TPC-DI Benchmarking with Postgres Database

Data Warehouses (INFO-H419)

Erasmus Mundus Joint Master's Degree
in
Big Data Management and Analytics

by

Ahmad (ahmad@ulb.be)
Rishika Gupta (rishika.gupta@ulb.be)
Chidiebere Ogbuchi (chidiebere.ogbuchi@ulb.be)
Mir Wise Khan (mir.khan@ulb.be)

under the guidance of

Prof. Esteban Zimanyi

École polytechnique de Bruxelles
Université Libre de Bruxelles
December 2022

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

CTE	Common Table Expressions
DAG	Directed Acyclic Graphs
DB	Database
DI	Data Integration
DBMS	Database Management Systems
DS	Decision Support
GB	Gigabytes
ETL	Extracting, Transforming, and Loading
MB	Megabytes
MS	Milli-seconds
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
ORDBMS	Object Relational Database Management System
RDBMS	Relational Database Management Systems
SF	Scale Factor
SQL	Structured Query Language
SUT	System Under Test
TPC	Transactional Processing Council
TPC-DI	Transactional Processing Council's Data Integration Benchmark

Abstract

TPC Benchmark™ DI (TPC-DI) is a profound performance test of tools that transfers and integrate volumes of data between several systems. In previous years, these tools were usually called ETL tools; since they support the procedure of Extracting, Transforming, and Loading (ETL) data from operational systems and synchronizing them to a decision support system. However, in recent times, this name has been replaced by a more comprehensive term - Data Integration (DI). DI describes the process of extracting and amalgamating different data types formats from various sources, transforming them into a more unified data model representation as well as loading the outputs into a suitable data warehouse in the context of various scenarios and use cases. Moreover, Data Integration (DI) tools are usually made available by various distributing vendors but there hasn't been a standard way to evaluate and compare performances. The TPC-DI provides a benchmark tool that includes data characterizing - an extract from an On-Line Transaction Processing (OLTP) system being manipulated along with data from other supporting data sources (comprising relational and hierarchical structures), and loaded into a data store. In addition, the schemas and transformation rules have been formed to depict and represent the requirements of a modern data integration system. In this report, we establish important aspects of this workload and outline the business modeling systems and performance aspects adopted into this benchmark. It captures the essential complexities (such as increase in data volume, performance, metrics etc) that are characterized in the industry standards. In general, the benchmarks model very essential aspects of a typical data integration system which entails transformation of transactional data into a data warehouse as well as synchronization, manipulation and maintenance processes of data structures.

Chapter 1

Introduction

This section of the report provides an overview of the benchmarking process following the TPC Benchmark™ DI (TPC-DI) v.1.1.0 specification as an implementation guide.

1.1 Overview

In this project, we implement the TPC Benchmark™ DI (TPC-DI) on a preferable data integration tool - Apache Airflow, Pentaho Data Integration, Snaplogic, Oracle Data integrator, SQL Server Integration Services, Talend Data Studio, SQL scripts, etc which loads the data warehouse on a suitable Database Management System (DBMS) such as PostgreSQL, SQL Server, Redis, Oracle, etc. The benchmark is executed using different scale factors (SF) which vary relatively in the size of the data warehouse. Performance is evaluated and compared based on a reference SF of data volume.

The project entails a group of four members and presents a well-detailed report of the essential aspects of the benchmarking process. For this study and report, we implement the TPC-DI benchmarking on Apache Airflow as the preferred data integration tool as well as PostgreSQL as the befitting choice on DBMS.

1.2 Aim and Objectives

The major aim of the TPC-DI benchmark is to extract, transform, and load (ETL) data processed from an On-line Transaction Processing (OLTP) system and various sources of data into a data warehouse loaded on a selected DBMS using a preferred data integration tool. The other objectives include:

- Evaluate benchmarking performance and analyze the results.
- Perform the TPC-DI benchmark to better understand the data integration process in order to help us compare and choose the best available tools for our businesses.

1.3 Tools Used

The tools installed and utilized to perform the benchmark operation are summarized in table 1.1.

Tool	Version	Description
TPC-DI standard benchmark tool	3.2.0	The official tools set offered by TPC-DI for data generation, query generation and an answer set to compare results.
PostgreSQL	14.0	Open-source PostgreSQL relational database for data warehouse.
Apache Airflow	2.4	Open-source workflow management platform for data integration pipelines.
Docker Desktop	20.10.17	Docker was used to run Ubuntu to generate TPC-DI data and install Apache airflow.
Visual Studio Code	1.74.2	Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
Python	3.10.5	Python was used as the programming language for running scripts on Apache airflow as well as for visualizing results.
GitHub	2.38.1	GitHub Desktop was used to share code files as well as images conveniently with the team members.

Table 1.1: Tools Used for TPC-DI Benchmarking with PostgreSQL

1.4 Limitations and Justifications

Given that the entire project was implemented on a local machine, there was an associated certain limit on the resources - tools that could be used - thereby preventing scaling to higher benchmarks. It is definitely possible for us to procure cloud-based services like Google Cloud, and Amazon Azure and implement the data warehouse into Postgres in their environments. Also, with the provision of more cores as well as storage, we could have certainly benchmarked until 100 GB's least. However, despite the complexities related to local resources, we tried to benchmark at least 30 GB volume of data.

Chapter 2

Technology Fundamentals

This chapter briefly describes the underlying tools and technology used to perform the database benchmark.

2.1 PostgreSQL

PostgreSQL is a free enterprise open-source object relational database management system (ORDBMS) akin to a relational database, bar that it is object-oriented such that it offers classes and objects models including inheritance in query-language and database schemas [Bartolini et al., 2017]. Initially developed at the University of California, Berkeley by the Database Research Team of the computer science department, is now adapted and developed by a vast horde of contributory developers. It provides a huge diversity of support languages ranging from C, Python, PHP, C++, Perl and Java amongst others that permits a variety selection of constructs that can proffer solutions to problems [The PostgreSQL Global Development Group, 2022]. In benchmarks, PostgreSQL is fast and provides similar excellent performance as when compared to other proprietary and open source databases [Obe & Hsu, 2017]. Also, it shoulders a huge part of the SQL standard and offers advanced present-day features such as but not limited to:

- Complex queries
- Transactional integrity Triggers
- Multiversion concurrency control
- Foreign keys
- Updatable views [Matthew & Stones, 2005]

Furthermore, PostgreSQL allows user extension in several ways such as adding and connecting new:

- operators
- data types
- index methods
- procedural languages
- aggregate functions

- functions

As as a result of the open license, PostgreSQL can be utilized, distributed & modified by any individuals without charge for any reason [The PostgreSQL Global Development Group, 2022].

2.1.1 Why PostgreSQL

PostgreSQL has numerous benefits including:

- Outstanding SQL standards compliance.
- Client-server architectural structure.
- High degree of synchronous interface and design where users don't interfere with each other.
- High extent of configuration and extensions for several kinds of applications
- Outstanding scalability and performance with high-level tuning and optimization features.
- Excellent support for different types of data formats including relational, post-relational (arrays, nested relations via record types) documents (JSON, CSV and XML), and dictionary keys/values.

In addition, the PostgreSQL system is a robust and high-quality tool with rich documentation, maintainability, interoperability and high availability. It requires low maintenance as well as provides excellent performance, security and compatibility for major operating systems on both enterprise and embedded usage [Bartolini et al., 2017]. In this project, PostgreSQL shall be used as a database management system for implementing the TPC-DI benchmark.

2.2 Apache Airflow

Apache Airflow is an open-source platform for developing, scheduling, managing and monitoring data engineering workflows. It's a batch-oriented pipeline management tool that enables users to build workflows connecting to most technologies. Airflow is built in a modular way, possessing an extensible python framework and a web interface that helps oversee the state of workflows. Also, it can be deployed in several ways, ranging from simple command-line processes on a PC to a distributed setup to support large complex data integration pipelines. Workflows are majorly created via python scripts and are designed under the "Principle of configuration" as code. Unlike other platforms that utilize markup languages like XML, implementing python allows users to import various libraries and packages that enable easy creation and processing of workflows. Furthermore, Airflow utilizes directed acyclic graphs (DAGs) to control workflow planning and coordination. These DAGs can be executed either on an explicate schedule (e.g monthly, daily, hourly, etc) or external event triggers (e.g. a file appearing in Apache Hive). Airflow DAGs can be written in one python file unlike previous DAG-based schedulers like Azkaban, Oozle, etc which tend to depend on several configuration files and file system trees to generate a DAG. Also, tasks and dependencies can be defined in python and written via Apache Airflow core scheduler functionality which can be extended by installing additional packages called ‘Providers’. These providers can contain operators, hooks, sensors, and transfer operators to interact with multiple external systems. [Apache Software Foundation. (2022, September 18).]

2.2.1 Why Apache Airflow?

Airflow workflows as code are very flexible (workflow parameterization is built-in leveraging the Jinja templating engine), extensible (operators can connect to several environments) and dynamic (pipelines can be dynamically generated since it uses python). Also, as a batch-oriented orchestration tool for workflows, it can easily be programmed to execute dags at different schedules provided these are clearly defined start, end, and interval times. Since Airflow uses a python framework, it is great for coding over clicking and offers other benefits like:

- Workflows can be rolled back to the previous version in case of implementation error (version control).
- It can be simultaneously developed by multiple contributors.
- Functionalities can be validated through tests and errors can be easily fixed.
- It allows easy definition and creation of complex tasks and pipelines.
- Easy inspection of logs and management of tasks.
- It is generalizable i.e. it ensures developers can work on components developed, tested, and used by many other companies peers around the world.

In this project, we utilize Apache Airflow as a workflow management platform to extract, transform and load data warehouse into Postgres.

2.3 Other Tools

2.3.1 Docker Desktop

This application allows for the transformation and optimization of workflows by allowing users to connect to a collection of pre-built developer tools and systems from the Docker Extension Marketplace. It allows for the creation and sharing of customized tools with other team members in its dev environment.

Also, Docker provides a fast way to build solutions and projects in containers as well as offers flexible control, secure access and management of container images [Install Docker Desktop, 2022]. Docker was used to run the latest version of the TPC-DI tool on a centOs container for the purpose of generating the data from the OLTP system and other sources as well as used to build and run the Apache Airflow tool.

2.3.2 Visual Studio Code

Visual Studio Code is a compact but extremely powerful source code editor that runs on computer desktops and is accessible on macOS, Windows and Linux operating systems. It has a built-in interface standard for Typescript, Node.js and JavaScript as well as a offers a wide array of extensions for other programming languages (Python, C++, C, Java, etc.). In action, visual studio code has an impressive UX and allows the customization of workflows [Visual Studio Code, 2022]. This tool was useful in the project for building and verifying the entire scripts for the workflows.

2.3.3 Python Interpreter

Python is a general-purpose programming language that allows quick working and integration of systems effectively. This high-level language is dynamically input and supports procedural, functional and object-oriented programmed. It can be compiled using an interactive development emulator [Python, 2022]. For this benchmark project, VS Code with Conda extension was used to create and compile python scripts. Python allowed us to cleanse and transform the initial load data generated and push them into the database. Also, it was used to transform the generated data for effective loading into the data warehouse.

Chapter 3

Benchmarking and Implementation

3.1 Introduction to Benchmarking

Benchmarking involves comparing performance indicators and processes to industry best practices usually in relation to time, quality and cost metrics. It is generally used to estimate similarities and contrast between a specific performance metric. In databases, benchmarking may be difficult especially if it follows different relational and object model approaches. Despite this fact, organizations and individuals still experience the challenge of selecting a suitable DBMS platform for implementing models, as most databases offer many similar features on many fronts. However, performance is a great differentiator when choosing between available databases for data integration. Leveraging benchmarks can be used in recommending a suitable selection of a given technology [Tortosa, 2020]. In other words, benchmarking a database is the process of performing well-defined tests on that particular database for the purpose of evaluating its performance [Kabangu, 2009]. The performance evaluation can help an organization decide if the particular choice of database can meet the business needs of the organization in the long run.

3.2 TPC-DI

TPC Benchmark™ DI (TPC-DI) is basically a data integration benchmarking model that caters to the relevant areas of a simple data integration structure, constituting the entire process of ETL along with data maintenance. The TPC-DI benchmark offers a comprehensive data-integration system that represents a typical appraisal of the System Under the Test's (SUT) performance model. Generally, it is an archetypal data integration platform that are similar to industry standards and are characterized by:

- The processing and injecting of huge volumes of data
- Blend of transformation and manipulation types including error checking, aggregation operations, surrogate key lookups, data type conversions, data modification & updates, etc.
- Historical loading and incremental updates of a destination Data Warehouse using the transformed data

- Consistency needs guaranteeing that the integration process outputs are reliable, precise and accurate data
- Multiple sources of data with various formats
- Multiple data structures with varied data types, features, tables and inter-table relationships

The TPC-DI operations are modeled as follows:

- Source data is generated using TPC guide code. The data is made available in flat files, akin to the output of other extraction tools.
- Transformation of the data starts with the System Under Test (SUT) reading the Source Data.
- The transformations check the accuracy of the Source Data and properly structure the data for loading into a Data Warehouse.
- The process culminates when all Source Data has been transformed and pushed into the assigned data Warehouse.

In addition, a benchmark result assesses various aspects including the extract, transform and load (historical) time, incremental loads (data maintenance) in an isolated user level as well as in multiple user levels evaluation for a designated hardware, data processing, and operating system setting under a monitored and controlled decision support workload [Transaction Processing Performance Council (TPC), 2021].

3.2.1 DIGEN Generation

The data is generated from the TPC-DI tool in the following way:

- Requirement: DIGen - Data Generation utility v1.1.0 used for generating source data for the TPC-DI benchmark.
- Dependencies:
  - Docker - used to deploy the CentOs container
  - Requires Java SE 7 or above: java-1.8.0-openjdk-devel-1:1.8.0.312.b07-2.el8_5.x86_64
  - PDGF which is located in the same directory as the DIGen
- Command line Usage:
  - Access container image:

    ```
    $ docker exec -it <container Id: xxx> <command line: bin/bash>
    ```

  - Generate the files using java:

    ```
    $ java -jar DIGen.jar -sf <scalefactor: 3,5,10...>
           -o <directory: cd:...>
    ```

Scale factors varied and generated in several file formats.

3.2.2 DIGEN Source Data Models

The TPC-DI benchmark tool generated the following source data models - Historical Load, Incremental Updates and Automated Auditing.

- Historical Load - This encompasses various transformations other than the Incremental Updates. Destination tables are originally empty and being loaded with new data, and the source files have varying ordering properties.
- Incremental Updates - These are different from the Historical load. The resulting files from the OLTP database are modeled as CDC extracts, which show the changes in the table data since the last extract. There are two Incremental Update phases in this benchmark ensuring the process is consistent and repeatable.
- Automated Auditing - After completion of the other source models , the automated audit queries the Data Warehouse to perform extensive tests on the resulting data and creates a simple report of the results. This validates the accuracy of the integration.

However, for this study, we implement only the Historical load which contains the following and is integrated in no particular order:

1. DimDate
2. DimTime
3. StatusType
4. TaxRate
5. TradeType
6. DimBroker
7. DimCompany
8. DimCustomer
9. DimAccount
10. DimSecurity
11. DimTrade
12. FactCashBalances
13. FactMarketHistory
14. FactWatches
15. Industry
16. Financial
17. Prospect

3.3 Data Integration Phases

As aforementioned, data integration essentially involves data warehouse initialization, extraction of data from several data sources (in this case - text, CSV, and XML files), transformation (between extraction and loading into staging schema as well as between staging schema and final historical load into master schema) and loading into the data warehouse (in this case - master schema). ETL pipeline is discussed in depth in the further sections and is briefly depicted in the following diagram:

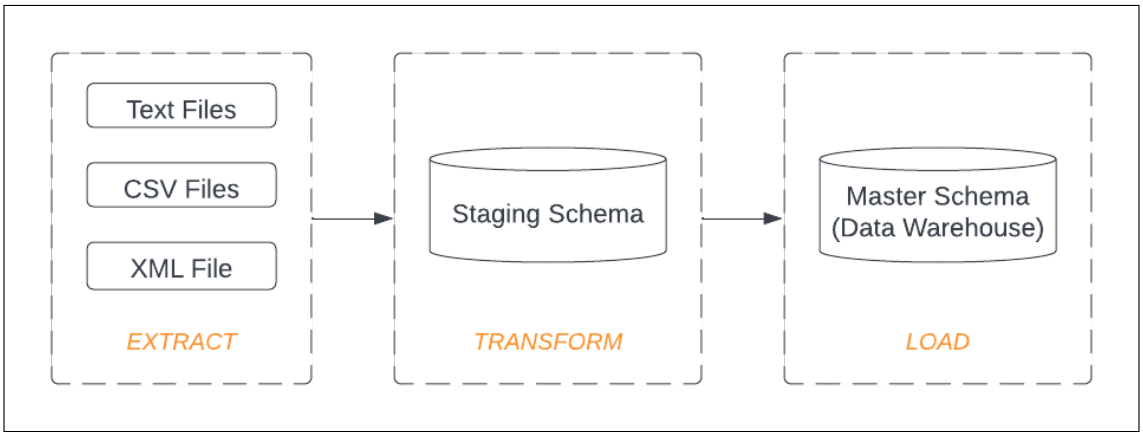


Figure 3.1: ETL Process Diagram

3.3.1 Data Warehouse Initialization

These involve the creation and preparation of the data warehouse. Creation of the Data Warehouse database and tables, including allocation of disk space as well as data integration software may require additional preparation. However, this initialization phase is not timed and is not absolutely part of the data integration process.

3.3.2 Extraction

Pragmatically thinking, businesses cannot work with a single source of data and are bound to utilize data that originates from several points of sources. Thus, extraction is the first and foremost step of ETL. It constitutes obtaining data (small or large-scales) from possible sources like:

- Pre-existing legacy or on-premise databases in the business systems
- In-house applications used for sales/marketing or mobile devices
- Customer Relationship Management Systems
- Flat files (text or CSV)
- XML files

and many more...

This step holds much importance as the source data is usually unstructured (might be corrupted in some cases) and direct loading into our final database (warehouse) can be disastrous. Hence, all the data (either structured or unstructured) is collated to form a single source. This process can be performed without automation but is time-consuming and includes a lot of errors. Thus, in this project, Apache Airflow is used to automate the **EXTRACT**, thereby creating a more effective and well-grounded workflow.

3.3.3 Transformation

After converting several sources of data into a single point of source, transformation comes into the picture. This is where guidelines relevant to the business can be applied to the data, meaning raw data can be converted to the requisite form as per the business use case, that meets business data standards and accessibility. Usually, it involves the following processes/tasks:

- Filter - Selecting only certain attributes to be loaded into the data warehouse, instead of dumping it entirely as is
- Preprocess and Sort -
    - Clean - Removing inconsistent data and imputing missing values
    - Handle redundancy - Dealing with redundant values
    - Join or Split - Creating multiple attributes from one or splitting a single attribute into multiple for ease
    - Sort - Organising the data according to a particular attribute(s)
- Standardize - Applying business rules to format data into a particular form
- Others - Applying other rules for improving the quality of data

Thus, in this project, by performing the process - **TRANSFORM**, we are essentially improving its integrity by ensuring that the raw data before arriving at the final storage location is ready to use and compatible.

3.3.4 Loading

The last and final part of the ETL pipeline is to execute the loading of business data into a data warehouse. The frequency and time period between subsequent data loads are very much dependent on business requirements, which vary from loading entire data at once, also known as a full load, or loading at intervals - incremental load.

3.3.4.1 Full Load

This type of load involves pushing all the transformed data into the data warehouse all at once. It comes in handy for small datasets but is not useful when it comes to maintaining a large database.

3.3.4.2 Incremental Load

This is a better and more efficient approach than a full load for large datasets. By definition, it involves a comparison of new data with what is already existing in the data warehouse. Also, the first load is known as the initial load. In some cases where data does not change and is loaded during the historical load (initial), incremental load holds no importance for such data. This project is restricted to performing only historical **LOAD**.

3.4 Scaling Model

The number of rows in the files is variable, determined by the data generator based on the chosen Scale Factor (SF). This parameter is usually supplied to DIGen to control the volume of Source Data generated. This value determines the sizes of the source files. The default SF is 3 which means that the generated size of data is proportional. Therefore, a more prominent Scale Factor will produce a proportionally larger set of source data. The same Scale Factor is used to generate all source data to ensure coherency.

Each scale factor has a corresponding SF which has no units and is almost equal to the bytes stored in the database. For this project, the various scale factors and SFs are presented in the table 3.1, wherein a megabyte (MB) is equivalent to 2^6 bytes.

SF	3	5	10	20
Scale Factor	300 MB	500 MB	1000 MB	2000 MB

Table 3.1: Implemented Scale Factors

3.5 Implementing TPC-DI on PostgreSQL

Figure 3.2 shows a business process model depicting a brief rundown of the implementation of the TPC-DI benchmark on PostgreSQL.

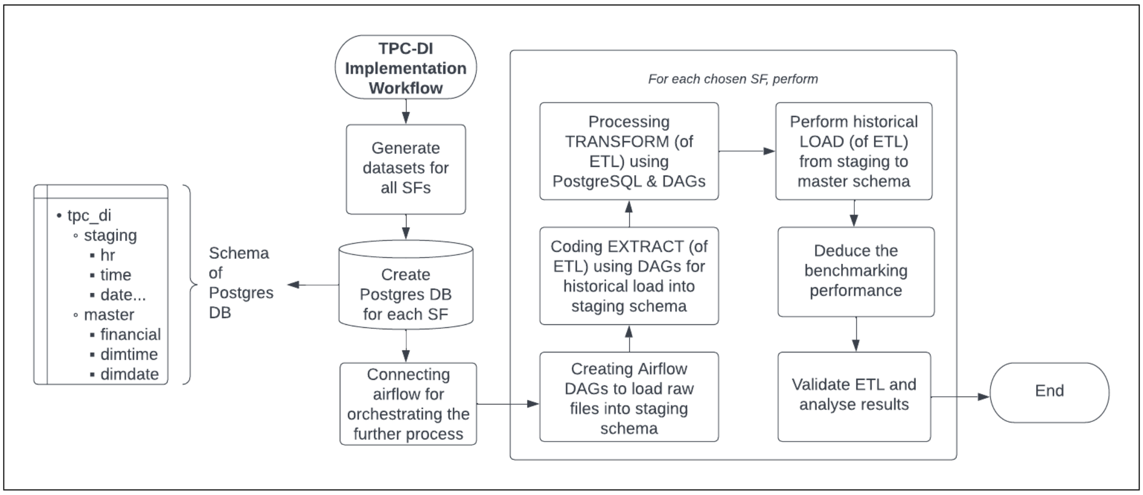

TPC-DI Implementation Workflow
Generate datasets for all SFs
Create Postgres DB for each SF
Connecting airflow for orchestrating the further process
• tpc_di
◦ staging
▪ hr
▪ time
▪ date...
◦ master
▪ financial
▪ dimtime
▪ dimdate
Schema of Postgres DB
For each chosen SF, perform
Creating Airflow DAGs to load raw files into staging schema
Coding EXTRACT (of ETL) using DAGs for historical load into staging schema
Processing TRANSFORM (of ETL) using PostgreSQL & DAGs
Perform historical LOAD (of ETL) from staging to master schema
Deduce the benchmarking performance
Validate ETL and analyse results
End

Figure 3.2: TPC-DI Process Diagram

Some of the major processes undertaken are summarized as follows:

- Data for various SFs is first extracted using DIGEN files into a flat-file format
- PostgresDB is created for each selected SF with 2 schemas:
  - Staging - holds the initially transformed data
  - Master - represents the data warehouse, in this case
- Apache airflow connections are made using Python DAGs for running the ETL tasks
- Then, firstly as flat files are loaded as is into the staging schema (EXTRACT)
- Transformation of each of the tables is created (TRANSFORM)
- Historical load of the data is performed from staging to the master (LOAD)

- Benchmarking results are obtained and analyzed.
- Visualize all outputs and provide detailed reports.

The benchmark was implemented on a local machine with specifications illustrated in table 3.2:

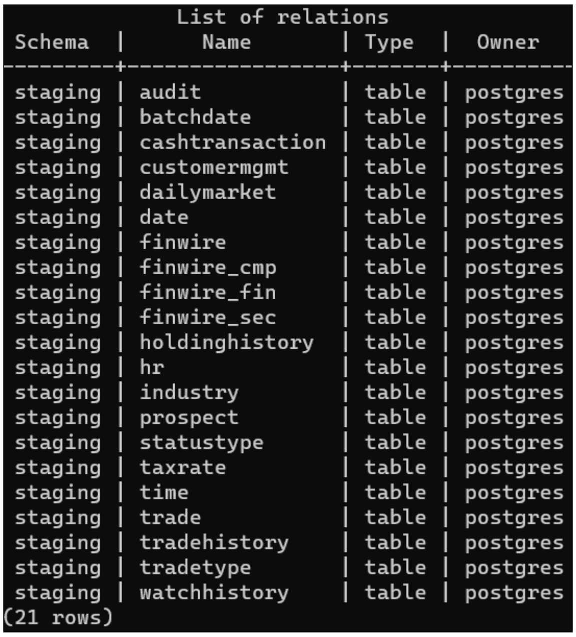
CPU (AMD Ryzen 7 6800HS)	**RAM (DDR5 SODIMM)**	**GPU (NVIDIA GeoForceRTX3060)**
• 8 cores, 16 threads • Base clocking speed at 3.2GHz and can overclock up to 4.7GHz • 16MB L3 Cache	• 16 GB memory • 4800MHz speed	• Dedicated graphics • 6GB VRAM

Table 3.2: Local Machine Specifications

3.6 Staging

3.6.1 Staging Schema

The raw data available from DIGEN files is loading as it is into a staging area as depicted in the implementation overview diagram, within Postgres same database, but as a different schema. Figure 3.3 shows the tables of the EXTRACT stage and B.1 elaborates further on the schema.



```
              List of relations
 Schema  |       Name        | Type  |  Owner
---------+-------------------+-------+----------
 staging | audit             | table | postgres
 staging | batchdate         | table | postgres
 staging | cashtransaction   | table | postgres
 staging | customermgmt      | table | postgres
 staging | dailymarket       | table | postgres
 staging | date              | table | postgres
 staging | finwire           | table | postgres
 staging | finwire_cmp       | table | postgres
 staging | finwire_fin       | table | postgres
 staging | finwire_sec       | table | postgres
 staging | holdinghistory    | table | postgres
 staging | hr                | table | postgres
 staging | industry          | table | postgres
 staging | prospect          | table | postgres
 staging | statustype        | table | postgres
 staging | taxrate           | table | postgres
 staging | time              | table | postgres
 staging | trade             | table | postgres
 staging | tradehistory      | table | postgres
 staging | tradetype         | table | postgres
 staging | watchhistory      | table | postgres
(21 rows)
```

Figure 3.3: Staging Schema

3.6.2 Staging Database Population

Figures 3.4 and 3.5 show the order in which extract into the staging area is performed.

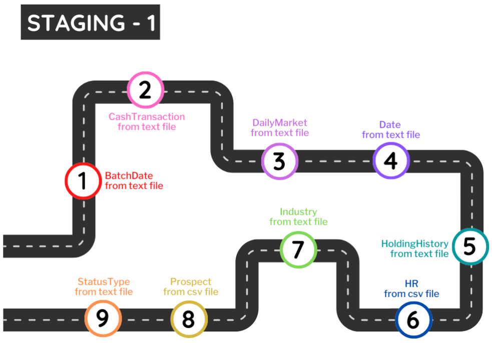


Figure 3.4: Staging Extract - Part 1

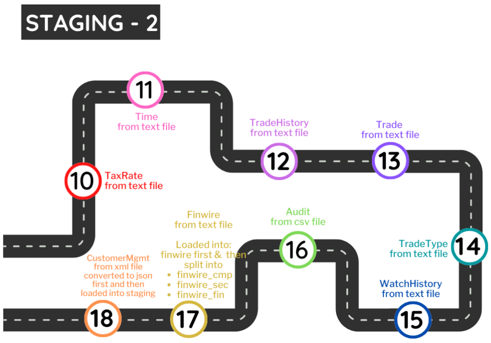


Figure 3.5: Staging Extract - Part 2

- Batch Date is extracted from its text file of BatchDate.txt using the command:

  ```
  COPY staging.batchdate FROM ’/output_data/Batch1/BatchDate.txt’;
  ```

- Cash Transaction is extracted from its text file of CashTransaction.txt using the command:

  ```
  COPY staging.cashtransaction FROM ’/output_data/Batch1/CashTransaction.txt’
  delimiter ’|’;
  ```

- Daily Market is extracted from its text file of DailyMarket.txt using the command:

  ```
  COPY staging.dailymarket FROM ’/output_data/Batch1/DailyMarket.txt’ delimiter ’|’;
  ```

- Date is extracted from its text file of Date.txt using the command:

  ```
  COPY staging.date FROM ’/output_data/Batch1/Date.txt’ delimiter ’|’;
  ```

- Holding History is extracted from its text file of HoldingHistory.txt using the command:

  ```
  COPY staging.holdinghistory FROM ’/output_data/Batch1/HoldingHistory.txt’
  delimiter ’|’;
  ```

- Hr is extracted from its csv file of HR.csv using the command:

  ```
  COPY staging.hr FROM ’/output_data/HR.csv’ delimiter ’,’ csv;
  ```

- Industry is extracted from its text file of Industry.txt using the command:

```
COPY staging.industry FROM ’/output_data/Batch1/Industry.txt’ delimiter ’|’;
```

- Prospect is extracted from its csv file of Prospect.csv using the command:

  ```
  COPY staging.prospect FROM ’/output_data/Batch1/Prospect.csv’ delimiter ’,’ csv;
  ```

- Status Type is extracted from its text file of StatusType.txt using the command:

  ```
  COPY staging.statustype FROM ’/output_data/Batch1/StatusType.txt’ delimiter ’|’;
  ```

- Tax Rate is extracted from its text file of TaxRate.txt using the command:

  ```
  COPY staging.taxrate FROM ’/output_data/Batch1/TaxRate.txt’ delimiter ’|’;
  ```

- Time is extracted from its text file of Time.txt using the command:

  ```
  COPY staging.time FROM ’/output_data/Batch1/Time.txt’ delimiter ’|’;
  ```

- Trade History is extracted from its text file of TradeHistory.txt using the command:

  ```
  COPY staging.tradehistory FROM ’/output_data/Batch1/TradeHistory.txt’ delimiter ’|’;
  ```

- Trade is extracted from its text file of Trade.txt using the command:

  ```
  COPY staging.trade FROM ’/output_data/Batch1/Trade.txt’ delimiter ’|’ null as ”;
  ```

- Trade Type is extracted from its text file of TradeType.txt using the command:

  ```
  COPY staging.tradetype FROM ’/output_data/Batch1/TradeType.txt’ delimiter ’|’;
  ```

- Watch History is extracted from its text file of WatchHistory.txt using the command:

  ```
  COPY staging.watchhistory FROM ’/output_data/Batch1/WatchHistory.txt’ delimiter ’|’;
  ```

- Audit is extracted from its csv file of Audit.csv using the command:

  ```
  COPY staging.audit FROM ’/output_data/Batch1/Audit.csv’ delimiter ’,’ header csv null as ”;
  ```

- For Finwire files, first all the data is copied from all finwire files (belonging from all years and quarters) using the sample command:

  COPY staging.finwire FROM '/output_data/Batch1/FINWIRE1967Q1';

  After this, pre-processing the finwire files is done to split it into CMP, SEC and FIN records using PL/PGSQL in Postgres itself. load_staging_finwire_db.sql is created for the purpose (B.3 as added in the appendix).

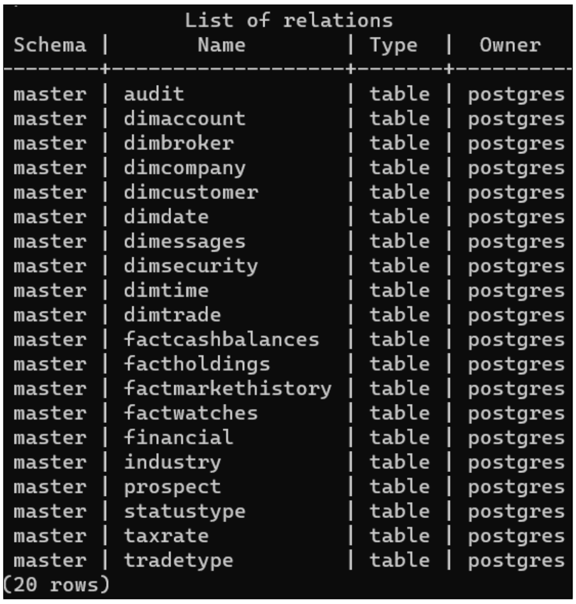
- For CustomerMgmt, the CustomerMgmt.xml is first converted to JSON. Python is then used to transform the JSON file into CSV customermgmt_conversion.py (A.1 as added in appendix) elaborates on the parsing further. Post that, staging.customermgmt is finally populated using the command:

  COPY staging.customermgmt FROM '/output_data/Batch1/CustomerMgmt.csv' delimiter ',' header csv null as '';

3.7 Master - Data Warehouse

3.7.1 Master Schema

The data from staging area is then transformed within Postgres and populated into the master tables, which is the Data Warehouse for our project. Figure 3.6 shows the tables of the LOAD stage B.2 elaborates further on the schema.



```
              List of relations
 Schema |        Name        | Type  |  Owner
--------+--------------------+-------+----------
 master | audit              | table | postgres
 master | dimaccount         | table | postgres
 master | dimbroker          | table | postgres
 master | dimcompany         | table | postgres
 master | dimcustomer        | table | postgres
 master | dimdate            | table | postgres
 master | dimessages         | table | postgres
 master | dimsecurity        | table | postgres
 master | dimtime            | table | postgres
 master | dimtrade           | table | postgres
 master | factcashbalances   | table | postgres
 master | factholdings       | table | postgres
 master | factmarkethistory  | table | postgres
 master | factwatches        | table | postgres
 master | financial          | table | postgres
 master | industry           | table | postgres
 master | prospect           | table | postgres
 master | statustype         | table | postgres
 master | taxrate            | table | postgres
 master | tradetype          | table | postgres
(20 rows)
```

Figure 3.6: Master Schema

3.7.2 Master Database Population

Before populating the master tables, as discussed earlier transformations need to be performed.

3.7.2.1 Simple Table Population

Static tables like tradetype were loaded as is from the staging area directly as they didn't require any transformation, plus were without any dependencies and hence could be inserted using the following sample Postgres code:

```
truncate table master.tradetype;
insert into master.tradetype select * from staging.tradetype;
```

The following tables were loaded in this way and this particular order:

- tradetype
- statustype
- taxrate
- industry
- dimdate
- dimtime

load_master_static_tables.sql details the entire code for loading the aforementioned static tables is added in the appendix - B.4.

3.7.2.2 Complex Table Population

For the remaining tables like dimcompany, several approaches of transformations were used to fit the data as per the requisite schema. Combinations of methods listed below were employed.

- JOINS - INNER, LEFT and CROSS
- Simple AGGREGATIONS - SUM, MIN, MAX, COUNT, AVERAGE
- CASTING - TO_CHAR(), ::DATE, ::TIME, NUMERIC
- UNION operator
- LIKE operator (%)
- CONCATENTATION of multiple columns/string value(s)
- TRIM function
- NULLIF function
- COALESCE function
- CASE WHEN method
- ROW_NUMBER function
- CTEs (Common Table Expressions)
- EXTRACT function
- WINDOW functions involving PARTITION OVER - MIN, MAX, SUM, LEAD
- WINDOW functions involving FRAMES - ROWS UNBOUNDED PRECEDING

load_master_complex_tables.sql details the entire code for loading the aforementioned tables is added in the appendix - B.5.

3.8 Data Integration using Airflow

3.8.1 Airflow Overview

load_master_historical_dag.py as added in the appendix A.2 details the airflow script to implement the full historical load into the data warehouse after transformations.

The following points highlight the use and Airflow and its components in the project:

- Each table in the master schema is represented using one or more tasks, and in total, we have 30 tasks.
- As per the implementation of our project, tasks progress from creation staging schema →loading staging (EXTRACT) →creating master schema → TRANSFORM and then historical LOAD
- Airflow Operators - PostgresOperator and PythonOperator are used in general to call corresponding sql code files and a Python function respectively. PostgresOperator is used for all the transformations except for the CustomerMgmt XML, wherein PythonOperator is used. Task indicated as pink in the figure 3.7 indicates use of PythonOperator, while for others PostgresOperator was used.
- We saw in the previous subsection that some tables are independent of other tables - like tradetype, dimdate, etc. Tasks catering to such tables can be run parallelly in airflow, thereby reducing the overall runtime of the historical load.
- Some tables are dependent on others - like factcashbalances depends on dimdate and dimaccount other than cashtransaction (from staging area). To execute such tasks, dependencies need to created in airflow using bitshift (>>or <<) operators. The task dependencies ensure that any task that is dependent on other, cannot run until its upstream (dependent task) is completed. In other words, using the example from the above point, the task dependencies for factcashbalances to be completed, dimaccount and dimdate need to be fully completed (transformed and loaded).

  transform_load_master_dimaccount >>transform_load_master_factcashbalances
  transform_load_master_dimdate >>transform_load_master_factcashbalances

3.8.2 Tasks Integration

Tasks were added into the DAG script in this following order:

1. create_staging_schema
2. load_txt_csv_sources_to_staging
3. load_finwire_to_staging
4. parse_finwire
5. convert_customermgmt_xml_to_csv
6. load_customer_mgmt_to_staging
7. create_master_schema
8. load_master_tradetype

9. load_master_statustype
10. load_master_taxrate
11. load_master_industry
12. transform_load_master_dimdate
13. transform_load_master_dimtime
14. transform_load_master_dimcompany
15. load_master_dimessages_dimcompany
16. transform_load_master_dimbroker
17. transform_load_master_prospect
18. transform_load_master_dimcustomer
19. load_master_dimessages_dimcustomer
20. update_master_prospect
21. transform_load_master_dimaccount
22. transform_load_master_dimsecurity
23. transform_load_master_dimtrade
24. load_master_dimessages_dimtrade
25. transform_load_master_financial
26. transform_load_master_factcashbalances
27. transform_load_master_factholdings
28. transform_load_master_factwatches
29. transform_load_master_factmarkethistory
30. load_master_dimessages_factmarkethistory

Although that's, the case, it does not mean each task would run in this sequence. Tasks that do not have dependencies are able to run in parallel, while those with dependencies would require to run in sequence after its dependencies.

Dependencies can be found below ('x >>y' indicates that y is dependent on x):

Staging schema dependency

- create_staging_schema >>load_txt_csv_sources_to_staging
- create_staging_schema >>load_finwire_to_staging >>parse_finwire
- create_staging_schema >>convert_customermgmt_xml_to_csv >> load_customer_mgmt_to_staging

Master schema dependency

- load_txt_csv_sources_to_staging >>create_master_schema
- parse_finwire >>create_master_schema
- load_customer_mgmt_to_staging >>create_master_schema

Transformation/Loading to master dependency

- create_master_schema >>load_master_tradetype
- create_master_schema >>load_master_statustype
- create_master_schema >>load_master_taxrate
- create_master_schema >>load_master_industry

- create_master_schema >>transform_load_master_dimdate
- create_master_schema >>transform_load_master_dimtime
- create_master_schema >>transform_load_master_dimcompany
- transform_load_master_dimcompany >>load_master_dimessages_dimcompany
- transform_load_master_dimdate >>transform_load_master_dimbroker
- transform_load_master_dimdate >>transform_load_master_prospect
- load_master_taxrate >>transform_load_master_dimcustomer
- transform_load_master_prospect >>transform_load_master_dimcustomer
- transform_load_master_dimcustomer >>load_master_dimessages_dimcustomer
- transform_load_master_dimcustomer >>update_master_prospect
- transform_load_master_dimbroker >>transform_load_master_dimaccount
- transform_load_master_dimcustomer >>transform_load_master_dimaccount
- transform_load_master_dimcompany >>transform_load_master_dimsecurity
- transform_load_master_dimaccount >>transform_load_master_dimtrade
- load_master_statustype >>transform_load_master_dimtrade
- load_master_tradetype >>transform_load_master_dimtrade
- transform_load_master_dimsecurity >>transform_load_master_dimtrade
- transform_load_master_dimtrade >>load_master_dimessages_dimtrade
- transform_load_master_dimcompany >>transform_load_master_financial
- transform_load_master_dimaccount >>transform_load_master_factcashbalances
- transform_load_master_dimdate >>transform_load_master_factcashbalances
- transform_load_master_dimtrade >>transform_load_master_factholdings
- transform_load_master_dimcustomer >>transform_load_master_factwatches
- transform_load_master_dimsecurity >>transform_load_master_factwatches
- transform_load_master_dimdate >>transform_load_master_factwatches
- transform_load_master_dimdate >>transform_load_master_factmarkethistory
- transform_load_master_financial >>transform_load_master_factmarkethistory
- transform_load_master_dimcompany >>transform_load_master_factmarkethistory
- transform_load_master_dimsecurity >>transform_load_master_factmarkethistory
- transform_load_master_factmarkethistory >>load_master_dimessages_factmarkethistory

Figure 3.7 summarises the tasks integration.

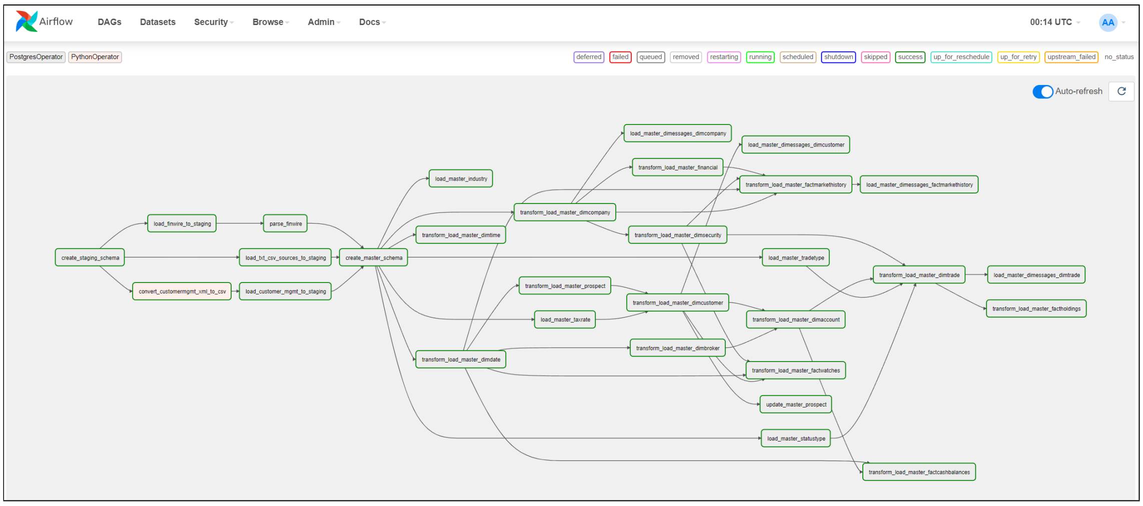

Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
00:14 UTC
AA
PostgresOperator
PythonOperator
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Auto-refresh
create_staging_schema
load_finwire_to_staging
parse_finwire
load_txt_csv_sources_to_staging
convert_customermgmt_xml_to_csv
load_customer_mgmt_to_staging
create_master_schema
load_master_industry
transform_load_master_dimtime
transform_load_master_dimcompany
transform_load_master_prospect
load_master_taxrate
transform_load_master_dimdate
load_master_dimessages_dimcompany
transform_load_master_financial
transform_load_master_dimsecurity
transform_load_master_dimcustomer
transform_load_master_dimbroker
load_master_dimessages_dimcustomer
transform_load_master_factmarkethistory
load_master_dimessages_factmarkethistory
load_master_tradetype
transform_load_master_dimaccount
transform_load_master_factwatches
update_master_prospect
load_master_statustype
transform_load_master_dimtrade
load_master_dimessages_dimtrade
transform_load_master_factholdings
transform_load_master_factcashbalances

Figure 3.7: Historical Load DAG

The results of data extraction, transformation and loading (Data Integration Tasks) using Airflow are discussed in the next chapter.

Chapter 4

Results and Discussions

The benchmark was performed on a local machine, with a total of four different scale factors (3, 5, 10 and 20) and this chapter discusses the performance of the same.

4.1 Overall performance across all scale factors

Inspecting the total average run time for all the tasks which were run for multiple scale factors, it is evident that the performance is nowhere close to being linear. In fact, as the scale factor increases further, an exponential pattern starts to emerge, as seen in Figure 4.1.

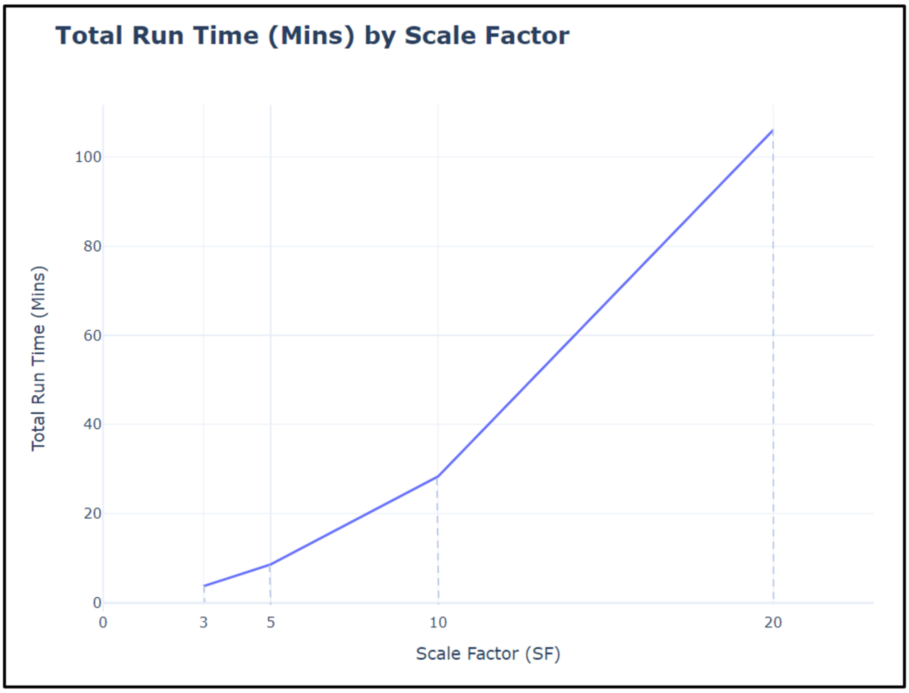


Figure 4.1: Total Average Runtime by Scale Factor

Furthermore, the table 4.1 details the average runtime durations across each scale factor in seconds.

TASK ID	SF 3	SF 5	SF 10	SF 20
create_staging_schema	0	1	1	0
load_txt_csv_sources_to_staging	15	25	47	105
load_finwire_to_staging	1	2	2	7
convert_customermgmt_xml_to_csv	94	257	974	4107
parse_finwire	10	18	38	88
load_customer_mgmt_to_staging	0	0	1	1
create_master_schema	1	1	0	0
transform_load_master_dimcompany	1	1	0	0
transform_load_master_dimdate	1	0	0	1
transform_load_master_dimtime	1	1	1	1
load_master_statustype	1	0	0	1
load_master_taxrate	1	1	0	1
load_master_tradetype	1	0	0	1
load_master_industry	1	0	0	1
transform_load_master_dimbroker	1	1	0	1
transform_load_master_dimsecurity	5	13	44	176
transform_load_master_prospect	1	1	1	1
transform_load_master_financial	33	95	385	1547
load_master_dimessages_dimcompany	1	1	0	0
transform_load_master_dimcustomer	1	1	2	3
load_master_dimessages_dimcustomer	0	0	0	1
update_master_prospect	1	1	2	8
transform_load_master_dimaccount	1	1	1	3
transform_load_master_factwatches	14	11	28	46
transform_load_master_dimtrade	22	21	53	79
transform_load_master_factcashbalances	9	6	8	15
transform_load_master_factholdings	2	2	4	10
load_master_dimessages_dimtrade	1	0	1	2
transform_load_master_factmarkethistory	84	150	327	693
load_master_dimessages_factmarkethistory	1	1	1	2

Table 4.1: Overall Runtime Results

At a scale factor (SF) of 3 (equivalent to a database size of 300MB), all of the tasks were managed to complete within a span of 100 seconds (on average). This quickly rises to a total of 250 seconds for SF 5, which then follows along with a large increase in run time for SF 10 and 20 respectively, but rises again steeply when progressing to SF 20. Although this is the case, looking only at the overall run time for all tasks queries together, gives a biased assumption regarding the performance of many of the individual tasks. Thus, the performance results are further broken down into individual tasks in the next section.

4.2 Task-wise performance across all scale factors

Firstly, it can be observed that the majority of the tasks are not running exponentially as the scale factor increases and it is actually due to a few specific task such as convert_customermgmt_xml_to_csv, transform_load_master_financial and transform_load_master_factmarkethistory that causes the total overall run time to massively increase as the difference can be spotted in figures 4.2 and 4.3. The convert_customermgmt_xml_to_csv is the only task (from all 30) that is being run on Python and hence the most expensive. It would definitely perform better if it were being run on Postgres itself. For the tasks transform_load_master_financial and transform_load_master_factmarkethistory the transformations are quite heavy which results in the higher run time cost. After excluding the highest 3 expensive tasks, we can see in 4.3 that in most cases the run time is mostly linear for the remaining tasks.

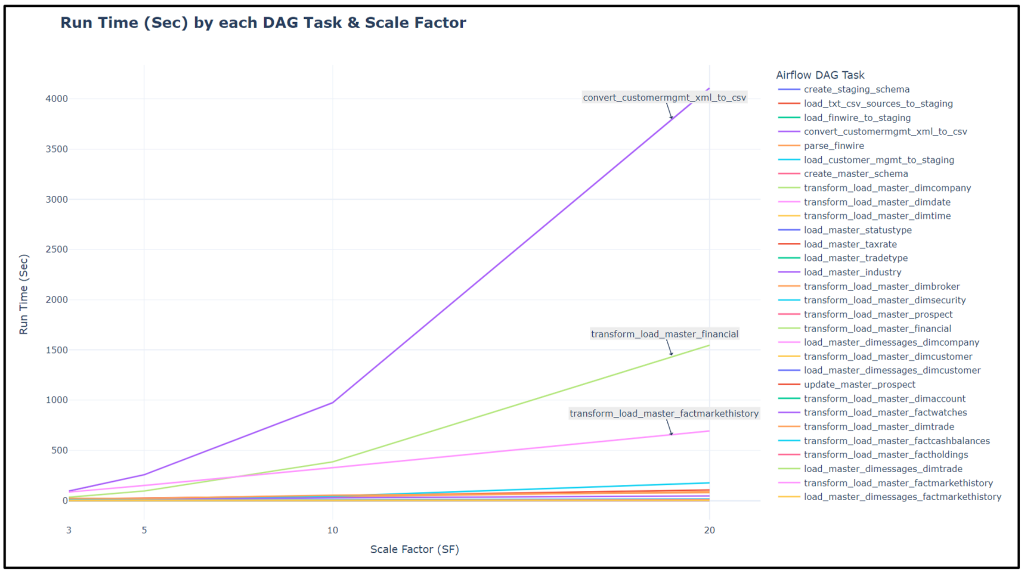


Figure 4.2: Runtime by Task

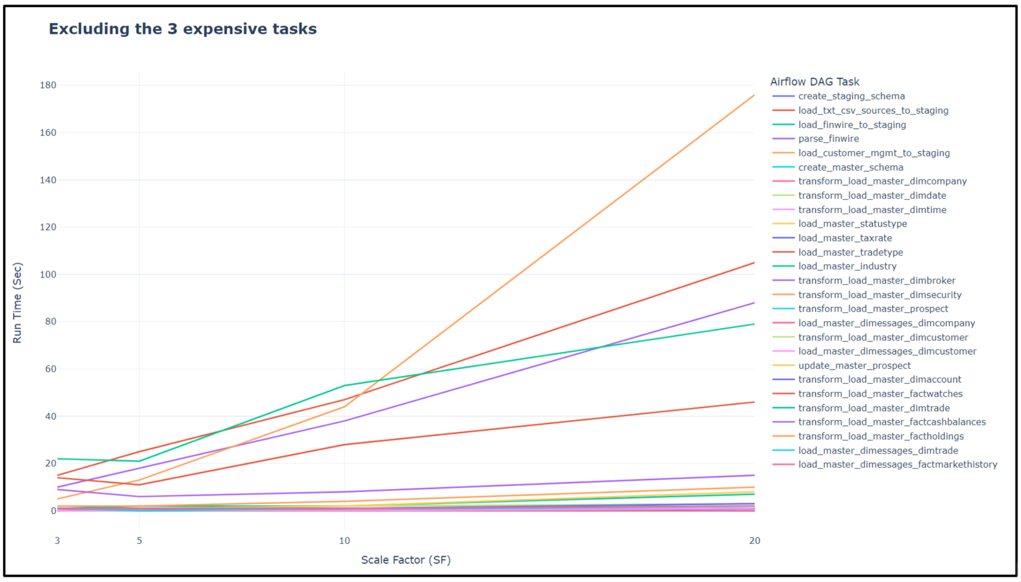


Figure 4.3: Runtime by Task (excluding 3 tasks with exponential runtime)

4.3 Task-wise performance across each scale factors

- Figures 4.4 to 4.7 depict the start time and end time for each task across each SF.
- Parallel execution of some tasks like create_master_schema, transform_load_master_dimcompany, transform_load_master_dimdate, transform_load_master_dimtime, load_master_statustype, load_master_taxrate, load_master_tradetype and load_master_industry can be observed.
- convert_customermgmt_xml_to_csv takes the longest time in each SF.

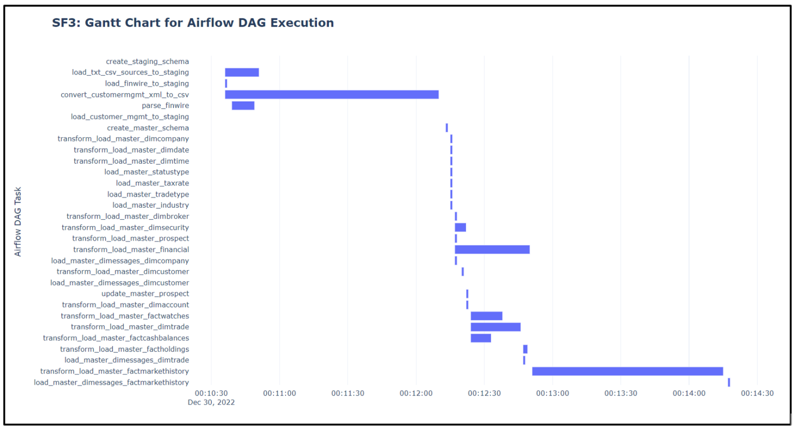


Figure 4.4: Gantt Chart for SF3

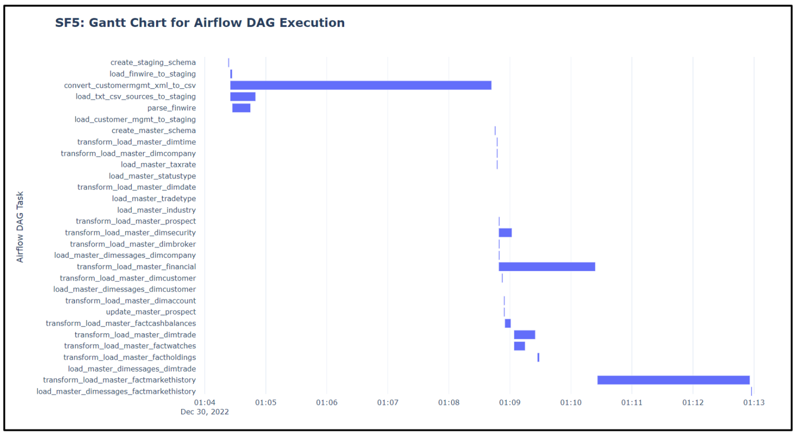


Figure 4.5: Gantt Chart for SF5

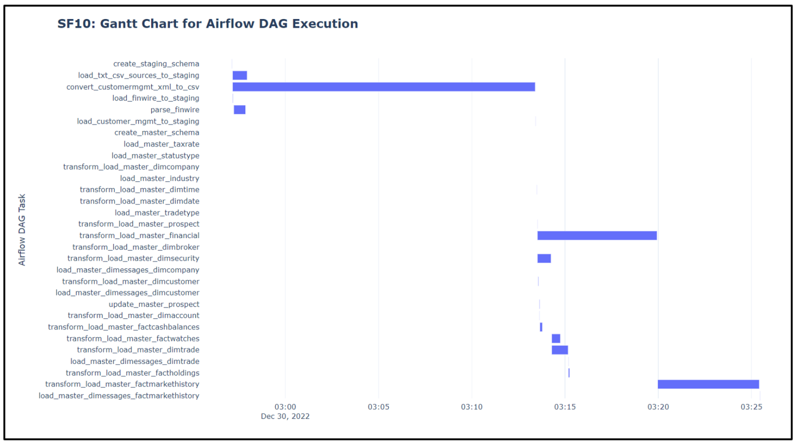


Figure 4.6: Gantt Chart for SF10

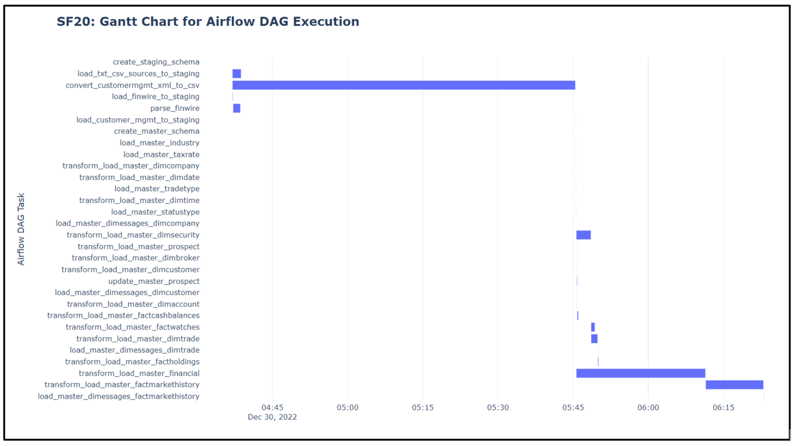


Figure 4.7: Gantt Chart for SF20

4.4 Summary of Results

Running a benchmark test on a large variety of tasks, on multiple scale factors provided great insights into how Postgres deals with both increasing volume of data, and the alteration of node steps chosen depending on the size of the table data. Although majority of the tasks demonstrated non-exponential run-time performance (more than 90%), there were a few that stood out.

Chapter 5

Conclusion

In conclusion, the TPC-DI framework provides a comprehensive ETL model to perform a fair and transparent benchmark on PostgreSQL DB. This project report documents the approach used to benchmark PostgreSQL with the integration of Airflow. Almost all the transformations were performed within Postgres itself, and it has shown to perform well in most of the transformation and loading tasks. Nevertheless, there were a few exceptions in which it displayed an emerging exponential pattern as the volume of data scaled upwards. This again proves that transformations withing SQL itself can be very powerful and depending on external tools for transformations are not always the best choice even if they are marketed to perform exceptionally.

Bibliography

[Bartolini et al., 2017] Bartolini, G., Ciolli, G., & Riggs, S. (2017). PostgreSQL Administration Cookbook - Third Edition. Packt Publishing, Limited.

[Install Docker Desktop, 2022] install Docker Desktop. (2022, January 22). Docker. Retrieved October 27, 2022, from https://www.docker.com/products/docker-desktop/

[Kabangu, 2009] Kabangu, S. (2009). Benchmarking Databases.

[Matthew & Stones, 2005] Matthew, N., & Stones, R. (2005). Beginning Databases with PostgreSQL: From Novice to Professional. Apress.

[Obe & Hsu, 2017] Obe, R. O., & Hsu, L. S. (2017). PostgreSQL: Up and Running : a Practical Guide to the Advanced Open Source Database. In (pp. 11 - 14). O'Reilly Media, Incorporated.

[The PostgreSQL Global Development Group, 2022] The PostgreSQL Global Development Group. (2022). PostgreSQL 14.5 Documentation. PostgreSQL. Retrieved October 26, 2022, from https://www.postgresql.org/docs/14

[Python, 2022] Python. (2022, February). Welcome to Python.org. Retrieved October 27, 2022, from https://www.python.org/

[Apache Software Foundation. (2022, September 18).] Apache Software Foundation. (2022, September 18). Airflow Documentation. Apache Airflow. Retrieved December 24, 2022, from https://airflow.apache.org/docs/apache-airflow/stable/index.html

[Tortosa, 2020] Tortosa, Á. H. (2020). Performance Benchmark Postgresql / Mongodb. Ongres.com. Retrieved October 28, 2022, from https://info.enterprisedb.com/rs/069-ALB-339/images/PostgreSQL_MongoDB_Benchmark-WhitepaperFinal.pdf

[Transaction Processing Performance Council (TPC), 2021] TPC-DS. (2021, January). tpc.org. Retrieved October 28, 2022, from https://www.tpc.org/tpcds/

[Visual Studio Code, 2022] Visual Studio Code. (2022, January). Visual Studio Code - Code Editing. Redefined. Retrieved October 27, 2022, from https://code.visualstudio.com/

[Zimanyi, n.d.] Zimanyi, E. (n.d.). INFO-H-419: Data Warehouses. INFO-H-419: Data Warehouses [Université Libre de Bruxelles - Service CoDE - Laboratoire WIT]. Retrieved October 27, 2022, from https://cs.ulb.ac.be/public/teaching/infoh419

Appendix A

Python Scripts

```python
import os
import numpy as np
import pandas as pd
import xmltodict
import json

def customermgmt_convert():
    with open('dags/sf_3/Batch1/CustomerMgmt.xml') as fd:
        doc = xmltodict.parse(fd.read())
        fd.close()

    with open("dags/sf_3/Batch1/CustomerData.json", "w") as outfile:
        outfile.write(json.dumps(doc))
        outfile.close()

    f = open('dags/sf_3/Batch1/CustomerData.json','r')

    cust = json.load(f)
    actions = cust['TPCDI:Actions']
    action = actions['TPCDI:Action']
    cust_df = pd.DataFrame(columns = np.arange(0, 36))


    for a in action:

        cust_row = {}

        # action element
        cust_row.update({0: [f"{a.get('@ActionType')}"]})
        cust_row.update({1: [f"{a.get('@ActionTS')}"]})

        # action.customer element
        cust_row.update({2: [f"{a.get('Customer').get('@C_ID')}"]})
        cust_row.update({3: [f"{a.get('Customer').get('@C_TAX_ID')}"]})
        cust_row.update({4: [f"{a.get('Customer').get('@C_GNDR')}"]})
        cust_row.update({5: [f"{a.get('Customer').get('@C_TIER')}"]})
        cust_row.update({6: [f"{a.get('Customer').get('@C_DOB')}"]})

        # action.customer.name element
        if a.get('Customer').get('Name') != None:
            cust_row.update({7: [f"{a.get('Customer').get('Name').get('C_L_NAME')}"]})
            cust_row.update({8: [f"{a.get('Customer').get('Name').get('C_F_NAME')}"]})
            cust_row.update({9: [f"{a.get('Customer').get('Name').get('C_M_NAME')}"]})
        else:
            cust_row.update({7: [None]})
            cust_row.update({8: [None]})
            cust_row.update({9: [None]})

        # action.customer.address element
        if a.get('Customer').get('Address') != None:
            cust_row.update({10: [f"{a.get('Customer').get('Address').get('C_ADLINE1')}"]})
            cust_row.update({11: [f"{a.get('Customer').get('Address').get('C_ADLINE2')}"]})
            cust_row.update({12: [f"{a.get('Customer').get('Address').get('C_ZIPCODE')}"]})
            cust_row.update({13: [f"{a.get('Customer').get('Address').get('C_CITY')}"]})
            cust_row.update({14: [f"{a.get('Customer').get('Address').get('C_STATE_PROV')}"]})
            cust_row.update({15: [f"{a.get('Customer').get('Address').get('C_CTRY')}"]})
        else:
            cust_row.update({10: [None]})
            cust_row.update({11: [None]})
            cust_row.update({12: [None]})
            cust_row.update({13: [None]})
            cust_row.update({14: [None]})
            cust_row.update({15: [None]})

        # action.customer.contactinfo element
        if a.get('Customer').get('ContactInfo') != None:
            cust_row.update({16: [f"{a.get('Customer').get('ContactInfo').get('C_PRIM_EMAIL')}"
    ]})
            cust_row.update({17: [f"{a.get('Customer').get('ContactInfo').get('C_ALT_EMAIL')}"
    ]})

            # action.customer.contactinfo.phone element

            # phone_1
            cust_row.update({18: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_1').get('C_CTRY_CODE')}"]})
            cust_row.update({19: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_1').get('C_AREA_CODE')}"]})
            cust_row.update({20: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_1').get('C_LOCAL')}"]})
            cust_row.update({21: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_1').get('C_EXT')}"]})

            # phone_2
            cust_row.update({22: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_2').get('C_CTRY_CODE')}"]})
            cust_row.update({23: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_2').get('C_AREA_CODE')}"]})
            cust_row.update({24: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_2').get('C_LOCAL')}"]})
            cust_row.update({25: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_2').get('C_EXT')}"]})

            # phone_3
            cust_row.update({26: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_3').get('C_CTRY_CODE')}"]})
            cust_row.update({27: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_3').get('C_AREA_CODE')}"]})
            cust_row.update({28: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_3').get('C_LOCAL')}"]})
            cust_row.update({29: [f"{a.get('Customer').get('ContactInfo').get('C_PHONE_3').get('C_EXT')}"]})
        else:
            cust_row.update({16: [None]})
            cust_row.update({17: [None]})
            cust_row.update({18: [None]})
            cust_row.update({19: [None]})
            cust_row.update({20: [None]})
            cust_row.update({21: [None]})
            cust_row.update({22: [None]})
            cust_row.update({23: [None]})
            cust_row.update({24: [None]})
            cust_row.update({25: [None]})
            cust_row.update({26: [None]})
            cust_row.update({27: [None]})
            cust_row.update({28: [None]})
            cust_row.update({29: [None]})

        # action.customer.taxinfo element
        if a.get('Customer').get('TaxInfo') != None:
            cust_row.update({30: [f"{a.get('Customer').get('TaxInfo').get('C_LCL_TX_ID')}"]})
            cust_row.update({31: [f"{a.get('Customer').get('TaxInfo').get('C_NAT_TX_ID')}"]})
        else:
            cust_row.update({30:  [None]})
            cust_row.update({31:  [None]})

        # action.customer.account attribute
        if a.get('Customer').get('Account') != None:
            cust_row.update({32: [f"{a.get('Customer').get('Account').get('@CA_ID')}"]})
            cust_row.update({33: [f"{a.get('Customer').get('Account').get('@CA_TAX_ST')}"]})

            # action.customer.account element
            cust_row.update({34: [f"{a.get('Customer').get('Account').get('CA_B_ID')}"]})
            cust_row.update({35: [f"{a.get('Customer').get('Account').get('CA_NAME')}"]})
        else:
            cust_row.update({32: [None]})
            cust_row.update({33: [None]})
            cust_row.update({34: [None]})
            cust_row.update({35: [None]})

        # append to dataframe
        cust_df = pd.concat([cust_df, pd.DataFrame.from_dict(cust_row)], axis = 0)

    cust_df.replace(to_replace = np.NaN, value = "", inplace = True)
    cust_df.replace(to_replace = "None", value = "", inplace = True)
    cust_df.to_csv('dags/sf_3/Batch1/CustomerMgmt.csv', index = False)
    print('Customer Management data converted from XML to CSV')
```

Listing A.1: customermgmt_conversion.py

```
import os
import airflow
from datetime import timedelta, datetime
from airflow import DAG
# from airflow.operators.postgres_operator import PostgresOperator #deprecated
from airflow.providers.postgres.operators.postgres import PostgresOperator
from airflow.operators.python_operator import PythonOperator
from airflow.decorators import task
from customermgmt_conversion import customermgmt_convert


# Default arguments for dag
default_args = {
    'owner': 'airflow',
    'start_date': datetime(2022, 12, 30)
}

# Create dag
dag_psql = DAG(
    dag_id = "dw_sf_3",
    default_args = default_args,
    schedule_interval = None,
)

# Task1 - Create staging schema
create_staging_schema = PostgresOperator(
    task_id = "create_staging_schema",
    postgres_conn_id = "pg_sf_3",
    sql = "create_staging_schema.sql",
  dag = dag_psql
)

# Task2 - Load txt and csv sources to staging
load_txt_csv_sources_to_staging = PostgresOperator(
    task_id = "load_txt_csv_sources_to_staging",
    postgres_conn_id = "pg_sf_3",
    sql = "staging_data_commands.sql",
  dag = dag_psql
)

# Task3 - Load finwire source to staging
load_finwire_to_staging = PostgresOperator(
    task_id = "load_finwire_to_staging",
    postgres_conn_id = "pg_sf_3",
    sql = "staging_finwire_load1.sql",
  dag = dag_psql
)

# Task4 - Parse finwire and load to seperate tables
parse_finwire = PostgresOperator(
    task_id = "parse_finwire",
    postgres_conn_id = "pg_sf_3",
    sql = "load_staging_finwire_db.sql",
  dag = dag_psql
)

# Task5 - Convert customer management source from xml to csv
convert_customermgmt_xml_to_csv = PythonOperator(
    task_id = "convert_customermgmt_xml_to_csv",
    python_callable = customermgmt_convert,
    dag = dag_psql,
)

# Task6 - Load customer management source to staging
load_customer_mgmt_to_staging = PostgresOperator(
    task_id = "load_customer_mgmt_to_staging",
    postgres_conn_id = "pg_sf_3",
    sql = "load_staging_customermgmt_db.sql",
  dag = dag_psql
)

# Task7 - Create master schema
create_master_schema = PostgresOperator(
    task_id = "create_master_schema",
    postgres_conn_id = "pg_sf_3",
    sql = "create_master_schema.sql",
  dag = dag_psql
)

# Task8 - Direct load master.tradetype
load_master_tradetype = PostgresOperator(
    task_id = "load_master_tradetype",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/1_load_master_tradetype.sql",
  dag = dag_psql
)

# Task9 - Direct load master.statustype
load_master_statustype = PostgresOperator(
    task_id = "load_master_statustype",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/2_load_master_statustype.sql",
  dag = dag_psql
)

# Task10 - Direct load master.taxrate
load_master_taxrate = PostgresOperator(
    task_id = "load_master_taxrate",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/3_load_master_taxrate.sql",
  dag = dag_psql
)

# Task11 - Direct load master.industry
load_master_industry = PostgresOperator(
    task_id = "load_master_industry",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/4_load_master_industry.sql",
  dag = dag_psql
)

# Task12 - Transform & load master.dimdate
transform_load_master_dimdate = PostgresOperator(
    task_id = "transform_load_master_dimdate",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/5_transform_load_master_dimdate.sql",
  dag = dag_psql
)

# Task13 - Transform & load master.dimtime
```

```
transform_load_master_dimtime = PostgresOperator(
    task_id = "transform_load_master_dimtime",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/6_transform_load_master_dimtime.sql",
  dag = dag_psql
)

# Task14 – Transform & load master.dimcompany
transform_load_master_dimcompany = PostgresOperator(
    task_id = "transform_load_master_dimcompany",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/7_transform_load_master_dimcompany.sql",
  dag = dag_psql
)

# Task15 – Load master.dimessages with alert from master.dimcompany
load_master_dimessages_dimcompany = PostgresOperator(
    task_id = "load_master_dimessages_dimcompany",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/8_load_master_dimessages_dimcompany.sql",
  dag = dag_psql
)

# Task16 – Transform & load master.dimbroker
transform_load_master_dimbroker = PostgresOperator(
    task_id = "transform_load_master_dimbroker",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/9_transform_load_master_dimbroker.sql",
  dag = dag_psql
)

# Task17 – Transform & load master.prospect
transform_load_master_prospect = PostgresOperator(
    task_id = "transform_load_master_prospect",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/10_transform_load_master_prospect.sql",
  dag = dag_psql
)

# Task18 – Transform & load master.dimcustomer
transform_load_master_dimcustomer = PostgresOperator(
    task_id = "transform_load_master_dimcustomer",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/11_transform_load_master_dimcustomer.sql",
  dag = dag_psql
)

# Task19 – Load master.dimessages with alert from master.dimcustomer
load_master_dimessages_dimcustomer = PostgresOperator(
    task_id = "load_master_dimessages_dimcustomer",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/12_load_master_dimessages_dimcustomer.sql",
  dag = dag_psql
)

# Task20 – Update master.prospect
update_master_prospect = PostgresOperator(
    task_id = "update_master_prospect",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/13_update_master_prospect.sql",
  dag = dag_psql
)

# Task21 – Transform & load master.dimaccount
transform_load_master_dimaccount = PostgresOperator(
    task_id = "transform_load_master_dimaccount",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/14_transform_load_master_dimaccount.sql",
  dag = dag_psql
)

# Task22 – Transform & load master.dimsecurity
transform_load_master_dimsecurity = PostgresOperator(
    task_id = "transform_load_master_dimsecurity",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/15_transform_load_master_dimsecurity.sql",
  dag = dag_psql
)

# Task23 – Transform & load master.dimtrade
transform_load_master_dimtrade = PostgresOperator(
    task_id = "transform_load_master_dimtrade",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/16_transform_load_master_dimtrade.sql",
  dag = dag_psql
)

# Task24 – Load master.dimessages with alert from master.dimtrade
load_master_dimessages_dimtrade = PostgresOperator(
    task_id = "load_master_dimessages_dimtrade",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/17_load_master_dimessages_dimtrade.sql",
  dag = dag_psql
)

# Task25 – Transform & load master.financial
transform_load_master_financial = PostgresOperator(
    task_id = "transform_load_master_financial",
    postgres_conn_id = "pg_sf_3",
```

```python
    sql = "/transformations/18_transform_load_master_financial.sql",
  dag = dag_psql
)

# Task26 - Transform & load master.factcashbalances
transform_load_master_factcashbalances = PostgresOperator(
    task_id = "transform_load_master_factcashbalances",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/19_transform_load_master_factcashbalances.sql",
  dag = dag_psql
)

# Task27 - Transform & load master.factholdings
transform_load_master_factholdings = PostgresOperator(
    task_id = "transform_load_master_factholdings",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/20_transform_load_master_factholdings.sql",
  dag = dag_psql
)

# Task28 - Transform & load master.factwatches
transform_load_master_factwatches = PostgresOperator(
    task_id = "transform_load_master_factwatches",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/21_transform_load_master_factwatches.sql",
  dag = dag_psql
)

# Task29 - Transform & load master.factmarkethistory
transform_load_master_factmarkethistory = PostgresOperator(
    task_id = "transform_load_master_factmarkethistory",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/22_transform_load_master_factmarkethistory.sql",
  dag = dag_psql
)

# Task30 - Load master.dimessages with alert from master.factmarkethistory
load_master_dimessages_factmarkethistory = PostgresOperator(
    task_id = "load_master_dimessages_factmarkethistory",
    postgres_conn_id = "pg_sf_3",
    sql = "/transformations/23_load_master_dimessages_factmarkethistory.sql",
  dag = dag_psql
)


# Task Dependencies

# Staging schema dependency
create_staging_schema >> load_txt_csv_sources_to_staging
create_staging_schema >> load_finwire_to_staging >> parse_finwire
create_staging_schema >> convert_customermgmt_xml_to_csv >> load_customer_mgmt_to_staging

# Master schema dependency
load_txt_csv_sources_to_staging >> create_master_schema
parse_finwire >> create_master_schema
load_customer_mgmt_to_staging >> create_master_schema

# Transformation/Loading to master dependency
create_master_schema >> load_master_tradetype
create_master_schema >> load_master_statustype
create_master_schema >> load_master_taxrate
create_master_schema >> load_master_industry
create_master_schema >> transform_load_master_dimdate
create_master_schema >> transform_load_master_dimtime
create_master_schema >> transform_load_master_dimcompany
transform_load_master_dimcompany >> load_master_dimessages_dimcompany
transform_load_master_dimdate >> transform_load_master_dimbroker
transform_load_master_dimdate >> transform_load_master_prospect
load_master_taxrate >> transform_load_master_dimcustomer
transform_load_master_prospect >> transform_load_master_dimcustomer
transform_load_master_dimcustomer >> load_master_dimessages_dimcustomer
transform_load_master_dimcustomer >> update_master_prospect
transform_load_master_dimbroker >> transform_load_master_dimaccount
transform_load_master_dimcustomer >> transform_load_master_dimaccount
transform_load_master_dimcompany >> transform_load_master_dimsecurity
transform_load_master_dimaccount >> transform_load_master_dimtrade
load_master_statustype >> transform_load_master_dimtrade
load_master_tradetype >> transform_load_master_dimtrade
transform_load_master_dimsecurity >> transform_load_master_dimtrade
transform_load_master_dimtrade >> load_master_dimessages_dimtrade
transform_load_master_dimcompany >> transform_load_master_financial
transform_load_master_dimaccount >> transform_load_master_factcashbalances
transform_load_master_dimdate >> transform_load_master_factcashbalances
transform_load_master_dimtrade >> transform_load_master_factholdings
transform_load_master_dimcustomer >> transform_load_master_factwatches
transform_load_master_dimsecurity >> transform_load_master_factwatches
transform_load_master_dimdate >> transform_load_master_factwatches
transform_load_master_dimdate >> transform_load_master_factmarkethistory
transform_load_master_financial >> transform_load_master_factmarkethistory
transform_load_master_dimcompany >> transform_load_master_factmarkethistory
transform_load_master_dimsecurity >> transform_load_master_factmarkethistory
transform_load_master_factmarkethistory >> load_master_dimessages_factmarkethistory
```

Listing A.2: load_master_historical_dag.py

Appendix B

SQL Scripts

```sql
drop schema if exists staging cascade;
create schema staging authorization postgres;

drop table if exists staging.batchdate;
create table staging.batchdate(
  batchdate date not null
);

drop table if exists staging.cashtransaction;
create table staging.cashtransaction(
  ct_ca_id numeric(11) not null check(ct_ca_id >= 0),
  ct_dts timestamp not null,
  ct_amt numeric(10, 2) not null,
  ct_name char(100) not null
);

drop table if exists staging.customermgmt;
create table staging.customermgmt(
  --action element
  actiontype char(9) check(actiontype in ('NEW','ADDACCT','UPDCUST','UPDACCT','CLOSEACCT','INACT')),
  actionts varchar check(length(actionts) > 0),
  --action.customer element
  c_id numeric(11) not null check(c_id >= 0),
  c_tax_id char(20) check((actiontype = 'NEW' and length(c_tax_id) > 0) or (actiontype != 'NEW')),
  c_gndr char(1) check(length(c_gndr) > 0),
  c_tier numeric(1) check(c_tier >= 0),
  c_dob date check((actiontype = 'NEW' and c_dob is not null) or (actiontype != 'NEW')),
  --action.customer.name element
  c_l_name char(25) check((actiontype = 'NEW' and length(c_l_name) > 0) or (actiontype != 'NEW')),
  c_f_name char(20) check((actiontype = 'NEW' and length(c_f_name) > 0) or (actiontype != 'NEW')),
  c_m_name char(1),
  --action.customer.address element
  c_adline1 char(80) check((actiontype = 'NEW' and length(c_adline1) > 0) or (actiontype != 'NEW')),
  c_adline2 char(80),
  c_zipcode char(12) check((actiontype = 'NEW' and length(c_zipcode) > 0) or (actiontype != 'NEW')),
  c_city char(25) check((actiontype = 'NEW' and length(c_city) > 0) or (actiontype != 'NEW')),
  c_state_prov char(20) check((actiontype = 'NEW' and length(c_state_prov) > 0) or (actiontype != 'NEW')),
  c_ctry char(24),
  --action.customer.contactinfo element
  c_prim_email char(50),
  c_alt_email char(50),
  --action.customer.contactinfo.phone element
  --phone1
  c_p_1_ctry_code char(20),
  c_p_1_area_code char(20),
  c_p_1_local char(20),
  c_p_1_ext char(20),
  --phone2
  c_p_2_ctry_code char(20),
  c_p_2_area_code char(20),
  c_p_2_local char(20),
  c_p_2_ext char(20),
  --phone3
  c_p_3_ctry_code char(20),
  c_p_3_area_code char(20),
  c_p_3_local char(20),
  c_p_3_ext char(20),
  --action.customer.taxinfo element
  c_lcl_tx_id char(4),
  c_nat_tx_id char(4),
  --action.customer.account attribute
  ca_id numeric(11),
  ca_tax_st numeric(1) check((actiontype = 'NEW' and ca_tax_st >= 0) or (actiontype != 'NEW')),
  --action.customer.account element
  ca_b_id numeric(11) check((actiontype = 'NEW' and ca_b_id >= 0) or (actiontype != 'NEW')),
  ca_name char(50)
```

```sql
);

drop table if exists staging.dailymarket;
create table staging.dailymarket(
  dm_date date not null,
  dm_s_symb char(15) not null,
  dm_close numeric(8, 2) not null,
  dm_high numeric(8, 2) not null,
  dm_low numeric(8, 2) not null,
  dm_vol numeric(12) not null check(dm_vol >= 0)
);

drop table if exists staging.date;
create table staging.date(
  sk_dateid numeric(11) not null check(sk_dateid >= 0),
  datevalue char(20) not null,
  datedesc char(20) not null,
  calendaryearid numeric(4) not null check(calendaryearid >= 0),
  calendaryeardesc char(20) not null,
  calendarqtrid numeric(5) not null check(calendarqtrid >= 0),
  calendarqtrdesc char(20) not null,
  calendarmonthid numeric(6) not null check(calendarmonthid >= 0),
  calendarmonthdesc char(20) not null,
  calendarweekid numeric(6) not null check(calendarweekid >= 0),
  calendarweekdesc char(20) not null,
  dayofweeknum numeric(1) not null check(dayofweeknum >= 0),
  dayofweekdesc char(10) not null,
  fiscalyearid numeric(4) not null check(fiscalyearid >= 0),
  fiscalyeardesc char(20) not null,
  fiscalqtrid numeric(5) not null check(fiscalqtrid >= 0),
  fiscalqtrdesc char(20) not null,
  holidayflag boolean
);

drop table if exists staging.finwire;
create table staging.finwire(
  text varchar
);

drop table if exists staging.finwire_cmp;
create table staging.finwire_cmp(
  pts char(15) check(length(pts) > 0),
  rectype char(3) check(length(rectype) > 0),
  companyname char(60) check(length(companyname) > 0),
  cik char(10) check(length(cik) > 0),
  status char(4) check(length(status) > 0),
  industryid char(2) check(length(industryid) > 0),
  sprating char(4) check(length(sprating) > 0),
  foundingdate char(8),
  addressline1 char(80) check(length(addressline1) > 0),
  addressline2 char(80),
  postalcode char(12) check(length(postalcode) > 0),
  city char(25) check(length(city) > 0),
  stateprovince char(20) check(length(stateprovince) > 0),
  country char(24),
  ceoname char(46) check(length(ceoname) > 0),
  description char(150) check(length(description) > 0)
);

drop table if exists staging.finwire_sec;
create table staging.finwire_sec(
  pts char(15) check(length(pts) > 0),
  rectype char(3) check(length(rectype) > 0),
  symbol char(15) check(length(symbol) > 0),
  issuetype char(6) check(length(issuetype) > 0),
  status char(4) check(length(status) > 0),
  name char(70) check(length(name) > 0),
  exid char(6) check(length(exid) > 0),
  shout char(13) check(length(shout) > 0),
  firsttradedate char(8) check(length(firsttradedate) > 0),
  firsttradeexchg char(8) check(length(firsttradeexchg) > 0),
  dividend char(12) check(length(dividend) > 0),
  conameorcik char(60) check(length(conameorcik) > 0)
);

drop table if exists staging.finwire_fin;
create table staging.finwire_fin(
  pts char(15) check(length(pts) > 0),
  rectype char(3) check(length(rectype) > 0),
  year char(4) check(length(year) > 0),
  quarter char(1) check(length(quarter) > 0),
  qtrstartdate char(8) check(length(qtrstartdate) > 0),
  postingdate char(8) check(length(postingdate) > 0),
  revenue char(17) check(length(revenue) > 0),
  earnings char(17) check(length(earnings) > 0),
  eps char(12) check(length(eps) > 0),
  dilutedeps char(12) check(length(dilutedeps) > 0),
  margin char(12) check(length(margin) > 0),
  inventory char(17) check(length(inventory) > 0),
  assets char(17) check(length(assets) > 0),
  liability char(17) check(length(liability) > 0),
  shout char(13) check(length(shout) > 0),
  dilutedshout char(13) check(length(dilutedshout) > 0),
  conameorcik char(60) check(length(conameorcik) > 0)
);

drop table if exists staging.holdinghistory;
create table staging.holdinghistory(
  hh_h_t_id numeric(15) not null check(hh_h_t_id >= 0),
  hh_t_id numeric(15) not null check(hh_t_id >= 0),
  hh_before_qty numeric(6) not null check(hh_before_qty >= 0),
  hh_after_qty numeric(6) not null check(hh_after_qty >= 0)
);

drop table if exists staging.hr;
create table staging.hr(
  employeeid numeric(11) not null check(employeeid >= 0),
  managerid numeric(11) not null check(managerid >= 0),
  employeefirstname char(30) not null,
  employeelastname char(30) not null,
  employeemi char(1),
  employeejobcode numeric(3) check(employeejobcode >= 0),
  employeebranch char(30),
  employeeoffice char(10),
  employeephone char(14)
);

drop table if exists staging.industry;
create table staging.industry(
  in_id char(2) not null,
  in_name char(50) not null,
  in_sc_id char(4) not null
);

drop table if exists staging.prospect;
create table staging.prospect(
  agencyid char(30) not null,
  lastname char(30) not null,
  firstname char(30) not null,
  middleinitial char(1),
  gender char(1),
  addressline1 char(80),
  addressline2 char(80),
  postalcode char(12),
  city char(25) not null,
  state char(20) not null,
  country char(24),
  phone char(30),
  income numeric(9) check(income >= 0),
  numbercars numeric(2) check(numbercars >= 0),
  numberchildren numeric(2) check(numberchildren >= 0),
  maritalstatus char(1),
  age numeric(3) check(age >= 0),
  creditrating numeric(4) check(creditrating >= 0),
  ownorrentflag char(1),
  employer char(30),
  numbercreditcards numeric(2) check(numbercreditcards >= 0),
  networth numeric(12) check(networth >= 0)
);

drop table if exists staging.statustype;
create table staging.statustype(
  st_id char(4) not null,
  st_name char(10) not null
);

drop table if exists staging.taxrate;
create table staging.taxrate(
  tx_id char(4) not null,
  tx_name char(50) not null,
  tx_rate numeric(6,5) not null check(tx_rate >= 0)
);

drop table if exists staging.time;
create table staging.time(
  sk_timeid numeric(11) not null check(sk_timeid >= 0),
  timevalue char(20) not null,
  hourid numeric(2) not null check(hourid >= 0),
  hourdesc char(20) not null,
  minuteid numeric(2) not null check(minuteid >= 0),
  minutedesc char(20) not null,
  secondid numeric(2) not null check(secondid >= 0),
  seconddesc char(20) not null,
  markethoursflag boolean,
  officehoursflag boolean
);

drop table if exists staging.tradehistory;
create table staging.tradehistory(
  th_t_id numeric(15) not null check(th_t_id >= 0),
  th_dts timestamp not null,
  th_st_id char(4) not null
);

drop table if exists staging.trade;
create table staging.trade(
  t_id numeric(15) not null check(t_id >= 0),
  t_dts timestamp not null,
  t_st_id char(4) not null,
  t_tt_id char(3) not null,
  t_is_cash integer check(t_is_cash in (0, 1)),
  t_s_symb char(15) not null,
  t_qty numeric(6) check(t_qty >= 0),
  t_bid_price numeric(8,2) check(t_bid_price >= 0),
  t_ca_id numeric(11) not null check(t_ca_id >= 0),
  t_exec_name char(49) not null,
  t_trade_price numeric(8,2) check((t_st_id = 'CMPT' and t_trade_price >= 0) or (t_st_id != 'CMPT' and t_trade_price is null)),
```

```sql
  t_chrg numeric(10,2) check((t_st_id = 'CMPT' and t_chrg >= 0) or (t_st_id != 'CMPT' and t_chrg is null)),
  t_comm numeric(10,2) check((t_st_id = 'CMPT' and t_comm >= 0) or (t_st_id != 'CMPT' and t_comm is null)),
  t_tax numeric(10,2) check((t_st_id = 'CMPT' and t_tax >= 0) or (t_st_id != 'CMPT' and t_tax is null))
);

drop table if exists staging.tradetype;
create table staging.tradetype(
  tt_id char(3) not null,
  tt_name char(12) not null,
  tt_is_sell numeric(1) not null check(tt_is_sell >= 0),
  tt_is_mrkt numeric(1) not null check(tt_is_mrkt >= 0)
);

drop table if exists staging.watchhistory;
create table staging.watchhistory(
  w_c_id numeric(11) not null check(w_c_id >= 0),
  w_s_symb char(15) not null,
  w_dts timestamp not null,
  w_action char(4) check(w_action in ('ACTV', 'CNCL'))
);

drop table if exists staging.audit;
create table staging.audit(
  dataset char(20) not null,
  batchid numeric(5) check(batchid >= 0),
  date date,
  attribute char(50) not null,
  value numeric(15),
  dvalue numeric(15,5)
);
```

Listing B.1: create_staging_schema.sql

```sql
drop schema if exists master cascade;
create schema master authorization postgres;

drop table if exists master.tradetype;
create table master.tradetype(
  tt_id char(3) not null,
  tt_name char(12) not null,
  tt_is_sell numeric(1) not null check(tt_is_sell >= 0),
  tt_is_mrkt numeric(1) not null check(tt_is_mrkt >= 0)
);

drop table if exists master.statustype;
create table master.statustype(
  st_id char(4) not null,
  st_name char(10) not null
);

drop table if exists master.taxrate;
create table master.taxrate(
  tx_id char(4) not null,
  tx_name char(50) not null,
  tx_rate numeric(6,5) not null check(tx_rate >= 0)
);

drop table if exists master.industry;
create table master.industry(
  in_id char(2) not null,
  in_name char(50) not null,
  in_sc_id char(4) not null
);

drop table if exists master.dimdate;
create table master.dimdate(
  sk_dateid numeric(11) not null check(sk_dateid >= 0),
  datevalue date not null,
  datedesc char(20) not null,
  calendaryearid numeric(4) not null check(calendaryearid >= 0),
  calendaryeardesc char(20) not null,
  calendarqtrid numeric(5) not null check(calendarqtrid >= 0),
  calendarqtrdesc char(20) not null,
  calendarmonthid numeric(6) not null check(calendarmonthid >= 0),
  calendarmonthdesc char(20) not null,
  calendarweekid numeric(6) not null check(calendarweekid >= 0),
  calendarweekdesc char(20) not null,
  dayofweeknum numeric(1) not null check(dayofweeknum >= 0),
  dayofweekdesc char(10) not null,
  fiscalyearid numeric(4) not null check(fiscalyearid >= 0),
  fiscalyeardesc char(20) not null,
  fiscalqtrid numeric(5) not null check(fiscalqtrid >= 0),
  fiscalqtrdesc char(20) not null,
  holidayflag boolean
);

drop table if exists master.dimtime;
create table master.dimtime(
  sk_timeid numeric(11) not null check(sk_timeid >= 0),
  timevalue time not null,
  hourid numeric(2) not null check(hourid >= 0),
  hourdesc char(20) not null,
  minuteid numeric(2) not null check(minuteid >= 0),
  minutedesc char(20) not null,
  secondid numeric(2) not null check(secondid >= 0),
  seconddesc char(20) not null,
  markethoursflag boolean,
  officehoursflag boolean
);

drop table if exists master.dimcompany;
create table master.dimcompany(
  sk_companyid numeric(11) not null check(sk_companyid >= 0),
  companyid numeric(11) not null check(companyid >= 0),
  status char(10) not null,
  name char(60) not null,
  industry char(50) not null,
  sprating char(4),
  islowgrade boolean,
  ceo char(100) not null,
  addressline1 char(80),
  addressline2 char(80),
  postalcode char(12) not null,
  city char(25) not null,
  stateprov char(20) not null,
  country char(24),
  description char(150) not null,
  foundingdate date,
  iscurrent boolean not null,
  batchid numeric(5) not null check(batchid >= 0),
  effectivedate date not null,
  enddate date not null
);

drop table if exists master.dimbroker;
create table master.dimbroker(
  sk_brokerid numeric(11) not null check(sk_brokerid >= 0),
  brokerid numeric(11) not null check(brokerid >= 0),
  managerid numeric(11) check(managerid >= 0),
  firstname char(50) not null,
  lastname char(50) not null,
  middleinitial char(1),
  branch char(50),
  office char(50),
  phone char(14),
  iscurrent boolean not null,
  batchid numeric(5) not null check(batchid >= 0),
  effectivedate date not null,
  enddate date not null
);

drop table if exists master.dimcustomer;
create table master.dimcustomer(
  sk_customerid numeric(11) not null check(sk_customerid >= 0),
  customerid numeric(11) not null check(customerid >= 0),
  taxid char(20) not null,
  status char(10) not null,
  lastname char(30) not null,
  firstname char(20) not null,
  middleinitial char(1),
  gender char(1),
  tier numeric(1) check(tier >= 0),
  dob date not null,
  addressline1 char(80) not null,
  addressline2 char(80),
  postalcode char(12) not null,
  city char(25) not null,
  stateprov char(20) not null,
  country char(24),
  phone1 char(30),
  phone2 char(30),
  phone3 char(30),
  email1 char(50),
  email2 char(50),
  nationaltaxratedesc char(50),
  nationaltaxrate numeric(6,5) check(nationaltaxrate >= 0),
  localtaxratedesc char(50),
  localtaxrate numeric(6,5) check(localtaxrate >= 0),
  agencyid char(30),
  creditrating numeric(5) check(creditrating >= 0),
  networth numeric(10),
  marketingnameplate char(100),
  iscurrent boolean not null,
  batchid numeric(5) not null check(batchid >= 0),
  effectivedate date not null,
  enddate date not null
);

drop table if exists master.dimaccount;
create table master.dimaccount(
  sk_accountid numeric(11) not null check(sk_accountid >= 0),
  accountid numeric(11) not null check(accountid >= 0),
  sk_brokerid numeric(11) not null check(sk_brokerid >= 0),
  sk_customerid numeric(11) not null check(sk_customerid >= 0),
  status char(10) not null,
  accountdesc char(50),
  taxstatus numeric(1) check(taxstatus in(0, 1, 2)),
  iscurrent boolean not null,
  batchid numeric(5) not null check(batchid >= 0),
  effectivedate date not null,
  enddate date not null
);

drop table if exists master.dimsecurity;
```

```
create table master.dimsecurity(
  sk_securityid numeric(11) not null check(sk_securityid >= 0),
  symbol char(15) not null,
  issue char(6) not null,
  status char(10) not null,
  name char(70) not null,
  exchangeid char(6) not null,
  sk_companyid numeric(11) not null check(sk_companyid >= 0),
  sharesoutstanding numeric(12) not null check(sharesoutstanding >= 0),
  firsttrade date not null,
  firsttradeonexchange date not null,
  dividend numeric(10,2) not null,
  iscurrent boolean not null,
  batchid numeric(5) not null check(batchid >= 0),
  effectivedate date not null,
  enddate date not null
);

drop table if exists master.dimtrade;
create table master.dimtrade(
  tradeid numeric(11) not null check(tradeid >= 0),
  sk_brokerid numeric(11) check(sk_brokerid >= 0),
  sk_createdateid numeric(11) not null check(sk_createdateid >= 0),
  sk_createtimeid numeric(11) not null check(sk_createtimeid >= 0),
  sk_closedateid numeric(11) check(sk_closedateid >= 0),
  sk_closetimeid numeric(11) check(sk_closetimeid >= 0),
  status char(10) not null,
  type char(12) not null,
  cashflag boolean not null,
  sk_securityid numeric(11) not null check(sk_securityid >= 0),
  sk_companyid numeric(11) not null check(sk_companyid >= 0),
  quantity numeric(6, 0) not null check(quantity >= 0),
  bidprice numeric(8, 2) not null check(bidprice >= 0),
  sk_customerid numeric(11) not null check(sk_customerid >= 0),
  sk_accountid numeric(11) not null check(sk_accountid >= 0),
  executedby char(64) not null,
  tradeprice numeric(8,2) check(tradeprice >= 0),
  fee numeric(10,2) check(fee >= 0),
  commission numeric(10,2) check(commission >= 0),
  tax numeric(10,2) check(tax >= 0),
  batchid numeric(5) not null check(batchid >= 0)
);

drop table if exists master.financial;
create table master.financial(
  sk_companyid numeric(11) not null check(sk_companyid >= 0),
  fi_year numeric(4) not null check(fi_year >= 0),
  fi_qtr numeric(1) not null check(fi_qtr >= 0),
  fi_qtr_start_date date not null,
  fi_revenue numeric(15, 2) not null,
  fi_net_earn numeric(15, 2) not null,
  fi_basic_eps numeric(10, 2) not null,
  fi_dilut_eps numeric(10, 2) not null,
  fi_margin numeric(10, 2) not null,
  fi_inventory numeric(15, 2) not null,
  fi_assets numeric(15, 2) not null,
  fi_liability numeric(15, 2) not null,
  fi_out_basic numeric(12) not null,
  fi_out_dilut numeric(12) not null
);

drop table if exists master.factcashbalances;
create table master.factcashbalances(
  sk_customerid numeric(11) not null check(sk_customerid >= 0),
  sk_accountid numeric(11) not null check(sk_accountid >= 0),
  sk_dateid numeric(11) not null check(sk_dateid >= 0),
  cash numeric(15, 2) not null,
  batchid numeric(5) not null check(batchid >= 0)
);

drop table if exists master.factholdings;
create table master.factholdings(
  tradeid numeric(11) not null check(tradeid >= 0),
  currenttradeid numeric(11) not null check(currenttradeid >= 0),
  sk_customerid numeric(11) not null check(sk_customerid >= 0),
  sk_accountid numeric(11) not null check(sk_accountid >= 0),
  sk_securityid numeric(11) not null check(sk_securityid >= 0),
  sk_companyid numeric(11) not null check(sk_companyid >= 0),
  sk_dateid numeric(11) not null check(sk_dateid >= 0),
  sk_timeid numeric(11) not null check(sk_timeid >= 0),
  currentprice numeric(8, 2) not null check(currentprice >= 0),
  currentholding numeric(6) not null,
  batchid numeric(5) not null check(batchid >= 0)
);

drop table if exists master.factmarkethistory;
create table master.factmarkethistory(
  sk_securityid numeric(11) not null check(sk_securityid >= 0),
  sk_companyid numeric(11) not null check(sk_companyid >= 0),
  sk_dateid numeric(11) not null check(sk_dateid >= 0),
  peratio numeric(10, 2) check(peratio >= 0),
  yield numeric(5, 2) not null check(yield >= 0),
  fiftytwoweekhigh numeric(8, 2) not null check(fiftytwoweekhigh >= 0),
  sk_fiftytwoweekhighdate numeric(11) not null check(sk_fiftytwoweekhighdate >= 0),
  fiftytwoweeklow numeric(8, 2) not null check(fiftytwoweeklow >= 0),
  sk_fiftytwoweeklowdate numeric(11) not null check(sk_fiftytwoweeklowdate >= 0),
  closeprice numeric(8, 2) not null check(closeprice >= 0),
  dayhigh numeric(8, 2) not null check(dayhigh >= 0),
  daylow numeric(8, 2) not null check(daylow >= 0),
```

```sql
  volume numeric(12) not null check(volume >= 0),
  batchid numeric(5) not null check(batchid >= 0)
);

drop table if exists master.factwatches;
create table master.factwatches(
  sk_customerid numeric(11) not null check(sk_customerid >= 0),
  sk_securityid numeric(11) not null check(sk_securityid >= 0),
  sk_dateid_dateplaced numeric(11) not null check(sk_dateid_dateplaced >= 0),
  sk_dateid_dateremoved numeric(11) check(sk_dateid_dateremoved >= 0),
  batchid numeric(5) not null check(batchid >= 0)
);

drop table if exists master.prospect;
create table master.prospect(
  agencyid char(30) not null,
  sk_recorddateid numeric(11) not null check(sk_recorddateid >= 0),
  sk_updatedateid numeric(11) not null check(sk_updatedateid >= 0),
  batchid numeric(5) not null check(batchid >= 0),
  iscustomer boolean not null,
  lastname char(30) not null,
  firstname char(30) not null,
  middleinitial char(1),
  gender char(1),
  addressline1 char(80),
  addressline2 char(80),
  postalcode char(12),
  city char(25) not null,
  state char(20) not null,
  country char(24),
  phone char(30),
  income numeric(9) check(income >= 0),
  numbercars numeric(2) check(numbercars >= 0),
  numberchildren numeric(2) check(numbercars >= 0),
  maritalstatus char(1),
  age numeric(3) check(age >= 0),
  creditrating numeric(4) check(creditrating >= 0),
  ownorrentflag char(1),
  employer char(30),
  numbercreditcards numeric(2) check(numbercreditcards >= 0),
  networth numeric(12) check(networth >= 0),
  marketingnameplate char(100)
);

-- operational tables
drop table if exists master.audit;
create table master.audit(
  dataset char(20) not null,
  batchid numeric(5) check(batchid >= 0),
  date date,
  attribute char(50) not null,
  value numeric(15),
  dvalue numeric(15, 5)
);

drop table if exists master.dimessages;
create table master.dimessages(
  messagedateandtime timestamp not null,
  batchid numeric(5) not null check(batchid >= 0),
  messagesource char(30),
  messagetext char(50) not null,
  messagetype char(12) not null,
  messagedata char(100)
);
```

Listing B.2: create_master_schema.sql

```sql
truncate table staging.finwire_cmp;
truncate table staging.finwire_sec;
truncate table staging.finwire_fin;

CREATE OR REPLACE FUNCTION staging_finwire_split()
  RETURNS VOID
AS
$$
DECLARE
   x varchar := '';
BEGIN
    FOR x in SELECT * FROM staging.finwire LOOP
    if substring(x,16,3) = 'CMP' then
      insert into staging.finwire_cmp
        select
          nullif(trim(both from substring(x,1,15)), '') as pts,
          nullif(trim(both from substring(x,16,3)), '') as rectype,
          nullif(trim(both from substring(x,19,60)), '') as companyname,
          nullif(trim(both from substring(x,79,10)), '') as cik,
          nullif(trim(both from substring(x,89,4)), '') as status,
          nullif(trim(both from substring(x,93,2)), '') as industryid,
          nullif(trim(both from substring(x,95,4)), '') as sprating,
          nullif(trim(both from substring(x,99,8)), '') as foundingdate,
          nullif(trim(both from substring(x,107,80)), '') as addressline1,
          nullif(trim(both from substring(x,187,80)), '') as addressline2,
          nullif(trim(both from substring(x,267,12)), '') as postalcode,
          nullif(trim(both from substring(x,279,25)), '') as city,
          nullif(trim(both from substring(x,304,20)), '') as stateprovince,
          nullif(trim(both from substring(x,324,24)), '') as country,
          nullif(trim(both from substring(x,348,46)), '') as ceoname,
          nullif(trim(both from substring(x,394,150)), '') as description
        from staging.finwire limit 1;
    elsif substring(x,16,3) = 'SEC' then
      insert into staging.finwire_sec
        select
          nullif(trim(both from substring(x,1,15)), '') as pts,
          nullif(trim(both from substring(x,16,3)), '') as rectype,
          nullif(trim(both from substring(x,19,15)), '') as symbol,
          nullif(trim(both from substring(x,34,6)), '') as issuetype,
          nullif(trim(both from substring(x,40,4)), '') as status,
          nullif(trim(both from substring(x,44,70)), '') as name,
          nullif(trim(both from substring(x,114,6)), '') as exid,
          nullif(trim(both from substring(x,120,13)), '') as shout,
          nullif(trim(both from substring(x,133,8)), '') as firsttradedate,
          nullif(trim(both from substring(x,141,8)), '') as firsttradeexchg,
          nullif(trim(both from substring(x,149,12)), '') as dividend,
          nullif(trim(both from substring(x,161,60)), '') as conameorcik
        from staging.finwire limit 1;
    elsif substring(x,16,3) = 'FIN' then
      insert into staging.finwire_fin
        select
          nullif(trim(both from substring(x,1,15)), '') as pts,
          nullif(trim(both from substring(x,16,3)), '') as rectype,
          nullif(trim(both from substring(x,19,4)), '') as year,
          nullif(trim(both from substring(x,23,1)), '') as quarter,
          nullif(trim(both from substring(x,24,8)), '') as qtrstartdate,
          nullif(trim(both from substring(x,32,8)), '') as postingdate,
          nullif(trim(both from substring(x,40,17)), '') as revenue,
          nullif(trim(both from substring(x,57,17)), '') as earnings,
          nullif(trim(both from substring(x,74,12)), '') as eps,
          nullif(trim(both from substring(x,86,12)), '') as dilutedeps,
          nullif(trim(both from substring(x,98,12)), '') as margin,
          nullif(trim(both from substring(x,110,17)), '') as inventory,
          nullif(trim(both from substring(x,127,17)), '') as assets,
          nullif(trim(both from substring(x,144,17)), '') as liability,
          nullif(trim(both from substring(x,161,13)), '') as shout,
          nullif(trim(both from substring(x,174,13)), '') as dilutedshout,
          nullif(trim(both from substring(x,187,60)), '') as conameorcik
        from staging.finwire limit 1;
    end if;
    END LOOP;
END;
$$
LANGUAGE plpgsql;

SELECT staging_finwire_split() as output;
```

Listing B.3: load_staging_finwire_db.sql

```
-- tradetype
truncate table master.tradetype;
insert into master.tradetype
  select * from staging.tradetype;

-- statustype
truncate table master.statustype;
insert into master.statustype
  select * from staging.statustype;

-- taxrate
truncate table master.taxrate;
insert into master.taxrate
  select * from staging.taxrate;

-- industry
truncate table master.industry;
insert into master.industry
  select * from staging.industry;

-- dimdate
truncate table master.dimdate;
insert into master.dimdate
  select
    sk_dateid
  , datevalue::date
  , datedesc
  , calendaryearid
  , calendaryeardesc
  , calendarqtrid
  , calendarqtrdesc
  , calendarmonthid
  , calendarmonthdesc
  , calendarweekid
  , calendarweekdesc
  , dayofweeknum
  , dayofweekdesc
  , fiscalyearid
  , fiscalyeardesc
  , fiscalqtrid
  , fiscalqtrdesc
  , holidayflag
  from staging.date;

-- dimtime
truncate table master.dimtime;
insert into master.dimtime
  select
  sk_timeid,
  timevalue::time,
```

```sql
  hourid,
  hourdesc,
  minuteid,
  minutedesc,
  secondid,
  seconddesc,
  markethoursflag,
  officehoursflag
  from staging.time;
```

Listing B.4: load_master_static_tables.sql

```sql
-- dimcompany
truncate table master.dimcompany;
insert into master.dimcompany
  select
  row_number() over(order by cik) as sk,
  cik::numeric(11) as companyid,
  s.st_name as status,
  companyname as name,
  i.in_name as industry,
  (CASE
    WHEN sprating not in ('AAA','AA','AA+','AA-','A','A+','A-','BBB','BBB+','BBB-','BB','BB+','BB-','B','B+','B-','CCC','CCC+','CCC-','CC','C','D')
      THEN null
    ELSE f.sprating END) as sprating,
  (CASE
    WHEN sprating not in ('AAA','AA','AA+','AA-','A','A+','A-','BBB','BBB+','BBB-','BB','BB+','BB-','B','B+','B-','CCC','CCC+','CCC-','CC','C','D')
      THEN null
    WHEN f.sprating like 'A%' or f.sprating like 'BBB%'
      THEN false
    ELSE
      true
  END) as islowgrade,
  ceoname as ceo,
  addressline1,
  addressline2,
  postalcode,
  city,
  stateprovince,
  country,
  description,
  foundingdate::date,
  case when lead( (select batchdate from staging.batchdate) ) over ( partition by cik order by pts asc ) is null then true else false end as iscurrent,
  1 as batchid,
  left(f.pts, 8)::date as effectivedate,
  '9999-12-31'::date as enddate
  from
    staging.finwire_cmp f,
    staging.statustype s,
    staging.industry i
  where
    f.status = s.st_id
  and f.industryid = i.in_id;

-- dimessages alert for dimcompany
truncate table master.dimessages;
insert into master.dimessages
  select
  now(),
  1 as batchid,
  'DimCompany' as messagesource,
  'Invalid SPRating' as messagetext,
  'Alert' as messagetype,
  'CO_ID = ' || cik::varchar || ', CO_SP_RATE = ' || sprating::varchar
  from staging.finwire_cmp
  where sprating not in ('AAA','AA','AA+','AA-','A','A+','A-','BBB','BBB+','BBB-','BB','BB+','BB-','B','B+','B-','CCC','CCC+','CCC-','CC','C','D');

-- dimbroker
truncate table master.dimbroker;
insert into master.dimbroker
  select
  row_number() over(order by employeeid) as sk,
  employeeid as brokerid,
  managerid,
  employeefirstname,
  employeelastname,
  employeemi,
  employeebranch,
  employeeoffice,
  employeephone,
  true as iscurrent,
  1 as batchid,
  (select min(datevalue) FROM master.dimdate) as effectivedate,
  '9999-12-31'::date as enddate
  from staging.hr
  where employeejobcode = 314;

-- prospect part 1
truncate table master.prospect;
insert into master.prospect
  with date_record_id as (
    select
    dd.sk_dateid
    from master.dimdate dd
    inner join staging.batchdate bd
      on dd.datevalue = bd.batchdate
  )

  select
    p.agencyid
  , dri.sk_dateid
  , dri.sk_dateid
  , 1
  , false -- temporary before dimcustomer load dependency
  , p.lastname
  , p.firstname
  , p.middleinitial
  , p.gender
  , p.addressline1
  , p.addressline2
  , p.postalcode
  , p.city
  , p.state
  , p.country
  , p.phone
  , p.income
  , p.numbercars
  , p.numberchildren
  , p.maritalstatus
  , p.age
  , p.creditrating
  , p.ownorrentflag
  , p.employer
  , p.numbercreditcards
  , p.networth
  , nullif(btrim(btrim(btrim(btrim(btrim(
    case
    when p.networth > 1000000 or p.income > 200000
    then 'HighValue'
    else ''
    end
    || '+' ||
    case
    when p.numberchildren > 3 or p.numbercreditcards > 5
    then 'Expenses'
    else ''
    end
    , '+')
    || '+' ||
    case
    when p.age > 45
    then 'Boomer'
    else ''
    end
    , '+')
    || '+' ||
    case
    when p.income < 50000 or p.creditrating < 600 or p.networth < 100000
    then 'MoneyAlert'
    else ''
    end
    , '+')
    || '+' ||
    case
    when p.numbercars > 3 or p.numbercreditcards > 7
    then 'Spender'
    else ''
    end
    , '+')
    || '+' ||
    case
    when p.age < 25 and p.networth > 1000000
    then 'Inherited'
    else ''
    end
    , '+'), '')
  from staging.prospect p
  cross join date_record_id dri;

-- dimcustomer
truncate table master.dimcustomer;
insert into master.dimcustomer
  with customer as (
    select
      row_number() over(order by cm.c_id) as sk
    , cm.c_id
    , cm.c_tax_id
    , case
      when cm.actiontype = 'INACT' then 'INACTIVE'
      else 'ACTIVE'
      end as status
    , cm.c_l_name
    , cm.c_f_name
    , cm.c_m_name
    , case
      when upper(cm.c_gndr) = 'M' or upper(cm.c_gndr) = 'F'
      then upper(cm.c_gndr)
      else 'U'
      end as gender
    , cm.c_tier
    , cm.c_dob
    , cm.c_adline1
    , cm.c_adline2
    , cm.c_zipcode
    , cm.c_city
    , cm.c_state_prov
    , cm.c_ctry
    , case
      when cm.c_p_1_ctry_code is not null and cm.c_p_1_area_code is not null and cm.c_p_1_local is not null
      then '+' || cm.c_p_1_ctry_code || ' (' || cm.c_p_1_area_code || ') ' || cm.c_p_1_local || coalesce(cm.c_p_1_ext, '')

      when cm.c_p_1_ctry_code is null and cm.c_p_1_area_code is not null and cm.c_p_1_local is not null
      then '(' || cm.c_p_1_area_code || ') ' || cm.c_p_1_local || coalesce(cm.c_p_1_ext, '')

      when cm.c_p_1_area_code is null and cm.c_p_1_local is not null
      then cm.c_p_1_local || coalesce(cm.c_p_1_ext, '')

      else null
      end as phone1
    , case
      when cm.c_p_2_ctry_code is not null and cm.c_p_2_area_code is not null and cm.c_p_2_local is not null
      then '+' || cm.c_p_2_ctry_code || ' (' || cm.c_p_2_area_code || ') ' || cm.c_p_2_local || coalesce(cm.c_p_2_ext, '')

      when cm.c_p_2_ctry_code is null and cm.c_p_2_area_code is not null and cm.c_p_2_local is not null
      then '(' || cm.c_p_2_area_code || ') ' || cm.c_p_2_local || coalesce(cm.c_p_2_ext, '')

      when cm.c_p_2_area_code is null and cm.c_p_2_local is not null
      then cm.c_p_2_local || coalesce(cm.c_p_2_ext, '')

      else null
      end as phone2
    , case
      when cm.c_p_3_ctry_code is not null and cm.c_p_3_area_code is not null and cm.c_p_3_local is not null
      then '+' || cm.c_p_3_ctry_code || ' (' || cm.c_p_3_area_code || ') ' || cm.c_p_3_local || coalesce(cm.c_p_3_ext, '')

      when cm.c_p_3_ctry_code is null and cm.c_p_3_area_code is not null and cm.c_p_3_local is not null
      then '(' || cm.c_p_3_area_code || ') ' || cm.c_p_3_local || coalesce(cm.c_p_3_ext, '')

      when cm.c_p_3_area_code is null and cm.c_p_3_local is not null
      then cm.c_p_3_local || coalesce(cm.c_p_3_ext, '')

      else null
      end as phone3
    , cm.c_prim_email
    , cm.c_alt_email
    , ntr.tx_name as nat_tx_name
    , ntr.tx_rate as nat_tx_rate
    , ltr.tx_name as lcl_tx_name
    , ltr.tx_rate as lcl_tx_rate
    , case
      when cm.actions::date = max(cm.actions::date) over(partition by cm.c_id range between unbounded preceding and unbounded following)
      then true
      else false
      end as iscurrent
    , 1 as batchid
    , cm.actions::date as effectivedate
    , '9999-12-31'::date as enddate
    , cm.actiontype
    from staging.customermgmt cm
    cross join staging.batchdate bd
    left join master.taxrate ntr
      on cm.c_nat_tx_id = ntr.tx_id
    left join master.taxrate ltr
      on cm.c_lcl_tx_id = ltr.tx_id
    where cm.actiontype in ('NEW', 'UPDCUST', 'INACT')
  )

  , c_new as (
    select
    *
    from customer
    where actiontype = 'NEW'
  )

  , c_not_new as (
    select
      coalesce(c.sk, cn.sk) as sk
    , coalesce(c.c_id, cn.c_id) as c_id
    , coalesce(c.c_tax_id, cn.c_tax_id) as c_tax_id
    , coalesce(c.status, cn.status) as status
    , coalesce(c.c_l_name, cn.c_l_name) as c_l_name
    , coalesce(c.c_f_name, cn.c_f_name) as c_f_name
    , coalesce(c.c_m_name, cn.c_m_name) as c_m_name
    , coalesce(c.gender, cn.gender) as gender
    , coalesce(c.c_tier, cn.c_tier) as c_tier
    , coalesce(c.c_dob, cn.c_dob) as c_dob
    , coalesce(c.c_adline1, cn.c_adline1) as c_adline1
    , coalesce(c.c_adline2, cn.c_adline2) as c_adline2
    , coalesce(c.c_zipcode, cn.c_zipcode) as c_zipcode
    , coalesce(c.c_city, cn.c_city) as c_city
    , coalesce(c.c_state_prov, cn.c_state_prov) as c_state_prov
    , coalesce(c.c_ctry, cn.c_ctry) as c_ctry
    , coalesce(c.phone1, cn.phone1) as phone1
    , coalesce(c.phone2, cn.phone2) as phone2
    , coalesce(c.phone3, cn.phone3) as phone3
    , coalesce(c.c_prim_email, cn.c_prim_email) as c_prim_email
    , coalesce(c.c_alt_email, cn.c_alt_email) as c_alt_email
    , coalesce(c.nat_tx_name, cn.nat_tx_name) as nat_tx_name
    , coalesce(c.nat_tx_rate, cn.nat_tx_rate) as nat_tx_rate
    , coalesce(c.lcl_tx_name, cn.lcl_tx_name) as lcl_tx_name
    , coalesce(c.lcl_tx_rate, cn.lcl_tx_rate) as lcl_tx_rate
    , c.iscurrent
    , c.batchid
    , c.effectivedate
    , c.enddate
    , c.actiontype
    from customer c
    inner join c_new cn
      on c.c_id = cn.c_id
    where c.actiontype != 'NEW'
  )

  , c_all as (
    select * from c_new
    union all
    select * from c_not_new
  )

  , final_output as (
    select
      cm.sk
    , cm.c_id
    , cm.c_tax_id
    , cm.status
    , cm.c_l_name
    , cm.c_f_name
    , cm.c_m_name
    , cm.gender
    , cm.c_tier
    , cm.c_dob
    , cm.c_adline1
    , cm.c_adline2
    , cm.c_zipcode
    , cm.c_city
    , cm.c_state_prov
    , cm.c_ctry
    , cm.phone1
    , cm.phone2
    , cm.phone3
    , cm.c_prim_email
    , cm.c_alt_email
    , cm.nat_tx_name
    , cm.nat_tx_rate
    , cm.lcl_tx_name
    , cm.lcl_tx_rate
    , case
      when cm.effectivedate = max(cm.effectivedate) over(partition by cm.c_id range between unbounded preceding and unbounded following)
      then p.agencyid
      else null
      end as agencyid
    , case
      when cm.effectivedate = max(cm.effectivedate) over(partition by cm.c_id range between unbounded preceding and unbounded following)
      then p.creditrating
      else null
      end as creditrating
    , case
      when cm.effectivedate = max(cm.effectivedate) over(partition by cm.c_id range between unbounded preceding and unbounded following)
      then p.networth
      else null
      end as networth
    , case
      when cm.effectivedate = max(cm.effectivedate) over(partition by cm.c_id range between unbounded preceding and unbounded following)
      then p.marketingnameplate
      else null
      end as marketingnameplate
    , cm.iscurrent
    , cm.batchid
    , cm.effectivedate
    , cm.enddate
    from c_all cm
    left join master.prospect p
      on upper(cm.c_l_name) = upper(p.lastname)
      and upper(cm.c_f_name) = upper(p.firstname)
      and upper(cm.c_adline1) = upper(p.addressline1)
      and upper(cm.c_adline2) = upper(p.addressline2)
      and upper(cm.c_zipcode) = upper(p.postalcode)
  )

  select * from final_output;

-- dimessages alert for dimcustomer
insert into master.dimessages
  select
    now()
  , 1
  , 'DimCustomer'
  , 'Invalid customer tier'
```

```sql
  , 'Alert'
  , 'C_ID = ' || customerid || ', C_TIER = ' || tier
  from master.dimcustomer
  where tier not between 1 and 3;

insert into master.dimessages
  select
    now()
  , 1
  , 'DimCustomer'
  , 'DOB out of range'
  , 'Alert'
  , 'C_ID = ' || customerid || ', C_DOB = ' || dob
  from master.dimcustomer
  where dob < (select * from staging.batchdate) - interval '100 years'
  or dob > (select * from staging.batchdate);

-- update prospect
with current_active_customer as (
  select p.*
  from master.prospect p
  inner join master.dimcustomer c
  on upper(c.lastname) = upper(p.lastname)
  and upper(c.firstname) = upper(p.firstname)
  and upper(c.addressline1) = upper(p.addressline1)
  and upper(c.addressline2) = upper(p.addressline2)
  and upper(c.postalcode) = upper(p.postalcode)
  where c.status = 'ACTIVE'
  and c.iscurrent = true
)

update master.prospect
  set iscustomer = true
  where lastname in (select lastname from current_active_customer)
  and firstname in (select firstname from current_active_customer)
  and addressline1 in (select addressline1 from current_active_customer)
  and addressline2 in (select addressline2 from current_active_customer)
  and postalcode in (select postalcode from current_active_customer);

-- dimaccount
truncate table master.dimaccount;
insert into master.dimaccount
  with account as (
    select
      row_number() over(order by cm.ca_id) as sk
    , cm.ca_id
    , b.sk_brokerid
    , c.sk_customerid
    , case
      when cm.actiontype in ('NEW', 'ADDACCT', 'UPDACCT', 'UPDCUST')
      then 'ACTIVE'
      else 'INACTIVE'
      end as status
    , cm.ca_name
    , cm.ca_tax_st
    , case
      when cm.actionts::date = max(cm.actionts::date) over(partition by cm.ca_id range between unbounded preceding and unbounded following)
      then true
      else false
      end as iscurrent
    , 1 as batchid
    , cm.actionts::date as effectivedate
    , '9999-12-31'::date as enddate
    , cm.actiontype
    from staging.customermgmt cm
    cross join staging.batchdate bd
    left join master.dimbroker b
      on cm.ca_b_id = b.brokerid
    left join master.dimcustomer c
      on cm.c_id = c.customerid
      and cm.actionts::date >= c.effectivedate
      and cm.actionts::date <= c.enddate
    where cm.actiontype in ('NEW', 'ADDACCT', 'UPDACCT', 'CLOSEACCT', 'UPDCUST', 'INACT')
  )

  , ca_new as (
      select
      *
      from account
      where actiontype = 'NEW'
  )

  , ca_not_new as (
    select
      coalesce(a.sk, cn.sk) as sk
    , coalesce(a.ca_id, cn.ca_id) as ca_id
    , coalesce(a.sk_brokerid, cn.sk_brokerid) as sk_brokerid
    , coalesce(a.sk_customerid, cn.sk_customerid) as sk_customerid
    , coalesce(a.status, cn.status) as status
    , coalesce(a.ca_name, cn.ca_name) as ca_name
    , coalesce(a.ca_tax_st, cn.ca_tax_st) as ca_tax_st
    , coalesce(a.iscurrent, cn.iscurrent) as iscurrent
    , coalesce(a.batchid, cn.batchid) as batchid
    , coalesce(a.effectivedate, cn.effectivedate) as effectivedate
    , coalesce(a.enddate, cn.enddate) as enddate
    , a.actiontype
    from account a
    inner join ca_new cn
```

```
      on a.ca_id = cn.ca_id
    where a.actiontype != 'NEW'
  )

  , ca_all as (
    select * from ca_new
    union all
    select * from ca_not_new
  )

  select
    sk
  , ca_id
  , sk_brokerid
  , sk_customerid
  , status
  , ca_name
  , ca_tax_st
  , iscurrent
  , batchid
  , effectivedate
  , enddate
  from ca_all;

-- master.dimsecurity transform and load
truncate table master.dimsecurity;
insert into master.dimsecurity
  select
    row_number() over() as sk_securityid,
    symbol,
    issuetype as issue,
    s.st_name as status,
    f.name,
    exid as exchangeid,
    c.sk_companyid as sk_companyid,
    shout::numeric(12) as sharesoutstanding,
    left(firsttradedate, 8)::date,
    left(firsttradeexchg, 8)::date,
    dividend::numeric(10,2),
    case
      when lead( (select batchdate from staging.batchdate) ) over ( partition by symbol order by pts asc ) is null
      then true
      else false
      end as iscurrent,
    1 as batchid,
    left(f.pts, 8)::date,
    '9999-12-31'::date as enddate
  from staging.finwire_sec f,
    staging.statustype s,
    master.dimcompany c
  where f.status = s.st_id
  and ((ltrim(f.conameorcik, '0') = c.companyid::varchar)
    or (f.conameorcik = c.name))
  and left(pts, 8)::date >= c.effectivedate
  and left(pts, 8)::date < c.enddate;

-- transform and load
-- master.dimtrade
truncate table master.dimtrade;
insert into master.dimtrade
  with trades as (
    select
      t.t_id
    , a.sk_brokerid
    , case
      when (th.th_st_id = 'SBMT' and t.t_tt_id in ('TMB', 'TMS')) or th.th_st_id = 'PNDG'
      then to_char(th.th_dts::date, 'yyyymmdd')::numeric
      else null
      end as sk_createdateid
    , case
      when (th.th_st_id = 'SBMT' and t.t_tt_id in ('TMB', 'TMS')) or th.th_st_id = 'PNDG'
      then to_char(th.th_dts::time, 'hh24miss')::numeric
      else null
      end as sk_createtimeid
    , case
      when th.th_st_id in ('CMPT', 'CNCL')
      then to_char(th.th_dts::date, 'yyyymmdd')::numeric
      else null
      end as sk_closedateid
    , case
      when th.th_st_id in ('CMPT', 'CNCL')
      then to_char(th.th_dts::time, 'hh24miss')::numeric
      else null
      end as sk_closetimeid
    , st.st_name
    , tt.tt_name
    , case
      when t.t_is_cash = 1 then true
      else false
      end as t_is_cash
    , s.sk_securityid
    , s.sk_companyid
    , t.t_qty
    , t.t_bid_price
    , a.sk_customerid
    , a.sk_accountid
    , t.t_exec_name
    , t.t_trade_price
    , t.t_chrg
    , t.t_comm
    , t.t_tax
    , 1 as batchid
    , row_number() over(partition by t.t_id order by th.th_dts desc) as rn
    from staging.trade t
    inner join staging.tradehistory th
      on t.t_id = th.th_t_id
    inner join master.dimaccount a
      on t.t_ca_id = a.accountid
      and th.th_dts::date >= a.effectivedate
      and th.th_dts::date < a.enddate
    inner join master.statustype st
      on t.t_st_id = st.st_id
    inner join master.tradetype tt
      on t.t_tt_id = tt.tt_id
    inner join master.dimsecurity s
      on t.t_s_symb = s.symbol
      and th.th_dts::date >= s.effectivedate
      and th.th_dts::date < s.enddate
  )

  , trade_creation as (
    select
    t_id,
    min(sk_createdateid::varchar || sk_createtimeid::varchar) as trade_creation
    from trades
    group by t_id
  )

  , latest_trades as (
    select
      t.t_id
    , sk_brokerid
    , coalesce(t.sk_createdateid::varchar, left(tc.trade_creation, 8))::numeric
    , coalesce(t.sk_createtimeid::varchar, right(tc.trade_creation, -8))::numeric
    , sk_closedateid
    , sk_closetimeid
    , st_name
    , tt_name
    , t_is_cash
    , sk_securityid
    , sk_companyid
    , t_qty
    , t_bid_price
    , sk_customerid
    , sk_accountid
    , t_exec_name
    , t_trade_price
    , t_chrg
    , t_comm
    , t_tax
    , batchid
    from trades t
    left join trade_creation tc
      on t.t_id = tc.t_id
    where rn = 1
  )

  select * from latest_trades;

-- dimessages alert for dimtrade
insert into master.dimessages
  select
    now()
  , 1
  , 'DimTrade'
  , 'Invalid trade commission'
  , 'Alert'
  , 'T_ID = ' || tradeid || ', T_COMM = ' || commission
  from master.dimtrade
  where commission is not null
  and commission > (tradeprice * quantity);

insert into master.dimessages
  select
    now()
  , 1
  , 'DimTrade'
  , 'Invalid trade fee'
  , 'Alert'
  , 'T_ID = ' || tradeid || ', T_CHRG = ' || fee
  from master.dimtrade
  where fee is not null
  and fee > (tradeprice * quantity);

-- financial
truncate table master.financial;
insert into master.financial
  select
    c.sk_companyid as sk_companyid,
    year::numeric as fi_year,
    quarter::numeric as fi_qtr,
    qtrstartdate::date as fi_qtr_start_date,
    revenue::numeric as fi_revenue,
    earnings::numeric as fi_net_earn,
    eps::numeric as fi_basic_eps,
    dilutedeps::numeric as fi_dilut_eps,
    margin::numeric as fi_margin,
    inventory::numeric as fi_inventory,
    assets::numeric as fi_assets,
    liability::numeric as fi_liability,
    shout::numeric as fi_out_basic,
    dilutedshout::numeric as fi_out_dilut
  from staging.finwire_fin f,
    master.dimcompany c
  where ((f.conameorcik = c.companyid::varchar)
    or (f.conameorcik = c.name))
  and left(pts, 8)::date >= c.effectivedate
  and left(pts, 8)::date < c.enddate;

-- factcashbalances
truncate table master.factcashbalances;
insert into master.factcashbalances
  with agg as (
    select
    a.sk_customerid as sk_customerid,
    a.sk_accountid as sk_accountid,
    d.sk_dateid as sk_dateid,
    sum(ct_amt) as ct_amt_day
    from staging.cashtransaction c,
      master.dimaccount a,
      master.dimdate d
    where c.ct_ca_id = a.accountid
    and ct_dts::date >= a.effectivedate
    and ct_dts::date < a.enddate
    and ct_dts::date = d.datevalue
    group by
      a.sk_customerid,
      a.sk_accountid,
      d.sk_dateid
  )

  , final_output as (
    select
    sk_customerid,
    sk_accountid,
    sk_dateid,
    sum(ct_amt_day) over(partition by sk_accountid order by sk_dateid rows between unbounded preceding and current row) as cash,
    1 as batchid
    from agg
  )

  select * from final_output;

-- factholdings
truncate table master.factholdings;
insert into master.factholdings
  select
  h.hh_h_t_id as tradeid,
  t.tradeid as currenttradeid,
  t.sk_customerid as sk_customerid,
  t.sk_accountid as sk_accountid,
  t.sk_securityid as sk_securityid,
  t.sk_companyid as sk_companyid,
  t.sk_closedateid as sk_dateid,
  t.sk_closetimeid as sk_timeid,
  t.tradeprice as currentprice,
  h.hh_after_qty as currentholding,
  1 as batchid
  from staging.holdinghistory h,
    master.dimtrade t
  where h.hh_t_id = t.tradeid;

-- factwatches
truncate table master.factwatches;
insert into master.factwatches
  with watches as (
    select w1.w_c_id,
    TRIM(w1.w_s_symb) as w_s_symb,
    w1.w_dts::date as dateplaced,
    w2.w_dts::date as dateremoved
    from staging.watchhistory w1,
      staging.watchhistory w2
    where w1.w_c_id = w2.w_c_id
    and w1.w_s_symb = w2.w_s_symb
    and w1.w_action = 'ACTV'
    and w2.w_action = 'CNCL'
  )

  select
    c.sk_customerid as sk_customerid,
    s.sk_securityid as sk_securityid,
    to_char(w.dateplaced, 'yyyymmdd')::numeric as sk_dateid_dateplaced,
    to_char(w.dateremoved, 'yyyymmdd')::numeric as sk_dateid_dateremoved,
    1 as batchid
  from watches w,
    master.dimcustomer c,
    master.dimsecurity s,
    master.dimdate d1,
    master.dimdate d2
  where w.w_c_id = c.customerid
  and w.w_s_symb = s.symbol
  and w.dateplaced = d1.datevalue
  and w.dateremoved = d2.datevalue;

-- factmarkethistory
```

```
truncate table master.factmarkethistory;
insert into master.factmarkethistory
  with market_dates_daily as (
    select
      dm.dm_s_symb
    , dm.dm_date
    , dm.dm_close
    , dm.dm_high
    , dm.dm_low
    , dm.dm_vol
    , dd.sk_dateid
    from staging.dailymarket dm
    inner join master.dimdate dd
      on dm.dm_date = dd.datevalue
    order by
        dm.dm_s_symb
      , dm.dm_date desc
  )

  , high_low as (
    select
      dm_s_symb
    , dm_date
    , dm_close
    , dm_high
    , dm_low
    , dm_vol
    , max(dm_high) over(partition by dm_s_symb order by dm_date rows between 363 preceding and current row) as fiftytwoweekhigh
    , min(dm_low) over(partition by dm_s_symb order by dm_date rows between 363 preceding and current row) as fiftytwoweeklow
    from market_dates_daily
  )

  , high_date as (
    select
      hl.dm_s_symb
    , hl.dm_date
    , hl.dm_close
    , hl.dm_high
    , hl.dm_low
    , hl.dm_vol
    , hl.fiftytwoweekhigh
    , hl.fiftytwoweeklow
    , max(mdd.dm_date) as sk_fiftytwoweekhighdate
    from high_low hl
    inner join market_dates_daily mdd
      on hl.dm_s_symb = mdd.dm_s_symb
      and hl.fiftytwoweekhigh = mdd.dm_high
      and mdd.dm_date <= hl.dm_date
      and mdd.dm_date >= hl.dm_date - interval '52 weeks'
    group by
        hl.dm_s_symb
      , hl.dm_date
      , hl.dm_close
      , hl.dm_high
      , hl.dm_low
      , hl.dm_vol
      , hl.fiftytwoweekhigh
      , hl.fiftytwoweeklow
  )

  , low_date as (
    select
      hl.dm_s_symb
    , hl.dm_date
    , hl.dm_close
    , hl.dm_high
    , hl.dm_low
    , hl.dm_vol
    , hl.fiftytwoweekhigh
    , hl.fiftytwoweeklow
    , hl.sk_fiftytwoweekhighdate
    , max(mdd.dm_date) as sk_fiftytwoweeklowdate
    from high_date hl
    inner join market_dates_daily mdd
      on hl.dm_s_symb = mdd.dm_s_symb
      and hl.fiftytwoweeklow = mdd.dm_low
      and mdd.dm_date <= hl.dm_date
      and mdd.dm_date >= hl.dm_date - interval '52 weeks'
    group by
        hl.dm_s_symb
      , hl.dm_date
      , hl.dm_close
      , hl.dm_high
      , hl.dm_low
      , hl.dm_vol
      , hl.fiftytwoweekhigh
      , hl.fiftytwoweeklow
      , hl.sk_fiftytwoweekhighdate
  )

  , quarters as (
      select
        f.sk_companyid
      , f.fi_qtr_start_date
      , sum(fi_basic_eps) over (partition by c.companyid order by f.fi_qtr_start_date rows between 3 preceding and current row ) as eps_qtr_sum
      , lead(fi_qtr_start_date, 1, '9999-12-31'::date) over (partition by c.companyid order by f
     .fi_qtr_start_date asc) as next_qtr_start
      from master.financial f
      inner join master.dimcompany c
        on f.sk_companyid = c.sk_companyid
  )

  , final_output as (
    select
      s.sk_securityid
    , s.sk_companyid
    , to_char(ld.dm_date, 'yyyymmdd')::numeric as sk_dateid
    , case
      when q.eps_qtr_sum != 0 and q.eps_qtr_sum is not null
      then (ld.dm_close / q.eps_qtr_sum)::numeric(10, 2)
      else null
      end as peratio
    , case
      when ld.dm_close != 0
      then round((s.dividend / ld.dm_close) * 100, 2)
      else null
      end as yield
    , ld.fiftytwoweekhigh
    , to_char(ld.sk_fiftytwoweekhighdate, 'yyyymmdd')::numeric as sk_fiftytwoweekhighdate
    , ld.fiftytwoweeklow
    , to_char(ld.sk_fiftytwoweeklowdate, 'yyyymmdd')::numeric as sk_fiftytwoweeklowdate
    , ld.dm_close as closeprice
    , ld.dm_high as dayhigh
    , ld.dm_low as daylow
    , ld.dm_vol as volume
    , 1 as batchid
    from low_date ld
      inner join master.dimsecurity s
        on ld.dm_s_symb = s.symbol
        and ld.dm_date >= s.effectivedate
        and ld.dm_date < s.enddate
      inner join quarters q
        on s.sk_companyid = q.sk_companyid
        and q.fi_qtr_start_date <= ld.dm_date
        and q.next_qtr_start > ld.dm_date
  )

  select * from final_output;

-- dimessages alert for factmarkethistory
insert into master.dimessages
  select
    now()
  , 1
  , 'FactMarketHistory'
  , 'No earnings for company'
  , 'Alert'
  , 'DM_S_SYMB = ' || s.symbol
  from master.factmarkethistory fmh
  inner join master.dimsecurity s
    on fmh.sk_securityid = s.sk_securityid
  where fmh.peratio is null
  or fmh.peratio = 0;
```

Listing B.5: load_master_complex_tables.sql