The purpose of our project is to help people understand the ARGO ocean data without even needing to be an expert. The idea is that a user can directly type a question in normal English and the system will return back an answer with visualizations and summaries. Oceanographic data complex, and heterogeneous in nature, containing measurements like CTD casts, Argo floats, and BGC sensors. The Argo program, which deploys autonomous profiling floats across the world's oceans, generates an extensive dataset in NetCDF format containing temperature, salinity, and other essential ocean variables. Now Accessing, querying, and visualizing this data requires domain knowledge, technical skills, and familiarity with complex formats and tools, making it hard for laymen users. With the rise of AI and Large Language Models (LLMs), combined with structured databases and dashboards, it is now possible to create an intuitive system that truly democratizes access to this data. Our problem statement is directly addressing this challenge.

The first step is to create an ETL pipeline that will directly fetch the newly added NetCDF file's data and transform them into structured as well as unstructured formats, so that our LLM's context window never becomes outdated. We can do this by automating an update script that runs in the background on a scheduled basis, such as once every night. Once this is done, our ETL pipeline would be ready. The next step is the real-time querying part, where we are going to use a RAG model that will take the user's prompt in natural language and then transform it into vector form using the sentence-transformer library (Python). This vectorized prompt will be sent to the Chroma database where the vectorized prompt's semantic similarity will be matched, and based upon that, the Chroma database will send back the relevant context to the RAG. At this point, the RAG has two things: the user's prompt and the context given by the Chroma database. Based upon these two, the RAG will generate an augmented prompt, which will then be given to the LLM. Using that prompt, the LLM will generate a SQL query, which will then be executed, and the data from the SQL query will again be sent to the LLM so that it can generate the desired visualizations and responses from it.

The current system we are proposing will therefore deliver an interactive dashboard where ARGO profiles like mapped trajectories, depth-time plots, and profile comparisons can be visualized. The chatbot-style interface will allow natural questions such as: "show me salinity profiles near the equator in March 2023," "compare BGC parameters in the Arabian Sea for the last 6 months," or "what are the nearest ARGO floats to this location?" This tool will bridge the gap between domain experts, decision-makers, and raw data by allowing non-technical users to extract meaningful insights without knowing the internal complexity. The expected solution will be an end-to-end pipeline with NetCDF processing, a relational (PostgreSQL) and vector database (Chroma), a backend RAG + LLM system for query interpretation along with a MCP server which will provide additional context to the LLM, and a frontend dashboard made using Streamlit with geospatial libraries like Plotly or Leaflet. We also plan to extend this later towards in-situ observations like BGC floats, gliders, buoys, and even satellite datasets to show the future potential of the project.

Now the key problems we might face are that initially our LLM will hallucinate, as we are using a generic LLM which will then need to be fine-tuned in order to be good at accurately generating responses. We also need to provide a high-level prompt with examples to the RAG initially so that it does not generate wrong responses. In addition, setting up the ETL pipeline will not be that easy, as we will need to somehow know which files have already been processed by our system. Another major challenge will be scalability, because oceanographic data is constantly growing. Our architecture must be able to handle larger volumes of profiles, ensure fast retrieval, and avoid overloading the LLM. We believe we can build such a system which will solve all these problems and work efficiently.