**Read carefully before you begin**

* For question 1 and 2: you only need to answer them if you have not provided answers in your application screening questions.
* For questions 3 to 8:
  + Pick 4 questions that you are most comfortable with to answer. And state your assumptions if anything is not clear.
  + If you have used any large language models (e.g. ChatGPT) to assist with your answers, declare it with a note at the end of the answer, along with the LLM model and all the prompts you used to get to the final answer.

**Question 1:** Have you written any Python code in your personal GitHub repository that you can share with a link?

**Answer:** <https://github.com/guangliangyang/pingpong-foul>

**Question 2:** How many years of commercial experience do you have in Snowflake and Python programming respectively?

**Answer:** Snowflake <1 month, Python > 5 Years

**Question 3**

Design and create a logical data model using the star schema methodology to store the electricity outage dataset below in Snowflake. Define primary/foreign/surrogate keys for each table and the data types for each column. Note that the number of customers on a transformer may change over time, and the model and table structure must support preserving these changes during the ETL process.

| **Outage ID** | **Outage Suburb** | **Transformer Code** | **# of Customers on Transformer** | **Outage Start Time** | **Outage End Time** | **Outage Status** | **Outage Duration in Minutes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 12345 | Ponsonby | KCN ME01 | 1,200 | 25/06/2024 8:00 | 25/06/2024 9:00 | Closed | 60 |
| 12346 | Albany | KNN CEP1 | 500 | 25/07/2024 8:30 | 25/07/2024 10:30 | Closed | 120 |
| 12347 | Remuera | REMU MK01 | 30 | 27/08/2024 8:30 | 27/08/2024 10:30 | Closed | 120 |
| 12347 | Remuera | REMU MK09 | 2,000 | 27/08/2024 8:30 | 27/08/2024 9:00 | Closed | 30 |
| 12347 | Remuera | REMU MU78 | 100 | 27/08/2024 8:30 | 27/08/2024 8:50 | Closed | 20 |
| 13349 | Ponsonby | KCN ME01 | 13 | 31/08/2024 20:00 |  | Open | 300 |
| 13350 | Takapuna | KNN CXP8 | 150 | 31/08/2024 22:10 |  | Open | 430 |

**Answer:**

1. **Assumptions**

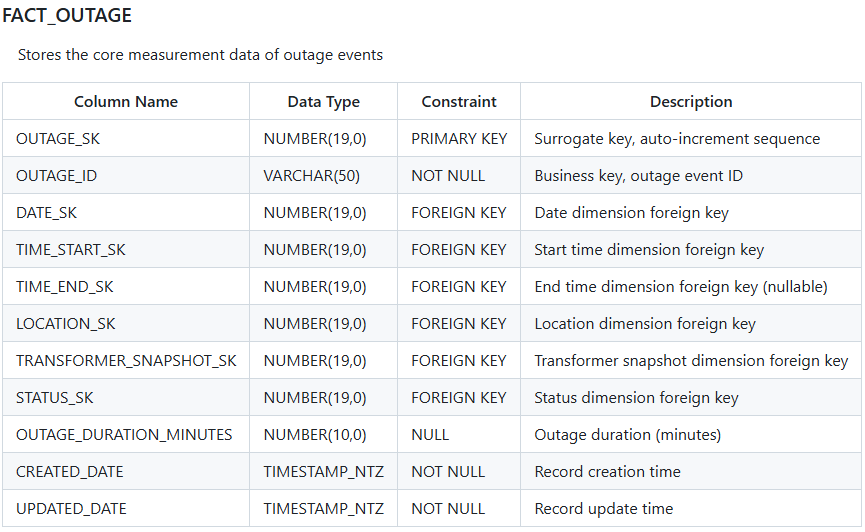
* The model must support 3+ years of historical outage analysis with geographic (region/city/suburb) and temporal (year-to-hour) aggregations while tracking quarterly transformer customer count changes.
* Data characteristics include 50-500 daily outages across 5K-50K transformers, with strict quality rules for outage IDs, transformer codes, and sub-2-hour timestamp updates.
* Performance targets require 95% of queries under 5 seconds for 50+ concurrent users, accommodating ≤50% annual data growth over 3 years.
* Transformer Code is a stable business identifier for transformers, The start time of power outages is always accurately recorded.

1. **LLM**

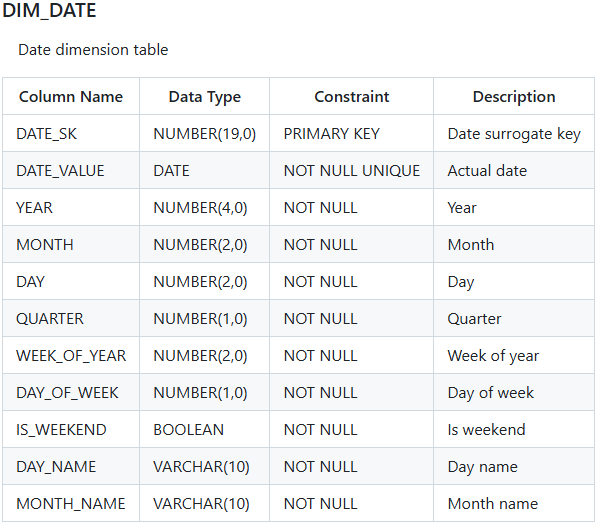
Claude Sonnet 4

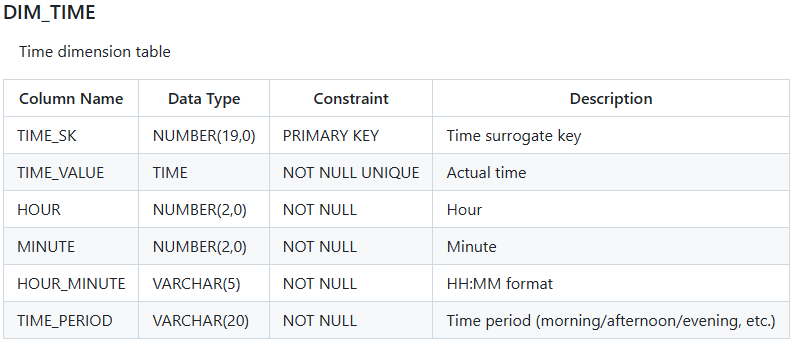
Prompt: question and assumptions

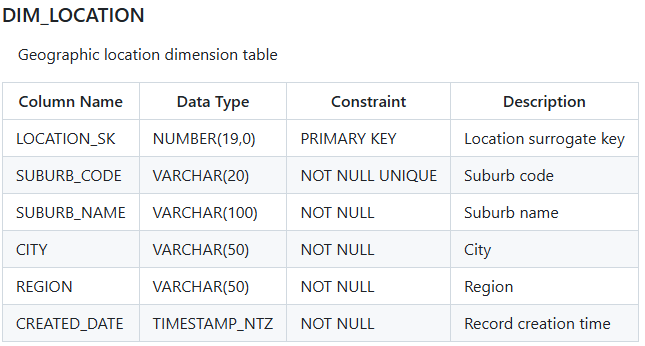
1. **Design** 
   1. **Fact Table**

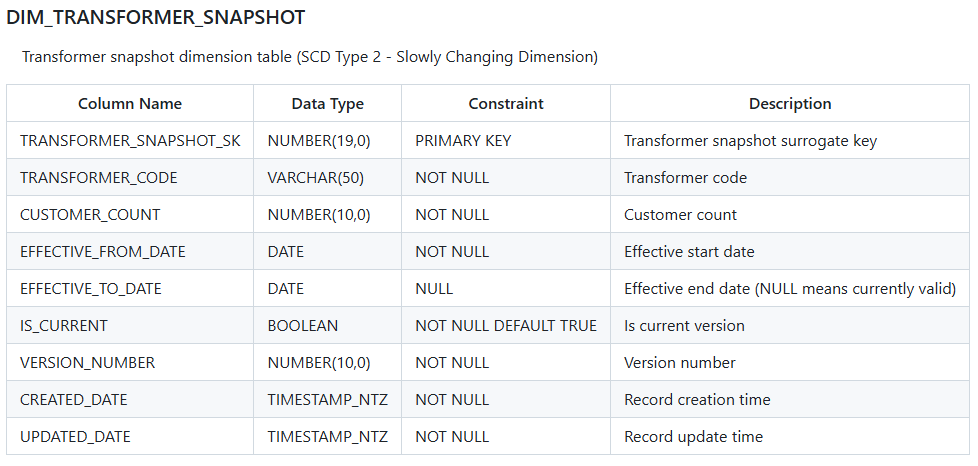
****

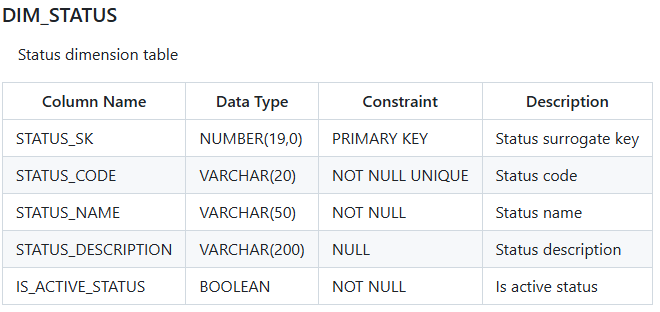
* 1. **Dimension Table**

****

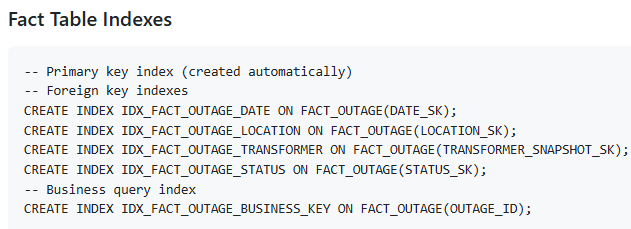
****

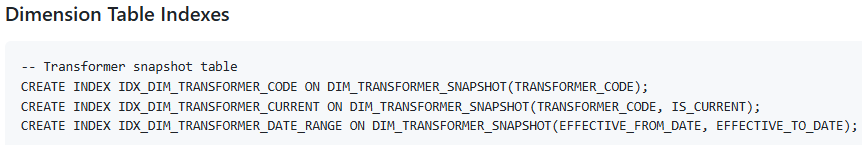
****

****

****

* 1. **Index Design**

****

****

**Question 4**

On 1 Sep 2024, the daily ETL receives the following outage record from the Outage Management System (OMS):

| **Outage ID** | **Outage Suburb** | **Transformer Code** | **# of Customers on Transformer** | **Outage Start Time** | **Outage End Time** | **Outage Status** | **Outage Duration in Minutes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 14508 | Milford | KNN CXP9 | 1,000 | 01/09/2024 8:00 |  | Open | 60 |

However, the TRANS system—your source for transformer attributes (size, install date, etc.)—is offline and is not expected back until 6 Sep 2024. Describe an ETL strategy that:

* preserves referential integrity between the fact and transformer dimension tables,
* allows outage reporting to continue between 1 Sep and 6 Sep, and
* back-fills the correct transformer attributes once TRANS is online.

**Answer:**

1. **Assumptions**
   * Sufficient ETL window, there is enough time for backfill processing.
2. **LLM**

Claude Sonnet 4

Prompt: question and “Does the Snowflake system have a mature solution for this situation?”

1. **Solution**

**## 2. Data Model Design**

**### 2.1 Extend Transformer Dimension Table**

```sql

-- Extend the existing dimension table to support data source tracking

ALTER TABLE DIM\_TRANSFORMER\_SNAPSHOT ADD COLUMN (

DATA\_SOURCE VARCHAR(50) DEFAULT 'TRANS',

IS\_PLACEHOLDER BOOLEAN DEFAULT FALSE,

NEEDS\_BACKFILL BOOLEAN DEFAULT FALSE,

TRANSFORMER\_SIZE\_KVA NUMBER(10,2) NULL,

INSTALL\_DATE DATE NULL,

MANUFACTURER VARCHAR(100) NULL,

MODEL VARCHAR(100) NULL,

VOLTAGE\_RATING VARCHAR(20) NULL

);

-- Create index to optimize query performance

CREATE INDEX IDX\_DIM\_TRANSFORMER\_SOURCE

ON DIM\_TRANSFORMER\_SNAPSHOT(DATA\_SOURCE, IS\_PLACEHOLDER);

```

**### 2.2 Staging Table Design**

```sql

-- Raw data staging table

CREATE TABLE STG\_PENDING\_OUTAGES (

STAGING\_ID NUMBER(19,0) IDENTITY PRIMARY KEY,

OUTAGE\_ID VARCHAR(50) NOT NULL,

OUTAGE\_SUBURB VARCHAR(100),

TRANSFORMER\_CODE VARCHAR(50),

CUSTOMER\_COUNT NUMBER(10,0),

OUTAGE\_START\_TIME TIMESTAMP\_NTZ,

OUTAGE\_END\_TIME TIMESTAMP\_NTZ,

OUTAGE\_STATUS VARCHAR(20),

OUTAGE\_DURATION\_MINUTES NUMBER(10,0),

RECEIVED\_DATE TIMESTAMP\_NTZ DEFAULT CURRENT\_TIMESTAMP,

PROCESSED\_FLAG BOOLEAN DEFAULT FALSE,

BACKFILL\_REQUIRED BOOLEAN DEFAULT FALSE

) DATA\_RETENTION\_TIME\_IN\_DAYS = 7;

-- TRANS system attribute staging table

CREATE TABLE STG\_TRANSFORMER\_ATTRIBUTES (

TRANSFORMER\_CODE VARCHAR(50) PRIMARY KEY,

TRANSFORMER\_SIZE\_KVA NUMBER(10,2),

INSTALL\_DATE DATE,

MANUFACTURER VARCHAR(100),

MODEL VARCHAR(100),

VOLTAGE\_RATING VARCHAR(20),

LOCATION\_DETAILS VARCHAR(200),

ACTUAL\_CUSTOMER\_COUNT NUMBER(10,0),

LAST\_MAINTENANCE\_DATE DATE,

LOADED\_DATE TIMESTAMP\_NTZ DEFAULT CURRENT\_TIMESTAMP

) DATA\_RETENTION\_TIME\_IN\_DAYS = 30;

```

**## 3. Phase 1: Temporary Processing Strategy (1-6 Sep)**

**### 3.1 Data Protection and Backup**

```sql

-- Create a backup at the time point before the system goes offline

CREATE TABLE DIM\_TRANSFORMER\_SNAPSHOT\_BACKUP

CLONE DIM\_TRANSFORMER\_SNAPSHOT

AT(TIMESTAMP => '2024-08-31 23:59:59');

-- Set appropriate retention period

ALTER TABLE DIM\_TRANSFORMER\_SNAPSHOT

SET DATA\_RETENTION\_TIME\_IN\_DAYS = 7;

```

**### 3.2 Temporary Data Processing Workflow**

```sql

-- Stored procedure to check if transformer exists

CREATE OR REPLACE PROCEDURE check\_and\_create\_transformer(

transformer\_code VARCHAR(50),

customer\_count NUMBER(10,0),

effective\_date DATE

)

RETURNS NUMBER

LANGUAGE SQL

AS

$$

DECLARE

transformer\_exists NUMBER DEFAULT 0;

new\_transformer\_sk NUMBER;

BEGIN

-- Check if transformer already exists

SELECT COUNT(\*) INTO transformer\_exists

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE TRANSFORMER\_CODE = transformer\_code

AND IS\_CURRENT = TRUE;

IF (transformer\_exists = 0) THEN

-- Create placeholder record

INSERT INTO DIM\_TRANSFORMER\_SNAPSHOT (

TRANSFORMER\_CODE,

CUSTOMER\_COUNT,

EFFECTIVE\_FROM\_DATE,

EFFECTIVE\_TO\_DATE,

IS\_CURRENT,

VERSION\_NUMBER,

IS\_PLACEHOLDER,

DATA\_SOURCE,

NEEDS\_BACKFILL,

CREATED\_DATE,

UPDATED\_DATE

) VALUES (

transformer\_code,

customer\_count,

effective\_date,

NULL,

TRUE,

1,

TRUE,

'OMS\_TEMP',

TRUE,

CURRENT\_TIMESTAMP,

CURRENT\_TIMESTAMP

);

SELECT TRANSFORMER\_SNAPSHOT\_SK INTO new\_transformer\_sk

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE TRANSFORMER\_CODE = transformer\_code

AND IS\_CURRENT = TRUE;

RETURN new\_transformer\_sk;

ELSE

-- Return existing record SK

SELECT TRANSFORMER\_SNAPSHOT\_SK INTO new\_transformer\_sk

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE TRANSFORMER\_CODE = transformer\_code

AND IS\_CURRENT = TRUE;

RETURN new\_transformer\_sk;

END IF;

END;

$$;

```

**### 3.3 Process Outage Records**

```sql

-- Main ETL process for new outage records

CREATE OR REPLACE PROCEDURE process\_outage\_record()

RETURNS STRING

LANGUAGE SQL

AS

$$

DECLARE

transformer\_sk NUMBER;

BEGIN

-- Stage raw data

INSERT INTO STG\_PENDING\_OUTAGES (

OUTAGE\_ID, OUTAGE\_SUBURB, TRANSFORMER\_CODE, CUSTOMER\_COUNT,

OUTAGE\_START\_TIME, OUTAGE\_END\_TIME, OUTAGE\_STATUS,

OUTAGE\_DURATION\_MINUTES, BACKFILL\_REQUIRED

) VALUES (

'14508', 'Milford', 'KNN CXP9', 1000,

'2024-09-01 08:00:00', NULL, 'Open', 60, TRUE

);

-- Check and create transformer record

CALL check\_and\_create\_transformer('KNN CXP9', 1000, '2024-09-01')

INTO transformer\_sk;

-- Process fact table record

INSERT INTO FACT\_OUTAGE (

OUTAGE\_ID, DATE\_SK, TIME\_START\_SK, TIME\_END\_SK,

LOCATION\_SK, TRANSFORMER\_SNAPSHOT\_SK, STATUS\_SK,

OUTAGE\_DURATION\_MINUTES, CREATED\_DATE, UPDATED\_DATE

)

SELECT

'14508',

d.DATE\_SK,

ts.TIME\_SK,

NULL,

l.LOCATION\_SK,

transformer\_sk,

st.STATUS\_SK,

60,

CURRENT\_TIMESTAMP,

CURRENT\_TIMESTAMP

FROM DIM\_DATE d

CROSS JOIN DIM\_TIME ts

CROSS JOIN DIM\_LOCATION l

CROSS JOIN DIM\_STATUS st

WHERE d.DATE\_VALUE = '2024-09-01'

AND ts.TIME\_VALUE = '08:00:00'

AND l.SUBURB\_NAME = 'Milford'

AND st.STATUS\_CODE = 'Open';

RETURN 'Outage record processed successfully with temporary transformer data';

END;

$$;

```

**## 4. Monitoring Using Native Snowflake Features**

**### 4.1 Create Change Tracking Streams**

```sql

-- Create change tracking stream for transformer dimension

CREATE STREAM dim\_transformer\_changes\_stream

ON TABLE DIM\_TRANSFORMER\_SNAPSHOT

COMMENT = 'Track changes to transformer dimension for TRANS backfill';

-- Create change tracking stream for fact table (for audit)

CREATE STREAM fact\_outage\_changes\_stream

ON TABLE FACT\_OUTAGE

COMMENT = 'Track outage fact table changes during TRANS outage';

```

**### 4.2 Dynamic Table for Automated Monitoring**

```sql

-- Create monitoring view

CREATE OR REPLACE DYNAMIC TABLE transformer\_status\_monitor

TARGET\_LAG = '5 minutes'

WAREHOUSE = ETL\_SMALL\_WH

AS

SELECT

COUNT(\*) as TOTAL\_TRANSFORMERS,

COUNT(CASE WHEN IS\_PLACEHOLDER = TRUE THEN 1 END) as PLACEHOLDER\_COUNT,

COUNT(CASE WHEN NEEDS\_BACKFILL = TRUE THEN 1 END) as PENDING\_BACKFILL,

MIN(CASE WHEN IS\_PLACEHOLDER = TRUE THEN CREATED\_DATE END) as OLDEST\_PLACEHOLDER,

CURRENT\_TIMESTAMP as LAST\_UPDATED

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE IS\_CURRENT = TRUE;

```

**## 5. Phase 2: Automated Backfill Strategy (After 6 Sep)**

**### 5.1 TRANS System Recovery Detection**

```sql

-- Create system availability detection task

CREATE OR REPLACE TASK check\_trans\_availability

WAREHOUSE = ETL\_SMALL\_WH

SCHEDULE = '15 MINUTE'

AS

DECLARE

trans\_available BOOLEAN DEFAULT FALSE;

pending\_count NUMBER DEFAULT 0;

BEGIN

-- Simulate TRANS system health check

-- In actual implementation, this would check TRANS system health

-- Check for new transformer attribute data

SELECT COUNT(\*) > 0 INTO trans\_available

FROM STG\_TRANSFORMER\_ATTRIBUTES

WHERE LOADED\_DATE >= DATEADD(hour, -1, CURRENT\_TIMESTAMP);

-- Check for pending backfill records

SELECT COUNT(\*) INTO pending\_count

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE NEEDS\_BACKFILL = TRUE;

IF (trans\_available AND pending\_count > 0) THEN

-- Trigger backfill process

CALL execute\_backfill\_process();

END IF;

END;

-- Start the task

ALTER TASK check\_trans\_availability RESUME;

```

**### 5.2 Intelligent Backfill Processing**

```sql

CREATE OR REPLACE PROCEDURE execute\_backfill\_process()

RETURNS STRING

LANGUAGE SQL

AS

$$

DECLARE

processed\_count NUMBER DEFAULT 0;

error\_count NUMBER DEFAULT 0;

result\_message STRING;

BEGIN

-- Handle cases where customer count matches

UPDATE DIM\_TRANSFORMER\_SNAPSHOT ts

SET

IS\_PLACEHOLDER = FALSE,

DATA\_SOURCE = 'TRANS',

NEEDS\_BACKFILL = FALSE,

TRANSFORMER\_SIZE\_KVA = ta.TRANSFORMER\_SIZE\_KVA,

INSTALL\_DATE = ta.INSTALL\_DATE,

MANUFACTURER = ta.MANUFACTURER,

MODEL = ta.MODEL,

VOLTAGE\_RATING = ta.VOLTAGE\_RATING,

UPDATED\_DATE = CURRENT\_TIMESTAMP

FROM STG\_TRANSFORMER\_ATTRIBUTES ta

WHERE ts.TRANSFORMER\_CODE = ta.TRANSFORMER\_CODE

AND ts.IS\_PLACEHOLDER = TRUE

AND ts.CUSTOMER\_COUNT = ta.ACTUAL\_CUSTOMER\_COUNT;

processed\_count := SQLROWCOUNT;

-- Handle cases where customer count does not match (SCD Type 2)

-- 1. Close old record

UPDATE DIM\_TRANSFORMER\_SNAPSHOT ts

SET

EFFECTIVE\_TO\_DATE = CURRENT\_DATE - 1,

IS\_CURRENT = FALSE,

UPDATED\_DATE = CURRENT\_TIMESTAMP

FROM STG\_TRANSFORMER\_ATTRIBUTES ta

WHERE ts.TRANSFORMER\_CODE = ta.TRANSFORMER\_CODE

AND ts.IS\_PLACEHOLDER = TRUE

AND ts.CUSTOMER\_COUNT != ta.ACTUAL\_CUSTOMER\_COUNT;

-- 2. Create new accurate record

INSERT INTO DIM\_TRANSFORMER\_SNAPSHOT (

TRANSFORMER\_CODE, CUSTOMER\_COUNT, EFFECTIVE\_FROM\_DATE,

EFFECTIVE\_TO\_DATE, IS\_CURRENT, VERSION\_NUMBER,

IS\_PLACEHOLDER, DATA\_SOURCE, NEEDS\_BACKFILL,

TRANSFORMER\_SIZE\_KVA, INSTALL\_DATE, MANUFACTURER,

MODEL, VOLTAGE\_RATING, CREATED\_DATE, UPDATED\_DATE

)

SELECT

ta.TRANSFORMER\_CODE,

ta.ACTUAL\_CUSTOMER\_COUNT,

CURRENT\_DATE,

NULL,

TRUE,

ts.VERSION\_NUMBER + 1,

FALSE,

'TRANS',

FALSE,

ta.TRANSFORMER\_SIZE\_KVA,

ta.INSTALL\_DATE,

ta.MANUFACTURER,

ta.MODEL,

ta.VOLTAGE\_RATING,

CURRENT\_TIMESTAMP,

CURRENT\_TIMESTAMP

FROM STG\_TRANSFORMER\_ATTRIBUTES ta

JOIN DIM\_TRANSFORMER\_SNAPSHOT ts

ON ta.TRANSFORMER\_CODE = ts.TRANSFORMER\_CODE

WHERE ts.IS\_PLACEHOLDER = TRUE

AND ts.CUSTOMER\_COUNT != ta.ACTUAL\_CUSTOMER\_COUNT;

-- Update staging table processing status

UPDATE STG\_PENDING\_OUTAGES

SET PROCESSED\_FLAG = TRUE,

BACKFILL\_REQUIRED = FALSE

WHERE TRANSFORMER\_CODE IN (

SELECT TRANSFORMER\_CODE

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE DATA\_SOURCE = 'TRANS'

);

result\_message := 'Backfill completed. Processed: ' || processed\_count || ' records';

-- Log processing result

INSERT INTO ETL\_PROCESS\_LOG (

PROCESS\_NAME, STATUS, MESSAGE, PROCESSED\_COUNT,

ERROR\_COUNT, CREATED\_DATE

) VALUES (

'TRANS\_BACKFILL', 'SUCCESS', result\_message,

processed\_count, error\_count, CURRENT\_TIMESTAMP

);

RETURN result\_message;

EXCEPTION

WHEN OTHER THEN

INSERT INTO ETL\_PROCESS\_LOG (

PROCESS\_NAME, STATUS, MESSAGE, ERROR\_DETAILS, CREATED\_DATE

) VALUES (

'TRANS\_BACKFILL', 'ERROR', 'Backfill process failed',

SQLERRM, CURRENT\_TIMESTAMP

);

RETURN 'Backfill failed: ' || SQLERRM;

END;

$$;

```

**## 6. Business Continuity Assurance**

**### 6.1 Reporting View**

```sql

-- Provide a transparent reporting view for business users

CREATE OR REPLACE SECURE VIEW V\_OUTAGE\_REPORTING AS

SELECT

f.OUTAGE\_ID,

l.SUBURB\_NAME,

ts.TRANSFORMER\_CODE,

ts.CUSTOMER\_COUNT,

CASE

WHEN ts.IS\_PLACEHOLDER = TRUE

THEN ts.CUSTOMER\_COUNT || ' (Temporary Estimate)'

ELSE ts.CUSTOMER\_COUNT || ''

END as CUSTOMER\_COUNT\_DISPLAY,

d.DATE\_VALUE as OUTAGE\_DATE,

t1.TIME\_VALUE as START\_TIME,

t2.TIME\_VALUE as END\_TIME,

s.STATUS\_NAME,

f.OUTAGE\_DURATION\_MINUTES,

ts.DATA\_SOURCE,

CASE

WHEN ts.IS\_PLACEHOLDER = TRUE THEN 'Y'

ELSE 'N'

END as IS\_ESTIMATED\_DATA,

ts.TRANSFORMER\_SIZE\_KVA,

ts.MANUFACTURER,

ts.INSTALL\_DATE

FROM FACT\_OUTAGE f

JOIN DIM\_DATE d ON f.DATE\_SK = d.DATE\_SK

JOIN DIM\_TIME t1 ON f.TIME\_START\_SK = t1.TIME\_SK

LEFT JOIN DIM\_TIME t2 ON f.TIME\_END\_SK = t2.TIME\_SK

JOIN DIM\_LOCATION l ON f.LOCATION\_SK = l.LOCATION\_SK

JOIN DIM\_TRANSFORMER\_SNAPSHOT ts ON f.TRANSFORMER\_SNAPSHOT\_SK = ts.TRANSFORMER\_SNAPSHOT\_SK

JOIN DIM\_STATUS s ON f.STATUS\_SK = s.STATUS\_SK;

```

**### 6.2 Data Quality Monitoring**

```sql

-- Data quality monitoring dashboard

CREATE OR REPLACE VIEW V\_DATA\_QUALITY\_DASHBOARD AS

SELECT

'Placeholder Records' as METRIC\_NAME,

COUNT(\*) as METRIC\_VALUE,

'Count' as METRIC\_TYPE,

CURRENT\_TIMESTAMP as LAST\_UPDATED

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE IS\_PLACEHOLDER = TRUE AND IS\_CURRENT = TRUE

UNION ALL

SELECT

'Pending Backfill Records' as METRIC\_NAME,

COUNT(\*) as METRIC\_VALUE,

'Count' as METRIC\_TYPE,

CURRENT\_TIMESTAMP as LAST\_UPDATED

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE NEEDS\_BACKFILL = TRUE

UNION ALL

SELECT

'Unprocessed Staging Records' as METRIC\_NAME,

COUNT(\*) as METRIC\_VALUE,

'Count' as METRIC\_TYPE,

CURRENT\_TIMESTAMP as LAST\_UPDATED

FROM STG\_PENDING\_OUTAGES

WHERE PROCESSED\_FLAG = FALSE

UNION ALL

SELECT

'Oldest Placeholder Age (Hours)' as METRIC\_NAME,

DATEDIFF(hour, MIN(CREATED\_DATE), CURRENT\_TIMESTAMP) as METRIC\_VALUE,

'Hours' as METRIC\_TYPE,

CURRENT\_TIMESTAMP as LAST\_UPDATED

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE IS\_PLACEHOLDER = TRUE AND IS\_CURRENT = TRUE;

```

**## 7. Data Validation and Exception Handling**

**### 7.1 Automated Data Validation**

```sql

-- Data variance analysis view

CREATE OR REPLACE VIEW V\_BACKFILL\_ANALYSIS AS

SELECT

ts.TRANSFORMER\_CODE,

ts.CUSTOMER\_COUNT as OMS\_COUNT,

ta.ACTUAL\_CUSTOMER\_COUNT as TRANS\_COUNT,

ABS(ts.CUSTOMER\_COUNT - ta.ACTUAL\_CUSTOMER\_COUNT) as DIFFERENCE,

ROUND(ABS(ts.CUSTOMER\_COUNT - ta.ACTUAL\_CUSTOMER\_COUNT) \* 100.0 /

NULLIF(ta.ACTUAL\_CUSTOMER\_COUNT, 0), 2) as DIFFERENCE\_PERCENT,

CASE

WHEN ABS(ts.CUSTOMER\_COUNT - ta.ACTUAL\_CUSTOMER\_COUNT) = 0

THEN 'EXACT\_MATCH'

WHEN ABS(ts.CUSTOMER\_COUNT - ta.ACTUAL\_CUSTOMER\_COUNT) <= 10

THEN 'MINOR\_VARIANCE'

WHEN ABS(ts.CUSTOMER\_COUNT - ta.ACTUAL\_CUSTOMER\_COUNT) <= 100

THEN 'MODERATE\_VARIANCE'

ELSE 'HIGH\_VARIANCE'

END as VARIANCE\_CATEGORY,

ts.CREATED\_DATE as PLACEHOLDER\_CREATED,

ta.LOADED\_DATE as TRANS\_DATA\_LOADED

FROM DIM\_TRANSFORMER\_SNAPSHOT ts

JOIN STG\_TRANSFORMER\_ATTRIBUTES ta

ON ts.TRANSFORMER\_CODE = ta.TRANSFORMER\_CODE

WHERE ts.IS\_PLACEHOLDER = TRUE;

```

**### 7.2 Exception Handling and Alerts**

```sql

-- Exception record table

CREATE TABLE ETL\_EXCEPTIONS (

EXCEPTION\_ID NUMBER(19,0) IDENTITY PRIMARY KEY,

PROCESS\_NAME VARCHAR(100),

TABLE\_NAME VARCHAR(100),

RECORD\_IDENTIFIER VARCHAR(200),

EXCEPTION\_TYPE VARCHAR(50),

EXCEPTION\_MESSAGE VARCHAR(1000),

SEVERITY\_LEVEL VARCHAR(20),

CREATED\_DATE TIMESTAMP\_NTZ DEFAULT CURRENT\_TIMESTAMP,

RESOLVED\_DATE TIMESTAMP\_NTZ,

RESOLUTION\_NOTES VARCHAR(1000),

RESOLVED\_BY VARCHAR(100)

);

-- Automated anomaly detection task

CREATE OR REPLACE TASK detect\_data\_anomalies

WAREHOUSE = ETL\_SMALL\_WH

SCHEDULE = '60 MINUTE'

AS

BEGIN

-- Detect high variance records

INSERT INTO ETL\_EXCEPTIONS (

PROCESS\_NAME, TABLE\_NAME, RECORD\_IDENTIFIER,

EXCEPTION\_TYPE, EXCEPTION\_MESSAGE, SEVERITY\_LEVEL

)

SELECT

'BACKFILL\_VALIDATION',

'DIM\_TRANSFORMER\_SNAPSHOT',

TRANSFORMER\_CODE,

'HIGH\_VARIANCE',

'Customer count variance > 100: OMS=' || OMS\_COUNT ||

', TRANS=' || TRANS\_COUNT || ', Diff=' || DIFFERENCE,

'HIGH'

FROM V\_BACKFILL\_ANALYSIS

WHERE VARIANCE\_CATEGORY = 'HIGH\_VARIANCE'

AND TRANSFORMER\_CODE NOT IN (

SELECT RECORD\_IDENTIFIER

FROM ETL\_EXCEPTIONS

WHERE EXCEPTION\_TYPE = 'HIGH\_VARIANCE'

AND RESOLVED\_DATE IS NULL

);

-- Detect long-running placeholders

INSERT INTO ETL\_EXCEPTIONS (

PROCESS\_NAME, TABLE\_NAME, RECORD\_IDENTIFIER,

EXCEPTION\_TYPE, EXCEPTION\_MESSAGE, SEVERITY\_LEVEL

)

SELECT

'PLACEHOLDER\_MONITORING',

'DIM\_TRANSFORMER\_SNAPSHOT',

TRANSFORMER\_CODE,

'LONG\_RUNNING\_PLACEHOLDER',

'Placeholder record exists for > 7 days: Created=' || CREATED\_DATE,

'MEDIUM'

FROM DIM\_TRANSFORMER\_SNAPSHOT

WHERE IS\_PLACEHOLDER = TRUE

AND CREATED\_DATE < DATEADD(day, -7, CURRENT\_TIMESTAMP)

AND TRANSFORMER\_CODE NOT IN (

SELECT RECORD\_IDENTIFIER

FROM ETL\_EXCEPTIONS

WHERE EXCEPTION\_TYPE = 'LONG\_RUNNING\_PLACEHOLDER'

AND RESOLVED\_DATE IS NULL

);

END;

ALTER TASK detect\_data\_anomalies RESUME;

```

**## 8. Recovery and Rollback Mechanisms**

**### 8.1 Recovery Using Time Travel**

```sql

-- Data recovery stored procedure

CREATE OR REPLACE PROCEDURE emergency\_data\_recovery(recovery\_timestamp TIMESTAMP\_NTZ)

RETURNS STRING

LANGUAGE SQL

AS

$$

DECLARE

backup\_table\_name STRING;

BEGIN

-- Generate unique backup table name

backup\_table\_name := 'DIM\_TRANSFORMER\_SNAPSHOT\_RECOVERY\_' ||

TO\_VARCHAR(CURRENT\_TIMESTAMP, 'YYYYMMDDHH24MISS');

-- Create current state backup

EXECUTE IMMEDIATE 'CREATE TABLE ' || backup\_table\_name ||

' CLONE DIM\_TRANSFORMER\_SNAPSHOT';

-- Restore to specified time point

EXECUTE IMMEDIATE 'CREATE OR REPLACE TABLE DIM\_TRANSFORMER\_SNAPSHOT\_TEMP CLONE DIM\_TRANSFORMER\_SNAPSHOT AT(TIMESTAMP => ''' || recovery\_timestamp || ''')';

-- Replace current table

DROP TABLE DIM\_TRANSFORMER\_SNAPSHOT;

ALTER TABLE DIM\_TRANSFORMER\_SNAPSHOT\_TEMP RENAME TO DIM\_TRANSFORMER\_SNAPSHOT;

-- Log recovery operation

INSERT INTO ETL\_PROCESS\_LOG (

PROCESS\_NAME, STATUS, MESSAGE, CREATED\_DATE

) VALUES (

'EMERGENCY\_RECOVERY', 'SUCCESS',

'Data recovered to ' || recovery\_timestamp || '. Backup: ' || backup\_table\_name,

CURRENT\_TIMESTAMP

);

RETURN 'Recovery completed. Backup table: ' || backup\_table\_name;

EXCEPTION

WHEN OTHER THEN

RETURN 'Recovery failed: ' || SQLERRM;

END;

$$;

```

**### 8.2 Progressive Validation and Correction**

```sql

-- Progressive data correction process

CREATE OR REPLACE PROCEDURE progressive\_data\_correction()

RETURNS STRING

LANGUAGE SQL

AS

$$

DECLARE

correction\_count NUMBER DEFAULT 0;

BEGIN

-- Correct small variances (<= 5%)

UPDATE DIM\_TRANSFORMER\_SNAPSHOT ts

SET

CUSTOMER\_COUNT = ta.ACTUAL\_CUSTOMER\_COUNT,

UPDATED\_DATE = CURRENT\_TIMESTAMP

FROM STG\_TRANSFORMER\_ATTRIBUTES ta

WHERE ts.TRANSFORMER\_CODE = ta.TRANSFORMER\_CODE

AND ts.IS\_PLACEHOLDER = TRUE

AND ABS(ts.CUSTOMER\_COUNT - ta.ACTUAL\_CUSTOMER\_COUNT) \* 100.0 /

NULLIF(ta.ACTUAL\_CUSTOMER\_COUNT, 0) <= 5;

correction\_count := SQLROWCOUNT;

RETURN 'Progressive correction completed. ' || correction\_count || ' records updated.';

END;

$$;

```

**## 9. Cost Optimization and Performance Tuning**

**### 9.1 Storage Optimization**

```sql

-- Set appropriate retention period

ALTER TABLE STG\_PENDING\_OUTAGES SET DATA\_RETENTION\_TIME\_IN\_DAYS = 7;

ALTER TABLE STG\_TRANSFORMER\_ATTRIBUTES SET DATA\_RETENTION\_TIME\_IN\_DAYS = 30;

-- Use transient tables for temporary calculations

CREATE TRANSIENT TABLE temp\_calculation\_results AS

SELECT /\* Temporary calculation logic \*/;

```

**### 9.2 Warehouse Size Optimization**

```sql

-- Create warehouses of different sizes for different workloads

CREATE WAREHOUSE ETL\_SMALL\_WH WITH

WAREHOUSE\_SIZE = 'XSMALL'

AUTO\_SUSPEND = 60

AUTO\_RESUME = TRUE

COMMENT = 'For routine ETL and monitoring tasks';

CREATE WAREHOUSE ETL\_BACKFILL\_WH WITH

WAREHOUSE\_SIZE = 'MEDIUM'

AUTO\_SUSPEND = 60

AUTO\_RESUME = TRUE

COMMENT = 'For backfill and recovery operations';

```

**## 10. Deployment and Maintenance Guide**

**### 10.1 Deployment Checklist**

1. **\*\*Database Object Creation\*\***

- Extend dimension table structure

- Create staging tables

- Create all stored procedures and functions

2. **\*\*Monitoring Setup\*\***

- Create Streams

- Deploy Dynamic Tables

- Start Tasks

3. **\*\*Permission Configuration\*\***

- ETL role permissions

- Monitoring user access

- Business user view permissions

**### 10.2 Maintenance Operations**

```sql

-- Regularly clean up expired data

CREATE OR REPLACE TASK cleanup\_expired\_data

WAREHOUSE = ETL\_SMALL\_WH

SCHEDULE = 'USING CRON 0 2 \* \* SUN' -- Every Sunday at 2am

AS

BEGIN

-- Clean up expired staging data

DELETE FROM STG\_PENDING\_OUTAGES

WHERE PROCESSED\_FLAG = TRUE

AND RECEIVED\_DATE < DATEADD(day, -30, CURRENT\_TIMESTAMP);

-- Clean up resolved exception records

DELETE FROM ETL\_EXCEPTIONS

WHERE RESOLVED\_DATE IS NOT NULL

AND RESOLVED\_DATE < DATEADD(day, -90, CURRENT\_TIMESTAMP);

-- Clean up expired logs

DELETE FROM ETL\_PROCESS\_LOG

WHERE CREATED\_DATE < DATEADD(day, -180, CURRENT\_TIMESTAMP);

END;

ALTER TASK cleanup\_expired\_data RESUME;

```

**Question 5**

Starting 1 Jul 2025, Microsoft Yammer (Viva Engage) will deprecate the OAuth2 based authentication, which means that only Azure Active Directory (AAD) token option is available. See below link for details of how Yammer AAD token authentication works.

<https://techcommunity.microsoft.com/blog/viva_engage_blog/yammer-api-with-aad-tokens-postman-collection/857923>

You are tasked with creating an automatic data pipeline that ingests messages daily from Yammer API v1. The data pipeline does not have an available web URL that you could use as callback URL when you pass parameters to get the AAD token. How would you design your data pipeline in Python or PowerShell to ingest data automatically without human intervention? Note code is not required but please explain:

* which OAuth2 grant type you would use and why
* the key values in header/body that you would pass to the authentication endpoint.
* how you would secure the credentials used in the authentication process.

**Answer:**

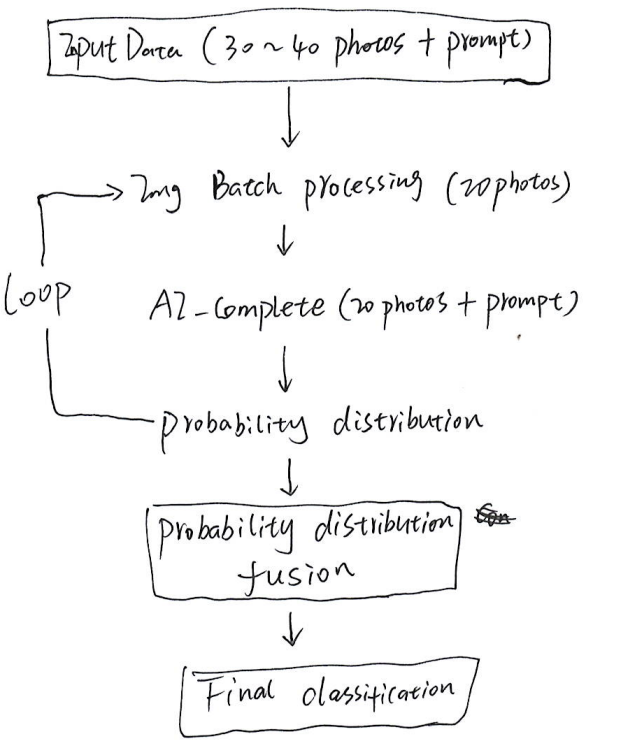
**Question 6**

You are working on an AI use case to classify causes of electricity outages by using the combined information from field notes from electricity technicians and the photos they have taken on the fault sites. Snowflake’s multimodal capability (AI\_COMPLETE) only allows you to process up to 20 images at a time (see below the documentation), how would you design your AI solution in Snowflake if the field crews sometimes attach 30-40 photos per outage? Draw an architecture diagram (hand-drawn is acceptable) of the end-to-end solution. Provide enough details for a junior data & AI engineer to implement the solution.

<https://docs.snowflake.com/en/user-guide/snowflake-cortex/ai-images>

**Answer:**

1. **KeyPoints**
   * The system automatically partitions images into batches of ≤20 using SQL windowing functions, enabling parallel AI\_COMPLETE processing across multiple batches simultaneously. Each batch generates classification probabilities for outage causes.
   * Probability Fusion Strategy，Individual batch probabilities are aggregated using confidence-weighted averaging, where high-confidence predictions receive greater influence in the final classification. The system calculates weighted probability distributions across all batches, ensuring robust predictions even when individual batches have varying quality or relevance.
2. **Workflow Design**

****

**Question 7**

Below is an example of first a few lines of an EIEP3 file, which is standard file format for half hourly electricity meter readings in NZ. Please write a SQL or Python script to load the file from SFTP into Snowflake without losing any useful information of the dataset.

*HDR,ICPHH,11,ENERGYCO,ENRG,DIST,09/06/2025,14:00:00,ID2025060914001,5,202505,E,I*

*DET,1234567890ABCD1,MTR123456789012,F,31/05/2025,1,20.35,5.12,,X,*

*DET,1234567890ABCD1,MTR123456789012,F,31/05/2025,2,19.87,4.98,,X,*

*DET,1234567890ABCD1,MTR123456789012,F,31/05/2025,3,21.10,5.34,,X,*

*DET,1234567890ABCD1,MTR123456789012,F,31/05/2025,4,22.45,5.68,,X,*

**Answer:**

1. **Assumptions**
   * Snowflake Enterprise Edition or higher (needs to support Tasks, Snowpipe, external functions)
   * Can be integrated with cloud storage, have appropriate IAM roles and permissions, and have a stable and secure network connection
   * Data format complies with EIEP3 standards
   * Data volume: file size is within a reasonable range (<100MB), and the frequency of file arrival does not exceed processing capacity
2. **LLM**

Claude Sonnet 4

Prompt: question and “ use snowflake native or advanced function to solve this problem if necessary”

1. **Solution**

**## 2. Snowflake Native Solution Architecture**

**### 2.1 Data Model Design**

**```sql**

**-- Create raw data staging table (retain all original information)**

**CREATE OR REPLACE TABLE STG\_EIEP3\_RAW (**

**FILE\_NAME VARCHAR(500),**

**LINE\_NUMBER NUMBER,**

**RECORD\_TYPE VARCHAR(10),**

**RAW\_LINE VARCHAR(5000),**

**LOAD\_TIMESTAMP TIMESTAMP\_NTZ DEFAULT CURRENT\_TIMESTAMP,**

**PROCESS\_STATUS VARCHAR(20) DEFAULT 'PENDING'**

**);**

**-- Create file header information table**

**CREATE OR REPLACE TABLE EIEP3\_FILE\_HEADERS (**

**FILE\_HEADER\_ID NUMBER(19,0) IDENTITY PRIMARY KEY,**

**FILE\_NAME VARCHAR(500) NOT NULL,**

**RECORD\_TYPE VARCHAR(10) DEFAULT 'HDR',**

**MESSAGE\_TYPE VARCHAR(20),**

**VERSION\_NUMBER VARCHAR(10),**

**SENDER\_CODE VARCHAR(50),**

**SENDER\_TYPE VARCHAR(10),**

**RECIPIENT\_CODE VARCHAR(50),**

**CREATION\_DATE DATE,**

**CREATION\_TIME TIME,**

**MESSAGE\_ID VARCHAR(100),**

**PRIORITY VARCHAR(10),**

**PERIOD VARCHAR(20),**

**TEST\_INDICATOR VARCHAR(5),**

**DUPLICATE\_INDICATOR VARCHAR(5),**

**LOAD\_TIMESTAMP TIMESTAMP\_NTZ DEFAULT CURRENT\_TIMESTAMP**

**);**

**-- Create meter reading detail table**

**CREATE OR REPLACE TABLE EIEP3\_METER\_READINGS (**

**READING\_ID NUMBER(19,0) IDENTITY PRIMARY KEY,**

**FILE\_HEADER\_ID NUMBER(19,0),**

**FILE\_NAME VARCHAR(500),**

**ICP\_IDENTIFIER VARCHAR(50) NOT NULL,**

**METER\_SERIAL\_NUMBER VARCHAR(50),**

**REGISTER\_CODE VARCHAR(10),**

**READING\_DATE DATE,**

**TRADING\_PERIOD NUMBER(2,0),**

**CONSUMPTION\_KWH NUMBER(12,3),**

**MAXIMUM\_DEMAND\_KW NUMBER(12,3),**

**RESERVED\_FIELD VARCHAR(50),**

**QUALITY\_FLAG VARCHAR(10),**

**LOAD\_TIMESTAMP TIMESTAMP\_NTZ DEFAULT CURRENT\_TIMESTAMP,**

**FOREIGN KEY (FILE\_HEADER\_ID) REFERENCES EIEP3\_FILE\_HEADERS(FILE\_HEADER\_ID)**

**);**

**-- Create processing log table**

**CREATE OR REPLACE TABLE EIEP3\_PROCESS\_LOG (**

**LOG\_ID NUMBER(19,0) IDENTITY PRIMARY KEY,**

**FILE\_NAME VARCHAR(500),**

**PROCESS\_STEP VARCHAR(100),**

**STATUS VARCHAR(20),**

**RECORDS\_PROCESSED NUMBER,**

**ERROR\_COUNT NUMBER,**

**ERROR\_MESSAGE VARCHAR(1000),**

**PROCESSING\_TIME\_SECONDS NUMBER(10,2),**

**CREATED\_TIMESTAMP TIMESTAMP\_NTZ DEFAULT CURRENT\_TIMESTAMP**

**);**

**```**

**### 2.2 SFTP Integration Setup**

**```sql**

**-- Create storage integration (for SFTP access)**

**CREATE OR REPLACE STORAGE INTEGRATION EIEP3\_SFTP\_INTEGRATION**

**TYPE = EXTERNAL\_STAGE**

**STORAGE\_PROVIDER = 'S3' -- or other cloud provider**

**ENABLED = TRUE**

**STORAGE\_AWS\_ROLE\_ARN = 'arn:aws:iam::your-account:role/snowflake-sftp-role'**

**STORAGE\_ALLOWED\_LOCATIONS = ('s3://your-eiep3-bucket/');**

**-- Create file format**

**CREATE OR REPLACE FILE FORMAT EIEP3\_CSV\_FORMAT**

**TYPE = 'CSV'**

**FIELD\_DELIMITER = ','**

**RECORD\_DELIMITER = '\n'**

**SKIP\_HEADER = 0**

**FIELD\_OPTIONALLY\_ENCLOSED\_BY = NONE**

**TRIM\_SPACE = TRUE**

**ERROR\_ON\_COLUMN\_COUNT\_MISMATCH = FALSE**

**ESCAPE = NONE**

**ESCAPE\_UNENCLOSED\_FIELD = NONE**

**NULL\_IF = ('');**

**-- Create external stage**

**CREATE OR REPLACE STAGE EIEP3\_SFTP\_STAGE**

**STORAGE\_INTEGRATION = EIEP3\_SFTP\_INTEGRATION**

**URL = 's3://your-eiep3-bucket/incoming/'**

**FILE\_FORMAT = EIEP3\_CSV\_FORMAT;**

**```**

**## 3. Advanced Parsing Stored Procedures**

**### 3.1 Main Processing Procedure**

**```sql**

**CREATE OR REPLACE PROCEDURE PROCESS\_EIEP3\_FILE(FILE\_NAME\_PATTERN VARCHAR)**

**RETURNS TABLE(STATUS VARCHAR, MESSAGE VARCHAR, RECORDS\_PROCESSED NUMBER)**

**LANGUAGE SQL**

**AS**

**$$**

**DECLARE**

**file\_cursor CURSOR FOR**

**SELECT DISTINCT metadata$filename as filename**

**FROM @EIEP3\_SFTP\_STAGE**

**WHERE metadata$filename LIKE FILE\_NAME\_PATTERN;**

**current\_file VARCHAR(500);**

**header\_id NUMBER;**

**total\_processed NUMBER DEFAULT 0;**

**process\_start TIMESTAMP\_NTZ;**

**BEGIN**

**process\_start := CURRENT\_TIMESTAMP;**

**-- Process each matching file**

**FOR file\_record IN file\_cursor DO**

**current\_file := file\_record.filename;**

**-- Step 1: Load raw data**

**CALL LOAD\_RAW\_EIEP3\_DATA(current\_file);**

**-- Step 2: Parse and process header record**

**CALL PARSE\_EIEP3\_HEADER(current\_file) INTO header\_id;**

**-- Step 3: Parse and process detail records**

**CALL PARSE\_EIEP3\_DETAILS(current\_file, header\_id);**

**-- Step 4: Data validation**

**CALL VALIDATE\_EIEP3\_DATA(current\_file);**

**total\_processed := total\_processed + 1;**

**-- Log processing**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, RECORDS\_PROCESSED,**

**PROCESSING\_TIME\_SECONDS**

**) VALUES (**

**current\_file, 'FILE\_COMPLETE', 'SUCCESS', 1,**

**DATEDIFF(second, process\_start, CURRENT\_TIMESTAMP)**

**);**

**END FOR;**

**RETURN TABLE(**

**SELECT 'SUCCESS' as STATUS,**

**'Processed ' || total\_processed || ' files' as MESSAGE,**

**total\_processed as RECORDS\_PROCESSED**

**);**

**EXCEPTION**

**WHEN OTHER THEN**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, ERROR\_MESSAGE**

**) VALUES (**

**current\_file, 'FILE\_PROCESSING', 'ERROR', SQLERRM**

**);**

**RETURN TABLE(**

**SELECT 'ERROR' as STATUS,**

**'Processing failed: ' || SQLERRM as MESSAGE,**

**total\_processed as RECORDS\_PROCESSED**

**);**

**END;**

**$$;**

**```**

**### 3.2 Raw Data Load Procedure**

**```sql**

**CREATE OR REPLACE PROCEDURE LOAD\_RAW\_EIEP3\_DATA(FILE\_NAME VARCHAR)**

**RETURNS NUMBER**

**LANGUAGE SQL**

**AS**

**$$**

**DECLARE**

**records\_loaded NUMBER DEFAULT 0;**

**BEGIN**

**-- Use COPY INTO to load raw file content**

**COPY INTO STG\_EIEP3\_RAW (FILE\_NAME, RAW\_LINE, LINE\_NUMBER)**

**FROM (**

**SELECT**

**metadata$filename,**

**$1 as raw\_line,**

**metadata$file\_row\_number as line\_number**

**FROM @EIEP3\_SFTP\_STAGE**

**WHERE metadata$filename = FILE\_NAME**

**)**

**FILE\_FORMAT = (**

**TYPE = 'CSV'**

**FIELD\_DELIMITER = '\t' -- Use tab, treat whole line as single field**

**RECORD\_DELIMITER = '\n'**

**SKIP\_HEADER = 0**

**);**

**records\_loaded := SQLROWCOUNT;**

**-- Identify record type**

**UPDATE STG\_EIEP3\_RAW**

**SET RECORD\_TYPE = CASE**

**WHEN RAW\_LINE LIKE '\*HDR,%' THEN 'HDR'**

**WHEN RAW\_LINE LIKE '\*DET,%' THEN 'DET'**

**ELSE 'UNKNOWN'**

**END**

**WHERE FILE\_NAME = FILE\_NAME**

**AND RECORD\_TYPE IS NULL;**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, RECORDS\_PROCESSED**

**) VALUES (**

**FILE\_NAME, 'RAW\_LOAD', 'SUCCESS', records\_loaded**

**);**

**RETURN records\_loaded;**

**EXCEPTION**

**WHEN OTHER THEN**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, ERROR\_MESSAGE**

**) VALUES (**

**FILE\_NAME, 'RAW\_LOAD', 'ERROR', SQLERRM**

**);**

**RETURN 0;**

**END;**

**$$;**

**```**

**### 3.3 Header Record Parsing Procedure**

**```sql**

**CREATE OR REPLACE PROCEDURE PARSE\_EIEP3\_HEADER(FILE\_NAME VARCHAR)**

**RETURNS NUMBER**

**LANGUAGE SQL**

**AS**

**$$**

**DECLARE**

**header\_id NUMBER;**

**header\_line VARCHAR(5000);**

**header\_parts ARRAY;**

**BEGIN**

**-- Get header record**

**SELECT RAW\_LINE INTO header\_line**

**FROM STG\_EIEP3\_RAW**

**WHERE FILE\_NAME = FILE\_NAME**

**AND RECORD\_TYPE = 'HDR'**

**LIMIT 1;**

**-- Remove leading/trailing asterisks and split fields**

**header\_line := TRIM(header\_line, '\*');**

**header\_parts := SPLIT(header\_line, ',');**

**-- Insert parsed header record**

**INSERT INTO EIEP3\_FILE\_HEADERS (**

**FILE\_NAME, RECORD\_TYPE, MESSAGE\_TYPE, VERSION\_NUMBER,**

**SENDER\_CODE, SENDER\_TYPE, RECIPIENT\_CODE,**

**CREATION\_DATE, CREATION\_TIME, MESSAGE\_ID,**

**PRIORITY, PERIOD, TEST\_INDICATOR, DUPLICATE\_INDICATOR**

**) VALUES (**

**FILE\_NAME,**

**header\_parts[0], -- HDR**

**header\_parts[1], -- ICPHH**

**header\_parts[2], -- 11**

**header\_parts[3], -- ENERGYCO**

**header\_parts[4], -- ENRG**

**header\_parts[5], -- DIST**

**TRY\_TO\_DATE(header\_parts[6], 'DD/MM/YYYY'), -- 09/06/2025**

**TRY\_TO\_TIME(header\_parts[7], 'HH24:MI:SS'), -- 14:00:00**

**header\_parts[8], -- ID2025060914001**

**header\_parts[9], -- 5**

**header\_parts[10], -- 202505**

**header\_parts[11], -- E**

**header\_parts[12] -- I**

**);**

**-- Get generated ID**

**SELECT FILE\_HEADER\_ID INTO header\_id**

**FROM EIEP3\_FILE\_HEADERS**

**WHERE FILE\_NAME = FILE\_NAME**

**ORDER BY LOAD\_TIMESTAMP DESC**

**LIMIT 1;**

**-- Update processing status**

**UPDATE STG\_EIEP3\_RAW**

**SET PROCESS\_STATUS = 'PROCESSED'**

**WHERE FILE\_NAME = FILE\_NAME**

**AND RECORD\_TYPE = 'HDR';**

**RETURN header\_id;**

**EXCEPTION**

**WHEN OTHER THEN**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, ERROR\_MESSAGE**

**) VALUES (**

**FILE\_NAME, 'HEADER\_PARSE', 'ERROR', SQLERRM**

**);**

**RETURN NULL;**

**END;**

**$$;**

**```**

**### 3.4 Detail Record Parsing Procedure**

**```sql**

**CREATE OR REPLACE PROCEDURE PARSE\_EIEP3\_DETAILS(FILE\_NAME VARCHAR, HEADER\_ID NUMBER)**

**RETURNS NUMBER**

**LANGUAGE SQL**

**AS**

**$$**

**DECLARE**

**records\_processed NUMBER DEFAULT 0;**

**BEGIN**

**-- Bulk insert parsed detail records**

**INSERT INTO EIEP3\_METER\_READINGS (**

**FILE\_HEADER\_ID, FILE\_NAME, ICP\_IDENTIFIER, METER\_SERIAL\_NUMBER,**

**REGISTER\_CODE, READING\_DATE, TRADING\_PERIOD,**

**CONSUMPTION\_KWH, MAXIMUM\_DEMAND\_KW, RESERVED\_FIELD, QUALITY\_FLAG**

**)**

**SELECT**

**HEADER\_ID,**

**FILE\_NAME,**

**detail\_parts[1] as ICP\_IDENTIFIER,**

**detail\_parts[2] as METER\_SERIAL\_NUMBER,**

**detail\_parts[3] as REGISTER\_CODE,**

**TRY\_TO\_DATE(detail\_parts[4], 'DD/MM/YYYY') as READING\_DATE,**

**TRY\_TO\_NUMBER(detail\_parts[5]) as TRADING\_PERIOD,**

**TRY\_TO\_NUMBER(detail\_parts[6], 12, 3) as CONSUMPTION\_KWH,**

**TRY\_TO\_NUMBER(detail\_parts[7], 12, 3) as MAXIMUM\_DEMAND\_KW,**

**NULLIF(detail\_parts[8], '') as RESERVED\_FIELD,**

**detail\_parts[9] as QUALITY\_FLAG**

**FROM (**

**SELECT**

**FILE\_NAME,**

**SPLIT(TRIM(RAW\_LINE, '\*'), ',') as detail\_parts**

**FROM STG\_EIEP3\_RAW**

**WHERE FILE\_NAME = FILE\_NAME**

**AND RECORD\_TYPE = 'DET'**

**AND PROCESS\_STATUS = 'PENDING'**

**);**

**records\_processed := SQLROWCOUNT;**

**-- Update processing status**

**UPDATE STG\_EIEP3\_RAW**

**SET PROCESS\_STATUS = 'PROCESSED'**

**WHERE FILE\_NAME = FILE\_NAME**

**AND RECORD\_TYPE = 'DET';**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, RECORDS\_PROCESSED**

**) VALUES (**

**FILE\_NAME, 'DETAIL\_PARSE', 'SUCCESS', records\_processed**

**);**

**RETURN records\_processed;**

**EXCEPTION**

**WHEN OTHER THEN**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, ERROR\_MESSAGE**

**) VALUES (**

**FILE\_NAME, 'DETAIL\_PARSE', 'ERROR', SQLERRM**

**);**

**RETURN 0;**

**END;**

**$$;**

**```**

**## 4. Data Validation and Quality Checks**

**### 4.1 Data Validation Procedure**

**```sql**

**CREATE OR REPLACE PROCEDURE VALIDATE\_EIEP3\_DATA(FILE\_NAME VARCHAR)**

**RETURNS TABLE(VALIDATION\_RULE VARCHAR, STATUS VARCHAR, ERROR\_COUNT NUMBER)**

**LANGUAGE SQL**

**AS**

**$$**

**DECLARE**

**validation\_results RESULTSET;**

**BEGIN**

**-- Create temporary table to store validation results**

**CREATE OR REPLACE TEMPORARY TABLE temp\_validation\_results (**

**validation\_rule VARCHAR(100),**

**status VARCHAR(20),**

**error\_count NUMBER**

**);**

**-- Validation 1: Check required fields**

**INSERT INTO temp\_validation\_results**

**SELECT**

**'REQUIRED\_FIELDS\_CHECK' as validation\_rule,**

**CASE WHEN error\_count = 0 THEN 'PASS' ELSE 'FAIL' END as status,**

**error\_count**

**FROM (**

**SELECT COUNT(\*) as error\_count**

**FROM EIEP3\_METER\_READINGS**

**WHERE FILE\_NAME = FILE\_NAME**

**AND (ICP\_IDENTIFIER IS NULL**

**OR METER\_SERIAL\_NUMBER IS NULL**

**OR READING\_DATE IS NULL**

**OR TRADING\_PERIOD IS NULL)**

**);**

**-- Validation 2: Check trading period range (1-48)**

**INSERT INTO temp\_validation\_results**

**SELECT**

**'TRADING\_PERIOD\_RANGE' as validation\_rule,**

**CASE WHEN error\_count = 0 THEN 'PASS' ELSE 'FAIL' END as status,**

**error\_count**

**FROM (**

**SELECT COUNT(\*) as error\_count**

**FROM EIEP3\_METER\_READINGS**

**WHERE FILE\_NAME = FILE\_NAME**

**AND (TRADING\_PERIOD < 1 OR TRADING\_PERIOD > 48)**

**);**

**-- Validation 3: Check reasonableness of reading values**

**INSERT INTO temp\_validation\_results**

**SELECT**

**'CONSUMPTION\_RANGE\_CHECK' as validation\_rule,**

**CASE WHEN error\_count = 0 THEN 'PASS' ELSE 'FAIL' END as status,**

**error\_count**

**FROM (**

**SELECT COUNT(\*) as error\_count**

**FROM EIEP3\_METER\_READINGS**

**WHERE FILE\_NAME = FILE\_NAME**

**AND (CONSUMPTION\_KWH < 0 OR CONSUMPTION\_KWH > 10000)**

**);**

**-- Validation 4: Check for duplicate records**

**INSERT INTO temp\_validation\_results**

**SELECT**

**'DUPLICATE\_RECORDS\_CHECK' as validation\_rule,**

**CASE WHEN error\_count = 0 THEN 'PASS' ELSE 'FAIL' END as status,**

**error\_count**

**FROM (**

**SELECT COUNT(\*) - COUNT(DISTINCT ICP\_IDENTIFIER, METER\_SERIAL\_NUMBER, READING\_DATE, TRADING\_PERIOD) as error\_count**

**FROM EIEP3\_METER\_READINGS**

**WHERE FILE\_NAME = FILE\_NAME**

**);**

**-- Log validation results**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, RECORDS\_PROCESSED**

**)**

**SELECT**

**FILE\_NAME,**

**'VALIDATION\_' || validation\_rule,**

**status,**

**error\_count**

**FROM temp\_validation\_results;**

**-- Return validation results**

**validation\_results := (SELECT \* FROM temp\_validation\_results);**

**RETURN TABLE(validation\_results);**

**END;**

**$$;**

**```**

**## 5. Automation and Scheduling**

**### 5.1 File Monitoring and Automated Processing**

**```sql**

**-- Create automated processing task**

**CREATE OR REPLACE TASK EIEP3\_AUTO\_PROCESSOR**

**WAREHOUSE = ETL\_WH**

**SCHEDULE = '15 MINUTE'**

**AS**

**DECLARE**

**files\_to\_process RESULTSET;**

**current\_file VARCHAR(500);**

**processing\_result RESULTSET;**

**BEGIN**

**-- Find unprocessed EIEP3 files**

**files\_to\_process := (**

**SELECT DISTINCT metadata$filename as filename**

**FROM @EIEP3\_SFTP\_STAGE**

**WHERE metadata$filename LIKE '%.EIEP3'**

**AND metadata$filename NOT IN (**

**SELECT DISTINCT FILE\_NAME**

**FROM EIEP3\_PROCESS\_LOG**

**WHERE STATUS = 'SUCCESS'**

**AND PROCESS\_STEP = 'FILE\_COMPLETE'**

**)**

**);**

**-- Process each file**

**FOR file\_record IN files\_to\_process DO**

**current\_file := file\_record.filename;**

**-- Call main processing procedure**

**CALL PROCESS\_EIEP3\_FILE(current\_file) INTO processing\_result;**

**-- Optional: send notification or trigger downstream processing**

**END FOR;**

**EXCEPTION**

**WHEN OTHER THEN**

**INSERT INTO EIEP3\_PROCESS\_LOG (**

**FILE\_NAME, PROCESS\_STEP, STATUS, ERROR\_MESSAGE**

**) VALUES (**

**'AUTO\_PROCESSOR', 'TASK\_EXECUTION', 'ERROR', SQLERRM**

**);**

**END;**

**-- Start the task**

**ALTER TASK EIEP3\_AUTO\_PROCESSOR RESUME;**

**```**

**### 5.2 Real-time Processing with Snowpipe**

**```sql**

**-- Create Snowpipe for real-time file processing**

**CREATE OR REPLACE PIPE EIEP3\_AUTO\_PIPE**

**AUTO\_INGEST = TRUE**

**AS**

**COPY INTO STG\_EIEP3\_RAW (FILE\_NAME, RAW\_LINE, LINE\_NUMBER)**

**FROM (**

**SELECT**

**metadata$filename,**

**$1 as raw\_line,**

**metadata$file\_row\_number as line\_number**

**FROM @EIEP3\_SFTP\_STAGE**

**)**

**FILE\_FORMAT = (**

**TYPE = 'CSV'**

**FIELD\_DELIMITER = '\t'**

**RECORD\_DELIMITER = '\n'**

**SKIP\_HEADER = 0**

**);**

**-- Show Snowpipe notification channel (for configuring cloud storage events)**

**SELECT SYSTEM$PIPE\_STATUS('EIEP3\_AUTO\_PIPE');**

**```**

**## 6. Monitoring and Reporting Views**

**### 6.1 Processing Status Monitoring**

**```sql**

**-- File processing status monitoring view**

**CREATE OR REPLACE VIEW V\_EIEP3\_PROCESSING\_STATUS AS**

**SELECT**

**FILE\_NAME,**

**MAX(CASE WHEN PROCESS\_STEP = 'RAW\_LOAD' THEN STATUS END) as RAW\_LOAD\_STATUS,**

**MAX(CASE WHEN PROCESS\_STEP = 'HEADER\_PARSE' THEN STATUS END) as HEADER\_PARSE\_STATUS,**

**MAX(CASE WHEN PROCESS\_STEP = 'DETAIL\_PARSE' THEN STATUS END) as DETAIL\_PARSE\_STATUS,**

**MAX(CASE WHEN PROCESS\_STEP = 'FILE\_COMPLETE' THEN STATUS END) as OVERALL\_STATUS,**

**SUM(CASE WHEN PROCESS\_STEP = 'DETAIL\_PARSE' THEN RECORDS\_PROCESSED ELSE 0 END) as TOTAL\_READINGS,**

**MAX(CREATED\_TIMESTAMP) as LAST\_PROCESSED**

**FROM EIEP3\_PROCESS\_LOG**

**WHERE PROCESS\_STEP IN ('RAW\_LOAD', 'HEADER\_PARSE', 'DETAIL\_PARSE', 'FILE\_COMPLETE')**

**GROUP BY FILE\_NAME**

**ORDER BY LAST\_PROCESSED DESC;**

**```**

**### 6.2 Data Quality Report**

**```sql**

**-- Data quality overview**

**CREATE OR REPLACE VIEW V\_EIEP3\_DATA\_QUALITY AS**

**SELECT**

**DATE(h.CREATION\_DATE) as FILE\_DATE,**

**COUNT(DISTINCT h.FILE\_NAME) as FILES\_PROCESSED,**

**COUNT(r.READING\_ID) as TOTAL\_READINGS,**

**COUNT(CASE WHEN r.QUALITY\_FLAG = 'X' THEN 1 END) as FLAGGED\_READINGS,**

**AVG(r.CONSUMPTION\_KWH) as AVG\_CONSUMPTION,**

**MAX(r.CONSUMPTION\_KWH) as MAX\_CONSUMPTION,**

**COUNT(CASE WHEN r.CONSUMPTION\_KWH = 0 THEN 1 END) as ZERO\_READINGS**

**FROM EIEP3\_FILE\_HEADERS h**

**JOIN EIEP3\_METER\_READINGS r ON h.FILE\_HEADER\_ID = r.FILE\_HEADER\_ID**

**GROUP BY DATE(h.CREATION\_DATE)**

**ORDER BY FILE\_DATE DESC;**

**```**

**## 7. Usage Examples**

**### 7.1 Manually Process a Single File**

**```sql**

**-- Process a specific file**

**CALL PROCESS\_EIEP3\_FILE('EIEP3\_20250609\_140000.dat');**

**-- View processing result**

**SELECT \* FROM V\_EIEP3\_PROCESSING\_STATUS**

**WHERE FILE\_NAME = 'EIEP3\_20250609\_140000.dat';**

**```**

**### 7.2 Batch Process Multiple Files**

**```sql**

**-- Process all pending files**

**CALL PROCESS\_EIEP3\_FILE('%.EIEP3');**

**-- View today's processing statistics**

**SELECT \* FROM V\_EIEP3\_DATA\_QUALITY**

**WHERE FILE\_DATE = CURRENT\_DATE;**

**```**

**### 7.3 Data Query Example**

**```sql**

**-- Query meter reading data for a specific ICP**

**SELECT**

**h.CREATION\_DATE,**

**h.MESSAGE\_ID,**

**r.ICP\_IDENTIFIER,**

**r.METER\_SERIAL\_NUMBER,**

**r.READING\_DATE,**

**r.TRADING\_PERIOD,**

**r.CONSUMPTION\_KWH,**

**r.MAXIMUM\_DEMAND\_KW,**

**r.QUALITY\_FLAG**

**FROM EIEP3\_FILE\_HEADERS h**

**JOIN EIEP3\_METER\_READINGS r ON h.FILE\_HEADER\_ID = r.FILE\_HEADER\_ID**

**WHERE r.ICP\_IDENTIFIER = '1234567890ABCD1'**

**AND r.READING\_DATE BETWEEN '2025-05-01' AND '2025-06-01'**

**ORDER BY r.READING\_DATE, r.TRADING\_PERIOD;**

**```**

**## 8. Optimization and Best Practices**

**### 8.1 Performance Optimization**

**```sql**

**-- Create clustering key to optimize query performance**

**ALTER TABLE EIEP3\_METER\_READINGS**

**CLUSTER BY (READING\_DATE, ICP\_IDENTIFIER);**

**-- Create key indexes**

**CREATE INDEX IDX\_EIEP3\_READINGS\_ICP\_DATE**

**ON EIEP3\_METER\_READINGS(ICP\_IDENTIFIER, READING\_DATE);**

**CREATE INDEX IDX\_EIEP3\_READINGS\_METER\_DATE**

**ON EIEP3\_METER\_READINGS(METER\_SERIAL\_NUMBER, READING\_DATE);**

**```**

**### 8.2 Data Retention Policy**

**```sql**

**-- Set data retention period**

**ALTER TABLE STG\_EIEP3\_RAW SET DATA\_RETENTION\_TIME\_IN\_DAYS = 7;**

**ALTER TABLE EIEP3\_PROCESS\_LOG SET DATA\_RETENTION\_TIME\_IN\_DAYS = 90;**

**ALTER TABLE EIEP3\_METER\_READINGS SET DATA\_RETENTION\_TIME\_IN\_DAYS = 2555; -- 7 years**

**```**

**This solution fully leverages Snowflake's native features, including:**

**- \*\*Snowpipe\*\* for real-time file ingestion**

**- \*\*Tasks\*\* for automated processing**

**- \*\*Stored Procedures\*\* for complex business logic**

**- \*\*Time Travel\*\* for data protection**

**- \*\*Clustering\*\* for performance optimization**

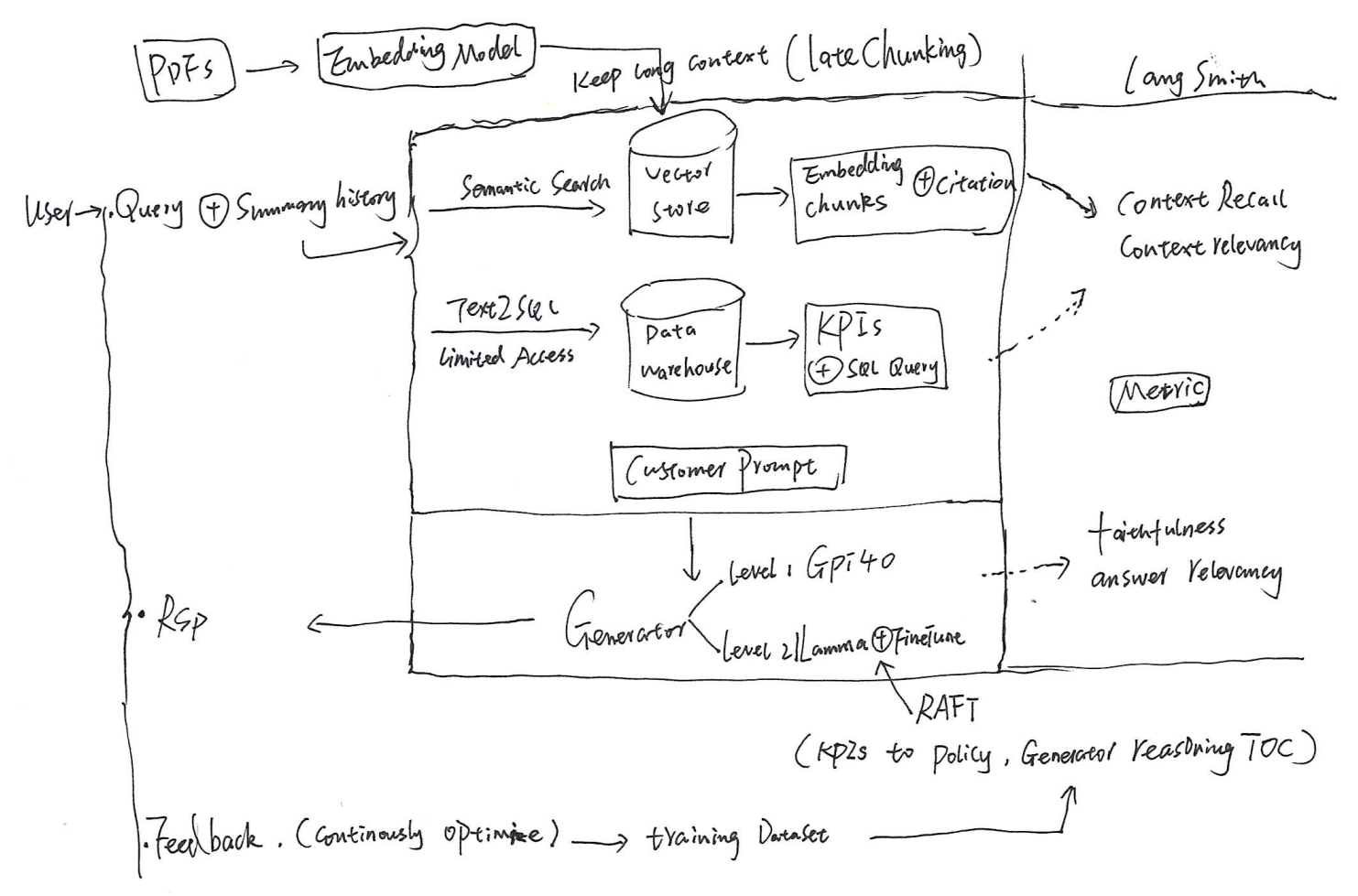
**- \*\*Views\*\* for data presentation and monitoring**

**Question 8**

You are working on an AI chatbot that can help the policy department to draft internal policies. The policy needs to quote KPIs from the company’s data warehouse, including total number of customers, total revenue, total spend by department, etc. The policy also needs to refer to the previous versions (there were about 20 of them and in PDF) published in the past 20 years. Draw an architecture diagram of the solution and describe the key points of how an AI engineer should build it if you are the technical lead of the project. Explain how you would measure the accuracy and quality of the chatbot response.

**Answer:**

1. **Assumptions**
   1. Snowflake stores historical KPI data, including current and target values.
   2. Internal policies are based on KPIs and do not currently consider deeper business logic.
   3. PDF content is only available in English.
2. **Key Points**
   1. We prefer to use LangChain ecosystem which is fully compatible with Snowflake, simple and feature-rich.
   2. To process long PDF files while maintaining contextual semantic accuracy, we use late chunking. However, this approach adds a small amount of query latency.
   3. To extract relevant KPIs, a Text-to-SQL model needs to be integrated. GPT-4o combined with LangChain is a strong candidate. Strict access control must be implemented on the data warehouse to avoid security issues.
   4. We use LangSmith to monitor and evaluate quality and accuracy. The system returns PDF citation and generated SQL queries for retrieving KPIs, helping to evaluate the source and logic .
   5. On the front end, users can provide feedback by clicking on relevant topics or making modifications. This interaction helps build a continuously growing training dataset.
   6. In the first version, it relies on GPT-4o without any fine-tuning. In the second version, it can plan to enhance the system performance by fine-tuning LLM to identify the chain of thoughts (CoT) between KPIs and policies. Eventually, the RAG system will be able to automatically generate new strategies based on the latest KPIs.
   7. To continuously improve performance, it is recommended to A/B test different model variants and feature strategies.
3. **Design**

****