**ESPECIALISTA EM ENGENHARIA DE DADOS – DEVOPS**

1. **Auto avaliação**

Auto-avalie suas habilidades nos requisitos de acordo com os níveis especificados usando o

link abaixo.

Qual o seu nível de domínio nas técnicas/ferramentas listadas abaixo, onde:

• 0, 1, 2 - não tem conhecimento e experiência;

• 3, 4 ,5 - conhece a técnica e tem pouca experiência;

• 6 - domina a técnica e já desenvolveu vários projetos utilizando-a.

**Tópicos de Conhecimento:**

• Manipulação e tratamento de dados com Python: \_6\_

• Manipulação e tratamento de dados com Pyspark: \_6\_

• Desenvolvimento de data workflows em Ambiente Azure com databricks: \_6\_

• Desenvolvimento de data workflows com Airflow: \_5\_

• Manipulação de bases de dados NoSQL: \_4\_

• Web crawling e web scraping para mineração de dados: \_5\_

• Construção de APIs: REST, SOAP e Microservices: \_4\_

1. Desenvolvimento de pipelines de ETL de dados com Python, Apache Airflow, Hadoop e Spark.
   1. Olhando para todos os dados disponíveis na fonte citada acima, em qual estrutura de banco de dados você orienta guardá-los no nosso Data Lake? SQL ou NoSQL? Discorra sobre sua orientação. (1 pts)
      1. Resposta: Opto pelo NoSQL como irei trabalhar com a plataforma DATABRICK, opto na verdade pelas estruturas colunares, trabalhando de forma de colunas, pensando que nossas dados irão crescer e trabalhando com o ecossistemas databricks a própria plataforma se encarrega de estruturar, escalonar e clusterizar a arquitetura, portanto, vou consumir recursos de memórias para gerar os dataframes precisando materializar o mínimo necessário, usando arquivos tipo parquet e no hive .delta, assim customizo mais recursos para processamentos.
      2. Exemplo da estrutura de linha vs coluna: Interface gráfica do usuário

         Descrição gerada automaticamente
   2. Nosso cliente estipulou que necessita de informações apenas sobre as atracações e cargas contidas nessas atracações dos últimos 3 anos (2017- 2019). Logo, o time de cientistas de dados, em conjunto com você, analisaram e decidiram que duas tabelas, uma para atracação e outra para carga, seriam suficientes tanto para o trabalho do Observatório como para trabalho do time externo. Assim, desenvolva um script em python que extraia os dados do anuário, e transforme-os em duas tabelas fato, atracacao\_fato e carga\_fato, com as respectivas colunas abaixo. Lembrando que os dados têm periodicidade mensal, então script’s automatizados e robustos ganham pontos extras. (2 pontos + 1 ponto para solução automatizada e elegante
      1. Resposta: aqu achei estranho pedi informações (2017 - 2019) 3 anos, então optei em pegar informações mais recentes, claro o script vai funcionar para qual anos, mais fiz as cargas mais recentes de do ano atual – 2, assim o sistema ficará automático, mantendo os 3 anos e fazendo o delta dos próximos anos, nada impedi de fazer uma carga full, mais sempre pegando como base o atual -2, portanto, foi criado m notebook para criação das 2 tabelas físicas (Create table atracacao\_fato, Create table carga\_fato), 1 notebook para as cargas (Carga), criei também um notebook para manutenção de pastas fiec e 1 notabook para procedure o tratamento das 2 tabelas materializadas.
         1. Os notebooks estarão no github na paast compactada FIEC.zip
   3. Essas duas tabelas ficaram guardadas no nosso Banco SQL SERVER. Nossos economistas gostaram tanto dos dados novos que querem escrever uma publicação sobre eles. Mais especificamente sobre o tempo de espera dos navios para atracar. Mas eles não sabem consultar o nosso banco e apenas usam o Excel. Nesse caso, pediram a você para criar uma consulta (query) otimizada em sql em que eles vão rodar no excel e por isso precisa ter o menor número de linhas possível para não travar o programa. Eles querem uma tabela com dados do Ceará, Nordeste e Brasil contendo número de atracações, para cada localidade, bem como tempo de espera para atracar e tempo atracado por meses nos anos de 2018 e 2019.
      1. 
      2. Resposta: OBS.: segue com os dados de 2023 – 2 anos, segue o script e se encontra também no notebook query - exercicio 2.c.py

%sql

WITH AtracacoesPorMes AS (

    SELECT

        atracacao\_municipio AS Localidade,

        ano\_inicio\_operacao,

        mes\_inicio\_operacao\_int,

        COUNT(DISTINCT idatracacao) AS nr\_atracacao,

        AVG(tatracado) AS tempo\_medio\_atracado,

        AVG(tesperaatracacao) AS tempo\_medio\_espera\_atracado

    FROM fiec.atracacao\_fato

    GROUP BY atracacao\_municipio, ano\_inicio\_operacao, mes\_inicio\_operacao\_int

)

SELECT

    AtracacoesPorMes.Localidade,

    AtracacoesPorMes.ano\_inicio\_operacao,

    AtracacoesPorMes.mes\_inicio\_operacao\_int,

    AtracacoesPorMes.nr\_atracacao,

    AtracacoesPorMes.tempo\_medio\_atracado,

    AtracacoesPorMes.tempo\_medio\_espera\_atracado,

    AtracacoesPorMes.nr\_atracacao - COALESCE(LAG(AtracacoesPorMes.nr\_atracacao) OVER (

        PARTITION BY AtracacoesPorMes.Localidade

        ORDER BY AtracacoesPorMes.ano\_inicio\_operacao, AtracacoesPorMes.mes\_inicio\_operacao\_int

    ), 0) AS variacao\_numero\_atracacoes

FROM AtracacoesPorMes

ORDER BY AtracacoesPorMes.ano\_inicio\_operacao, AtracacoesPorMes.mes\_inicio\_operacao\_int, AtracacoesPorMes.nr\_atracacao DESC;

1. Criação de ambiente de desenvolvimento com Linux e Docker
   1. Resposta: nessa etapa fiz o uso da plataforma AIRFLOW com DOCKER-COMPOSE, gerando a orquestração de uma DAG construída em python e comunicand0se com DOCKER o AIRFLOW, pegando de ima API da ANTAQ o arquivo em .ZIP do ultimo ano corrente, aqui podemos definir a periodicidade dessa carga, coloquei ela fazendo uma vez por semana, mas claro não tem necessidade dessa periodicidade, portanto, esse processo fará o ETL completo até a disponibilidade para o storage do DBFS do DATABRICKS que foi a plataforma que escolhi para fazer o desenvolvimento do datalike e likehouse nosso entre aspa DW. Segue algumas figuras como documentos desse processo.
   2. FIGURA 1 – ambiente do DOCKER

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

* 1. FIGURA 2 – Ambiente do AIRFLOW pagina inicialInterface gráfica do usuário, Texto, Aplicativo, Email

     Descrição gerada automaticamente
  2. FIGURA 3 – Ambiente do AIFLOW pagina de operações - GráficoTabela

     Descrição gerada automaticamente com confiança baixa
  3. FIGURA 4 – Ambiente do AIFLOW pagina de operações - Details Interface gráfica do usuário, Texto, Aplicativo, Email

     Descrição gerada automaticamente
  4. FIGURA 4 – Ambiente do AIFLOW pagina de operações - Grantt Interface gráfica do usuário, Aplicativo

     Descrição gerada automaticamente
  5. FIGURA 4 – Ambiente do AIFLOW pagina de operações - Code

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

* 1. FIGURA 4 – Ambiente do AIFLOW pagina de operações - Logs Interface gráfica do usuário, Aplicativo

     Descrição gerada automaticamente
  2. O Arquivo em python e o processo todo dessa etapa estará no github com o nome da pasta: AIRFLOW-DOCKER

1. Configuração de pipelines de CI/CD com Gitlab ou Github