DOCUMENTAÇÃO

1. Criação do **projeto -** “**projeto3-supercaja”**
    a. Vídeo auxiliar para criação **-** LINK
2. Logo em seguida, criação do **banco de dados** - **“projeto3_dataset”** - em que seriam importados as seguintes tabelas disponibilizadas em **CSV**:
    a. default
    b. loans_detail
    c. loans_outstanding
    d. user_info

3. 🟦 2.1 Processar e preparar a base de dados

a. 2.1.1 🔵 Conectar/importar dados para ferramentas ✅
b. 2.1.2 🔵 **Identificar** e tratar valores nulos ✅
    i. Primeiro, identificar os valores **nulos** de cada tabela:
        1. default:
            a. `default_flag: 0`
            b. `user_id: 0`
        2. loans_detail
            a. `more_90_days_overdue: 0`
            b. `using_lines_not_secured_personal_assets: 0`
            c. `number_times_delayed_payment_loan_30_59_days: 0`
            d. `debt_ratio: 0`
            e. `number_times_delayed_payment_loan_60_89_days: 0`
        3. loans_outstanding:
            a. `loan_id: 0`
            b. `user_id: 0`
            c. `loan_type: 0`
        4. user_info:
            a. `user_id: 0`
            b. `age: 0`
            c. `sex: 0`
            d. `last_month_salary: 7199`
            e. `number_dependents: 943`

    ii. Query usadas - LINK (Bigquery):
        1.

```sql
# Tabela: DEFAULT
SELECT
  COUNT(*)
FROM
  `projeto3-supercaja.projeto3_dataset.default`
WHERE default_flag IS NULL
   OR user_id IS NULL

-------------------------------------------------------

# Tabela: LOANS_DETAIL
SELECT
  COUNT(*)
FROM
  `projeto3-supercaja.projeto3_dataset.loans_detail`
WHERE more_90_days_overdue IS NULL
   OR using_lines_not_secured_personal_assets IS NULL
   OR number_times_delayed_payment_loan_30_59_days IS
NULL
   OR debt_ratio IS NULL
   OR number_times_delayed_payment_loan_60_89_days IS
NULL

-------------------------------------------------------

# Tabela: LOANS_OUTSTANDING
SELECT
  COUNT(*)
FROM

`projeto3-supercaja.projeto3_dataset.loans_outstanding
`
WHERE loan_id IS NULL
   OR user_id IS NULL
   OR loan_type IS NULL

-------------------------------------------------------

# Tabela: USER_INFO >>> deu erro
SELECT COUNT(*)
FROM `projeto3-supercaja.projeto3_dataset.user_info`
WHERE user_id IS NULL
   OR age IS NULL
```

```sql
   OR sex IS NULL
   OR last_month_salary IS NULL
   OR number_dependents IS NULL

# NULOS DE CADA COLUNA
SELECT
  'user_id' AS coluna,
  COUNT(*) AS quantidade_nulos
FROM `projeto3-supercaja.projeto3_dataset.user_info`
WHERE user_id IS NULL

UNION ALL

SELECT
  'age' AS coluna,
  COUNT(*)
FROM `projeto3-supercaja.projeto3_dataset.user_info`
WHERE age IS NULL

UNION ALL

SELECT
  'sex' AS coluna,
  COUNT(*)
FROM `projeto3-supercaja.projeto3_dataset.user_info`
WHERE sex IS NULL

UNION ALL

SELECT
  'last_month_salary' AS coluna,
  COUNT(*)
FROM `projeto3-supercaja.projeto3_dataset.user_info`
WHERE last_month_salary IS NULL

UNION ALL

SELECT
  'number_dependents' AS coluna,
  COUNT(*)
FROM `projeto3-supercaja.projeto3_dataset.user_info`
WHERE number_dependents IS NULL
```

```
#TESTE 2(opcional)
SELECT
  COUNT(CASE WHEN user_id IS NULL THEN 1 END) AS
nulos_user_id,
  COUNT(CASE WHEN age IS NULL THEN 1 END) AS
nulos_age,
  COUNT(CASE WHEN sex IS NULL THEN 1 END) AS
nulos_sex,
  COUNT(CASE WHEN last_month_salary IS NULL THEN 1
END) AS nulos_salary,
  COUNT(CASE WHEN number_dependents IS NULL THEN 1
END) AS nulos_dependents
FROM `projeto3-supercaja.projeto3_dataset.user_info`
```

iii. **Tratamento** dado aos campos nulos:

1. Calculei a **média e mediana** no Python. Caso resolva/precise usar, pois a mediana não é calculada no Bigquery - LINK;
    a. Média: 6675.05
    b. Mediana: 5400.0
2. Para tratar o valor nulo da tabela “user_infor”, já trouxe a mediana calculada no Python, pois o SQL não possui essa função. Criei uma nova tabela, preservando a original:
3. Criada uma **NOVA TABELA** - user_info2

```
# TRATAMENTO do NULO:

CREATE OR REPLACE TABLE
`projeto3-supercaja.projeto3_dataset.user_info2`
AS
SELECT
  user_id,
  age,
  number_dependents,
  -- substituição de NULL pela mediana (5400.0)
  IFNULL(last_month_salary, 5400.0) AS
last_month_salary
FROM
`projeto3-supercaja.projeto3_dataset.user_info`;
```

4. Tomei essa decisão, por temer algum erro ao tentar encontrar Outlier pelo Bigquery.

c. 2.1.3 🔵 Identificar e tratar **valores duplicados** ✅:
   i. Nas tabelas **Default, Loans_detail e User_info** foi feita a busca por duplicados através da variável **user_id** (identificação única de cada cliente);
   ii. Já na tabela **Loans_outstanding**, a variável única e que não poderia se repetir é **loan_id** (identificação única para os empréstimos);
   iii. Em ambos, **não há** registros de valores duplicados;
   iv. Querys usadas - (LINK - BigQuery):

1.
```
# tabela DEFAULT

SELECT
  user_id,
  COUNT(*) AS qtd
FROM
  `projeto3-supercaja.projeto3_dataset.default`
GROUP BY
  user_id
HAVING
  COUNT(*) > 1;
```

2.
```
# tabela LOANS_DETAIL

SELECT
  user_id,
  COUNT(*) AS qtd
FROM
  `projeto3-supercaja.projeto3_dataset.loans_detail`
GROUP BY
  user_id
HAVING
  COUNT(*) > 1;
```

3.
```
# tabela LOANS_OUTSTANDING

SELECT *
FROM
`projeto3-supercaja.projeto3_dataset.loans_outstanding
```

```
`
WHERE loan_id IN (
  SELECT loan_id
  FROM
`projeto3-supercaja.projeto3_dataset.loans_outstanding
`
  GROUP BY loan_id
  HAVING COUNT(*) > 1
);
```

4.

```
# Tabela USER_INFO

SELECT *
FROM `projeto3-supercaja.projeto3_dataset.user_info`
WHERE user_id IN (
  SELECT user_id
  FROM `projeto3-supercaja.projeto3_dataset.user_info`
  GROUP BY user_id
  HAVING COUNT(*) > 1
);
```

5. **Resultado: não** foram encontrados valores duplicados nas tabelas analisadas..

d. 2.1.4🔵 Identificar e gerenciar dados fora do escopo da análise ✅

i. Optei por fazer a correlação de todas variáveis. Fórmula extensa, mas ajuda a avaliar correlações que possam escapar a olho nu;

1.

```
WITH base AS (
  SELECT
    d.user_id,
    SAFE_CAST(d.default_flag AS FLOAT64) AS
default_flag,
    SAFE_CAST(l.more_90_days_overdue AS FLOAT64) AS
more_90_days_overdue,

SAFE_CAST(l.using_lines_not_secured_personal_assets AS
FLOAT64) AS using_lines_not_secured_personal_assets,

SAFE_CAST(l.number_times_delayed_payment_loan_30_59_da
```

```
ys AS FLOAT64) AS delay_30_59,
    SAFE_CAST(l.debt_ratio AS FLOAT64) AS debt_ratio,

SAFE_CAST(l.number_times_delayed_payment_loan_60_89_da
ys AS FLOAT64) AS delay_60_89,
    SAFE_CAST(u.age AS FLOAT64) AS age,
    SAFE_CAST(u.last_month_salary AS FLOAT64) AS
last_month_salary,
    SAFE_CAST(u.number_dependents AS FLOAT64) AS
number_dependents
  FROM `projeto3-supercaja.projeto3_dataset.default` d
  LEFT JOIN
`projeto3-supercaja.projeto3_dataset.loans_detail` l
ON d.user_id = l.user_id
  LEFT JOIN
`projeto3-supercaja.projeto3_dataset.user_info` u ON
d.user_id = u.user_id
),

pairs AS (
  SELECT 'default_flag' AS var1,
'more_90_days_overdue' AS var2, CORR(default_flag,
more_90_days_overdue) AS corr FROM base UNION ALL
  SELECT 'default_flag',
'using_lines_not_secured_personal_assets',
CORR(default_flag,
using_lines_not_secured_personal_assets) FROM base
UNION ALL
  SELECT 'default_flag', 'delay_30_59',
CORR(default_flag, delay_30_59) FROM base UNION ALL
  SELECT 'default_flag', 'debt_ratio',
CORR(default_flag, debt_ratio) FROM base UNION ALL
  SELECT 'default_flag', 'delay_60_89',
CORR(default_flag, delay_60_89) FROM base UNION ALL
  SELECT 'default_flag', 'age', CORR(default_flag,
age) FROM base UNION ALL
  SELECT 'default_flag', 'last_month_salary',
CORR(default_flag, last_month_salary) FROM base UNION
ALL
  SELECT 'default_flag', 'number_dependents',
CORR(default_flag, number_dependents) FROM base UNION
ALL
```

```
  SELECT 'more_90_days_overdue',
'using_lines_not_secured_personal_assets',
CORR(more_90_days_overdue,
using_lines_not_secured_personal_assets) FROM base
UNION ALL
  SELECT 'more_90_days_overdue', 'delay_30_59',
CORR(more_90_days_overdue, delay_30_59) FROM base
UNION ALL
  SELECT 'more_90_days_overdue', 'debt_ratio',
CORR(more_90_days_overdue, debt_ratio) FROM base UNION
ALL
  SELECT 'more_90_days_overdue', 'delay_60_89',
CORR(more_90_days_overdue, delay_60_89) FROM base
UNION ALL
  SELECT 'more_90_days_overdue', 'age',
CORR(more_90_days_overdue, age) FROM base UNION ALL
  SELECT 'more_90_days_overdue', 'last_month_salary',
CORR(more_90_days_overdue, last_month_salary) FROM
base UNION ALL
  SELECT 'more_90_days_overdue', 'number_dependents',
CORR(more_90_days_overdue, number_dependents) FROM
base UNION ALL

  SELECT 'using_lines_not_secured_personal_assets',
'delay_30_59',
CORR(using_lines_not_secured_personal_assets,
delay_30_59) FROM base UNION ALL
  SELECT 'using_lines_not_secured_personal_assets',
'debt_ratio',
CORR(using_lines_not_secured_personal_assets,
debt_ratio) FROM base UNION ALL
  SELECT 'using_lines_not_secured_personal_assets',
'delay_60_89',
CORR(using_lines_not_secured_personal_assets,
delay_60_89) FROM base UNION ALL
  SELECT 'using_lines_not_secured_personal_assets',
'age', CORR(using_lines_not_secured_personal_assets,
age) FROM base UNION ALL
  SELECT 'using_lines_not_secured_personal_assets',
'last_month_salary',
CORR(using_lines_not_secured_personal_assets,
last_month_salary) FROM base UNION ALL
  SELECT 'using_lines_not_secured_personal_assets',
```

```
'number_dependents',
CORR(using_lines_not_secured_personal_assets,
number_dependents) FROM base UNION ALL

  SELECT 'delay_30_59', 'debt_ratio',
CORR(delay_30_59, debt_ratio) FROM base UNION ALL
  SELECT 'delay_30_59', 'delay_60_89',
CORR(delay_30_59, delay_60_89) FROM base UNION ALL
  SELECT 'delay_30_59', 'age', CORR(delay_30_59, age)
FROM base UNION ALL
  SELECT 'delay_30_59', 'last_month_salary',
CORR(delay_30_59, last_month_salary) FROM base UNION
ALL
  SELECT 'delay_30_59', 'number_dependents',
CORR(delay_30_59, number_dependents) FROM base UNION
ALL

  SELECT 'debt_ratio', 'delay_60_89', CORR(debt_ratio,
delay_60_89) FROM base UNION ALL
  SELECT 'debt_ratio', 'age', CORR(debt_ratio, age)
FROM base UNION ALL
  SELECT 'debt_ratio', 'last_month_salary',
CORR(debt_ratio, last_month_salary) FROM base UNION
ALL
  SELECT 'debt_ratio', 'number_dependents',
CORR(debt_ratio, number_dependents) FROM base UNION
ALL

  SELECT 'delay_60_89', 'age', CORR(delay_60_89, age)
FROM base UNION ALL
  SELECT 'delay_60_89', 'last_month_salary',
CORR(delay_60_89, last_month_salary) FROM base UNION
ALL
  SELECT 'delay_60_89', 'number_dependents',
CORR(delay_60_89, number_dependents) FROM base UNION
ALL

  SELECT 'age', 'last_month_salary', CORR(age,
last_month_salary) FROM base UNION ALL
  SELECT 'age', 'number_dependents', CORR(age,
number_dependents) FROM base UNION ALL

  SELECT 'last_month_salary', 'number_dependents',
```

```
CORR(last_month_salary, number_dependents) FROM base
),

classified AS (
  SELECT *,
    CASE
      WHEN ABS(corr) = 1 THEN 'Correlação perfeita'
      WHEN ABS(corr) >= 0.9 THEN 'Muito forte'
      WHEN ABS(corr) >= 0.7 THEN 'Forte'
      WHEN ABS(corr) >= 0.4 THEN 'Moderada'
      WHEN ABS(corr) >= 0.2 THEN 'Fraca'
      ELSE 'Muito fraca ou nula'
    END AS classificacao
  FROM pairs
)

SELECT * FROM classified
ORDER BY var1, var2;
```

2. Ao final, tenho o seguinte resumo:

a.

var1	var2	corr	classificacao	Tabela
delay_30_59	delay_60_89	0,9865536455	Muito forte	loans_detail
more_90_days_overdue	delay_30_59	0,9829168066	Muito forte	loans_detail
more_90_days_overdue	delay_60_89	0,9921755263	Muito forte	loans_detail

3. Cálculo do **Desvio Padrão** para escolha da variável que irá prevalecer (feito no **Python**):

a.

Variáveis	Desvio Padrão	
delay_30_59	4.144020438226020	1ª
more_90_days_overdue	4.12136466.428520	2ª
delay_60_89	4.1055147551019315	3ª

b.

var1	var2	corr	Variável escolhida
delay_30_59	delay_60_89	0,9865536455	delay_30_59
more_90_days_overdue	delay_30_59	0,9829168066	delay_30_59
more_90_days_overdue	delay_60_89	0,9921755263	more_90_days_overdue

e. 2.1.5 🔵 Identificar e tratar dados inconsistentes em variáveis categóricas✅

i. Primeiro, apenas as tabelas **“loans_outstanding”** e **“user_info”** possuem variável caracterizada como STRING(Texto);

ii. Porém, a tabela “user_info” possui a variável **“sex”** que deve ser **desconsiderada** por se referir a características pessoais do cliente;

iii. Então, primeiramente foi aplicada a query para saber exatamente qual as variações de preenchimento dos campos:

1. LINK - Bigquery

2.

```
# TABELA: loans_outstanding - (possui STRING)

# 1º para saber quais os dados que temos e suas variações:
SELECT DISTINCT loan_type
FROM
`projeto3-supercaja.projeto3_dataset.loans_outstanding`
ORDER BY loan_type;
```

3. Retornou:

a.

```
OTHER
Other
REAL ESTATE
Real Estate
other
others
real estate
```

4. Então, o passo seguinte foi criar uma **NOVA TABELA** com os **campos padronizados (**loans_outstanding**_padronizado)**:

a.

```
# 2º agora sim criando uma nova tabela padronizada:Se tivesse repetido o caminho da tabela, teria substituído,# como coloquei "loans_outstanding_padronizado", foi criado uma
```

```
nova)
CREATE OR REPLACE TABLE
`projeto3-supercaja.projeto3_dataset.loans_outst
anding_padronizado` AS
SELECT
  *,
  CASE
    WHEN LOWER(TRIM(loan_type)) IN ('other',
'others') THEN 'other'
    WHEN LOWER(TRIM(loan_type)) = 'real estate'
THEN 'real estate'
    ELSE loan_type
  END AS loan_type_padronizado
FROM
`projeto3-supercaja.projeto3_dataset.loans_outst
anding`;
```

5. A query a seguir gera **apenas visualização**:

   a.

```
# 3º - query só para VISUALIZAÇÃO:
SELECT
  loan_id,
  user_id,
  loan_type,
  CASE
    WHEN LOWER(TRIM(loan_type)) IN ('other',
'others') THEN 'other'
    WHEN LOWER(TRIM(loan_type)) = 'real estate'
THEN 'real estate'
    ELSE loan_type
  END AS loan_type_padronizado
FROM
projeto3-supercaja.projeto3_dataset.loans_outsta
nding
```

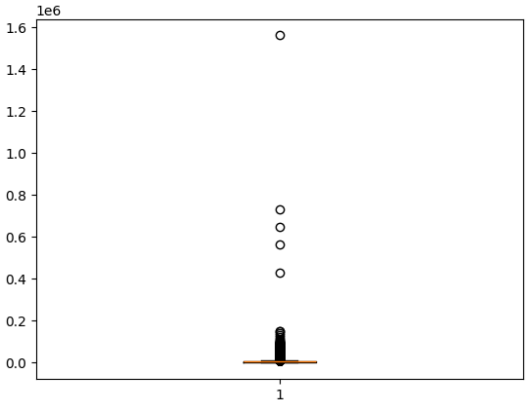
f. 2.1.6 🔵 Identificar e tratar dados discrepantes em variáveis numéricas ✅

   i. Primeiro, deve-se decidir quais variáveis vão ser verificadas quanto a **presença de Outliers**;

ii. ⚠️ - **Uma dúvida**: a correlação calculada no item anterior, é para decidir qual variável será priorizada, inclusive no tratamento de Outiliers? Sim.

iii. Variáveis que devem ser analisadas quanto a outliers:.

iv. **PRIMEIRO,** analisando **"last_month_salary"** da tabela **user_info;**

1. Para tanto, para **1ª CONFIRMAÇÃO** utilizei a **Média de Desvio Padrão**. Foi realizada no **Colab, com Python. (**⚠️Não vou aplicar nas próximas variáveis):
    a. `Media artmetica: 6.420.07`
    b. `Desvio padrão: 11.604.57`
    c. `Limite SUPERIOR:41.233.81`
    d. `Limite INFERIOR: -28.393.66`
2. Já para uma **2ª CONFIRMAÇÃO**, temos primeiro a visualização através do **Boxplot:**
3. 

4. Já para uma **3ª CONFIRMAÇÃO**, temos a visualização através da **Mediana + QUARTIS,** feita no **Python:**
    a. Q1: 3.900.0
    b. Q3: 7.416.0
    c. IQR (Interquartil): 3.516.0
    d. Limite SUPERIOR: 12.690.0
    e. Limite INFERIOR: -1.374.0
5. No Colab, identifiquei o outlier **mais discrepante**:
    a.

```
# Para visualizar os MAIORES VALORES:

dados['last_month_salary'].sort_values(ascendin
```

```
g=False).head(20)
```

b.

	last_month_salary
21095	**1560100.0**
6542	730483.0
23383	649587.0
22075	562466.0
24042	428250.0
2291	150000.0
15931	150000.0
34712	143000.0
34557	133000.0
2854	121000.0

dtype: float64

c. Esse número **(1560100.0)** está **de acordo** com o Boxplot;

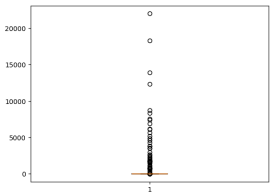
d. São **2186** registros de **outliers**, segundo Python;

e. A única modificação foi relacionado aos valores nulos que foram preenchidos com a mediana, enfim, uma ação já realizada anteriormente;

v. **SEGUNDO**, analisando a variável "**number times delayed payment loan 30 59 days"** na tabela **loans_detail:**

1. **A**s variações relacionadas ao **número de vezes que o cliente atrasou o pagamento de um empréstimo**, é importante para a caracterização e posterior segmentação do cliente, até mesmo os valores nulos;

vi. **TERCEIRO**, analisando a variável **"using_ lines_ not_ secured_personal_assets"** na tabela **loans_detail:**

1. Essa variável trata do quanto o cliente está **utilizando em relação ao seu limite de crédito**.
2. 

3. São valores de fato discrepantes: números inteiros se misturam a decimais, como por exemplo:
4.

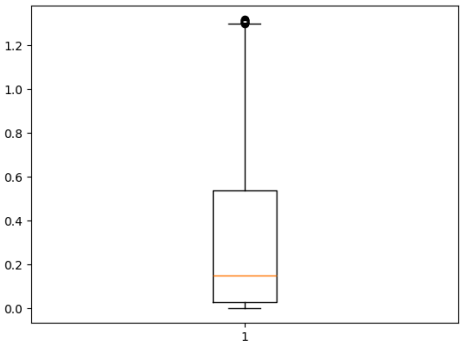
1,511145273
2.242
1.761
1,694922031
1,680709534
1,591362126
1,891169351
18.300
22.000

5. ⚠️ - **Provável decisão**: **Imputação** - Substitui os outliers por outros mais representativos;
6. Foram identificados 177 registros como outliers (no Bigquery, no Python trouxe 95 registros)
7. Concluí a **substituição** esses valores discrepantes pela **média**, assim, como fiz ao substituir os outliers na tabela user_info pela média dos não-outliers para seguir a mesma regra de imputação aplicada anteriormente;
8. Assim, criada uma **NOVA TABELA** (loans_detail_**tratado**), ainda conservando a original
9.

```
CREATE OR REPLACE TABLE
`projeto3-supercaja.projeto3_dataset.loans_detail_trat
ado` AS
WITH quartis AS (
```

```
  SELECT

PERCENTILE_CONT(using_lines_not_secured_personal_asset
s, 0.25) OVER() AS Q1,

PERCENTILE_CONT(using_lines_not_secured_personal_asset
s, 0.75) OVER() AS Q3
  FROM
`projeto3-supercaja.projeto3_dataset.loans_detail`
  LIMIT 1
),
limites AS (
  SELECT
    Q1,
    Q3,
    (Q3 - Q1) AS IQR,
    Q1 - 1.5 * (Q3 - Q1) AS limite_inferior,
    Q3 + 1.5 * (Q3 - Q1) AS limite_superior
  FROM quartis
),
media_nao_outlier AS (
  SELECT
    AVG(using_lines_not_secured_personal_assets) AS
media_tratada
  FROM
`projeto3-supercaja.projeto3_dataset.loans_detail`,
limites
  WHERE using_lines_not_secured_personal_assets
BETWEEN limite_inferior AND limite_superior
),
tabela_tratada AS (
  SELECT
    ld.*,
    CASE
      WHEN ld.using_lines_not_secured_personal_assets
< l.limite_inferior
        OR ld.using_lines_not_secured_personal_assets
> l.limite_superior
      THEN m.media_tratada
      ELSE ld.using_lines_not_secured_personal_assets
    END AS
using_lines_not_secured_personal_assets_tratada,
    CASE
```

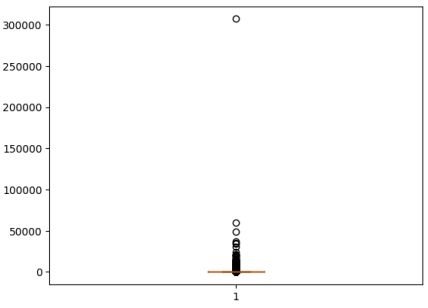
```
      WHEN ld.using_lines_not_secured_personal_assets
< l.limite_inferior
        OR ld.using_lines_not_secured_personal_assets
> l.limite_superior
      THEN 'Outlier'
      ELSE 'Normal'
    END AS outlier_status
  FROM
`projeto3-supercaja.projeto3_dataset.loans_detail` ld
  CROSS JOIN limites l
  CROSS JOIN media_nao_outlier m
)
SELECT *
FROM tabela_tratada;
```

10. 

11. Boxplot da tabela tratada

vii. **QUARTO,** analisando a variável **"debt ratio"** na tabela **loans_detail:**

1. Essa variável trata da relação entre dívidas e ativos do cliente (**Taxa de endividamento** = Dívidas / Patrimônio);

2. 

3. Foram identificados **7579 registros** de Outliers, tanto no Bigquery como no Python;

4. Concluí a **substituição** esses valores discrepantes pela **média**, assim, como fiz ao substituir os outliers na tabela user_info pela média dos não-outliers para seguir a mesma regra de imputação aplicada anteriormente;

g. 2.1.8 🔵Criar novas variáveis ✅

i. **Primeira variável** criada: apliquei a Query abaixo para saber quais as categorias de empréstimos existentes na tabela já tratada (padronizada) “loans_outstanding_padronizado”;

1.

```
SELECT DISTINCT loan_type_padronizado
FROM
`projeto3-supercaja.projeto3_dataset.loans_outstanding_padron
izado`;
```

2. Retornou:

3.

	loan_type_padronizado
1	other
2	real estate

4. Em seguida, gerei query em ChatGPT para Contagem de "other" e "real estate" por "user_id":

5.

```
# CONTAGEM de "other e real estate" por "user_id"

SELECT
  user_id,
  COUNT(CASE WHEN LOWER(loan_type) = 'other' THEN 1
END) AS count_other_loans,
  COUNT(CASE WHEN LOWER(loan_type) = 'real estate'
THEN 1 END) AS count_real_estate_loans,
  COUNT(*) AS total_loans
FROM

`projeto3-supercaja.projeto3_dataset.loans_outstanding
```

```
_padronizado`
WHERE
  LOWER(loan_type) IN ('other', 'real estate')
GROUP BY
  user_id
ORDER BY
  user_id;
```

6. Agora criar uma nova tabela com o resultado da contagem acima:

7.

```
# CRIANDO TABELA COM A QUERY DE CONTAGEM

CREATE OR REPLACE TABLE
`projeto3-supercaja.projeto3_dataset.loans_outst
anding_emprestimo` AS
WITH loan_counts AS (
  SELECT
    user_id,
    COUNT(CASE WHEN LOWER(loan_type) = 'other'
THEN 1 END) AS count_other_loans,
    COUNT(CASE WHEN LOWER(loan_type) = 'real
estate' THEN 1 END) AS count_real_estate_loans,
    COUNT(*) AS total_filtered_loans
  FROM

`projeto3-supercaja.projeto3_dataset.loans_outst
anding_padronizado`
  WHERE
    LOWER(loan_type) IN ('other', 'real estate')
  GROUP BY
    user_id
),
distinct_users AS (
  SELECT
    user_id,
    -- Use QUALIFY ROW_NUMBER() para pegar
apenas uma linha por user_id
    -- Você pode adicionar outros campos
originais que precisar aqui
    ROW_NUMBER() OVER(PARTITION BY user_id) as
```

```
rn
  FROM

`projeto3-supercaja.projeto3_dataset.loans_outst
anding_padronizado`
)

SELECT
  du.user_id,
  -- outros campos originais que você selecionou
no CTE distinct_users
  IFNULL(lc.count_other_loans, 0) AS
count_other_loans,
  IFNULL(lc.count_real_estate_loans, 0) AS
count_real_estate_loans,
  IFNULL(lc.total_filtered_loans, 0) AS
total_filtered_loans
FROM
  distinct_users du
LEFT JOIN
  loan_counts lc
ON
  du.user_id = lc.user_id
WHERE
  du.rn = 1;  -- Apenas uma linha por user_id
```

8. Assim, foi criada a tabela “loans_outstanding_**emprestimo**” com as seguintes variáveis:
    a. user_id (original);
    b. count_other_loans **(nova)**;
    c. count_real_estate_loans **(nova)**;
    d. total_filtered_loans **(nova)**;

ii. **Segunda variável** criada: foi a partir da tabela Default. Foi aplicado a seguinte query:

1.

```
# NOVA VARIÁVEL NA TABELA DEFAULT

CREATE TABLE
```

```
`projeto3-supercaja.projeto3_dataset.default_classificado` AS
SELECT
  *,
  CASE
    WHEN default_flag = 0 THEN 'Adimplente'
    WHEN default_flag = 1 THEN 'Inadimplente'
    ELSE 'Desconhecido'
  END AS status_pagamento
FROM `projeto3-supercaja.projeto3_dataset.default`;
```

2. Assim, foi criada a tabela “default_**classificado**” com as seguintes variáveis:
    a. user_id (original);
    b. default_flag (original);
    c. status_pagamento **(nova)**;

iii. **Terceira variável** criada: user_info3

1.

```
CREATE OR REPLACE TABLE
`projeto3-supercaja.projeto3_dataset.user_info3` AS
WITH dados_tratados AS (
  SELECT
    user_id,
    -- Substituição de NULL por 0
    IFNULL(number_dependents, 0) AS number_dependents,
    age,
    -- Substituição de NULL pela mediana
    IFNULL(last_month_salary, 5400.0) AS
last_month_salary
  FROM
    `projeto3-supercaja.projeto3_dataset.user_info2`
)

SELECT
  user_id,
  number_dependents,

  -- Classificação por dependentes
  CASE
```

```
    WHEN number_dependents > 0 THEN 'Com filhos'
    ELSE 'Sem filhos'
  END AS tem_filhos,

  age,

  -- Nova classificação etária ajustada
  CASE
    WHEN age BETWEEN 21 AND 34 THEN 'Jovens adultos'
    WHEN age BETWEEN 35 AND 54 THEN 'Adultos maduros'
    WHEN age BETWEEN 55 AND 74 THEN 'Pré-idosos'
    WHEN age >= 75 THEN 'Idosos'
    ELSE 'Fora da faixa (menor de 21 ou idade
inválida)'
  END AS faixa_etaria,

  last_month_salary

FROM
  dados_tratados;
```

2.

h. 2.1.9 🔵 Unir tabelas: ✅

i. Foi usado LEFT JOIN para que **todos os registros** fossem considerados. (Total: 36.000)

ii. Para constar:

1. 21-34 anos: Millennials / Jovens adultos
2. 35-54 anos: Geração X / Adultos maduros
3. 55-74 anos: Baby Boomers / Pré-idosos
4. 75+ anos: Idosos

iii. Query para UNIR as tabelas que tratei:

1.

```
# TABELA UNIFICADA 3

CREATE OR REPLACE TABLE
`projeto3-supercaja.projeto3_dataset.tabela_unificada_3`
AS
```

```sql
SELECT
  -- From user_info3
  ui.user_id,
  ui.number_dependents,
  ui.tem_filhos,
  ui.age,
  ui.faixa_etaria,
  ui.last_month_salary,

  -- From default_classificado
  df.default_flag,
  df.status_pagamento,

  -- From loans_detail_tratado2_debt_ratio
  ld.number_times_delayed_payment_loan_30_59_days,
  ld.using_lines_not_secured_personal_assets_tratada,
  ld.debt_ratio_corrigido,

  -- From loans_outstanding_emprestimo
  lo.count_other_loans,
  lo.count_real_estate_loans,
  lo.total_filtered_loans

FROM
  `projeto3-supercaja.projeto3_dataset.user_info3` ui

LEFT JOIN

`projeto3-supercaja.projeto3_dataset.default_classificad
o` df
ON
  ui.user_id = df.user_id

LEFT JOIN

`projeto3-supercaja.projeto3_dataset.loans_detail_tratad
o2_debt_ratio` ld
ON
  ui.user_id = ld.user_id

LEFT JOIN

`projeto3-supercaja.projeto3_dataset.loans_outstanding_e
```

```
mprestimo` lo
ON
  ui.user_id = lo.user_id;
```

i. 2.1.10🔵 Construir **tabelas auxiliares**:✅

i. Depois, para **atualização** de user_info2:

1.

```
WITH dados_tratados AS (
  SELECT
    -- Substitui NULL por 0
    IFNULL(number_dependents, 0) AS number_dependents,
    age
  FROM
    `projeto3-supercaja.projeto3_dataset.user_info2`
)

SELECT
  number_dependents,

  -- Classifica como 'Com filhos' ou 'Sem filhos'
  CASE
    WHEN number_dependents > 0 THEN 'Com filhos'
    ELSE 'Sem filhos'
  END AS tem_filhos,

  age,

  -- Classifica como Adulto ou Idoso
  CASE
    WHEN age >= 65 THEN 'Idoso'
    WHEN age >= 18 THEN 'Adulto'
    ELSE 'Menor de idade'
  END AS faixa_etaria

FROM
  dados_tratados;
```

4. 🟪 2.2 Fazer uma análise exploratória

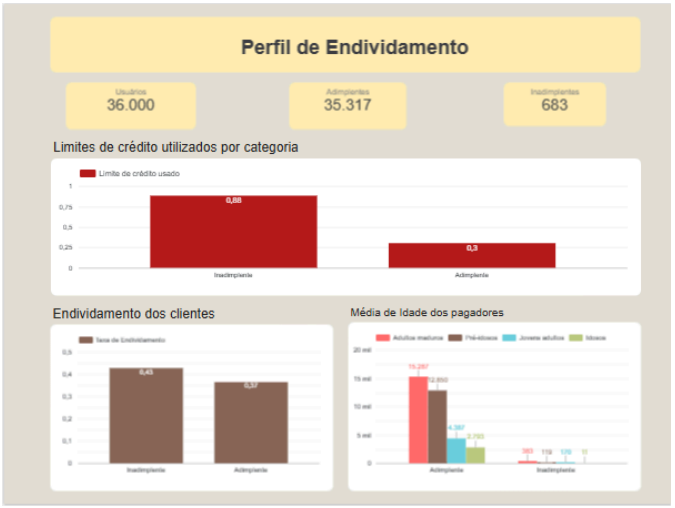
a. 2.2.1 🟣 Agrupar dados de acordo com variáveis categóricas:✅

   i. Exportei a tabela do Bigquery para Google Sheets;

   ii. Depois, importei essa planilha para o Looker Studio através da URL;

b. 2.2.2 🟣 Ver variáveis categóricas:✅

   i. Print e LINK (Dashboar Looker) :

1. 

2. **Observação:** tive que importar novamente uma nova tabela com a classificação “Adimplente e Inadimplente”, pois não me atentei para fazer isso na tabela anterior. Mas não causou problemas, pois o Looker permite o manuseio de 2 tabelas na mesma página;

c. 2.2.3 🟣 Aplicar medidas de tendência central (moda, média, mediana):✅

   i. Tabelas feitas no Looker - LINK:

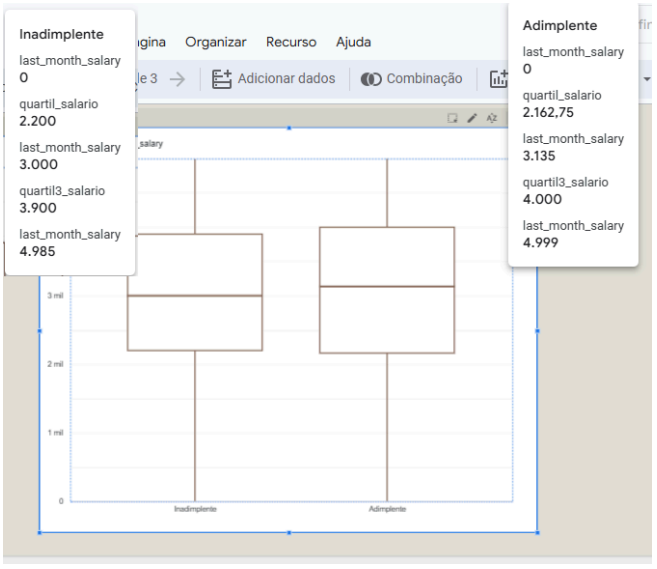
1. 

2. Looker **não** calcula **moda**;

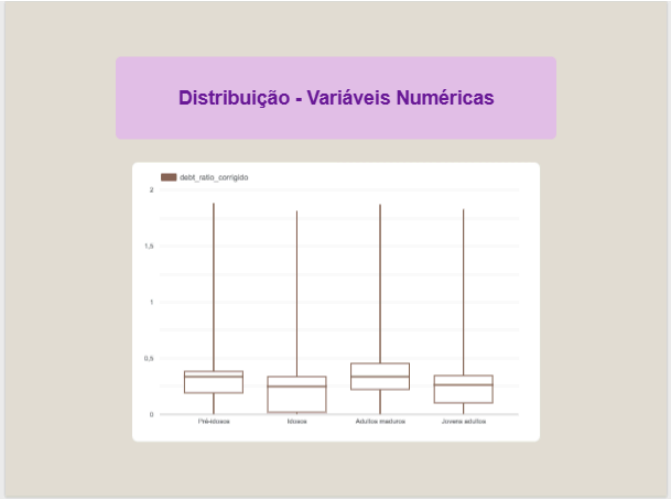
3. Vídeo usado como referência - LINK

d. 2.2.4 🟣 Ver distribuição:✅

i. Bloxpot / Salário - LINK - Looker Studio:

1. 

ii. Bloxpot /Endividamento - LINK - Looker Studio:

1. 

iii. Conclusões:

e. 2.2.5 🟣 Aplicar medidas de dispersão (desvio padrão): ✅

i. LINK - Looker Studio:

1. 

f. 2.2.6 🟣 Calcular quartis, decis ou percentis ✅

i. Variáveis usadas:

1. age;
2. last_month_salary ;
3. default_flag;
4. `debt_ratio_corrigido`

ii. Query:

1.

```
```

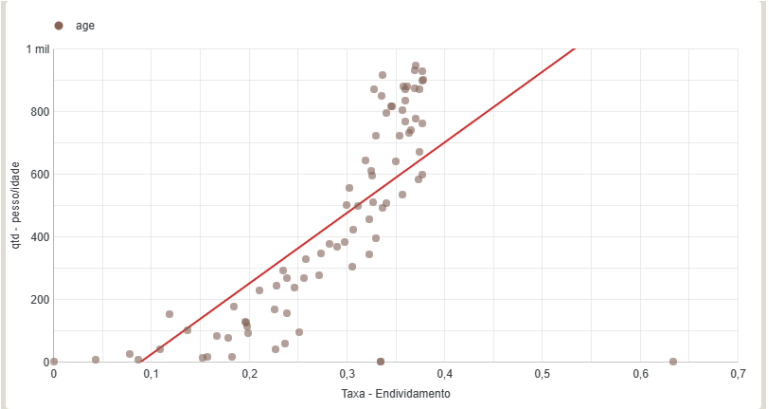
```sql
WITH ClientesComQuartis AS (
    SELECT
        age,
        last_month_salary,
        default_flag,
        debt_ratio_corrigido,
        -- Quartis baseados na idade
        NTILE(4) OVER (ORDER BY age) AS quartil_idade,
        -- Quartis baseados no salário do mês anterior
        NTILE(4) OVER (ORDER BY last_month_salary) AS
quartil_salario_mes_anterior,
        -- Quartis baseados no debt_ratio_corrigido
(NOVO)
        NTILE(4) OVER (ORDER BY debt_ratio_corrigido)
AS quartil_debt_ratio
    FROM

`projeto3-supercaja.projeto3_dataset.tabela_unificada_
3`
)
-- Selecionando os dados finais
SELECT
    age,
    last_month_salary,
    default_flag,
    debt_ratio_corrigido,
    quartil_idade,
    quartil_salario_mes_anterior,
    quartil_debt_ratio  -- Nova coluna de quartil para
debt_ratio
FROM
    ClientesComQuartis
```

g. 2.2.7🟣 Calcular correlação entre variáveis numéricas ✅

   i. Vídeo referência interessante explicando como analisar o gráfico de dispersão: ▶ Falando sobre Gráfico de Dispersão

   ii. Correlação:

      1. Idade x Taxa de Endividamento:

a. 

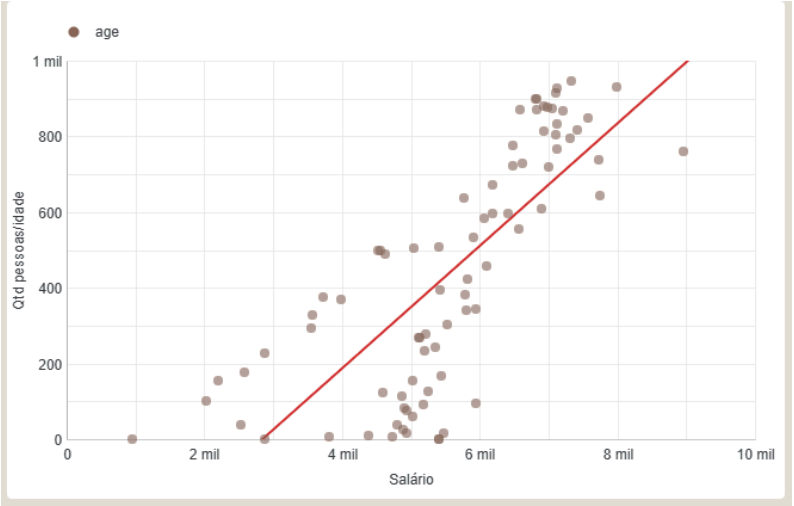
b. ***Interpretação/ Direcionamento:*** Dispersão dos Dados: Os pontos estão distribuídos de forma relativamente dispersa, sem um padrão linear claro. Isso sugere que a correlação entre idade e taxa de endividamento pode ser fraca ou moderada.

c. ***Possível Tendência:*** Apesar da dispersão, há uma leve inclinação ascendente nos pontos à medida que a idade aumenta, indicando que pessoas mais velhas podem ter taxas de endividamento ligeiramente maiores. No entanto, essa tendência não é forte.

d. ***Outliers*****:** Alguns pontos estão distantes da maioria, o que pode indicar casos atípicos (ex.: pessoas jovens com alto endividamento ou idosos com baixo endividamento).

e. ***Conclusão:*** A correlação entre idade e endividamento parece positiva, mas fraca. Fatores como renda, perfil financeiro ou contexto econômico podem influenciar mais fortemente o endividamento do que a idade isoladamente;

2. Idade x Salário:

a. 

b. ***<u>Interpretação/Direcionamento</u>***: Padrão Geral - Há uma tendência positiva entre idade e salário, ou seja, à medida que a idade aumenta, o salário também tende a crescer. Isso é comum em muitas carreiras, onde experiência e tempo no mercado levam a melhores remunerações.

c. ***Dispersão dos Dados:*** Os pontos estão relativamente alinhados, mas com alguma dispersão, especialmente em idades mais avançadas. Isso indica que, embora a correlação seja positiva, não é perfeita (ex.: algumas pessoas mais velhas têm salários menores, enquanto outras ganham significativamente mais).

d. ***Picos e Variações:***

   i. Em idades intermediárias (por volta dos 40-50 anos), há uma concentração de salários mais altos, sugerindo que esse pode ser o ápice da carreira para muitos.

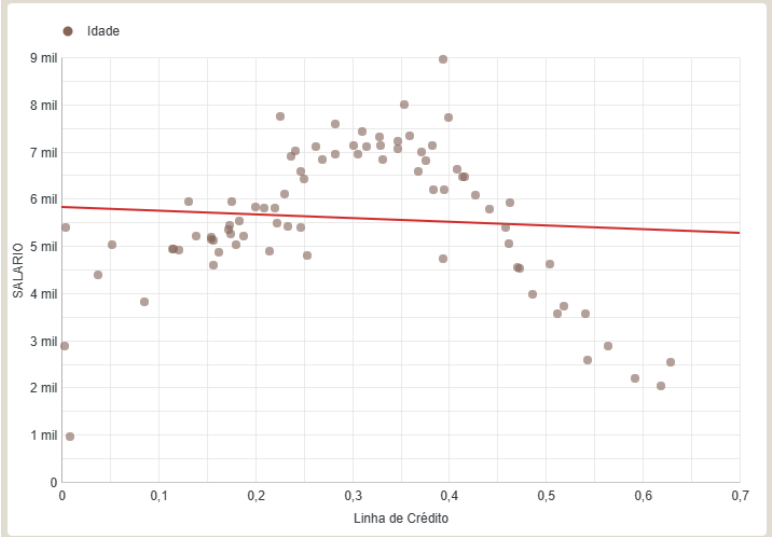
   ii. Jovens (lado esquerdo do gráfico) têm salários mais baixos, o que condiz com a entrada no mercado de trabalho.

e. ***Possíveis Outliers***: Alguns pontos isolados (ex.: pessoas jovens com salários altos ou idosos com salários baixos) podem representar casos atípicos,

como profissionais precoces ou carreiras com estagnação financeira.

f. ***Conclusão***: A correlação entre idade e salário é positiva e moderadamente forte, refletindo que a experiência geralmente se traduz em maior remuneração. No entanto, fatores como área de atuação, qualificação e contexto econômico também influenciam significativamente;

3. Salário x Linha de Crédito:

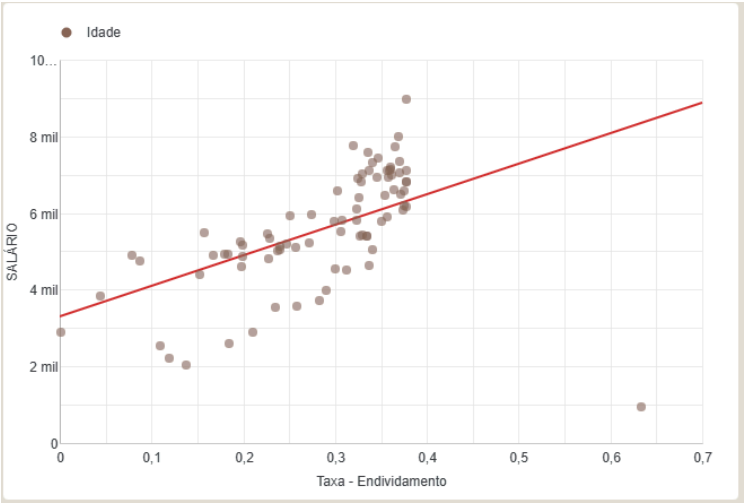
a. 

b. ***Interpretação/Direcionamento:*** Tendência Geral: Os dados sugerem uma correlação positiva entre salário e linha de crédito, ou seja, quanto maior o salário, maior tende a ser o limite de crédito oferecido. Isso faz sentido, pois instituições financeiras geralmente vinculam o crédito à capacidade de pagamento (renda).

c. **Dispersão dos Dados**:Apesar da tendência positiva, há uma dispersão significativa em alguns intervalos (ex.: pessoas com salários médios podem ter linhas de crédito variáveis). Isso indica que outros fatores (como score de crédito, histórico bancário ou dívidas) também influenciam o resultado.

d. **Padrão Não Linear:** Em salários mais altos (acima de 6 mil), o crescimento da linha de crédito parece se acelerar, sugerindo que instituições podem ser mais generosas com quem tem renda elevada.Já para salários baixos (abaixo de 2 mil), a linha de crédito é limitada, mesmo com variações.
e. **Possíveis Outliers:** Alguns pontos fora do padrão (ex.: salário alto com crédito baixo ou vice-versa) podem refletir situações específicas, como restrições no CPF ou clientes com perfil de risco diferenciado.

4. Salário x taxa de Endividamento:

a. 

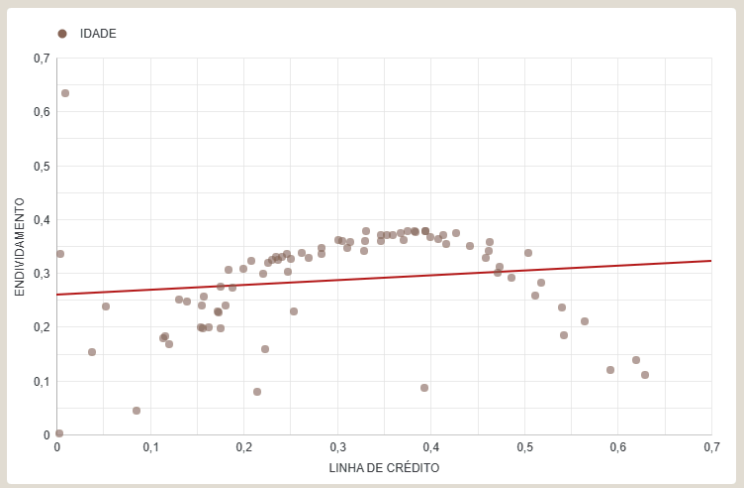
b. ***Interpretação/Direcionamento:*** Tendência Geral: Os dados indicam uma correlação negativa entre salário e taxa de endividamento. Ou seja, quanto maior o salário, menor a taxa de endividamento, e vice-versa. Isso é consistente com a expectativa de que pessoas com maior renda têm mais capacidade de pagamento e menor necessidade de contrair dívidas.

c. **Dispersão dos Dados:** Há uma dispersão moderada, especialmente em faixas salariais intermediárias (4 mil a 6 mil), onde algumas pessoas apresentam taxas de endividamento mais altas ou mais baixas. Isso sugere que outros fatores (como estilo de vida, gastos fixos ou

emergências) também influenciam o endividamento, além da renda.

d. **Padrão Não Linear**: Para salários muito baixos (abaixo de 2 mil), a taxa de endividamento é consistentemente alta, refletindo a dificuldade de equilibrar orçamento com renda limitada. Em salários altos (acima de 8 mil), a taxa de endividamento é baixa, mas não necessariamente zero, indicando que mesmo pessoas com boa renda podem ter dívidas (por exemplo, empréstimos para investimentos ou imóveis).
e. **Possíveis Outliers**:Alguns pontos fora da tendência principal (ex.: salário alto com endividamento elevado ou salário baixo com endividamento reduzido) podem representar casos específicos, como:
    1. Pessoas com alta renda mas alto nível de gastos.
    2. Indivíduos com baixa renda mas que priorizam o controle financeiro.

5. Endividamento x Linha de Crédito:

a. 

b. ***Interpretação/Direcionamento***:Padrão Geral por Faixa Etária

    i. **Jovens (20-35 anos**): **Endividamento**: Tendem a ter níveis moderados a altos (0,4-0,6), principalmente por dívidas de consumo (cartão

de crédito, financiamento estudantil). **Linha de Crédito:** Geralmente limitada (0,2-0,5), pois instituições veem risco maior devido à renda instável e histórico curto. ***Correlação***: Positiva, mas fraca. Crédito cresce com dívida, mas é contido pela idade/risco.

ii. **Adultos (36-55 anos):Endividamento**: Pico de 0,5-0,7 (hipotecas, empréstimos pessoais, despesas familiares).**Linha de Crédito**: Mais ampla (0,5-0,8), pois renda estável e histórico de crédito consolidado aumentam a confiança.***Correlação***: Forte e positiva. Dívida ativa "alimenta" acesso a mais crédito.

iii. **Idosos (56+ anos):Endividamento**: Reduzido (0,2-0,4), mas com outliers (ex.: dívidas médicas ou empréstimos consignados). **Linha de Crédito**: Variável (0,3-0,6). Bancos podem restringir crédito devido à aposentadoria (renda fixa), mas oferecem consignado. ***Correlação*:** Fraca ou negativa. Menos dívidas não necessariamente aumentam crédito.

iv. **Insights Chave - Efeito Ciclo de Vida**:Jovens usam crédito para consumo → Adultos para investimentos (casa, negócios) → Idosos reduzem dívidas, mas têm crédito restrito.
Risco x Oportunidade: Adultos são o grupo com maior sinergia entre endividamento e linha de crédito, pois equilibram capacidade de pagamento e necessidades financeiras.
Armadilhas Potenciais: Jovens com alta linha de crédito e endividamento podem entrar em espiral de dívidas (juros compostos). Idosos com crédito fácil (ex.: consignado) podem comprometer renda fixa;

v.

5. 🟥 2.3 Aplicar técnica de análise

a. 2.3.1 🔴 Calcular risco relativo ✅

i. Vídeo - Referência - Conceito;

ii. Query - calculo de risco: **age**:

1.

```
WITH ClientesComQuartis AS (
    SELECT
        age,
        default_flag,
        NTILE(4) OVER (ORDER BY age) AS quartil_idade
    FROM

projeto3-supercaja.projeto3_dataset.tabela_unificada_3
),

TaxasDefault AS (
    SELECT
        quartil_idade,
        COUNT(*) AS total,
        SUM(CASE WHEN default_flag = 1 THEN 1 ELSE 0
END) AS inadimplentes,
        SAFE_DIVIDE(SUM(CASE WHEN default_flag = 1
THEN 1 ELSE 0 END), COUNT(*)) AS taxa_inadimplencia
    FROM ClientesComQuartis
    GROUP BY quartil_idade
),

RiscoRelativo AS (
    SELECT
        t1.*,
        FIRST_VALUE(taxa_inadimplencia) OVER (ORDER BY
quartil_idade) AS taxa_ref,
        SAFE_DIVIDE(taxa_inadimplencia,
FIRST_VALUE(taxa_inadimplencia) OVER (ORDER BY
quartil_idade)) AS risco_relativo
    FROM TaxasDefault t1
),
```

```
QuartilComCategoria AS (
    SELECT
        *,
        CASE
            WHEN risco_relativo < 1 THEN 'Menor risco'
            WHEN risco_relativo = 1 THEN 'Risco igual
ao ref'
            WHEN risco_relativo > 1 AND risco_relativo
<= 1.5 THEN 'Risco moderadamente maior'
            WHEN risco_relativo > 1.5 THEN 'Alto
risco'
            ELSE 'Desconhecido'
        END AS categoria_risco_idade
    FROM RiscoRelativo
)

-- Resultado final com quartis, taxa, risco relativo e
categoria
SELECT
    quartil_idade,
    total,
    inadimplentes,
    ROUND(taxa_inadimplencia, 4) AS
taxa_inadimplencia,
    ROUND(risco_relativo, 2) AS risco_relativo,
    categoria_risco_idade
FROM QuartilComCategoria
ORDER BY quartil_idade;
```

iii. Query - calculo de risco: **`last_month_salary`**

1.

```
  WITH ClientesComQuartis AS (
    SELECT
        last_month_salary,
        default_flag,
        NTILE(4) OVER (ORDER BY last_month_salary) AS
quartil_salario
    FROM
projeto3-supercaja.projeto3_dataset.tabela_unificada_3
),
```

```
TaxasSalario AS (
    SELECT
        quartil_salario,
        COUNT(*) AS total,
        SUM(CASE WHEN default_flag = 1 THEN 1 ELSE 0
END) AS inadimplentes,
        SAFE_DIVIDE(SUM(CASE WHEN default_flag = 1
THEN 1 ELSE 0 END), COUNT(*)) AS taxa
    FROM ClientesComQuartis
    GROUP BY quartil_salario
),

RiscoRelativo AS (
    SELECT
        *,
        FIRST_VALUE(taxa) OVER (ORDER BY
quartil_salario) AS taxa_referencia,
        SAFE_DIVIDE(taxa, FIRST_VALUE(taxa) OVER
(ORDER BY quartil_salario)) AS risco_relativo
    FROM TaxasSalario
)

SELECT
    quartil_salario,
    total,
    inadimplentes,
    ROUND(taxa, 4) AS taxa_inadimplencia,
    ROUND(risco_relativo, 2) AS risco_relativo,
    CASE
        WHEN risco_relativo < 1 THEN 'Menor risco'
        WHEN risco_relativo = 1 THEN 'Risco igual ao
ref'
        WHEN risco_relativo > 1 AND risco_relativo <=
1.5 THEN 'Risco moderadamente maior'
        WHEN risco_relativo > 1.5 THEN 'Alto risco'
        ELSE 'Desconhecido'
    END AS categoria_risco_salario
FROM RiscoRelativo
ORDER BY quartil_salario;
```

iv. Query - calculo de risco: **`debt_ratio_corrigido`**

1.

```sql
WITH ClientesComQuartis AS (
    SELECT
        debt_ratio_corrigido,
        default_flag,
        NTILE(4) OVER (ORDER BY debt_ratio_corrigido)
AS quartil_debt
    FROM
projeto3-supercaja.projeto3_dataset.tabela_unificada_3
),

TaxasDebt AS (
    SELECT
        quartil_debt,
        COUNT(*) AS total,
        SUM(CASE WHEN default_flag = 1 THEN 1 ELSE 0
END) AS inadimplentes,
        SAFE_DIVIDE(SUM(CASE WHEN default_flag = 1
THEN 1 ELSE 0 END), COUNT(*)) AS taxa
    FROM ClientesComQuartis
    GROUP BY quartil_debt
),

RiscoRelativo AS (
    SELECT
        *,
        FIRST_VALUE(taxa) OVER (ORDER BY quartil_debt)
AS taxa_referencia,
        SAFE_DIVIDE(taxa, FIRST_VALUE(taxa) OVER
(ORDER BY quartil_debt)) AS risco_relativo
    FROM TaxasDebt
)

SELECT
    quartil_debt,
    total,
    inadimplentes,
    ROUND(taxa, 4) AS taxa_inadimplencia,
    ROUND(risco_relativo, 2) AS risco_relativo,
    CASE
        WHEN risco_relativo < 1 THEN 'Menor risco'
        WHEN risco_relativo = 1 THEN 'Risco igual ao
ref'
        WHEN risco_relativo > 1 AND risco_relativo <=
```

```
1.5 THEN 'Risco moderadamente maior'
        WHEN risco_relativo > 1.5 THEN 'Alto risco'
        ELSE 'Desconhecido'
    END AS categoria_risco_debt
FROM RiscoRelativo
ORDER BY quartil_debt;
```

v. Query - calculo de risco: **using_lines_tratada**

1.

```
WITH linhas_quartis AS (
  SELECT *,
         NTILE(4) OVER (ORDER BY
using_lines_not_secured_personal_assets_tratada) AS
quartil_linhas
  FROM
`projeto3-supercaja.projeto3_dataset.tabela_unificada_
3`
),

agrupado AS (
  SELECT
    quartil_linhas AS grupo,
    SUM(CASE WHEN default_flag = 1 THEN 1 ELSE 0 END)
AS inadimplentes,
    SUM(CASE WHEN default_flag = 0 THEN 1 ELSE 0 END)
AS adimplentes,
    COUNT(*) AS total
  FROM linhas_quartis
  GROUP BY grupo
),

riscos AS (
  SELECT
    grupo,
    inadimplentes,
    adimplentes,
    total,
    SAFE_DIVIDE(inadimplentes, total) AS risco
  FROM agrupado
),
```

```sql
referencia AS (
  SELECT
    risco AS risco_ref
  FROM riscos
  WHERE grupo = 4
)

SELECT
  r.grupo,
  r.inadimplentes,
  r.adimplentes,
  r.total,
  r.risco,
  ref.risco_ref,
  SAFE_DIVIDE(r.risco, ref.risco_ref) AS
risco_relativo
FROM riscos r
CROSS JOIN referencia ref
ORDER BY r.grupo;
```

b. 2.3.2 🔴 Aplicar segmentação por Score: ✅

i. Artigo referência - LINK

1. Pontuação (Score):

a.

```sql
SELECT
    user_id,
    age,
    last_month_salary,
    debt_ratio_corrigido,

using_lines_not_secured_personal_assets_tratada,
    (
        CASE WHEN age < 25 THEN 10
             WHEN age >= 25 AND age <= 55 THEN 0
             WHEN age > 55 THEN 5
             ELSE 5 END
        +
        CASE WHEN last_month_salary < 1500 THEN
15
             WHEN last_month_salary >= 1500 AND
last_month_salary < 3000 THEN 5
```

```
            WHEN last_month_salary >= 3000 THEN
0
            ELSE 7 END
        +
        CASE WHEN debt_ratio_corrigido > 0.5
THEN 20
            WHEN debt_ratio_corrigido > 0.2
THEN 10
            WHEN debt_ratio_corrigido <= 0.2
THEN 0
            ELSE 10 END
        +
        CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
            WHEN
using_lines_not_secured_personal_assets_tratada
> 2 THEN 8
            WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
            ELSE 7 END
    ) AS pontuacao_risco_total,
    CASE
        WHEN (
            CASE WHEN age < 25 THEN 10
                 WHEN age >= 25 AND age <= 55
THEN 0
                 WHEN age > 55 THEN 5
                 ELSE 5 END
            +
            CASE WHEN last_month_salary < 1500
THEN 15
                 WHEN last_month_salary >= 1500
AND last_month_salary < 3000 THEN 5
                 WHEN last_month_salary >= 3000
THEN 0
                 ELSE 7 END
            +
            CASE WHEN debt_ratio_corrigido > 0.5
THEN 20
                 WHEN debt_ratio_corrigido > 0.2
THEN 10
```

```
                WHEN debt_ratio_corrigido <=
0.2 THEN 0
                ELSE 10 END
            +
            CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
                WHEN
using_lines_not_secured_personal_assets_tratada
> 2 THEN 8
                WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
                ELSE 7 END
        ) <= 20 THEN 'Bom Pagador'
        WHEN (
            CASE WHEN age < 25 THEN 10
                WHEN age >= 25 AND age <= 55
THEN 0
                WHEN age > 55 THEN 5
                ELSE 5 END
            +
            CASE WHEN last_month_salary < 1500
THEN 15
                WHEN last_month_salary >= 1500
AND last_month_salary < 3000 THEN 5
                WHEN last_month_salary >= 3000
THEN 0
                ELSE 7 END
            +
            CASE WHEN debt_ratio_corrigido > 0.5
THEN 20
                WHEN debt_ratio_corrigido > 0.2
THEN 10
                WHEN debt_ratio_corrigido <=
0.2 THEN 0
                ELSE 10 END
            +
            CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
                WHEN
using_lines_not_secured_personal_assets_tratada
```

```
> 2 THEN 8
                WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
                ELSE 7 END
        ) > 20 AND (
            CASE WHEN age < 25 THEN 10
                 WHEN age >= 25 AND age <= 55
THEN 0
                 WHEN age > 55 THEN 5
                 ELSE 5 END
            +
            CASE WHEN last_month_salary < 1500
THEN 15
                 WHEN last_month_salary >= 1500
AND last_month_salary < 3000 THEN 5
                 WHEN last_month_salary >= 3000
THEN 0
                 ELSE 7 END
            +
            CASE WHEN debt_ratio_corrigido > 0.5
THEN 20
                 WHEN debt_ratio_corrigido > 0.2
THEN 10
                 WHEN debt_ratio_corrigido <=
0.2 THEN 0
                 ELSE 10 END
            +
            CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
                 WHEN
using_lines_not_secured_personal_assets_tratada
> 2 THEN 8
                 WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
                 ELSE 7 END
        ) <= 40 THEN 'Risco Moderado'
        ELSE 'Potencial Mau Pagador'
    END AS categoria_pagador
FROM
```

```
`projeto3-supercaja.projeto3_dataset.tabela_unificada_3`;
```

2. Matriz de Confusão - (A/P/R)

   a.

```
SELECT
    -- Componentes da Matriz de Confusão
    SUM(CASE WHEN predicted_category = 'Bom Pagador' AND default_flag = 0 THEN 1 ELSE 0 END) AS VerdadeiroPositivo,
    SUM(CASE WHEN predicted_category = 'Potencial Mau Pagador' AND default_flag = 1 THEN 1 ELSE 0 END) AS VerdadeiroNegativo,
    SUM(CASE WHEN predicted_category = 'Bom Pagador' AND default_flag = 1 THEN 1 ELSE 0 END) AS FalsoPositivo,
    SUM(CASE WHEN predicted_category = 'Potencial Mau Pagador' AND default_flag = 0 THEN 1 ELSE 0 END) AS FalsoNegativo,

    -- Métricas de Performance
    -- Acurácia: (VP + VN) / Total de Observações
    (CAST(SUM(CASE WHEN predicted_category = 'Bom Pagador' AND default_flag = 0 THEN 1 ELSE 0 END) AS NUMERIC) +
     CAST(SUM(CASE WHEN predicted_category = 'Potencial Mau Pagador' AND default_flag = 1 THEN 1 ELSE 0 END) AS NUMERIC))
    /
    COUNT(user_id) AS Acuracia,

    -- Precisão (para 'Bom Pagador'): VP / (VP + FP)
    -- Capacidade do modelo de identificar corretamente os 'Bom Pagador' dentre todos que ele classificou como 'Bom Pagador'
    CAST(SUM(CASE WHEN predicted_category = 'Bom Pagador' AND default_flag = 0 THEN 1 ELSE 0 END) AS NUMERIC)
    /
    NULLIF((CAST(SUM(CASE WHEN
```

```
predicted_category = 'Bom Pagador' AND
default_flag = 0 THEN 1 ELSE 0 END) AS NUMERIC)
+
            CAST(SUM(CASE WHEN
predicted_category = 'Bom Pagador' AND
default_flag = 1 THEN 1 ELSE 0 END) AS
NUMERIC)), 0) AS Precisao_BomPagador,

    -- Recall (para 'Bom Pagador'): VP / (VP +
FN)
    -- Capacidade do modelo de encontrar todos
os 'Bom Pagador' reais
    CAST(SUM(CASE WHEN predicted_category = 'Bom
Pagador' AND default_flag = 0 THEN 1 ELSE 0 END)
AS NUMERIC)
    /
    NULLIF((CAST(SUM(CASE WHEN
predicted_category = 'Bom Pagador' AND
default_flag = 0 THEN 1 ELSE 0 END) AS NUMERIC)
+
            CAST(SUM(CASE WHEN
predicted_category = 'Potencial Mau Pagador' AND
default_flag = 0 THEN 1 ELSE 0 END) AS
NUMERIC)), 0) AS Recall_BomPagador

FROM
    (
        SELECT
            user_id,
            default_flag,
            CASE
                WHEN (
                    CASE WHEN age < 25 THEN 10
                         WHEN age >= 25 AND age
<= 55 THEN 0
                         WHEN age > 55 THEN 5
                         ELSE 5 END
                    +
                    CASE WHEN last_month_salary
< 1500 THEN 15
                         WHEN last_month_salary
>= 1500 AND last_month_salary < 3000 THEN 5
                         WHEN last_month_salary
```

```
>= 3000 THEN 0
                            ELSE 7 END
                    +
                    CASE WHEN
debt_ratio_corrigido > 0.5 THEN 20
                            WHEN
debt_ratio_corrigido > 0.2 THEN 10
                            WHEN
debt_ratio_corrigido <= 0.2 THEN 0
                            ELSE 10 END
                    +
                    CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
                            WHEN
using_lines_not_secured_personal_assets_tratada
> 2 THEN 8
                            WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
                            ELSE 7 END
                ) <= 20 THEN 'Bom Pagador'
                ELSE 'Potencial Mau Pagador'
            END AS predicted_category
        FROM

`projeto3-supercaja.projeto3_dataset.tabela_unif
icada_3`
    ) subquery;
```

3. tabela_risco_clientes

    a.

```
CREATE TABLE IF NOT EXISTS
`projeto3-supercaja.projeto3_dataset.tabela_risc
o_clientes` AS
SELECT
    user_id,
    age,
    last_month_salary,
    debt_ratio_corrigido,

using_lines_not_secured_personal_assets_tratada,
```

```
    (
        CASE WHEN age < 25 THEN 10
             WHEN age >= 25 AND age <= 55 THEN 0
             WHEN age > 55 THEN 5
             ELSE 5 END
        +
        CASE WHEN last_month_salary < 1500 THEN
15
             WHEN last_month_salary >= 1500 AND
last_month_salary < 3000 THEN 5
             WHEN last_month_salary >= 3000 THEN
0
             ELSE 7 END
        +
        CASE WHEN debt_ratio_corrigido > 0.5
THEN 20
             WHEN debt_ratio_corrigido > 0.2
THEN 10
             WHEN debt_ratio_corrigido <= 0.2
THEN 0
             ELSE 10 END
        +
        CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
             WHEN
using_lines_not_secured_personal_assets_tratada
> 2 THEN 8
             WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
             ELSE 7 END
    ) AS pontuacao_risco_total,
    CASE
        WHEN (
            CASE WHEN age < 25 THEN 10
                 WHEN age >= 25 AND age <= 55
THEN 0
                 WHEN age > 55 THEN 5
                 ELSE 5 END
            +
            CASE WHEN last_month_salary < 1500
THEN 15
```

```
                WHEN last_month_salary >= 1500
AND last_month_salary < 3000 THEN 5
                WHEN last_month_salary >= 3000
THEN 0
                ELSE 7 END
            +
            CASE WHEN debt_ratio_corrigido > 0.5
THEN 20
                WHEN debt_ratio_corrigido > 0.2
THEN 10
                WHEN debt_ratio_corrigido <=
0.2 THEN 0
                ELSE 10 END
            +
            CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
                WHEN
using_lines_not_secured_personal_assets_tratada
> 2 THEN 8
                WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
                ELSE 7 END
        ) <= 20 THEN 'Bom Pagador'
        WHEN (
            CASE WHEN age < 25 THEN 10
                WHEN age >= 25 AND age <= 55
THEN 0
                WHEN age > 55 THEN 5
                ELSE 5 END
            +
            CASE WHEN last_month_salary < 1500
THEN 15
                WHEN last_month_salary >= 1500
AND last_month_salary < 3000 THEN 5
                WHEN last_month_salary >= 3000
THEN 0
                ELSE 7 END
            +
            CASE WHEN debt_ratio_corrigido > 0.5
THEN 20
                WHEN debt_ratio_corrigido > 0.2
```

```
THEN 10
                WHEN debt_ratio_corrigido <=
0.2 THEN 0
                ELSE 10 END
            +
            CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
                WHEN
using_lines_not_secured_personal_assets_tratada
> 2 THEN 8
                WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
                ELSE 7 END
        ) > 20 AND (
            CASE WHEN age < 25 THEN 10
                WHEN age >= 25 AND age <= 55
THEN 0
                WHEN age > 55 THEN 5
                ELSE 5 END
            +
            CASE WHEN last_month_salary < 1500
THEN 15
                WHEN last_month_salary >= 1500
AND last_month_salary < 3000 THEN 5
                WHEN last_month_salary >= 3000
THEN 0
                ELSE 7 END
            +
            CASE WHEN debt_ratio_corrigido > 0.5
THEN 20
                WHEN debt_ratio_corrigido > 0.2
THEN 10
                WHEN debt_ratio_corrigido <=
0.2 THEN 0
                ELSE 10 END
            +
            CASE WHEN
using_lines_not_secured_personal_assets_tratada
> 5 THEN 15
                WHEN
using_lines_not_secured_personal_assets_tratada
```

```
> 2 THEN 8
                  WHEN
using_lines_not_secured_personal_assets_tratada
<= 2 THEN 0
                  ELSE 7 END
        ) <= 40 THEN 'Risco Moderado'
        ELSE 'Potencial Mau Pagador'
    END AS categoria_pagador
FROM

`projeto3-supercaja.projeto3_dataset.tabela_unif
icada_3`;
```

c. 2.4.1 🟠 Representar dados por meio de tabela resumo ou scorecards✅

d. 2.4.2 🟠 Representar dados através de gráficos simples ✅

e. 2.4.3 🟠 Representar dados através de gráficos avançados ✅

f. 2.4.4 🟠 Aplicar opções de filtros (controle de dados) para gerenciamento e interação ✅

6. 🟩 2.5 Apresentar resultados

a. 2.5.1 🟢 Selecionar gráficos e informações relevantes✅

b. 2.5.2🟢Criar uma apresentação ✅

c. 2.5.3 🟢Apresentar resultados com conclusões e recomendações ✅