Global Shipping Database Design

Report

Kulanika Gnanaratna

Introduction

This project is based on a shipping and logistics company that handles the movement of containers and goods across global ports on behalf of their customers. The system is designed to manage real-life operations like scheduling vessels along routes, assigning crew, booking containers for customers, tracking goods, and monitoring container movement at different ports.

Aim:

To design and implement a fully functional and normalized relational database that supports the key operational needs of a global shipping company, while enabling complex queries, enforcing business rules, and providing insight into logistics performance.

Chapter 1 – ER Diagram

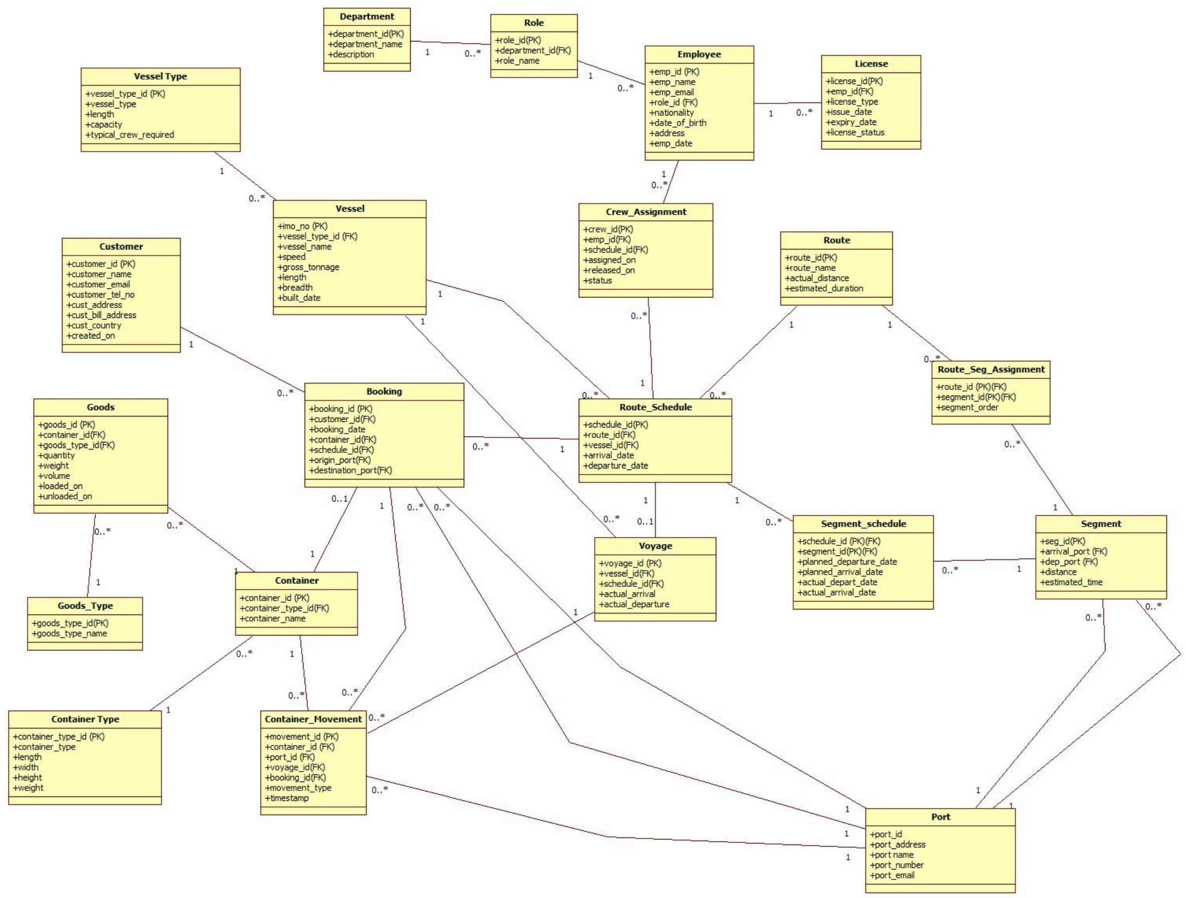


Figure 1: ER Diagram

General Assumptions

- The company owns all containers and vessels.
- The system supports global logistics operations, with ports across different countries.
- The database is designed for both planning and operational tracking.
- Each container carries one type of goods per booking.

Ports and Routes

- All ports are uniquely identified by name and have basic contact information.
- Routes are made up of multiple reusable segments connecting two distinct ports.
- Segment order is maintained to preserve route sequencing.

Scheduling and Voyage Tracking

- Route schedules define the planned deployment of vessels along a route.
- Voyages represent the actual execution of those schedules.
- Every schedule must be assigned to a specific vessel.

Containers and Goods

- Container types are used to define the size and capacity constraints.
- Goods are linked to containers and classified using a predefined Goods Type table.
- The system tracks goods at the category level not item level to support operational planning and container routing.

Customers and Bookings

- Customers can book one container per booking to transport goods from one port to another.
- Origin and destination ports in a booking must be different.
- Customer data includes basic contact and billing information.

Crew and Assignments

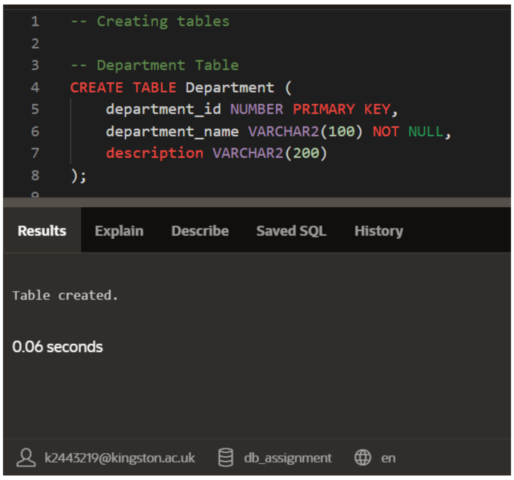
- Employees belong to departments and have defined roles.
- Some roles might require valid licenses.
- Only one license per employee can be considered valid at any given time.
- Crew members are assigned to route schedules based on the vessel type's required crew count.
- Assignment dates and status are tracked using predefined statuses such as Assigned, Active and Completed.

Movement Tracking

- Container movement is logged at each port it passes through during a voyage.

Chapter 2 – Implementation

Table Creation was done as follows,



```sql
-- Creating tables

-- Department Table
CREATE TABLE Department (
    department_id NUMBER PRIMARY KEY,
    department_name VARCHAR2(100) NOT NULL,
    description VARCHAR2(200)
);
```

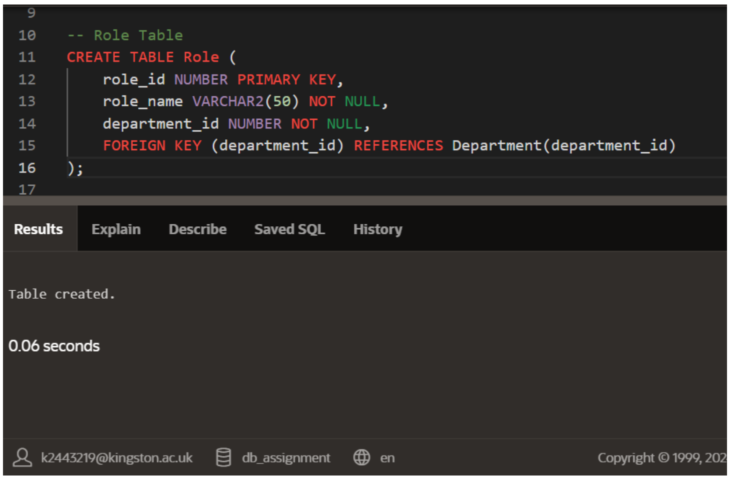
Results Explain Describe Saved SQL History

Table created.

0.06 seconds

k2443219@kingston.ac.uk db_assignment en

Figure 2: CREATE TABLE Statements for Department



```sql
-- Role Table
CREATE TABLE Role (
    role_id NUMBER PRIMARY KEY,
    role_name VARCHAR2(50) NOT NULL,
    department_id NUMBER NOT NULL,
    FOREIGN KEY (department_id) REFERENCES Department(department_id)
);
```

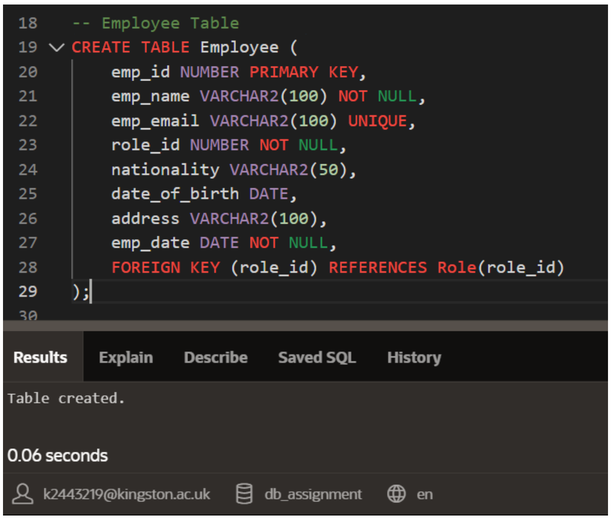
Results Explain Describe Saved SQL History

Table created.

0.06 seconds

k2443219@kingston.ac.uk db_assignment en Copyright © 1999, 202

Figure 3: CREATE TABLE Statements for Role



```sql
-- Employee Table
CREATE TABLE Employee (
    emp_id NUMBER PRIMARY KEY,
    emp_name VARCHAR2(100) NOT NULL,
    emp_email VARCHAR2(100) UNIQUE,
    role_id NUMBER NOT NULL,
    nationality VARCHAR2(50),
    date_of_birth DATE,
    address VARCHAR2(100),
    emp_date DATE NOT NULL,
    FOREIGN KEY (role_id) REFERENCES Role(role_id)
);
```

Results Explain Describe Saved SQL History

Table created.

0.06 seconds

k2443219@kingston.ac.uk db_assignment en

Figure 4: CREATE TABLE Statements for Employee

```sql
-- Table License
CREATE TABLE License (
    license_id NUMBER PRIMARY KEY,
    emp_id NUMBER NOT NULL,
    license_type VARCHAR2(50),
    issue_date DATE,
    expiry_date DATE,
    license_status VARCHAR2(20),
    FOREIGN KEY (emp_id) REFERENCES Employee(emp_id),
    CONSTRAINT Chk_Lic_Status CHECK (license_status IN ('Valid', 'Invalid')),
    CONSTRAINT Chk_Lic_date CHECK (issue_date < expiry_date)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.08 seconds

k2443219@kingston.ac.uk db_assignment en Copyright © 1999, 2024, Oracle and/

Figure 5: CREATE TABLE Statements for License

```sql
-- Table Vessel Type
CREATE TABLE Vessel_Type (
    vessel_type_id VARCHAR2(5) PRIMARY KEY,
    vessel_type VARCHAR2(50),
    length NUMBER(6,2),
    capacity NUMBER,
    typical_crew_required NUMBER,
    CONSTRAINT Chk_V_Length CHECK (length > 0),
    CONSTRAINT Chk_V_Capacity CHECK (capacity > 0),
    CONSTRAINT Chk_V_Crew CHECK (typical_crew_required > 0),
    CONSTRAINT Chk_V_Name CHECK (vessel_type_id = UPPER(vessel_type_id))
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.05 seconds

k2443219@kingston.ac.uk db_assignment en Copyright © 1999, 2024, Oracle an

Figure 6: CREATE TABLE Statements for Vessel Type

```sql
-- Table Vessel
CREATE TABLE Vessel (
    imo_no NUMBER PRIMARY KEY,
    vessel_type_id VARCHAR2(5) NOT NULL,
    vessel_name VARCHAR2(100) NOT NULL,
    speed NUMBER(5,2),
    gross_tonnage NUMBER,
    length NUMBER(6,2),
    breadth NUMBER(6,2),
    built_date DATE,
    FOREIGN KEY (vessel_type_id) REFERENCES Vessel_Type(vessel_type_id),
    CONSTRAINT Chk_S_Speed CHECK (speed > 0),
    CONSTRAINT Chk_S_Tonnage CHECK (gross_tonnage > 0),
    CONSTRAINT Chk_S_Length CHECK (length > 0),
    CONSTRAINT Chk_S_breadth CHECK (breadth > 0)
);
```

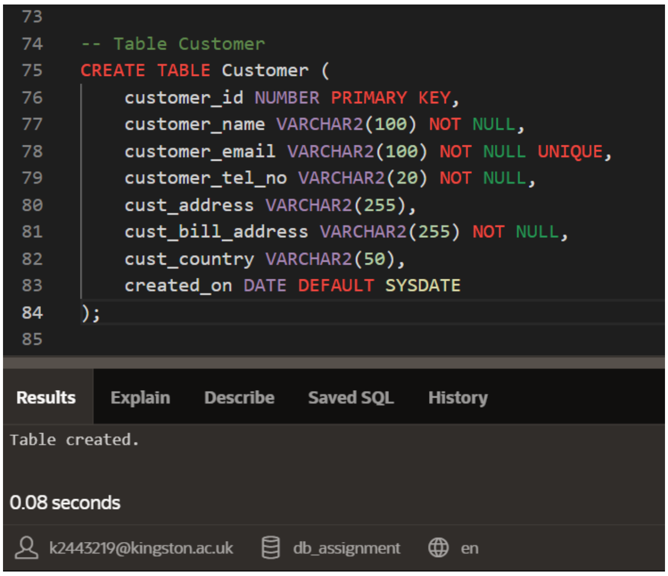
Results Explain Describe Saved SQL History

```
Table created.
```

0.07 seconds

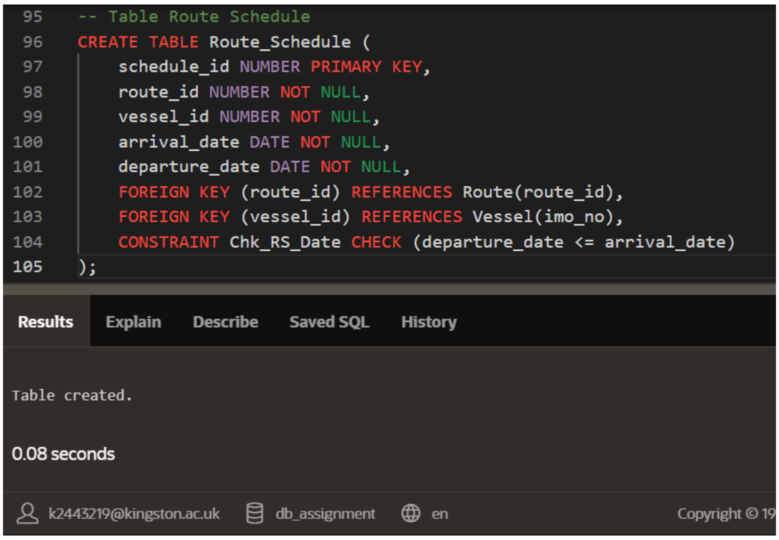
k2443219@kingston.ac.uk db_assignment en Copyri

Figure 7: CREATE TABLE Statements for Vessel



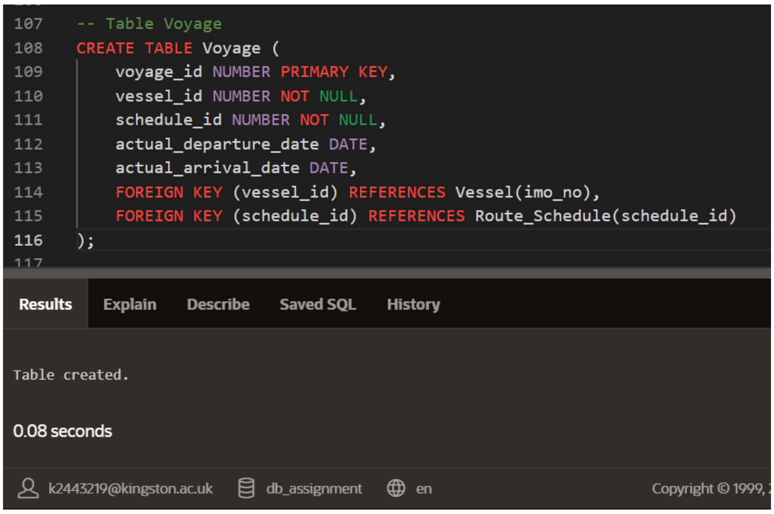
```sql
-- Table Customer
CREATE TABLE Customer (
    customer_id NUMBER PRIMARY KEY,
    customer_name VARCHAR2(100) NOT NULL,
    customer_email VARCHAR2(100) NOT NULL UNIQUE,
    customer_tel_no VARCHAR2(20) NOT NULL,
    cust_address VARCHAR2(255),
    cust_bill_address VARCHAR2(255) NOT NULL,
    cust_country VARCHAR2(50),
    created_on DATE DEFAULT SYSDATE
);
```


Figure 8: CREATE TABLE Statements for Customer



```sql
-- Table Route Schedule
CREATE TABLE Route_Schedule (
    schedule_id NUMBER PRIMARY KEY,
    route_id NUMBER NOT NULL,
    vessel_id NUMBER NOT NULL,
    arrival_date DATE NOT NULL,
    departure_date DATE NOT NULL,
    FOREIGN KEY (route_id) REFERENCES Route(route_id),
    FOREIGN KEY (vessel_id) REFERENCES Vessel(imo_no),
    CONSTRAINT Chk_RS_Date CHECK (departure_date <= arrival_date)
);
```


Figure 9: CREATE TABLE Statements for Route Schedule



```sql
-- Table Voyage
CREATE TABLE Voyage (
    voyage_id NUMBER PRIMARY KEY,
    vessel_id NUMBER NOT NULL,
    schedule_id NUMBER NOT NULL,
    actual_departure_date DATE,
    actual_arrival_date DATE,
    FOREIGN KEY (vessel_id) REFERENCES Vessel(imo_no),
    FOREIGN KEY (schedule_id) REFERENCES Route_Schedule(schedule_id)
);
```


Figure 10: CREATE TABLE Statements for Voyage

```sql
-- Table Port
CREATE TABLE Port (
    port_id NUMBER PRIMARY KEY,
    port_name VARCHAR2(100) NOT NULL UNIQUE,
    port_address VARCHAR2(100),
    port_number VARCHAR2(20),
    port_email VARCHAR2(100) UNIQUE
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.10 seconds

k2443219@kingston.ac.uk db_assignment en

Figure 11: CREATE TABLE Statements for Port

```sql
-- Table Segment
CREATE TABLE Segment (
    seg_id NUMBER PRIMARY KEY,
    dep_port NUMBER NOT NULL,
    arrival_port NUMBER NOT NULL,
    distance NUMBER(6,2),
    estimated_time INTERVAL DAY TO SECOND,
    FOREIGN KEY (dep_port) REFERENCES Port(port_id),
    FOREIGN KEY (arrival_port) REFERENCES Port(port_id),
    CONSTRAINT Chk_Seg_Port CHECK (dep_port <> arrival_port)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.08 seconds

k2443219@kingston.ac.uk db_assignment en Copy

Figure 12: CREATE TABLE Statements for Segment

```sql
-- Table Route Seg Assignment
CREATE TABLE Route_Seg_Assignment (
    route_id NUMBER NOT NULL,
    segment_id NUMBER NOT NULL,
    segment_order NUMBER,
    PRIMARY KEY (route_id, segment_id),
    FOREIGN KEY (route_id) REFERENCES Route(route_id),
    FOREIGN KEY (segment_id) REFERENCES Segment(seg_id),
    CONSTRAINT Chk_RSeg_Order CHECK (segment_order > 0)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.07 seconds

k2443219@kingston.ac.uk db_assignment en

Figure 13: CREATE TABLE Statements for Route Seg Assignment

```sql
-- Table Segment Schedule
CREATE TABLE Segment_Schedule (
    schedule_id NUMBER NOT NULL,
    segment_id NUMBER NOT NULL,
    planned_departure_date DATE,
    planned_arrival_date DATE,
    actual_depart_date DATE,
    actual_arrival_date DATE,
    PRIMARY KEY (schedule_id, segment_id),
    FOREIGN KEY (schedule_id) REFERENCES Route_Schedule(schedule_id),
    FOREIGN KEY (segment_id) REFERENCES Segment(seg_id),
    CONSTRAINT Chk_SegSchedule_Date CHECK (planned_departure_date <= planned_arrival_date)
);
```

Results | Explain | Describe | Saved SQL | History

```
Table created.
```

0.05 seconds

k2443219@kingston.ac.uk db_assignment en

Figure 14: CREATE TABLE Statements for Segment Schedule

```sql
-- Table Crew Assignment
CREATE TABLE Crew_Assignment (
    crew_id NUMBER PRIMARY KEY,
    emp_id NUMBER NOT NULL,
    schedule_id NUMBER NOT NULL,
    assigned_on DATE DEFAULT SYSDATE,
    released_on DATE,
    status VARCHAR2(20),
    UNIQUE (emp_id, schedule_id),
    FOREIGN KEY (emp_id) REFERENCES Employee(emp_id),
    FOREIGN KEY (schedule_id) REFERENCES Route_Schedule(schedule_id),
    CONSTRAINT Chk_CA_Status CHECK (status IN ('Assigned','Active','Completed','Cancelled')),
    CONSTRAINT Chk_CA_Date CHECK (released_on IS NULL OR assigned_on <= released_on)
);
```

Results | Explain | Describe | Saved SQL | History

```
Table created.
```

0.08 seconds

k2443219@kingston.ac.uk db_assignment en

Figure 15: CREATE TABLE Statements for Crew Assignment

```sql
-- Table Container Type
CREATE TABLE Container_Type (
    container_type_id NUMBER PRIMARY KEY,
    container_type VARCHAR2(50) NOT NULL,
    length NUMBER(5,2),
    width NUMBER(5,2),
    height NUMBER(5,2),
    weight NUMBER(6,2),
    CONSTRAINT Chk_CT_length CHECK (length > 0),
    CONSTRAINT Chk_CT_width  CHECK (width > 0),
    CONSTRAINT Chk_CT_height CHECK (height > 0),
    CONSTRAINT Chk_CT_weight CHECK (weight > 0)
);
```

Results | Explain | Describe | Saved SQL | History

```
Table created.
```

0.06 seconds

k2443219@kingston.ac.uk db_assignment en

Figure 16: CREATE TABLE Statements for Container Type

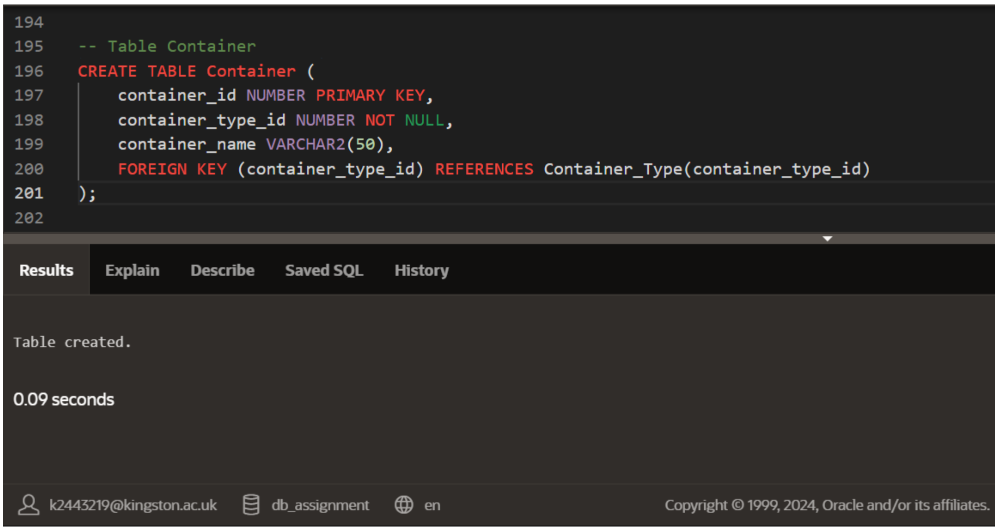


Figure 17: CREATE TABLE Statements for Container

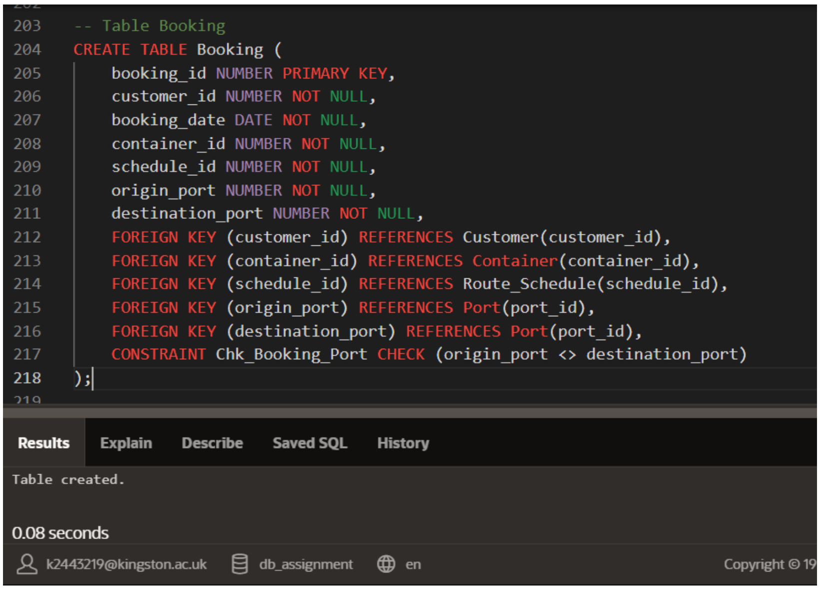


Figure 18: CREATE TABLE Statements for Booking

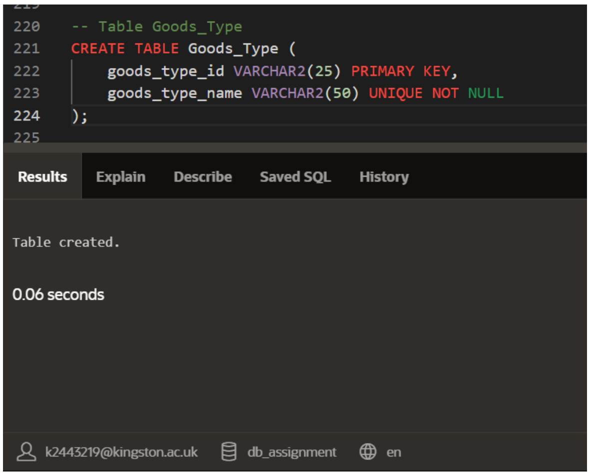
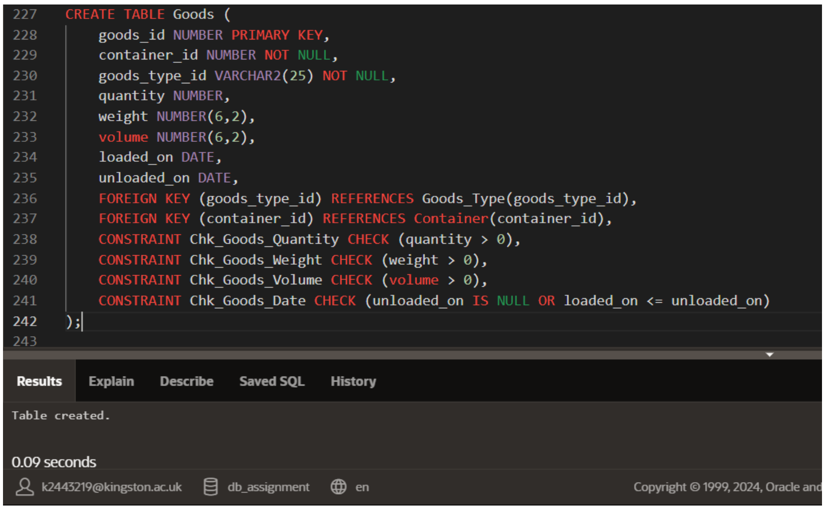
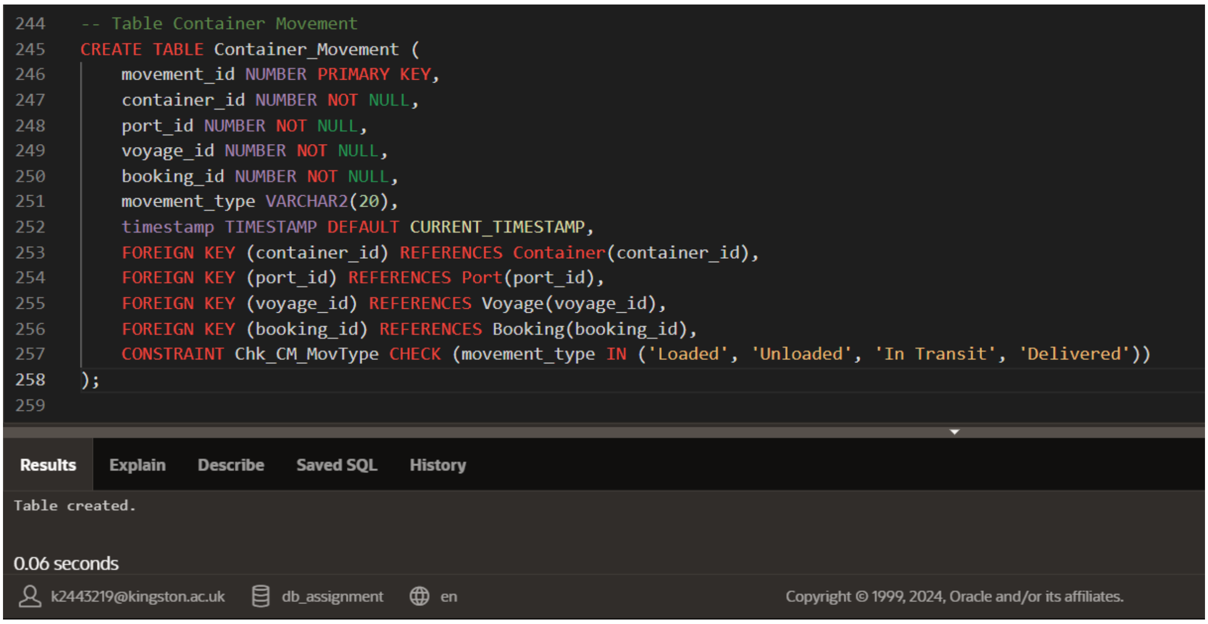


Figure 19: CREATE TABLE Statements for Goods Type



```sql
CREATE TABLE Goods (
    goods_id NUMBER PRIMARY KEY,
    container_id NUMBER NOT NULL,
    goods_type_id VARCHAR2(25) NOT NULL,
    quantity NUMBER,
    weight NUMBER(6,2),
    volume NUMBER(6,2),
    loaded_on DATE,
    unloaded_on DATE,
    FOREIGN KEY (goods_type_id) REFERENCES Goods_Type(goods_type_id),
    FOREIGN KEY (container_id) REFERENCES Container(container_id),
    CONSTRAINT Chk_Goods_Quantity CHECK (quantity > 0),
    CONSTRAINT Chk_Goods_Weight CHECK (weight > 0),
    CONSTRAINT Chk_Goods_Volume CHECK (volume > 0),
    CONSTRAINT Chk_Goods_Date CHECK (unloaded_on IS NULL OR loaded_on <= unloaded_on)
);
```


Figure 20: CREATE TABLE Statements for Goods



```sql
-- Table Container Movement
CREATE TABLE Container_Movement (
    movement_id NUMBER PRIMARY KEY,
    container_id NUMBER NOT NULL,
    port_id NUMBER NOT NULL,
    voyage_id NUMBER NOT NULL,
    booking_id NUMBER NOT NULL,
    movement_type VARCHAR2(20),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (container_id) REFERENCES Container(container_id),
    FOREIGN KEY (port_id) REFERENCES Port(port_id),
    FOREIGN KEY (voyage_id) REFERENCES Voyage(voyage_id),
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id),
    CONSTRAINT Chk_CM_MovType CHECK (movement_type IN ('Loaded', 'Unloaded', 'In Transit', 'Delivered'))
);
```


Figure 21: CREATE TABLE Statements for Container Movement

Chapter 3 – Data Entry

The Department Table holds realistic data representing the key divisions within the shipping company.

DEPARTMENT_ID	DEPARTMENT_NAME	DESCRIPTION
1	Deck Department	Responsible for navigation and overall ship operation.
2	Engine Department	Manages propulsion, fuel, and mechanical systems.
3	Steward Department	Handles accommodations and hospitality services.
4	Operations Department	Coordinates shipping schedules, logistics, and port operations.
5	Customer Service	Manages client bookings and inquiries.
6	Human Resources	Handles employee records, recruitment, and compliance.

6 rows returned in 0.01 seconds Download

k2443219@kingston.ac.uk db_assignment en Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.2

Figure 22: Data for Table Department

The Role Table holds realistic data representing the job roles dispersed within the company and its departments. The multiplicity of the relationship of the Role and Department tables [One to Many: A Department can have many Roles] is shown below.

ROLE_ID	ROLE_NAME	DEPARTMENT_ID
1	Captain	1
2	First Officer	1
3	Deck Cadet	1
4	Navigation Officer	1
5	Safety Officer	1
6	Chief Engineer	2
7	Second Engineer	2
8	Engine Cadet	2
9	Oiler	2
10	Electrician	2
11	Chief Steward	3
12	Cook	3

k2443219@kingston.ac.uk db_assignment en Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.2

Figure 23: Data for Table Role

The Employee table stores realistic detailed records of all staff members, including their personal information, role assignments, nationality, and employment dates. The multiplicity of the relationship of the Role and Employee tables [One to Many: A Role can be linked to many Employees] is shown below.

EMP_ID	EMP_NAME	EMP_EMAIL	ROLE_ID	NATIONALITY	DATE_OF_BIRTH	ADDRESS	EMP_DATE
56	Rebecca Thomas	rebecca.thomas56@shippingco.com	22	Andorra	3/9/1980	55982 Shawn Creek Suite 310, East Gregory, FL 44063	1/14/2023
57	Mary Castillo	mary.castillo57@shippingco.com	22	Bahamas	12/6/1984	78468 Cochran Unions Apt. 236, Josetown, MA 56934	1/12/2020
58	Teresa Lee	teresa.lee58@shippingco.com	22	Maldives	6/22/1970	0142 Diaz Mountains, West Jennifermouth, OH 63196	1/11/2022
59	Russell King	russell.king59@shippingco.com	23	Romania	3/22/1980	613 Courtney Way, Millerbury, VT 12567	12/22/2023
60	Tony Greene	tony.greene60@shippingco.com	23	Trinidad and Tobago	6/21/1990	498 Lisa Centers Suite 292, Isabelview, VA 70533	6/14/2023
61	Jennifer Jenkins	jennifer.jenkins61@shippingco.com	23	Ecuador	7/18/1997	81049 Gary Shores Suite 554, Port Gregoryfort, MD 27438	1/14/2024
62	Keith Allen	keith.allen62@shippingco.com	24	Uruguay	10/8/1986	1701 Kelly Station Apt. 356, Lake Saraview, CT 50779	10/25/2020
63	Kyle Bernard	kyle.bernard63@shippingco.com	24	Peru	2/5/1990	523 Nicole Forge, West Anne, WI 02563	6/25/2023
64	Jason Cole	jason.cole64@shippingco.com	25	Uruguay	6/2/1978	24562 Cooper Lane, Port Lukeview, FL 64782	5/30/2021
65	Mark Ellison	mark.ellison65@shippingco.com	25	Canada	6/27/1992	980 Mark Park, Lake Scott, AK 38057	8/29/2023
66	Brenda Hanson	brenda.hanson66@shippingco.com	25	Finland	5/26/1983	80285 Joyce Underpass, West Amber, NM 70877	12/24/2020
67	James Smith	james.smith67@shippingco.com	26	Cyprus	12/4/1975	3076 Christina Throughway Apt. 741, New Jay, OK 10207	5/28/2021
68	Lori Pearson	lori.pearson68@shippingco.com	26	Canada	1/25/1976	899 Neal Gateway Apt. 950, West Noah, ID 86712	9/12/2020
1	Leah White	leah.white1@shippingco.com	1	Dominican Republic	4/15/1972	81767 Stephanie River, South Deborah, NV 25416	10/3/2022

Figure 24: Data for Table Employees

The license table holds realistic details about licenses assigned to employees in roles that require official certification. The multiplicity of the relationship of the License and Employee tables [One to Many: Each employee can have multiple licenses over time whether they are renewals or expired records with only one valid entry. Not all job roles require license] is shown below.

LICENSE_ID	EMP_ID	LICENSE_TYPE	ISSUE_DATE	EXPIRY_DATE	LICENSE_STATUS
1	1	Captain License	10/3/2022	10/3/2027	Valid
2	2	Captain License	8/23/2019	8/23/2024	Invalid
3	2	Captain License	8/24/2024	8/24/2029	Valid
4	3	Captain License	10/13/2022	10/13/2027	Valid
5	4	First Officer License	1/2/2024	1/2/2029	Valid
6	5	First Officer License	10/16/2017	10/16/2022	Invalid
7	5	First Officer License	10/17/2022	10/17/2027	Valid
8	6	First Officer License	5/12/2023	5/12/2028	Valid
9	7	First Officer License	9/12/2022	9/12/2027	Valid
10	8	Deck Cadet License	11/17/2017	11/17/2022	Invalid
11	8	Deck Cadet License	11/18/2022	11/18/2027	Valid
12	9	Deck Cadet License	9/17/2014	9/17/2019	Invalid
13	9	Deck Cadet License	9/18/2019	9/18/2024	Invalid
14	9	Deck Cadet License	9/19/2024	9/19/2028	Valid

Figure 25: Data for Table License

The Vessel Type table defines some realistic standard categories for vessels in the fleet.

Results Explain Describe Saved SQL History

VESSEL_TYPE_ID	VESSEL_TYPE	LENGTH	CAPACITY	TYPICAL_CREW_REQUIRED
CNT	Container Ship	220	18000	14
FEED	Feeder Container Vessel	120	1500	8
ULCV	Ultra Large Container Vessel	400	22000	16
PANM	Panamax Container Ship	200	4500	10
NPAX	Neo-Panamax Container Ship	295	13000	12
RFR	Reefer Container Ship	160	3000	9

6 rows returned in 0.00 seconds Download

k2443219@kingston.ac.uk db_assignment en Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.2

Figure 26: Data for Table Vessel Type

The vessel table stores realistic details for each individual ship, including its name, IMO number, vessel type, speed, dimensions, and build date. The multiplicity of the relationship of the Vessel and Vessel Type tables [One to Many: Multiple Vessels can share the same Vessel Type] is shown below.

Results Explain Describe Saved SQL History

IMO_NO	VESSEL_TYPE_ID	VESSEL_NAME	SPEED	GROSS_TONNAGE	LENGTH	BREADTH	BUILT_DATE
7139540	FEED	Poseidons Path	22.62	131845	123.43	25.55	1/10/2007
7063471	CNT	Emerald Horizon	23.91	155223	227.57	45.97	12/19/2005
7080208	CNT	Venture Eagle	22.45	99238	212.96	47.28	6/27/2021
7095736	CNT	Titan Crest	24	17118	210.73	57.5	9/4/2013
7029633	CNT	Nova Trader	23.15	133989	228.6	52.21	8/29/2019
7089268	CNT	Sapphire Glory	18.3	13519	222.27	32.64	6/29/2006
7116666	FEED	Coral Voyager	23.16	40210	118.61	54.22	11/9/2022
7181565	FEED	Pacific Queen	24.17	119560	113.37	45.36	11/23/2008
7170210	FEED	Wind Rider	21.12	40574	110.27	45.69	9/14/2022
7256120	ULCV	Golden Crest	20.95	85943	408.78	44.39	2/1/2016
7238700	ULCV	Ocean Titan	18.34	109795	397.84	44.3	11/2/2015
7224253	ULCV	Majestic Tide	24.43	97553	409.12	45.3	3/4/2009
7327970	PANM	Blue Phoenix	19.89	77050	203.99	46.13	9/11/2019
7365969	PANM	Global Seaway	23.32	32241	190.65	38.99	1/29/2006
7334395	PANM	Royal Gold	24.75	33021	205.05	52.49	11/18/2020

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 27: Data for Table Vessel

The customer table holds the profiles of individuals or companies booking containers for goods transport.

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_EMAIL	CUSTOMER_TEL_NO	CUST_ADDRESS	CUST_BILL_ADDRESS	CUST_COUNTRY	CREATED_ON
1	Joshua Santos	joshua.santos1@example.com	220-541-4254x981	33959 Chapman Overpass Apt. 416, East Kenneth, NJ 36124	258 Lopez Green Suite 850, Davisburgh, Germany	Germany	10/14/2023
2	Joseph Morgan	joseph.morgan2@example.com	6064955261	185 Eric Plain, Lake Amanda, FL 68618	8720 Leah Port, Henryborough, United Kingdom	United Kingdom	1/18/2024
3	Christopher Vasquez	christopher.vasquez3@example.com	061-854-3470x62534	01200 Brett Station, South Steventon, OR 66703	75006 Howard Canyon, Port Jeremyview, China	China	3/28/2023
4	Christopher Proctor	christopher.proctor4@example.com	+1-887-820-0693x2091	2951 Richard River, West Juanview, NY 88467	30965 Calderon Port, Edwardview, United Kingdom	United Kingdom	2/9/2022
5	Michelle Logan	michelle.logan5@example.com	(803)541-2731	USS Rodriguez, FPO AP 64123	2210 Williams Union, Christopherland, China	China	6/6/2020
6	Sherry Jackson	sherry.jackson6@example.com	001-461-018-0190x739	0512 Ortiz Brook, Wilsonmouth, NM 23642	94776 Sara Way, New Vanessa, United States of America	United States of America	10/13/2024
8	Richard Scott	richard.scott8@example.com	545.532.2788	615 Brown Drive Suite 029, Lake Christopherland, CO 16221	7774 Robert Vista Apt. 261, North Ryanton, Japan	Japan	5/2/2024
9	Sierra Lowery	sierra.lowery9@example.com	(737)065-1472x27170	226 Perez Route Apt. 429, Grantside, NC 27991	39315 Martinez Point Suite 860, South Annstad, United Kingdom	United Kingdom	4/27/2021
12	Monica Martinez	monica.martinez12@example.com	4334360836	939 Owens Brooks, North Claudia, WV 88982	101 Jackson Bridge Apt. 975, North Jamesside, Brazil	Brazil	1/26/2022
13	Kristen Baxter	kristen.baxter13@example.com	(674)422-5860x24480	829 Chapman Loop, New Monica, ND 14958	6272 Schmidt Brooks Apt. 485, West John, United Kingdom	United Kingdom	4/9/2022
14	Paul West	paul.west14@example.com	3304299136	PSC 5714, Box 6777, APO AA 58952	7453 Hurley Isle, Monroeberg, France	France	5/15/2020
15	John Ferguson	john.ferguson15@example.com	+1-294-999-4100	48372 Estrada Manor Apt. 783, West Alexamouth, NJ 61212	4785 Bryant Mission Suite 645, Lake Tina, United States of America	United States of America	6/1/2022
16	Eric Roberts	eric.roberts16@example.com	(541)272-1755x02861	Unit 7580 Box 7212, DPO AP 58281	7970 Stephanie Shoal, Adrianchester, United Kingdom	United Kingdom	1/17/2025
17	Nicole Smith	nicole.smith17@example.com	+1-340-786-1596x7992	05760 Best Underpass Suite 674, South Gary, WI 81325	27934 Jonathan Squares, Higginsfort, Brazil	Brazil	3/13/2022
18	Laura Bass	laura.bass18@example.com	(657)165-3302	202 Jake Gardens, Leehaven, PA 62383	736 Williams Drive Suite 930, Josephburgh, India	India	12/20/2020

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 28: Data for Table Customer

The route table defines planned shipping paths that vessels might take to transport goods.

Results Explain Describe Saved SQL History

ROUTE_ID	ROUTE_NAME	ACTUAL_DISTANCE	ESTIMATED_DURATION
1	Shanghai to Los Angeles	10360	+46 00:00:00.000000
2	Rotterdam to Singapore	9630	+44 00:00:00.000000
3	Hamburg to New York	7500	+31 00:00:00.000000
4	Dubai to Mumbai	1920	+15 00:00:00.000000
5	Tokyo to Sydney	7820	+29 00:00:00.000000
6	Los Angeles to Hong Kong	11600	+39 00:00:00.000000
7	London to Cape Town	9750	+19 00:00:00.000000
8	Singapore to Colombo	2700	+17 00:00:00.000000
9	Busan to Seattle	8770	+15 00:00:00.000000
10	Barcelona to Jeddah	4650	+26 00:00:00.000000

10 rows returned in 0.03 seconds Download

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 29: Data for Table Route

The Route Schedule table represents the planned movement of a vessel along a predefined route, including its assigned vessel, departure, and arrival dates. The multiplicity of the relationship of the Route Schedule and Vessel tables [One to Many: A Vessel can be assigned to many Schedules over time] and the multiplicity of the relationship of the Route Schedule and Route tables [One to Many: A Route can have multiple Schedules] is shown below.

SCHEDULE_ID	ROUTE_ID	VESSEL_ID	ARRIVAL_DATE	DEPARTURE_DATE
19	1	7565969	6/29/2026	5/14/2026
20	1	7224253	8/19/2026	7/4/2026
21	2	7116666	5/15/2023	4/1/2023
22	2	7426658	7/3/2023	5/20/2023
23	2	7170210	8/21/2023	7/8/2023
24	2	7588652	10/9/2023	8/26/2023
25	2	7256120	11/27/2023	10/14/2023
26	2	7139540	1/15/2024	12/2/2023
27	2	7181565	3/4/2024	1/20/2024
28	2	7528216	4/22/2024	3/9/2024
29	2	7029633	6/10/2024	4/27/2024
30	2	7327970	7/29/2024	6/15/2024
31	2	7494686	5/15/2025	4/1/2025
32	2	7063471	7/3/2025	5/20/2025
33	2	7334395	8/21/2025	7/8/2025
34	2	7521273	10/9/2025	8/26/2025
35	2	7080208	11/27/2025	10/14/2025
36	2	7089268	1/15/2026	12/2/2025
37	2	7095756	3/5/2026	1/20/2026
38	2	7218700	4/25/2026	3/10/2026
39	2	7454467	6/11/2026	4/28/2026
40	2	7311051	7/30/2026	6/16/2026
41	3	7574301	5/2/2023	4/1/2023
42	3	7467368	6/7/2023	5/3/2023
43	3	7491894	7/15/2023	6/12/2023
44	3	7565969	8/18/2023	7/18/2023
45	3	7224253	9/25/2023	8/25/2023
46	3	7116666	10/29/2023	9/28/2023
47	3	7426658	12/4/2023	11/3/2023

Figure 30: Data for Table Route Schedule

The voyage table logs realistic actual executions of scheduled journeys, with details such as the vessel used, the schedule it follows, and the real departure and arrival dates. The multiplicity of the relationship of the Voyage and Vessel tables [One to Many: A Vessel can execute many Voyages over time] and the multiplicity of the relationship of the Voyage and Route Schedule tables [One to One: One voyage is executed per Route Schedule] is shown below.

VOYAGE_ID	VESSEL_ID	SCHEDULE_ID	ACTUAL_DEPARTURE_DATE	ACTUAL_ARRIVAL_DATE
4000	7139540	1	4/1/2023	5/20/2023
4001	7181565	2	5/22/2023	7/10/2023
4002	7528216	3	7/12/2023	8/30/2023
4003	7029633	4	9/1/2023	10/19/2023
4004	7327970	5	10/22/2023	12/10/2023
4005	7494686	6	12/12/2023	1/30/2024
4006	7063471	7	2/1/2024	3/20/2024
4007	7334395	8	3/23/2024	5/11/2024
4008	7521273	9	5/13/2024	7/1/2024
4009	7080208	10	7/3/2024	8/21/2024
4010	7089268	11	4/1/2025	5/20/2025
4011	7116666	21	4/1/2023	5/19/2023
4012	7426658	22	5/20/2023	7/7/2023
4013	7170210	23	7/8/2023	8/24/2023
4014	7588652	24	8/26/2023	10/13/2023
4015	7256120	25	10/14/2023	12/1/2023
4016	7139540	26	12/2/2023	1/19/2024
4017	7181565	27	1/20/2024	3/8/2024
4018	7528216	28	3/9/2024	4/26/2024
4019	7029633	29	4/27/2024	6/13/2024
4020	7327970	30	6/15/2024	8/2/2024
4021	7494686	31	4/1/2025	5/19/2025
4022	7574301	41	4/1/2023	4/30/2023

Figure 31: Data for Table Voyage

The Port Table stores realistic key details for each port globally, including contact information, location, and a unique name.

Results Explain Describe Saved SQL History

PORT_ID	PORT_NAME	PORT_ADDRESS	PORT_NUMBER	PORT_EMAIL
1	Alexandria	746 Alexandria Harbor Road	+20 551-1991	alexandria@portauthority.com
2	Algeciras	738 Algeciras Harbor Road	+34 588-3724	algeciras@portauthority.com
3	Antwerp	834 Antwerp Harbor Road	+32 391-4911	antwerp@portauthority.com
4	Auckland	656 Auckland Harbor Road	+64 730-9124	auckland@portauthority.com
5	Barcelona	565 Barcelona Harbor Road	+34 218-4031	barcelona@portauthority.com
6	Bremerhaven	899 Bremerhaven Harbor Road	+49 793-6222	bremerhaven@portauthority.com
7	Brisbane	538 Brisbane Harbor Road	+61 328-3792	brisbane@portauthority.com
8	Busan	956 Busan Harbor Road	+82 429-7337	busan@portauthority.com
9	Casablanca	116 Casablanca Harbor Road	+212 524-2111	casablanca@portauthority.com
10	Charleston	356 Charleston Harbor Road	+1 431-8780	charleston@portauthority.com
11	Chennai	997 Chennai Harbor Road	+91 839-9762	chennai@portauthority.com
12	Colombo	522 Colombo Harbor Road	+94 861-8635	colombo@portauthority.com
13	Dakar	279 Dakar Harbor Road	+221 573-5544	dakar@portauthority.com
14	Dammam	334 Dammam Harbor Road	+966 897-3039	dammam@portauthority.com
15	Genoa	738 Genoa Harbor Road	+39 732-8541	genoa@portauthority.com
16	Gioia Tauro	183 Gioia Tauro Harbor Road	+39 686-2349	gioiatauro@portauthority.com
17	Haifa	779 Haifa Harbor Road	+972 330-8907	haifa@portauthority.com
18	Hamburg	777 Hamburg Harbor Road	+49 468-3722	hamburg@portauthority.com

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 32: Data for Table Port

The Segment Table holds realistic information of a direct, traversable leg of a route that connects two ports. The multiplicity of the relationship of the Segment and Port tables [One to Many in both directions: One port can be the departure point for many segments; one port can also be the arrival point for many segments] is shown below.

Results Explain Describe Saved SQL History

SEG_ID	DEP_PORT	ARRIVAL_PORT	DISTANCE	ESTIMATED_TIME
1	44	54	1915.39	+10 00:00:00.000000
2	54	22	455.66	+10 00:00:00.000000
3	22	8	3248.33	+09 00:00:00.000000
4	8	51	3062.82	+07 00:00:00.000000
5	39	3	4087.24	+06 00:00:00.000000
6	3	26	501.6	+01 00:00:00.000000
7	26	2	4065.24	+05 00:00:00.000000
8	41	12	542.02	+09 00:00:00.000000
9	12	45	3505.98	+10 00:00:00.000000
10	18	39	3629.77	+03 00:00:00.000000
11	26	46	2362.27	+08 00:00:00.000000
12	21	23	1365.71	+03 00:00:00.000000
13	34	32	721.68	+04 00:00:00.000000
14	37	25	3639.68	+04 00:00:00.000000
15	25	22	4033.69	+01 00:00:00.000000
16	22	19	3864.19	+09 00:00:00.000000

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 33: Data for Table Segment

Routes are made of multiple Segments and Segments could be present in many Routes [Many-to Many]. To address this, Route Seg Assignment, a bridging table is formed to store each combination of a Route and a Segment. The multiplicity of the relationship of the Route and Route Seg Assignment Tables [One to Many: A Route can be linked to many Segments] and the multiplicity of the relationship of the Segments and Route Seg Assignment Tables [One to Many: A Segment can be reused across multiple Routes] is shown below.

ROUTE_ID	SEGMENT_ID	SEGMENT_ORDER
1	48	5
1	30	6
1	31	7
2	5	1
2	6	2
2	7	3
2	32	4
2	51	5
2	34	6
2	8	7
2	9	8
3	10	1
3	5	2
3	6	3
3	11	4
3	35	5
3	36	6
4	12	1
4	37	2
4	38	3
4	13	4
5	48	1
5	40	2

Figure 34: Data for Table Route Seg Assignment

A Route Schedule involves multiple Segments, and a Segment can appear in multiple Route Schedules [Many-to-Many]. To address this, Segment Schedule, a bridging table was formed to store each combination of Route Schedule and Segment. The multiplicity of the relationship of the Route Schedule and Segment Schedule Tables [One to Many: A Route Schedule can involve many Segments] and the multiplicity of the relationship of the Segments and Segment Schedule Tables [One to Many: A Segment can be reused across multiple Route Schedules] is shown below.

SCHEDULE_ID	SEGMENT_ID	PLANNED_DEPARTURE_DATE	PLANNED_ARRIVAL_DATE	ACTUAL_DEPART_DATE	ACTUAL_ARRIVAL_DATE
107	22	1/20/2024	1/26/2024	1/20/2024	1/25/2024
108	22	3/4/2024	3/10/2024	3/4/2024	3/10/2024
109	22	4/17/2024	4/23/2024	4/17/2024	4/23/2024
110	22	5/31/2024	6/6/2024	5/31/2024	6/6/2024
111	22	5/1/2025	5/7/2025	5/1/2025	5/6/2025
101	16	5/7/2023	5/13/2023	5/7/2023	5/13/2023
102	16	6/20/2023	6/26/2023	6/20/2023	6/25/2023
103	16	8/3/2023	8/9/2023	8/3/2023	8/9/2023
104	16	9/16/2023	9/22/2023	9/16/2023	9/22/2023
105	16	10/30/2023	11/5/2023	10/30/2023	11/6/2023
106	16	12/13/2023	12/19/2023	12/13/2023	12/18/2023
107	16	1/26/2024	2/1/2024	1/26/2024	2/1/2024
108	16	3/10/2024	3/16/2024	3/11/2024	3/15/2024
109	16	4/23/2024	4/29/2024	4/23/2024	4/29/2024
110	16	6/6/2024	6/12/2024	6/7/2024	6/12/2024
111	16	5/7/2025	5/13/2025	5/7/2025	5/12/2025
121	23	4/1/2023	4/5/2023	4/1/2023	4/4/2023
122	23	4/25/2023	4/29/2023	4/25/2023	4/29/2023
123	23	5/19/2023	5/23/2023	5/19/2023	5/22/2023
124	23	6/12/2023	6/16/2023	6/12/2023	6/15/2023
125	23	7/6/2023	7/10/2023	7/6/2023	7/10/2023
126	23	7/30/2023	8/3/2023	7/30/2023	8/3/2023

Figure 35: Data for Table Segment Schedule

The Crew Assignment Table captures the allocation of crew members to the planned route schedules. Not all employees would be Crew Members. The multiplicity of the relationship of the Crew Assignment and Route Schedule Tables [One to Many: A Route Schedule can be assigned to many Crew Members over time] and the multiplicity of the relationship of the Crew Assignment and Employee Tables [One to Many: An Employee can be given multiple Crew Assignments over time] is shown below.

CREW_ID	EMP_ID	SCHEDULE_ID	ASSIGNED_ON	RELEASED_ON	STATUS
6000	18	1	3/22/2023	5/18/2023	Completed
6001	9	1	3/26/2023	5/18/2023	Completed
6002	31	1	3/28/2023	5/18/2023	Completed
6003	41	1	3/28/2023	5/22/2023	Completed
6004	1	1	3/23/2023	5/19/2023	Completed
6005	51	1	3/29/2023	5/18/2023	Completed
6006	56	1	3/29/2023	5/18/2023	Completed
6007	34	1	3/25/2023	5/19/2023	Completed
6008	18	41	3/25/2023	5/3/2023	Completed
6009	9	41	3/24/2023	5/7/2023	Completed
6010	31	41	3/28/2023	5/4/2023	Completed
6011	41	41	3/28/2023	5/6/2023	Completed
6012	1	41	3/23/2023	5/5/2023	Completed
6013	51	41	3/23/2023	5/3/2023	Completed
6014	56	41	3/25/2023	5/4/2023	Completed
6015	34	41	3/22/2023	5/5/2023	Completed
6016	56	41	3/22/2023	5/4/2023	Completed
6017	18	81	3/27/2023	5/1/2023	Completed
6018	9	81	3/29/2023	5/5/2023	Completed
6019	31	81	3/25/2023	5/5/2023	Completed

Figure 36: Data for Table Crew Assignment

The Container Type table defines realistic physical characteristics of shipping containers, including their length, width, height, and weight capacity.

CONTAINER_TYPE_ID	CONTAINER_TYPE	LENGTH	WIDTH	HEIGHT	WEIGHT
1	Standard 20ft	20	8	8.5	2300
2	Standard 40ft	40	8	8.5	3800
3	High Cube 40ft	40	8	9.5	3900
4	Refrigerated 20ft	20	8	8.5	3000
5	Refrigerated 40ft	40	8	8.5	4500
6	Open Top 20ft	20	8	8.5	2200
7	Open Top 40ft	40	8	8.5	3600
8	Flat Rack 20ft	20	8	8.5	2400
9	Flat Rack 40ft	40	8	8.5	4000
10	Tank Container	20	8	8.5	3500

10 rows returned in 0.01 seconds

Figure 37: Data for Table Container Type

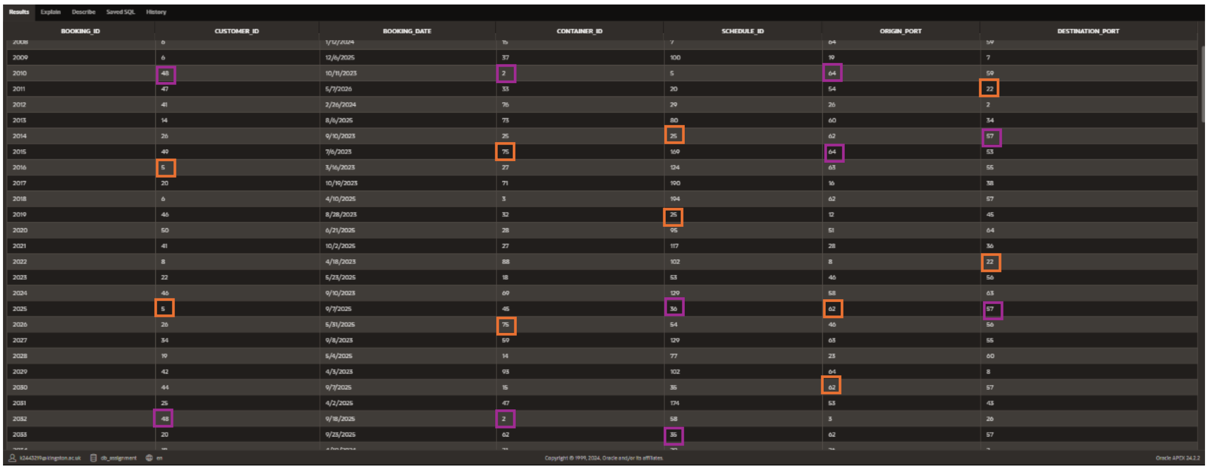
The Container table stores each physical container used by the shipping company which are reused over time for different customer bookings. The multiplicity of the relationship of the Container and Container Type tables [One to Many: Multiple Containers can share the same Container Type] is shown below.

CONTAINER_ID	CONTAINER_TYPE_ID	CONTAINER_NAME
1	5	REEF40-001
2	1	STD20-002
3	10	TANK-003
4	2	STD40-004
5	2	STD40-005
6	10	TANK-006
7	4	REEF20-007
8	1	STD20-008
9	1	STD20-009
10	5	REEF40-010
11	2	STD40-011

Figure 38: Data for Table Container

The Booking Table represents a customer's request to transport goods using a specific container, from one port to another, on a scheduled route. The multiplicity of the relationship of the Booking and Customer Tables [One to Many: A Customer can have multiple Bookings], multiplicity of the relationship of the Booking and Container Tables [One to One: A Booking must involve one Container], multiplicity of the relationship of the Booking and Route Schedule Tables [One to Many: A Route Schedule can involve multiple

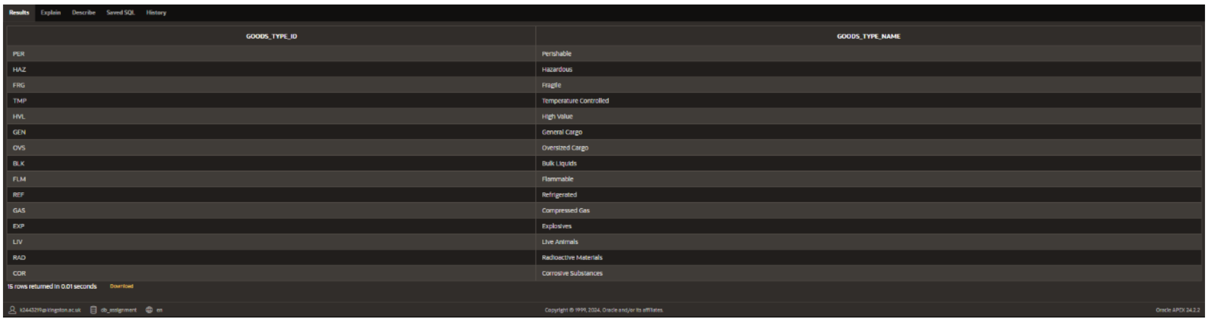
Bookings] and the multiplicity of the relationship of the Booking and Port Tables [One to Many: One port can be the origin for many bookings and One port can also be the destination for many bookings] is shown below.



BOOKING_ID	CUSTOMER_ID	BOOKING_DATE	CONTAINER_ID	SCHEDULE_ID	ORIGIN_PORT	DESTINATION_PORT
2009	6	12/6/2025	37	100	19	7
2010	48	10/11/2023	2	5	64	59
2011	47	5/7/2026	33	20	54	22
2012	41	2/26/2024	76	29	26	2
2013	14	8/6/2025	73	80	60	34
2014	26	9/10/2023	25	25	62	57
2015	49	7/6/2023	75	169	64	53
2016	5	3/16/2023	27	124	63	55
2017	20	10/19/2023	71	190	16	38
2018	6	4/10/2025	3	194	62	57
2019	46	8/28/2023	32	25	12	45
2020	50	6/21/2025	28	95	51	64
2021	41	10/2/2025	27	117	28	36
2022	8	4/18/2023	88	102	8	22
2023	22	5/23/2025	18	53	46	56
2024	46	9/10/2023	69	129	58	63
2025	5	9/7/2025	45	36	62	57
2026	26	5/31/2025	75	54	46	56
2027	34	9/8/2023	59	129	63	55
2028	19	5/4/2025	14	77	23	60
2029	42	4/3/2023	93	102	64	8
2030	44	9/7/2025	15	35	62	57
2031	25	4/2/2025	47	174	53	43
2032	48	9/18/2025	2	58	3	26
2033	20	9/23/2025	62	35	62	57

Figure 39: Data for Table Booking

The Goods Type Table represents realistic classifications of the shipments.



GOODS_TYPE_ID	GOODS_TYPE_NAME
PER	Perishable
HAZ	Hazardous
FRG	Fragile
TMP	Temperature Controlled
HVL	High Value
GEN	General Cargo
OVS	Oversized Cargo
BLK	Bulk Liquids
FLM	Flammable
REF	Refrigerated
GAS	Compressed Gas
EXP	Explosives
LIV	Live Animals
RAD	Radioactive Materials
COR	Corrosive Substances

15 rows returned in 0.01 seconds

Figure 40: Data for Table Goods Type

The goods table records realistic details about the actual cargo being transported, including quantity, weight, volume, type, and loading and unloading dates. The multiplicity of the relationship of the Goods and Goods Type tables [One to Many: Multiple Goods can share the same Goods Type] is shown below.

GOODS_ID	CONTAINER_ID	GOODS_TYPE_ID	QUANTITY	WEIGHT	VOLUME	LOADED_ON	UNLOADED_ON
3000	27	REF	173	5776.56	28.89	-	-
3001	79	EXP	37	8892.83	18.5	7/16/2023	8/19/2023
3002	98	FRG	86	2179.07	12.15	10/13/2023	11/29/2023
3004	61	LIV	171	5447.15	28.37	-	-
3007	63	PER	167	1516.98	37.88	5/10/2024	7/1/2024
3008	15	FRG	77	5245.4	58.18	-	-
3009	37	EXP	166	5262.98	14.43	-	-
3010	2	GEN	30	6799.38	62.29	-	-
3011	33	GAS	81	5455.9	36.36	-	-
3012	76	EXP	109	3289.78	42.91	4/26/2024	6/11/2024
3013	73	BLK	95	5476.37	15.62	-	-
3014	25	LIV	15	5870.79	62.28	10/11/2023	11/28/2023
3016	27	GEN	27	2724.99	16.38	-	-
3018	3	HVL	63	6895.64	49.96	-	-
3019	32	FLM	115	6148.51	44.06	10/12/2023	11/30/2023
3020	28	COR	194	3450.2	38.84	-	-
3022	88	EXP	171	616.22	5.35	-	-
3023	18	COR	15	2513.12	33.75	-	-
3025	45	TMP	82	8446.28	30.9	-	-
3026	75	GEN	11	4707.71	62.96	-	-
3027	59	TMP	95	4692.56	55.07	10/7/2023	10/30/2023

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 41: Data for Table Goods

The Container Movement table captures the real-time tracking of each container as it passes through various ports during a voyage. The multiplicity of the relationship between Container Movement and Container [One to Many: A Container can have multiple Container Movements], the multiplicity of the relationship between Container Movement and Port [One to Many: A Port can have multiple Container Movements], the multiplicity of the relationship between Container Movement and Voyage [One to Many: A Voyage can be linked to multiple Container Movements], and the multiplicity of the relationship between Container Movement and Booking [One to Many: One Booking is linked to multiple Container Movements] is shown below.

MOVEMENT_ID	CONTAINER_ID	PORT_ID	VOYAGE_ID	BOOKING_ID	MOVEMENT_TYPE	TIMESTAMP
5000	27	44	4001	2000	Loaded	06-FEB-25 01.17.06.000000 PM
5001	27	51	4001	2000	In Transit	02-APR-25 01.17.06.000000 PM
5002	27	28	4001	2000	Unloaded	11-APR-25 01.17.06.000000 PM
5003	49	44	4001	2107	Loaded	28-MAR-25 01.17.06.000000 PM
5004	49	51	4001	2107	In Transit	09-MAR-25 01.17.06.000000 PM
5005	49	28	4001	2107	Delivered	06-APR-25 01.17.06.000000 PM
5006	79	18	4025	2001	Loaded	03-MAR-25 01.17.06.000000 PM
5007	79	46	4025	2001	In Transit	01-APR-25 01.17.06.000000 PM
5008	79	33	4025	2001	Unloaded	27-MAR-25 01.17.06.000000 PM
5009	98	39	4015	2002	Loaded	15-JAN-25 01.17.06.000000 PM
5010	98	57	4015	2002	In Transit	12-MAR-25 01.17.06.000000 PM
5011	98	45	4015	2002	Delivered	23-MAR-25 01.17.06.000000 PM
5012	25	39	4015	2014	Loaded	19-JAN-25 01.17.06.000000 PM
5013	25	57	4015	2014	In Transit	22-FEB-25 01.17.06.000000 PM
5014	25	45	4015	2014	Delivered	24-MAR-25 01.17.06.000000 PM
5015	32	39	4015	2019	Loaded	16-JAN-25 01.17.06.000000 PM
5016	32	57	4015	2019	In Transit	04-APR-25 01.17.06.000000 PM
5017	32	45	4015	2019	Delivered	12-APR-25 01.17.06.000000 PM
5018	1	39	4015	2076	Loaded	19-MAR-25 01.17.06.000000 PM
5019	1	57	4015	2076	In Transit	16-MAR-25 01.17.06.000000 PM
5020	1	45	4015	2076	Unloaded	15-MAR-25 01.17.06.000000 PM

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 42: Data for Table Container Movements

Chapter 4 – SQL Queries

Summary of Voyages for Operational Monitoring

This view summarizes each voyage with relevant schedule, route, and vessel data, while also showing its operational status and total containers delivered. This helps the operations team quickly monitor voyage performance and identify active or completed trips.

INITCAP(): to format vessel and route to Title Case

CASE: to classify each voyage as either 'Completed' or 'In Progress', based on whether actual arrival date is present.

```
CREATE OR REPLACE VIEW v_voyage_summary AS
SELECT
    v.voyage_id AS voyage_code,
    INITCAP(vs.vessel_name) AS vessel,
    rs.schedule_id AS schedule,
    INITCAP(r.route_name) AS route,
    v.actual_departure_date AS departed_on,
    v.actual_arrival_date AS arrived_on,
    CASE
        WHEN v.actual_arrival_date IS NULL THEN 'In Progress'
        ELSE 'Completed'
    END AS voyage_status,
    COUNT(DISTINCT cm.container_id) AS containers_delivered
FROM voyage v
JOIN route_schedule rs ON v.schedule_id = rs.schedule_id
JOIN vessel vs ON v.vessel_id = vs.imo_no
JOIN route r ON rs.route_id = r.route_id
LEFT JOIN container_movement cm
    ON v.voyage_id = cm.voyage_id AND cm.movement_type = 'Unloaded'
GROUP BY v.voyage_id, vs.vessel_name, rs.schedule_id, r.route_name, v.actual_departure_date, v.actual_arrival_date;
```

Results Explain Describe Saved SQL History

View created.

0.06 seconds

k2443219@kingston.ac.uk db_assignment en

Figure 43: View for Voyage Summary

Results:

```
SELECT * FROM v_voyage_summary;
```

Results Explain Describe Saved SQL History

VOYAGE_CODE	VESSEL	SCHEDULE	ROUTE	DEPARTED_ON	ARRIVED_ON	VOYAGE_STATUS	CONTAINERS_DELIVERED
4033	Sapphire Glory	61	Dubai To Mumbai	4/1/2023	4/17/2023	Completed	1
4057	Celestial Freighter	103	Los Angeles To Hong Kong	6/28/2023	8/9/2023	Completed	2
4094	Leviathan Sky	167	Busan To Seattle	7/30/2023	8/15/2023	Completed	2
4109	Neptune Voyager	191	Barcelona To Jeddah	4/1/2025	4/26/2025	Completed	2
4105	Titan Crest	187	Barcelona To Jeddah	10/4/2023	10/29/2023	Completed	0
4101	Royal Gold	183	Barcelona To Jeddah	6/2/2023	6/26/2023	Completed	1
4003	Nova Trader	4	Shanghai To Los Angeles	9/1/2023	10/19/2023	Completed	1
4069	Aegean Pearl	124	London To Cape Town	6/12/2023	7/2/2023	Completed	1
4082	Coral Voyager	146	Singapore To Colombo	7/20/2023	8/7/2023	Completed	0
4108	Aurora Spirit	190	Barcelona To Jeddah	1/5/2024	1/29/2024	Completed	1
4023	Leviathan Sky	42	Hamburg To New York	5/7/2023	6/6/2023	Completed	0

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 44: Results for the View

Validating Crew Assignment

This Trigger ensures that during data entry, no more than the required number of crew members are assigned to a route schedule based on the vessel's crew requirement. It also optionally warns if fewer than expected crew have been assigned so far when inserting new entries to the table.

```
CREATE OR REPLACE TRIGGER trg_check_minimum_crew
BEFORE INSERT ON crew_assignment
FOR EACH ROW
DECLARE
    v_required NUMBER;
    v_existing NUMBER;
BEGIN
    SELECT vt.typical_crew_required INTO v_required
    FROM route_schedule rs
    JOIN vessel v ON rs.vessel_id = v.imo_no
    JOIN vessel_type vt ON v.vessel_type_id = vt.vessel_type_id
    WHERE rs.schedule_id = :NEW.schedule_id;

    SELECT COUNT(*) INTO v_existing
    FROM crew_assignment
    WHERE schedule_id = :NEW.schedule_id;

    IF v_existing >= v_required THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot exceed typical crew requirement (' || v_required || ')');
    END IF;

    IF v_existing BETWEEN 0 AND v_required - 2 THEN
        DBMS_OUTPUT.PUT_LINE('Less than expected crew assigned.');
    END IF;
END;
/
```

Results | Explain | Describe | Saved SQL | History

```
Trigger created.
```

0.08 seconds

k2443219@kingston.ac.uk | db_assignment | en

Figure 45: Trigger to validate the Crew Assignment

Result:

```
INSERT INTO crew_assignment (crew_id, emp_id, schedule_id, assigned_on, released_on, status)
VALUES (8008, 1, 1, TO_DATE('2024-04-01', 'YYYY-MM-DD'), NULL, 'Assigned');
```

Results | Explain | Describe | Saved SQL | History

```
ORA-20001: Cannot exceed typical crew requirement (8)
ORA-06512: at "WKSP_DBASSIGNMENT.TRG_CHECK_MINIMUM_CREW", line 16
ORA-04088: error during execution of trigger
'WKSP_DBASSIGNMENT.TRG_CHECK_MINIMUM_CREW'


1. INSERT INTO crew_assignment (crew_id, emp_id, schedule_id, assigned_on,
released_on, status)
2. VALUES (8008, 1, 1, TO_DATE('2024-04-01', 'YYYY-MM-DD'), NULL, 'Assigned');
```

0.04 seconds

Figure 46: Result for the Trigger

Top Ports by Route Coverage and Container Traffic

This stored procedure identifies ports that are involved in multiple routes and reports detailed traffic statistics for each, including container handling volume, number of distinct routes, and movement history. It supports logistics performance monitoring and can guide operational decisions such as identifying high-traffic hubs or bottlenecks.

INITCAP(): to format the port name in Title Case.

COUNT(): to calculate distinct total routes, containers handled and total movements.

MIN(): to obtain the first movement of the container that was recorded.

MAX(): to obtain the last movement of the container that was recorded.

GROUP BY: to group the data by port name.

HAVING: to show only those ports involved in more than one route.

```sql

CREATE OR REPLACE PROCEDURE sp_top_ports_by_route_traffic IS
    CURSOR port_route_summary_cur IS
        SELECT
            INITCAP(p.port_name) AS port,
            COUNT(DISTINCT rsa.route_id) AS total_routes,
            COUNT(DISTINCT cm.container_id) AS containers_handled,
            COUNT(cm.movement_id) AS total_movements,
            MIN(cm.timestamp) AS first_movement,
            MAX(cm.timestamp) AS last_movement
        FROM port p
        JOIN segment s ON p.port_id IN (s.dep_port, s.arrival_port)
        JOIN route_seg_assignment rsa ON rsa.segment_id = s.seg_id
        JOIN route r ON rsa.route_id = r.route_id
        LEFT JOIN container_movement cm ON cm.port_id = p.port_id
        GROUP BY p.port_name
        HAVING COUNT(DISTINCT rsa.route_id) > 1
        ORDER BY total_routes DESC, total_movements DESC;
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Top Ports by Route Coverage and Traffic ---');
    FOR rec IN port_route_summary_cur LOOP
        DBMS_OUTPUT.PUT_LINE('Port: ' || rec.port);
        DBMS_OUTPUT.PUT_LINE('  Routes Involved: ' || rec.total_routes);
        DBMS_OUTPUT.PUT_LINE('  Containers Handled: ' || rec.containers_handled);
        DBMS_OUTPUT.PUT_LINE('  Total Movements: ' || rec.total_movements);
        DBMS_OUTPUT.PUT_LINE('  First Movement: ' || rec.first_movement);
        DBMS_OUTPUT.PUT_LINE('  Last Movement: ' || rec.last_movement);
        DBMS_OUTPUT.PUT_LINE('---------------------------------------------');
    END LOOP;
END;
/
```

Results Explain Describe Saved SQL History

```
Procedure created.

0.07 seconds
```

k2443219@kingston.ac.uk db_assignment en

Figure 47: Stored Procedure to report detailed traffic statistics of top ports

Results:

```
BEGIN
    sp_top_ports_by_route_traffic;
END;
/
```

Results Explain Describe Saved SQL History

```
--- Top Ports by Route Coverage and Traffic ---
Port: Tokyo
  Routes Involved: 4
  Containers Handled: 17
  Total Movements: 119
  First Movement: 26-JAN-25 01.17.06.000000 PM
  Last Movement: 02-APR-25 01.17.06.000000 PM
---------------------------------------------
Port: Yokohama
  Routes Involved: 4
  Containers Handled: 11
  Total Movements: 88
  First Movement: 23-FEB-25 01.17.06.000000 PM
  Last Movement: 07-APR-25 01.17.06.000000 PM
---------------------------------------------
Port: Busan
  Routes Involved: 3
  Containers Handled: 10
  Total Movements: 50
  First Movement: 13-JAN-25 01.17.06.000000 PM
  Last Movement: 01-APR-25 01.17.06.000000 PM
---------------------------------------------
Port: Kaohsiung
  Routes Involved: 3
  Containers Handled: 8
  Total Movements: 48
  First Movement: 22-FEB-25 01.17.06.000000 PM
  Last Movement: 06-APR-25 01.17.06.000000 PM
---------------------------------------------
Port: Le Havre
  Routes Involved: 3
  Containers Handled: 0
```

k2443219@kingston.ac.uk db_assignment en

Figure 48: Result for the Stored Procedure

Container Type Weight Utilization Efficiency

This query shows how efficiently different types of containers are being utilized, based on the average goods weight and the container's maximum weight capacity helping logistics managers detect patterns like underutilized containers, or potential overloads. It uses,

AVG(): to calculate the average weight of goods loaded into containers of a given type.

COUNT(): to calculate the total number of goods entries per container type and how many unique containers were used.

ROUND(): to present the average goods weight and average utilisation percentage to 2 decimal places.

CASE: to classify the container usage efficiency into 3 categories.

```sql
SELECT
    ct.container_type_id,
    ct.container_type,
    ROUND(AVG(g.weight), 2) AS avg_goods_weight_kg,
    ct.weight AS container_max_weight_kg,
    COUNT(g.goods_id) AS total_goods_entries,
    COUNT(DISTINCT c.container_id) AS distinct_containers_used,
    ROUND((AVG(g.weight) / ct.weight) * 100, 2) AS avg_utilization_percent,
    CASE
        WHEN (AVG(g.weight) / ct.weight) * 100 > 100 THEN 'Overloaded'
        WHEN (AVG(g.weight) / ct.weight) * 100 BETWEEN 80 AND 100 THEN 'Efficient'
        ELSE 'Underutilized'
    END AS usage_status
FROM goods g
JOIN container c ON g.container_id = c.container_id
JOIN container_type ct ON c.container_type_id = ct.container_type_id
GROUP BY ct.container_type_id, ct.container_type, ct.weight
ORDER BY avg_utilization_percent DESC;
```

Figure 49: Query that shows the Container Weight Utilization

Result:

Results Explain Describe Saved SQL History

CONTAINER_TYPE_ID	CONTAINER_TYPE	AVG_GOODS_WEIGHT_KG	CONTAINER_MAX_WEIGHT_KG	TOTAL_GOODS_ENTRIES	DISTINCT_CONTAINERS_USED	AVG_UTILIZATION_PERCENT	USAGE_STATUS
8	Flat Rack 20ft	7170.04	2400	5	3	298.75	Overloaded
1	Standard 20ft	4871.62	2300	33	21	211.81	Overloaded
6	Open Top 20ft	4093.96	2200	3	1	186.09	Overloaded
4	Refrigerated 20ft	4980.67	3000	10	7	166.02	Overloaded
2	Standard 40ft	5547.06	3800	35	23	145.98	Overloaded
3	High Cube 40ft	5323.44	3900	10	6	136.5	Overloaded
10	Tank Container	4699.85	3500	20	13	134.28	Overloaded
7	Open Top 40ft	4579.99	3600	10	7	127.22	Overloaded
5	Refrigerated 40ft	4786.81	4500	15	10	106.37	Overloaded
9	Flat Rack 40ft	4086.45	4000	9	5	102.16	Overloaded

10 rows returned in 0.06 seconds Download

k2443219@kingston.ac.uk db_assignment en

Oracle APEX 24.2.2

Figure 50: Results for the Query

Top Customers who transport Hazardous Goods by Weight Shipped

This query shows the top customers who frequently ship hazardous and high-risk cargo types according to the weight of goods that has been shipped and the volume of bookings.

INITCAP(): to format customer names in Title Case

COUNT(): to count goods and unique bookings

SUM(): to obtain the total weight of shipments and the total volume.

AVG(): to obtain the average weight of the shipments.

ROUND(): to present the average weight and total volume of shipment with a precision of 2 decimal places.

MIN(): to obtain the lightest shipment

MAX(): to obtain the heaviest shipment

```sql
SELECT
    INITCAP(c.customer_name) AS customer,
    COUNT(DISTINCT b.booking_id) AS total_bookings,
    COUNT(g.goods_id) AS total_goods,
    SUM(g.weight) AS total_weight_kg,
    ROUND(AVG(g.weight), 2) AS avg_weight_kg,
    MIN(g.weight) AS lightest_shipment,
    MAX(g.weight) AS heaviest_shipment,
    ROUND(SUM(g.volume), 2) AS total_volume_m3
FROM customer c
JOIN booking b ON c.customer_id = b.customer_id
JOIN goods g ON b.container_id = g.container_id
JOIN goods_type gt ON g.goods_type_id = gt.goods_type_id
WHERE g.weight BETWEEN 500 AND 15000
  AND LOWER(gt.goods_type_name) IN (
    'bulk liquids', 'compressed gas', 'flammable'
  )
GROUP BY c.customer_id, c.customer_name
HAVING COUNT(b.booking_id) > 1 AND SUM(g.weight) > 10000
ORDER BY total_weight_kg DESC;
```

Figure 51: Query that shows Top Customers who transport Hazardous Goods by weight shipped and volume of bookings

Result:

Results Explain Describe Saved SQL History

CUSTOMER	TOTAL_BOOKINGS	TOTAL_GOODS	TOTAL_WEIGHT_KG	AVG_WEIGHT_KG	LIGHTEST_SHIPMENT	HEAVIEST_SHIPMENT	TOTAL_VOLUME_M3
Andres Sanders	4	4	21919.53	5479.88	3364.33	9386.97	171.8
Bryan Griffin	2	3	12863.51	4287.84	1945.46	5608.67	83.6
Mandy Lewis	2	2	12374.67	6187.34	3389.58	8985.09	141.51
Samantha Johnson	2	2	11968.27	5984.14	5967.86	6000.41	119.63
Christopher Vasquez	2	2	11444.23	5722.12	5476.37	5967.86	71.24
Brian Moore	2	2	10373.12	5186.56	4594.47	5778.65	107.98

6 rows returned in 0.08 seconds Download

k2443219@kingston.ac.uk db_assignment en Oracle APEX 24.2.2

Figure 52: Results for query

Top 3 most frequently used containers for each Customers in the past year, along with usage frequency, type, average weight, and usage classification

The query shows each customer's container usage over the last 12 months, highlighting their top 3 most frequently used containers, the average cargo weight, and a categorized usage label (Heavy, Moderate, or Light Use). It has used,

With Clause: to define a temporary result set (CustomerContainerUsage) that can be referenced in the main query.

INITCAP: to format names in Title Case.

AVG(): to calculate the average weight of the shipments.

COUNT(): to calculate how often a container is used.

ROUND(): to roundup the average weight of shipments to 2 decimal places.

RANK(): to rank the containers within each customer based on their usage count.

CASE: to classify each container into a usage category.

```sql
WITH CustomerContainerUsage AS (
    SELECT
        c.customer_id,
        INITCAP(c.customer_name) AS customer,
        con.container_id,
        con.container_name,
        ct.container_type,
        COUNT(*) AS usage_count,
        ROUND(AVG(g.weight), 2) AS avg_weight,
        RANK() OVER (
            PARTITION BY c.customer_id
            ORDER BY COUNT(*) DESC
        ) AS usage_rank
    FROM booking b
    JOIN customer c ON b.customer_id = c.customer_id
    JOIN container con ON b.container_id = con.container_id
    JOIN container_type ct ON con.container_type_id = ct.container_type_id
    LEFT JOIN goods g ON con.container_id = g.container_id
    WHERE b.booking_date >= ADD_MONTHS(SYSDATE, -12)
    GROUP BY c.customer_id, c.customer_name, con.container_id, con.container_name, ct.container_type
)
SELECT
    customer,
    container_id,
    container_name,
    container_type,
    usage_count,
    avg_weight,
    CASE
        WHEN usage_count >= 10 THEN 'Heavy Use'
        WHEN usage_count BETWEEN 5 AND 9 THEN 'Moderate Use'
        ELSE 'Light Use'
    END AS usage_category
FROM CustomerContainerUsage
WHERE usage_rank <= 3
ORDER BY customer, usage_count DESC;
```

Figure 53: Query with CTE to show the most frequently used containers

Result:

CUSTOMER	CONTAINER_ID	CONTAINER_NAME	CONTAINER_TYPE	USAGE_COUNT	AVG_WEIGHT	USAGE_CATEGORY
Andres Sanders	75	STD20-075	Standard 20ft	2	5243.18	Light Use
Anita Anderson	2	STD20-002	Standard 20ft	2	4162.89	Light Use
Antonio Peck	6	TANK-006	Tank Container	3	5773.5	Light Use
Brian Moore	94	STD40-094	Standard 40ft	1	4594.47	Light Use
Bryan Griffin	14	STD20-014	Standard 20ft	2	4880.1	Light Use
Bryan Griffin	89	STD20-089	Standard 20ft	1	8704.55	Light Use
Bryan Griffin	7	REEF20-007	Refrigerated 20ft	1	6840.44	Light Use
Charles Rose	80	REEF20-080	Refrigerated 20ft	1	6736.84	Light Use
Cheryl Pratt	27	FLAT40-027	Flat Rack 40ft	3	4347.41	Light Use
Cheryl Pratt	67	HC40-067	High Cube 40ft	2	4501.87	Light Use
Cheryl Pratt	18	OT40-018	Open Top 40ft	2	2439.46	Light Use
Christopher Johnson	6	TANK-006	Tank Container	3	5773.5	Light Use
Christopher Vasquez	97	REEF40-097	Refrigerated 40ft	3	4152.06	Light Use
Daniel Watson	18	OT40-018	Open Top 40ft	2	2439.46	Light Use
Eric Roberts	39	FLAT40-039	Flat Rack 40ft	2	6658.75	Light Use
Hannah White	1	REEF40-001	Refrigerated 40ft	3	5489.76	Light Use
Heather Schneider	95	STD40-095	Standard 40ft	4	6689.95	Light Use
Heather Schneider	47	REEF20-047	Refrigerated 20ft	2	2552.98	Light Use
Jennifer Johnson	14	STD20-014	Standard 20ft	2	4880.1	Light Use
Jennifer Johnson	13	HC40-013	High Cube 40ft	1	6623.29	Light Use
Jesse Fowler	100	STD40-100	Standard 40ft	1	5454.19	Light Use
John Ferguson	97	REEF40-097	Refrigerated 40ft	3	4152.06	Light Use

db_assignment en

Oracle APEX 24.2.2

Figure 54: Results for Query

Chapter 5 - Conclusion

This project offered a comprehensive and hands-on experience in designing and implementing a relational database system for a global shipping company. It covered all stages from entity-relationship modelling and normalization to data entry and writing advanced SQL queries.

One of the most challenging aspects was the design of the ER diagram. Picturing and mapping out relationships for some entities was challenging. Ensuring that each relationship was realistic, scalable, and logically sound required multiple iterations and critical thinking.

The part of the project that stood out most positively for me was the SQL query implementation. Having worked with SQL professionally for over four years, I found this part of the assignment both familiar and rewarding. I was able to apply real-world techniques such as subqueries, aggregate functions, window functions, and CASE statements to generate meaningful insights and answer practical business questions.

Self-Evaluation

This project pushed me to not only apply my technical knowledge, but also to critically evaluate system requirements and design accordingly. I was able to leverage my SQL experience effectively, especially in the querying section. However, I also had to challenge myself when working with data integrity constraints, complex relationships, and realistic data entry areas which required more attention than anticipated.

What Did I Think of the Assignment?

I found the assignment to be well-structured and thoughtfully challenging. It mirrored the kind of work that happens in real-world database design and gave me the chance to step into the role of a database architect. It was about designing something that actually works and makes sense in a practical context, rather than just theory.

What Went Well

- The SQL queries were the strongest part of this coursework. I was able to design queries that not only worked but were also meaningful from a business operations standpoint.
- The structure of the database aligned well with the system assumptions.
- I was able to demonstrate multiplicity clearly in the data and ensure referential integrity across all relationships.

What Didn't Go So Well

- The ER diagram took several attempts to get right. Some relationships were more complex than expected and required reworking.
- Data entry and testing were time-consuming. Matching foreign keys, validating dates, and ensuring logic (like employment dates before crew assignment) required careful checking and occasional correction.
- Implementing and testing triggers needed extra effort to make sure the logic executed as intended under different data conditions.

What Would I Do Differently

- Extend cargo tracking in the container movement table by adding GPS co-ordinates.
- Begin testing the data model earlier using mock data to identify relationship flaws sooner.
- Set aside more time for data entry since it was quite challenging as the number of entities increased and relationships became more complex.
- Keep system assumptions upfront throughout the process to ensure nothing is overlooked in later stages like query development.