

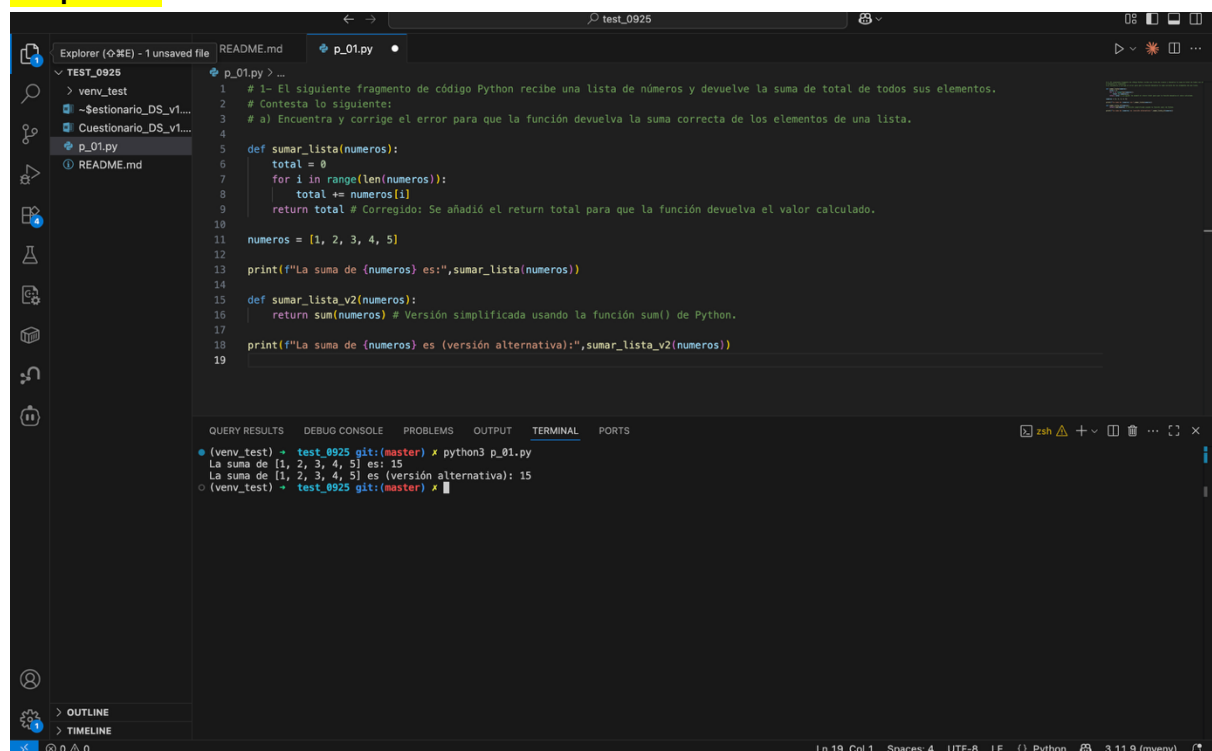
Por favor contesta este examen de manera individual. Esto es puedes informarte con otras personas o buscar en Internet información sobre las respuestas, pero en última instancia debes ser tú quien conteste el examen, tú debes de entender las preguntas y lo que estás respondiendo, es posible que te preguntemos sobre tus respuestas. **Adicionalmente, no está permitido el uso de herramientas generativas de texto como ChatGPT, Gemini, copilot, colab entre otras.** Finalmente, un mal resultado te puede descartar como candidato por lo que trata de hacer un buen examen.

1-El siguiente fragmento de código Python recibe una lista de números y devuelve la suma de total de todos sus elementos. Contesta lo siguiente:

a) Encuentra y corrige el error para que la función devuelva la suma correcta de los elementos de una lista.

```
def sumar_lista(numeros):  
    total = 0  
    for i in range(len(numeros)):  
        total += numeros[i]  
    return i
```

Respuesta:



```
1 # 1- El siguiente fragmento de código Python recibe una lista de números y devuelve la suma de total de todos sus elementos.  
2 # Contesta lo siguiente:  
3 # a) Encuentra y corrige el error para que la función devuelva la suma correcta de los elementos de una lista.  
4  
5 def sumar_lista(numeros):  
6     total = 0  
7     for i in range(len(numeros)):  
8         total += numeros[i]  
9     return total # Corregido: Se añadió el return total para que la función devuelva el valor calculado.  
10  
11 numeros = [1, 2, 3, 4, 5]  
12  
13 print(f"La suma de {numeros} es:",sumar_lista(numeros))  
14  
15 def sumar_lista_v2(numeros):  
16     return sum(numeros) # Versión simplificada usando la función sum() de Python.  
17  
18 print(f"La suma de {numeros} es (versión alternativa):",sumar_lista_v2(numeros))  
19
```

Terminal Output:

```
(venv_test) ➔ test_0925 git:(master) x python3 p_01.py  
La suma de [1, 2, 3, 4, 5] es: 15  
La suma de [1, 2, 3, 4, 5] es (versión alternativa): 15  
(venv_test) ➔ test_0925 git:(master) x
```

b) Explica brevemente cómo usarías Git y GitHub para subir el cambio al repositorio “funciones_etl”. Describe un flujo de trabajo típico que incluya el uso de ramas (branching), confirmaciones de cambios (commits), y solicitudes de extracción (pull requests) para colaborar con otros desarrolladores de datos.

Respuesta:

Liga del repositorio: https://github.com/Jorge-Polanco-Roque/funciones_etl

Pasos para subir al repo remoto:

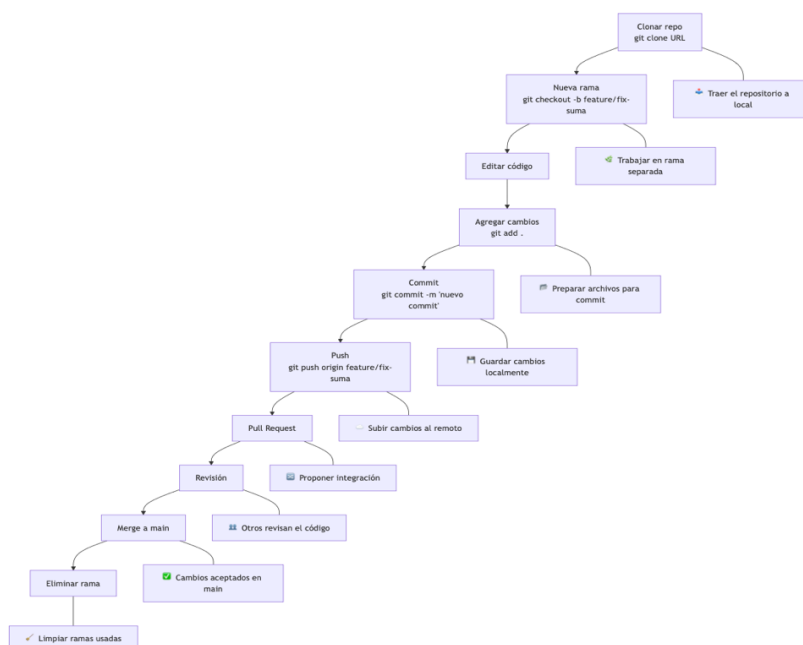
Nota: Primero llame al repo como “test_0925”, después le cambié el nombre de “funciones_etl”. De allí, la diferencia con el nombre del repo vs lo que sale en el script.

```
QUERY RESULTS  DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL  PORTS
(venv_test) + test_0925 git:(master) * git init
Initialized empty Git repository in /Users/A1064331/Desktop/Jorge/Otros/test_0925/.git/
(venv_test) + test_0925 git:(main) * git add .
(venv_test) + test_0925 git:(main) * git commit -m "Primer commit"
[main (root-commit) 02992a9] Primer commit
3 files changed, 27 insertions(+)
create mode 100644 Cuestionario_DS_v1.0 3.docx
create mode 100644 README.md
create mode 100644 p_01.py
(venv_test) + test_0925 git:(main) git branch -M main
(venv_test) + test_0925 git:(main) git remote add origin https://github.com/Jorge-Polanco-Roque/test_0925.git
(venv_test) + test_0925 git:(main) git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 3.96 MiB | 24.11 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Jorge-Polanco-Roque/test_0925.git
 * [new branch] main -> main
branch 'main' set up to track 'origin/main'.
(venv_test) + test_0925 git:(main) |
```

Explicación:

Nota: El Markdown Mermaid para hacer la siguiente gráfica, y el pdf, están en el repo.

Flujo de trabajo con Git y GitHub



2- El equipo de negocio te comparte una tabla con información de clientes y sus recargas, para ayudarte a construir un modelo de predicción de valor futuro del cliente (Customer Lifetime Value). La tabla contiene los siguientes campos:

cliente_id	fecha_recarga	monto_recarga	canal	plan_tarifario
1001	2025-05-10	100	App	AT&T Más
1002	2025-05-12	50	Tienda	Unefon Ilimitado 1.0
1001	2025-05-15	100	App	AT&T Más
...

Creando tabla de recarga:

```

1  -- Crear una tabla para registrar recargas de clientes ---
2  CREATE TABLE recargas (
3      cliente_id INT,
4      fecha_recarga DATE,
5      monto_recarga DECIMAL(10,2),
6      canal VARCHAR(50),
7      plan_tarifario VARCHAR(100)
8  );
9
10 -- Insertar datos de ejemplo ---
11 INSERT INTO recargas (cliente_id, fecha_recarga, monto_recarga, canal, plan_tarifario)
12 VALUES
13 (1001, '2025-05-10', 100, 'App', 'AT&T Más'),
14 (1002, '2025-05-12', 50, 'Tienda', 'Unefon Ilimitado 1.0'),
15 (1001, '2025-05-15', 100, 'App', 'AT&T Más');
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

QUERY RESULTS DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL PORTS

```

(venv_test) - test_0925 git:(main) x sqlite3 p_02_a.db < p_02_a.sql
(venv_test) - test_0925 git:(main) x sqlite3 p_02_a.db
SQLite version 3.43.2 2023-10-10 13:00:14
Enter ".help" for usage hints.
sqlite> SELECT cliente_id, canal, SUM(monto_recarga) AS total FROM recargas GROUP BY cliente_id, canal;
1001|App|200
1002|Tienda|50
sqlite>

```

Responde lo siguiente y en caso donde aplique, escribir el código SQL:

- a) **Validaciones de calidad:** Realiza al menos cinco validaciones que realizarías sobre esta tabla antes de utilizarla con su correspondiente sentencia SQL.

Revisión de nullos:

```

16
17 -- ¿Hay nulos en las columnas? ---
18 SELECT "cliente_id", COUNT(*) - COUNT(cliente_id) AS nulos FROM recargas
19 UNION ALL
20 SELECT "fecha_recarga", COUNT(*) - COUNT(fecha_recarga) FROM recargas
21 UNION ALL
22 SELECT "monto_recarga", COUNT(*) - COUNT(monto_recarga) FROM recargas
23 UNION ALL
24 SELECT "canal", COUNT(*) - COUNT(canal) FROM recargas
25 UNION ALL
26 SELECT "plan_tarifario", COUNT(*) - COUNT(plan_tarifario) FROM recargas;
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

QUERY RESULTS DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL PORTS

```

(venv_test) - test_0925 git:(main) x sqlite3 p_02_a.db
SQLite version 3.43.2 2023-10-10 13:00:14
Enter ".help" for usage hints.
sqlite> SELECT "cliente_id", COUNT(*) - COUNT(cliente_id) AS nulos FROM recargas
...> UNION ALL
...> SELECT "fecha_recarga", COUNT(*) - COUNT(fecha_recarga) FROM recargas
...> UNION ALL
...> SELECT "monto_recarga", COUNT(*) - COUNT(monto_recarga) FROM recargas
...> UNION ALL
...> SELECT "canal", COUNT(*) - COUNT(canal) FROM recargas
...> UNION ALL
...> SELECT "plan_tarifario", COUNT(*) - COUNT(plan_tarifario) FROM recargas;
cliente_id|
fecha_recarga|
monto_recarga|
canal|
plan_tarifario|
sqlite>

```

Revisión de duplicados:

```

26 --- Conteo de duplicados por columna ---
27 SELECT 'cliente_id' AS columna, COUNT(*) - COUNT(DISTINCT cliente_id) AS duplicados FROM recargas
28 UNION ALL
29 SELECT 'fecha_recarga', COUNT(*) - COUNT(DISTINCT fecha_recarga) FROM recargas
30 UNION ALL
31 SELECT 'monto_recarga', COUNT(*) - COUNT(DISTINCT monto_recarga) FROM recargas
32 UNION ALL
33 SELECT 'canal', COUNT(*) - COUNT(DISTINCT canal) FROM recargas
34 UNION ALL
35 SELECT 'plan_tarifario', COUNT(*) - COUNT(DISTINCT plan_tarifario) FROM recargas;
36

```

QUERY RESULTS DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL PORTS

```

(venv_test) + test_0925 git:(main) # sqlite3 p_02_a.db
SQLite version 3.43.2 2023-10-10 13:00:14
Enter ".help" for usage hints.
sqlite> SELECT 'cliente_id' AS columna, COUNT(*) - COUNT(DISTINCT cliente_id) AS duplicados FROM recargas
--> UNION ALL
--> SELECT 'fecha_recarga', COUNT(*) - COUNT(DISTINCT fecha_recarga) FROM recargas
--> UNION ALL
--> SELECT 'monto_recarga', COUNT(*) - COUNT(DISTINCT monto_recarga) FROM recargas
--> UNION ALL
--> SELECT 'canal', COUNT(*) - COUNT(DISTINCT canal) FROM recargas
--> UNION ALL
--> SELECT 'plan_tarifario', COUNT(*) - COUNT(DISTINCT plan_tarifario) FROM recargas;
cliente_id|1
fecha_recarga|0
monto_recarga|1
canal|1
plan_tarifario|1
sqlite>

```

Niveles de cuartiles de monto_recarga:

```

39 --- Distribución de monto_recarga por cuartiles ---
40 WITH o AS (
41     SELECT monto_recarga,
42           CUME_DIST() OVER (ORDER BY monto_recarga) AS cd
43     FROM recargas
44 )
45 SELECT
46     'Min' AS cuartil, MIN(monto_recarga) AS valor FROM o
47 UNION ALL
48 SELECT 'Q1 (25%)', MIN(monto_recarga) FROM o WHERE cd >= 0.25
49 UNION ALL
50 SELECT 'Q2 (50% - Mediana)', MIN(monto_recarga) FROM o WHERE cd >= 0.50
51 UNION ALL
52 SELECT 'Q3 (75%)', MIN(monto_recarga) FROM o WHERE cd >= 0.75
53 UNION ALL
54 SELECT 'Max', MAX(monto_recarga) FROM o;
55

```

QUERY RESULTS DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL PORTS

```

(venv_test) + test_0925 git:(main) # sqlite3 p_02_a.db
SQLite version 3.43.2 2023-10-10 13:00:14
Enter ".help" for usage hints.
sqlite> --- Distribución de monto_recarga por cuartiles ---
sqlite> WITH o AS (
--> SELECT monto_recarga,
--> CUME_DIST() OVER (ORDER BY monto_recarga) AS cd
--> FROM recargas
--> )
--> SELECT
--> 'Min' AS cuartil, MIN(monto_recarga) AS valor FROM o
--> UNION ALL
--> SELECT 'Q1 (25%)', MIN(monto_recarga) FROM o WHERE cd >= 0.25
--> UNION ALL
--> SELECT 'Q2 (50% - Mediana)', MIN(monto_recarga) FROM o WHERE cd >= 0.50
--> UNION ALL
--> SELECT 'Q3 (75%)', MIN(monto_recarga) FROM o WHERE cd >= 0.75
--> UNION ALL
--> SELECT 'Max', MAX(monto_recarga) FROM o;
Min|50
Q1 (25%)|50
Q2 (50% - Mediana)|100
Q3 (75%)|100
Max|100
sqlite>

```

Conteo de valores de plan_tarifario:

```

56 --- Conteo de registros por plan_tarifario ---
57 SELECT
58     plan_tarifario,
59     COUNT(*) AS total_registros
60 FROM recargas
61 GROUP BY plan_tarifario
62 ORDER BY total_registros DESC;
63

```

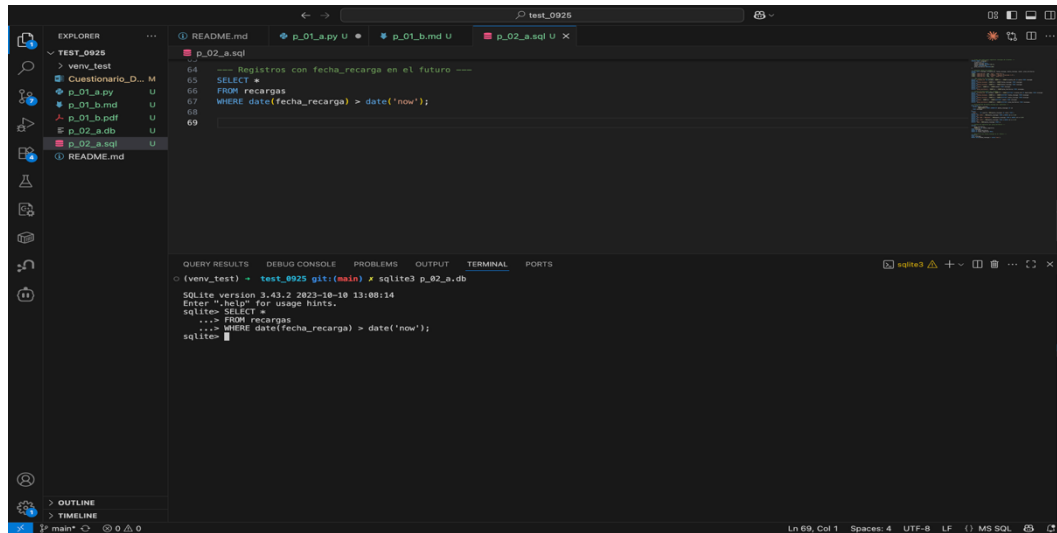
QUERY RESULTS DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL PORTS

```

(venv_test) + test_0925 git:(main) # sqlite3 p_02_a.db
SQLite version 3.43.2 2023-10-10 13:00:14
Enter ".help" for usage hints.
sqlite> SELECT
--> plan_tarifario,
--> COUNT(*) AS total_registros
--> FROM recargas
--> GROUP BY plan_tarifario
--> ORDER BY total_registros DESC;
AT&T Masi2
Unefon Ilimitado 1.0|1
sqlite>

```

¿Hay registros futuros?:



The screenshot shows a VS Code editor with a file explorer on the left containing files like `venv_test`, `Questionario_D...`, `p_01_a.py`, `p_01_b.md`, `p_01_b.pdf`, `p_02_a.db`, `p_02_a.sql`, and `README.md`. The main editor displays a SQL query in `p_02_a.sql`:

```
64 --- Registros con fecha_recarga en el futuro ---
65 SELECT *
66 FROM recargas
67 WHERE date(fecha_recarga) > date('now');
68
69
```

The bottom panel shows the terminal with the following output:

```
(venv_test) ~ test_0925 git:(main) # sqlite3 p_02_a.db
SQLite version 3.43.2 2023-10-10 13:08:14
Enter ".help" for usage hints.
sqlite> SELECT *
...> FROM recargas
...> WHERE date(fecha_recarga) > date('now');
sqlite>
```

b) Validaciones de negocio: ¿Qué aspectos revisarías para asegurarte de que la información cumple con lo esperado para el caso de uso (valor del cliente)?

- Para garantizar que la tabla sea útil en la estimación del valor de vida del cliente, lo primero es confirmar que los montos de recarga están dentro de los límites definidos por el negocio (evitando valores negativos, exageradamente altos, etc.).
- Algunas causísticas pudieran ser necesario aplicar en la limpieza/filtro. Pensando en la frecuencia de recargas, un mismo cliente no debería registrar múltiples operaciones idénticas en el mismo día, lo cual podría reflejar errores o fraudes. Mediante el conocimiento de negocio, y análisis estadístico, se pudieran crear dichas reglas.
- Además, los campos categóricos como canal y plan_tarifario deben corresponder a catálogos vigentes y controlados, los cuales debería estar validados y aprobados por el negocio.
- En términos temporales, las fechas no deben situarse en el futuro y deben respetar la coherencia en estructura.
- Finalmente, el volumen diario debe ser consistente con la tendencia histórica, de modo que caídas o picos inusuales activen alertas antes de alimentar un modelo.

c) Preparación de la tabla: ¿Qué manipulaciones o transformaciones aplicarías para que la tabla esté lista para análisis/modelado? con su correspondiente sentencia SQL

Recency, Frequency, Monetary (RFM) Analysis:

The screenshot shows a SQL IDE with a query editor and a results pane. The query is as follows:

```

1  -- Recency, Frequency and Monetary (RFM) Analysis --
2  WITH base AS (
3      SELECT
4          cliente_id,
5          COUNT(*) AS frecuencia,
6          SUM(monto_recarga) AS valor_monetario,
7          AVG(monto_recarga) AS ticket_promedio,
8          MAX(fecha_recarga) AS ultima_recarga,
9          MIN(fecha_recarga) AS primera_recarga
10     FROM recargas
11     GROUP BY cliente_id
12 )
13 , rfm AS (
14     SELECT
15         b.cliente_id AS Cliente,
16         b.frecuencia AS Frecuencia,
17         b.valor_monetario AS Total_Recargado,
18         b.ticket_promedio AS Ticket_Promedio,
19         CAST(JULIANDAY(CURRENT_DATE) - JULIANDAY(b.ultima_recarga) AS INT) AS Recencia_Dias,
20         CAST(JULIANDAY(CURRENT_DATE) - JULIANDAY(b.primera_recarga) AS INT) AS Dias_Actividad,
21         b.frecuencia * b.ticket_promedio AS Ingreso_Estimado,
22         COUNT(DISTINCT r.canal) AS Num_Canales,
23         COUNT(DISTINCT r.plan_tarifario) AS Num_Planes
24     FROM base b
25     JOIN recargas r ON r.cliente_id = b.cliente_id
26     GROUP BY b.cliente_id, b.frecuencia, b.valor_monetario,
27             b.ticket_promedio, b.ultima_recarga, b.primera_recarga
28 )
29 SELECT * FROM rfm;
30

```

The results pane shows the output of the query, including columns like ticket_promedio, Recencia_Dias, Dias_Actividad, Ingreso_Estimado, Num_Canales, and Num_Planes, along with sample data rows.

- El análisis RFM permite resumir el comportamiento histórico de los clientes en tres dimensiones clave: qué tan reciente fue su última transacción, con qué frecuencia interactúa con la empresa y cuánto ha gastado en total o en promedio. Esta segmentación ofrece una visión clara para identificar clientes valiosos, inactivos o con alto potencial de retención.
- El siguiente paso natural es avanzar hacia un modelo de Customer Lifetime Value (CLV), sino que busca predecir el valor económico futuro de cada cliente.
- Para ello, se utilizan las métricas RFM como variables de entrada con información adicional como canales de interacción, planes contratados o periodos de inactividad. Como variable objetivo, buscaría algún indicador de CLV (ej. Ventas durante los primeros X años desde que iniciaron relación con la empresa). Ya con ello, se aplican métodos de predicción Supervisado, en donde probaría con varios algoritmos/arquitecturas para ello (ej. Xgboost). Seleccionando el modelo con menor nivel de error (ej. MAPE).

d) Supón que la tabla se actualiza diariamente a día vencido. ¿Qué mecanismo implementarías para verificar que los datos están actualizados correctamente?

- **Monitoreo de ingesta:** validar que el proceso de carga se ejecutó en la fecha esperada y sin errores (logs, status jobs, tiempos de ejecución).
- **Conteo de registros:** comparar el volumen de registros contra días previos o contra umbrales esperados (detecta faltantes o duplicados).
- **Validación de fecha de actualización:** asegurar que la columna de fecha tenga datos hasta el día vencido más reciente.
- **Alertamiento automático:** configurar reglas de control que disparen notificaciones si no se cumple el Service Level Agreement (ej. datos incompletos, retrasos, fallos de conexión).

- **Chequeos de calidad (umbrales):** revisar nulos, duplicados y valores fuera de rango en las columnas críticas.
- **Análisis de tendencias:** verificar consistencia en montos y frecuencia de recargas frente al histórico (detección de outliers).
- **Escalamiento de alertas:** si hay desviaciones, invocar matriz de comunicación con responsables (TI, Calidad, Negocio).
- **Reporte de servicio diario:** documentar si la carga fue exitosa, con métricas de calidad y disponibilidad.

3-Lista ejemplos concretos de situaciones o tipos de datos en las que elegirías pandas y otros donde preferirías PySpark.

Pandas: lo elegiría para análisis exploratorio y manipulación de datasets pequeños o medianos que pueden cargarse en memoria local. Ejemplos concretos: limpieza de un CSV de encuestas de clientes, transformación de un Excel con métricas financieras, preparación de features en notebooks durante fases iniciales de modelado, o validación puntual de la calidad de datos. Pandas es rápido y muy flexible en entornos locales, ideal para prototipado y pruebas.

PySpark: lo aplicaría en escenarios de big data y procesamiento distribuido, cuando los volúmenes de información exceden la memoria de una sola máquina o requieren escalabilidad horizontal. Ejemplos: procesamiento de logs de millones de eventos diarios en un data lake, consolidación de historiales de transacciones bancarias de varios años, generación de datasets analíticos sobre terabytes de información, o ejecución de pipelines ETL en producción. PySpark, al correr sobre clústeres, permite tolerancia a fallos y optimización para cargas masivas.

4-¿Que hace la operación “cache” en pyspark y cuando la ocuparías?

Se usa para acelerar procesos distribuidos en PySpark guardando los resultados intermedios en memoria cuando sabes que vas a reutilizar esos datos varias veces.

5-Explique el concepto de “feature engineering” y mencione algunas técnicas comunes que ha usado.

Feature engineering es el proceso de crear, transformar o seleccionar variables a partir de los datos crudos para mejorar el desempeño de un modelo de machine learning. Su objetivo es representar mejor la información relevante, capturar patrones útiles y facilitar que los algoritmos aprendan de forma más eficiente.

Algunas técnicas comunes que he usado son:

- **Transformaciones numéricas:** normalización, estandarización, escalado logarítmico.
- **Codificación de variables categóricas:** one-hot encoding, label encoding, target encoding.
- **Manejo de fechas y tiempo:** extracción de día, mes, día de la semana, periodos, diferencias de tiempo.

- **Generación de variables agregadas:** medias, sumas, conteos, ratios o rolling windows sobre series temporales.
- **Selección de características:** Principal Component Analysis, importancia de features en árboles o eliminación de colinealidad.

6-Si un modelo que implementó tiene un desempeño pobre en producción, ¿cómo abordaría el diagnóstico y la mejora?

- Si un modelo tiene un desempeño pobre en producción, lo primero es hacer un diagnóstico estructurado para entender si el problema viene de los datos, del entrenamiento o del despliegue.
- Evaluaría si el modelo está sufriendo de underfitting (demasiado simple, no captura patrones) o de overfitting (aprendió ruido del set de entrenamiento y no generaliza bien).
- En el caso de modelos de clasificación, revisaría el balanceo de etiquetas: un dataset muy desbalanceado puede llevar a que el modelo prediga casi siempre la clase mayoritaria.
- A partir de ahí, aplicaría estrategias como ajustar la complejidad del modelo (underfitting/overfitting), usar regularización (overfitting), aumentar datos o aplicar técnicas de re-muestreo (desbalanceo de etiquetas).
- Finalmente, validaría si en producción los datos tienen la misma distribución que en entrenamiento (data drift) y consideraría retrainings periódicos con datos más recientes.

7-Supón que tienes monedas con valores [1, 5, 7, 13] (asume que las monedas siempre están en orden ascendente dentro de la lista de Python), escribe un algoritmo en Python para que encuentre el cambio necesario para pagar cierto artículo. Por ejemplo, si un artículo cuesta 18, entonces sería necesario usar una moneda de 5 y otra de 13. Trata de que sea con el menor número de monedas posibles. Por ejemplo, 43 se puede lograr de manera mínima con 5 monedas: 2 monedas de 13, 1 de 7 y 2 de 5. El resultado debe ser una lista de Python con el número de monedas de cada tipo, para el ejemplo anterior la solución sería: [0,2,1,2], indicando 0 monedas de 1, 2 monedas de 5, 1 moneda de 13 y 2 de 5. **No uses herramientas generativas de texto como colab, chatgpt, etc. Es mejor una respuesta a medias que una respuesta copiada.**


```

1 # 7-Supón que tienes monedas con valores [1, 5, 7, 13] (asume que las monedas siempre están en orden ascendente dentro de la lista de Python).
2
3 # Definimos la función que calcula el cambio mínimo
4 def cambio_minimo(monedas, monto):
5     # Creamos una lista llena de ceros para guardar cuántas monedas de cada tipo usaremos
6     resultado = [0] * len(monedas)
7
8     # Recorremos la lista de monedas desde la más grande hasta la más pequeña (índices en orden inverso)
9     for i in range(len(monedas)-1, -1, -1):
10        # Si la moneda actual cabe dentro del monto restante
11        if monto >= monedas[i]:
12            # Calculamos cuántas monedas de ese valor podemos usar
13            resultado[i] = monto // monedas[i]
14            # Actualizamos el monto restante con el residuo
15            monto = monto % monedas[i]
16        # Retornamos la lista con el número de monedas de cada tipo
17        return resultado
18
19 # Definimos las monedas disponibles
20 monedas = [1, 5, 7, 13]
21
22
23 print(cambio_minimo(monedas, 18))
24 print(cambio_minimo(monedas, 43))
25

```

QUERY RESULTS DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL PORTS

```

• (venv_test) → test_0925 git:(main) # python p_07.py
[0, 1, 0, 1]
[4, 0, 0, 3]
• (venv_test) → test_0925 git:(main) # []

```

Ln 25, Col 1 Spaces: 4 UTF-8 LF Python 3.11.9 (myenv)

8-Supón que has hecho un modelo de aprendizaje de máquina sobre un problema de clasificación binaria que tiene los siguientes pasos de preprocesamiento:

- conversión de fecha de nacimiento con formato de string a formato fecha
- normalización de la edad (pasarla a un rango de 0 a 1)
- cálculo de la edad ((fecha_actual – fecha de nacimiento) / 365.25)
- eliminación de edades que no hacen sentido (< 0, > 100)
- imputación de datos faltantes de edad (reemplazar nulos por mediana).

¿Qué parte debe de ir forzosamente dentro de un pipeline de preprocesamiento más entrenamiento de una regresión logística dentro de iteraciones de validación cruzada (1 en el código) y por otra parte qué pasos pueden ser preprocesados sin ser parte del pipeline de entrenamiento de la validación cruzada (2 en el código)?

```

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from datetime import datetime as dtm
import pandas as pd

# Puede ser que algunos imports falten

pipeline = Pipeline([
    # ... qué pasos van aquí (1)
    ('regression', LogisticRegression())
])

data = pd.read_csv('datos.csv')
X, y = data.drop('objetivo', axis=1), data[['objetivo']]

# qué pasos van aquí (2)

skf = StratifiedKFold(n_splits=3)
for train, val in skf.split(X, y):
    pipeline.fit(X.iloc[train], y.iloc[train])

    print(accuracy_score(pipeline.predict(X.iloc[val]), y.iloc[val]))

```

- a) i y ii pueden quedar fuera del pipeline de entrenamiento (2); iii, iv y v deben quedar dentro del pipeline de entrenamiento (1).
- b) Todos pueden ir afuera del pipeline de entrenamiento (2).
- c) Todos deben de ir en el pipeline de entrenamiento (1).
- d) **i, iii y iv pueden quedar fuera del pipeline de entrenamiento (2); ii y v deben quedar dentro del pipeline de entrenamiento (1).**
 - **Para evitar data leakage, solo deben ir dentro del pipeline los pasos que calculan estadísticas a partir de los datos de entrenamiento en cada fold. Por eso, la normalización y la imputación por mediana deben incluirse en el pipeline, mientras que la conversión a fecha, el cálculo de edad y la eliminación de valores imposibles son transformaciones determinísticas o reglas fijas que no dependen del split y pueden hacerse antes.**

9-Imagina que has creado un modelo que determina la probabilidad de que una persona compre un cierto producto cuando se le contacta por teléfono. Este modelo se quiere usar para contactar a las personas con mayor probabilidad de comprar dicho producto y correr campañas mensualmente. **¿Cómo debería de ser el conjunto de validación?**

- a) Separar aleatoriamente los datos, un 10% para validación y 90% para entrenamiento
- b) Separar los datos por id de cliente (campo numérico con enteros), usar los ids de clientes más chicos para entrenamiento (90%) y los ids más grandes para validación (10%).
- c) Separar los datos por fecha de envío, usar los clientes con mayor antigüedad en una campaña de este tipo para entrenamiento (90%) y los más recientes para validación (10%).
- d) **Hacer una validación cruzada con 5 folds.**
 - **Acá estoy asumiendo que se está pensando en un modelo logístico, sin ninguna estructura temporal. La variable objetivo sería dicotómica (compra o no compra), mientras que las explicativas serían algunas como: dicotómica si recibió o no una llamada, duración de llamada, número de quejas previas, demográficos, etc.**

10-Explique con sus propias palabras qué es una inteligencia artificial generativa y mencione dos ejemplos de aplicaciones prácticas actuales. Además, indique una limitación importante de estas tecnologías.

La Inteligencia Artificial Generativa es un tipo de Inteligencia Artificial diseñado para producir contenido nuevo (texto, imágenes, código, audio, etc.); ello, a partir de patrones aprendidos en grandes volúmenes de datos. No se limita a clasificar o predecir, sino que crea resultados originales que simulan razonamientos o expresiones humanas.

Entre las aplicaciones prácticas en las que he trabajado destacan:

- Un proyecto para una fintech, donde se aplicó IA Generativa para automatizar parte del proceso de KYC (Know Your Customer), generando reportes estructurados, provenientes de fuentes no estructuradas, reduciendo así la carga manual en la verificación de clientes. Todo ello desplegado como un RPA usando Google Cloud Platform.

- Un proyecto para una empresa del tipo buró de crédito, donde diseñamos un sistema que emplea agentes de IA Generativa para automatizar análisis financieros y búsquedas en internet. Todo ello embebido en una plataforma web usando AWS.

Una limitación importante de estas tecnologías es la posibilidad de alucinación o generación de información incorrecta, lo cual obliga a implementar mecanismos de validación, sobre todo en sectores sensibles como finanzas y crédito, donde la precisión es crítica. Esto se puede llegar a mitigar a través de las siguientes estrategias:

- Modularizar tareas complejas en muchas pequeñas tareas.
- Contextualización mediante técnicas como Chain of Thought, RAG, Agentes.
- Buenas prácticas de prompt engineering: frameworks de prompts, few-shot learning, etc.
- Fine-tuning de modelos, Destilados de LLMs.

Yo, Jorge Polanco Roque, certifico poniendo mi nombre en el espacio anterior que fui quien contestó este examen esto es yo fui responsable de entender las preguntas y contestarlas.