Projeto-Dom-Manolo

January 12, 2024

1 Projeto Dom Manolo

1.1 Sobre

Este projeto voluntário foi idealizado com o objetivo de colocar em prática conhecimentos adquiridos e obter experiência de mercado. A proposta para o cliente foi de entregar uma análise de dados mais unificada e personalizada, que fosse de encontro às suas necessidades. Este documento apresenta a parte técnica, utilizada para realizar o processo de ETL, desenvolvido em PySpark e também a análise dos dados, feita no PowerBi, com DAX.

1.2 PySpark

1.2.1 Importando Libs

[ ]:
```
from pyspark.sql import SparkSession
from functools import reduce
from pyspark.sql.functions import to_timestamp, date_format, col, initcap,␣
 ↪lower, translate, regexp_replace, when, to_date, month, year, sum
from pyspark.sql.types import FloatType
import os
```

###Instanciando Spark Context

[ ]:
```
spark = SparkSession.builder.appName("Dom-Manolo").getOrCreate()
```

1.2.2 Mapeando Arquivos

[ ]:
```
data_raw = "data/raw/csv/"
files_prefix = "AppBarber_Relatorio_de_Agendamentos_"
all_files = [file for file in os.listdir(data_raw) if file.
 ↪startswith(files_prefix)]
```

1.2.3 Lendo Arquivos CSV

[ ]:
```
df = spark.read.csv(os.path.join(data_raw, all_files[1]))
```

1.2.4 Criando DataFrames

[ ]:
```
data_frames = [spark.read.csv(os.path.join(data_raw, file), header=True) for␣
 ↪file in all_files]
```

[ ]:
```
def union_all(data_frames):
    return reduce(lambda df1, df2: df1.union(df2), data_frames)
```

Criando DataFrame Único Inicial

[ ]:
```
final_data_frame = union_all(data_frames)
```

1.2.5 Salvando Arquivo Único na Camada Raw

[ ]:
```
final_data_frame.coalesce(1).write \
    .option("header", True) \
    .option("delimiter", ",") \
    .mode("overwrite") \
    .csv(data_raw + "Relatorio_de_Agendamentos_2018_2023")
```

1.2.6 Renomeando Colunas

[ ]:
```
collums_names = ["servico",
           "cliente",
           "telefone",
           "profissional",
           "data_e_hora_atendimento",
           "valor",
           "forma_pagamento",
           "pontos",
           "status",
           "quem_cadastrou",
           "data_e_hora_cadastro",
           "comanda",
           "tipo",
           "observacao"]
```

[ ]:
```
count = 0
final_data_frame_1 = final_data_frame
for old_column in final_data_frame.columns:
    final_data_frame_1 = final_data_frame_1.withColumnRenamed(old_column,␣
 ↪collums_names[count])
    count += 1
```

1.2.7 Ajustando colunas de data

[ ]:
```
date_columns = ["data_e_hora_atendimento", "data_e_hora_cadastro"]

for column in date_columns:
    final_data_frame_2 = (
        final_data_frame_1
        .withColumn(column + "_timestamp",␣
 ↪to_timestamp("data_e_hora_atendimento", "dd/MM/yyyy HH:mm"))
        .withColumn("data_atendimento", date_format(column + "_timestamp", "dd/
 ↪MM/yyyy"))
        .withColumn("hora_atendimento", date_format(column + "_timestamp", "HH:
 ↪mm"))
        .drop(column + "_timestamp", "data_e_hora_atendimento")
    )
final_data_frame_2 = final_data_frame_2.withColumn("data_atendimento",␣
 ↪to_date("data_atendimento", "dd/MM/yyyy"))
```

[ ]:
```
final_data_frame_3 = (
    final_data_frame_2
    .withColumn("data_e_hora_cadastro_timestamp",␣
 ↪to_timestamp("data_e_hora_cadastro", "dd/MM/yyyy HH:mm"))
    .withColumn("data_cadastro", date_format("data_e_hora_cadastro_timestamp",␣
 ↪"dd/MM/yyyy"))
    .withColumn("hora_cadastro", date_format("data_e_hora_cadastro_timestamp",␣
 ↪"HH:mm"))
    .drop("data_e_hora_cadastro_timestamp", "data_e_hora_cadastro")
)

final_data_frame_3 = final_data_frame_3.withColumn("data_cadastro",␣
 ↪to_date("data_cadastro", "dd/MM/yyyy"))
```

1.2.8 Tratando de Strings

[ ]:
```
columns_to_initcap = ["servico", "cliente", "profissional", "quem_cadastrou"]

for column in columns_to_initcap:
    final_data_frame_3 = final_data_frame_3.withColumn(column,␣
 ↪initcap(col(column)))
    final_data_frame_3 = final_data_frame_3.withColumn(column,␣
 ↪regexp_replace(translate(col(column), "çáéíóúâêîôûãõàèìòùäëïöü",␣
 ↪"caeiouaeiouaoaeiouaeiou"), "[^a-zA-Z0-9 ]", ""))
```

[ ]:
```
columns_to_lower = ["observacao"]

for column in columns_to_lower:
```

```
    final_data_frame_3 = final_data_frame_3.withColumn(column,␣
 ↪lower(col(column)))
```

[ ]:
```
columns_to_deemphasize = ["forma_pagamento"]

for column in columns_to_deemphasize:
    final_data_frame_3 = final_data_frame_3.withColumn(column,␣
 ↪regexp_replace(translate(col(column), "áéíóúâêîôûãõàèìòùäëïöü",␣
 ↪"aeiouaeiouaoaeiouaeiou"), "[^a-zA-Z0-9 ]", ""))
```

[ ]:
```
final_data_frame_3 = final_data_frame_3.withColumn("valor",␣
 ↪regexp_replace("valor", ",", "."))
final_data_frame_3 = final_data_frame_3.withColumn("valor", col("valor").
 ↪cast(FloatType()))
```

1.2.9 Criando apenas um profissional para o mesmo profissional com nomes diferentes

[ ]:
```
nomes = {
    "Chave nome omitida":"Valor nome omitido",
    "Chave nome omitida":"Valor nome omitido"
}

final_data_frame_4 = final_data_frame_3

for nome, nome_corrigido in nomes.items():
    final_data_frame_4 = final_data_frame_4.withColumn("profissional",␣
 ↪when(col("profissional").startswith(nome), nome_corrigido).
 ↪otherwise(col("profissional")))
```

1.2.10 Mascarando nome dos Profissionais (Opcional)

[ ]:
```
lista_profissionais = final_data_frame_4.select(col("profissional")).distinct().
 ↪rdd.map(lambda x: x[0]).collect()

c = 1
for nome in lista_profissionais:
    final_data_frame_4 = final_data_frame_4.withColumn("profissional",␣
 ↪when(col("profissional") == nome, f"Barbeiro_{c}").
 ↪otherwise(col("profissional")))
    c += 1
```

1.2.11 Criando coluna Comissão

```
[ ]: final_data_frame_5 = final_data_frame_4
     final_data_frame_5 = final_data_frame_5.withColumn("comissao", (col("valor") *␣
      ↪0.40))
```

1.2.12 Extraindo a data do último relatório

```
[ ]: ultima_data = final_data_frame_5.agg({"data_atendimento": "max"}).
      ↪collect()[0][0]
     print(ultima_data)
```

1.2.13 Salvando Arquivos na Camada Trusted

```
[ ]: data_trusted = "data/trusted/csv"

     final_data_frame_5.coalesce(1).write \
         .option("header", True) \
         .option("delimiter", ",") \
         .mode("overwrite") \
         .csv(data_trusted + "Trusted_Relatorio_de_Agendamentos_2018_2023_v3")
```

1.3 DAX utilizado para construção de métricas de fluxo de clientes.

1.3.1 Clientes antigos

```
clientes_antigos =
VAR _data_contexto = MIN(DomManolo[data_atendimento])
VAR _anteriores =
    CALCULATETABLE(
        VALUES(DomManolo[cliente]),
        FILTER(
            ALL(DomManolo),
        DomManolo[data_atendimento] < _data_contexto
        )
)
RETURN
    COUNTROWS(_anteriores)
```

1.3.2 Clientes novos

```
clientes_novos =
VAR _clientes_atuais = VALUES(DomManolo[cliente])
VAR _data_contexto = MIN(DomManolo[data_atendimento]) - 1
VAR _clientes_antigos =
    CALCULATETABLE(
        VALUES(DomManolo[cliente]),
        FILTER(
```

```
            ALL(DomManolo),
        DomManolo[data_atendimento] <= _data_contexto
        )
)
VAR _novos = EXCEPT(_clientes_atuais, _clientes_antigos)
RETURN
    COUNTROWS(_novos)
```

1.3.3 Clientes perdidos (últimos dois meses antes do contexto)

```
clientes_perdidos =
VAR _data_contexto = MIN(DomManolo[data_atendimento])-1
VAR _clientes_perdidos =
    CALCULATETABLE(
        VALUES(DomManolo[cliente]),
        FILTER(
            ALL(DomManolo),
            DomManolo[data_atendimento]<=_data_contexto &&
            DATEDIFF(_data_contexto, DomManolo[ultima_data],MONTH)>2
        )
    )
RETURN
    COUNTROWS(_clientes_perdidos)
```