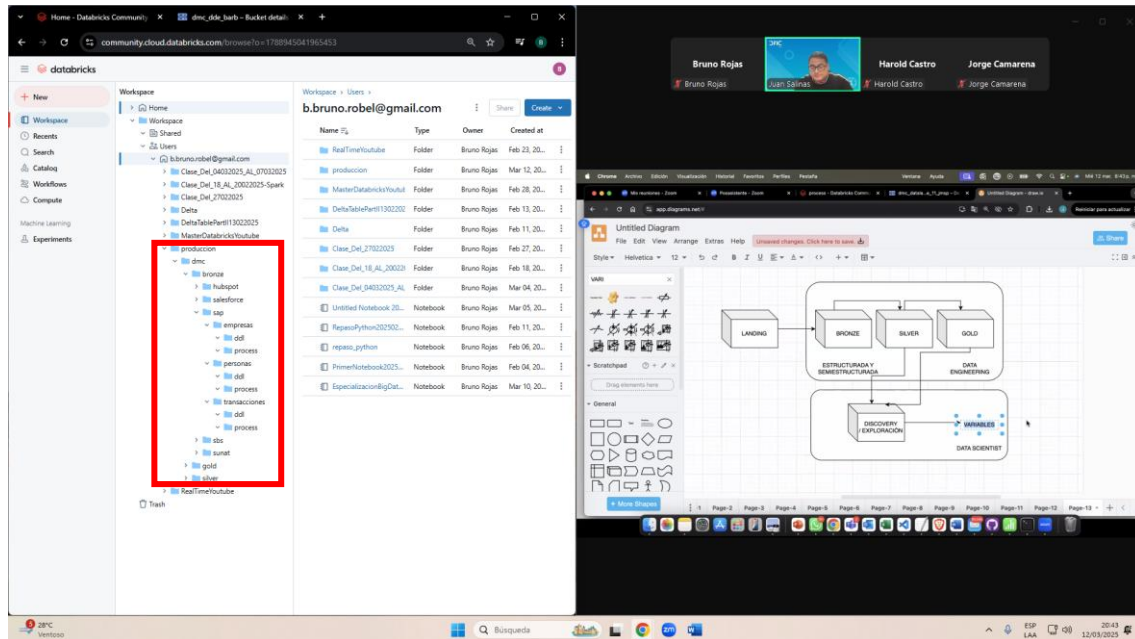
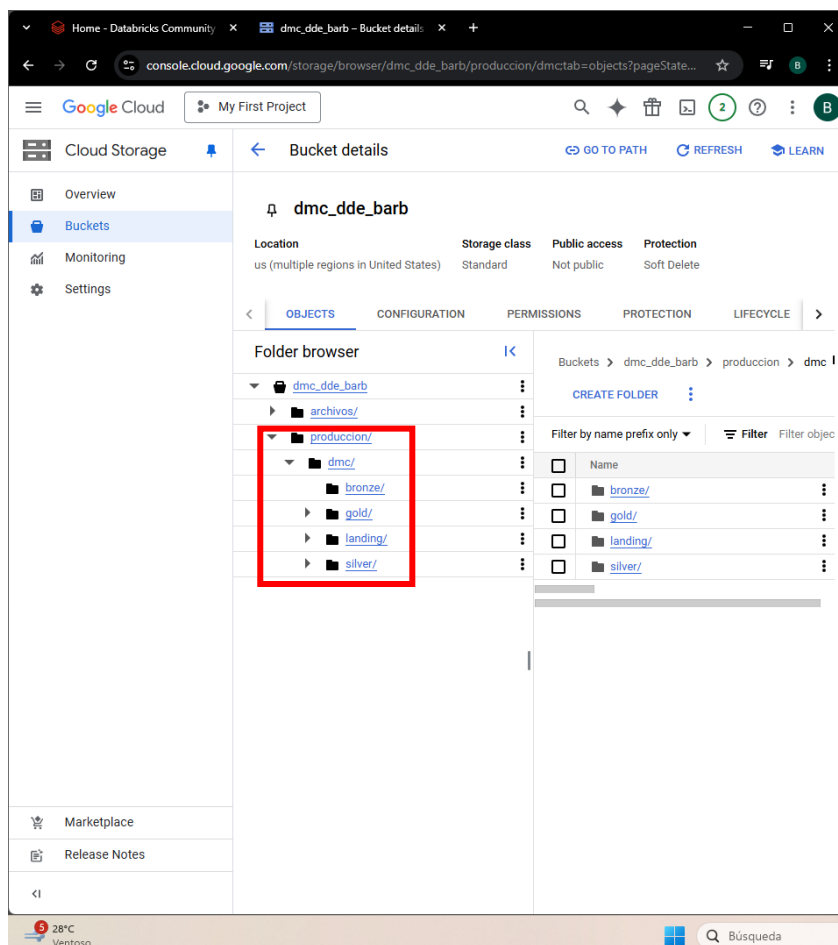


Lakehouse con Databricks y Google Cloud Storage

Creación de estructura de carpetas



Creación de bucket en Google Cloud Storage para alimentar el Lakehouse

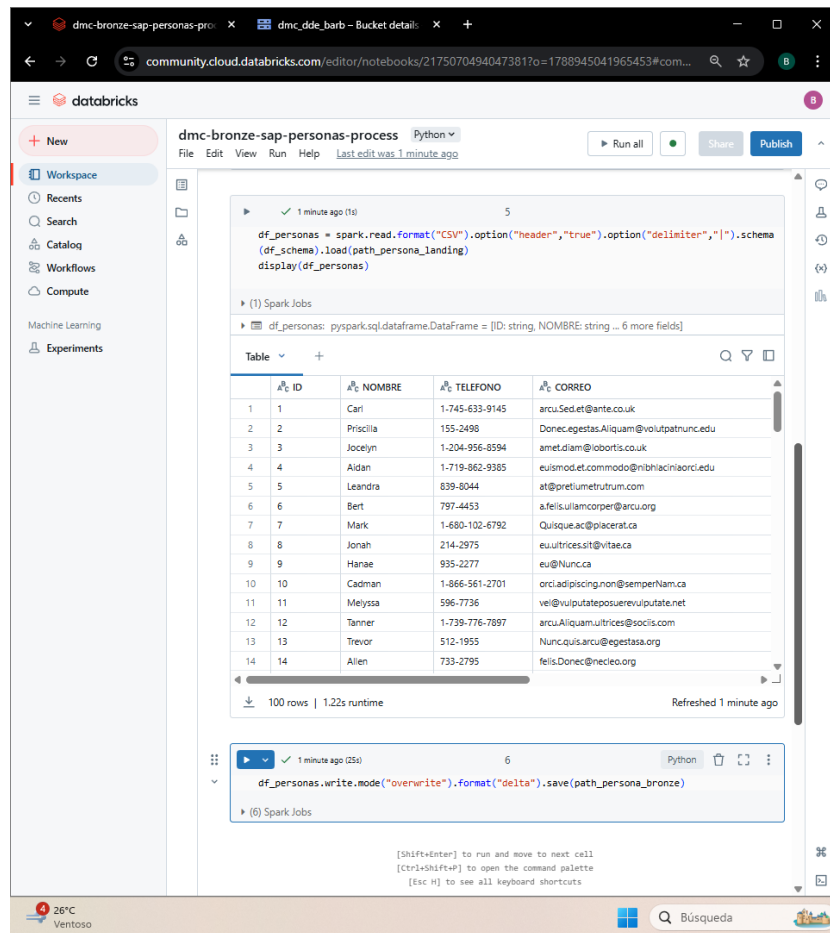


Leyendo datos de la capa landing de personas desde el bucket y armando el esquema del dataframe.

The screenshot shows a Databricks notebook interface with the following components:

- Left Sidebar:** Contains navigation links for 'New', 'Workspace', 'Recents', 'Search', 'Catalog', 'Workflows', 'Compute', 'Machine Learning', and 'Experiments'.
- Top Bar:** Displays the notebook title 'dmc-bronze-sap-personas-process', the language 'Python', and buttons for 'Run all', 'Share', and 'Publish'.
- Code Blocks:**
 - Block 1:** Imports SparkSession and StructField, StructType, StringType, IntegerType, and DoubleType from pyspark.sql.
 - Block 2:** Defines variables for the Spark session and file paths. It sets 'name_bucket' to 'dmc_dde_barb', 'path_lakehouse' to 'gs://{name_bucket}/production/dmc', 'path_persona_landing' to 'gs://{path_lakehouse}/landing/personas/persona.data', and 'path_persona_bronze' to 'gs://{path_lakehouse}/bronze/personas/'.
 - Block 3:** Defines a schema 'df_schema' as a StructType with fields: 'ID' (StringType, True), 'NOMBRE' (StringType, True), 'TELEFONO' (StringType, True), 'CORREO' (StringType, True), 'FECHA_INGRESO' (StringType, True), 'EDAD' (StringType, True), 'SALARIO' (StringType, True), and 'ID_EMPRESA' (StringType, True).
 - Block 4:** Prints the 'path_persona_bronze' variable, showing the output 'gs://dmc_dde_barb/production/dmc/bronze/personas/'.
 - Block 5:** (Empty code block).
- Bottom Bar:** Shows system information (26°C, Ventoso) and a search bar labeled 'Búsqueda'.

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



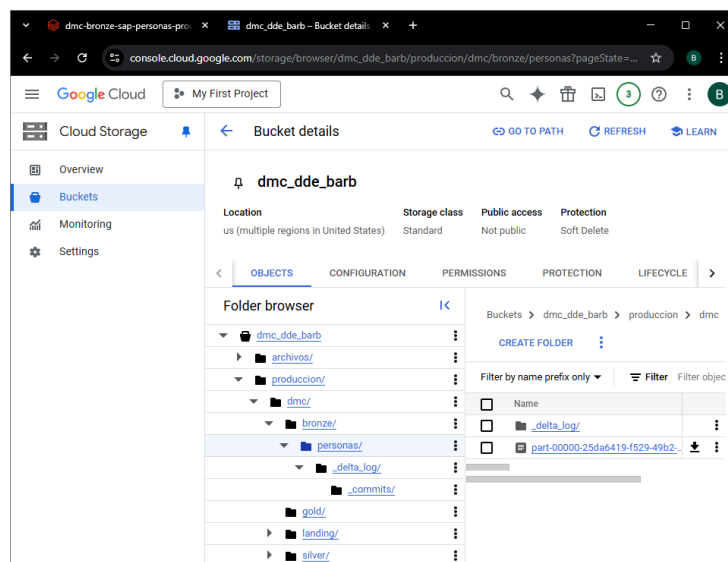
The screenshot shows the Databricks interface with a notebook titled "dmc-bronze-sap-personas-process". The first code cell reads a CSV file and displays a table of 14 rows. The second code cell writes the data to a Delta table.

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

ID	NOMBRE	TELEFONO	CORREO
1	Carl	1-745-633-9145	arcu.Sed.et@ante.co.uk
2	Priscila	155-2498	Donec.egestas.Aliquam@volutpatrunc.edu
3	Jocelyn	1-204-956-8594	amet.diam@lobortis.co.uk
4	Aidan	1-719-862-9365	euismod.et.commodo@nibhiacinaorci.edu
5	Leandra	839-8044	at@pretiumtrulum.com
6	Bert	797-4453	a.felis.ulam.corper@arcu.org
7	Mark	1-680-102-6792	Quisque.ac@placera.ca
8	Jonah	214-2975	eu.ultrices.sit@vitas.ca
9	Hanae	935-2277	eu@Nunc.ca
10	Cadman	1-866-561-2701	orci.adipiscing.non@semperNam.ca
11	Melyssa	596-7736	vel@vulputateposuerevulputate.net
12	Tanner	1-739-776-7897	arcu.Aliquam.ultrices@sociis.com
13	Trevor	512-1955	Nuncquis.arcu@egestasa.org
14	Allen	733-2795	felis.Donec@necio.org

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

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



The screenshot shows the Google Cloud Storage console for the bucket "dmc_dde_barb". The folder structure is as follows:

- dmc_dde_barb
 - archivos/
 - produccion/
 - dmc/
 - bronze/
 - personas/
 - delta_log/
 - commits/
 - gold/
 - landing/
 - silver/

Clonando archivos en databricks

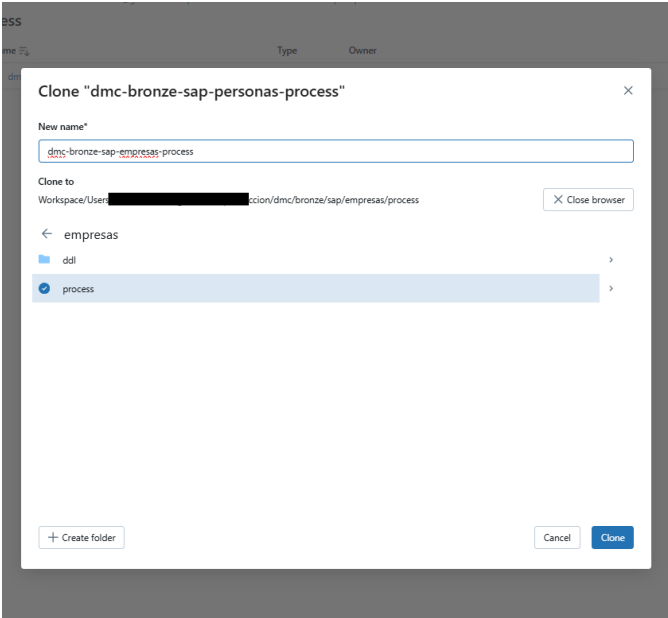
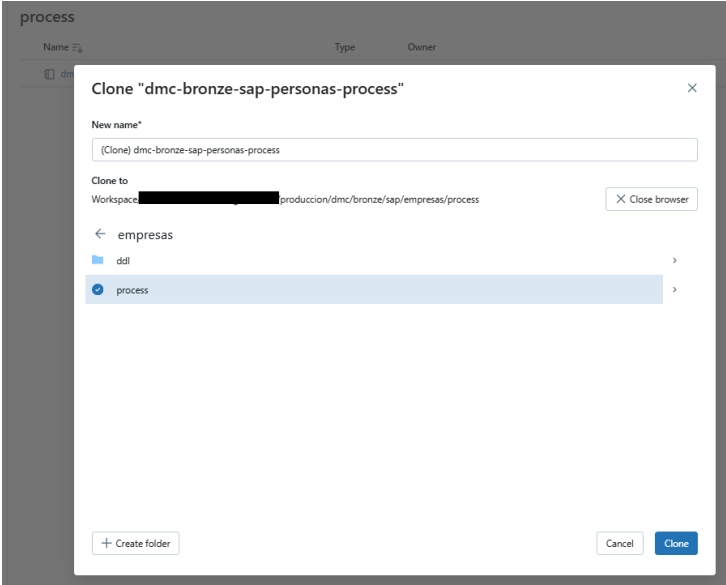


Tabla empresas a la capa bronce

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

Home

Workspace

Shared

Users

[redacted]

Clase_Del_04032025_AI_07032025

Clase_Del_18_AI_20022025-Spark

Clase_Del_27022025

Delta

DeltaTablePart113022025

MasterDataBricksYoutube

produccion

dmc

bronze

hubspot

salesforce

sap

empresas

ddl

process

personas

transacciones

sbs

sunat

gold

silver

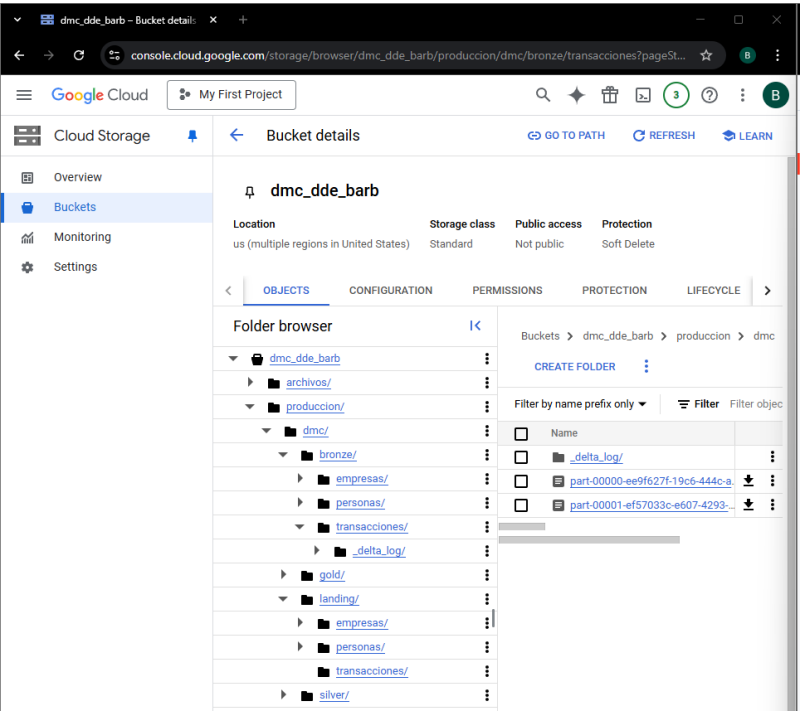
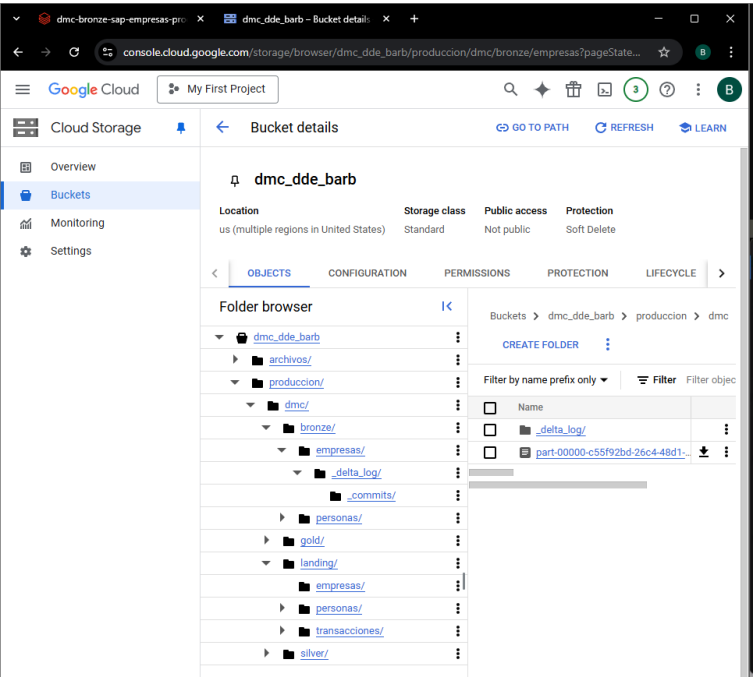
RealTimeYoutube

Trash

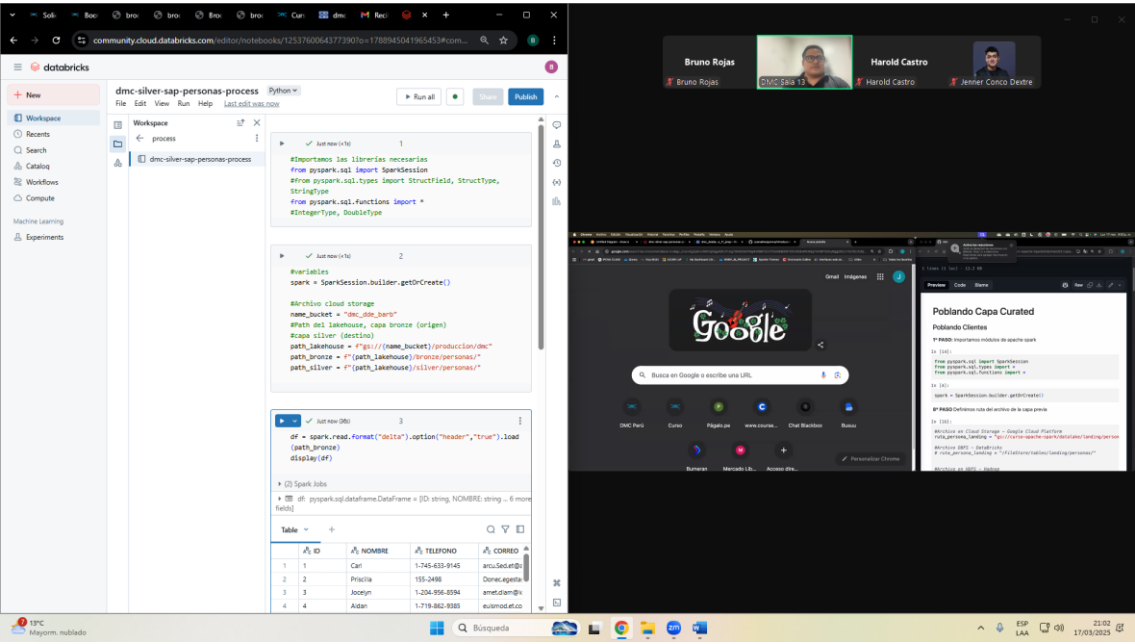
process

Name	Type	Owner
dmc-bronze-sap-empresas-process	Notebook	Bruno Rojas

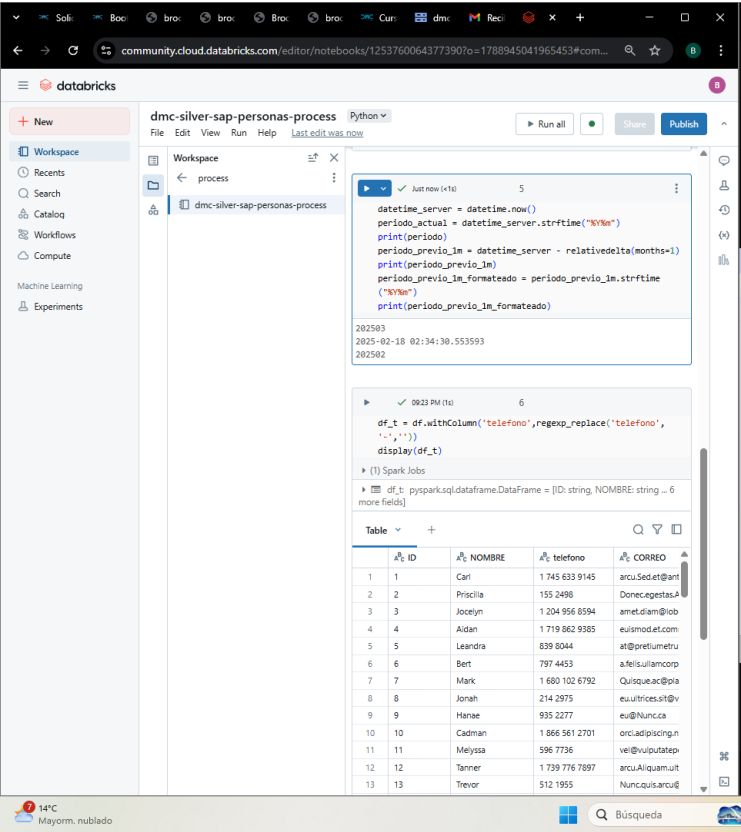
Tabla transacciones a la capa bronce



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 datetime.datetime import relativedelta
```

Añadiendo columnas para enriquecer la data

The screenshot shows a Databricks workspace with a notebook titled "dmc-silver-sap-personas-process". The code defines two DataFrames, `df_t` and `df_c`, and displays them. `df_t` has 8 fields and `df_c` has 11 fields. Below the code, a table view shows the first 5 rows of the data.

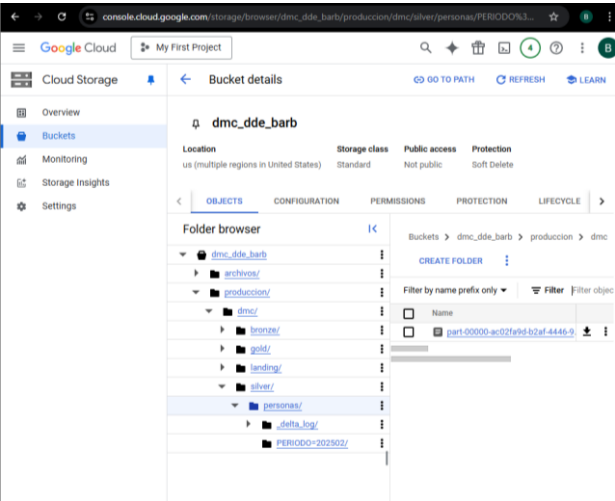
ID	NOMBRE	TELEFONO	CORREO
1	Carl	17456339145	arcu.Sed.et@
2	Priscila	1552498	Donec.egesta
3	Jocelyn	12049568594	amet.diam@i
4	Aidan	17198629385	eiusmod.et.co
5	Leandra	8398044	at@pretiumet

Creación de partición en la tabla personas

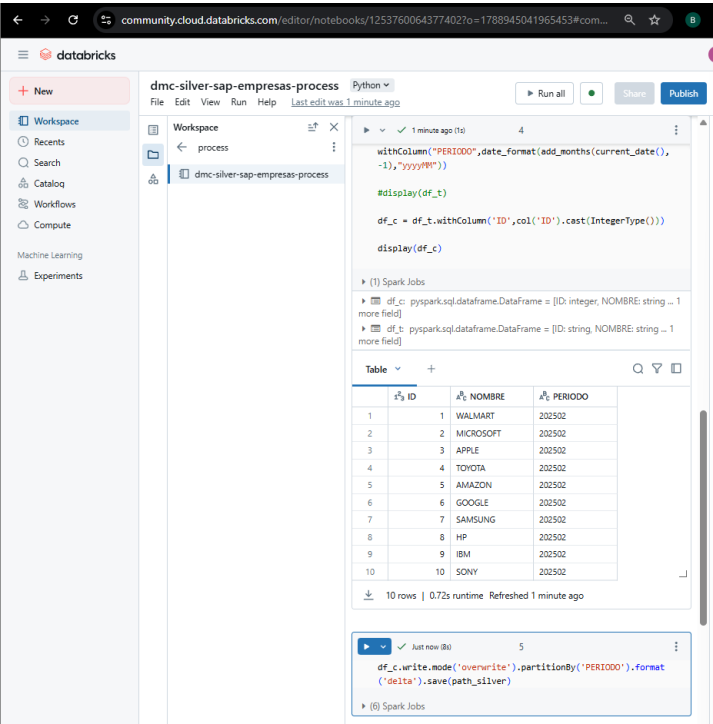
The screenshot shows the same Databricks workspace. The code defines a new DataFrame `df_c` and writes it to a table named "personas" in the "silver" path, partitioned by "PERIODO". The table view shows the first 14 rows of the data.

ID	NOMBRE	TELEFONO	CORREO
6	Bert	7974453	a.felisulamco
7	Mark	16801026792	Quisque.ac@;
8	Jonah	2142975	euultrices.sit
9	Hanae	9352277	eu@Nunc.ca
10	Cadman	18665612701	orci.adipiscing
11	Melyssa	5967736	vel@vulputate
12	Tanner	17397767897	arcu.Aliquam
13	Trevor	5121955	Nunc.quis.arci
14	Allen	7332795	felis.Donec@r

Corroborando data en el bucket



Realizando casteo de datos en la tabla empresas



Validando creación de la tabla en el bucket

The screenshot shows the Google Cloud Storage console for a bucket named 'dmc_dde_barb'. The bucket is located in 'us (multiple regions in United States)', has a 'Standard' storage class, 'Not public' access, and 'Soft Delete' protection. The 'OBJECTS' tab is selected, showing a folder browser view. The path is 'Buckets > dmc_dde_barb > produccion > dmc'. The folder browser shows a tree structure with folders like 'archivos/', 'produccion/', 'dmc/', 'bronze/', 'gold/', 'landing/', 'silver/', 'empresas/', and 'personas/'. The 'personas/' folder is expanded, showing sub-folders like 'PERIODO=202502/' and 'delta_log/'.

Añadiendo partición a la tabla transacciones y columnas nuevas

The screenshot shows a Databricks workspace with a notebook titled 'dmc-silver-sap-transaccioens-process'. The notebook is in Python mode and shows a Spark job that displays and writes data to a table. The table has columns: ID_PERSONA, ID_EMPRESA, MONTO, FECHA, ANIO, MES, DIA. The data is displayed in a table view with 15 rows. The table is named 'transacciones' and is located in the 'dmc-silver-sap-transaccioens-process' workspace. The notebook code shows a Spark job that displays the data and then writes it to a table with a partition by 'ANIO', 'MES', and 'DIA'.

ID_PERSONA	ID_EMPRESA	MONTO	FECHA	ANIO	MES	DIA
1	18	3	1383	2018-01-21	2018	1
2	30	6	2331	2018-01-21	2018	1
3	47	2	2280	2018-01-21	2018	1
4	28	1	730	2018-01-21	2018	1
5	91	4	3081	2018-01-21	2018	1
6	74	8	2409	2018-01-21	2018	1
7	41	2	3754	2018-01-22	2018	1
8	42	9	4079	2018-01-22	2018	1
9	24	6	4475	2018-01-22	2018	1
10	67	9	561	2018-01-22	2018	1
11	9	4	3765	2018-01-22	2018	1
12	97	3	3669	2018-01-22	2018	1
13	91	5	3497	2018-01-22	2018	1
14	61	3	735	2018-01-23	2018	1
15	15	5	367	2018-01-23	2018	1

Guardando tabla en formato delta en el bucket

New

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

dmc-silver-sap-transaccioens-process

Python

Run all

Share

Publish

File

Edit

View

Run

Help

Last edit was 1 minute ago

Workspace

process

dmc-silver-sap-transaccioens-proc...

1 minute ago (1s)

4

(1) Spark Jobs

df_c: pyspark.sql.dataframe.DataFrame = [ID_PERSONA: integer, ID_EMPRESA: integer ... 5 more fields]

Table

	ID_PERSONA	ID_EMPRESA	MONTO	
1	18	3	1383	20
2	30	6	2331	20
3	47	2	2280	20
4	28	1	730	20
5	91	4	3081	20
6	74	8	2409	20
7	41	2	3754	20
8	42	9	4079	20
9	24	6	4475	20
10	67	9	561	20
11	9	4	3765	20
12	97	3	3669	20
13	91	5	3497	20
14	61	3	735	20

10,000+ rows | Truncated data | 1.38s runtime

Refreshed 1 minute ago

Just now (10s)

5

(6) Spark Jobs

df_c.write.mode("overwrite").partitionBy("ANIO","MES","DIA").format("delta").save(path_silver)

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

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

[Esc H] to see all keyboard shortcuts

Validando data en el bucket

Google Cloud

My First Project

Cloud Storage

Overview

Buckets

Monitoring

Storage Insights

Settings

Marketplace

Release Notes

Bucket details

GO TO PATH

REFRESH

LEARN

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/

gold/

landing/

silver/

empresas/

_delta_log/

PERIODO=202502/

personas/

_delta_log/

PERIODO=202502/

transacciones/

_delta_log/

ANIO=2018/

MES=1/

DIA=21/

DIA=22/

DIA=23/

Buckets > dmc_dde_barb > produccion > dmc

CREATE FOLDER

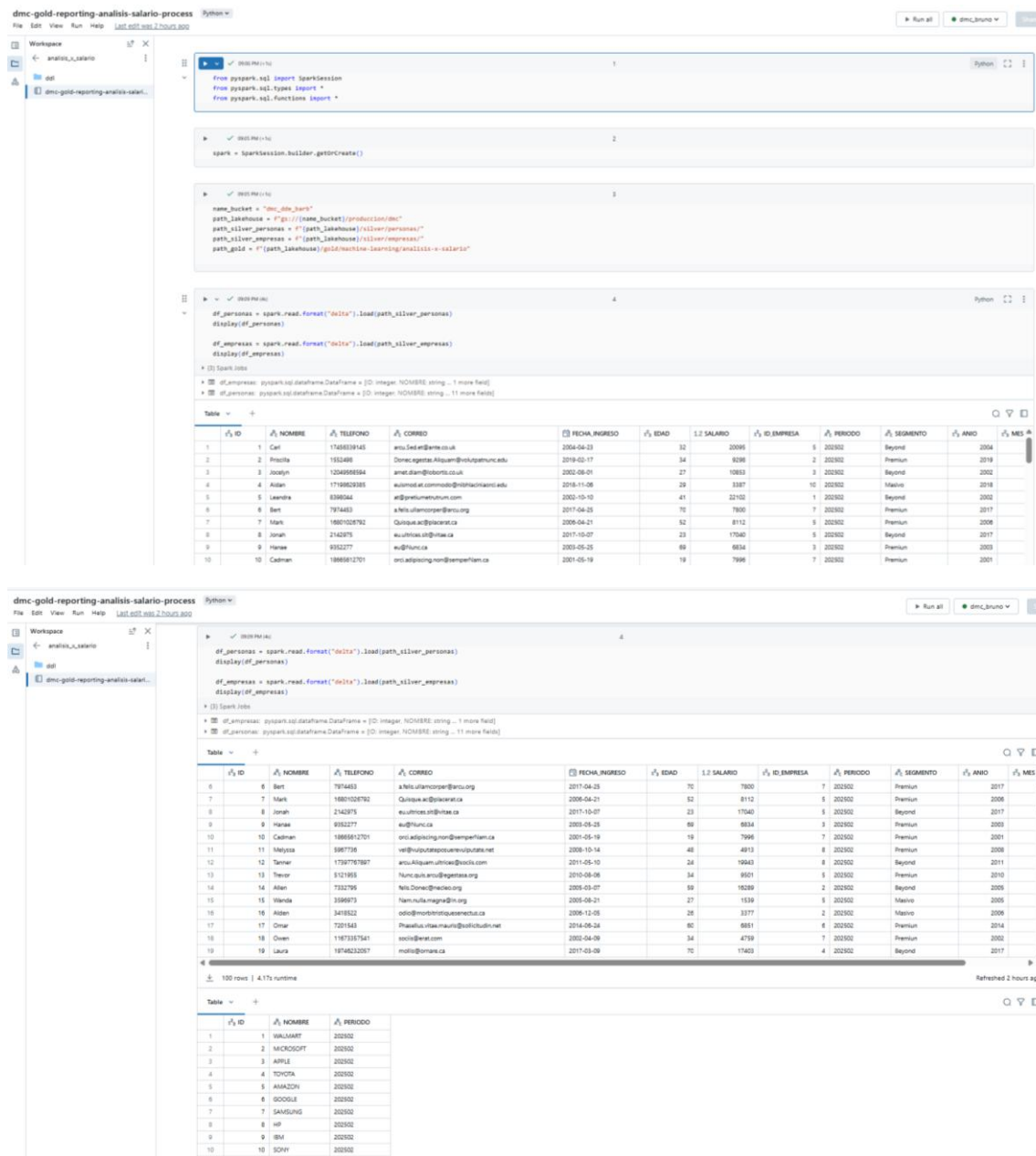
Filter by name prefix only

Filter

Filter objec

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

Poblando la capa Gold



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

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

```
new_bucket = "dmc_db_gold"
path_lakehouse = f"{path_bucket}/production/etl/"
path_silver_personas = f"{path_lakehouse}/silver/personas/"
path_silver_empresas = f"{path_lakehouse}/silver/empresas/"
path_gold = f"{path_lakehouse}/gold/etl/etl-reporting-analysis-salario/"
```

```
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)
```

ID	NOMBRE	TELEFONO	CORREO	FECHA_INGRESO	EDAD	SALARIO	ID_EMPRESA	PERIODO	SEGMENTO	AÑO	MES
1	Carl	1745638145	carl.sed@bmc.co.uk	2004-04-23	32	20095	5	202302	Beyond	2004	
2	Procta	1532489	Domecgentat.Akqum@vulqummc.edu	2019-02-17	34	9298	2	202302	Premium	2019	
3	Jocelyn	1204598564	arnet.dam@roborts.co.uk	2002-08-01	27	10853	3	202302	Beyond	2002	
4	Aidan	1718929385	auemod.at.commod@qimackmcc.edu	2016-11-06	29	3387	10	202302	Mativo	2016	
5	Leandra	439544	ar@priummcum.com	2002-10-10	41	27152	1	202302	Beyond	2002	
6	Bert	7974453	a.hicul.dymor@bmc.org	2017-04-25	70	7920	7	202302	Premium	2017	
7	Mark	18801026762	Quique.ac@bmc.co.uk	2006-04-21	52	8112	5	202302	Premium	2006	
8	Jonah	2142975	eu@lumca.ca	2017-10-07	23	17040	5	202302	Premium	2006	
9	Hanrae	6952277	eu@lumca.ca	2003-05-25	89	6834	3	202302	Premium	2003	
10	Cadman	18865812701	onr.adp@mcgmc.com	2001-05-19	19	7996	7	202302	Premium	2001	

```
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)
```

ID	NOMBRE	TELEFONO	CORREO	FECHA_INGRESO	EDAD	SALARIO	ID_EMPRESA	PERIODO	SEGMENTO	AÑO	MES
6	Bert	7974453	a.hicul.dymor@bmc.org	2017-04-25	70	7920	7	202302	Premium	2017	
7	Mark	18801026762	Quique.ac@bmc.co.uk	2006-04-21	52	8112	5	202302	Premium	2006	
8	Jonah	2142975	eu@lumca.ca	2017-10-07	23	17040	5	202302	Premium	2006	
9	Hanrae	6952277	eu@lumca.ca	2003-05-25	89	6834	3	202302	Premium	2003	
10	Cadman	18865812701	onr.adp@mcgmc.com	2001-05-19	19	7996	7	202302	Premium	2001	
11	Malyca	5967736	ver@vulqummcum.com	2008-10-14	48	4913	8	202302	Premium	2008	
12	Tenar	1739797887	arac.alqum.dymor@bmc.com	2011-05-10	34	19443	8	202302	Beyond	2011	
13	Tenar	5121955	hunc.ac@bmc.org	2010-09-26	34	8501	5	202302	Premium	2010	
14	Aidan	7327795	Nels.Domec@mcgmc.org	2005-03-07	69	16289	2	202302	Beyond	2005	
15	Wanda	3596973	ham.mulla.magna@bmc.org	2005-09-21	27	1539	5	202302	Mativo	2005	
16	Aidan	3418522	odm@mcgmc.com	2006-12-05	26	3377	2	202302	Mativo	2006	
17	Omar	7201543	Phasalus.vta@mauro@bmc.com	2014-09-24	80	6851	6	202302	Premium	2014	
18	Owen	11673357541	soch@bmc.com	2002-04-09	34	4759	7	202302	Premium	2002	
19	Laura	19745232007	mody@bmc.com	2017-03-09	70	17403	4	202302	Beyond	2017	

```
df_result_1 = spark.sql(sql)
```

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)
```


community.cloud.databricks.com/editor/notebooks/200277875424738376?w=178945041965433&command=2002778754247384

diagnostics

New

Workspace

Recent

Search

Cloning

Workflows

Compute

Machine Learning

Experiments

dmc-gold-reporting-analisis-salario-process

File Edit View Run Help

Workspace

analis_salaro

analis

dmc-gold-reporting-analisis-salaro

Python 3

Run all

Run

Full Run

10 rows | 1.41s runtime

Refreshed 2 hours ago

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

10

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

10

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

Schema

Details

History

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

periodo	nombre_empresa	sum_salario	avg_salario	avg_estado
202302	AMAZON	19869	9781.76714285714	41.2142857142857
202302	SONY	62012	9112.44444444444	40.8888888888889
202302	TOYOTA	19830	7637.675	38.875
202302	MICROSOFT	19837	11188.7671428571	38.7671428571429
202302	HP	73219	8146.05555555556	39.8888888888889
202302	WALMART	79804	11326.1428571429	35.8571428571429
202302	IBM	91476	12739.8888888889	37.6666666666667
202302	SAVINGS	385710	11858.8888888889	34.5555555555556
202302	GOOGLE	130243	15403.3769230769	50
202302	APPLE	151700	13790.8000000000	35.8333333333333

10 rows | 1.37s runtime

Refreshed 2 hours ago

[Shift+Enter] to run and move to next cell
[Enter] to open the current pointer
[Esc] to view all keyboard shortcuts

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.

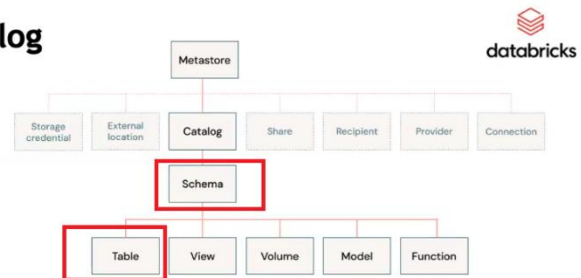
Catálogo: los catálogos son el nivel más alto en la jerarquía de datos (catálogo > esquema > tabla/vista/volumen) administrado por el metaalmacén de Unity Catalog. Están pensados como la unidad principal de aislamiento de datos en un modelo típico de gobierno de datos de Databricks.

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



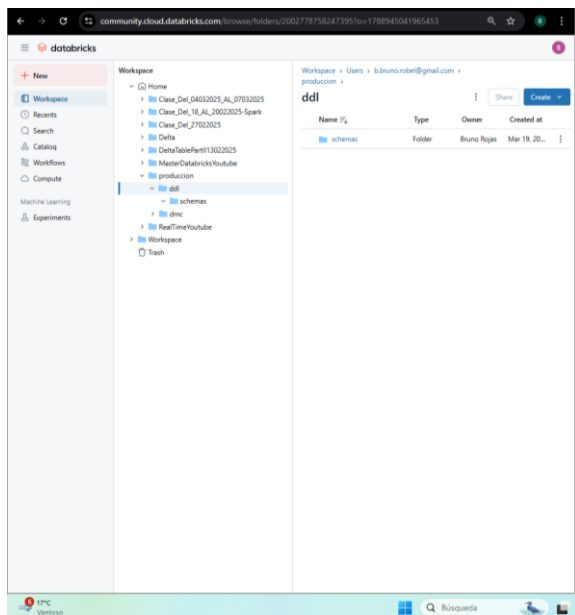
• **Vistas:** una vista es un objeto de solo lectura derivado de una o más tablas y vistas en un metastore.

• **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



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.

The screenshot shows a Databricks notebook with three SQL queries. The first query, 'show catalogs;', returns one row with 'spark_catalog'. The second query, 'show databases;', returns one row with 'default'. The third query, 'show tables in default;', returns no rows.

```
1
show catalogs;

Table
  catalog
1 spark_catalog

2
show databases;

Table
  databaseName
1 default

3
show tables in default;

Table
  database  tableName  temporary
No rows returned
```

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

The screenshot shows a Databricks notebook with three SQL queries to create schemas. The first query creates the 'bronze' schema, the second creates the 'silver' schema, and the third creates the 'gold' schema if it does not exist.

```
3
show tables in default;

Table
  database  tableName  temporary
No rows returned

4
create schema bronze
comment "Descripción del esquema"
LOCATION 'gs://dmc_ahm_barb/production/dmc/bronze';
OK

5
create schema silver
comment "Descripción del esquema"
LOCATION 'gs://dmc_ahm_barb/production/dmc/silver';
OK

6
create schema if not exists gold
comment "Descripción del esquema"
LOCATION 'gs://dmc_ahm_barb/production/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.

The screenshot shows the Databricks SQL interface. The top panel displays the SQL code for creating an external Delta table named 'bronze.personas'. The code specifies columns: ID (STRING), NOMBRE (STRING), TELEFONO (STRING), CORREO (STRING), FECHA_INGRESO (STRING), EDAD (STRING), SALARIO (STRING), ID_EMPRESA (STRING), and LOCATION. The table is created using the Delta format and is located at 'gs://dmc_data_bark/production/dmc/bronze/personas'. The bottom panel shows the execution results, displaying a table with 14 rows and 4 columns: ID, NOMBRE, TELEFONO, and CORREO.

ID	NOMBRE	TELEFONO	CORREO
1	Carl	1-745-633-9145	carl.sed@bertco.uk
2	Priscia	133-3488	Concepcion.Aquino@imphelpmcastu
3	Joselyn	1-204-856-8594	amel.dam@bortpico.uk
4	Aidan	1-719-862-8385	auimodul.commod@rhmactmactau
5	Leandra	838-8244	at@retumetum.com
6	Bert	797-4483	Afrilulancop@arcu.org
7	Mark	1-680-152-6792	Quocuew@pacom.ca
8	Jaral	214-2979	eu@troung@fiteas
9	Harar	935-2277	eu@num.ca
10	Cadman	1-866-561-2701	en@edlincingnon@emperiam.ca
11	Melissa	596-7738	vt@vupatoposumvupate.net
12	Tanner	1-739-776-7897	arcu.Aliquam@ruos@ociu.com
13	Trevar	512-1955	huncupisarcu@gestas.org
14	Aiden	733-2785	Rea.Donac@reacresing

The screenshot shows the Databricks SQL interface. The top panel displays the SQL code for creating an external Delta table named 'bronze.empresas'. The code specifies columns: ID (STRING) and NOMBRE (STRING). The table is created using the Delta format and is located at 'gs://dmc_data_bark/production/dmc/bronze/empresas'. The bottom panel shows the execution results, displaying a table with 10 rows and 2 columns: ID and NOMBRE.

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

The screenshot shows the Databricks workspace interface. On the left, a sidebar contains navigation options like 'New', 'Workspace', 'Recent', 'Search', 'Catalog', 'Workflow', 'Compute', 'Machine Learning', and 'Experiments'. The main area displays a SQL query for creating a bronze table named 'bronze.transacciones' with columns: ID_PERSONA, ID_EMPRESA, MONTO, FECHA, and PUNTO. The query uses a Delta table and is located at 'gs://dms_dms_barb/produccion/dms/bronze/transacciones'. Below the query, a Spark job is shown with a table preview of 10,000 rows (truncated) and a 1.6s runtime. The table has columns: ID_PERSONA, ID_EMPRESA, MONTO, and FECHA. A video call overlay is present on the right side of the screen, showing participants Bruno Rojas, Carmen Milagro, and others.

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

The screenshot shows the Databricks workspace interface. On the left, a sidebar contains navigation options like 'New', 'Workspace', 'Recent', 'Search', 'Catalog', 'Workflow', 'Compute', 'Machine Learning', and 'Experiments'. The main area displays a SQL query for creating a silver table named 'silver.personas' with columns: ID, ID_EMPRESA, NOMBRE, EDAD, TELEFONO, CORREO, SALARIO, FECHA_INGRESO, MES, DIA, SEGUNDO, and PERIODO. The query uses a Delta table and is located at 'gs://dms_dms_barb/produccion/dms/silver/personas'. Below the query, a Spark job is shown with a table preview of 10 rows and a 1.6s runtime. The table has columns: ID, NOMBRE, TELEFONO, CORREO, FECHA_INGRESO, EDAD, SALARIO, ID_EMPRESA, PERIODO, and SEGUNDO.

community.cloud.databricks.com/notebooks/1820682946755018?uc=1780945041965453#command=1820682946755019

dotabricks

File Edit View Run Help Last edit: 18/03/2025 22:37

Workspace

Workspace

dotabricks-sap-empress-ddl

dotabricks-sap-empress-ddl

1

```
CREATE EXTERNAL TABLE IF NOT EXISTS silver_empress;  
USING DELTA;  
PARTITIONED BY (PERIODO);  
LOCATION 'gs://dms_data_bak6/production/dms/silver/empress';
```

2

```
SELECT * FROM silver_empress;
```

Table

ID	NOMBRE	PERIODO
1	WALMART	202002
2	MICROSOFT	202002
3	APPLE	202002
4	YOHDA	202002
5	AMAZON	202002
6	GOOGLE	202002
7	SAMSUNG	202002
8	HP	202002
9	IBM	202002
10	SONY	202002

10 rows | 1.13s runtime

3

```
show partitions silver_empress;
```

Table

PERIODO
202002

1 row | 0.0s runtime

18°C Ventoso

Búsqueda

ESP LAA

22:37 18/03/2025

community.cloud.databricks.com/notebooks/1820682946755018?uc=1780945041965453#command=1820682946755027

dotabricks

File Edit View Run Help Last edit: 18/03/2025 22:37

Workspace

Workspace

dotabricks-sap-transacciones-ddl

dotabricks-sap-transacciones-ddl

1

```
CREATE EXTERNAL TABLE IF NOT EXISTS silver_transacciones;  
USING DELTA;  
PARTITIONED BY (MES, DIA);  
LOCATION 'gs://dms_data_bak6/production/dms/silver/transacciones';
```

2

```
show partitions silver_transacciones;
```

Table

MES	DIA
2018	1
2018	22
2018	23

3 rows | 0.82s runtime

3

```
SELECT * FROM silver_transacciones LIMIT 3;
```

Table

ID_PERSONA	ID_EMPRESA	MONTO	FECHA	MES	DIA
18	3	1383	2018-01-21	2018	1
30	6	2391	2018-01-21	2018	1
47	2	2280	2018-01-21	2018	1

3 rows | 0.82s runtime

18°C Ventoso

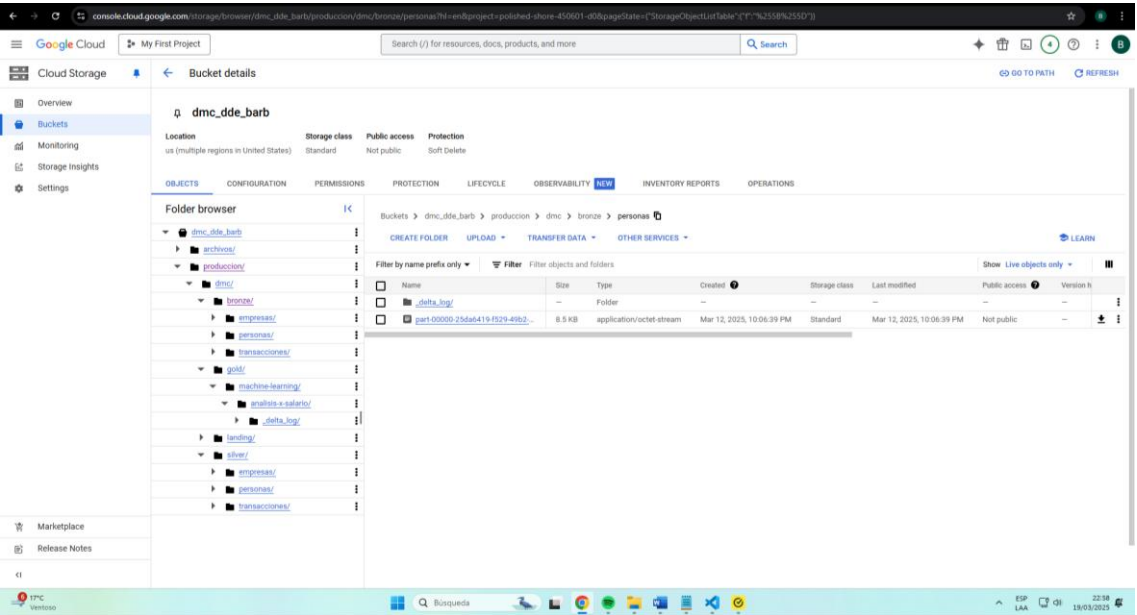
Búsqueda

ESP LAA

22:37 18/03/2025

Creamos una subcarpeta dentro del bucket para la capa gold

Gold→machine-learning→analysis-x-salario



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

