# AZURE DATA FACTORY / AZURE DATABRICKS


# INTEGRANTES

Carlos Eduardo Denett

Cecilia Marcela Espada

Federico Pfund

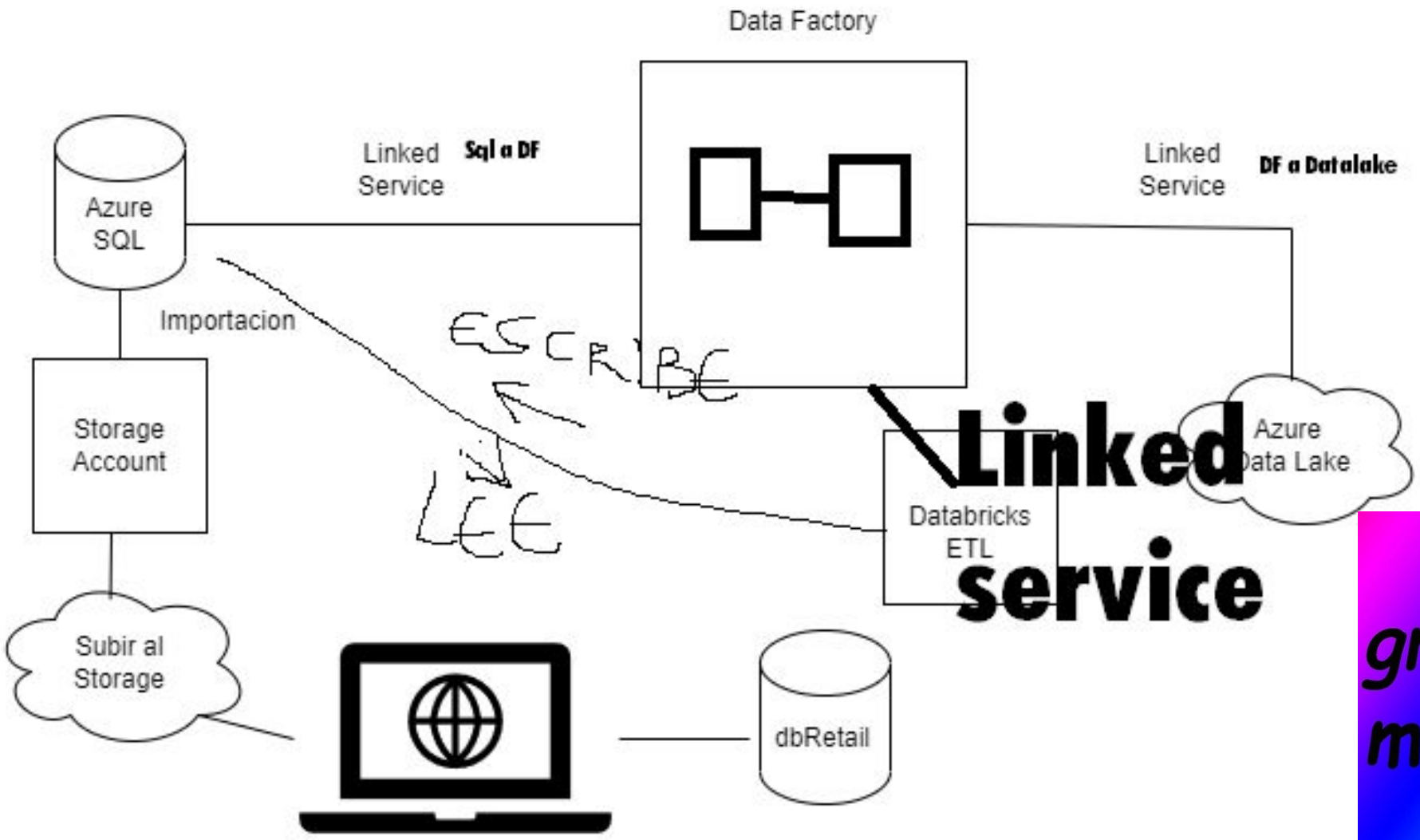
Juan Martín Elena

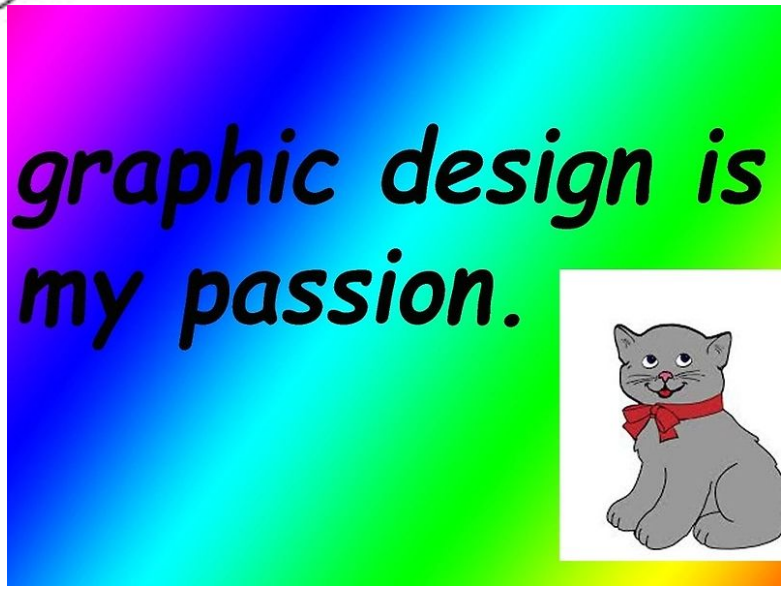
Agustín Fernández

Patricio Perrone

# Review Sprint 2 - Grupo 1
# Proyecto Integrador Final

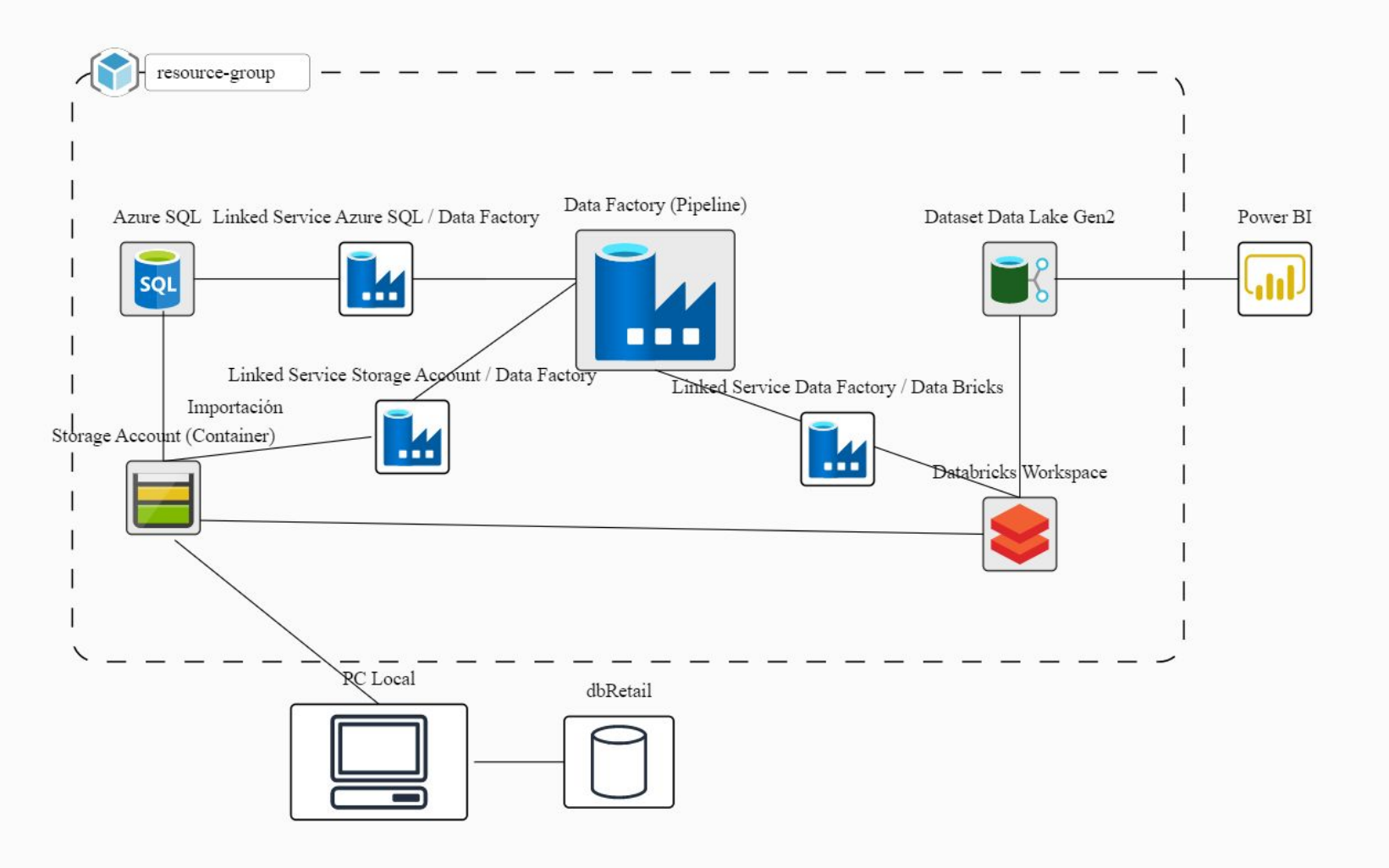
# Cambios de Arquitectura - Antigua



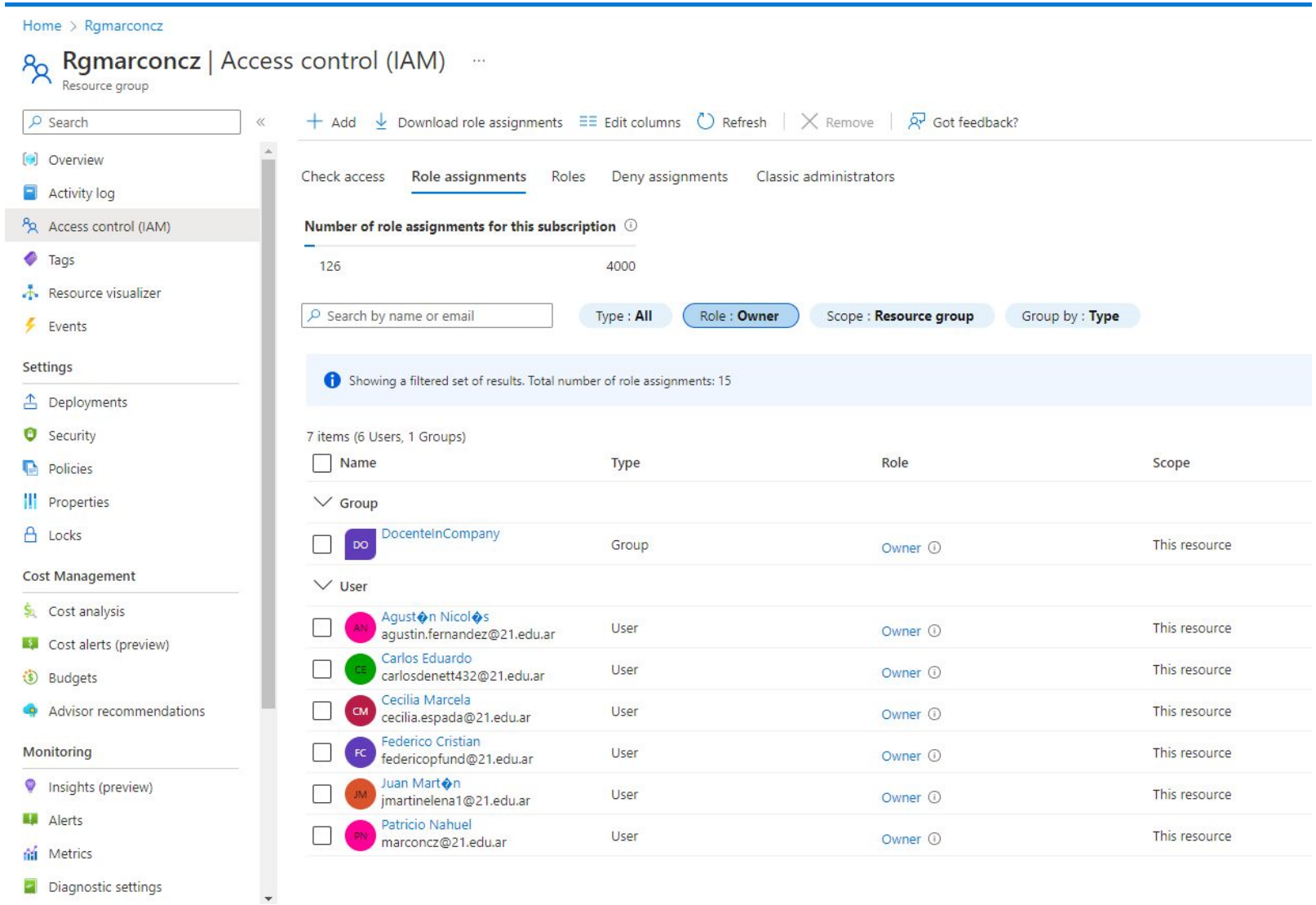
# Cambios de Arquitectura

- En este segundo sprint, realizamos cambios en la arquitectura para trabajar directamente con archivos csv y evitar el gasto de consumo continuo del recurso Azure SQL. También logramos mejorar los tiempos de ejecución.



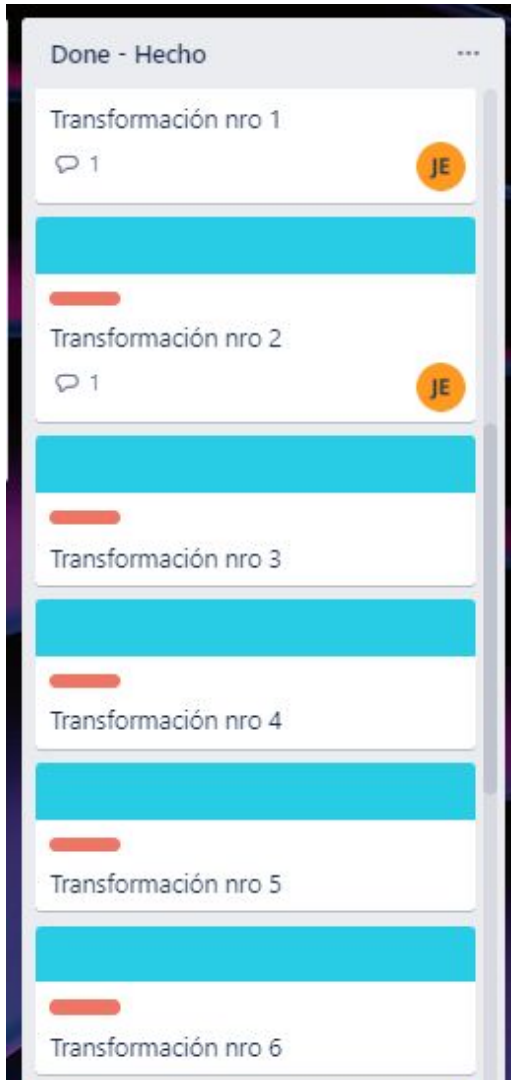
# Grupo de Recursos Compartidos

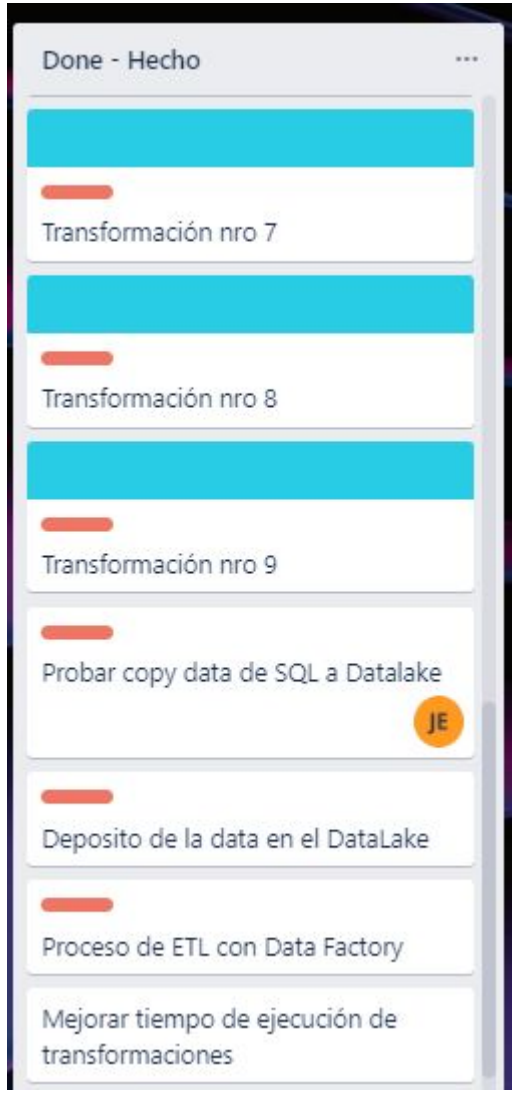
- Decidimos utilizar un sólo grupo de recursos y compartir los accesos con el resto del equipo para ahorrar presupuesto y focalizarnos en una sola solución en común.



# Tareas del Sprint 2

- En este sprint nos concentramos en el cambio de la arquitectura, el pipeline con las transformaciones y la presentación de los datos de manera visual
- Trello oficial con todos los grupos.
- Trello del grupo



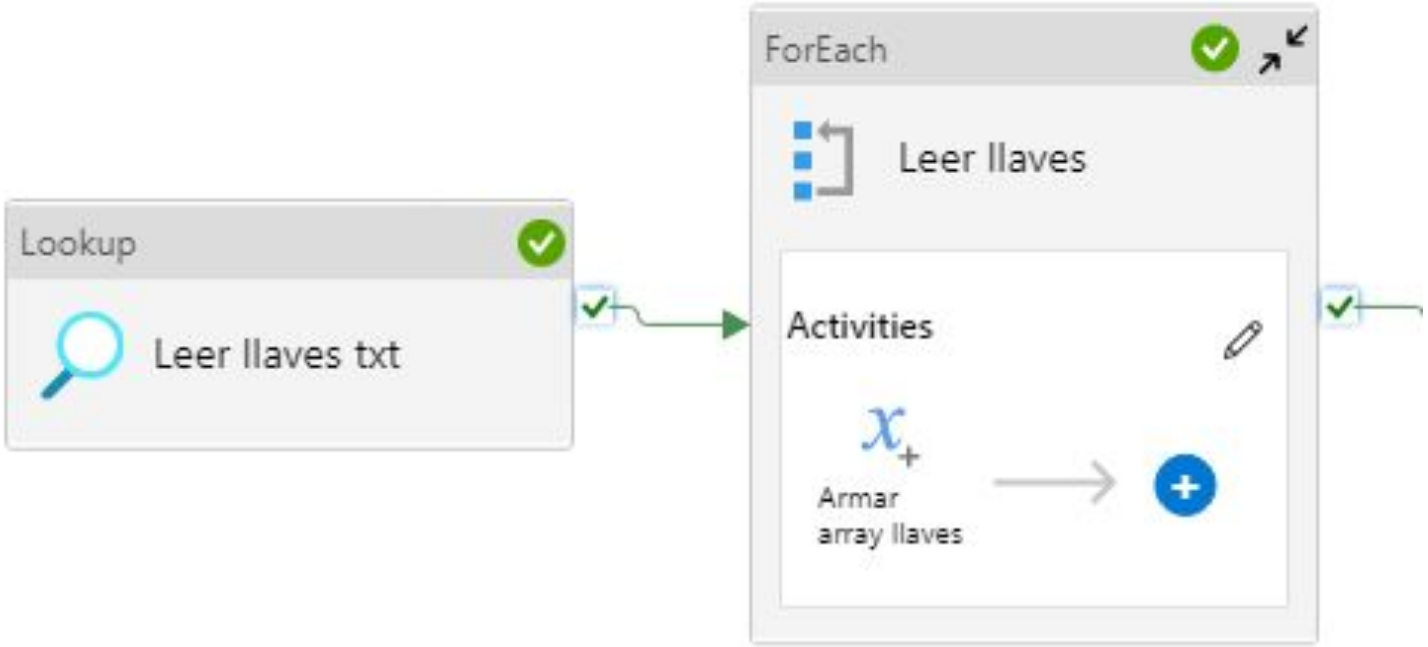
# Pipeline ETL en Data Factory

# Creación de Pipeline en Data Factory

- Desarrollamos un Pipeline que crea las tablas en CSV si no existen y se las transfiere a Databricks.

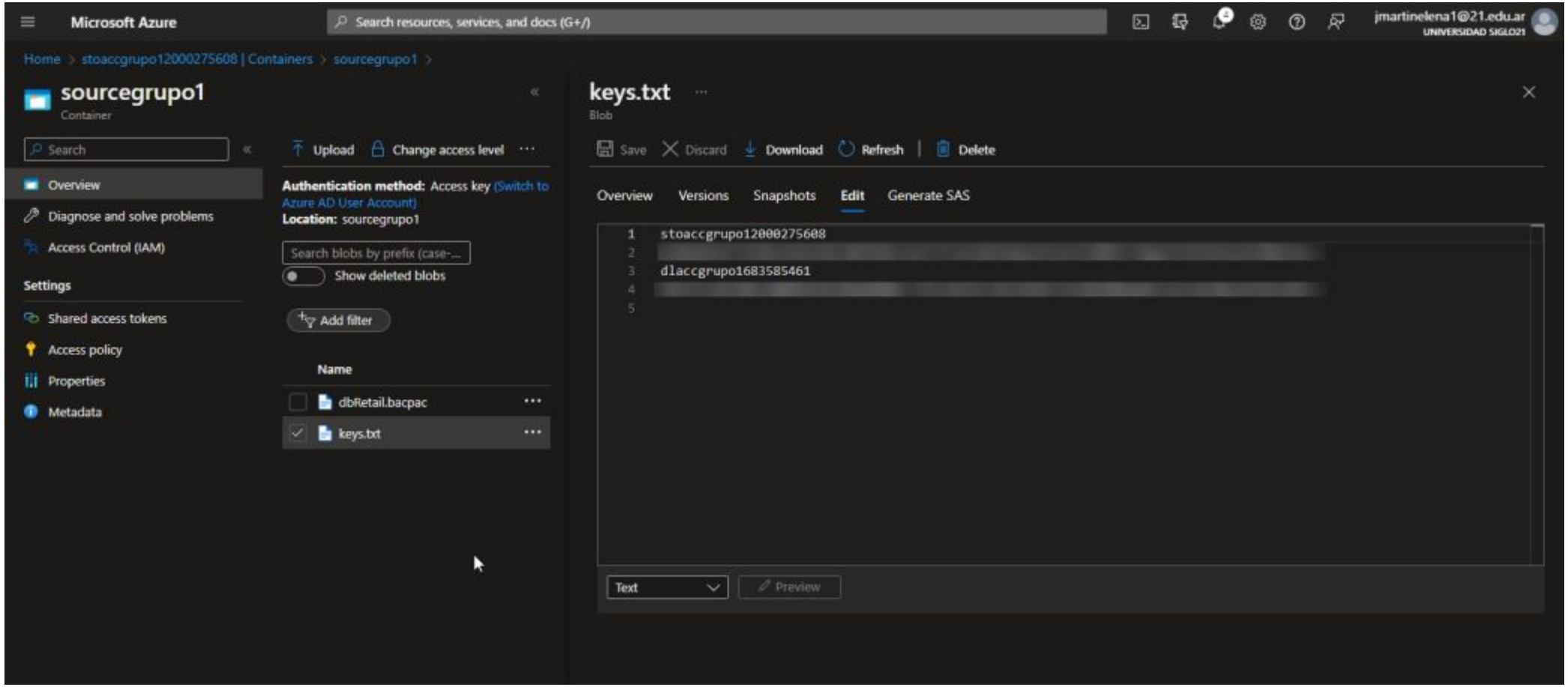
# Lectura de datos de acceso

- En este primer paso, se accede a un archivo txt donde se almacenan los datos de acceso que emula un Key Vault, como lo son nombres del storage account y data lake, y sus access keys.



# Lectura de datos de acceso

Archivo txt dentro del Blob Storage:



# Lectura de datos de acceso

Vista previa del archivo del archivo de texto tras el Lookup activity:

Preview data

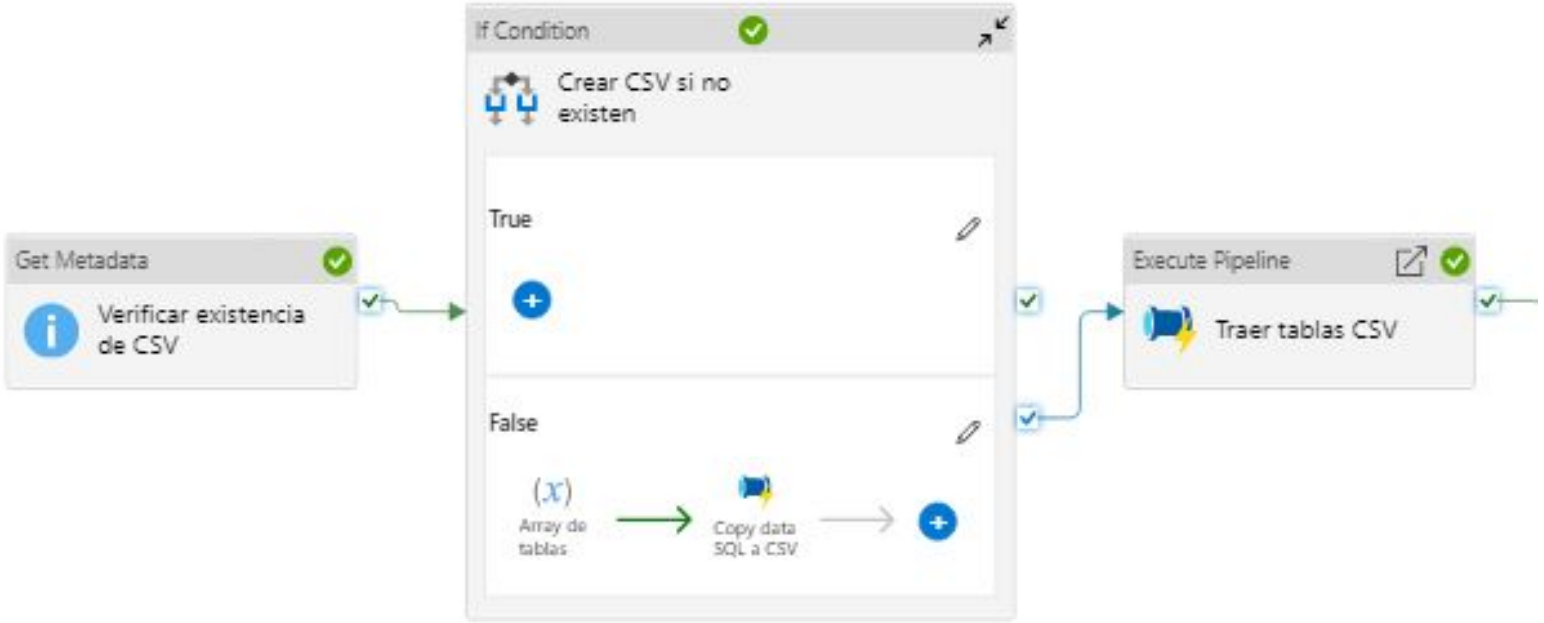
Linked service: AzureBlobStorage1

Object: keys.txt

| | Prop_0 |
|---|---|
| 1 | stoaccgrupo12000275608 |
| 2 | ql8AmjPv |
| 3 | dlaccgrupo1683585461 |
| 4 | 8SPIZI |

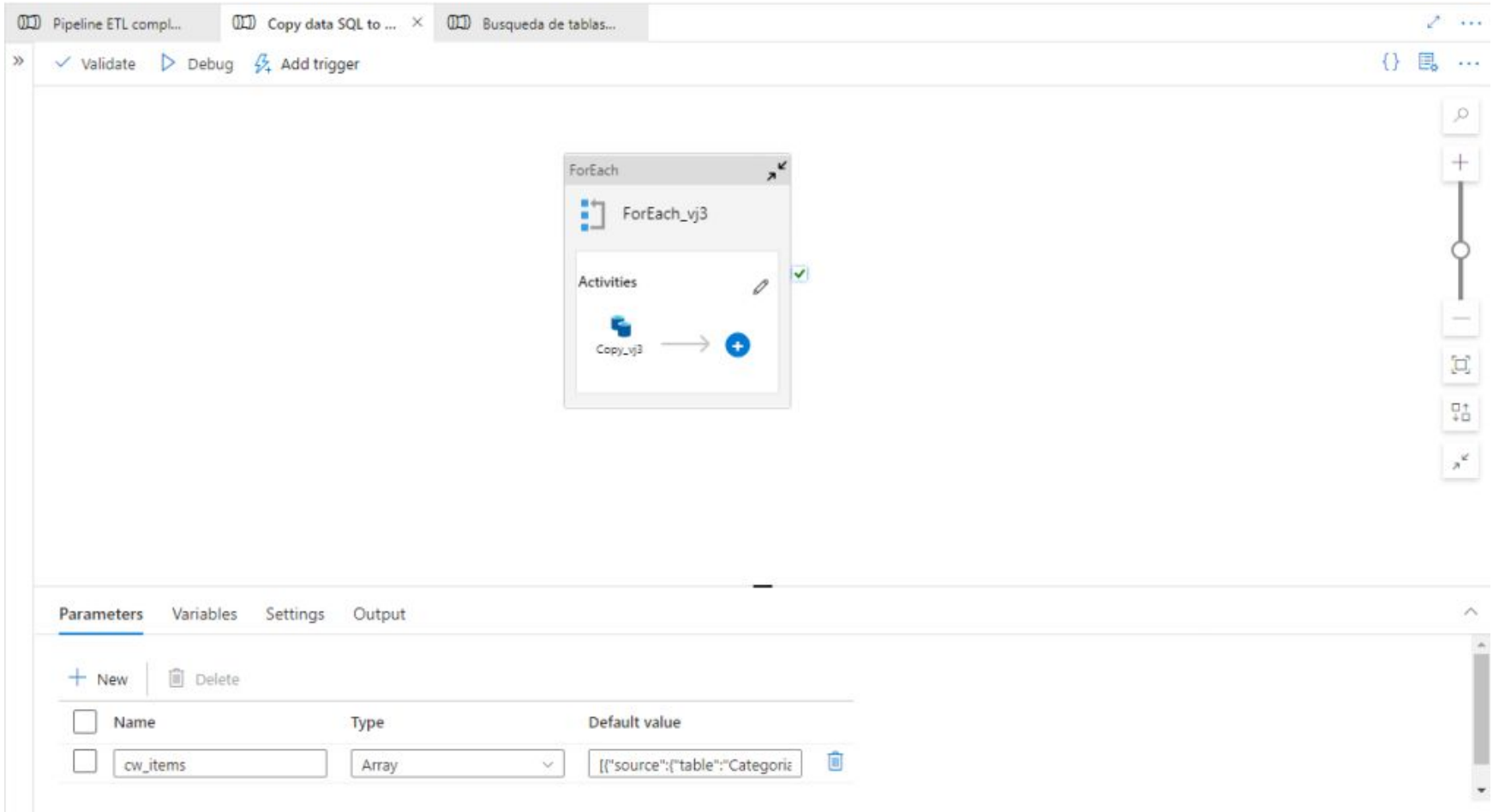
# Creación de csv con base de datos

- El segundo paso verifica si las tablas ya existen en el Blob Storage para no ejecutar el Copy Data de no ser necesario.



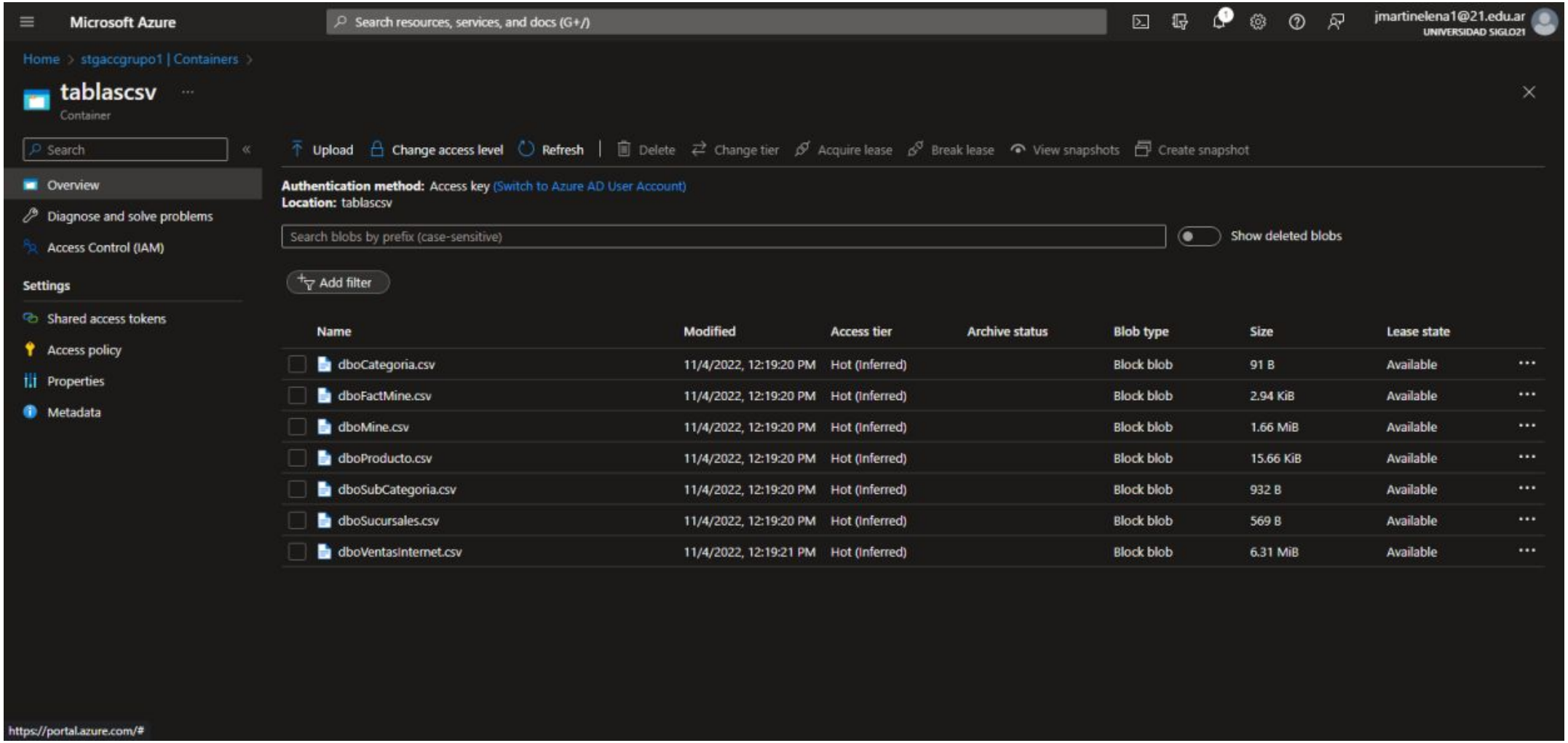
# Creación de csv con base de datos

- Luego, convierte las tablas de la base de datos a tablas en archivos csv (en caso de que no existan). Este ForEach se integra al Pipeline ETL Completo a través de un Pipeline Activity.



# Creación de csv con base de datos

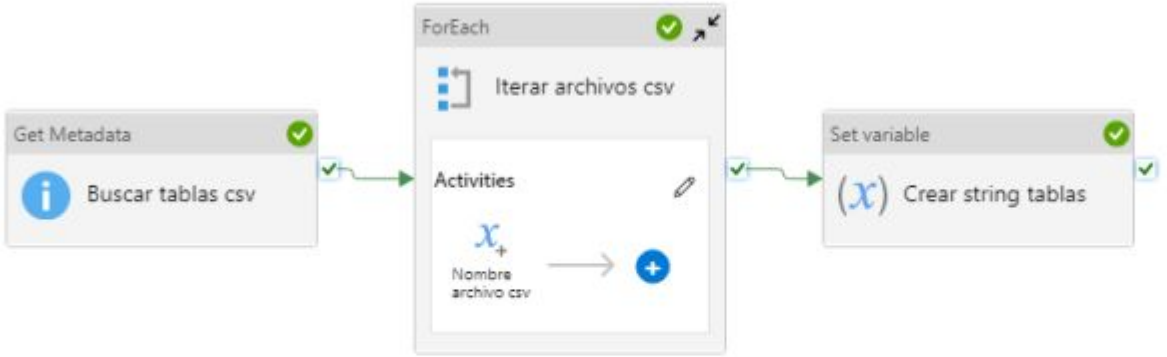
- Los archivos csv se depositan en un container en el storage account.



# Creación de csv con base de datos

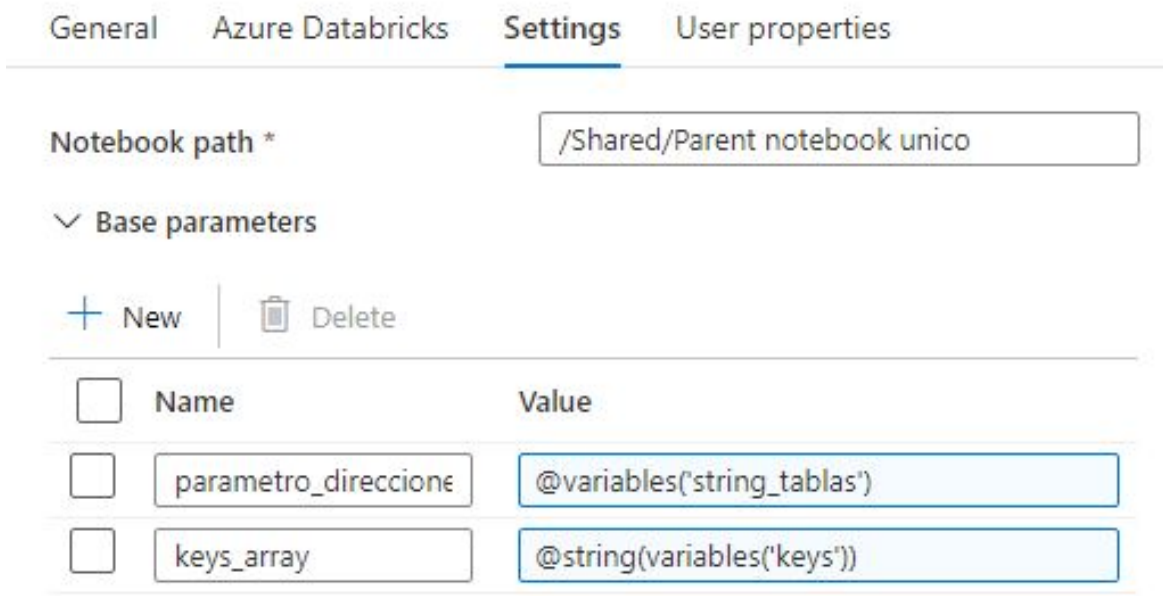
- Después, se preparan las tablas para enviar a Databricks como parámetro.


Despliegue de “Traer tablas CSV”



# Envío de tablas csv a Databricks

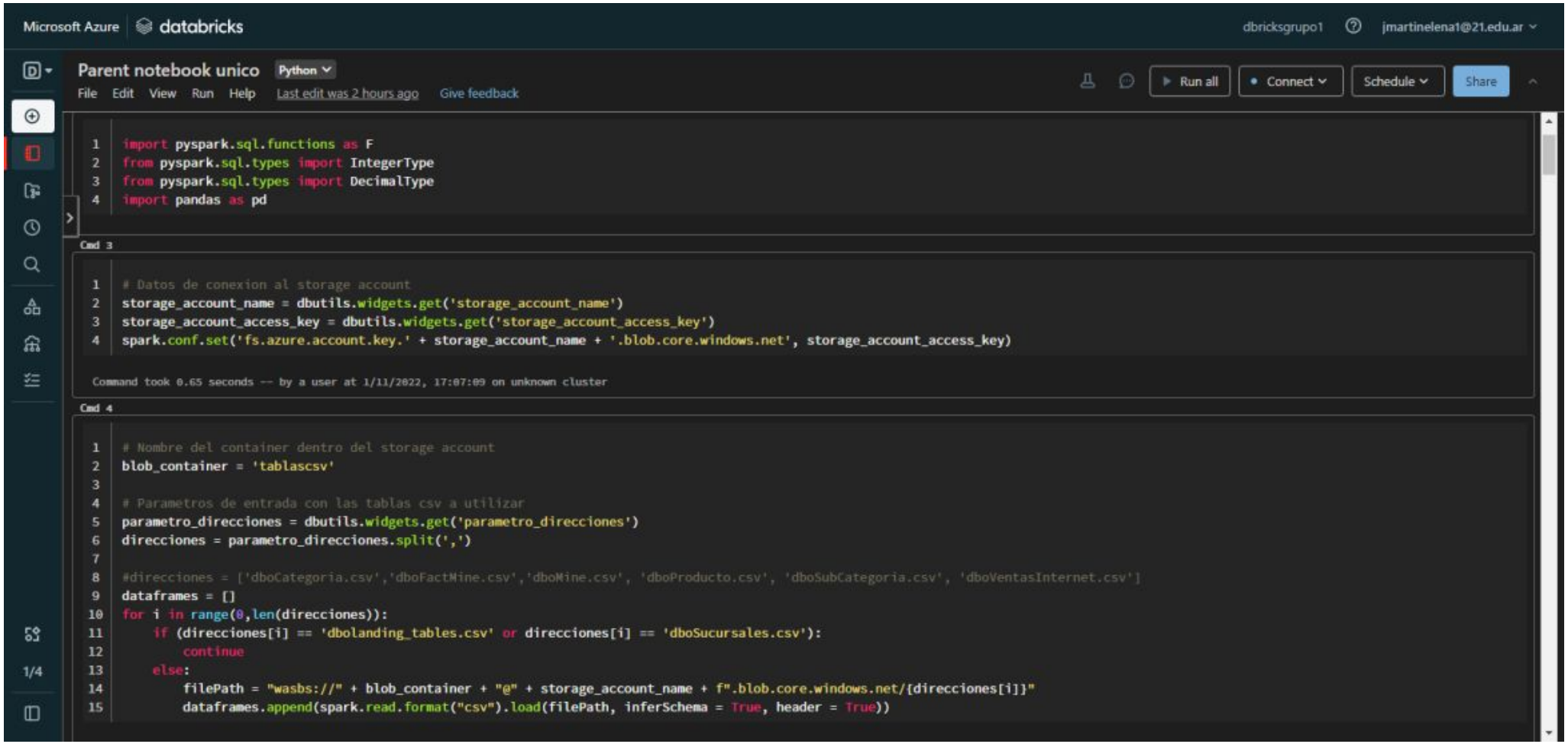
- En este último paso, se envían las tablas y parámetros a una notebook de Databricks.

# Transformaciones en Databricks

# Transformación de tablas en Databricks

- Tomamos los csv y realizamos las transformaciones. Optamos por usar un solo notebook, en lugar de un notebook padre que llama a notebooks hijos, para mejorar los tiempos de ejecución.



# Ejemplo de Transformación 3

- A modo de ejemplo, se muestra la Transformación 3 en Databricks:

```
1  # Transformacion 3
2  selectedColumns = dataframes[2].select("Country","FirstName","LastName","Age")
3  transformaciones.append(selectedColumns)
4  display(selectedColumns)
5  selectedColumns.printSchema()
```

Table +

| | Country | FirstName | LastName | Age |
|---|---|---|---|---|
| 1 | Australia | Curtis | Lu | 41 |
| 2 | Australia | Curtis | Lu | 41 |
| 3 | Australia | Curtis | Lu | 41 |
| 4 | Australia | Jill | Jimenez | 59 |
| 5 | Australia | Jill | Jimenez | 59 |
| 6 | Australia | Jill | Jimenez | 59 |
| 7 | Australia | Bethany | Yuan | 52 |

Truncated results, showing first 1,000 rows. | 1.13 seconds runtime

```
root
 |-- Country: string (nullable = true)
 |-- FirstName: string (nullable = true)
 |-- LastName: string (nullable = true)
 |-- Age: integer (nullable = true)
```

# Salida al Data Lake Storage

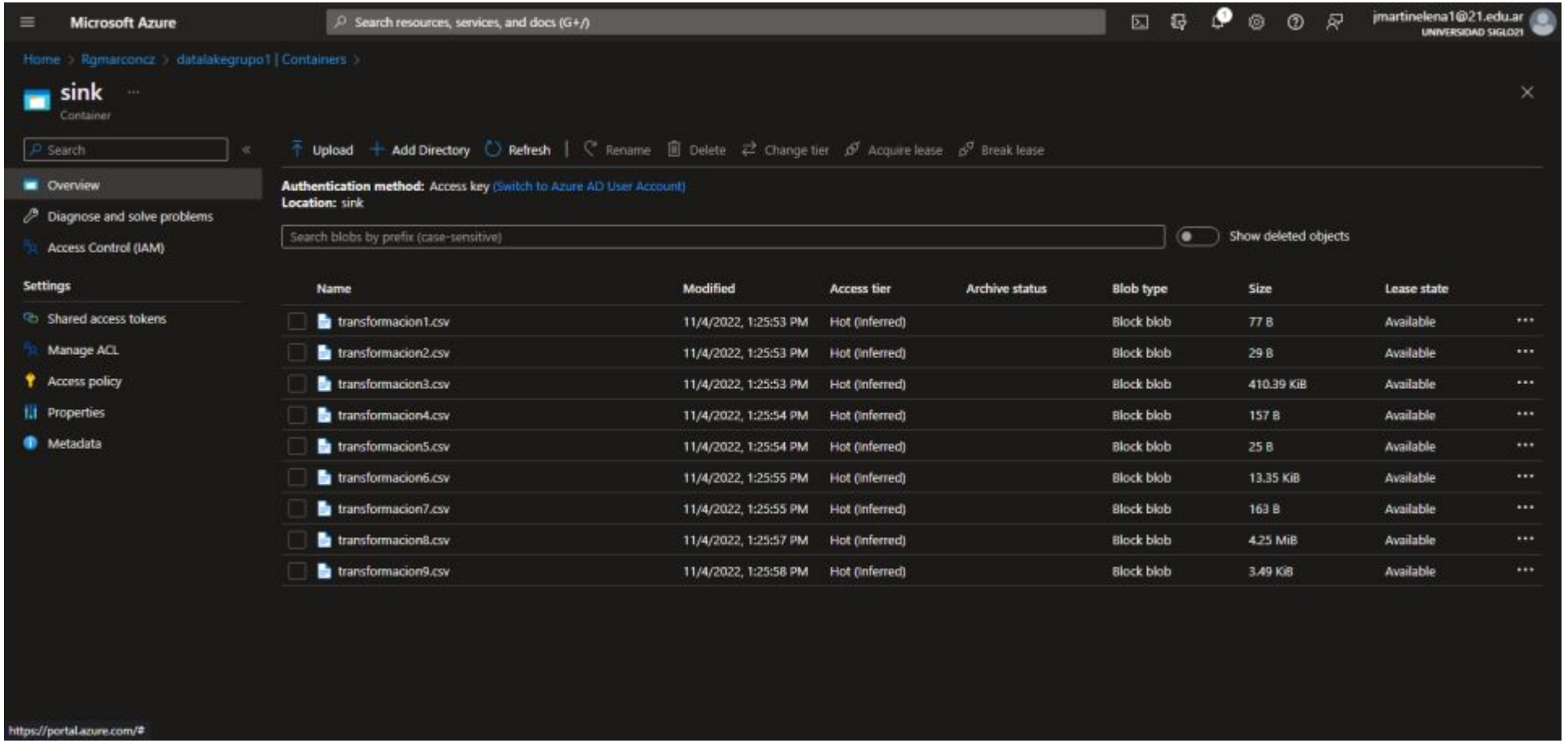
- Configuramos la salida desde Databricks al Data Lake Storage, y escribimos las tablas en formato CSV.

Cmd 21

```
# Cargar las tablas en formato CSV dentro del data lake
datalake_container = 'sink'
datalake_account_name = dbutils.widgets.get('datalake_account_name')
datalake_access_key = dbutils.widgets.get('datalake_access_key')

for i in range(len(transformaciones)):
    transformaciones[i].toPandas().to_csv(f'abfs://{datalake_container}@{datalake_account_name}.dfs.core.windows.net/transformacion{i+1}.csv',storage_options = {'account_key':
datalake_access_key} ,index=False)
```

# Salida de Data Lake

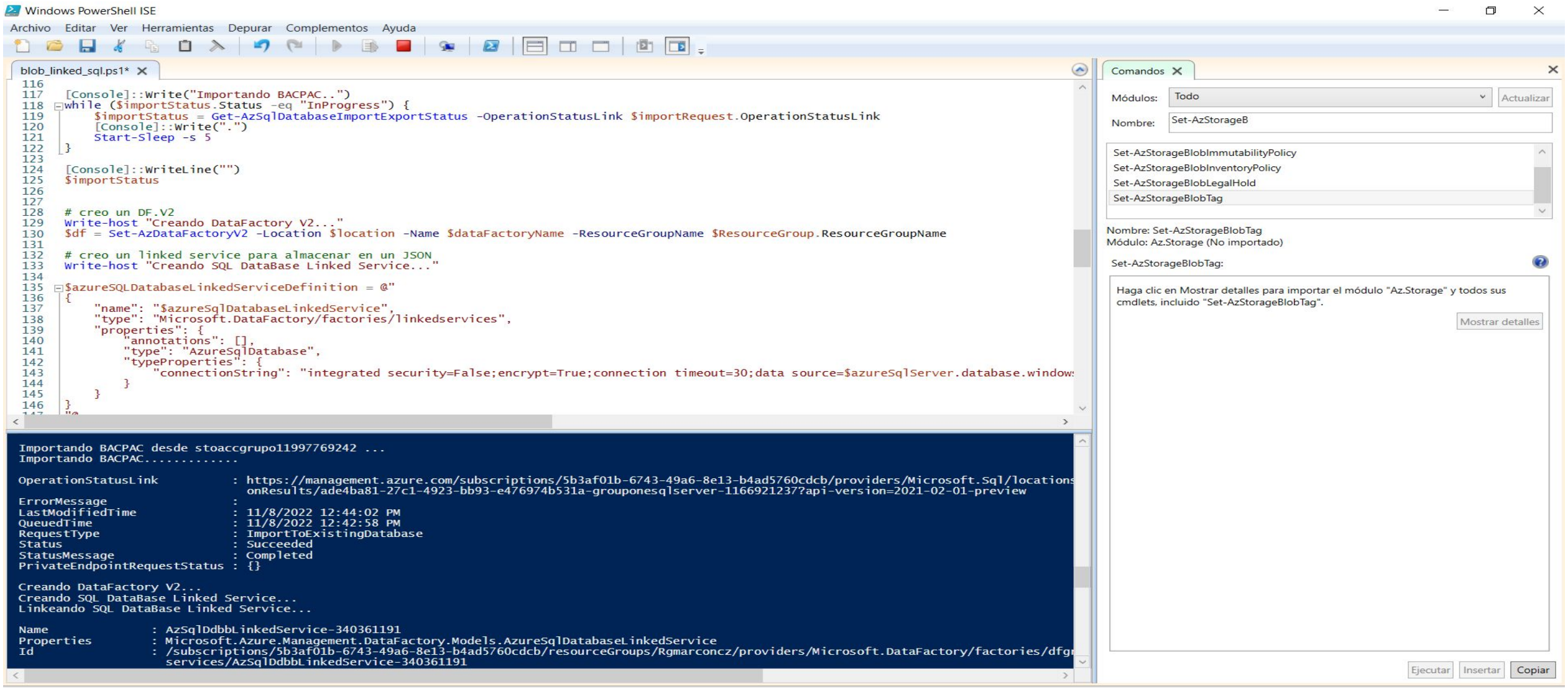
Tablas transformadas en formato CSV en el Data Lake Storage:



# Adicionales

# Script para automatizar creación de recursos

https://youtu.be/b8gvkS6xk9Y

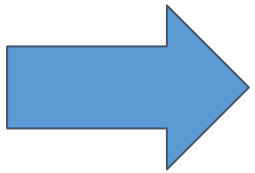


# Script para automatizar creación de recursos

https://youtu.be/b8gvkS6xk9Y

| Name | Type | Location |
| --- | --- | --- |
| dbricksgrupo11270365620 | Azure Databricks Service | West US 3 |
| dfgrupo1siglo21367831347 | Data factory (V2) | West US 3 |
| dlaccgrupo11160844581 | Storage account | West US 3 |
| grouponesqlserver-331627171 | SQL server | West US 3 |
| groupOneSqlServerDatabase (grouponesqlserver-331627171/groupOneSqlServerDatabase) | SQL database | West US 3 |
| stoaccgrupo11803122672 | Storage account | West US 3 |

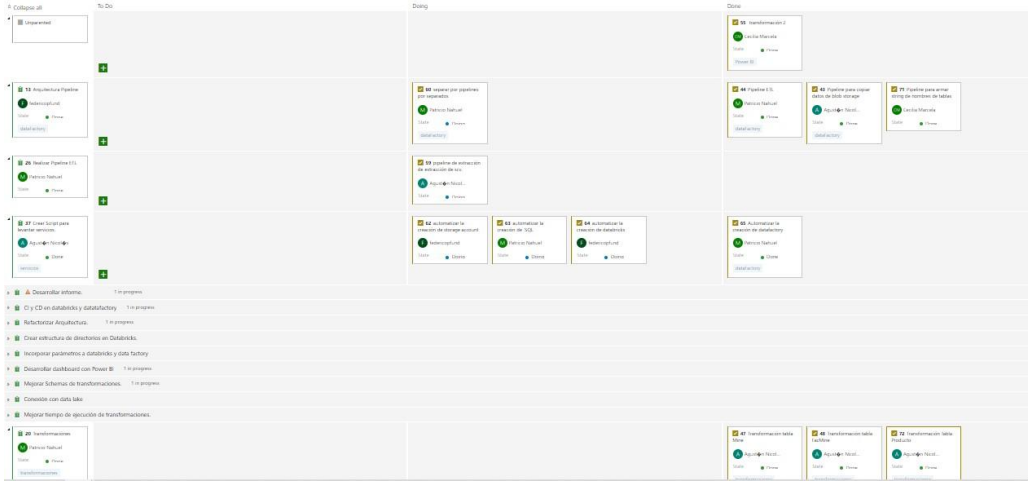
# ¿Y las claves del almacenamiento? Simulando el Keyvault

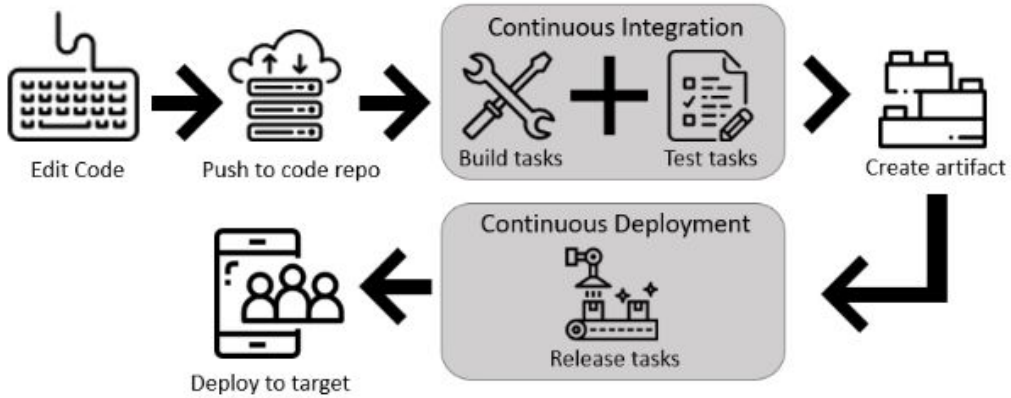



```powershell
# extraigo claves de las dos storage accounts
$storageAccountKey = `
    (Get-AzStorageAccountKey `
    -ResourceGroupName $ResourceGroup.ResourceGroupName `
    -Name $StorageAccountName).Value[0]

$dataLakeAccountKey = `
    (Get-AzStorageAccountKey `
    -ResourceGroupName $ResourceGroup.ResourceGroupName `
    -Name $azureDataLakeName).Value[0]

# guardo las claves en un txt
Write-host "Guardando claves en un archivo TXT"
"$StorageAccountName"|Set-Content keys.txt
"$storageAccountKey"|Add-Content keys.txt
"$azureDataLakeName"|Add-Content keys.txt
"$dataLakeAccountKey"|Add-Content keys.txt

# subo el .txt al blob
Write-host "Subiendo claves a $ContainerName"
$Keys = @{
  File            = ".\keys.txt"
  Container       = $ContainerName
  Blob            = "keys.txt"
  Context         = $StorageAccount.context
  StandardBlobTier = 'Hot'
}
Set-AzStorageBlobContent @Keys
```

# Integración con Azure DevOps

| Order | ID | Title | Assigned To | State | Tags |
|---|---|---|---|---|---|
| 2 | 13 | Arquitectura Pipeline | federicopfund | Done | dataFactory |
| | 44 | Pipeline ETL | Patricio Nahuel | Done | dataFactory |
| | 60 | separar por pipelines por separados | Patricio Nahuel | Doing | dataFactory |
| | 43 | Pipeline para copiar datos de blob storage | Agust◆n Nic... | Done | dataFactory |
| | 71 | Pipeline para armar string de nombres de tablas | Cecilia Marcela | Done | |
| 3 | 26 | Realizar Pipeline ETL | Patricio Nahuel | Done | |
| | 59 | pipeline de extracción de extracción de scv. | Agust◆n Nic... | Doing | |
| 4 | 37 | Crear Script para levantar servicios. | Agust◆n Nic... | Done | servicios |
| | 62 | automatizar la creación de storage account | federicopfund | Doing | |
| | 63 | automatizar la creación de SQL | Patricio Nahuel | Doing | |
| | 64 | automatizar la creación de databricks | federicopfund | Doing | |
| | 65 | Automatizar la creación de datafactory | Patricio Nahuel | Done | dataFactory |
| 5 | 17 | Desarrollar informe. | Agust◆n Nic... | Done | Informe verciones |
| | 45 | Informe de las versiones | Patricio Nahuel | Doing | Informe |
| 6 | 52 | CI y CD en databricks y datatafactory | Cecilia Marcela | Doing | |
| 7 | 16 | Refactorizar Arquitectura. | Cecilia Marcela | Doing | Rafactors |
| | 56 | sacar el servicio de SQL | | Doing | |
| | 61 | eliminación de un link servicie con SQL | | Done | |
| 8 | 11 | Crear estructura de directorios en Databricks. | Cecilia Marcela | Done | |
| 9 | 12 | Incorporar parámetros a databricks y data factory | Agust◆n Nic... | Done | |
| 10 | 53 | Desarrollar dashboard con Power BI | Agust◆n Nic... | Doing | |

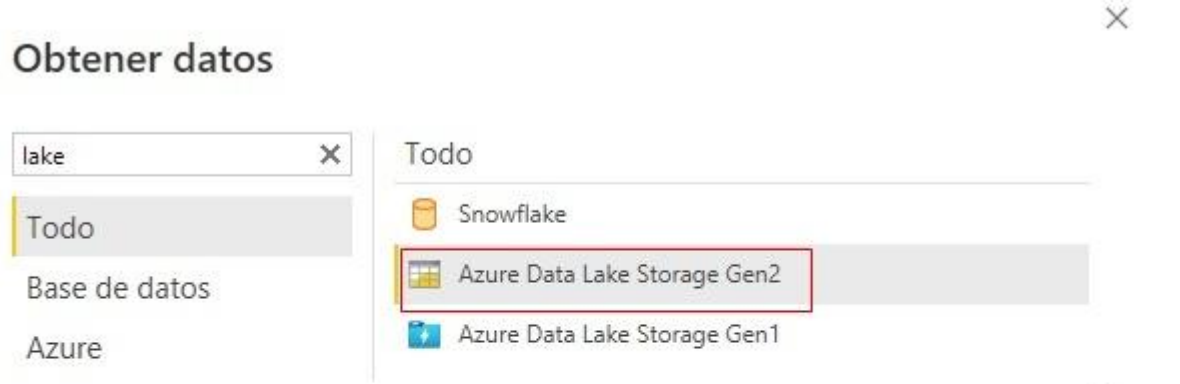




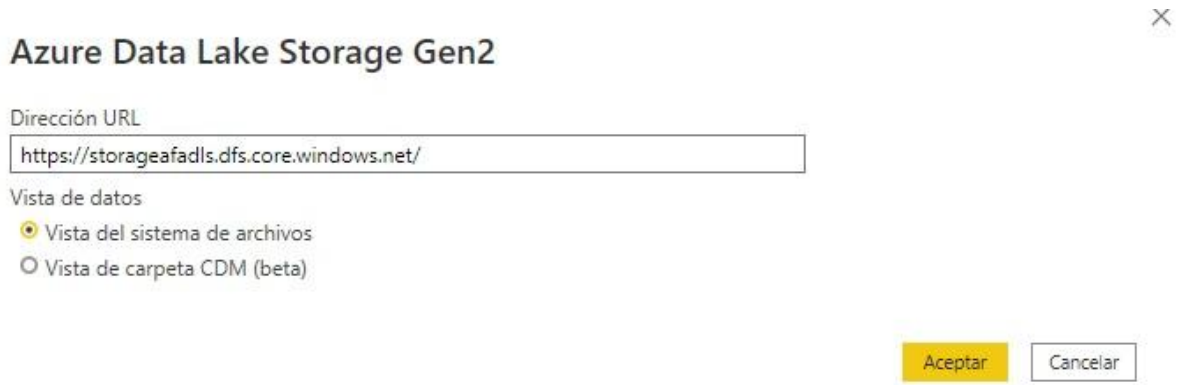
# Visualización de Datos con Power BI

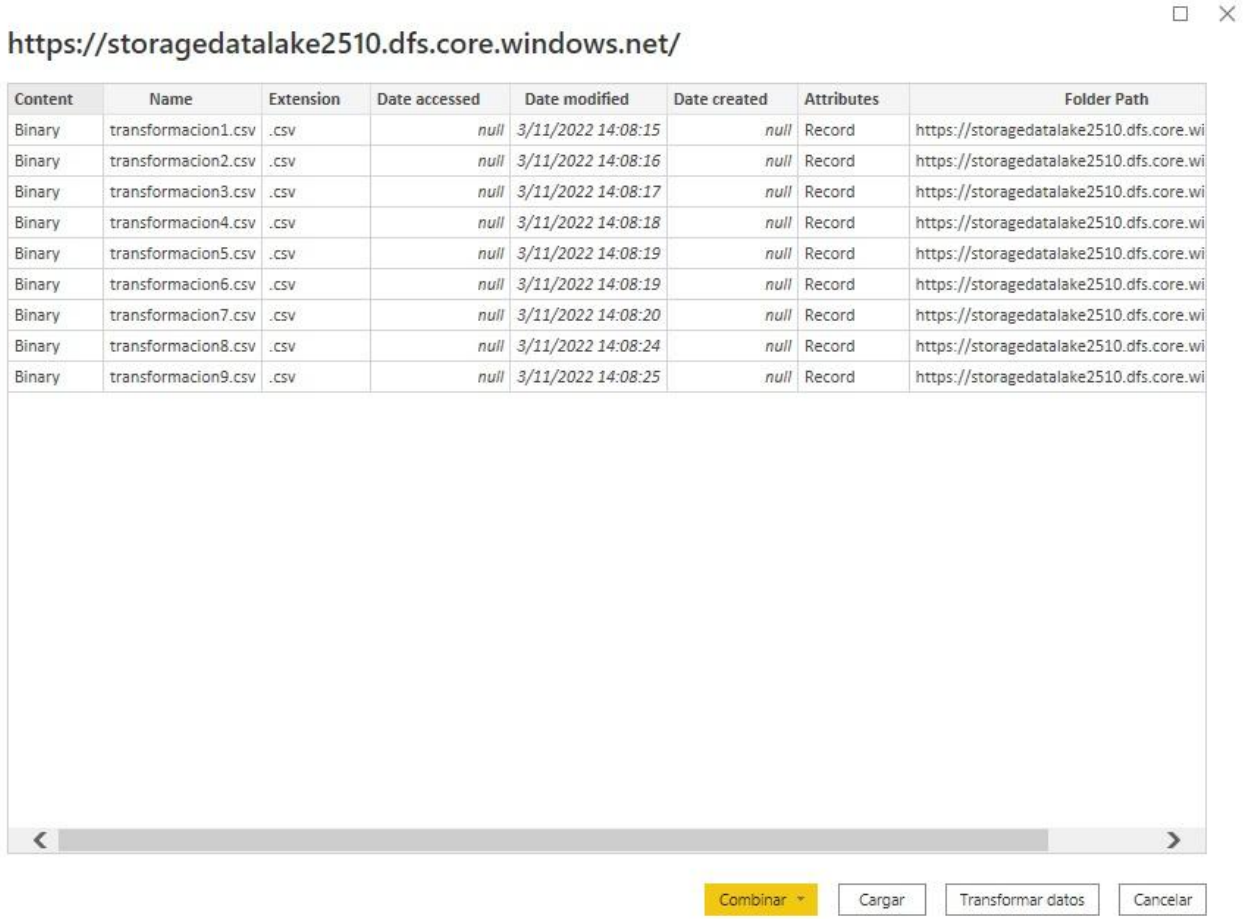
# Visualización de datos en PowerBi

- Conexión de blob storage y Data Lake Storage con PowerBi Desktop



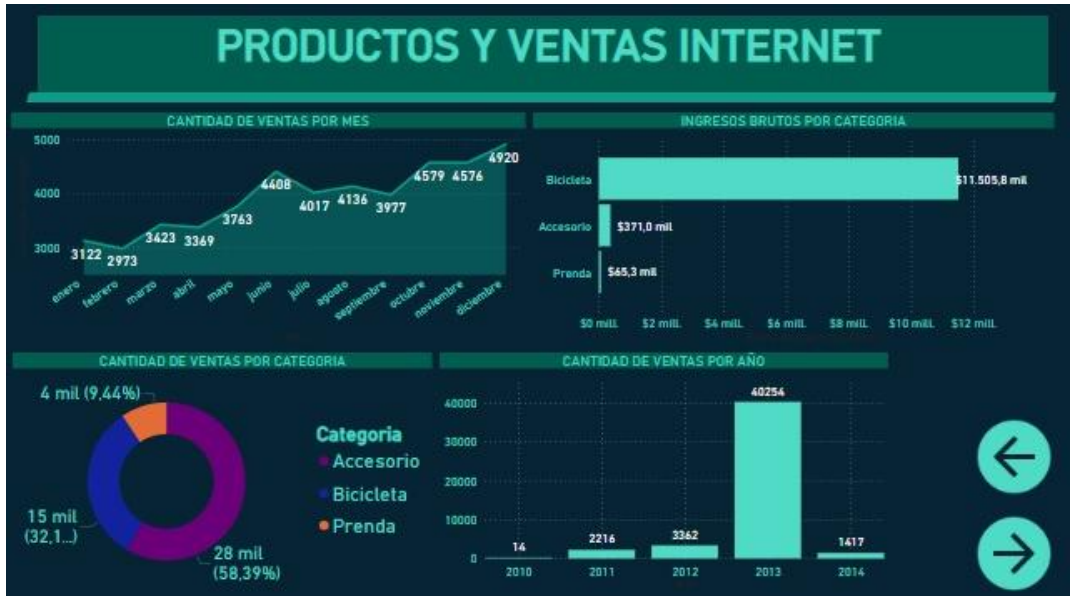




# Dashboard Interactivo






# MUCHAS GRACIAS