Maritime Vessel Route Simulation & Data Engineering Challenge

Background

Our company specializes in AI-powered maritime solutions to monitor, protect, and optimize operations at sea. A key aspect of our work involves processing vessel tracking data, specifically the **Automatic Identification System (AIS)**. AIS is a standardized system used by ships and vessel traffic services to identify and locate vessels. AIS messages are broadcast in a standardized format of AIVDM sentences, containing information such as vessel position, speed, and heading. While the AIVDM documentation is provided for reference, you do not need to delve deeply into it.

Our products must operate in both online and offline environments, often under resource constraints, requiring creative problem-solving and adaptability.

This challenge is designed to evaluate your problem-solving skills, ability to learn new tools, and approach to real-world data engineering tasks. We value clear thinking, curiosity, and communication over tool-specific expertise.

The Challenge

1. Route Generation

- You are provided with a CSV file containing a list of ports (with names and coordinates). The World Port Index is a good source for this data.
- For **N vessels**, randomly select two ports and generate a realistic vessel route using the `searoute-py` library (not straight lines).
- For the minimum requirement, you may simulate only one vessel and one route.

2. AIS Simulation

- Simulate a vessel moving along the generated route at fixed intervals (e.g., every 5 minutes), assuming a reasonable vessel speed.
- At each interval, generate an AIS position report and encode it as an AIVDM sentence using the `py-ais` library.
- Treat each vessel as an object with a unique MMSI (Maritime Mobile Service Identity) number and a constant speed for simplicity.
- The `pyais` library requires the following inputs to generate a position report:
  - MMSI
  - Latitude
  - Longitude
  - Message ID (assume `1` for simplicity)
- Ensure the simulation captures the actual movement characteristics of the vessel at the given speed. Note that the route generated by `searoute-py` is a list of waypoints and does not represent the actual positions of the vessel.
- Implement a playback system that streams these AIS messages over a WebSocket. The messages should follow this structure:

```
 {
 "message": "AIVDM",
 "mmsi": <MMSI>,
 "timestamp": <timestamp>,
 "payload": <AIVDM message>
}
```

- The playback system must support a **simulation speed factor**:
- `1.0` = real time (5 minutes in simulation = 5 minutes in real time)
- `2.0` = 2x speed (5 minutes in simulation = 2.5 minutes in real time)
- `-1` = send all messages immediately (no delay)
- You may choose to pre-calculate positions or generate them on the fly.

3. Data Engineering & Storage

Data Ingestion

- Implement a receiver that connects to the WebSocket, parses incoming AIVDM messages (using `py-ais`), and ingests the data into a database of your choice.
- Design your ingestion pipeline to handle out-of-order or duplicate messages gracefully.
- Log and handle malformed or corrupt AIS messages.

Data Modeling

- Design a schema that efficiently stores AIS messages and supports time-based queries.
- Justify your choice of database (e.g., PostgreSQL, SQLite, MongoDB, DuckDB, etc.) and schema design, considering scalability, query performance, and offline/online deployment scenarios.

Data Validation & Quality

- Implement basic data validation (e.g., valid coordinates, timestamps, MMSI uniqueness).
- Track and report on data quality metrics (e.g., number of invalid messages, duplicates, missing fields).

Indexing & Optimization

- Implement indexing or partitioning strategies to optimize retrieval of vessel tracks and statistics.
- **Bonus:** Demonstrate query performance with sample data and explain any optimizations.

Tooling & Automation

- Provide scripts or tools to initialize the database, run the ingestion pipeline, and execute analytics queries.

Testing

- Include unit or integration tests for your ingestion and analytics logic.

4. Query & Analytics

- Implement queries or scripts to:
- Retrieve the full track (trajectory) for a given vessel.
- Calculate the total distance covered and average speed for a vessel within a specified time window.
- **Bonus:** Add any other relevant statistics, optimizations, or features you think are useful (e.g., support for multiple vessels/routes, efficient indexing, summary dashboards, etc.).

5. Documentation

- Provide a README that explains:
- Your approach and assumptions.
- How to run your solution.
- Design decisions and trade-offs.
- **If you deviate from the requirements due to time or other constraints, clearly explain what you changed and why.**
- *(Optional)* What you would improve or extend with more time.

Hints

- You are encouraged to use the `searoute-py` and `py-ais` libraries.
- You may use any programming language, but Python is recommended.
- You may simulate positions on the fly or pre-calculate them.
- Focus on clarity, reasoning, and communication.

Minimum Requirements

- One vessel, one route, basic simulation, and data storage.
- Clear documentation.

Bonus Features

- Support for multiple vessels/routes.
- Optimized queries and analytics.
- Additional statistics, features, or creative solutions.

Flexibility

If you are unable to complete any part of the requirements, you are allowed to **deviate slightly**. Please **justify your choices and explain the technical issues you faced** in your README. We are interested in your problem-solving process as much as the final result.

Submission Instructions

- Submit your project as a public or private **GitHub repository**.
- Include all source code, scripts, and a README as described above.
- The duration of the challenge is **1 week** from the date you receive it. If you need more time, please let us know and justify why.
- If you are unable to tackle the entirety of the problem due to professional obligations, propose a feasible limited scope and justify your choices. For example:
    - If you cannot create a live simulation, you can pre-calculate a trajectory and work from there instead of building a simulation tool.
    - Justifications are required for any reduction in scope. Failure to provide adequate technical reasoning will result in negative marks.
- We have no objection to the use of LLMs to assist with the challenge. However, you must understand the code you submit and be able to explain it. We will test your understanding beyond the codebase, so ensure you grasp the intricacies of the problem and your solutions.
- Ensure your repository is public, as we will not be able to access private repositories.
- If you have any questions, reach out via email with the subject line "AIS Challenge Question." If you have multiple questions, compile them into one email for efficiency. Thoughtful questions demonstrate critical thinking.

Grading Rubric

Category	Weight	Description
Approach & Problem Solving	40%	Clarity of reasoning, assumptions, and design decisions.
Code Quality & Documentation	20%	Readability, structure, comments, and clear README.
Functionality & Correctness	20%	Working simulation, correct AIS encoding/decoding, and data storage.
Creativity & Initiative	10%	Bonus features, optimizations, or insightful analytics.
Communication	10%	Clarity in explaining approach, trade-offs, and results.