knab®

Financial Data Warehouse

- A case study

By : Kulamani Pradhan

Agenda

- About Me
- The Challenge
- High level Solution
- Solution Details
  - Assumptions
  - Staging Layer
  - Dimensional Model
  - Control Model
  - Job Process Model
  - DGS Report for DNB
- Demo
- Questions
- Future Improvements
- Code base links

About me

- Kulamani Pradhan
  - Native from India
  - Bachelor of Engineering in Information Technology
  - Professional experience > 7 years
  - BI Developer
  - Hobbies include playing cricket, cooking.

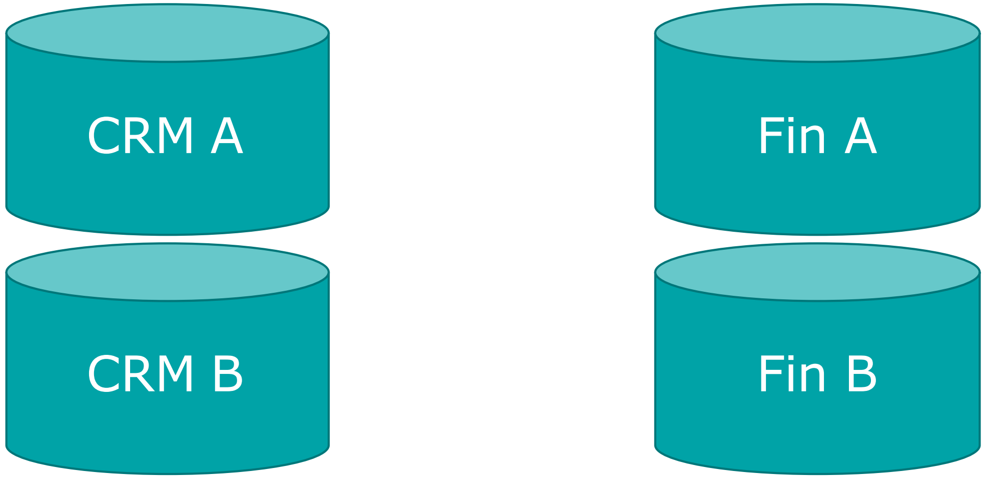
Case

Situation

Don is the COO at Knab. He is responsible for complying with DNB regulations, one of which is around the DGS (Deposito Garantie Stelsel). The DNB guarantees a maximum amount of € 100K per person in case the bank goes bankrupt.

In order for the DNB to make good on that promise, they want to know exactly how much money they would need to pay out. So Don wants to give them a list of all customers and the amount of money that is guaranteed by the DNB for those customers at the end of last year.

What makes it slightly more interesting, there are multiple source databases with information about customers (CRM) and about financial transactions (let’s call them Fin) and customers can appear in multiple systems at once.



CRM A

Fin A

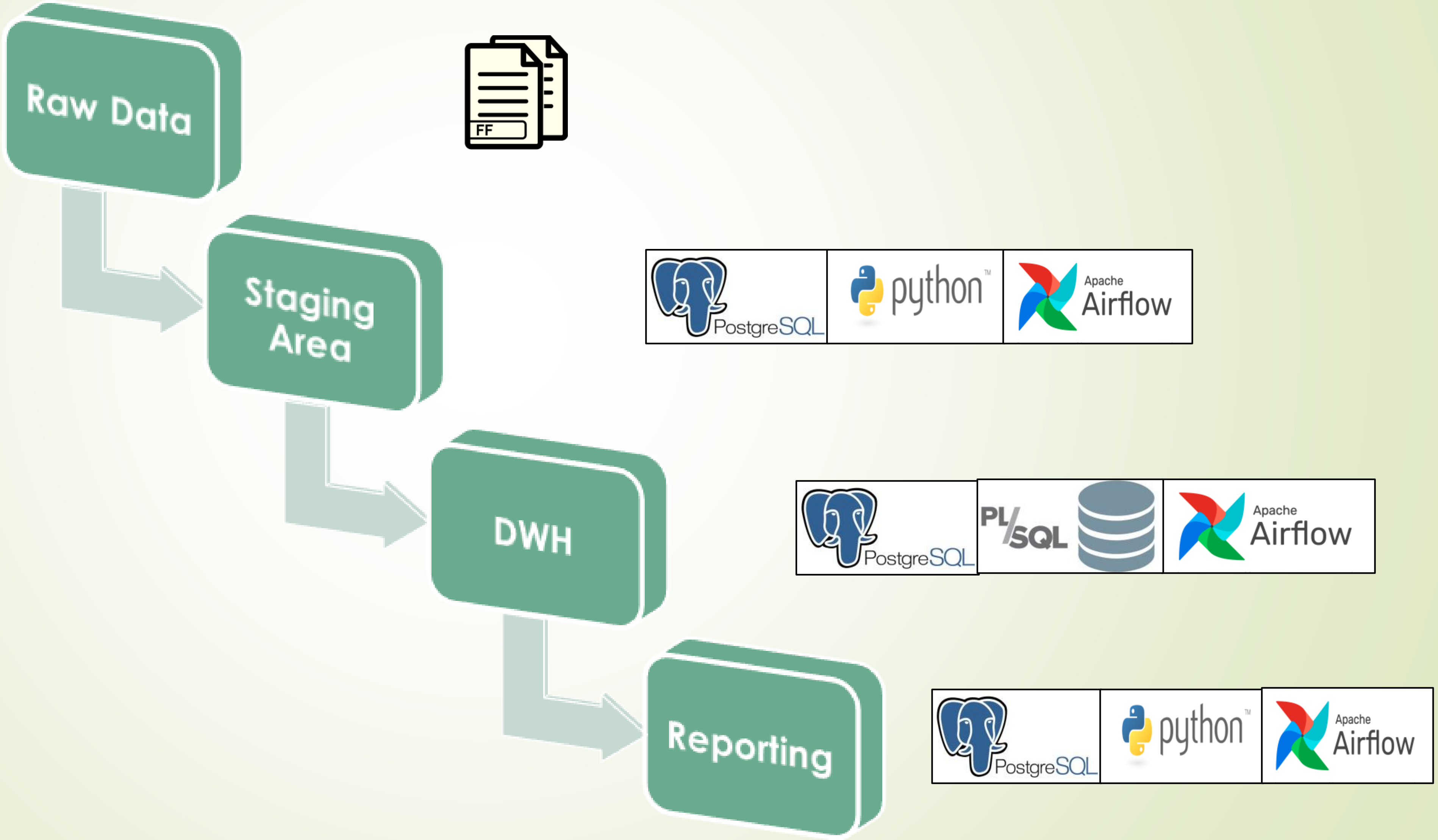
CRM B

Fin B

Challenges

- Describe which data you think should be used to answer Don’s question
- Draw a dimensional data model describing these data sets
- Describe which controls you would implement to ensure the final result is correct.
- Write a SQL query that produces the desired result

High Level Solution : Approach



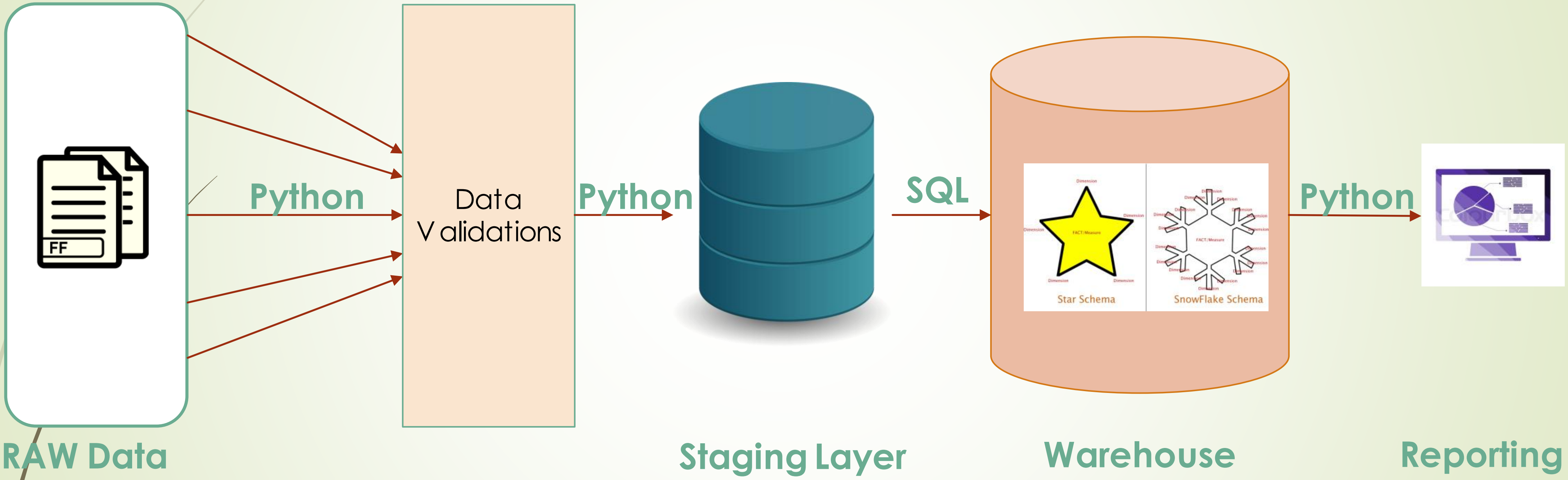
Raw Data

Staging Area

DWH

Reporting

High Level Solution : Approach



Python

Data Validations

Python

SQL

Python

RAW Data

Staging Layer

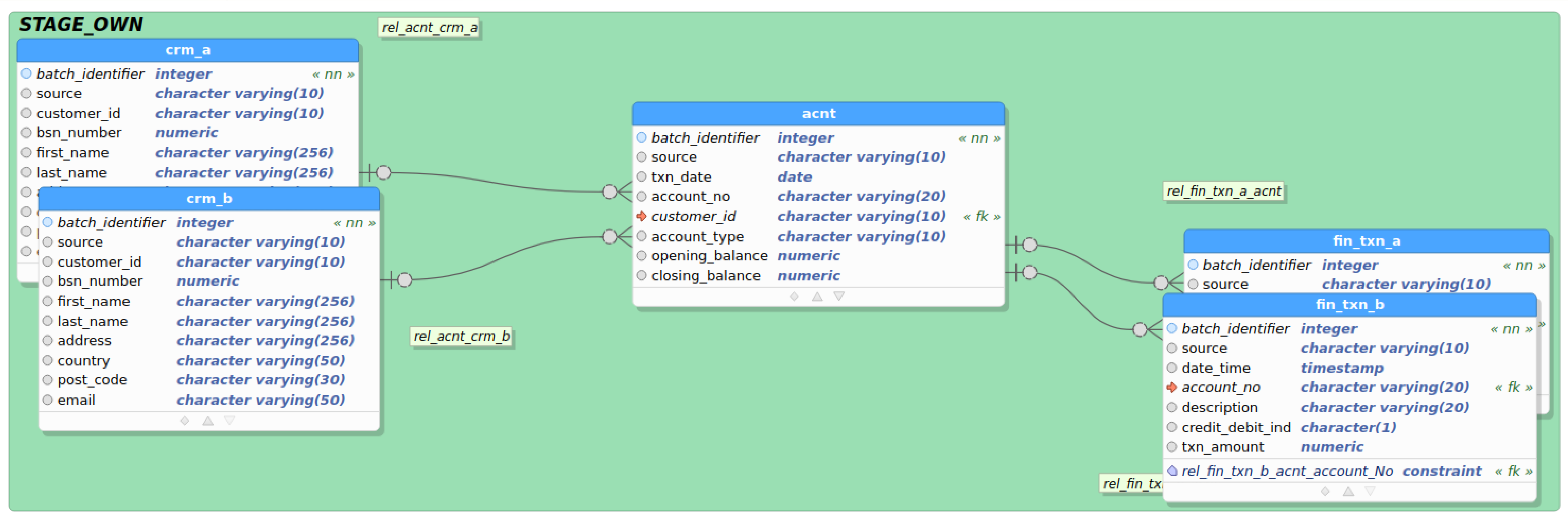
Warehouse

Reporting

Assumptions

- CRM system is an internal OLTP system of Knab. If we have external files then approach will change.
- A financial transaction is related to an account and that account must belong to a customer. A customer can have multiple accounts on their name.
- We will receive an acknowledgement file with filename & record count of each and every source files.
- We will receive an account file with opening balance and closing balance in it on daily basis.
- DNB is giving guarantee 100K per customer, doesn't matter how many accounts that customer has.
- All attributes within staging table.
- Flat file are in csv format and both crm_a & crm_b has similar structure. Same for fin_txn_a & fin_txn_b.

Solution Details : Staging Model



crm_a

○ batch_identifier	integer	« nn »
○ source	character varying(10)	
○ customer_id	character varying(10)	
○ bsn_number	numeric	
○ first_name	character varying(256)	
○ last_name	character varying(256)	

crm_b

○ batch_identifier	integer	« nn »
○ source	character varying(10)	
○ customer_id	character varying(10)	
○ bsn_number	numeric	
○ first_name	character varying(256)	
○ last_name	character varying(256)	
○ address	character varying(256)	
○ country	character varying(50)	
○ post_code	character varying(30)	
○ email	character varying(50)	

acnt

○ batch_identifier	integer	« nn »
○ source	character varying(10)	
○ txn_date	date	
○ account_no	character varying(20)	
◆ customer_id	character varying(10)	« fk »
○ account_type	character varying(10)	
○ opening_balance	numeric	
○ closing_balance	numeric	

fin_txn_a

○ batch_identifier	integer	« nn »
○ source	character varying(10)	

fin_txn_b

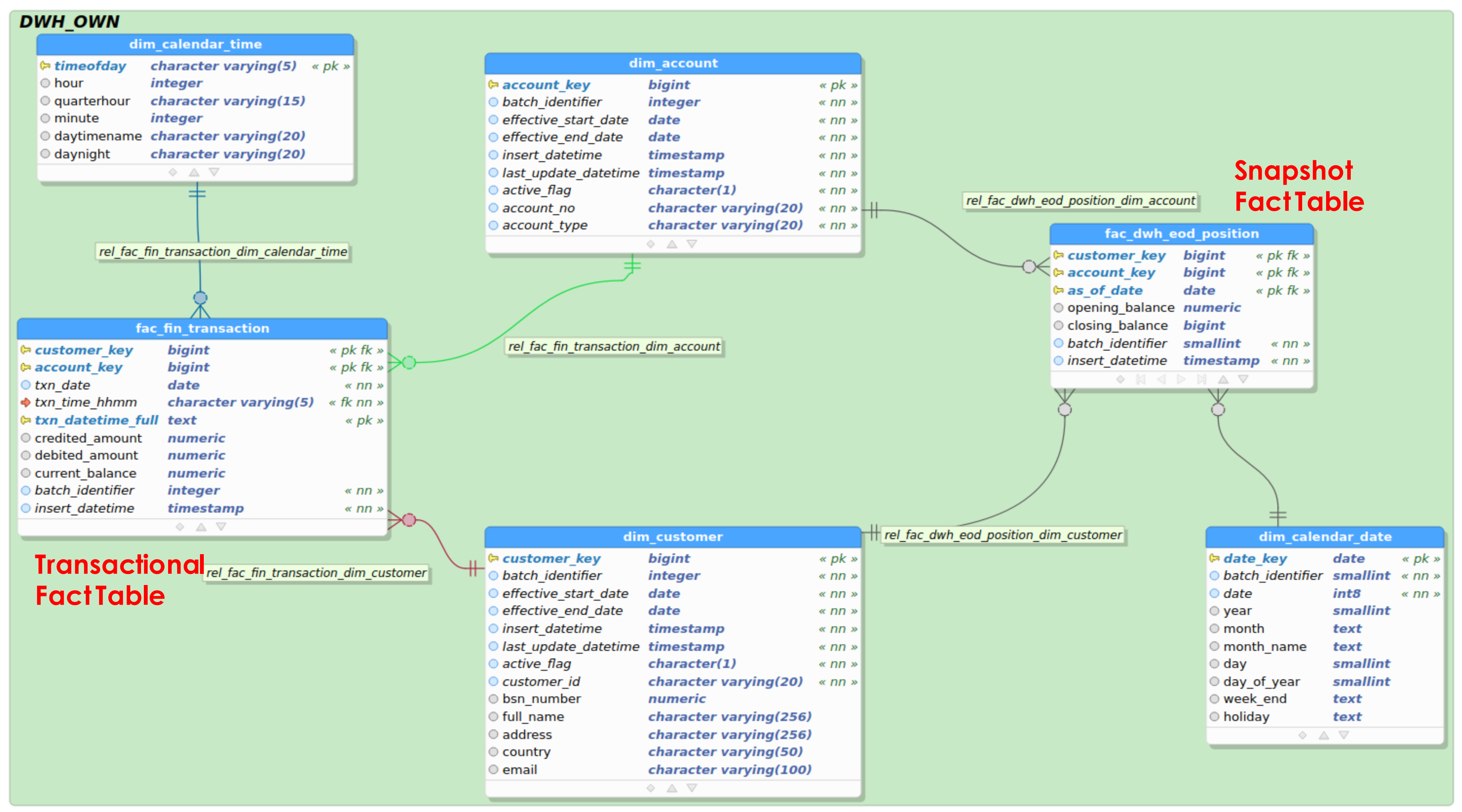
○ batch_identifier	integer	« nn »
○ source	character varying(10)	
○ date_time	timestamp	
◆ account_no	character varying(20)	« fk »
○ description	character varying(20)	
○ credit_debit_ind	character(1)	
○ txn_amount	numeric	
◇ rel_fin_txn_b_acnt_account_No	constraint	« fk »

Staging Loading Job :
https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/staging/stg_load.py

Solution Details : Data Validations

- Data Completeness Validation
  - knab_dwh_ack.txt file containing (file_name, record_count)
- Data Correctness Validation
  - *Mandatory Constraints (customer_id , account_no etc)*
  - Data-Type Constraints (Txn Amount must be Numeric)
  - Duplicate Record Constraints
  - *Attribute Value level* Constraints (customer_id must starts with "CA")
  - *Date/Time Format* Constraints (Ex : yyyy-mm-dd)
  - Referential Integrity Constraints
- Storing bad records in error_log table to monitor and discuss with source systems to fix if required.

Solution Details :Dimensional Model



DWH_OWN
rel_fac_fin_transaction_dim_calendar_time
rel_fac_fin_transaction_dim_account
rel_fac_dwh_eod_position_dim_account
rel_fac_dwh_eod_position_dim_customer
rel_fac_fin_transaction_dim_customer

dim_calendar_time

timeofday	character varying(5)	« pk »
hour	integer	
quarterhour	character varying(15)	
minute	integer	
daytimename	character varying(20)	
daynight	character varying(20)	

dim_account

account_key	bigint	« pk »
batch_identifier	integer	« nn »
effective_start_date	date	« nn »
effective_end_date	date	« nn »
insert_datetime	timestamp	« nn »
last_update_datetime	timestamp	« nn »
active_flag	character(1)	« nn »
account_no	character varying(20)	« nn »
account_type	character varying(20)	« nn »

Snapshot Fact Table

fac_dwh_eod_position

customer_key	bigint	« pk fk »
account_key	bigint	« pk fk »
as_of_date	date	« pk fk »
opening_balance	numeric	
closing_balance	bigint	
batch_identifier	smallint	« nn »
insert_datetime	timestamp	« nn »

fac_fin_transaction

customer_key	bigint	« pk fk »
account_key	bigint	« pk fk »
txn_date	date	« nn »
txn_time_hhmm	character varying(5)	« fk nn »
txn_datetime_full	text	« pk »
credited_amount	numeric	
debited_amount	numeric	
current_balance	numeric	
batch_identifier	integer	« nn »
insert_datetime	timestamp	« nn »

Transactional Fact Table

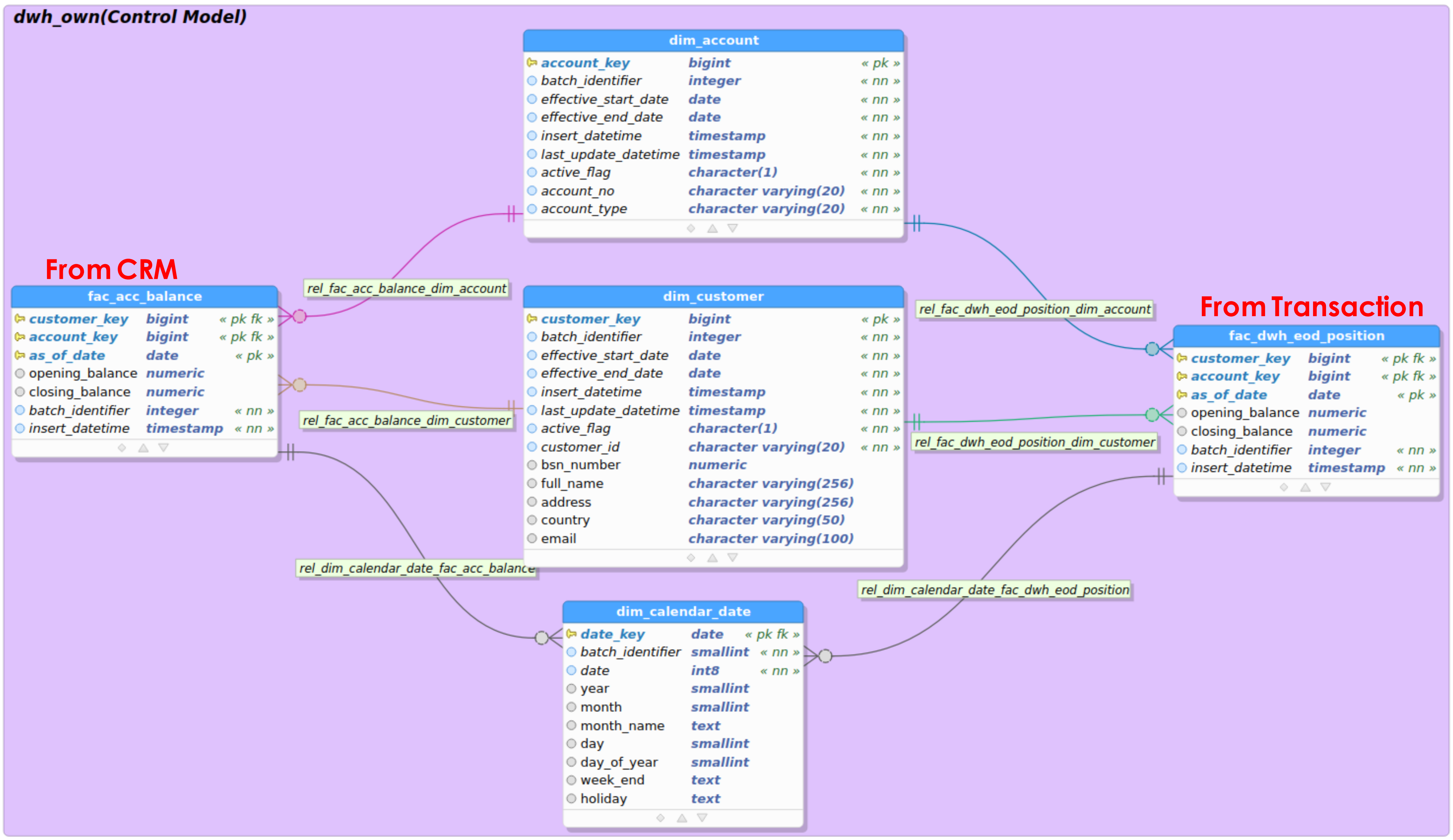
dim_customer

customer_key	bigint	« pk »
batch_identifier	integer	« nn »
effective_start_date	date	« nn »
effective_end_date	date	« nn »
insert_datetime	timestamp	« nn »
last_update_datetime	timestamp	« nn »
active_flag	character(1)	« nn »
customer_id	character varying(20)	« nn »
bsn_number	numeric	
full_name	character varying(256)	
address	character varying(256)	
country	character varying(50)	
email	character varying(100)	

dim_calendar_date

date_key	date	« pk »
batch_identifier	smallint	« nn »
date	int8	« nn »
year	smallint	
month	text	
month_name	text	
day	smallint	
day_of_year	smallint	
week_end	text	
holiday	text	

Solution Details :Control Model



dwh_own(Control Model)

From CRM

fac_acc_balance

customer_key	*bigint*	« pk fk »
account_key	*bigint*	« pk fk »
as_of_date	*date*	« pk »
opening_balance	*numeric*	
closing_balance	*numeric*	
batch_identifier	*integer*	« nn »
insert_datetime	*timestamp*	« nn »

dim_account

account_key	*bigint*	« pk »
batch_identifier	*integer*	« nn »
effective_start_date	*date*	« nn »
effective_end_date	*date*	« nn »
insert_datetime	*timestamp*	« nn »
last_update_datetime	*timestamp*	« nn »
active_flag	*character(1)*	« nn »
account_no	*character varying(20)*	« nn »
account_type	*character varying(20)*	« nn »

dim_customer

customer_key	*bigint*	« pk »
batch_identifier	*integer*	« nn »
effective_start_date	*date*	« nn »
effective_end_date	*date*	« nn »
insert_datetime	*timestamp*	« nn »
last_update_datetime	*timestamp*	« nn »
active_flag	*character(1)*	« nn »
customer_id	*character varying(20)*	« nn »
bsn_number	*numeric*	
full_name	*character varying(256)*	
address	*character varying(256)*	
country	*character varying(50)*	
email	*character varying(100)*	

From Transaction

fac_dwh_eod_position

customer_key	*bigint*	« pk fk »
account_key	*bigint*	« pk fk »
as_of_date	*date*	« pk »
opening_balance	*numeric*	
closing_balance	*numeric*	
batch_identifier	*integer*	« nn »
insert_datetime	*timestamp*	« nn »

dim_calendar_date

date_key	*date*	« pk fk »
batch_identifier	*smallint*	« nn »
date	*int8*	« nn »
year	*smallint*	
month	*smallint*	
month_name	*text*	
day	*smallint*	
day_of_year	*smallint*	
week_end	*text*	
holiday	*text*	

rel_fac_acc_balance_dim_account

rel_fac_acc_balance_dim_customer

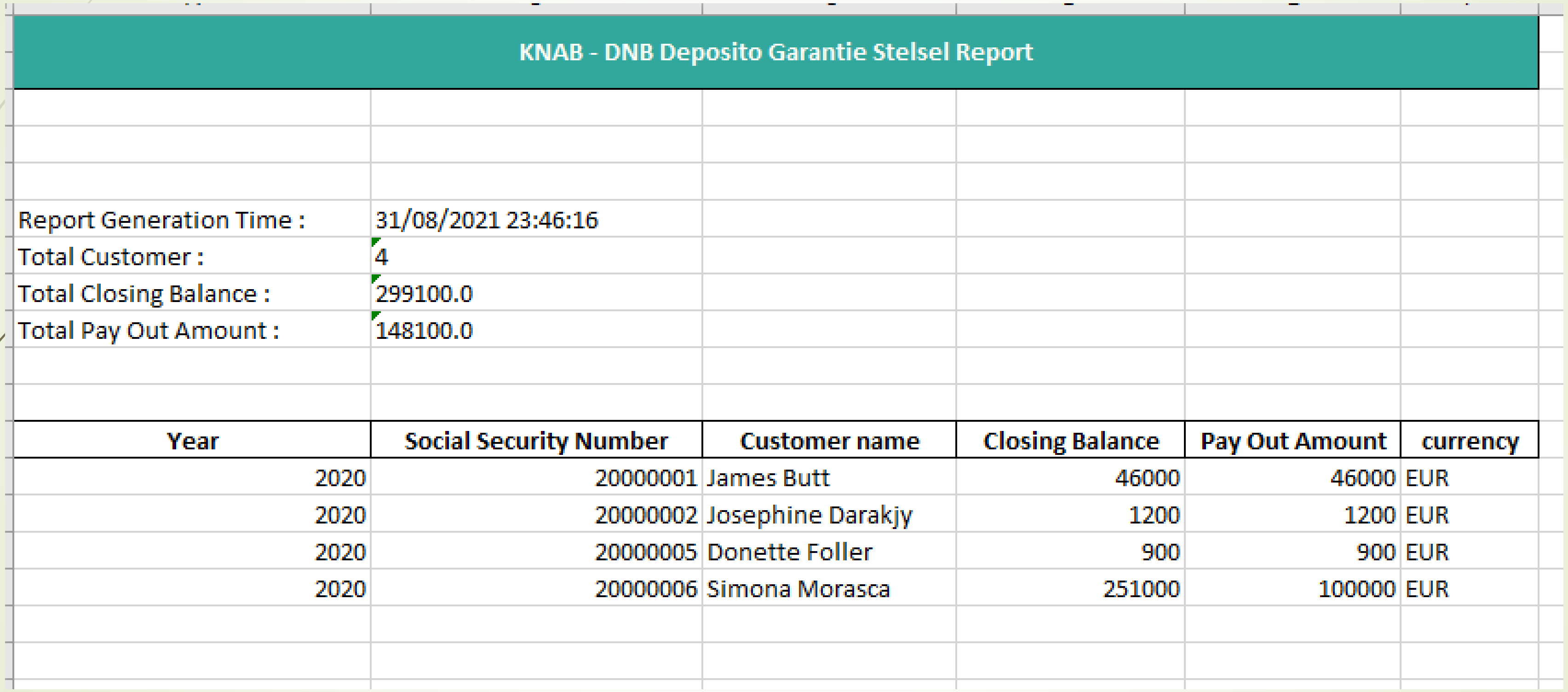
rel_dim_calendar_date_fac_acc_balance

rel_fac_dwh_eod_position_dim_account

rel_fac_dwh_eod_position_dim_customer

rel_dim_calendar_date_fac_dwh_eod_position

Solution Details : Report



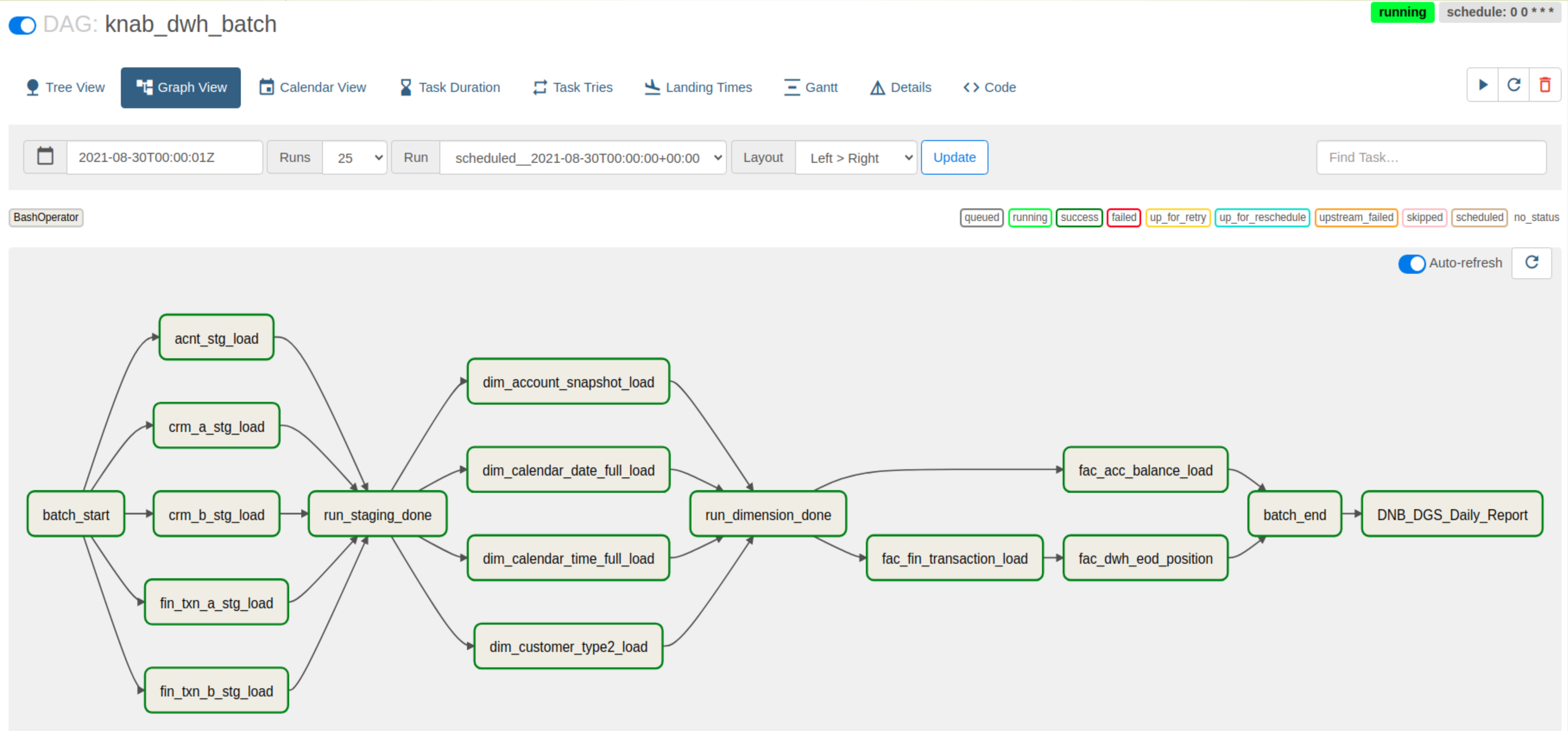
KNAB - DNB Deposito Garantie Stelsel Report

Report Generation Time :	31/08/2021 23:46:16
Total Customer :	4
Total Closing Balance :	299100.0
Total Pay Out Amount :	148100.0

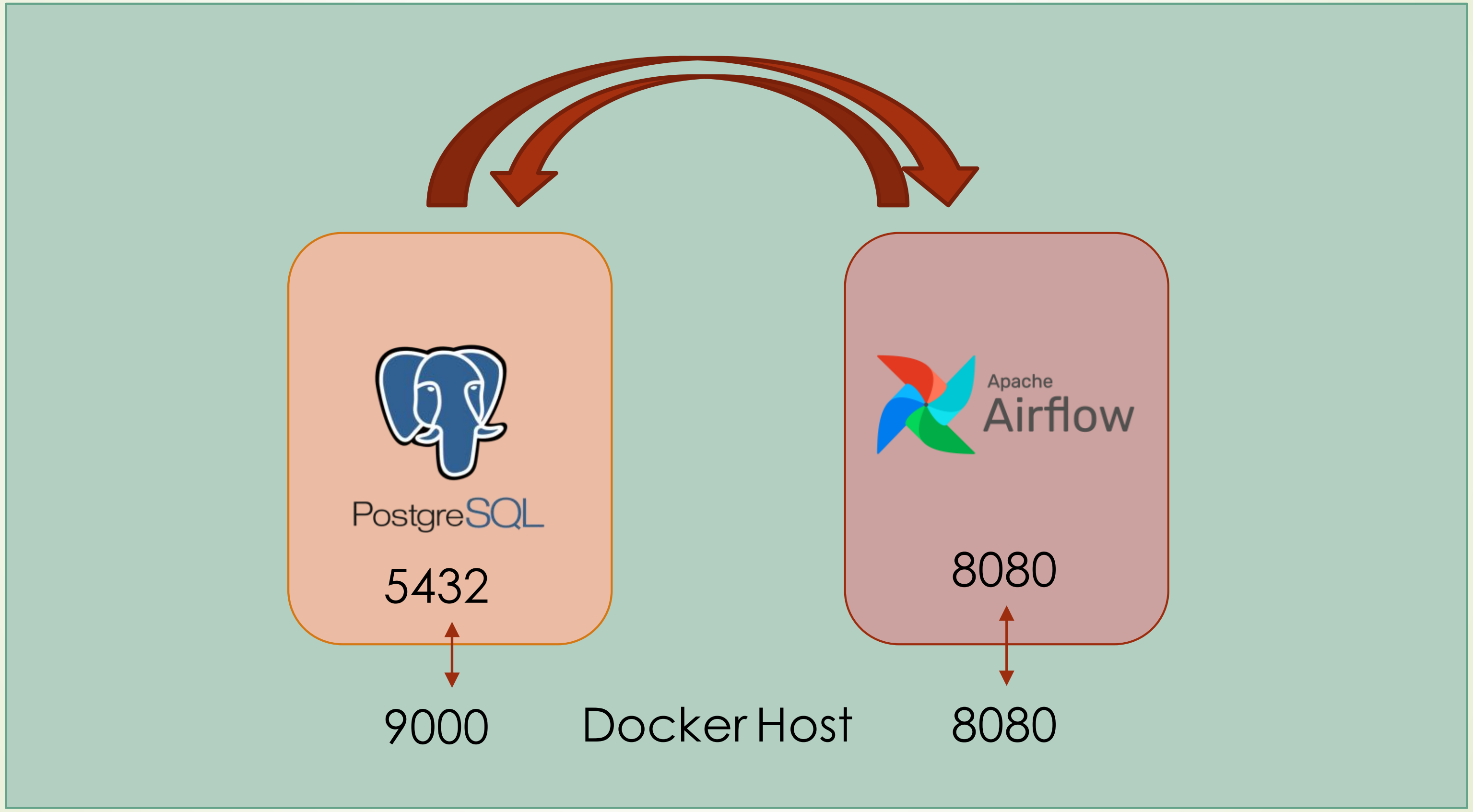
Year	Social Security Number	Customer name	Closing Balance	Pay Out Amount	currency
2020	20000001	James Butt	46000	46000	EUR
2020	20000002	Josephine Darakjy	1200	1200	EUR
2020	20000005	Donette Foller	900	900	EUR
2020	20000006	Simona Morasca	251000	100000	EUR

DNB_Deposito
iarantie Stelsel_202

Solution Details : Job Process Model



Containerization :

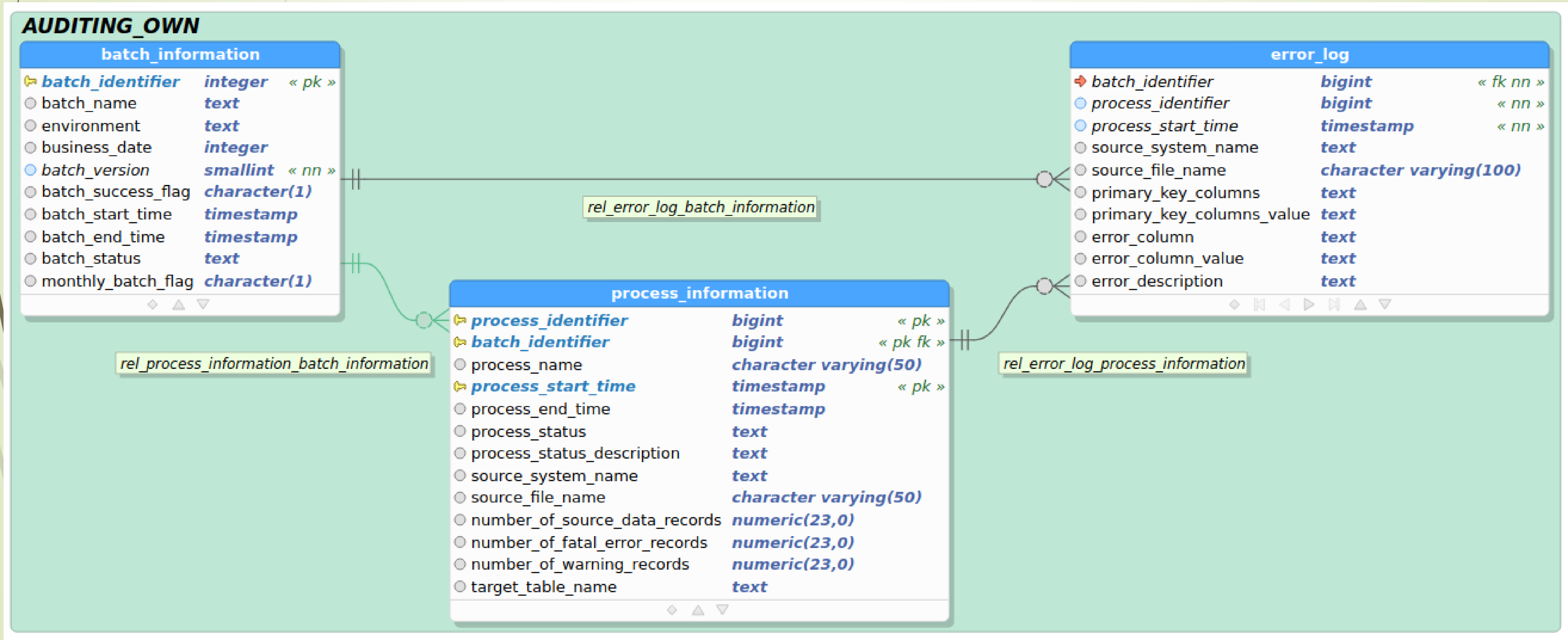


Container App	Docker File name	Image name(Docker hub)
PostgreSQL	Dockerfile_pg	mekpradhan/knab_dwh:knab_dwh_pg
Airflow	Dockerfile_airflow	mekpradhan/knab_dwh:Knab_dwh_airflow

Demo

- Done on a small set of Data (*in the interest of time*)
- Will be an E2E demo inclusive of:
  - SQL code walkthrough
  - Job execution using Airflow

Solution Details : Data Model(Auditing)



AUDITING_OWN

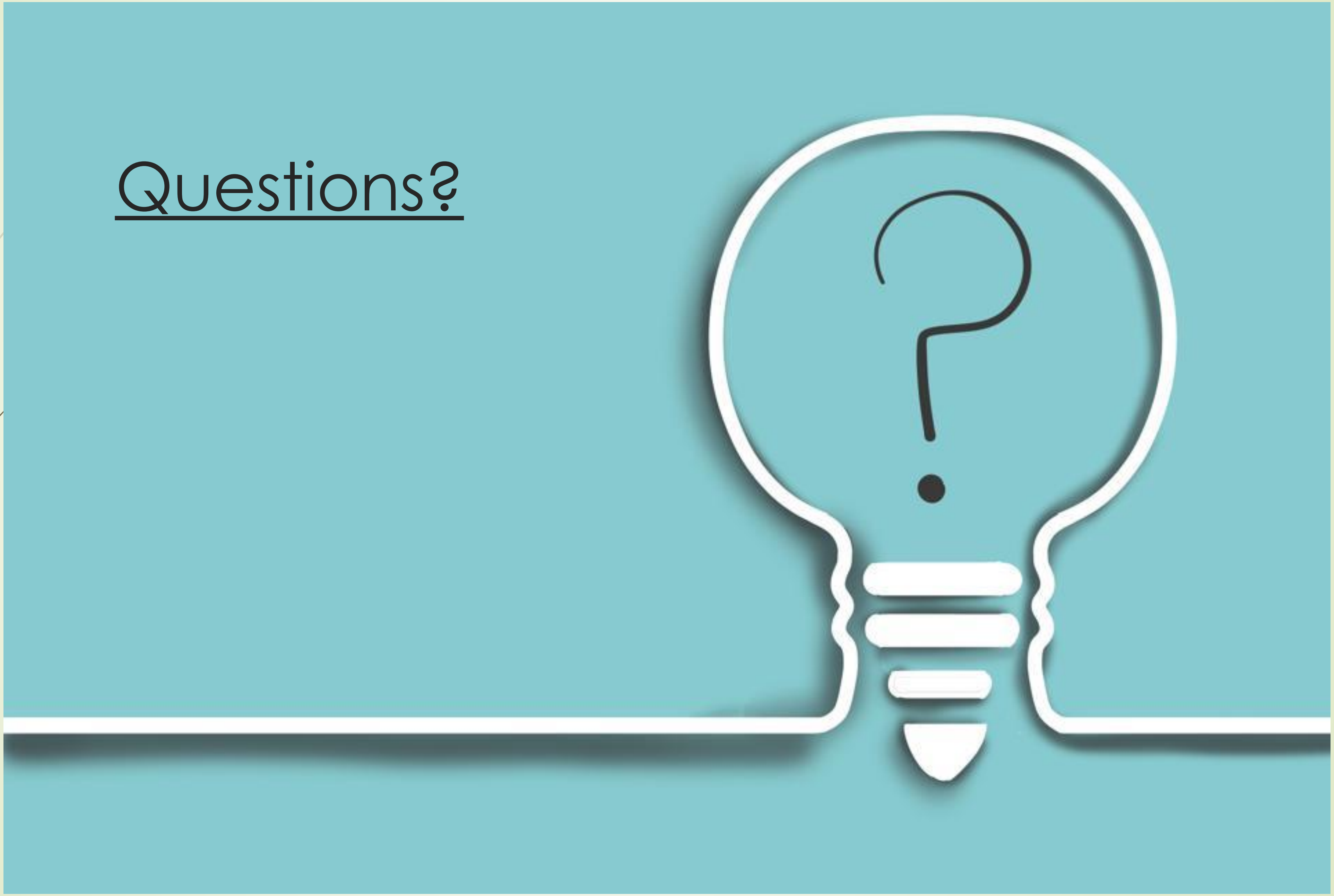
batch_information		
batch_identifier	integer	« pk »
batch_name	text	
environment	text	
business_date	integer	
batch_version	smallint	« nn »
batch_success_flag	character(1)	
batch_start_time	timestamp	
batch_end_time	timestamp	
batch_status	text	
monthly_batch_flag	character(1)	

error_log		
batch_identifier	bigint	« fk nn »
process_identifier	bigint	« nn »
process_start_time	timestamp	« nn »
source_system_name	text	
source_file_name	character varying(100)	
primary_key_columns	text	
primary_key_columns_value	text	
error_column	text	
error_column_value	text	
error_description	text	

process_information		
process_identifier	bigint	« pk »
batch_identifier	bigint	« pk fk »
process_name	character varying(50)	
process_start_time	timestamp	« pk »
process_end_time	timestamp	
process_status	text	
process_status_description	text	
source_system_name	text	
source_file_name	character varying(50)	
number_of_source_data_records	numeric(23,0)	
number_of_fatal_error_records	numeric(23,0)	
number_of_warning_records	numeric(23,0)	
target_table_name	text	

rel_error_log_batch_information
rel_process_information_batch_information
rel_error_log_process_information

Questions?



Future Improvements

- Event Based Trigger : (Automatic triggering of DWH batch)
  - We can create an airflow dag task to check every 2-5 seconds, if already all files have been delivered then start the batch.
- Batch Status Mail :
  - We can create a job which will send (Team) an email confirmation that batch has started, completed staging, completed fact and OLAP etc. In each stage we can get notified.
- Staging Area :
  - Parallel processing : If we use Amazon redshift(distributed network) to store whole DWH area, then these huge data can be processed parallel across multiple nodes.
- Batch Monitoring Dashboard :
  - Create a Batch monitoring dashboard using auditing_own schema tables, where we can easily monitor the batch status(if not airflow).
- Source File Movement :
  - Once Python processes the source file, the file should be moved to done location. In that way we will not be overriding the files of one date to another.
  - In case there is a failure we should move the file to working location. In that way it would be clear which one is still pending for processing.

Future Improvements

- Independent Execution of airflow containers :
  - In a production environment set up , we should separate airflow scheduler & webserver, so that they should run in two different containers.
  - Proper networking between containers.
  - Creating an docker-compose.yml file to start all these containers at once
- PDF Report to DNB

Code Base Links :

- Docker Images(Airflow + Postgres)
  - Dockerfiles : https://github.com/Kulamanipradhan0/knab_dwh
    - Dockerfile_pg
    - Dockerfile_airflow
  - Images : https://hub.docker.com/repository/docker/mekpradhan/knab_dwh
- DDL Code base : This is used to create all DB objects i.e. schema, users, tables, partitions etc
  - https://github.com/Kulamanipradhan0/knab_dwh/tree/main/db_scripts
- ETL Repository : All code base related to staging, dimension, fact & report loading
  - https://github.com/Kulamanipradhan0/knab_dwh/tree/main/etl_repo
- Sample Source Files : All sample flat files
  - https://github.com/Kulamanipradhan0/knab_dwh/tree/main/src_files/polling
- Airflow Dag file : Used for running the complete DWH batch
  - https://github.com/Kulamanipradhan0/knab_dwh/tree/main/airflow/dags

Code Base Links :

- Loading data from Transaction to EOD Position : (Making sure all active customers are getting loaded everyday)
  - https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/fact/fac_dwh_eod_position.sql
- Data Validation Layer(nice to have). Ex:
  - https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/staging/staging_data_validation.sql
- ETL Log files : During Job run ETL logs will be created in this directory.
  - https://github.com/Kulamanipradhan0/knab_dwh/tree/main/log_files
- SCD Type 2 Implementation using SQL : (dim_customer)
  - https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/dimension/dim_customer_type2_load.sql
- Report Code : (DNB_DGS_Last_Year_Report)
  - Main Job : https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/report/DNB_DGS_Last_Year_Report.py
  - SQL : https://github.com/Kulamanipradhan0/knab_dwh/blob/main/etl_repo/report/dnb_dgs_report_query.sql

Thank you