Hatice Hilal AKSOY

LV Product Data Feeds Project Simulation

1. **Proje Nedir?**
2. **Kod Mimarisi Açıklaması**
3. **AWS Mimari Diagramları ve Açıklamaları**
4. **Zor-Kolay Gelen Adımlar**
5. **Kaynaklar**

Proje Nedir?

> *Ottoo'da hali hazırda kullanılan Lead Venture projesinin simüle edilmiş halidir. Bu projenin simülasyonunun kodlanması için AVS sistemlerinde çalışılmamıştır. Sadece lokalde değişiklikler ve kodlamalar yapılmıştır.*

> *Proje, mühendis tarafından sağlanan iki adet CSV türündeki dosyanın belirli başlı işlemlerle temizlendikten sonra ve rollerin de bazılarındaki semboller kaldırıldıktan sonra Parquet olarak kaydedilip, ardından kaydedilen bu Parquet datalarda iki dosya arasındaki farkı bulup bir JSON dosyası halinde yazdırmaktır.*

Kod Mimarisi Açıklaması

```
import pandas as pd #Parquet formatında kayıt için kullandım
import dask.dataframe as dd #Verileri karşılaştırırken kullandım.
import re #temizlemelerde kullandım.
import orjson #işlemek ve saklamak için kullandım _json dan daha verimli imiş chatGPT önerdi ggoogle den teyit ettim_.

# Dosya yollarını belirledim.
file1 = './BasePriceFile_20240607.csv'
file2 = './BasePriceFile_20240608.csv'

# Kolon veri tiplerini tanımlayın
dtype_spec = {
    'punctuated_part_number': str,            # Noktalama işaretli parça numarası
    'current_suggested_retail': float,        # Mevcut önerilen perakende fiyatı
    'wi_availability': int,                   # Wisconsin stok durumu
    'ny_availability': int,                   # New York stok durumu
    'tx_availability': int,                   # Texas stok durumu
    'nv_availability': int,                   # Nevada stok durumu
    'nc_availability': int,                   # North Carolina stok durumu
    'national_availability': int,             # Ulusal stok durumu
    'part_status': str,                       # Parça durumu
    'part_description': str,                  # Parça açıklaması eklemek istedim çünkü Ürünü satıcının ve alıcının tanımasında önemli rolü olabilir.
    'brand_name': str                         # Marka adı ekledim  opsiyonel satıcı ve alıcı için önem teşkil eder.
    # 'upc_code': str                         # UPC kodu (bunu ekleyemedim çünkü lokalimde veriler büyük gelmeye başladı) eklemek istedim çünkü ürün tanımada önemli bir rolü var.
```

```
    #ağırlık vb özellikleri de eklemek istedim)
}

# CSV dosyalarını pandas ile okudum
# https://stackoverflow.com/questions/24251219/pandas-read-csv-low-memory-and-dtype-
options  -71  low memeory=false yapma sebebim oldu
# https://pandas.pydata.org/docs/user_guide/io.html pandas işlemlerini yapmadan önce
dökumanda arattım.
df1 = pd.read_csv(file1, dtype=dtype_spec, low_memory=False)
df2 = pd.read_csv(file2, dtype=dtype_spec, low_memory=False)

# Stage2: Clearing Column Names
# Kolon isimlerini temizleme fonksiyonunu tanımladım.
# https://stackoverflow.com/questions/71273328/change-the-structure-of-column-name
# her kaynakta df var bu ne sordum chatGPT ye.
# https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.columns.html
def clean_column_names(df):
    df.columns = [re.sub(r'\W+', '_', col.lower().strip()) for col in df.columns]  #
Kolon isimlerindeki özel karakterleri alt çizgi ile değiştirin
    return df

# Kolon isimlerini temizleme fonksiyonunu uygulayın
df1 = clean_column_names(df1)
df2 = clean_column_names(df2)

# Stage3: Clearing Data
# Adım 3: punctuated_part_number kolonunda null veya duplicate değerleri temizle
# Clear null or duplicate values in r column with pandas
# Remove Null Values:
# dropna(subset=['r']): This function removes rows where the r column has null
(missing) values.
# Remove Duplicate Values:
# drop_duplicates(subset=['r']): This function removes rows where the r column has
duplicate values.
def clean_data(df, id_column):
    df = df.dropna(subset=[id_column])  # ID kolonu boş olan satırları silin
    df = df.drop_duplicates(subset=[id_column])  # ID kolonu tekrarlayan satırları
silin
    return df

# Temizleme fonksiyonunu uygulamak için ID kolonunu belirleyin
id_column = 'punctuated_part_number' # ID kolonunun Pythonic adı
df1 = clean_data(df1, id_column)
df2 = clean_data(df2, id_column)

# Stage4: Clearing Availability Data
# Stok durumu verilerini temizleme fonksiyonunu tanımladım
#ChatGpt ye yazdırdım.
def enrichment_method(df, quantity_columns):
    for col in quantity_columns:
        df[col] = df[col].astype(str).str.replace('+', '', regex=False)  # '+'
karakterini silin
```

```
    return df

# Stok durumu kolonlarını listeledim. Bir den fazla availability için aynı işlemi
yapacağım
quantity_columns = [
    'wi_availability', 'ny_availability', 'tx_availability',
    'nv_availability', 'nc_availability', 'national_availability'
]

# Temizleme fonksiyonunu stok durumu kolonlarına uygulayın
df1 = enrichment_method(df1, quantity_columns)
df2 = enrichment_method(df2, quantity_columns)

# Stage 5: Saving Data in Parquet Format
# Temizlenmiş veriyi parquet formatında kaydedin
# https://pandas.pydata.org/docs/user_guide/io.html
df1.to_parquet('cleaned_base_price_file_20240607.parquet', engine='pyarrow')
df2.to_parquet('cleaned_base_price_file_20240608.parquet', engine='pyarrow')

# Stage6: Reading Parquet Files Using Dask
# Parquet dosyalarını Dask ile okuyun ve indeks kolonunu belirleyin
# Aslında ben başka yöntem kullandım fakat kaç veri listeleniyor diye logladığımda
kayıp veriler olduğunu düşündüm. Araştırdım Dask veri kaybını azaltıyor ayrı yeten
kodum hızlandı.
df1_parquet =
dd.read_parquet('cleaned_base_price_file_20240607.parquet').set_index('punctuated_part_r

df2_parquet =
dd.read_parquet('cleaned_base_price_file_20240608.parquet').set_index('punctuated_part_r


# Combining DataFrames
# İki DataFrame'i birleştirin
#Çıktıda aldığım ve açık gördüğüm null çıktılarından sonra ekledim. ChatGpt yazdı.
merged_df = dd.merge(df1_parquet, df2_parquet, left_index=True, right_index=True,
suffixes=('_file1', '_file2'), how='outer', indicator=True)  #how= dış birleştirme
yapıyor hiçbir veriyi kayıp etmiyor.
#indicator=True   sadece ilk veri çerçevesinden gelmiş ise  felft_only, sadece ikinci
veri çerçevesinden gelmişse right_only

#Ben tanımladım eksiklerimi- yanlışlarımı chat düzeltti yaparken bir kağıt kalemle tüm
ihtimalleri yazdım.
# Defining the Conditions for Finding Differences
#3 colomn için de fark var mı yok mu belirleme noktası
# Farkları bulmak için koşulları tanımlayın
conditions = [
    merged_df['part_status_file1'] != merged_df['part_status_file2'],  # Parça durumu
farkı
    merged_df['current_suggested_retail_file1'] !=
merged_df['current_suggested_retail_file2']  # Mevcut önerilen perakende fiyatı farkı
]
```

```
# Stok durumu farklarını koşullara ekleyin
for col in quantity_columns:
    conditions.append(merged_df[f'{col}_file1'] != merged_df[f'{col}_file2'])
# Farklılıkların bulunduğu satırları içerir.
# conditions[0] | conditions[1] | dd.concat(conditions[2:] 3 koşumumun da herhangi
birisi (or) sağlanıyorsa onu seçiyorum.
#axis=1 sütun bazında
#any(axis=1) her bir satırda sağlanıp sağlanmadığını kontrol eder.
# loc belirli koşulları sağlayanları seçer. merged_df veri çerçevesinden seçtim.
changed = merged_df.loc[conditions[0] | conditions[1] | dd.concat(conditions[2:],
axis=1).any(axis=1)]

# Adding the Differences
#Değişim olmadan eklediklerim.
differences = {}
for index, row in changed.compute().iterrows():  # Satırları iteratif olarak işlendi
(tekrarlamalı)
    part_number = index
    difference = {
        'punctuated_part_number': part_number,
        'part_description': {
            'file1': row['part_description_file1'],
            'file2': row['part_description_file2']
        },
        'brand_name': {
            'file1': row['brand_name_file1'],
            'file2': row['brand_name_file2']
        },
        '_merge': row['_merge']
    }
    change_detected = False  # Değişiklik tespiti için

    # part_status için if
    if row['part_status_file1'] != row['part_status_file2']:
        if pd.notnull(row['part_status_file1']) or
pd.notnull(row['part_status_file2']):
            difference['part_status'] = {
                'file1': row['part_status_file1'],
                'file2': row['part_status_file2']
            }
            change_detected = True

    # current_suggested_retail için if
    if row['current_suggested_retail_file1'] != row['current_suggested_retail_file2']:
        if pd.notnull(row['current_suggested_retail_file1']) or
pd.notnull(row['current_suggested_retail_file2']):
            difference['current_suggested_retail'] = {
                'file1': row['current_suggested_retail_file1'],
                'file2': row['current_suggested_retail_file2']
            }
            change_detected = True
```

```
    # {Region} Availability için if
    for col in quantity_columns:
        if row[f'{col}_file1'] != row[f'{col}_file2']:
            if pd.notnull(row[f'{col}_file1']) or pd.notnull(row[f'{col}_file2']):
                difference[col] = {
                    'file1': row[f'{col}_file1'],
                    'file2': row[f'{col}_file2']
                }
                change_detected = True

    # Eğer değişiklik tespit edildiyse, farkları kaydedin
    if change_detected:
        differences[part_number] = difference


# Farkları JSON dosyasına yazdım
#orjon kütüphanesi kullandım
with open('differences.json', 'wb') as f:
    f.write(orjson.dumps(differences, option=orjson.OPT_INDENT_2))


# İkinci dosya için farkları aldım
differences_file2_only = {}
for index, row in changed.compute().iterrows():
    part_number = index
    difference = {
        'punctuated_part_number': part_number,
        'part_description': row['part_description_file2'],
        'brand_name': row['brand_name_file2'],
        '_merge': row['_merge'] #sadece file1 den sadece file2 den ya da ikisinde de
mi var bu parametre ile anlayacağım.
    }
    change_detected = False  # Değişiklik tespiti için flag koydum

    # part_status için if ama sadece file2 yi ekle (farklarını)
    if row['part_status_file1'] != row['part_status_file2']:
        if pd.notnull(row['part_status_file1']) or
pd.notnull(row['part_status_file2']):
            difference['part_status'] = row['part_status_file2']
            change_detected = True

    # current_suggested_retail için if ama sadece file2 yi ekle (farklarını)
    if row['current_suggested_retail_file1'] != row['current_suggested_retail_file2']:
        if pd.notnull(row['current_suggested_retail_file1']) or
pd.notnull(row['current_suggested_retail_file2']):
            difference['current_suggested_retail'] =
row['current_suggested_retail_file2']
            change_detected = True

    # {Region} Availability için if ama sadece file2 yi ekle (farklarını)
    for col in quantity_columns:
        if row[f'{col}_file1'] != row[f'{col}_file2']:
```

```
            if pd.notnull(row[f'{col}_file1']) or pd.notnull(row[f'{col}_file2']):
                difference[col] = row[f'{col}_file2']
                change_detected = True

    # Eğer değişiklik tespit edildiyse, farkları kaydedin
    if change_detected:
        differences_file2_only[part_number] = difference

# Writing to JSON File
# İkinci dosya için farkları JSON dosyasına yazın
with open('differences_file2_only.json', 'wb') as f:
    f.write(orjson.dumps(differences_file2_only, option=orjson.OPT_INDENT_2))

# Counting the Differences
# Farkları sayın
part_status_diff = 0
current_suggested_retail_diff = 0
quantity_diffs = {
    'wi_availability': 0,
    'ny_availability': 0,
    'tx_availability': 0,
    'nv_availability': 0,
    'nc_availability': 0,
    'national_availability': 0
}

# Farkları sayma işlemi
for diff in differences.values():
    if 'part_status' in diff:
        part_status_diff += 1
    if 'current_suggested_retail' in diff:
        current_suggested_retail_diff += 1
    for col in quantity_diffs.keys():
        if col in diff:
            quantity_diffs[col] += 1

# Toplam değişiklik sayısını hesaplayın
num_changes = part_status_diff + current_suggested_retail_diff +
sum(quantity_diffs.values())
print(f"Total number of changes: {num_changes}")

print(f"part_status differences: {part_status_diff}")
print(f"current_suggested_retail differences: {current_suggested_retail_diff}")
for col, diff_count in quantity_diffs.items():
    print(f"{col} differences {diff_count}")
```

1. Kodlama yaparken genel olarak ilk başta kendim mantığını oluşturdum.
2. Kullanabileceğim kütüphaneler ve kodları da araştırarak ilerlemeye çalıştım.

AWS Mimari Diagramları ve Açıklamaları

Kullandığım AWS Servisleri

AWS Servis	LINK
AWS Lambda Fonksiyon	https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
AWS S3 Bucket	https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
AWS SQS	https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperG
AWS SNS	https://docs.aws.amazon.com/sns/latest/dg/welcome.html
AWS EventBridge	https://docs.aws.amazon.com/scheduler/latest/UserGuide/what-is-schedul
AWS Step Fonksiyon	https://docs.aws.amazon.com/codepipeline/latest/userguide/action-refer StepFunctions.html
AWS CloudWatch	https://aws.amazon.com/tr/cloudwatch/
AWS Amazon API	https://aws.amazon.com/tr/api-gateway/

FİRST Diagram

1. İlk olarak kullanılacak Csv file yüklendi bir s3 bucket a.
2. Ardından lambda kodum eventbridge tarafından tetiklendi ve gerekli izinler doğrultusunda çalıştı.
3. Ardından Historical Data SQS den alındı ve json s3 bucket a gönderildi.
4. Gerekli Permission ve izinler verildikten sonra s3 den URL alındı.
5. SNS ile kullanıcıya Url ulaştı.
6. us-west-2 region da gerçekleşti olaylar.

SECOND Diagram

1. İlk olarak kullanılacak Csv file yüklendi bir s3 bucket a.
2. Ardından step fonksiyonum eventbridge tarafından tetiklendi ve sırası ile lambda larım çalıştı.
3. Birinci lambda da parquet dataya kayıt edildi, ikinci lambda da farklar alındı ve json formatında s3 bucket a gönderildi.
4. Step fonksiyonda oluşan loglar CloudWatch da depolandı (Historical Data)
5. s3 bucket a kayıt edilen datalar API Gatway Resources aracılığı ile AWS lambda yı tetikleyip bir url oluştu
6. SNS ile kullanıcıya Url ulaştı.
7. us-west-2 region da gerçekleşti olaylar.

LİNKS

Code Flowchart

- https://lucid.app/lucidchart/e0880c89-85dd-4393-9f66-57a61aa47a62/edit?invitationId=inv_005643d9-bb7a-43e8-bc13-e5d938d83f2e&page=0_0#

AWS Mimari Diagrams

- https://lucid.app/lucidchart/aad399aa-9e8b-4888-b2d6-ee00e72e71bb/edit?invitationId=inv_32594e22-7236-4c2a-b3a4-04aaff331435&page=0_0#
- https://lucid.app/lucidchart/e2a31b26-90b6-4914-bd63-4de33d9ddb2b/edit?viewport_loc=-1091%2C-462%2C2742%2C1274%2C0_0&invitationId=inv_854aabe7-e2e8-45bc-9c8f-a33e48506b75

Hatice Hilal AKSOY DATA FELLOW