

# ETL

## Pentaho Data Integration

**HITACHI**

Pentaho Data integration



### **Version: 9.4**

General Availability Release - 9.4.0.0-343

Build Date: November 8, 2022 07:50:27

Copyright (C) 2007 - 2024 Hitachi Vantara. All rights reserved.

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this application and all files except in  
compliance with the License. You may obtain a copy of the  
License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing,  
software distributed under the License is distributed on an "AS  
IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,  
either express or implied.

See the License for the specific language governing permissions  
and limitations under the License.

# History and ETL Projects

## History

- The original data warehouse structure and ETL processes were created by 4Sight (a third-party vendor).
- It appears that it was initially designed for a tiny insurance company and later adapted to be more generalized for any insurance company.
- I did not receive any support or assistance from the vendor.
- The basic solution was substantially enhanced and tailored to meet the specific requirements of CSE.

## ETL Projects

1. "SPInn" - Customized 4Sight data warehouse (DW) and ETL solution tailored to SPInn transactional system, with data staging from Microsoft SQL Server and later transitioned to AWS Aurora.
2. "SPInn Extra" - Custom DW tables and ETL process developed specifically for CSE, based on SPInn transactional system with data staging from AWS Aurora
3. "UU" - DW and ETL system developed based on text export files originating from InsurPAS (reinsurance).
4. "ICO" - DW and ETL system developed based on text export files originating from InsurPAS (reinsurance).
5. "MGA" - DW and ETL solution built upon SPInn transactional system, with data staging from AWS Aurora.

# Issues

- The original ETL process was developed for a tiny company. When I began working at CSE, the incremental daily load took an astonishing 17 hours to complete.
- There were no conditions implemented to verify if the data in the source system were ready for processing. Consequently, in case of any issues during the daily cycle, the only approach was to intervene in the middle of the night to halt a scheduled process or restore from backups and restart the following day.
- The original ETL was designed around a different insurance source system. Despite attempts to adjust staging queries, some crucial information was lost in the data warehouse.
- The database structure and ETL processes were intended to be agnostic to any specific database platform. However, instead of utilizing SQL, T-SQL or PL/SQL stored procedures, JavaScript and Pentaho Spoon "lookups" were employed. This approach resulted in slow performance and excessive memory consumption. Notably, the AWS EC2 instance used for our ETL tasks was the largest in the company.
- Numerous dimensions, metrics, columns, lookups, and JavaScript functions within the ETL were redundant and remained unused.
- Conversely, essential attributes of the CSE business model were absent. Consequently, only basic financial metrics could be provided.
- The existing implementation of SCD type 2 did not make sense. The information within these dimensions was static in CSE.

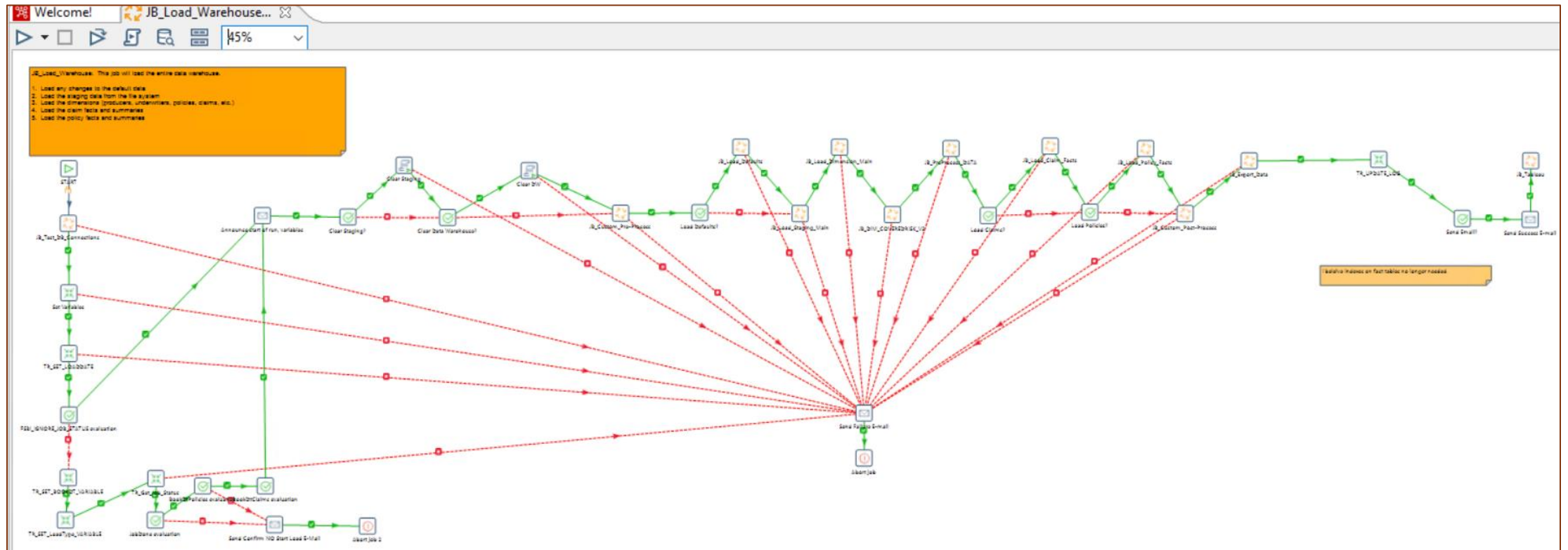
# Enhancements and Adjustments

- The first enhancement I implemented was a source system log check in a Pentaho Job to prevent false starts, along with a specialized schedule to verify source system readiness.
- I conducted a comprehensive cleanup of the existing ETL process, removing unnecessary lookups, calculations, and other artifacts.
- To boost ETL performance, I introduced parallel operations and implemented structural and database optimizations such as adding indexes and temporary tables.
- As a result of these optimizations, the ETL process now completes in 2 hours, where 1 hour is dedicated to loading data into Redshift.
- Unneeded SCD type 2 was removed and instead additional SCD2 dimensions were added in the structure. It allowed to build large number of data feeds and Tableau dashboards related to the quality of risk portfolio.
- In addition to dimensional attributes, I incorporated new metrics derived from complex calculations performed across Aurora, Microsoft SQL Server, and Redshift.
- I created 2 more, similar, but smaller, DW and ETLs for other companies' data sets based on text files.
- Recognizing the limits of further improvement within Pentaho Data Integration, I developed a new ELT process using Redshift/Matillion. Data movement from Aurora to Redshift is facilitated by Fivetran.

# ETL Steps

- Set Variables
- Test Database Connections
- Set Load Date
- Evaluate Start/No Start Automatic Load
- Set Incremental Load date range
- Load Staging from AWS Aurora
- Load Dimensions in MS SQL
- Load Policy Facts in MS SQL
- Load Claim Facts in MS SQL
- Export Data to Redshift
- Update Log tables for automatic incremental load
- Start Tableau dashboards refresh

# Load Warehouse Job



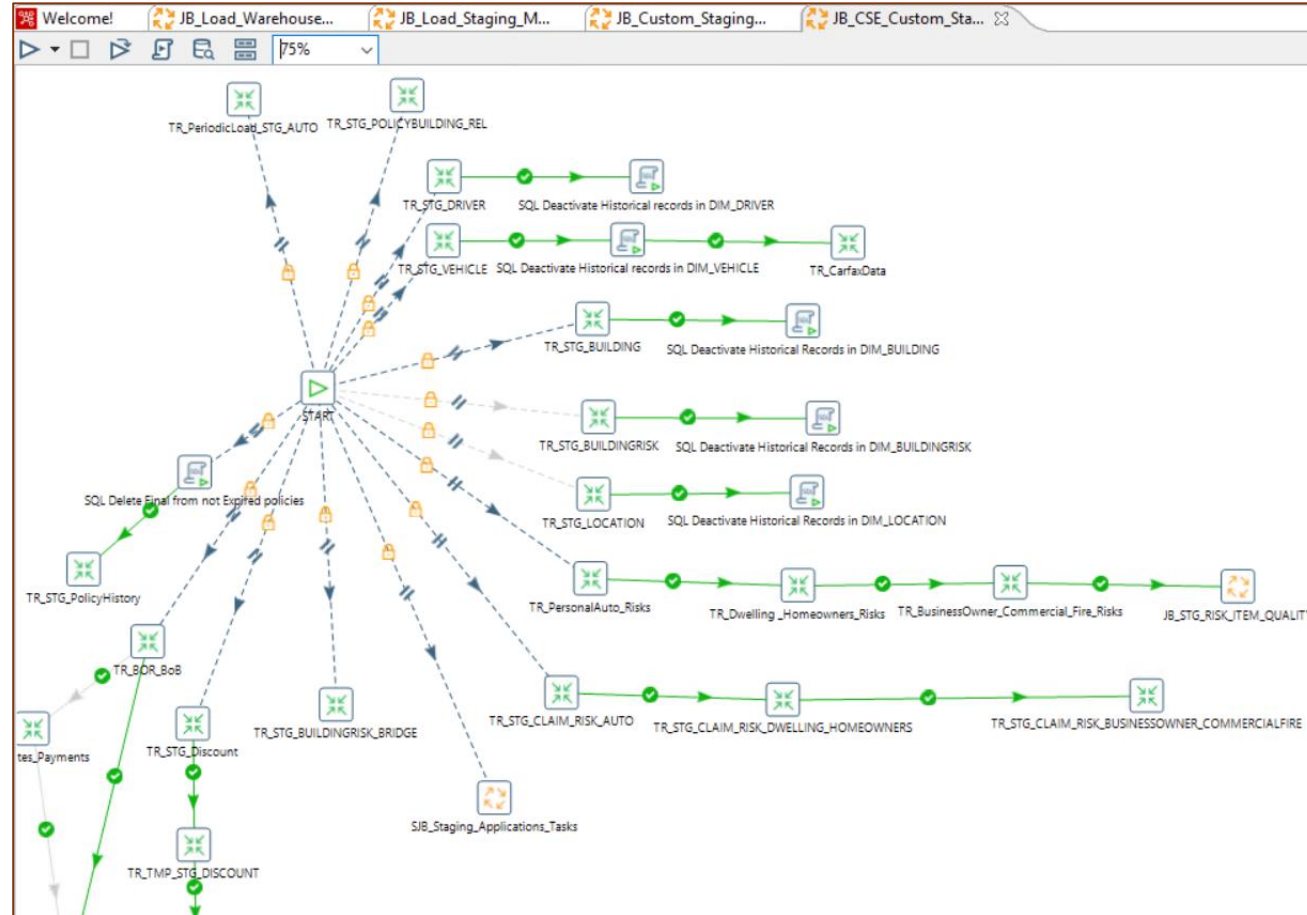
# Fragments of Configuration File

```
4sightbi_config.SPINN.properties 4sightbi_config.SPINN_EXTRA.properties 4sightbi_config.UUIICO.properties 4sightbi_config.UU.properties
38
39 FSBI_PROC_COUNT=2
40 FSBI_HIGH_PROC_COUNT=4
41
42 ## Define which parts of the ETL to run, put a Y to run or N not to run
43
44 FSBI_CLEAR_DATAWAREHOUSE=N
45 FSBI_CLEAR_STAGING=N
46 FSBI_LOAD_DEFAULTS=N
47
48 FSBI_LOAD_CUSTOM_STAGING=Y
49
50 FSBI_LOAD_DIMENSIONS=Y
51 FSBI_LOAD_CUSTOM_DIMENSIONS=Y
52 FSBI_LOAD_DIM_COVEREDRISK=Y
53
54 ## FACT_POLICYAUTO_REL FACT_POLICYBUILDING_RELO FACT_CATASTROPHY_REL
55 FSBI_PreProcess_DATA=Y
56
57 FSBI_LOAD_CLAIMS=Y
58 FSBI_LOAD_CLAIM_TRANSACTIONS=Y
59 FSBI_LOAD_CLAIM_SUMMARIES=Y
60
61 FSBI_LOAD_POLICIES=Y
62 FSBI_LOAD_POLICY_TRANSACTIONS=Y
63 FSBI_LOAD_POLICY_SUMMARIES=Y
64
65
66
67 ## FSBI_EXPORT_DATA can be one of these three values: NONE,DIFF,FULL
68 FSBI_EXPORT_DATA=DIFF
69 FSBI_UPLOAD_TO_S3=Y
70 FSBI_LOAD_REDSHIFT=Y
71
```

```
4sightbi_config.SPINN.properties 4sightbi_config.SPINN_EXTRA.properties 4sightbi_config.UUIICO.properties 4sightbi_config.UU.properties
113
114 ## Paths to file locations needed for ETLs
115
116 FSBI_LOG_PATH=C:\\4SightBI_Data\\SPINN\\Logs\\
117 FSBI_STAGING_PATH=C:\\4SightBI_Data\\SPINN\\Source_Data\\
118 FSBI_DEFAULTS_PATH=C:\\4SightBI_Data\\SPINN\\Default_Data\\
119 FSBI_SQL_SCRIPT_PATH=C:\\4SightBI_Data\\SPINN\\SQL_Scripts\\
120 FSBI_CUSTOM_SQL_SCRIPT_PATH=C:\\4SightBI_Data\\SPINN\\Custom_SQL_Scripts\\
121 FSBI_CSV_OUTPUT_PATH=E:\\Exports\\SPINN\\
122 FSBI_TEMP_DIR=E:\\temp\\SPINN\\FSBI_TEMP_DIR\\
123
124 ## Delimiter used in CSV staging files
125
126 FSBI_CSV_DELIMITER=|
127
128 ## Date format for dates in staging files
129
130 ##FSBI_DATE_FORMAT=yyyy-MM-dd HH:mm:ss.SSS
131 FSBI_DATE_FORMAT=yyyy-MM-dd
132
133 ## Default values used if data is blank/missing in staging files or tables
134
135 FSBI_DEFAULT_DATE=01/01/1900
136 FSBI_DEFAULT_NUMBER=0
137 FSBI_DEFAULT_TEXT=~
138
139 ## E-mail connection information for sending ETL success and failure messages
140
141 FSBI_EMAIL_SMTP_SERVER=email-smtp.us-west-2.amazonaws.com
142 FSBI_EMAIL_AUTHENTICATION_USER=
143 FSBI_EMAIL_AUTHENTICATION_PASSWORD=
144 FSBI_EMAIL_SMTP_PORT=465
145
146 FSBI_EMAIL_SENDER_ADDRESS=prod-etl-notifications@cseinsurance.com
147 FSBI_EMAIL_DESTINATION_ADDRESS=BITeam@cseinsurance.com
148 FSBI_EMAIL_SENDER_NAME=ETL Process (PROD-ETL SPINN)
149 FSBI_EMAIL_DESTINATION_CC_ADDRESS=
```

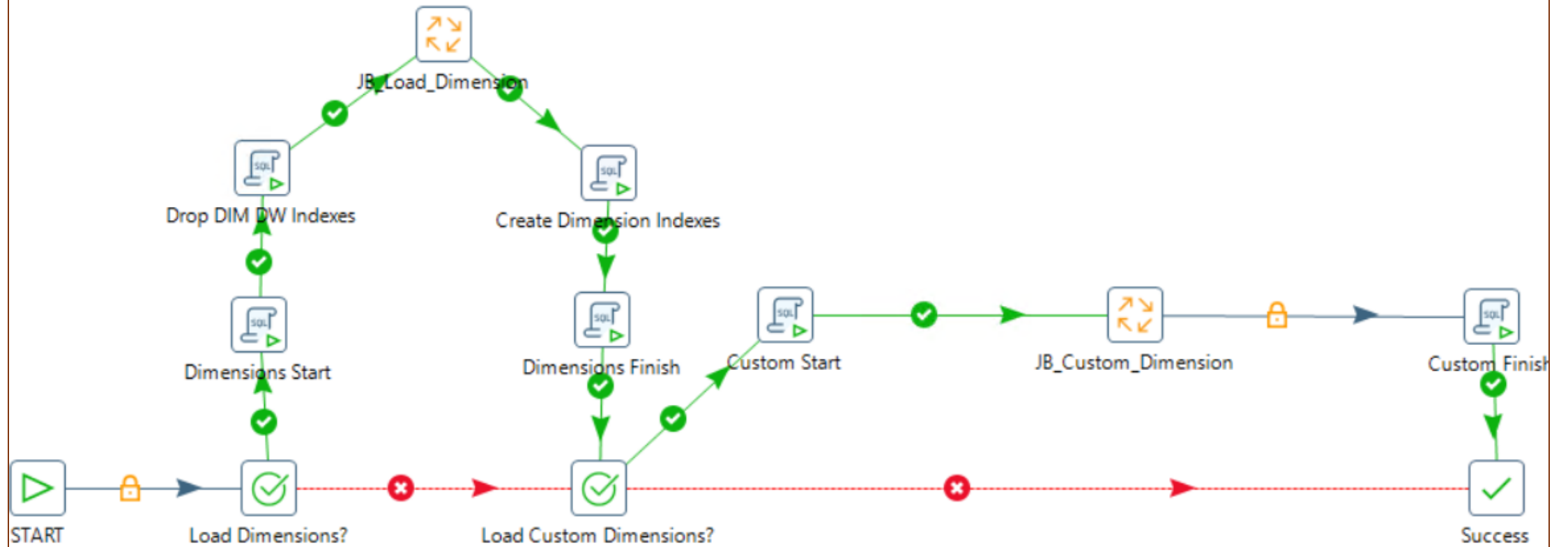


# Load Staging

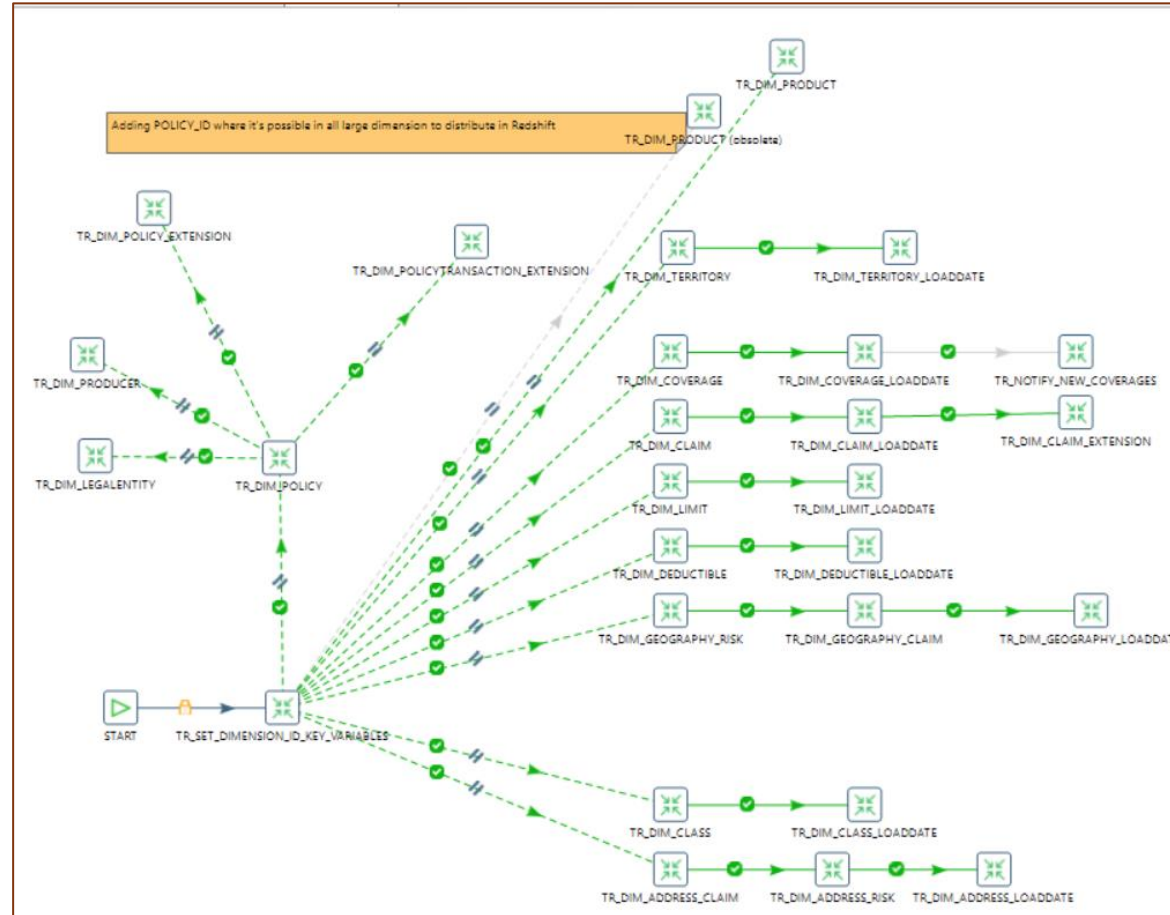




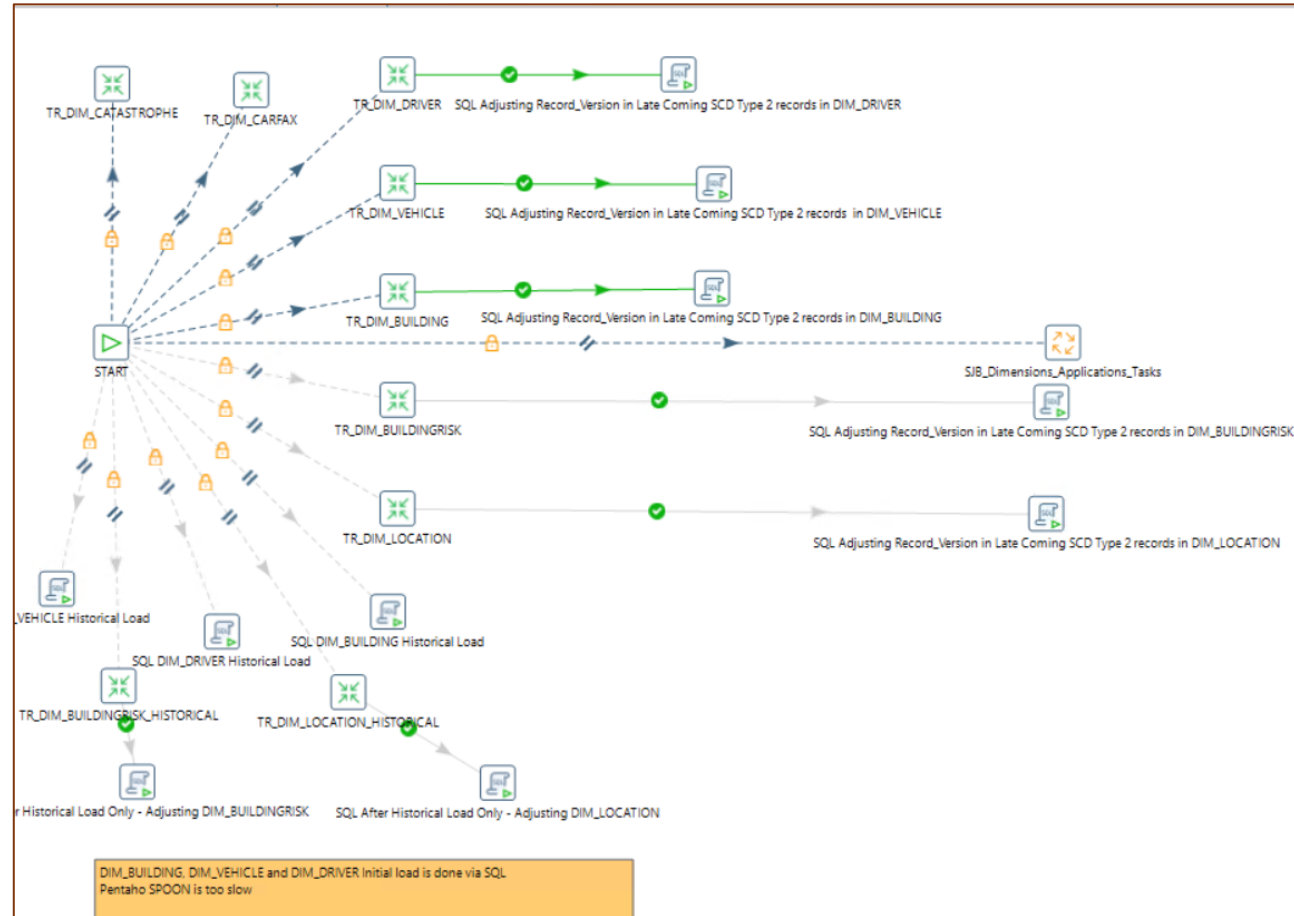
# Load Dimensions



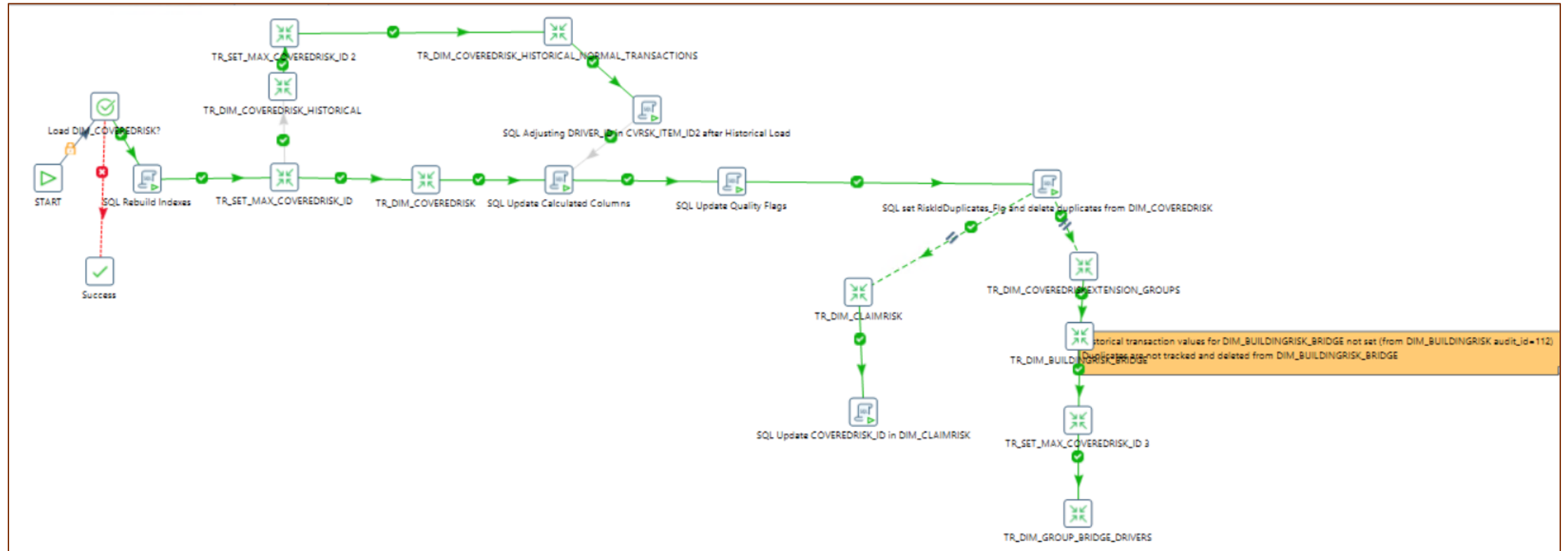
# Load Dimensions



# Load Dimensions



# Load Risk related Dimensions



# SCD Type 2 Dimension

Dimension lookup/update

Step name: DIM\_VEHICLE Dimension lookup/update

Update the dimension? ☒

Connection: 4SBI\_DataWarehouse

Target schema:

Target table: DIM\_VEHICLE

Commit size: 100000

Enable the cache? ☐

Pre-load the cache? ☐

Cache size in rows (0 = cache all):

Keys Fields

Key fields (to look up row in dimension):

#	Dimension field	Field in stream
1	Vehicle_Uniquelid	Vehicle_Uniquelid

Technical key field: VEHICLE\_ID

New name:

Creation of technical key

☒ Use table maximum + 1

☐ Use sequence

☐ Use auto increment field

Version field: RECORD\_VERSION

Stream Datefield: changeDate

Date range start field: VALID\_FROMDATE

Min. year: 1900

Help

Dimension lookup/update

Step name: DIM\_VEHICLE Dimension lookup/update

Update the dimension? ☒

Connection: 4SBI\_DataWarehouse

Target schema:

Target table: DIM\_VEHICLE

Commit size: 100000

Enable the cache? ☐

Pre-load the cache? ☐

Cache size in rows (0 = cache all):

Keys Fields

Lookup/Update fields

#	Dimension field	Stream field to compare with	Type of dimension update
1	Policy_Uniqueid	Policy_Uniqueid	Insert
2	Policy_Id	Policy_Id	Insert
3	SPInnVehicle_Id	SPInnVehicle_Id	Insert
4	Status	Status	Insert
5	stateprovcd	stateprovcd	Insert
6	county	county	Insert

Technical key field: VEHICLE\_ID

New name:

Creation of technical key

☒ Use table maximum + 1

☐ Use sequence

☐ Use auto increment field

Version field: RECORD\_VERSION

Stream Datefield: changeDate

Date range start field: VALID\_FROMDATE

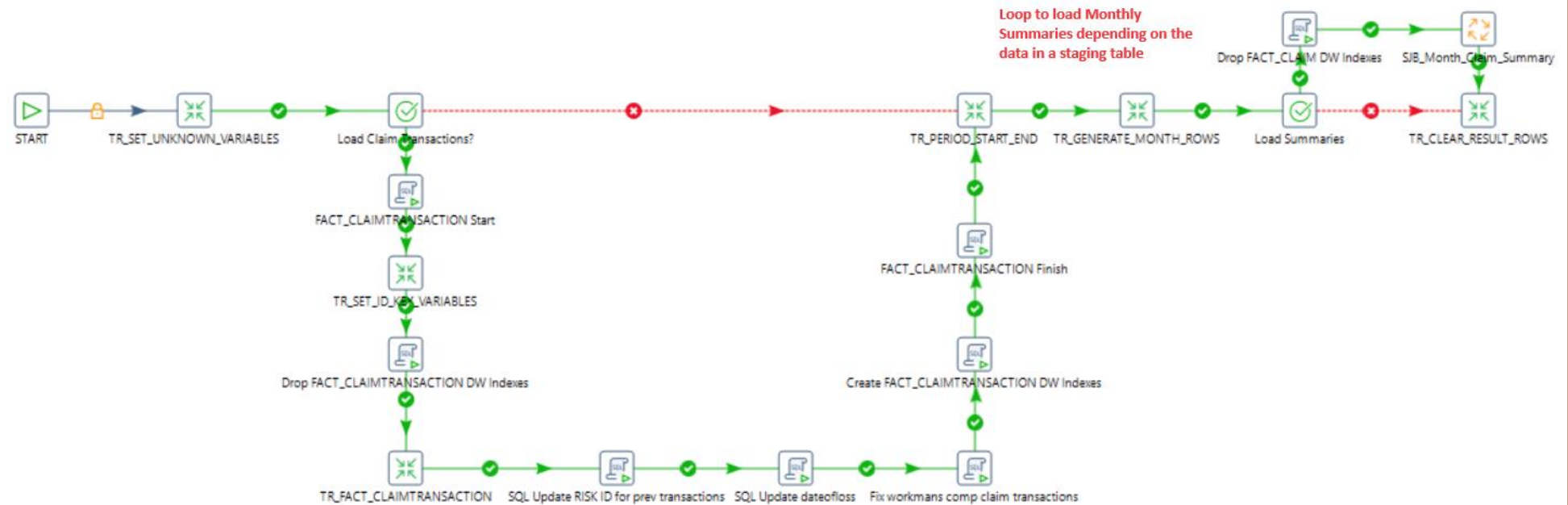
Min. year: 1900

Help

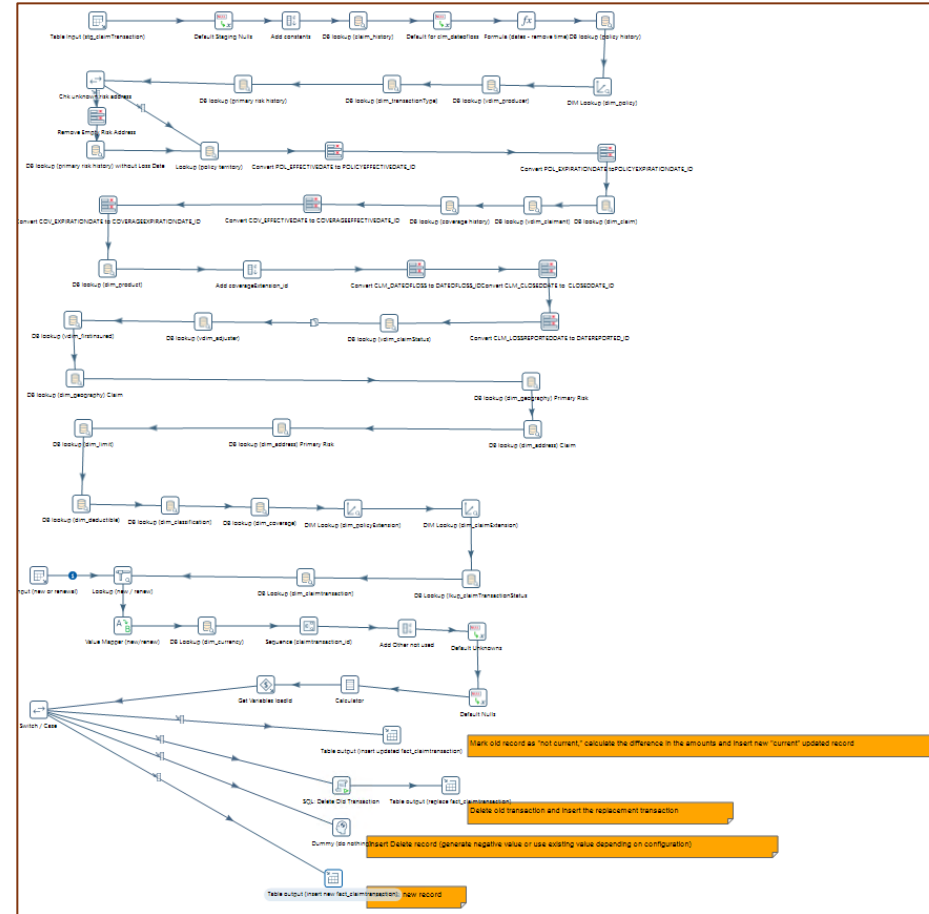
# Load Claim Fact Tables

JB\_Load\_Claim\_Facts: loads the claim transactions and claim summary.

1. Loads all of the unknown ID's in case a record is not found
2. Loads all the claim transactions from the last staging load
3. Determines the min and max accounting dates from the staging load
4. Generates a record for each month between the start and end dates
5. Runs the summary load for each record (aka month)

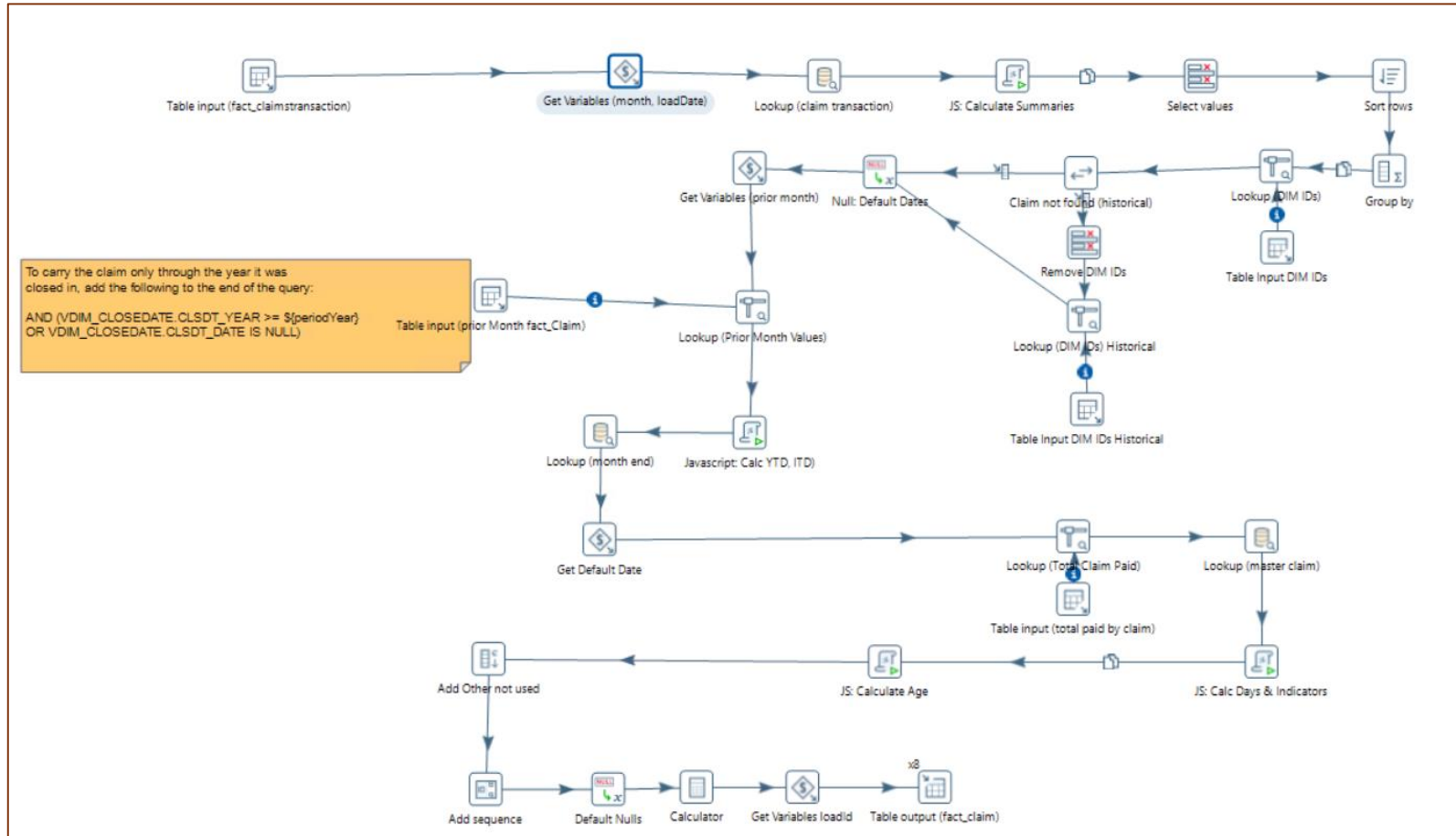


# Load Claim Fact Transactions





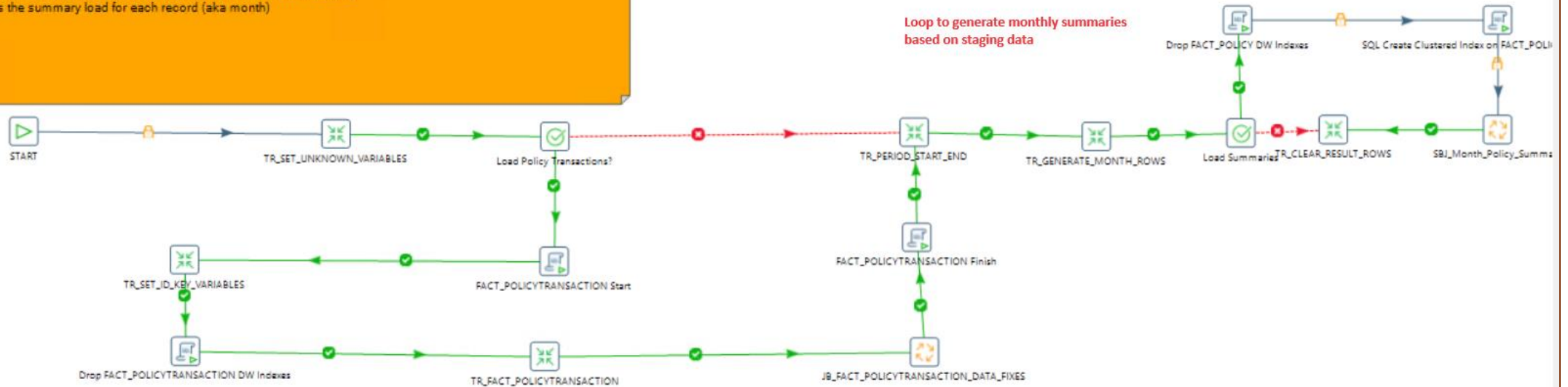
# Load Claim Fact Monthly Summaries



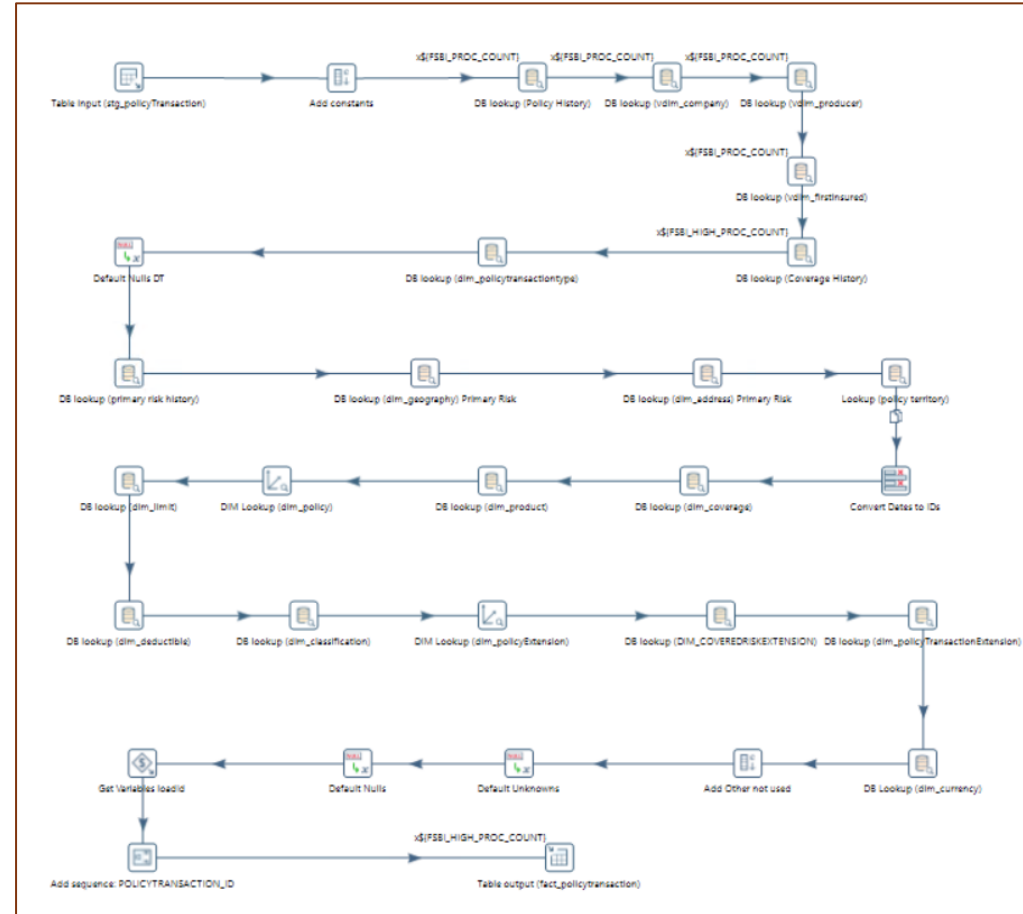
# Load Policy Fact Tables

JB\_Load\_Policy\_Facts: loads the policy transactions, policy coverage summary and policy summary tables.

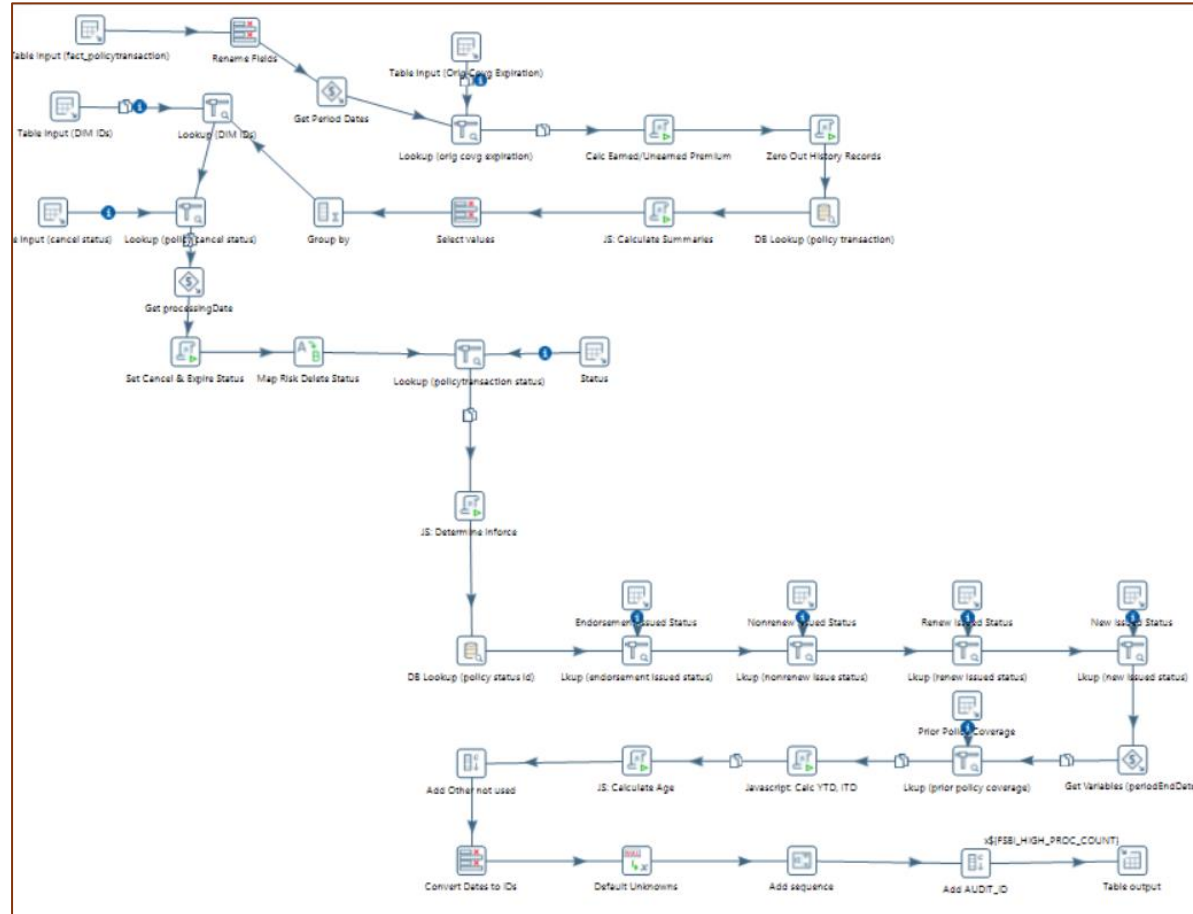
1. Loads all of the unknown ID's in case a record is not found
2. Loads all the policy transactions from the last staging load
3. Determines the min and max accounting dates from the staging load
4. Generates a record for each month between the start and end dates
5. Runs the summary load for each record (aka month)



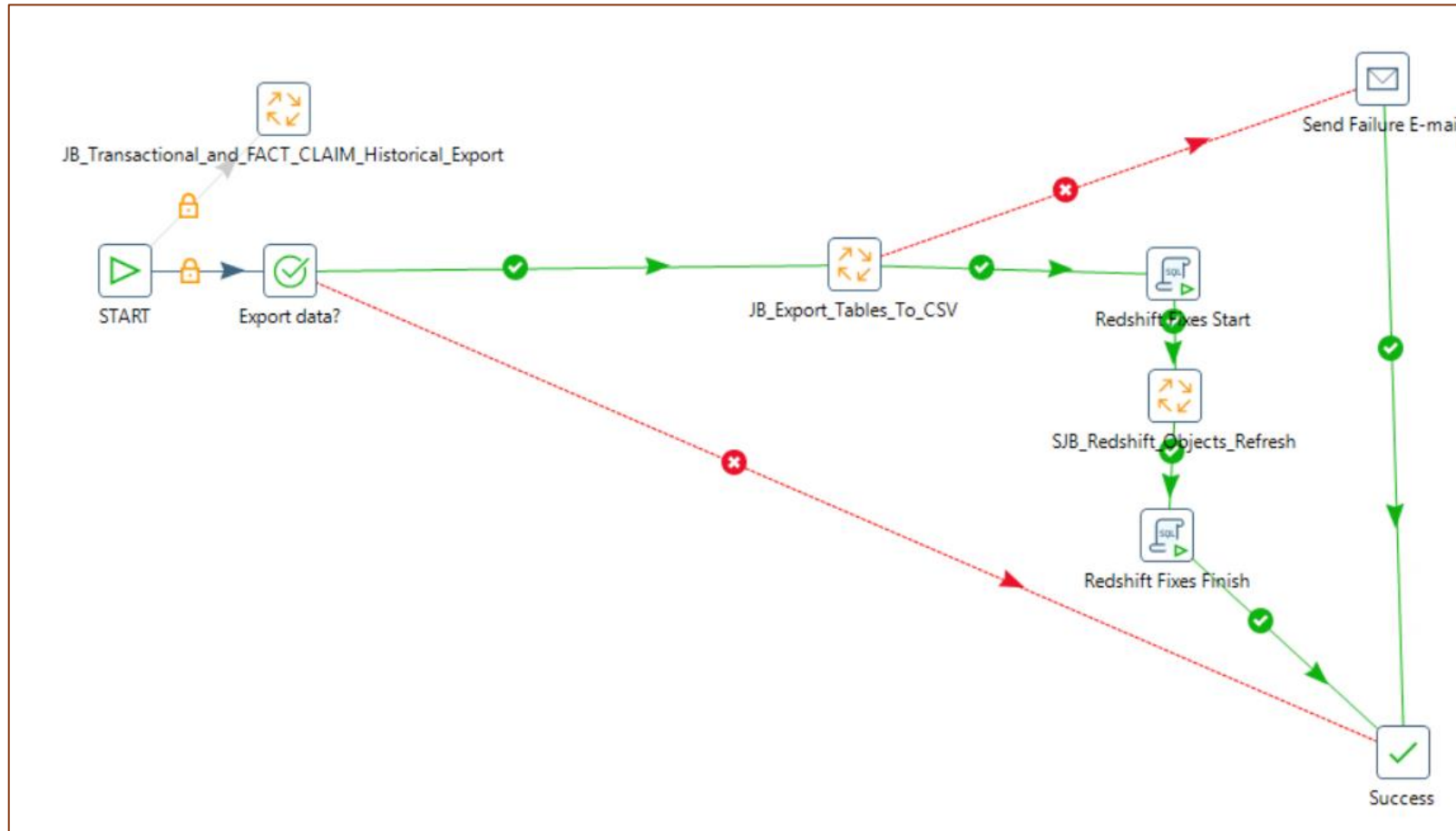
# Load Fact Transactions



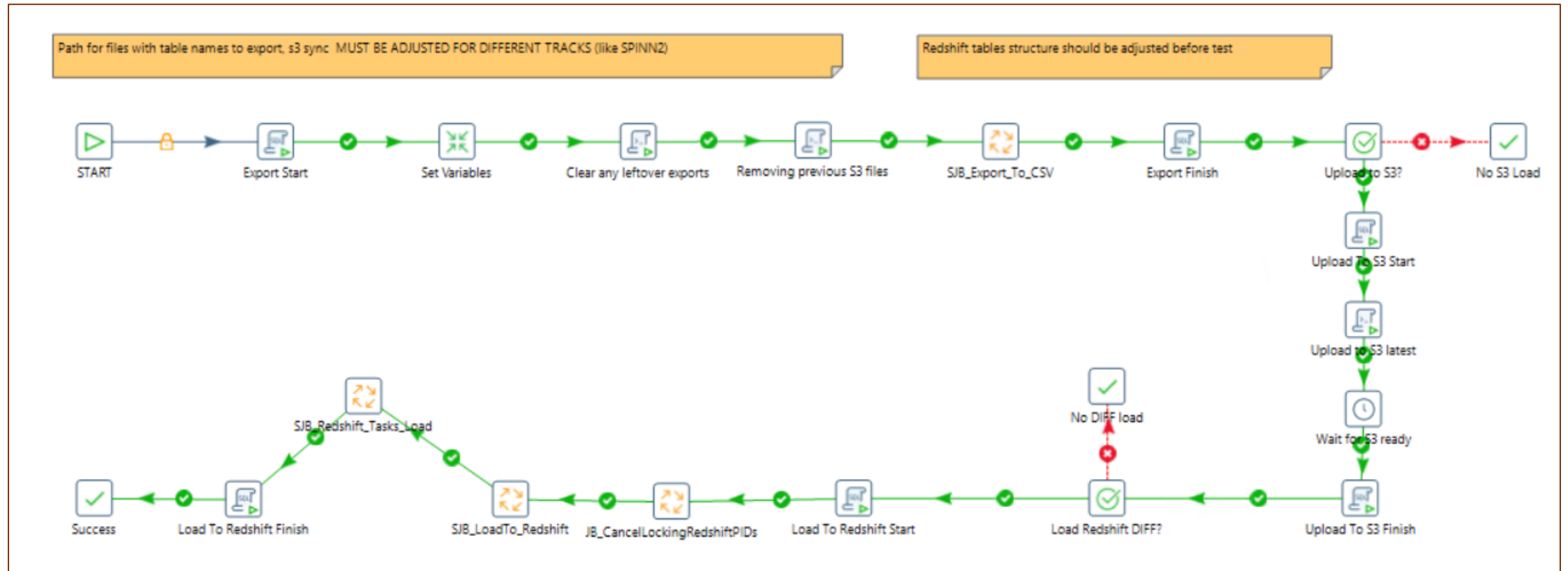
# Load Policy Fact Monthly Summaries



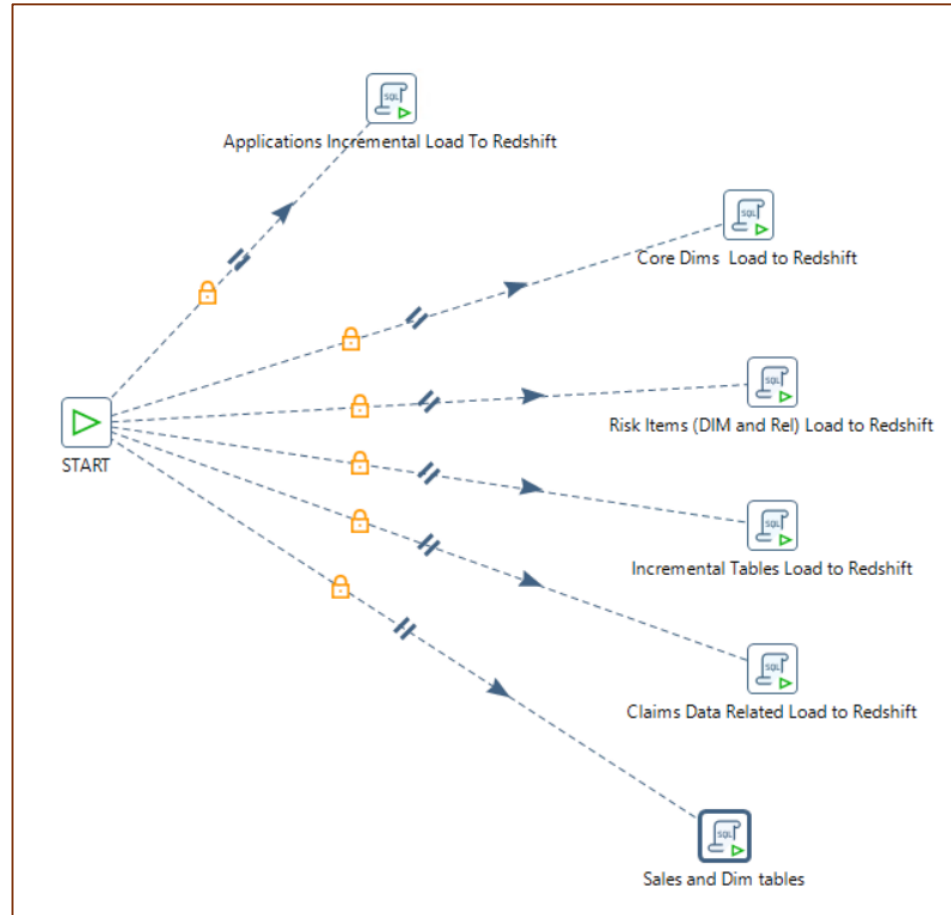
# Export to CSV -> AWS S3 -> AWS Redshift



# Export to CSV -> AWS S3 -> AWS Redshift



# Export to CSV -> AWS S3 -> AWS Redshift





# Start Tableau Dashboards Refresh



Shell

Job entry name: Refresh Tableau Dashboards

General Script

Insert script ☒

Script file name:  Browse...

Working directory: \${FSBI\_TEMP\_DIR}

Logging settings

Specify logfile? ☒

Append logfile? ☒

Name of logfile: \${FSBI\_LOG\_PATH}Log

Extension of logfile: txt

Include date in logfile? ☒

Include time in logfile? ☒

Loglevel: Basic

Copy previous results to args? ☐

Execute for every input row? ☐

Fields:

#	Argument

Help OK Cancel