# datos_sinteticos

July 11, 2025

## 1 CREACIÓN DE DATOS SINTÉTICOS

[1]:
```
# INSTALACIÓN DE LIBRERÍAS
%pip install pandas
%pip install numpy
```

```
Requirement already satisfied: pandas in
c:\users\carolina\documents\proyectos_programacion\predestu\.venv\lib\site-
packages (2.3.1)
Requirement already satisfied: numpy>=1.26.0 in
c:\users\carolina\documents\proyectos_programacion\predestu\.venv\lib\site-
packages (from pandas) (2.3.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\carolina\documents\proyectos_programacion\predestu\.venv\lib\site-
packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
c:\users\carolina\documents\proyectos_programacion\predestu\.venv\lib\site-
packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\carolina\documents\proyectos_programacion\predestu\.venv\lib\site-
packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in
c:\users\carolina\documents\proyectos_programacion\predestu\.venv\lib\site-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: numpy in
c:\users\carolina\documents\proyectos_programacion\predestu\.venv\lib\site-
packages (2.3.1)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
# IMPORTACIÓN DE LIBRERÍAS
import pandas as pd
import numpy as np
import random
```

[3]:
```
# CONFIGURACIÓN INICIAL
np.random.seed(42)
random.seed(42)
```

[30]:
```
# CARGAR DATASETS
df_escuelas = pd.read_csv('../fuentes_datos/datasets_finales/CSV/escuela.csv',␣
 ↪delimiter=';')
df_semestres = pd.read_csv('../fuentes_datos/datasets_finales/CSV/semestre.
 ↪csv', delimiter=';')
df_planes = pd.read_csv('../fuentes_datos/datasets_finales/CSV/plan.csv',␣
 ↪delimiter=';')
df_estudiantes = pd.read_csv('../fuentes_datos/datasets_finales/CSV/estudiante.
 ↪csv', delimiter=';')
df_cursos = pd.read_csv('../fuentes_datos/datasets_finales/CSV/curso.csv',␣
 ↪delimiter=';')
df_reprobaciones = pd.read_csv('../fuentes_datos/datasets_finales/CSV/
 ↪reprobacion.csv', delimiter=';')
df_tutorias = pd.read_csv('../fuentes_datos/datasets_finales/CSV/tutoria.csv',␣
 ↪delimiter=';')
df_curso_tomado = pd.read_csv('../fuentes_datos/datasets_finales/CSV/
 ↪curso_tomado.csv', delimiter=';')
```

[31]:
```
# PARAMETROS BASE
escuelas = df_escuelas.set_index('id_escuela')['nombre'].to_dict()
#planes = df_planes.set_index('id_escuela')['anio_inicio'].to_dict()
semestres = df_semestres.set_index('id_semestre')['nombre'].to_dict()
#semestres = df_semestres['id_semestre'].astype(str).tolist()
```

[32]:
```
print("DICCIONARIO ESCUELAS:")
print(escuelas)
#print("\nDICCIONARIO PLANES:")
#print(planes)
print("\nLISTA SEMESTRES:")
print(semestres)
```

```
DICCIONARIO ESCUELAS:
{1: 'Ingeniería de Sistemas', 2: 'Ingeniería de Software', 3: 'Ciencias de la
Computación'}

LISTA SEMESTRES:
{1: 20190, 2: 20191, 3: 20192, 4: 20200, 5: 20201, 6: 20202, 7: 20210, 8: 20211,
9: 20212, 10: 20220, 11: 20221, 12: 20222, 13: 20230, 14: 20231, 15: 20232, 16:
20240, 17: 20241, 18: 20242, 19: 20250, 20: 20251, 21: 20190, 22: 20191, 23:
20192, 24: 20200, 25: 20201, 26: 20202, 27: 20210, 28: 20211, 29: 20212, 30:
20220, 31: 20221, 32: 20222, 33: 20230, 34: 20231, 35: 20232, 36: 20240, 37:
20241, 38: 20242, 39: 20250, 40: 20251, 41: 20241, 42: 20242, 43: 20250, 44:
20251}
```

[33]:
```
# SEMESTRE POR ESCUELA
```

```
semestres_sistemas = {7: 20210, 8: 20211, 9: 20212, 10: 20220, 11: 20221, 12:␣
 ↪20222, 13: 20230, 14: 20231, 15: 20232, 16: 20240, 17: 20241, 18: 20242, 19:␣
 ↪20250}
semestres_software = {27: 20210, 28: 20211, 29: 20212, 30: 20220, 31: 20221, 32:
 ↪ 20222, 33: 20230, 34: 20231, 35: 20232, 36: 20240, 37: 20241, 38: 20242, 39:
 ↪ 20250, 40: 20251}
semestres_ccomputacion = {41: 20241, 42: 20242, 43: 20250, 44: 20251}
```

[39]:
```
# ULTIMO ID_CT DEL DATAFRAME DE CURSOS TOMADOS
ultimo_id_ct1 = df_curso_tomado['id_ct'].max() if not df_curso_tomado.empty␣
 ↪else 0
print(f"\nÚltimo ID_CT en df_curso_tomado: {ultimo_id_ct1}")
```

```
Último ID_CT en df_curso_tomado: 268644
```

[44]:
```
# CREACIÓN DE CALIFICACIONES SINTÉTICAS - SISTEMAS
# Simular cursos tomados por estudiante (2021-2025)
cursos_tomados = []
indice = ultimo_id_ct1 + 1  # Comenzar desde el último ID_CT

for _, est in df_estudiantes.iterrows():
    for id_semestre, nombre_semestre in semestres_sistemas.items():
        # Extrae el año del nombre del semestre (ejemplo: '2021-1'[:4] ->␣
 ↪'2021')
        if int(str(nombre_semestre)[:4]) >= est.anio_ingreso:
            for _ in range(random.randint(3, 7)):
                curso = df_cursos.sample(1).iloc[0]  # elige curso al azar
                id_curso = curso["id_curso"]
                id_ciclo = curso["id_ciclo"]
                t_aprob = curso["t_aprobacion"]  # entre 0 y 1

                # Simular si aprueba o no según tasa
                aprueba = random.random() < t_aprob

                # Nota entre 11-20 si aprueba, entre 0-10 si desaprueba
                if aprueba:
                    nota = np.clip(np.random.normal(14.5, 2), 11, 20)
                else:
                    nota = np.clip(np.random.normal(9, 2.5), 0, 10)

                indice += 1

                cursos_tomados.append({
                    "id_ct": indice,  # Asignar ID_CT secuencial
                    "id_estudiante": est.id_estudiante,
                    "id_curso": id_curso,
```

```
                    "id_semestre": id_semestre,  # Usar la clave del diccionario
                    "nota": int(round(nota)),
                    "id_ciclo": id_ciclo
                })
```

[45]:
```
# GUARDAR EN UN DATAFRAME
df_cursos_tomados_sistemas = pd.DataFrame(cursos_tomados)
```

[46]:
```
print("DATASET CURSOS_TOMADOS 2021-2025:")
print(df_cursos_tomados_sistemas.tail(5))  # Muestra los últimos 5 registros
```

```
DATASET CURSOS_TOMADOS 2021-2025:
         id_ct  id_estudiante  id_curso  id_semestre  nota  id_ciclo
105341  373987       18200108    201003           18    16         2
105342  373988       18200108   20W0902           19    13        63
105343  373989       18200108   2020205           19    16        45
105344  373990       18200108  202W0301           19    15        67
105345  373991       18200108   20W0E12           19    14        54
```

[47]:
```
# ULTIMO ID_CT DEL DATAFRAME DE CURSOS TOMADOS
ultimo_id_ct2 = df_cursos_tomados_sistemas['id_ct'].max() if not␣
 ↪df_cursos_tomados_sistemas.empty else 0
print(f"\nÚltimo ID_CT en df_curso_tomado: {ultimo_id_ct2}")
```

```
Último ID_CT en df_curso_tomado: 373991
```

[48]:
```
# CREACIÓN DE CALIFICACIONES SINTÉTICAS - SOFTWARE
# Simular cursos tomados por estudiante (2021-2025)
cursos_tomados2 = []
indice = ultimo_id_ct2 + 1  # Comenzar desde el último ID_CT

for _, est in df_estudiantes.iterrows():
    for id_semestre, nombre_semestre in semestres_software.items():
        # Extrae el año del nombre del semestre (ejemplo: '2021-1'[:4] ->␣
 ↪'2021')
        if int(str(nombre_semestre)[:4]) >= est.anio_ingreso:
            for _ in range(random.randint(3, 7)):
                curso = df_cursos.sample(1).iloc[0]  # elige curso al azar
                id_curso = curso["id_curso"]
                id_ciclo = curso["id_ciclo"]
                t_aprob = curso["t_aprobacion"]  # entre 0 y 1

                # Simular si aprueba o no según tasa
                aprueba = random.random() < t_aprob

                # Nota entre 11-20 si aprueba, entre 0-10 si desaprueba
                if aprueba:
```

```
                    nota = np.clip(np.random.normal(14.5, 2), 11, 20)
                else:
                    nota = np.clip(np.random.normal(9, 2.5), 0, 10)

                indice += 1

                cursos_tomados2.append({
                    "id_ct": indice,  # Asignar ID_CT secuencial
                    "id_estudiante": est.id_estudiante,
                    "id_curso": id_curso,
                    "id_semestre": id_semestre,  # Usar la clave del diccionario
                    "nota": int(round(nota)),
                    "id_ciclo": id_ciclo
                })
```

[49]:
```
# GUARDAR EN UN DATAFRAME
df_cursos_tomados_software = pd.DataFrame(cursos_tomados2)
```

[50]:
```
print("DATASET CURSOS_TOMADOS 2021-2025:")
print(df_cursos_tomados_software.tail(5))  # Muestra los últimos 5 registros
```

```
DATASET CURSOS_TOMADOS 2021-2025:
         id_ct  id_estudiante   id_curso  id_semestre  nota  id_ciclo
113143  487136       18200108  202SW0E09           40     1        75
113144  487137       18200108    2010202           40    15        14
113145  487138       18200108    2020103           40    13        44
113146  487139       18200108  202SW0203           40    17        76
113147  487140       18200108    201203M           40    14         1
```

[51]:
```
# ULTIMO ID_CT DEL DATAFRAME DE CURSOS TOMADOS
ultimo_id_ct3 = df_cursos_tomados_software['id_ct'].max() if not␣
 ↪df_cursos_tomados_software.empty else 0
print(f"\nÚltimo ID_CT en df_curso_tomado: {ultimo_id_ct3}")
```

```
Último ID_CT en df_curso_tomado: 487140
```

[56]:
```
# CREACIÓN DE CALIFICACIONES SINTÉTICAS - CCOMPUTACIÓN
# Simular cursos tomados por estudiante (2021-2025)
cursos_tomados3 = []
indice = ultimo_id_ct3 + 1  # Comenzar desde el último ID_CT

for _, est in df_estudiantes.iterrows():
    for id_semestre, nombre_semestre in semestres_software.items():
        # Extrae el año del nombre del semestre (ejemplo: '2021-1'[:4] ->␣
 ↪'2021')
        if int(str(nombre_semestre)[:4]) >= est.anio_ingreso:
            for _ in range(random.randint(3, 7)):
```

```
                curso = df_cursos.sample(1).iloc[0]  # elige curso al azar
                id_curso = curso["id_curso"]
                id_ciclo = curso["id_ciclo"]
                t_aprob = curso["t_aprobacion"]  # entre 0 y 1

                # Simular si aprueba o no según tasa
                aprueba = random.random() < t_aprob

                # Nota entre 11-20 si aprueba, entre 0-10 si desaprueba
                if aprueba:
                    nota = np.clip(np.random.normal(14.5, 2), 11, 20)
                else:
                    nota = np.clip(np.random.normal(9, 2.5), 0, 10)

                indice += 1

                cursos_tomados3.append({
                    "id_ct": indice,  # Asignar ID_CT secuencial
                    "id_estudiante": est.id_estudiante,
                    "id_curso": id_curso,
                    "id_semestre": id_semestre,  # Usar la clave del diccionario
                    "nota": int(round(nota)),
                    "id_ciclo": id_ciclo
                })
```

[57]:
```
# GUARDAR EN UN DATAFRAME
df_cursos_tomados_ccomputacion = pd.DataFrame(cursos_tomados3)
```

[58]:
```
print("DATASET CURSOS_TOMADOS 2021-2025:")
print(df_cursos_tomados_ccomputacion.tail(5))  # Muestra los últimos 5 registros
```

```
DATASET CURSOS_TOMADOS 2021-2025:
         id_ct  id_estudiante   id_curso  id_semestre  nota  id_ciclo
113827  600969       18200108    2010802           40     8        20
113828  600970       18200108    2020898           40    18        43
113829  600971       18200108  203230301           40    16        87
113830  600972       18200108     208010           40    12         9
113831  600973       18200108  203230403           40    10        88
```

[59]:
```
# UNIR DATASETS DE CURSOS TOMADOS (HISTÓRICO Y NUEVO)
df_cursos_tomados_union1 = pd.concat([df_curso_tomado,␣
 ↪df_cursos_tomados_sistemas], ignore_index=True)
df_cursos_tomados_union2 = pd.concat([df_cursos_tomados_union1,␣
 ↪df_cursos_tomados_software], ignore_index=True)
df_cursos_tomados_union3 = pd.concat([df_cursos_tomados_union2,␣
 ↪df_cursos_tomados_ccomputacion], ignore_index=True)
```

[60]:
```
# GUARDAR EL DATASET NUEVO DE CURSOS_TOMADOS (2019-2025))
df_cursos_tomados_union3.to_csv('../fuentes_datos/datasets_finales/CSV/
 ↪curso_tomado_completo.csv', index=False, sep=';')
```

[61]:
```
# CREACIÓN DE REPETICIONES SINTÉTICAS
# Cada estudiante puede tener repitencias si su nota fue < 11
repitencias = df_cursos_tomados_union3[df_cursos_tomados_union3["nota"] < 11]
repitencias = repitencias.groupby(["id_estudiante", "id_curso", "id_semestre"]).
 ↪size().reset_index(name="n_reprobaciones")
```

[62]:
```
# GUARDAR EN UN DATAFRAME
df_reprobaciones2 = pd.DataFrame(repitencias)
```

[63]:
```
print("DATASET REPROBACIONES 2021-2025:")
print(df_reprobaciones2.tail(5))  # Muestra los últimos 5 registros
```

```
DATASET REPROBACIONES 2021-2025:
      id_estudiante id_curso  id_semestre  n_reprobaciones
88211        962909   206008            2                1
88212        973021   206008            2                1
88213        973021   206008            3                1
88214        973021   206010            2                1
88215        997985   201105            2                1
```

[64]:
```
# UNIR DATASETS DE REPROBACIONES (HISTÓRICO Y NUEVO)
df_reprobaciones3 = pd.concat([df_reprobaciones, df_reprobaciones2],␣
 ↪ignore_index=True)
```

[65]:
```
# Sumar reprobaciones por estudiante y curso, conservando el último semestre␣
 ↪donde reprobó
df_reprobaciones4 = df_reprobaciones3.groupby(["id_estudiante", "id_curso"],␣
 ↪as_index=False).agg({
    "n_reprobaciones": "sum",
    "id_semestre": "max"  # Último semestre con reprobación
})
```

[66]:
```
# GUARDAR EL DATASET NUEVO DE REPROBACIONES (2019-2025))
df_reprobaciones4.to_csv('../fuentes_datos/datasets_finales/CSV/
 ↪reprobacion_completo.csv', index=False, sep=';')
```

[67]:
```
# IDENTIFICACIÓN DE NUM_RES_AUTORIZA DEL DATASET DE TUTORIAS
df_tutorias['num_res_autoriza'].unique().tolist()  # Muestra los valores únicos␣
 ↪de la columna num_res_autoriza
```

[67]:
```
['RD N°138-D-FISI-19                    ',
 'RD 125-D-FISI-19                      ',
 'RD125-D-FISI-19                       ',
 'RD 125-D-FISI19                       ',
```

```
 'RD 138-D-FISI-19                    ',
 'RD 405-D-FISI-2019                  ',
 'RD 164-D-FISI-19                    ',
 'RD 174-D-FISI-19                    ',
 'RD N°126-D-FISI-19                  ',
 'RD 125-D-FISI-2019                  ',
 'RD 138-D-FISI-2019                  ',
 'RD N° 00126-D-FISI                  ',
 'RD N°174-D-FISI-19                  ',
 'RD N 126-D-FISI-19                  ',
 'RD 125-D-19                         ',
 'RD 126-D-FISI-19                    ',
 'RD 126-D-FISI-2019                  ',
 'RD 126-D-FISI19                     ',
 'RD126-D-FISI-19                     ',
 'RD 139-D-FISI-19                    ',
 'RD 139-D-FISI-2019                  ',
 'RD139-D-FISI-19                     ',
 'RD 165-D-FISI-19                    ',
 'RD 175-D-FISI-19                    ',
 'RD 406-D-FISI-2019                  ',
 'RD 405-D-FISI-19                    ',
 'RD 405-FISI-2019                    ',
 'RD  716-D-FISI-19                   ',
 'RD 00744-D-FISI-19                  ',
 'RD 405 D-FISI-2019                  ',
 'RD 716-D-FISI-19                    ',
 'RD 406 D-FISI-2019                  ',
 'RD 00406-D-FISI                     ',
 'RD 406-D-FISI-19                    ',
 'RD 406-FISI-2019                    ',
 'RD 406-D-FISI-2016                  ',
 '716-D-FISI-2019                     ',
 'RD 107-D-FISI-2020                  ',
 'RD 107-D-FISI2020                   ',
 'RD 000439-2020-D                    ',
 'RD 00439-2020-D                     ',
 'RD 000451-2020-D                    ',
 'RD 00439                            ',
 'RD 000445-2020-D                    ',
 'RD 000539-2020-D-                   ',
 nan]
```

[68]:
```
# QUITAR ESPACIOS EN BLANCO DE NUM_RES_AUTORIZA
df_tutorias['num_res_autoriza'] = df_tutorias['num_res_autoriza'].str.strip()
```

[69]:
```
# IDENTIFICACIÓN DE NUM_RES_AUTORIZA DEL DATASET DE TUTORIAS
df_tutorias['num_res_autoriza'].unique().tolist()  # Muestra los valores únicos␣
↪de la columna num_res_autoriza
```

[69]:
```
['RD N°138-D-FISI-19',
 'RD 125-D-FISI-19',
 'RD125-D-FISI-19',
 'RD 125-D-FISI19',
 'RD 138-D-FISI-19',
 'RD 405-D-FISI-2019',
 'RD 164-D-FISI-19',
 'RD 174-D-FISI-19',
 'RD N°126-D-FISI-19',
 'RD 125-D-FISI-2019',
 'RD 138-D-FISI-2019',
 'RD N° 00126-D-FISI',
 'RD N°174-D-FISI-19',
 'RD N 126-D-FISI-19',
 'RD 125-D-19',
 'RD 126-D-FISI-19',
 'RD 126-D-FISI-2019',
 'RD 126-D-FISI19',
 'RD126-D-FISI-19',
 'RD 139-D-FISI-19',
 'RD 139-D-FISI-2019',
 'RD139-D-FISI-19',
 'RD 165-D-FISI-19',
 'RD 175-D-FISI-19',
 'RD 406-D-FISI-2019',
 'RD 405-D-FISI-19',
 'RD 405-FISI-2019',
 'RD  716-D-FISI-19',
 'RD 00744-D-FISI-19',
 'RD 405 D-FISI-2019',
 'RD 716-D-FISI-19',
 'RD 406 D-FISI-2019',
 'RD 00406-D-FISI',
 'RD 406-D-FISI-19',
 'RD 406-FISI-2019',
 'RD 406-D-FISI-2016',
 '716-D-FISI-2019',
 'RD 107-D-FISI-2020',
 'RD 107-D-FISI2020',
 'RD 000439-2020-D',
 'RD 00439-2020-D',
 'RD 000451-2020-D',
 'RD 00439',
```

```
 'RD 000445-2020-D',
 'RD 000539-2020-D-',
 nan]
```

[70]:
```
# REEMPLAZAR VALORES DE NUM_RES_AUTORIZA
df_tutorias['num_res_autoriza'] = df_tutorias['num_res_autoriza'].replace({
    'RD N°138-D-FISI-19': 'RD 138-D-FISI-19',
    'RD125-D-FISI-19': 'RD 125-D-FISI-19',
    'RD 125-D-FISI19': 'RD 125-D-FISI-19',
    'RD 405-D-FISI-2019': 'RD 405-D-FISI-19',
    'RD N°126-D-FISI-19': 'RD 126-D-FISI-19',
    'RD 125-D-FISI-2019': 'RD 125-D-FISI-19',
    'RD 138-D-FISI-2019': 'RD 138-D-FISI-19',
    'RD N° 00126-D-FISI': 'RD 126-D-FISI-19',
    'RD N°174-D-FISI-19': 'RD 174-D-FISI-19',
    'RD N 126-D-FISI-19': 'RD 126-D-FISI-19',
    'RD 125-D-19': 'RD 125-D-FISI-19',
    'RD 126-D-FISI-2019': 'RD 126-D-FISI-19',
    'RD 126-D-FISI19': 'RD 126-D-FISI-19',
    'RD126-D-FISI-19': 'RD 126-D-FISI-19',
    'RD 139-D-FISI-2019': 'RD 139-D-FISI-19',
    'RD139-D-FISI-19': 'RD 139-D-FISI-19',
    'RD 406-D-FISI-2019': 'RD 406-D-FISI-19',
    'RD 000451-2020-D': 'RD 451-D-FISI-20',
    'RD 00439': 'RD 439-D-FISI-20',
    'RD 000445-2020-D': 'RD 445-D-FISI-20',
    'RD 000445-2020-D': 'RD 445-D-FISI-20'
})
```

[71]:
```
# IDENTIFICACIÓN DE NUM_RES_AUTORIZA DEL DATASET DE TUTORIAS
df_tutorias['num_res_autoriza'].unique().tolist()  # Muestra los valores únicos␣
 ↪de la columna num_res_autoriza
```

[71]:
```
['RD 138-D-FISI-19',
 'RD 125-D-FISI-19',
 'RD 405-D-FISI-19',
 'RD 164-D-FISI-19',
 'RD 174-D-FISI-19',
 'RD 126-D-FISI-19',
 'RD 139-D-FISI-19',
 'RD 165-D-FISI-19',
 'RD 175-D-FISI-19',
 'RD 406-D-FISI-19',
 'RD 405-FISI-2019',
 'RD  716-D-FISI-19',
 'RD 00744-D-FISI-19',
 'RD 405 D-FISI-2019',
```

```
 'RD 716-D-FISI-19',
 'RD 406 D-FISI-2019',
 'RD 00406-D-FISI',
 'RD 406-FISI-2019',
 'RD 406-D-FISI-2016',
 '716-D-FISI-2019',
 'RD 107-D-FISI-2020',
 'RD 107-D-FISI2020',
 'RD 000439-2020-D',
 'RD 00439-2020-D',
 'RD 451-D-FISI-20',
 'RD 439-D-FISI-20',
 'RD 445-D-FISI-20',
 'RD 000539-2020-D-',
 nan]
```

[72]:
```
# REEMPLAZAR VALORES DE NUM_RES_AUTORIZA
df_tutorias['num_res_autoriza'] = df_tutorias['num_res_autoriza'].replace({
    'RD 405-FISI-2019': 'RD 405-D-FISI-19',
    'RD  716-D-FISI-19': 'RD 716-D-FISI-19',
    'RD 00744-D-FISI-19': 'RD 744-D-FISI-19',
    'RD 405 D-FISI-2019': 'RD 405-D-FISI-19',
    'RD 406 D-FISI-2019': 'RD 406-D-FISI-19',
    'RD 00406-D-FISI': 'RD 406-D-FISI-19',
    'RD 406-FISI-2019': 'RD 406-D-FISI-19',
    'RD 406-D-FISI-2016': 'RD 406-D-FISI-19',
    '716-D-FISI-2019': 'RD 716-D-FISI-19',
    'RD 107-D-FISI-2020': 'RD 107-D-FISI-20',
    'RD 107-D-FISI2020': 'RD 107-D-FISI-20',
    'RD 000439-2020-D': 'RD 439-D-FISI-20',
    'RD 00439-2020-D': 'RD 439-D-FISI-20',
    'RD 000539-2020-D-': 'RD 539-D-FISI-20'
})
```

[73]:
```
# IDENTIFICACIÓN DE NUM_RES_AUTORIZA DEL DATASET DE TUTORIAS
df_tutorias['num_res_autoriza'].unique().tolist()  # Muestra los valores únicos␣
 ↪de la columna num_res_autoriza
```

[73]:
```
['RD 138-D-FISI-19',
 'RD 125-D-FISI-19',
 'RD 405-D-FISI-19',
 'RD 164-D-FISI-19',
 'RD 174-D-FISI-19',
 'RD 126-D-FISI-19',
 'RD 139-D-FISI-19',
 'RD 165-D-FISI-19',
 'RD 175-D-FISI-19',
```

```
 'RD 406-D-FISI-19',
 'RD 716-D-FISI-19',
 'RD 744-D-FISI-19',
 'RD 107-D-FISI-20',
 'RD 439-D-FISI-20',
 'RD 451-D-FISI-20',
 'RD 445-D-FISI-20',
 'RD 539-D-FISI-20',
 nan]
```

[74]:
```
# Función auxiliar para generar un num_res_autoriza sintético pero realista
def generar_resolucion(id_semestre):
    anio = str(id_semestre)[:4][-2:]  # Extrae los dos últimos dígitos del año
    numero = random.randint(100, 800)
    return f"RD {numero}-D-FISI-{anio}"
```

[80]:
```
# ULTIMO ID_CT DEL DATAFRAME DE CURSOS TOMADOS
ultimo_id_tutoria = df_tutorias['id_tutoria'].max() if not df_tutorias.empty␣
 ↪else 0
print(f"\nÚltimo ID_TUTORIAS en df_tutorias: {ultimo_id_tutoria}")
```

```
Último ID_TUTORIAS en df_tutorias: 1218.0
```

[81]:
```
# CREACIÓN DE TUTORÍAS SINTÉTICAS
tutorias = []
indice = ultimo_id_tutoria + 1  # Comenzar desde el último ID_TUTORIA

for _, row in df_reprobaciones4.iterrows():
    indice += 1  # Incrementar el índice para cada tutoría nueva
    if random.random() < 0.7:  # 70% probabilidad de tener tutoría
        tutorias.append({
            "id_tutoria": indice,  # Asignar ID_TUTORIA secuencial
            "id_estudiante": row.id_estudiante,
            "id_semestre": row.id_semestre,
            "tipo_autorizacion": random.choice(['AM', 'TO']),
            "num_res_autoriza": generar_resolucion(row.id_semestre)
        })
```

[82]:
```
# GUARDAR EN UN DATAFRAME
df_tutorias2 = pd.DataFrame(tutorias)
```

[83]:
```
print("DATASET TUTORIAS 2021-2025:")
print(df_tutorias2.tail(5))  # Muestra los últimos 5 registros
```

```
DATASET TUTORIAS 2021-2025:
       id_tutoria id_estudiante  id_semestre tipo_autorizacion  \
61218     88511.0        962909            3                AM
```

```
61219     88512.0        962909            3               AM
61220     88513.0        962909            2               TO
61221     88514.0        973021            3               TO
61222     88516.0        997985            2               AM

      num_res_autoriza
61218  RD 502-D-FISI-3
61219  RD 316-D-FISI-3
61220  RD 257-D-FISI-2
61221  RD 496-D-FISI-3
61222  RD 449-D-FISI-2
```

[84]:
```
# UNIR DATASETS DE TUTORIAS (HISTÓRICO Y NUEVO)
df_tutorias3 = pd.concat([df_tutorias, df_tutorias2], ignore_index=True)
```

[86]:
```
# ELIMINAR FILAS VACÍAS
df_tutorias4 = df_tutorias3.dropna(subset=['id_tutoria','id_estudiante',␣
 ↪'id_semestre', 'tipo_autorizacion', 'num_res_autoriza'])
```

[87]:
```
# GUARDAR EL DATASET NUEVO DE TUTORIAS (2019-2025))
df_tutorias4.to_csv('../fuentes_datos/datasets_finales/CSV/tutorias_completo.
 ↪csv', index=False, sep=';')
```