

Documentación Prueba Técnica DataKnow

Sebastian Mariño

1. Carga de Información

La realización de este ejercicio fue desarrollada mediante el lenguaje de programación Python. En un primer acercamiento al problema, se utilizó la función `open` para abrir el archivo `OFEI1204.txt` ubicado en la carpeta `../Datos3`. El modo `'r'` especifica que el archivo se abrirá en modo lectura. El contenido del archivo se leyó y almacenó en la variable `txt_data`. Posteriormente, se imprimió el contenido del archivo para identificar las particularidades de la Data, en donde se notó que cada tabla de información venía delimitado por el título agente y líneas vacías.

Luego, se procedió a la extracción de información relevante del texto. Se dividió el contenido del archivo en secciones utilizando el método `split` con el delimitador `"AGENTE:"`. Cada sección resultante contenía información sobre un agente específico. Se recorrieron estas secciones, extrayendo el nombre del agente y la tabla de datos asociada. La información de cada agente se agregó a la lista `agentes`, y las tablas de datos se procesaron y almacenaron en la lista `dfs`.

Después de completar la extracción de información, se utilizó la biblioteca `pandas` para manipular y analizar los datos de manera eficiente. Se concatenaron todas las tablas de datos en un único `DataFrame` llamado `df_final`. Se filtraron las filas del `DataFrame` donde la columna `"AGENTE"` era igual a `"EEPPM"` para verificar la correcta lectura del archivo ya que se comparó esta salida con el archivo `.txt` original. Luego, se filtraron las filas donde la columna con índice 1 era igual a `"D"` y se reorganizaron las columnas. Se eliminó la columna `'TIPO'` y se renombraron las columnas de acuerdo con los requisitos del problema.

Finalmente, se realizaron ajustes en el formato de algunas columnas, convirtiendo las columnas que representan horas a tipo de dato `float`. El `DataFrame` resultante se guardó en un archivo CSV llamado `'tabla_punto_1.csv'` en la carpeta `../resultados`.

En resumen, el código realiza la extracción, procesamiento y manipulación de datos de un archivo de texto, utilizando Python y `pandas`, para obtener un `DataFrame` final que satisface los requisitos específicos del problema planteado.

2. Manipulación de datos

La implementación de este código en Python se realizó con el objetivo de manipular y analizar datos contenidos en un archivo Excel (`'Datos Maestros VF.xlsx'`) y un archivo de texto (`'dDEC1204.TXT'`). Aquí se detalla paso a paso el procedimiento:

En primer lugar, se seleccionaron las columnas relevantes del archivo Excel utilizando la biblioteca `pandas`. Se leyó el archivo `'Datos Maestros VF.xlsx'` y se extrajeron las columnas especificadas en la lista `columnas_seleccionadas`. El `DataFrame` resultante se llamó `df_master_data`.

Luego, se filtraron las filas de `df_master_data` para obtener información específica sobre la empresa "EMGESA" y sobre centrales de tipo hidroeléctrico ("H") o térmico ("T"). El DataFrame resultante se denominó `df_master_data_emgesa`.

A continuación, se leyó el archivo de texto 'dDEC1204.TXT' con la función `pd.read_csv`, especificando el delimitador y la codificación adecuados. Se realizaron ajustes en el DataFrame resultante (`df_dec_1204`), eliminando la última fila y renombrando las columnas con nombres descriptivos.

Se llevó a cabo una fusión (`pd.merge`) entre los DataFrames `df_master_data_emgesa` y `df_dec_1204` utilizando la columna 'CENTRAL (dDEC, dSEGDES, dPRU...)' como clave de unión, y se especificó un tipo de fusión de tipo 'left'. El DataFrame resultante se llamó `df_merged_emgesa`.

A continuación, se seleccionaron las columnas que representan las horas ('Hora_1', 'Hora_2', ..., 'Hora_24') y se calculó la suma de estas columnas para obtener la columna 'Total_Horas'. Se filtraron las filas donde 'Total_Horas' era mayor que 0, y el DataFrame resultante se llamó `df_tabla_punto_2`.

Finalmente, se guardó el DataFrame `df_tabla_punto_2` en un archivo CSV llamado 'tabla_punto_2.csv' en la carpeta '../resultados'.

En resumen, este código realiza una serie de manipulaciones de datos utilizando pandas, fusionando información de un archivo Excel y un archivo de texto, y generando un DataFrame final que satisface los requisitos establecidos en el problema.

3. Prueba de SQL

Como primer paso se corrió la Query otorgada, la cual crea las tablas EMPLEADO, VACACIONES y se llena con la información especificada.

```
1 CREATE TABLE EMPLEADO (  
2   ID INT(8),  
3   NOMBRE VARCHAR(50),  
4   APELLIDO VARCHAR(50),  
5   SEXO CHAR(1),  
6   FECHA_NACIMIENTO DATE,  
7   SALARIO DOUBLE(10,2)  
8 );  
9 CREATE TABLE VACACIONES(  
10  ID INT(8),  
11  ID_EMP INT(8),  
12  FECHA_INICIO DATE,  
13  FECHA_FIN DATE,  
14  ESTADO CHAR(1),  
15  CANTIDAD_DIAS INT(8)  
16 );  
17 /*EN ESTA TABLA SE ALMACENA LA INFORMACIÓN BASICA DE LOS EMPLEADOS*/  
18 INSERT INTO EMPLEADO VALUES (1,"JUAN","PELAEZ","M",'1985-01-29',3500000);  
19 INSERT INTO EMPLEADO VALUES (2,"ANDRES","GARCIA","M",'1975-05-22',5500000);  
20 INSERT INTO EMPLEADO VALUES (3,"LAURA","PEREZ","F",'1991-09-10',2500000);  
21 INSERT INTO EMPLEADO VALUES (4,"PEPE","MARTINEZ","M",'1987-12-01',3800000);
```

Figura 1. Creación Tablas

A continuación, se muestran todas las Querys realizadas para obtener los datos solicitados y los respectivos resultados.

1. Seleccionar nombre, apellido y salario de todos los empleados:

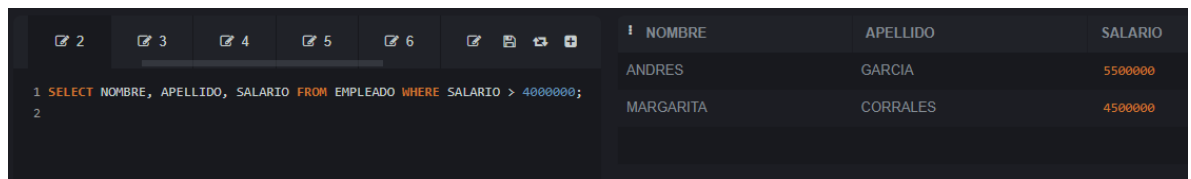


The screenshot shows a SQLite database window with a query editor on the left and a results table on the right. The query is: `1 SELECT NOMBRE, APELLIDO, SALARIO FROM EMPLEADO;`. The results table has three columns: NOMBRE, APELLIDO, and SALARIO. It contains five rows of employee data.

NOMBRE	APELLIDO	SALARIO
JUAN	PELAEZ	3500000
ANDRES	GARCIA	5500000
LAURA	PEREZ	2500000
PEPE	MARTINEZ	3800000
MARGARITA	CORRALES	4500000

Figura 2. Nombre apellido y salario

2. Seleccionar nombre, apellido y salario de todos los empleados que ganen más de 4 millones:

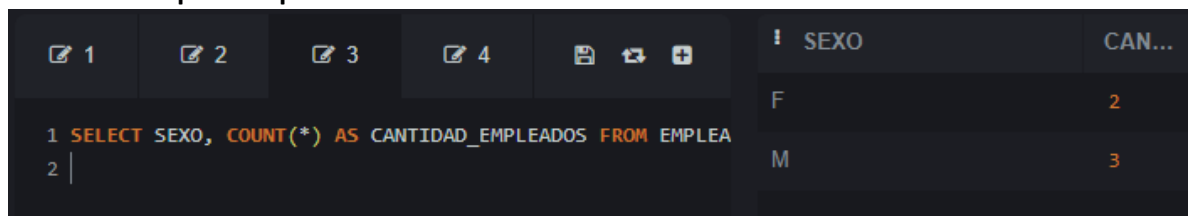


The screenshot shows a SQLite database window with a query editor on the left and a results table on the right. The query is: `1 SELECT NOMBRE, APELLIDO, SALARIO FROM EMPLEADO WHERE SALARIO > 4000000;`. The results table has three columns: NOMBRE, APELLIDO, and SALARIO. It contains two rows of employee data.

NOMBRE	APELLIDO	SALARIO
ANDRES	GARCIA	5500000
MARGARITA	CORRALES	4500000

Figura 3. Empleados que ganas más de 4M

3. Contar los empleados por sexo:

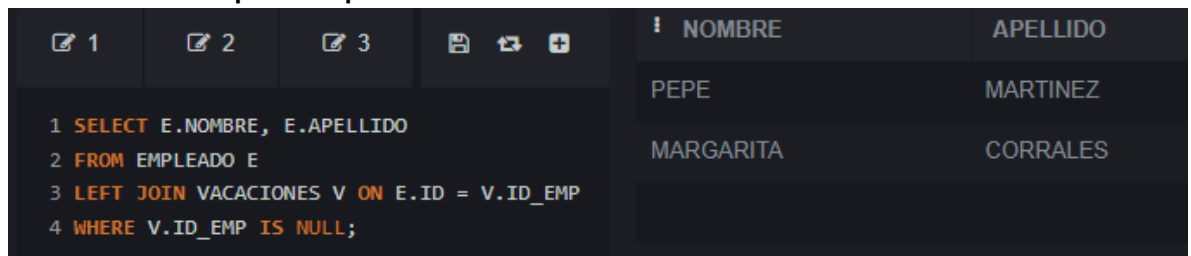


The screenshot shows a SQLite database window with a query editor on the left and a results table on the right. The query is: `1 SELECT SEXO, COUNT(*) AS CANTIDAD_EMPLEADOS FROM EMPLEA`. The results table has two columns: SEXO and CAN... (CANTIDAD_EMPLEADOS). It contains two rows of data.

SEXO	CAN...
F	2
M	3

Figura 4. Conteo por sexo

4. Seleccionar los empleados que no han hecho solicitud de vacaciones:

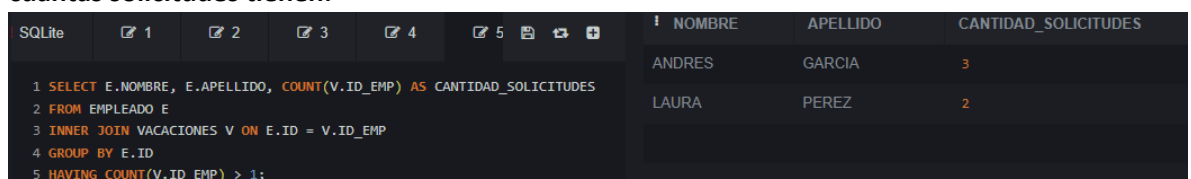


The screenshot shows a SQLite database window with a query editor on the left and a results table on the right. The query is: `1 SELECT E.NOMBRE, E.APELLIDO`
`2 FROM EMPLEADO E`
`3 LEFT JOIN VACACIONES V ON E.ID = V.ID_EMP`
`4 WHERE V.ID_EMP IS NULL;`. The results table has two columns: NOMBRE and APELLIDO. It contains three rows of employee data.

NOMBRE	APELLIDO
PEPE	MARTINEZ
MARGARITA	CORRALES

Figura 5. Empleados que no han solicitado vacaciones

5. Seleccionar los empleados que tengan más de una solicitud de vacaciones y mostrar cuántas solicitudes tienen:

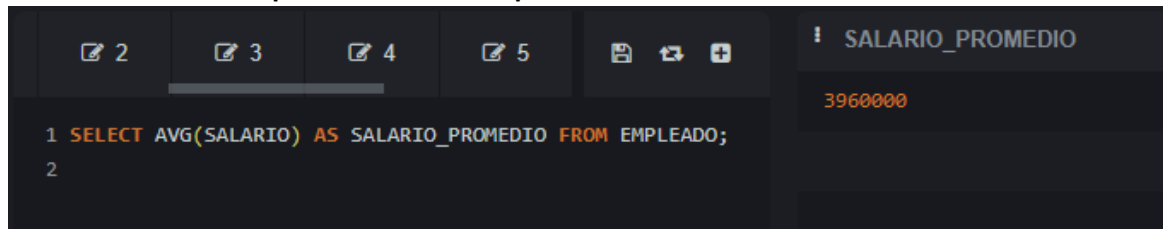


The screenshot shows a SQLite database window with a query editor on the left and a results table on the right. The query is: `1 SELECT E.NOMBRE, E.APELLIDO, COUNT(V.ID_EMP) AS CANTIDAD_SOLICITUDES`
`2 FROM EMPLEADO E`
`3 INNER JOIN VACACIONES V ON E.ID = V.ID_EMP`
`4 GROUP BY E.ID`
`5 HAVING COUNT(V.ID_EMP) > 1;`. The results table has three columns: NOMBRE, APELLIDO, and CANTIDAD_SOLICITUDES. It contains two rows of employee data.

NOMBRE	APELLIDO	CANTIDAD_SOLICITUDES
ANDRES	GARCIA	3
LAURA	PEREZ	2

Figura 6. Empleados que han solicitado vacaciones más de una vez

6. Determinar el salario promedio de los empleados:



The screenshot shows a SQL query editor with a toolbar at the top containing icons for undo, redo, and other editing functions. The query is as follows:

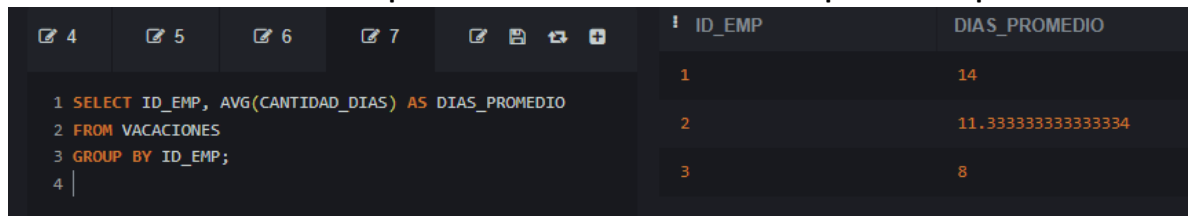
```
1 SELECT AVG(SALARIO) AS SALARIO_PROMEDIO FROM EMPLEADO;
2
```

To the right of the editor, a preview window displays the result of the query:

SALARIO_PROMEDIO
3960000

Figura 7. Salario promedio

7. Determinar la cantidad de días promedio solicitados de vacaciones por cada empleado:



The screenshot shows a SQL query editor with a toolbar. The query is as follows:

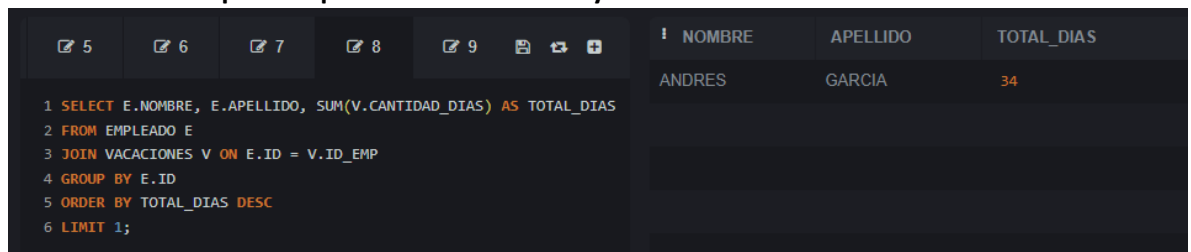
```
1 SELECT ID_EMP, AVG(CANTIDAD_DIAS) AS DIAS_PROMEDIO
2 FROM VACACIONES
3 GROUP BY ID_EMP;
4
```

To the right, a preview window displays the results:

ID_EMP	DIAS_PROMEDIO
1	14
2	11.333333333333334
3	8

Figura 8. Dias por promedio solicitados por empleado

8. Seleccionar el empleado que ha solicitado la mayor cantidad de días de vacaciones:



The screenshot shows a SQL query editor with a toolbar. The query is as follows:

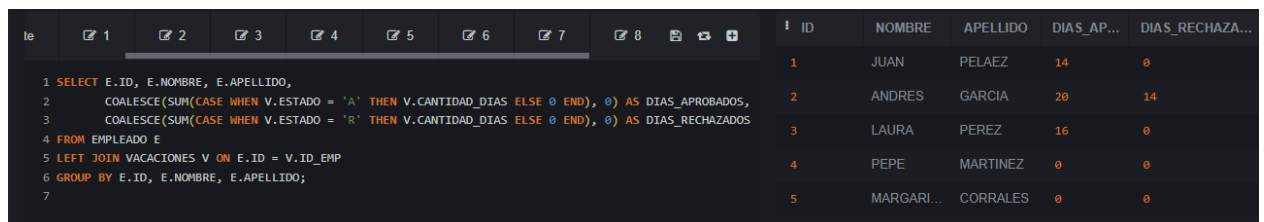
```
1 SELECT E.NOMBRE, E.APELLIDO, SUM(V.CANTIDAD_DIAS) AS TOTAL_DIAS
2 FROM EMPLEADO E
3 JOIN VACACIONES V ON E.ID = V.ID_EMP
4 GROUP BY E.ID
5 ORDER BY TOTAL_DIAS DESC
6 LIMIT 1;
```

To the right, a preview window displays the result:

NOMBRE	APELLIDO	TOTAL_DIAS
ANDRES	GARCIA	34

Figura 9. Empleado que más ha solicitado vacaciones en total

9. Consultar la cantidad de días aprobados y rechazados por cada empleado, mostrar 0 en caso de no tener solicitudes:



The screenshot shows a SQL query editor with a toolbar. The query is as follows:

```
1 SELECT E.ID, E.NOMBRE, E.APELLIDO,
2 COALESCE(SUM(CASE WHEN V.ESTADO = 'A' THEN V.CANTIDAD_DIAS ELSE 0 END), 0) AS DIAS_APROBADOS,
3 COALESCE(SUM(CASE WHEN V.ESTADO = 'R' THEN V.CANTIDAD_DIAS ELSE 0 END), 0) AS DIAS_RECHAZADOS
4 FROM EMPLEADO E
5 LEFT JOIN VACACIONES V ON E.ID = V.ID_EMP
6 GROUP BY E.ID, E.NOMBRE, E.APELLIDO;
7
```

To the right, a preview window displays the results:

ID	NOMBRE	APELLIDO	DIAS_AP...	DIAS_RECHAZA...
1	JUAN	PELAEZ	14	0
2	ANDRES	GARCIA	20	14
3	LAURA	PEREZ	16	0
4	PEPE	MARTINEZ	0	0
5	MARGARI...	CORRALES	0	0

Figura 10. Tabla de días aprobados y rechazados de vacaciones

4. Prueba de AWS

1. Configuración roles en AWS

En la Figura 11 se observan los roles creados para las dos instancias requeridas para el desarrollo de este ejercicio, estas son s3 y redshift

Roles (4) Información

Un rol de IAM es una identidad que podemos crear y que tiene permisos específicos con credenciales que son válidas para periodos cortos. Las entidades en las que confía pueden asumir roles.

Buscar

<input type="checkbox"/>	Nombre del rol	Entidades de confianza	Última actividad
<input type="checkbox"/>	AWSServiceRoleForSupport	Servicio de AWS: support (Rol vinculi	-
<input type="checkbox"/>	AWSServiceRoleForTrustedAdvisor	Servicio de AWS: trustedadvisor (Rol	-
<input type="checkbox"/>	redshift-full-access	Servicio de AWS: redshift	-
<input type="checkbox"/>	s3-roles	Servicio de AWS: s3	-

Figura 11. Roles

2. Configure un Clúster de pruebas de Redshift con los requerimientos mínimos necesarios

Se crea el Clúster de prueba para ingresar los datos necesarios.

Clústeres (2) Información

Filtrar clústeres por propiedad o valor

Crear clúster

<input type="checkbox"/>	Clúster	Estado	Espacio de nombre...	Zona de disponibilidad	Multi-AZ	Capacidad de alma...	Utilización de la CPU	Instant...	Notificaci...	Etiqu...
<input type="checkbox"/>	redshift-cluster ra3.16xlarge 2 nodos 256 ...	Modifying	49473606-914c-458a-...	us-east-2c	No	< 1 %	1 %	1 instantánea		
<input type="checkbox"/>	redshift-cluster-2 ra3.16xlarge 2 nodos 256 ...	Available	81138b00-25fe-4014-...	us-east-2c	No	< 1 %	1 %	1 instantánea		

Figura 12. Clúster

3. Copie la información a Redshift usando algún editor de queries como SQL Workbench/J - Home (sql-workbench.eu)

Mediante SQL workbench se conecta al clúster creado como se muestra en la siguiente Figura.

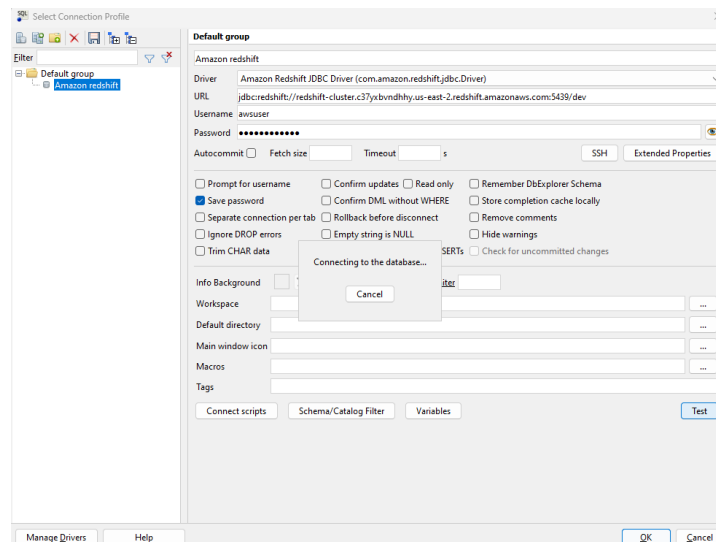


Figura 13. Workbench

Posteriormente se cargaron las tablas, a continuación, se muestra un ejemplo de Query SQL para cargar una tala, esto se repitió para los 7 archivos de la base de datos.

```
COPY events FROM 'C:\Users\{tu ruta}\ticketdb \allevnts_pipe.txt'
DELIMITER ',' -- Cambia a tu delimitador
IGNOREHEADER 1 -- Si hay un encabezado en el archivo
CSV; -- 0 el formato que estés utilizando
```

Ya que ticketdb es una base de base de datos de prueba para redshift, otra forma de cargarla es cargando directamente desde la creación del Clúster del siguiente modo.

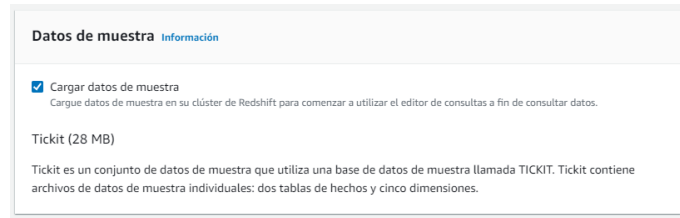


Figura 14. Carga de Tickit directamente desde la creación del Clúster

A continuación, se muestra la base de datos relacional tickit.

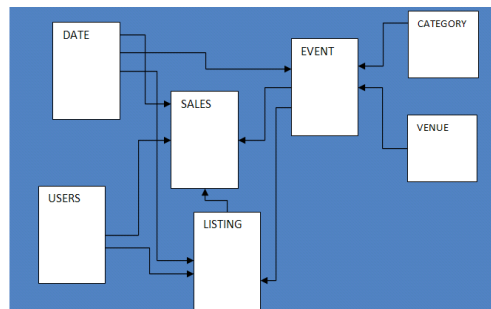


Figura 15. TicketDb

4. Responda las siguientes preguntas usando comandos de consultas SQL:

Median el redshift query editor se realizarán las consultas, a continuación, se muestran las tablas cargadas:

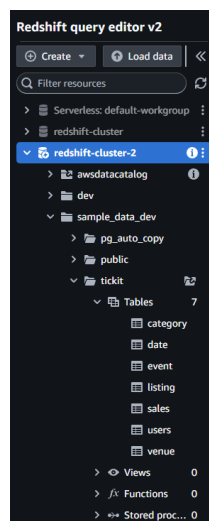


Figura 16. Redshift query editor v2

A continuación, se muestran las Querys para resolver las preguntas planteadas:

¿Cuántos Usuarios gustan del Jazz?

► Run ☒ Limit 100

```
1 SET search_path TO tickit;
2
3 SELECT COUNT(*)
4 FROM users
5 WHERE likejazz = true;
```

Result 1 Result 2 (1)

count	
12441	

¿Cuántos Usuarios gustan de la ópera y del rock al mismo tiempo?

► Run ☒ Limit 100

```
1 SELECT COUNT(*)
2 FROM users
3 WHERE likeopera = true AND likerock = true;
```

Result 1 (1)

count	
3113	

¿Cuál es el promedio, moda y mediana del total de Ventas?

► Run ☒ Limit 100

```
1 -- Promedio del total de ventas
2 SELECT AVG(s.pricepaid) AS promedio_ventas
3 FROM sales s;
4
5 -- Mediana del total de ventas
6 SELECT
7     pricepaid AS mediana_ventas
8 FROM (
9     SELECT
10        pricepaid,
11        ROW_NUMBER() OVER (ORDER BY pricepaid) AS row_num,
12        COUNT(*) OVER () AS total_rows
13     FROM sales
14     ) subquery
15 WHERE row_num = (total_rows + 1) / 2
16        OR row_num = total_rows / 2 + 1;
17
18 -- Moda del total de ventas
19 SELECT
20     pricepaid AS moda_ventas
21 FROM (
22     SELECT
23        pricepaid,
24        COUNT(*) AS frequency
25     FROM sales
26     GROUP BY pricepaid
27     ORDER BY frequency DESC
28     LIMIT 1
29     ) subquery;
```

promedio_ventas	mediana_ventas	moda_ventas
642.28	386	144
	386	

► Run ☒ Limit 100

```
1 SELECT AVG(s.pricepaid) AS promedio_ventas_rock
2 FROM sales s
3 JOIN users u ON s.buyerid = u.userid
4 WHERE u.likerock = true AND u.likejazz = false;
```

Result 1 (1)

promedio_ventas_rock	
648.86	

Figura 17. Querys para resolver las preguntas

4. En una nueva tabla junte la información (Nombre de usuario, Apellido de usuario, Correo de usuario, Nombre del evento, lugar del evento, Fecha del evento, Cantidad y Total vendidos) y expórtela usando Redshift a un bucket predefinido de S3.:

Se crea una instancia de S3 y se crea el bucket para almacenar una nueva tabla.

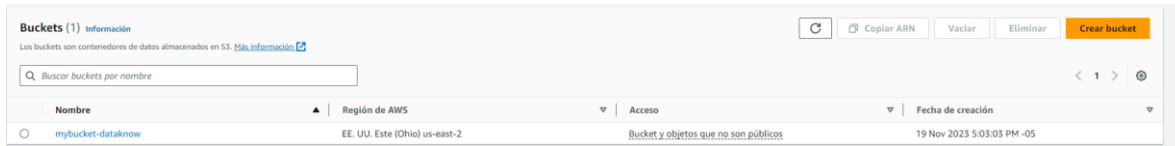


Figura 18. creación bucket

Posteriormente con la siguiente Query se crea la “nueva_tabla”

```
1  -- Crear una nueva tabla con la información requerida
2  CREATE TABLE nueva_tabla AS
3  SELECT
4      u.username,
5      u.lastname,
6      u.email,
7      e.eventname,
8      v.venueid,
9      d.caldate AS fecha_evento,
10     s.qtysold AS cantidad_vendida,
11     s.pricepaid AS total_vendido
12 FROM
13     sales s
14 JOIN
15     users u ON s.buyerid = u.userid
16 JOIN
17     event e ON s.eventid = e.eventid
18 JOIN
19     venue v ON e.venueid = v.venueid
20 JOIN
21     date d ON e.dateid = d.dateid;
```

Figura 19. Crear nueva tabla

Mediante la siguiente Query se carga la nueva en la ubicación deseada dentro del Bucket

```
► Run  Limit 100
1  UNLOAD ('SELECT * FROM nueva_tabla')
2  to 's3://mybucket-dataknow/tablas/tabla.csv'
3  CREDENTIALS 'aws_access_key_id=AKIAZGOMSSKYA6NMXP42;aws_secret_access_key=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'
4  CSV
5  PARALLEL OFF;
```

Figura 20. Envío de tabla a S3

Se revisa dentro del bucket el archivo cargado de la siguiente manera:

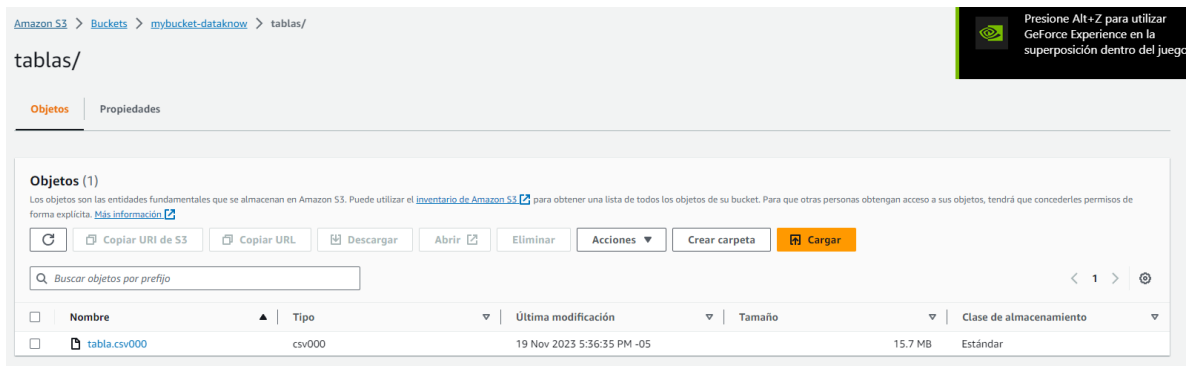


Figura 21. Carga del archivo csv

```

CXQ97IWP, Landry, euismod@turpis.org, Thurgood, Majestic Theatre, 2008-09-06, 1, 394.00
OLM65HXC, Schultz, dictum.mi.ac@fringillaporttitorvulputate.edu, Dirty Dancing, Nederlander Theatre, 2008-08-28, 1, 89.00
ICV065TI, Richmond, scelerisque@interdum.ca, Black Kids, Verizon Center, 2008-08-01, 1, 2264.00
IGN99NKK, Juarez, Vivamus.rhonus@dolornonummyac.com, Killers, Edward Jones Dome, 2008-08-12, 1, 26.00
QDM10KBI, Duran, diam.luctus@nismagna.com, Iggy and the Stooges, Progressive Field, 2008-08-03, 2, 600.00
XVZ87DTP, Blevins, amet.massa.Quisque@nec.ca, A Tale of Two Cities, Hilton Theatre, 2008-08-22, 1, 1775.00
KVZ00JJA, Palmer, Fusce.feugiat.Lorem@nec.edu, Vicente Fernandez, Villa Montalvo, 2008-08-27, 1, 145.00
BRD06LXJ, Talley, molestie.tortor.nibh@scelerisque.edu, Nick Cave, Oracle Arena, 2008-08-16, 1, 143.00
LKS46PXD, Lawrence, nunc.ullamcorper.eu@Inmipede.ca, Matisyahu, Progressive Field, 2008-08-23, 2, 266.00
QJW54PEU, Davenport, consequat.enim.diam@ac.ca, Rock the Bells, Dolphin Stadium, 2008-09-15, 1, 46.00
CLQ46JPN, Hendricks, facilisis.non@quisturpisvitae.ca, Ana Gabriel, Busch Stadium, 2008-09-14, 1, 439.00
COY13BOI, Pratt, feugiat@nonummy.ca, Calibash, Georgia Dome, 2008-09-08, 1, 344.00
YIE05RWM, Spears, vitae@commodo.com, L Elisir d Amore, War Memorial Opera House, 2008-09-23, 1, 2137.00
RFZ19QRS, Hodges, enim@elitelitfermentum.edu, The Magic Flute, San Francisco Opera, 2008-09-07, 2, 118.00
WYJ46COX, Adkins, erat@dictumeleifend.org, A Bronx Tale, Helen Hayes Theatre, 2008-08-29, 1, 887.00
SME22JEA, Wagner, nulla.ante@Donec.ca, Little Big Town, Louisiana Superdome, 2008-08-25, 1, 199.00
BKM33NTB, Oneil, felis.orci@urna.com, Die Walkure, Ellie Caulkins Opera House, 2008-09-30, 1, 281.00
OLM65HXC, Schultz, dictum.mi.ac@fringillaporttitorvulputate.edu, Steven Curtis Chapman, Ralph Wilson Stadium, 2008-09-29, 1, 494.00
DQX19ALZ, Roman, Praesent.luctus@magnaLorem.com, Bette Midler, ARCO Arena, 2008-10-05, 2, 168.00
SOY02BIK, Cooley, convallis@egestas.com, Foo Fighters, INVESCO Field, 2008-09-18, 1, 249.00
EFX80DRX, Lara, nunc@nislNullaeu.com, Rigoletto, Lyric Opera House, 2008-09-10, 1, 148.00
LNQ590BS, Murphy, aliquam.enim.nec@nonarcu.org, Robert Cray Band, Air Canada Centre, 2008-09-23, 1, 374.00
JFF76FIR, Figueroa, magna.Cras@quisaccumsanconvallis.com, Il Trovatore, Lyric Opera House, 2008-09-03, 2, 250.00
NNN47GED, Elliott, adipiscing.elit.Aliquam@diam.edu, John Mellencamp, Wrigley Field, 2008-09-05, 2, 434.00
OGQ29CAF, Kline, taciti.sociosqu@Nullamsuscipit.ca, Grease, Carnegie Hall, 2008-09-25, 2, 426.00
OMX12VHV, Cameron, leo.Cras@orci.com, Temptations, Jobing.com Arena, 2008-10-08, 1, 29.00
OMX27CBS, Wooten, Curabitur@Aliquam.edu, Justin Timberlake, Robertson Stadium, 2008-09-22, 1, 122.00
OW003EWH, Trevino, Nullam@eueleifendnec.com, Zappa Plays Zappa, TD Banknorth Garden, 2008-09-07, 4, 728.00
KMZ00QPU, Holmes, sed@semperet.org, Avenue Q, George Gershwin Theatre, 2008-09-10, 2, 156.00
PKJ80CDG, Fisher, tincidunt@idrisusquis.com, Jersey Boys, Nederlander Theatre, 2008-09-28, 1, 496.00
AGL87LNF, Gill, fames.ac.turpis@posuerecubiliaCurae;.edu, Hairspray, Brooks Atkinson Theatre, 2008-10-20, 2, 374.00
NXC79VTO, Delaney, cursus.purus@mollisnecscursus.edu, South Pacific, Ethel Barrymore Theatre, 2008-10-01, 2, 590.00
WSM73HPK, Skinner, tempor@Nulla.com, Wilco, Progressive Field, 2008-09-19, 2, 730.00
YHH04SQB, Dillard, pede.blandit@nullaDonecnon.org, Stray Cats, BMO Field, 2008-10-03, 2, 488.00
GTW71MNA, Reitt, lobortis.mauris@danibusgravidu.edu, Blake Shelton, Pizza Hut Park, 2008-10-17, 2, 48.00

```

Figura 22. CSV generado

6. (Opcional) Sobre la data exportada en el punto 5, y usando cualquier información adicional que desee, cree una sesión de SageMaker y realice un modelo de Forecast con cualquier técnica de su preferencia, para pronosticar las ventas para los siguientes 7 días desde el final del histórico de datos. Tenga en cuenta que la fecha de la venta se encuentra en la variable:

Se crea una instancia de Sagemaker como se muestra en la siguiente figura:

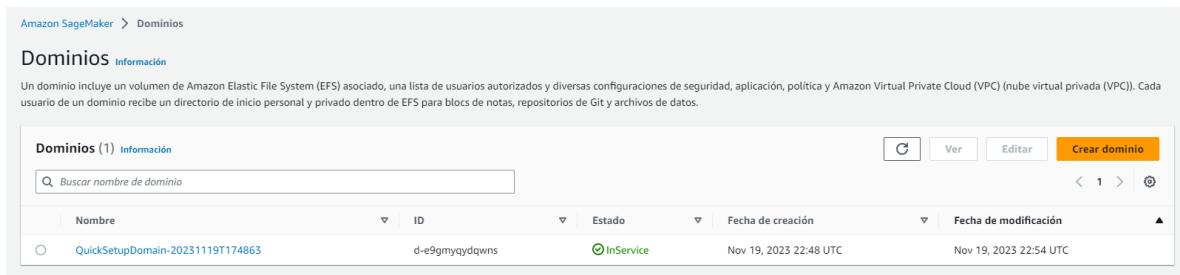


Figura 23. Sagemaker

Se implementa el código en un jupyter dentro del entorno de sagemaker.

```
[8]: !pip install cython
!pip install statsmodels
...

[9]: import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt

[15]: df = pd.read_csv('tabla_punto_4.csv', header=None)
df

[15]:
```

	0	1	2	3	4	5	6	7
0	CKQ97WNP	Landry	eulsmo@turpis.org	Thurgood	Majestic Theatre	2008-09-06	1	394.0
1	OLM6SHXC	Schultz	dictum.mt.ac@fringillaportitorvoluptate.edu	Dirty Dancing	Nederlander Theatre	2008-08-28	1	89.0
2	ICV06STI	Richmond	sclerisque@interdum.ca	Black Kids	Verizon Center	2008-08-01	1	2264.0
3	IGN99NKK	Juarez	Vivamus.rhonus@dolomonomumyac.com	Killers	Edward Jones Dome	2008-08-12	1	26.0
4	QDM10KBI	Duran	diam.luctus@nismagna.com	Iggy and the Stooges	Progressive Field	2008-08-03	2	600.0
...
169674	VJA56TNP	Dodson	lorem.eget.mollis@semegestasblandit.com	Memphis - The Birth of Rock n Roll	John Golden Theatre	2008-09-30	2	328.0
169675	UEQ6SXRK	Ball	enim.diam.vel@incursuset.org	KT Tunstall	Yankee Stadium	2008-10-30	2	76.0
169676	NV047ASC	Franco	Nam.ligula@temporarcuVestibulum.edu	Wallflowers	Safeco Field	2008-10-13	4	1576.0
169677	HZZ05IYP	Spears	lobortis.mauris@lpsum.ca	Jesse Lacey	Jobing.com Arena	2008-09-28	2	4098.0
169678	QJG52XOL	Benjamin	arcu.Aliquam.ultrices@ategestas.ca	Dandy Warhols	PETCO Park	2008-10-05	2	484.0

```
169679 rows x 8 columns

[28]: df.columns = ["ID", "Name", "Email", "Event", "Venue", "SaleTime", "Quantity", "Amount"]
df["SaleTime"] = pd.to_datetime(df["SaleTime"])
df

[28]:
```

	ID	Name	Email	Event	Venue	SaleTime	Quantity	Amount
0	CKQ97WNP	Landry	eulsmo@turpis.org	Thurgood	Majestic Theatre	2008-09-06	1	394.0
1	OLM6SHXC	Schultz	dictum.mt.ac@fringillaportitorvoluptate.edu	Dirty Dancing	Nederlander Theatre	2008-08-28	1	89.0
2	ICV06STI	Richmond	sclerisque@interdum.ca	Black Kids	Verizon Center	2008-08-01	1	2264.0
3	IGN99NKK	Juarez	Vivamus.rhonus@dolomonomumyac.com	Killers	Edward Jones Dome	2008-08-12	1	26.0
4	QDM10KBI	Duran	diam.luctus@nismagna.com	Iggy and the Stooges	Progressive Field	2008-08-03	2	600.0

Figura 24. Sagemaker

El código implementado se puede encontrar dentro de la carpeta code. A continuación, se muestra la predicción.

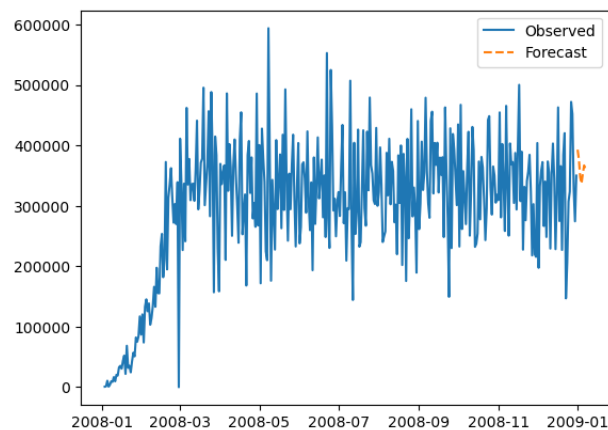


Figura 25. Forecast

5. Prueba de Azure

1.Desplegar base de datos en SQL, con la base de pruebas Adventure Works

Se crea la base de datos SQL en azure como se muestra en la siguiente figura, posteriormente ya se tiene una base de datos con la dataset de pruebas.

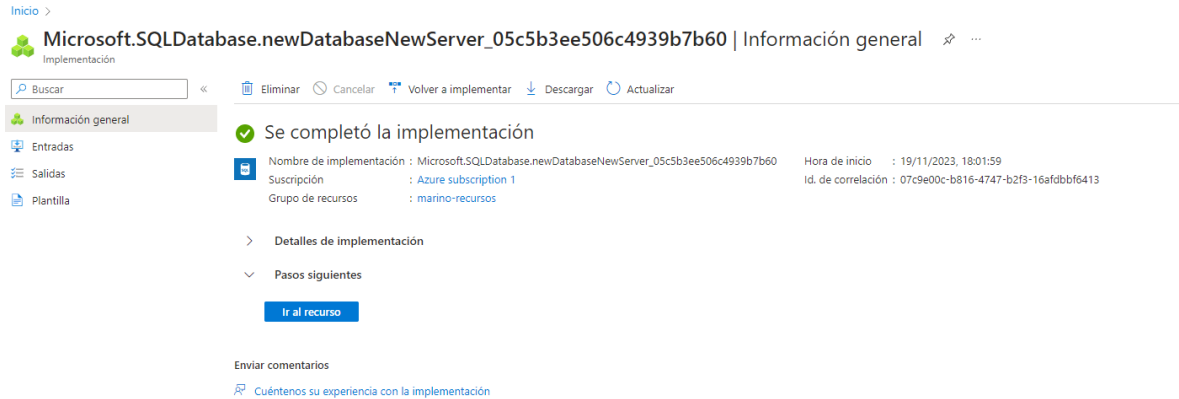


Figura 26. Creación base de datos

2. Realizar un pipeline con Azure Datafactory, utilizando data flow, para realizar la carga de una base de datos, crear 5 indicadores.

En la siguiente figura se muestra la creación del Dataflow desde Azure:

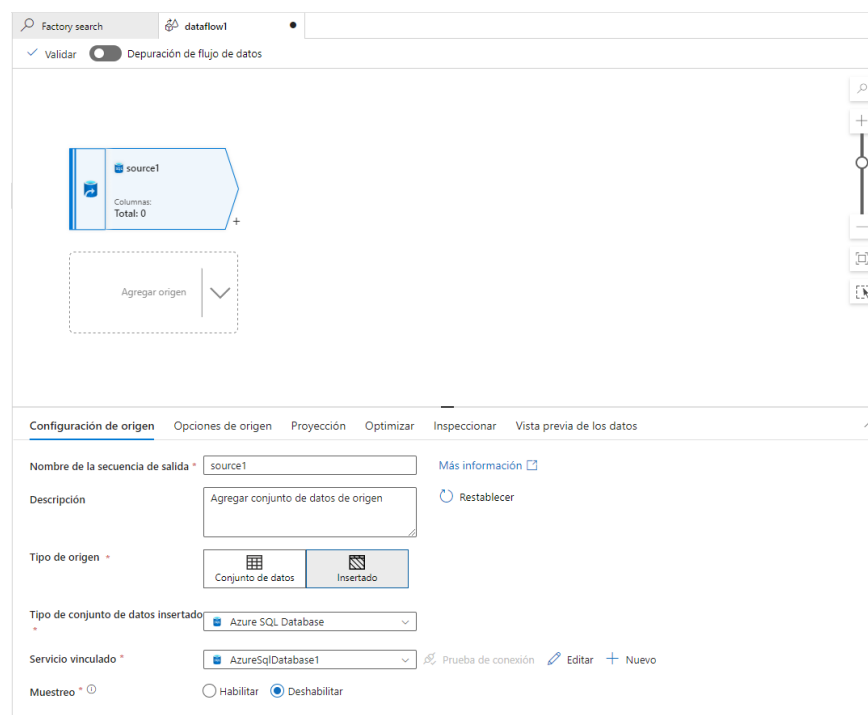


Figura 27. Dataflow

Los siguientes ítems no se lograron desarrollar debido a una falla en la cuenta personal de Azure, pero los pasos siguientes si se hubiera desarrollado el punto serian:

- se selecciona "Nuevo" para crear un nuevo Data Flow. En el Data Flow, se utilizan las actividades "Source" para extraer datos de la base de datos SQL y las actividades "Sink" para cargar datos en el Data Lake. Se configuran las transformaciones necesarias entre las actividades Source y Sink utilizando las funcionalidades de transformación de Data Flow.
- se agregan actividades "Copy Data" entre las transformaciones según sea necesario. Se configuran las actividades "Copy Data" para mover datos entre la base de datos SQL y el Data Lake.
- se guarda y publica el Data Flow y el pipeline después de configurar todas las actividades y transformaciones.
- se puede ejecutar el pipeline manualmente para probar las transformaciones y asegurarse de que todo funcione correctamente.

6. Prueba de modelación analítica

En el transcurso del análisis y modelado de datos, se procedió a cargar el conjunto de datos de entrenamiento desde el archivo 'train.csv'. Se identificaron varias columnas que no solo eran consideradas irrelevantes para la predicción de la variable objetivo FRAUDE, sino que también presentaban un alto número de valores nulos. La decisión de eliminar estas columnas se basó en la premisa de que su inclusión no aportaría significativamente al modelo y podría introducir ruido innecesario en el análisis. Después de remover filas con valores nulos y eliminar estas columnas, el conjunto de datos de entrenamiento no experimentó una reducción significativa en su tamaño, si no se hubiesen eliminado las columnas irrelevantes el conjunto de datos sufriría una drástica disminución de casi la mitad de los datos. Este proceso de limpieza y simplificación de características no solo contribuyó a mejorar la eficiencia computacional, sino que también permitió centrarse en las variables más relevantes para el objetivo de predecir transacciones fraudulentas.

Posteriormente, se dividió el conjunto de datos en conjuntos de entrenamiento y prueba, y se ajustaron modelos de Random Forest y SVM. La optimización de hiperparámetros se realizó mediante las técnicas de búsqueda GridSearchCV y RandomizedSearchCV. Se evaluó el rendimiento de los modelos, destacándose el Random Forest por su alta precisión, recall y f1-score. En el siguiente paso, se aplicó el modelo seleccionado a un conjunto de datos de prueba previamente cargado. Se garantizó la consistencia en el orden de las columnas y se aplicó la codificación one-hot, ajustando la columna 'COD_PAIS' según el mapeo ISO previamente creado. Se generaron probabilidades en lugar de predicciones binarias para ofrecer flexibilidad en la interpretación de los resultados. Los resultados se guardaron en un archivo CSV, incluyendo las probabilidades de fraude. Se optó por el modelo de Random Forest con hiperparámetros optimizados mediante GridSearchCV debido a su robusto rendimiento, especialmente en la minimización de falsos positivos. Este enfoque se seleccionó con la consideración de la crítica asociada a acusar falsamente de fraude, priorizando así la precisión en la detección de transacciones legítimas. Los resultados detallados del rendimiento de los modelos se presentan en el informe, y el archivo CSV resultante se generó para su análisis y revisión adicionales.

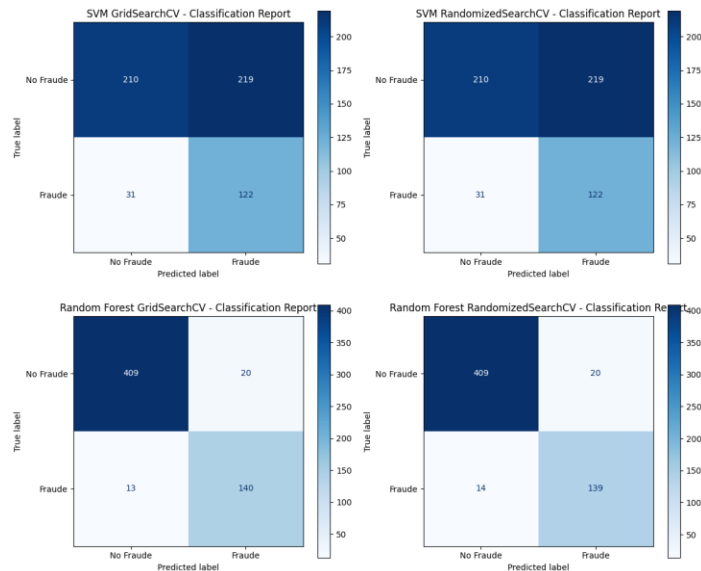


Figura 28. Matriz de confusión con modelos entrenados

7. Arquitectura

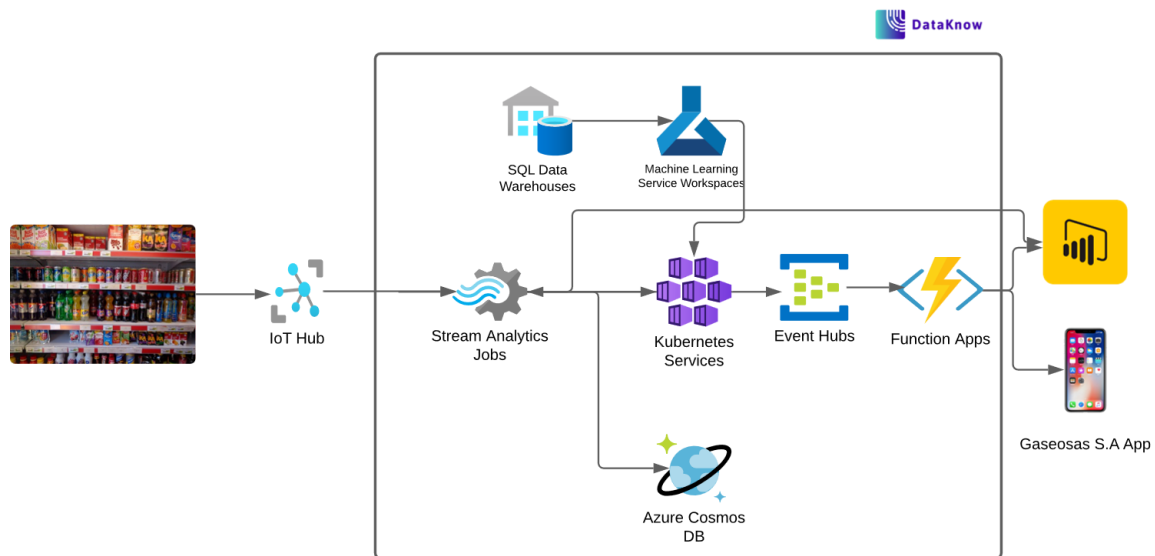


Figura 29. Arquitectura Solución.

Para el diseño de la arquitectura de la solución analítica en Azure que cumpla con los requerimientos de Gaseosas SA, se tienen la siguiente explicación.

1. Fuentes de Datos:

- **Datos de Ventas:** Almacenados en Azure SQL Data Warehouse para la base de datos histórica debido al gran volumen de datos y la capacidad de escalamiento.

- **Datos de Dispositivos de Puntos de Venta:** Transmitidos en tiempo real a través de Azure Stream Analytics, que procesa y enruta los datos a diferentes destinos según sea necesario.

2. **Procesamiento de Datos:**

- Utilizar Azure Databricks para procesar grandes volúmenes de datos de ventas y realizar análisis avanzados para el entrenamiento de modelos predictivos.

3. **Almacenamiento:**

- **Almacenamiento Histórico:** Azure SQL Data Warehouse para almacenar la base de datos histórica de ventas debido a su capacidad de escalamiento horizontal y capacidades de procesamiento analítico.
- **Almacenamiento en Tiempo Real:** Azure Cosmos DB para almacenar datos en tiempo real, ya que ofrece baja latencia y es altamente escalable.

4. **Modelado y Entrenamiento:**

- **Entorno de Modelado:** Azure Machine Learning para desarrollar, entrenar y desplegar modelos predictivos.
- **Modelos Analíticos:** Los modelos entrenados se implementan en Azure Kubernetes Service para permitir la escalabilidad y el despliegue eficiente.

5. **Gobierno de Datos y Modelos:**

- **Gestión de Metadatos:** Azure Purview para gestionar y descubrir metadatos en todo el entorno analítico.
- **Gobierno de Modelos:** Azure Machine Learning registra y rastrea los modelos, y Azure Policy define políticas para la implementación y monitorización.
- **Seguridad de Datos:** Azure Active Directory para la gestión de identidades y roles, y Azure Key Vault para la gestión segura de claves y secretos.

6. **Análisis en Tiempo Real:**

- Azure Stream Analytics para el análisis en tiempo real de datos de dispositivos de puntos de venta y la comparación con patrones históricos.

7. **Alertas y Acciones Preventivas:**

- Azure Logic Apps para la automatización de procesos y la generación de alertas en caso de posibles desabastecimientos.

8. **Visualización de Datos:**

- Utilizar Power BI para crear paneles de control interactivos que permitan a los usuarios explorar los datos y patrones identificados por el sistema analítico.

En cuanto al gobierno de datos tenemos lo siguiente:

1. Gestión de Metadatos:

- Azure Purview para catalogar, descubrir y gestionar metadatos en todos los servicios de datos.

2. Gobierno de Modelos:

- Azure Machine Learning para rastrear, versionar y gestionar el ciclo de vida de los modelos predictivos.

3. Seguridad de Datos:

- Azure Active Directory para gestionar la autenticación y autorización, y Azure Key Vault para la gestión segura de claves y secretos.

4. Perfiles de Usuario:

- Definir roles y permisos en Azure Active Directory para garantizar el acceso adecuado según los perfiles de usuario.

5. Formación y Concienciación:

- Desarrollar programas de formación para los usuarios sobre el uso ético de datos y modelos, con recursos disponibles en Azure Learning Paths.

Esta arquitectura aprovecha servicios específicos de Azure para garantizar la eficiencia, la escalabilidad y la integración fluida entre los diferentes componentes. La gestión de datos y modelos se logra a través de herramientas y servicios dedicados en Azure, proporcionando un gobierno robusto y centralizado.

8. Prueba BI

Para el desarrollo de este punto se usó PowerBI, por lo tanto, el primer paso a seguir fue subir las tablas a dicho Software y conectarlas generando una base de datos relacional, esto del siguiente modo:

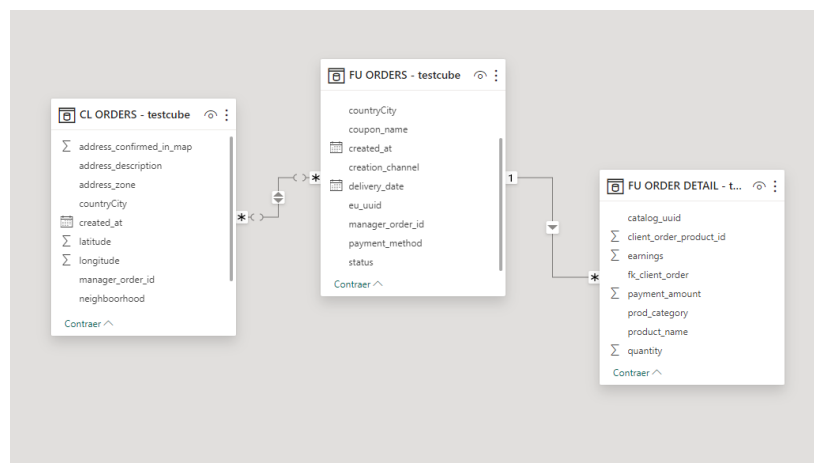


Figura 30. Unión Tablas

Posteriormente se calcula el número total de usuarios mediante el siguiente Código en DAX, se cuentan todos los valores “user_uuid” de la tabla CL ORDERS, pero sin contar los repetidos, cada registro solo se cuenta una vez.

```
1 TotalUsuarios = COUNTX(VALUES('CL ORDERS - testcube'[user_uuid]), 'CL ORDERS - testcube'[user_uuid])
```

Figura 31. Dax 1

Ahora para calcular el GMV cuadrado.

```
1 GMV_cuadrado = POWER(SUM('FU ORDER DETAIL - testcube'[payment_amount]) + SUM('FU ORDER DETAIL - testcube'[earnings]), 2)
```

Figura 32. Dax 2

Para la cantidad de artículos vendidos.

```
1 CantidadArticulosVendidos = SUM('FU ORDER DETAIL - testcube'[quantity])
```

Figura 33. Dax 3

Para la evolución de las ganancias a lo largo del tiempo se hizo sin DAX de la siguiente manera:

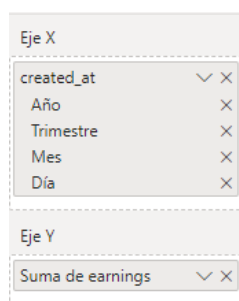


Figura 34. Dax 4

Del mismo modo se generaron varios reporte y análisis, estos muestran en la presentación de Power point presente en la misma carpeta donde se encuentra este documento, para más información dirigirse al PowerBI o la presentación.

El dashboard creado a partir de las tablas proporcionadas ofrece una visión integral y detallada de la información clave relacionada con las órdenes y ventas. Este tablero de control, diseñado para facilitar el análisis de datos, incorpora seis filtros esenciales que permiten una personalización completa: País-Ciudad, Zona-Dirección, Barrio, Manager ID, Categoría de Producto y Nombre del Producto.

En la sección de tarjetas, se presentan métricas fundamentales, como el número de usuarios, el GMV al cuadrado, las ganancias totales y el total de artículos vendidos. Estas tarjetas proporcionan una instantánea rápida de los indicadores clave del rendimiento del negocio.

El dashboard incluye un pie chart que visualiza las ganancias por ciudad, lo que facilita la identificación de patrones geográficos en el rendimiento financiero. También se destacan el valor

promedio de la orden y el número de productos por orden, brindando información sobre los hábitos de compra de los usuarios.

En la sección de análisis de costos, una tabla presenta las categorías que generan mayores costos. Esto ofrece insights cruciales para la toma de decisiones estratégicas. Las gráficas de barras comparativas muestran las ganancias de diferentes categorías y productos, así como la frecuencia de compra asociada a cada categoría y nombre de producto.

Además, se presentan pie charts que desglosan las ganancias según el canal de creación, el nombre del cupón, el tipo de propiedad y el estado de la orden. Estos gráficos proporcionan una comprensión detallada de las fuentes y condiciones de las transacciones.

En la sección de detalles de usuario, una tabla muestra información específica, como el user ID, dirección, barrio, ganancias generadas por el usuario, tipo de propiedad y ciudad. Acompañando esta tabla, un mapa visualiza las direcciones proporcionadas, facilitando la comprensión geoespacial de la distribución de usuarios.

Finalmente, las gráficas de líneas ofrecen una perspectiva temporal de las ganancias y los pagos, permitiendo la identificación de tendencias a lo largo del tiempo. Este enfoque temporal es esencial para discernir si las fluctuaciones en las ganancias se deben a cambios en el volumen de compras o a la variación en la rentabilidad. En resumen, el dashboard proporciona una herramienta integral para analizar datos, facilitando la identificación de patrones, la toma de decisiones informada y la comprensión profunda del rendimiento del negocio. A partir de este completo dashboard, se pueden derivar varios análisis que ofrecen una visión profunda de las operaciones comerciales y el comportamiento de los clientes.

Algunos de los análisis clave incluyen:

- **Análisis Geográfico:** Identificación de las ubicaciones geográficas más rentables y la distribución de ganancias por ciudad. Evaluación de la frecuencia de compra y las preferencias de producto en diferentes zonas, barrios o países-ciudades.
- **Análisis de Producto:** Identificación de las categorías y productos más rentables, lo que permite ajustar estrategias de marketing y gestión de inventario. Comparación de la frecuencia de compra por categoría y por nombre de producto para comprender las preferencias del cliente.
- **Análisis de Usuarios:** Evaluación del rendimiento individual de los usuarios mediante la tabla detallada, que muestra sus contribuciones en términos de ganancias. Exploración de la relación entre la ubicación del usuario, el tipo de propiedad y el comportamiento de compra.
- **Análisis de Órdenes y Transacciones:** Observación de métricas clave, como el valor promedio de una orden y el número de productos por orden, para entender el comportamiento de compra típico. Seguimiento del estado de las órdenes (confirmado, cancelado, enviado, etc.) para evaluar el rendimiento operativo y la satisfacción del cliente.

- **Análisis de Canales y Cupones:** Evaluación del impacto de diferentes canales de creación en las ganancias. Identificación de la efectividad de los cupones en términos de generación de ingresos.
- **Análisis Temporal:** Posible análisis de la evolución de las ventas a lo largo del tiempo utilizando las tarjetas y visualizaciones temporales. Identificación de patrones estacionales o tendencias de compra a lo largo del tiempo.
- **Análisis de Costos y Rentabilidad:** Determinación de las categorías que más contribuyen a las ganancias y los productos más rentables. Evaluación del impacto financiero de las estrategias de descuento o promoción.

Estos análisis proporcionan una visión integral de la salud comercial, permitiendo a los responsables de la toma de decisiones identificar oportunidades de mejora, ajustar estrategias y optimizar el rendimiento general del negocio.

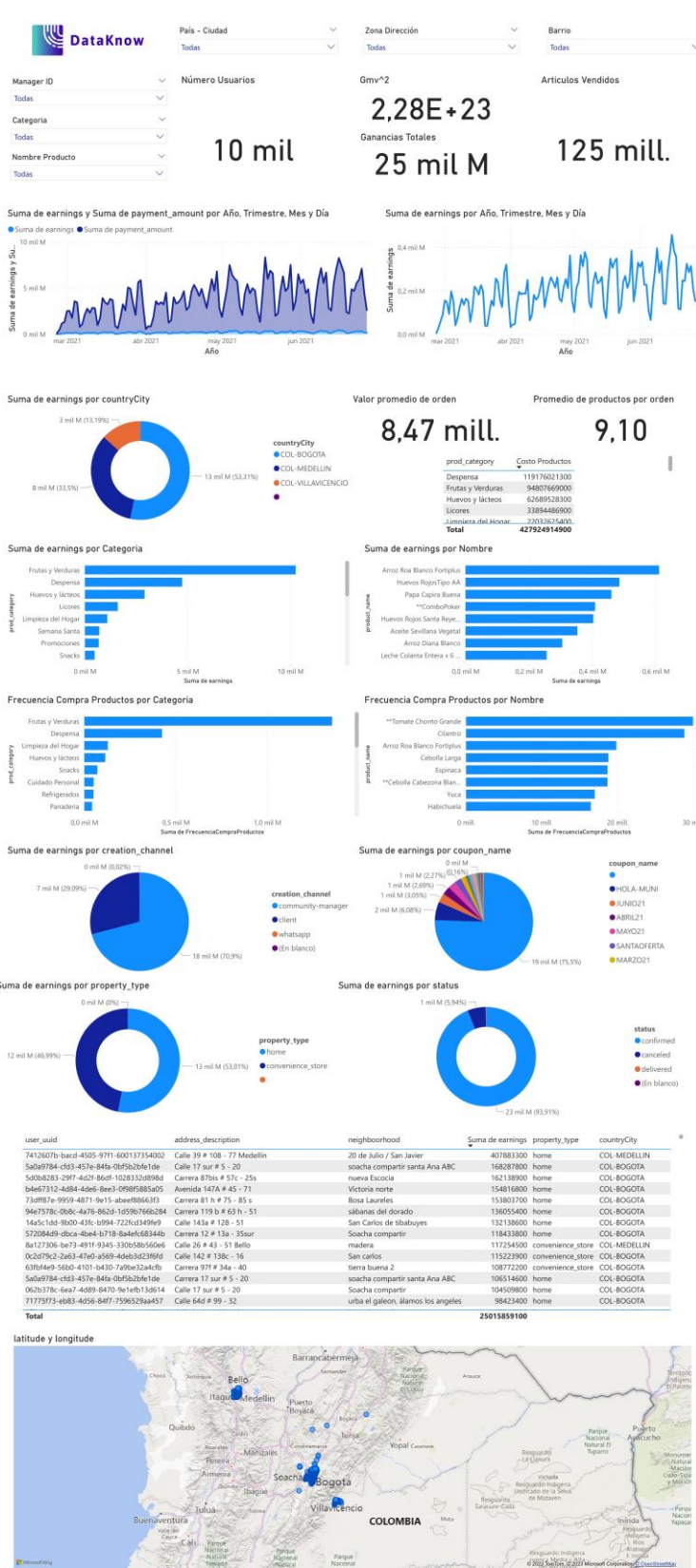


Figura 35. Dashboard Creado