Project Report Guidelines

First and foremost, following the individual original work policy clearly stated at the start of the course, the topic and questions you ask in your project must be of your own invention. **If you used ideas from a particular web site or previous project, or did your project as part of an existing research collaboration, you must identify your sources and/or collaborators and provide links and citation(s) where appropriate.**

As a guide, the report should be around 8-10 pages depending on space used for any visualizations, tables, etc.

The format of the report is semi-flexible - you can include additional information, but at a minimum it should have the following sections:

# Motivation

1. Motivation (5 points):  (a) Briefly state the nature of your project and why you chose it. (b)  What specific question or goal did you try to address?

Our proposed project was to use Argo data (a data set of ocean floats that collect variables such as salinity and temperature in oceans around the world), map the change of salinity and temperature in the Estuary and Gulf of St. Lawrence, and identify potential correlations to fish populations totals in the same area.

The Estuary and Gulf of St. Lawrence (the “Gulf”) is an incredibly diverse and complex marine and estuary ecosystem and is one of the largest of its kind in the world. The area is made up of freshwater from the Canadian Shield, the Great Lakes basin and the St. Lawrence River system emptying out into the Atlantic Ocean, where it combines with the cold Labrador Current from the Arctic and the warm Gulf Stream from the tropics. These currents merge in a semi-enclosed and mostly shallow area, creating the perfect conditions for incredible diversity of life. The Gulf of St. Lawrence is the most important source of fish on the Atlantic side for the commercial fishing industry in both the U.S. and Canada, both top exporters of fish and seafood in the world[[1]](#footnote-1). By studying and reporting about this ecosystem and the species that depend on it, we hope to better inform others about its critical importance.

This project aimed to answer the following questions:

* How have ocean properties such as temperature and salinity changed over a period of ten years in the Gulf?
* How have pelagic fish populations changed in the Gulf within the same time period?
* Is there a meaningful correlation between ocean properties and fish population evolutions that merits further study?

# Data Sources

2. Data Sources (10 points):  Describe the properties of the two dataset(s) or API services you used. Be specific. Your information at a minimum should include but not be limited to:

* where the datasets or API resources are located,
* what formats they returned/used,
* what were the important variables contained in them,
* how many records you used or retrieved (if using an API), and
* what time periods they covered (if there is a time element)

For example, if you downloaded data or used API services, you should state the specific URLs to those files or resources. It should require zero effort on my part to find and access the exact resources you used if I need to do so.

## Argo Data

Argo is an international program that collects information about Earth’s oceans using a fleet of robotic instruments that drift with the ocean currents and move between the surface and mid-water level. For the scope of this project, we used Atlantic Ocean data between latitude 38 to 59 and longitude -70 to -35 (corresponding to a large zone including the St Lawrence Gulf).

### **Name**: Argo float data and metadata from the Global Data Assembly Centre (Argo GDAC) –Atlantic Ocean, 2009-2019

### **Data Location**: <ftp://usgodae.org/pub/outgoing/argo>

### **Format**: netCDF files (one per day)

### **Important variables**: Float identifier, depth, salinity, temperature, adjusted salinity, adjusted temperature (based on scientific calculations), quality flags, latitude and longitude. Date is derived from each file name.

### **Time period used**: 2009-2019

### **Size**: 36.8 GB (before pre-processing)

### **Total records**: 626,703 (post pre-processing)

### **Access method**: Files were downloaded from the FTP site, then records were pre-processed and stored in a relational database (AWS RDS).

## Global Temperature and Salinity Profile Programme Data

Based on an initial exploration of the Argo dataset, it became quickly clear that the dataset might be relatively sparse on float data within the Gulf itself. Data from the Global Temperature and Salinity Profile Programme (GTSPP), a joint international cooperative effort supporting the World Climate Research Programme, was used to supplement ARGO data for the same time period (2009-2019). Similar to Argo data, GTSPP data also collects sea measurements such as salinity and temperature in the Earth’s oceans, but instead of floats, data is collected from both ships and buoys. Measurements are not quite as deep as Argo data, as depth measurements max out typically well before a thousand meters.

### **Name:** Global Temperature and Salinity Profile Programme (GTSPP) data

### **Data Location**: <ftp://ftp.nodc.noaa.gov/pub/data.nodc/gtspp/best_nc/>

### **Formats**: individual netCDF files aggregated in a monthly tarball

### **Important variables contained in each file**: latitude, longitude, station or ship identifier, timestamp of measurement, depth of measurement (in meters), salinity, and temperature. Supplementing all these measurements are additional measurements denoting the quality of the measurement (e.g., ‘good’, ‘probably good’, ‘probably bad’, etc.).

### **Time period used**: 2009-2019

### **Size**: 132 tar.gz files, 7.83 GB (compressed)

### **Total records**: 43,573 (post pre-processing)

## Government of Canada’s Department of Fisheries and Oceans (DFO) – Quebec Coastal Thermograph Network

DFO’s dataset was used to supplement both the Argo float and GTSPP dataset at the surface level, as we were not sure we would have sufficient measurement points for the Gulf from the first two datasets. Data are collected from buoys and, unlike both Argo and GTSPP data, strictly surface level data only (< than 100 meters).

### **Name**: Department of Fisheries and Oceans (DFO) - Quebec Coastal Thermograph Network

### **Data Location**: <https://open.canada.ca/data/en/dataset/848e943b-1a98-43b8-acb3-ac89af17ea41> (HTTP access)

### **Formats Used**: a zipped file containing 2 folders, 1 with CSV files of each station and another folder with graphs.

### **Important variables contained in each file**: Timestamp of measurement, station id, latitude, longitude, depth of measurement, temperature and salinity.

### **Time period used**: 2009-2019

### **Total records**: 158,764 (post pre-processing)

## Government of Canada’s Department of Fisheries and Oceans (DFO) – Pelagic Fish Populations in the Gulf

The Canadian Department of Fisheries and Ocean conducts annual multidisciplinary surveys of the Northern and Southern Gulf of St. Lawrence to capture information on groundfish and invertebrates’ abundance, spatial distribution and diversity. The pelagic species represented in the dataset are: Arctic Cod, Atlantic Argentine, Atlantic Herring, Atlantic Mackerel, Atlantic Soft Pout, Capelin, Lumpfish, Pollock, Rainbow Smelt, Sand Lances, Silver Hake, Threespine Stickleback and White Barracudina.

### **Name**: Pelagic fish species abundance in the Estuary and Gulf of St. Lawrence between 2009 and 2018

### **Data Location**: <https://open.canada.ca/data/en/dataset/f1fc359c-0ed1-4045-a421-adef2497b68d>

### **Formats Used**: a zipped file containing 2 CSV files from http download, one for each location.

### **Important variables contained in each file**: Date of measurement, the station id, latitude, longitude, and quantity of specific fish species.

### **Time Periods Covered**: 2009 - 2018

### **Size**: approx. 1,700 lines of data per file (0.2MB zipped)

### **Access method**: HTTP download and from relational database post processing

### **Total records**: 41,145 (post pre-processing)

## Estuary and Gulf of St. Lawrence Shape File

A private label shape file denoting the boundaries of the Gulf and Estuary of St. Lawrence.

### **Name**: Gulf of St. Lawrence shapefile

### **Short Description**: Shapefile of the Gulf and Estuary of St. Lawrence for visualization purposes.

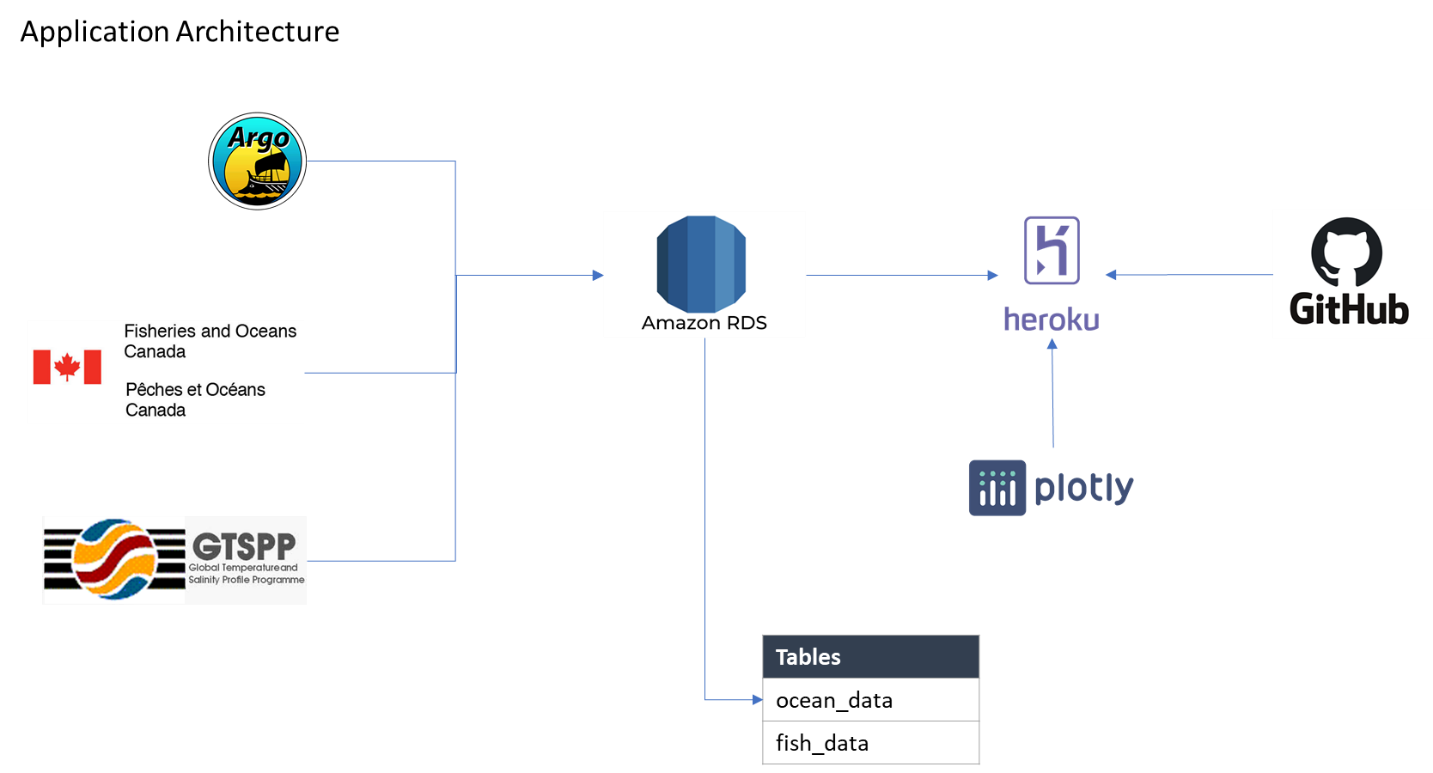
### **Size**: <1MB

### **Location**: <https://www.marineregions.org/gazetteer.php?p=details&id=4290>

### **Format**: Shapefile

### **Access method**: Downloaded and stored in GitHub repository. Opened and manipulated with Python’s shapely and pyshape.

Below is the architecture of our live application.



# Data Manipulation

3. Data Manipulation Methods (30 points):  For each of your two sources, describe how you manipulated the data.  For example:

## Data helpers/packages (files are located in the /helper folder)

Since our datasets are quite large and not conducive towards local Jupyter manipulation at large, our data processing and manipulation jobs were chunked and we used several helpers and processes to move data for cleanup and visualization.

### db.py: Since our data is stored in AWS RDS, a helper was written to move data to and from the databases. We chose to use PostgreSQL as our underlying database and the psycopg2 python package to read from and write to the database. The helper file covers:

1. Executing queries from AWS (*run\_query()*): Executing and returning basic SQL queries for Jupyter exploration and use. Typical SELECT statements will return in pandas dataframe format for immediate exploration and use. *run\_query()* can also be used to execute basic statements with no data returns, e.g., ALTER TABLE statements. For this project, *run\_query()* is relatively generic and does not allow for iterated/chunked returns.
2. Inserting dataframes: The *db* helper leverages the use of pandas *to\_sql()* although Postgres only works with that particular method in a very limited way. It works for bulk insert as long as there are no collisions on keys.
3. Upserts: A method used for slightly more specialized dataframe rows inserts (replace on collision with keys). Quite slow as it uses a single connection per dataframe row, but useful to update specific rows while keeping keys in place. Further optimization was not needed for this project.
4. Formatting strings: Some internal methods are built in to format items inside a dataframe (e.g., *\_val\_format()*, *\_clean\_df()*). This is used to build SQL query strings easily.
5. Minor boto3 helper to handle AWS connections: Set up but never used; the skeleton remains in place as it had some use in regards to AWS DynamoDB applications and it might be useful for AWS RDS.
6. *database\_demo.ipynb* in the */notebook* folder runs through how it can be used.

### spkly: A helper package written as extra credit for SIADS 521 that creates tiny sparklines for dataframes columns when using *spkly.display(df)*. It is meant to be extremely lightweight and allows to visualize a dataset quickly requiring to write matplotlib code.

## Argo

### Each netCDF file contained several variables encoded in different ways. After selecting the variables we needed for our analysis, we had to unmask and decode their data. Cleaned data was then loaded into a dataframe.

### The Atlantic data files cover a wide area (the Atlantic ocean!), but we did not need the full dataset for our analysis. We filtered the data to retain only points with a latitude between 38 to 59 and a longitude between -70 to -35 (corresponding to a large zone including the St Lawrence Gulf).

### The date column was created based on the netCDF file name, which contains the date the measures were published to the FTP site. There is a slight delay between publication date and measuring date as measures are reviewed by the scientific community. For the purpose of our project, we considered a week of delay would not significantly impact our analysis of parameter variation since water temperature and salinity are unlikely to drop or increase drastically in such a short period of time.

### Argo data stores one entry per float, but each float captures multiple data points as they rise to the surface. Once the dataframe was created, we had to unnest the parameters variables (temperature, temperature adjusted, salinity, salinity adjusted) to create one record for each measure.

### Parameters variables in argo data are meant to contain up to 2170 values. When a float only goes down to 1000m, only 1000 values are filled in the netCDF schema and the remaining values are set to 99999.00 (default value). We removed the default value to only keep recorded measures. Then, when the parameter (temperature or salinity) had been adjusted, we kept the adjusted record instead of the initial record as our value, taking advantage of the processing done by the scientific community before publication.

### Depth was not included in the dataset, but pressure in decibar was recorded. After confirming with a oceanography expert, we used the pressured in decibar as depth in m: Because a one-metre (three-foot) column of seawater produces a pressure of about one decibar (0.1 atmosphere), the pressure in decibars is approximately equal to the depth in metres.[[2]](#footnote-2)

### We then binned the data by depths to be measure trends across meaningful intervals (e.g. 0-100m, 100-200m).

### To avoid timeout and disconnections from the FTP site, each site was downloaded locally, pre-processed then wrote to the ocean\_data database. We used Python’s time.sleep() function to work around time outs and do all the upload at once. We also leveraged try/except capabilities to handle cases where we do not have data to write to the database from a file (we had two files that did not contain float data within our chosen latitude and longitude).

### An additional processing step was done after the data was uploaded to the ocean\_data table to check that the data was within the Gulf. The ETL and update process is stored in /notebooks/argo\_mapper.py, which contains both the shape lookup and the accompany upsert.

Our final pipeline for pre-processing the data included the following steps:

* Download file
* Process file:
  + unmask\_variables()
  + select\_columns()
  + if we had data to write (i.e. our dataframe is of length >= 1): unnest\_param(), depth\_bins()
  + insert\_table(‘ocean\_data’, file)

## GTSPP Data

### **How specifically did you need to manipulate the data?** Measurements were stored in netCDF format, an array based file format, which in turn were zipped and aggregated on a monthly basis. Each file contains a set of measurements (e.g., salinity, temperature) at a single latitude/longitudinal point at varying depths, along with quality measurements. The monthly files were retrieved from an FTP via a python script in the /notebooks folder (gtspp\_extraction.py) and dumped into the /data/gtspp folder. For the period 2009 to 2019, 132 tarballs were downloaded, at approximately 7.8 GB in size while still zipped.

### **How did you handle missing, incomplete, or incorrect data?** All measurements came with a quality measurement ranging from 0-9: 0 meant no quality control was perform, 1 meant the data was good, 2 meant ‘probably good,’ 3, ‘probably bad’, and so on and so forth. For quality control, only measurements that returned a quality measurement of 1 was used at the very end of the process.

### **How did you perform conversion or processing steps?** Each tarball was unpacked into a subfolder in the /data/gtspp folder. Each netCDF file was opened and its latitude and longitude point extracted. Because GTSPP files covered the entire Atlantic, each latitude/longitude point was checked via pyshp/shapely to see if it was within the Estuary and Gulf of St. Lawrence (the shapefile contains latitude/longitude coordinates of this shape). If the data point was within the boundaries, then the remaining variables were unpacked from the netcdf as masked arrays and then joined together into a dataframe. Later in the ETL process, certain variables had to be transformed to human readable data – e.g., measurement time, for example was stored in a day format most likely used to convert to date properly in Excel and therefore had to be converted via Timestamp() to datetime format.

### **What variables and steps did you use to join the data resources to perform your data analysis?** The initial ‘join’ across data sets was via the Estuary and Gulf of St. Lawrence shapefile – as long as they fell within this shapefile, all points were included. A secondary join to the fish population file was by year, to join fish population figures to temperature/salinity averages by year.

### **Briefly describe the workflow of your source code and what the main parts do.** The entirety of the ETL is stored in the gtspp\_extraction.py for GTSPP data. The process was:

#### Download tarfiles from the FTP to /gtspp folder.

#### Unpack each individual file to a subfile, iterate through the file contents checking for measurements done within the Estuary and Gulf of St. Lawrence. Once every file was examined within the tarfile, the folder was deleted in interests of space.

#### Qualifying measurements were unpacked and appended to a csv file named after the tarfile, stored in the data/gtspp/csv\_results file. Measurement times were converted to datetime and the results were inserted into the gtspp table in the AWS RDS database.

#### Data cleaning was done by pulling down the files from the gtspp table, grouping each measurement time into depth bins of 100 meters each and averaging the temperatures and salinity measurements within each bin. Final results were uploaded into the ocean\_data table.

#### Overall statistics: 132 tarfiles from FTP with 15.8 milllion latitude/longitude measurement files. Of the 12 million files examined, approximately 26,745 files with 1.6 million data points qualified and were uploaded to the gtspp database. After clean up and aggregation, 43,500 measurements were added to the final ocean\_data table.

### **What challenges did you encounter and how did you solve them?** Mostly spotchecking to make sure the cleanup was performing as expected – was the shapefile check truly returning accurately latitude/longitude points? This was solved simply by spotchecking via a visualization – mapped all 91k points in a single file in folium and spot checked the qualifying latitude/longitude points that were returned by the script. The other challenge was largely related to file cleanup as the script ran when processes didn’t give up control of a folder – this would throw an error and end the script (as intended – didn’t want to leave unpacked tarfiles around). This was mostly solved in a rather trivial way – adding time.sleep() to specific areas to allow processes time to close out.

### **How to access the data?** Raw data can be downloaded via the script from get\_data(), and sample data used in our visualization can be found in AWS RDS using SELECT \* from gtspp LIMIT 10;

## Department of Fisheries and Oceans (DFO) – Quebec Coastal Thermograph Network

### **How specifically did you need to manipulate the data?**Like Argo and GTSPP data, DFO data contains measurements at a single latitude/longitudinal point of temperature and salinity. Because the data is coastal in nature within the Estuary and Gulf of St. Lawrence, depth measurements are quite shallow (<100 meters).

### **How did you handle missing, incomplete, or incorrect data?** Rows with missing salinity or temperature values were simply dropped.

### **How did you perform conversion or processing steps?** The zipped file contains multiple csv files of different areas with measurements; the script runs through each and every one and aggregates them into a single dataframe table. The raw files were uploaded in aggregate to the dfo\_quebec table. From the aggregate, missing data values were discarded and then the remaining data set filtered for the appropriate time ranges (2009-2019).

### **What variables and steps did you use to join the two data resources to perform your data analysis?**Because this was already within the Estuary and Gulf of St Lawrence, a latitude/longitude check of the measurements was not necessary. The data was aggregated and averaged together with the other remaining data sources by year and then joined against the fish population dataset by year, however.

### **Briefly describe the workflow of your source code and what the main parts do.**

#### The entirety of the ETL is done in /notebooks/oceans\_fisheries\_nc\_extraction.py.

#### Extract\_raw() opens the zip file and opens every .csv file into a dataframe and renaming columns. Raw values are dumped in the dfo\_quebec table in the AWS RDS database.

#### Clean\_data() cleans the full dataframe, which has aggregated all the csv files together. Rows with missing values are discarded and then further filtered for the appropriate time. Values are binned by depth bins of approximately 100 meters each and averaged by the date of the timestamp before being uploaded to the ocean\_data database.

#### Overall statistics: Within the zip file: 42 csv files of approximately 8 million rows of data. After discarding missing measurements and filtering by date, 1.5 million rows were left. After cleanup and aggregation, approximately 150k rows of data were left and uploaded into ocean\_data.

### **What challenges did you encounter and how did you solve them?** Timestamps in this appear to be across a single day rather than there being one set of measurements per date. In order to aggregate by date and by datetime, I simply recasted the datetime to date before performing any aggregate calculations.

## Pelagic Fish Populations in the Estuary and Gulf of St. Lawrence

### **How specifically did you need to manipulate the data?**

### **How did you handle missing, incomplete, or incorrect data?**For this particular dataset, no datapoints were missing, but there were many points where population points were recorded as 0 in a given area. For the purposes of overall averages, we considered these null points.

### **How did you perform conversion or processing steps?**

#### Because the dataset was relatively small and simple

### **What variables and steps did you use to join the two data resources to perform your data analysis?**

#### Fish population data was averaged by year and joined against temperature/salinity averages by year.

### **Briefly describe the workflow of your source code and what the main parts do.**

#### The data exploration and etl process is stored in the /notebooks/pelagic\_analysis.ipynb file. The first couple of cells are exploring the data and putting it together to map and view. The spkly helper is used to get an idea about the variables.

#### The files were retrieved from the zipped file. Data is stored in 2 csv files, each containing fish population information for the north and south regions of the Gulf. The fields are the same for both, so the files were appended together. For fish names, the corresponding data dictionary was opened to remap the fish variable names to fish species name, and then further renamed to make it easier to search in a database.

#### Because each row reports a population count for all fish species at a specific longitude/latitude, the dataframe was melted down so every row would report a single value for a single species at a specific latitude/longitude (to make sql queries and dataframe filtering easier). The code for the melting, renaming, and push to database is in the cell after the folium map.

### **What challenges did you encounter and how did you solve them?**

## Estuary and Gulf of St. Lawrence Shape File

We used a shapefile as a filter/part of the cleaning process for our datasets and therefore did not process the shapefile itself.

One particular conversion of the shapefile was required, however – we converted the shapefile to its geojson equivalent, in order for our Dash visualization to use it for mapping. We used mapshaper.org for this conversion process and highly recommend it for very quick and easy conversions (and scaling!).

# Analysis and Visualization

4. Analysis and Visualization (25 points):

* A key goal of this project was bringing together two different data resources to answer an interesting question or find a new insight that could not have been answered with either data resource alone (which you summarized in part 1).  Now describe the analysis steps you performed on your combined dataset to address that goal/question. Be specific, and include references to key functions or parts of your code.

For our datasets, finding data within the appropriate location was very important, and most of our extract-transform-load routines were used in order to have working datasets that could be merged with each other.

For the float data (whether they be from Argo, GTSPP, or DFO Quebec):

### The raw datasets needed to be filtered to find data located within the Estuary and Gulf of St. Lawrence (“Gulf”). The shapefile was used as the filter at this point.

### Once we had this subset of measurements from the Gulf, then further averaging of temperature and salinity by depth was done, with depth bins of 100 meters by day.

### This final dataset was averaged by the year and returned, which in turn was joined with the fish population dataset.

Fish data needed very little additional cleanup, as it came in a single CSV file. Averages were calculated by the year per fish species. Because every latitude/longitudinal measurement has a fish population measurement (even if 0), we made the decision that 0 figures did not count towards population measurements and therefore did not count towards the averages.

* What didn't work, and why?

# What Didn’t Work

## The fish population dataset of choice with Argo.

Because Argo data generally captures ocean data and the Gulf location is sheltered away from the Atlantic, very few float data points were available for the Gulf. Of the 620,000 initial data points filtered from Argo data in the ocean\_data table, only 528 measurements were ultimately found in the Gulf.

## Postgres.

### This has less to do with postgres and more to do with the psycopg2 package for Postgres. Although a mature package, it is definitely less helpful, in some ways, to the pymysql package equivalent. It is always an unpleasant surprise to attempt to use a DictCursor (and later, a ‘RealDictCursor’), only to find out neither of these extras in psycopg2 actually return dicts in the first place. Certain other quirks are also a little more frustrating to deal with (e.g., upserts in Postgres require you to know where the collisions will be vs Insert On Duplicate Update)

# What interesting relationships or insights did you get from your analysis?

It’s one thing to hear reports that ocean temperatures were rising, it’s another thing to actually see it in the data. Average temperature measurements trended upward in the last ten or so years, regardless of depth.

* To summarize your findings, include at least one visualization (chart, plot, tag cloud, map or other graphic) that summarizes your analysis.

# Statement of Work

5. Statement of Work (0 points)

* You must include a statement that describes the contribution that each team member made to the project.

Sharon Sung wrote the helper files (db.py, spkly), uploaded most of the datafiles and mapped all the data to the Estuary and Gulf of St. Lawrence. She also wrote the initial Dash skeleton of the visualization and set up the requirements.txt file and containers for Heroku to host the Dash visualization.

Claire was the subject matter expert and led in setting up the AWS RDS database. She cleaned and uploaded the Argo data to the database, and worked on the advanced visualization setup for Dash.

Both looked for and acquired the datasets used for the project.

1. <https://www.dfo-mpo.gc.ca/stats/trade-commerce/world-mondial/export/wxv1517-eng.htm> [↑](#footnote-ref-1)
2. <https://www.britannica.com/science/seawater/Density-of-seawater-and-pressure> [↑](#footnote-ref-2)