

Curso de Especialização em Big Data – Escola Politécnica da USP

Disciplina de Ingestão de Dados eEDB-011

Prof. Dra. Jeaneth Machicao - Prof. Leandro

Projeto Final

Grupo 2

Ingrid Silva

Lucas Pereira

Miguel Ferreira

João Martins

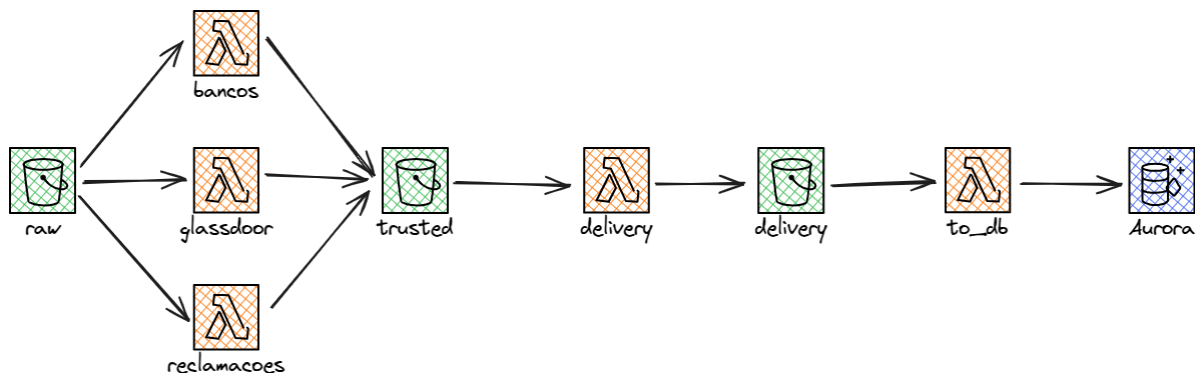
Exercício 2

Link do repositório no GitHub

https://github.com/MFC-MiguelFerreira/eEDB-011_2025-3_ingestao_de_dados

Arquitetura da Solução

A solução foi desenvolvida na nuvem da AWS, utilizando o ambiente de laboratório disponibilizado na disciplina. Para esta atividade em específico, é possível observar a arquitetura da solução implementada, composta basicamente por um nível de armazenamento em um data lake estruturado segundo a arquitetura medallion no Amazon S3, e por um nível de processamento realizado por funções Lambda desenvolvidas em Python. Como destino final, os dados são armazenados em um banco de dados PostgreSQL provisionado no Amazon RDS.



Criação do banco de dados no RDS

[Repete-se da Atividade 1]

Esse passo é semelhante ao da primeira atividade, em que optamos por criar o banco de dados utilizando PostgreSQL no Amazon RDS, serviço gerenciado da AWS. Esse serviço não foi criado utilizando IaC.

database-1

Summary

DB Identifier

database-1

CPU

3.53%

Status

Available

Class

db.t4g.micro

Role

Instance

Current activity

0.00 sessions

Engine

PostgreSQL

Region & AZ

us-east-1b

Recommendations

Connectivity & security

Monitoring

Logs & events

Configuration

Zero-ETL integrations

Maintenance & backups

Data migrations - new

Tags

Connectivity & security

Endpoint & port

Endpoint

database-1.cow15th1wubu.us-east-1.rds.amazonaws.com

Port

5432

Networking

Availability Zone

us-east-1b

VPC

vpc-0f1656b9318d0b4c2

Subnet group

default-vpc-0f1656b9318d0b4c2

Subnets

subnet-03cacd8897dbb5e9f
subnet-07bd42120c3d1d120
subnet-0b38f6e311d149cbf
subnet-0173a2cec145c7e6d
subnet-0844c11dda16c40ee
subnet-0421061737a616939

Security

VPC security groups

default (sg-0135aebcef6d4d241)

Active

Publicly accessible

Yes

Certificate authority

info
rds-ca-rsa2048-g1

Certificate authority date

May 25, 2061, 20:34 (UTC-03:00)

DB instance certificate expiration date

August 05, 2026, 20:07 (UTC-03:00)

Databases (1)

Group resources

Modify

Actions

Create database

Filter by databases

DB Identifier

Status

Role

Engine

Region ...

Size

Recommendations

CP

database-1

Available

Instance

PostgreSQL

us-east-1b

db.t4g.micro

Organização do Repositório

O repositório, dentro da pasta *atividade2*, foi organizado de forma a armazenar tanto os arquivos Python exigidos nesta etapa do processo quanto os arquivos relacionados à infraestrutura. Dessa forma, a pasta *terraform* contém os arquivos de infraestrutura, enquanto a pasta *lambdas* reúne os scripts utilizados, bem como o arquivo *requirements.txt* com as dependências necessárias.

MFC-MiguelFerreira / eEDB-011_2025-3_ingestao_de_dados

Q Type to search

Code Issues Pull requests Actions Projects Security Insights Settings

Files

main

Go to file

atividade1

atividade2

lambdas

terraform

data

.gitignore

README.md

requirements.txt

eEDB-011_2025-3_ingestao_de_dados / atividade2

Add file

MFC-MiguelFerreira Add Lambda function to insert data into PostgreSQL: implement S3 i... 78d7b53 · 8 minutes ago History

Name

Last commit message

Last commit da...

..

lambdas

Add Lambda function to insert data into PostgreSQL: impl...

8 minutes ago

terraform

Add Lambda function to insert data into PostgreSQL: impl...

8 minutes ago

Armazenamento dos dados no data lake (S3)

[Repete-se da Atividade 1]

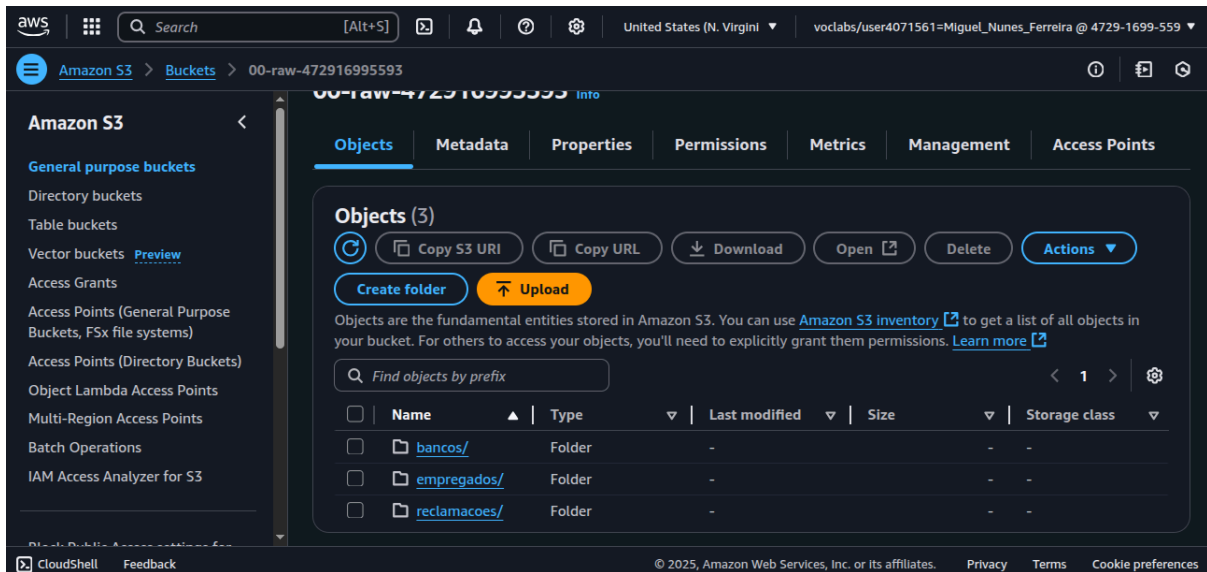
Consideramos como ponto de partida da ingestão, os arquivos fornecidos em aula armazenados no data lake da AWS, o S3. Organizamos os temas dos arquivos por pastas.

The top screenshot shows a code editor with a file explorer on the left. The file explorer shows a directory structure with folders 'atividade1', 'atividade2', 'lambdas', and 'terraform'. Inside 'terraform', there are files: '.terraform.lock.hcl', 'data.tf', 'lambda_layers.tf', 'lambdas.tf', 'provider.tf', 's3.tf' (selected), 'variable.tf', and 'version.tf'. The main editor shows the content of 's3.tf', which defines four S3 buckets using Terraform:

```
1 resource "aws_s3_bucket" "raw_datalake_bucket" {
2   bucket = "00-raw-${local.account_id}"
3   force_destroy = true
4 }
5
6 resource "aws_s3_bucket" "trusted_datalake_bucket" {
7   bucket = "01-trusted-${local.account_id}"
8   force_destroy = true
9 }
10
11 resource "aws_s3_bucket" "delivery_datalake_bucket" {
12   bucket = "02-delivery-${local.account_id}"
13   force_destroy = true
14 }
15
16 resource "aws_s3_bucket" "lambda_bucket" {
17   bucket = "lambda-${local.account_id}"
18   force_destroy = true
19 }
```

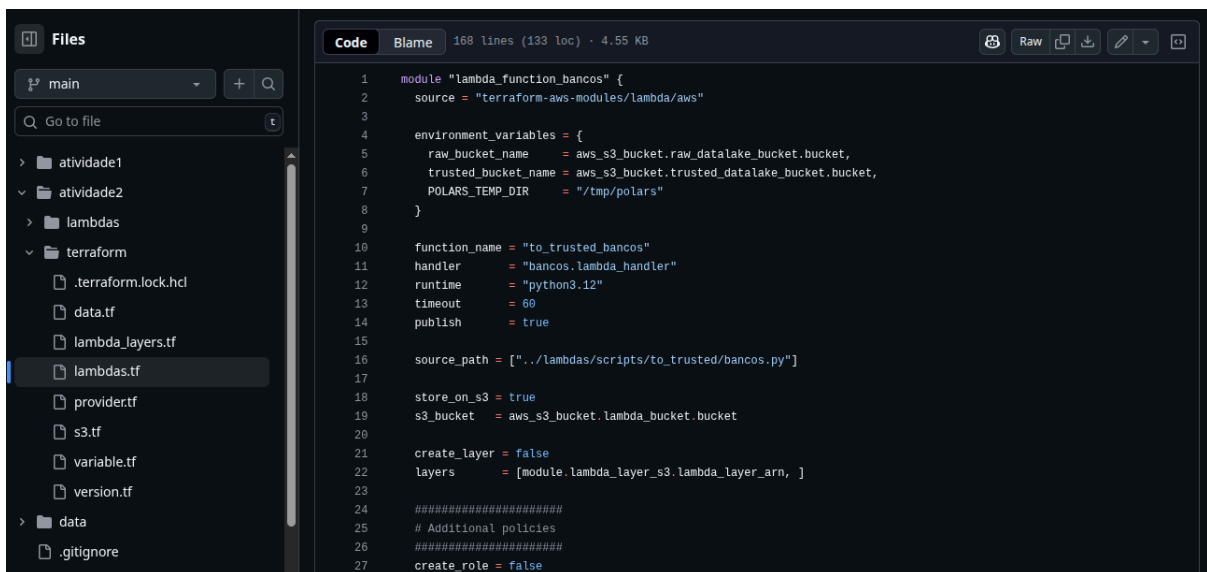
The bottom screenshot shows the AWS S3 console. The left sidebar shows the 'Amazon S3' menu with options like 'General purpose buckets', 'Directory buckets', 'Table buckets', 'Vector buckets', 'Access Grants', 'Access Points (General Purpose Buckets, FSx file systems)', 'Access Points (Directory Buckets)', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', and 'IAM Access Analyzer for S3'. The main content area shows 'General purpose buckets (4)' with a table of buckets:

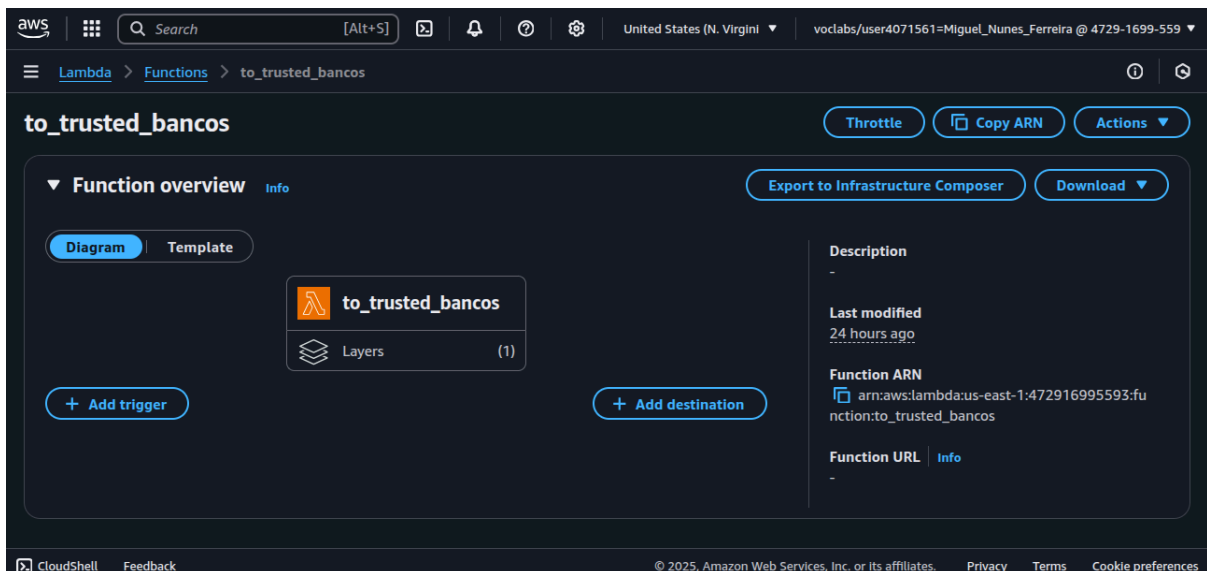
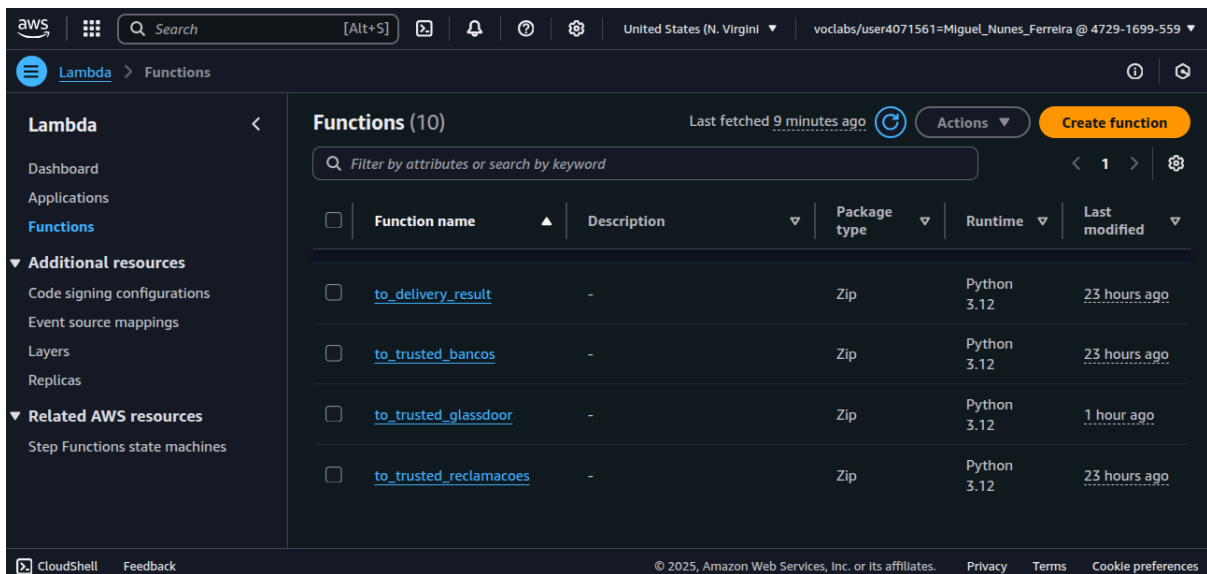
Name	AWS Region	Creation date
00-raw-472916995593	US East (N. Virginia) us-east-1	August 3, 2025, 21:28:13 (UTC-03:00)
01-trusted-472916995593	US East (N. Virginia) us-east-1	August 3, 2025, 21:28:13 (UTC-03:00)
02-delivery-472916995593	US East (N. Virginia) us-east-1	August 3, 2025, 21:28:14 (UTC-03:00)
lambda-472916995593	US East (N. Virginia) us-east-1	August 3, 2025, 21:28:13 (UTC-03:00)



Processamento raw to trusted

Para realizar o processamento de *raw* para *trusted*, foram criadas três funções Lambda utilizando Terraform.





Dependências

Todas as funções Lambda contam com *layers* que contêm as dependências necessárias. Para as funções referentes aos dados de bancos e do Glassdoor, os *layers* foram criados por meio do comando ``pip install -r requirements.txt -t layer/python``, tendo como principal dependência a biblioteca Polars. Já para a função Lambda responsável pelo processamento das reclamações, foi utilizada a dependência gerenciada do AWS SDK for pandas (aws wrangler).

The top screenshot shows a code editor with the following Terraform code:

```
1 module "lambda_layer_s3" {
2   source = "terraform-aws-modules/lambda/aws"
3
4   create_layer = true
5
6   layer_name      = "ingestao-de-dados-atividade2"
7   description     = "Lambda Layer with necessary dependencies to Atividade2"
8   compatible_runtimes = ["python3.12"]
9
10  source_path = "../lambdas/layer"
11
12  store_on_s3 = true
13  s3_bucket   = aws_s3_bucket.lambda_bucket.bucket
14 }
15
16 module "lambda_layer_s3_to_db" {
17   source = "terraform-aws-modules/lambda/aws"
18
19   create_layer = true
20
21   layer_name      = "ingestao-de-dados-atividade2-to-db"
22   description     = "Lambda Layer with necessary dependencies to Atividade2 to db"
```

The middle screenshot shows the AWS Lambda console 'Layers' page for the layer 'ingestao-de-dados-atividade2'.

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	ingestao-de-dados-atividade2	6	python3.12	-	arn:aws:lambda:us-east-1:4729169955

The bottom screenshot shows the AWS Lambda console 'Layers' page for the layer 'AWSSDKPandas-Python312'.

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	AWSSDKPandas-Python312	16	python3.12	x86_64	arn:aws:lambda:us-east-1:336392948345

Código

Bancos

O script `bancos.py` realiza o processamento dos dados brutos de bancos, lidos diretamente de arquivos TSV armazenados no bucket raw. Utilizando a biblioteca Polars, os dados são carregados, renomeados para padronização dos nomes das colunas e tratados para uniformizar os campos de CNPJ e nome das instituições financeiras, removendo sufixos e caracteres indesejados. O resultado é gravado em formato Parquet com compressão Snappy na camada trusted.

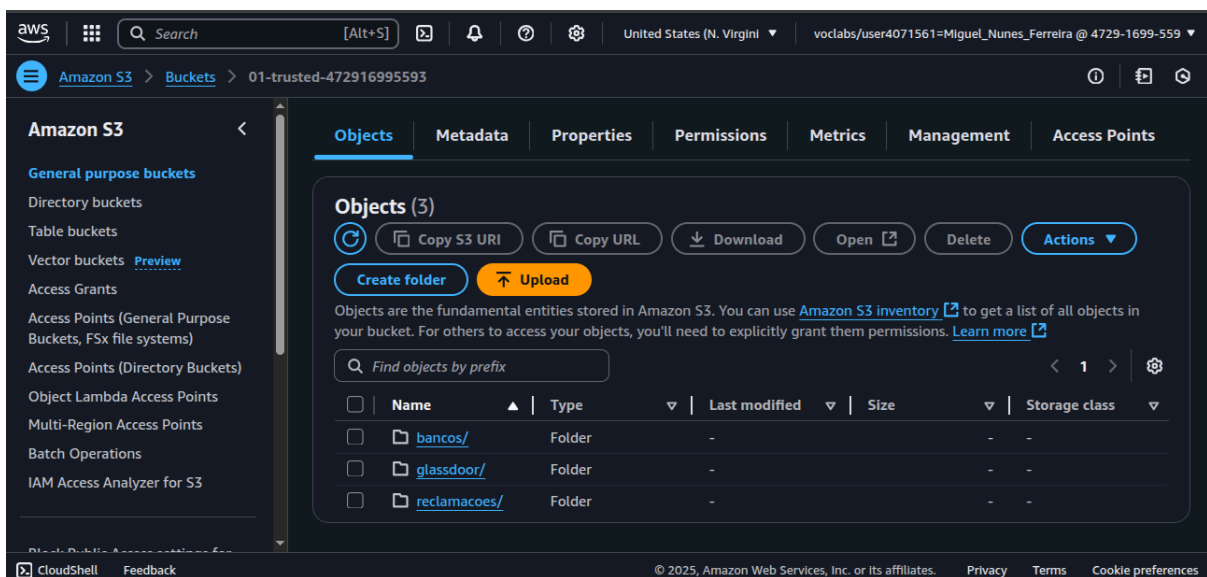
Glassdoor

O arquivo `glassdoor.py` é responsável por processar os dados de empregados provenientes de múltiplos arquivos CSV no bucket raw. Utiliza as bibliotecas `boto3` para interação com o S3 e `polars` para leitura, transformação e agregação dos dados. Após renomear e selecionar as colunas relevantes, o script aplica tratamentos nos campos de CNPJ e nome, e realiza agregações por nome de empregador, somando e calculando médias de métricas como avaliações, cultura, diversidade e outros indicadores. O resultado final é salvo em formato Parquet comprimido no bucket S3 na camada trusted.

Reclamações

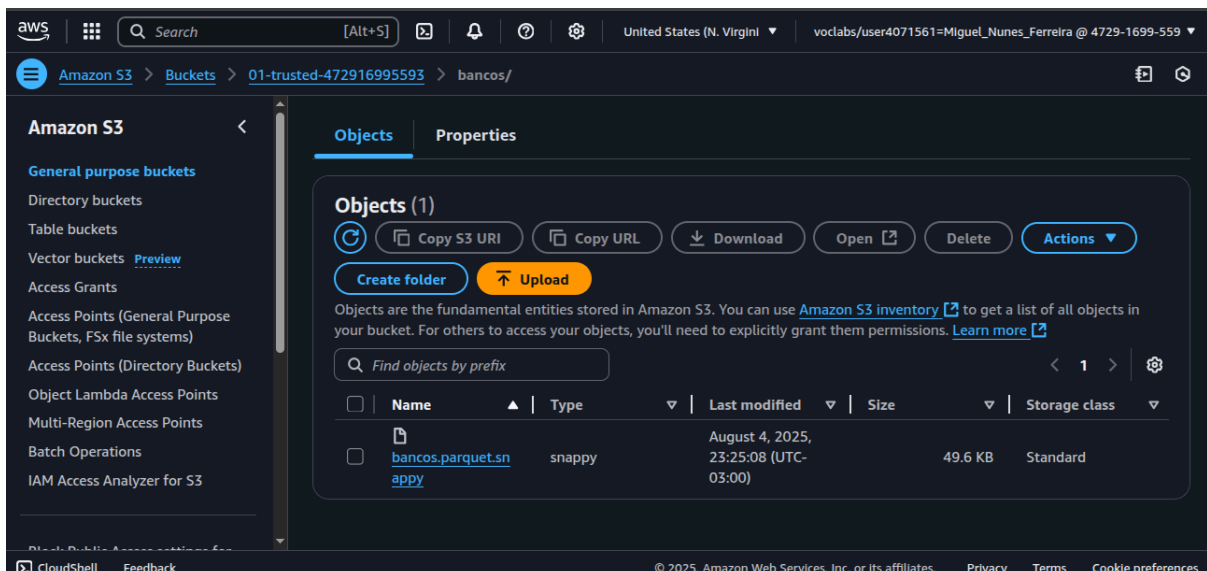
O script `reclamacoes.py` processa os dados de reclamações financeiras, que são lidos de arquivos CSV no bucket `raw` utilizando a biblioteca `aws wrangler`. Os dados passam por renomeação de colunas, padronização dos campos de CNPJ e nome das instituições, e tipagem explícita das colunas conforme schema definido. Após o tratamento, os dados são gravados em formato Parquet com compressão Snappy no bucket `trusted`. As dependências principais são `aws wrangler` para leitura e escrita no S3 e `numpy` para eventuais operações numéricas.

Resultado



The screenshot shows the Amazon S3 console interface. The left sidebar lists various bucket types and access points. The main content area is titled 'Objects (3)' and shows a list of three folders: 'bancos/', 'glassdoor/', and 'reclamacoes/'. The 'reclamacoes/' folder is highlighted. The console also displays action buttons like 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', and 'Actions'.

Name	Type	Last modified	Size	Storage class
bancos/	Folder	-	-	-
glassdoor/	Folder	-	-	-
reclamacoes/	Folder	-	-	-



The screenshot shows the Amazon S3 console interface with the 'Properties' tab selected for the object 'bancos.parquet.snappy'. The console displays the object's details, including its name, type, last modified date, size, and storage class.

Name	Type	Last modified	Size	Storage class
bancos.parquet.snappy	snappy	August 4, 2025, 23:25:08 (UTC-03:00)	49.6 KB	Standard

Bancos

```
bancos.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1474 entries, 0 to 1473
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   segment     1474 non-null   string
1   cnpj         1474 non-null   string
2   name        1474 non-null   string
dtypes: string(3)
memory usage: 34.7 KB
```

bancos

	segment	cnpj	name
0	S1	0	banco do brasil
1	S1	60746948	bradesco
2	S1	30306294	btg pactual
3	S1	360305	caixa economica federal
4	S1	60872504	itau
...
1469	S2	59588111	banco votorantim
1470	S3	28127603	banestes s/a banco do estado do espirito santo

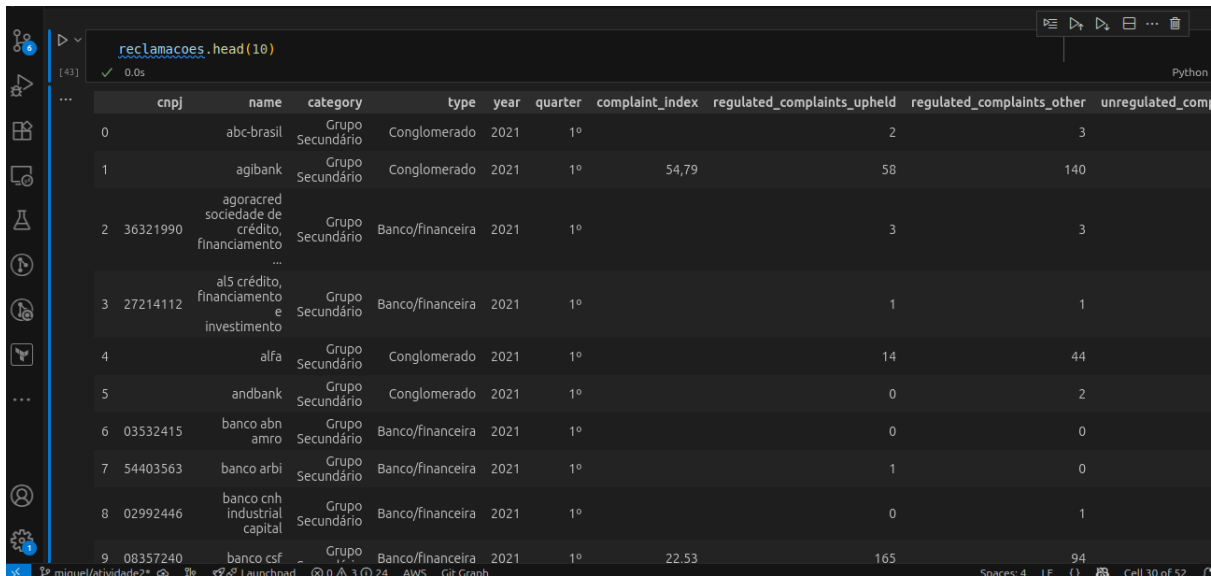
Glassdoor

```
glassdoor.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   name         33 non-null    string
1   cnpj         5 non-null     string
2   segment      32 non-null    string
3   revenue      33 non-null    string
4   reviews_count 33 non-null    Int64
5   culture_count 33 non-null    Int64
6   salaries_count 33 non-null    Int64
7   benefits_count 33 non-null    Int64
8   culture_score 33 non-null    float64
9   diversity_score 33 non-null    float64
10  quality_of_life_score 33 non-null    float64
11  leadership_score 33 non-null    float64
12  compensation_score 33 non-null    float64
13  career_opportunities_score 33 non-null    float64
14  recommend_percent 33 non-null    float64
15  positive_outlook_percent 33 non-null    float64
16  match_percent 33 non-null    float64
dtypes: Int64(4), float64(9), string(4)
memory usage: 4.6 KB
```

```
glassdoor.head()

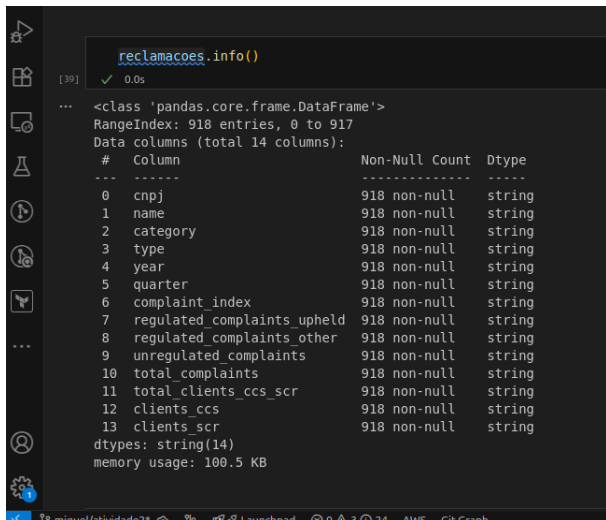
name cnpj segment revenue reviews_count culture_count salaries_count benefits_count culture_score diversity_score quality_of_life_score le
0 citibank <NA> S2 Mais de US$ 10 bilhões 31000 8900 52000 9900 3.8 4.1 3.5
1 pan <NA> S3 Desconhecida/não se aplica 1100 520 1700 493 4.2 4.4 3.9
2 bmg <NA> S3 De US$ 1 a US$ 5 milhões 445 232 704 277 3.9 4.1 3.7
3 societe generale <NA> S3 Mais de US$ 10 bilhões 11000 3400 16000 2800 3.7 3.9 3.9
4 banco daycoval <NA> S3 De US$ 25 a US$ 100 milhoes 300 157 521 107 3.7 3.6 3.4
```

reclamacoes.head(10)

	cnpj	name	category	type	year	quarter	complaint_index	regulated_complaints_upheld	regulated_complaints_other	unregulated_complaints
0		abc-brasil	Grupo Secundário	Conglomerado	2021	1º		2	3	
1		agibank	Grupo Secundário	Conglomerado	2021	1º	54,79	58	140	
2	36321990	agoracred sociedade de crédito, financiamento e investimento	Grupo Secundário	Banco/financeira	2021	1º		3	3	
3	27214112	al5 crédito, financiamento e investimento	Grupo Secundário	Banco/financeira	2021	1º		1	1	
4		alfa	Grupo Secundário	Conglomerado	2021	1º		14	44	
5		andbank	Grupo Secundário	Conglomerado	2021	1º		0	2	
6	03532415	banco abn amro	Grupo Secundário	Banco/financeira	2021	1º		0	0	
7	54403563	banco arbi	Grupo Secundário	Banco/financeira	2021	1º		1	0	
8	02992446	banco cnh industrial capital	Grupo Secundário	Banco/financeira	2021	1º		0	1	
9	08357240	banco csf	Grupo Secundário	Banco/financeira	2021	1º	22,53	165	94	

Reclamações



reclamacoes.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   cnpj                                  918 non-null    string
1   name                                  918 non-null    string
2   category                             918 non-null    string
3   type                                  918 non-null    string
4   year                                  918 non-null    string
5   quarter                              918 non-null    string
6   complaint_index                      918 non-null    string
7   regulated_complaints_upheld          918 non-null    string
8   regulated_complaints_other          918 non-null    string
9   unregulated_complaints              918 non-null    string
10  total_complaints                    918 non-null    string
11  total_clients_ccs_scr               918 non-null    string
12  clients_ccs                         918 non-null    string
13  clients_scr                         918 non-null    string
dtypes: string(14)
memory usage: 100.5 KB
```

Processamento trusted to delivery

Para realizar o processamento de *trusted* para *delivery*, foi criado uma função Lambda utilizando Terraform.

aws

Search [Alt+S]

United States (N. Virgini

voclabs/user4071561=Miguel_Nunes_Ferreira @ 4729-1699-559

Lambda > Functions > to_delivery_result

to_delivery_result

Throttle Copy ARN Actions

Function overview Info

Diagram Template

to_delivery_result

Layers (1)

+ Add trigger

+ Add destination

Description

Last modified 23 hours ago

Function ARN arn:aws:lambda:us-east-1:472916995593:fu
nction:to_delivery_result

Function URL Info

Export to Infrastructure Composer Download

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Layers Info

Edit Add a layer

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	AWSSDKPandas-Python312	16	python3.12	x86_64	arn:aws:lambda:us-east-1:336392948345

Código

O arquivo `result.py` realiza o processamento final de integração dos dados da camada `truste`, consolidando informações de bancos, reclamações e avaliações do Glassdoor. Utilizando as bibliotecas `Polars` para manipulação eficiente de `DataFrames` e `Boto3` para acesso aos arquivos `Parquet` no `S3`, o script executa união, através da estratégia *left join*, entre as diferentes fontes, padroniza os campos de `CNPJ` e nome das instituições. O resultado consolidado é gravado em formato `Parquet` na camada `delivery`.

Resultado

aws

Search

[Alt+S]

United States (N. Virgini

voclabs/user4071561=Miguel_Nunes_Ferreira @ 4729-1699-559

Amazon S3

Buckets

02-delivery-472916995593

Amazon S3

General purpose buckets

Directory buckets

Table buckets

Vector buckets Preview

Access Grants

Access Points (General Purpose Buckets, FSx file systems)

Access Points (Directory Buckets)

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

02-delivery-472916995593 Info

Objects

Metadata

Properties

Permissions

Metrics

Management

Access Points

Objects (1)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

1

Find objects by prefix

Name

Type

Last modified

Size

Storage class

result/

Folder

-

-

-

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 167 entries, 0 to 166

Data columns (total 31 columns):

Column Non-Null Count Dtype

0 name 167 non-null string

1 cnpj 167 non-null string

2 segment 167 non-null string

3 revenue 167 non-null string

4 reviews_count 167 non-null Int64

5 culture_count 167 non-null Int64

6 salaries_count 167 non-null Int64

7 benefits_count 167 non-null Int64

8 culture_score 167 non-null float64

9 diversity_score 167 non-null float64

10 quality_of_life_score 167 non-null float64

11 leadership_score 167 non-null float64

12 compensation_score 167 non-null float64

13 career_opportunities_score 167 non-null float64

14 recommend_percent 167 non-null float64

15 positive_outlook_percent 167 non-null float64

16 match_percent 167 non-null float64

17 name_reclamacoes 9 non-null string

18 category 135 non-null string

19 type 135 non-null string

29 clients_scr 135 non-null string

30 cnpj_reclamacoes 126 non-null string

dtypes: Int64(4), float64(9), string(18)

memory usage: 41.2 KB

miguel/atividade2*

Launchpad

0.0.3

AWS

Git Graph

Spaces: 4

Cell 37 of 54

result

[47]

0.0s

Python

name cnpj segment revenue reviews_count culture_count salaries_count benefits_count culture_score diversity_score ... quarter comp

0 citibank 33479023 S2 Mais de US\$ 10 bilhões 31000 8900 52000 9900 3.8 4.1 ... <NA>

1 pan 59285411 S3 Desconhecida/não se aplica 1100 520 1700 493 4.2 4.4 ... <NA>

2 bmg 61186680 S3 De US\$ 1 a US\$ 5 milhões 445 232 704 277 3.9 4.1 ... <NA>

3 societe generale 61533584 S3 Mais de US\$ 10 bilhões 11000 3400 16000 2800 3.7 3.9 ... <NA>

4 banco daycoval 62232889 S3 De US\$ 25 a US\$ 100 milhões 300 157 521 107 3.7 3.6 ... <NA>

... ..

162 sicredi 1181521 S3 Mais de US\$ 10 bilhões 2900 1300 4800 1100 4.4 4.2 ... 2º

163 sicredi 1181521 S3 Mais de US\$ 10 bilhões 2900 1300 4800 1100 4.4 4.2 ... 3º

164 sicredi 1181521 S3 Mais de US\$ 10 bilhões 2900 1300 4800 1100 4.4 4.2 ... 1º

165 sicredi 1181521 S3 Mais de US\$ 10 bilhões 2900 1300 4800 1100 4.4 4.2 ... 3º

166 sicredi 1181521 S3 Mais de US\$ 10 bilhões 2900 1300 4800 1100 4.4 4.2 ... 4º

167 rows x 31 columns

miguel/atividade2*

Launchpad

0.0.3

AWS

Git Graph

Spaces: 4

Cell 37 of 54

Processamento delivery to database

Para realizar o processamento de *trusted* para o banco de dados, foi criado uma função Lambda utilizando Terraform.

Atenção: código apenas executado localmente, conforme explicado na seção de desafios.

The screenshot displays the AWS Lambda console interface for a function named 'to_db'. The top navigation bar shows the user is logged in as 'voclabs/user4071561=Miguel_Nunes_Ferreira @ 4729-1699-559'. The breadcrumb trail indicates the path: Lambda > Functions > to_db. The function overview section includes tabs for 'Diagram' and 'Template', a 'Throttle' button, a 'Copy ARN' button, and an 'Actions' dropdown. The 'Function overview' section shows the function name 'to_db', a 'Layers' section with one layer, and buttons for '+ Add trigger' and '+ Add destination'. The 'Description' section is empty. The 'Last modified' section shows '2 hours ago'. The 'Function ARN' section shows 'arn:aws:lambda:us-east-1:472916995593:fu nction:to_db'. The 'Function URL' section is empty. The 'Layers' section shows a table with one layer.

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	ingestao-de-dados-atividade2-to-db	1	python3.12	-	arn:aws:lambda:us-east-1:47291

Código

O arquivo `to_db.py` é responsável por realizar a carga dos dados finais, previamente processados e armazenados em formato Parquet na camada delivery, para uma tabela específica em um banco de dados Aurora PostgreSQL na AWS. Utilizando as bibliotecas Polars para leitura eficiente do arquivo Parquet, Boto3 para acesso ao Secrets Manager (recuperando a senha do banco de dados) e Psycopg para conexão e inserção dos dados, o script lê os dados do S3, conecta-se ao banco de dados utilizando credenciais e insere linha a linha na tabela de destino. As principais dependências são Polars, Boto3, Psycopg e módulos padrão do Python.

Resultado

atividade2
AZ name
AZ cnpj
AZ segment
AZ revenue
123 reviews_count
123 culture_count
123 salaries_count
123 benefits_count
123 culture_score
123 diversity_score
123 quality_of_life_score
123 leadership_score
123 compensation_score
123 career_opportunities_score
123 recommend_percent
123 positive_outlook_percent
123 match_percent
AZ name_reclamacoes
AZ category
AZ type
AZ year
AZ quarter
AZ complaint_index
AZ regulated_complaints_upheld
AZ regulated_complaints_other
AZ unregulated_complaints
AZ total_complaints
AZ total_clients_ccs_scr
AZ clients_ccs
AZ clients_scr
AZ cnpj_reclamacoes

PostgreSQL Data Viewer - atividade2

Grid view showing 167 rows of data. Columns include: AZ name, AZ cnpj, AZ segment, AZ revenue, 123 reviews_count, 123 culture_count, 123 salaries_count, 123 benefit.

AZ name	AZ cnpj	AZ segment	AZ revenue	123 reviews_count	123 culture_count	123 salaries_count	123 benefit
1	citibank	33479023	S2	Mais de US\$ 10 bilhões	31,000	8,900	52,000
2	pan	59285411	S3	Desconhecida/não se aplica	1,100	520	1,700
3	bmg	61186680	S3	De US\$ 1 a US\$ 5 milhões	445	232	704
4	societe generale	61533584	S3	Mais de US\$ 10 bilhões	11,000	3,400	16,000
5	banco daycoval	62232889	S3	De US\$ 25 a US\$ 100 milhões	300	157	521
6	alfa	60770336	S3	Desconhecida/não se aplica	175	74	271
7	goldman sachs	4332281	S3	Mais de US\$ 10 bilhões	17,000	4,400	36,000
8	jp morgan chase	33172537	S3	Mais de US\$ 10 bilhões	22,859	9,122	66,600
9	bnp paribas	1522368	S3	Mais de US\$ 10 bilhões	13,000	4,100	20,000
10	morgan stanley	2801938	S3	Mais de US\$ 10 bilhões	17,000	4,600	33,000
11	credit suisse	33987793	S2	Mais de US\$ 10 bilhões	11,000	3,400	22,000
12	china construction bank (brasil) banco multiplo s/	7450604	S3	Mais de US\$ 10 bilhões	1,742	388	1,440
13	deutsche	62331228	S3	Mais de US\$ 10 bilhões	12,000	3,200	21,000
14	btg pactual	30306294	S1	Desconhecida/não se aplica	1,600	683	2,800
15	santander	90400888	S1	Mais de US\$ 10 bilhões	17,000	5,900	29,000
16	banrisul	92702067	S2	De US\$ 25 a US\$ 100 milhões	1,000	155	1,700
17	ape pouplex	655522	S3	Mais de US\$ 10 bilhões	882	552	1,153
18	original	92894922	S3	Desconhecida/não se aplica	651	286	1,100

167 row(s) fetched - 0.760s (0.010s fetch), on 2025-08-05 at 23:26:50

PostgreSQL Data Viewer - atividade2

Grid view showing 167 rows of data. Columns include: 123 quality_of_life_score, 123 leadership_score, 123 compensation_score, 123 career_opportunities_score, 123 recommend_percent, 123 positive_outlook_per.

123 quality_of_life_score	123 leadership_score	123 compensation_score	123 career_opportunities_score	123 recommend_percent	123 positive_outlook_per
4.1	3.5	3.5	3.8	3.9	76
4.4	3.9	3.9	4.6	4	91
4.1	3.7	3.6	4.2	3.5	79
3.9	3.9	3.3	3.3	3.5	75
3.6	3.4	3.3	4.1	3.4	78
3.2	3.1	2.8	3.8	2.7	68
4	2.9	3.5	3.8	4	72
4.2	3.55	3.65	3.9	4	81
4	3.8	3.4	3.4	3.5	77
4.1	3.6	3.7	3.7	3.9	81
4.1	3.9	3.3	3.6	3.7	68
2.95	3.4	2.9	3.5	2.8	54
4	3.8	3.5	3.6	3.7	78
3.5	2.8	3.7	4.4	4.2	78
4	3.4	3.4	4	3.8	75
3.8	3.7	3.4	4.1	3.3	84
3.45	3	2.9	2.7	3.2	51
3.5	3.4	3.1	4	3.3	71

167 row(s) fetched - 0.760s (0.010s fetch), on 2025-08-05 at 23:26:50

PostgreSQL Data Viewer - atividade2

Grid view showing 167 rows of data. Columns include: regulated_complaints_other, AZ unregulated_complaints, AZ total_complaints, AZ total_clients_ccs_scr, AZ clients_ccs, AZ clients_scr, AZ cnpj_reclamacoes.

regulated_complaints_other	AZ unregulated_complaints	AZ total_complaints	AZ total_clients_ccs_scr	AZ clients_ccs	AZ clients_scr	AZ cnpj_reclamacoes
55	0	1	670	552	161	
56	0	1	534	418	159	
57	1	3	480	375	154	
58	0	1	428	341	137	
59	15	73	412135	268186	145105	
60	16	67	416540	268174	149521	
61	11	85	420486	268221	153423	
62	12	82	422572	268328	155393	
63	10	57	418830	267342	152627	
64	9	80	436746	267423	170417	
65	4	36	439630	267442	173252	
66	0	1	3931	3914	129	
67	1	3	4029	4004	105	
68	2	4	3672	3649	109	
69	0	1	0	0	0	
70	90	787	2456247	12581	2444170	
71	94	818	2373655	11796	2362365	
72	117	896	2283925	11993	2272412	

167 row(s) fetched - 0.760s (0.010s fetch), on 2025-08-05 at 23:26:50

Desafios encontrados

1. Lidar com novas bibliotecas como o Polars foi um desafio. Algumas vezes operações simples como ler um csv necessitava muita pesquisa e em outras algumas operações que realizadas localmente não executavam da mesma forma na lambda, como por exemplo a leitura dos arquivos no S3.
2. Não foi possível executar a lambda para inserir os dados no banco de dados relacional. Ocorreu algum comportamento inesperado que causava *timeout* nas execuções, com operações simples como coletar o segredo no secrets manager não executando. As hipóteses levantadas são relacionadas a questões de permissões ou rede, porém não houve tempo hábil para investigá-las.
3. Esse ponto foi mais um inconveniente, pois ao utilizar o terraform é necessário ter as credenciais configuradas localmente, entretanto no ambiente do laboratório as mesmas se renovam.