databricks MVPMontoya

(https://databricks.com)

MVP Trabalho Final

Aluno: Rodolfo Montoya

Disciplina: Engenharia de dados

Data de entrega: 04 de Julho de 2024**

Objetivo

Objetivo deste MVP, é avaliar a capacidade estrutural das pontes nas estradas dos Estados Unidos, verificando se existem profissionais suficientes para realizar trabalhos de inspeção e avaliando orçamentos necessários para realizar inspeções, projetos e manutenções. Nossas perguntas que queremos responder seriam: • Risco estrutural das pontes? • Frequência necessária de inspeção? • Quantidades de oportunidades e profissionais? • Necessidade de investimento?

Plataforma

Direcionamos a Plataforma Databricks. Sendo que dentro do Microsoft Azure, temos esta ferramenta de Databricks e toda a arquitetura de dados será realizada na nuvem do Azure. Detalhamento A escolha de nossos dados foi obtida de pesquisas de informações internas, raspagem de dados do site da ASCE, classificados americanos, assim como do site kaggle. Dados utilizados: • Data.NBI.csv obtido do kaggle - https://www.kaggle.com/datasets/broach/build-bridges-not-walls (https://www.kaggle.com/datasets/broach/build-bridges-not-walls); • mtguide.pdf, obtido do site da internet https://www.fhwa.dot.gov/bridge/mtguide.pdf (https://www.fhwa.dot.gov/bridge/mtguide.pdf) – federal highway administration, deste arquivo foram raspadas diferentes tabelas para alimentação de nossos dados principais. Aqui foram raspadas diferentes tabelas.

Coleta, Modelagem e Carga

Uma vez definido o conjunto de dados, devemos coletar e armazená-los na nuvem, este processo de armazenagem segue as disposições de uma arquitetura para ETL, desenvolvendo assim está no Azure, utilizando a carga dos dados para o Data Warehouse/Data Lake. Utilizamos pipelines de ETL (Extração, Transformação e Carga) na Azure e Databricks. Criada conta de armazenamento com três camadas. Criado o pipeline. E criado nosso cluster com nosso notebook Na camada bronze foi colocado nossos dados brutos E posteriormente com o código chegamos até nossa camada silver com dados já previamente tratados A camada gold foi mais o cálculo e tratamento final dos dados para avaliação de risco em estruturas e disponibilizados para nossos clientes.

Análise

Qualidade de dados Os atributos encontrados tiveram alguns dados desnecessários para nossa análise, não é uma boa prática alterar a camada bruta, por isso que o tratamento dos dados é feito na silver, aqui deletamos dados que não seriam úteis para nossos questionamentos. Nas oportunidades de trabalho foi mais complexo o tratamento porque existem muitas funções e precisamos de avaliar as que são úteis para nosso questionamento.

Solução do problema

Montagem das bases das camadas bronze, silver e gold

```
dbutils.fs.unmount('/mnt/azuredatabricksmvp2024/bronze')
dbutils.fs.mount(
  source = 'wasbs://bronze@azuredatabricksmvp2024.blob.core.windows.net/',
  mount_point = '/mnt/azuredatabricksmvp2024/bronze',
  extra_configs = {'fs.azure.account.key.azuredatabricksmvp2024.blob.core.windows.net': 'aKryGss0+fXjV8YXg6uRiDl4p2ZDAifGTH/7fVGGkonQMLzmyldgy8OvUu7EPkSzlod0U0kCxcvx+AStFta46Q=='}
)
```

```
dbutils.fs.unmount('/mnt/azuredatabricksmvp2024/silver')

dbutils.fs.mount(
  source = 'wasbs://silver@azuredatabricksmvp2024.blob.core.windows.net/',
  mount_point = '/mnt/azuredatabricksmvp2024/silver',
  extra_configs = {'fs.azure.account.key.azuredatabricksmvp2024.blob.core.windows.net': 'aKryGss0+fXjV8YXg6uRiDl4p2ZDAifGTH/7fVGGkonQMLzmyldgy8OvUu7EPkSzlod0U0kCxcvx+AStFta46Q=='}
)
```

```
dbutils.fs.unmount('/mnt/azuredatabricksmvp2024/gold')
dbutils.fs.mount(
  source = 'wasbs://gold@azuredatabricksmvp2024.blob.core.windows.net/',
  mount_point = '/mnt/azuredatabricksmvp2024/gold',
  extra_configs = {'fs.azure.account.key.azuredatabricksmvp2024.blob.core.windows.net': 'aKryGss0+fXjV8YXg6uRiDl4p2ZDAifGTH/7fVGGkonQMLzmyldgy8OvUu7EPkSzlod0U0kCxcvx+AStFta46Q=='}
)
```

Visualizando os dados que tenho na minha camada bronze, feito o carregamento com tabelas que serão utilizadas na análise

```
#criar database
spark.sql('CREATE DATABASE IF NOT EXISTS bridge')
```

```
#ler camada bronze
file_location = 'dbfs:/mnt/azuredatabricksmvp2024/bronze/data_NBI.csv'
file_type = 'csv'
infer_schema = 'true'
first_row_is_header = 'true'
delimiter = ','
df_bridge_bronze = spark.read.format(file_type).option('inferSchema', infer_schema).option('header', first_row_is_header).option('sep', delimiter).load(file_location)
display(df_bridge_bronze)
```

OBSERVEI QUE NO MOMENTO QUE ANALIZANDO OS DADOS TINHA MUITO ERROS E SEM CABEÇALHOS VOU FAZER UM TRATAMENTO INICIAL DO BRONZE, CRIANDO UMA TABELA MAIS ESTRUTURADA

```
field_map = {
    'ITEM1':  { 'State Code'},
    'ITEM8':  { 'Structure Number'},
    'ITEM5':  { 'Inventory Route'},
    'ITEM5A': { 'Record Type'},
    'ITEM5B': { 'Route Signing Prefix'},
    'ITEM5C': { 'Designated Level of Service'},
    'ITEM5D': { 'Route Number'},
    'ITEM5E': { 'Directional Suffix'},
    'ITEM2':  { 'Highway Agency District'},
    'ITEM3':  { 'County (Parish) Code'},
    'ITEM4':  { 'Place Code'},
    'ITEM6':  { 'Features Intersected'},
    'ITEM6A': { 'Features Intersected'},
    'ITEM6B': { 'Critical Facility Indicator'},
    'ITEM7':  { 'Facility Carried By Structure'},
    'ITEM9':  { 'Location'},
    'ITEM10': { 'Inventory Rte, Min Vert Clearance'},
    'ITEM11': { 'Kilometerpoint'},
    'ITEM12': { 'Base Highway Network'},
    'ITEM13': { 'Inventory Route'},
    'ITEM13A': { 'LRS Inventory Route'},
    'ITEM13B': { 'Subroute Number'},
    'ITEM16': { 'Latitude'},
    'ITEM17': { 'Longitude'},
    'ITEM19': { 'Bypass/Detour Length'},
    'ITEM20': { 'Toll'},
    'ITEM21': { 'Maintenance Responsibility'},
    'ITEM22': { 'Owner'},
    'ITEM26': { 'Functional Class Of Inventory Rte.'},
    'ITEM27': { 'Year Built'},
    'ITEM28': { 'Lanes On/Under Structure'},
    'ITEM28A': { 'Lanes On Structure'},
    'ITEM28B': { 'Lanes Under Structure'},
    'ITEM29': { 'Average Daily Traffic'},
    'ITEM30': { 'Year Of Average Daily Traffic'},
    'ITEM31': { 'Design Load'},
    'ITEM32': { 'Approach Roadway Width'},
    'ITEM33': { 'Bridge Median'},
    'ITEM34': { 'Skew'},
    'ITEM35': { 'Structure Flared'},
    'ITEM36': { 'Traffic Safety Features'},
    'ITEM36A': { 'Bridge Railings'},
    'ITEM36B': { 'Transitions'},
    'ITEM36C': { 'Approach Guardrail'},
    'ITEM36D': { 'Approach Guardrail Ends'},
    'ITEM37': { 'Historical significance'},
    'ITEM38': { 'Navigation Control'},
```

```
'ITEM39': { 'Navigation Vertical Clearance'},
'ITEM40': { 'Navigation Horizontal Clearance'},
'ITEM41': { 'Structure Open/Posted/Closed'},
'ITEM42': { 'Type Of Service'},
'ITEM42A': { 'Type of Service On Bridge'},
'ITEM42B': { 'Type of Service Under Bridge'},
'ITEM43':  { 'Structure Type, Main'},
'ITEM43A': { 'Kind of Material/Design'},
'ITEM43B': { 'Type of Design/Construction'},
'ITEM44':  { 'Structure Type, Approach Spans'},
'ITEM44A': { 'Kind of Material/Design'},
'ITEM44B': { 'Type of Design/Construction'},
'ITEM45':  { 'Number Of Spans In Main Unit'},
'ITEM46':  { 'Number Of Approach Spans'},
'ITEM47':  { 'Inventory Rte Total Horz Clearance'},
'ITEM48':  { 'Length Of Maximum Span'},
'ITEM49':  { 'Structure Length'},
'ITEM50':  { 'Curb/Sidewalk Widths'},
'ITEM50A': { 'Left Curb/Sidewalk Width'},
'ITEM50B': { 'Right Curb/Sidewalk Width'},
'ITEM51':  { 'Bridge Roadway Width Curb-To-Curb'},
'ITEM52':  { 'Deck Width, Out-To-Out'},
'ITEM53':  { 'Min Vert Clear Over Bridge Roadway'},
'ITEM54':  { 'Minimum Vertical Underclearance'},
'ITEM54A': { 'Reference Feature'},
'ITEM54B': { 'Minimum Vertical Underclearance'},
'ITEM55':  { 'Min Lateral Underclear On Right'},
'ITEM55A': { 'Reference Feature'},
'ITEM55B': { 'Minimum Lateral Underclearance'},
'ITEM56':  { 'Min Lateral Underclear On Left'},
'ITEM58':  { 'Deck'},
'ITEM59':  { 'Superstructure'},
'ITEM60':  { 'Substructure'},
'ITEM61':  { 'Channel/Channel Protection'},
'ITEM62':  { 'Culverts'},
'ITEM63':  { 'Method Used To Determine Operating Rating'},
'ITEM64':  { 'Operating Rating'},
'ITEM65':  { 'Method Used To Determine Inventory Rating'},
'ITEM66':  { 'Inventory Rating'},
'ITEM67':  { 'Structural Evaluation'},
'ITEM68':  { 'Deck Geometry'},
'ITEM69':  { 'Underclear, Vertical & Horizontal'},
'ITEM70':  { 'Bridge Posting'},
'ITEM71':  { 'Waterway Adequacy'},
'ITEM72':  { 'Approach Roadway Alignment'},
'ITEM75':  { 'Type of Work'},
'ITEM75A': { 'Type of Work Proposed'},
'ITEM75B': { 'Work Done By'},
'ITEM76':  { 'Length Of Structure Improvement'}
```

```
    'ITEM76' :  { 'Length Of Structure Improvement' },
    'ITEM90':  { 'Inspection Date'},
    'ITEM91':  { 'Designated Inspection Frequency'},
    'ITEM92':  { 'Critical Feature Inspection'},
    'ITEM92A': { 'Fracture Critical Details'},
    'ITEM92B': { 'Underwater Inspection'},
    'ITEM92C': { 'Other Special Inspection'},
    'ITEM93':  { 'Critical Feature Inspection Dates'},
    'ITEM93A': { 'Fracture Critical Details Date'},
    'ITEM93B': { 'Underwater Inspection Date'},
    'ITEM93C': { 'Other Special Inspection Date'},
    'ITEM94':  { 'Bridge Improvement Cost'},
    'ITEM95':  { 'Roadway Improvement Cost'},
    'ITEM96':  { 'Total Project Cost'},
    'ITEM97':  { 'Year Of Improvement Cost Estimate'},
    'ITEM98':  { 'Border Bridge'},
    'ITEM98A': { 'Neighboring State Code'},
    'ITEM98B': { 'Percent Responsibility'},
    'ITEM99':  { 'Border Bridge Structure Number'},
    'ITEM100': { 'STRAHNET Highway Designation'},
    'ITEM101': { 'Parallel Structure Designation'},
    'ITEM102': { 'Direction Of Traffic'},
    'ITEM103': { 'Temporary Structure Designation'},
    'ITEM104': { 'Highway System Of Inventory Route'},
    'ITEM105': { 'Federal Lands Highways'},
    'ITEM106': { 'Year Reconstructed'},
    'ITEM107': { 'Deck Structure Type'},
    'ITEM108': { 'Wearing Surface/Protective System'},
    'ITEM108A': { 'Type of Wearing Surface'},
    'ITEM108B': { 'Type of Membrane'},
    'ITEM108C': { 'Deck Protection'},
    'ITEM109': { 'AVERAGE DAILY TRUCK TRAFFIC'},
    'ITEM110': { 'DESIGNATED NATIONAL NETWORK'},
    'ITEM111': { 'PIER/ABUTMENT PROTECTION'},
    'ITEM112': { 'NBIS BRIDGE LENGTH'},
    'ITEM113': { 'SCOUR CRITICAL BRIDGES'},
    'ITEM114': {'FUTURE AVERAGE DAILY TRAFFIC'},
    'ITEM115': { 'YEAR OF FUTURE AVG DAILY TRAFFIC'},
    'ITEM116': { 'MINIMUM NAVIGATION VERTICAL CLEARANCE VERTICAL LIFT BRIDGE'}
}

limpeza=['ITEM1','ITEM8','ITEM5A','ITEM5B','ITEM5C','ITEM5D','ITEM5E','ITEM2','ITEM3','ITEM4','ITEM6A','ITEM6B','ITEM7','ITEM9','ITEM10','ITEM11','ITEM12','ITEM13','ITEM13A','ITEM13B','ITEM16','I
'ITEM27','ITEM28','ITEM28A','ITEM28B','ITEM29','ITEM30','ITEM31','ITEM32','ITEM33','ITEM34','ITEM35','ITEM36','ITEM36A','ITEM36B','ITEM36C','ITEM36D','ITEM37','ITEM38','ITEM39','ITEM40','ITEM41',
'ITEM44','ITEM44A','ITEM44B','ITEM45','ITEM46','ITEM47','ITEM48','ITEM49','ITEM50','ITEM50A','ITEM50B','ITEM51','ITEM52','ITEM53','ITEM54','ITEM54A','ITEM54B','ITEM55','ITEM55A','ITEM55B','ITEM56
'ITEM64','ITEM65','ITEM66','ITEM67','ITEM68',  'ITEM69','ITEM70', 'ITEM71', 'ITEM72','ITEM75', 'ITEM75A','ITEM75B','ITEM76','ITEM90','ITEM91','ITEM92','ITEM92A','ITEM92B','ITEM92C','ITEM93','ITEM
'ITEM97','ITEM98','ITEM98A','ITEM98B','ITEM99','ITEM100','ITEM101','ITEM102','ITEM103','ITEM104','ITEM105','ITEM106','ITEM107','ITEM108','ITEM108A','ITEM108B','ITEM108C','ITEM109','ITEM110','ITEM
print(field_map.get('ITEM110'))
```

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

df_bridge_bronze01 = spark.sql("SELECT * FROM pipelinedatabricks.bridge.silverbridge")

display(df_bridge_bronze01)
df_bridge_bronze01.printSchema
```

```
%fs ls dbfs:/mnt/azuredatabricksmvp2024/bronze
```

Dados complementares

```
#ler camada bronze
from pyspark.sql.functions import *

file_location01 = 'dbfs:/mnt/azuredatabricksmvp2024/bronze/Tabela de Estados.CSV'
file_location02 = 'dbfs:/mnt/azuredatabricksmvp2024/bronze/TabelaRespon.csv'
file_location03 = 'dbfs:/mnt/azuredatabricksmvp2024/bronze/HISTORIA.CSV'
file_location04 = 'dbfs:/mnt/azuredatabricksmvp2024/bronze/DADOS01.CSV'
file_location05 = 'dbfs:/mnt/azuredatabricksmvp2024/bronze/DADOS02.CSV'
file_location06 = 'dbfs:/mnt/azuredatabricksmvp2024/bronze/DADOS03.CSV'
file_type = 'csv'
infer_schema = 'true'
first_row_is_header = 'true'
delimiter = ';'

df_nomes_estados = spark.read.format(file_type).option('inferSchema', infer_schema).option('header', first_row_is_header).option('sep', delimiter).load(file_location01)
df_tabResponsável = spark.read.format(file_type).option('inferSchema', infer_schema).option('header', first_row_is_header).option('sep', delimiter).load(file_location02)
df_tabHistoria = spark.read.format(file_type).option('inferSchema', infer_schema).option('header', first_row_is_header).option('sep', delimiter).load(file_location03)
df_dados01 = spark.read.format(file_type).option('inferSchema', infer_schema).option('header', first_row_is_header).option('sep', delimiter).load(file_location04)
df_dados02 = spark.read.format(file_type).option('inferSchema', infer_schema).option('header', first_row_is_header).option('sep', delimiter).load(file_location05)
df_dados03 = spark.read.format(file_type).option('inferSchema', infer_schema).option('header', first_row_is_header).option('sep', delimiter).load(file_location06)

display(df_nomes_estados)
display(df_tabResponsável)
display(df_tabHistoria)
display(df_dados01)
display(df_dados02)
display(df_dados03)
```

Aqui posso começar a tratar para evoluir para o silver.

```
# Excluindo item desnecessarios para nossa análise e criando a camada silver
df_bridge_silver=df_bridge_bronze01

deletar=['ITEM6B','ITEM7','ESTADO','STAT','SR2','EXTRA','DATE', 'LONGITUDE', 'LATITUDE', 'LOCAL', 'RESPONSÁVEL']
for ajuste in deletar:
    if ajuste in df_bridge_silver.columns:
        df_bridge_silver  = df_bridge_silver.drop (ajuste)
    else:
        print(f"Column {ajuste} does not exist in the dataframe.")

#alterar alguns cabeçalhos
df_bridge_silver=df_bridge_silver.withColumn('ANO INSPEÇÂO',substring('DATA INSPEÇÃO',-2,2))
df_bridge_silver = df_bridge_silver.withColumnRenamed("AVALIAÇÃO ESTRUCTURAL", "AVALIAÇÃO ESTRUTURAL")

#Ajuste de tipo de informação
datas=['CUSTO PONTE', 'ANO MANUTENÇÃO', 'FTMD', 'ANO CONSTRUÇÃO', 'QTD LINHAS', 'TMD', 'YTMD', 'TT', 'HISTORICA','MATERIAL','TIPO ESTRUTURAL','QTD VÃO','CLASSIFICAÇÃO OPERAÇÃO', 'CLASSIFICAÇÃO IN
INSPEÇÂO','SUPERESTRUTURA','INFRAESTRUTURA' ]

for ajuste in datas:
    if ajuste in df_bridge_silver.columns:  # Check if the column exists
        df_bridge_silver=df_bridge_silver\
            .withColumn(ajuste, df_bridge_silver[ajuste].cast('int'))\
            .fillna(0,subset=[ajuste])
    else:
        print(f"Column {ajuste} does not exist in the dataframe.")

df_bridge_silver = df_bridge_silver.withColumn('CUSTO_PONTE_REAIS', col('CUSTO PONTE') * 5500
                                               )

display(df_bridge_silver)
```

AQUI FINALIZEI O TRATAMENTO DE DADOS DO SILVER

```
from pyspark.sql.functions import col

# Rename columns with invalid characters
df_bridge_silver = df_bridge_silver.withColumnRenamed("ANO CONSTRUÇÃO", "ANO_CONSTRUCAO") \
                                   .withColumnRenamed("QTD LINHAS", "QTD_LINHAS")

# Assuming there might be other columns with invalid characters, ensure all column names are compliant
# This is a generic approach to replace spaces with underscores in all column names
for col_name in df_bridge_silver.columns:
    new_col_name = col_name.replace(" ", "_").replace(",", "_").replace(";", "_") \
                           .replace("{", "_").replace("}", "_").replace("(", "_") \
                           .replace(")", "_").replace("\n", "_").replace("\t", "_") \
                           .replace("=", "_")
    df_bridge_silver = df_bridge_silver.withColumnRenamed(col_name, new_col_name)

# Write the DataFrame to Delta
df_bridge_silver.write.format('delta') \
    .mode('overwrite') \
    .option('mergeSchema', 'true') \
    .save('/mnt/azuredatabricksmvp2024/silver/bridge_silver')
```

O cliente pode ja trabalhar com estas tabelas que estão tratadas e que podem ser utilizadas para diferentes perguntas. Nos utilizaremos uma nova camada para avaliar as questões inseridas no inicio do trabalho

```
%fs ls dbfs:/mnt/azuredatabricksmvp2024/silver
```

```
display(spark.read.format('delta').load('dbfs:/mnt/azuredatabricksmvp2024/silver/bridge_silver'))
```

```
# Alias each DataFrame
df_bridge_silver_alias = df_bridge_silver.alias("bridge")
df_tabResponsável_alias = df_tabResponsável.alias("responsavel")
df_tabHistoria_alias = df_tabHistoria.alias("historia")
df_dados01_alias = df_dados01.alias("dados01")
df_dados02_alias = df_dados02.alias("dados02")
df_dados03_alias = df_dados03.alias("dados03")

# Perform the joins using the aliased DataFrames
df_bridge_gold = (
    df_bridge_silver_alias.join(
        df_tabResponsável_alias,
        on=df_bridge_silver_alias["PROPRIETARIO"] == df_tabResponsável_alias["INDICADOR"],
        how="left"
    )
    .join(
        df_tabHistoria_alias,
        on=df_bridge_silver_alias["HISTORICA"] == df_tabHistoria_alias["PESO"],
        how="left"
    )
    .join(
        df_dados01_alias,
        on=df_bridge_silver_alias["FRATURA_CRÍTICA"] == df_dados01_alias["ITEM"],
        how="left"
    )
    .join(
        df_dados02_alias,
        on=df_bridge_silver_alias["MATERIAL"] == df_dados02_alias["ITEM"],
        how="left"
    )
    .join(
        df_dados03_alias,
        on=df_bridge_silver_alias["RESTRIÇÕES"] == df_dados03_alias["ITEM"],
        how="left"
    )
    .select(
        df_bridge_silver_alias["*"],
        df_tabResponsável_alias["RESPONSAVEL"].alias("NOME_PROP"),
        df_tabHistoria_alias["HISTORIA"].alias("NOM_HISTORICA"),
        df_dados01_alias["PESO"].alias("FRAT_CRÍTICA_ID"),
        df_dados02_alias["GERAL"].alias("NOM_TIPO_ESTRUTURAL"),
        df_dados02_alias["PESO"].alias("PESO_TIPO_ESTR"),
        df_dados03_alias["PESO"].alias("PESO_RESTRIÇÃO"),
    )
)

display(df_bridge_gold)
```

```
#Tratamento gold
df_bridge_gold=df_bridge_gold.withColumn('Risco', col('HISTORICA')  + col('PESO_RESTRIÇÃO')+ col('PESO_TIPO_ESTR') +  col('FRAT_CRÍTICA_ID')
                                        )
display(df_bridge_gold)
```

Finalmente salvamos a camada gold na nossa pasta

```
from pyspark.sql.functions import col

# Rename columns with invalid characters
df_bridge_gold = df_bridge_gold.withColumnRenamed("ANO CONSTRUÇÃO", "ANO_CONSTRUCAO") \
                                  .withColumnRenamed("QTD LINHAS", "QTD_LINHAS")

# Assuming there might be other columns with invalid characters, ensure all column names are compliant
# This is a generic approach to replace spaces with underscores in all column names
for col_name in df_bridge_gold.columns:
    new_col_name = col_name.replace(" ", "_").replace(",", "_").replace(";", "_") \
                           .replace("{", "_").replace("}", "_").replace("(", "_") \
                           .replace(")", "_").replace("\n", "_").replace("\t", "_") \
                           .replace("=", "_")
    df_bridge_gold = df_bridge_gold.withColumnRenamed(col_name, new_col_name)

# Write the DataFrame to Delta
df_bridge_gold.write.format('delta') \
    .mode('overwrite') \
    .option('mergeSchema', 'true') \
    .save('/mnt/azuredatabricksmvp2024/gold/bridge_gold')
```

```
%fs ls dbfs:/mnt/azuredatabricksmvp2024/gold
```

```
display(spark.read.format('delta').load('dbfs:/mnt/azuredatabricksmvp2024/gold/bridge_gold'))
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Display the first few rows of the dataframe
df_bridge_gold.show()

# Show the general info about dataframe
df_bridge_gold.printSchema()

# Calculate basic statistics for the numeric columns
df_bridge_gold.describe().show()
```

```
df_bridge_gold.createOrReplaceTempView("df_bridge_gold")
```

```
plt.figure(figsize=(12, 8))
sns.heatmap(df_bridge_gold_pd.corr(), cmap='coolwarm', annot=True, fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
plt.figure(figsize=(10, 6))
sns.countplot(x='NOM_TIPO_ESTRUTURAL', data=df_bridge_gold_pd)
plt.title('Common types of bridges in the area')
plt.show()
```

Respondendo e autoavaliação

• Risco estrutural das pontes? Observou-se que existe um risco maior em estruturas mais antigas e materiais concretos e aço. Os dados dizem isto mas historicamente as estruturas americanas de concreto e aço são muitíssimo superdimensionadas;

• Frequência necessária de inspeção? A frequência necessária foi determinada pelas inspeções anteriores, observando que quanto mais frequente a inspeção a estrutura tem menor risco. Observamos também que o período de inspeção mais utilizado é 24 meses, que para a quantidade de pontes é um bom parametro para concluir da necessidade de empresas que realizem este serviço;

• Quantidades de oportunidades e profissionais? Não conseguimos visualizar com gráficos, porém pelo entendimento do problema, observamos que pela quantidade de pontes e os risco altos segundo a metodologia adotada, haverá uma demanda crescente pela busca destes profissionais e empresas que trabalhem nesta área;

• Necessidade de investimento? O investimento é grande em função da quantidade de pontes, para cada estado poderá ser uma quantia mais viável para orçamentos plurianuais. Podemos observar que seria necessário um valor de investimento na faixa de 30B de reais;

- Adicional A base de dados, ainda observou-se algumas deficiências como valores negativos de investimento e o cálculo de risco maior para estruturas mais robustas. Também observou-se uma forte relação entre variáveis: tráfego médio, material estrutural e tipologias. Consideramos ter cumprido nossa análise, utilizado a nuvem adequadamente com os programas Azure e databricks e