|  |  |
| --- | --- |
| B2B Warehouse and ETL  A Toptal Screening Project | Abstract  Technical Design Document for the Data Warehouse, ETL Framework and Reporting View.  Arun Kumar Basnet  2022-07-03 |

Contents

[Introduction 2](#_Toc107872600)

[Purpose of document 2](#_Toc107872601)

[Requirements 2](#_Toc107872602)

[ETL Flow 3](#_Toc107872603)

[ETL Flow Diagram 3](#_Toc107872604)

[Database Architecture 5](#_Toc107872605)

[B2B Platform Source System: 5](#_Toc107872606)

[Target Data Mart: 5](#_Toc107872607)

[ETL Framework 7](#_Toc107872608)

[ETL Framework Folder Structure 7](#_Toc107872609)

[Variable Module 8](#_Toc107872610)

[Database Module 8](#_Toc107872611)

[Log Module 9](#_Toc107872612)

[Batch Monitor Module 9](#_Toc107872613)

[Scripts for extraction and load 10](#_Toc107872614)

[Web Log Framework 11](#_Toc107872615)

[Report Source Setup 12](#_Toc107872616)

[Installation Guide 13](#_Toc107872617)

[User and Security 14](#_Toc107872618)

[Shortcoming and Future Enhancements 15](#_Toc107872619)

[Assumptions 15](#_Toc107872620)

# Introduction

## Purpose of document

This document contains the details regarding the technical design of the B2B web platform Data Warehouse Design and ETL flow as part of Toptal Screening Project

This document will cover:

1. ETL Flow
2. B2B Data Model Definition
3. ETL Framework
4. Batch Monitoring and Recovery and Batch Metadata
5. Weblog generation and consumption by DataMart
6. Views for the Reporting

# Requirements

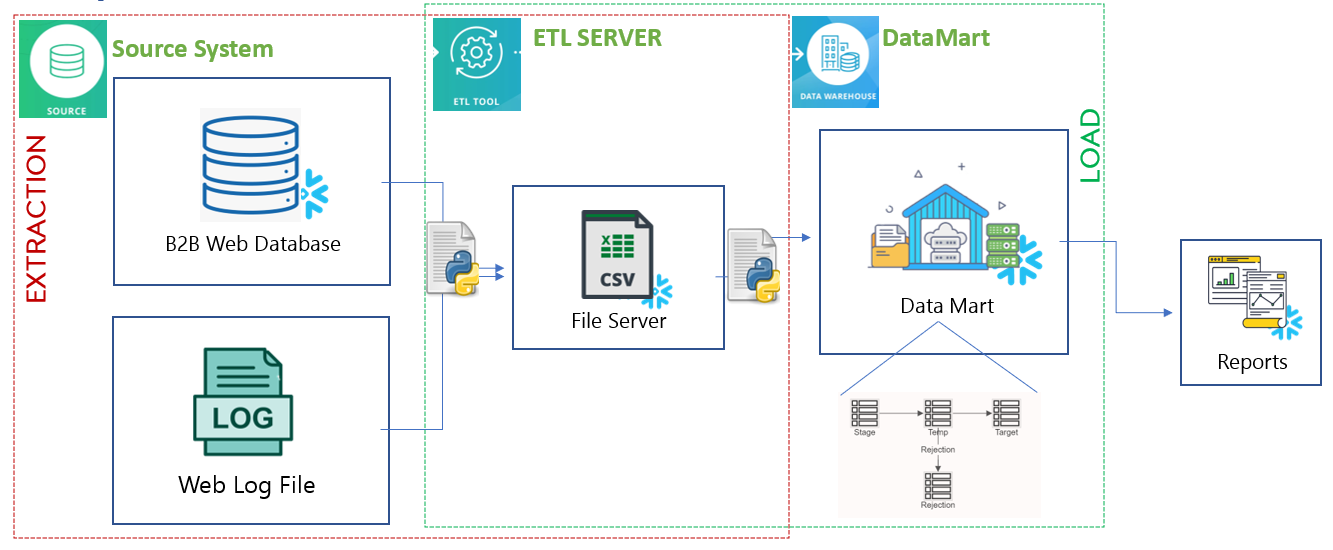
Below are the summarizations of the Requirements:

1. Data Warehouse has 2 Major Data sources
   1. B2B platform database
   2. Log data from the webserver
2. Solution to be provided:
   1. Database implementation with generated data for the B2B database  
      source
   2. Weblog generated via script in a language of your choosing
   3. A target database which represents the data warehouse or data mart, you can choose relation or NoSQL solution
   4. ETL/ELT\* process with transformations that will:  
      • Fill the initial load of the target datastore  
      • Be restart able if the jobs or sub-job fails  
      • Handle erroneous data  
      • Track ETL/ELT metadata (when did the load start, break, finish)  
      • Transform the data into a readable data format for reporting  
      • Demonstrate the ability to transform large data size
   5. Data Mart as foundation of following answers:
      1. What are the most popular used devices for B2B clients (top 5)?
      2. What are the most popular products in the country from which most users log into?
      3. All sales of B2B platform displayed monthly for the last year.

# ETL Flow

The section will explain the ETL framework and the data flow in detail.

## ETL Flow Diagram

*fig: ETL Flow Diagram*

The above figure illustrated the entire ETL flow. There are 3 major components in the ETL system.

1. **Source System:** The source system consists two components.
   1. First one is the Snowflake Database for the B2B platform. It is the transactional Database for the B2B platform to maintain the records for the orders and the attributes related to orders.
   2. Weblog file is also a source of the Data Mart which holds the data regarding the client logins and details into the web application.
2. **ETL Server:** The server holds the ETL scripts and data file which has been extracted from the Source system. The data file is generated by the extraction ETL scripts querying into the source system and they are finally consumed by the Data Mart with the aid of load python scripts. Hence the components of the ETL server are:
   1. **Extraction Script**
   2. **Framework libraries**
   3. **Data file**
   4. **Load Scripts**
3. **Data Mart:**  The Data Mart holds the data form the B2B platform and also holds the data form web log to perform the analysis and Reporting: The multiple component in the Data Warehouse are:
   1. **STAGE Schema:** It holds the extracted data from the source system and is output of the extraction part. The data in Stage is then used to populate the target tables.
   2. **TEMP Schema:** It is the schema where the multiple ETL process like, constrain checks, dimension lookup is done and the Data form the STAGE is transformed and loaded into this schema.
   3. **REJ Schema:** This schema is the schema to hold the data Rejected from the main Data flow during the Batch execution.
   4. **Target Schema:** This schema holds the Data Mart and the Data from the sources are finally loaded into this schema.
   5. **Reporting Schema:** This schema holds the views and the other Database objects required for the Reporting purpose.

# Database Architecture

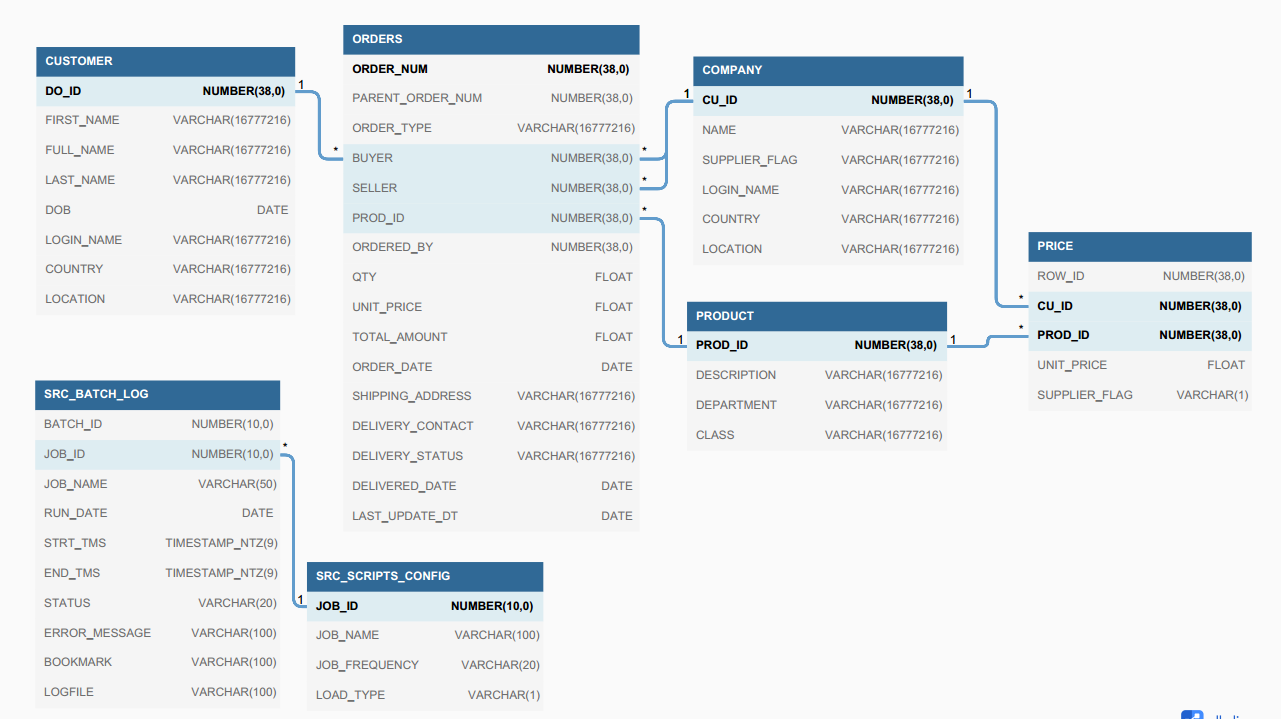
## B2B Platform Source System:

Following are the list of Source tables in the Source System:

* PRODUCT
* CUSTOMER
* COMPANY
* PRICE
* ORDERS

Following are the list of tables used for extraction batch monitoring and storing extraction batch metadata

* SRC\_BATCH\_LOG
* SRC\_SCRIPTS\_CONFIG



*fig: Source Database ER Diagram*

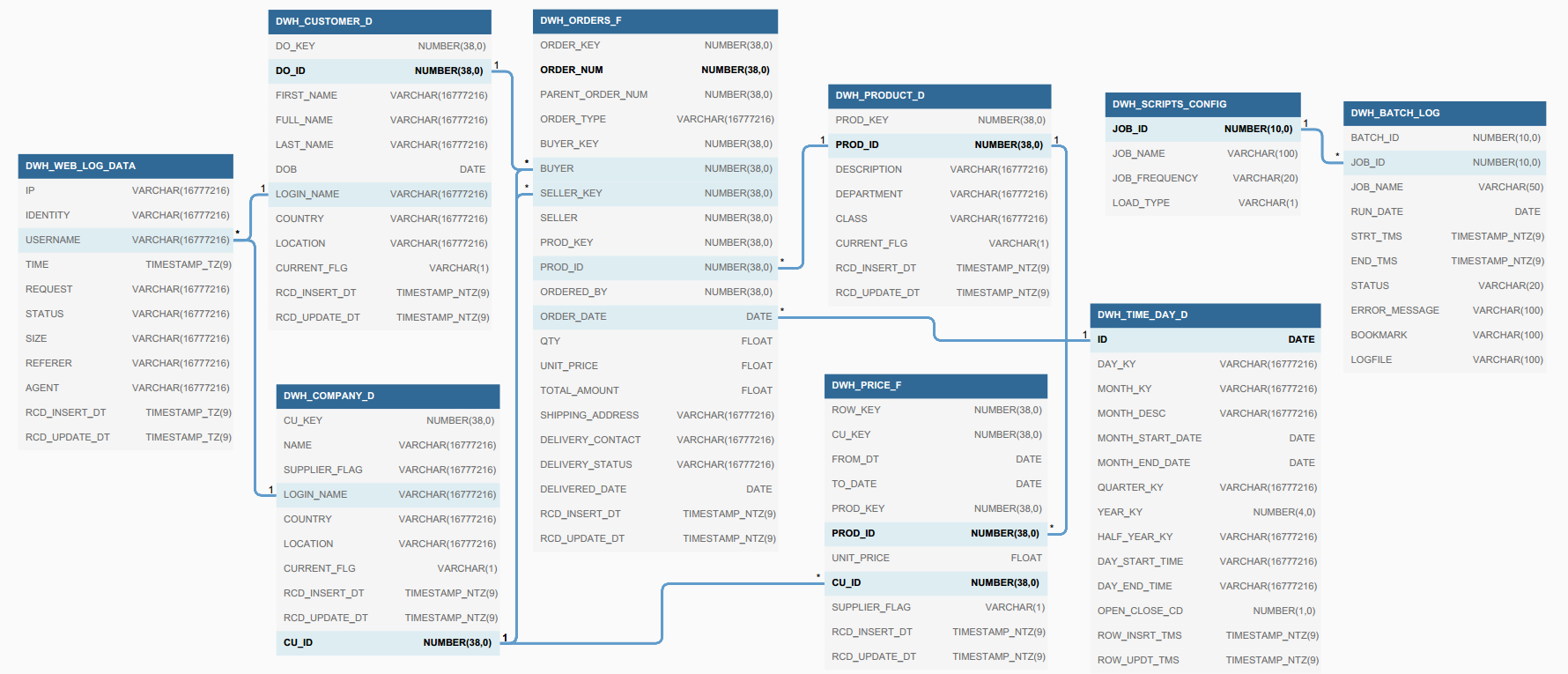
## Target Data Mart:

Following are the list of tables populated in Data Mart for the B2B Platform

* DWH\_PRODUCT\_D
* DWH\_CUSTOMER\_D
* DWH\_COMPANY\_D
* DWH\_PRICE\_F
* DWH\_ORDERS\_F
* DWH\_WEB\_LOG\_DATA
* DWH\_TIME\_DAY\_D

Following are the list of tables used for load batch monitoring and storing load batch metadata

* DWH\_BATCH\_LOG
* DWH\_SCRIPTS\_CONFIG



*fig: Datamart tables and ER Diagram*

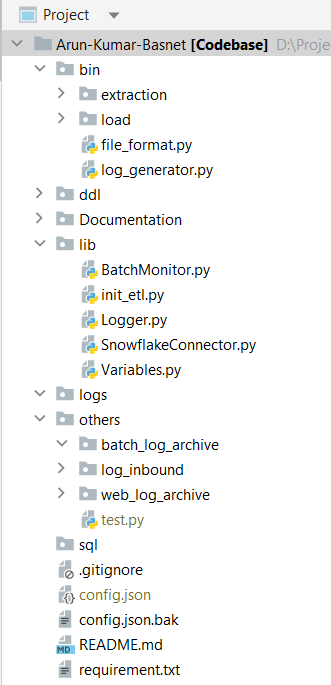
# ETL Framework

As part of ETL framework, a python-based framework has been designed. There are multiple class for each library module. The different module present in the framework are:

1. Variable module to maintain the variables throughout the project
2. Database module to control the database operations
3. Log Module to maintain the logging in the project
4. Batch Monitor module to handle the batch and maintain the batch integrity

Below are the folders and the library scripts from the project:

## ETL Framework Folder Structure

Below are the folder structures of the ETL framework and details about its contents:

* **bin**: Directory for load and extraction python scripts.
  + - **\extraction:** Contains all the extraction scripts
    - **\load:** Contains all the load scripts
* **ddl**: Directory for the DDL used for Database objects.
* **Documentation**: Directory for the project documents i.e Power Point for Project Demo and Technical Design Document.
* **lib**: Library Directory for the ETL framework. It contains the scripts for all the framework modules. (which will be discussed later)
* **logs**: Directory for the batch log for latest run.
* **others**: Directory to load all the log file except for the current batch execution log
  + - **Batch\_og\_archive:** Directory to store logs from previous batch executions.
    - **log\_inbound:** Inbound directory of the weblog file for the batch
    - **Web\_log\_archive**: Directory for the past log files processed by batch
* **sql**: Contains the sql files used to perform initial seed on the various tables and setup the Database.
* **config.json**: file with all the credentials and environment/project variables.
* **config.file.bak**: backup/template of config.json file
* **requirement.txt**: Project Requirement file

fig *fig: Project Directory Structure*

### Variable Module

This module controls the usecase of various environment variables, environment credentials and other project variables. All the environment variables are stored on the config.json file and is used in the various modules and scripts.

**Configuration file**: ../config.json

The configuration file includes:

* Database connections, credentials, and schema details
* Different Directory paths and project variables

#### Module Detail

|  |  |
| --- | --- |
| **Module Name** | Variables |
| **Functionality** | Initialize the project/environment variables maintains the value of the variables through getter and setters method |
| **Module methods** | * *\_\_init\_\_()* * *get()* * *set()* * *exists()* |

### Database Module

The database module contains the usecase and implementation of various database operations of the Snowflake Database (i.e Project Database). It handles the database connection, query executions and result evaluation from the query.

#### Module Detail

|  |  |
| --- | --- |
| **Module Name** | SnowflakeConnector |
| **Functionality** | Database connection creation and close Execute query Handle the Results returned from query |
| **Module methods** | * *\_\_init\_\_()* * *startConnection()* * *endConnection()* * *execute\_query()* * *get\_value()* * *get\_data()* * *truncate\_table()* * *create\_data\_file()* * *create\_data\_file\_delta()* * *load\_stage\_from\_file()* * *load\_table()* * *load\_stage\_from\_log()* |

### Log Module

This module handles all the logging mechanism during the job executions. It helps create each log file for each script executed and logs all the query and rows affected by the query in different steps from script along with the job status and the error message.

#### Module Detail

|  |  |
| --- | --- |
| **Module Name** | Logger |
| **Functionality** | Logging the job status and details from the script execution |
| **Module methods** | * *\_\_init\_\_()* * *log\_message()* * *close()* |

### Batch Monitor Module

This module handles all the batch execution flow control and error handling of the batch flow. It is also the part of storing the job metadata and the batch metadata regarding the status of jobs execution.

Job Metadata: It is stored in the SCRIPT\_CONFIG table in source and target schema respectively for the extraction and load batch separately. It holds the JOB\_ID which is identifying id for each job, along with script name and its load type.

Batch Metadata: It is stored in BATCH\_LOG table in source and target same as SCRIPT\_CONFIG. It holds the status of the job (i.e Running, ERROR, Restart, Restarted, Completed) along with the various bookmark in the script to handle the re run of the jobs.

#### Module Detail

|  |  |
| --- | --- |
| **Module Name** | BatchMonitor |
| **Functionality** | Logging the job status and details from the script execution |
| **Module methods** | * *\_\_init\_\_()* * *get\_job\_id()* * *get\_batch\_id()* * *get\_last\_status()* * *set\_bookmark()* * *get\_last\_bookmark()* * *start\_audit()* * *audit\_complete()* * *script\_error()* |

#### Batch Recovery and Bookmarking

1. Batch monitoring can be done by looking into the tables
   1. For Source: SOURCE.SRC\_BATCH\_LOG
   2. For Target: TARGET.DWH\_BATCH\_LOG
2. In case of failure,
   1. Note the batch ID
   2. Note the Job ID
   3. Analyze the bug and fix the issues
   4. In case of Batch restart from point of failure, run following query
      1. UPDATE DWH\_BATCH\_LOG

SET STATUS=’RESTART’

WHERE JOB\_ID = <Failed Job ID> AND BATCH\_ID = <Failed Batch ID>

### Scripts for extraction and load

Following are the list of scripts used in the ETL framework

|  |  |  |
| --- | --- | --- |
| **S.N** | **Module** | **Script\_Name** |
| **1** | Extraction | extraction\_batch\_start.py |
| **2** | product\_ex.py |
| **3** | company\_ex.py |
| **4** | customer\_ex.py |
| **5** | price\_ex.py |
| **6** | order\_ex.py |
| **7** | order\_delta\_ex.py |
| **8** | web\_log\_ex.py |
| **9** | extraction\_batch\_end.py |
| **10** | Load | load\_batch\_start.py |
| **11** | product\_ld.py |
| **12** | company\_ld.py |
| **13** | customer\_ld.py |
| **14** | price\_ld.py |
| **15** | order\_ld.py |
| **16** | web\_log\_ld.py |
| **17** | load\_batch\_end.py |
| **18** | Others | file\_formate.py |
| **19** | log\_generator.py |

# Web Log Framework

This module is implemented on the log\_generator.py file for the log generation and the load and extraction are implemented in the same way as others.  
The dummy log is created on one of the directories (**inbound\_folder)** and the log is consumed by the extraction script to load into the staging area.

The staged log file is then sourced by the load script and then the target table **DWH\_WEB\_LOG\_DATA** is populated.   
the log files are cleaned for the **inbound\_folder** to archived folder to stop the loaded log from being loaded again during next execution.

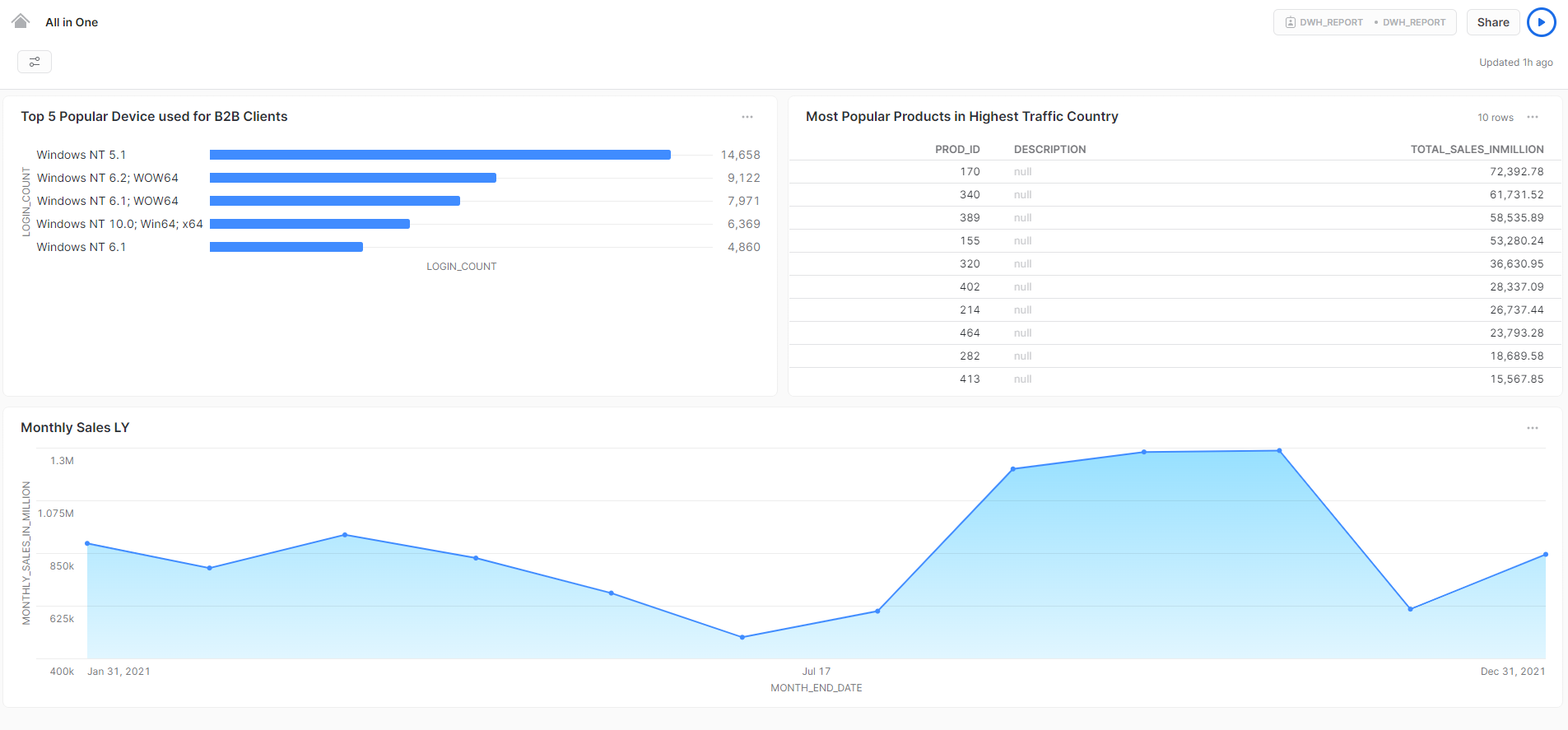
# Report Source Setup

The views in the Reporting Schema are created as part of data preparation for the reporting purpose.  
The logic and the view generation queries are present in the ddl folder inside the file **Views\_SQL.sql**.

The Views for the Reporting Purpose for each requirement are:

1. DWH\_V\_DEVICES\_USUAGE\_F
2. DWH\_V\_POPULAR\_PRODUCT\_F
3. DWH\_V\_MONTHLY\_SALES\_LY\_F

As part of the platform creation on the Database for Reporting purpose I have also created **Snowflake dashboard.** *Few data seem off will get synced once Batch is executed with proper data.*



*Fig: Snowflake Dashboard for the Requirement*

# Installation Guide

1. Copy the Development folder in your system in any directory as required
2. Install Python 3.7 in your system
3. Install all required libraries using the requirements.txt file in folder Development as follows
   1. Open cmd and go to the folder where the Development folder resides
   2. Pip install -r requirements.txt
4. Set up the database, Target database in your Snowflake database.
   1. Run all SQL in file ddl
5. There are optional scripts inside sql Directory
6. Rename file config.json.bak to config.json and open the file in your text editor and add paths, database paramters etc in the file
7. All Extraction scripts and Load Scripts are currently run sequentially using init\_py

Comment all the methods and only un comment *#file\_format.create\_file\_format()* line in script and run to test the database connection and setup test.

1. Logs generated for all scripts will be present in log folder
2. The archived logs are available in the archived\_logs folder.

# User and Security

All the objects are created by SYSADMIN hence only the user with SYSADMIN role can modify the database Objects.

There are 3 user Roles created in the Data Mart for the purpose of access Management.

1. DWH\_BATCH: It is the role to execute the batch. It also has the SYSADMIN Role hence it can modify and operate on the Database object.
2. DWH\_USER: It is the role for the individual user which has limited access on database and limited edit access on the Data Mart. User with this role are created to user who need to perform various Data operation like. Data fix, Prod Deployments and others.
3. DWH\_REPORT: It is the role for the individual user which has only select access to the Data Mart Views that are used for reporting purpose. The Reporting user is also created with this role to access the database view from the Reporting server.

# Shortcoming and Future Enhancements

1. Scheduling of the jobs. The jobs can be scheduled using various scheduling tools like ADF, Automic etc. As of now we are just triggering the job from the wrapper class. Later we can use scheduler to trigger the batch as well as notify the users when running and with status.
2. Rejection handling: As of now the rejected data are only loaded to rejection table and no notification are generated. In future we can validate Rejection table and generate and notification to user regarding data Rejection in case of Bad data.
3. Implementation of multiple Database: As of now we system that only supports Snowflake Database. In future we can create modules to support multiple Databases to have cross platform data flow.
4. Stage File: As of now we have only used the Snowflake Internal File Stage for Data File storage. We can use multiple cloud implementation storage Device like AWS, Azure, Google Cloud Platform to access it from multiple Database at once.

# Assumptions

1. Source system have same table for the Supplier and Company as it both involves in selling. The differentiator between Supplier and Company is SUPPLIER\_FLAG column where Supplier has ‘Y’ value in this field.
2. There is an option for a Company to buy a order to Directly deliver to customer in B2B platform.  
   Since Company buying price and selling price are different there will be two transaction initiated for this cause.
   1. First the Company Order will be placed to supplier
   2. Secondly a child transaction is created as Company Sales to customer with the Company selling price.
   3. The transaction is separated by ORDER\_TYPE column. If it has ‘COMPANY\_PO’ then it is a Purchase by company from Supplier.  
      If it has value ‘COMPANY\_SALES’ then it is sale by Company to customer.