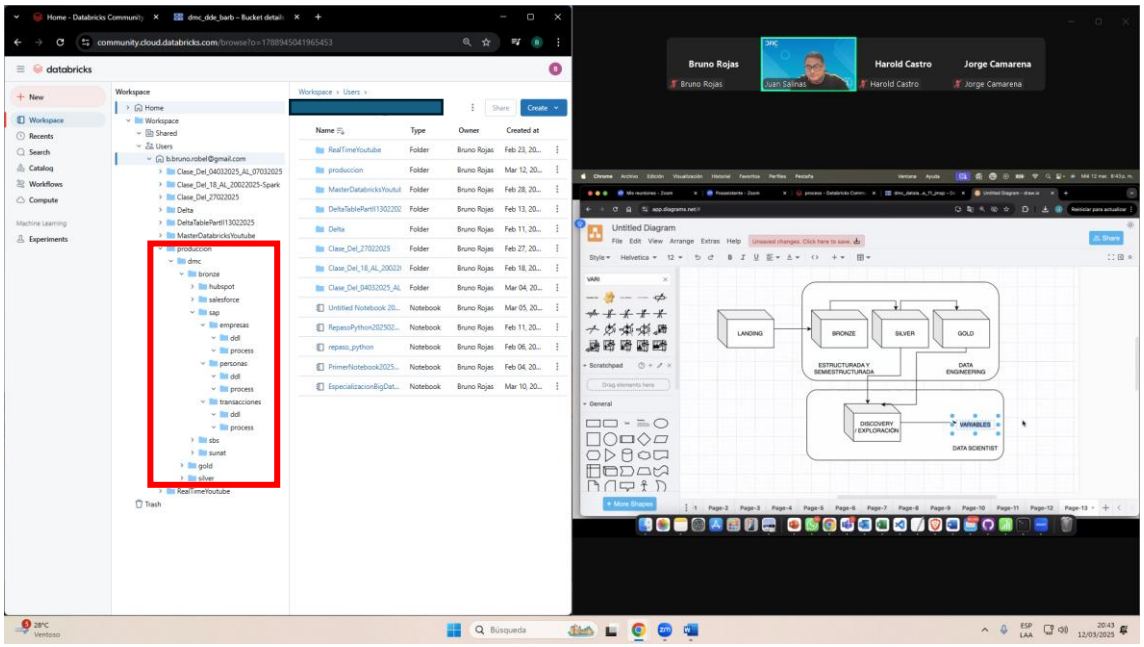
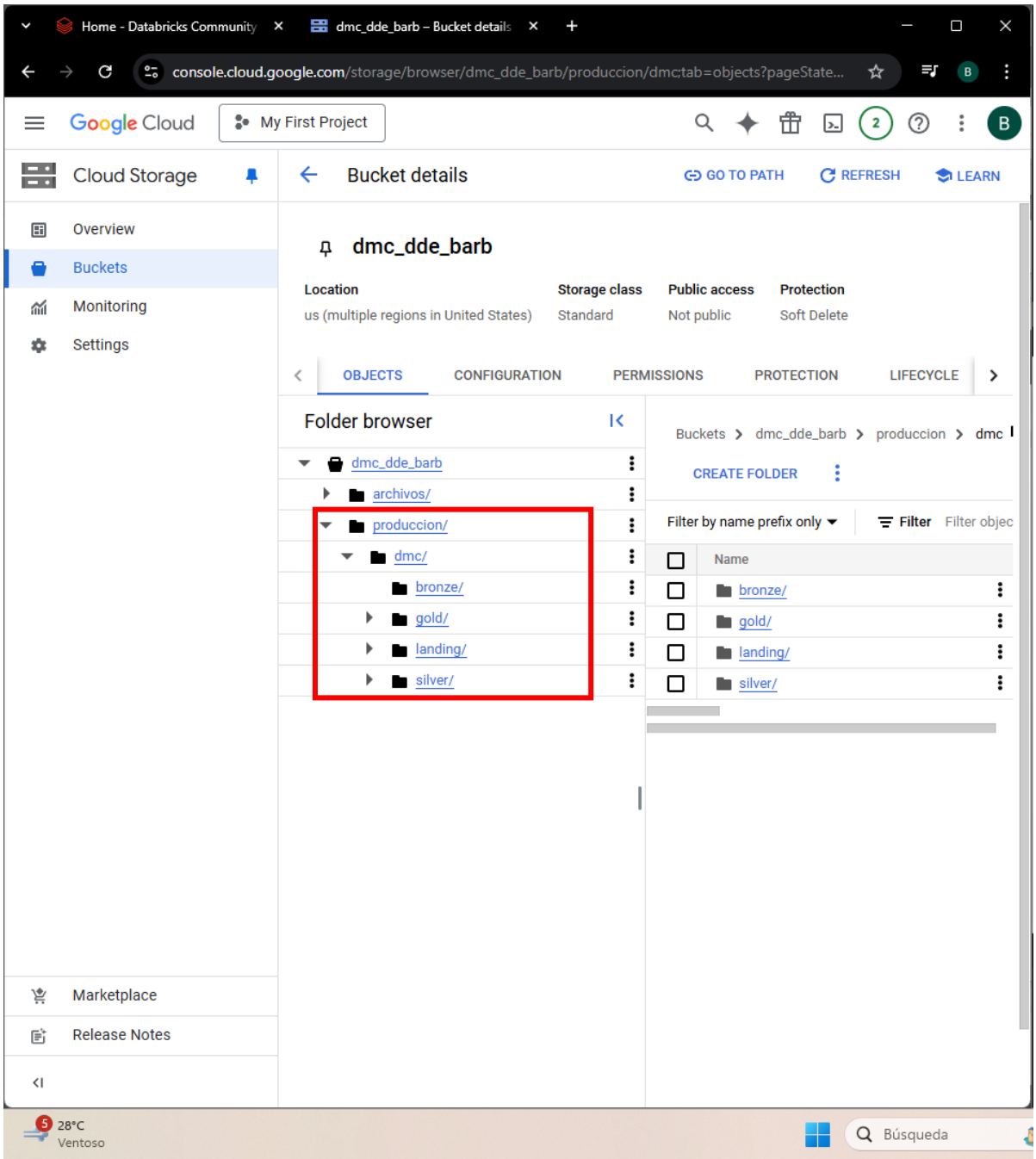
4.-Lakehouse

Lakehouse con Databricks y Google Cloud Storage

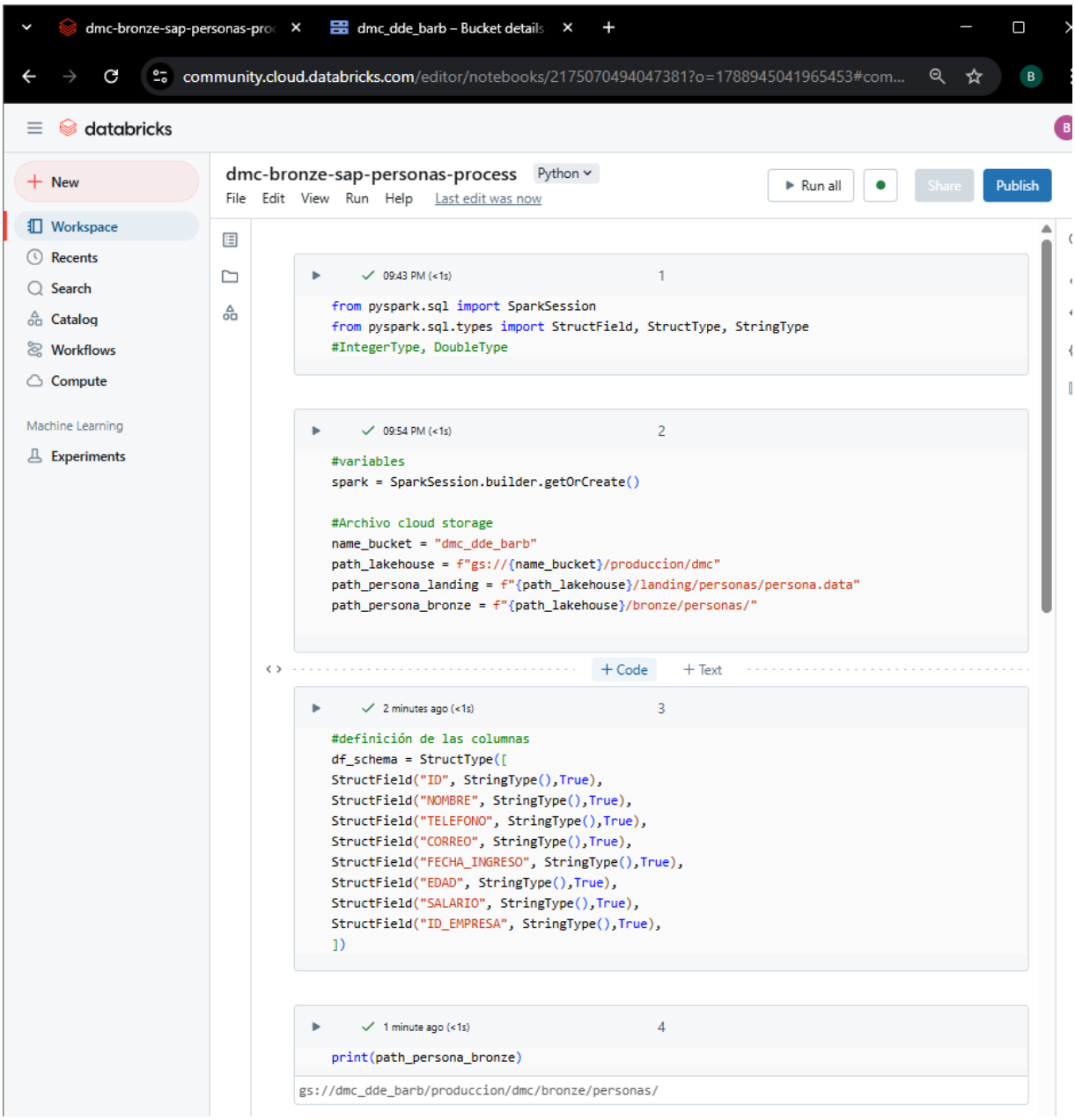
Creación de estructura de carpetas



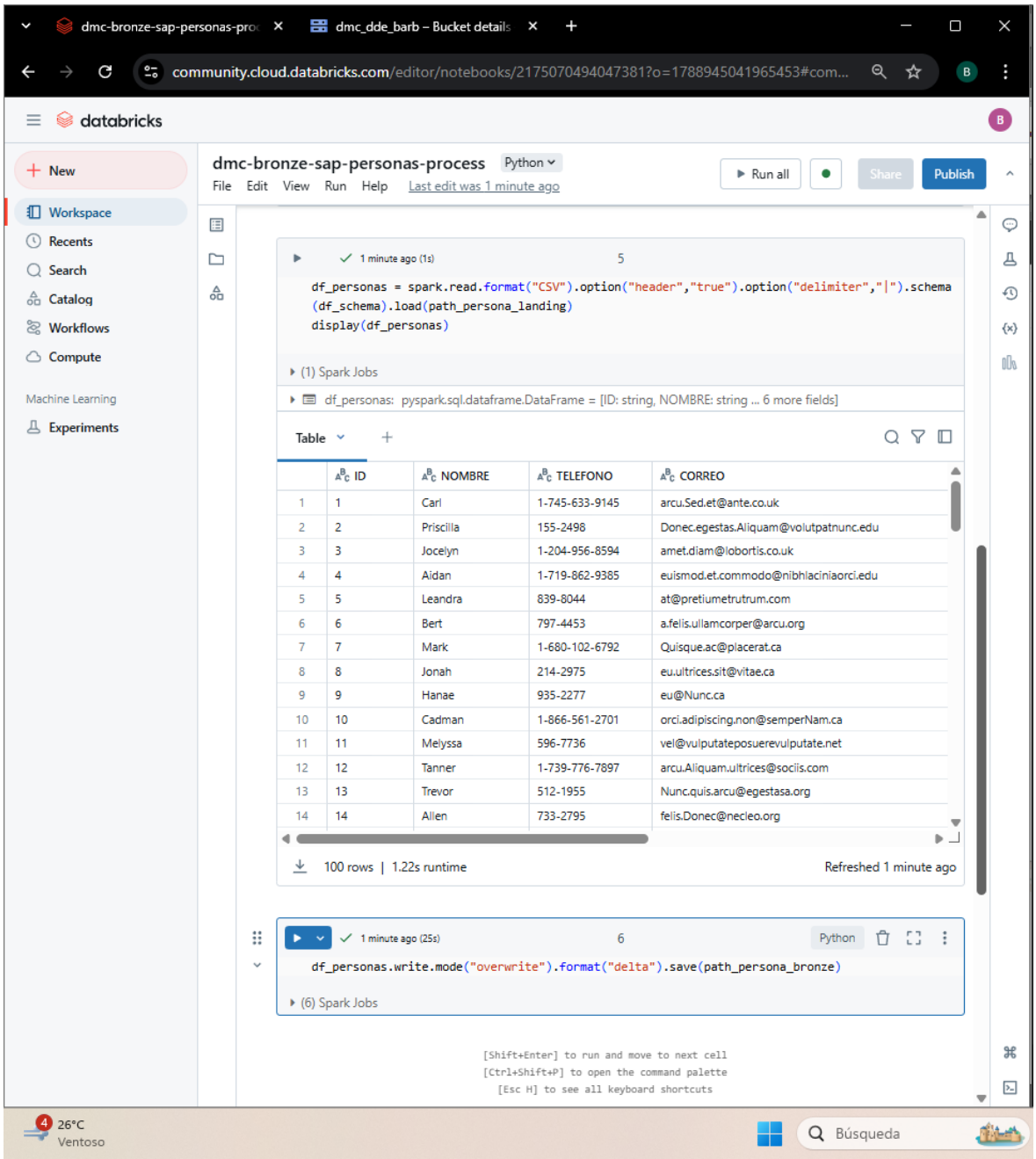
Creación de bucket en Google Cloud Storage para alimentar el Lakehouse



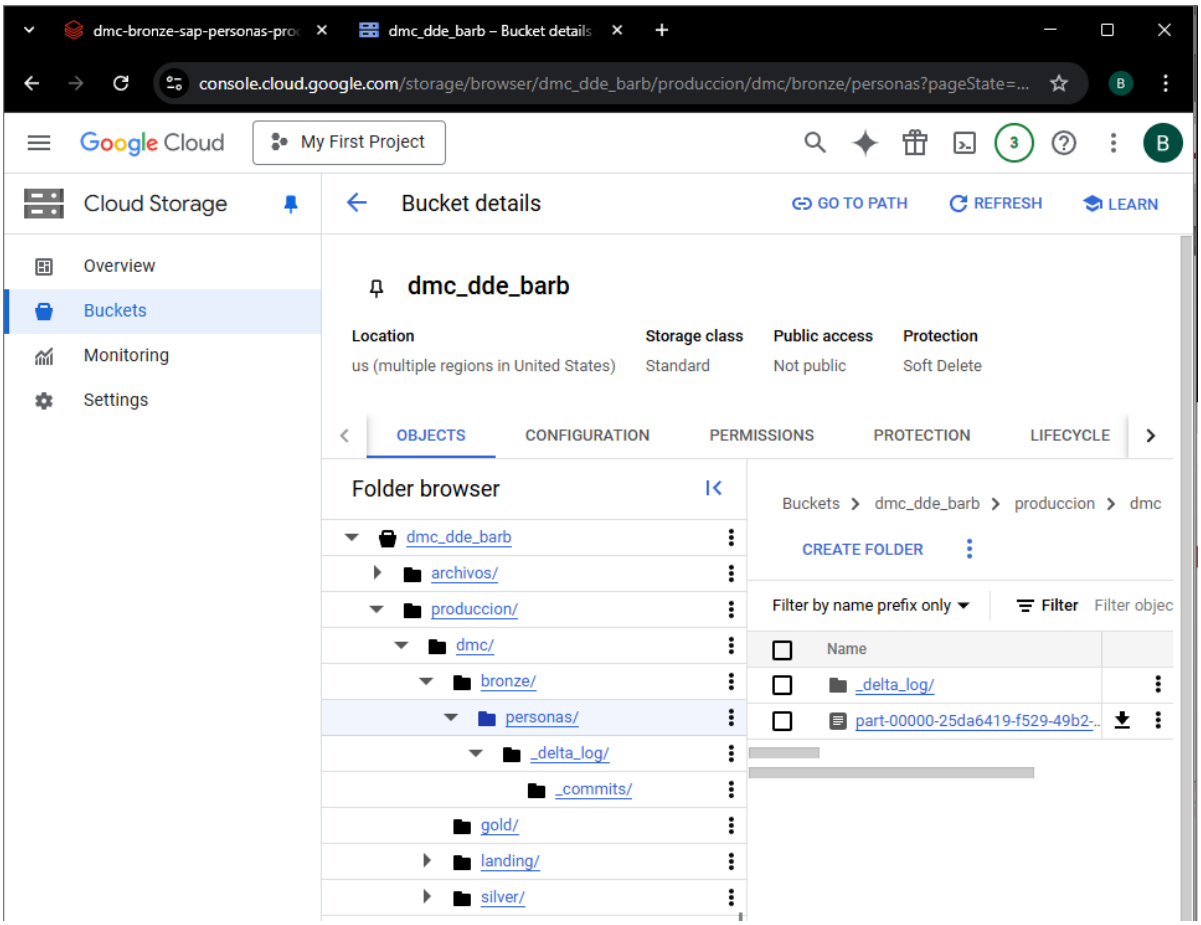
Leyendo datos de la capa landing de personas desde el bucket y armando el esquema del dataframe.


dmc-bronze-sap-personas-proc
dmc_dde_barb – Bucket details
community.cloud.databricks.com/editor/notebooks/2175070494047381?o=1788945041965453#com...
databricks
New
Workspace
Recents
Search
Catalog
Workflows
Compute
Machine Learning
Experiments
dmc-bronze-sap-personas-process
Python
File Edit View Run Help
Last edit was now
Run all
Share
Publish
09:43 PM (<1s) 1
from pyspark.sql import SparkSession
from pyspark.sql.types import StructField, StructType, StringType
#IntegerType, DoubleType
09:54 PM (<1s) 2
#variables
spark = SparkSession.builder.getOrCreate()
#Archivo cloud storage
name_bucket = "dmc_dde_barb"
path_lakehouse = f"gs://{name_bucket}/produccion/dmc"
path_persona_landing = f"{path_lakehouse}/landing/personas/persona.data"
path_persona_bronze = f"{path_lakehouse}/bronze/personas/"
+ Code
+ Text
2 minutes ago (<1s) 3
#definición de las columnas
df_schema = StructType([
StructField("ID", StringType(),True),
StructField("NOMBRE", StringType(),True),
StructField("TELEFONO", StringType(),True),
StructField("CORREO", StringType(),True),
StructField("FECHA_INGRESO", StringType(),True),
StructField("EDAD", StringType(),True),
StructField("SALARIO", StringType(),True),
StructField("ID_EMPRESA", StringType(),True),
])
1 minute ago (<1s) 4
print(path_persona_bronze)
gs://dmc_dde_barb/produccion/dmc/bronze/personas/

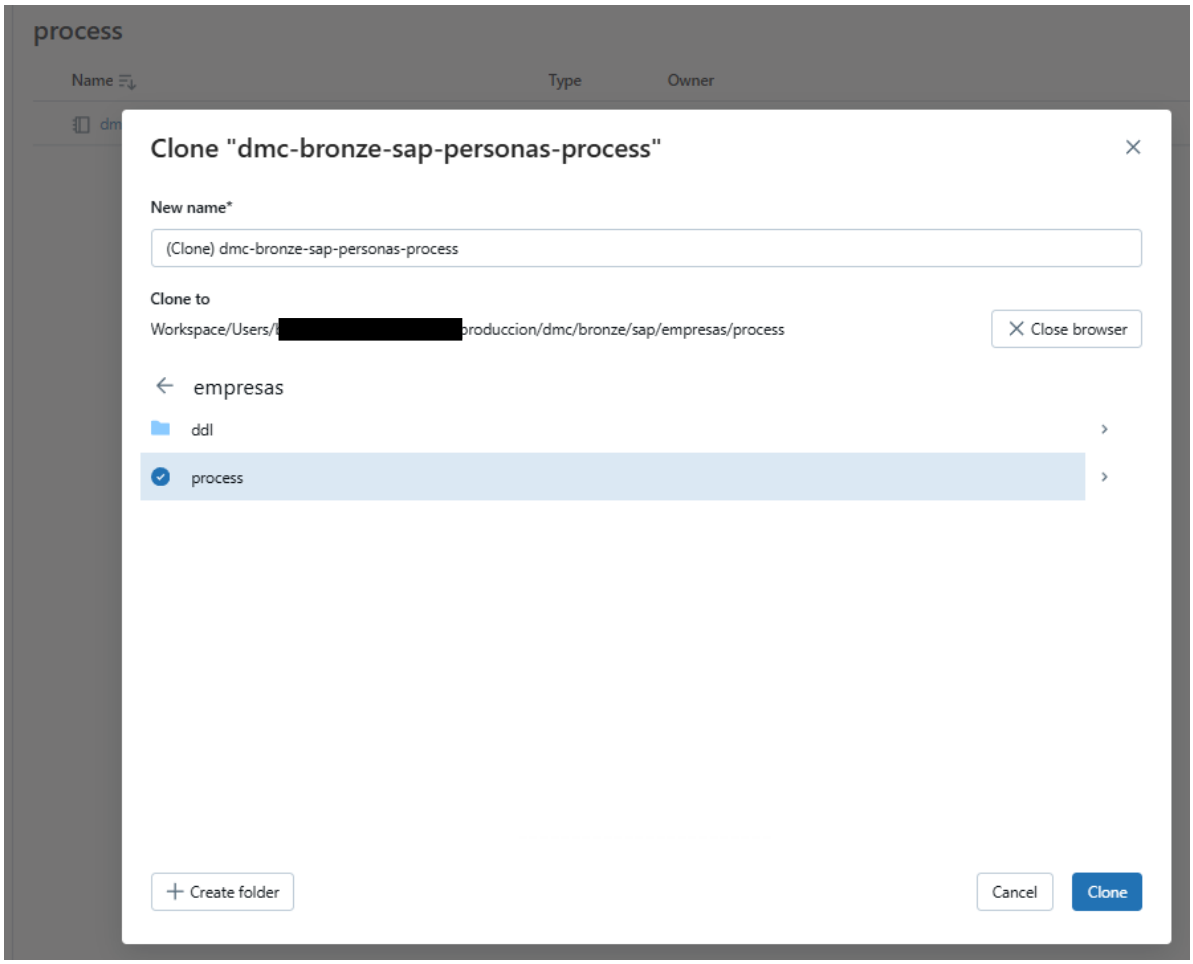
Guardando datos de la capa bronce en formato delta en el bucket de gcp. La tabla personas.



Validando en Google Cloud Storage que se creó la carpeta y la data.



Clonando archivos en databricks



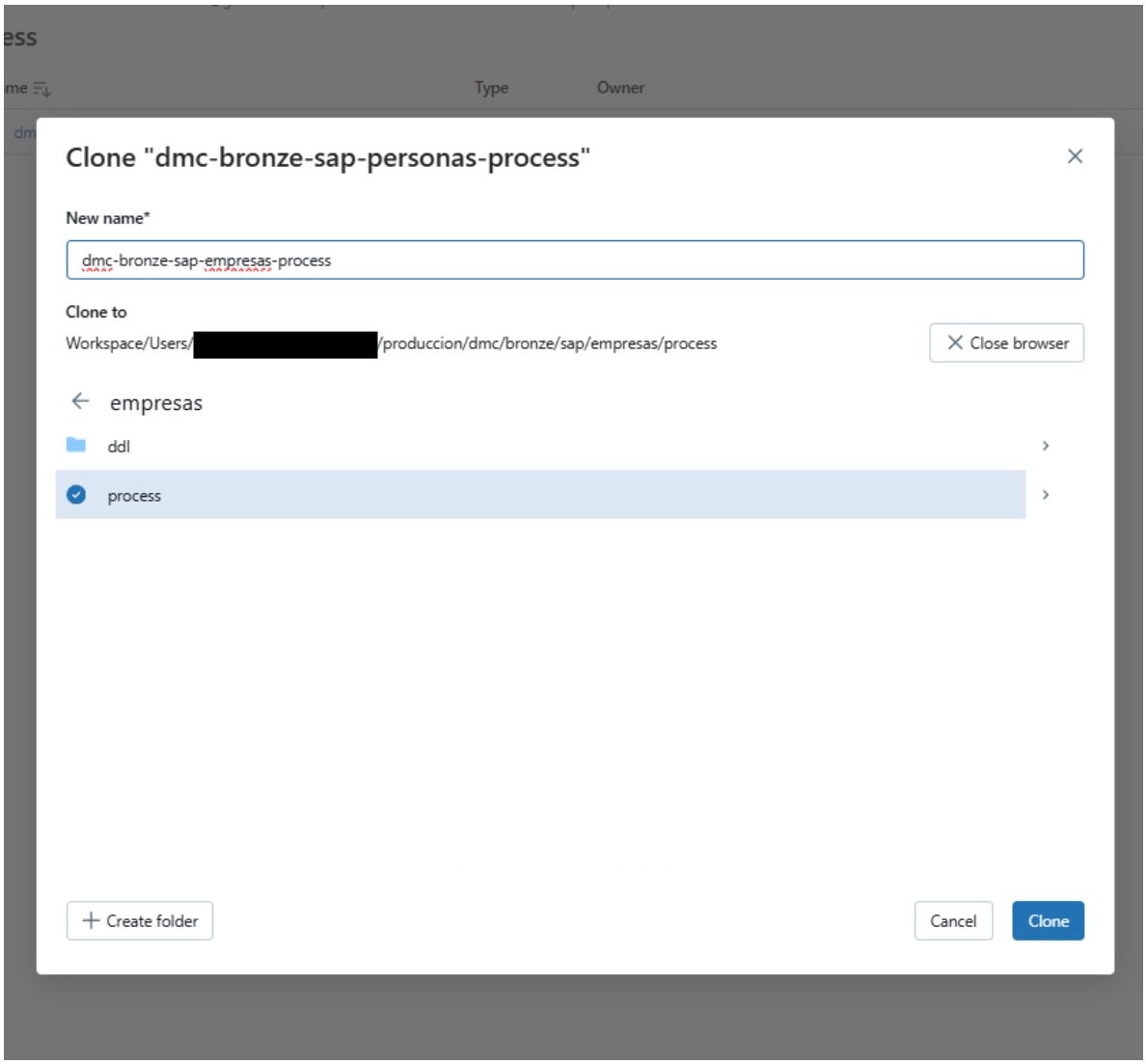

ess
me
Type
Owner
Clone "dmc-bronze-sap-personas-process"
New name*
dmc-bronze-sap-empresas-process
Clone to
Workspace/Users/[redacted]/produccion/dmc/bronze/sap/empresas/process
Close browser
empresas
ddl
process
Create folder
Cancel
Clone

Tabla empresas a la capa bronce

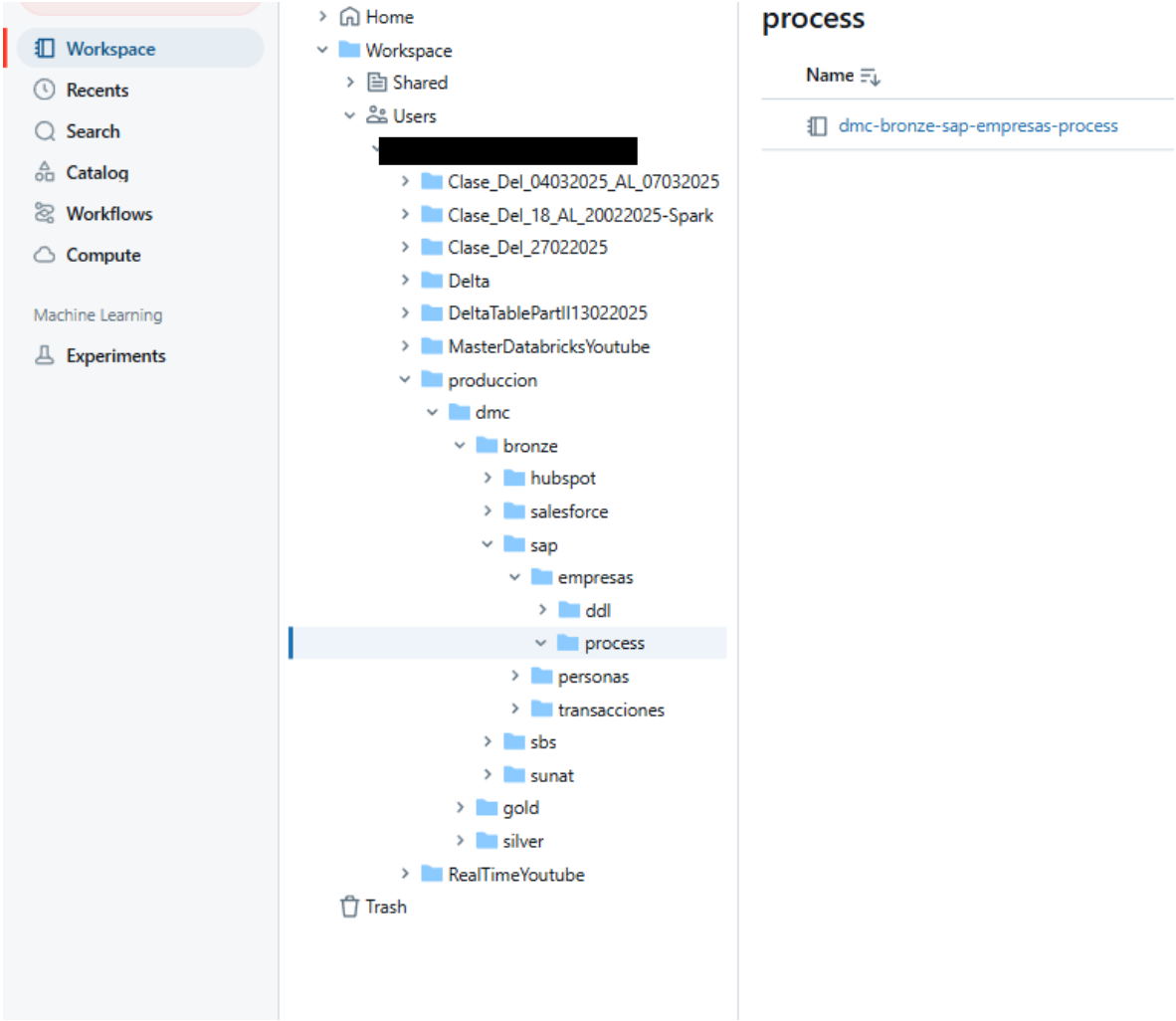
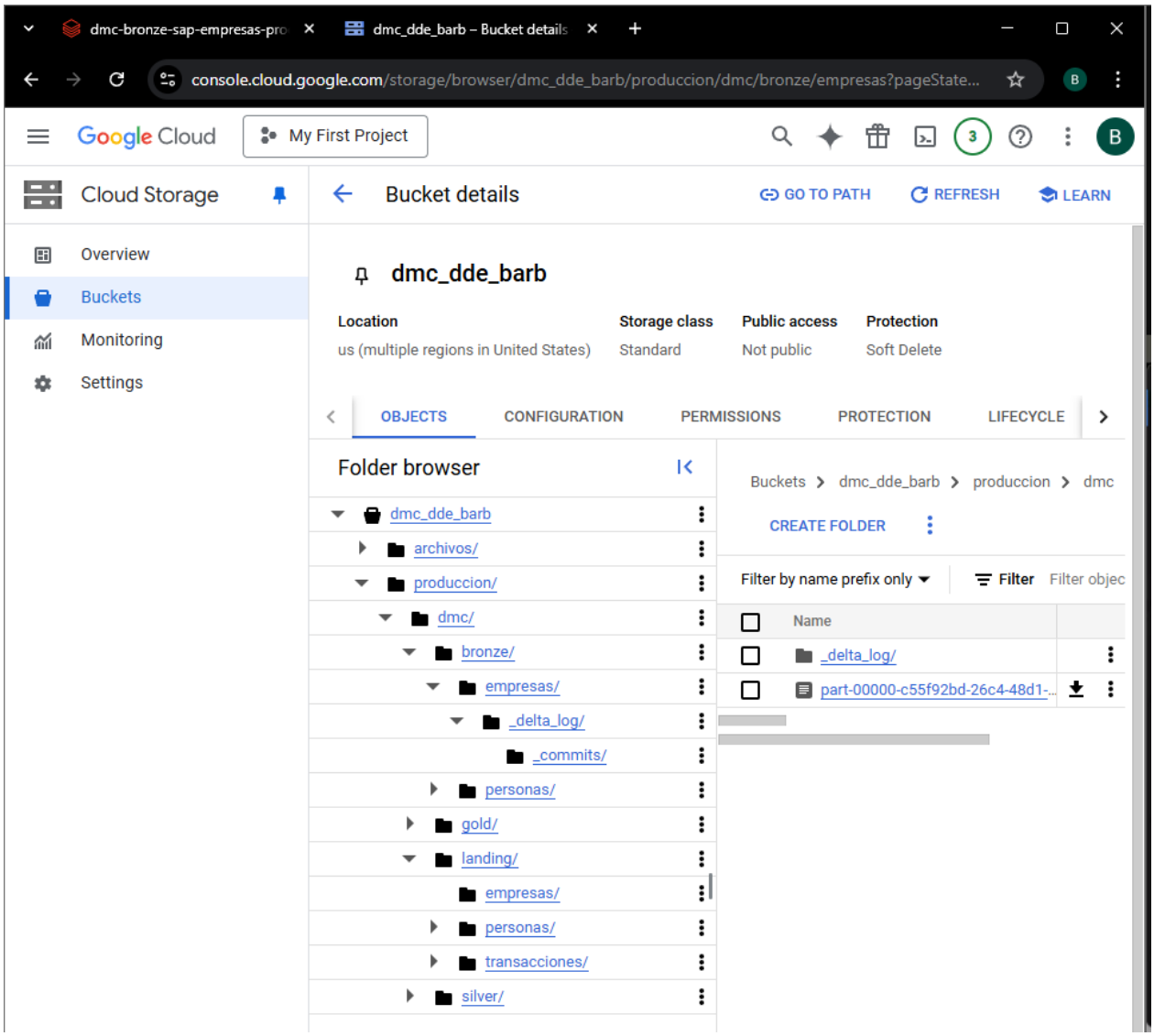
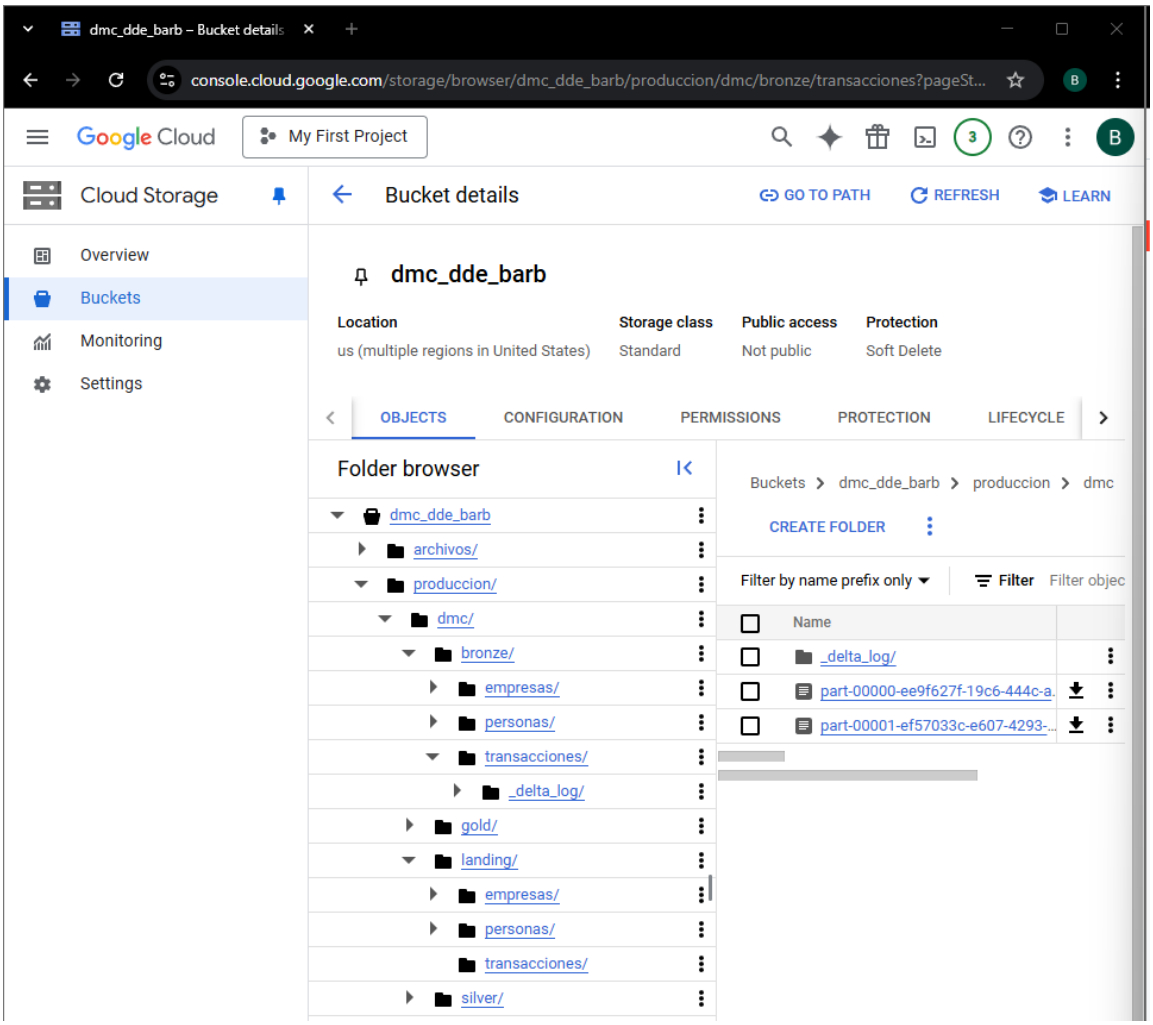
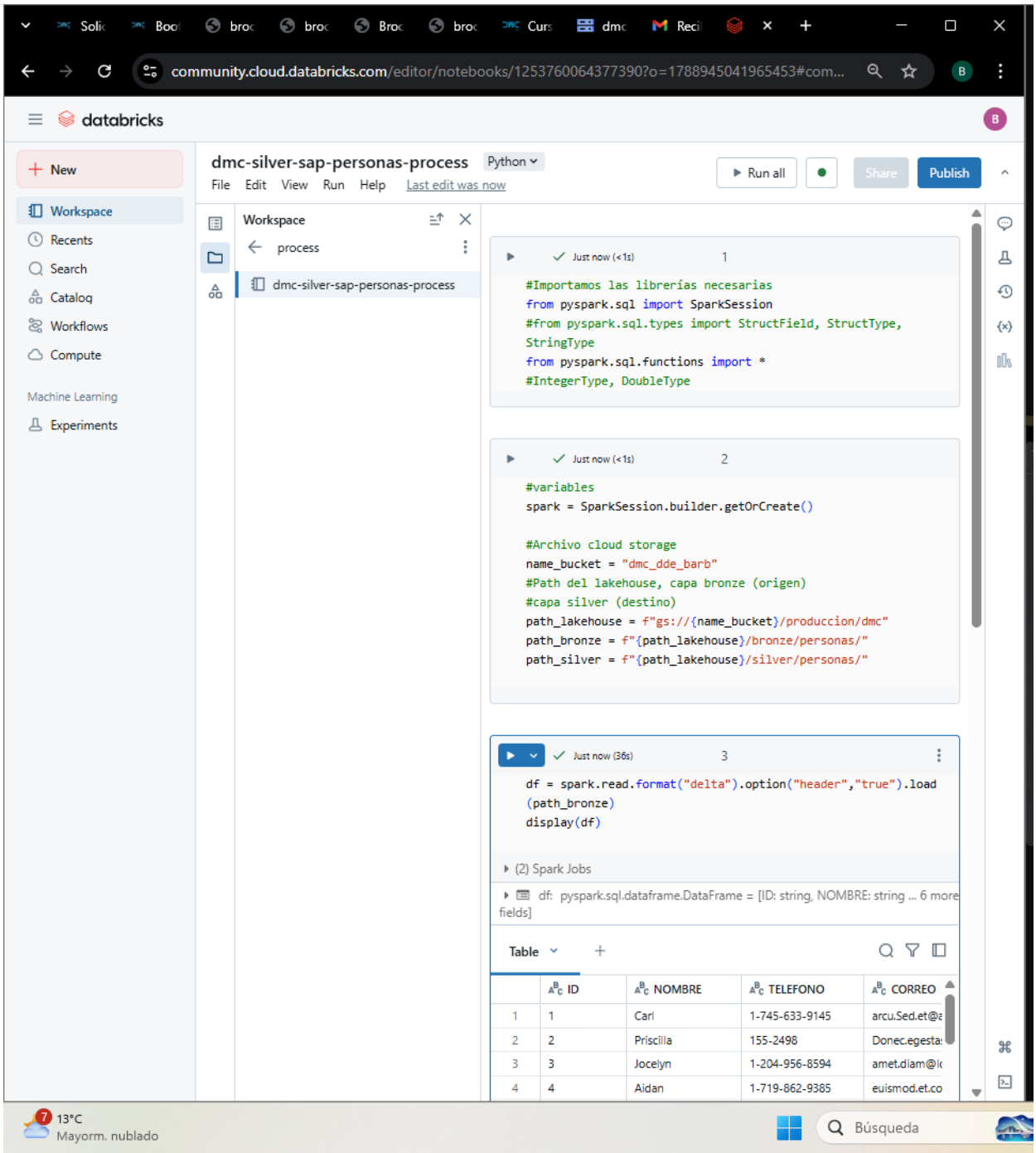


Tabla transacciones a la capa bronce

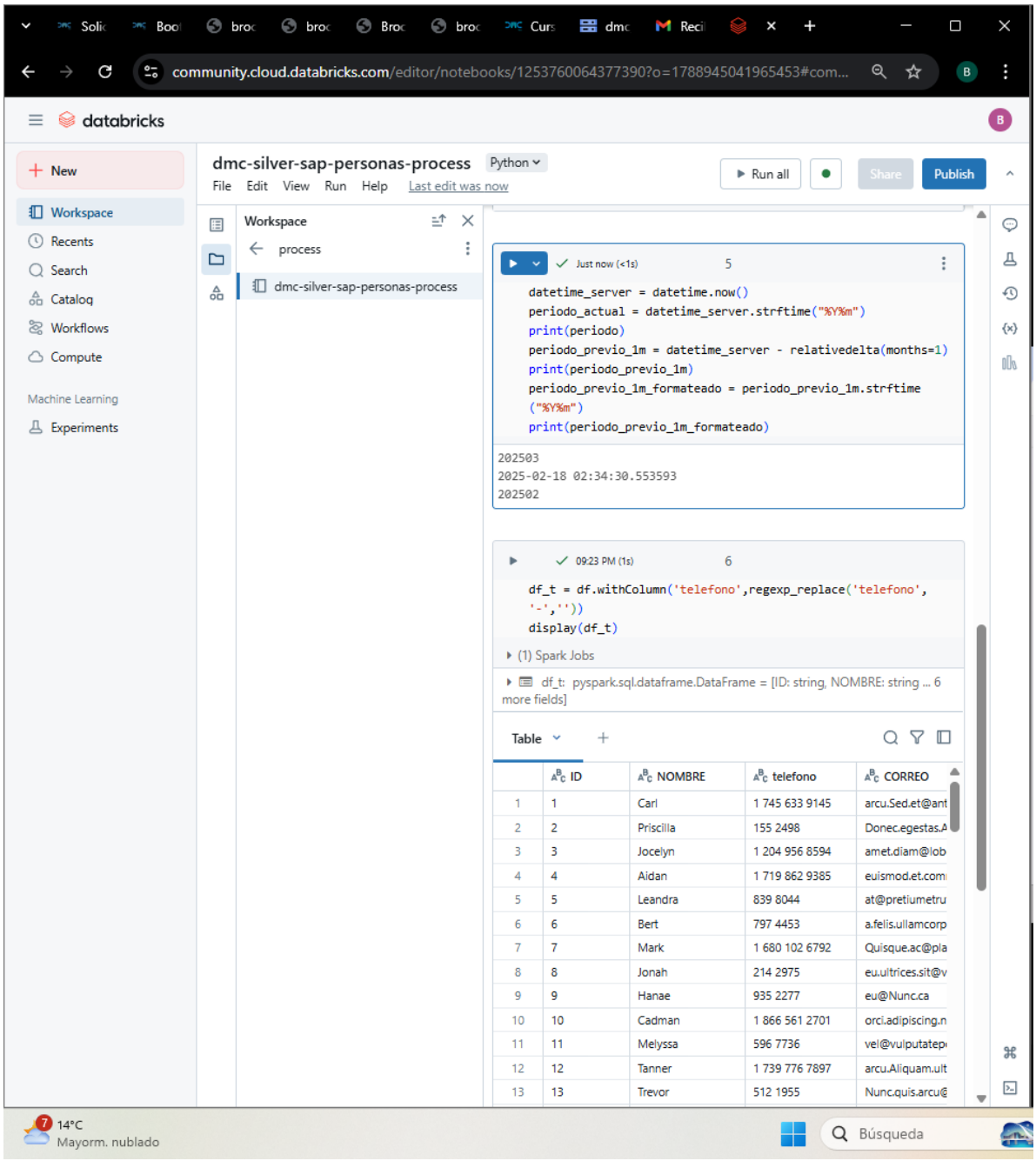



dmc_dde_barb – Bucket details
console.cloud.google.com/storage/browser/dmc_dde_barb/produccion/dmc/bronze/transacciones?pageSt...
Google Cloud
My First Project
Cloud Storage
Overview
Buckets
Monitoring
Settings
Bucket details
GO TO PATH
REFRESH
LEARN
dmc_dde_barb
Location
us (multiple regions in United States)
Storage class
Standard
Public access
Not public
Protection
Soft Delete
OBJECTS
CONFIGURATION
PERMISSIONS
PROTECTION
LIFECYCLE
Folder browser
dmc_dde_barb
archivos/
produccion/
dmc/
bronze/
empresas/
personas/
transacciones/
_delta_log/
gold/
landing/
empresas/
personas/
transacciones/
silver/
Buckets > dmc_dde_barb > produccion > dmc
CREATE FOLDER
Filter by name prefix only
Filter
Filter objec
Name
_delta_log/
part-00000-ee9f627f-19c6-444c-a...
part-00001-ef57033c-e607-4293-...

La Capa Silver



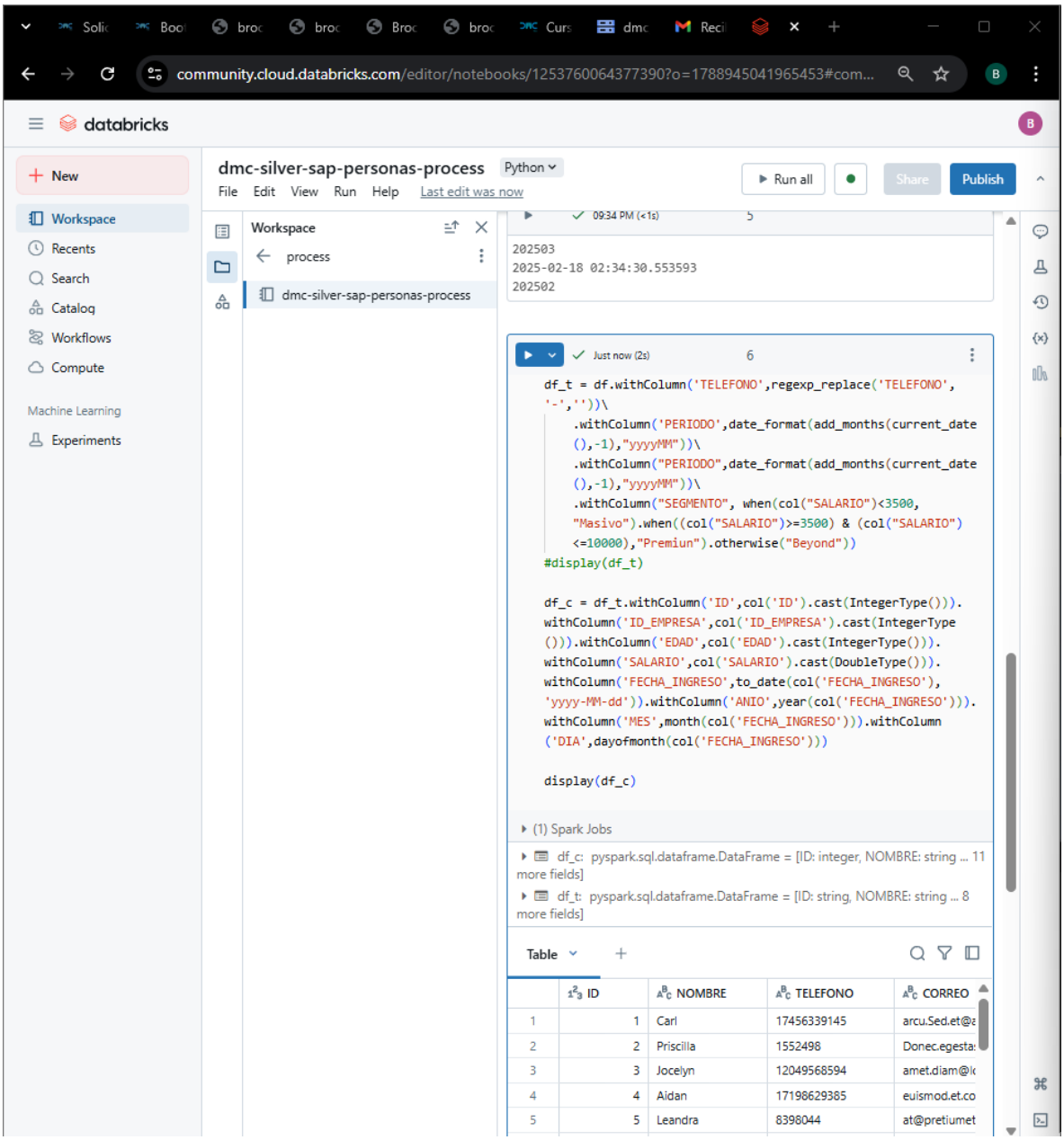
Limpieza y casteo de datos



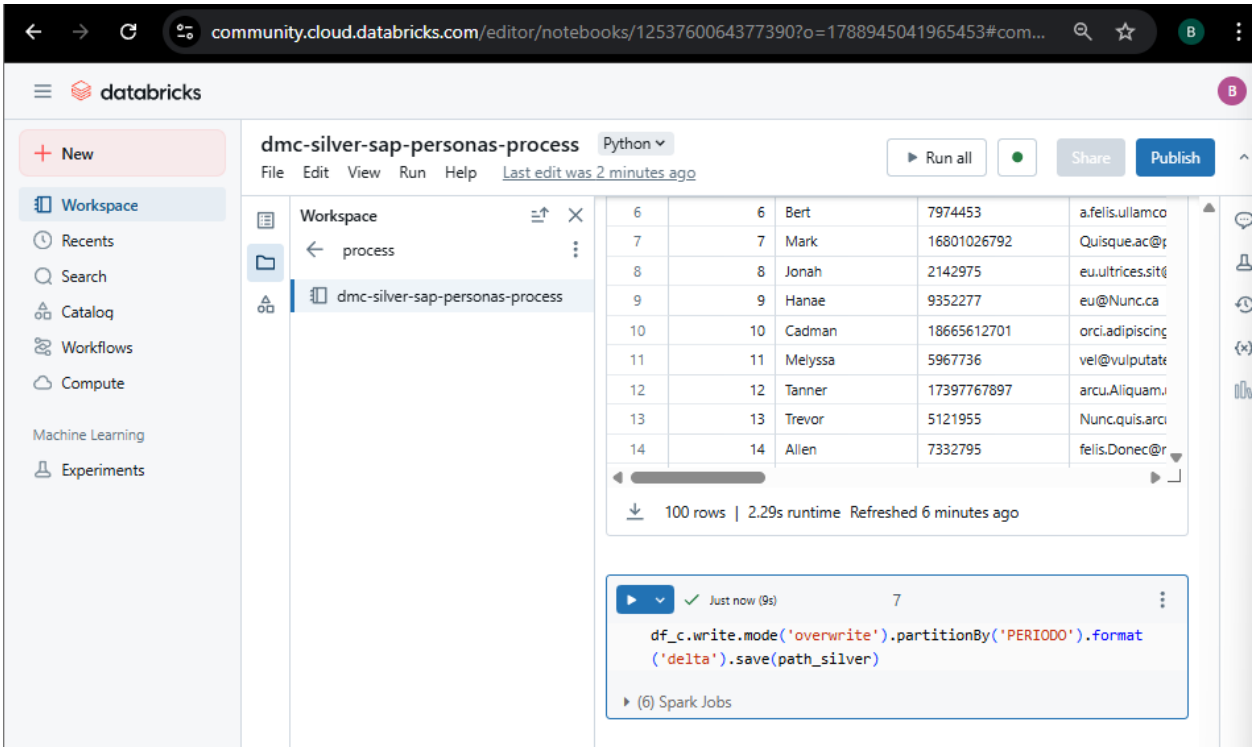
09:52 PM (<1s) 1

```
#Importamos las librerías necesarias
from pyspark.sql import SparkSession
#from pyspark.sql.types import StructField, StructType,
StringType
from pyspark.sql.functions import regexp_replace,
date_format, current_date, add_months, when, col
from pyspark.sql.types import IntegerType, DoubleType
from datetime import datetime
from dateutil.relativedelta import relativedelta
```

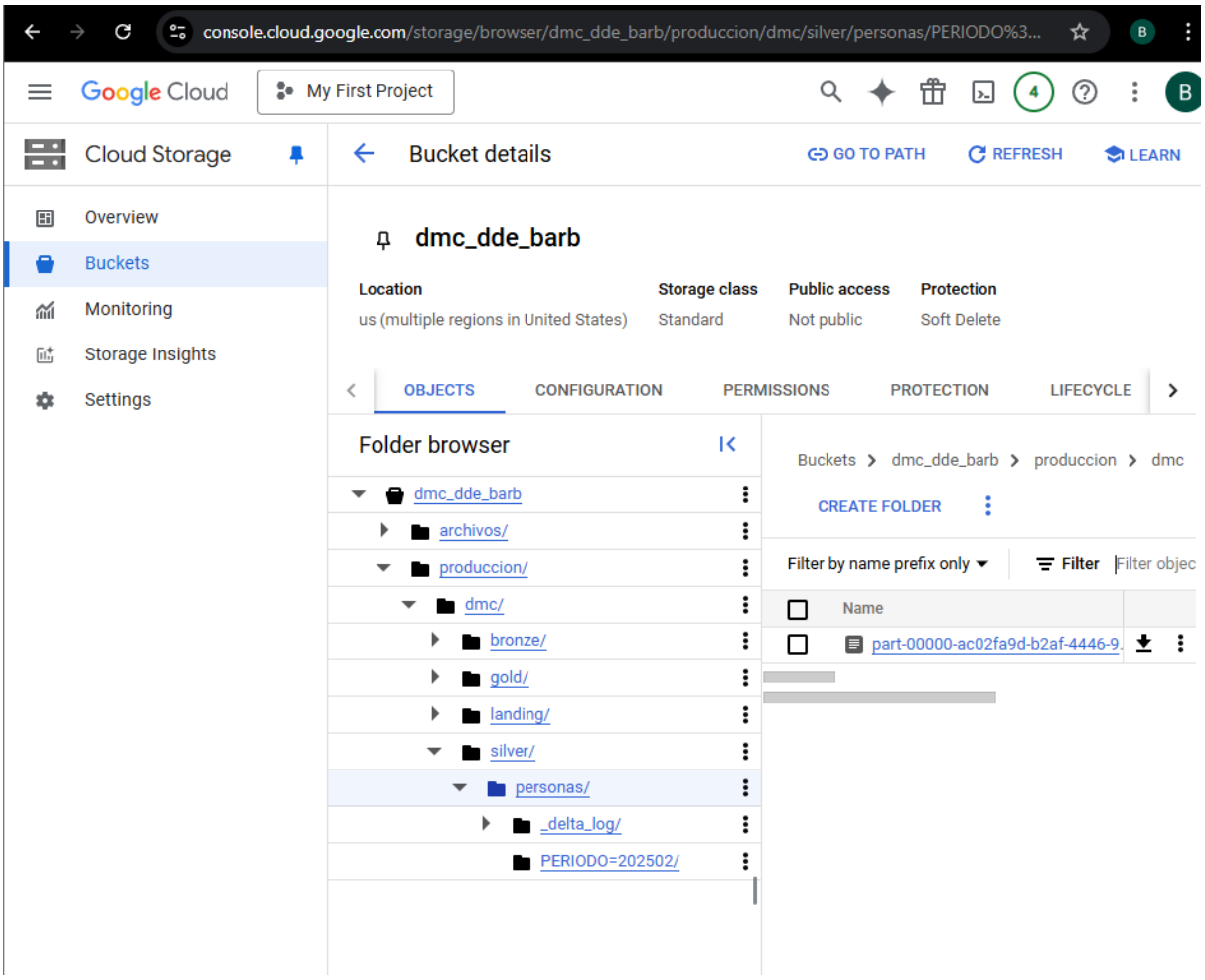
Añadiendo columnas para enriquecer la data



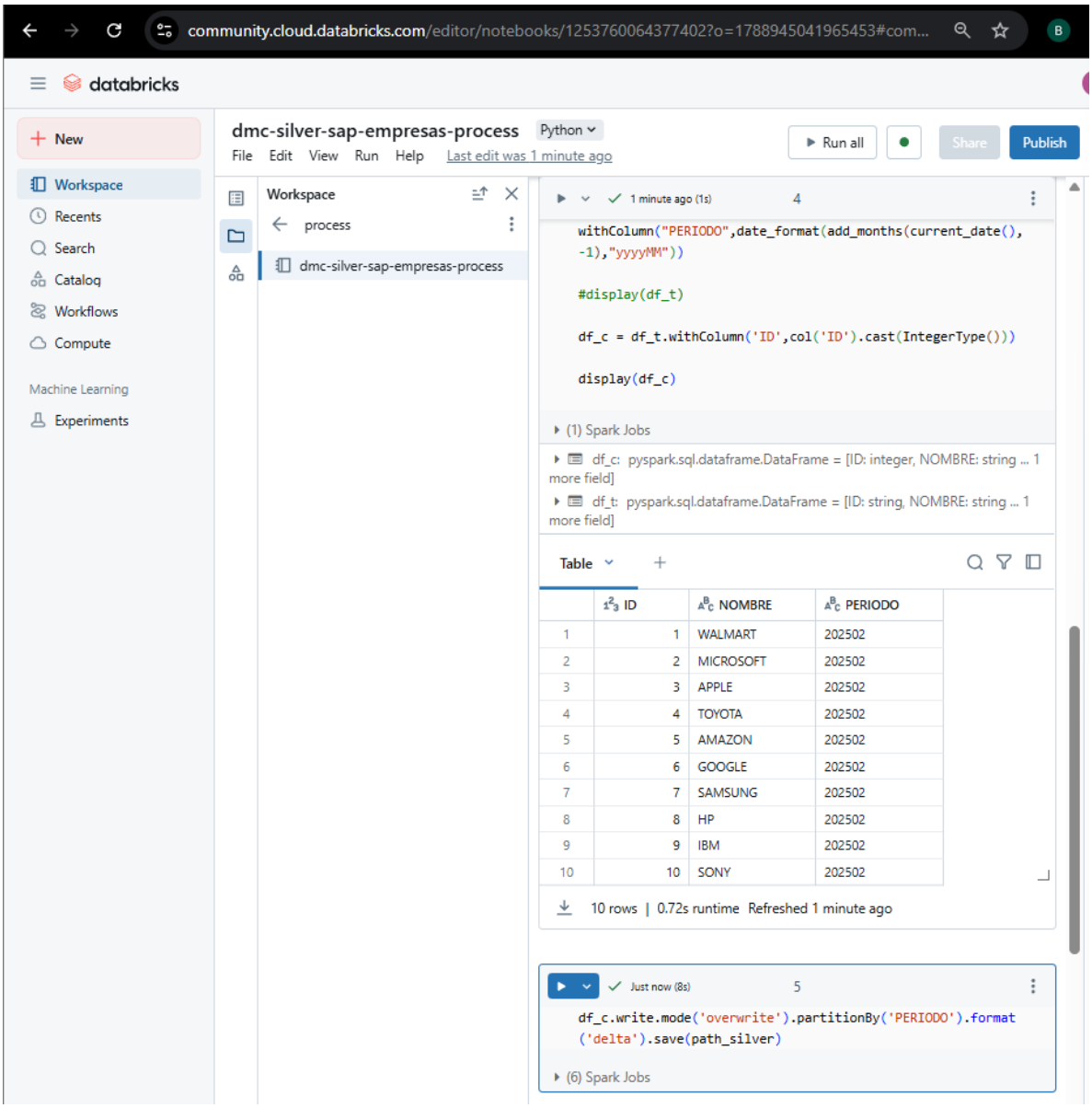
Creación de partición en la tabla personas



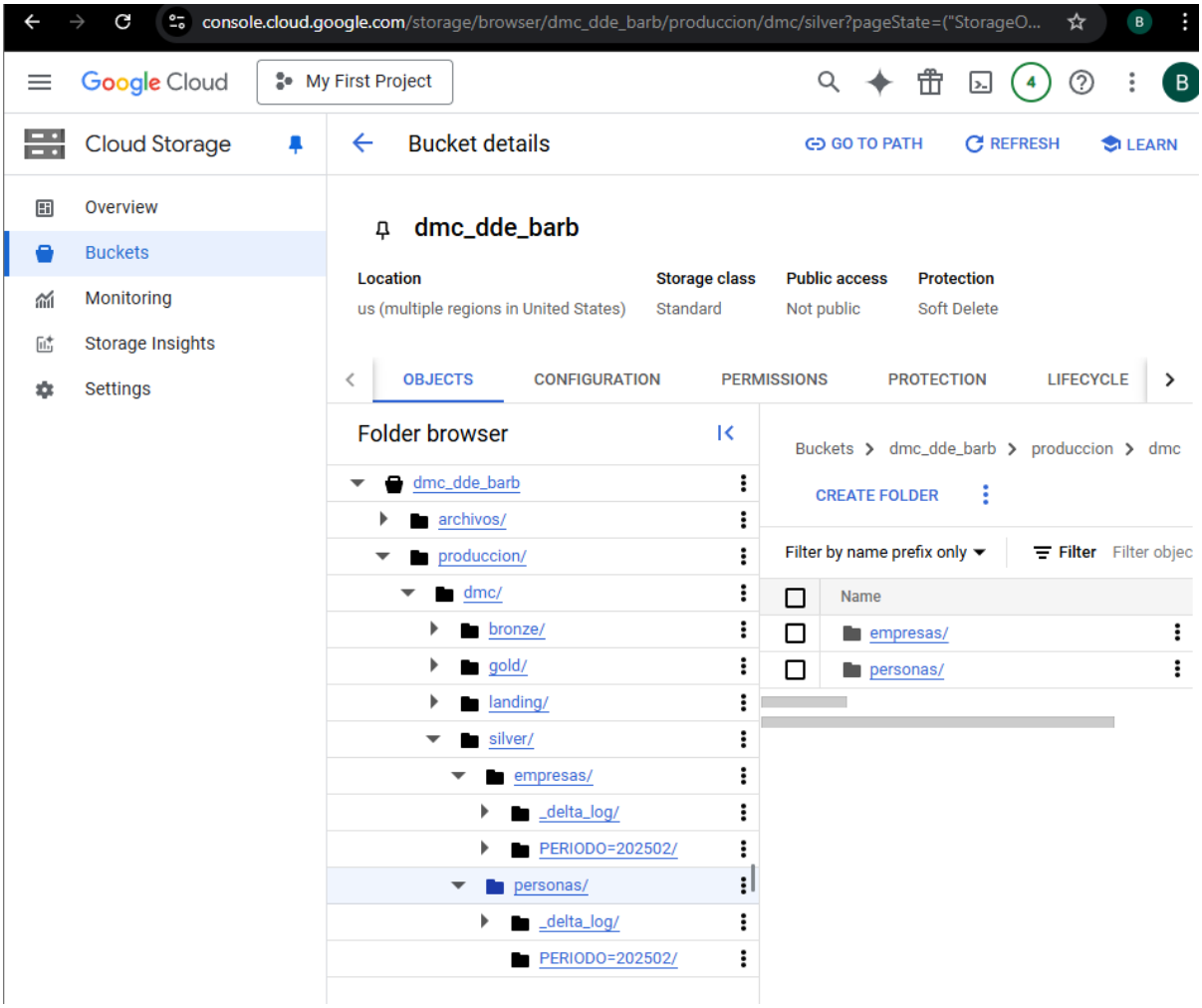
Corroborando data en el bucket



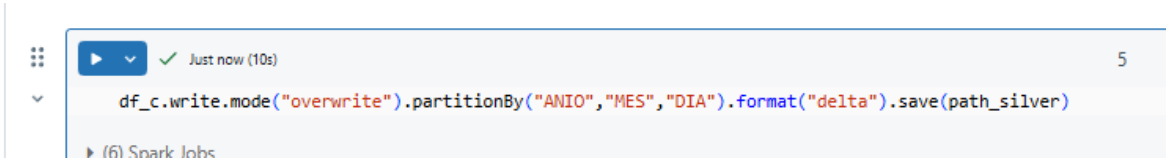
Realizando casteo de datos en la tabla empresas



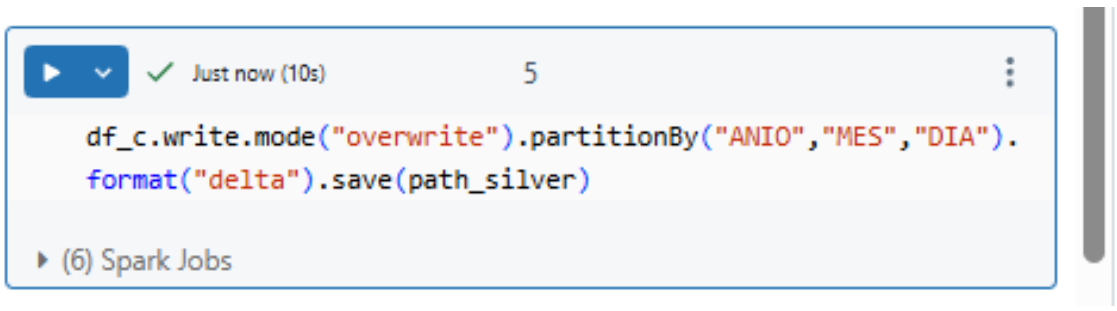
Validando creación de la tabla en el bucket



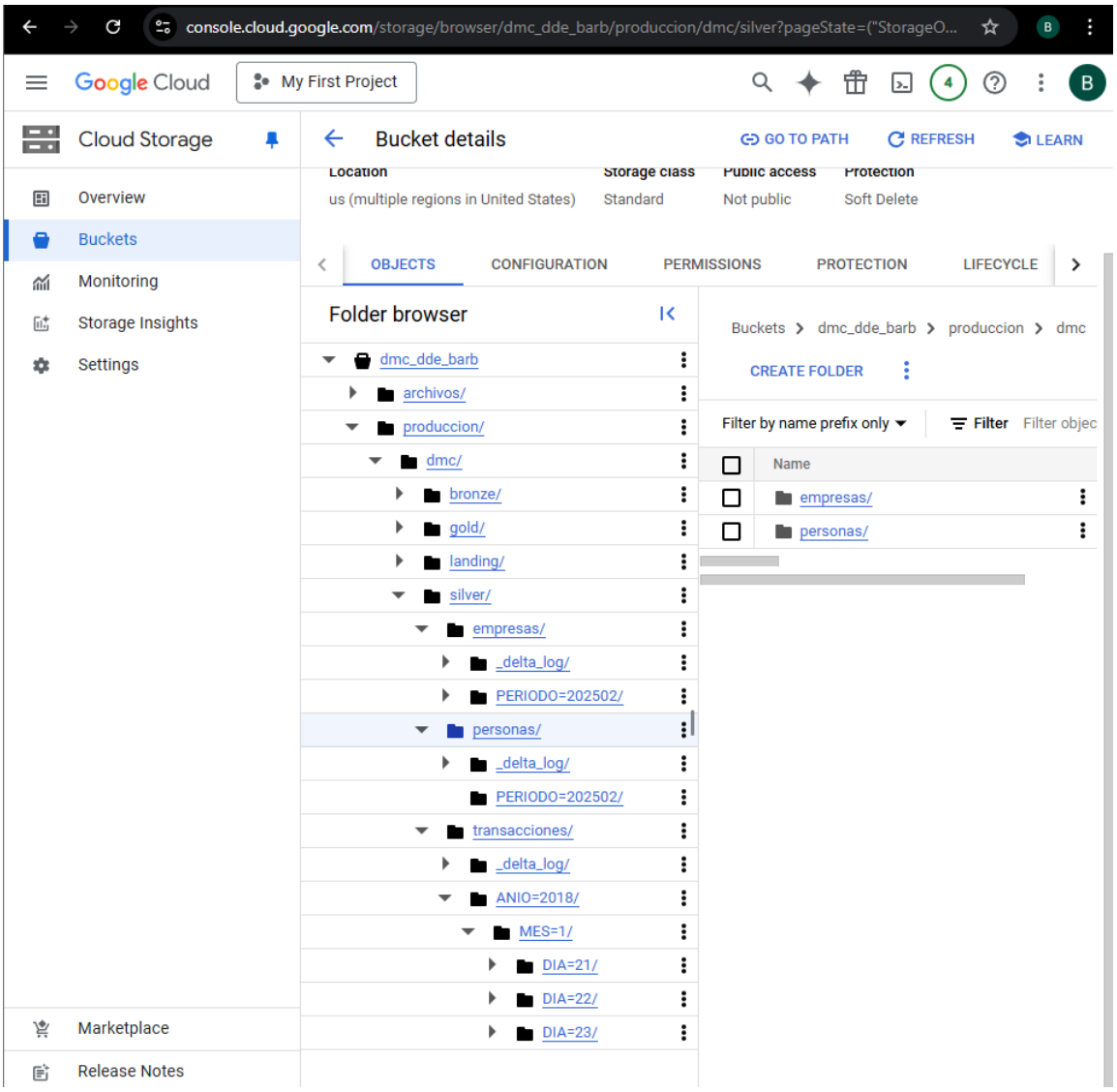
Añadiendo partición a la tabla transacciones y columnas nuevas



Guardando tabla en formato delta en el bucket



Validando data en el bucket



Poblando la capa Gold

io-process Python

: 2 hours ago

09:06 PM (<1s)

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *
```

09:05 PM (<1s)

```
spark = SparkSession.builder.getOrCreate()
```

09:05 PM (<1s)

```
name_bucket = "dmc_dde_barb"
path_lakehouse = f"gs://{name_bucket}/produccion/dmc"
path_silver_personas = f"{path_lakehouse}/silver/personas/"
path_silver_empresas = f"{path_lakehouse}/silver/empresas/"
path_gold = f"{path_lakehouse}/gold/machine-learning/analisis-x-salario"
```

09:09 PM (4s)

```
df_personas = spark.read.format("delta").load(path_silver_personas)
display(df_personas)

df_empresas = spark.read.format("delta").load(path_silver_empresas)
display(df_empresas)
```

(3) Spark Jobs

df_empresas: pyspark.sql.dataframe.DataFrame = [ID: integer, NOMBRE: string ... 1 more field]

df_personas: pyspark.sql.dataframe.DataFrame = [ID: integer, NOMBRE: string ... 11 more fields]

Table

	ID	NOMBRE	TELEFONO	CORREO	FECHA_IN
1	1	Carl	17456339145	arcu.Sed.et@ante.co.uk	2004-04-23
2	2	Priscilla	1552498	Donec.egestas.Aliquam@volutpatnunc.edu	2019-02-17
3	3	Jocelyn	12049568594	amet.diam@lobortis.co.uk	2002-08-01
4	4	Aidan	17198629385	euismod.et.commodo@nibhlaciniaorci.edu	2018-11-06
5	5	Leandra	8398044	at@pretiumetrutrum.com	2002-10-10
6	6	Bert	7974453	a.felis.ullamcorper@arcu.org	2017-04-25
7	7	Mark	16801026792	Quisque.ac@placerat.ca	2006-04-21
8	8	Jonah	2142975	eu.ultrices.sit@vitae.ca	2017-10-07
9	9	Hanae	9352277	eu@Nunc.ca	2003-05-25
10	10	Cadman	18665612701	orci.adipiscing.non@semperNam.ca	2001-05-19

Python

09:09 PM (4s)

```
df_personas = spark.read.format("delta").load(path_silver_personas)
display(df_personas)

df_empresas = spark.read.format("delta").load(path_silver_empresas)
display(df_empresas)
```

(3) Spark Jobs

df_empresas: pyspark.sql.dataframe.DataFrame = [ID: integer, NOMBRE: string ... 1 more field]

df_personas: pyspark.sql.dataframe.DataFrame = [ID: integer, NOMBRE: string ... 11 more fields]

Table

	ID	NOMBRE	TELEFONO	CORREO
6	6	Bert	7974453	a.felis.ullamcorper@arcu.org
7	7	Mark	16801026792	Quisque.ac@placerat.ca
8	8	Jonah	2142975	eu.ultrices.sit@vitae.ca
9	9	Hanae	9352277	eu@Nunc.ca
10	10	Cadman	18665612701	orci.adipiscing.non@semperNam.ca
11	11	Melyssa	5967736	vel@vulputateposuerevulputate.net
12	12	Tanner	17397767897	arcu.Aliquam.ultrices@sociis.com
13	13	Trevor	5121955	Nunc.quis.arcu@egestasa.org
14	14	Allen	7332795	felis.Donec@necleo.org
15	15	Wanda	3596973	Nam.nulla.magna@In.org
16	16	Alden	3418522	odio@morbitristiquesenectus.ca
17	17	Omar	7201543	Phasellus.vitae.mauris@sollicitudin.net
18	18	Owen	11673357541	sociis@erat.com
19	19	Laura	19746232057	mollis@ornare.ca

100 rows | 4.17s runtime

Table

	ID	NOMBRE	PERIODO
1	1	WALMART	202502
2	2	MICROSOFT	202502
3	3	APPLE	202502
4	4	TOYOTA	202502
5	5	AMAZON	202502
6	6	GOOGLE	202502

Para hacer consultas sobre sql sobre los datos de un dataframe es necesario crear vistas temporales a partir de dichos dataframes.

```
df_personas.createOrReplaceTempView("tb_personas")
df_empresas.createOrReplaceTempView("tb_empresas")
```

Luego de la ejecución de la consulta se pueden almacenar en otro dataframe de la siguiente forma:

```
sql= """ SELECT p.periodo,e.empresa_name as nombre_empresa, p.salario,p.edad
     FROM tb_personas p
     inner join tb_empresas e on e.ID = p.ID_EMPRESA and e.periodo=p.periodo
    order by P.ID_EMPRESA"""

df_result_1 = spark.sql(sql)
```

9	9	IBM	202502
10	10	SONY	202502

10 rows | 4.17s runtime

09:05 PM (5s)

```
df_personas.createOrReplaceTempView("tb_personas")
df_empresas.createOrReplaceTempView("tb_empresas")
```

09:27 PM (2s)

```
sql = """SELECT p.periodo,e.nombre as nombre_empresa, p.salario,p.edad
        FROM tb_personas p
        INNER JOIN tb_empresas e ON e.ID = p.ID_EMPRESA
        ORDER BY p.id_empresa"""
df_result_1 = spark.sql(sql)
display(df_result_1)
```

(2) Spark Jobs

df_result_1: pyspark.sql.dataframe.DataFrame = [periodo: string, nombre_empresa: string ... 2 mor

Table

	periodo	nombre_empresa	salario	edad
6	202502	WALMART	8818	24
7	202502	WALMART	5570	26
8	202502	MICROSOFT	9298	34
9	202502	MICROSOFT	16289	59
10	202502	MICROSOFT	3377	26
11	202502	MICROSOFT	7449	22
12	202502	MICROSOFT	6483	39
13	202502	MICROSOFT	22038	42
14	202502	MICROSOFT	21452	61
15	202502	MICROSOFT	10825	27
16	202502	MICROSOFT	22953	46
17	202502	MICROSOFT	15116	38

09:27 PM (<1s) 7

```
df_result_1.createOrReplaceTempView("tb_analisis_detalle")
```

09:27 PM (4s) 8

```
sql_final = """ select periodo
,nombre_empresa,sum(salario) as sum_salario,avg(salario) as avg_salario, avg(edad) as avg_edad from tb_analisis_detalle group by periodo
,nombre_empresa
"""
df_result_final = spark.sql(sql_final)
display(df_result_final)
```

(5) Spark Jobs

df_result_final: pyspark.sql.dataframe.DataFrame = [periodo: string, nombre_empresa: string ... 3 more fields]

Table +

	periodo	nombre_empresa	sum_salario	avg_salario	avg_edad
1	202502	AMAZON	136609	9757.785714285714	41.2142857142857...
2	202502	SONY	82012	9112.444444444445	40.8888888888888...
3	202502	TOYOTA	155503	19437.875	38.875
4	202502	MICROSOFT	156377	11169.785714285714	39.7857142857142...
5	202502	HP	73319	8146.555555555556	39.8888888888888...
6	202502	WALMART	79304	11329.142857142857	35.8571428571428...
7	202502	IBM	91678	15279.666666666666	37.6666666666666...
8	202502	SAMSUNG	106710	11856.666666666666	34.55555555555556
9	202502	GOOGLE	135243	10403.307692307691	50
10	202502	APPLE	151700	13790.90909090909	39.63636363636363

10 rows | 4.19s runtime

09:27 PM (18s) 9

```
df_result_final.write.mode("overwrite").format("delta").save(path_gold)
```

(10) Spark Jobs

09:28 PM (2s) 10

```
df = spark.read.format("delta").load(path_gold)
display(df)
```

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame

Schema Details History

09:27 PM (18s)

```
df_result_final.write.mode("overwrite").format("delta").save(path_gold)
```

(10) Spark Jobs

09:28 PM (2s)

```
df = spark.read.format("delta").load(path_gold)
display(df)
```

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame

Schema Details History

```
periodo: string
nombre_empresa: string
sum_salario: double
avg_salario: double
avg_edad: double
```

Table

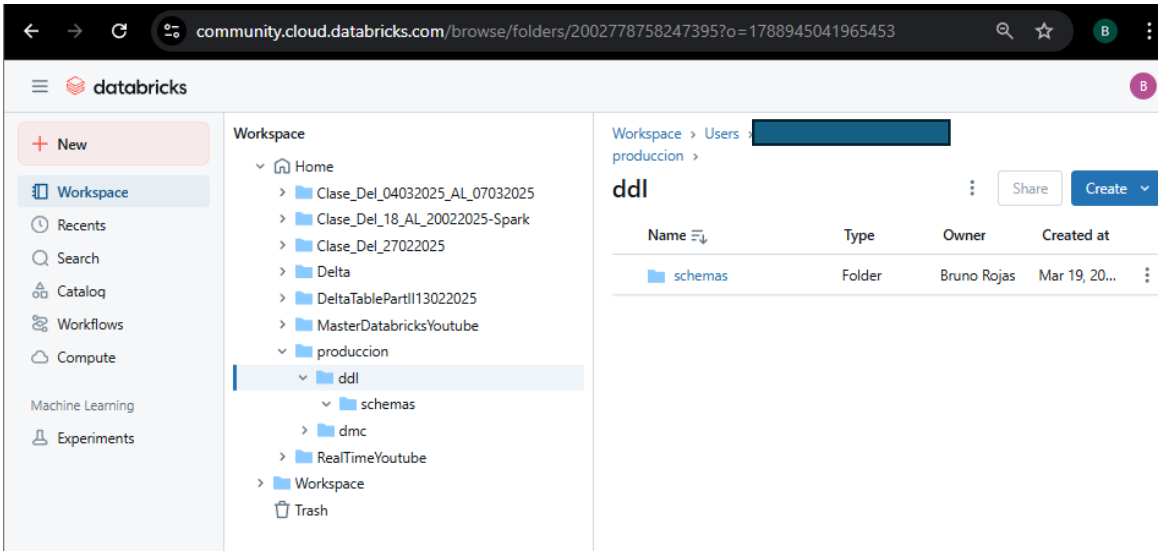
	periodo	nombre_empresa	sum_salario	avg_salario	avg_edad
1	202502	AMAZON	136609	9757.785714285714	41.2142857142857...
2	202502	SONY	82012	9112.444444444445	40.8888888888888...
3	202502	TOYOTA	155503	19437.875	38.875
4	202502	MICROSOFT	156377	11169.785714285714	39.7857142857142...
5	202502	HP	73319	8146.555555555556	39.8888888888888...
6	202502	WALMART	79304	11329.142857142857	35.8571428571428...
7	202502	IBM	91678	15279.666666666666	37.6666666666666...
8	202502	SAMSUNG	106710	11856.666666666666	34.55555555555556
9	202502	GOOGLE	135243	10403.307692307691	50
10	202502	APPLE	151700	13790.90909090909	39.63636363636363

10 rows | 1.77s runtime

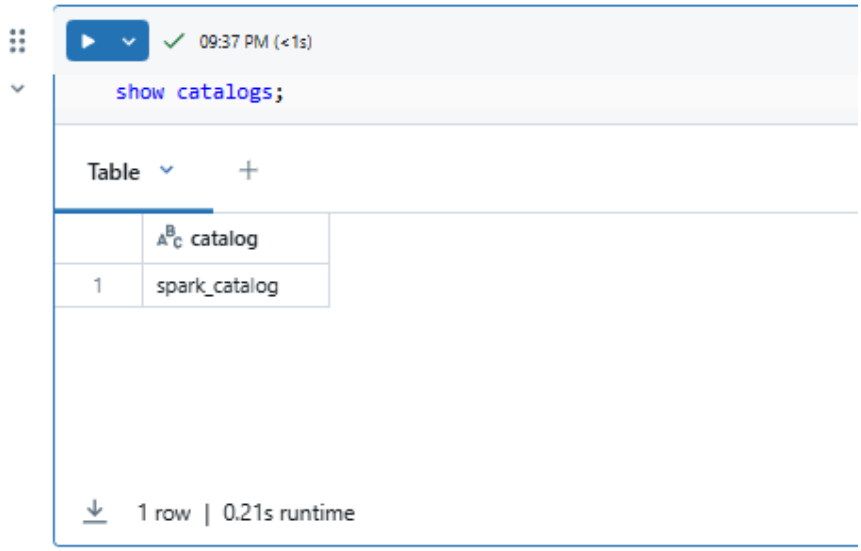
CATALOGO DE DATOS – METADA REFERENCIA A ARCHIVOS ALMACENADOS (INTERNA O EXTERNAMENTE)

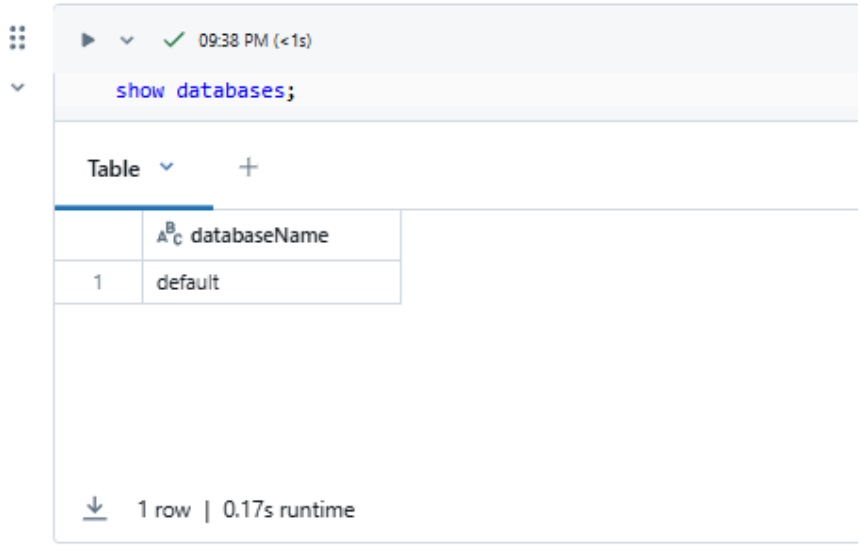
Brinda posibilidad de ejecutar consultas sql.

En el caso del community solo podemos alcanzar el nivel de bases de datos (schemas) y tablas (también vistas).

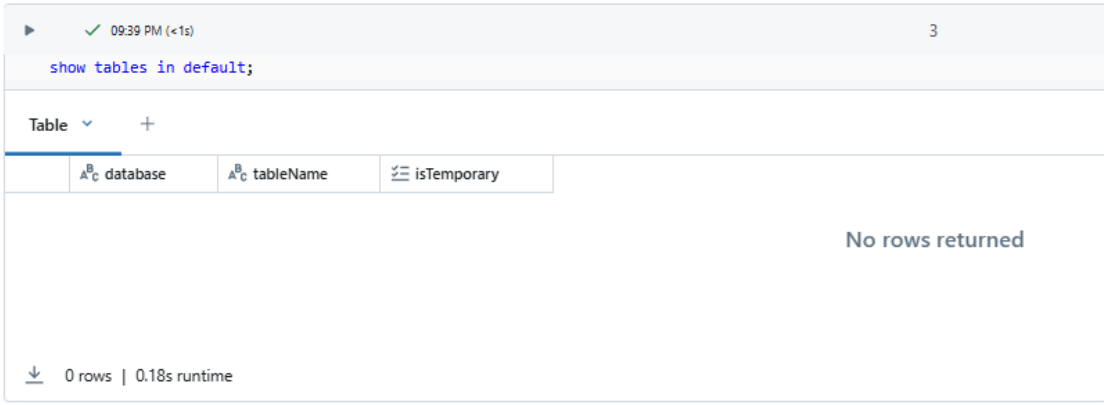


En la carpeta producción->ddl creamos la carpeta schemas. Dentro creamos un notebook de tipo de lenguaje SQL. Usamos sentencias para listar los catálogos, las bases de datos y las tablas dentro de las bases de datos existentes.



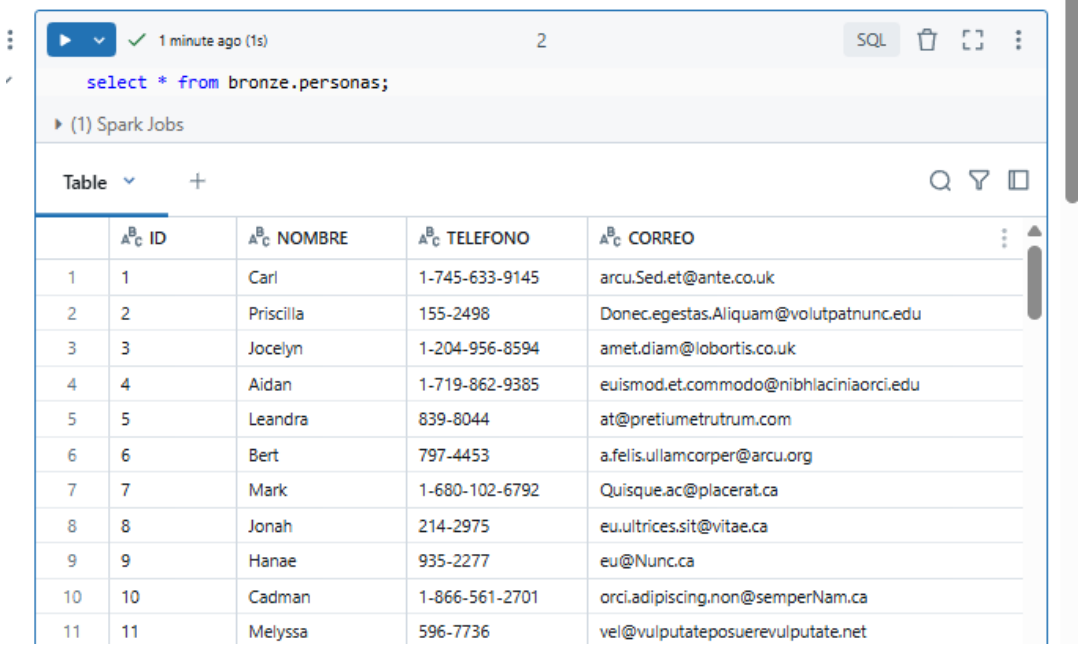


Creamos los esquemas para nuestras tres capas: bronze, silver, gold



En la carpeta ddl de cada tabla creamos el esquema de la tabla. Empezando por la capa bronze.

Las tablas creadas son de tipo external y delta. Por ser de tipo delta basta con indicar el location para que se considere como external aunque no se asigne la palabra external en la creación de la tabla.



	ID	NOMBRE	TELEFONO	CORREO
1	1	Carl	1-745-633-9145	arcu.Sed.et@ante.co.uk
2	2	Priscilla	155-2498	Donec.egestas.Aliquam@volutpatnunc.edu
3	3	Jocelyn	1-204-956-8594	amet.diam@lobortis.co.uk
4	4	Aidan	1-719-862-9385	euismod.et.commodo@nibhlaciniaorci.edu
5	5	Leandra	839-8044	at@pretiumetrutrum.com
6	6	Bert	797-4453	a.felis.ullamcorper@arcu.org
7	7	Mark	1-680-102-6792	Quisque.ac@placerat.ca
8	8	Jonah	214-2975	eu.ultrices.sit@vitae.ca
9	9	Hanae	935-2277	eu@Nunc.ca
10	10	Cadman	1-866-561-2701	orci.adipiscing.non@semperNam.ca
11	11	Melyssa	596-7736	vel@vulputateposuerevulputate.net

Just now (5s) 1

```
create external table bronze.empresas(
  ID STRING,
  NOMBRE STRING
)
USING delta
location 'gs://dmc_dde_barb/produccion/dmc/bronze/empresas'
```

(3) Spark Jobs

OK

Just now (1s) 2 SQL

```
SELECT * FROM bronze.empresas;
```

(2) Spark Jobs

Table

	ID	NOMBRE
1	1	Walmart
2	2	Microsoft
3	3	Apple
4	4	Toyota
5	5	Amazon
6	6	Google
7	7	Samsung
8	8	HP
9	9	IBM
10	10	Sony

10 rows | 1.19s runtime Refreshed now

10:03 PM (5s) 1

```
CREATE EXTERNAL TABLE bronze.transacciones(
  ID_PERSONA STRING,
  ID_EMPRESA STRING,
  MONTO STRING,
  FECHA STRING
)
USING DELTA
LOCATION 'gs://dmc_dde_barb/produccion/dmc/bronze/
transacciones'
```

(3) Spark Jobs

OK

10:03 PM (2s) 2

```
select * from bronze.transacciones;
```

(2) Spark Jobs

Table

	ID_PERSONA	ID_EMPRESA	MONTO	
23	83	5	2079	20
24	48	4	2543	20
25	15	6	1434	20
26	89	4	780	20
27	4	2	863	20
28	22	4	2058	20
29	10	3	2027	20
30	48	1	3833	20
31	66	1	3847	20
32	87	2	3581	20
33	43	2	3310	20
34	71	10	4028	20
35	74	7	3360	20
36	57	7	3331	20

Iniciamos con la creación de la definición de las tablas para la capa silver.

Just now (1s)

```
CREATE EXTERNAL TABLE IF NOT EXISTS silver.personas(
    ID INT,
    ID_EMPRESA INT,
    NOMBRE STRING,
    EDAD INT,
    TELEFONO STRING,
    CORREO STRING,
    SALARIO DOUBLE,
    FECHA_INGRESO DATE,
    ANIO INT,
    MES INT,
    DIA INT,
    SEGMENTO STRING,
    PERIODO STRING
)
USING DELTA
PARTITIONED BY (PERIODO)
LOCATION 'gs://dmc_dde_barb/produccion/dmc/silver/personas';
```

OK

Just now (1s)

```
SELECT * FROM silver.personas;
```

(2) Spark Jobs

Table

	ID	NOMBRE	TELEFONO	CORREO
1	1	Carl	17456339145	arcu.Sed.et@ante.co.uk
2	2	Priscilla	1552498	Donec.egestas.Aliquam@volutpatn
3	3	Jocelyn	12049568594	amet.diam@lobortis.co.uk
4	4	Aidan	17198629385	euismod.et.commodo@nibhlacinia
5	5	Leandra	8398044	at@pretiumetrutrum.com
6	6	Bert	7974453	a.felis.ullamcorper@arcu.org
7	7	Mark	16801026792	Quisque.ac@placerat.ca
8	8	Jonah	2142975	eu.ultrices.sit@vitae.ca
9	9	Hanae	9352277	eu@Nunc.ca
10	10	Cadman	18665612701	orci.adipiscing.non@semperNam.c

10:51 PM (5s)

```
CREATE EXTERNAL TABLE IF NOT EXISTS silver.empresas(
    ID INT,
    NOMBRE STRING,
    PERIODO STRING
)
USING DELTA
PARTITIONED BY (PERIODO)
LOCATION 'gs://dmc_dde_barb/produccion/dmc/silver/empresas';
```

(3) Spark Jobs

OK

10:51 PM (1s)

```
SELECT * FROM silver.empresas;
```

(2) Spark Jobs

Table

	ID	NOMBRE	PERIODO
1	1	WALMART	202502
2	2	MICROSOFT	202502
3	3	APPLE	202502
4	4	TOYOTA	202502
5	5	AMAZON	202502
6	6	GOOGLE	202502
7	7	SAMSUNG	202502
8	8	HP	202502
9	9	IBM	202502
10	10	SONY	202502

10 rows | 1.13s runtime

10:51 PM (1s)

```
show partitions silver.empresas;
```

(2) Spark Jobs

Table

	PERIODO
1	202502

✓ 2 minutes ago (5s)

```
CREATE EXTERNAL TABLE IF NOT EXISTS silver.transacciones(
    ID_PERSONA INT,
    ID_EMPRESA INT,
    MONTO DOUBLE,
    FECHA DATE,
    ANIO INT,
    MES INT,
    DIA INT
)
USING DELTA
PARTITIONED BY (ANIO, MES, DIA)
LOCATION 'gs://dmc_dde_barb/produccion/dmc/silver/transacciones';
```

▸ (3) Spark Jobs

OK

✓ Just now (1s)

```
show partitions silver.transacciones;
```

▸ (2) Spark Jobs

Table +

	ANIO	MES	DIA
1	2018	1	21
2	2018	1	22
3	2018	1	23

3 rows | 0.82s runtime

✓ Just now (1s)

```
SELECT * FROM silver.transacciones LIMIT 3;
```

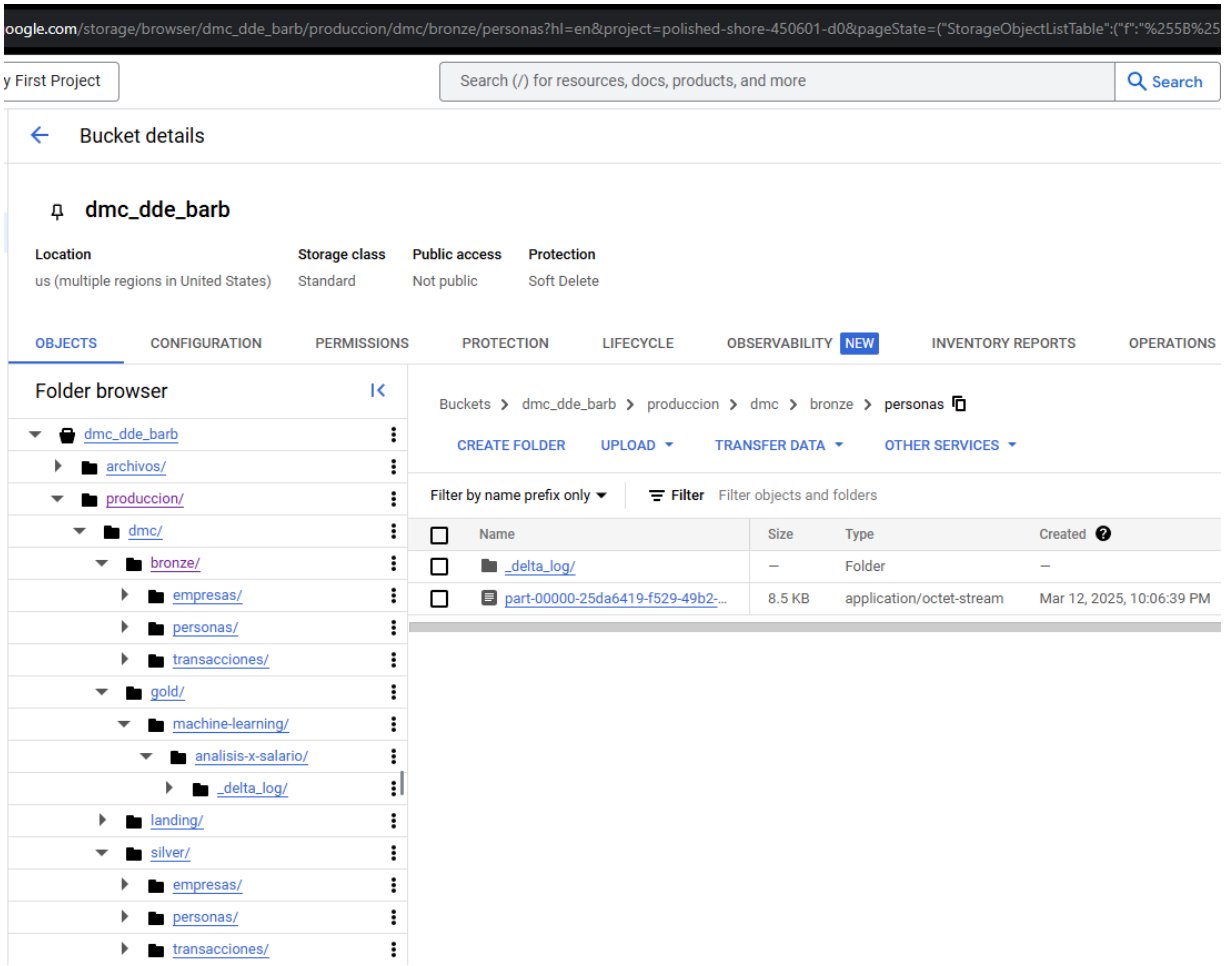
▸ (1) Spark Jobs

Table +

	ID_PERSONA	ID_EMPRESA	MONTO	FECHA	AN
1	18	3	1383	2018-01-21	
2	30	6	2331	2018-01-21	
3	47	2	2280	2018-01-21	

Creamos una subcarpeta dentro del bucket para la capa gold

Gold→machine-learning→analisis-x-salario



Creamos el ddl de la capa gold para análisis-x-salario

Just now (1s)

```
CREATE EXTERNAL TABLE IF NOT EXISTS gold.analisis_x_salario(
    periodo STRING,
    nombre_empresa STRING,
    sum_salario DOUBLE,
    avg_salario DOUBLE,
    avg_edad DOUBLE
)
USING DELTA
LOCATION 'gs://dmc_dde_barb/produccion/dmc/gold/machine-learning/analisis-x-salario';
```

OK

Just now (1s)

```
SELECT * FROM gold.analisis_x_salario;
```

(2) Spark Jobs

Table

	periodo	nombre_empresa	sum_salario	avg_salario	avg_edad
1	202502	AMAZON	136609	9757.785714285714	41.2142857142857...
2	202502	SONY	82012	9112.444444444445	40.8888888888888...
3	202502	TOYOTA	155503	19437.875	38.875
4	202502	MICROSOFT	156377	11169.785714285714	39.7857142857142...
5	202502	HP	73319	8146.555555555556	39.8888888888888...
6	202502	WALMART	79304	11329.142857142857	35.8571428571428...
7	202502	IBM	91678	15279.666666666666	37.6666666666666...
8	202502	SAMSUNG	106710	11856.666666666666	34.55555555555556
9	202502	GOOGLE	135243	10403.307692307691	50
10	202502	APPLE	151700	13790.90909090909	39.63636363636363

10 rows | 1.06s runtime