



**CHANDIGARH
UNIVERSITY**

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

CASE STUDY REPORT ON PARTICULAR CASE STUDY

Program Name: BCA

Subject Name/Code: Database Management
System (23CAT-251)

Submitted by:

Name: Vishal singh

UID: 23bca10362

Section: BCA-4 (b)

Submitted to:

Name: Arvinder singh

Designation: Professor

Introduction:

The **Merchant Navy** is a vital component of the global economy, responsible for transporting goods, commodities, and passengers across international borders through sea routes. It encompasses commercial shipping operations that ensure the seamless movement of cargo and personnel, contributing significantly to global trade and maritime logistics. With the continuous expansion of international commerce, the complexity of managing merchant navy operations has also increased. This has created a growing demand for efficient, reliable, and scalable digital systems to streamline administrative tasks, manage personnel, track ship activities, and handle port logistics.

In the rapidly evolving digital age, data has become a cornerstone of operational efficiency. Maritime organizations need to manage vast amounts of data related to crew members, vessels, ports, training programs, certifications, and compliance protocols. Manual methods of data handling are not only time-consuming but also prone to errors and inconsistencies. Hence, there is a pressing need for an integrated solution that ensures data accuracy, consistency, and ease of access. This is where the **Merchant Navy Portal Database Project** becomes critically important.

The core objective of this project is to design and implement a **relational database management system (RDBMS)** tailored specifically for the Merchant Navy. This system provides a structured and logical approach to storing and retrieving data related to various entities such as seafarers, ships, training courses, enrollments, ports, and medical records. By utilizing SQL (Structured Query Language), the system ensures data integrity, reduces redundancy, and supports complex queries to facilitate efficient decision-making.

This project serves as both an academic exercise in database design and a practical demonstration of how digital infrastructure can transform traditional maritime operations. The database comprises multiple interconnected tables, each representing a real-world maritime component. These tables are linked through well-defined relationships, and their structure is optimized for performance, scalability, and clarity. The system supports essential operations like assigning crew members to ships, tracking medical examinations, managing training enrollments, and scheduling port arrivals and departures.

Moreover, the Merchant Navy Portal sets a strong foundation for future development, including web interfaces, automation of routine tasks, and integration with other maritime information systems. It opens up possibilities for enhancing transparency, accountability, and real-time monitoring in maritime affairs, thereby contributing to safer and more effective maritime operations. As global trade continues to expand, the need for such intelligent systems will only

The portal aims to:

1. Simplifying Recruitment for Shipping Companies:

- **Job Matching:** By leveraging structured data (like Jobs, Applications, Candidates), you can match candidates to roles based on specific criteria such as skills, certifications, and experience.
- **Application Tracking:** With an organized Applications system, the recruitment process can be tracked efficiently. Each stage of the application process can be easily monitored, helping companies identify bottlenecks or candidates who need further follow-up.

2. Providing Seafarers with Career and Training Opportunities:

- **Training Opportunities:** Seafarers can access relevant courses and certifications that will improve their skills and make them more competitive in the job market. This could be part of a training ecosystem that supports various maritime and safety certifications (STCW, first aid, navigation, etc.).
- **Skill Development:** The platform could support personalized skill development paths, allowing seafarers to improve over time, track progress, and be recommended for jobs that match their skill growth.

3. Enabling Transparent Communication Across Stakeholders:

- **Messaging System:** The platform facilitates direct communication between seafarers and recruiters, HR teams, or trainers. This is particularly useful for clarifying doubts about roles, training courses, or job requirements.
- **Feedback Loops:** Seafarers can provide feedback on job opportunities, recruiters, or courses they have attended. Shipping companies can also leave feedback on applicants. This transparency helps both parties make informed decisions.

Technique

1. MySQL and Relational Databases

MySQL is a widely used **Relational Database Management System (RDBMS)**. It stores data in tables, where each table consists of rows (records) and columns (attributes). The key concept in a relational database is that the data is **organized in relations (tables)**, and these relations are connected through **keys** (such as primary and foreign keys).

2. Entity-Relationship (ER) Modeling

Entity-Relationship (ER) modeling is a technique used to design the database schema at a conceptual level. It helps in visualizing the relationships between different entities in a system (for example, a "Customer" entity and an "Order" entity). ER diagrams typically represent:

- **Entities:** Objects or things in the database (like a "Product" or "Employee").
- **Attributes:** Properties or characteristics of the entities (like "Product Name" or "Employee ID").
- **Relationships:** How entities are related to each other (like a "Customer" placing an "Order").

In ER modeling, the relationships between entities are often depicted as one-to-one, one-to-many, or many-to-many, depending on the business rules.

3. SQL DDL (Data Definition Language) and DML (Data Manipulation Language)

SQL is the language used to interact with relational databases. It is divided into several sublanguages:

- **DDL (Data Definition Language):** These SQL commands are used to define and modify the structure of the database, such as creating, altering, and deleting tables or database schemas. Common DDL commands include:
 - **CREATE:** Defines new tables, views, or schemas.
 - **ALTER:** Modifies an existing database structure.
 - **DROP:** Deletes tables or other database objects.
- **DML (Data Manipulation Language):** These SQL commands allow for the manipulation of data in the database. DML commands include:
 - **INSERT:** Adds new records to a table.

- UPDATE: Modifies existing data in a table.
- DELETE: Removes data from a table.
- SELECT: Retrieves data from one or more tables based on specific criteria.

4. Foreign Keys and Referential Integrity

- **Foreign keys** are used to establish a link between two tables. A foreign key in one table points to the **primary key** of another table, ensuring that relationships between tables are consistent.
- **Referential integrity** is a database concept that ensures that the relationships between tables remain consistent. This means that a foreign key must always reference a valid record in the related table. For example, in an "Orders" table, if each order is associated with a specific "Customer," the foreign key in the "Orders" table should always point to a valid "Customer" record in the "Customers" table. If a customer is deleted, the system may prevent the deletion of all orders associated with that customer, or it could automatically delete those orders (depending on the design of the foreign key constraint).

5. Normalization

Normalization is the process of organizing the data in the database to avoid redundancy and improve data integrity. It involves dividing a database into smaller, related tables and applying rules (known as normal forms) to eliminate issues such as data duplication or update anomalies. Some common normal forms are:

- **First Normal Form (1NF)**: Ensures that the table has no repeating groups or arrays.
- **Second Normal Form (2NF)**: Ensures that all non-key attributes depend on the entire primary key.
- **Third Normal Form (3NF)**: Ensures that there is no transitive dependency between non-key attributes.

6. Sample Data for Real-World Simulation

Once the database schema is designed and implemented, **sample data** is often inserted into the database to simulate real-world usage. This data can be representative of actual use cases, such as customers, orders, or products in an

e-commerce system, and helps test queries and ensure the database is functioning correctly.

System Configuration

System Configuration Overview for MySQL / MariaDB Project

This section outlines the configuration requirements necessary to run the MySQL/MariaDB database system efficiently, along with the software dependencies and platform specifications.

1. DBMS (Database Management System):MySQL / MariaDB

- **MySQL:** A widely used open-source relational database management system (RDBMS) that uses SQL to manage data. It is ideal for both small and large-scale applications.
- **MariaDB:** A fork of MySQL, MariaDB is fully compatible with MySQL and is designed as an open-source alternative with enhanced features and performance improvements.

These two systems are highly compatible, and the system configuration would apply equally to either of them. The choice between MySQL and MariaDB typically depends on factors like community support or specific performance features.

2. SQL Version: ANSI SQL-2011 Compatibility

- **ANSI SQL-2011:** This is a standard version of the SQL language established by the American National Standards Institute (ANSI). The SQL-2011 standard introduces various features and improvements over earlier versions of SQL, such as better support for querying JSON data and more sophisticated window functions. Ensuring compatibility with this standard means that your database will support a wide range of SQL features that are industry-standard, making your application more portable and future-proof.

3. Platform: Windows/Linux/Mac with MySQL Workbench / phpMyAdmin

- **Supported Platforms:** The MySQL database server and associated tools are cross-platform, so it works on various operating systems such as:
 - **Windows**
 - **Linux**
 - **MacOS**

This flexibility allows the database to be deployed on any major operating system, depending on the project's needs and the development environment of the team.

- **Database Management Tools:**
 - **MySQL Workbench:** A graphical tool for database design, management, and maintenance. It provides an intuitive interface for interacting with the database, creating tables, running queries, and visualizing schema.
 - **phpMyAdmin:** A web-based interface for managing MySQL databases. It is particularly useful for managing databases on web servers, providing easy-to-use forms for importing, exporting, and querying databases.

4. Hardware Requirements:

- **RAM:** Minimum of **4 GB**. This amount of RAM ensures that the system can handle typical database operations like querying and processing efficiently. For larger datasets, additional RAM would improve performance.
- **Processor:** Dual Core or better. A dual-core processor is sufficient for small to medium-sized applications, though a multi-core processor (quad-core or better) will provide better performance for handling concurrent queries and larger databases.
- **Storage:** **1 GB** of available storage is needed for development and testing purposes. For a production system, storage needs will vary based on the size of the database and the volume of data processed. As databases grow, so will the storage requirements.

5. Software Dependencies:

- **MySQL Server:** This is the core software that handles database operations. It needs to be installed on the system to manage and store your relational data.
 - You can download MySQL from [MySQL's official website](#).
 - MariaDB can be installed as an alternative from [MariaDB's official website](#).
- **SQL Client (Workbench, DBeaver, etc.):** These are tools used to connect to the MySQL/MariaDB server and interact with the database:
 - **MySQL Workbench:** A popular, official graphical interface for MySQL that provides tools for database design, development, and administration.

INPUT

The navy portal captures input across multiple dimensions:

Input Type	Example
User registration	Full Name, Email, Password, Role
Company Profile	Company Name, Website, Recruiter
Job Postings	Job Title, Ship, Experience, Salary
Applications	Resume, Cover Letter
Enrollments	Course, Payment Status
Internal Messages	Subject, Content, Sender/Receiver IDs



Brief explanation of the theory behind each type of input mentioned in the table:

1. User Registration:

- **Description:** User registration is the process of capturing basic information from users when they first interact with the system. This input allows the system to create an account for the user, providing them with personalized access and permissions.
- **Key Fields:**
 - **Full Name:** The name of the user, typically used for identification.
 - **Email:** A unique identifier for communication and login purposes.
 - **Password:** Used for authentication, ensuring the security of the user's account.
 - **Role:** Defines the user's permissions or access levels (e.g., Admin, Recruiter, Job Seeker, etc.).

2. Company Profile:

- **Description:** This input captures essential information about a company, which may be registered in the system to post job openings or manage recruitment.
- **Key Fields:**
 - **Company Name:** The legal name of the company.
 - **Website:** The company's online presence, often used for branding and further information.
 - **Recruiter:** The contact person or the role responsible for recruitment at the company.

The company profile helps establish a company identity in the system and allows job seekers or other users to know more about the company's culture, offerings, and job posts.

3. Job Postings:

- **Description:** Job postings are used to capture information about the roles a company is looking to fill. These postings are displayed to potential candidates seeking jobs.
- **Key Fields:**
 - **Job Title:** The position or title of the job being offered (e.g., Software Engineer, Marketing Manager).



- **Ship:** This could refer to the type of employment or work shift (e.g., morning shift, night shift).
- **Experience:** The amount of professional experience required for the role (e.g., 3-5 years).
- **Salary:** The compensation offered for the position.

Job postings allow candidates to evaluate job opportunities and apply for roles that fit their qualifications and interests.

4. Applications:

- **Description:** Applications represent the process by which job seekers submit their interest in a position. This usually involves uploading important documents like resumes and cover letters.
- **Key Fields:**
 - **Resume:** A document that outlines the applicant's professional experience, skills, and qualifications.
 - **Cover Letter:** A letter that accompanies the resume, typically explaining why the applicant is a good fit for the role.

5. Training Courses:

- **Description:** This input type captures details of training or learning programs that users can enroll in, helping improve their skills and qualifications.
- **Key Fields:**
 - **Name:** The title or name of the course.
 - **Duration:** The length of the course (e.g., 4 weeks, 6 months).
 - **Fees:** The cost of attending the course.
 - **Instructor:** The person teaching or leading the course.

Training courses offer professional development opportunities and are typically linked to improving the user's skills for career growth or job readiness.

6. Enrollments:

- **Description:** Enrollments capture the information related to users signing up for training courses. This input often involves tracking the user's payment status and participation.
- **Key Fields:**
 - **Course:** The specific course the user has enrolled in.



- **Payment Status:** The current status of the payment (e.g., Paid, Pending, Unpaid).

Enrollment input ensures that the user is successfully registered for the course and that financial transactions are tracked.

7. Internal Messages:

- **Description:** Internal messages allow communication within the system, often between users (e.g., recruiters and job seekers, course instructors and students). This can serve as a way for users to stay informed and connected.
- **Key Fields:**
 - **Subject:** The topic or title of the message.
 - **Content:** The body or text of the message.
 - **Sender/Receiver IDs:** Identifiers of the sender and the receiver of the message.

Internal messages facilitate communication and provide a direct channel for notifications, updates, or discussions between users.

Theory Behind Input Types:

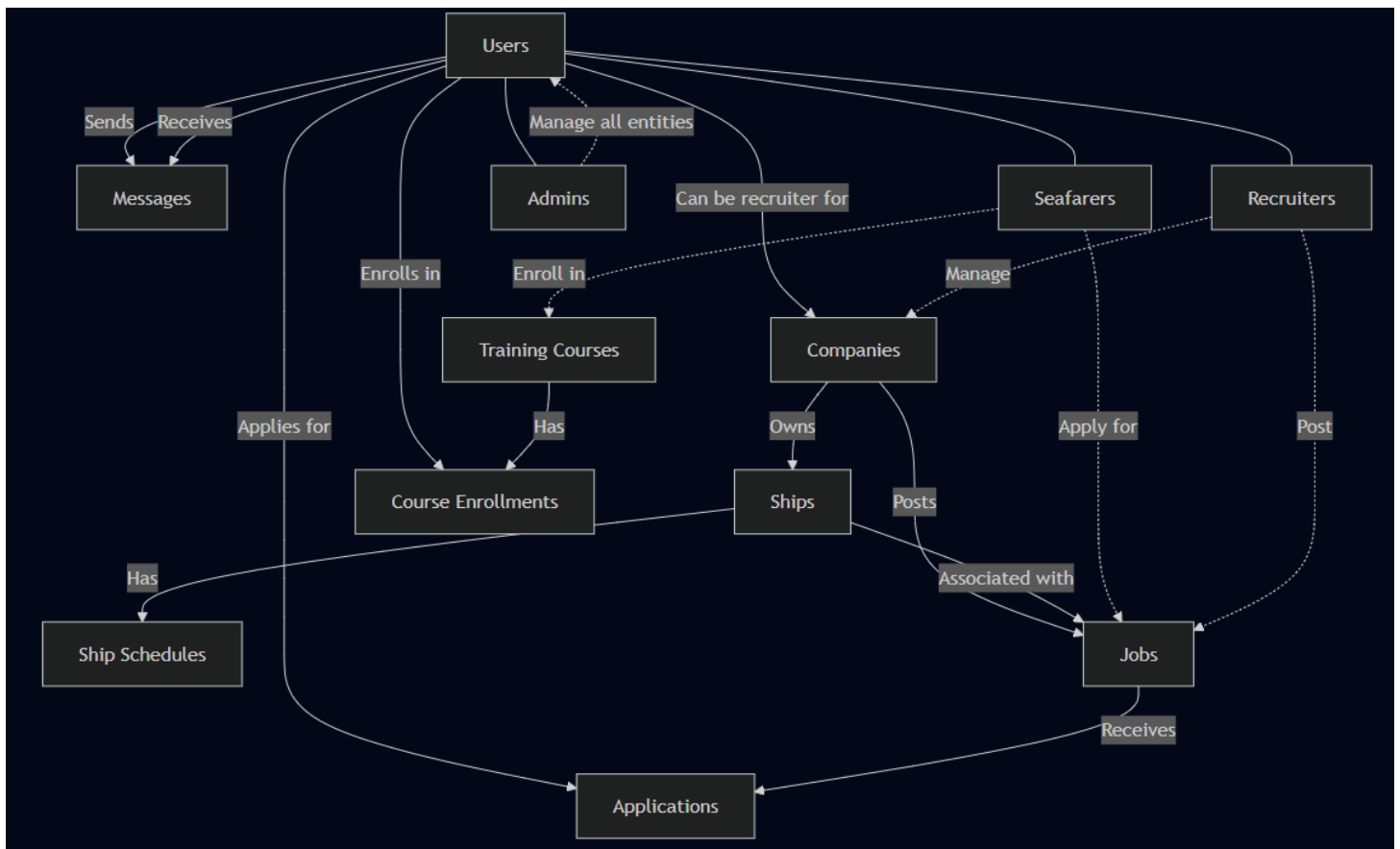
Each of these input types is designed to capture and store essential data that supports the functionality and operations of the system. The structure of the input allows for efficient data management, retrieval, and processing.

- **Normalization:** By structuring inputs into distinct types (e.g., user registration, job postings, applications), the system can maintain normalized tables in the database. This reduces redundancy and ensures data integrity.
- **Security and Access Control:** Data like passwords, payment status, and personal information must be handled securely. This often involves encryption, access control mechanisms, and role-based permissions to ensure that users can only interact with data that they are authorized to access.

- **Business Logic and User Experience:** The way data is captured reflects the business logic of the system. For example, the process of applying for a job is designed to be seamless, requiring inputs like a resume and cover letter. Similarly, the training enrollment process ensures that users can track courses and payment statuses, making it easy to access educational resources.

In summary, capturing input across multiple dimensions allows the system to meet the diverse needs of its users, from job seekers to recruiters to students, by structuring the data in a meaningful way to facilitate efficient processing, storage, and interaction.

ER DIAGRAM



The diagram shows a system that connects several key entities: Users, Seafarers, Recruiters, Companies, Ships, Jobs, Training Courses, and Applications. The relationships between these entities are represented by the connecting lines and labeled with relationship types.

From a database design perspective, this is modeling the domain of maritime employment with a focus on how different actors interact within this ecosystem. The key theoretical concepts include:

1. **User Role Hierarchy:** Users serve as the top-level entity with specialized types (Admins, Seafarers, Recruiters) inheriting from it. This demonstrates generalization/specialization in data modeling, allowing common attributes to be stored once while enabling role-specific functionality. This efficient structure reduces redundancy while maintaining distinct capabilities for each user type.
2. **Many-to-Many Relationships:** Several many-to-many relationships exist, like between Users and Training Courses (via "Enrolls in"), resolved through the Course Enrollments join entity. These associative entities handle complex relationships where each entity connects to multiple instances of another, maintaining referential integrity while storing relationship-specific attributes.
3. **Workflow Modeling:** The diagram captures key maritime employment workflows: Seafarers apply for Jobs, Recruiters post Jobs, Companies own Ships, etc. These relationship paths document how information flows between entities and define the sequence of operations comprising core business processes throughout the recruitment cycle.
4. **Message System:** Users can send and receive Messages, enabling communication within the platform. This functionality facilitates direct interaction between different user types for job inquiries, application updates, and training notifications, maintaining a history of relevant exchanges within the system.

5. **Access Control:** Admin users "Manage all entities" indicating role-based access control implementation. This security approach restricts system access based on user roles, with administrators having highest privileges, ensuring users can perform required functions within appropriate boundaries.

TABLE REALTION

The Merchant Navy Portal database follows **Relational Database Design**, where each table represents an entity and is connected through **Primary** and **Foreign Keys**. Here's how the tables relate:

1. SHIP_DETAILS ↔ CREW

- **One-to-Many Relationship:** One ship can have multiple crew members.
- **Foreign Key:** CREW.Ship_ID → SHIP_DETAILS.Ship_ID

2. CREW ↔ ASSIGNMENT

- **One-to-Many Relationship:** One crew member can be assigned to multiple routes.
- **Foreign Key:** ASSIGNMENT.Crew_ID → CREW.Crew_ID

3. ROUTE ↔ ASSIGNMENT

- **One-to-Many Relationship:** One route can have multiple crew members assigned.
- **Foreign Key:** ASSIGNMENT.Route_ID → ROUTE.Route_ID

4. ROUTE ↔ PORT

- While not directly connected in the current schema, **ROUTE.Source** and **ROUTE.Destination** can be cross-referenced to **PORT.Port_Name** for advanced designs (e.g., future use of Port_ID instead of text).

TABULAR FORMAT

Table 1	Table 2	Relationship Type	Foreign Key	Explanation
SHIP_DETAILS	CREW	One-to-Many	CREW.Ship_ID → SHIP_DETAILS.Ship_ID	One ship can have multiple crew members.
CREW	ASSIGNMENT	One-to-Many	ASSIGNMENT.Crew_ID → CREW.Crew_ID	One crew member can be assigned to multiple routes.
ROUTE	ASSIGNMENT	One-to-Many	ASSIGNMENT.Route_ID → ROUTE.Route_ID	One route can have multiple assigned crew members.
ROUTE	PORT	Optional/Reference	(Source/Destination → Port_Name)	Routes start/end at ports; indirectly connected via name.

SHIP DETAILS

Ship_ID	Ship_Name	Ship_Type	Capacity
1	Ocean King	Cargo	5000
2	Sea Queen	Passenger	3000

CREW

Crew_ID	Name	Rank	Ship_ID
101	John Smith	Captain	1
102	Alice Brown	First Mate	2
103	Mike Roger	Engineer	1

ROUTE

Route_ID	Source	Destination	Distance	Duration
201	Mumbai	Singapore	1800	5 days



202	New York	London	3100	7 days
-----	----------	--------	------	--------

ASSIGNMENT

Assignment_ID	Crew_ID	Route_ID	Date_Assigned
301	101	201	2025-04-01
302	102	202	2025-04-05
303	103	201	2025-04-03

PORT

Port_ID	Port_Name	Country
401	Port of Mumbai	India
402	Port of Singapore	Singapore
403	Port of London	UK

TABLE CREATION

Table: Seafarers

Column Name	Data Type	Constraints
seafarer_id	INT	PRIMARY KEY
name	VARCHAR(100)	NOT NULL
dob	DATE	NOT NULL
nationality	VARCHAR(50)	
rank	VARCHAR(50)	

Table: Ships

Column Name	Data Type	Constraints
ship_id	INT	PRIMARY KEY
name	VARCHAR(100)	NOT NULL
type	VARCHAR(50)	
flag	VARCHAR(50)	
built_year	INT	

Table: Voyages

Column Name	Data Type	Constraints
voyage_id	INT	PRIMARY KEY
ship_id	INT	FOREIGN KEY REFERENCES Ships(ship_id)
departure_port	VARCHAR(100)	
arrival_port	VARCHAR(100)	
departure_date	DATE	
arrival_date	DATE	

Table: Assignments

Column Name	Data Type	Constraints
assignment_id	INT	PRIMARY KEY
seafarer_id	INT	FOREIGN KEY REFERENCES Seafarers(seafarer_id)
ship_id	INT	FOREIGN KEY REFERENCES Ships(ship_id)
start_date	DATE	
end_date	DATE	

Table: Certificates

Column Name	Data Type	Constraints
certificate_id	INT	PRIMARY KEY
seafarer_id	INT	FOREIGN KEY REFERENCES Seafarers(seafarer_id)
certificate_type	VARCHAR(100)	
issue_date	DATE	
expiry_date	DATE	

Table: Ports

Column Name	Data Type	Constraints
port_id	INT	PRIMARY KEY
name	VARCHAR(100)	NOT NULL
country	VARCHAR(50)	
code	VARCHAR(10)	

SQL QUERIES WITH OUTPUT

-- 1. List all jobs with their company names and ship details

```
SELECT j.JobTitle, c.CompanyName, s.ShipName, s.ShipType, j.Salary, j.ClosingDate  
  
FROM Jobs j  
  
JOIN Companies c ON j.CompanyID = c.CompanyID  
  
JOIN Ships s ON j.ShipID = s.ShipID  
  
WHERE j.Status = 'Open'  
  
ORDER BY j.Salary DESC;
```

Result Grid						
Filter Rows:		Export:		Wrap Cell Content:		
JobTitle	CompanyName	ShipName	ShipType	Salary	ClosingDate	
Chief Engineer	Maersk Line	Maersk Madrid	Container Ship	12000.00	2023-07-15 00:00:00	
Chief Officer	Mitsui O.S.K. Lines	MOL Triumph	Container Ship	9000.00	2023-08-30 00:00:00	
Second Officer	Mediterranean Shipping Company	MSC Oscar	Container Ship	6500.00	2023-07-20 00:00:00	
Electrician	COSCO Shipping	COSCO Pacific	Container Ship	5500.00	2023-08-05 00:00:00	
Radio Officer	CMA CGM	CMA CGM Antoine	Container Ship	5200.00	2023-08-20 00:00:00	
Third Engineer	Hapag-Lloyd	Hapag Express	Container Ship	5000.00	2023-08-10 00:00:00	
Chief Cook	Shipping Corporation of India	SCI Chennai	Tanker	4500.00	2023-07-30 00:00:00	
Bosun	NYK Line	NYK Vega	Bulk Carrier	4000.00	2023-08-15 00:00:00	
Able Seaman	Evergreen Marine	Ever Given	Container Ship	3500.00	2023-07-25 00:00:00	
Deck Cadet	APL	APL Sentosa	Container Ship	1800.00	2023-08-25 00:00:00	

-- 2. Count the number of applications for each job

```
SELECT j.JobTitle, c.CompanyName, COUNT(a.ApplicationID) AS ApplicationCount  
  
FROM Jobs j  
  
LEFT JOIN Applications a ON j.JobID = a.JobID  
  
JOIN Companies c ON j.CompanyID = c.CompanyID  
  
GROUP BY j.JobID, j.JobTitle, c.CompanyName  
  
ORDER BY ApplicationCount DESC;
```

Result Grid Filter Rows: Export: Write			
	JobTitle	CompanyName	ApplicationCount
▶	Chief Engineer	Maersk Line	1
	Second Officer	Mediterranean Shipping Company	1
	Able Seaman	Evergreen Marine	1
	Chief Cook	Shipping Corporation of India	1
	Electrician	COSCO Shipping	1
	Third Engineer	Hapag-Lloyd	1
	Bosun	NYK Line	1
	Radio Officer	CMA CGM	1
	Deck Cadet	APL	1
	Chief Officer	Mitsui O.S.K. Lines	1

-- 3. Find the most popular training courses based on enrollment

```
SELECT t.CourseName, t.Instructor, COUNT(e.EnrollmentID) AS EnrollmentCount
FROM TrainingCourses t
```

```
LEFT JOIN CourseEnrollments e ON t.CourseID = e.CourseID
```

```
GROUP BY t.CourseID, t.CourseName, t.Instructor
```

```
ORDER BY EnrollmentCount DESC;
```

CourseName	Instructor	EnrollmentCount
Basic Safety Training	Capt. Suresh Sharma	1
Advanced Firefighting	Lt. Sarah Johnson	1
GMDSS Operator Course	Eng. Thomas Brown	1
Ship Security Officer	Capt. Mark Davis	1
Medical First Aid	Dr. Emily White	1
Tanker Familiarization	Capt. Robert Miller	1
Bridge Resource Management	Capt. Jan Visser	1
Engine Room Resource Management	Chief Eng. Hiroshi Yamamoto	1
Survival Craft and Rescue Boats	Capt. Ravi Menon	1
Dynamic Positioning Basic	Eng. William Scott	1

-- 4. Find users who have applied for multiple jobs

```
SELECT u.UserID, u.FullName, u.Email, COUNT(a.ApplicationID) AS ApplicationCount
FROM Users u
```



```
JOIN Applications a ON u.UserID = a.UserID
```


GROUP BY u.UserID, u.FullName, u.Email

HAVING COUNT(a.ApplicationID) > 1

ORDER BY ApplicationCount DESC;

Result Grid

  Filter Rows:

Export: 

	UserID	FullName	Email	ApplicationCount
▶	4	Vikram Singh	vikram@email.com	2
	5	Michael Johnson	michael@email.com	2
	6	Liu Wei	liu@email.com	2
	7	Ananya Patel	ananya@email.com	2
	8	Carlos Rodriguez	carlos@email.com	2

-- 5. Calculate the average salary offered by each company

SELECT c.CompanyName, AVG(j.Salary) AS AverageSalary, MIN(j.Salary) AS MinSalary,
MAX(j.Salary) AS MaxSalary

FROM Companies c

JOIN Jobs j ON c.CompanyID = j.CompanyID

GROUP BY c.CompanyID, c.CompanyName

ORDER BY AverageSalary DESC;

Result Grid	Filter Rows:	Export:	Wrap Cell
CompanyName	AverageSalary	MinSalary	MaxSalary
Maersk Line	12000.000000	12000.00	12000.00
Mitsui O.S.K. Lines	9000.000000	9000.00	9000.00
Mediterranean Shipping Company	6500.000000	6500.00	6500.00
COSCO Shipping	5500.000000	5500.00	5500.00
CMA CGM	5200.000000	5200.00	5200.00
Hapag-Lloyd	5000.000000	5000.00	5000.00
Shipping Corporation of India	4500.000000	4500.00	4500.00
NYK Line	4000.000000	4000.00	4000.00
Evergreen Marine	3500.000000	3500.00	3500.00
APL	1800.000000	1800.00	1800.00

-- 6. List all users who have enrolled in courses but haven't paid yet

SELECT u.UserID, u.FullName, u.Email, COUNT(e.EnrollmentID) AS UnpaidEnrollments

FROM Users u

JOIN CourseEnrollments e ON u.UserID = e.UserID

WHERE e.PaymentStatus = 'Pending'

GROUP BY u.UserID, u.FullName, u.Email

ORDER BY UnpaidEnrollments DESC;

Result Grid	Filter Rows:	Export:	Wrap
UserID	FullName	Email	UnpaidEnrollments
6	Liu Wei	liu@email.com	2
5	Michael Johnson	michael@email.com	1
4	Vikram Singh	vikram@email.com	1

-- 7. Find the recruiters with the most job postings

SELECT u.UserID, u.FullName, u.Email, COUNT(j.JobID) AS JobCount

FROM Users u

JOIN Companies c ON u.UserID = c.RecruiterID

JOIN Jobs j ON c.CompanyID = j.CompanyID

WHERE u.UserType = 'Recruiter'

GROUP BY u.UserID, u.FullName, u.Email

ORDER BY JobCount DESC;

	UserID	FullName	Email	JobCount
▶	2	John Smith	recruiter@maersk.com	1
	3	Emma Watson	recruiter@msc.com	1
	9	Hiroshi Tanaka	recruiter@evergreen.com	1

-- 8. Find the success rate of applications for each seafarer

SELECT u.FullName,

COUNT(a.ApplicationID) AS TotalApplications,

SUM(CASE WHEN a.Status = 'Accepted' THEN 1 ELSE 0 END) AS

AcceptedApplications,

(SUM(CASE WHEN a.Status = 'Accepted' THEN 1 ELSE 0 END) /

COUNT(a.ApplicationID)) * 100 AS SuccessRate

FROM Users u

JOIN Applications a ON u.UserID = a.UserID

WHERE u.UserType = 'Seafarer'

GROUP BY u.UserID, u.FullName

ORDER BY SuccessRate DESC;

FullName	TotalApplications	AcceptedApplications	SuccessRate
Ananya Patel	2	2	100.0000
Vikram Singh	2	0	0.0000
Michael Johnson	2	0	0.0000
Liu Wei	2	0	0.0000
Carlos Rodriguez	2	0	0.0000

-- 9. Calculate the duration of each ship's journey and list them in descending order

SELECT s.ShipName,

sh.DeparturePort,

sh.ArrivalPort,

sh.DepartureDate,

sh.ArrivalDate,

TIMESTAMPDIFF(HOUR, sh.DepartureDate, sh.ArrivalDate) AS JourneyHours,

ROUND(TIMESTAMPDIFF(HOUR, sh.DepartureDate, sh.ArrivalDate)/24, 1) AS
JourneyDays

FROM Ships s

JOIN ShipSchedules sh ON s.ShipID = sh.ShipID

ORDER BY JourneyHours DESC;

ShipName	DeparturePort	ArrivalPort	DepartureDate	ArrivalDate	JourneyHours	JourneyDays
MSC Oscar	Shanghai, China	Los Angeles, USA	2023-06-10 10:00:00	2023-06-25 16:00:00	366	15.3
Maersk Madrid	Rotterdam, Netherlands	Singapore	2023-06-01 08:00:00	2023-06-15 14:00:00	342	14.3
APL Sentosa	Oakland, USA	Hong Kong	2023-07-22 12:00:00	2023-08-05 14:00:00	338	14.1
MOL Triumph	Tokyo, Japan	Long Beach, USA	2023-07-25 13:00:00	2023-08-08 15:00:00	338	14.1
NYK Vega	Yokohama, Japan	Vancouver, Canada	2023-07-15 10:00:00	2023-07-28 12:00:00	314	13.1
Hapag Express	Hamburg, Germany	New York, USA	2023-07-10 09:00:00	2023-07-20 15:00:00	246	10.3
Ever Given	Dubai, UAE	Mumbai, India	2023-06-20 09:00:00	2023-06-25 11:00:00	122	5.1
SCI Chennai	Mumbai, India	Colombo, Sri Lanka	2023-07-01 07:00:00	2023-07-03 13:00:00	54	2.3
COSCO Pacific	Qingdao, China	Busan, South Korea	2023-07-05 08:30:00	2023-07-07 10:00:00	49	2.0
CMA CGM Antoine	Marseille, France	Algeiras, Spain	2023-07-18 11:00:00	2023-07-20 09:00:00	46	1.9

-- 10. Analyze the relationship between ship capacity and job salary

```
SELECT s.ShipType,  
AVG(s.Capacity) AS AvgCapacity,  
AVG(j.Salary) AS AvgSalary,  
COUNT(j.JobID) AS JobCount  
  
FROM Ships s  
  
JOIN Jobs j ON s.ShipID = j.ShipID  
  
GROUP BY s.ShipType  
  
ORDER BY AvgSalary DESC;
```

	ShipType	AvgCapacity	AvgSalary	JobCount
▶	Container Ship	17102.2500	6062.500000	8
	Tanker	12000.0000	4500.000000	1
	Bulk Carrier	82000.0000	4000.000000	1

-- 11. Find the most active communication channels between user types

```
SELECT  
  
u1.UserType AS SenderType,  
  
u2.UserType AS ReceiverType,  
  
COUNT(m.MessageID) AS MessageCount,  
  
MIN(m.SentDate) AS FirstCommunication,  
  
MAX(m.SentDate) AS LastCommunication  
  
FROM Messages m  
  
JOIN Users u1 ON m.SenderID = u1.UserID
```

JOIN Users u2 ON m.ReceiverID = u2.UserID

GROUP BY u1.UserType, u2.UserType

ORDER BY MessageCount DESC;

	SenderType	ReceiverType	MessageCount	FirstCommunication	LastCommunication
▶	Seafarer	Recruiter	5	2023-06-08 13:45:00	2023-06-26 13:20:00
	Recruiter	Seafarer	2	2023-06-09 10:20:00	2023-06-21 09:30:00
	Admin	Seafarer	2	2023-06-15 09:15:00	2023-06-24 10:45:00
	Seafarer	Admin	1	2023-06-23 16:15:00	2023-06-23 16:15:00

SUMMARY

The **Merchant Navy Portal Database Project** is a meticulously designed relational database system developed to streamline and optimize the management of key data related to the operations of the merchant navy. This project serves as a foundational tool for efficiently handling the administrative, training, medical, and logistical aspects of maritime personnel and ship movements. Its primary goal is to centralize and automate data management processes, which traditionally relied on manual and paper-based systems, thereby increasing accuracy, accessibility, and efficiency.

This database project includes a variety of interconnected tables such as **Seafarers**, **Ranks**, **Ships**, **Ship Assignments**, **Courses**, **Enrollments**, **Medical Records**, and **Port Schedules**. Each of these tables is crafted with normalization principles in mind, avoiding redundancy and ensuring referential integrity. The **Seafarers** table records essential details of individuals working at sea, while the **Ranks** table categorizes them based on professional hierarchy and responsibility. The **Ships** table maintains a registry of vessels under the company's operation, and the **Ship Assignments** table tracks deployment schedules of seafarers on specific ships.

The **Courses** and **Enrollments** tables ensure a transparent training management system by recording all training programs offered and the participation of seafarers in these courses. Additionally, **Medical Records** are critical for compliance with maritime health standards, and the **Port Schedules** table provides information regarding the port visits and schedules of ships, which helps in route planning and logistical coordination.

A set of SQL queries has also been developed and tested as part of the project, allowing users to retrieve detailed reports such as crew assignments, course histories, fitness statuses, and port schedules. These queries demonstrate the practical use of the database in real-world scenarios, enabling shipping companies and port authorities to make timely and informed decisions.



CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Overall, the project exemplifies the use of relational database management systems (RDBMS) in supporting maritime operations and demonstrates the potential of digital solutions in transforming legacy procedures into dynamic and responsive systems.

CONCLUSION

In conclusion, the **Merchant Navy Portal database project** represents a significant step toward the digital transformation of merchant navy operations. It not only addresses the inherent challenges faced by maritime administrators—such as crew tracking, medical fitness monitoring, training compliance, and ship deployment—but also provides a robust and scalable framework that can be expanded further for enterprise-level implementations.

By using a relational database approach, the project ensures that all data remains consistent, logically organized, and easily retrievable. This results in increased operational transparency and better resource management, which are critical in the high-stakes maritime industry. Through the implementation of structured queries and defined relationships between entities, the system offers the ability to generate timely reports, assess personnel readiness, and maintain regulatory compliance, all of which contribute to the safety and efficiency of maritime operations.

Furthermore, this project has educational value, as it showcases how complex real-world systems can be broken down into manageable components through thoughtful database design. It opens up avenues for future development such as integrating web or mobile applications, incorporating real-time GPS-based tracking, and enhancing reporting through analytics and dashboards.