

1. MVP2 – Engenharia de Dados

1.1 Objetivo

A finalidade deste projeto é, através da utilização de dados abertos do Portal da Transparência do Governo Federal, responder às seguintes perguntas:

- Quais são os “top 10” órgãos públicos que mais gastaram no cartão corporativo em janeiro/2024;
- Quais são os “top 10” servidores que mais gastaram no cartão corporativo em janeiro/2024, por unidade organizacional (UORG)
- Quanto e quais foram os órgãos públicos que gastaram mais do que a média do mês de janeiro/2024;
- Houve alguma compra no cartão corporativo efetuada com empresa punida e/ou sancionada por algum órgão público, isto é, presente no Cadastro de Empresas Inidôneas e/ou Suspensas (CEIS).

Essas são categorias de questões muito presentes em qualquer negócio atual, que visam melhorar a gestão e os procedimentos de *compliance* nas organizações.

1.2 Coleta, carga e modelagem de dados

A carga inicial de dados foi feita a partir de arquivos **.csv** baixados do Portal da Transparência (Dados Abertos). Devido à dificuldade em fazer *web scraping* através do Databricks, esses dados foram baixados e colocados diretamente no DBFS do ambiente, tendo sido criadas tabelas para eles.

Criar nova tabela

Origem de dados ⓘ

Carregar arquivo S3 Outras origens de dados

Diretório de destino do DBFS ⓘ

/FileStore/tables/ (opcional) Selecionar

Todas as pessoas com acesso a este espaço de trabalho podem acessar os arquivos carregados para o DBFS. Saiba mais

Arquivos ⓘ

Arquivo	Tamanho	Ação
202401_Cada.csv	0.4 GB	Remover arquivo
202401_CPGF.csv	2.7 MB	Remover arquivo
202401_Remu.csv	0.2 GB	Remover arquivo

✓ Arquivo carregado para /FileStore/tables/202401_CPGF.csv
✓ Arquivo carregado para /FileStore/tables/202401_Remuneracao.csv
✓ Arquivo carregado para /FileStore/tables/202401_Cadastro.csv

Criar tabela com IU Criar tabela no notebook ⓘ

Esta imagem é apenas exemplificativa, pois outros arquivos foram adicionados e removidos posteriormente. Esses arquivos foram considerados a camada “bronze”, a camada intermediária, com manipulações feitas utilizando PySpark, a “prata” e a tabela persistida, a camada “ouro”.

1.2.1 ETL Servidores Federais

O primeiro processo de Extração, Transformação e Carga (Loading) foi sobre os dados dos servidores federais. A primeira tabela tratada foi a de “Cadastro de Servidores”. Esta tabela possuía originalmente 43 colunas com dados não normalizados, isto é, não atômicos. A imagem a seguir ilustra a situação:

	Id_SERVIDOR_PORTAL	NOME	CPF	MATRICULA	DESCRICAO_CARGO	CLASSE_CARGO
1	3174964	AARAO CARLOS LUZ MACAMBIRA	***.017.623-...	016****	BIBLIOTECARIO-DOCUMENTALISTA	E
2	2903139	AARAO FERREIRA LIMA NETO	***.116.132-...	014****	Sem informaç	null
3	2903139	AARAO FERREIRA LIMA NETO	***.116.132-...	014****	PROFESSOR DO MAGISTERIO SUPERIOR	7
4	2868174	AARAO MEIR SERRUYA	***.693.832-...	033****	FISIOTERAPEUTA - 30H	S
5	3137242	AARAO PEREIRA DE ARAUJO JUNIOR	***.031.184-...	002****	PROFESSOR ENS BASICO TECN TECNOLOGICO	D

Pode-se perceber que há nomes e IDs duplicados, uma vez que o mesmo servidor pode mudar de cargo no mesmo órgão ou mesmo para órgãos diferentes. Diante disso, através de script Python, foi possível criar um dataframe de servidores, somente com id no portal, nome e CPF, sem duplicações:

```
df_cad_servidores = df[["id_SERVIDOR_PORTAL", "NOME",
"CPF"]].drop_duplicates(subset=["id_SERVIDOR_PORTAL"])
```

A partir desses dados, foi criada a tabela “cadastro_servidores” que poderá ser acessada via SQL:

```
permanent_table_name = "cadastro_servidores"
df_cad_servidores.write.format("parquet").saveAsTable(permanent_table_name)
```

Observação: Como foi utilizado o Databricks Community, toda vez que havia inatividade, um novo recurso de *compute* deveria ser criado. Sendo assim, foi necessário utilizar o código abaixo para criar tabelas pela segunda vez:

```
permanente_table_name = "cadastro_servidores"
df_cad_servidores.write.mode("overwrite").saveAsTable(permanent_table_name)
```

Consultando a tabela, restringindo-a pelo ID do exemplo (2903139):

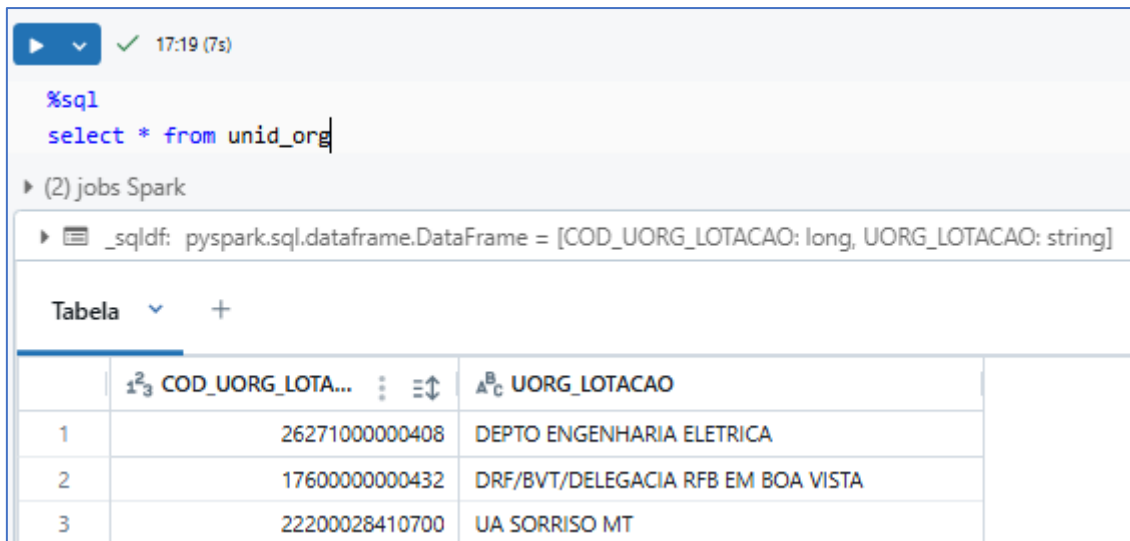
```
%sql
select * from cadastro_servidores where Id_SERVIDOR_PORTAL = '2903139'
```

	id_SERVIDOR_PORTAL	NOME	CPF
1	2903139	AARAO FERREIRA LIMA NETO	***.116.132-...

Outros campos também contém duplicações desnecessárias, como por exemplo, a Unidade Organizacional. Existe o código desta unidade e sua descrição. Também foi criada uma tabela à parte somente com esses dados e, posteriormente, eliminada da tabela original, a descrição, mantendo apenas o código.

```
df_uorgs = df[["COD_UORG_LOTACAO", "UORG_LOTACAO"]].drop_duplicates()
```

```
permanent_table_name = "unid_org"
df_uorgs.write.format("parquet").saveAsTable(permanent_table_name)
```



The screenshot shows a Databricks notebook interface. At the top, there's a play button and a checkmark with the text '17:19 (7s)'. Below that, a SQL query is entered: `%sql select * from unid_org`. Under the query, it says '(2) jobs Spark'. Below that, a message indicates the schema: `_sqldf: pyspark.sql.dataframe.DataFrame = [COD_UORG_LOTACAO: long, UORG_LOTACAO: string]`. At the bottom, there's a table view titled 'Tabela' with a dropdown arrow and a plus sign. The table has two columns: 'COD_UORG_LOTACAO' and 'UORG_LOTACAO'. It contains three rows of data.

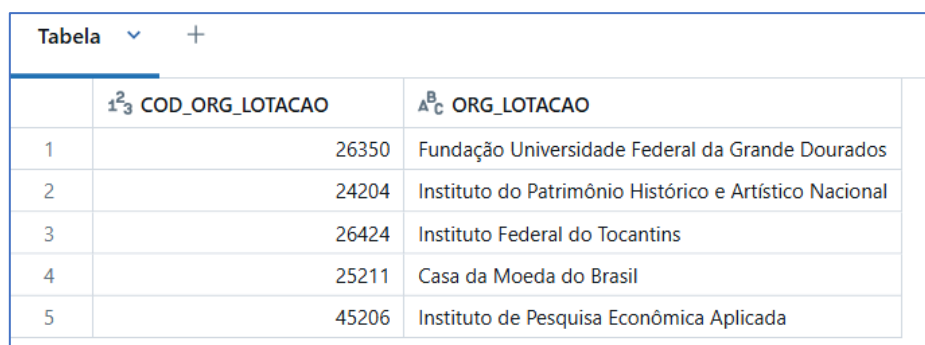
	¹ ₃ COD_UORG_LOTACAO	^A _C UORG_LOTACAO
1	26271000000408	DEPTO ENGENHARIA ELETRICA
2	17600000000432	DRF/BVT/DELEGACIA RFB EM BOA VISTA
3	22200028410700	UA SORRISO MT

Há também a tabela de Órgãos de Lotação, que mostra o código do órgão federativo e sua respectiva descrição. Esses dados também estão aninhados em apenas uma tabela. Aplicou-se a mesma transformação das tabelas anteriores:

```
# separar organização lotação
df_org_lot = df[["COD_ORG_LOTACAO", "ORG_LOTACAO"]].drop_duplicates()
```

Persistir a tabela **org_lotacao**

```
# criar tabela de unidades organizacionais
permanent_table_name = "org_lotacao"
# df_org_lot.write.format("parquet").saveAsTable(permanent_table_name)
df_org_lot.write.mode("overwrite").saveAsTable(permanent_table_name)
```



The screenshot shows a Databricks notebook interface. At the top, there's a table view titled 'Tabela' with a dropdown arrow and a plus sign. The table has two columns: 'COD_ORG_LOTACAO' and 'ORG_LOTACAO'. It contains five rows of data.

	¹ ₃ COD_ORG_LOTACAO	^A _C ORG_LOTACAO
1	26350	Fundação Universidade Federal da Grande Dourados
2	24204	Instituto do Patrimônio Histórico e Artístico Nacional
3	26424	Instituto Federal do Tocantins
4	25211	Casa da Moeda do Brasil
5	45206	Instituto de Pesquisa Econômica Aplicada

Os Órgãos Poder Executivo Federal também têm seus respectivos órgãos superiores. Uma tabela semelhante à anterior foi criada **org_lotacao_superior**:

```
permanent_table_name = "org_lotacao_superior"
# df_org_sup.write.format("parquet").saveAsTable(permanent_table_name)
df_org_sup.write.mode("overwrite").saveAsTable(permanent_table_name)
```

```
%sql
select * from org_lotacao_superior LIMIT 5
```

► (1) jobs Spark

► _sqlidf: pyspark.sql.dataframe.DataFrame = [COD_ORGSUP_LOTACAO: integer, ORGSUP_LOTACAO]

Tabela ▾ +

	1 ² ₃ COD_ORGSUP_LOTACAO	A ^B _C ORGSUP_LOTACAO
1	23000	Ministério da Previdência Social
2	15000	Ministério da Educação
3	49200	MINISTERIO DOS TRANSPORT...
4	17600	MINISTERIO DA FAZENDA
5	40200	MINISTERIO DAS CIDADES

Após todas as transformações, inclusive eliminação de colunas que não faziam muito sentido para análise proposta, o dataframe ficou com 10 colunas residuais. A partir deste dataframe também foi criada a tabela de qualificação dos servidores (qualificacao_servidores):

```
permanent_table_name = "qualificacao_servidores"
# df.write.format("parquet").saveAsTable(permanent_table_name)
df.write.mode("overwrite").saveAsTable(permanent_table_name)
```

Tabela ▾ +

	1 ² ₃ Id_SERVIDOR_PORTAL	A ^B _C DESCRICAO_CARGO	A ^B _C FUNCAO
1	3174964	BIBLIOTECARIO-DOCUMENTALISTA	Sem informação
2	2903139	Sem informaç	FUNCAO GRATIFICADA - II
3	2903139	PROFESSOR DO MAGISTERIO SUPERIOR	Sem informação
4	2868174	FISIOTERAPEUTA - 30H	Sem informação
5	2127242	PROFESSOR ENS BASICO TECN TECNOLOGICO	Sem informação

Para efeitos de teste, podemos tentar obter, por exemplo, o nome, a função e o órgão de lotação dos servidores com o cargo “TECNICO DO SEGURO SOCIAL” que **não estão** lotados no Instituto Nacional de Seguridade Social (INSS):

```
%sql
SELECT a.NOME, b.FUNCAO, c.ORG_LOTACAO
FROM cadastro_servidores a
JOIN qualificacao_servidores b ON a.Id_SERVIDOR_PORTAL = b.Id_SERVIDOR_PORTAL
JOIN org_lotacao c ON b.COD_ORG_LOTACAO = c.COD_ORG_LOTACAO
WHERE b.DESCRICAO_CARGO = 'TECNICO DO SEGURO SOCIAL' AND c.ORG_LOTACAO <>
'Instituto Nacional do Seguro Social'
```

O resultado é o que segue:

	A ^B _C NOME	A ^B _C FUNCAO	A ^B _C ORG_LOTACAO
1	ROBERTO CARLOS DE ARAUJO	Sem informação	MINISTERIO DA FAZENDA
2	SIDNEY CESAR FERNANDES	Sem informação	MINISTERIO DA FAZENDA
3	VALERIA SILVA FERREIRA DA COSTA	Sem informação	MINISTERIO DA FAZENDA
4	ALESSANDRA PEREIRA DE PAULA CARDOSO	Sem informação	MINISTERIO DA FAZENDA
5	BRASILIANO DIAS DOS SANTOS	Sem informação	MINISTERIO DA FAZENDA

Retorno: 625 linhas

Com relação à tabela de qualificação dos servidores, seria possível aproxima-la ainda mais da FN3, criando outras tabelas para Função e Atividades, uma vez que esses dados são dinâmicos e têm relação de 1 para muitos, para cada servidor; por exemplo: o mesmo servidor pode ocupar N funções ao longo do tempo ou trabalhar em N atividades.

1.2.2 ETL CEIS

Também foi carregado o arquivo de cadastro de empresas punidas (CEIS). Este arquivo também sofreu algumas transformações, como alteração do nome das colunas e exclusão de algumas delas, além de ter sido dividido em duas tabelas. A primeira, uma espécie de cadastro de empresas, contendo apenas CNPJ, Nome e Razão Social e a outra, apenas o CNPJ e informações sobre as sanções aplicadas às empresas. Este foi o resultado:

```
permanent_table_name = "cadastro_empresas"
# df_empresas.write.format("parquet").saveAsTable(permanent_table_name)
df_empresas.write.mode("overwrite").saveAsTable(permanent_table_name)
```

Tabela **cadastro_empresas**

	A ^B _C cnpj	A ^B _C nome	A ^B _C razao_social
1	251812980001...	LEALMAQ - LEAL MÁQUINAS LTDA	LEALMAQ - LEAL MAQUINAS LTDA
2	242623160001...	STAFF MEDICAL DISTRIBUIDORA LTDA	STAFF MEDICAL DISTRIBUIDORA LTI
3	679808620001...	ASSOCIAÇÃO LUZ DO MUNDO	ASSOCIACAO LUZ DO MUNDO

Temos, então, uma tabela contendo dados exclusivos, isto é, sem repetições.

A outra tabela ficou apenas com os dados da dimensão “sanções”, mantendo um vínculo com a tabela **cadastro_empresas** pelo CNPJ da empresa sancionada.

```
permanent_table_name = "ceis"
# df.write.format("parquet").saveAsTable(permanent_table_name)
df.write.mode("overwrite").saveAsTable(permanent_table_name)
```

Tabela **ceis**

	A ^B _C n_processo	A ^B _C categ_sancao	A ^B _C dt_in_san
1	00032223320096200197	Impedimento/proibição de contratar com prazo determina...	21/06/2019
2	23314.000919.2022	Impedimento/proibição de contratar com prazo determina...	24/10/2022
3	0200170175165	Declaração de Inidoneidade com prazo determinado	16/03/2018
4	579/2022	Suspensão	18/11/2022
5	64482004985202354	Impedimento/proibição de contratar com prazo determina...	06/10/2023

1.2.3 ETL Cartão de Pagamentos do Governo Federal

Após upload dos dados de cartão corporativo de janeiro/2024 (202401_CPGF.csv), percebemos que há diversos campos descritivos redundantes, uma vez que esses dados já estão em tabelas separadas.

	A ^B _C CÓDIGO ÓRGÃO SUPERIOR	A ^B _C NOME ÓRGÃO SUPERIOR	A ^B _C CÓDIGO ÓRGÃO	A ^B _C NOME ÓRGÃO
1	63000	Advocacia-Geral da União	63000	Advocacia-Geral da União - Unidades com vínculo direto
2	63000	Advocacia-Geral da União	63000	Advocacia-Geral da União - Unidades com vínculo direto
3	63000	Advocacia-Geral da União	63000	Advocacia-Geral da União - Unidades com vínculo direto
4	63000	Advocacia-Geral da União	63000	Advocacia-Geral da União - Unidades com vínculo direto
5	63000	Advocacia-Geral da União	63000	Advocacia-Geral da União - Unidades com vínculo direto
6	63000	Advocacia-Geral da União	63000	Advocacia-Geral da União - Unidades com vínculo direto
7	63000	Advocacia-Geral da União	63000	Advocacia-Geral da União - Unidades com vínculo direto

Após tratamento, a tabela fica da seguinte forma:

	A ^B _C CÓDIGO ÓRGÃO SUPERIOR	A ^B _C CÓDIGO ÓRGÃO	A ^B _C CÓDIGO UNIDADE GESTORA	A ^B _C ANO EXTRATO	A ^B _C MÊS EXTRATO	A ^B _C CPF PORTADOR
1	63000	63000	110161	2024	01	***.562.861-**
2	63000	63000	110161	2024	01	***.195.852-**
3	63000	63000	110161	2024	01	***.195.852-**
4	63000	63000	110161	2024	01	***.195.852-**
5	63000	63000	110161	2024	01	***.195.852-**
6	63000	63000	110161	2024	01	***.195.852-**
7	63000	63000	110161	2024	01	***.212.021-**

Outras transformações: coluna de valor, armazenada como texto, dificultando processos de agregação (soma, média):

Transformar do Spark Data Frame para Pandas Dataframe para utilizar a função Lambda

```
df = df.to_pandas_on_spark()
df["VALOR TRANSAÇÃO"] = df["VALOR TRANSAÇÃO"].apply(lambda x:
float(str(x).replace(",",".")))
```

Para persistir a tabela no DBFS:

```
permanent_table_name = "fato_cpgf"
# df.write.format('parquet').saveAsTable(permanent_table_name)
df.write.mode("overwrite").saveAsTable(permanent_table_name)
```

Podemos considerar essa uma tabela **Fato**, pois ela contém os valores de cada transação ocorrida.

1.2.4 Organização dos Notebooks e Metadados

Para cada arquivo manipulado foi gerado um *notebook* de ETL, a fim de manter a organização e facilitar a depuração dos códigos futuramente. Todos esses notebooks estão disponíveis no projeto público do **Git Hub**.

Foi ainda criado um notebook para adicionar comentários às tabelas, melhorando, assim, a compreensão dos dados, metadados. Nem todos os campos receberam comentários.

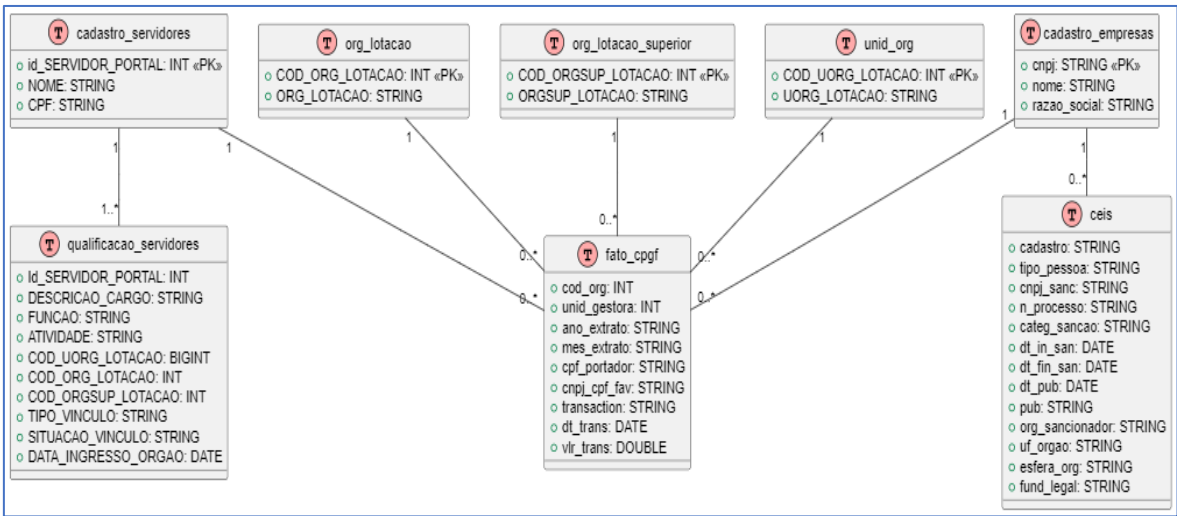
Portal da Transparência				
			Compartilhar	Criar
Nome	Tipo	Proprietário	Criada às	
ETL Cartão de Pagamentos do Governo Federal	Notebook	Thiago Nascimento	2024-07-05 14:52:41	
ETL CEIS	Notebook	Thiago Nascimento	2024-07-08 13:36:44	
ETL Servidores Federais	Notebook	Thiago Nascimento	2024-06-22 22:44:04	
Metadados	Notebook	Thiago Nascimento	2024-07-09 20:04:44	

Metadados das tabelas (exemplo tabela cadastro_servidores)

```
spark.sql("DESCRIBE cadastro_servidores").show()

+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| id_SERVIDOR_PORTAL | int | id do servidor |
| NOME | string | Nome completo do ... |
| CPF | string | Código CPF do ser... |
+-----+-----+-----+
```

Ao final do processo, ficamos com o seguinte esquema:



Esquema renderizado com a extensão PlantUML, do Visual Studio Code

A esse esquema ainda poderia ser adicionada uma tabela de calendário (dimensão data), a fim de abranger todas as possibilidades de exibição da informação, ao longo do tempo (visão por mês, ano, trimestre etc).

1.3 Análise

Após todas as transformações, já conseguimos responder às perguntas propostas. A primeira: quais são os 10 órgãos da administração federal que mais gastaram em jan/24 com cartão corporativo?

```
%sql
SELECT
    fc.cod_org, ol.ORG_LOTACAO,
    ROUND(SUM(vlr_trans), 2) AS total_gasto
FROM
    fato_cpgr fc
INNER JOIN
    org_lotacao ol
ON
    fc.cod_org = ol.COD_ORG_LOTACAO
GROUP BY
    fc.cod_org, ol.ORG_LOTACAO
ORDER BY
    total_gasto DESC
LIMIT 10
```

Tabela			
	1.2 cod_org	2.1 ORG_LOTACAO	1.2 total_gasto
1	20101	Presidência da República	2221202.95
2	25000	Ministério da Saúde	100584.24
3	30802	Departamento de Polícia Rodoviária Federal	67798.53
4	36201	Fundação Oswaldo Cruz	47141.75
5	26429	Instituto Federal de Goiás	46737.75
6	26443	Empresa Brasileira de Serviços Hospitalares	42848.09
7	32396	Agência Nacional de Mineração	38104.58
8	44207	Instituto Chico Mendes de Conservação da Biodiversidade	34163
9	26249	Universidade Federal Rural do Rio de Janeiro	29963.61
10	26263	Universidade Federal de Lavras - MG	28829.16

É importante utilizar a cláusula ORDER BY DESC para que os valores sejam exibidos do maior para o menor, atendendo integralmente à proposição.

Daqui, conseguimos responder à questão “b”: os 10 servidores que mais gastaram com cartão corporativo no mês de janeiro/2024, por Unidade Organizacional (UORG):

```
%sql
SELECT
```



```
cs.NOME, ug.UORG_LOTACAO,
round(sum(fc.vlr_trans), 2) AS total_gasto
FROM
fato_cpgef fc
INNER JOIN
cadastro_servidores cs
ON
fc.cpf_portador = cs.CPF
INNER JOIN
qualificacao_servidores qs
ON
qs.Id_SERVIDOR_PORTAL = cs.id_SERVIDOR_PORTAL
INNER JOIN
unid_org ug
ON
qs.COD_UORG_LOTACAO = ug.COD_UORG_LOTACAO
GROUP BY
cs.NOME, ug.UORG_LOTACAO
ORDER BY
total_gasto DESC
LIMIT 10
```

Tabela + 🔍 ⚙️

	A _C NOME	A _C UORG_LOTACAO	1.2 total_gasto
1	REBECA MEDEIROS FONSECA	CENTRO DE REGISTRO E CONTROLE ACADEMICO	40233.66
2	NISVAL FERREIRA GUIMARAES	DIRECAO GERAL CAMPUS INHUMAS	38000.58
3	RICARDO YANEZ NOGUEIRA	INSTITUTO NAC DE CONTROL QUALID EM SAUDE	27917.96
4	CRISTIANE FRENCH PEREIRA	INSTITUTO DE TECNOLOGIA EM IMUNOBIOLOGIC	27250.36
5	FERNANDO FRANCISCO DE SOUZA	DIRECAO GERAL CAMPUS INHUMAS	24808.42
6	MARCIA MARIA HENGEMUHLE	HOSPITAL UNIVERSITARIO DA UFSM	24260.94
7	LEONARDO LIMA BERGAMINI	GERENCIA DE MEIO AMBIENTE E GEOGRAFIA	21431.98
8	PAULO CESAR DO NASCIMENTO CORREA	COORDENACAO GERAL INFRAESTRUTURA CAMPI	21417.6
9	LUCIANA BANDEIRA DE SOUZA	DIRETORIA DE PLANEJAMENTO	20116.83
10	ANTENOR FONSECA DE OLIVEIRA FILHO	DIVISAO OPERACIONAL	20116.83

A questão “c”, quantos e quais foram os órgãos públicos que gastaram mais do que a média de janeiro/2024, é um pouco mais complicada. Vamos por partes:

Passo 1: criar uma “tabela” apenas para computar a média dos gastos, por órgão:

```
%sql
WITH avgspending AS (
    SELECT
        ROUND(AVG(total_gasto), 2) AS avg_gasto
    FROM (
        SELECT
```

```
        fc.cod_org,  
        ol.ORG_LOTACAO,  
        SUM(fc.vlr_trans) AS total_gasto  
FROM  
    fato_cpgef fc  
INNER JOIN  
    org_lotacao ol  
ON  
    fc.cod_org = ol.COD_ORG_LOTACAO  
GROUP BY  
    fc.cod_org,  
    ol.ORG_LOTACAO  
)  
)
```

A variável *avgspending* contém a média dos gastos, agrupada por órgão.

Passo 2: obter quais foram os órgãos cujo gasto superou a média calculada no passo anterior

```
SELECT  
    fc.cod_org,  
    ol.ORG_LOTACAO,  
    ROUND(SUM(fc.vlr_trans), 2) AS total_gasto  
FROM  
    fato_cpgef fc  
INNER JOIN  
    org_lotacao ol  
ON  
    fc.cod_org = ol.COD_ORG_LOTACAO  
GROUP BY  
    fc.cod_org,  
    ol.ORG_LOTACAO  
HAVING  
    SUM(fc.vlr_trans) > (SELECT avg_gasto FROM avgspending)  
ORDER BY  
    total_gasto DESC
```

Como foi utilizada a cláusula 'GROUP BY', deve-se utilizar a cláusula HAVING para filtrar os dados. A tabela resultado é:

Tabela ▾ +			
	A ^B _C cod_org	A ^B _C ORG_LOTACAO	1.2 total_gasto
1	20101	Presidência da República	2221202.95
2	25000	Ministério da Saúde	100584.24
3	30802	Departamento de Polícia Rodoviária Federal	67798.53
4	36201	Fundação Oswaldo Cruz	47141.75
5	26429	Instituto Federal de Goiás	46737.75
6	26443	Empresa Brasileira de Serviços Hospitalares	42848.09

Temos que 6 órgãos gastaram mais do que a média, que é de R\$ 38.217,14.

```
SELECT avg_gasto FROM avgspending
```

Para responder à última questão (d), se houve alguma compra no cartão corporativo com empresa punida (CEIS), precisamos consultar as tabelas **ceis** e **fato_cpgef**, vinculando-as pelo CNPJ.

```
%sql
```

```
SELECT
```

```
    ce.cnpj_sanc, cde.razao_social, ce.n_processo,  
    ce.categ_sancao, ce.dt_in_san, cp.dt_trans, ce.dt_fin_san,  
    cp.vlr_trans
```

```
FROM
```

```
    ceis ce
```

```
INNER JOIN
```

```
    fato_cpgef cp
```

```
ON
```

```
    ce.cnpj_sanc = cp.cnpj_cpf_fav
```

```
INNER JOIN
```

```
    cadastro_empresas cde
```

```
ON
```

```
    ce.cnpj_sanc = cde.cnpj
```

Tabela	▼	+								
	A ^B _C cnpj_sanc	A ^B _C razao_social	A ^B _C n_proces...	A ^B _C categ_sancao	A ^B _C dt_in_san	A ^B _C dt_trans	A ^B _C dt_fin_san	1.2 vlr_trans		
1	02458330000150	> MULTISERV - C...	> 02001605...	> Declaração de Inidoneid...	14/02/2020	null	null	930		
2	07358914000178	CEQUIMICA LTDA	> 019.5050....	Suspensão	23/04/2024	null	04/08/2024	2315.9		
3	32602639000133	> LUKATONER SU...	303/23	> Impedimento/proibição ...	09/02/2024	null	09/02/2025	218		

Infelizmente houve pagamentos realizados com o cartão corporativo do Governo Federal a empresas sancionadas. Poderíamos aprofundar a análise para comparar se a data da compra está entre a data de início e término da sanção, porém os dados não estão completos, como destacam os retângulos vermelhos sobre as colunas analisadas. Somente a título de exemplo, esta query poderia ser melhorada da seguinte forma:

```
%sql
```

```
SELECT
```

```
    ce.cnpj_sanc, cde.razao_social, ce.n_processo,  
    ce.categ_sancao, ce.dt_in_san, cp.dt_trans, ce.dt_fin_san,  
    cp.vlr_trans
```

```
FROM
```

```
    ceis ce
```

```
INNER JOIN
```

```
    fato_cpgef cp
```

```
ON
```

```
    ce.cnpj_sanc = cp.cnpj_cpf_fav
```

```
INNER JOIN
```

```
    cadastro_empresas cde
```

```
ON
```

```
ce.cnpj_sanc = cde.cnpj  
WHERE  
date(cp.dt_trans) BETWEEN date(ce.dt_in_san) AND date(ce.dt_fin_san)
```

Esta query não retornou resultados, devido às datas de transação estarem com valor “null”, mas, considerando o preenchimento adequado desses campos, com essa adaptação, a query retornaria somente os casos em que a compra ocorreu dentro do período em que a empresa esteve sancionada, gerando, dessa forma, indícios mais fortes de que ocorrera uma transação alheia ao interesse público.

1.4 Autoavaliação

O trabalho se propôs a responder quatro perguntas do negócio a respeito de gastos com cartão corporativo no mês de janeiro de 2024, envolvendo aspectos quantitativos e qualitativos. Os resultados ocorreram conforme o esperado, uma vez que as perguntas foram devidamente respondidas com os dados. Cabe ressaltar que a utilização do Databricks na versão “Community” limitou um pouco as possibilidades para este MVP. Uma das limitações foi a de não poder utilizar o *pipeline*, uma ferramenta na qual os notebooks podem ser agendados e executados de maneira orquestrada. Outra limitação observada foi a desativação dos ambientes de *compute* sempre após um período maior ou igual a 60 minutos de inatividade. Com essa desativação, as tabelas permanentes eram apagadas do DBFS, sendo necessário criar novo ambiente e executar novamente os notebooks.

Diante dessas ponderações, acredito que o MVP2 – Engenharia de Dados conseguiu cumprir o objetivo, considerando os recursos disponíveis.