Data Pipeline Architecture

Source Data Server

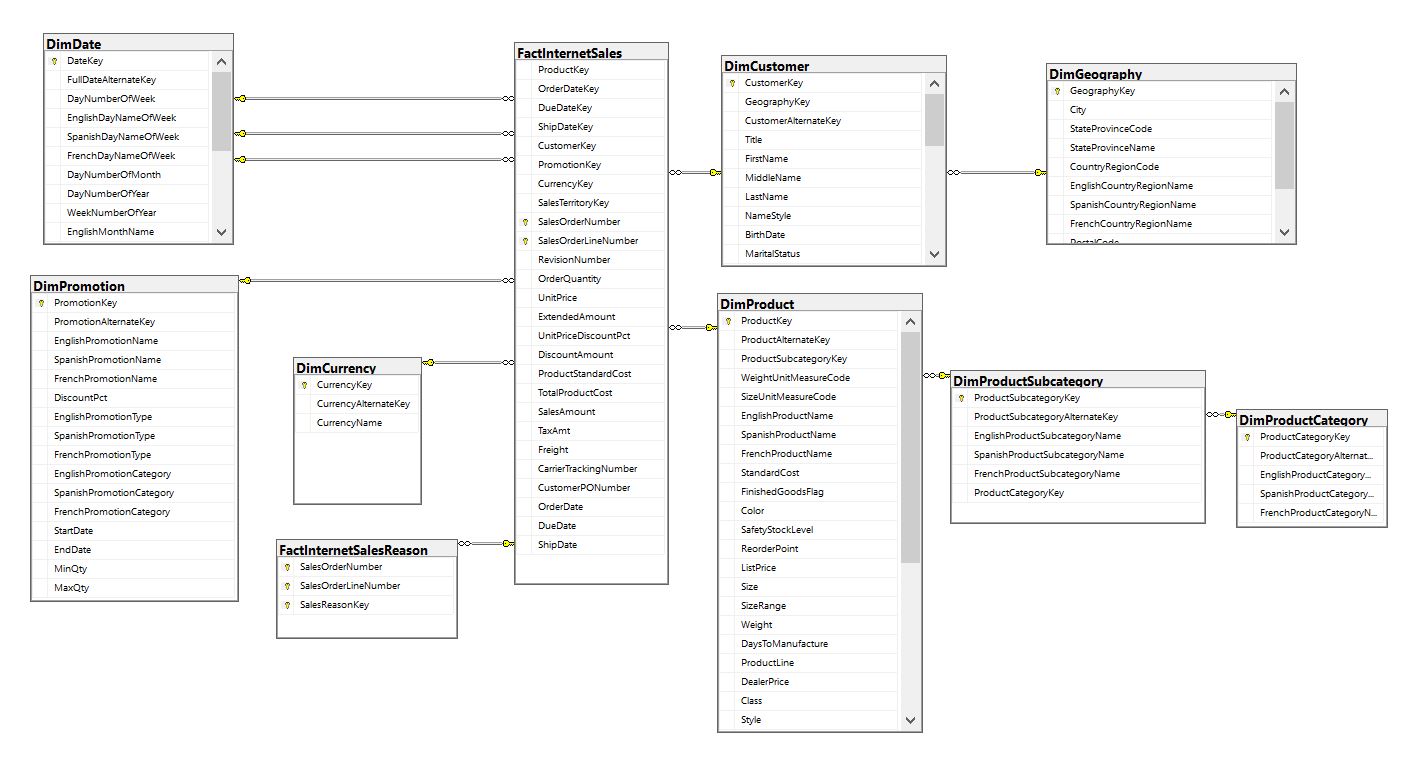
O que é Amazon QuickSight? - dataRain - Serviço base na nuvem

A picture containing text, clipart

Description automatically generated

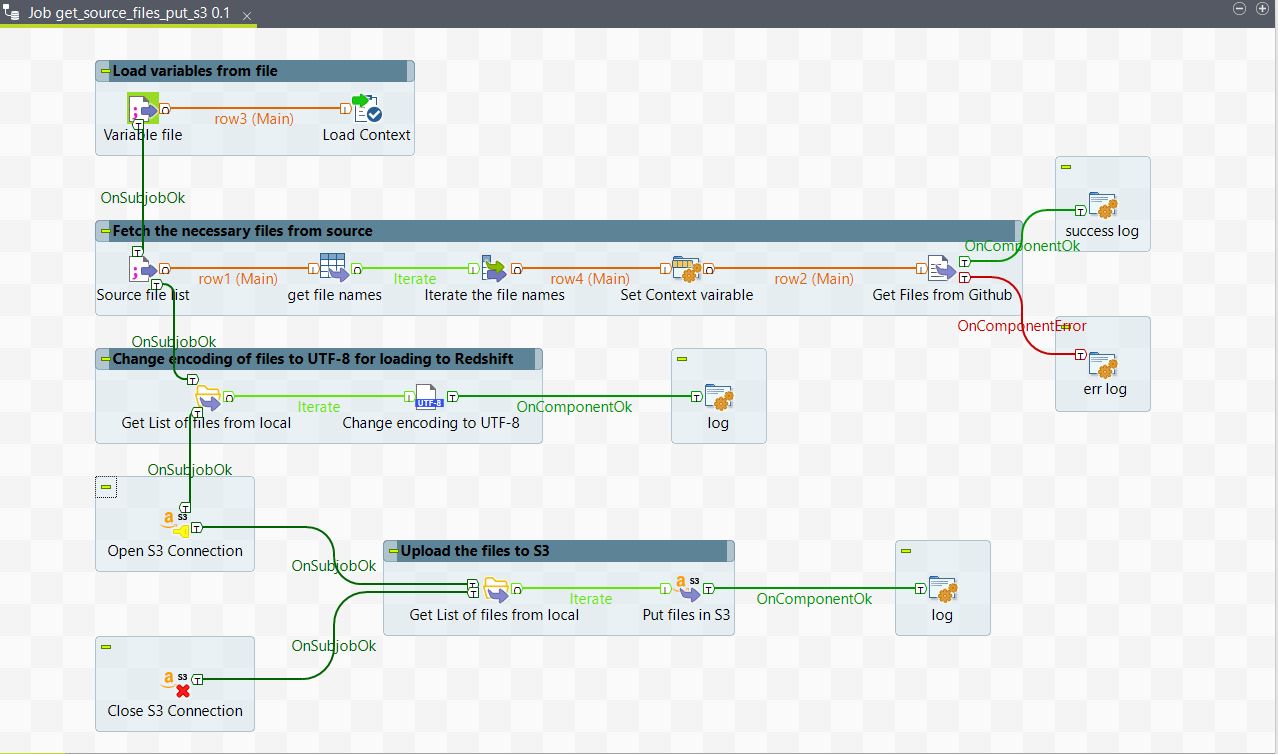
For the given task I have chosen the above architecture for the data pipeline.

I have taken the following tables from the AdventureWorks Datawarehouse



Below are the steps taken in details:

1. Extract data from Github(source) from a given list of files to the server(local)
2. Convert data to UTF8 encoding (needed to load data into Redshift)
3. Transfer the files to S3 for loading



The above image shows the first 3 steps described. All the file names, paths and configs are passed through the context variables.

1. Once the files are transferred to S3, we can then load the files to Redshift tables which are created in advance (refer create\_table\_scripts.sql)
   1. The tables are created with an addition column called load\_time as I have implemented the SCD Type 2 approach. The latest data will be taken with respect to the load\_time for future changes using the below queries:

drop table if exists staging\_" +{REDSHIFT\_TABLE\_NAME}+ ";

create table staging\_"+{REDSHIFT\_TABLE\_NAME}+" (like "+{REDSHIFT\_TABLE\_NAME}+");

alter table staging\_"+{REDSHIFT\_TABLE\_NAME}+" drop column load\_time;

copy staging\_"+{REDSHIFT\_TABLE\_NAME}+" from 's3://source-datasets/"+{REDSHIFT\_TABLE\_NAME}+".csv'

credentials 'aws\_access\_key\_id="+context.aws\_access\_key+";aws\_secret\_access\_key="+context.aws\_secret\_key+"'

delimiter '|' region 'eu-west-1';

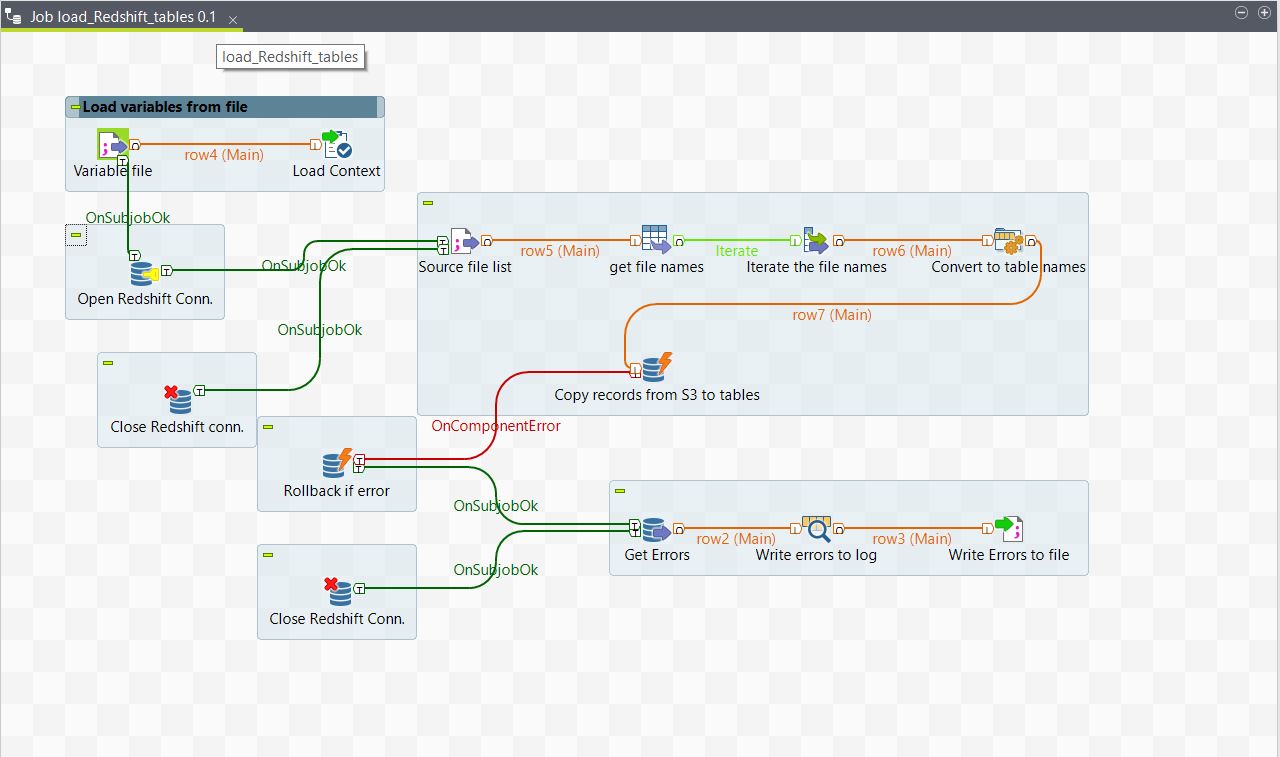
insert into "+{REDSHIFT\_TABLE\_NAME}+"

(select s.\*, sysdate as load\_time from staging\_"+{REDSHIFT\_TABLE\_NAME}+" s) ;

drop table staging\_"+{REDSHIFT\_TABLE\_NAME}+";

commit;

* 1. The data will first be loaded to the staging tables and then inserted into the main tables with the load\_time addition.



1. The errors during loading can be loaded to a different file from the temp table stl\_load\_errors using the below query:

select query, substring(filename,22,25) as filename,line\_number as line,

substring(colname,0,12) as column, type, position as pos, substring(raw\_line,0,30) as line\_text,

substring(raw\_field\_value,0,15) as field\_text,

substring(err\_reason,0,45) as reason

from stl\_load\_errors

order by query desc

1. Once the data is loaded, there is an option to transform the data into new tables for analysis by using materialized views on Redshift. Since I have taken the approach of using AWS Quicksight as my BI Tool we can do the transformations on Quicksight considering this is relatively small data.

A picture containing graphical user interface

Description automatically generated

Using different analysis and small transformations of the data, I have created the following dashboard views.