

# Engenharia de Dados

## Cessão de Crédito Consignado

**Autor: Alexander Almeida**

Última atualização: segunda-feira, 1 de julho de 2024

# ÍNDICE

<b>1. MVP</b>	<b>3</b>
1.1. Objetivo	3
1.2. Análise	3
1.3. Metadado	3
1.4. Arquitetura do ETL	4
1.4.1. Camada Bronze (Extração)	5
1.4.2. Camada Prata (Transformação)	6
1.4.3. Camada Ouro (Carga)	8
1.4.3.1. Dataproc	8
1.4.3.2. Linhagem	10
1.4.3.3. Dataplex (Data Quality)	10
1.4.4. Looker Stúdio (Análises)	11
1.4.4.1. Volume de idade por UF	12
1.4.4.2. Volume da margem de saldo por UF	14
1.4.4.3. Volume da margem de saldo por UF e grupo de idade	16
1.4.5. Conclusão	17
1.4.5.1. Considerações	17
1.4.5.2. Recomendações	17

## 1. MVP

### 1.1. Objetivo

O objetivo principal é analisar os dados para ser mais assertivo na venda do crédito consignado. A regra principal consiste em ser menor que 85 anos de idade, com margem de saldo superior à 35% dos parcelamentos vigentes.

O dataset possui 6176 instâncias e 9 atributos, com um mix de tipos de dados categóricos e numéricos.

### 1.2. Análise

- Qual o volume de Idade por UF?
- Qual o volume da Margem de Saldo por UF?
- Qual a relação entre UF, Idade e Margem de Saldo?

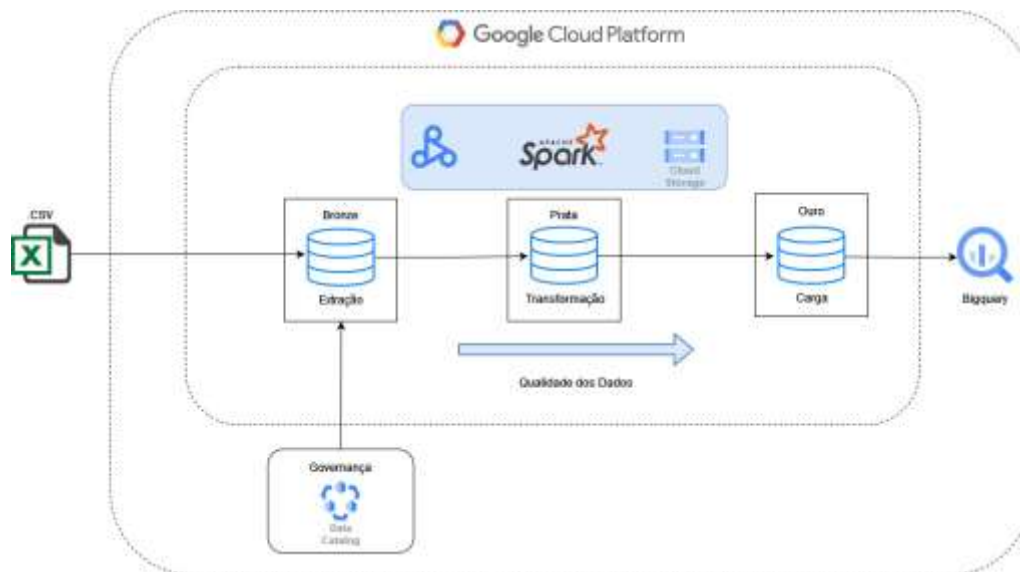
Com estas definições, poderemos montar ações estratégicas para cessão de crédito

### 1.3. Metadado

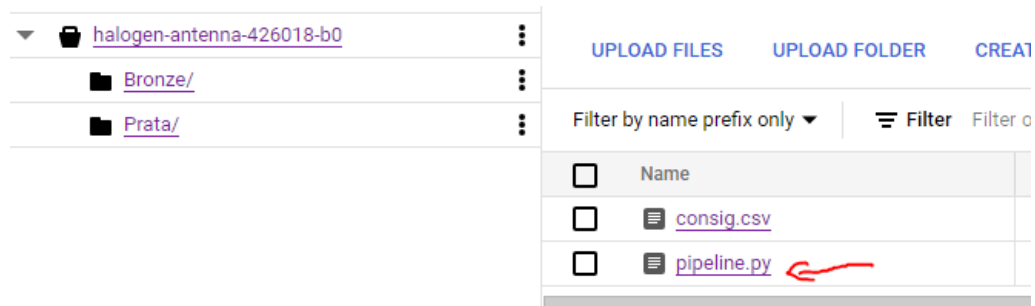
<input type="checkbox"/>	Field name	Type	Mode	Min	Max	Default value	Primary key	Description
<input type="checkbox"/>	cpf	STRING	RELIABLE	-	-	-	-	CPF do beneficiário
<input type="checkbox"/>	nome	STRING	RELIABLE	-	-	-	-	Nome do beneficiário
<input type="checkbox"/>	sexo	STRING	RELIABLE	-	-	-	-	Sexo do beneficiário
<input type="checkbox"/>	idade	INTEGER	RELIABLE	-	-	-	-	Idade do beneficiário
<input type="checkbox"/>	uf	STRING	RELIABLE	-	-	-	-	Unidade Federativa de origem do beneficiário
<input type="checkbox"/>	situacao_financeira	STRING	RELIABLE	-	-	-	-	Situação financeira: Aposentado/Previdente
<input type="checkbox"/>	tipo_contrato	STRING	RELIABLE	-	-	-	-	Contrato: Aposentadoria Contratual / Não Contratual. Pode perder a aposentadoria.
<input type="checkbox"/>	prel_valor	FLOAT	RELIABLE	-	-	-	-	Valor total dos parcelamentos vigentes
<input type="checkbox"/>	margem_saldo	FLOAT	RELIABLE	-	-	-	-	Valor total disponível para cancelo do crédito
<input type="checkbox"/>	conceder	INTEGER	RELIABLE	-	-	-	-	Regra de seleção para cessão de crédito: sendo 0=CONCEDER e 1=NÃO conceder

### 1.4. Arquitetura do ETL

Estaremos utilizando a arquitetura medalhão neste MVP, conforme figura abaixo:



Toda construção é realizada através de uma pipeline em pyspark (pipeline.py)



### 1.4.1. Camada Bronze (Extração)

Na camada bronze carreguei o arquivo .CSV no Cloud Storage GCP, pasta Bronze, em seu estado bruto. A base utilizada é particular e real, mas para apresentação neste MVP, foi necessária a anonimização do campo CPF. Esse processo de extração não foi automatizado, devido a limitação da conta "free" na GCP. Desta forma, realizei o UPLOAD manualmente, conforme figura abaixo:

```

#####
# Desde a criação do novo GCP, devido ao plano free, não conseguirei demonstrar o upload do arquivo pela pipeline. Desta forma, realizei o UPLOAD manualmente no bucket
#####

# Definição do projeto no GCP
project_id = "balogun-arquivos-426018-02"
bucket_name = "balogun-arquivos-426018-02"
destination_blob_name = "Prata"

github_file_url = "https://raw.githubusercontent.com/AlexanderAlmeida/pos_graduacao/master/pipeline.py"

def download_and_upload_to_gcp(github_file_url, bucket_name, destination_blob_name, project_id):
    # Download do arquivo do GitHub
    response = requests.get(github_file_url)
    if response.status_code == 200:
        csv_data = response.content
        # Salva o arquivo temporário em um arquivo local para análise
        temp_file_path = os.path.join(tempfile.gettempdir(), 'temp_file.csv')
        with open(temp_file_path, 'wb') as temp_file:
            temp_file.write(csv_data)
            filename = temp_file_path

    # Autenticação no Cloud Storage e criação do bucket
    storage_client = storage.Client(project=project_id)
    bucket = storage_client.get_bucket(bucket_name)

    # Upload do arquivo para o Cloud Storage
    blob = bucket.blob(destination_blob_name)
    blob.upload_from_filename(filename)

    # Delete the temporary file after upload
    # (Optional, the file will be automatically deleted when the program exits)
    os.remove(filename)

download_and_upload_to_gcp(github_file_url, bucket_name, destination_blob_name, project_id)

print(f"File downloaded from github and uploaded to bucket: {bucket_name}/{destination_blob_name}")

```

Folder browser		Buckets > balogun-arquivos-426018-02 > Bronze						
<div><div><div>balogun-arquivos-426018-02</div><div><div>Prata</div><div>Prata</div></div></div></div>		<div><div>UPLOAD FILES</div><div>UPLOAD FOLDER</div><div>CREATE FOLDER</div><div>TRANSFER DATA</div><div>SCHEDULE HOLD</div><div>EXPORT RETENTION</div><div>IMPORT/EXPORT</div><div>DELETE</div></div>						
Filter by name prefix only		Filter: Show objects and folders						
		<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified
		<input type="checkbox"/>	consig.csv	405.1 KB	Text file	25 Jun 2024, 12:34:17	Standard	25 Jun 2024, 12:34:17
		<input type="checkbox"/>	pipeline.py	4.2 KB	Application/Python	1 Jun 2024, 12:41:11	Standard	1 Jun 2024, 12:41:11

Referências:

pipeline.py:

[https://raw.githubusercontent.com/AlexanderAlmeida/pos\\_graduacao/master/pipeline.py](https://raw.githubusercontent.com/AlexanderAlmeida/pos_graduacao/master/pipeline.py)

Consig.csv:

[https://raw.githubusercontent.com/AlexanderAlmeida/pos\\_graduacao/master/consig.csv](https://raw.githubusercontent.com/AlexanderAlmeida/pos_graduacao/master/consig.csv)

### 1.4.2. Camada Prata (Transformação)

Na camada prata carreguei o dataframe com spark para início dos tratamentos de dados. Criei um contador de instâncias para observabilidade.

```
In [3]: #####
# CAMADA BRONZE (Ingestão)
#####

# Lê o dataset do Cloud Storage
df_fe = spark.read.csv("gs://halogen-antenna-426818-b0/bronze/consig.csv", header=True, inferSchema=True)
df_fe.show()

# Contando o número de linhas do dataset para observabilidade
num_rows_csv = df_fe.count()
print(f"Number of rows in CSV: {num_rows_csv}")
```

0000***2074	CRISTIANO SANTOS ...	MASCULINO	37	R5	ATIVO PERMANENTE	ESTAVEL	45.07	1361.11
0000***3748	MARIA DA GLORIA V...	FEMININO	52	R3	ATIVO PERMANENTE	ESTAVEL	37.15	2192.17
0000***3284	CERESA KAZUO MAK...	FEMININO	88	R5	APOSENTADO	ESTAVEL	1922.6	5100.92
0000***3155	SILBERTO DOS SANT...	MASCULINO	36	MG	ATIVO PERMANENTE	ESTAVEL	748.4200000000001	2054.88
0000***1234	IRIS PEDRO DE OLI...	FEMININO	75	GO	APOSENTADO	ESTAVEL	8723.39	19903.72
0000***7742	JANICE CAMPOS POTTA	FEMININO	55	R3	APOSENTADO	ESTAVEL	576.3200000000000	4161.24
0000***4786	FABIO DE JESUS RI...	MASCULINO	52	R3	ATIVO PERMANENTE	ESTAVEL	158.29	3251.0
0000***6725	GIOVANI AZEVEDO S...	MASCULINO	54	SC	NAO INFORMADO	ESTAVEL	110.0	108.8
0000***2780	TERONINO DOS SANT...	MASCULINO	58	R3	ATIVO PERMANENTE	ESTAVEL	1862.14	2888.3500000000004
0000***8580	MARCELLO PORTELA ...	MASCULINO	40	MS	ATIVO PERMANENTE	ESTAVEL	831.53	11245.95
0000***4353	JOSE MEVERGTHON A...	MASCULINO	82	SP	APOSENTADO	ESTAVEL	683.21	15200.95
0000***8065	VANDERLEI DOMINGU...	MASCULINO	45	GO	CELETISTA EMPREGADO	ESTAVEL	1640.6699999999998	2337.4800000000005
0000***4729	JOSELEI TRINDADE ...	FEMININO	50	R3	CEDIDO SUS LEI 8270	ESTAVEL	302.18	2018.96
0000***4287	JOANA CORREA DE S...	FEMININO	82	PA	APOSENTADO	ESTAVEL	45.99	1384.39
0000***3127	LARISSA DE SOUZA ...	FEMININO	36	DF	CELETISTA EMPREGADO	ESTAVEL	3887.82	8848.839999999999
0000***9582	MARIA EVINICE DE J...	FEMININO	70	BA	APOSENTADO	ESTAVEL	58.86	712.1

-----+  
only showing top 20 rows  
Number of rows in CSV: 6176

Em seguida, armazenei o dataset no Bigquery, eu seu estado ainda bruto, para qualidade dos dados e linhagem.

```
In [4]: # Autenticação no BigQuery
cliente_bq = bigquery.Client()

# Define o nome do projeto e do conjunto de dados no BigQuery
dataset_id = "CREDITO_CONSIGNADO"

# Armazenando os dados brutos no bigquery
df_fe.write.format("bigquery").option("temporaryGcsBucket", "").option("writeMethod", "DIRECT").option("project", project_id).save()

# Verificando a quantidade de registros importados para observabilidade
query = f"SELECT COUNT(*) as total_rows FROM `halogen-antenna-426818-b0.CREDITO_CONSIGNADO.tbl_consignado_raw`"
query_job = cliente_bq.query(query)
result = query_job.result()
num_rows_bq = list(result)[0].total_rows
print(f"Number of rows in BigQuery table: {num_rows_bq}")

Number of rows in BigQuery table: 6176
```

Nome	Idade	Sexo	UF	Situação Funcional	Tipo Contrato	Pmt Valor	Margem Saldo
0	70	M	AM	APOSENTADO	ESTAVEL	1340.4199999999998	4998.18
1	37	F	MT	CELETISTA EMPREGADO	ESTAVEL	3508.71	3194.28
2	55	F	RJ	APOSENTADO	ESTAVEL	2048.42	-3172.8800000000006
3	38	M	PI	ATIVO PERMANENTE	ESTAVEL	4678.62	10253.44
4	84	F	MG	BENEFICIARIO PENSÃO	NÃO ESTAVEL	1584.85	381.75
5	78	F	PE	APOSENTADO	ESTAVEL	652.84	8514.0

Na sequência foi observado no dataset, CPFs duplicados, devido a anonimização. Dito isto, foram removidas as duplicidades.

```
In [5]: #####
#CAMADA PRATA (Transformação)
#####

## Mantenha apenas a primeira ocorrência por CPF
# Verificação de duplicidade no CPF, devido a anonimização do campo para este MVP
dataset_sem_duplicatas = df_fe.withColumn("row_num", F.row_number().over(Window.partitionBy("CPF").orderBy(F.col("CPF").desc)))
dataset_sem_duplicatas = dataset_sem_duplicatas.filter(F.col("row_num") == 1)

dataset_sem_duplicatas.show()

dataset_sem_duplicatas = dataset_sem_duplicatas.filter(F.col("row_num") == 1).drop("row_num")
df_fe = dataset_sem_duplicatas
```

Como a análise será realizada por faixa de idade, foi necessária adição das mesmas ao dataset. Ao término, armazenei o novo dataset em formato parquet ao Cloud Storage GCP, na pasta Prata.

```
# Crie colunas para cada faixa etária
df_fe = df_fe.withColumn("Faixa_10_20", F.when(df_fe["idade"].between(10, 20), 1).otherwise(0))
df_fe = df_fe.withColumn("Faixa_21_30", F.when(df_fe["idade"].between(21, 30), 1).otherwise(0))
df_fe = df_fe.withColumn("Faixa_31_40", F.when(df_fe["idade"].between(31, 40), 1).otherwise(0))
df_fe = df_fe.withColumn("Faixa_41_50", F.when(df_fe["idade"].between(41, 50), 1).otherwise(0))
df_fe = df_fe.withColumn("Faixa_51_60", F.when(df_fe["idade"].between(51, 60), 1).otherwise(0))
df_fe = df_fe.withColumn("Faixa_61_70", F.when(df_fe["idade"].between(61, 70), 1).otherwise(0))
df_fe = df_fe.withColumn("Faixa_71_80", F.when(df_fe["idade"].between(71, 80), 1).otherwise(0))
df_fe = df_fe.withColumn("Faixa_81_90", F.when(df_fe["idade"].between(81, 90), 1).otherwise(0))

df_fe.show()
```

cpf	nome	sexo	idade	uf	situacao_funcional	tipo_contrato	pmt_valor	margem_saldo	
0	conceder	Faixa_10_20	Faixa_21_30	Faixa_31_40	Faixa_41_50	Faixa_51_60	Faixa_61_70	Faixa_71_80	Faixa_81_90
0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

### 1.4.3. Camada Ouro (Carga)

Basicamente o dataset foi agrupado com as colunas UF, Margem\_Saldo, Grupo de faixas e adicionado o campo id (identity). Após isso, gravamos o dataset no bigquery como tabela fato (FLAT), para realização das análises.

```
#####
# Busca pelo (type)
#####

# Cálculo e montagem por faixa de id e UF
df_fa = df_fa.groupby(["uf"])[["egg(f,sm(f,ml("mergem_idb"))),.alias("mergem_idb"),f.sm("faixa_10_20"),.alias("faixa_10_20"), f.sm("faixa_20_30"),.alias("faixa_20_30"),f.sm("faixa_30_40"),.alias("faixa_30_40"),f.sm("faixa_40_50"),.alias("faixa_40_50"),f.sm("faixa_50_60"),.alias("faixa_50_60"),f.sm("faixa_60_70"),.alias("faixa_60_70"),f.sm("faixa_70_80"),.alias("faixa_70_80"),f.sm("faixa_80_90"),.alias("faixa_80_90"),f.sm("faixa_90_100"),.alias("faixa_90_100")]]

# Cálculo de nome id
df_fa = df_fa.alias("id", f.monotonically_increasing_id())

# Montagem de coluna e deleção de última coluna
df_fa = df_fa.select(["id", *df_fa.columns[1:-1]])

# Cálculo e deleção completa
df_fa.show()

# Salvo o dataset processado no BigQuery
df_fa.write.format("bigquery").option("temporaryIdentifier", "").option("writeMethod", "DIRECT").option("project", project_id).option("dataset", dataset_id).option("table", "tbl_fa")

# Fim o script Apuracao
print("Fim")
```

id	uf	Margem_Saldo	Faixa_10_20	Faixa_21_30	Faixa_31_40	Faixa_41_50	Faixa_51_60	Faixa_61_70	Faixa_71_80	Faixa_81_90
0	MS	253197.910000000003	1	1	84	5	4	5	3	4
1	CE	1248672.97	1	0	105	7	3	6	73	93
2	MG	1888138.9190999974	0	1	21	167	34	32	62	110
3	DF	3392125.6299999985	0	8	417	47	20	6	82	108
4	RO	40581.68	1	13	21	1	1	2	4	0
5	AM	610785.12000000005	1	13	37	0	0	0	42	34
6	MT	375235.17000000016	1	0	55	4	4	4	10	13
7	SP	1064686.47000000007	1	2	48	28	18	142	78	50
8	PB	315985.21	0	1	9	59	9	6	14	9
9	BA	863103.37999999999	0	0	67	58	10	10	38	52
10	SE	280744.18000000005	0	0	35	19	1	2	6	1
11	RJ	1361111.95999999997	0	1	15	57	681	130	120	64
12	AC	64217.089999999999	0	8	11	1	2	1	5	3
13	PR	581894.58000000003	0	1	26	17	3	7	34	42
14	AP	116248.910000000002	0	6	22	0	1	0	18	6
15	RR	16245.420000000002	1	8	19	4	1	1	1	0
16	TO	154264.8	0	1	36	8	4	0	2	1
17	ES	215858.67	0	0	1	8	27	5	6	5
18	MD	1498.5	2	0	0	0	0	0	0	0
19	AL	124588.56	0	0	5	16	4	5	1	8

only showing top 20 rows

Field name	Type	Mode	Key	Collate	Default value	Policy type	Description
id	INTEGER	SECURE					campo de identificação único
uf	STRING	MULTI					unidade federativa
Margem,Saldo	FLOAT	MULTI					Valor disponível para cancelo de crédito
Faixa_10_20	INTEGER	MULTI					Faixa etária agrupada entre 10 e 20 anos
Faixa_21_30	INTEGER	MULTI					Faixa etária agrupada entre 21 e 30 anos
Faixa_31_40	INTEGER	MULTI					Faixa etária agrupada entre 31 e 40 anos
Faixa_41_50	INTEGER	MULTI					Faixa etária agrupada entre 41 e 50 anos
Faixa_51_60	INTEGER	MULTI					Faixa etária agrupada entre 51 e 60 anos
Faixa_61_70	INTEGER	MULTI					Faixa etária agrupada entre 71 e 80 anos
Faixa_71_80	INTEGER	MULTI					Faixa etária agrupada entre 81 e 90 anos
Faixa_81_90	INTEGER	MULTI					

OBS.: Utilizamos o **modelo flat** para o data warehouse.

#### 1.4.3.1. Dataproc

Selecioneamos o Google DataProc como gerenciador do Spark e Hadoop, para execução de toda pipeline deste projeto. Criamos o JOB denominado Pipeline\_FULL, apontando para o arquivo pyspark (pipeline.py), armazenado no bucket Bronze. Abaixo a saída de todo fluxo:



[illegible]

Number of rows in CWA: 6072

24/07/02 18:02:08 INFO SparkBigQueryConnectorModule: Registering cleanup task listener, should happen just once

```
24/07/01 10:02:18 INFO SparkQuery$DefaultQueryPlan: Registering Cleanup Job listener, should happen just once
24/07/01 10:03:01 INFO ElgQuery/DirectDataSourceWriterContext: ElgQuery DataSource writer 87b4210c-d94e-4894-a182-12a34e0c3455 committed with messages
```

```

2017/07/19 15:01:02 1540 httpgo.DirectDatabaseWriter[terusted]: httpgo DatabaseWriter has committed at time: seconds: 1719646662
name: 7X7863660

```

Number of rows in BigQuery table: 612

	id	cpf	nome	assin_lidado	uf	situacao_funcional	tipo_contrato	prez_valor	numemp_receboconce	num_emp
1	#####12871	LEANDRO ALBERTO S ...	F	PR	AM	APONTADO	ESTAVEL	15.688.189.000.00...	8908.121	0
2	#####7164	ARACILIA MARIA DE C...	F	PR	PE	INTERDITARIA PERMANENTE	ESTAVEL	7506.731	3104.281	0
3	#####9999	JOSE GABRIEL SOARES DA S...	F	PR	PE	APONTADO	ESTAVEL	11.725.000.000.00...	1016.14	0
4	#####9999	DANIELLE MACHADO DE ...	F	PR	PE	ATIVO PERMANENTE	ESTAVEL	1076.612	18203.401	0
5	#####9473	SILVANA DE WITTON ...	F	PR	DA	PROBETECIARIA PERMAN	NÃO ESTAVEL	1258.301	3561.73	1
6	#####9463	OLIVIO EDUARDO GONÇAL...	F	PR	PE	APONTADO	ESTAVEL	621.041	8524.0	0
7	#####1186	ANDRÉ VICTOR VIEIRA E...	F	PR	PE	APONTADO	ESTAVEL	137.091	3378.31	0
8	#####4281	CHARLES DAMASCENO ...	F	PR	AC	ATIVO PERMANENTE	ESTAVEL	327.127.281.666.000.00...	1016.14	0
9	#####4262	PAULINE ALVES SARAIVA	F	PR	BA	ATIVO PERMANENTE	ESTAVEL	3080.51	4150.37	0
10	#####9999	JOSE GREGO DA S...	F	PR	PE	INTERDITARIA PERMANENTE	ESTAVEL	517.825	3811.18	0
11	#####9999	WILLIAM CALVO DE ...	F	PR	PE	ATIVO PERMANENTE	ESTAVEL	1.499.679.000.00...	1016.14	0
12	#####9233	OLIVIA FLORENCE H...	F	PR	BA	ATIVO PERMANENTE	ESTAVEL	634.21	2095.21	0
13	#####12755	MARIA HELENA COM...	F	PR	SP	PROBETECIARIA PERMAN	NÃO ESTAVEL	1837.101	3477.0	1
14	#####40888	LEONARDO RABELO	F	PR	PE	APONTADO	ESTAVEL	16.77	2088.85	0
15	#####8888	SARA JOSE SOARES	F	PR	TO	ATIVO PERMANENTE	ESTAVEL	2566.87	-2007.30	1
16	#####1275	RISSA DE CARVALHO ...	F	PR	PE	INTERDITARIA PERMAN	NÃO ESTAVEL	1577.431	15.647.599.999.00...	1
17	#####3823	MARIA DO GASTO L...	F	PR	BA	APONTADO	ESTAVEL	24.161.698.000.00...	13.781.898.000.00...	0
18	#####9999	JOSE WILSON ...	F	PR	PE	INTERDITARIA PERMAN	NÃO ESTAVEL	124.101.698.000.00...	1016.14	1
19	#####95739	CATIA REGINA CAR...	F	PR	DA	PROBETECIARIA PERMAN	NÃO ESTAVEL	14.000.180.000.00...	-15.249.799.999.00...	1
20	#####9999	JUSTICA HILBERT ...	F	PR	PE	ATIVO PERMANENTE	ESTAVEL	3080.50	5760.1	0

[illegible]

with a standing foot 20 inch

[illegible][illegible]

only window for 10 min.

2016/07/05 13:09:10 INFO DirectDataSinkWriterContext: BigQuery DataSource writer: 944c1f30-86d3-43da-af52-882a607ad370 committed with messages

```
2019-08-16 15:03:19 INFO BigQueryDirectDataWriterContext: BigQuery DataSource writer has committed at time: second: 1719578989
[BigQueryWriteCommitMessage{partitionId=0, taskId=0, epoch=1719578989534, taskIdAsProjectPath=warehouse-426818-bb/datasets/CRCCTO2_CONSOLIDATED/taskId_0}]
2019-08-16 15:03:20 INFO BigQueryDirectDataWriterContext: BigQuery DataSource writer has committed at time: second: 1719578989
```

089673-572095-0000

2007/01 10:00:20 INFO SparkBigQueryConnectModule: In SparkListener notification, going to activate cleanup loop

24/07/93 10:00:20 INFO HighwaysClient: Running cleanup task. task count is 0

26/07/91 18:00:20 1993 Highway-Client: Clearing the cleanup jobs list

```
26/07/92 18:03:20 INFO BlahemClient: Cleaning the cleaning jobs list
26/07/92 18:03:30 INFO BlahemClient: Finished to run cleaning jobs
```

[illegible]

ID	Job ID	Status	Project	Type	Client	Start Time	Report Time	Cost
1	Project 1	Completed	Acme Corp	Project	John Doe	2023-01-01 10:00:00	2023-01-05 15:00:00	\$1000

#### 1.4.3.2. Linhagem

Como os dados não sofreram alteração, por ser uma base real já tratada, não conseguirei detalhamento sobre o tema. Deixarei o print abaixo, como forma de representação do controle no bigquery:



#### 1.4.3.3. Dataplex (Data Quality)

Utilizamos o Dataplex para realização do Data Quality. Foram adicionadas regras de negócio para validação dos campos do dataset original. Importante lembrar que os dados de origem já estavam previamente tratados, por se tratar de uma base privada.

Configuração do Dataplex com as regras de negócio:

Filter	Filter items								
	Clickable name ↑	File name	File type	Publication	Element	Properties	Thumbnail	Actions	
<input type="checkbox"/>	calculator	Calculator (File)	File icon	Validity	get of 1.0		100%		
<input type="checkbox"/>	all	Full Check	File icon	Completed			100%		
<input type="checkbox"/>	all	Run Conditions Check	File icon	Validity	((getOf(1) > 1) AND ((getOf(1) > 1) OR getOf(100)))		100%		
<input type="checkbox"/>	back	Full Check	File icon	Completed			100%		
<input type="checkbox"/>	manager_valid	Full Check	File icon	Completed			100%		
<input type="checkbox"/>	manager_valid	Run Conditions Check	File icon	Validity	((getOf(1) > manager_valid) > 1 AND ((getOf(1) > manager_valid) > 1) OR ((getOf(1) > manager_valid) > 100))		100%		
<input type="checkbox"/>	name	Full Check	File icon	Completed			100%		
<input type="checkbox"/>	print_valid	Full Check	File icon	Completed			100%		
<input type="checkbox"/>	print_valid	Run Conditions Check	File icon	Validity	((getOf(1) > print_valid) > 1 AND ((getOf(1) > print_valid) > 1) OR ((getOf(1) > print_valid) > 100))		100%		
<input type="checkbox"/>	size	Full Check	File icon	Completed			100%		
<input type="checkbox"/>	size	Calculator (File)	File icon	Validity	get of ((getOf(1) > 1) OR 100)		100%		
<input type="checkbox"/>	size	Run Conditions Check	File icon	Validity	((getOf(1) > size) > 1 AND ((getOf(1) > size) > 1) OR ((getOf(1) > size) > 100))		100%		
<input type="checkbox"/>	ui	Full Check	File icon	Completed			100%		
<input type="checkbox"/>	ui	Run Conditions Check	File icon	Validity	((getOf(1) > ui) > 1 AND ((getOf(1) > ui) > 1) OR ((getOf(1) > ui) > 100))		100%		

Sucesso na validação do Job:

Job ID	Start Date	Records scanned	Job Status	Auto quality result
02273145-53-426-8084-MPH-149116	1 July 2022 at 16:05:24 (UTC-2)	8475	Completed	Passed

### Detalhamento dos testes realizados:

Job ID: 02372147-1c30-42a1-b086-1479614841fe results

Overall Score

100%

10 PASSED / 16.13

Validity

100%

Completeness

100%

Rules

Filter

Other items

Control name	Rule name	Rule type	Status	Validation	Dimension	Parameters	Failed count	Threshold	Steps to get failed records
connector	Value Not Check	Passed	Passed	Per row	Validity	set of 1,2	0%	100%	WITH 32572747-1-35-42x1-8000
col1	Null Check	Passed	Passed	Per row	Completeness		0%	100%	WITH 32572747-1-35-42x1-8000
col1	Row Cardinality Check	Passed	Passed	Per row	Validity	2,8000000 col1 <= 11 8000	0%	100%	WITH 32572747-1-35-42x1-8000
col2	Null Check	Passed	Passed	Per row	Completeness		0%	100%	WITH 32572747-1-35-42x1-8000
margin_col2	Null Check	Passed	Passed	Per row	Completeness		0%	100%	WITH 32572747-1-35-42x1-8000
margin_col2	Row Cardinality Check	Passed	Passed	Per row	Validity	2,8000000 margin_col2 <= 11 8000	0%	100%	WITH 32572747-1-35-42x1-8000
row1	Null Check	Passed	Passed	Per row	Completeness		0%	100%	WITH 32572747-1-35-42x1-8000
row1_col1	Null Check	Passed	Passed	Per row	Completeness		0%	100%	WITH 32572747-1-35-42x1-8000
row1_col1	Row Cardinality Check	Passed	Passed	Per row	Validity	2,8000000 row1_col1 <= 11 8000	0%	100%	WITH 32572747-1-35-42x1-8000
row2	Null Check	Passed	Passed	Per row	Completeness		0%	100%	WITH 32572747-1-35-42x1-8000
row2	Value Not Check	Passed	Passed	Per row	Validity	set of 1	0%	100%	WITH 32572747-1-35-42x1-8000

#### 1.4.4. Looker Stúdio (Análises)

O Looker Stúdio é uma ferramenta de análise acoplada ao Bigquery.

The screenshot shows the Tableau Desktop interface. The main view displays a data table named 'tbl\_fato'. The table has columns for various metrics, including 'Fato\_10\_28', 'Fato\_11\_28', 'Fato\_12\_28', 'Fato\_13\_28', 'Fato\_14\_28', 'Fato\_15\_28', 'Fato\_16\_28', 'Fato\_17\_28', 'Fato\_18\_28', and 'Fato\_19\_28'. The table is filtered by 'Fato\_10\_28' and 'Fato\_11\_28'. The right sidebar shows the 'Filters' shelf with 'Fato\_10\_28' and 'Fato\_11\_28' listed. The 'Columns' shelf is empty. The 'Rows' shelf is empty. The 'Marks' shelf is set to 'Sum'. The 'Marks' card is expanded, showing 'Sum' and 'Count' options. The 'Columns' card is expanded, showing 'Fato\_10\_28' and 'Fato\_11\_28' options. The 'Rows' card is expanded, showing 'Fato\_10\_28' and 'Fato\_11\_28' options. The 'Marks' card is expanded, showing 'Sum' and 'Count' options. The 'Columns' card is expanded, showing 'Fato\_10\_28' and 'Fato\_11\_28' options. The 'Rows' card is expanded, showing 'Fato\_10\_28' and 'Fato\_11\_28' options.

### 1.4.4.1. Volume de idade por UF

Neste item, precisamos entender se faz sentido segmentarmos a base pela UF e grupo de faixas de atrasos.

Baseado no gráfico abaixo, percebemos nitidamente uma aglomeração muito alta na faixa de 31\_40, para minha "surpresa". Seguidos pela faixa de 81\_90, 51\_60 e 71\_80.

Consulta:

```
SELECT
  distinct(uf) as uf,
  Margem_Saldo as saldo,
  Faixa_10_20,
  Faixa_21_30,
  Faixa_31_40,
  Faixa_41_50,
  Faixa_51_60,
  Faixa_61_70,
  Faixa_71_80,
  Faixa_81_90
FROM `halogen-antenna-426018-b0.CREDITO_CONSIGNADO.tbl_fato`
--group by uf
order by Margem_Saldo desc
```

Execução no Bigquery:

Query results

Row	uf	Faixa_10_20	Faixa_21_30	Faixa_31_40	Faixa_41_50	Faixa_51_60	Faixa_61_70	Faixa_71_80	Faixa_81_90
0	AC	1	1	54	5	4	5	3	4
1	CE	1	0	100	1	0	0	70	61
2	MG	0	0	21	157	84	83	60	110
3	SP	0	0	417	27	20	6	82	106
4	RS	1	19	31	1	1	0	4	0
5	AM	1	13	37	0	0	0	42	34
6	MT	0	0	33	4	4	4	10	14
7	DF	1	2	44	30	10	162	76	20
8	PR	0	0	9	36	6	9	16	9
9	BA	0	0	37	36	10	10	38	30
10	GO	0	0	39	16	1	2	0	1
11	SC	0	0	13	37	601	136	120	64
12	AL	0	0	11	1	2	0	0	0
13	PA	0	0	26	17	9	0	34	21
14	AP	0	0	33	0	1	0	19	0

	uf	Faixa_10_20	Faixa_21_30	Faixa_31_40...	Faixa_51_60	Faixa_61_70	Faixa_71_80	Faixa_81_90
1.	RS	0	1	467	9	5	19	37
2.	DF	0	8	417	20	6	82	108
3.	GO	0	3	123	4	2	6	5
4.	CE	1	0	105	3	6	73	93
5.	PI	0	1	88	2	1	26	29
6.	MS	1	1	84	4	5	3	4
7.	MA	1	0	76	2	2	16	28
8.	BA	0	0	67	10	10	38	52
9.	PA	1	21	57	2	0	95	83
10.	MT	0	0	55	4	4	10	13
11.	SP	1	2	48	18	142	78	30
12.	AM	1	13	37	0	0	42	34
13.	TO	0	1	36	4	0	2	1
14.	SE	0	0	35	1	2	6	1
15.	SC	0	0	32	10	5	12	14
16.	PR	0	1	26	3	7	34	42
17.	AP	0	6	22	1	0	18	6
18.	RO	1	13	21	1	2	4	0
19.	MG	0	1	21	34	32	62	110
20.	RR	1	8	19	1	1	1	0
Total geral		8	90	1.898	865	384	813	874

Já em relação a UF, notamos como aglomeração nas UFs: RS,DF,GO,CE para faixa 31\_40:

	uf	Faixa_10_20	Faixa_21_30	Faixa_31_40...	Faixa_51_60	Faixa_61_70	Faixa_71_80	Faixa_81_90
1.	RS	0	1	467	9	5	19	37
2.	DF	0	8	417	20	6	82	108
3.	GO	0	3	123	4	2	6	5
4.	CE	1	0	105	3	6	73	93
5.	PI	0	1	88	2	1	26	29
6.	MS	1	1	84	4	5	3	4
7.	MA	1	0	76	2	2	16	28
8.	BA	0	0	67	10	10	38	52
9.	PA	1	21	57	2	0	95	83
10.	MT	0	0	55	4	4	10	13
11.	SP	1	2	48	18	142	78	30
12.	AM	1	13	37	0	0	42	34
13.	TO	0	1	36	4	0	2	1
14.	SE	0	0	35	1	2	6	1
15.	SC	0	0	32	10	5	12	14
16.	PR	0	1	26	3	7	34	42
17.	AP	0	6	22	1	0	18	6
18.	RO	1	13	21	1	2	4	0
19.	MG	0	1	21	34	32	62	110
20.	RR	1	8	19	1	1	1	0
Total geral		8	90	1.898	865	384	813	874

Na faixa de 81\_90, notamos interseção apenas na UF de DF:

	uf	Faixa_10_20	Faixa_21_30	Faixa_31_40	Faixa_51_60	Faixa_61_70	Faixa_71_80	Faixa_81_90...
1	MG	0	1	21	34	32	62	110
2	DF	0	8	417	20	6	82	108
3	CE	1	0	105	3	6	73	93
4	PE	0	0	9	6	3	35	92
5	PA	1	21	57	2	0	95	83
6	RJ	0	1	15	681	130	120	64
7	BA	0	0	67	10	10	38	52
8	PR	0	1	26	3	7	34	42
9	RS	0	1	467	9	5	19	37
10	AM	1	13	37	0	0	42	34
11	SP	1	2	48	18	142	78	30
12	PI	0	1	88	2	1	26	29
13	MA	1	0	76	2	2	56	28
14	SC	0	0	32	10	5	12	14
15	MT	0	0	55	4	4	10	13
16	PB	0	1	9	9	6	14	9
17	AL	0	0	5	4	5	1	8
18	AP	0	6	22	1	0	18	6
19	GO	0	3	125	4	2	6	5
20	ES	0	0	1	27	5	6	5
Total geral		8	90	1.898	865	384	813	874

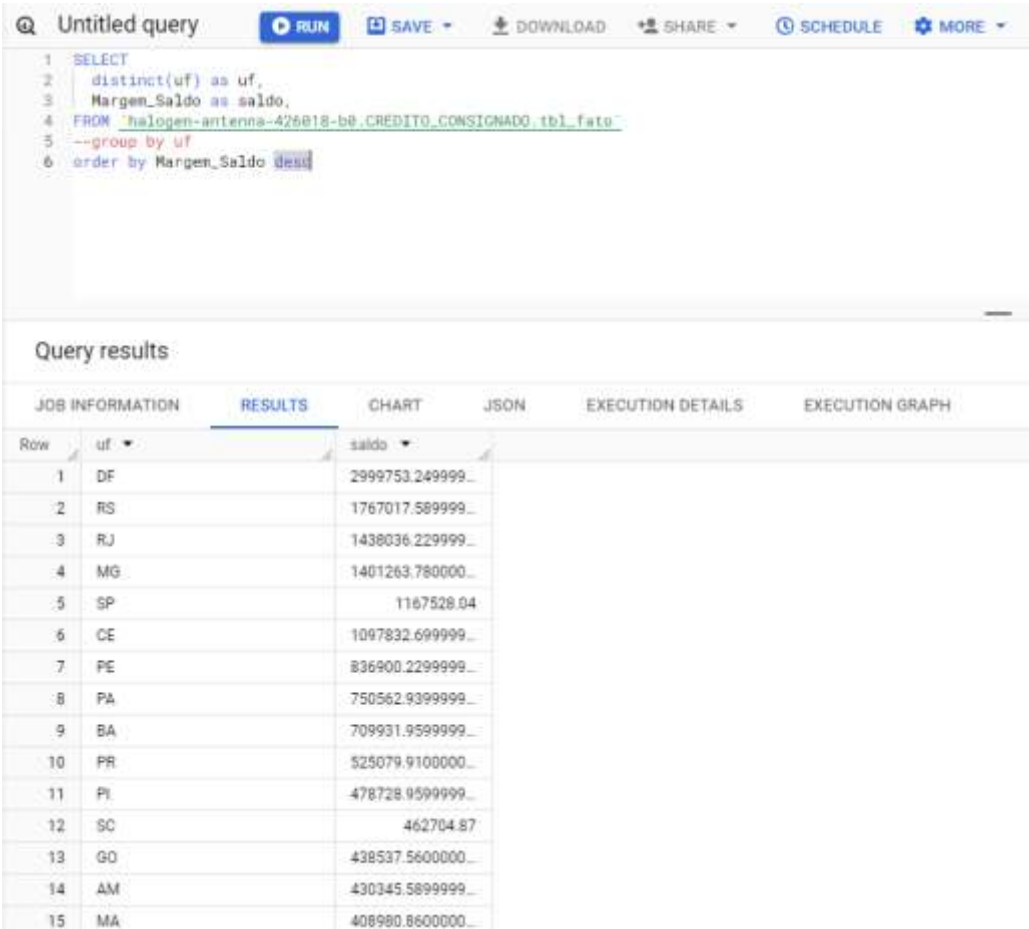
#### 1.4.4.2. Volume da margem de saldo por UF

Neste item, precisamos entender se faz sentido segmentarmos a base pela UF e margem de saldo. Através do gráfico abaixo, percebemos que DF, RS, RJ, MG, SP e CE possuem um valor financeiro considerável.

Consulta:

```
SELECT
  distinct(uf) as uf,
  Margem_Saldo as saldo,
FROM `halogen-antenna-426018-b0.CREDITO_CONSIGNADO.tbl_fato`
order by Margem_Saldo desc
```

Execução no Bigquery:



tbl\_fato



### 1.4.4.3. Volume da margem de saldo por UF e grupo de idade

Neste último item, precisamos entender se faz sentido segmentarmos a base pelo saldo, uf e grupo de idade. Utilizamos como referência a faixa de 31\_40, devido ao alto volume encontrado no item 1.4.4.1, deste documento.

Consulta:

```
SELECT
  distinct(uf) as uf,
  Margem_Saldo as saldo,
  Faixa_10_20,
  Faixa_21_30,
  Faixa_31_40,
  Faixa_41_50,
  Faixa_51_60,
  Faixa_61_70,
  Faixa_71_80,
  Faixa_81_90
FROM `halogen-antenna-426018-b0.CREDITO_CONSIGNADO.tbl_fato`
order by Margem_Saldo desc
```

Execução no Bigquery:

Pos	uf	Margem_Saldo	Faixa_10_20	Faixa_21_30	Faixa_31_40	Faixa_41_50	Faixa_51_60	Faixa_61_70	Faixa_71_80	Faixa_81_90
1	RS	2993753.249999	0	9	417	47	20	6	82	108
2	DF	1767877.589999	0	8	467	75	9	5	19	37
3	GO	1488836.229999	0	1	125	57	681	130	120	64
4	CE	1481263.780000	0	1	105	607	34	82	82	719
5	PI	1167528.34	1	2	48	28	18	142	78	30
6	CE	1087862.669999	1	0	105	7	3	6	73	88
7	PE	896982.229999	0	0	9	13	6	0	39	82
8	PA	750562.509999	1	21	57	4	2	0	95	83
9	BA	709431.659999	0	0	67	58	10	10	38	52
10	PR	678079.910000	0	1	55	11	9	9	34	42
11	PI	478720.889999	0	1	88	5	2	1	28	29
12	SC	462704.87	0	0	33	28	10	3	12	14
13	SP	438387.980000	0	9	138	9	4	0	9	9
14	MS	400345.589999	1	15	57	5	0	0	42	34

Através do gráfico abaixo, percebemos que RS, DF, GO e CE como UFs relevantes.

uf	saldo	Faixa_10_20	Faixa_21_30	Faixa_31_40	Faixa_41_50	Faixa_51_60	Faixa_61_70	Faixa_71_80	Faixa_81_90
1. RS	1.767.017,59	0	1	467	75	9	5	19	37
2. DF	2.999.753,25	0	8	417	47	20	6	82	108
3. GO	438.537,36	0	3	125	9	4	2	6	5
4. CE	1.097.832,7	1	0	105	7	3	6	73	88
5. PI	478.728,96	0	1	88	5	2	1	28	29
6. MS	200.709,07	1	1	84	5	4	5	3	4
7. MA	408.980,86	1	0	76	5	2	2	16	28
8. BA	709.931,96	0	0	67	58	10	10	38	52
9. PA	750.562,94	1	21	57	4	2	0	95	83
10. MT	262.231,06	0	0	55	4	4	4	10	13
11. SP	1.167.528,04	1	2	48	28	18	142	78	30



## **1.4.5. Conclusão**

### **1.4.5.1. Considerações**

A faixa de 31 à 40 anos se destaca em termos de volume e margem de saldo, indicando um potencial interessante para oferta do crédito consignado. Confesso que fiquei muito surpreso, pois não tinha essa visão. Concentrava grande parte das ações nas faixas superiores à 50 anos.

As UF: RS,DS,GO,CE apresentam concentração significativa na faixa etária desejada e com margem de saldo favorável, tornando as áreas prioritárias para prospecção.

A análise detalhada por UF e faixa etária permite identificar oportunidades específicas para cada região, otimizando as estratégias de venda.

### **1.4.5.2. Recomendações**

- Focar na faixa etária de 31 a 40 anos: Direcionar esforços de marketing e vendas para este público, que apresenta maior potencial de conversão.
- Priorizar as UFs RS, DF, GO e CE: Concentrar ações nestas regiões, onde existe maior concentração de clientes com perfil adequado ao crédito consignado.
- Analisar dados por UF e faixa etária: Segmentar as análises para identificar oportunidades específicas em cada região e faixa etária, personalizando as ofertas e otimizando os resultados.
- Monitorar indicadores de performance: Acompanhar métricas como taxa de aprovação, volume de concessões e inadimplência para avaliar a efetividade das estratégias e realizar ajustes quando necessário.