**ETL/ELT (Snowflake and Python) Assignment**

This project is likely to take longer than the time we have to complete the project. I expect that each person spend about 10 hours on the assignment and complete as much as possible of part 2 given this time constraint.

**Part 1 – Introduction to Snowflake (individual)**

Read through the following interworks tutorials (if you are already familiar with Snowflake then you can skip some or all of these tutorials):

[Zero to Snowflake: Creating Your First Database](https://interworks.com/blog/chastie/2019/10/18/zero-to-snowflake-creating-your-first-database/)

[Zero to Snowflake: ETL vs. ELT](https://interworks.com/blog/chastie/2019/11/12/zero-to-snowflake-etl-or-elt/)

[Zero to Snowflake: Staging Explained](https://interworks.com/blog/chastie/2019/12/05/zero-to-snowflake-staging-explained/)

[Zero to Snowflake: Importing Data without Code via the User Interface](https://interworks.com/blog/chastie/2019/12/12/zero-to-snowflake-importing-data-without-code-via-the-user-interface/)

[Zero to Snowflake: Importing Data with Code via SnowSQL](https://interworks.com/blog/chastie/2019/12/20/zero-to-snowflake-importing-data-with-code-via-snowsql/)

[Zero to Snowflake: Python and Snowflake](https://interworks.com/blog/chastie/2020/01/02/zero-to-snowflake-python-and-snowflake/)

[Zero to Snowflake: Role-Based Security Access](https://interworks.com/blog/cmurray/2020/01/10/zero-to-snowflake-role-based-security-access/)

[Zero to Snowflake: Structured Data and Snowflake](https://interworks.com/blog/chastie/2020/01/15/zero-to-snowflake-structured-data-and-snowflake/)

[Zero to Snowflake: An Introduction to Semi-Structured JSON Data Formats](https://interworks.com/blog/chastie/2020/01/21/zero-to-snowflake-an-introduction-to-semi-structured-json-data-formats/)

[Zero to Snowflake: Loading and Querying Semi-Structured JSON Data](https://interworks.com/blog/chastie/2020/01/28/zero-to-snowflake-loading-and-querying-semi-structured-json-data/)

[Zero to Snowflake: Defining Virtual Warehouses](https://interworks.com/blog/2020/02/12/zero-to-snowflake-defining-virtual-warehouses/)

[Zero to Snowflake: Simple SQL Stored Procedures](https://interworks.com/blog/2020/02/18/zero-to-snowflake-simple-sql-stored-procedures/)

[Zero to Snowflake: Multi-Threaded Bulk Loading with Python](https://interworks.com/blog/2020/03/04/zero-to-snowflake-multi-threaded-bulk-loading-with-python/)

[Zero to Snowflake: Tips for Query Building in Snowflake](https://interworks.com/blog/2020/03/11/zero-to-snowflake-tips-for-query-building-in-snowflake/)

In addition to browsing Snowflake’s documentation as needed when working on part 2, the following sections of the documentation are recommended:

[Understanding Snowflake’s architecture (e.g., database vs. virtual warehouses)](https://docs.snowflake.com/en/user-guide/intro-key-concepts.html#snowflake-architecture)

[Loading Data into Snowflake](https://docs.snowflake.com/en/user-guide-data-load.html)

[Snowflake Python Connector](https://docs.snowflake.com/en/user-guide/python-connector.html)

For a deeper (and slower) introduction to Snowflake, consider spending additional time on the case and go through the [Data Warehousing Workshop](https://learn.snowflake.com/courses/course-v1:snowflake+ESS_DWW_101+2021/about) Snowflake learning track. If you decide to go through this learning track then do so before the interworks tutorials. The learning track is, however, fairly time consuming and the interwork tutorials cover most of what is explained in the learning track. In addition to covering some easy concepts very slowly, the course is slow because you have to complete multiple-choice questions and wait for 5 minutes if you answer a question incorrectly. This should not happen very often as most questions are easy, but when it does it slows you down… with this in mind, have some other work that you can switch to in case you miss a question. If you have sufficient time then I recommend going through this track.

**Part 2 – Python and Snowflake Case (group)**

This part of the assignment is a group project with three people in each group (up to four group members is allowed, but three is recommended). Submit one deliverable per group using the assignment drop box on the course website by the due date listed in the syllabus. Note that you do not have to (and that you are not expected to) complete all the requirements outlined below.

You will use Python (you can use any Python editor) and Snowflake to extract, load, and transform data. You should have as much of the work as possible done using Python (and SQL statements in Python), e.g., use Python (including SQL in Python) to create Snowflake warehouses, databases, schemas, and tables rather than doing so manually in Snowflake. Snowflake should, however, be used as much as possible for processing the work (e.g., instead of using Pandas to merge tables, merge the tables using a SQL statement written in Python that is sent to Snowflake. You, however, do not need to create an entire ETL process in terms of tasks that runs automatically or according to a schedule, process monitoring, etc.

You will be working with purchase order, supplier, invoice, and weather data that should be loaded into Snowflake. You should then create queries that calculate differences between invoice and purchase order amounts and see if the weather has anything to do with these differences. The data are stored in four different formats/sources:

* csv (comma delimited) – 41 files with monthly purchase order data (at the line item level)
* XML – one file with supplier invoice
* postgres – one table with supplier information (to prepare the data for the assignment, first download the supplier\_case.psql code from Canvas and then run the in the Postgres extension in VS Code)
* Snowflake Marketplace – weather data from Environment Data Atlas. The following will help when working with this data: <https://towardsdatascience.com/noaa-weather-data-in-snowflake-free-20e90ee916ed>

*Specific Tasks to Perform*

A. Extract and Load

Before completing the extract and load tasks below, examine the data and the other requirements. During extract and load, exclude columns that do not appear to be relevant (and that you do not think would be relevant for other problems, e.g., columns with all Null values or columns with only one value). Also convert datatypes as needed, e.g., Snowflake is more efficient when dates are stored as dates rather than varchar. You can combine multiple tasks below into single processing steps (or split a single task into multiple processing steps).

1. extract and load the 41 comma delimited purchases data files and form a single table of purchases data; preferably follow these [guidelines](https://docs.snowflake.com/en/user-guide/data-load-considerations-stage.html) when staging the files (this staging approach does not make sense for our data as the files are small, but it is good practice if you have more data and if the data is loaded over time)
2. create a calculated field that shows purchase order totals, i.e., for each order, sum the line item amounts (defined as ReceivedOuters \* ExpectedUnitPricePerOuter), and name this field POAmount
3. extract and load the supplier invoice XML data, flatten the data into a table where each row corresponds to a single invoice
4. join the purchases data from step 2 and the supplier invoices data from step 3 (only include matching rows); assuming that step 2 was completely correctly, you can assume the following relationships among the four tables:

Daily Weather

Supplier

Invoice

Purchase Order Header

1, 1 0, 1 0, M 1, 1 0, M 0, M

1. using the joined data from step 4, create a calculated field that shows the difference between AmountExcludingTax and POAmount, name this field invoiced\_vs\_quoted, and save the result as a materialized view named purchase\_orders\_and\_invoices
2. extract the supplier\_case data from postgres
3. connect to the Environment Data Atlas Marketplace data (do this manually inside Snowflake) and then extract weather data for each unique zip code in the supplier\_case table (suppliers can have the same zip code but you only need to extract weather data for each zip code once)
   1. the weather data does not contain zip codes but you can use the approach in <https://towardsdatascience.com/noaa-weather-data-in-snowflake-free-20e90ee916ed> to find weather stations closest to each zip code (only use one weather station per zip code);
   2. create a materialized view named supplier\_zip\_code\_weather that contains the unique zip codes (PostalPostalCode) from the supplier data, date, and daily high temperatures, i.e., the view should have three columns (zip code, date, and high temperature) and one row per day and unique supplier zip code
4. join purchase\_orders\_and\_invoices, supplier\_case, and supplier\_weather based on zip codes and the transaction date. Only include transactions that have matching temperature readings