Relatório de Case

Nome: Alef Anderson Fernandes Clarindo da Silva

Data: 02/06/2025

Vaga: Engenheiro de Dados Pleno

Empresa: Telefônica Brasil (Vivo)

Sumário

Introdução

Este projeto foi desenvolvido com o objetivo de demonstrar habilidades em manipulação, transformação e análise de dados utilizando ferramentas amplamente adotadas na área de dados.

O desafio proposto consistiu em integrar e transformar três bases de dados distintas (funcionarios.csv, cargos.csv e estruturas.csv) respeitando os períodos de vigência de cargos e estruturas, a fim de consolidar as informações em um único DataFrame. A partir da base unificada, foi possível gerar indicadores relevantes, como a quantidade de admissões e desligamentos por mês e do total de funcionários ativos por mês.

Para a execução do projeto, foram utilizadas as seguintes ferramentas:

- **Jupyter Notebooks-7.0.3:** Ambiente para desenvolvimento e documentação do código.
- **Python-3.10.12:** Linguagem principal para manipulação e análise dos dados.
- **Pandas-2.2.3:** Biblioteca utilizada manipulação de dados em DataFrames.
- **Docker**: Utilizado para garantir a portabilidade e reprodutibilidade do ambiente de desenvolvimento.

Etapas Realizadas

1. Preparar o Ambiente

A primeira etapa realizada foi de preparação do ambiente. Aproveitei a oportunidade para aprender mais sobre containers e optei por utilizar a ferramenta Docker. Com o Docker, foi possível encapsular todas as dependências e configurações necessárias em um contêiner, evitando problemas de incompatibilidade entre diferentes sistemas operacionais ou versões de bibliotecas.

Além disso, o **Docker facilita a integração com ambientes em nuvem**, como o Azure que é amplamente utilizado na Telefônica, permitindo que os projetos sejam facilmente escalados ou migrados na infraestrutura.

O arquivo *"Dockerfile"* possui as instruções da imagem utilizada para replicar o ambiente do case.

2. Extrair e Identificar os Dados

A segunda etapa do projeto consistiu na importação da biblioteca Pandas e leitura dos três arquivos CSV fornecidos: *funcionarios.csv*, *cargos.csv* e *estruturas.csv*. Utilizando a função *"pandas.read_csv()"* do Pandas, os dados foram carregados sem dificuldades em DataFrames, permitindo uma manipulação eficiente, estruturada e possibilitar a visualização dos dados nos outputs do notebook para a primeira análise dos dados obtidos.

3. Análise Exploratória

O mais importante nesse momento foi analisar todos os dados obtidos, nome, colunas, informações, tipos de dados, tamanho das tabelas, dados faltantes, informações inconsistentes ou duplicadas, dados de data e quais são as chaves primárias. Assim pude entender o que fazer nas etapas seguintes para chegar no resultado esperado.

	id_funcionario	id_cargo	id_estrutura	estado	data_admissao	data_desligamento
0	1	16	6	São Paulo	2026-01-12	NaN
1	2	11	10	Santa Catarina	2025-05-22	NaN
2	3	1	3	São Paulo	2024-07-05	NaN
3	4	1	8	São Paulo	2024-12-01	NaN
4	5	2	5	Paraná	2024-12-13	NaN
...	...	...	...	...	...	...
95	96	12	2	São Paulo	2024-12-30	NaN
96	97	14	4	Santa Catarina	2025-04-20	NaN
97	98	7	9	São Paulo	2024-09-28	NaN
98	99	12	1	São Paulo	2025-04-19	2025-11-30
99	100	14	10	São Paulo	2026-01-04	NaN

100 rows × 6 columns

4. Limpeza e Tratamento dos Dados

Nesse processo, após ter um entendimento das tabelas, segui para o passo de converter as colunas que continham datas para o tipo “datetime” utilizando a função *“pandas.to_datetime()”.* Antes, criei uma lógica de “for” para percorrer por todos os dataframes e todas as colunas para aplicar a função nas colunas necessárias. Após isso, confirmei se os dados foram devidamente alterados para manipulação posterior.

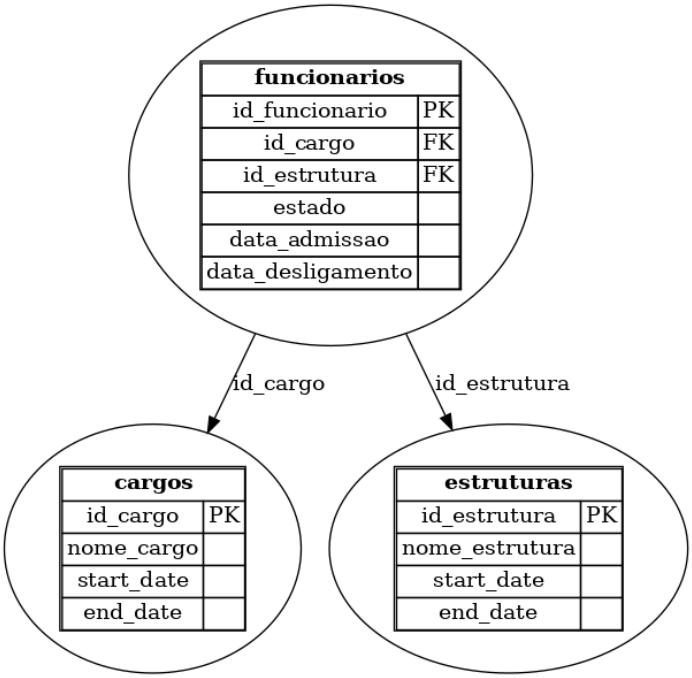
```
# Transforma o tipo de todas as colunas de data em todos os dataframes para o formato 'datetime'
for df in df_list:

    # Obter todas os nomes das colunas do dataframe atual
    df_columns = df.columns.tolist()

    for column in df_columns:
        # Verifica se o nome da atual coluna possui na lista de 'date_columns'
        if column in date_columns:
            # Transforma a coluna atual para o tipo 'datetime'
            df[column] = pd.to_datetime(df[column], errors='coerce')

✓ 0.0s
```

5. Relacionamento Entre as Bases

Após uma validação das chaves primárias e estrangeiras, pude seguir com a união dos três dataframes. Utilizei a função *“pandas.merge()”* para unir no primeiro momento, as tabelas “*funcionários*” (No qual é a tabela com mais linhas) e “*cargos*” (Segunda maior planilha). Foi importante utilizar nessa função o parâmetro “*how=left*” para que os dados da primeira tabela sejam mantidos independente de possuir um equivalente na segunda tabela. A chave primária foi a coluna *id_cargos* para união das informações.

Assim que os filtros foram aplicados e os dados transformados (Etapa mencionada a seguir), realizei o mesmo processo de *merge()* com a nova tabela gerada e o dataframe de “*estruturas*”. Dessa vez foi utilizado *id_estruturas* como chave primária.

Em ambos os momentos, foi importante conferir os novos dataframes gerados com as junções realizadas para garantir a integridade dos dados e que não houve nenhuma perda de informação valiosa.



6. Transformação e Enriquecimento

Após cada junção, eu realizei a filtragem temporal exigida na descrição do case. Com muita análise, optei por preencher os valores ausentes de *data_desligamento* com a data final de vigência do cargo (*end_date*). Essa decisão foi tomada com base na premissa de que, na ausência de uma data de desligamento explícita, o vínculo do funcionário com aquele cargo deve ser considerado ativo até o fim do período de validade do cargo.

Assim, filtrei os dados para garantir que o período de admissão e desligamento do funcionário estivesse dentro da faixa de tempo de *end_date* e *start_date* dos cargos e estruturas.

Finalizada a filtragem, agora o novo dataframe agora possuía um total de 26 funcionários dentro dos períodos de vigência dos cargos e estruturas.

7. Limpeza e Carregamento dos Dados Finais

Para finalizar a manipulação dos dados obtidos, executei a remoção das colunas *start_date* e *end_date* pois já não eram mais necessários para o dataframe final com as três bases unidas e reorganizei as colunas para uma melhor visualização. Por fim, realizei o carregamento do dataframe para o formato csv na pasta do projeto.

	id_funcionario	data_admissao	data_desligamento	estado	id_cargo	nome_cargo	id_estrutura	nome_estrutura
2	3	2024-07-05	2024-12-31	São Paulo	1	Analista de Banco de Dados	3	GER TECNOLOGIA DA INFORMAÇÃO
3	4	2024-12-01	2024-12-31	São Paulo	1	Analista de Banco de Dados	8	GER VENDAS
4	5	2024-12-13	2025-01-31	Paraná	2	Analista de Suporte	5	GER FINANCEIRO
5	6	2024-10-22	2025-02-28	Santa Catarina	3	Analista de Segurança da Informação SR	7	GER OPERAÇÕES
11	13	2024-10-12	2025-05-31	São Paulo	6	Engenheiro de Software	10	GER DESENVOLVIMENTO DE NEGÓCIOS
12	14	2024-05-15	2025-01-31	São Paulo	2	Analista de Suporte	6	GER MARKETING
19	21	2025-01-24	2025-08-31	São Paulo	9	Gerente de TI	10	GER DESENVOLVIMENTO DE NEGÓCIOS
24	27	2024-11-07	2024-12-31	Paraná	1	Analista de Banco de Dados	2	GER INTELIGENCIA COMERCIAL B2B
28	32	2025-01-04	2025-01-31	Rio de Janeiro	2	Analista de Suporte	7	GER OPERAÇÕES
33	37	2024-12-19	2024-12-31	São Paulo	1	Analista de Banco de Dados	8	GER VENDAS
40	44	2024-08-31	2025-03-31	São Paulo	4	Desenvolvedor de Software	6	GER MARKETING
43	47	2024-06-29	2024-12-31	São Paulo	1	Analista de Banco de Dados	4	GER RECURSOS HUMANOS

8. Criação dos indicadores

Para a etapa final, a partir dos dados formatados, foram calculados os principais indicadores de gestão de pessoas, adicionado duas novas colunas para classificar apenas os meses de admissão e desligamento. Com esses dados, foi possível calcular o total de funcionários admitidos e desligados desde o primeiro mês registrado até o último. Também foi possível criar a tabela da quantidade de funcionários ativos no período total. Foi utilizado o formato *pandas.Series()* para os indicadores.

Com os três indicadores definidos, decidi juntar todas as tabelas para um único dataframe visto que o index *meses* estava presente igualmente nas três tabelas. Por fim, exportei os dados para CSV. Dessa forma acredito que a visualização dos indicadores em uma tabela unificada iria facilitar a visualização e análise para a equipe de analytics.

	mes	ativos	admitidos	desligados
0	2024-03	1	1	0
1	2024-04	1	0	0
2	2024-05	2	1	0
3	2024-06	4	2	0
4	2024-07	7	3	0
5	2024-08	9	2	0
6	2024-09	12	3	0
7	2024-10	14	2	0
8	2024-11	15	1	0
9	2024-12	19	4	7
10	2025-01	15	3	5
11	2025-02	11	1	2
12	2025-03	9	0	5
13	2025-04	5	1	1
14	2025-05	6	2	1
15	2025-06	5	0	2
16	2025-07	3	0	1
17	2025-08	2	0	2

Desafios Encontrados

1. **Operações de Data/Tempo no Pandas:** Inicialmente não conhecia algumas operações com o tipo data do Pandas. Principalmente no momento de criar os cálculos e dados dos indicadores.

2. **Informações nulas:** A etapa para executar os filtros de tempo foi com certeza o momento mais desafiador do projeto. O principal desafio encontrado foi a ausência de valores na coluna *data_desligamento* de diversos funcionários. Essa informação era essencial para aplicar corretamente os filtros de vigência dos cargos e das estruturas, já que o objetivo era garantir que os vínculos estivessem ativos dentro dos períodos definidos por *start_date* e *end_date*.

Soluções Adotadas

1. **Operações de Data/Tempo no Pandas:** Consegui pesquisar as funções para desenvolvimento e manipulação dos cálculos de tempo com a ajuda da IA ChatGPT4 e a documentação oficial do Pandas.

   Link: https://pandas.pydata.org/docs/reference/api/pandas.date_range.html

2. **Informações nulas:** Diante desse cenário, considerei algumas alternativas:

   - **Remover os registros com data_desligamento nula**, o que iria comprometer a análise ao descartar funcionários ainda ativos e perder muitos dados.
   - **Assumir a data atual como data de desligamento**, o que poderia gerar distorções.
   - **Manter o valor nulo e adaptar os filtros**, o que poderia afetar a consistência da análise e das regras de filtro definidas.

Após avaliar os impactos de cada abordagem, optei por **preencher os valores ausentes de *data_desligamento* com a data final de vigência do cargo (*end_date*)**. Essa decisão foi tomada com base na premissa de que, na ausência de uma data de desligamento explícita, o vínculo do funcionário com aquele cargo deve ser considerado ativo até o fim do período de validade do cargo.

Essa solução permitiu manter os registros relevantes para análise dos indicadores, respeitando os critérios de integridade temporal definidos no desafio.

Porém é importante destacar que, em um cenário real, eu priorizaria uma conversa com o gestor responsável pelo projeto ou com a equipe de analytics para entender melhor o contexto e as necessidades da análise. Essa troca seria fundamental para tomar uma decisão mais alinhada com os objetivos do negócio e evitar interpretações equivocadas.

Resultados e Conclusão

A tarefa da proposta foi concluída com sucesso, atingindo todas as etapas previstas. Como resultado, foram gerados indicadores mensais que permitem visualizar de forma clara:

- A quantidade de **admissões** e **desligamentos** por mês;
- A evolução do número de **funcionários ativos** ao longo do tempo.
- Gerar gráficos valiosos e estratégicos para a gestão de pessoas da organização.

O código desenvolvido foi documentado passo a passo em um Jupyter Notebook, e os dados finais foram exportados em arquivos CSV, prontos para uso em análises futuras ou integração com outras ferramentas. Além disso, aproveitei a proposta do projeto para estudos e pesquisas da ferramenta Docker e da criação de gráficos com Python utilizando as bibliotecas matplotlib e seaborn.

Link GitHub: https://github.com/alef-and/case_employees

Link Docker: https://hub.docker.com/repository/docker/alefand/case_employees_python/general

Gráficos

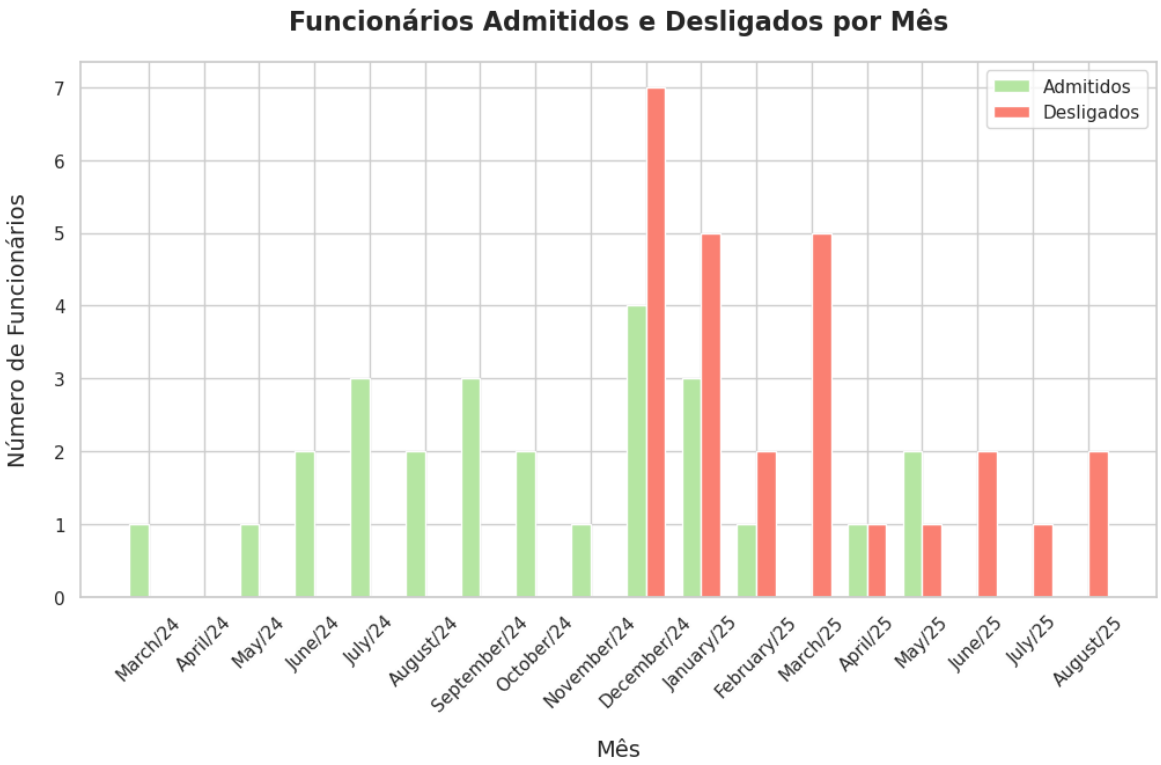


Gráfico 1 – Indicador Funcionários Admitidos e Desligados por Mês

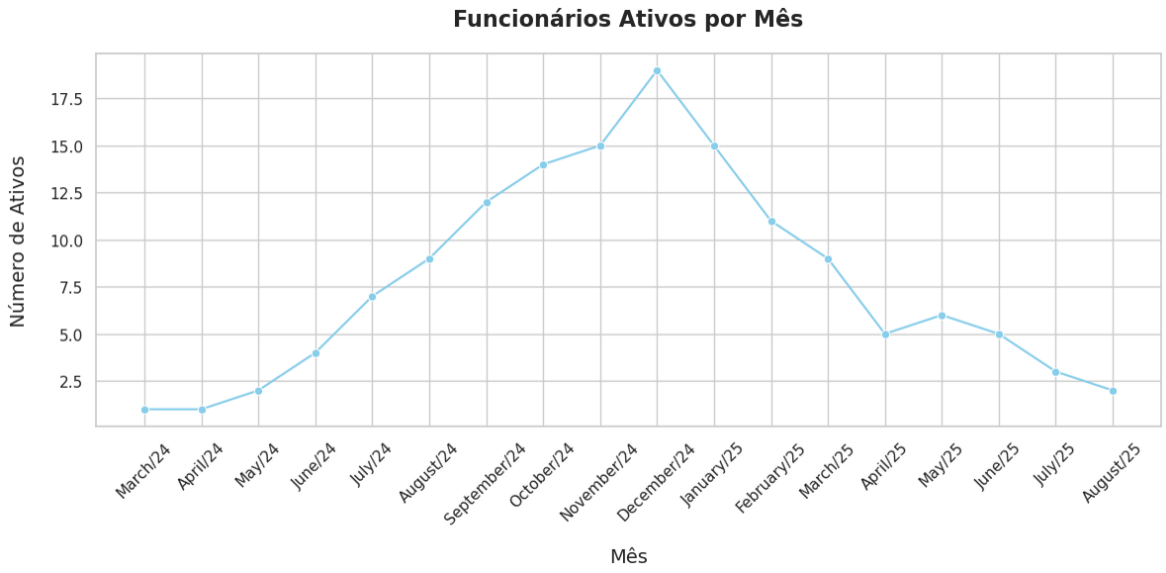


Gráfico 2 – Indicador Funcionários Ativos por Mês