Projeto: Democratização de Dados na AWS

Este documento é um guia completo para construir um pipeline de dados na AWS, projetado para extrair dados de um banco de dados relacional (RDS), processá-los e disponibilizá-los para análise de forma otimizada e segura.

Fluxo de Dados

O pipeline seguirá o seguinte fluxo:

1. **AWS RDS (MySQL):** O banco de dados de origem onde os dados transacionais são armazenados.
2. **AWS Lambda (.NET):** Uma função serverless que é acionada para extrair dados do RDS.
3. **Amazon S3:** O Data Lake onde os dados extraídos são armazenados em formato Parquet, particionados por data.
4. **AWS Glue:** Um serviço de ETL que utiliza um Crawler para catalogar os dados no S3, tornando-os "consultáveis".
5. **Amazon Athena:** Um serviço de consulta interativo que permite analisar os dados no S3 usando SQL padrão.
6. **Amazon QuickSight:** Uma ferramenta de Business Intelligence (BI) para criar dashboards e visualizações a partir dos dados consultados no Athena.

Estrutura do Projeto (Repositórios no GitHub)

A organização sugerida para este projeto no GitHub é a seguinte:

```
aws-democratizacao-dados/
├── README.md (Este guia geral)
├── 00-infraestrutura-iam/
│   └── README.md (Instruções para VPC, Subnets, Security Groups e Role IAM)
├── 01-rds-mysql/
│   └── README.md (Criação do banco, scripts SQL para tabelas e população de dados)
├── 02-lambda-net-parquet/
│   └── README.md (Código .NET, build, deploy e configuração da função)
├── 03-glue-crawler/
│   └── README.md (Como criar e configurar o Crawler)
├── 04-athena-queries/
│   └── README.md (Configuração do Athena e queries de validação)
└── 05-quicksight-dashboard/
    └── README.md (Como conectar a fonte de dados e criar o dashboard)
```

00 - Infraestrutura e Permissões (VPC e IAM)

<details>
<summary>Clique para expandir:
00-infraestrutura-iam/README.md</summary>
Passo a Passo: Configurando a Base da Infraestrutura

Antes de criar os serviços, precisamos de uma rede segura (VPC) para eles operarem e de uma permissão (IAM Role) para que possam se comunicar.

1. Criação da VPC (Virtual Private Cloud)

A VPC é sua rede privada na nuvem. O RDS ficará em uma sub-rede privada para não ser acessível pela internet, e a Lambda também rodará na VPC para acessar o RDS.

1. **Acesse o Console da AWS** e procure por **"VPC"**.
2. No painel da VPC, clique em **"Criar VPC"**.
3. Selecione a opção **"VPC e mais"**. Isso cria a VPC, sub-redes, tabelas de rotas e um Internet Gateway automaticamente.
4. **Configurações:**
    - **Nome:** democratizacao-vpc
    - **Bloco CIDR IPv4:** 10.0.0.0/16 (um bom padrão).
    - **Zonas de Disponibilidade (AZs):** Escolha 2. Isso garante alta disponibilidade.
    - **Número de sub-redes públicas:** 2.
    - **Número de sub-redes privadas:** 2. O RDS ficará aqui.
    - **NAT Gateways:** Selecione **"1 por AZ"**. Isso é **essencial** para que a Lambda na sub-rede privada possa acessar os serviços da AWS (como o S3) que estão fora da VPC.
    - **Endpoints da VPC:** Deixe como **"Nenhum"** por enquanto.
5. Clique em **"Criar VPC"**. Aguarde alguns minutos até que todos os recursos sejam criados.

2. Criação da IAM Role para a Lambda

A função Lambda precisa de permissões para ler dados do RDS, escrever arquivos no S3 e interagir com o Glue.

1. **Acesse o Console da AWS** e procure por **"IAM"**.
2. No menu à esquerda, clique em **"Funções" (Roles)** e depois em **"Criar função"**.
3. **Tipo de entidade confiável:** Selecione **"Serviço da AWS"**.
4. **Caso de uso:** Selecione **"Lambda"** e clique em **"Avançar"**.

5. **Adicionar permissões:** Vamos adicionar as políticas gerenciadas pela AWS. Pesquise e adicione as seguintes políticas:
    - AWSLambdaVPCAccessExecutionRole: Permite que a Lambda opere dentro da VPC.
    - AmazonS3FullAccess: Permite acesso total ao S3. (Para produção, você criaria uma política mais restrita, permitindo acesso apenas ao bucket vitor-democratizacao).
    - AmazonRDSDataFullAccess: Permite que a Lambda acesse os dados do RDS.
    - AWSGlueConsoleFullAccess: Permite que a Lambda interaja com o Glue (necessário para acionar o crawler, por exemplo).
6. Clique em **"Avançar"**.
7. **Nome da função:** lambda-democratizacao-role.
8. Revise as políticas e clique em **"Criar função"**.

</details>

01 - Banco de Dados (RDS MySQL)

<details>
<summary>Clique para expandir: 01-rds-mysql/README.md</summary>
Passo a Passo: Criando e Populando o Banco de Dados

1. Criação do Bucket S3

Primeiro, vamos criar o bucket onde os dados serão armazenados.

1. Vá para o serviço **S3** no console da AWS.
2. Clique em **"Criar bucket"**.
3. **Nome do bucket:** vitor-democratizacao (nomes de bucket são globais, então se este já existir, adicione um sufixo, como vitor-democratizacao-123). Anote o nome exato.
4. **Região da AWS:** Escolha a mesma região onde você criou a VPC.
5. Mantenha as outras configurações padrão e clique em **"Criar bucket"**.

2. Criação do Banco de Dados RDS MySQL

1. **Acesse o Console da AWS** e procure por **"RDS"**.
2. Clique em **"Criar banco de dados"**.
3. **Método de criação:** "Criação padrão".
4. **Opções do mecanismo:**
    - **Tipo de mecanismo:** MySQL.
    - **Modelo:** Nível gratuito (para evitar custos iniciais).
5. **Configurações:**

- **Identificador da instância de BD:** democratizacao-db.
- **Nome de usuário mestre:** admin.
- **Senha mestre:** Crie uma senha forte e **salve-a em um local seguro**. Você precisará dela.

6. **Conectividade:**
   - **Virtual Private Cloud (VPC):** Selecione a democratizacao-vpc que você criou.
   - **Grupo de sub-redes:** O RDS deve criar um novo grupo de sub-redes usando as sub-redes privadas.
   - **Acesso público:** Selecione **Não**. O banco não deve ser acessível pela internet.
   - **Grupo de segurança da VPC (firewall):**
     - Selecione **"Criar novo"**.
     - **Nome do novo grupo de segurança:** rds-sg.
7. Deixe as outras configurações padrão e clique em **"Criar banco de dados"**. A criação pode levar de 10 a 15 minutos.

3. Configurando o Acesso (Security Group)

Precisamos permitir que a Lambda acesse o banco de dados.

1. Vá para o console da **VPC** -> **Grupos de Segurança**.
2. Crie um novo Security Group para a Lambda:
   - **Nome:** lambda-sg.
   - **VPC:** democratizacao-vpc.
   - Clique em **"Criar grupo de segurança"**.
3. Agora, edite o Security Group do RDS (rds-sg):
   - Selecione rds-sg e vá para a aba **"Regras de entrada"**.
   - Clique em **"Editar regras de entrada"** e **"Adicionar regra"**.
   - **Tipo:** MYSQL/Aurora (3306).
   - **Origem:** Selecione o grupo de segurança da Lambda (lambda-sg). Isso permite que qualquer recurso com o lambda-sg acesse o RDS na porta 3306.
   - Clique em **"Salvar regras"**.

4. Scripts SQL (Criação e População)

Para se conectar e rodar os scripts, você precisará do **endpoint** do seu banco de dados. Vá para a página do RDS, clique na sua instância democratizacao-db e copie o valor de "Endpoint e porta".

Use uma ferramenta como DBeaver, MySQL Workbench ou o terminal para se conectar ao banco com o endpoint, usuário (admin) e a senha que você salvou.

Script de Criação de Tabelas:

```
CREATE DATABASE IF NOT EXISTS democratizacao_db;
USE democratizacao_db;

CREATE TABLE cargos (
    id INT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    salario DECIMAL(12,2) NOT NULL,
    bonus DECIMAL(13,2)
);

CREATE TABLE funcionarios (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(255),
    cargo_id INT,
    data_contratacao DATE,
    data_referencia DATE NOT NULL,
    FOREIGN KEY (cargo_id) REFERENCES cargos(id)
);
```

Script para Popular as Tabelas (50 exemplos):

```
USE democratizacao_db;

-- Inserir Cargos
INSERT INTO cargos (id, nome, salario, bonus) VALUES
(1, 'Engenheiro de Software Jr', 5000.00, 1000.00),
(2, 'Engenheiro de Software Pl', 8500.00, 2500.00),
(3, 'Engenheiro de Software Sr', 12000.00, 5000.00),
(4, 'Analista de Dados', 7000.00, 2000.00),
(5, 'Cientista de Dados', 11000.00, 4500.00),
(6, 'Gerente de Projetos', 13000.00, 6000.00);

-- Inserir Funcionários
INSERT INTO funcionarios (nome, cargo_id, data_contratacao, data_referencia)
VALUES
('Ana Silva', 1, '2023-01-15', CURDATE()), ('Bruno Costa', 2, '2022-05-20', CURDATE()),
('Carla Dias', 3, '2021-11-10', CURDATE()), ('Daniel Faria', 4, '2023-02-01', CURDATE()),
```

('Eduarda Lima', 5, '2022-08-30', CURDATE()), ('Fábio Martins', 6, '2020-03-12', CURDATE()),
('Gabriela Nunes', 1, '2023-03-18', CURDATE()), ('Heitor Oliveira', 2, '2022-07-22', CURDATE()),
('Isabela Pereira', 3, '2021-09-05', CURDATE()), ('João Ramos', 4, '2023-04-11', CURDATE()),
('Larissa Santos', 5, '2022-10-14', CURDATE()), ('Marcos Souza', 6, '2020-01-20', CURDATE()),
('Natália Almeida', 1, '2023-05-25', CURDATE()), ('Otávio Barbosa', 2, '2022-09-28', CURDATE()),
('Patrícia Carvalho', 3, '2021-07-15', CURDATE()), ('Rafael Ferreira', 4, '2023-06-30', CURDATE()),
('Sofia Gomes', 5, '2022-12-01', CURDATE()), ('Tiago Mendes', 1, '2023-07-07', CURDATE()),
('Úrsula Pinto', 2, '2022-11-11', CURDATE()), ('Victor Ribeiro', 3, '2021-05-21', CURDATE()),
('Yasmin Rocha', 4, '2023-08-14', CURDATE()), ('Zeca Vieira', 5, '2022-02-18', CURDATE()),
('Amanda Azevedo', 6, '2019-12-01', CURDATE()), ('Bento Cunha', 1, '2023-09-01', CURDATE()),
('Celina Esteves', 2, '2022-01-10', CURDATE()), ('Davi Lemos', 3, '2021-03-25', CURDATE()),
('Elisa Matos', 4, '2023-10-05', CURDATE()), ('Frederico Neves', 5, '2022-04-15', CURDATE()),
('Giovana Pires', 1, '2023-11-12', CURDATE()), ('Hugo Queiroz', 2, '2022-06-20', CURDATE()),
('Íris Sampaio', 3, '2021-02-08', CURDATE()), ('Jonas Tavares', 4, '2023-12-18', CURDATE()),
('Lia Vasconcelos', 5, '2022-08-01', CURDATE()), ('Mauro Xavier', 6, '2019-10-10', CURDATE()),
('Noah Arantes', 1, '2024-01-05', CURDATE()), ('Olívia Bastos', 2, '2023-03-15', CURDATE()),
('Pietro Campos', 3, '2022-05-19', CURDATE()), ('Raquel Dantas', 4, '2024-02-02', CURDATE()),
('Samuel Eanes', 5, '2023-07-28', CURDATE()), ('Tainá Furtado', 1, '2024-03-01', CURDATE()),
('Uriel Galvão', 2, '2023-09-11', CURDATE()), ('Valentina Henriques', 3, '2022-11-22', CURDATE()),
('Theo Iglésias', 4, '2024-04-04', CURDATE()), ('Stella Jardim', 5, '2023-01-20',

CURDATE()),
('Benjamin Klein', 6, '2019-08-08', CURDATE()), ('Alice Lopes', 1, '2024-05-01',
CURDATE()),
('Gael Morais', 2, '2023-02-14', CURDATE()), ('Helena Nogueira', 3, '2022-10-30',
CURDATE()),
('Miguel Otero', 4, '2024-06-05', CURDATE()), ('Laura Paiva', 5, '2023-04-19',
CURDATE());

</details>

02 - Processamento (Lambda .NET)

<details>
<summary>Clique para expandir:
02-lambda-net-parquet/README.md</summary>
Passo a Passo: Criando a Função de Extração

Esta função irá se conectar ao RDS, ler os dados do dia e salvá-los como um arquivo Parquet no S3.

1. Preparando o Ambiente de Desenvolvimento (.NET 8)

1. Abra um terminal ou PowerShell.
2. Crie uma pasta para o projeto: mkdir lambda-net-parquet && cd lambda-net-parquet
3. Crie um novo projeto de Lambda: dotnet new lambda.EmptyFunction --name RdsToS3Parquet
4. Entre na pasta do projeto: cd src/RdsToS3Parquet
5. Adicione os pacotes NuGet necessários:
   dotnet add package MySql.Data
   dotnet add package Parquet.Net
   dotnet add package AWSSDK.S3

2. Código da Função (Function.cs)

Substitua o conteúdo do arquivo src/RdsToS3Parquet/Function.cs pelo código abaixo.

using Amazon.Lambda.Core;
using Amazon.S3;
using Amazon.S3.Model;
using MySql.Data.MySqlClient;

```
using Amazon.Lambda.Core;
using Amazon.S3;
using Amazon.S3.Model;
using MySql.Data.MySqlClient;
using Parquet;
using Parquet.Data;
using System;
using System.Collections.Generic;
using System.Data;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

[assembly:
LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLamb
daJsonSerializer))]

namespace RdsToS3Parquet;

public class Function
{
    public class Funcionario
    {
```

```
        public int Id { get; set; }
        public string? Nome { get; set; }
        public int? CargoId { get; set; }
        public DateTime? DataContratacao { get; set; }
        public DateTime DataReferencia { get; set; }
    }

    private readonly IAmazonS3 _s3Client;

    public Function()
    {
        _s3Client = new AmazonS3Client();
    }

    public async Task<string> FunctionHandler(ILambdaContext context)
    {
        var dbConnectionString =
Environment.GetEnvironmentVariable("DB_CONNECTION_STRING");
        var bucketName = Environment.GetEnvironmentVariable("S3_BUCKET_NAME");

        if (string.IsNullOrEmpty(dbConnectionString) ||
string.IsNullOrEmpty(bucketName))
        {
            var errorMessage = "Erro: Variáveis de ambiente DB_CONNECTION_STRING
```

```
e/ou S3_BUCKET_NAME não estão definidas.";
            context.Logger.LogError(errorMessage);
            return errorMessage;
        }

        var dataReferencia = DateTime.UtcNow.Date;
        var funcionarios = new List<Funcionario>();

        try
        {
            using var connection = new MySqlConnection(dbConnectionString);
            await connection.OpenAsync();

            var sql = "SELECT id, nome, cargo_id, data_contratacao, data_referencia FROM
funcionarios WHERE data_referencia = @data_referencia";
            using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@data_referencia", dataReferencia);

            using var reader = await command.ExecuteReaderAsync();
            while (await reader.ReadAsync())
            {
                funcionarios.Add(new Funcionario
                {
```

```
                Id = reader.GetInt32("id"),

                Nome = reader.IsDBNull(reader.GetOrdinal("nome")) ? null :
reader.GetString("nome"),

                CargoId = reader.IsDBNull(reader.GetOrdinal("cargo_id")) ? null :
reader.GetInt32("cargo_id"),

                DataContratacao =
reader.IsDBNull(reader.GetOrdinal("data_contratacao")) ? null :
reader.GetDateTime("data_contratacao"),

                DataReferencia = reader.GetDateTime("data_referencia")

            });

        }


        context.Logger.LogInformation($"{funcionarios.Count} registros encontrados
para {dataReferencia:yyyy-MM-dd}.");

    }

    catch (Exception ex)

    {

        context.Logger.LogError($"Erro ao acessar o banco: {ex.Message}");

        return $"Erro no banco: {ex.Message}";

    }


    if (funcionarios.Count == 0)

    {

        context.Logger.LogInformation("Nenhum dado encontrado para a data de
referência.");
```

```
        return "Nenhum dado encontrado.";
    }

    using var memoryStream = new MemoryStream();
    try
    {

        var schema = new Schema(
            new DataField<int>("id"),
            new DataField<string>("nome"),
            new DataField<int?>("cargo_id"),
            new DataField<DateTimeOffset?>("data_contratacao"),
            new DataField<DateTimeOffset>("data_referencia")
        );

        using (var parquetWriter = await ParquetWriter.CreateAsync(schema,
memoryStream))
        {
            using (var groupWriter = parquetWriter.CreateRowGroup())
            {
                await groupWriter.WriteColumnAsync(new Parquet.Data.DataColumn(
                    (Parquet.Data.DataField)schema.Fields[0],
                    funcionarios.Select(f => f.Id).ToArray()
```

```
            ));
            await groupWriter.WriteColumnAsync(new Parquet.Data.DataColumn(
                (Parquet.Data.DataField)schema.Fields[1],
                funcionarios.Select(f => f.Nome).ToArray()
            ));
            await groupWriter.WriteColumnAsync(new Parquet.Data.DataColumn(
                (Parquet.Data.DataField)schema.Fields[2],
                funcionarios.Select(f => f.CargoId).ToArray()
            ));
            await groupWriter.WriteColumnAsync(new Parquet.Data.DataColumn(
                (Parquet.Data.DataField)schema.Fields[3],
                funcionarios.Select(f => f.DataContratacao.HasValue
                    ? (DateTimeOffset?)new DateTimeOffset(f.DataContratacao.Value)
                    : null).ToArray()
            ));

            await groupWriter.WriteColumnAsync(new Parquet.Data.DataColumn(
                (Parquet.Data.DataField)schema.Fields[4],
                funcionarios.Select(f => new
DateTimeOffset(f.DataReferencia)).ToArray()
            ));
        }
    }
```

```
            context.Logger.LogInformation("Parquet gerado com sucesso.");
        }
        catch (Exception ex)
        {
            context.Logger.LogError($"Erro ao gerar Parquet: {ex.Message}");
            return $"Erro no Parquet: {ex.Message}";
        }

        var s3Key =
$"funcionarios/ano={dataReferencia:yyyy}/mes={dataReferencia:MM}/dia={dataReferencia:dd}/data.parquet";

        try
        {
            memoryStream.Position = 0;
            var putRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = s3Key,
                InputStream = memoryStream
            };
```

```
            await _s3Client.PutObjectAsync(putRequest);

            context.Logger.LogInformation($"Arquivo salvo: s3://{bucketName}/{s3Key}");

        }

        catch (Exception ex)

        {

            context.Logger.LogError($"Erro ao salvar no S3: {ex.Message}");

            return $"Erro no S3: {ex.Message}";

        }



        return $"Sucesso! {funcionarios.Count} registros salvos em
s3://{bucketName}/{s3Key}";

    }

}
```

3. Publicando e Criando a Função Lambda

1. **Compile e publique o projeto:**
   # Navegue de volta para a pasta raiz do projeto (lambda-net-parquet)
   dotnet tool install -g Amazon.Lambda.Tools
   dotnet lambda deploy-function \
     --function-name rds-to-s3-parquet \
     --function-role
   arn:aws:iam::[SEU_AWS_ACCOUNT_ID]:role/lambda-democratizacao-role
   Atenção: Substitua [SEU_AWS_ACCOUNT_ID] pelo ID da sua conta AWS. Você pode encontrá-lo no canto superior direito do console da AWS.
2. **Configure a Função no Console da AWS:**
   - Vá para o serviço **Lambda** e encontre a função rds-to-s3-parquet.
   - **Configuração > Variáveis de ambiente:**
     - DB_CONNECTION_STRING:
       Server=[ENDPOINT_DO_SEU_RDS];Database=democratizacao_db;Uid=admin;Pwd=[SUA_SENHA_DO_RDS];Substitua [ENDPOINT_DO_SEU_RDS] e [SUA_SENHA_DO_RDS] pelos valores corretos.

        - S3_BUCKET_NAME: vitor-democratizacao (ou o nome que você usou).
    - **Configuração > VPC:**
        - Clique em **Editar**.
        - Selecione a democratizacao-vpc.
        - **Sub-redes:** Selecione as **duas sub-redes privadas**.
        - **Grupos de segurança:** Selecione o lambda-sg que criamos.
    - **Configuração > Geral > Editar:**
        - Aumente o **Tempo Limite (Timeout)** para 1 minuto. A primeira execução pode ser mais lenta.
3. **Teste a Função:**
    - Na aba **"Teste"**, crie um novo evento de teste (pode deixar o JSON padrão {}) e clique em **"Testar"**.
    - Verifique os logs. Se tudo deu certo, você verá a mensagem de sucesso.
    - Vá até o bucket S3 e confirme que o arquivo data.parquet foi criado na pasta funcionarios/ano=.../mes=.../dia=.../.

</details>

03 - Catálogo de Dados (Glue Crawler)

<details>
<summary>Clique para expandir: 03-glue-crawler/README.md</summary>
Passo a Passo: Catalogando os Dados do S3

O Crawler vai inspecionar os arquivos Parquet no S3, inferir o esquema (colunas e tipos de dados) e as partições, e criar uma tabela no Catálogo de Dados do Glue.

1. **Acesse o Console da AWS** e procure por **"Glue"**.
2. No menu à esquerda, em "Data Catalog", clique em **"Crawlers"** e depois em **"Criar crawler"**.
3. **Nome do crawler:** democratizacao-funcionarios-crawler.
4. **Fontes de dados:**
    - Clique em **"Adicionar uma fonte de dados"**.
    - **Fonte de dados:** S3.
    - **Localização dos dados:** s3://vitor-democratizacao/funcionarios/.
    - **Rastrear subpastas:** Sim.
5. **Função do IAM:**
    - Selecione **"Criar uma nova função do IAM"**.
    - **Nome da função:** glue-democratizacao-role.
6. **Saída e agendamento:**
    - **Banco de dados de destino:** Clique em **"Adicionar banco de dados"**.
    - **Nome do banco de dados:** democratizacao_db.

    - **Agendamento:** Sob demanda.
7. Revise tudo e clique em **"Criar crawler"**.
8. **Execute o Crawler:**
    - Selecione o crawler recém-criado e clique em **"Executar crawler"**.
    - A execução pode levar um ou dois minutos.
    - Quando terminar, ele deve indicar que **1 tabela foi criada**.
9. **Verifique a Tabela:**
    - No menu do Glue, vá para **"Tabelas"**.
    - Você verá a tabela funcionarios no banco de dados democratizacao_db.
    - Clique nela e explore o esquema e as partições que ele detectou automaticamente (ano, mes, dia).

</details>

04 - Consultas Ad Hoc (Athena)

<details>
<summary>Clique para expandir:
04-athena-queries/README.md</summary>
Passo a Passo: Consultando os Dados com SQL

O Athena permite que você execute consultas SQL diretamente nos arquivos do S3, usando o catálogo do Glue como se fosse um banco de dados.

1. Configuração Inicial do Athena

1. **Acesse o Console da AWS** e procure por **"Athena"**.
2. Se for sua primeira vez, talvez apareça um pop-up de configuração. Clique em **"Explorar o editor de consultas"**.
3. **Configurar local do resultado da consulta:**
    - No topo da tela, clique em **"Configurações"**.
    - Em **"Local do resultado da consulta"**, insira o caminho s3://vitor-democratizacao/athena-results/.
    - **Importante:** Vá antes ao console do S3 e crie a pasta athena-results dentro do seu bucket.
    - Clique em **"Salvar"**.

2. Executando as Consultas

No editor de consultas do Athena:

1. **Fonte de dados:** Selecione AwsDataCatalog.
2. **Banco de dados:** Selecione democratizacao_db.

3. A tabela funcionarios deve aparecer na lista à esquerda.

Consulta 1: Contagem de funcionários por nome

```
SELECT
  nome,
  COUNT(*) AS total
FROM "democratizacao_db"."funcionarios"
GROUP BY nome
ORDER BY total DESC;
```

Consulta 2: Join com a tabela de cargos

Para fazer o join, primeiro precisamos carregar os dados da tabela cargos também. A maneira mais simples é criar um arquivo CSV, fazer o upload para o S3 e criar outra tabela no Glue para ele.

1. Crie um arquivo cargos.csv com o conteúdo:
   ```
   id,nome,salario,bonus
   1,"Engenheiro de Software Jr",5000.00,1000.00
   2,"Engenheiro de Software Pl",8500.00,2500.00
   3,"Engenheiro de Software Sr",12000.00,5000.00
   4,"Analista de Dados",7000.00,2000.00
   5,"Cientista de Dados",11000.00,4500.00
   6,"Gerente de Projetos",13000.00,6000.00
   ```
2. Faça o upload para s3://vitor-democratizacao/cargos/cargos.csv.
3. Crie um novo **Crawler do Glue** para s3://vitor-democratizacao/cargos/, que criará a tabela cargos.

Após o crawler da tabela cargos rodar, execute a query de join:

```
SELECT
  f.nome AS nome_funcionario,
  c.nome AS nome_cargo,
  c.salario
FROM "democratizacao_db"."funcionarios" f
JOIN "democratizacao_db"."cargos" c
  ON f.cargo_id = c.id
WHERE f.ano = 'ANO_ATUAL' AND f.mes = 'MES_ATUAL' -- Filtre por partição para otimizar
```

LIMIT 10;

Substitua ANO_ATUAL e MES_ATUAL pelos valores da partição existente.

</details>

05 - Visualização (QuickSight) e Agendamento (EventBridge)

<details>
<summary>Clique para expandir:
05-quicksight-dashboard/README.md</summary>
Passo a Passo: Criando o Dashboard e Automatizando a Carga

1. Configurando o QuickSight

1. **Acesse o Console da AWS** e procure por **"QuickSight"**.
2. Se for seu primeiro acesso, você precisará se inscrever. Escolha a edição **Standard** e siga os passos.
3. **Permissões:** Durante a configuração, ou em **"Gerenciar QuickSight" > "Segurança e permissões"**, garanta que o QuickSight tem permissão para acessar o **Athena** e o bucket S3 (vitor-democratizacao).

2. Criando o Conjunto de Dados (Dataset)

1. Na página principal do QuickSight, clique em **"Conjuntos de dados" > "Novo conjunto de dados"**.
2. Selecione **"Athena"** como fonte.
3. **Nome da fonte de dados:** democratizacao_athena.
4. **Catálogo de dados:** AwsDataCatalog.
5. **Banco de dados:** democratizacao_db.
6. **Tabelas:** Selecione a tabela funcionarios e clique em **"Selecionar"**.
7. Na próxima tela, você pode escolher importar os dados para o SPICE (memória do QuickSight) ou fazer uma consulta direta. Para começar, **"Consultar diretamente seus dados"** é mais simples.
8. Clique em **"Visualizar"**.

3. Criando a Análise (Gráfico)

1. Você será levado para a tela de Análise.
2. **Tipo de elemento visual:** Na parte inferior, selecione o ícone de **"Gráfico de barras verticais"**.
3. **Configuração do gráfico:**
    - Arraste o campo nome da lista de campos para o poço **"Eixo X"**.

    - Arraste o campo id para o poço **"Valor"**. Por padrão, ele deve usar a agregação **"Contagem"**, que é o que queremos.
4. Pronto! Você terá um gráfico mostrando a contagem de registros por nome de funcionário. Você pode renomear o gráfico e o dashboard.

4. Agendando a Execução Diária da Lambda (EventBridge)

Para que a carga seja diária e automática, usaremos o EventBridge Scheduler.

1. **Acesse o Console da AWS** e procure por **"EventBridge"**.
2. No menu à esquerda, clique em **"Agendamentos" (Schedules)** e **"Criar agendamento"**.
3. **Nome do agendamento:** executa-lambda-diariamente-12h.
4. **Padrão de recorrência:**
    - Selecione **"Agendamento recorrente"**.
    - **Tipo de agendamento:** Agendamento baseado em cron.
    - **Expressão cron:** 15 15 * * ? *Esta expressão significa "às 15:15 UTC todos os dias", que corresponde a 12:15 no horário de Brasília (UTC-3).
5. **Destino:**
    - Selecione **"API de destino da AWS"**.
    - Pesquise e selecione **"Lambda Invoke"**.
    - **Função Lambda:** Selecione a função rds-to-s3-parquet.
6. **Configurações > Permissões:**
    - Selecione **"Criar uma nova função para este agendamento"**. O EventBridge criará a permissão necessária para invocar a Lambda.
7. Clique em **"Criar agendamento"**.

Agora, sua função Lambda será executada todos os dias automaticamente, populando seu Data Lake para que seu dashboard no QuickSight esteja sempre atualizado.

</details>