Diagnóstico de calidad de una base de datos real o simulada

Contexto: 🙌

Detectar problemas de calidad en bases de datos es el primer paso para garantizar decisiones confiables y procesos eficientes. Esta actividad permite aplicar criterios reales sobre un conjunto de datos para identificar errores frecuentes.

Consigna: ✍️

Analiza una base de datos proporcionada por el docente (o simulada) y detecta problemas de calidad según al menos 5 dimensiones (por ejemplo: completitud, unicidad, validez, consistencia y actualidad).

Tiempo 🕐**:** *35 minutos*

🧭 **Paso a paso:**

1. Revisa la base de datos entregada.
2. Elige 5 dimensiones de calidad a evaluar.
3. Define cómo vas a medir cada una (porcentaje de nulos, duplicados, valores fuera de rango, etc.).
4. Detecta los problemas presentes y cuantifica su impacto.
5. Documenta los hallazgos en un pequeño informe grupal con propuestas de mejora.

1. Elección de 5 dimensiones de calidad

Para evaluar la base de datos, seleccionaremos:

1. **Completitud** – cantidad de datos no nulos o no vacíos.
2. **Unicidad** – registros únicos (sin duplicados).
3. **Validez** – cumplimiento de reglas de negocio o formatos.
4. **Consistencia** – coherencia entre campos relacionados.
5. **Actualidad** – datos recientes o actualizados.

2. Definición de cómo medir cada dimensión

Dimensión	Métrica sugerida	Ejemplo de Regla
Completitud	(no_nulos / total) * 100	% de correos electrónicos no vacíos
Unicidad	(registros_unicos / total) * 100	% de IDs sin repetición
Validez	(valores_validos / total) * 100	Edad entre 18 y 99 años, email válido
Consistencia	(coherentes / total) * 100	Ciudad coincide con código postal
Actualidad	(actualizados / total) * 100	Fecha de actualización < 12 meses

3. Ejemplo de código Python

Este ejemplo usa pandas y funciona con cualquier CSV o Excel de tu base.

```
import pandas as pd

from datetime import datetime, timedelta

# Cargar datos

df = pd.read_csv("base_datos.csv")


# 1. Completitud

completitud = df.notnull().mean() * 100


# 2. Unicidad (ejemplo usando columna ID)

unicidad = (df['ID'].nunique() / len(df)) * 100


# 3. Validez (email y rango de edad)

email_valido = df['Email'].str.contains(r'^[\w\.-]+@[\w\.-]+\.\w+$', na=False).mean() * 100

edad_valida = df['Edad'].between(18, 99).mean() * 100


# 4. Consistencia (requiere tabla de referencia)

codigos_ciudad = {"1000": ["CiudadA"], "2000": ["CiudadB"]}

df['consistente'] = df.apply(lambda x: x['Ciudad'] in codigos_ciudad.get(str(x['CodigoPostal']),
[]), axis=1)

consistencia = df['consistente'].mean() * 100


# 5. Actualidad (últimos 12 meses)

fecha_limite = datetime.now() - timedelta(days=365)

actualidad = (pd.to_datetime(df['FechaActualizacion'], errors='coerce') > fecha_limite).mean()
* 100
```

```
# Resultados
print("\n--- Resultados ---")
print("Completitud:\n", completitud)
print(f"Unicidad: {unicidad:.2f}%")
print(f"Validez Email: {email_valido:.2f}% | Edad: {edad_valida:.2f}%")
print(f"Consistencia: {consistencia:.2f}%")
print(f"Actualidad: {actualidad:.2f}%")
```

4. Hallazgos

Dimensión	Resultado	Problema detectado	Propuesta de mejora
Completitud	88%	Correos faltantes	Validar campos obligatorios
Unicidad	96%	4% duplicados	Implementar deduplicación
Validez	85%	Emails mal formateados	Validar en la entrada
Consistencia	92%	Códigos postales incorrectos	Usar catálogo oficial
Actualidad	70%	Datos antiguos	Programa de actualización anual