## 1.1 Limpieza de datos

1. Tabla Default:

```
-- Identificar Nulos
SELECT
 'user_id' AS column_name,
 COUNT(*) AS null_count
FROM
 `riesgo-relativo-p3.Data_Set.default`
WHERE
 SAFE_CAST(user_id AS INT64) IS NULL
UNION ALL
SELECT
 'default_flag' AS column_name,
 COUNT(*) AS null_count
FROM
 `riesgo-relativo-p3.Data_Set.default`
WHERE
 SAFE_CAST(default_flag AS INT64) IS NULL;

-- Identificar y manejar valores duplicados
WITH base_data AS (
 SELECT
  user_id,
  default_flag
 FROM
   `riesgo-relativo-p3.Data_Set.default`
),
duplicados AS (
 SELECT
   user_id,
   default_flag,
   COUNT(*) AS duplicado
 FROM
   base_data
 GROUP BY
   user_id,
   default_flag
 HAVING
   COUNT(*) > 1
)
SELECT
 user_id,
 default_flag,
 duplicado
FROM
 duplicados;

-- Identificar y manejar datos fuera del alcance del análisis (No se aplica correlación entre variables)
SELECT
 *
FROM
 `riesgo-relativo-p3.Data_Set.default`
WHERE
 user_id IS NULL OR default_flag IS NULL;

-- Identificar y manejar datos inconsistentes en variables categóricas
SELECT
 user_id,
 default_flag
FROM
 `riesgo-relativo-p3.Data_Set.default`
WHERE
 SAFE_CAST(user_id AS INT64) IS NULL OR SAFE_CAST(default_flag AS INT64) IS NULL;

#Identificar y manejar datos discrepantes en variables numéricas (OUTLIERS)
WITH ordered_values AS (
   SELECT
       default_flag,
       NTILE(4) OVER (ORDER BY default_flag) AS quartile
   FROM `riesgo-relativo-p3.Data_Set.default`
),
quartiles AS (
   SELECT
```

```
        MAX(CASE WHEN quartile = 1 THEN default_flag ELSE NULL END) AS Q1,
         MAX(CASE WHEN quartile = 2 THEN default_flag ELSE NULL END) AS Q2,  -- Usaremos Q2 como
mediana
        MAX(CASE WHEN quartile = 3 THEN default_flag ELSE NULL END) AS Q3,
        MAX(CASE WHEN quartile = 4 THEN default_flag ELSE NULL END) AS Q4
    FROM ordered_values
),
iqr_calc AS (
    SELECT
        Q1,
        Q3,
        Q3 - Q1 AS IQR,
        Q2 AS median_value
    FROM quartiles
),
outliers AS (
    SELECT
        user_id,  -- Asumiendo que tienes una columna `user_id` para identificar las filas
        default_flag,
        CASE
                  WHEN default_flag < (SELECT Q1 FROM iqr_calc) - 1.5 * (SELECT IQR FROM iqr_calc) THEN
(SELECT Q1 FROM iqr_calc) - 1.5 * (SELECT IQR FROM iqr_calc) - default_flag
                 WHEN default_flag > (SELECT Q3 FROM iqr_calc) + 1.5 * (SELECT IQR FROM iqr_calc) THEN
default_flag - (SELECT Q3 FROM iqr_calc) - 1.5 * (SELECT IQR FROM iqr_calc)
        END AS distance_from_limit
    FROM `riesgo-relativo-p3.Data_Set.default`
    WHERE default_flag < (SELECT Q1 FROM iqr_calc) - 1.5 * (SELECT IQR FROM iqr_calc)
       OR default_flag > (SELECT Q3 FROM iqr_calc) + 1.5 * (SELECT IQR FROM iqr_calc)
)
SELECT *
FROM outliers
ORDER BY distance_from_limit DESC
LIMIT 10;  -- Puedes ajustar este límite según cuántos outliers relevantes desees ver

# Comprobar y cambiar tipo de dato (Se corrobora que sea INT64)

#Crear nuevas variables (No se aplican nuevas variables)
```

2. Tabla Loans_detail::

```
 -- Identificar Nulos
SELECT
  'user_id' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_detail`
WHERE
  SAFE_CAST(user_id AS INT64) IS NULL
UNION ALL
  --
SELECT
  'more_90_days_overdue' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_detail`
WHERE
  SAFE_CAST(more_90_days_overdue AS INT64) IS NULL
UNION ALL
  --
SELECT
  'using_lines_not_secured_personal_assets' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_detail`
WHERE
  SAFE_CAST(using_lines_not_secured_personal_assets AS INT64) IS NULL
UNION ALL
  --
SELECT
  'number_times_delayed_payment_loan_30_59_days' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_detail`
WHERE
  SAFE_CAST(number_times_delayed_payment_loan_30_59_days AS INT64) IS NULL
UNION ALL
```

```
  --
SELECT
  'debt_ratio' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_detail`
WHERE
  SAFE_CAST(debt_ratio AS INT64) IS NULL
UNION ALL
  --
SELECT
  'number_times_delayed_payment_loan_60_89_days' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_detail`
WHERE
  SAFE_CAST(number_times_delayed_payment_loan_60_89_days AS INT64) IS NULL;

-- Identificar y manejar valores duplicados
WITH
  base_data AS (
  SELECT
    user_id,
    more_90_days_overdue,
    using_lines_not_secured_personal_assets,
    number_times_delayed_payment_loan_30_59_days,
    debt_ratio,
    number_times_delayed_payment_loan_60_89_days
  FROM
    `riesgo-relativo-p3.Data_Set.loans_detail` ),
  duplicado AS (
  SELECT
    user_id,
    more_90_days_overdue,
    using_lines_not_secured_personal_assets,
    number_times_delayed_payment_loan_30_59_days,
    debt_ratio,
    number_times_delayed_payment_loan_60_89_days,
    COUNT(*) AS duplicado
  FROM
    base_data
  GROUP BY
    user_id,
    more_90_days_overdue,
    using_lines_not_secured_personal_assets,
    number_times_delayed_payment_loan_30_59_days,
    debt_ratio,
    number_times_delayed_payment_loan_60_89_days
  HAVING
    COUNT(*) > 1 )
SELECT
  user_id,
  more_90_days_overdue,
  using_lines_not_secured_personal_assets,
  number_times_delayed_payment_loan_30_59_days,
  debt_ratio,
  number_times_delayed_payment_loan_60_89_days,
  duplicado
FROM
  duplicado;

-- Identificar y manejar datos fuera del alcance del análisis
 -- Correlación y Desviación Estándar entre
number_times_delayed_payment_loan_30_59_days y
number_times_delayed_payment_loan_60_89_days
SELECT
  CORR(number_times_delayed_payment_loan_30_59_days,
number_times_delayed_payment_loan_60_89_days) AS correlation_30_59_60_89,
  STDDEV(number_times_delayed_payment_loan_30_59_days) AS stddev_30_59,
  STDDEV(number_times_delayed_payment_loan_60_89_days) AS stddev_60_89
FROM `riesgo-relativo-p3.Data_Set.loans_detail`;
```

```sql
-- Correlación y Desviación Estándar entre more_90_days_overdue y
number_times_delayed_payment_loan_60_89_days
SELECT
  CORR(more_90_days_overdue, number_times_delayed_payment_loan_60_89_days) AS
correlation_90_60_89,
  STDDEV(more_90_days_overdue) AS stddev_90,
  STDDEV(number_times_delayed_payment_loan_60_89_days) AS stddev_60_89
FROM `riesgo-relativo-p3.Data_Set.loans_detail`;

-- Correlación y Desviación Estándar entre more_90_days_overdue y
number_times_delayed_payment_loan_30_59_days
SELECT
  CORR(more_90_days_overdue, number_times_delayed_payment_loan_30_59_days) AS
correlation_90_30_59,
  STDDEV(more_90_days_overdue) AS stddev_90,
  STDDEV(number_times_delayed_payment_loan_30_59_days) AS stddev_30_59
FROM `riesgo-relativo-p3.Data_Set.loans_detail`;

  -- Identificar y manejar datos inconsistentes en variables categóricas
SELECT
  user_id,
  more_90_days_overdue,
  using_lines_not_secured_personal_assets,
  number_times_delayed_payment_loan_30_59_days,
  debt_ratio,
  number_times_delayed_payment_loan_60_89_days,
FROM
  `riesgo-relativo-p3.Data_Set.loans_detail`
WHERE
  SAFE_CAST(user_id AS INT64) IS NULL
  OR SAFE_CAST(more_90_days_overdue AS INT64) IS NULL
  OR SAFE_CAST(using_lines_not_secured_personal_assets AS INT64) IS NULL
OR SAFE_CAST(number_times_delayed_payment_loan_30_59_days AS INT64) IS NULL
OR SAFE_CAST(debt_ratio AS INT64) IS NULL
OR SAFE_CAST(number_times_delayed_payment_loan_60_89_days AS INT64) IS NULL;

#Identificar y manejar datos discrepantes en variables numéricas (OUTLIERS)
WITH ordered_values AS (
  SELECT
    more_90_days_overdue,
    using_lines_not_secured_personal_assets,
    number_times_delayed_payment_loan_30_59_days,
    debt_ratio,
    number_times_delayed_payment_loan_60_89_days,
    NTILE(4) OVER (ORDER BY more_90_days_overdue) AS quartile_90_days,
    NTILE(4) OVER (ORDER BY using_lines_not_secured_personal_assets) AS
quartile_unsecured,
    NTILE(4) OVER (ORDER BY number_times_delayed_payment_loan_30_59_days) AS
quartile_30_59,
    NTILE(4) OVER (ORDER BY debt_ratio) AS quartile_debt_ratio,
    NTILE(4) OVER (ORDER BY number_times_delayed_payment_loan_60_89_days) AS
quartile_60_89
  FROM
    `riesgo-relativo-p3.Data_Set.loans_detail`
),
quartiles AS (
  SELECT
    MAX(CASE WHEN quartile_90_days = 1 THEN more_90_days_overdue ELSE NULL END) AS
Q1_90_days,
    MAX(CASE WHEN quartile_90_days = 2 THEN more_90_days_overdue ELSE NULL END) AS
Q2_90_days,
    MAX(CASE WHEN quartile_90_days = 3 THEN more_90_days_overdue ELSE NULL END) AS
Q3_90_days,
    MAX(CASE WHEN quartile_90_days = 4 THEN more_90_days_overdue ELSE NULL END) AS
Q4_90_days,

    MAX(CASE WHEN quartile_unsecured = 1 THEN
using_lines_not_secured_personal_assets ELSE NULL END) AS Q1_unsecured,
    MAX(CASE WHEN quartile_unsecured = 2 THEN
using_lines_not_secured_personal_assets ELSE NULL END) AS Q2_unsecured,
    MAX(CASE WHEN quartile_unsecured = 3 THEN
using_lines_not_secured_personal_assets ELSE NULL END) AS Q3_unsecured,
    MAX(CASE WHEN quartile_unsecured = 4 THEN
using_lines_not_secured_personal_assets ELSE NULL END) AS Q4_unsecured,

    MAX(CASE WHEN quartile_30_59 = 1 THEN
number_times_delayed_payment_loan_30_59_days ELSE NULL END) AS Q1_30_59,
    MAX(CASE WHEN quartile_30_59 = 2 THEN
number_times_delayed_payment_loan_30_59_days ELSE NULL END) AS Q2_30_59,
    MAX(CASE WHEN quartile_30_59 = 3 THEN
number_times_delayed_payment_loan_30_59_days ELSE NULL END) AS Q3_30_59,
    MAX(CASE WHEN quartile_30_59 = 4 THEN
number_times_delayed_payment_loan_30_59_days ELSE NULL END) AS Q4_30_59,

    MAX(CASE WHEN quartile_debt_ratio = 1 THEN debt_ratio ELSE NULL END) AS
Q1_debt_ratio,
    MAX(CASE WHEN quartile_debt_ratio = 2 THEN debt_ratio ELSE NULL END) AS
Q2_debt_ratio,
    MAX(CASE WHEN quartile_debt_ratio = 3 THEN debt_ratio ELSE NULL END) AS
Q3_debt_ratio,
    MAX(CASE WHEN quartile_debt_ratio = 4 THEN debt_ratio ELSE NULL END) AS
Q4_debt_ratio,

    MAX(CASE WHEN quartile_60_89 = 1 THEN
number_times_delayed_payment_loan_60_89_days ELSE NULL END) AS Q1_60_89,
    MAX(CASE WHEN quartile_60_89 = 2 THEN
number_times_delayed_payment_loan_60_89_days ELSE NULL END) AS Q2_60_89,
    MAX(CASE WHEN quartile_60_89 = 3 THEN
number_times_delayed_payment_loan_60_89_days ELSE NULL END) AS Q3_60_89,
    MAX(CASE WHEN quartile_60_89 = 4 THEN
number_times_delayed_payment_loan_60_89_days ELSE NULL END) AS Q4_60_89
  FROM
    ordered_values
),
iqr_calc AS (
  SELECT
    Q1_90_days, Q3_90_days, Q3_90_days - Q1_90_days AS IQR_90_days, Q2_90_days AS
median_90_days,
    Q1_unsecured, Q3_unsecured, Q3_unsecured - Q1_unsecured AS IQR_unsecured,
Q2_unsecured AS median_unsecured,
    Q1_30_59, Q3_30_59, Q3_30_59 - Q1_30_59 AS IQR_30_59, Q2_30_59 AS median_30_59,
    Q1_debt_ratio, Q3_debt_ratio, Q3_debt_ratio - Q1_debt_ratio AS IQR_debt_ratio,
Q2_debt_ratio AS median_debt_ratio,
    Q1_60_89, Q3_60_89, Q3_60_89 - Q1_60_89 AS IQR_60_89, Q2_60_89 AS median_60_89
  FROM
    quartiles
)
SELECT
  COUNT(CASE WHEN more_90_days_overdue < (SELECT Q1_90_days FROM iqr_calc) - 1.5 *
(SELECT IQR_90_days FROM iqr_calc) OR more_90_days_overdue > (SELECT Q3_90_days
FROM iqr_calc) + 1.5 * (SELECT IQR_90_days FROM iqr_calc) THEN 1 ELSE NULL END) AS
outliers_90_days,
  COUNT(CASE WHEN using_lines_not_secured_personal_assets < (SELECT Q1_unsecured
FROM iqr_calc) - 1.5 * (SELECT IQR_unsecured FROM iqr_calc) OR
using_lines_not_secured_personal_assets > (SELECT Q3_unsecured FROM iqr_calc) + 1.5
* (SELECT IQR_unsecured FROM iqr_calc) THEN 1 ELSE NULL END) AS outliers_unsecured,
  COUNT(CASE WHEN number_times_delayed_payment_loan_30_59_days < (SELECT Q1_30_59
FROM iqr_calc) - 1.5 * (SELECT IQR_30_59 FROM iqr_calc) OR
number_times_delayed_payment_loan_30_59_days > (SELECT Q3_30_59 FROM iqr_calc) +
1.5 * (SELECT IQR_30_59 FROM iqr_calc) THEN 1 ELSE NULL END) AS outliers_30_59,
  COUNT(CASE WHEN debt_ratio < (SELECT Q1_debt_ratio FROM iqr_calc) - 1.5 * (SELECT
IQR_debt_ratio FROM iqr_calc) OR debt_ratio > (SELECT Q3_debt_ratio FROM iqr_calc)
+ 1.5 * (SELECT IQR_debt_ratio FROM iqr_calc) THEN 1 ELSE NULL END) AS
outliers_debt_ratio,
  COUNT(CASE WHEN number_times_delayed_payment_loan_60_89_days < (SELECT Q1_60_89
FROM iqr_calc) - 1.5 * (SELECT IQR_60_89 FROM iqr_calc) OR
number_times_delayed_payment_loan_60_89_days > (SELECT Q3_60_89 FROM iqr_calc) +
1.5 * (SELECT IQR_60_89 FROM iqr_calc) THEN 1 ELSE NULL END) AS outliers_60_89
FROM
  `riesgo-relativo-p3.Data_Set.loans_detail`;

# Comprobar y cambiar tipo de dato (Se corrobora que sea INT64)

#Crear nuevas variables
WITH datos_filtrados AS (
  SELECT
    user_id,
    debt_ratio,
    using_lines_not_secured_personal_assets,
    -- Filtrar valores mayores a 20
    CASE
      WHEN number_times_delayed_payment_loan_30_59_days > 20 THEN NULL
      ELSE number_times_delayed_payment_loan_30_59_days
    END AS number_times_delayed_payment_loan_30_59_days,

    CASE
      WHEN number_times_delayed_payment_loan_60_89_days > 20 THEN NULL
      ELSE number_times_delayed_payment_loan_60_89_days
    END AS number_times_delayed_payment_loan_60_89_days,

    CASE
      WHEN more_90_days_overdue > 20 THEN NULL
      ELSE more_90_days_overdue
    END AS more_90_days_overdue
  FROM
    `proyecto2-super-caja.Data_set.loans_detail`
),

datos_suma AS (
  SELECT
    user_id,
    debt_ratio,
    using_lines_not_secured_personal_assets,
    COALESCE(number_times_delayed_payment_loan_30_59_days, 0) +
    COALESCE(number_times_delayed_payment_loan_60_89_days, 0) +
    COALESCE(more_90_days_overdue, 0) AS delay_30_59_90
  FROM
    datos_filtrados
),

segmentacion AS (
  SELECT
    user_id,
    debt_ratio,
    using_lines_not_secured_personal_assets,
    delay_30_59_90,

    -- Segmentación Basada en Debt Ratio
    CASE
      WHEN debt_ratio < 0.30 THEN 'Bajo Riesgo'
      WHEN debt_ratio >= 0.30 AND debt_ratio <= 0.60 THEN 'Medio Riesgo'
      ELSE 'Alto Riesgo'
    END AS Incumplimiento_debt_ratio,

    -- Segmentación Basada en Uso de Líneas de Crédito No Aseguradas
    CASE
      WHEN using_lines_not_secured_personal_assets < 0.30 THEN 'Bajo Riesgo'
      WHEN using_lines_not_secured_personal_assets >= 0.30 AND
using_lines_not_secured_personal_assets <= 0.60 THEN 'Medio Riesgo'
      ELSE 'Alto Riesgo'
    END AS Incumplimiento_unsec_line,

    -- Segmentación Basada en la Nueva Columna de Suma
    CASE
      WHEN delay_30_59_90 = 0 THEN 'Ningún Retraso'
      WHEN delay_30_59_90 > 0 AND delay_30_59_90 <= 5 THEN 'Poco Retraso'
      ELSE 'Muchos Retrasos'
    END AS segmentacion_delay
  FROM
    datos_suma
)

-- Seleccionar los resultados finales
SELECT
  user_id,
  debt_ratio,
  Incumplimiento_debt_ratio,
  using_lines_not_secured_personal_assets,
  Incumplimiento_unsec_line,
  delay_30_59_90,
  segmentacion_delay
FROM
  segmentacion
ORDER BY
  user_id;
```

## 3. Tabla Loans_outstanding:

```
-- Identificar Nulos
SELECT
  'user_id' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_outstanding`
WHERE
  SAFE_CAST(user_id AS INT64) IS NULL
UNION ALL
SELECT
  'loan_id' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_outstanding`
WHERE
  SAFE_CAST(loan_id AS INT64) IS NULL
UNION ALL
SELECT
  'loan_type' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.loans_outstanding`
WHERE
  SAFE_CAST(loan_type AS INT64) IS NULL;

-- Identificar y manejar valores duplicados
WITH base_data AS (
  SELECT
    user_id,
    loan_id,
    loan_type
  FROM
    `riesgo-relativo-p3.Data_Set.loans_outstanding`
),
duplicado AS (
  SELECT
    user_id,
    loan_id,
    loan_type,
    COUNT(*) AS duplicado
  FROM
    base_data
  GROUP BY
    user_id,
    loan_id,
    loan_type
  HAVING
    COUNT(*) > 1
)
SELECT
  user_id,
  loan_id,
  loan_type,
  duplicado
FROM
  duplicado;

-- Identificar y manejar datos inconsistentes en variables categóricas
SELECT
  user_id,
  loan_id,
  loan_type
```

```sql
FROM
  `riesgo-relativo-p3.Data_Set.loans_outstanding`
WHERE
  SAFE_CAST(user_id AS INT64) IS NULL
  OR SAFE_CAST(loan_id AS INT64) IS NULL;

-- Identificar y manejar datos discrepantes en variables numéricas (OUTLIERS)
WITH ordered_values AS (
  SELECT
    loan_id,
    NTILE(4) OVER (ORDER BY loan_id) AS quartile_loan_id
  FROM
    `riesgo-relativo-p3.Data_Set.loans_outstanding`
),
quartiles AS (
  SELECT
    MAX(CASE WHEN quartile_loan_id = 1 THEN loan_id ELSE NULL END) AS Q1_loan,
    MAX(CASE WHEN quartile_loan_id = 2 THEN loan_id ELSE NULL END) AS Q2_loan,
    MAX(CASE WHEN quartile_loan_id = 3 THEN loan_id ELSE NULL END) AS Q3_loan,
    MAX(CASE WHEN quartile_loan_id = 4 THEN loan_id ELSE NULL END) AS Q4_loan
  FROM
    ordered_values
),
iqr_calc AS (
  SELECT
    Q1_loan,
    Q3_loan,
    Q3_loan - Q1_loan AS IQR_loan,
    Q2_loan AS median_loan
  FROM
    quartiles
)
SELECT
  COUNT(CASE WHEN loan_id < (SELECT Q1_loan FROM iqr_calc) - 1.5 * (SELECT IQR_loan
FROM iqr_calc) OR loan_id > (SELECT Q3_loan FROM iqr_calc) + 1.5 * (SELECT IQR_loan
FROM iqr_calc) THEN 1 ELSE NULL END) AS outliers_loan
FROM
  `riesgo-relativo-p3.Data_Set.loans_outstanding`;

-- Comprobar y cambiar tipo de dato
-- LOWER (convierte las letras mayúsculas en minúsculas)
SELECT
  LOWER(
    REPLACE(loan_type, 'other', 'others')
  ) AS normalized_loan_type
FROM
  `riesgo-relativo-p3.Data_Set.loans_outstanding`;

-- Crear nuevas variables
SELECT
  user_id,
  SUM(CASE
      WHEN loan_type = 'real estate' THEN 1
      ELSE 0
  END) AS real_estate_loan_type,
  SUM(CASE
      WHEN loan_type = 'other' THEN 1
      ELSE 0
  END) AS others_loan_type,
  SUM(CASE
      WHEN loan_type = 'real estate' THEN 1
      ELSE 0
  END) + SUM(CASE
      WHEN loan_type = 'other' THEN 1
      ELSE 0
  END) AS total_loan_type
FROM
  `riesgo-relativo-p3.Data_Set.loans_outstanding`
GROUP BY
  user_id;

-- Seleccionar los resultados finales
SELECT
```

```
  user_id,
  real_estate_loan_type,
  others_loan_type,
  total_loan_type
FROM
  (
    SELECT
      user_id,
      SUM(CASE
          WHEN loan_type = 'real estate' THEN 1
          ELSE 0
      END) AS real_estate_loan_type,
      SUM(CASE
          WHEN loan_type = 'other' THEN 1
          ELSE 0
      END) AS others_loan_type,
      SUM(CASE
          WHEN loan_type = 'real estate' THEN 1
          ELSE 0
      END) + SUM(CASE
          WHEN loan_type = 'other' THEN 1
          ELSE 0
      END) AS total_loan_type
    FROM
      `riesgo-relativo-p3.Data_Set.loans_outstanding`
    GROUP BY
      user_id
  ) AS segmentacion
ORDER BY
  user_id;
```

4. Tabla user_info

```
 -- Identificar Nulos
SELECT
  'user_id' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.user_info`
WHERE
  SAFE_CAST(user_id AS INT64) IS NULL
UNION ALL
  --
SELECT
  'age' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.user_info`
WHERE
  SAFE_CAST(age AS INT64) IS NULL
UNION ALL
  --
SELECT
  'last_month_salary' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.user_info`
WHERE
  SAFE_CAST(last_month_salary AS INT64) IS NULL
UNION ALL
  --
SELECT
  'number_dependents' AS column_name,
  COUNT(*) AS null_count
FROM
  `riesgo-relativo-p3.Data_Set.user_info`
WHERE
  SAFE_CAST(number_dependents AS INT64) IS NULL; -- Identificar y manejar valores
duplicados
WITH
  base_data AS (
  SELECT
    user_id,
    age,
    sex,
    last_month_salary,
    number_dependents,
  FROM
    `riesgo-relativo-p3.Data_Set.user_info` ),
  duplicado AS (
  SELECT
    user_id,
    age,
    sex,
    last_month_salary,
    number_dependents,
    COUNT(*) AS duplicado
  FROM
    base_data
  GROUP BY
    user_id,
    age,
    sex,
    last_month_salary,
    number_dependents
  HAVING
    COUNT(*) > 1 )
SELECT
  user_id,
  age,
  sex,
  last_month_salary,
  number_dependents,
  duplicado
FROM
  duplicado; -- Identificar y manejar datos fuera del alcance del análisis (NO
aplica) -- Identificar y manejar datos inconsistentes en variables categóricas
SELECT
  age,
  sex,
  last_month_salary,
  number_dependents,
FROM
  `riesgo-relativo-p3.Data_Set.user_info`
WHERE
  SAFE_CAST(age AS INT64) IS NULL
  OR SAFE_CAST(sex AS INT64) IS NULL
  OR SAFE_CAST(last_month_salary AS INT64) IS NULL
  OR SAFE_CAST(number_dependents AS INT64) IS NULL; #Identificar y manejar datos
discrepantes en variables numéricas (OUTLIERS)
WITH
  ordered_values AS (
  SELECT
    age,
    sex,
    last_month_salary,
    number_dependents,
    NTILE(4) OVER (ORDER BY age) AS quartile_age,
    NTILE(4) OVER (ORDER BY last_month_salary) AS quartile_last_month_salary,
    NTILE(4) OVER (ORDER BY number_dependents) AS quartile_number_dependents
  FROM
    `riesgo-relativo-p3.Data_Set.user_info` ),
  quartiles AS (
  SELECT
    MAX(CASE
        WHEN quartile_age = 1 THEN age
        ELSE NULL
    END
      ) AS Q1_age,
    MAX(CASE
        WHEN quartile_age = 2 THEN age
        ELSE NULL
    END
      ) AS Q2_age,
    MAX(CASE
        WHEN quartile_age = 3 THEN age
        ELSE NULL
    END
      ) AS Q3_age,
    MAX(CASE
        WHEN quartile_age = 4 THEN age
        ELSE NULL
    END
      ) AS Q4_age,
    MAX(CASE
        WHEN quartile_last_month_salary = 1 THEN last_month_salary
        ELSE NULL
    END
      ) AS Q1_last_month_salary,
    MAX(CASE
        WHEN quartile_last_month_salary = 2 THEN last_month_salary
        ELSE NULL
    END
      ) AS Q2_last_month_salary,
    MAX(CASE
        WHEN quartile_last_month_salary = 3 THEN last_month_salary
        ELSE NULL
    END
      ) AS Q3_last_month_salary,
    MAX(CASE
        WHEN quartile_last_month_salary = 4 THEN last_month_salary
        ELSE NULL
    END
      ) AS Q4_last_month_salary,
    MAX(CASE
        WHEN quartile_number_dependents = 1 THEN number_dependents
        ELSE NULL
    END
      ) AS Q1_number_dependents,
    MAX(CASE
        WHEN quartile_number_dependents = 2 THEN number_dependents
        ELSE NULL
    END
      ) AS Q2_number_dependents,
    MAX(CASE
        WHEN quartile_number_dependents = 3 THEN number_dependents
        ELSE NULL
    END
      ) AS Q3_number_dependents,
    MAX(CASE
        WHEN quartile_number_dependents = 4 THEN number_dependents
        ELSE NULL
    END
      ) AS Q4_number_dependents
  FROM
    ordered_values ),
  iqr_calc AS (
  SELECT
    Q1_age,
    Q3_age,
    Q3_age - Q1_age AS IQR_age,
    Q2_age AS median_age,
    Q1_last_month_salary,
    Q3_last_month_salary,
    Q3_last_month_salary - Q1_last_month_salary AS IQR_last_month_salary,
    Q2_last_month_salary AS median_last_month_salary,
    Q1_number_dependents,
    Q3_number_dependents,
    Q3_number_dependents - Q1_number_dependents AS IQR_number_dependents,
    Q2_number_dependents AS median_number_dependents
  FROM
    quartiles )
SELECT
  COUNT(CASE
      WHEN age < ( SELECT Q1_age FROM iqr_calc) - 1.5 * ( SELECT IQR_age FROM
iqr_calc) OR age > ( SELECT Q3_age FROM iqr_calc) + 1.5 * ( SELECT IQR_age FROM
iqr_calc) THEN 1
```

```
      ELSE NULL
  END
    ) AS outliers_age,
  COUNT(CASE
      WHEN last_month_salary < ( SELECT Q1_last_month_salary FROM iqr_calc) - 1.5 *
( SELECT IQR_last_month_salary FROM iqr_calc) OR last_month_salary > ( SELECT
Q3_last_month_salary FROM iqr_calc) + 1.5 * ( SELECT IQR_last_month_salary FROM
iqr_calc) THEN 1
      ELSE NULL
  END
    ) AS outliers_last_month_salary,
  COUNT(CASE
      WHEN number_dependents < ( SELECT Q1_number_dependents FROM iqr_calc) - 1.5 *
( SELECT IQR_number_dependents FROM iqr_calc) OR number_dependents > ( SELECT
Q3_number_dependents FROM iqr_calc) + 1.5 * ( SELECT IQR_number_dependents FROM
iqr_calc) THEN 1
      ELSE NULL
  END
    ) AS outliers_number_dependents
FROM
  `riesgo-relativo-p3.Data_Set.user_info`;

  # Comprobar y cambiar tipo de dato (NO aplica)
```

## 5. Unión de tablas

1

```
SELECT
  DF.user_id,
  DF.default_flag,
  UI.age,
  UI.generational_group,
  UPPER(UI.sex) AS sex,
  last_month_salary,
  COALESCE(UI.number_dependents, 0) AS number_dependents,
  LD.debt_ratio,
  LD.Incumplimiento_debt_ratio,
  LD.using_lines_not_secured_personal_assets,
  LD.Incumplimiento_unsec_line,
  LD.delay_30_59_90,
  LD.segmentacion_delay,
FROM
  riesgo-relativo-p3.Data_Set.default AS DF
LEFT JOIN
  riesgo-relativo-p3.Data_Set.user_info1 AS UI
ON
  DF.user_id = UI.user_id
LEFT JOIN
  riesgo-relativo-p3.Data_Set.Loans_detail AS LD
ON
  DF.user_id = LD.user_id
ORDER BY
  user_id;
```

2

```
-- Crear una tabla temporal con la moda de last_month_salary por default_flag
WITH salary_mode AS (
  -- Subconsulta para calcular la moda de last_month_salary por default_flag
  SELECT
    default_flag,
    last_month_salary AS mode_salary
  FROM (
    SELECT
      default_flag,
      last_month_salary,
      COUNT(*) AS frequency,
      ROW_NUMBER() OVER (PARTITION BY default_flag ORDER BY COUNT(*) DESC,
last_month_salary) AS rn
    FROM
      `riesgo-relativo-p3.Data_Set.Data`
    WHERE
```

```
      last_month_salary IS NOT NULL AND last_month_salary > 0
    GROUP BY
      default_flag, last_month_salary
  )
  WHERE
    rn = 1
)

-- Seleccionar los datos y reemplazar los nulos en last_month_salary
SELECT
  DF.user_id,
  DF.default_flag,
  DF.age,
  DF.generational_group,
  DF.sex,
  COALESCE(
    CASE
      WHEN DF.last_month_salary IS NULL THEN salary_mode.mode_salary
      ELSE DF.last_month_salary
    END,
    DF.last_month_salary
  ) AS last_month_salary,
  COALESCE(DF.number_dependents, 0) AS number_dependents,
  LT.real_estate_loan_type,
  LT.others_loan_type,
  LT.total_loan_type,
  DF.debt_ratio,
  DF.Incumplimiento_debt_ratio,
  DF.using_lines_not_secured_personal_assets,
  DF.Incumplimiento_unsec_line,
  DF.delay_30_59_90,
  DF.segmentacion_delay
FROM
  `riesgo-relativo-p3.Data_Set.Data` AS DF
LEFT JOIN
  `riesgo-relativo-p3.Data_Set.Loans_type` AS LT
ON
  DF.user_id = LT.user_id
LEFT JOIN
  salary_mode
ON
  DF.default_flag = salary_mode.default_flag
ORDER BY
  DF.user_id;
```

## 1.2 Análisis exploratorio

### 1.Cuartiles de malos pagadores

```
#CALCULAR LOS CUARTILES DE MALOS PAGADORES POR VARIABLE
WITH base_data AS (
    SELECT
        age,
        default_flag
    FROM `riesgo-relativo-p3.Data_Set.Data_Set_Completo`
),
---Calcular los cuartiles dependiendo la edad.
quartiles AS (
    SELECT
        age,
        default_flag,
        NTILE(4) OVER (ORDER BY age) AS age_quartile
    FROM base_data ---datos de donde provienen las variables
),
-- Calcula el número total de malos pagadores
quartile_risk AS (
    SELECT
```

```
        age_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
    FROM quartiles
    GROUP BY age_quartile
),
---rango de edad (mínimo y máximo) para cada cuartil.
quartile_ranges AS (
    SELECT
        age_quartile,
        MIN(age) AS min_age,
        MAX(age) AS max_age
    FROM quartiles
    GROUP BY age_quartile
)
SELECT
    q.age_quartile,
    q.total_count,
    q.total_bad_payers,
    r.min_age,
    r.max_age
FROM quartile_risk q
JOIN quartile_ranges r
ON q.age_quartile = r.age_quartile
ORDER BY age_quartile ASC;

#CALCULAR LOS CUARTILES DE MALOS PAGADORES LAST_MONTH
WITH base_data AS (
    SELECT
        last_month_salary,
        default_flag
    FROM `riesgo-relativo-p3.Data_Set.Data_Set_Completo`
),
---Calcular los cuartiles dependiendo la edad.
quartiles AS (
    SELECT
        last_month_salary,
        default_flag,
        NTILE(4) OVER (ORDER BY last_month_salary) AS last_month_salary_quartile
    FROM base_data ---datos de donde provienen las variables
),
-- Calcula el número total de malos pagadores
quartile_risk AS (
    SELECT
        last_month_salary_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
    FROM quartiles
    GROUP BY last_month_salary_quartile
),
---rango de edad (mínimo y máximo) para cada cuartil.
quartile_ranges AS (
    SELECT
        last_month_salary_quartile,
        MIN(last_month_salary) AS min_last_month_salary,
        MAX(last_month_salary) AS max_last_month_salary
    FROM quartiles
    GROUP BY last_month_salary_quartile
)
SELECT
    q.last_month_salary_quartile,
    q.total_count,
    q.total_bad_payers,
    r.min_last_month_salary,
    r.max_last_month_salary
FROM quartile_risk q
JOIN quartile_ranges r
ON q.last_month_salary_quartile = r.last_month_salary_quartile
ORDER BY last_month_salary_quartile ASC;

#CALCULAR LOS CUARTILES DE MALOS PAGADORES NUMBER_DEPENDENTS
WITH base_data AS (
    SELECT
```

```
        number_dependents,
        default_flag
    FROM `riesgo-relativo-p3.Data_Set.Data_Set_Completo`
),
---Calcular los cuartiles dependiendo la edad.
quartiles AS (
    SELECT
        number_dependents,
        default_flag,
        NTILE(4) OVER (ORDER BY number_dependents) AS number_dependents_quartile
    FROM base_data ---datos de donde provienen las variables
),
-- Calcula el número total de malos pagadores
quartile_risk AS (
    SELECT
        number_dependents_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
    FROM quartiles
    GROUP BY number_dependents_quartile
),
---rango de edad (mínimo y máximo) para cada cuartil.
quartile_ranges AS (
    SELECT
        number_dependents_quartile,
        MIN(number_dependents) AS min_number_dependents,
        MAX(number_dependents) AS max_number_dependents
    FROM quartiles
    GROUP BY number_dependents_quartile
)
SELECT
    q.number_dependents_quartile,
    q.total_count,
    q.total_bad_payers,
    r.min_number_dependents,
    r.max_number_dependents
FROM quartile_risk q
JOIN quartile_ranges r
ON q.number_dependents_quartile = r.number_dependents_quartile
ORDER BY number_dependents_quartile ASC;

#CALCULAR LOS CUARTILES DE MALOS PAGADORES DEBT_RATIO
WITH base_data AS (
    SELECT
        debt_ratio,
        default_flag
    FROM `riesgo-relativo-p3.Data_Set.Data_Set_Completo`
),
---Calcular los cuartiles dependiendo la edad.
quartiles AS (
    SELECT
        debt_ratio,
        default_flag,
        NTILE(4) OVER (ORDER BY debt_ratio) AS debt_ratio_quartile
    FROM base_data ---datos de donde provienen las variables
),
-- Calcula el número total de malos pagadores
quartile_risk AS (
    SELECT
        debt_ratio_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
    FROM quartiles
    GROUP BY debt_ratio_quartile
),
---rango de edad (mínimo y máximo) para cada cuartil.
quartile_ranges AS (
    SELECT
        debt_ratio_quartile,
        MIN(debt_ratio) AS min_debt_ratio,
        MAX(debt_ratio) AS max_debt_ratio
    FROM quartiles
    GROUP BY debt_ratio_quartile
```

```
)
SELECT
    q.debt_ratio_quartile,
    q.total_count,
    q.total_bad_payers,
    r.min_debt_ratio,
    r.max_debt_ratio
FROM quartile_risk q
JOIN quartile_ranges r
ON q.debt_ratio_quartile = r.debt_ratio_quartile
ORDER BY debt_ratio_quartile ASC;

#CALCULAR LOS CUARTILES DE MALOS PAGADORES using_lines_not_secured_personal_assets
WITH base_data AS (
    SELECT
        using_lines_not_secured_personal_assets,
        default_flag
    FROM `riesgo-relativo-p3.Data_Set.Data_Set_Completo`
),
---Calcular los cuartiles dependiendo la edad.
quartiles AS (
    SELECT
        using_lines_not_secured_personal_assets,
        default_flag,
        NTILE(4) OVER (ORDER BY using_lines_not_secured_personal_assets) AS
using_lines_not_secured_personal_assets_quartile
    FROM base_data ---datos de donde provienen las variables
),
-- Calcula el número total de malos pagadores
quartile_risk AS (
    SELECT
        using_lines_not_secured_personal_assets_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
    FROM quartiles
    GROUP BY using_lines_not_secured_personal_assets_quartile
),
---rango de edad (mínimo y máximo) para cada cuartil.
quartile_ranges AS (
    SELECT
        using_lines_not_secured_personal_assets_quartile,
        MIN(using_lines_not_secured_personal_assets) AS
min_using_lines_not_secured_personal_assets,
        MAX(using_lines_not_secured_personal_assets) AS
max_using_lines_not_secured_personal_assets
    FROM quartiles
    GROUP BY using_lines_not_secured_personal_assets_quartile
)
SELECT
    q.using_lines_not_secured_personal_assets_quartile,
    q.total_count,
    q.total_bad_payers,
    r.min_using_lines_not_secured_personal_assets,
    r.max_using_lines_not_secured_personal_assets
FROM quartile_risk q
JOIN quartile_ranges r
ON q.using_lines_not_secured_personal_assets_quartile =
r.using_lines_not_secured_personal_assets_quartile
ORDER BY using_lines_not_secured_personal_assets_quartile ASC;

#CALCULAR LOS CUARTILES DE MALOS PAGADORES delay_30_59_90
-- Calcular los cuartiles de malos pagadores delay_30_59_90
WITH base_data AS (
    SELECT
        delay_30_59_90,
        default_flag
    FROM `riesgo-relativo-p3.Data_Set.Data_Set_Completo`
    WHERE default_flag = 1
),
-- Calcular los cuartiles dependiendo del delay_30_59_90
quartiles AS (
    SELECT
        delay_30_59_90,
        default_flag,
        NTILE(4) OVER (ORDER BY delay_30_59_90) AS delay_30_59_90_quartile
    FROM base_data
),
-- Calcula el número total de malos pagadores y el total de registros para cada
cuartil
quartile_risk AS (
    SELECT
        delay_30_59_90_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers
    FROM quartiles
    GROUP BY delay_30_59_90_quartile
),
-- Rango de delay_30_59_90 (mínimo y máximo) para cada cuartil
quartile_ranges AS (
    SELECT
        delay_30_59_90_quartile,
        MIN(delay_30_59_90) AS min_delay_30_59_90,
        MAX(delay_30_59_90) AS max_delay_30_59_90
    FROM quartiles
    GROUP BY delay_30_59_90_quartile
)
SELECT
    q.delay_30_59_90_quartile,
    q.total_count,
    q.total_bad_payers,
    r.min_delay_30_59_90,
    r.max_delay_30_59_90
FROM quartile_risk q
JOIN quartile_ranges r
ON q.delay_30_59_90_quartile = r.delay_30_59_90_quartile
ORDER BY q.delay_30_59_90_quartile ASC;
```

## 2.-Calcular la correlaciòn entre variables nùmericas continuas.

```
SELECT
CORR(age,last_month_salary) AS age_last_month,
CORR (age,debt_ratio) AS age_debt_ratio,
CORR (debt_ratio,last_month_salary) AS debt_last_month,
CORR (debt_ratio, using_lines_not_secured_personal_assets) AS debt_usin_lines
 FROM `riesgo-relativo-p3.Data_Set.Data_Set_Completo`
```

## 3.- Calcular riesgo relativo e hipótesis

```
-- Paso 1: Crear una tabla temporal con los datos base
WITH base_data AS (
    SELECT
        age,
        default_flag
    FROM riesgo-relativo-p3.Data_Set.Data_Set_Completo
),

-- Paso 2: Calcular los cuartiles para la variable age
quartiles AS (
    SELECT
        age,
        default_flag,
        NTILE(4) OVER (ORDER BY age) AS age_quartile
    FROM base_data
),

-- Paso 3: Calcular el número total de malos y buenos pagadores por cuartil
quartile_risk AS (
    SELECT
        age_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
        AVG(default_flag) AS tasa_bad_payers,
        COUNT(*) - SUM(default_flag) AS total_good_payers,
    FROM quartiles
    GROUP BY age_quartile
),

-- Paso 4: Obtener el rango de edad (mínimo y máximo) para cada cuartil
quartile_ranges AS (
    SELECT
        age_quartile,
        MIN(age) AS min_age,
        MAX(age) AS max_age
    FROM quartiles
    GROUP BY age_quartile
),

-- Paso 5: Calcular el riesgo relativo usando la nueva fórmula
risk_relative AS (
    SELECT
        q.age_quartile,
        q.total_count,
        q.total_bad_payers,
        q.tasa_bad_payers,
        q.total_good_payers,
        r.min_age,
        r.max_age,
        CASE
            WHEN q.age_quartile = 1 THEN
(((q1.total_bad_payers/q1.total_count))/((q2.total_bad_payers+q3.total_bad_payers+q
4.total_bad_payers)/(q2.total_count+q3.total_count+q4.total_count)))
            WHEN q.age_quartile = 2 THEN (((q2.total_bad_payers / q2.total_count))
/ ((q1.total_bad_payers + q3.total_bad_payers + q4.total_bad_payers) /
(q1.total_count + q3.total_count +q4.total_count)))
            WHEN q.age_quartile = 3 THEN (((q3.total_bad_payers / q3.total_count))
/ ((q1.total_bad_payers + q2.total_bad_payers + q4.total_bad_payers) /
(q1.total_count + q2.total_count +q4.total_count)))
            WHEN q.age_quartile = 4 THEN (((q4.total_bad_payers / q4.total_count))
/ ((q1.total_bad_payers + q2.total_bad_payers + q3.total_bad_payers) /
(q1.total_count + q2.total_count +q3.total_count)))

        END AS riesgo_relativo
    FROM quartile_risk q
    JOIN quartile_ranges r ON q.age_quartile = r.age_quartile
    LEFT JOIN quartile_risk q1 ON q1.age_quartile = 1
    LEFT JOIN quartile_risk q2 ON q2.age_quartile = 2
    LEFT JOIN quartile_risk q3 ON q3.age_quartile = 3
    LEFT JOIN quartile_risk q4 ON q4.age_quartile = 4
)

-- Paso 6: Seleccionar los resultados finales
SELECT
    age_quartile,
    total_count,
    total_bad_payers,
    tasa_bad_payers,
    total_good_payers,
    riesgo_relativo,
    min_age,
    max_age
FROM risk_relative
ORDER BY age_quartile ASC;

-----LAST MONTH SALATY Paso 1: Crear una tabla temporal con los datos base
WITH base_data AS (
    SELECT
        last_month_salary,
        default_flag
    FROM riesgo-relativo-p3.Data_Set.Data_Set_Completo
),

-- Paso 2: Calcular los cuartiles para la variable
quartiles AS (
    SELECT
```

```sql
        last_month_salary,
        default_flag,
        NTILE(4) OVER (ORDER BY last_month_salary) AS last_month_salary_quartile
    FROM base_data
),

-- Paso 3: Calcular el número total de malos y buenos pagadores por cuartil
quartile_risk AS (
    SELECT
        last_month_salary_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
        AVG(default_flag) AS tasa_bad_payers,
        COUNT(*) - SUM(default_flag) AS total_good_payers,
    FROM quartiles
    GROUP BY last_month_salary_quartile
),

-- Paso 4: Obtener el rango de edad (mínimo y máximo) para cada cuartil
quartile_ranges AS (
    SELECT
        last_month_salary_quartile,
        MIN(last_month_salary) AS min_last_month_salary,
        MAX(last_month_salary) AS max_last_month_salary
    FROM quartiles
    GROUP BY last_month_salary_quartile
),

-- Paso 5: Calcular el riesgo relativo usando la nueva fórmula
risk_relative AS (
    SELECT
        q.last_month_salary_quartile,
        q.total_count,
        q.total_bad_payers,
        q.tasa_bad_payers,
        q.total_good_payers,
        r.min_last_month_salary,
        r.max_last_month_salary,
        CASE
            WHEN q.last_month_salary_quartile = 1 THEN
(((q1.total_bad_payers/q1.total_count))/((q2.total_bad_payers+q3.total_bad_payers+q
4.total_bad_payers)/(q2.total_count+q3.total_count+q4.total_count)))
            WHEN q.last_month_salary_quartile = 2 THEN (((q2.total_bad_payers /
q2.total_count)) / ((q1.total_bad_payers + q3.total_bad_payers +
q4.total_bad_payers) / (q1.total_count + q3.total_count +q4.total_count)))
            WHEN q.last_month_salary_quartile = 3 THEN (((q3.total_bad_payers /
q3.total_count)) / ((q1.total_bad_payers + q2.total_bad_payers +
q4.total_bad_payers) / (q1.total_count + q2.total_count +q4.total_count)))
            WHEN q.last_month_salary_quartile = 4 THEN (((q4.total_bad_payers /
q4.total_count)) / ((q1.total_bad_payers + q2.total_bad_payers +
q3.total_bad_payers) / (q1.total_count + q2.total_count +q3.total_count)))

        END AS riesgo_relativo
    FROM quartile_risk q
    JOIN quartile_ranges r ON q.last_month_salary_quartile =
r.last_month_salary_quartile
    LEFT JOIN quartile_risk q1 ON q1.last_month_salary_quartile = 1
    LEFT JOIN quartile_risk q2 ON q2.last_month_salary_quartile = 2
    LEFT JOIN quartile_risk q3 ON q3.last_month_salary_quartile = 3
    LEFT JOIN quartile_risk q4 ON q4.last_month_salary_quartile = 4
)

-- Paso 6: Seleccionar los resultados finales
SELECT
    last_month_salary_quartile,
    total_count,
    total_bad_payers,
    tasa_bad_payers,
    total_good_payers,
    riesgo_relativo,
    min_last_month_salary,
    max_last_month_salary
FROM risk_relative
```

```sql
ORDER BY last_month_salary_quartile ASC;

-----NUMBER DEPENDENTS Paso 1: Crear una tabla temporal con los datos base
WITH base_data AS (
    SELECT
        number_dependents,
        default_flag
    FROM riesgo-relativo-p3.Data_Set.Data_Set_Completo
),

-- Paso 2: Calcular los cuartiles para la variable
quartiles AS (
    SELECT
        number_dependents,
        default_flag,
        NTILE(4) OVER (ORDER BY number_dependents) AS number_dependents_quartile
    FROM base_data
),

-- Paso 3: Calcular el número total de malos y buenos pagadores por cuartil
quartile_risk AS (
    SELECT
        number_dependents_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
        AVG(default_flag) AS tasa_bad_payers,
        COUNT(*) - SUM(default_flag) AS total_good_payers,
    FROM quartiles
    GROUP BY number_dependents_quartile
),

-- Paso 4: Obtener el rango de edad (mínimo y máximo) para cada cuartil
quartile_ranges AS (
    SELECT
        number_dependents_quartile,
        MIN(number_dependents) AS min_number_dependents,
        MAX(number_dependents) AS max_number_dependents
    FROM quartiles
    GROUP BY number_dependents_quartile
),

-- Paso 5: Calcular el riesgo relativo usando la nueva fórmula
risk_relative AS (
    SELECT
        q.number_dependents_quartile,
        q.total_count,
        q.total_bad_payers,
        q.tasa_bad_payers,
        q.total_good_payers,
        r.min_number_dependents,
        r.max_number_dependents,
        CASE
            WHEN q.number_dependents_quartile = 1 THEN
(((q1.total_bad_payers/q1.total_count))/((q2.total_bad_payers+q3.total_bad_payers+q
4.total_bad_payers)/(q2.total_count+q3.total_count+q4.total_count)))
            WHEN q.number_dependents_quartile = 2 THEN (((q2.total_bad_payers /
q2.total_count)) / ((q1.total_bad_payers + q3.total_bad_payers +
q4.total_bad_payers) / (q1.total_count + q3.total_count +q4.total_count)))
            WHEN q.number_dependents_quartile = 3 THEN (((q3.total_bad_payers /
q3.total_count)) / ((q1.total_bad_payers + q2.total_bad_payers +
q4.total_bad_payers) / (q1.total_count + q2.total_count +q4.total_count)))
            WHEN q.number_dependents_quartile = 4 THEN (((q4.total_bad_payers /
q4.total_count)) / ((q1.total_bad_payers + q2.total_bad_payers +
q3.total_bad_payers) / (q1.total_count + q2.total_count +q3.total_count)))

        END AS riesgo_relativo
    FROM quartile_risk q
    JOIN quartile_ranges r ON q.number_dependents_quartile =
r.number_dependents_quartile
    LEFT JOIN quartile_risk q1 ON q1.number_dependents_quartile = 1
    LEFT JOIN quartile_risk q2 ON q2.number_dependents_quartile = 2
    LEFT JOIN quartile_risk q3 ON q3.number_dependents_quartile = 3
    LEFT JOIN quartile_risk q4 ON q4.number_dependents_quartile = 4
```

```
)

-- Paso 6: Seleccionar los resultados finales
SELECT
    number_dependents_quartile,
    total_count,
    total_bad_payers,
    tasa_bad_payers,
    total_good_payers,
    riesgo_relativo,
    min_number_dependents,
    max_number_dependents
FROM risk_relative
ORDER BY number_dependents_quartile ASC;

-----debt_ratio Paso 1: Crear una tabla temporal con los datos base
WITH base_data AS (
    SELECT
        debt_ratio,
        default_flag
    FROM riesgo-relativo-p3.Data_Set.Data_Set_Completo
),

-- Paso 2: Calcular los cuartiles para la variable
quartiles AS (
    SELECT
        debt_ratio,
        default_flag,
        NTILE(4) OVER (ORDER BY debt_ratio) AS debt_ratio_quartile
    FROM base_data
),

-- Paso 3: Calcular el número total de malos y buenos pagadores por cuartil
quartile_risk AS (
    SELECT
        debt_ratio_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
        AVG(default_flag) AS tasa_bad_payers,
        COUNT(*) - SUM(default_flag) AS total_good_payers,
    FROM quartiles
    GROUP BY debt_ratio_quartile
),

-- Paso 4: Obtener el rango de edad (mínimo y máximo) para cada cuartil
quartile_ranges AS (
    SELECT
        debt_ratio_quartile,
        MIN(debt_ratio) AS min_debt_ratio,
        MAX(debt_ratio) AS max_debt_ratio
    FROM quartiles
    GROUP BY debt_ratio_quartile
),

-- Paso 5: Calcular el riesgo relativo usando la nueva fórmula
risk_relative AS (
    SELECT
        q.debt_ratio_quartile,
        q.total_count,
        q.total_bad_payers,
        q.tasa_bad_payers,
        q.total_good_payers,
        r.min_debt_ratio,
        r.max_debt_ratio,
        CASE
            WHEN q.debt_ratio_quartile = 1 THEN
(((q1.total_bad_payers/q1.total_count))/((q2.total_bad_payers+q3.total_bad_payers+q
4.total_bad_payers)/(q2.total_count+q3.total_count+q4.total_count)))
            WHEN q.debt_ratio_quartile = 2 THEN (((q2.total_bad_payers /
q2.total_count)) / ((q1.total_bad_payers + q3.total_bad_payers +
q4.total_bad_payers) / (q1.total_count + q3.total_count +q4.total_count)))
```

```
            WHEN q.debt_ratio_quartile = 3 THEN (((q3.total_bad_payers /
q3.total_count)) / ((q1.total_bad_payers + q2.total_bad_payers +
q4.total_bad_payers) / (q1.total_count + q2.total_count +q4.total_count)))
            WHEN q.debt_ratio_quartile = 4 THEN (((q4.total_bad_payers /
q4.total_count)) / ((q1.total_bad_payers + q2.total_bad_payers +
q3.total_bad_payers) / (q1.total_count + q2.total_count +q3.total_count)))

        END AS riesgo_relativo
    FROM quartile_risk q
    JOIN quartile_ranges r ON q.debt_ratio_quartile = r.debt_ratio_quartile
    LEFT JOIN quartile_risk q1 ON q1.debt_ratio_quartile = 1
    LEFT JOIN quartile_risk q2 ON q2.debt_ratio_quartile = 2
    LEFT JOIN quartile_risk q3 ON q3.debt_ratio_quartile = 3
    LEFT JOIN quartile_risk q4 ON q4.debt_ratio_quartile = 4
)

-- Paso 6: Seleccionar los resultados finales
SELECT
    debt_ratio_quartile,
    total_count,
    total_bad_payers,
    tasa_bad_payers,
    total_good_payers,
    riesgo_relativo,
    min_debt_ratio,
    max_debt_ratio
FROM risk_relative
ORDER BY debt_ratio_quartile ASC;

-----using_lines_not_secured_personal_assets Paso 1: Crear una tabla temporal con
los datos base
WITH base_data AS (
    SELECT
        using_lines_not_secured_personal_assets,
        default_flag
    FROM riesgo-relativo-p3.Data_Set.Data_Set_Completo
),

-- Paso 2: Calcular los cuartiles para la variable
quartiles AS (
    SELECT
        using_lines_not_secured_personal_assets,
        default_flag,
        NTILE(4) OVER (ORDER BY using_lines_not_secured_personal_assets) AS
using_lines_not_secured_personal_assets_quartile
    FROM base_data
),

-- Paso 3: Calcular el número total de malos y buenos pagadores por cuartil
quartile_risk AS (
    SELECT
        using_lines_not_secured_personal_assets_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
        AVG(default_flag) AS tasa_bad_payers,
        COUNT(*) - SUM(default_flag) AS total_good_payers,
    FROM quartiles
    GROUP BY using_lines_not_secured_personal_assets_quartile
),

-- Paso 4: Obtener el rango de edad (mínimo y máximo) para cada cuartil
quartile_ranges AS (
    SELECT
        using_lines_not_secured_personal_assets_quartile,
        MIN(using_lines_not_secured_personal_assets) AS
min_using_lines_not_secured_personal_assets,
        MAX(using_lines_not_secured_personal_assets) AS
max_using_lines_not_secured_personal_assets
    FROM quartiles
    GROUP BY using_lines_not_secured_personal_assets_quartile
),

-- Paso 5: Calcular el riesgo relativo usando la nueva fórmula
```

```sql
risk_relative AS (
    SELECT
        q.using_lines_not_secured_personal_assets_quartile,
        q.total_count,
        q.total_bad_payers,
        q.tasa_bad_payers,
        q.total_good_payers,
        r.min_using_lines_not_secured_personal_assets,
        r.max_using_lines_not_secured_personal_assets,
        CASE
            WHEN q.using_lines_not_secured_personal_assets_quartile = 1 THEN
(((q1.total_bad_payers/q1.total_count))/((q2.total_bad_payers+q3.total_bad_payers+q
4.total_bad_payers)/(q2.total_count+q3.total_count+q4.total_count)))
            WHEN q.using_lines_not_secured_personal_assets_quartile = 2 THEN
(((q2.total_bad_payers / q2.total_count)) / ((q1.total_bad_payers +
q3.total_bad_payers + q4.total_bad_payers) / (q1.total_count + q3.total_count
+q4.total_count)))
            WHEN q.using_lines_not_secured_personal_assets_quartile = 3 THEN
(((q3.total_bad_payers / q3.total_count)) / ((q1.total_bad_payers +
q2.total_bad_payers + q4.total_bad_payers) / (q1.total_count + q2.total_count
+q4.total_count)))
            WHEN q.using_lines_not_secured_personal_assets_quartile = 4 THEN
(((q4.total_bad_payers / q4.total_count)) / ((q1.total_bad_payers +
q2.total_bad_payers + q3.total_bad_payers) / (q1.total_count + q2.total_count
+q3.total_count)))

        END AS riesgo_relativo
    FROM quartile_risk q
    JOIN quartile_ranges r ON q.using_lines_not_secured_personal_assets_quartile =
r.using_lines_not_secured_personal_assets_quartile
    LEFT JOIN quartile_risk q1 ON
q1.using_lines_not_secured_personal_assets_quartile = 1
    LEFT JOIN quartile_risk q2 ON
q2.using_lines_not_secured_personal_assets_quartile = 2
    LEFT JOIN quartile_risk q3 ON
q3.using_lines_not_secured_personal_assets_quartile = 3
    LEFT JOIN quartile_risk q4 ON
q4.using_lines_not_secured_personal_assets_quartile = 4
)

-- Paso 6: Seleccionar los resultados finales
SELECT
    using_lines_not_secured_personal_assets_quartile,
    total_count,
    total_bad_payers,
    tasa_bad_payers,
    total_good_payers,
    riesgo_relativo,
    min_using_lines_not_secured_personal_assets,
    max_using_lines_not_secured_personal_assets
FROM risk_relative
ORDER BY using_lines_not_secured_personal_assets_quartile ASC;

-----delay_30_59_90 Paso 1: Crear una tabla temporal con los datos base
WITH base_data AS (
    SELECT
        delay_30_59_90,
        default_flag
    FROM riesgo-relativo-p3.Data_Set.Data_Set_Completo
    WHERE default_flag = 1
),

-- Paso 2: Calcular los cuartiles para la variable
quartiles AS (
    SELECT
        delay_30_59_90,
        default_flag,
        NTILE(4) OVER (ORDER BY delay_30_59_90) AS delay_30_59_90_quartile
    FROM base_data
),

-- Paso 3: Calcular el número total de malos y buenos pagadores por cuartil
quartile_risk AS (
    SELECT
        delay_30_59_90_quartile,
        COUNT(*) AS total_count,
        SUM(default_flag) AS total_bad_payers,
        AVG(default_flag) AS tasa_bad_payers,
        COUNT(*) - SUM(default_flag) AS total_good_payers,
    FROM quartiles
    GROUP BY delay_30_59_90_quartile
),

-- Paso 4: Obtener el rango de edad (mínimo y máximo) para cada cuartil
quartile_ranges AS (
    SELECT
        delay_30_59_90_quartile,
        MIN(delay_30_59_90) AS min_delay_30_59_90,
        MAX(delay_30_59_90) AS max_delay_30_59_90
    FROM quartiles
    GROUP BY delay_30_59_90_quartile
),

-- Paso 5: Calcular el riesgo relativo usando la nueva fórmula
risk_relative AS (
    SELECT
        q.delay_30_59_90_quartile,
        q.total_count,
        q.total_bad_payers,
        q.tasa_bad_payers,
        q.total_good_payers,
        r.min_delay_30_59_90,
        r.max_delay_30_59_90,
        CASE
            WHEN q.delay_30_59_90_quartile = 1 THEN
(((q1.total_bad_payers/q1.total_count))/((q2.total_bad_payers+q3.total_bad_payers+q
4.total_bad_payers)/(q2.total_count+q3.total_count+q4.total_count)))
            WHEN q.delay_30_59_90_quartile = 2 THEN (((q2.total_bad_payers /
q2.total_count)) / ((q1.total_bad_payers + q3.total_bad_payers +
q4.total_bad_payers) / (q1.total_count + q3.total_count +q4.total_count)))
            WHEN q.delay_30_59_90_quartile = 3 THEN (((q3.total_bad_payers /
q3.total_count)) / ((q1.total_bad_payers + q2.total_bad_payers +
q4.total_bad_payers) / (q1.total_count + q2.total_count +q4.total_count)))
            WHEN q.delay_30_59_90_quartile = 4 THEN (((q4.total_bad_payers /
q4.total_count)) / ((q1.total_bad_payers + q2.total_bad_payers +
q3.total_bad_payers) / (q1.total_count + q2.total_count +q3.total_count)))

        END AS riesgo_relativo
    FROM quartile_risk q
    JOIN quartile_ranges r ON q.delay_30_59_90_quartile = r.delay_30_59_90_quartile
    LEFT JOIN quartile_risk q1 ON q1.delay_30_59_90_quartile = 1
    LEFT JOIN quartile_risk q2 ON q2.delay_30_59_90_quartile = 2
    LEFT JOIN quartile_risk q3 ON q3.delay_30_59_90_quartile = 3
    LEFT JOIN quartile_risk q4 ON q4.delay_30_59_90_quartile = 4
)

-- Paso 6: Seleccionar los resultados finales
SELECT
    delay_30_59_90_quartile,
    total_count,
    total_bad_payers,
    tasa_bad_payers,
    total_good_payers,
    riesgo_relativo,
    min_delay_30_59_90,
    max_delay_30_59_90
FROM risk_relative
ORDER BY delay_30_59_90_quartile ASC;
```

## 2.1 Hito 2

1.- Calcular variables dummys y Score de Riesgo

```
WITH dummies_table AS (
  SELECT
    *,
    -- Crear variable dummy para age
    CASE
      WHEN age >= 21 AND age <= 40 THEN 1
      ELSE 0
    END AS age_dummy,

    -- Crear variable dummy para last_month_salary
    CASE
      WHEN last_month_salary >= 0 AND last_month_salary <= 3878 THEN 1
      WHEN last_month_salary >= 5000 AND last_month_salary <= 7424 THEN 1
      ELSE 0
    END AS last_month_salary_dummy,

    -- Crear variable dummy para delay_30_59_90
    CASE
      WHEN delay_30_59_90 = 0 THEN 0
      ELSE 1
    END AS delay_30_59_90_dummy,

    -- Crear variable dummy para debt_ratio
    CASE
      WHEN debt_ratio >= 0.366558991 AND debt_ratio <= 307001.0 THEN 1
      ELSE 0
    END AS debt_ratio_dummy,

    -- Crear variable dummy para using_lines_not_secured_personal_assets
    CASE
      WHEN using_lines_not_secured_personal_assets >= 0.546718373 THEN 1
      ELSE 0
    END AS using_lines_not_secured_personal_assets_dummy
  FROM
    `riesgo-relativo-p3.Data_Set.Data_Set_Completo`
)

SELECT
  *,
  -- Calcular el score de riesgo basado en las variables dummies
  (age_dummy * 0.2 +
   last_month_salary_dummy * 0.2 +
   delay_30_59_90_dummy * 0.3 +
   debt_ratio_dummy * 0.2 +
   using_lines_not_secured_personal_assets_dummy * 0.1) AS risk_score,

  -- Clasificación basada en el score de riesgo
  CASE
    WHEN (age_dummy * 0.2 +
          last_month_salary_dummy * 0.2 +
          delay_30_59_90_dummy * 0.3 +
          debt_ratio_dummy * 0.2 +
          using_lines_not_secured_personal_assets_dummy * 0.1) <= 0.5 THEN 'Buen
pagador'
    ELSE 'Mal pagador'
  END AS payment_classification

FROM
  dummies_table
```

## 2.- Matriz de confusión

```
SELECT
  SUM(CASE
        WHEN default_flag = 0 AND payment_classification = 'Buen pagador'
        THEN 1
        ELSE 0
      END) AS TP, -- Verdaderos Positivos
  SUM(CASE
        WHEN default_flag = 0 AND payment_classification = 'Mal pagador'
        THEN 1
        ELSE 0
      END) AS FN, -- Falsos Negativos
  SUM(CASE
        WHEN default_flag = 1 AND payment_classification = 'Buen pagador'
        THEN 1
        ELSE 0
      END) AS FP, -- Falsos Positivos
  SUM(CASE
        WHEN default_flag = 1 AND payment_classification = 'Mal pagador'
        THEN 1
        ELSE 0
      END) AS TN -- Verdaderos Negativos
FROM `riesgo-relativo-p3.Data_Set.Prueba_jueves`
```

## 3.- Métricas de precisión

```
WITH confusion_matrix AS (
  SELECT
  SUM(CASE
        WHEN default_flag = 0 AND payment_classification = 'Buen pagador'
        THEN 1
        ELSE 0
      END) AS true_positive, -- Verdaderos Positivos
  SUM(CASE
        WHEN default_flag = 0 AND payment_classification = 'Mal pagador'
        THEN 1
        ELSE 0
      END) AS false_negative, -- Falsos Negativos
  SUM(CASE
        WHEN default_flag = 1 AND payment_classification = 'Buen pagador'
        THEN 1
        ELSE 0
      END) AS false_positive, -- Falsos Positivos
  SUM(CASE
        WHEN default_flag = 1 AND payment_classification = 'Mal pagador'
        THEN 1
        ELSE 0
      END) AS true_negative -- Verdaderos Negativos
FROM `riesgo-relativo-p3.Data_Set.Prueba_jueves`
)

SELECT
  true_positive,
  false_positive,
  false_negative,
  true_negative,
  SAFE_DIVIDE(true_positive + true_negative, true_positive + false_positive +
false_negative + true_negative) AS accuracy,
  SAFE_DIVIDE(true_positive, true_positive + false_positive) AS precision,
  SAFE_DIVIDE(true_positive, true_positive + false_negative) AS recall,
  SAFE_DIVIDE(
    2 * (SAFE_DIVIDE(true_positive, true_positive + false_positive) *
SAFE_DIVIDE(true_positive, true_positive + false_negative)),
    SAFE_DIVIDE(true_positive, true_positive + false_positive) +
SAFE_DIVIDE(true_positive, true_positive + false_negative)
```

## 3.1 Hito 3