

4.-Lakehouse

Lakehouse con Databricks y Google Cloud Storage

Creación de estructura de carpetas

The image shows two side-by-side screenshots. The left screenshot is a Databricks workspace interface. The left sidebar shows a file explorer with a tree structure. A red box highlights the 'production' folder, which contains subfolders for 'dmc', 'hubspot', 'salesforce', 'sap', 'empresas', 'ddt', 'personas', 'transacciones', 'rbs', 'sunat', and 'gold'. The main panel shows a list of files and folders, including 'RealTimeYoutube', 'production', 'DeltaTableFem1302202', 'Delta', 'Class_Del_27022023', 'Class_Del_18_AI_200203', 'Class_Del_04032025_AI', 'Untitled Notebook 20...', 'ReparosPython202502...', 'Reparos_python', 'PrimerNotebook2025...', and 'EspecializacionBigDat...'. The right screenshot shows a presentation slide titled 'Untitled Diagram' with a data engineering workflow diagram. The diagram includes a 'LANDING' box, a 'BRONZE' box, a 'SILVER' box, and a 'GOLD' box, all connected by arrows. Below these is a 'DISCOVERY EXPLORATION' box. The diagram is labeled 'ESTRUCTURADA Y SEMIESTRUCTURADA' and 'DATA ENGINEERING'. The slide also features a 'VARIAS' box and a 'DATA SCIENTIST' box. The slide is presented by Bruno Rojas, Harold Castro, and Jorge Camarena.

Creación de bucket en Google Cloud Storage para alimentar el Lakehouse

The screenshot shows the Google Cloud Storage console interface. The left sidebar contains navigation links: Overview, Buckets (selected), Monitoring, and Settings. The main content area displays the 'Bucket details' for 'dmc_dde_barb'. The bucket's location is 'us (multiple regions in United States)', storage class is 'Standard', public access is 'Not public', and protection is 'Soft Delete'. Below this, there are tabs for OBJECTS, CONFIGURATION, PERMISSIONS, PROTECTION, and LIFECYCLE. The 'OBJECTS' tab is active, showing a 'Folder browser' view. The folder structure is as follows:

- dmc_dde_barb
 - archivos/
 - production/ (highlighted with a red box)
 - dmc/
 - bronze/
 - gold/
 - landing/
 - silver/

On the right side of the folder browser, there is a 'CREATE FOLDER' button and a table listing the folders. The table has a 'Name' column and a checkbox column. The folders listed are bronze/, gold/, landing/, and silver/.

Name
bronze/
gold/
landing/
silver/

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

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

+

New

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

dmc-bronze-sap-personas-process

Python

File Edit View Run Help

Last edit was 1 minute ago

Run all

Share

Publish

1 minute ago (1s)

5

```
df_personas = spark.read.format("CSV").option("header","true").option("delimiter","|").schema(df_schema).load(path_persona_landing)
display(df_personas)
```

(1) Spark Jobs

df_personas: pyspark.sql.dataframe.DataFrame = [ID: string, NOMBRE: string ... 6 more fields]

Table

+

	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@nibhacinaorci.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
12	12	Tanner	1-739-776-7897	arcu.Aliquam.ultrices@sociis.com
13	13	Trevor	512-1955	Nunc.quis.arcu@egestasa.org
14	14	Allen	733-2795	felis.Donec@necleo.org

100 rows | 1.22s runtime

Refreshed 1 minute ago

1 minute ago (25s)

6

Python

Python

df_personas.write.mode("overwrite").format("delta").save(path_persona_bronze)

(6) Spark Jobs

[Shift+Enter] to run and move to next cell

[Ctrl+Shift+P] to open the command palette

[Esc H] to see all keyboard shortcuts

26°C

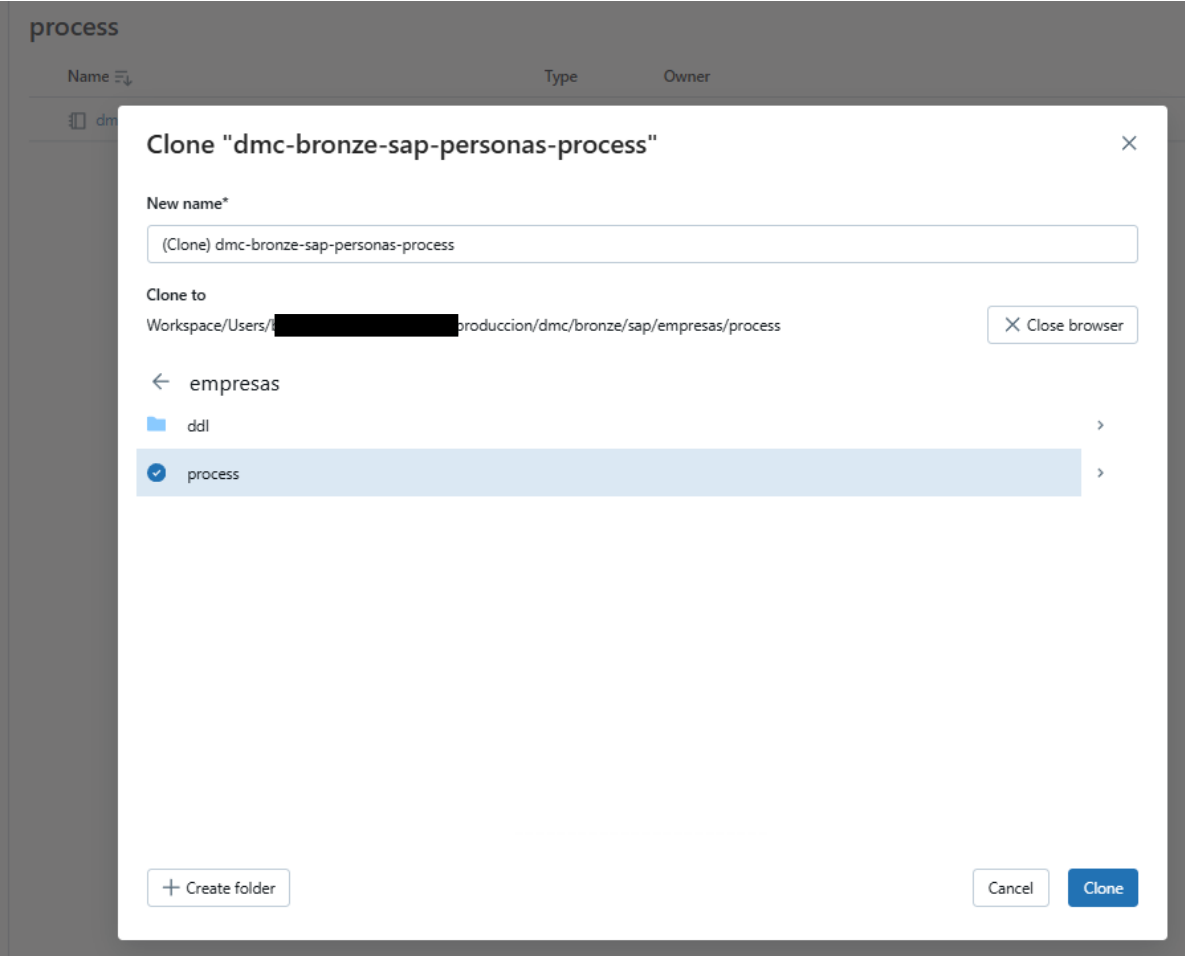
Ventoso

Búsqueda

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

The screenshot shows the Google Cloud console interface. At the top, there's a navigation bar with the Google Cloud logo, 'My First Project', and various utility icons. Below this, the 'Cloud Storage' section is active, displaying 'Bucket details' for 'dmc_dde_barb'. The bucket's metadata is shown: Location is 'us (multiple regions in United States)', Storage class is 'Standard', Public access is 'Not public', and Protection is 'Soft Delete'. The 'OBJECTS' tab is selected, showing a 'Folder browser' view. The folder structure is as follows: 'dmc_dde_barb' (root) contains 'archivos/' and 'produccion/'. 'produccion/' contains 'dmc/'. 'dmc/' contains 'bronze/'. 'bronze/' contains 'personas/' (highlighted). 'personas/' contains '_delta_log/' and '_commits/'. '_delta_log/' contains 'gold/'. 'gold/' contains 'landing/'. 'landing/' contains 'silver/'. The 'personas/' folder is expanded, showing a file named 'part-00000-25da6419-f529-49b2-...' with a download icon.

Clonando archivos en databricks



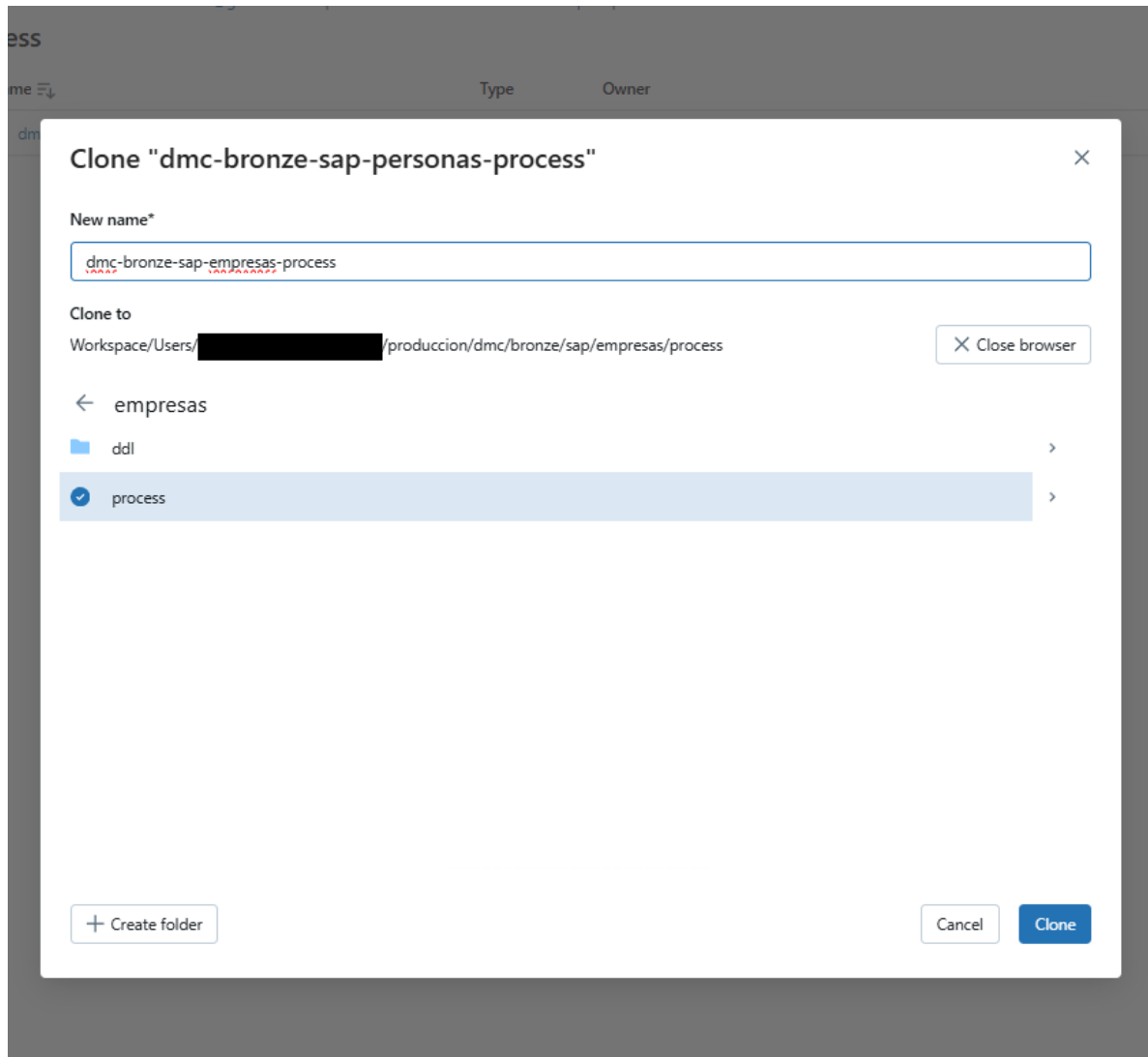


Tabla empresas a la capa bronce

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

Home

Workspace

Shared

Users

Clase_Del_04032025_AL_07032025

Clase_Del_18_AL_20022025-Spark

Clase_Del_27022025

Delta

DeltaTablePartII13022025

MasterDatabricksYoutube

produccion

dmc

bronze

hubspot

salesforce

sap

empresas

ddl

process

personas

transacciones

sbs

sunat

gold

silver

RealTimeYoutube

Trash

process

Name

dmc-bronze-sap-empresas-process

Tabla transacciones a la capa bronce

Google Cloud

My First Project

GO TO PATH

REFRESH

LEARN

Cloud Storage

Overview

Buckets

Monitoring

Settings

Bucket details

dmc_dde_barb

Location

Storage class

Public access

Protection

us (multiple regions in United States)

Standard

Not public

Soft Delete

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

Folder browser

dmc_dde_barb

archivos/

produccion/

dmc/

bronze/

empresas/

_delta_log/

_commits/

personas/

gold/

landing/

empresas/

personas/

transacciones/

silver/

Buckets > dmc_dde_barb > produccion > dmc

CREATE FOLDER

Filter by name prefix only

Filter

Filter objec

<input type="checkbox"/>	Name	
<input type="checkbox"/>	_delta_log/	
<input type="checkbox"/>	part-00000-c55f92bd-26c4-48d1-	

Google Cloud

My First Project

GO TO PATH

REFRESH

LEARN

Cloud Storage

Overview

Buckets

Monitoring

Settings

Bucket details

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

New

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

databricks

Workspace

process

dmc-silver-sap-personas-process

dmc-silver-sap-personas-process

Python

File Edit View Run Help

Last edit was now

Run all

Share

Publish

Just now (< 1s)

1

#Importamos las librerías necesarias

from pyspark.sql import SparkSession

#from pyspark.sql.types import StructField, StructType,

StringType

from pyspark.sql.functions import *

#IntegerType, DoubleType

Just now (< 1s)

2

#variables

spark = SparkSession.builder.getOrCreate()

#Archivo cloud storage

name_bucket = "dmc_dde_barb"

#Path del lakehouse, capa bronze (origen)

#capa silver (destino)

path_lakehouse = f"gs://{name_bucket}/produccion/dmc"

path_bronze = f"{path_lakehouse}/bronze/personas/"

path_silver = f"{path_lakehouse}/silver/personas/"

Just now (36s)

3

df = spark.read.format("delta").option("header", "true").load

(path_bronze)

display(df)

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [ID: string, NOMBRE: string ... 6 more

fields]

Table

	ID	NOMBRE	TELEFONO	CORREO
1	1	Carl	1-745-633-9145	arcu.Sed.et@e
2	2	Priscilla	155-2498	Donec.egesta:
3	3	Jocelyn	1-204-956-8594	amet.diam@k
4	4	Aidan	1-719-862-9385	euismod.et.co

13°C

Mayorm. nublado

Búsqueda

Limpieza y casteo de datos

dmc-silver-sap-personas-process Python ▾

File Edit View Run Help Last edit was now

Run all Share Publish

Workspace

- process
- dmc-silver-sap-personas-process

```
datetime_server = datetime.now()
periodo_actual = datetime_server.strftime("%Y%m")
print(periodo)
periodo_previo_1m = datetime_server - relativedelta(months=1)
print(periodo_previo_1m)
periodo_previo_1m_formateado = periodo_previo_1m.strftime("%Y%m")
print(periodo_previo_1m_formateado)
```

202503
2025-02-18 02:34:30.553593
202502

```
df_t = df.withColumn('telefono', regexp_replace('telefono', '-', ''))
display(df_t)
```

(1) Spark Jobs

df_t: pyspark.sql.dataframe.DataFrame = [ID: string, NOMBRE: string ... 6 more fields]

ID	NOMBRE	telefono	CORREO
1	Carl	1 745 633 9145	arcu.Sed.et@ant
2	Priscilla	155 2498	Donecegestas.A
3	Jocelyn	1 204 956 8594	amet.diam@lob
4	Aidan	1 719 862 9385	eusmod.et.com
5	Leandra	839 8044	at@pretiumetru
6	Bert	797 4453	a.felis.ullamcorp
7	Mark	1 680 102 6792	Quisque.ac@pla
8	Jonah	214 2975	eu.ultrices.sit@v
9	Hanae	935 2277	eu@Nunc.ca
10	Cadman	1 866 561 2701	orci.adipiscing.n
11	Melyssa	596 7736	vel@vulputatepi
12	Tanner	1 739 776 7897	arcu.Aliquam.uit
13	Trevor	512 1955	Nunc.quis.arcu@



✓ 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

+ New

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

databricks

dmc-silver-sap-personas-process Python

Run all

Share

Publish

Workspace

process

dmc-silver-sap-personas-process

202503
2025-02-18 02:34:30.553593
202502

Just now (2s)

df_t = df.withColumn('TELEFONO', regexp_replace('TELEFONO', '-',''))\n .withColumn('PERIODO', date_format(add_months(current_date\n (-1), "yyyyMM")))\n .withColumn("PERIODO", date_format(add_months(current_date\n (-1), "yyyyMM")))\n .withColumn("SEGMENTO", when((col("SALARIO")<3500,\n "Masivo").when((col("SALARIO")>=3500) & (col("SALARIO")\n <=10000),"Premium").otherwise("Beyond"))\n #display(df_t)\n\n df_c = df_t.withColumn('ID', col('ID').cast(IntegerType()))\n .withColumn('ID_EMPRESA', col('ID_EMPRESA').cast(IntegerType\n ()))\n .withColumn('EDAD', col('EDAD').cast(IntegerType()))\n .withColumn('SALARIO', col('SALARIO').cast(DoubleType()))\n .withColumn('FECHA_INGRESO', to_date(col('FECHA_INGRESO'),\n 'yyyy-MM-dd')).withColumn('ANIO', year(col('FECHA_INGRESO'))).\n withColumn('MES', month(col('FECHA_INGRESO')).withColumn\n ('DIA', dayofmonth(col('FECHA_INGRESO')))\n\n display(df_c)

(T) Spark Jobs

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

df_t: pyspark.sql.dataframe.DataFrame = [ID: string, NOMBRE: string ... 8 more fields]

Table +

	ID	NOMBRE	TELEFONO	CORREO
1	1	Carl	17456339145	arcu.Sed.et@z
2	2	Priscilla	1552498	Donec.egesta
3	3	Jocelyn	12049568594	amet.diam@k
4	4	Aidan	17198629385	euismod.et.co
5	5	Leandra	8398044	at@pretiumet

Creación de partición en la tabla personas

The screenshot shows the Databricks community cloud interface. The notebook is titled "dmc-silver-sap-personas-process" and is written in Python. The interface includes a sidebar with navigation options like Workspace, Recents, Search, Catalog, Workflows, Compute, Machine Learning, and Experiments. The main area displays a table with 100 rows of data, including columns for ID, Name, and Email. Below the table, a code cell shows a Python snippet for writing data to a Delta table with a new partition.

ID	Name	Email
6	Bert	a.felis.ullamco
7	Mark	Quisque.ac@
8	Jonah	eu.ultrices.sit@
9	Hanae	eu@Nunc.ca
10	Cadman	orci.adipiscing
11	Melissa	vel@vulputate
12	Tanner	arcu.Aliquam.
13	Trevor	Nuncquis.arci
14	Allen	felis.Donec@r

```
df_c.write.mode('overwrite').partitionBy('PERIODO').format('delta').save(path_silver)
```

100 rows | 2.29s runtime | Refreshed 6 minutes ago

Just now (9s) 7

(6) Spark Jobs

Corroborando data en el bucket

← → ↻ 🔍 console.cloud.google.com/storage/browser/dmc_dde_barb/produccion/dmc/silver/personas/PERIODO%3...

☰ Google Cloud

My First Project

🔍 ✨ 📦 📄 4 ? ⋮ B

Cloud Storage

Overview

Buckets

Monitoring

Storage Insights

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/

📁 gold/

📁 landing/

📁 silver/

📁 personas/

📁 _delta_log/

📁 PERIODO=202502/

Buckets > dmc_dde_barb > produccion > dmc

CREATE FOLDER

Filter by name prefix only Filter Filter object

<input type="checkbox"/>	Name	
<input type="checkbox"/>	part-00000-ac02fa9d-b2af-4446-9	⬇ ⋮

Realizando casteo de datos en la tabla empresas

The screenshot shows a Databricks notebook interface. The notebook is titled "dmc-silver-sap-empresas-process" and is written in Python. The left sidebar shows the workspace structure with a folder named "process" containing the notebook. The main area displays the code and its execution results.

Code:

```
withColumn("PERIODO",date_format(add_months(current_date(),-1),"yyyyMM"))

#display(df_t)

df_c = df_t.withColumn('ID',col('ID').cast(IntegerType()))

display(df_c)
```

Execution Results:

(1) Spark Jobs

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

Table View:

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

10 rows | 0.72s runtime Refreshed 1 minute ago

Next Code Block:

```
df_c.write.mode('overwrite').partitionBy('PERIODO').format('delta').save(path_silver)
```

(6) Spark Jobs

Validando creación de la tabla en el bucket

The screenshot shows the Google Cloud Storage console interface. The left sidebar contains navigation links: Overview, Buckets (selected), Monitoring, Storage Insights, and Settings. The main area is titled 'Bucket details' for the bucket 'dmc_dde_barb'. It displays metadata: Location (us), Storage class (Standard), Public access (Not public), and Protection (Soft Delete). Below this, there are tabs for OBJECTS, CONFIGURATION, PERMISSIONS, PROTECTION, and LIFECYCLE. The 'OBJECTS' tab is active, showing a 'Folder browser' view. The breadcrumb path is 'Buckets > dmc_dde_barb > produccion > dmc'. A 'CREATE FOLDER' button is visible. A table lists folders under the 'dmc/' prefix, including 'bronze/', 'gold/', 'landing/', 'silver/', 'empresas/', and 'personas/'. The 'personas/' folder is currently selected.

Añadiendo partición a la tabla transacciones y columnas nuevas

The screenshot shows a code execution console with a status bar at the top indicating 'Just now (10s)' and a count of '5'. The code being executed is: `df_c.write.mode("overwrite").partitionBy("ANIO", "MES", "DIA").format("delta").save(path_silver)`. Below the code, it shows '(6) Spark Jobs'.

Guardando tabla en formato delta en el bucket

The screenshot shows a code execution console with a status bar at the top indicating 'Just now (10s)' and a count of '5'. The code being executed is: `df_c.write.mode("overwrite").partitionBy("ANIO", "MES", "DIA").format("delta").save(path_silver)`. Below the code, it shows '(6) Spark Jobs'.

Validando data en el bucket

The screenshot shows the Google Cloud Storage console interface. The top navigation bar includes the Google Cloud logo, the project name "My First Project", and various utility icons like search, refresh, and help. On the left sidebar, the "Cloud Storage" section is active, with "Buckets" selected under the overview menu.

The main content area displays the "Bucket details" for the bucket named "dmc_dde_barb". Key information shown includes:

- Location:** us (multiple regions in United States)
- Storage class:** Standard
- Public access:** Not public
- Protection:** Soft Delete

Below the bucket details, there are tabs for "OBJECTS", "CONFIGURATION", "PERMISSIONS", "PROTECTION", and "LIFECYCLE". The "OBJECTS" tab is currently selected, showing a "Folder browser" view. The folder tree on the left lists several directories, with "personas/" highlighted. To the right of the folder browser, there's a breadcrumb trail: "Buckets > dmc_dde_barb > produccion > dmc". Below this, there's a "CREATE FOLDER" button and a filter section labeled "Filter by name prefix only" with a "Filter" button and a "Filter object" input field.

Name	
<input type="checkbox"/> empresas/	
<input type="checkbox"/> personas/	

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	Donecegestas.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 ▾ +

	¹ ₃ ID	^A _C NOMBRE	^A _C TELEFONO	^A _C 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@egestas.org
14	14	Allen	7332795	felis.Donec@necleo.org
15	15	Wanda	3596973	Nam.nulla.magna@In.org
16	16	Alden	3418522	odio@morbistriquesenectus.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 ▾ +

	¹ ₃ ID	^A _C NOMBRE	^A _C 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

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
```

	A _C periodo	A _C nombre_empresa	1.2 salario	1 ₃ 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	34.5555555555556
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 ▼ +

	A _C periodo	A _C nombre_empresa	1.2 sum_salario	1.2 avg_salario	1.2 avg_edad
1	202502	AMAZON	136609	9757.785714285714	41.2142857142857...
2	202502	SONY	82012	9112.444444444445	40.888888888888...
3	202502	TOYOTA	155503	19437.875	38.875
4	202502	MICROSOFT	156377	11169.785714285714	39.7857142857142...
5	202502	HP	73319	8146.555555555556	39.888888888888...
6	202502	WALMART	79304	11329.142857142857	35.8571428571428...
7	202502	IBM	91678	15279.666666666666	37.666666666666...
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).

Jerarquía de objetos del Unity Catalog

Metastore: un metastore es el contenedor de objetos de nivel superior en Unity Catalog. Los metastores viven en el nivel de la cuenta y funcionan como la cima de la pirámide en el modelo de gobierno de datos de Databricks.

Los metastores administran activos de datos (tablas, vistas y volúmenes) y los permisos que rigen el acceso a ellos.

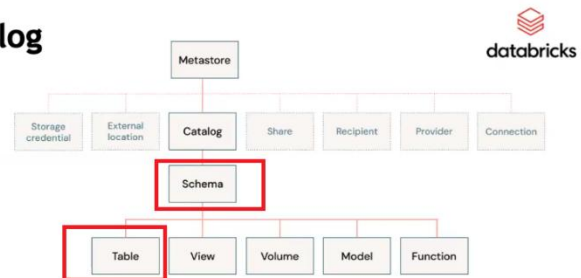
Catálogo: los catálogos son el nivel más alto en la jerarquía de datos (catálogo > esquema > tabla/vista/volumen) administrado por el metaalmacén de Unity Catalog. Están pensados como la unidad principal de aislamiento de datos en un modelo típico de gobierno de datos de Databricks.

Esquema (base de datos): los esquemas, también conocidos como bases de datos, son agrupaciones lógicas de datos tabulares (tablas y vistas), datos no tabulares (volúmenes), funciones y modelos de aprendizaje automático. Le brindan una forma de organizar y controlar el acceso a los datos que es más granular que los catálogos. Por lo general, representan un caso de uso único, un proyecto o un entorno limitado de equipo.

Tablas: las tablas residen en la tercera capa del espacio de nombres de tres niveles de Unity Catalog. Contienen filas de datos.

Unity Catalog le permite crear *tablas administradas* y *tablas*

externas.



• **Vistas:** una vista es un objeto de solo lectura derivado de una o más tablas y vistas en un metastore.

• **Filas y columnas:** el acceso a nivel de filas y columnas, junto con el enmascaramiento de datos, se otorga mediante vistas dinámicas o filtros de filas y máscaras de columnas. Las vistas dinámicas son de solo lectura.

• **Volúmenes:** los volúmenes residen en la tercera capa de Unity Catalog. Manejan datos no tabulares. Puede utilizar volúmenes para almacenar, organizar y acceder a archivos en cualquier formato.

• **Modelos y funciones:** aunque, estrictamente hablando, no son activos de datos, los modelos registrados y las funciones definidas por el usuario también se pueden administrar en Unity Catalog y residir en el nivel más bajo de la jerarquía de objetos.

By Juan Salinas

community.cloud.databricks.com/browse/folders/2002778758247395?o=1788945041965453

Workspace > Users > [usuario]

ddl

Name	Type	Owner	Created at
schemas	Folder	Bruno Rojas	Mar 19, 20...

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.

09:37 PM (<1s)

```
show catalogs;
```

Table +

	<div>A^B_C catalog</div>
1	spark_catalog

1 row | 0.21s runtime

09:38 PM (<1s)

```
show databases;
```

Table +

	<div>A^B_C databaseName</div>
1	default

1 row | 0.17s runtime

09:39 PM (<1s)

```
show tables in default;
```

Table +

	<div>A^B_C database</div>	<div>A^B_C tableName</div>	<div>isTemporary</div>
--	---	--	------------------------

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

▶

✓ 09:39 PM (<1s)

3

```
show tables in default;
```

Table ▼

+

	database	tableName	isTemporary
--	----------	-----------	-------------

No rows returned

0 rows | 0.18s runtime

▶

✓ 09:42 PM (1s)

4

```
create schema bronze
comment 'Descripcion del esquema'
LOCATION 'gs://dmc_dde_barb/produccion/dmc/bronze';
```

OK

⋮

▼

▶

✓ 09:44 PM (<1s)

5

```
create schema silver
comment 'Descripcion del esquema'
LOCATION 'gs://dmc_dde_barb/produccion/dmc/silver';
```

OK

▶

✓ 09:45 PM (<1s)

6

```
create schema if not exists gold
comment 'Descripcion del esquema'
LOCATION 'gs://dmc_dde_barb/produccion/dmc/gold';
```

OK

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.

```
create external table bronze.personas(  
  ID STRING,  
  NOMBRE STRING,  
  TELEFONO STRING,  
  CORREO STRING,  
  FECHA_INGRESO STRING,  
  EDAD STRING,  
  SALARIO STRING,  
  ID_EMPRESA STRING  
)  
USING DELTA  
LOCATION "gs://dmc_dde_barb/produccion/dmc/bronze/personas/"
```

▶ (3) Spark Jobs

OK

▶ 1 minute ago (1s) 2 SQL

```
select * from bronze.personas;
```

▶ (1) Spark Jobs

Table +

	A _C ID	A _C NOMBRE	A _C TELEFONO	A _C 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@nibhiaciniaorci.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 ▼ + 🔍 🏠

	^B _c ID	^B _c 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

	^A _C ID_PERSONA	^A _C ID_EMPRESA	^A _C MONTO	^A _C
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 ▼ +

	1 ² ANIO	1 ² MES	1 ² 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 ▼ +

	1 ² ID_PERSONA	1 ² ID_EMPRESA	1.2 MONTO	📅 FECHA	1 ² 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

oogle.com/storage/browser/dmc_dde_barb/produccion/dmc/bronze/personas?hl=en&project=polished-shore-450601-d0&pageState={"StorageObjectListTable":{"f":"%255B%25

y First Project

Search (/) for resources, docs, products, and more

Search

← Bucket details

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

OBSERVABILITY

NEW

INVENTORY REPORTS

OPERATIONS

Folder browser

1<

dmc_dde_barb

archivos/

produccion/

dmc/

bronze/

empresas/

personas/

transacciones/

gold/

machine-learning/

analisis-x-salario/

_delta_log/

landing/

silver/

empresas/

personas/

transacciones/

Buckets > dmc_dde_barb > produccion > dmc > bronze > personas

CREATE FOLDER

UPLOAD

TRANSFER DATA

OTHER SERVICES

Filter by name prefix only

Filter

Filter objects and folders

<input type="checkbox"/>	Name	Size	Type	Created
<input type="checkbox"/>	_delta_log/	—	Folder	—
<input type="checkbox"/>	part-00000-25da6419-f529-49b2-...	8.5 KB	application/octet-stream	Mar 12, 2025, 10:06:39 PM

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

▼

+

	A _C periodo	A _C nombre_empresa	1.2 sum_salario	1.2 avg_salario	1.2 avg_edad
1	202502	AMAZON	136609	9757.785714285714	41.2142857142857...
2	202502	SONY	82012	9112.444444444445	40.888888888888...
3	202502	TOYOTA	155503	19437.875	38.875
4	202502	MICROSOFT	156377	11169.785714285714	39.7857142857142...
5	202502	HP	73319	8146.555555555556	39.888888888888...
6	202502	WALMART	79304	11329.142857142857	35.8571428571428...
7	202502	IBM	91678	15279.666666666666	37.666666666666...
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