MVP 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; • mtguide.pdf, obtido do site da
internet 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

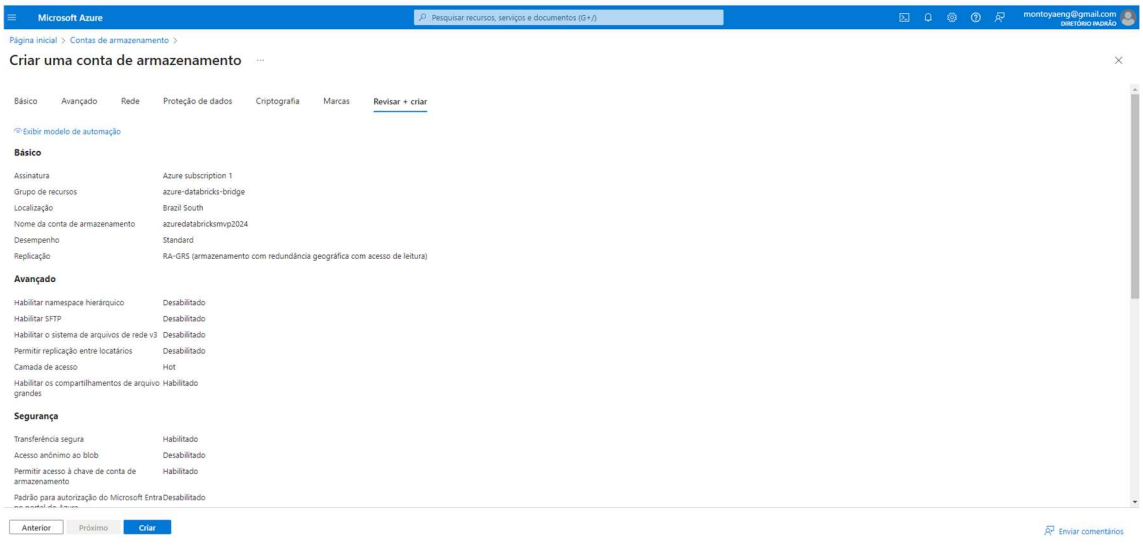
dados que não seriam úteis para nossos questionamentos. Nas oportunidades de trabalho foi mais complexo o tratamento porque existem muitas funções e precisamos de avaliar as que são úteis para nosso questionamento.

Nome	Modificado	Camada de acesso	Status do arquivo	Tipo de blob	Tamanho	Estado de concessão
BridgesExport_AllYear.csv	03/07/2024, 09:31:09	Principal (Inferidos)		Blob de blocos	97.62 MiB	Disponível
currency_exchange.csv	03/07/2024, 09:33:10	Principal (Inferidos)		Blob de blocos	13.73 KiB	Disponível
data_NBI.csv	02/07/2024, 09:59:54	Principal (Inferidos)		Blob de blocos	298.39 MiB	Disponível
glassdoor_construction_jobs_usa....	02/07/2024, 12:08:10	Principal (Inferidos)		Blob de blocos	1.7 MiB	Disponível
mtguide.pdf	02/07/2024, 09:25:35	Principal (Inferidos)		Blob de blocos	831.91 KiB	Disponível

2. Coleta, Modelagem e Carga

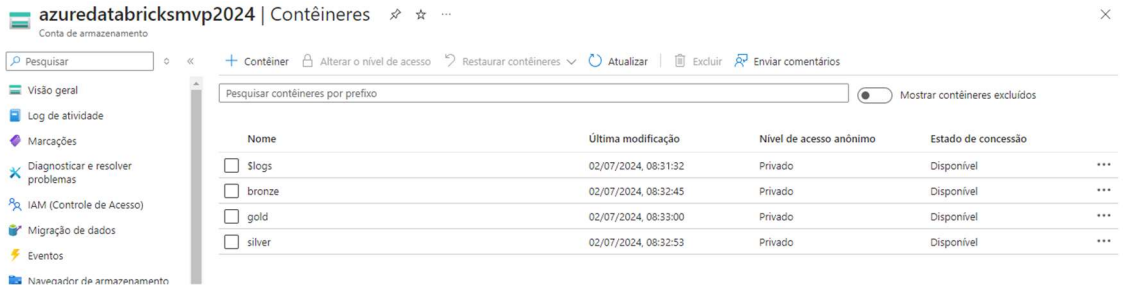
Uma vez definido o conjunto de dados, devemos coletar e armazená-los na nuvem, este processo de armazenagem segue as disposições de uma arquitetura par ETL, desenvolvendo assim está no Azure, utilizando a carga dos dados para o Data Warehouse/Data Lake. Utilizamos pipelines de ETL (Extração, Transformação e Carga) na Azure e Databricks.

Criada conta de armazenamento com as três camadas.

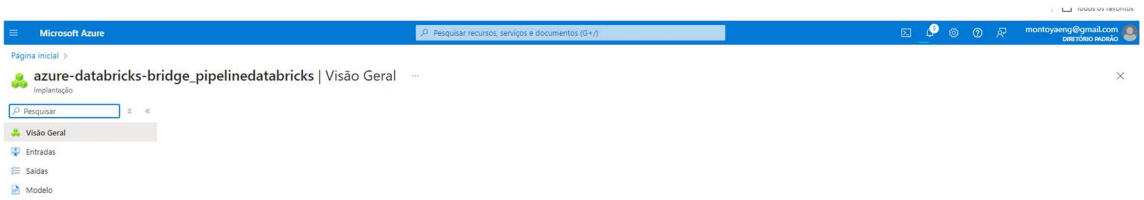



```
[
  {
    "id": "/subscriptions/2fd0ff2e-3433-4530-9e94-
ed6191c23e33/resourceGroups/azure-databricks-
bridge/providers/Microsoft.Resources/deployments/azuredatabricksmvp2024_1
719919771847/operations/4E4006BA29EEDDB8",
    "operationId": "4E4006BA29EEDDB8",
    "properties": {
      "provisioningOperation": "Create",
      "provisioningState": "Running",
      "timestamp": "2024-07-02T11:31:07.0479451Z",
      "duration": "PT5.0106552S",
      "trackingId": "53efd8f8-f03e-4fe2-a7e8-9496e97481d4",
```

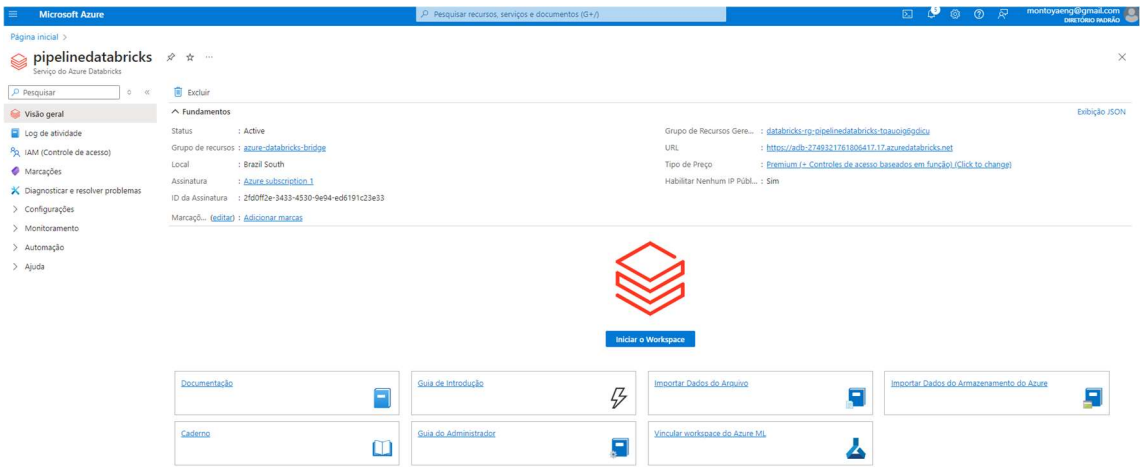
```
      "statusCode": "Accepted",
      "targetResource": {
        "id": "/subscriptions/2fd0ff2e-3433-4530-9e94-
ed6191c23e33/resourceGroups/azure-databricks-
bridge/providers/Microsoft.Storage/storageAccounts/azuredatabricksmvp2024
",
        "resourceType": "Microsoft.Storage/storageAccounts",
        "resourceName": "azuredatabricksmvp2024"
      }
    }
  }
]
```

Criado o pipeline.



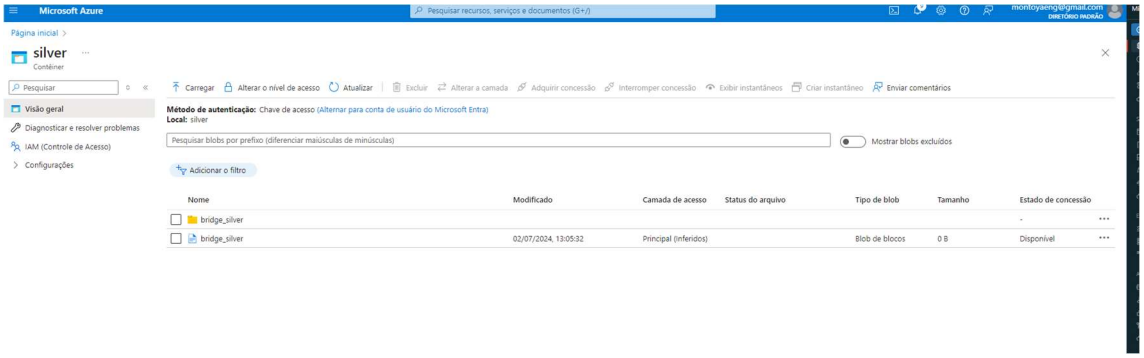
E criado nosso cluster com nosso notebook



Na camada bronze foi colocado nossos dados brutos

Nome	Modificado	Camada de acesso	Status do arquivo	Tipo de blob	Tamanho	Estado de concessão
BridgesExport_AllYear.csv	03/07/2024, 09:31:09	Principal (Inferidos)		Blob de blocos	97.62 MiB	Disponível
currency_exchange.csv	03/07/2024, 09:33:10	Principal (Inferidos)		Blob de blocos	13.73 KiB	Disponível
data_NBI.csv	02/07/2024, 09:59:54	Principal (Inferidos)		Blob de blocos	298.39 MiB	Disponível
glassdoor_construction_jobs_usa....	02/07/2024, 12:08:10	Principal (Inferidos)		Blob de blocos	1.7 MiB	Disponível
mtguide.pdf	02/07/2024, 09:25:35	Principal (Inferidos)		Blob de blocos	831.91 KiB	Disponível

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

5. Análise

Qualidade de dados

Os atributos encontrados tiveram alguns dados desnecessários para nossa análise, não é uma boa pratica alterar a camada bruta, por isso que o tratamento dos dados é feito na silver, aqui deletamos dados que não seriam uteis para nossos questionamentos.

Nas oportunidades de trabalho foi mais complexo o tratamento porque existem muitas funções e precisamos de avaliar as que são uteis para nosso questionamento.

Solução do problema no arquivo databricks

Montagem das bases das camadas bronze, silver e gold

```
4 minutes ago (21s)                                                    4
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=='}
)
```

```
/mnt/azuredatabricksmvp2024/bronze has been unmounted.
True
```

```
4 minutes ago (21s)                                                    5
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=='}
)
```

```
/mnt/azuredatabricksmvp2024/silver has been unmounted.
True
```

```
4 minutes ago (21s)                                                    6
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=='}
)
```

```
/mnt/azuredatabricksmvp2024/gold has been unmounted.
True
```

+ Code + Text

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

```
4 minutes ago (<1s)                                                    8
#criar database
spark.sql('CREATE DATABASE IF NOT EXISTS bridge')
```

```
DataFrame[]
```

```
4 minutes ago (15s)                                                    9
#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)

(3) Spark Jobs
```

(3) Spark Jobs

df_bridge_bronze: pyspark.sql.dataframe.DataFrame = [X: double, Y: double ... 133 more fields]

Table +

	1.2 X	1.2 Y	FID	STFIPS	REGION	ITEM8	ITEM5A	ITEM5B	ITEM5C	ITEM5D	ITEM5E	ITEM2
1	9.9988e-11	9.9984e-11	1001	16	0	0414050000010...	1	4	0	70227	0	04
2	9.9988e-11	9.9984e-11	1002	16	0	0414050000010...	1	4	0	70227	0	04
3	9.9988e-11	9.9984e-11	1003	16	0	0414050000010...	1	4	0	70227	0	04
4	9.9988e-11	9.9984e-11	1004	16	0	0414050000010...	1	6	0	70227	0	04
5	9.9988e-11	9.9984e-11	1005	16	0	0414050000010...	1	6	0	70227	0	04
6	9.9988e-11	9.9984e-11	1006	16	0	0414050000010...	1	6	0	70487	0	04
7	9.9988e-11	9.9984e-11	1007	16	0	0415000000145...	1	6	0	80235	0	00
8	9.9988e-11	9.9984e-11	1008	16	0	0415010000010...	1	4	0	80006	0	00
9	9.9988e-11	9.9984e-11	1009	16	0	0415010000010...	1	4	0	80006	0	00
10	9.9988e-11	9.9984e-11	1010	16	0	0415020000010...	1	6	0	80045	0	00
11	9.9988e-11	9.9984e-11	1011	16	0	0415020000010...	1	6	0	80059	0	00
12	9.9988e-11	9.9984e-11	1012	16	0	0415020000010...	1	6	0	80059	0	00
13	9.9988e-11	9.9984e-11	1013	16	0	0415030000010...	1	4	0	80082	0	00
14	9.9988e-11	9.9984e-11	1014	16	0	0415030000010...	1	6	0	80097	0	00

2,980+ rows | Truncated data due to byte limit | 15.09 seconds runtime Refreshed 3 minutes ago

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

4 minutes ago (<1s) 11 Python

```
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'},
    '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','ITEM17','ITEM19','ITEM20','ITEM21','ITEM22','ITEM26','ITEM27','ITEM28','ITEM28A','ITEM28B','ITEM29','ITEM30','ITEM31','ITEM32','ITEM33','ITEM34','ITEM35',
'ITEM36','ITEM36A','ITEM36B','ITEM36C','ITEM36D','ITEM37','ITEM38','ITEM39','ITEM40','ITEM41','ITEM42','ITEM42A','ITEM42B','ITEM43','ITEM43A','ITEM43B','ITEM44','ITEM44A','ITEM44B',
'ITEM45','ITEM46','ITEM47','ITEM48','ITEM49','ITEM50','ITEM50A','ITEM50B','ITEM51','ITEM52','ITEM53','ITEM54','ITEM54A','ITEM54B','ITEM55','ITEM55A','ITEM55B','ITEM56','ITEM58',
'ITEM59','ITEM60','ITEM61','ITEM62','ITEM63','ITEM64','ITEM65','ITEM66','ITEM67','ITEM68',  'ITEM69','ITEM70', 'ITEM71', 'ITEM72','ITEM75', 'ITEM75A','ITEM75B','ITEM76','ITEM90',
'ITEM91','ITEM92','ITEM92A','ITEM92B','ITEM92C','ITEM93','ITEM93A','ITEM93B','ITEM93C','ITEM94','ITEM95','ITEM96','ITEM97','ITEM98','ITEM98A','ITEM98B','ITEM99','ITEM100','ITEM101',
'ITEM102','ITEM103','ITEM104','ITEM105','ITEM106','ITEM107','ITEM108','ITEM108A','ITEM108B','ITEM108C','ITEM109','ITEM110','ITEM111','ITEM112','ITEM113','ITEM114','ITEM115',
'ITEM116']
print(field_map.get('ITEM110'))
```

```
{'DESIGNATED NATIONAL NETWORK'}
```

10:49 PM (3s) 12 Python

```
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
```

(1) Spark Jobs

df_bridge_bronze01: pyspark.sql.dataframe.DataFrame = [ID: string, ITEM6B: string ... 38 more fields]

Table

	ID	ITEM6B	ITEM7	LOCAL	LATITUDE	LONGITUDE	RESPONSÁVEL	PROPRIETARIO	ANO CONSTRUÇÃO
1	0414050000010...		FDR	BIG SMOKY JCT	0 0 0	0 0 0	64	64	1982
2	0414050000010...		FDR	2 MILES WEST BIG SMOKY JT	0 0 0	0 0 0	64	64	1953
3	0414050000010...		FDR	MOUTH BIG SMOKY CREEK	0 0 0	0 0 0	64	64	1951
4	0414050000010...		FDR	ELMORE-CAMAS COUNTY LI...	0 0 0	0 0 0	64	64	1955
5	0414050000010...		FDR	2 MILES EAST SHAKE CR G S	0 0 0	0 0 0	64	64	1955
6	0414050000010...		FDR	10 MILES WEST FEATHERVILE	0 0 0	0 0 0	64	64	1954
7	0415000000145...		ROAD	NO DATA ENTERED	0 0 0	0 0 0	64	64	2009
8	0415010000010...		ROAD	NO DATA ENTERED	0 0 0	0 0 0	64	64	1972
9	0415010000010...		ROAD	NO DATA ENTERED	0 0 0	0 0 0	64	64	1991
10	0415020000010...		ROAD	NO DATA ENTERED	0 0 0	0 0 0	64	64	1975
11	0415020000010...		ROAD	NO DATA ENTERED	0 0 0	0 0 0	64	64	1991
12	0415020000010...		ROAD	NO DATA ENTERED	0 0 0	0 0 0	64	64	1957
13	0415030000010...		ROAD	NO DATA ENTERED	0 0 0	0 0 0	64	64	1984
14	0415030000010...		ROAD	NO DATA ENTERED	0 0 0	0 0 0	64	64	1954

8,651+ rows | Truncated data due to byte limit | 2.71 seconds runtime

Refreshed 4 minutes ago

8,651+ rows | Truncated data due to byte limit | 2.71 seconds runtime

Refreshed 4 minutes ago

```
<bound method DataFrame.printSchema of DataFrame[ID: string, ITEM6B: string, ITEM7: string, LOCAL: string, LATITUDE: string, LONGITUDE: string, RESPONSÁVEL: bigint, PROPRIETARIO: bi
t, ANO CONSTRUÇÃO: string, QTD LINHAS: string, QTD LINHAS DEBAIXO: bigint, TMD: string, YTMD: string, TT: string, HISTORICA: string, RESTRIÇÕES: string, MATERIAL: bigint, TIPO ESTRUTUR
AL: bigint, QTD VÃO: string, VÃO: bigint, VÃO TOTAL: bigint, LARGURA: bigint, SUPERESTRUTURA: string, INFRAESTRUTURA: string, CLASSIFICAÇÃO OPERAÇÃO: string, CLASSIFICAÇÃO INICIAL: str
ing, AVALIAÇÃO ESTRUCTURAL: string, DATA INSPEÇÃO: string, FREQ INSPEÇÃO: string, FRATURA CRÍTICA: string, INSPEÇÃO SUBAQUATICA: string, INSPEÇÃO ESPECIAL: string, CUSTO PONTE: string,
ANO MANUTENÇÃO: string, ESTADO: string, FTMD: string, STAT: bigint, SR2: string, EXTRA: bigint, DATE: bigint]>
```

10:49 PM (2s) 13

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

Table

	path	name	size	modificationTime
1	dbfs:/mnt/azuredatabricksmvp2024/bronze/DADOS01.CSV	DADOS01.CSV	48	1720049376000
2	dbfs:/mnt/azuredatabricksmvp2024/bronze/DADOS02.CSV	DADOS02.CSV	239	1720049325000
3	dbfs:/mnt/azuredatabricksmvp2024/bronze/DADOS03.CSV	DADOS03.CSV	51	1720049257000
4	dbfs:/mnt/azuredatabricksmvp2024/bronze/HISTORIA.CSV	HISTORIA.CSV	352	1720032824000
5	dbfs:/mnt/azuredatabricksmvp2024/bronze/Tabela de Estados.CSV	Tabela de Estados.CSV	773	1720023393000
6	dbfs:/mnt/azuredatabricksmvp2024/bronze/TabelaRespon.csv	TabelaRespon.csv	826	1720028942000
7	dbfs:/mnt/azuredatabricksmvp2024/bronze/data_NBI.csv	data_NBI.csv	312885086	1719925194000
8	dbfs:/mnt/azuredatabricksmvp2024/bronze/mtguide.pdf	mtguide.pdf	851873	1719923135000

8 rows | 2.03 seconds runtime

Refreshed 4 minutes ago

Dados complementares

10:49 PM (3s) 15 Python

```
#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)
```

(18) Spark Jobs

- df_nomes_estados: pyspark.sql.dataframe.DataFrame = [Numero: integer, Estado: string]
- df_tabResponsável: pyspark.sql.dataframe.DataFrame = [INDICADOR: integer, RESPONSAVEL: string]
- df_tabHistoria: pyspark.sql.dataframe.DataFrame = [PESO: integer, HISTORIA: string]
- df_dados01: pyspark.sql.dataframe.DataFrame = [ITEM: string, PESO: integer]
- df_dados02: pyspark.sql.dataframe.DataFrame = [ITEM: integer, GERAL: string ... 1 more field]
- df_dados03: pyspark.sql.dataframe.DataFrame = [ITEM: string, PESO: integer]

Table

	Numero	Estado
1	14	Alabama
2	20	Alaska
3	49	Arizona
4	56	Arkansas
5	69	California
6	88	Colorado
7	91	Connecticut
8	103	Delaware
9	113	District of Columb...
10	124	Florida
11	134	Georgia
12	159	Hawaii
13	160	Idaho
14	175	Illinois
15	185	Indiana

52 rows | 2.84 seconds runtime

Refreshed 5 minutes ago

Table

	INDICADOR	RESPONSAVEL
1	1	State Highway Agency
2	2	County Highway Agency
3	3	Town or Township Highway Agency
4	4	City or Municipal Highway Agency
5	11	State Park, Forest, or Reservation Agency
6	12	Local Park, Forest, or Reservation Agency
7	21	Other State Agencies

Aqui posso começar a tratar para evoluir para o silver.

10:49 PM (4s) 17 Python

```
# 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 INSPEÇÃO',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 INICIAL','AVALIAÇÃO ESTRUTURAL','FREQ INSPEÇÃO','ANO INSPEÇÃO','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)
```

(1) Spark Jobs

(1) Spark Jobs

df_bridge_silver: pyspark.sql.dataframe.DataFrame = [ID: string, PROPRIETARIO: long ... 29 more fields]

Table

	ID	PROPRIETARIO	ANO CONSTRUÇÃO	QTD LINHAS	QTD LINHAS DEBAIXO	TMD	YTMD	TT	HISTORICA	RESTR
1	080074	1	2012	3	2	18974	2008	0	5	A
2	7700243	2	1985	2	0	190	2011	0	5	A
3	410A13700...	2	1991	1	0	290	2013	0	4	K
4	8400226	2	1986	2	0	173	2011	5	5	A
5	63F00190001	2	1935	2	0	2900	2013	2	4	A
6	080076	1	2011	2	0	18974	2008	9	5	A
7	410A13700...	2	1991	1	0	290	2013	0	4	P
8	8400339	2	2009	2	0	693	2011	5	5	A
9	054B00215N	1	1995	2	0	324	2012	9	5	A
10	63S61170001	4	2004	4	0	7600	2013	5	4	A
11	410A16500...	2	1962	2	0	440	2013	2	4	P
12	8400340	2	2009	4	0	703	2011	5	5	A
13	63S62690001	2	1942	2	0	1370	2013	2	4	A
14	410A16700...	2	1975	2	0	20	2013	2	4	A

10,000+ rows | Truncated data due to row limit | 4.32 seconds runtime

Refreshed 5 minutes ago

AQUI FINALIZEI O TRATAMENTO DE DADOS DO SILVER

10:49 PM (14s) 19 Python

```
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')
```

(6) Spark Jobs

df_bridge_silver: pyspark.sql.dataframe.DataFrame = [ID: string, PROPRIETARIO: long ... 29 more fields]

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

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

10:49 PM (2s) 21

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

Table

	path	name	size	modificationTime
1	dbfs:/mnt/azuredatabricksmvp2024/silver/bridge_silv...	bridge_silver/	0	1720057837000

1 row | 1.54 seconds runtime Refreshed 5 minutes ago

10:49 PM (1s) 22 Python

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

(2) Spark Jobs

Table

	ID	PROPRIETARIO	ANO_CONSTRUCAO	QTD_LINHAS	QTD_LINHAS_DEBAIXO	TMD	YTMD	TT	HISTORICA	RE
1	0414050000010...	64	1982	1	0	50	0	5	4	A
2	0414050000010...	64	1953	1	0	50	0	4	4	P
3	0414050000010...	64	1951	1	0	50	0	4	4	P
4	0414050000010...	64	1955	1	0	50	0	4	4	A
5	0414050000010...	64	1955	1	0	50	0	4	4	A
6	0414050000010...	64	1954	1	0	50	0	4	4	P
7	0415000000145...	64	2009	1	0	50	0	0	4	A
8	0415010000010...	64	1972	2	0	50	0	4	4	A
9	0415010000010...	64	1991	2	0	50	0	4	4	A
10	0415020000010...	64	1975	1	0	50	0	5	4	A
11	0415020000010...	64	1991	2	0	50	0	5	4	A
12	0415020000010...	64	1957	2	0	50	0	0	4	A
13	0415030000010...	64	1984	2	0	50	0	5	4	A
14	0415030000010...	64	1954	1	0	50	0	0	4	P

10,000+ rows | Truncated data due to row limit | 1.29 seconds runtime Refreshed 5 minutes ago

10:49 PM (2s) 23 Python

```python
# 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)
```

▸ (6) Spark Jobs

- ▸ df_bridge_silver_alias: pyspark.sql.dataframe.DataFrame = [ID: string, PROPRIETARIO: long ... 29 more fields]
- ▸ df_tabResponsável_alias: pyspark.sql.dataframe.DataFrame = [INDICADOR: integer, RESPONSAVEL: string]
- ▸ df_tabHistoria_alias: pyspark.sql.dataframe.DataFrame = [PESO: integer, HISTORIA: string]
- ▸ df_dados01_alias: pyspark.sql.dataframe.DataFrame = [ITEM: string, PESO: integer]
- ▸ df_dados02_alias: pyspark.sql.dataframe.DataFrame = [ITEM: integer, GERAL: string ... 1 more field]
- ▸ df_dados03_alias: pyspark.sql.dataframe.DataFrame = [ITEM: string, PESO: integer]
- ▸ df_bridge_gold: pyspark.sql.dataframe.DataFrame = [ID: string, PROPRIETARIO: long ... 35 more fields]

Table +

	ID	PROPRIETARIO	ANO_CONSTRUCAO	QTD_LINHAS	QTD_LINHAS_DEBAIXO	TMD	YTMD	TT	HISTORICA	REST
1	080074	1	2012	3	2	18974	2008	0	5	A
2	7700243	2	1985	2	0	190	2011	0	5	A
3	410A13700...	2	1991	1	0	290	2013	0	4	K
4	8400226	2	1986	2	0	173	2011	5	5	A
5	63F00190001	2	1935	2	0	2900	2013	2	4	A
6	080076	1	2011	2	0	18974	2008	9	5	A
7	410A13700...	2	1991	1	0	290	2013	0	4	P
8	8400339	2	2009	2	0	693	2011	5	5	A
9	054B00215N	1	1995	2	0	324	2012	9	5	A

11	410A16500...	2	1962	2	0	440	2013	2	4	P
12	8400340	2	2009	4	0	703	2011	5	5	A
13	63S62690001	2	1942	2	0	1370	2013	2	4	A
14	410A16700...	2	1975	2	0	20	2013	2	4	A

8,607+ rows | Truncated data due to byte limit | 2.27 seconds runtime

Refreshed 6 minutes ago

10:49 PM (3s) 24

```
#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)
```

(6) Spark Jobs

df_bridge_gold: pyspark.sql.dataframe.DataFrame = [ID: string, PROPRIETARIO: long ... 36 more fields]

Table

	ID	PROPRIETARIO	ANO_CONSTRUCAO	QTD_LINHAS	QTD_LINHAS_DEBAIXO	TMD	YTMD	TT	HISTORICA	REST
1	080074	1	2012	3	2	18974	2008	0	5	A
2	7700243	2	1985	2	0	190	2011	0	5	A
3	410A13700...	2	1991	1	0	290	2013	0	4	K
4	8400226	2	1986	2	0	173	2011	5	5	A
5	63F00190001	2	1935	2	0	2900	2013	2	4	A
6	080076	1	2011	2	0	18974	2008	9	5	A
7	410A13700...	2	1991	1	0	290	2013	0	4	P
8	8400339	2	2009	2	0	693	2011	5	5	A
9	054B00215N	1	1995	2	0	324	2012	9	5	A
10	63S61170001	4	2004	4	0	7600	2013	5	4	A
11	410A16500...	2	1962	2	0	440	2013	2	4	P
12	8400340	2	2009	4	0	703	2011	5	5	A
13	63562690001	2	1942	2	0	1370	2013	2	4	A
14	410A16700...	2	1975	2	0	20	2013	2	4	A

8,532+ rows | Truncated data due to byte limit | 2.60 seconds runtime

Refreshed 6 minutes ago

Finalmente salvamos a camada gold na nossa pasta

10:49 PM (18s) 26

```
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')
```

(11) Spark Jobs

10:49 PM (2s) 27 Python

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

Table

	path	name	size	modificationTime
1	dbfs:/mnt/azuredatabricksmvp2024/gold/bridge_gol...	bridge_gold/	0	1720057860000

1 row | 1.79 seconds runtime

Refreshed 6 minutes ago

10:49 PM (1s) 28

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

(2) Spark Jobs

Table | RiscoID | %Risco | RiscoxMaterial | Investimentos | Inspeção | Riscos | InspeçãoxRIsco 1 | investimento total

	ID	PROPRIETARIO	ANO_CONSTRUCAO	QTD_LINHAS	QTD_LINHAS_DEBAIXO	TMD	YTMD	TT	HISTORICA	RE
1	0414050000010...	64	1982	1	0	50	0	5	4	A
2	0414050000010...	64	1953	1	0	50	0	4	4	P
3	0414050000010...	64	1951	1	0	50	0	4	4	P
4	0414050000010...	64	1955	1	0	50	0	4	4	A
5	0414050000010...	64	1955	1	0	50	0	4	4	A
6	0414050000010...	64	1954	1	0	50	0	4	4	P
7	0415000000145...	64	2009	1	0	50	0	0	4	A
8	0415010000010...	64	1972	2	0	50	0	4	4	A
9	0415010000010...	64	1991	2	0	50	0	4	4	A
10	0415020000010...	64	1975	1	0	50	0	5	4	A
11	0415020000010...	64	1991	2	0	50	0	5	4	A
12	0415020000010...	64	1957	2	0	50	0	0	4	A
13	0415030000010...	64	1984	2	0	50	0	5	4	A
14	0415030000010...	64	1954	1	0	50	0	0	4	P

8,795+ rows | Truncated data due to byte limit | 1.38 seconds runtime

Refreshed 7 minutes ago

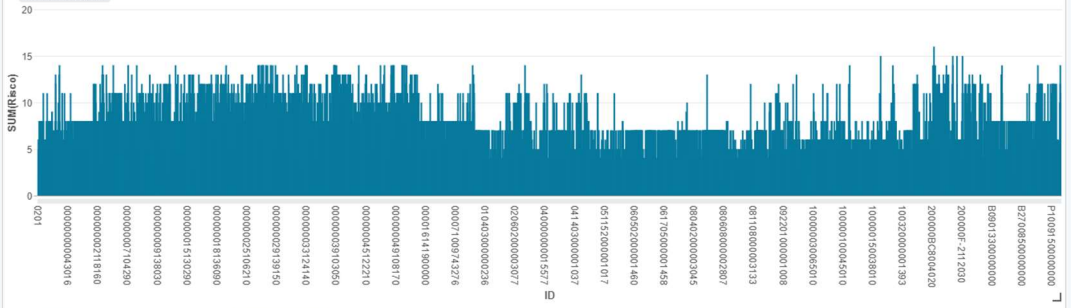
10:49 PM (1s) 28

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

(2) Spark Jobs

Table | RiscoID | %Risco | RiscoxMaterial | Investimentos | Inspeção | Riscos | InspeçãoxRIsco 1 | investimento total

New charts: ON



Edit Visualization | 8,795+ rows | Truncated data

Refreshed 8 minutes ago

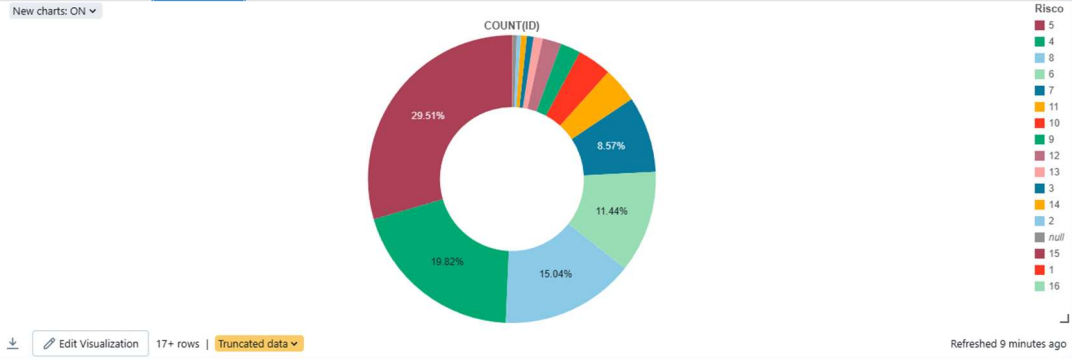
10:49 PM (1s) 28

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

(2) Spark Jobs

Table | RiscoID | %Risco | RiscoxMaterial | Investimentos | Inspeção | Riscos | InspeçãoxRIsco 1 | investimento total

New charts: ON



Edit Visualization | 17+ rows | Truncated data

Refreshed 9 minutes ago

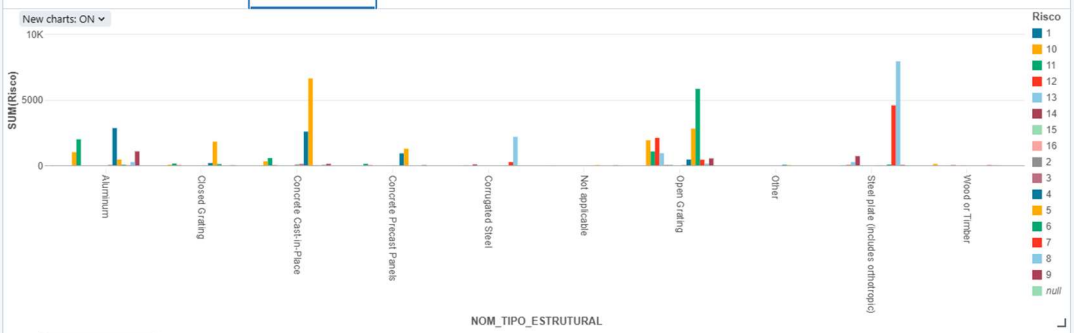
10:49 PM (1s) 28

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

(2) Spark Jobs

Table | RiscoID | %Risco | RiscoxMaterial | Investimentos | Inspeção | Riscos | InspeçãoxRIsco 1 | investimento total

New charts: ON



Edit Visualization | 90+ rows | Truncated data | Refreshed 9 minutes ago

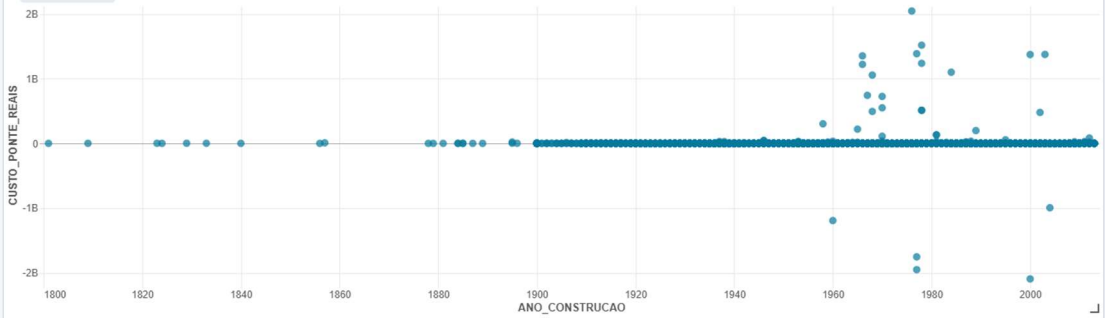
10:49 PM (1s) 28

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

(2) Spark Jobs

Table | RiscoID | %Risco | RiscoxMaterial | Investimentos | Inspeção | Riscos | InspeçãoxRIsco 1 | investimento total

New charts: ON



Edit Visualization | 8,795+ rows | Truncated data | Refreshed 9 minutes ago

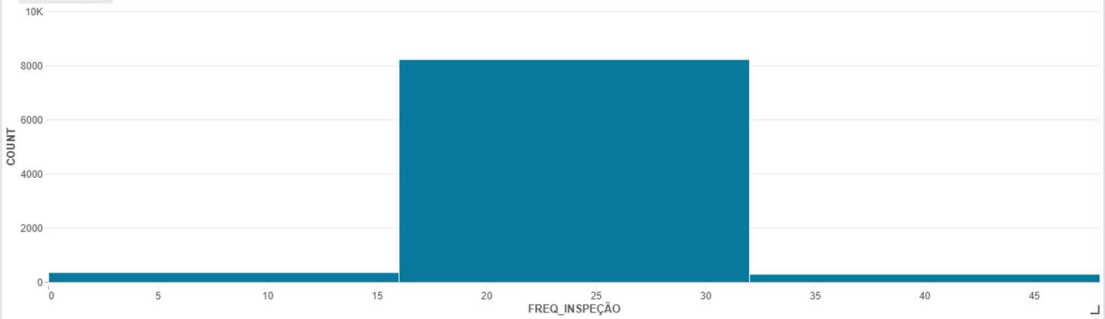
10:49 PM (1s) 28 Python

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

(2) Spark Jobs

Table | RiscoID | %Risco | RiscoxMaterial | Investimentos | Inspeção | Riscos | InspeçãoxRIsco 1 | investimento total

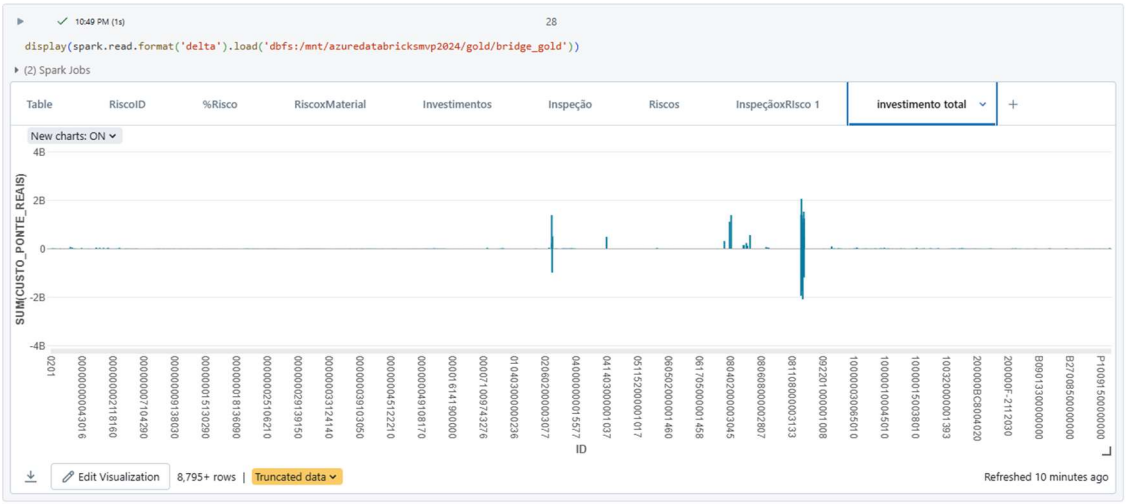
New charts: ON



Edit Visualization | 3+ rows | Truncated data | Refreshed 9 minutes ago


10:49 PM (1s)
28
Python
display(spark.read.format('delta').load('dbfs:/mnt/azuredatabricksmvp2024/gold/bridge_gold'))
(2) Spark Jobs
Table
RiscoID
%Risco
RiscoxMaterial
Investimentos
Inspeção
Riscos
InspeçãoxRIsco 1
investimento total
New charts: ON
COUNT
Risco
Edit Visualization
11+ rows | Truncated data
Refreshed 9 minutes ago


10:49 PM (1s)
28
display(spark.read.format('delta').load('dbfs:/mnt/azuredatabricksmvp2024/gold/bridge_gold'))
(2) Spark Jobs
Table
RiscoID
%Risco
RiscoxMaterial
Investimentos
Inspeção
Riscos
InspeçãoxRIsco 1
investimento total
New charts: ON
SUM(FREQ_INSPEÇÃO)
Risco
Edit Visualization
17+ rows | Truncated data
Refreshed 9 minutes ago


10:49 PM (1s)
28
display(spark.read.format('delta').load('dbfs:/mnt/azuredatabricksmvp2024/gold/bridge_gold'))
(2) Spark Jobs
Table
RiscoID
%Risco
RiscoxMaterial
Investimentos
Inspeção
Riscos
InspeçãoxRIsco 1
investimento total
New charts: ON
SUM(CUSTO_PONTE_REAIS)
ID
Edit Visualization
8,795+ rows | Truncated data
Refreshed 10 minutes ago

10:49 PM (21s) 29 Python

```
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()
```

(13) Spark Jobs

```
|   mean|        Infinity|3.262330538488148|1971.1583128798986| 2.274024412341246|0.6676709440426969|7630.773332564614|2000.9467722008994| 3.795374503763981| 4.768978865699178|      N
ULL|3.07006605498542 17|6.882891594050109| 2.773220551172023|146.75911838832425| 441.6228902762449|89.17733210338181|5.184563559391833|5.083926071128535|     497.5065989095162|    32
1.7705635264467|   5.975627192910208|670.9034905466079|24.182539081160 33|          NULL|             NULL|             NULL|488.9957253693973|718.7732106676331|10713.415930617557
|11.691404615612697| 1939535.7357866473|     NULL|             NULL|0.04173200949181833 7|             NULL|0.9617062282768 05|0.6796952098521356|6.450582363826482|
| stddev|             NaN| 8.830349025562 48| 25.53496184076315|1.1537644425539288|2.1687271363498692|21483.21418437405|117.88024539313182|2.4250991301965485|0.5500476748547611|      N
ULL|1.8102702711453869|7.3528699692170 55|4.9850589517004105|156.45968491224374|1552.3305129131916|70.90764018500532|2.9848616393233 21|2.944054124673846|     215.7931830631554|    14
6.2046160207077|  1.61237696185538 98|343.3115261158801| 6.56123577947927|          NULL|             NULL|             NULL|9342.662474213743|963.7348008108795|28751.747454986176
| 1.2178279710695 96|3.3342560192945 64E7|     NULL|             NULL| 0.2696674284 0948665|             NULL| 1.2478629775439 7|1.8885308193101142|2.2499412613213 96|
|    min|           0002|             1|                0|                0|                0|                0|                0|                0|                0|                0|
|                0|                0|                0|                0|               61|                0|                0|                0|                0|
0|                0|                |                0|                |                |                |                0|                0|                0|
0|       -2138967296|Air Force|Bridge is eligibl...|                0|         Aluminum|                0|                0|                0|
|    max|   YUMY-00.30-A|             80|             2014|               82|               99|           985500|             2013|                9|                5|
R|                9|               22|              998|            22915|           384216|             9999|                9|                9|              999|
999|                9|             1297|               48|              n00|              Y72|              Y99|           984000|             7636|           999999|
99|        2088757000|  Unknown|Historical signif...|                3|     Wood or Timber|                5|                7|               17|
+-------+----------------+-----------------+-----------------+------------------+------------------+-----------------+------------------+------------------+------------------+
---+-----------------+-----------------+-----------------+------------------+------------------+-----------------+------------------+------------------+------------------+
-------------+-----------------+-----------------+-----------------+------------------+------------------+-----------------+------------------+------------------+------------------+
-----------------+-----------------+-----------------+-----------------+------------------+------------------+-----------------+------------------+------------------+
```

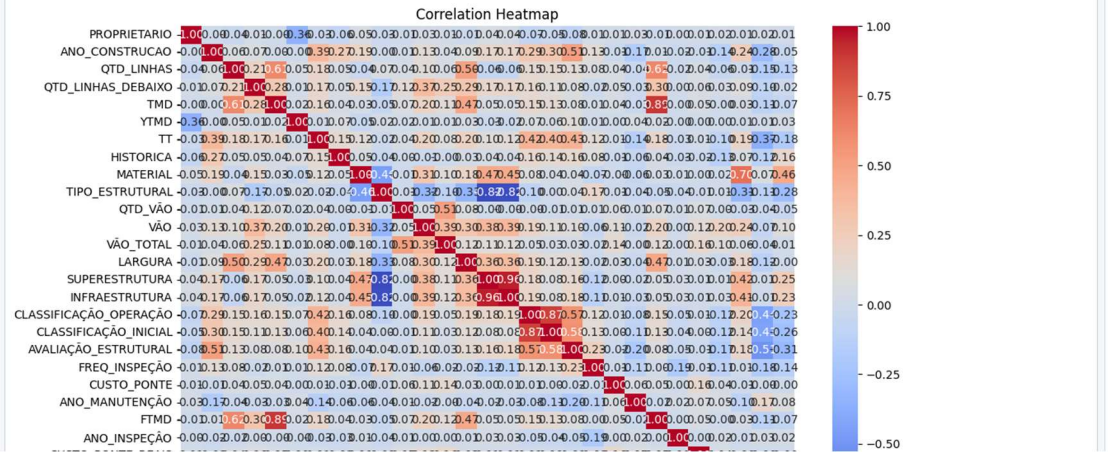
10:49 PM (<1s) 30

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

10:49 PM (10s) 31

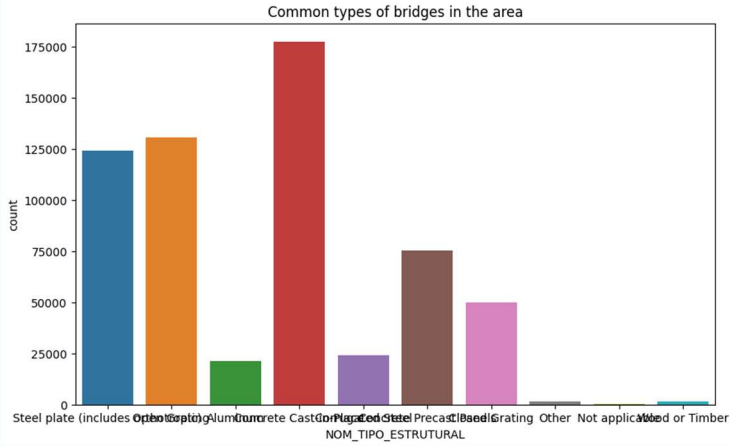
```
plt.figure(figsize=(12, 8))
sns.heatmap(df_bridge_gold_pd.corr(), cmap='coolwarm', annot=True, fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
/root/.ipykernel/238720/command-3653661460975371-1385371033:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to
False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df_bridge_gold_pd.corr(), cmap='coolwarm', annot=True, fmt=".2f")
```



10:49 PM (1s) 32 Python

```
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 desenvolvido uma análise com respostas adequadas.