Nombre:
Ruben Ortega Perez
Apellidos:
75147350W
DNI:

Jueves, 24 de octubre de 2024.

Test de conocimientos

SQL+Python
Prueba A

Responde marcando la solución correcta.

Definiciones:

- El examen consta de 10 preguntas.
- El examen durará 15 min.
- Solo existe **una** (1) **solución** correcta.
- Cada pregunta respondida **correctamente sumará** un (1) punto.
- Cada pregunta respondida **erróneamente restará** dos (2) puntos.

Preguntas

1) **Dado el código de Python:**

```
for i in range(3):
    for j in range(1, 4):
        if i % 2 == 0 and j % 2 != 0:
            print(i * j, end=" ")
```

a) 0 0 0 0 0 0
b) El código no ejecutará, contiene un error
c) 0 2 0 4 0 6
d) 0 0 2 6

2) **¿Qué línea de código creará correctamente un DataFrame a partir de un diccionario?**

a) pd.DataFrame('nombre': ['Ana', 'Luis'], 'edad': [25, 30])
b) pd.DataFrame({'nombre': ['Ana', 'Luis'], 'edad': [25, 30]})
c) DataFrame({'nombre': ['Ana', 'Luis'], 'edad': [25, 30]})
d) pd.createDataFrame({'nombre': ['Ana', 'Luis'], 'edad': [25, 30]})

3) **¿Qué función de Seaborn se utiliza comúnmente para ver las correlaciones entre las diferentes variables de un DataFrame?**

a) sns.correlate()
b) sns.pairplot()
c) sns.correlation_plot()
d) sns.heatmap()

4) **¿Cuál es la consulta de SQL que utiliza la función ventana correctamente para asignar un rango a los empleados de cada departamento basado en su salario de mayor a menor?**

a) **SELECT DENSE_RANK() OVER** (**PARTITION BY** departamento **ORDER BY** salario **DESC**) **AS** rango, nombre
FROM empleados;

b) **SELECT** nombre, **RANK() OVER (ORDER BY** salario **DESC**) **AS** rango
FROM empleados;

c) **SELECT DENSE_RANK() OVER (PARTITION BY** departamento **ORDER BY** salario **DESC) AS** rango, nombre, departamento
FROM empleados;

d) **SELECT** departamento, nombre, **RANK() OVER (ORDER BY** departamento, salario **DESC**)
FROM empleados;

5) **Considerando Matplotlib para personalizar las visualizaciones ¿cuál es el propósito de utilizar `ax.set_xticklabels(labels, rotation=45, ha='right')` en el contexto de una visualización?**

a) Cambiar el color de los ejes x e y del gráfico.
b) Establecer un límite fijo para los valores mostrados en el eje x del gráfico.
c) Ajustar la orientación y alineación de las etiquetas del eje x para mejorar la legibilidad, rotándolas 45 grados y alineándolas a la derecha.
d) Crear una leyenda personalizada para el eje x con etiquetas rotadas 45 grados.

6) **Utilizando Selenium WebDriver en Python ¿Cómo podrías esperar explícitamente que un elemento sea clicable antes de realizar una acción sobre él?**

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

a) *element = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.ID, 'submit')))*
b) *element = driver.wait_for_element('submit', timeout=10)*
c) *element = driver.click_on_element_when_ready('submit', timeout=10)*
d) *element = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, 'submit')))*

7) **Utilizando pandas para la manipulación de datos ¿cuál es la mejor manera de aplicar una función personalizada que requiere múltiples columnas de un DataFrame para generar una nueva columna?**

a) *df['D'] = df.apply(lambda row: custom_function(row), axis=1)*
b) *df['D'] = custom_function(df['A'], df['B'], df['C'])*
c) *df['D'] = df.apply(custom_function(df['A'], df['B'], df['C']))*
d) *df['D'] = df[['A', 'B', 'C']].apply(custom_function)*

8) **Tienes un Dataframe df muy grande que contiene transacciones de los clientes con las siguientes columnas 'ID_cliente', 'Fecha', 'Importe', 'Producto'. Necesitas generar un reporte que muestre el total de gastos de cada cliente. ¿Cuál es el enfoque más eficiente?**

a) Iterar sobre cada fila del DataFrame, sumar los montos por cliente y almacenar los resultados en un diccionario.
b) Utilizar df.groupby('ID_cliente')['Importe'].sum() para obtener directamente la suma de gastos por cliente.
c) Crear un bucle for que recorra todos los ID de clientes únicos y luego filtrar el DataFrame por cada ID para calcular la suma de montos individualmente.
d) Exportar los datos a una base de datos SQL y usar una consulta SQL para sumar los montos por cliente.

9) **Estás trabajando con un DataFrame df de pandas que contiene información de clientes incluyendo las columnas 'Edad', 'Ciudad', 'Ingresos'. Quieres filtrar por todos los clientes que tienen más de 30 años y viven en Madrid. ¿Cuál es el método más eficiente?**

a) Utilizar un bucle for para iterar sobre todas las filas del DataFrame y seleccionar manualmente las que cumplen con ambas condiciones.
b) Crear dos DataFrames separados, uno filtrando por edad y otro por ciudad, y luego usar *pd.merge()* para combinarlos.
c) Usar df.find*("Edad > 30 & Ciudad == 'Madrid'")* para filtrar el DataFrame mediante una consulta de texto.
d) Emplear *df[(df['Edad'] > 30) & (df['Ciudad'] == 'Madrid')]* para filtrar el DataFrame utilizando condiciones booleanas.

10) Dentro de una tabla `ventas` con columnas `fecha`, `categoria` e `importe`, ¿cómo calcularías el total de imgresos por categoría, además de un conteo condicional que solo cuente las ventas mayores a 100€?

a)
```
SELECT categoria,
    SUM (importe) AS total_ventas,
    COUNT (CASE WHEN importe > 100 THEN 1 ELSE NULL END)
AS ventas_mayores_100
FROM ventas
GROUP BY categoria;
```

b)
```
SELECT categoria,
   SUM (importe) AS total_ventas,
    SUM( IF (importe > 100, 1, 0)) AS ventas_mayores_100
FROM ventas
GROUP BY categoria;
```

c)
```
SELECT categoria,
    SUM (importe) AS total_ventas,
    COUNT(*) WHERE importe > 100 AS ventas_mayores_100
FROM ventas
GROUP BY categoria;
```

d)
```
SELECT categoria,
        SUM (monto) AS total_ventas,
        COUNT (monto > 100) AS ventas_mayores_100
FROM ventas
GROUP BY categoria;
```