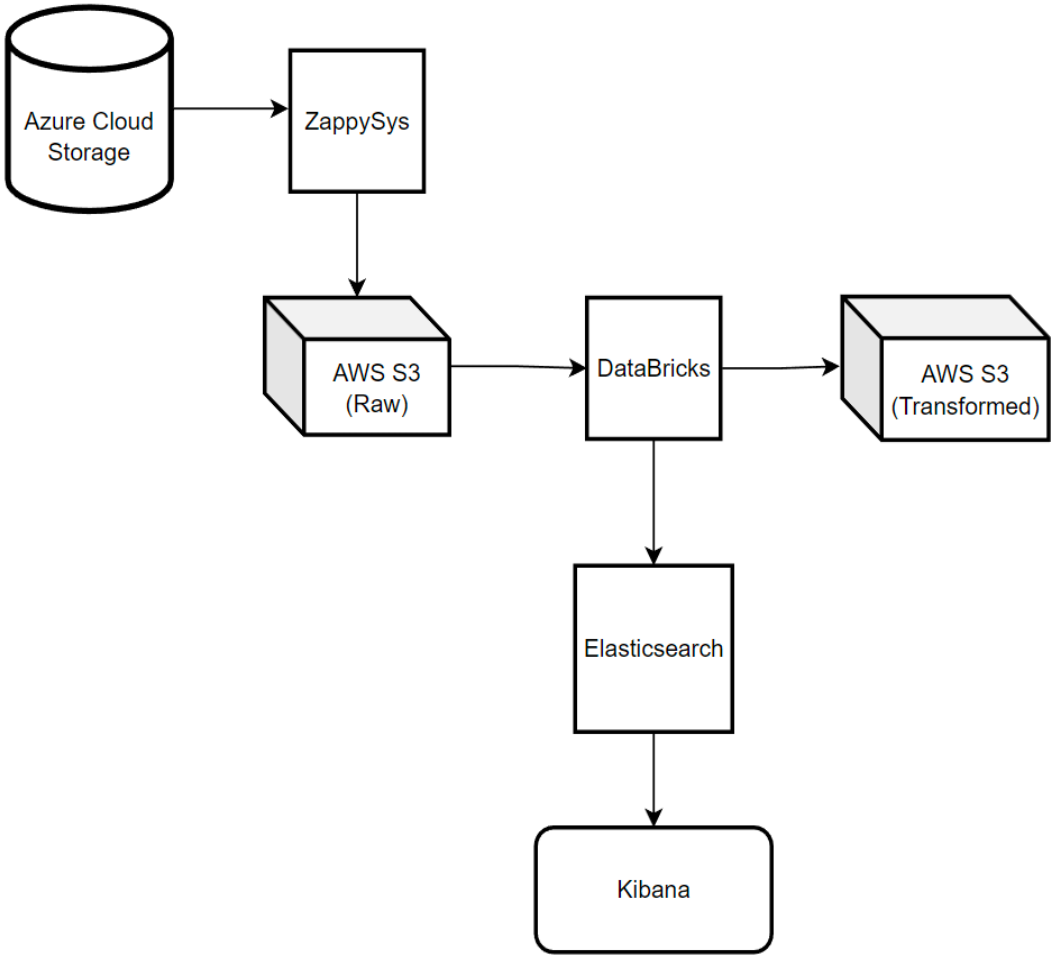
DATA ENGINEERING - TEST PROJECT ONE v 1.0

Bradley Winiarczyk

7/22/2022

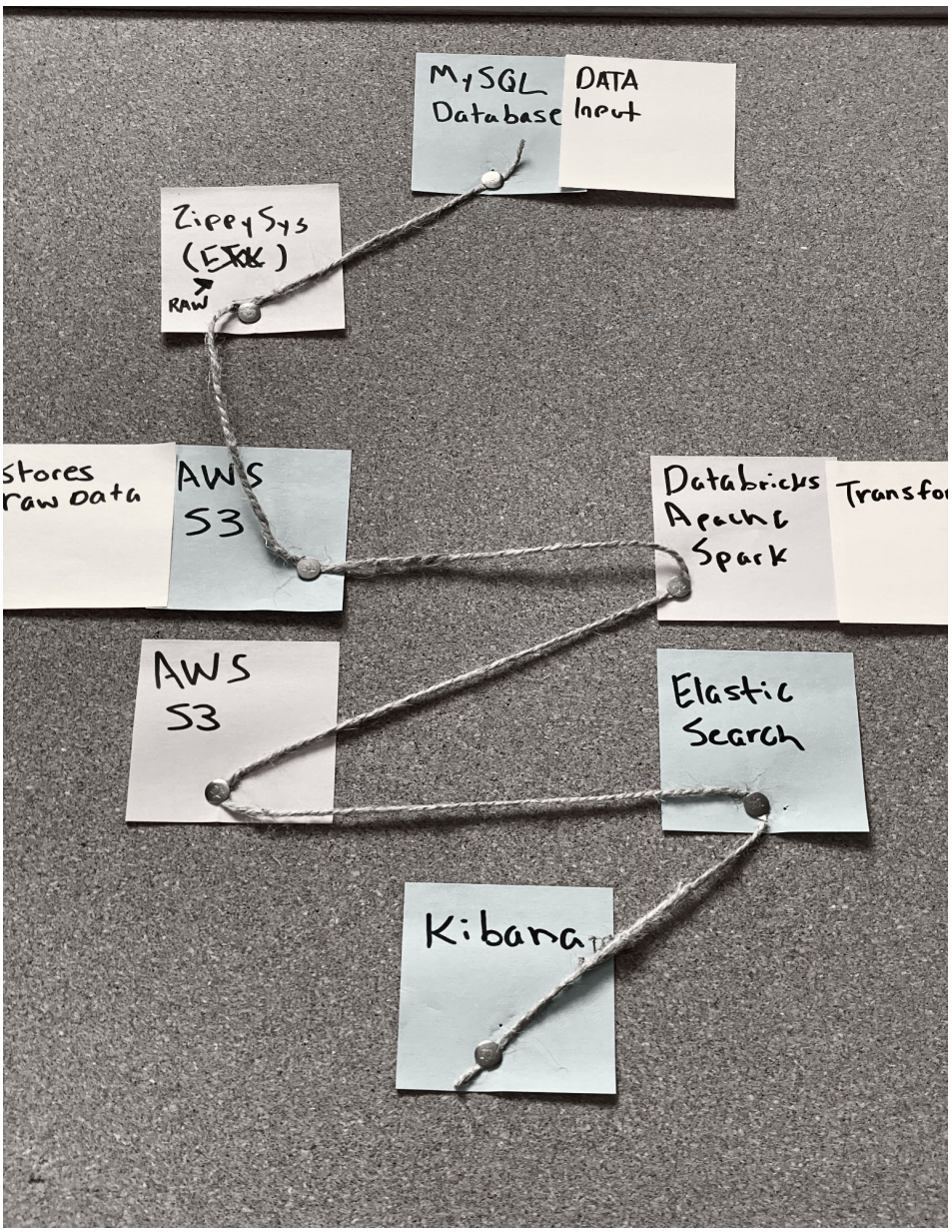
Azure Cloud Storage - Data Source System
ZappySys - Data Integration (SSIS)
Elasticsearch - Document oriented database
AWS S3 - Object Data Storage
DataBricks - Apache Spark (web-based)
Kibana - Data Visualization & Dashboard



Final Schema - 7/22/2022

Original Concept //////

After some consideration, the main and only change from my original concept was switching out a MySQL Database for Azure's cloud storage. This was a decision made to give myself some exposure to data streaming as well as the chance to familiarize myself with industry standards to the best of my ability.



Original concept cork board - 7/18/22

Future Improvements //////

Wraping up this project has allowed me to take a step back and focus in on some core improvements for future development. More specifically, automation. With future renditions of this project, I would like to implement a one click process for any user.

In version 1.0 there is a three main step process in order to be able to visualize the data:

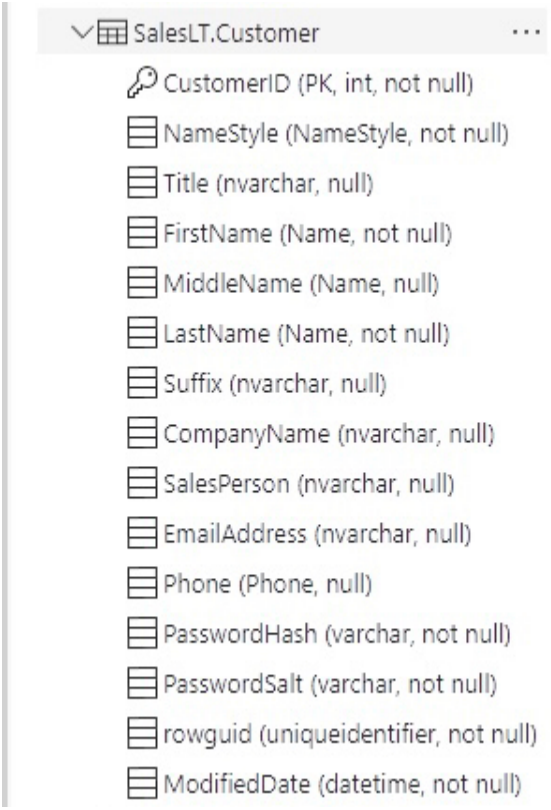
- ➔ Load data from Azure into S3
- ➔ Load data from S3 into Databricks
  - ◆ Data can be transformed and saved to a separate S3 bucket at this step
- ➔ Load data from S3 into Elastic search

From here, the user is able to manage a Kibana dashboard with ease.

Ideally, constructing a program to do all of this in one go would be the most optimal direction moving forward. When ever the Azure data is updated, I would like this process to be updated in Kibana for a fully functional dashboard.

//////

Azure and S3 //////



Starting off, I constructed a **ZippySys** task that extracts the **Customer** data from **Azure** and then sends it over to a ZippySys AWS S3 transfer node. From that node, the data is sent over to my S3 bucket labeled:

berstuk2001cardinaltest

//////

S3 DataBricks Code //////

#User credential files
display(dbutils.fs.ls("FileStore/tables"))

#pyspark functions
from pyspark.sql.functions import *

 # URL processing
import urllib

----

file_type = "csv"

first_row_is_header = "true"

delimiter = ","

aws_keys_df = spark.read.format(file_type)\
.option("header", first_row_is_header)\
.option("sep", delimiter)\
.load("/FileStore/tables/new_user_credentials.csv")

ACCESS_KEY = aws_keys_df.where(col('User name')=='Jerry2001').select('Access key ID').collect()[0]['Access key ID']

SECRET_KEY = aws_keys_df.where(col('User name')=='Jerry2001').select('Secret access key').collect()[0]['Secret access key']

#Encode the secrete key
ENCODED_SECRET_KEY = urllib.parse.quote(string=SECRET_KEY, safe="")

#SOURCE BUCKET MOUNTING

#AWS S3 bucket name

AWS_S3_BUCKET = "berstuk2001cardinaltest"

#Mount name for the bucket

MOUNT_NAME = "/mnt/berstuk2001cardinaltest"

#Source url

SOURCE_URL = "s3n://{0}:{1}@{2}".format(ACCESS_KEY, ENCODED_SECRET_KEY, AWS_S3_BUCKET)

#Mount the drive

dbutils.fs.mount(SOURCE_URL, MOUNT_NAME)

#Reading the CSV

file_location = "/mnt/berstuk2001cardinaltest/Test/test.csv"
file_type = "csv"

#CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

#Applied

df = spark.read.format(file_type)\
.option("inferschema", infer_schema)\
.option("header", first_row_is_header)\
.option("sep", delimiter)\
.load(file_location)

```
df = df.drop('ModifiedDate')
df = df.drop('rowguid')
df = df.drop('PasswordSalt')
df = df.drop('PasswordHash')

#df.printSchema()
```

#Mount S3 for transformed data

#TRANSFORM BUCKET MOUNTING

#AWS S3 bucket name

```
AWS_S3_BUCKET = "jerry2001cardinaltesttransform"
```

#Mount name for the bucket

```
MOUNT_NAME = "/mnt/jerry2001cardinaltesttransform"
```

#Source url

```
SOURCE_URL = "s3n://{0}:{1}@{2}".format(ACCESS_KEY, ENCODED_SECRET_KEY,
AWS_S3_BUCKET)
```

#Mount the drive

```
dbutils.fs.mount(SOURCE_URL, MOUNT_NAME)
```

#Removes if already in bucket
dbutils.fs.rm('/mnt/jerry2001cardinaltesttransform/Transform/transformeddata', True)

#Save to the mounted S3 Transform bucket
df.write.save(f'/mnt/jerry2001cardinaltesttransform/Transform/transformeddata/', format='csv')

#display(dbutils.fs.ls("/mnt/jerry2001cardinaltesttransform/Transform/transformeddata"))

%scala
import org.apache.hadoop.fs._;

//RENAME CSV

val fs = FileSystem.get(sc.hadoopConfiguration);

val file = fs.globStatus(new
Path("/mnt/jerry2001cardinaltesttransform/Transform/transformeddata/part*"))(0).getPath().getN
ame();

fs.rename(new Path("/mnt/jerry2001cardinaltesttransform/Transform/transformeddata/" + file),
new Path("/mnt/jerry2001cardinaltesttransform/Transform/transformeddata/Data.csv"));

display(dbutils.fs.ls("/mnt/jerry2001cardinaltesttransform/Transform/transformeddata"))

#umount S3 buckets
dbutils.fs.unmount("/mnt/berstuk2001cardinaltest")
dbutils.fs.unmount("/mnt/jerry2001cardinaltesttransform")

//////

Elasticsearch DataBricks Code //////

%scala

```
import org.elasticsearch.spark.sql.DefaultSource15
import org.elasticsearch.spark._
import org.elasticsearch.spark.sql._
import sqlContext.implicits._
import spark.implicits._
import org.apache.spark.sql._
```

#pyspark functions

```
from pyspark.sql.functions import *
```

 # URL processing

```
import urllib
```

```
file_type = "csv"

first_row_is_header = "true"

delimiter = ","

aws_keys_df = spark.read.format(file_type)\
.option("header", first_row_is_header)\
.option("sep", delimiter)\
.load("/FileStore/tables/new_user_credentials.csv")
```

```
ACCESS_KEY = aws_keys_df.where(col('User name')=='Jerry2001').select('Access key
ID').collect()[0]['Access key ID']
SECRET_KEY = aws_keys_df.where(col('User name')=='Jerry2001').select('Secret access
key').collect()[0]['Secret access key']
```

#Encode the secrete key

```
ENCODED_SECRET_KEY = urllib.parse.quote(string=SECRET_KEY, safe="")
```

#SOURCE BUCKET MOUNTING

#AWS S3 bucket name

```
AWS_S3_BUCKET = "jerry2001cardinaltesttransform"
```

#Mount name for the bucket

```
MOUNT_NAME = "/mnt/jerry2001cardinaltesttransform"
```

#Source url

```
SOURCE_URL = "s3n://{0}:{1}@{2}".format(ACCESS_KEY, ENCODED_SECRET_KEY,
AWS_S3_BUCKET)
```

#Mount the drive

```
dbutils.fs.mount(SOURCE_URL, MOUNT_NAME)
```

%scala

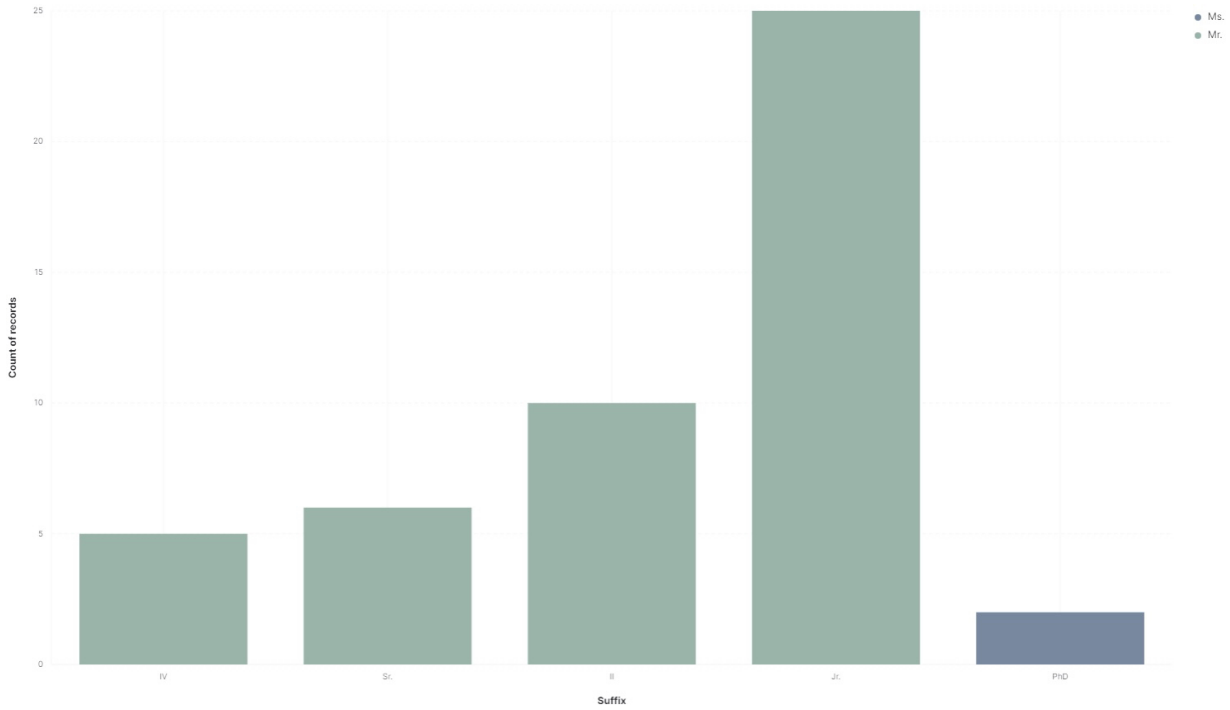
```
val df =
spark.read.format("csv").load("/mnt/jerry2001cardinaltesttransform/Transform/transformeddata/
Data.csv")

val esNodes7 = "https://cardinalprojecttest.es.us-central1.gcp.cloud.es.io"

EsSparkSQL.saveToEs(
    df,
    "customer", // ES index to write to
    Map(
      "es.write.operation" -> "index",
      "es.nodes.wan.only" -> "true",
```

```
      "es.nodes" -> esNodes7,
      "es.net.http.auth.user" -> "xxxxxxxxxx",
      "es.net.http.auth.pass" -> "xxxxxxxxxxxxxx",
      "es.net.ssl" -> true.toString,
      "es.port" -> "443"))
```

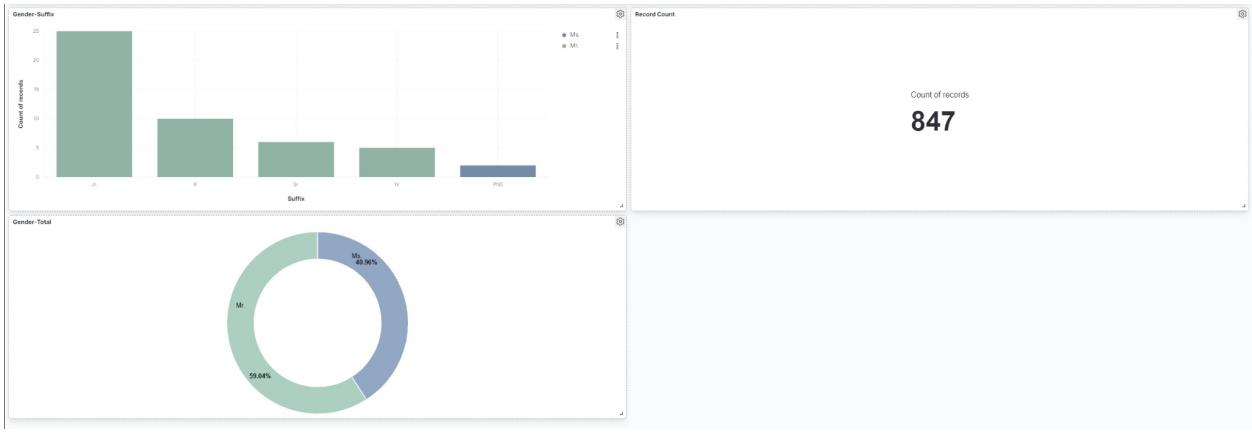
```
dbutils.fs.unmount("/mnt/jerry2001cardinaltesttransform")
```

//////

Simple Kibana Visualizations //////



➔ Displays the most common suffixs with gender of each being shown. This data set has few doctors, and interestingly all of them are female.



➔ Simple Kibana dashboard with similar gender information which could be useful for business intelligence and marketing.