Mosty AI

- dSocial - 1 - Train_text
- dSocial - 2 - Generate Data_text
- dSocial - 3 - Save Synthetic Data_text
- dSocial - 4 - Analysis_text
- tSocial - 1 - Train_text
- tSocial - 2 - Generate Data_text
- tSocial - 3 - Save Synthetic Data_text
- tSocial - 4 - Analysis - Original PKs_text
- tSocial - 4 - Analysis_text

dSocial - 1 - Train_text

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Installations

# COMMAND ----------

# MAGIC %pip install -U mostlyai[local]
# MAGIC %pip install ipywidgets
# MAGIC %restart_python

# COMMAND ----------

import pandas as pd
from mostlyai.sdk import MostlyAI

mostly = MostlyAI(local=True)

# COMMAND ----------

# MAGIC %md
# MAGIC # Read and process tables

# COMMAND ----------

# MAGIC %md
# MAGIC ## Personas

# COMMAND ----------

#ALL ROWS DISTINCT
df_personas = spark.table("admin_govern_sta_des.tmp_mostlyai.personas")
display(df_personas.limit(10))

# COMMAND ----------

# MAGIC %md
```

MAGIC ## Documents Personals

COMMAND ----------

#ALL ROWS DISTINCT
#SAME ROWID_OBJECT_PERSONA MULTIPLE IDENTIFICADOR
df_documents_personals = spark.table("admin_govern_sta_des.tmp_mostlyai.documents_personals")
display(df_documents_personals.limit(10))

COMMAND ----------

MAGIC %md
MAGIC ## Persona Adreça

COMMAND ----------

#MULTIPLE ROWID_OBJECT_PERSONA W/ MULTIPLE IDENTIFICADOR
df_persona_adreca = spark.table("admin_govern_sta_des.tmp_mostlyai.`persona-adreca`")
display(df_persona_adreca.limit(10))

COMMAND ----------

from pyspark.sql.functions import lit
df_persona_adreca = df_persona_adreca.withColumn("const", lit("A"))
display(df_persona_adreca)

COMMAND ----------

MAGIC %md
MAGIC ## Adreça

COMMAND ----------

df_adreca = spark.table("admin_govern_sta_des.tmp_mostlyai.adreca")
display(df_adreca.limit(10))

COMMAND ----------

```
# MAGIC %md

# MAGIC ## Contactes


# COMMAND ----------


#ROWID_OBJECT_PERSONA W/ MULTIPLE IDENTIFICADOR (SOMETIMES MORE THAN 100, ONCE W/ 614"!!)

df_contactes = spark.table("admin_govern_sta_des.tmp_mostlyai.contactes")

display(df_contactes.limit(10))


# COMMAND ----------


# MAGIC %md

# MAGIC # Generator


# COMMAND ----------


# MAGIC %md

# MAGIC ## Configuration


# COMMAND ----------


from mostlyai.sdk.domain import ModelEncodingType


personas_table_config = {

    "name": "personas",

    "data": df_personas,

    "tabular_model_configuration": {

        "max_training_time": 45

    },

    "primary_key": "ROWID_OBJECT_PERSONA",

    "columns": [{"name" : "ROWID_OBJECT_PERSONA", "model_encoding_type": ModelEncodingType.auto},

                {"name" : "NOM", "model_encoding_type": ModelEncodingType.tabular_character},

                {"name" : "COGNOM1", "model_encoding_type": ModelEncodingType.tabular_character},

                {"name" : "COGNOM2", "model_encoding_type": ModelEncodingType.tabular_character}]

}


documents_personals_table_config = {
```

```
    "name": "documents_personals",
    "data": df_documents_personals,
    "tabular_model_configuration": {
        "max_training_time": 45
    },
    "foreign_keys": [{"column": "ROWID_OBJECT_PERSONA", "referenced_table": "personas",
"is_context": True}],
    "columns": [{"name" : "ROWID_OBJECT_PERSONA", "model_encoding_type":
ModelEncodingType.auto},
                {"name" : "IDENTIFICADOR", "model_encoding_type": ModelEncodingType.tabular_character}]
}


# contactes_config = {
#     "name": "contactes",
#     "data": df_contactes,
#     "tabular_model_configuration": {
#         "max_training_time": 10
#     },
#     "foreign_keys": [{"column": "ROWID_OBJECT_PERSONA", "referenced_table": "personas",
"is_context": True}],
#     "columns": [{"name" : "ROWID_OBJECT_PERSONA", "model_encoding_type":
ModelEncodingType.auto},
#                 {"name" : "IDENTIFICADOR", "model_encoding_type":
ModelEncodingType.tabular_character}]
# }



persona_adreca_config = {
    "name": "persona_adreca",
    "data": df_persona_adreca,
    "tabular_model_configuration": {
        "max_training_time": 45
    },
    "foreign_keys": [{"column": "ROWID_OBJECT_PERSONA", "referenced_table": "personas",
"is_context": True},
                     {"column": "ROWID_OBJECT_ADRECA", "referenced_table": "adreca", "is_context": False}],
    "columns": [{"name" : "ROWID_OBJECT_PERSONA", "model_encoding_type":
ModelEncodingType.auto},
                {"name" : "ROWID_OBJECT_ADRECA", "model_encoding_type": ModelEncodingType.auto}]
                #{"name" : "const", "model_encoding_type": ModelEncodingType.tabular_categorical}]
}
```

```
adreca_config = {
   "name": "adreca",
   "data": df_adreca,
   "tabular_model_configuration": {
      "max_training_time": 45
   },
   "primary_key": "ROWID_OBJECT_ADRECA",
   "columns": [{"name" : "ROWID_OBJECT_ADRECA", "model_encoding_type": ModelEncodingType.auto},
               {"name" : "IDENTIFICADOR", "model_encoding_type": ModelEncodingType.tabular_character}]
}

generator_config = {
   "name": "Multi-table Generator",
   "tables": [personas_table_config, documents_personals_table_config, persona_adreca_config, adreca_config] #contactes_config,
}

# COMMAND ----------

# MAGIC %md
# MAGIC ## Train

# COMMAND ----------

generator = mostly.train(name = "ctti-dSocial", config=generator_config, start=True, wait=False)

# COMMAND ----------

generator.training.wait(interval=60)

# COMMAND ----------

# MAGIC %md
# MAGIC ## Save Generator

# COMMAND ----------

generator.export_to_file('/Volumes/admin_govern_sta_des/tmp_mostlyai/generator/dSocial_03_04.zip')
```

dSocial - 2 - Generate Data_text

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Installations

# COMMAND ----------

# MAGIC %pip install -U mostlyai[local]
# MAGIC %pip install ipywidgets
# MAGIC %restart_python

# COMMAND ----------

import time

# COMMAND ----------

time.sleep(600)

# COMMAND ----------

import pandas as pd
from mostlyai.sdk import MostlyAI

mostly = MostlyAI(local=True)

# COMMAND ----------

# MAGIC %md
# MAGIC # Get generator

# COMMAND ----------

[print(i) for i in mostly.generators.list()]

# COMMAND ----------
```

```
generator = mostly.generators.get("82340210-c89b-4db8-9bd8-e032ac6a2c6b")

# COMMAND ----------

# MAGIC %md
# MAGIC ## Save generator

# COMMAND ----------

generator.export_to_file('/Volumes/admin_govern_sta_des/tmp_mostlyai/generator/dSocial_31_03.zip')

# COMMAND ----------

df_personas = spark.table("admin_govern_sta_des.tmp_mostlyai.personas")
#df_contactes = spark.table("admin_govern_sta_des.tmp_mostlyai.contactes")
df_documents_personals = spark.table("admin_govern_sta_des.tmp_mostlyai.documents_personals")
df_persona_adreca = spark.table("admin_govern_sta_des.tmp_mostlyai.`persona-adreca`")
df_adreca = spark.table("admin_govern_sta_des.tmp_mostlyai.adreca")

# COMMAND ----------

# MAGIC %md
# MAGIC # Generation

# COMMAND ----------

# MAGIC %md
# MAGIC ## Configure data generation

# COMMAND ----------

config = {
    "tables": [
        {
            "name": "personas",
            "configuration": {
                "sample_size": 3378516
            }
```

```
        },
        {
            "name": "adreca",
            "configuration": {
                "sample_seed_data": df_adreca
            }
        }
    ]
}

# COMMAND ----------

# MAGIC %md
# MAGIC ## Generate data

# COMMAND ----------

df_samples = mostly.generate(generator, config=config)
```

dSocial - 3 - Save Synthetic Data_text

```
# Databricks notebook source
# MAGIC %md
# MAGIC ## Installations

# COMMAND ----------

# MAGIC %pip install -U mostlyai[local]
# MAGIC %pip install ipywidgets
# MAGIC %restart_python

# COMMAND ----------

import pandas as pd
from mostlyai.sdk import MostlyAI

mostly = MostlyAI(local=True)

# COMMAND ----------

# MAGIC %md
# MAGIC # Get synthetic data

# COMMAND ----------

df_samples = mostly.synthetic_datasets.get("ce967a36-b232-4871-abc6-a9c536928567")

# COMMAND ----------

df_samples.data()

# COMMAND ----------

# MAGIC %md
# MAGIC # Save synthetic data
```

COMMAND ----------

df_documents_personals = spark.createDataFrame(df_samples.data()['documents_personals'])

df_personas = spark.createDataFrame(df_samples.data()['personas'])

df_persona_adreca = spark.createDataFrame(df_samples.data()['persona_adreca'])

df_adreca = spark.createDataFrame(df_samples.data()['adreca'])

COMMAND ----------

df_documents_personals.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_documents_personals_31_03")

df_personas.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_personas_31_03")

df_persona_adreca.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_persona_adreca_31_03")

df_adreca.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_adreca_31_03")

dSocial - 4 - Analysis_text

```
# Databricks notebook source

# MAGIC %md

# MAGIC # Read original and synthetic data


# COMMAND ----------


df_personas = spark.table("admin_govern_sta_des.tmp_mostlyai.personas")

df_documents_personals = spark.table("admin_govern_sta_des.tmp_mostlyai.documents_personals")

df_persona_adreca = spark.table("admin_govern_sta_des.tmp_mostlyai.`persona-adreca`")

df_adreca = spark.table("admin_govern_sta_des.tmp_mostlyai.adreca")

df_contactes = spark.table("admin_govern_sta_des.tmp_mostlyai.contactes")


# COMMAND ----------


df_synthetic_personas = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_personas_31_03")

df_synthetic_documents_personals =
spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_documents_personals_31_03")

df_synthetic_persona_adreca =
spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_persona_adreca_31_03")

df_synthetic_adreca = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_adreca_31_03")



# COMMAND ----------


# MAGIC %md

# MAGIC # Count rows


# COMMAND ----------


tables = [

  df_personas,

  df_documents_personals,

  df_persona_adreca,

  df_adreca,
```

```
    #df_contactes
]

for df in tables:
    print(f"Table has {df.count()} rows")

# COMMAND ----------

tables = [
    df_synthetic_personas,
    df_synthetic_documents_personals,
    df_synthetic_persona_adreca,
    df_synthetic_adreca
]

for df in tables:
    print(f"Table has {df.count()} rows")

# COMMAND ----------

# MAGIC %md
# MAGIC # Checking cardinalities

# COMMAND ----------

# MAGIC %md
# MAGIC ## Personas

# COMMAND ----------

# MAGIC %md
# MAGIC ### Original

# COMMAND ----------
```

```
from pyspark.sql.functions import avg, median


df_grouped = df_personas.groupBy(["ROWID_OBJECT_PERSONA"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ### Synthetic


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_personas.groupBy(["ROWID_OBJECT_PERSONA"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ## Documents personals


# COMMAND ----------


# MAGIC %md

# MAGIC ### Original


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_documents_personals.groupBy(["ROWID_OBJECT_PERSONA"]).count()
```

```
df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ### Synthetic


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_documents_personals.groupBy(["ROWID_OBJECT_PERSONA"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ## Persona-Adreça


# COMMAND ----------


# MAGIC %md

# MAGIC ### Original


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_persona_adreca.groupBy(["ROWID_OBJECT_PERSONA"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

```
df_grouped = df_synthetic_persona_adreca.groupBy(["ROWID_OBJECT_PERSONA"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ## Adreça

COMMAND ----------

MAGIC %md

MAGIC ### Original

COMMAND ----------

```
from pyspark.sql.functions import avg, median

df_grouped = df_adreca.groupBy(["ROWID_OBJECT_ADRECA"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

from pyspark.sql.functions import avg, median

df_grouped = df_synthetic_adreca.groupBy(["ROWID_OBJECT_ADRECA"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)

COMMAND ----------

MAGIC %md

MAGIC # Further analysis

COMMAND ----------

df_grouped_personas =
df_personas.groupBy(["ROWID_OBJECT_PERSONA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_personas)

COMMAND ----------

df_grouped_synthetic_personas =
df_synthetic_personas.groupBy(["ROWID_OBJECT_PERSONA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_personas)

COMMAND ----------

df_grouped_documents_personals =
df_documents_personals.groupBy(["ROWID_OBJECT_PERSONA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_fks").orderBy("n_repetitions")

display(df_grouped_documents_personals)

COMMAND ----------

```
df_grouped_synthetic_documents_personals =
df_synthetic_documents_personals.groupBy(["ROWID_OBJECT_PERSONA"]).count().withColumnRenamed("
count", "n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_fks").orderBy("n_repetitions")

display(df_grouped_synthetic_documents_personals)
```

COMMAND ----------

```
df_grouped_persona_adreca =
df_persona_adreca.groupBy(["ROWID_OBJECT_PERSONA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_fks_ROWID_OBJECT_PERSONA").orderBy("n_repetitions")

display(df_grouped_persona_adreca)
```

COMMAND ----------

```
df_grouped_synthetic_persona_adreca =
df_synthetic_persona_adreca.groupBy(["ROWID_OBJECT_PERSONA"]).count().withColumnRenamed("count"
, "n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_fks_ROWID_OBJECT_PERSONA").orderBy("n_repetitions")

display(df_grouped_synthetic_persona_adreca)
```

COMMAND ----------

```
df_grouped_persona_adreca =
df_persona_adreca.groupBy(["ROWID_OBJECT_ADRECA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_fks_ROWID_OBJECT_ADRECA").orderBy("n_repetitions")

display(df_grouped_persona_adreca)
```

COMMAND ----------

```
df_grouped_synthetic_persona_adreca =
df_synthetic_persona_adreca.groupBy(["ROWID_OBJECT_ADRECA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_fks_ROWID_OBJECT_ADRECA").orderBy("n_repetitions")

display(df_grouped_synthetic_persona_adreca)
```

COMMAND ----------

```
df_grouped_adreca =
df_adreca.groupBy(["ROWID_OBJECT_ADRECA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_adreca)
```

COMMAND ----------

```
df_grouped_adreca =
df_adreca.groupBy(["ROWID_OBJECT_ADRECA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_adreca)
```

COMMAND ----------

```
df_grouped_synthetic_adreca =
df_synthetic_adreca.groupBy(["ROWID_OBJECT_ADRECA"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_adreca)
```

tSocial - 1 - Train_text

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Installations

# COMMAND ----------

# MAGIC %pip install -U mostlyai[local]
# MAGIC %pip install ipywidgets
# MAGIC %restart_python

# COMMAND ----------

import pandas as pd
from mostlyai.sdk import MostlyAI

mostly = MostlyAI(local=True)

# COMMAND ----------

# MAGIC %md
# MAGIC # Read and process tables

# COMMAND ----------

# MAGIC %md
# MAGIC ### Interopko

# COMMAND ----------

from pyspark.sql.functions import col, lit, concat

df_interopko=spark.table("admin_govern_sta_des.tmp_mostlyai.interopko")
df_interopko=(df_interopko
```

```
    .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"),
col("N_Procediment"))))

display(df_interopko.limit(5))


# COMMAND ----------


display(df_interopko.select("Tipus_interop").groupBy("Tipus_interop").count())


# COMMAND ----------


# MAGIC %md

# MAGIC ### Interopok


# COMMAND ----------


df_interopok = spark.table("admin_govern_sta_des.tmp_mostlyai.interopok")

df_interopok=(df_interopok

    .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"),
col("N_Procediment")))

    )

display(df_interopok)


# COMMAND ----------


display(df_interopok.select("Tipus_interop").groupBy("Tipus_interop").count())


# COMMAND ----------


# MAGIC %md

# MAGIC ### Fullquery


# COMMAND ----------


df_fullquery = spark.table("admin_govern_sta_des.tmp_mostlyai.fullquery")

df_fullquery=(df_fullquery

    .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment"))))
```

```
display(df_fullquery.limit(20))


# COMMAND ----------


# MAGIC %md

# MAGIC ### Pagament Nomina


# COMMAND ----------


from pyspark.sql.functions import col, regexp_replace, cast, lit, concat

df_pagamentnomina=spark.table("admin_govern_sta_des.tmp_mostlyai.pagamentnomina")

df_pagamentnomina=(df_pagamentnomina

        .withColumn("Import", regexp_replace(regexp_replace(col("Import"), "\.", ""), ",", ".").cast("double"))

        .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"),
col("N_Procediment"))))

display(df_pagamentnomina.limit(5))


# COMMAND ----------


# MAGIC %md

# MAGIC ### Actuacions nomina


# COMMAND ----------


from pyspark.sql.functions import concat, col, lit


df_actuacionsnomina_raw=spark.table("admin_govern_sta_des.tmp_mostlyai.actuacionsnomina")

df_fullquery_=spark.table("admin_govern_sta_des.tmp_mostlyai.fullquery").select("identificador",
"N_Expedient", "N_Procediment").distinct()


df_actuacionsnomina = (

  df_actuacionsnomina_raw.join(df_fullquery_, on=["N_Expedient", "N_Procediment"], how="left")

  .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment")))

  .select("pk", "identificador", "N_Expedient", "N_Procediment", "Data_actuacio", "Efecte_Nomina")

)
```

```
display(df_actuacionsnomina.limit(5))


# COMMAND ----------


# MAGIC %md

# MAGIC ### Relationship


# COMMAND ----------


from pyspark.sql.functions import col, lit, concat


df_interopko_renamed = df_interopko.withColumnRenamed('Usuari_solicitant', 'identificador')

df_interopok_renamed = df_interopok.withColumnRenamed('Usuari_solicitant', 'identificador')

df_pagamentnomina_renamed = df_pagamentnomina.withColumnRenamed('Usuari_solicitant',
'identificador')


df_relationship = (

 df_interopko_renamed.select(['identificador', 'N_Expedient', 'N_Procediment'])

        .union(df_interopok_renamed.select(['identificador', 'N_Expedient', 'N_Procediment']))

        .union(df_fullquery.select(['identificador', 'N_Expedient', 'N_Procediment']))

        .union(df_pagamentnomina_renamed.select(['identificador', 'N_Expedient', 'N_Procediment']))

        .select("*")

        .distinct()

        .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment")))

        .withColumn("fk", concat(col("identificador"), lit("_"),col("N_Expedient")))

        .select("pk", "fk", "N_Procediment", "identificador")

        )

display(df_relationship.select("pk").distinct())


# COMMAND ----------


display(df_relationship.limit(5))


# COMMAND ----------


# MAGIC %md
```

```
# MAGIC ### Solicitud

# COMMAND ----------

from pyspark.sql.functions import col, lit, concat
df_solicitud=spark.table("admin_govern_sta_des.tmp_mostlyai.solicitud")
df_solicitud=(df_solicitud
        .drop("_c166")
        .withColumn("fk", concat(col("solicitant1Identificador"), lit("_"),col("expedient"))))
display(df_solicitud.limit(5))

# COMMAND ----------

# MAGIC %md
# MAGIC ### Titular

# COMMAND ----------

df_titular=(
 spark.table("admin_govern_sta_des.tmp_mostlyai.titular")
  .withColumnRenamed("identificador", "solicitant1Identificador")
       )
display(df_titular.limit(5))

# COMMAND ----------

df_titular.count()

# COMMAND ----------

# MAGIC %md
# MAGIC ### Esborrany

# COMMAND ----------
```

```
df_esborrany =
spark.table("admin_govern_sta_des.tmp_mostlyai.esborrany").withColumnRenamed("identificador_docume
nt", "solicitant1Identificador")

display(df_esborrany)


# COMMAND ----------


# MAGIC %md

# MAGIC # Generator


# COMMAND ----------


# MAGIC %md

# MAGIC ## Configuration


# COMMAND ----------


from mostlyai.sdk.domain import ModelEncodingType


relationship_table_config = {

  "name": "relationship",

  "data": df_relationship,

  "tabular_model_configuration": {

    "max_training_time": 45

  },

  "primary_key": "pk",

  "foreign_keys": [{"column": "fk", "referenced_table": "solicitud", "is_context": True}],


  "columns": [{"name" : "pk", "model_encoding_type": ModelEncodingType.auto},

        {"name" : "fk", "model_encoding_type": ModelEncodingType.auto},

        {"name" : "N_Procediment", "model_encoding_type": ModelEncodingType.tabular_character},

        {"name" : "identificador", "model_encoding_type": ModelEncodingType.tabular_character}

        ]

}


solicitud_table_config = {
```

```
  "name": "solicitud",
  "data": df_solicitud,
  "tabular_model_configuration": {
    "max_training_time": 240
  },
  "primary_key": "fk",
  "foreign_keys": [{"column": "solicitant1Identificador", "referenced_table": "titular", "is_context": False}],
  "columns": [
   {"name" : "fk", "model_encoding_type": ModelEncodingType.auto},
   {"name": "canalEntrada", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "usuariIniciador", "model_encoding_type": ModelEncodingType.tabular_character},
   {"name": "solicitant1Identificador", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "enviamentAny", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "enviamentMes", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "enviamentDia", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "presentador", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "prestacio", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "dataRegistre", "model_encoding_type": ModelEncodingType.tabular_datetime},
   {"name": "dataInici", "model_encoding_type": ModelEncodingType.tabular_datetime},
   {"name": "solicitant1Genere", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "expedient", "model_encoding_type": ModelEncodingType.tabular_character},
   {"name": "sollicitant1IdentificadorTipus", "model_encoding_type":
ModelEncodingType.tabular_categorical},
   {"name": "sollicitant1Nom", "model_encoding_type": ModelEncodingType.tabular_character},
   {"name": "sollicitant1CognomPrimer", "model_encoding_type": ModelEncodingType.tabular_character},
   {"name": "sollicitant1CognomSegon", "model_encoding_type": ModelEncodingType.tabular_character},
   {"name": "sollicitant1NomSentit", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "sollicitant1NaixementData", "model_encoding_type": ModelEncodingType.tabular_datetime},
   {"name": "sollicitant1DniNacionalitzat", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "sollicitant1DniDataExpedicioDNI", "model_encoding_type":
ModelEncodingType.tabular_datetime},
   {"name": "sollicitant1DniNieAnterior", "model_encoding_type": ModelEncodingType.tabular_categorical},
   {"name": "sollicitant1NieDataCaducitat", "model_encoding_type": ModelEncodingType.tabular_datetime},
   {"name": "sollicitant1NieDataPeticioRenovacio", "model_encoding_type":
ModelEncodingType.tabular_datetime},
   {"name": "sollicitant1NieTipusPermisResidencia", "model_encoding_type":
ModelEncodingType.tabular_categorical},
```

{"name": "sollicitant1EstatCivil", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant1Nacionalitat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioTipus", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioCanalAvis", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "telefonMobil", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "correuElectronic", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "telefonFix", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaTipusVia", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaNomVia", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "residenciaNumero", "model_encoding_type": ModelEncodingType.tabular_numeric_auto},

{"name": "residenciaQualificador", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaBloc", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaEscala", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaPis", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaPorta", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaCodiPostal", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaPoblacio", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaAmpliacioAdreca", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaTipusDomicili", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaTipusLlar", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaRegimTinenca", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residenciaNormalitzada", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioTipusVia", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioNomVia", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "notificacioNumero", "model_encoding_type": ModelEncodingType.tabular_numeric_auto},

{"name": "notificacioQualificador", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioBloc", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioEscala", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioPis", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioPorta", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioCodiPostal", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioPoblacio", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioAmpliacioAdreca", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "notificacioNormalitzada", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "difuntTipusIdentificador", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "difuntIdentificador", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "difuntNom", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "difuntCognomPrimer", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "difuntCognomSegon", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "difuntNaixementData", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "difuntGenere", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "difuntRelacioAmbSollicitant", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "difuntConviviaAmbSollicitant", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioEstrangerDataInici", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "pensioEstrangerImport", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioEstrangerPeriodicitat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioEstrangerPagues", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioEstrangerPagador", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioPlaPrivatDataInici", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "pensioPlaPrivatImport", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioPlaPrivatPeriodicitat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioPlaPrivatPagues", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioPlaPrivatPagador", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioCompensatoriaDataInici", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "pensioCompensatoriaImport", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioCompensatoriaPeriodicitat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioCompensatoriaPagues", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "pensioCompensatoriaPagador", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant1CondicioCatalaRetornat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant1CondicioCatalaRetornatAcreditat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "activitatLaboralDataInici", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "activitatLaboralDataFi", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "activitatLaboralImportBrutAnual", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "activitatLaboralPerCompte", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "activitatLaboralPagador", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residit10Anys", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "residit2DarrersAnys", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "periodesResidenciaHiHa", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "guardaCustodiaInfant", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "guardaCustodiaInfantAcreditat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "unicProgenitor", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "familiaNombrosa", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "familiaNombrosaACatalunya", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "familiaMonoparental", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "violenciaGenere", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "violenciaGenereAcreditat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "progenitorPrivacioLlibertat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "progenitorPrivacioLlibertatQuin", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant2IdentificadorTipus", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant2Identificador", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "sollicitant2DniNacionalitzat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant2DniDataExpedicioDNI", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "sollicitant2DniNieAnterior", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant2NieDataCaducitat", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "sollicitant2NieDataPeticioRenovacio", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "sollicitant2Nom", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "sollicitant2CognomPrimer", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "sollicitant2CognomSegon", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "sollicitant2NaixementData", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "sollicitant2Genere", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant2EstatCivil", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant2Nacionalitat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant2RelacioAmbSollicitant1", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "sollicitant2CondicioCatalaRetornat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "fillMesUn", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "fillsQuants", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1IdentificadorTipus", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1Identificador", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "infant1Nom", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "infant1CognomPrimer", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "infant1CognomSegon", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "infant1NaixementData", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "infant1Genere", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1FiliacioTipus", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1FiliacioData", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "infant1FiliacioAcreditada", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1AcollimentCatalunya", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1AcollimentProvisional", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1NaixementPais", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1NaixementInscrit", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1NaixementInscritRegistre", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant1NaixementInscritConsolat", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant2IdentificadorTipus", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant2Identificador", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "infant2Nom", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "infant2CognomPrimer", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "infant2CognomSegon", "model_encoding_type": ModelEncodingType.tabular_character},

{"name": "infant2NaixementData", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "infant2Genere", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant2FiliacioTipus", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant2FiliacioData", "model_encoding_type": ModelEncodingType.tabular_datetime},

{"name": "infant2FiliacioAcreditada", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant2AcollimentCatalunya", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant2AcollimentProvisional", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant2NaixementPais", "model_encoding_type": ModelEncodingType.tabular_categorical},

{"name": "infant2NaixementInscrit", "model_encoding_type": ModelEncodingType.tabular_categorical},

```
    {"name": "infant2NaixementInscritRegistre", "model_encoding_type":
ModelEncodingType.tabular_categorical},

    {"name": "infant2NaixementInscritConsolat", "model_encoding_type":
ModelEncodingType.tabular_categorical},

    {"name": "personaRefugiada", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "personaRefugiadaAcreditat", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "CIP", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "situacioImv", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "situacioImvAcreditat", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "situacioImvNumSollicitud", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "situacioImvDataPresentacio", "model_encoding_type": ModelEncodingType.tabular_datetime},

    {"name": "situacioImvImport", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "valoratDiscapacitat", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "valoratDiscapacitatAcreditat", "model_encoding_type":
ModelEncodingType.tabular_categorical},

    {"name": "grauDiscapacitat", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "pensioCompensatoriaDret", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "pensioCompensatoriaRep", "model_encoding_type": ModelEncodingType.tabular_categorical},

    {"name": "pensioCompensatoriaAmbReclamacio", "model_encoding_type":
ModelEncodingType.tabular_categorical}]

}


titular_table_config = {

   "name": "titular",

   "data": df_titular,

   "tabular_model_configuration": {

      "max_training_time": 45

   },

   "primary_key": "solicitant1Identificador",

   "columns": [{"name" : "identificadorTipus", "model_encoding_type":
ModelEncodingType.tabular_categorical},

            {"name" : "solicitant1Identificador", "model_encoding_type": ModelEncodingType.auto},

            {"name" : "identificadorNieAnterior", "model_encoding_type":
ModelEncodingType.tabular_categorical},

            {"name" : "nom", "model_encoding_type": ModelEncodingType.tabular_character},

            {"name" : "cognomPrimer", "model_encoding_type": ModelEncodingType.tabular_character},

            {"name" : "cognomSegon", "model_encoding_type": ModelEncodingType.tabular_character},

            {"name" : "nomSentit", "model_encoding_type": ModelEncodingType.tabular_categorical},
```

```
        {"name" : "naixementData", "model_encoding_type": ModelEncodingType.tabular_datetime},

        {"name" : "genere", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "estatCivil", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "nacionalitat", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "situacioPersona", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "residenciaTipusVia", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "residenciaNomVia", "model_encoding_type": ModelEncodingType.tabular_character},

        {"name" : "residenciaNumero", "model_encoding_type": ModelEncodingType.tabular_numeric_auto},

        {"name" : "residenciaQualificador", "model_encoding_type":
ModelEncodingType.tabular_categorical},

        {"name" : "residenciaBloc", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "residenciaEscala", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "residenciaPis", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "residenciaPorta", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "residenciaCodiPostal", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "residenciaPoblacio", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "residenciaAmpliacioAdreca", "model_encoding_type":
ModelEncodingType.tabular_categorical},

        {"name" : "residenciaTipusDomicili", "model_encoding_type":
ModelEncodingType.tabular_categorical},

        {"name" : "telefonMobil", "model_encoding_type": ModelEncodingType.tabular_character},

        {"name" : "telefonFix", "model_encoding_type": ModelEncodingType.tabular_character},

        {"name" : "correuElectronic", "model_encoding_type": ModelEncodingType.tabular_character},

        {"name" : "notificacioTipusVia", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "notificacioNomVia", "model_encoding_type": ModelEncodingType.tabular_character},

        {"name" : "notificacioNumero", "model_encoding_type": ModelEncodingType.tabular_numeric_auto},

        {"name" : "notificacioQualificador", "model_encoding_type":
ModelEncodingType.tabular_categorical},

        {"name" : "notificacioBloc", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "notificacioEscala", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "notificacioPis", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "notificacioPorta", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "notificacioCodiPostal", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "notificacioPoblacio", "model_encoding_type": ModelEncodingType.tabular_categorical},

        {"name" : "notificacioAmpliacioAdreca", "model_encoding_type":
ModelEncodingType.tabular_categorical}]

}
```

```
esborrany_table_config = {
   "name": "esborrany",
   "data": df_esborrany,
   "tabular_model_configuration": {
      "max_training_time": 45
   },
   "foreign_keys": [{"column": "solicitant1Identificador", "referenced_table": "titular", "is_context": True}],
   "columns": [{"name" : "solicitant1Identificador", "model_encoding_type": ModelEncodingType.auto},
            {"name" : "nommunicipi", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "any_creacio", "model_encoding_type": ModelEncodingType.tabular_numeric_auto},
            {"name" : "mes_creacio", "model_encoding_type": ModelEncodingType.tabular_numeric_auto},
            {"name" : "dia_creacio", "model_encoding_type": ModelEncodingType.tabular_numeric_auto},
            {"name" : "prestacio", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "diferida", "model_encoding_type": ModelEncodingType.tabular_categorical}]
}

interopko_table_config = {
   "name": "interopko",
   "data": df_interopko,
   "tabular_model_configuration": {
      "max_training_time": 45
   },
   "foreign_keys": [{"column": "pk", "referenced_table": "relationship", "is_context": True}],
   "columns": [{"name" : "pk", "model_encoding_type": ModelEncodingType.auto},
            #{"name" : "Usuari_solicitant", "model_encoding_type": ModelEncodingType.tabular_categorical},
            #{"name" : "N_Expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
            #{"name" : "N_Procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Data_crida", "model_encoding_type": ModelEncodingType.tabular_datetime},
            {"name" : "Tipus_expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Tipus_procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Tipus_interop", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Estat_interop", "model_encoding_type": ModelEncodingType.tabular_categorical}]
}
```

```
interopok_table_config = {
   "name": "interopok",
   "data": df_interopok,
   "tabular_model_configuration": {
      "max_training_time": 45
   },
   "foreign_keys": [{"column": "pk", "referenced_table": "relationship", "is_context": True}],
   "columns": [{"name" : "pk", "model_encoding_type": ModelEncodingType.auto},
            #{"name" : "Usuari_solicitant", "model_encoding_type": ModelEncodingType.tabular_categorical},
            #{"name" : "N_Expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
            #{"name" : "N_Procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Data_crida", "model_encoding_type": ModelEncodingType.tabular_datetime},
            {"name" : "Tipus_expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Tipus_procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Tipus_interop", "model_encoding_type": ModelEncodingType.tabular_categorical}]
}


fullquery_table_config = {
   "name": "fullquery",
   "data": df_fullquery,
   "tabular_model_configuration": {
      "max_training_time": 90
   },
   "foreign_keys": [{"column": "pk", "referenced_table": "relationship", "is_context": True}],
   "columns": [{"name" : "pk", "model_encoding_type": ModelEncodingType.auto},
            #{"name" : "identificador", "model_encoding_type": ModelEncodingType.tabular_categorical},
            #{"name" : "N_Expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Tipus_Expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
            #{"name" : "N_Procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Tipus_Procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Usuari_Iniciador", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Situacio_Procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
            {"name" : "Motiu_Situacio_Procediment", "model_encoding_type":
ModelEncodingType.tabular_categorical},
            {"name" : "Estat_Calculat_Procediment", "model_encoding_type":
ModelEncodingType.tabular_categorical},
```

```
        {"name" : "Terminacio_Procediment", "model_encoding_type":
ModelEncodingType.tabular_categorical},
        {"name" : "Data_Registre", "model_encoding_type": ModelEncodingType.tabular_datetime},
        {"name" : "Descripcio_tramit", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Data_inicial_tramit", "model_encoding_type": ModelEncodingType.tabular_datetime},
        {"name" : "Data_fi_del_tramit", "model_encoding_type": ModelEncodingType.tabular_datetime},
        {"name" : "Estat_tramit", "model_encoding_type": ModelEncodingType.tabular_categorical}]
  }

pagamentnomina_table_config = {
  "name": "pagamentnomina",
  "data": df_pagamentnomina,
  "tabular_model_configuration": {
    "max_training_time": 45
  },
  "foreign_keys": [{"column": "pk", "referenced_table": "relationship", "is_context": True}],
  "columns": [{"name" : "pk", "model_encoding_type": ModelEncodingType.auto},
        #{"name" : "Usuari_solicitant", "model_encoding_type": ModelEncodingType.tabular_categorical},
        #{"name" : "N_Expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
        #{"name" : "N_Procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Data_pagament", "model_encoding_type": ModelEncodingType.tabular_datetime},
        {"name" : "Import", "model_encoding_type": ModelEncodingType.tabular_numeric_digit},
        {"name" : "Tipus_Expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Tipus_procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Municipi", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Provincia", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Ambit_Territorial", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Banc", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Data_nomina", "model_encoding_type": ModelEncodingType.tabular_datetime}]
}

actuacionsnomina_table_config = {
  "name": "actuacionsnomina",
  "data": df_actuacionsnomina,
  "tabular_model_configuration": {
    "max_training_time": 45
```

```
  },
  "foreign_keys": [{"column": "pk", "referenced_table": "relationship", "is_context": True}],
  "columns": [{"name" : "pk", "model_encoding_type": ModelEncodingType.auto},
        #{"name" : "identificador", "model_encoding_type": ModelEncodingType.tabular_categorical},
        #{"name" : "N_Expedient", "model_encoding_type": ModelEncodingType.tabular_categorical},
        #{"name" : "N_Procediment", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Efecte_Nomina", "model_encoding_type": ModelEncodingType.tabular_categorical},
        {"name" : "Data_actuacio", "model_encoding_type": ModelEncodingType.tabular_datetime}]
}

# COMMAND ----------

generator_config = {
  "name": "Multi-table Generator",
  "tables": [relationship_table_config, solicitud_table_config, titular_table_config, esborrany_table_config, interopko_table_config, interopok_table_config, fullquery_table_config, pagamentnomina_table_config, actuacionsnomina_table_config]
  #"tables": [relationship_table_config, solicitud_table_config, pagamentnomina_table_config, titular_table_config]
}

# COMMAND ----------

# MAGIC %md
# MAGIC ## Train

# COMMAND ----------

# train a synthetic data generator
generator = mostly.train(name="ctti-tsocial", config=generator_config)
print(f"{generator.id}{generator.name} - {generator.accuracy}"
```

tSocial - 2 - Generate Data_text

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Installations

# COMMAND ----------

# MAGIC %pip install -U mostlyai[local]
# MAGIC %pip install ipywidgets
# MAGIC %restart_python

# COMMAND ----------

import time
time.sleep(600)

# COMMAND ----------

import pandas as pd
from mostlyai.sdk import MostlyAI

mostly = MostlyAI(local=True)

# COMMAND ----------

# MAGIC %md
# MAGIC # Get generatior

# COMMAND ----------

[print(i) for i in mostly.generators.list()]

# COMMAND ----------

generator = mostly.generators.get("c5528a9d-7991-47e0-aa93-935dabc71484")
```

COMMAND ----------

MAGIC %md

MAGIC ## Save generator

COMMAND ----------

generator.export_to_file('/Volumes/admin_govern_sta_des/tmp_mostlyai/generator/tSocial_31_03.zip')

COMMAND ----------

MAGIC %md

MAGIC # Generation

COMMAND ----------

df_samples = mostly.generate(generator, size=87_000)

tSocial - 3 - Save Synthetic Data_text

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Installations

# COMMAND ----------

# MAGIC %pip install -U mostlyai[local]
# MAGIC %pip install ipywidgets
# MAGIC %restart_python

# COMMAND ----------

import pandas as pd
from mostlyai.sdk import MostlyAI

mostly = MostlyAI(local=True)

# COMMAND ----------

# MAGIC %md
# MAGIC # Get synthetic data

# COMMAND ----------

df_samples = mostly.synthetic_datasets.get("bbd81e54-1833-4bb8-964c-d96480a6821e")

# COMMAND ----------

df_samples.data()

# COMMAND ----------

# MAGIC %md
# MAGIC # Save synthetic data
```

COMMAND ----------

df_solicitud = spark.createDataFrame(df_samples.data()['solicitud'])

df_interopok = spark.createDataFrame(df_samples.data()['interopok'])

df_interopko = spark.createDataFrame(df_samples.data()['interopko'])

df_fullquery = spark.createDataFrame(df_samples.data()['fullquery'])

df_titular = spark.createDataFrame(df_samples.data()['titular'])

df_pagamentnomina = spark.createDataFrame(df_samples.data()['pagamentnomina'])

df_esborrany = spark.createDataFrame(df_samples.data()['esborrany'])

df_relationship = spark.createDataFrame(df_samples.data()['relationship'])

df_actuacionsnomina = spark.createDataFrame(df_samples.data()['actuacionsnomina'])

COMMAND ----------

df_solicitud.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_solicitud_01_04")

df_interopok.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_interopok_01_04")

df_interopko.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_interopko_01_04")

df_fullquery.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_fullquery_01_04")

df_titular.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_titular_01_04")

df_pagamentnomina.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_pagamentnomina_01_04")

df_esborrany.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_esborrany_01_04")

df_relationship.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_relationship_01_04")

df_actuacionsnomina.write.mode("overwrite").saveAsTable("admin_govern_sta_des.tmp_mostlyai.synthetic_actuacionsnomina_01_04")

tSocial - 4 - Analysis - Original PKs_text

```
# Databricks notebook source

# MAGIC %md

# MAGIC #Read and process original data


# COMMAND ----------


from pyspark.sql.functions import col, lit, concat


df_interopko=spark.table("admin_govern_sta_des.tmp_mostlyai.interopko")

# df_interopko=(df_interopko

#       .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment"))))

display(df_interopko.limit(5))


# COMMAND ----------


df_interopok = spark.table("admin_govern_sta_des.tmp_mostlyai.interopok")

# df_interopok=(df_interopok

#       .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment")))

#       )

display(df_interopok)


# COMMAND ----------


df_fullquery = spark.table("admin_govern_sta_des.tmp_mostlyai.fullquery")

# df_fullquery=(df_fullquery

#       .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment"))))

display(df_fullquery.limit(20))


# COMMAND ----------


from pyspark.sql.functions import col, regexp_replace, cast, lit, concat

df_pagamentnomina=spark.table("admin_govern_sta_des.tmp_mostlyai.pagamentnomina")
```

```
# df_pagamentnomina=(df_pagamentnomina

#                 .withColumn("Import", regexp_replace(regexp_replace(col("Import"), "\.", ""), ",", ".").cast("double"))

#                 .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment"))))

display(df_pagamentnomina.limit(5))


# COMMAND ----------


from pyspark.sql.functions import concat, col, lit


df_actuacionsnomina=spark.table("admin_govern_sta_des.tmp_mostlyai.actuacionsnomina")

# df_actuacionsnomina_raw=spark.table("admin_govern_sta_des.tmp_mostlyai.actuacionsnomina")

# df_fullquery_=spark.table("admin_govern_sta_des.tmp_mostlyai.fullquery").select("identificador", "N_Expedient", "N_Procediment").distinct()


# df_actuacionsnomina = (

#     df_actuacionsnomina_raw.join(df_fullquery_, on=["N_Expedient", "N_Procediment"], how="left")

#     .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment")))

#     .select("pk", "identificador", "N_Expedient", "N_Procediment", "Data_actuacio", "Efecte_Nomina")

# )


# display(df_actuacionsnomina.limit(5))


# COMMAND ----------


from pyspark.sql.functions import col, lit, concat


df_interopko_renamed = df_interopko.withColumnRenamed('Usuari_solicitant', 'identificador')

df_interopok_renamed = df_interopok.withColumnRenamed('Usuari_solicitant', 'identificador')

df_pagamentnomina_renamed = df_pagamentnomina.withColumnRenamed('Usuari_solicitant', 'identificador')


df_relationship = (

  df_interopko_renamed.select(['identificador', 'N_Expedient', 'N_Procediment'])

        .union(df_interopok_renamed.select(['identificador', 'N_Expedient', 'N_Procediment']))

        .union(df_fullquery.select(['identificador', 'N_Expedient', 'N_Procediment']))
```

```
        .union(df_pagamentnomina_renamed.select(['identificador', 'N_Expedient', 'N_Procediment']))
        .select("*")
        .distinct()
        .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment")))
        .withColumn("fk", concat(col("identificador"), lit("_"),col("N_Expedient")))
        .select("pk", "fk", "N_Procediment", "identificador")
        )
display(df_relationship.select("pk").distinct())


# COMMAND ----------


from pyspark.sql.functions import col, lit, concat
df_solicitud=spark.table("admin_govern_sta_des.tmp_mostlyai.solicitud")
# df_solicitud=(df_solicitud
#           .drop("_c166")
#           .withColumn("fk", concat(col("solicitant1Identificador"), lit("_"),col("expedient"))))
display(df_solicitud.limit(5))


# COMMAND ----------


df_titular=(
  spark.table("admin_govern_sta_des.tmp_mostlyai.titular")
   .withColumnRenamed("identificador", "solicitant1Identificador")
        )
display(df_titular.limit(5))


# COMMAND ----------


df_esborrany =
spark.table("admin_govern_sta_des.tmp_mostlyai.esborrany").withColumnRenamed("identificador_docume
nt", "solicitant1Identificador")
display(df_esborrany)


# COMMAND ----------


# MAGIC %md
```

MAGIC # Read and process synthetic data

COMMAND ----------

df_synthetic_actuacionsnomina = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_actuacionsnomina_01_04")

df_synthetic_esborrany = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_esborrany_01_04")

df_synthetic_fullquery = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_fullquery_01_04")

df_synthetic_interopko = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_interopko_01_04")

df_synthetic_interopok = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_interopok_01_04")

df_synthetic_solicitud = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_solicitud_01_04")

df_synthetic_pagamentnomina = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_pagamentnomina_01_04")

df_synthetic_titular = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_titular_01_04")

df_synthetic_relationship = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_relationship_01_04")

COMMAND ----------

df_synthetic_relationship_joined = df_synthetic_relationship.join(df_synthetic_solicitud.select("fk", "expedient"), on=['fk'], how='left')

display(df_synthetic_relationship_joined)

COMMAND ----------

df_synthetic_interopok_joined = df_synthetic_interopok.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_interopok_joined)

COMMAND ----------

df_synthetic_interopok_joined = df_synthetic_interopok.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_interopok_joined)

COMMAND ----------

```
df_synthetic_interopko_joined = df_synthetic_interopko.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_interopko_joined)


# COMMAND ----------


df_synthetic_fullquery_joined = df_synthetic_fullquery.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_fullquery_joined)


# COMMAND ----------


df_synthetic_pagamentnomina_joined = df_synthetic_pagamentnomina.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_pagamentnomina_joined)


# COMMAND ----------


df_synthetic_actuacionsnomina_joined = df_synthetic_actuacionsnomina.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_actuacionsnomina_joined)


# COMMAND ----------


display(df_synthetic_pagamentnomina_joined.limit(10))


# COMMAND ----------


display(df_pagamentnomina_renamed.limit(10))


# COMMAND ----------


# MAGIC %md

# MAGIC # Example


# COMMAND ----------
```

```
display(df_pagamentnomina_renamed.limit(10))


# COMMAND ----------


display(df_synthetic_pagamentnomina_joined.limit(10))


# COMMAND ----------


# MAGIC %md
# MAGIC # Count rows


# COMMAND ----------


tables = [
    df_actuacionsnomina,
    df_esborrany,
    df_fullquery,
    df_interopko,
    df_interopok,
    df_solicitud,
    df_pagamentnomina,
    df_titular,
    df_relationship
]


for df in tables:
    print(f"Table has {df.count()} rows")


# COMMAND ----------


tables = [
    df_synthetic_actuacionsnomina,
    df_synthetic_esborrany,
    df_synthetic_fullquery,
```

```
    df_synthetic_interopko,
    df_synthetic_interopok,
    df_synthetic_solicitud,
    df_synthetic_pagamentnomina,
    df_synthetic_titular,
    df_synthetic_relationship
]

for df in tables:
    print(f"Table has {df.count()} rows")

# COMMAND ----------

# MAGIC %md
# MAGIC # Checking cardinalities

# COMMAND ----------

# MAGIC %md
# MAGIC ## ActuacionsNomina

# COMMAND ----------

# MAGIC %md
# MAGIC ### Original

# COMMAND ----------

display(df_actuacionsnomina)

# COMMAND ----------

from pyspark.sql.functions import avg, median

df_grouped = df_actuacionsnomina.groupBy(["Efecte_Nomina", "N_Procediment", "N_Expedient"]).count()
```

```
df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ### Synthetic


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_actuacionsnomina_joined.groupBy(["Efecte_Nomina", "N_Procediment",
"expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ## Pagament nomina


# COMMAND ----------


# MAGIC %md

# MAGIC ### Original


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_pagamentnomina.groupBy(["Data_nomina", "N_Procediment", "N_Expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

```
from pyspark.sql.functions import avg, median

df_grouped = df_synthetic_pagamentnomina_joined.groupBy(["Data_nomina", "N_Procediment",
"expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ## Full Query

COMMAND ----------

MAGIC %md

MAGIC ### Original

COMMAND ----------

```
from pyspark.sql.functions import avg, median

df_grouped = df_fullquery.groupBy(["identificador", "N_Procediment", "N_Expedient",
"Descripcio_tramit"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

from pyspark.sql.functions import avg, median

df_grouped = df_synthetic_fullquery_joined.groupBy(["identificador", "N_Procediment", "expedient", "Descripcio_tramit"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)

COMMAND ----------

MAGIC %md

MAGIC ## Interopko

COMMAND ----------

MAGIC %md

MAGIC ### Original

COMMAND ----------

from pyspark.sql.functions import avg, median

df_grouped = df_interopko.groupBy(["Data_crida","Tipus_interop", "N_Procediment", "N_Expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

from pyspark.sql.functions import avg, median

df_grouped = df_synthetic_interopko_joined.groupBy(["Data_crida","Tipus_interop", "N_Procediment", "expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)

COMMAND ----------

MAGIC %md

MAGIC ## Interopok

COMMAND ----------

MAGIC %md

MAGIC ### Original

COMMAND ----------

from pyspark.sql.functions import avg, median

df_grouped = df_interopok.groupBy(["Tipus_interop", "N_Procediment", "N_Expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

```
# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_interopok_joined.groupBy(["Tipus_interop", "N_Procediment",
"expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ## Solicitud


# COMMAND ----------


# MAGIC %md

# MAGIC ### Original


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_solicitud.groupBy(["expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ### Synthetic


# COMMAND ----------
```

```
from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_solicitud.groupBy(["expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ## Titular


# COMMAND ----------


# MAGIC %md

# MAGIC ### Original


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_titular.groupBy(["solicitant1identificador"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ### Synthetic


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_titular.groupBy(["solicitant1identificador"]).count()
```

```
df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_relationship.groupBy(["pk"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_relationship.groupBy(["pk"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC # Further Analysis


# COMMAND ----------


display(df_actuacionsnomina.limit(10))


# COMMAND ----------


df_grouped_actuacionsnomina = df_actuacionsnomina.groupBy(["N_Expedient", "N_Procediment",
"Efecte_Nomina"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_actuacionsnomina)
```

COMMAND ----------

```
df_synthetic_actuacionsnomina_joined =
df_synthetic_actuacionsnomina.join(df_synthetic_relationship_joined.select("pk", "identificador",
"N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_actuacionsnomina_joined)
```

COMMAND ----------

```
df_grouped_synthetic_actuacionsnomina = df_synthetic_actuacionsnomina_joined.groupBy(["expedient",
"N_Procediment", "Efecte_Nomina"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_actuacionsnomina)
```

COMMAND ----------

COMMAND ----------

```
df_grouped_pagamentnomina = df_pagamentnomina.groupBy(["N_Expedient", "N_Procediment",
"Data_Nomina"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_pagamentnomina)
```

COMMAND ----------

```
df_grouped_synthetic_pagamentnomina =
df_synthetic_pagamentnomina_joined.groupBy(["Tipus_Expedient", "N_Procediment",
"Data_Nomina"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_pagamentnomina)
```

COMMAND ----------

```
df_grouped_fullquery = df_fullquery.groupBy(["N_Expedient", "N_Procediment", "identificador",
"Descripcio_tramit"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_fullquery)


# COMMAND ----------


df_grouped_synthetic_fullquery = df_synthetic_fullquery_joined.groupBy(["expedient", "N_Procediment",
"identificador", "Descripcio_tramit"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_fullquery)


# COMMAND ----------


df_grouped_interopok = df_interopok.groupBy(["N_Expedient", "N_Procediment",
"Tipus_interop"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_interopok)


# COMMAND ----------


df_grouped_synthetic_interopok = df_synthetic_interopok_joined.groupBy(["expedient", "N_Procediment",
"Tipus_interop"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_interopok)


# COMMAND ----------


df_grouped_interopko = df_interopko.groupBy(["N_Expedient", "N_Procediment", "Tipus_interop",
"Data_crida"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_interopko)


# COMMAND ----------


df_grouped_synthetic_interopko = df_synthetic_interopko_joined.groupBy(["expedient", "N_Procediment",
"Tipus_interop", "Data_crida"]).count().withColumnRenamed("count",
```

```
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_interopko)


# COMMAND ----------


df_grouped_solicitud = df_solicitud.groupBy(["expedient"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_solicitud)


# COMMAND ----------


df_grouped_synthetic_solicitud =
df_synthetic_solicitud.groupBy(["expedient"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_solicitud)


# COMMAND ----------


df_grouped_titular = df_titular.groupBy(["solicitant1identificador"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_titular)


# COMMAND ----------


df_grouped_synthetic_titular =
df_synthetic_titular.groupBy(["solicitant1identificador"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_titular)
```

tSocial - 4 - Analysis_text

```
# Databricks notebook source
# MAGIC %md
# MAGIC # Read and process original data (build new pk)

# COMMAND ----------

from pyspark.sql.functions import col, lit, concat

df_interopko=spark.table("admin_govern_sta_des.tmp_mostlyai.interopko")
df_interopko=(df_interopko
        .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"),
col("N_Procediment"))))
display(df_interopko.limit(5))

# COMMAND ----------

df_interopok = spark.table("admin_govern_sta_des.tmp_mostlyai.interopok")
df_interopok=(df_interopok
        .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"),
col("N_Procediment")))
        )
display(df_interopok)

# COMMAND ----------

df_fullquery = spark.table("admin_govern_sta_des.tmp_mostlyai.fullquery")
df_fullquery=(df_fullquery
        .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"),
col("N_Procediment"))))
display(df_fullquery.limit(20))

# COMMAND ----------

from pyspark.sql.functions import col, regexp_replace, cast, lit, concat
df_pagamentnomina=spark.table("admin_govern_sta_des.tmp_mostlyai.pagamentnomina")
df_pagamentnomina=(df_pagamentnomina
```

```
                .withColumn("Import", regexp_replace(regexp_replace(col("Import"), "\.", ""), ",", ".").cast("double"))
                .withColumn("pk", concat(col("Usuari_solicitant"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment"))))
display(df_pagamentnomina.limit(5))


# COMMAND ----------


from pyspark.sql.functions import concat, col, lit


df_actuacionsnomina_raw=spark.table("admin_govern_sta_des.tmp_mostlyai.actuacionsnomina")
df_fullquery_=spark.table("admin_govern_sta_des.tmp_mostlyai.fullquery").select("identificador", "N_Expedient", "N_Procediment").distinct()


df_actuacionsnomina = (
    df_actuacionsnomina_raw.join(df_fullquery_, on=["N_Expedient", "N_Procediment"], how="left")
    .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"), col("N_Procediment")))
    .select("pk", "identificador", "N_Expedient", "N_Procediment", "Data_actuacio", "Efecte_Nomina")
)


display(df_actuacionsnomina.limit(5))


# COMMAND ----------


from pyspark.sql.functions import col, lit, concat


df_interopko_renamed = df_interopko.withColumnRenamed('Usuari_solicitant', 'identificador')
df_interopok_renamed = df_interopok.withColumnRenamed('Usuari_solicitant', 'identificador')
df_pagamentnomina_renamed = df_pagamentnomina.withColumnRenamed('Usuari_solicitant', 'identificador')


df_relationship = (
  df_interopko_renamed.select(['identificador', 'N_Expedient', 'N_Procediment'])
          .union(df_interopok_renamed.select(['identificador', 'N_Expedient', 'N_Procediment']))
          .union(df_fullquery.select(['identificador', 'N_Expedient', 'N_Procediment']))
          .union(df_pagamentnomina_renamed.select(['identificador', 'N_Expedient', 'N_Procediment']))
          .select("*")
          .distinct()
```

```
        .withColumn("pk", concat(col("identificador"), lit("_"),col("N_Expedient"), lit("_"),
col("N_Procediment")))
        .withColumn("fk", concat(col("identificador"), lit("_"),col("N_Expedient")))
        .select("pk", "fk", "N_Procediment", "identificador")
        )
display(df_relationship.select("pk").distinct())


# COMMAND ----------


from pyspark.sql.functions import col, lit, concat
df_solicitud=spark.table("admin_govern_sta_des.tmp_mostlyai.solicitud")
df_solicitud=(df_solicitud
          .drop("_c166")
          .withColumn("fk", concat(col("solicitant1Identificador"), lit("_"),col("expedient"))))
display(df_solicitud.limit(5))


# COMMAND ----------


df_titular=(
  spark.table("admin_govern_sta_des.tmp_mostlyai.titular")
    .withColumnRenamed("identificador", "solicitant1Identificador")
        )
display(df_titular.limit(5))


# COMMAND ----------


df_esborrany =
spark.table("admin_govern_sta_des.tmp_mostlyai.esborrany").withColumnRenamed("identificador_docum
ent", "solicitant1Identificador")
display(df_esborrany)


# COMMAND ----------


# MAGIC %md
# MAGIC # Read and process synthetic data (fetch cols from relationship)


# COMMAND ----------
```

```
df_synthetic_actuacionsnomina =
spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_actuacionsnomina_01_04")

df_synthetic_esborrany = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_esborrany_01_04")

df_synthetic_fullquery = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_fullquery_01_04")

df_synthetic_interopko = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_interopko_01_04")

df_synthetic_interopok = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_interopok_01_04")

df_synthetic_solicitud = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_solicitud_01_04")

df_synthetic_pagamentnomina =
spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_pagamentnomina_01_04")

df_synthetic_titular = spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_titular_01_04")


df_synthetic_relationship =
spark.table("admin_govern_sta_des.tmp_mostlyai.synthetic_relationship_01_04")


# COMMAND ----------


df_synthetic_relationship_joined = df_synthetic_relationship.join(df_synthetic_solicitud.select("fk",
"expedient"), on=['fk'], how='left')

display(df_synthetic_relationship_joined)


# COMMAND ----------


df_synthetic_interopok_joined = df_synthetic_interopok.join(df_synthetic_relationship_joined.select("pk",
"identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_interopok_joined)


# COMMAND ----------


df_synthetic_interopok_joined = df_synthetic_interopok.join(df_synthetic_relationship_joined.select("pk",
"identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_interopok_joined)


# COMMAND ----------


df_synthetic_interopko_joined = df_synthetic_interopko.join(df_synthetic_relationship_joined.select("pk",
"identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_interopko_joined)


# COMMAND ----------
```

```
df_synthetic_fullquery_joined = df_synthetic_fullquery.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_fullquery_joined)


# COMMAND ----------


df_synthetic_pagamentnomina_joined = df_synthetic_pagamentnomina.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_pagamentnomina_joined)


# COMMAND ----------


df_synthetic_actuacionsnomina_joined = df_synthetic_actuacionsnomina.join(df_synthetic_relationship_joined.select("pk", "identificador", "N_Procediment", "expedient"), on=['pk'], how='left')

display(df_synthetic_actuacionsnomina_joined)


# COMMAND ----------


# MAGIC %md

# MAGIC # Example


# COMMAND ----------


display(df_pagamentnomina_renamed.limit(10))


# COMMAND ----------


display(df_synthetic_pagamentnomina_joined.limit(10))


# COMMAND ----------


# MAGIC %md

# MAGIC # Count rows


# COMMAND ----------
```

```
tables = [
    df_actuacionsnomina,
    df_esborrany,
    df_fullquery,
    df_interopko,
    df_interopok,
    df_solicitud,
    df_pagamentnomina,
    df_titular,
    df_relationship
]

for df in tables:
    print(f"Table has {df.count()} rows")

# COMMAND ----------

tables = [
    df_synthetic_actuacionsnomina,
    df_synthetic_esborrany,
    df_synthetic_fullquery,
    df_synthetic_interopko,
    df_synthetic_interopok,
    df_synthetic_solicitud,
    df_synthetic_pagamentnomina,
    df_synthetic_titular,
    df_synthetic_relationship
]

for df in tables:
    print(f"Table has {df.count()} rows")

# COMMAND ----------

# MAGIC %md
# MAGIC # Checking cardinalities
```

COMMAND ----------

MAGIC %md

MAGIC ## Actuacions Nomina

COMMAND ----------

MAGIC %md

MAGIC ### Original

COMMAND ----------

```
df_grouped = df_actuacionsnomina.groupBy(["identificador", "N_Procediment", "N_Expedient"]).count()

df = df_grouped.groupBy("count").count()

display(df)
```

COMMAND ----------

```
df_grouped = df_synthetic_actuacionsnomina.groupBy(["pk"]).count()

df = df_grouped.groupBy("count").count()

display(df)
```

COMMAND ----------

```
from pyspark.sql.functions import avg, median

df_grouped = df_actuacionsnomina.groupBy(["identificador", "N_Procediment", "N_Expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

from pyspark.sql.functions import avg, median

df_grouped = df_synthetic_actuacionsnomina_joined.groupBy(["identificador", "N_Procediment", "expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)

COMMAND ----------

MAGIC %md

MAGIC ## Pagament nomina

COMMAND ----------

MAGIC %md

MAGIC ### Original

COMMAND ----------

from pyspark.sql.functions import avg, median

df_grouped = df_pagamentnomina.groupBy(["Usuari_solicitant", "N_Procediment", "N_Expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

from pyspark.sql.functions import avg, median

```
df_grouped = df_synthetic_pagamentnomina_joined.groupBy(["identificador", "N_Procediment", "expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ## Full Query


# COMMAND ----------


# MAGIC %md

# MAGIC ### Original


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_fullquery.groupBy(["identificador", "N_Procediment", "N_Expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ### Synthetic


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_fullquery_joined.groupBy(["identificador", "N_Procediment", "expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"), median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ## Interopko

COMMAND ----------

MAGIC %md

MAGIC ### Original

COMMAND ----------

```
from pyspark.sql.functions import avg, median


df_grouped = df_interopko.groupBy(["Usuari_solicitant", "N_Procediment", "N_Expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

```
from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_interopko_joined.groupBy(["identificador", "N_Procediment",
"expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ## Interopok

COMMAND ----------

MAGIC %md

MAGIC ### Original

COMMAND ----------

```
from pyspark.sql.functions import avg, median

df_grouped = df_interopok.groupBy(["Usuari_solicitant", "N_Procediment", "N_Expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ### Synthetic

COMMAND ----------

```
from pyspark.sql.functions import avg, median

df_grouped = df_synthetic_interopok_joined.groupBy(["identificador", "N_Procediment",
"expedient"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)
```

COMMAND ----------

MAGIC %md

MAGIC ## Solicituds

COMMAND ----------

```
# MAGIC %md

# MAGIC ### Original


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_solicitud.groupBy(["expedient", "solicitant1Identificador"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ### Synthetic


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_solicitud.groupBy(["expedient", "solicitant1Identificador"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ## Titular


# COMMAND ----------


# MAGIC %md

# MAGIC ###Original


# COMMAND ----------
```

```
from pyspark.sql.functions import avg, median


df_grouped = df_titular.groupBy(["solicitant1identificador"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC ### Synthetic


# COMMAND ----------


from pyspark.sql.functions import avg, median


df_grouped = df_synthetic_titular.groupBy(["solicitant1identificador"]).count()

df_avg_repetitions = df_grouped.agg(avg("count").alias("avg_repetitions"),
median("count").alias("median_repetitions"))

display(df_avg_repetitions)


# COMMAND ----------


# MAGIC %md

# MAGIC # Further analysis


# COMMAND ----------


df_grouped_actuacionsnomina = df_actuacionsnomina.groupBy(["identificador", "N_Procediment",
"N_Expedient"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_actuacionsnomina)


# COMMAND ----------


df_grouped_synthetic_actuacionsnomina =
df_synthetic_actuacionsnomina.groupBy(["pk"]).count().withColumnRenamed("count",
```

```
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_actuacionsnomina)


# COMMAND ----------


df_grouped_pagamentnomina = df_pagamentnomina_renamed.groupBy(["identificador",
"N_Procediment", "N_Expedient"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_pagamentnomina)


# COMMAND ----------


df_grouped_synthetic_pagamentnomina =
df_synthetic_pagamentnomina.groupBy(["pk"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_pagamentnomina)


# COMMAND ----------


df_grouped_fullquery = df_fullquery.groupBy(["identificador", "N_Procediment",
"N_Expedient"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_fullquery)


# COMMAND ----------


df_grouped_synthetic_fullquery =
df_synthetic_fullquery.groupBy(["pk"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_fullquery)


# COMMAND ----------


df_grouped_interopok = df_interopok_renamed.groupBy(["identificador", "N_Procediment",
"N_Expedient"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_interopok)
```

```
# COMMAND ----------


df_grouped_synthetic_interopok =
df_synthetic_interopok.groupBy(["pk"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_interopok)


# COMMAND ----------


df_grouped_interopko = df_interopko_renamed.groupBy(["identificador", "N_Procediment",
"N_Expedient"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_interopko)


# COMMAND ----------


df_grouped_synthetic_interopko =
df_synthetic_interopko.groupBy(["pk"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_interopko)


# COMMAND ----------


display(df_solicitud)


# COMMAND ----------


df_grouped_solicitud = df_solicitud.groupBy(["fk"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_solicitud)


# COMMAND ----------


df_grouped_synthetic_solicitud =
df_synthetic_solicitud.groupBy(["fk"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")
```

```
display(df_grouped_synthetic_solicitud)


# COMMAND ----------


df_grouped_titular = df_titular.groupBy(["solicitant1identificador"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_titular)


# COMMAND ----------


df_grouped_synthetic_titular =
df_synthetic_titular.groupBy(["solicitant1identificador"]).count().withColumnRenamed("count",
"n_repetitions").groupBy("n_repetitions").count().withColumnRenamed("count",
"n_pks").orderBy("n_repetitions")

display(df_grouped_synthetic_titular)


# COMMAND ----------


# COMMAND ----------


# df = df_grouped.groupBy("count").count()

# display(df)
```