4FLOW
Your supply chain solution

Data Engineering for Data Solutions

Case study

Candidate version

Business problem

- Parcel service (delivery)
- Increasing number of complaints
- Root cause is unclear
- Tight monitoring to get hotspots

Deliverables

- Report 1:
  - What are the pickup stations that received complaints in the last 14 days?
    - Pickup Station ID
    - Pickup Station Name
    - Pickup Station Zip code
    - Number of claims
- Report 2:
  - What are the final recipients and the associated stations with more than 5 complaints in last 2 months?
    - Final recipient name
    - Final recipient address
    - Pickup Station IDs

Goals

- Design a data pipeline
  - Bring the delivery events data into the pipeline
  - Bring the complaints messages into the pipeline
  - Bring the pickup stations reference data into the pipeline
- Use a Cloud Service to aid in the process
- Build the reports

Available data

- Delivery events files (CSV, XML and JSON)
  - Delivery timestamp
  - Pickup station ID
  - Final recipient’s name and address
  - Driver ID
  - Parcel ID
  - Purchase Order ID
- Complaints (CSV, XML and JSON)
  - Timestamp of the complaint
  - Pickup station ID
  - Final recipient’s name and address
  - Parcel ID (optional)
  - Purchase Order ID (optional)
- List of Pickup stations (database)
  - Pickup station ID
  - Name
  - City
  - Zip code
  - Street

Assumptions

- Pickup stations in the reference table might be added, changed and removed
- Pickup stations are defined by their ID or their address
- Pickup station IDs in complaints data are congruent with the reference database
- Pickup station IDs in delivery events data are not consolidated with reference database
- Delivery events and complaints files fall into two different folders in the central server
- Pipeline will run once a day
- We are using Microsoft Azure as Cloud Service
  - Azure PostgreSQL as database management system
  - Azure Data Factory as orchestration tool and Azure Data Lake Storage Gen2 as data lake

Pipeline architecture

- Three layers for improved control and governance
  - Landing zone
  - Staging zone
  - Data Warehouse zone

- Orchestration will be made mostly by pipelines in Azure Data Factory

- Reports will be exposed as views

1) bring raw data into landing zone

- Azure Data Factory Pipelines
  - Daily triggered
  - Get metadata
  - ForEach file
  - Copy activity
  - SQL Script (INSERT)
- Copies raw data into Azure Data Lake Storage Gen2
- Updates *landing_file* control table

2) Parse raw CSV/XML/JSON into staging zone

- Azure Data Factory Pipeline
  - Trigger
  - LookUp “RECEIVED” in *landing_file*
  - If COUNT greater than zero
  - Calls Python parser function or container
- Python parser inserts data into suitable staging table (*stg_delivery_events* or *stg_complaints*)
- For each parsed file updates *landing_file* with “PARSED” or “FAILED”

3) copy reference data into staging zone

- Azure Data Factory Pipelines
  - Trigger
  - SQL Script (INSERT)

- Creates a fingerprint to be used in later JOINs

- Creates a HASH with name, city, zip code and street to easily test if anything has changed

4) update SCD2 *dim_pickup_station* in dw zone

- Azure Data Factory Pipeline
  - Trigger
  - SQL Script (ADD)
  - SQL Script (UPDATE CLOSE)
  - SQL Script (UPDATE INSERT)
  - SQL Script (REMOVE)

- Update the SCD2 by adding new entries, closing and inserting entries that changed and deactivating removed entries.

- Creates a canonical ID.

5) Update fact tables in dw zone

- Azure Data Factory Pipeline
  - Trigger
  - SQL Script (INSERT)

- Inserts data from *staging.stg_delivery_events* into *dw.fact_delivery_events*

- Inserts data from *staging.stg_complaints* into *dw.fact_complaints* joining with *dw.dim_pickup_*station via Pickup Station ID

6) monitoring data integration

- Azure Data Factory Pipeline
  - Trigger
  - SQL Script (INSERT)

- Inserts into *dw.monitoring_complaints* the number os complaints processed in the current day and the number of complaints that didn’t get a pickup ID match with *dw.dim_pickup_station*.

7) reports

- Once they are views, they are instantly updated
- Report 1 is exposed in *dw.* rpt_pickup_station_claims_14d
- Report 2 is exposed in *dw.* rpt_recipients_over5_complaints_2m

Next steps

- Improve monitoring stages in pipeline
- Correlate complaints and delivery events using *parcel_id* or *purchase_order_id* whenever possible
- Create a crosswalk table allowing a more precise connection between delivery events and complaints
- Draw insights from drivers, recipients and stations in terms of number of deliveries and number of complaints
- Use LLM to evaluate the claims contents