happened when I reached a stage where I wanted to calculate container tariffs based on the distance they had travelled. And I hit upon a gold mine here. With the static design we were required to create, my idea did not fit in with the project. But I did go on a treasure hunt using Oracle Blog Posts. I was trying to find a logic using Dijkstra's algorithm. As so happens, Oracle has launched Property Graphs that can tackle such complex relationships easily. Oracle also comes equipped with Oracle Maps, that we can integrate with software (requires understanding of docker).

Due to time constraints I never got to explore further. I did find some useful resources such as <https://inviqa.com/blog/storing-graphs-database-sql-meets-social-network>. I did try to work with this, but I was getting a few errors.

I learnt a lot of new ways SQL can be used, as it has advanced a lot. I have a lot of studying to do! I learnt how to add constraints after a table has been created which helped my case when creating certain tables I was skeptical about with regards to which primary key and foreign key constraints should be associated with it. I learnt how we can find specific values in all the columns of the table. I learnt about JOINS and UNION.

Something I attempted was an unsuccessful endeavor for me. I hypothesized that if I knew the existing data in a table and wanted to add new data linking to the existing data, I could just insert it into. I tried cross joins, which worked just once and then, the query threw a unique key constraint error. I wanted to achieve some sense of real time in the static tables. But it looks like that is another chapter about SQL I got to master.

Given an opportunity I would focus on the bigger picture. I realised after a significant amount of time that this was an ‘Optimizing the Routing’ problem. I would start with what data I was going to enter and create my tables and then perhaps go to the design. That way my design would reflect my latest understanding of the requirements. There would be an initial design just to put things in retrospect.

Overall, I enjoyed the learning curve. It was challenging and I enjoyed every minute of it.