Script that processes HL7 data from JSON files stored in S3 buckets or RedisJSON and stores the transformed/processed data in PostgreSQL schema.

See LinkedIn Post

Requires:

```
Use create_sample_data.py to generate anonymized sample data

1. Apache Spark Standalone or Cluster (3.4.1 or later)
    I set up a Standalone Spark Master with 5 worker nodes
    https://spark.apache.org/downloads.html
2. Have following JARS downloaded - in my case for Spark v3.4.1:
    /var/tmp/sparkjars/postgresql-42.6.0.jar
    /var/tmp/sparkjars/aws-java-sdk-bundle-1.12.262.jar
    /var/tmp/sparkjars/hadoop-aws-3.3.4.jar
3. Postgres JDBC driver (postgresql-42.6.0.jar or later)
4. S3 Bucket name (<s3_bucket_name>) (Mirth connect is storing
                                    RAW HL7 messages as JSON file
                                    in a S3 bucket)
5. S3 Bucket Prefix (yyyy/m/d)
6. S3 Bucket contains, single, or a set of JSON files that are
    formated like the Sample JSON file.
7. etl.config file to store configuration details etc:
    [reportdb]
    host=
    port=5432
    dbname=
    dbuser=
    dbuserpass=

    [spark]
    master=
    masterport=7077

    [redis]
    host=
    port=6379

    [aws]
    access.key=
    secret.key=
    [constants]
    IGNORE_SEG_FIELDS=['PID_1','PID_12','PV1_1','IN1_1','EVN_1','OBX_1','AL1_
    IGNORE_COMPONENT_FIELDS=['CX_4','CX_5','XTN_2','XTN_3','XTN_5','XTN_6','X
    HL7_SEGMENTS=['pid','pv1','pv2','pd1','evn','in1','in2','obx','al1','gt1'
8. List of fields to ignore stored in "hl7_field_names_to_ignore.txt" file
    Sample:
        al1_1_set_id_al1_si_none_1
        evn_1_event_type_code_id_none
        evn_5_operator_id_xcn_2_family_name_2
        evn_5_operator_id_xcn_3_given_name_2
        evn_5_operator_id_xcn_9_assigning_authority_1
        gt1_10_guarantor_type_is_none_1
        gt1_11_guarantor_relationship_ce_1_identifier_1
        gt1_12_guarantor_ssn_st_none_1
        gt1_16_guarantor_employer_name_xpn_1_family_name_1
        gt1_17_guarantor_employer_address_xad_1_street_address_1
        gt1_17_guarantor_employer_address_xad_3_city_1
        gt1_17_guarantor_employer_address_xad_4_state_or_province_1
        gt1_17_guarantor_employer_address_xad_5_zip_or_postal_code_1
        gt1_17_guarantor_employer_address_xad_6_country_1
        gt1_18_guarantor_employer_phone_number_xtn_1_telephone_number_1
        gt1_20_guarantor_employment_status_is_none_1
        gt1_23_guarantor_credit_rating_code_ce_1_identifier_1
        gt1_2_guarantor_number_cx_1_id_number_1
        gt1_3_guarantor_name_xpn_4_suffix_e_g_jr_or_iii_1
        gt1_3_guarantor_name_xpn_5_prefix_e_g_dr_1
        gt1_3_guarantor_name_xpn_7_name_type_code_1
        gt1_4_guarantor_spouse_name_xpn_1_family_name_1
        gt1_4_guarantor_spouse_name_xpn_2_given_name_1
        gt1_5_guarantor_address_xad_1_street_address_1
9. Field mapping file "field_map.txt",
    see field_name = f'{ac_name}_{ac_long_name}_{fc_name}_{fc_long_name}':
    Sample:
        ('pid_2_patient_id_cx_1_id_number','pt_id_4')
        ('pid_30_patient_death_indicator_id_none','pt_death_indicator')
        ('pid_3_patient_identifier_list_cx_1_id_number_2','pt_id_2')
        ('pid_3_patient_identifier_list_cx_1_id_number_3','pt_id_5')
        ('pid_4_alternate_patient_id_pid_cx_1_id_number','pt_id_3')
        ('pid_5_patient_name_xpn_1_family_name','pt_last_name')
```

How-to Run - S3:

```
>./s3_json_to_psql_etl.py --adt-feed-name "acme,roadrunner" --s3-bucket-prefi

23/09/02 16:15:08 WARN NativeCodeLoader: Unable to load native-hadoop library
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogL

**** Starting for acme

23/09/02 16:15:10 WARN MetricsConfig: Cannot locate configuration: tried hado
23/09/02 16:15:53 WARN package: Truncated the string representation of a plan

**** Stored data in table v4_acme
**** Completed for acme
**** Starting for roadrunner
**** Stored data in table v4_roadrunner
**** Completed for roadrunner
```

How-to Run - RedisJSON:

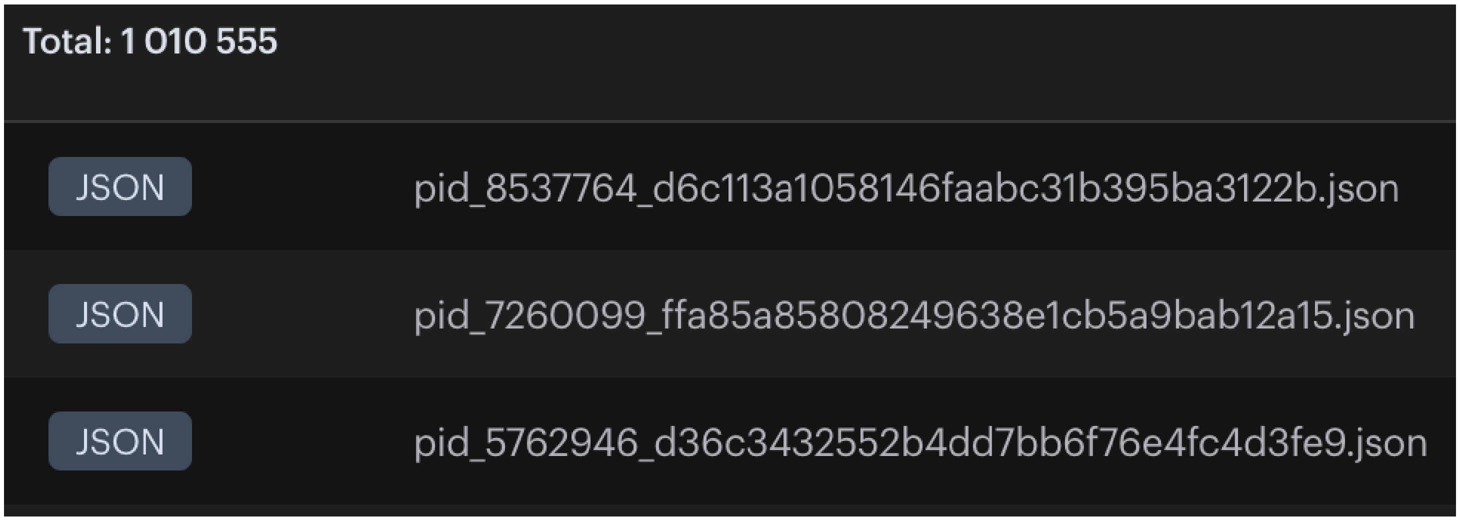
```
>./s3_redis_json_to_psql_etl.py --adt-feed-name redis_$(date '+%s')

23/10/05 10:22:27 WARN NativeCodeLoader: Unable to load native-hadoop library
Setting default log level to "WARN".

**** Starting for redis_1696526545
**** Get Redis Jsons ****
**** Lower Case Colunmn Names ****
**** Process HL7 ****
**** Start Process ****
```

Example RedisJSON:

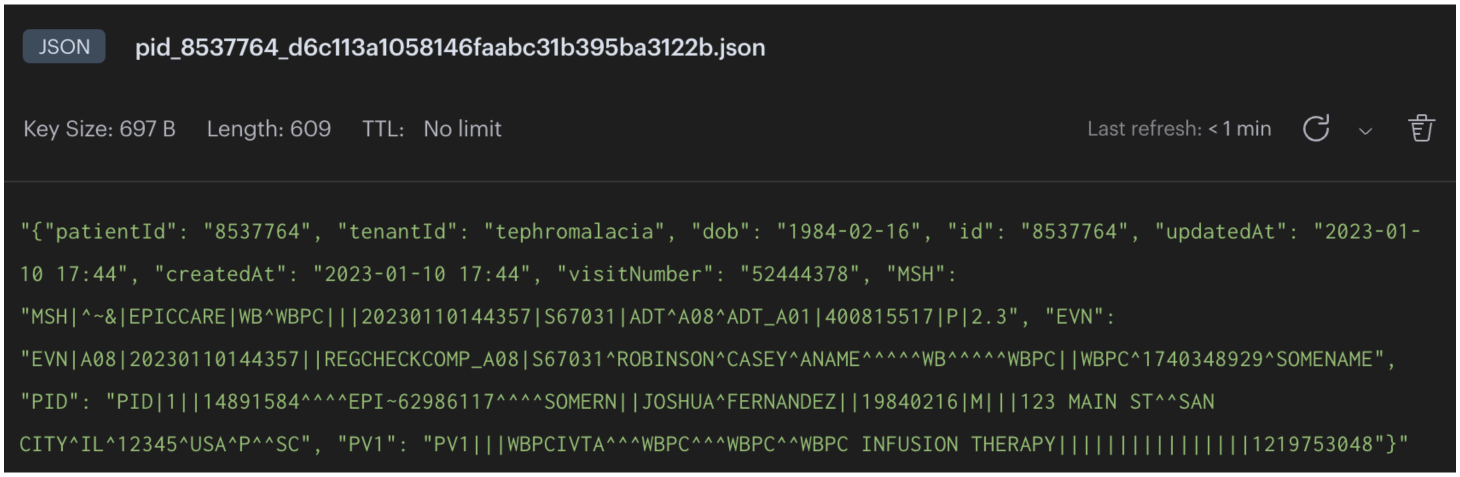
Sample RedisJSON Key Names:



```
Total: 1 010 555

JSON    pid_8537764_d6c113a1058146faabc31b395ba3122b.json
JSON    pid_7260099_ffa85a85808249638e1cb5a9bab12a15.json
JSON    pid_5762946_d36c3432552b4dd7bb6f76e4fc4d3fe9.json
```

Sample RedisJSON Key Content:



```
JSON  pid_8537764_d6c113a1058146faabc31b395ba3122b.json

Key Size: 697 B    Length: 609    TTL:  No limit                Last refresh: < 1 min
```

"{"patientId": "8537764", "tenantId": "tephromalacia", "dob": "1984-02-16", "id": "8537764", "updatedAt": "2023-01-10 17:44", "createdAt": "2023-01-10 17:44", "visitNumber": "52444378", "MSH": "MSH|^~&|EPICCARE|WB^WBPC|||20230110144357|S67031|ADT^A08^ADT_A01|400815517|P|2.3", "EVN": "EVN|A08|20230110144357||REGCHECKCOMP_A08|S67031^ROBINSON^CASEY^ANAME^^^^^WB^^^^^WBPC||WBPC^1740348929^SOMENAME", "PID": "PID|1||14891584^^^^EPI~62986117^^^^SOMERN||JOSHUA^FERNANDEZ||19840216|M|||123 MAIN ST^^SAN CITY^IL^12345^USA^P^^SC", "PV1": "PV1|||WBPCIVTA^^^WBPC^^^WBPC^^WBPC INFUSION THERAPY||||||||||||||||1219753048"}"

Example table names:

```
public | v4_acme       | table |
public | v4_roadrunner | table |
```

Sample JSON file:

```
{
  "patientId": "123456789",
  "tenantId": "<tenant id>",
  "dob": "1990-01-01",
  "id": 123456788,
  "updatedAt": "2023-09-01 00:59:22",
  "createdAt": "2023-09-01 00:59:22",
  "visitNumber": "987654321",
  "MSH": "MSH|^~&|EPICCARE|WB^WBVC|||20230901095922|S327628|ADT^A08^ADT_A01|7
  "EVN": "EVN|A08|20230901095922||REG_UPDATE_VISIT_CHANGE|S327628^RNLASTNAME^
  "PID": "PID|1||12345678^^^^EPI~123456789^^^^SOMEMRN||FIRSTNAME^LASTNAME||19
  "PV1": "PV1|||WBVCERDA^ED07^07^WBVC^^^WBVC^^WBVC EMERGENCY||||||||||||||||1
}
```

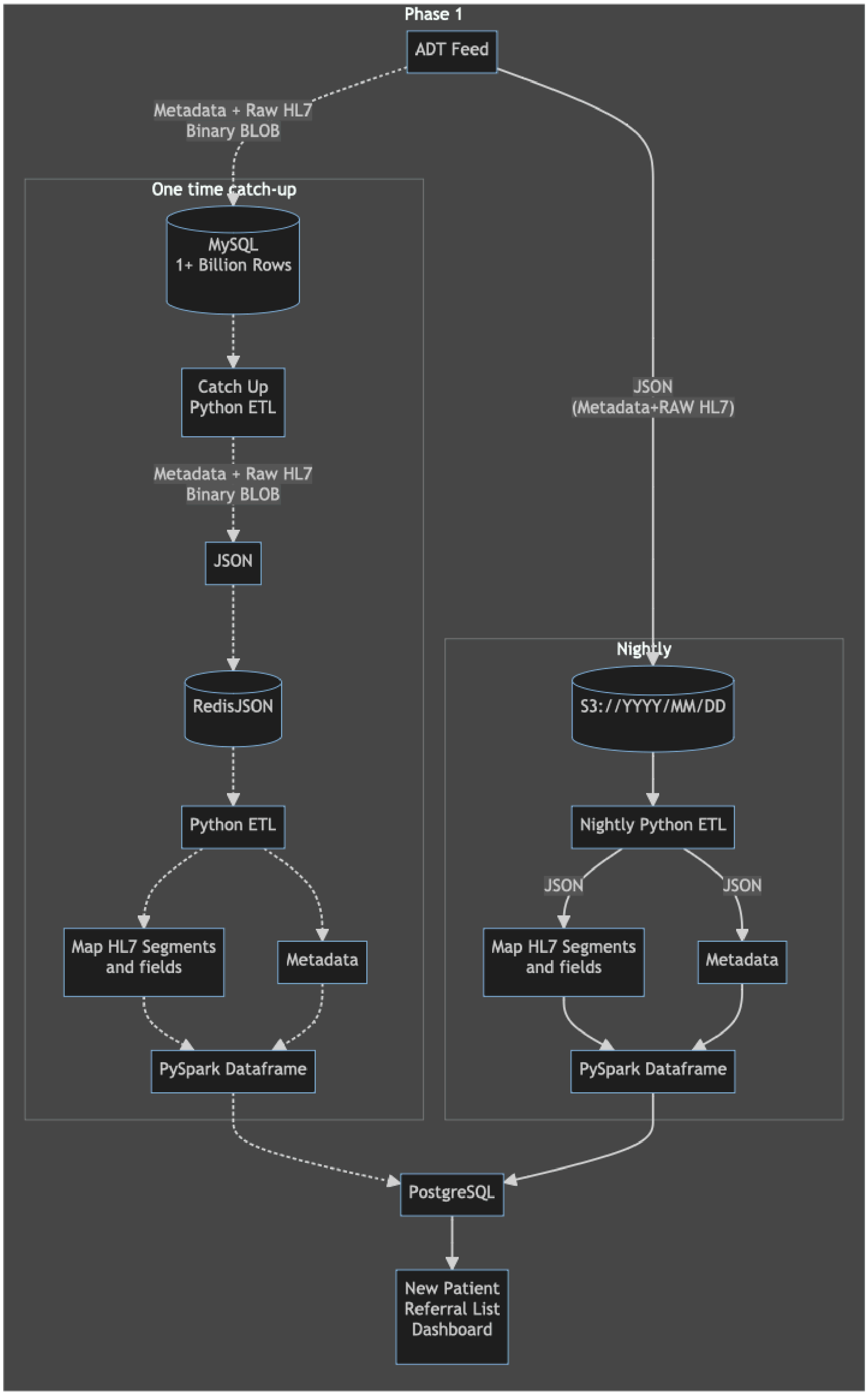
Sample PostgreSQL table:

```
              Column               | Type | Collation | Nullable | Default | S
-----------------------------------+------+-----------+----------+---------+--
 dob                               | text |           |          |         | e
 pt_event_recorded_date_time_1     | text |           |          |         | e
 pt_event_reason_code              | text |           |          |         | e
 pt_opr_assigning_facility_1       | text |           |          |         | e
 pt_event_operator_id_num_1        | text |           |          |         | e
 pt_event_operator_last_name       | text |           |          |         | e
 pt_event_operator_first_name      | text |           |          |         | e
 pt_opr_assigning_auth_1           | text |           |          |         | e
 pt_event_facil_namespace_id       | text |           |          |         | e
 pt_event_facil_universal_id       | text |           |          |         | e
 pt_event_facil_universal_id_type  | text |           |          |         | e
 patientid                         | text |           |          |         | e
 pt_address_st_1                   | text |           |          |         | e
 pt_address_city                   | text |           |          |         | e
 pt_address_state_prov             | text |           |          |         | e
 pt_address_zip_postal             | text |           |          |         | e
 pt_address_country                | text |           |          |         | e
 pt_address_type                   | text |           |          |         | e
 pt_county_parish                  | text |           |          |         | e
 pt_last_name                      | text |           |          |         | e
 pt_first_name                     | text |           |          |         | e
 pt_dob                            | text |           |          |         | e
 pt_gender                         | text |           |          |         | e
 pt_visit_number                   | text |           |          |         | e
 pt_loc_poc_1                      | text |           |          |         | e
 pt_loc_room_1                     | text |           |          |         | e
 pt_loc_bed_1                      | text |           |          |         | e
 pt_loc_facility_1                 | text |           |          |         | e
 pt_loc_description                | text |           |          |         | e
 pt_event_occur_date_time          | text |           |          |         | e
```

Phase 1 Workflow

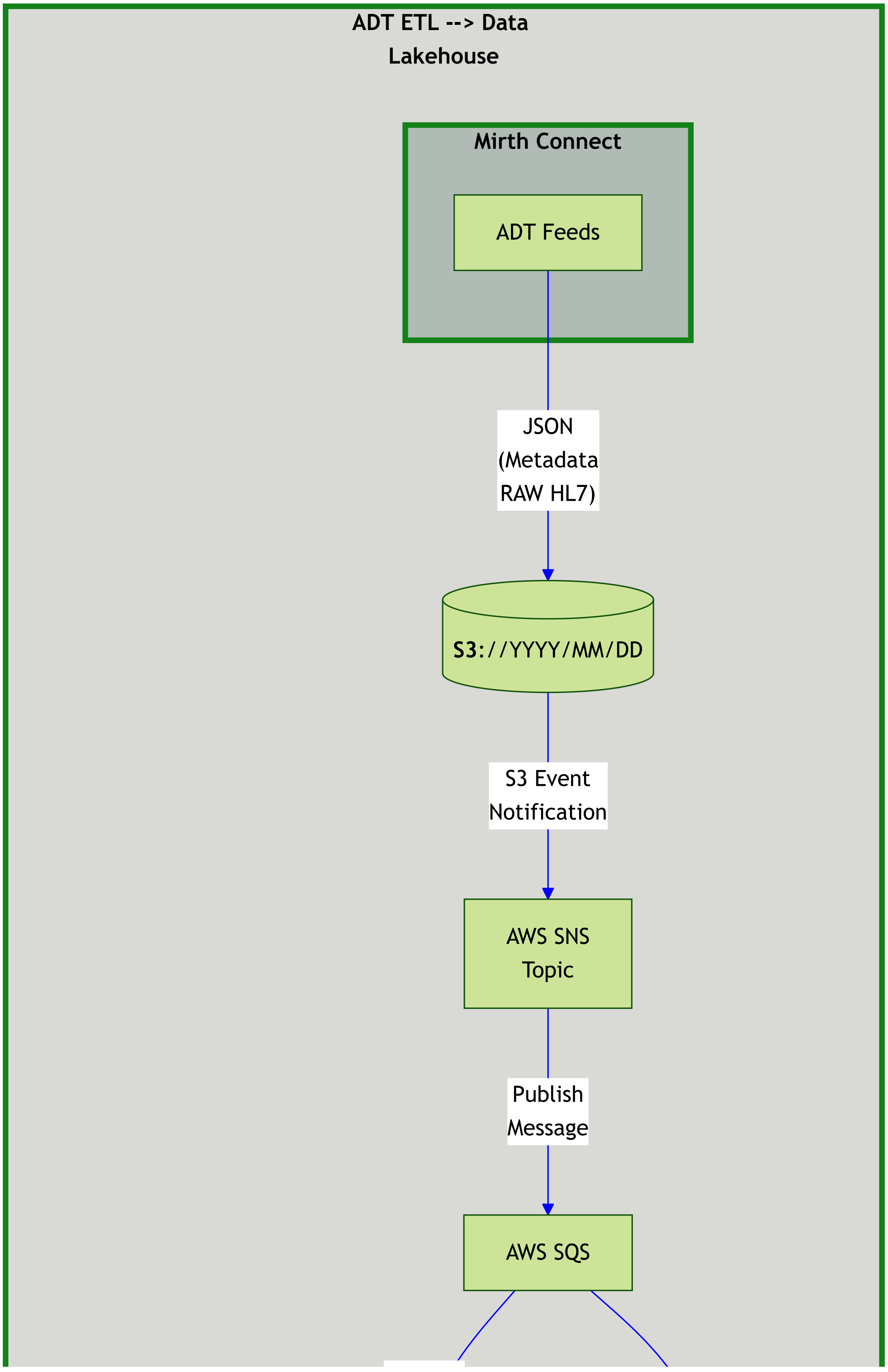
To catch up with over one billion rows (equivalent to three years' worth of raw data) in a MySQL Database and to facilitate the ingestion and processing of a large number of JSON files, I converted the rows into JSON files and stored each file in an S3 bucket as well as in the RedisJSON document store. The S3 bucket serves as permanent storage, while RedisJSON, due to

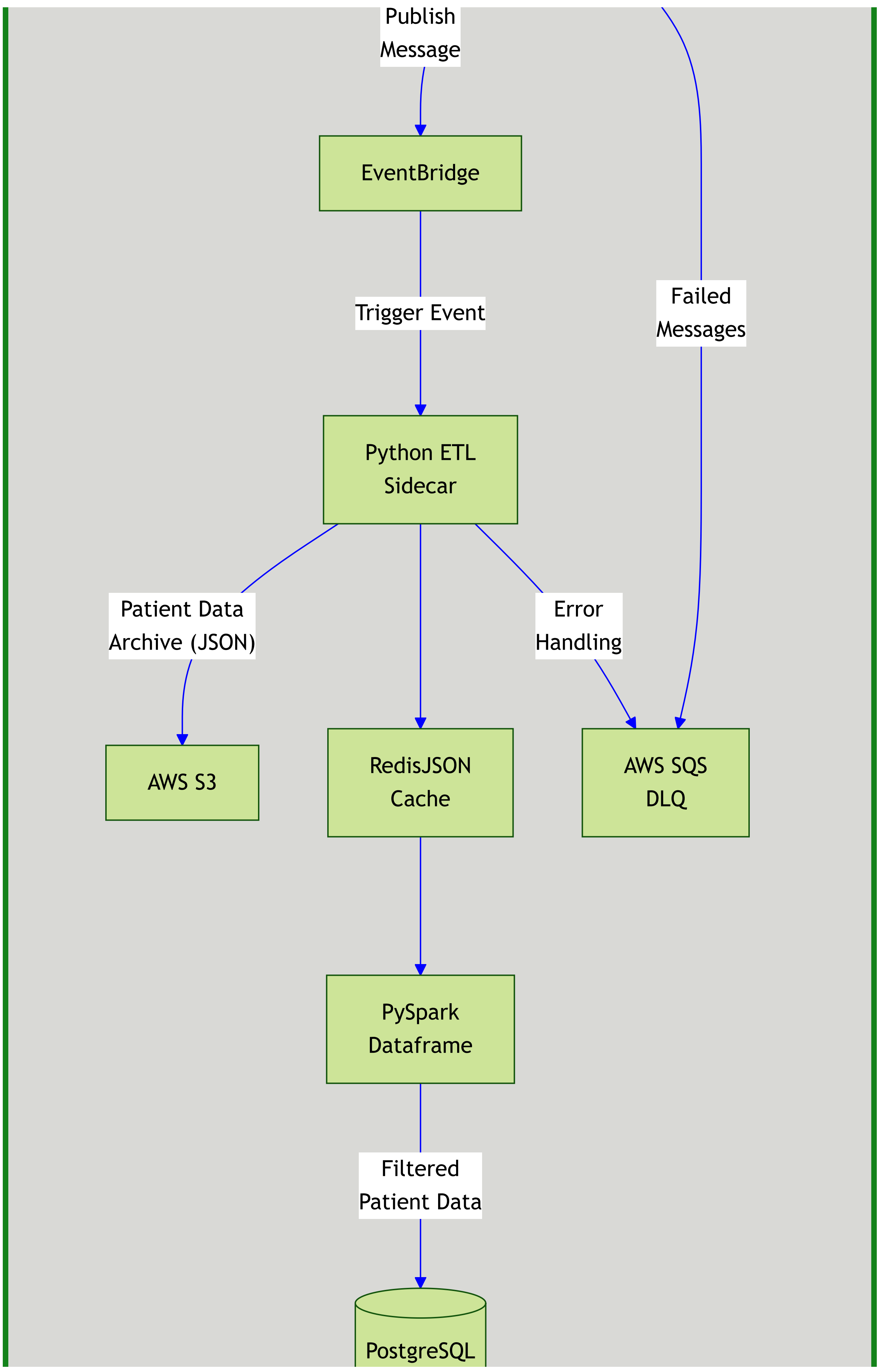
its
significantly faster read capabilities, was utilized for reading and processing tens of thousands of files simultaneously. Despite the initial cost of writing to both S3 and RedisJSON with over one billion rows, the reads, when comparing Python Boto3 with Python Redis, from RedisJSON
were exceptionally fast (`.8` second for `Boto3` , vs `.007` seconds for `PyRedis` for a `3kb` file). The read size was only limited by the available RAM of the EC2 instance.


Phase 1
ADT Feed
Metadata + Raw HL7 Binary BLOB
One time catch-up
MySQL 1+ Billion Rows
Catch Up Python ETL
Metadata + Raw HL7 Binary BLOB
JSON
RedisJSON
Python ETL
Map HL7 Segments and fields
Metadata
PySpark Dataframe
JSON (Metadata+RAW HL7)
Nightly
S3://YYYY/MM/DD
Nightly Python ETL
JSON
JSON
Map HL7 Segments and fields
Metadata
PySpark Dataframe
PostgreSQL
New Patient Referral List Dashboard

Final Implementation

Since this has been my very first experience with processing HL7 data from ADT feeds, now that I have
learned quite a lot, my final implementation would look something like this.


ADT ETL --> Data Lakehouse
Mirth Connect
ADT Feeds
JSON (Metadata RAW HL7)
S3://YYYY/MM/DD
S3 Event Notification
AWS SNS Topic
Publish Message
AWS SQS


Publish Message
EventBridge
Trigger Event
Failed Messages
Python ETL Sidecar
Patient Data Archive (JSON)
Error Handling
AWS S3
RedisJSON Cache
AWS SQS DLQ
PySpark Dataframe
Filtered Patient Data
PostgreSQL