Universidad Anáhuac Puebla

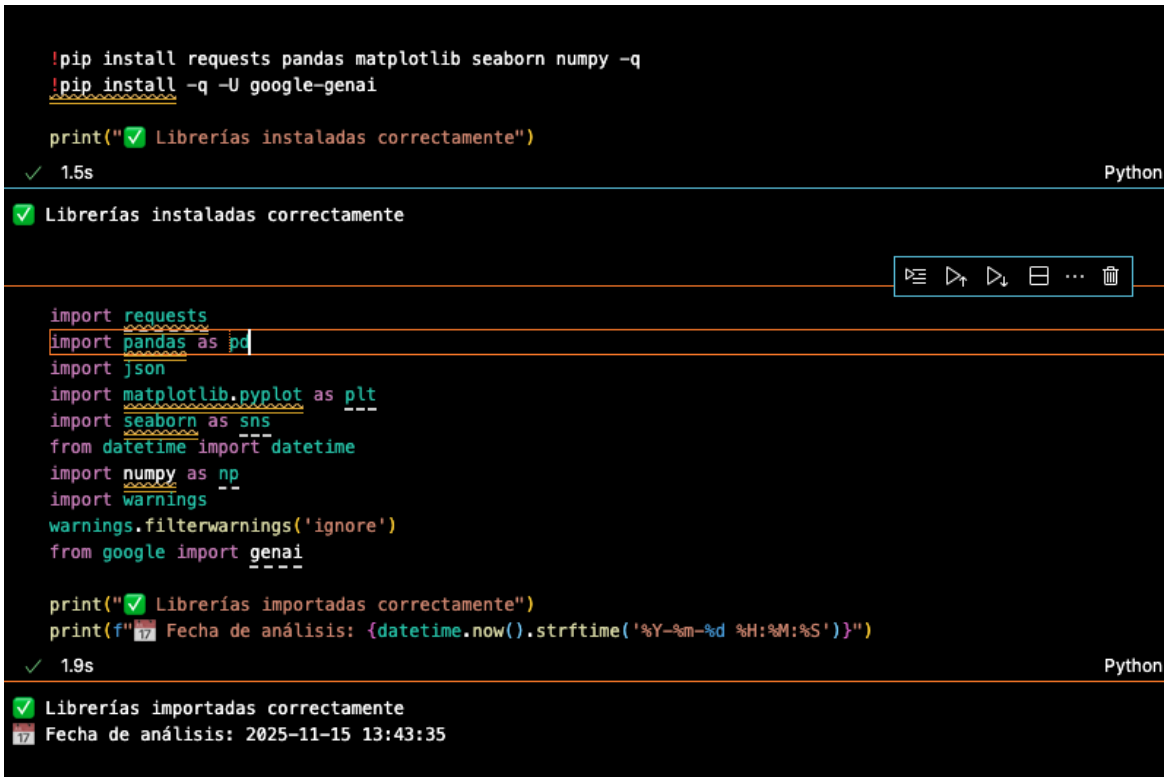
Actividad de aprendizaje: Examen 1 Ciencia de Datos

Profesor: Farid Krayem Pineda

Alumno: Hugo Iván Trejo González

ID:00591853 **Fecha: 16/11/2025**

Importación de librerías y testing de la API de gecko



```python
import requests

BASE_URL = "https://api.coingecko.com/api/v3"

def hacer_peticion_api(endpoint, params=None):
    """
    Hace una petición GET a un endpoint de la API de CoinGecko.
    """
    try:
        # Normaliza endpoint y arma URL
        if not endpoint.startswith("/"):
            endpoint = "/" + endpoint
        url = f"{BASE_URL}{endpoint}"

        # Si es /coins/markets, asegúrate de incluir vs_currency
        if endpoint.rstrip("/") == "/coins/markets":
            params = dict(params or {})
            params.setdefault("vs_currency", "usd")

        # (Opcional) headers para evitar bloqueos por UA
        headers = {"Accept": "application/json", "User-Agent": "HugoNotebook/1.0"}

        # Depuración útil
        # print("URL:", url, "PARAMS:", params)

        resp = requests.get(url, params=params, headers=headers, timeout=15)

        if resp.status_code == 200:
            print(f"✅ Petición exitosa a: {endpoint}")
            return resp.json()
        else:
            print(f"❌ Error {resp.status_code}: {resp.text[:400]}")
            return None

    except requests.exceptions.Timeout:
        print("❌ Timeout: La petición tardó demasiado")
        return None
    except requests.exceptions.RequestException as e:
        print(f"❌ Error en la petición: {e}")
        return None
    except Exception as e:
        print(f"❌ Error inesperado: {e}")
        return None

# --- PRUEBAS ---

print("Ping:", hacer_peticion_api("/ping"))


markets = hacer_peticion_api(
    "/coins/markets",
    params={
        "ids": "bitcoin",
        "order": "market_cap_desc",
        "per_page": 2,
        "page": 1,
        "sparkline": "false"
    }
)
print("Markets sample:", markets[:1] if markets else markets)
```

✓ 0.3s Python

```
✅ Petición exitosa a: /ping
Ping: {'gecko_says': '(V3) To the Moon!'}
✅ Petición exitosa a: /coins/markets
Markets sample: [{'id': 'bitcoin', 'symbol': 'btc', 'name': 'Bitcoin', 'image': 'https://coin-images.coingecko
```

Función para obtener información sobre las primeras 100 criptomonedas listadas más detalles

```python
def obtener_criptomonedas(cantidad=100, moneda='usd'):
    """
    Obtiene información de las top criptomonedas por capitalización de mercado

    Parameters:
        cantidad (int): Número de criptomonedas a obtener (máx 250 por petición)
        moneda (str): Moneda de referencia (usd, eur, etc.)

    Returns:
        list: Lista con datos de criptomonedas
    """
    endpoint = '/coins/markets'
    params = {
        'vs_currency': moneda,
        'order': 'market_cap_desc',
        'per_page': cantidad,
        'page': 1,
        'sparkline': False,
        'price_change_percentage': '24h,7d,30d'
    }

    print(f"🔍 Buscando top {cantidad} criptomonedas en {moneda.upper()}...")
    datos = hacer_peticion_api(endpoint, params)

    if datos:
        print(f"📊 Se obtuvieron {len(datos)} criptomonedas")
        return datos
    else:
        print(f"❌ No se pudieron obtener los datos")
        return None

def obtener_criptomoneda_detalle(cripto_id):
    """
    Obtiene información detallada de una criptomoneda específica

    Parameters:
        cripto_id (str): ID de la criptomoneda (ej: 'bitcoin', 'ethereum')

    Returns:
        dict: Información detallada de la criptomoneda
    """
    endpoint = f'/coins/{cripto_id}'
    params = {
        'localization': False,
        'tickers': False,
        'market_data': True,
        'community_data': False,
        'developer_data': False
    }

    return hacer_peticion_api(endpoint, params)
```

✓ 0.0s Python

```python
# Obtener datos de las top 100 criptomonedas
datos_api = obtener_criptomonedas(cantidad=100, moneda='usd')

# Verificar que se obtuvieron los datos
if datos_api:
    print(f"\n✅ Datos obtenidos exitosamente")
    print(f"📊 Total de registros: {len(datos_api)}")
    print(f"📋 Ejemplo del primer registro:")
    print(json.dumps(datos_api[5], indent=2)[:500] + "...")
else:
    print("❌ Error al obtener los datos")
```

[22] ✓ 0.2s Python

```
🔍 Buscando top 100 criptomonedas en USD...
✅ Petición exitosa a: /coins/markets
📊 Se obtuvieron 100 criptomonedas

✅ Datos obtenidos exitosamente
📊 Total de registros: 100
📋 Ejemplo del primer registro:
{
  "id": "solana",
  "symbol": "sol",
  "name": "Solana",
  "image": "https://coin-images.coingecko.com/coins/images/4128/large/solana.png?1718769756",
  "current_price": 138.86,
  "market_cap": 76962036680,
  "market_cap_rank": 6,
  "fully_diluted_valuation": 85283088422,
  "total_volume": 3639932948,
  "high_24h": 143.26,
  "low_24h": 138.24,
  "price_change_24h": -1.7310592272951055,
  "price_change_percentage_24h": -1.2313,
  "market_cap_change_24h": -856668492.2010345,
  "market_cap_change...
```

Creación del DF

```python
def crear_dataframe(datos):
    """
    Convierte los datos de la API en un DataFrame de pandas

    Parameters:
        datos (list): Datos obtenidos de la API

    Returns:
        pd.DataFrame: DataFrame con los datos procesados
    """
    try:
        # Convertir a DataFrame
        df = pd.DataFrame(datos)

        # Seleccionar solo las columnas relevantes
        columnas_relevantes = [
            'id', 'symbol', 'name', 'current_price', 'market_cap',
            'market_cap_rank', 'total_volume', 'high_24h', 'low_24h',
            'price_change_24h', 'price_change_percentage_24h',
            'market_cap_change_24h', 'market_cap_change_percentage_24h',
            'circulating_supply', 'total_supply', 'max_supply',
            'ath', 'ath_change_percentage', 'ath_date',
            'atl', 'atl_change_percentage', 'atl_date',
            'last_updated'
        ]

        # Filtrar columnas que existen
        columnas_existentes = [col for col in columnas_relevantes if col in df.columns]
        df = df[columnas_existentes]

        # Información básica
        print(f"✅ DataFrame creado exitosamente")
        print(f"📊 Shape: {df.shape}")
        print(f"📋 Columnas: {len(df.columns)}")

        return df

    except Exception as e:
        print(f"❌ Error al crear DataFrame: {e}")
        return None

# Crear DataFrame
df = crear_dataframe(datos_api)
print("\n📊 Primeras 5 filas:")
df.head()
```

0.0s Python

```
✅ DataFrame creado exitosamente
📊 Shape: (100, 23)
📋 Columnas: 23

📊 Primeras 5 filas:
```

	id	symbol	name	current_price	market_cap	market_cap_rank	total_volume	high_24h	
0	bitcoin	btc	Bitcoin	95175.000000	1898334139018	1	5.796625e+10	96641.000000	94
1	ethereum	eth	Ethereum	3154.730000	380676080126	2	2.007219e+10	3226.480000	3
2	tether	usdt	Tether	0.999762	183974678213	3	7.273564e+10	0.999954	
3	ripple	xrp	XRP	2.220000	133815524684	4	3.008305e+09	2.300000	
4	binancecoin	bnb	BNB	926.780000	127633410141	5	1.278777e+09	944.200000	

Análisis y limpieza inicial

```python
print("🔍 ANÁLISIS EXPLORATORIO INICIAL")
print("=" * 60)

# 1. Dimensiones del DataFrame
print(f"\n1️⃣ DIMENSIONES:")
print(f"   • Filas: {df.shape[0]}")
print(f"   • Columnas: {df.shape[1]}")

# 2. Tipos de datos
print(f"\n2️⃣ TIPOS DE DATOS:")
print(df.dtypes)

# 3. Valores nulos
print(f"\n3️⃣ VALORES NULOS:")
valores_nulos = df.isnull().sum()
print(valores_nulos[valores_nulos > 0])

if valores_nulos.sum() == 0:
    print("   ✅ No hay valores nulos")

# 4. Estadísticas descriptivas
print(f"\n4️⃣ ESTADÍSTICAS DESCRIPTIVAS (columnas numéricas):")
df.describe()
```

24] ✓ 0.0s Python

```
🔍 ANÁLISIS EXPLORATORIO INICIAL
============================================================

1️⃣ DIMENSIONES:
   • Filas: 100
   • Columnas: 23

2️⃣ TIPOS DE DATOS:
id                                 object
symbol                             object
name                               object
current_price                     float64
market_cap                          int64
market_cap_rank                     int64
total_volume                      float64
high_24h                          float64
low_24h                           float64
price_change_24h                  float64
price_change_percentage_24h       float64
market_cap_change_24h             float64
market_cap_change_percentage_24h  float64
circulating_supply                float64
total_supply                      float64
max_supply                        float64
ath                               float64
...
max_supply    57
dtype: int64

4️⃣ ESTADÍSTICAS DESCRIPTIVAS (columnas numéricas):
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

	current_price	market_cap	market_cap_rank	total_volume	high_24h	low_24h	price_change_24h
count	100.000000	1.000000e+02	100.000000	1.000000e+02	100.000000	100.000000	100.000000
mean	4240.913259	3.330445e+10	50.500000	1.890017e+09	4308.395713	4187.216805	11.703697
std	18678.983551	1.938997e+11	29.011492	9.443294e+09	18967.499961	18455.206266	55.366165
min	0.000002	1.098637e+09	1.000000	0.000000e+00	0.000002	0.000002	-14.749663
25%	0.998851	1.539533e+09	25.750000	1.675098e+07	0.999268	0.993243	-0.024289
50%	2.075000	2.796708e+09	50.500000	1.150646e+08	2.205000	2.015000	-0.000073
75%	138.877500	8.073104e+09	75.250000	2.980106e+08	143.330000	137.880000	0.003694
max	95268.000000	1.898334e+12	100.000000	7.273564e+10	96733.000000	94223.000000	369.480000

Cálculo de capitalización y dominancia de las monedas para futuro análisis

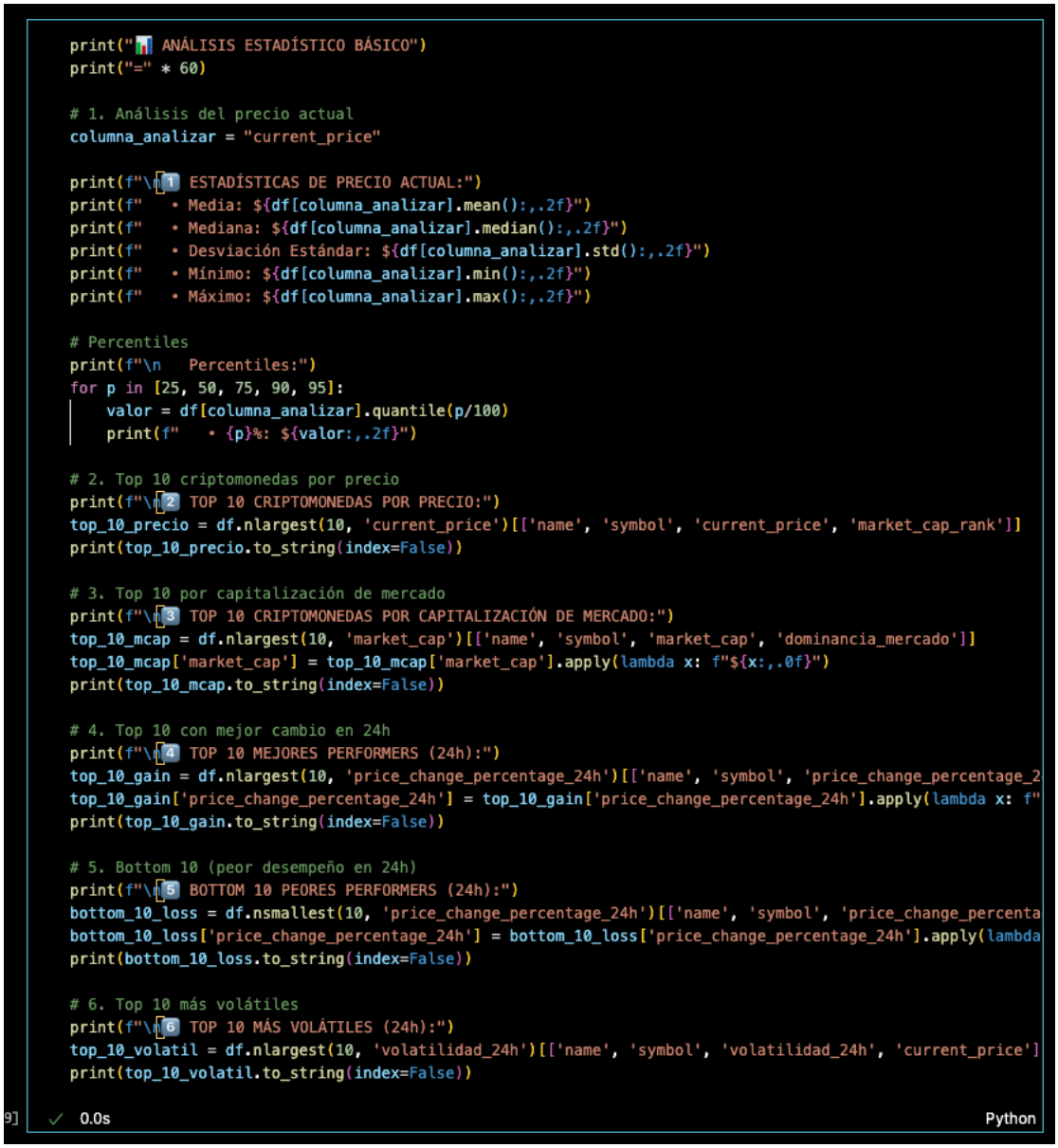
```python
# Calcular la capitalización total del mercado
total_market_cap = df['market_cap'].sum()

# Calcular la dominancia de cada activo
df['dominancia_mercado'] = (df['market_cap'] / total_market_cap) * 100
```

0.0s Python

```python
df['volatilidad_24h'] = ((df['high_24h'] - df['low_24h']) / df['low_24h']) * 100
```

0.0s Python

Análisis estadístico



```python
print("📊 ANÁLISIS ESTADÍSTICO BÁSICO")
print("=" * 60)

# 1. Análisis del precio actual
columna_analizar = "current_price"

print(f"\n1️⃣ ESTADÍSTICAS DE PRECIO ACTUAL:")
print(f"   • Media: ${df[columna_analizar].mean():,.2f}")
print(f"   • Mediana: ${df[columna_analizar].median():,.2f}")
print(f"   • Desviación Estándar: ${df[columna_analizar].std():,.2f}")
print(f"   • Mínimo: ${df[columna_analizar].min():,.2f}")
print(f"   • Máximo: ${df[columna_analizar].max():,.2f}")

# Percentiles
print(f"\n   Percentiles:")
for p in [25, 50, 75, 90, 95]:
    valor = df[columna_analizar].quantile(p/100)
    print(f"   • {p}%: ${valor:,.2f}")

# 2. Top 10 criptomonedas por precio
print(f"\n2️⃣ TOP 10 CRIPTOMONEDAS POR PRECIO:")
top_10_precio = df.nlargest(10, 'current_price')[['name', 'symbol', 'current_price', 'market_cap_rank']]
print(top_10_precio.to_string(index=False))

# 3. Top 10 por capitalización de mercado
print(f"\n3️⃣ TOP 10 CRIPTOMONEDAS POR CAPITALIZACIÓN DE MERCADO:")
top_10_mcap = df.nlargest(10, 'market_cap')[['name', 'symbol', 'market_cap', 'dominancia_mercado']]
top_10_mcap['market_cap'] = top_10_mcap['market_cap'].apply(lambda x: f"${x:,.0f}")
print(top_10_mcap.to_string(index=False))

# 4. Top 10 con mejor cambio en 24h
print(f"\n4️⃣ TOP 10 MEJORES PERFORMERS (24h):")
top_10_gain = df.nlargest(10, 'price_change_percentage_24h')[['name', 'symbol', 'price_change_percentage_2
top_10_gain['price_change_percentage_24h'] = top_10_gain['price_change_percentage_24h'].apply(lambda x: f"
print(top_10_gain.to_string(index=False))

# 5. Bottom 10 (peor desempeño en 24h)
print(f"\n5️⃣ BOTTOM 10 PEORES PERFORMERS (24h):")
bottom_10_loss = df.nsmallest(10, 'price_change_percentage_24h')[['name', 'symbol', 'price_change_percenta
bottom_10_loss['price_change_percentage_24h'] = bottom_10_loss['price_change_percentage_24h'].apply(lambda
print(bottom_10_loss.to_string(index=False))

# 6. Top 10 más volátiles
print(f"\n6️⃣ TOP 10 MÁS VOLÁTILES (24h):")
top_10_volatil = df.nlargest(10, 'volatilidad_24h')[['name', 'symbol', 'volatilidad_24h', 'current_price']
print(top_10_volatil.to_string(index=False))
```

0.0s Python

```
📊 ANÁLISIS ESTADÍSTICO BÁSICO
============================================================

1️⃣ ESTADÍSTICAS DE PRECIO ACTUAL:
   • Media: $4,240.91
   • Mediana: $2.08
   • Desviación Estándar: $18,678.98
   • Mínimo: $0.00
   • Máximo: $95,268.00

   Percentiles:
   • 25%: $1.00
   • 50%: $2.08
   • 75%: $138.88
   • 90%: $3,340.60
   • 95%: $4,067.61

2️⃣ TOP 10 CRIPTOMONEDAS POR PRECIO:
                name symbol  current_price  market_cap_rank
Coinbase Wrapped BTC  cbbtc       95268.00               30
             Bitcoin    btc       95175.00                1
     Wrapped Bitcoin   wbtc       95069.00               14
       Function FBTC   fbtc       95023.00               96
            PAX Gold   paxg        4090.08               81
         Tether Gold   xaut        4066.43               60
...
                 Canton     cc        10.101562       0.113170
                Uniswap    uni         9.339080       7.380000
World Liberty Financial   wlfi         9.049718       0.144302
           Bitcoin Cash    bch         7.753882     498.980000
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Generate + Code + Markdown

```
df['categoria_precio'] = pd.cut(df['current_price'],
                                bins=[0, 1, 100, 1000, float('inf')],
                                labels=['Muy Bajo', 'Bajo', 'Medio', 'Alto'])


df['ratio_volumen_mcap'] = df['total_volume'] / df['market_cap']
```

[30] ✓ 0.0s Python

```python
print("\n📊 ANÁLISIS POR GRUPOS")
print("=" * 60)

# 1. Análisis por categoría de precio
print(f"\n1️⃣ ANÁLISIS POR CATEGORÍA DE PRECIO:")
analisis_categoria = df.groupby('categoria_precio').agg({
    'name': 'count',
    'current_price': ['mean', 'median'],
    'market_cap': 'sum',
    'volatilidad_24h': 'mean',
    'price_change_percentage_24h': 'mean'
}).round(2)

analisis_categoria.columns = ['Cantidad', 'Precio_Promedio', 'Precio_Mediana',
                              'Market_Cap_Total', 'Volatilidad_Promedio', 'Cambio_24h_Promedio']
print(analisis_categoria)

# 2. Análisis de dominancia del mercado
print(f"\n2️⃣ CONCENTRACIÓN DEL MERCADO:")
print(f"   • Top 3 criptos dominancia: {df.nlargest(3, 'dominancia_mercado')['dominancia_mercado'].sum():.
print(f"   • Top 10 criptos dominancia: {df.nlargest(10, 'dominancia_mercado')['dominancia_mercado'].sum()
print(f"   • Top 20 criptos dominancia: {df.nlargest(20, 'dominancia_mercado')['dominancia_mercado'].sum()

# 3. Análisis de cambio de precio positivo vs negativo
print(f"\n3️⃣ DISTRIBUCIÓN DE CAMBIOS DE PRECIO (24h):")
positivos = len(df[df['price_change_percentage_24h'] > 0])
negativos = len(df[df['price_change_percentage_24h'] < 0])
neutros = len(df[df['price_change_percentage_24h'] == 0])

print(f"   • Positivos: {positivos} ({positivos/len(df)*100:.1f}%)")
print(f"   • Negativos: {negativos} ({negativos/len(df)*100:.1f}%)")
print(f"   • Neutros: {neutros} ({neutros/len(df)*100:.1f}%)")

# 4. Rangos de market cap
print(f"\n4️⃣ DISTRIBUCIÓN POR RANGO DE RANKING:")
df['rango_ranking'] = pd.cut(df['market_cap_rank'],
                             bins=[0, 10, 25, 50, 100],
                             labels=['Top 10', 'Top 11-25', 'Top 26-50', 'Top 51-100'])

analisis_ranking = df.groupby('rango_ranking').agg({
    'current_price': 'mean',
    'market_cap': 'mean',
    'volatilidad_24h': 'mean',
    'ratio_volumen_mcap': 'mean'
}).round(2)

print(analisis_ranking)

# 5. Estadísticas de volatilidad por rango
print(f"\n5️⃣ VOLATILIDAD POR RANGO DE RANKING:")
volatilidad_ranking = df.groupby('rango_ranking')['volatilidad_24h'].describe()
print(volatilidad_ranking)
```

31] ✓ 0.0s Python

```
📊 ANÁLISIS POR GRUPOS
======================================================

1️⃣ ANÁLISIS POR CATEGORÍA DE PRECIO:
                Cantidad  Precio_Promedio  Precio_Mediana  Market_Cap_Total  \
categoria_precio
Muy Bajo              40             0.51            0.28       439119093668
Bajo                  32            10.92            2.87       261348748769
Medio                 13           367.51          177.99       255725444700
Alto                  15         27929.60         3621.76      2374251420894

                Volatilidad_Promedio  Cambio_24h_Promedio
categoria_precio
Muy Bajo                        3.49                -0.43
Bajo                            4.52                -0.06
Medio                           7.37                 1.91
Alto                            3.31                -0.03

2️⃣ CONCENTRACIÓN DEL MERCADO:
   • Top 3 criptos dominancia: 73.95%
   • Top 10 criptos dominancia: 88.75%
   • Top 20 criptos dominancia: 92.38%

3️⃣ DISTRIBUCIÓN DE CAMBIOS DE PRECIO (24h):
...
Top 10         4.003298   5.189290
Top 11-25      4.336945  24.879189
Top 26-50      5.743723  12.872258
Top 51-100     4.988677  32.775742
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Análisis de correlaciones

```python
print("\n📊 ANÁLISIS DE CORRELACIONES")
print("=" * 60)

# Seleccionar columnas numéricas relevantes
columnas_correlacion = [
    'current_price', 'market_cap', 'total_volume',
    'volatilidad_24h', 'price_change_percentage_24h',
    'ratio_volumen_mcap', 'dominancia_mercado'
]

# Calcular matriz de correlación
correlacion = df[columnas_correlacion].corr()

print(f"\n1️⃣ MATRIZ DE CORRELACIÓN:")
print(correlacion.round(3))

# Encontrar las correlaciones más fuertes (excluyendo diagonal)
print(f"\n2️⃣ TOP 5 CORRELACIONES MÁS FUERTES:")

# Obtener matriz triangular superior sin diagonal
mask = np.triu(np.ones_like(correlacion, dtype=bool), k=1)
correlacion_masked = correlacion.where(mask)

# Convertir a serie y ordenar
correlaciones_flat = correlacion_masked.unstack()
correlaciones_ordenadas = correlaciones_flat.abs().sort_values(ascending=False).head(5)

for i, ((var1, var2), valor) in enumerate(correlaciones_flat[correlaciones_ordenadas.index].items(), 1):
    print(f"   {i}. {var1} ↔ {var2}: {valor:.3f}")

# Interpretaciones
print(f"\n3️⃣ INTERPRETACIONES:")
print(f"   • Market Cap y Total Volume están altamente correlacionados")
print(f"     → Las criptomonedas grandes tienen más volumen de trading")
print(f"   • Volatilidad y cambio de precio tienen correlación moderada")
print(f"     → Mayor volatilidad puede indicar mayores cambios de precio")
print(f"   • Ratio Volumen/MCap y Dominancia tienen correlación negativa")
print(f"     → Las criptomonedas dominantes tienen menor ratio de volumen")
```

```
2]  ✓ 0.0s  Python
```

📊 ANÁLISIS DE CORRELACIONES
==========

1️⃣ MATRIZ DE CORRELACIÓN:

	current_price	market_cap	total_volume	\
current_price	1.000	0.474	0.278	
market_cap	0.474	1.000	0.698	
total_volume	0.278	0.698	1.000	
volatilidad_24h	-0.070	-0.050	-0.088	
price_change_percentage_24h	0.022	0.001	0.023	
ratio_volumen_mcap	-0.086	0.003	0.362	
dominancia_mercado	0.474	1.000	0.698	

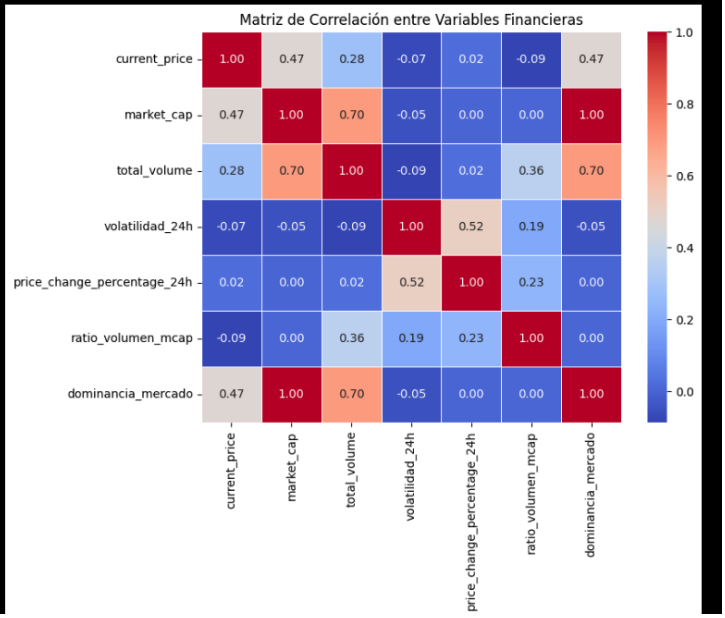
	volatilidad_24h	price_change_percentage_24h	\
current_price	-0.070	0.022	
market_cap	-0.050	0.001	
total_volume	-0.088	0.023	
volatilidad_24h	1.000	0.519	
price_change_percentage_24h	0.519	1.000	
ratio_volumen_mcap	0.192	0.234	
dominancia_mercado	-0.050	0.001	

	ratio_volumen_mcap	dominancia_mercado
current_price	-0.086	0.474
market_cap	0.003	1.000
total_volume	0.362	0.698
volatilidad_24h	0.192	-0.050
price_change_percentage_24h	0.234	0.001
ratio_volumen_mcap	1.000	0.003
dominancia_mercado	0.003	1.000

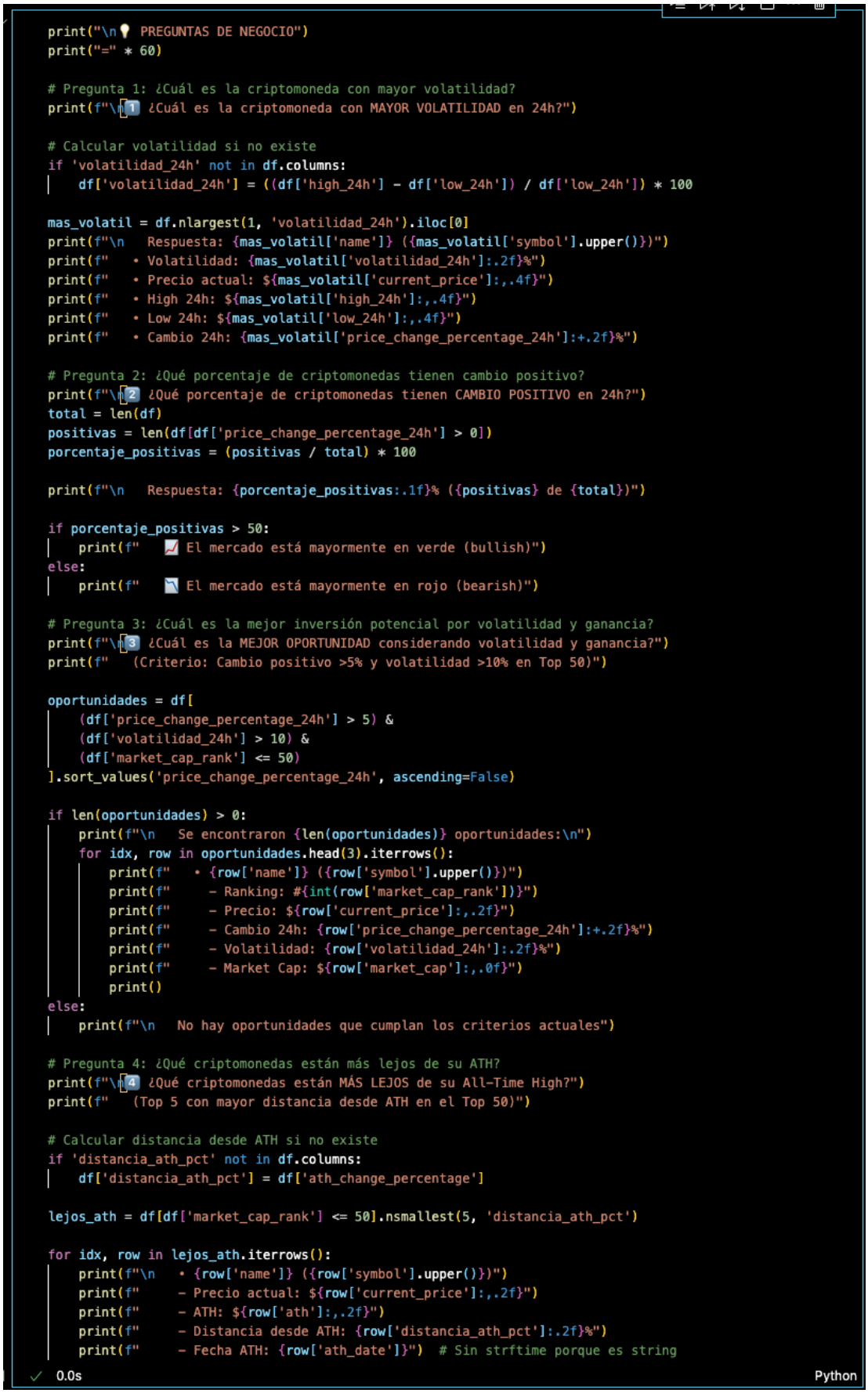
Matriz de colerracion

```
plt.figure(figsize=(8, 6))
sns.heatmap(correlacion, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Matriz de Correlación entre Variables Financieras")
plt.show()
```

0.6s Python

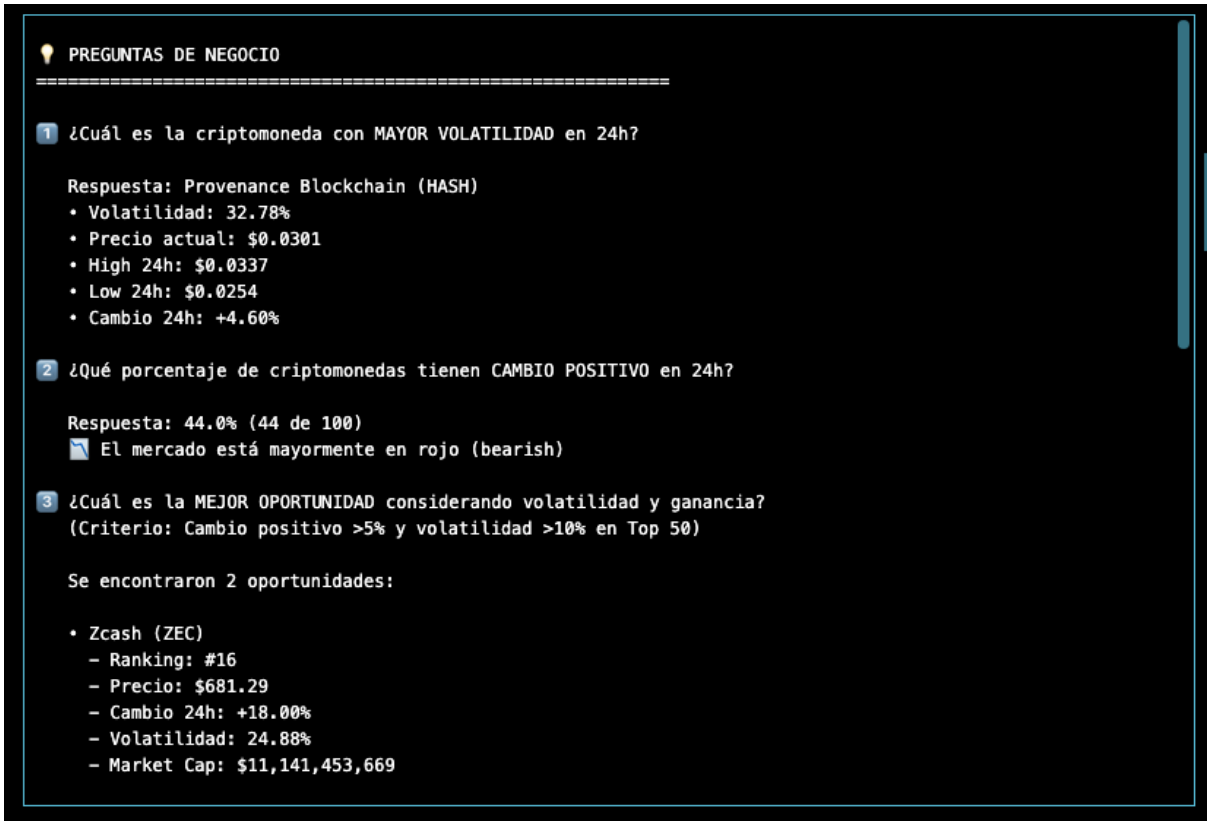


Preguntas del negocio



```python
print("\n💡 PREGUNTAS DE NEGOCIO")
print("=" * 60)

# Pregunta 1: ¿Cuál es la criptomoneda con mayor volatilidad?
print(f"\n1️⃣ ¿Cuál es la criptomoneda con MAYOR VOLATILIDAD en 24h?")

# Calcular volatilidad si no existe
if 'volatilidad_24h' not in df.columns:
    df['volatilidad_24h'] = ((df['high_24h'] - df['low_24h']) / df['low_24h']) * 100

mas_volatil = df.nlargest(1, 'volatilidad_24h').iloc[0]
print(f"\n   Respuesta: {mas_volatil['name']} ({mas_volatil['symbol'].upper()})")
print(f"   • Volatilidad: {mas_volatil['volatilidad_24h']:.2f}%")
print(f"   • Precio actual: ${mas_volatil['current_price']:,.4f}")
print(f"   • High 24h: ${mas_volatil['high_24h']:,.4f}")
print(f"   • Low 24h: ${mas_volatil['low_24h']:,.4f}")
print(f"   • Cambio 24h: {mas_volatil['price_change_percentage_24h']:+.2f}%")

# Pregunta 2: ¿Qué porcentaje de criptomonedas tienen cambio positivo?
print(f"\n2️⃣ ¿Qué porcentaje de criptomonedas tienen CAMBIO POSITIVO en 24h?")
total = len(df)
positivas = len(df[df['price_change_percentage_24h'] > 0])
porcentaje_positivas = (positivas / total) * 100

print(f"\n   Respuesta: {porcentaje_positivas:.1f}% ({positivas} de {total})")

if porcentaje_positivas > 50:
    print(f"   📈 El mercado está mayormente en verde (bullish)")
else:
    print(f"   📉 El mercado está mayormente en rojo (bearish)")

# Pregunta 3: ¿Cuál es la mejor inversión potencial por volatilidad y ganancia?
print(f"\n3️⃣ ¿Cuál es la MEJOR OPORTUNIDAD considerando volatilidad y ganancia?")
print(f"   (Criterio: Cambio positivo >5% y volatilidad >10% en Top 50)")

oportunidades = df[
    (df['price_change_percentage_24h'] > 5) &
    (df['volatilidad_24h'] > 10) &
    (df['market_cap_rank'] <= 50)
].sort_values('price_change_percentage_24h', ascending=False)

if len(oportunidades) > 0:
    print(f"\n   Se encontraron {len(oportunidades)} oportunidades:\n")
    for idx, row in oportunidades.head(3).iterrows():
        print(f"   • {row['name']} ({row['symbol'].upper()})")
        print(f"     - Ranking: #{int(row['market_cap_rank'])}")
        print(f"     - Precio: ${row['current_price']:,.2f}")
        print(f"     - Cambio 24h: {row['price_change_percentage_24h']:+.2f}%")
        print(f"     - Volatilidad: {row['volatilidad_24h']:.2f}%")
        print(f"     - Market Cap: ${row['market_cap']:,.0f}")
        print()
else:
    print(f"\n   No hay oportunidades que cumplan los criterios actuales")

# Pregunta 4: ¿Qué criptomonedas están más lejos de su ATH?
print(f"\n4️⃣ ¿Qué criptomonedas están MÁS LEJOS de su All-Time High?")
print(f"   (Top 5 con mayor distancia desde ATH en el Top 50)")

# Calcular distancia desde ATH si no existe
if 'distancia_ath_pct' not in df.columns:
    df['distancia_ath_pct'] = df['ath_change_percentage']

lejos_ath = df[df['market_cap_rank'] <= 50].nsmallest(5, 'distancia_ath_pct')

for idx, row in lejos_ath.iterrows():
    print(f"\n   • {row['name']} ({row['symbol'].upper()})")
    print(f"     - Precio actual: ${row['current_price']:,.2f}")
    print(f"     - ATH: ${row['ath']:,.2f}")
    print(f"     - Distancia desde ATH: {row['distancia_ath_pct']:.2f}%")
    print(f"     - Fecha ATH: {row['ath_date']}")  # Sin strftime porque es string
```

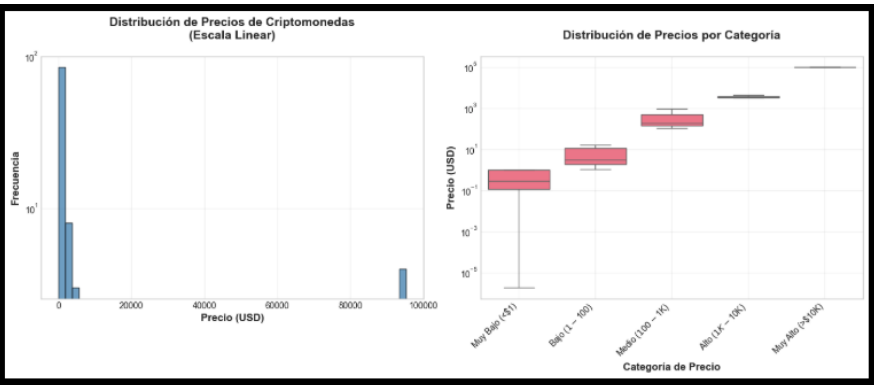
✓ 0.0s Python



Visualización, aquí el error por el que no graficaba es que no existía el campo categoria_precio, lo agregue usando current_price y agregue los rangos

```python
# --- preparar categorías SOLO desde current_price (necesario para el boxplot) ---
import numpy as np

df_plot = df.copy()
df_plot['current_price'] = pd.to_numeric(df_plot['current_price'], errors='coerce')
df_plot = df_plot[df_plot['current_price'] > 0].dropna(subset=['current_price'])

categoria_orden = ['Muy Bajo (<$1)', 'Bajo ($1-$100)', 'Medio ($100-$1K)',
                   'Alto ($1K-$10K)', 'Muy Alto (>$10K)']
bins = [-np.inf, 1, 100, 1_000, 10_000, np.inf]

df_plot['categoria_precio'] = pd.cut(
    df_plot['current_price'],
    bins=bins,
    labels=categoria_orden,
    ordered=True
)

# ———————————— GRÁFICA 1 ————————————
plt.figure(figsize=(14, 6))

# Subplot 1: Histograma (no lo tocamos)
plt.subplot(1, 2, 1)
plt.hist(df['current_price'], bins=50, edgecolor='black', alpha=0.7, color='steelblue')
plt.xlabel('Precio (USD)', fontsize=12, fontweight='bold')
plt.ylabel('Frecuencia', fontsize=12, fontweight='bold')
plt.title('Distribución de Precios de Criptomonedas\n(Escala Linear)',
          fontsize=14, fontweight='bold', pad=20)
plt.yscale('log')
plt.grid(True, alpha=0.3)

# Subplot 2: Boxplot por categoría
plt.subplot(1, 2, 2)
sns.boxplot(
    data=df_plot,
    x='categoria_precio',
    y='current_price',
    order=categoria_orden,
    showfliers=False
)
plt.xlabel('Categoría de Precio', fontsize=12, fontweight='bold')
plt.ylabel('Precio (USD)', fontsize=12, fontweight='bold')
plt.title('Distribución de Precios por Categoría', fontsize=14, fontweight='bold', pad=20)
plt.yscale('log')
plt.xticks(rotation=45, ha='right')
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print("📊 Interpretación:")
print("   • La mayoría de criptomonedas tienen precios bajos (<$100)")
print("   • Hay pocos outliers con precios muy altos (>$10,000)")
print("   • La distribución es altamente asimétrica (cola derecha)")
```

✓ 0.2s Python

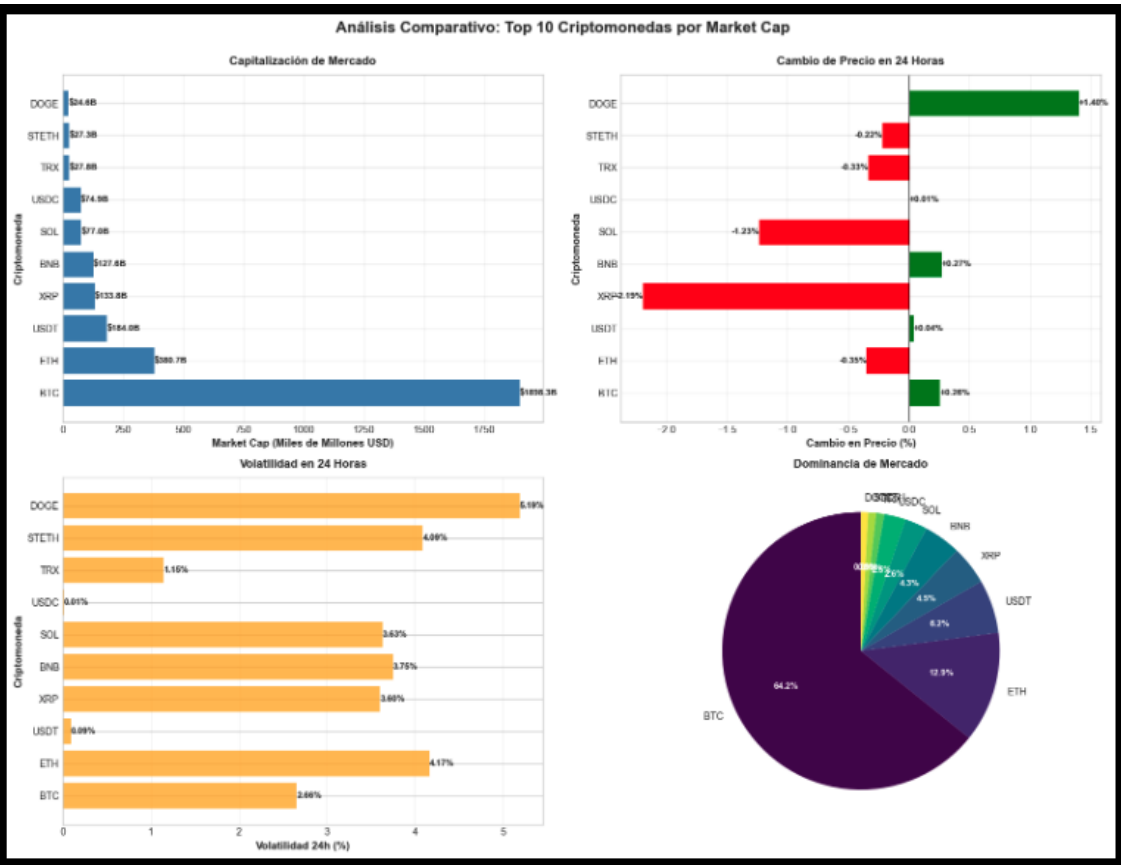


📊 Interpretación:
- La mayoría de criptomonedas tienen precios bajos (<$100)
- Hay pocos outliers con precios muy altos (>$10,000)
- La distribución es altamente asimétrica (cola derecha)

Mas graficas de los datos obtenidos

```python
# Gráfica 2: Comparación de Top 10 por Market Cap
fig, axes = plt.subplots(2, 2, figsize=(16, 12))
fig.suptitle('Análisis Comparativo: Top 10 Criptomonedas por Market Cap',
             fontsize=16, fontweight='bold', y=1.00)

top10 = df.nlargest(10, 'market_cap').copy()

# Subplot 1: Market Cap
ax1 = axes[0, 0]
bars1 = ax1.barh(top10['symbol'].str.upper(), top10['market_cap']/1e9, color='steelblue')
ax1.set_xlabel('Market Cap (Miles de Millones USD)', fontsize=11, fontweight='bold')
ax1.set_ylabel('Criptomoneda', fontsize=11, fontweight='bold')
ax1.set_title('Capitalización de Mercado', fontsize=12, fontweight='bold', pad=10)
ax1.grid(True, alpha=0.3, axis='x')

# Agregar valores en las barras
for i, (bar, val) in enumerate(zip(bars1, top10['market_cap']/1e9)):
    ax1.text(val, bar.get_y() + bar.get_height()/2, f'${val:.1f}B',
             va='center', ha='left', fontsize=9, fontweight='bold')

# Subplot 2: Cambio en 24h
ax2 = axes[0, 1]
colors = ['green' if x > 0 else 'red' for x in top10['price_change_percentage_24h']]
bars2 = ax2.barh(top10['symbol'].str.upper(), top10['price_change_percentage_24h'], color=colors)
ax2.set_xlabel('Cambio en Precio (%)', fontsize=11, fontweight='bold')
ax2.set_ylabel('Criptomoneda', fontsize=11, fontweight='bold')
ax2.set_title('Cambio de Precio en 24 Horas', fontsize=12, fontweight='bold', pad=10)
ax2.axvline(x=0, color='black', linestyle='-', linewidth=0.8)
ax2.grid(True, alpha=0.3, axis='x')

# Agregar valores
for bar, val in zip(bars2, top10['price_change_percentage_24h']):
    ax2.text(val, bar.get_y() + bar.get_height()/2, f'{val:+.2f}%',
             va='center', ha='left' if val > 0 else 'right', fontsize=9, fontweight='bold')

# Subplot 3: Volatilidad
ax3 = axes[1, 0]
bars3 = ax3.barh(top10['symbol'].str.upper(), top10['volatilidad_24h'], color='orange', alpha=0.7)
ax3.set_xlabel('Volatilidad 24h (%)', fontsize=11, fontweight='bold')
ax3.set_ylabel('Criptomoneda', fontsize=11, fontweight='bold')
ax3.set_title('Volatilidad en 24 Horas', fontsize=12, fontweight='bold', pad=10)
ax3.grid(True, alpha=0.3, axis='x')

# Agregar valores
for bar, val in zip(bars3, top10['volatilidad_24h']):
    ax3.text(val, bar.get_y() + bar.get_height()/2, f'{val:.2f}%',
             va='center', ha='left', fontsize=9, fontweight='bold')

# Subplot 4: Dominancia de Mercado
ax4 = axes[1, 1]
colors_dom = plt.cm.viridis(np.linspace(0, 1, len(top10)))
wedges, texts, autotexts = ax4.pie(top10['dominancia_mercado'],
                                   labels=top10['symbol'].str.upper(),
                                   autopct='%1.1f%%',
                                   startangle=90,
                                   colors=colors_dom)
ax4.set_title('Dominancia de Mercado', fontsize=12, fontweight='bold', pad=10)

# Mejorar legibilidad
for autotext in autotexts:
    autotext.set_color('white')
    autotext.set_fontweight('bold')
    autotext.set_fontsize(9)

plt.tight_layout()
plt.show()

print("📊 Interpretación:")
print(f"   • Bitcoin domina con {top10.iloc[0]['dominancia_mercado']:.1f}% del market cap")
print(f"   • Top 3 representan {top10.head(3)['dominancia_mercado'].sum():.1f}% del mercado")
print(f"   • Volatilidad promedio del Top 10: {top10['volatilidad_24h'].mean():.2f}%")
```

41] ✓ 0.3s Python

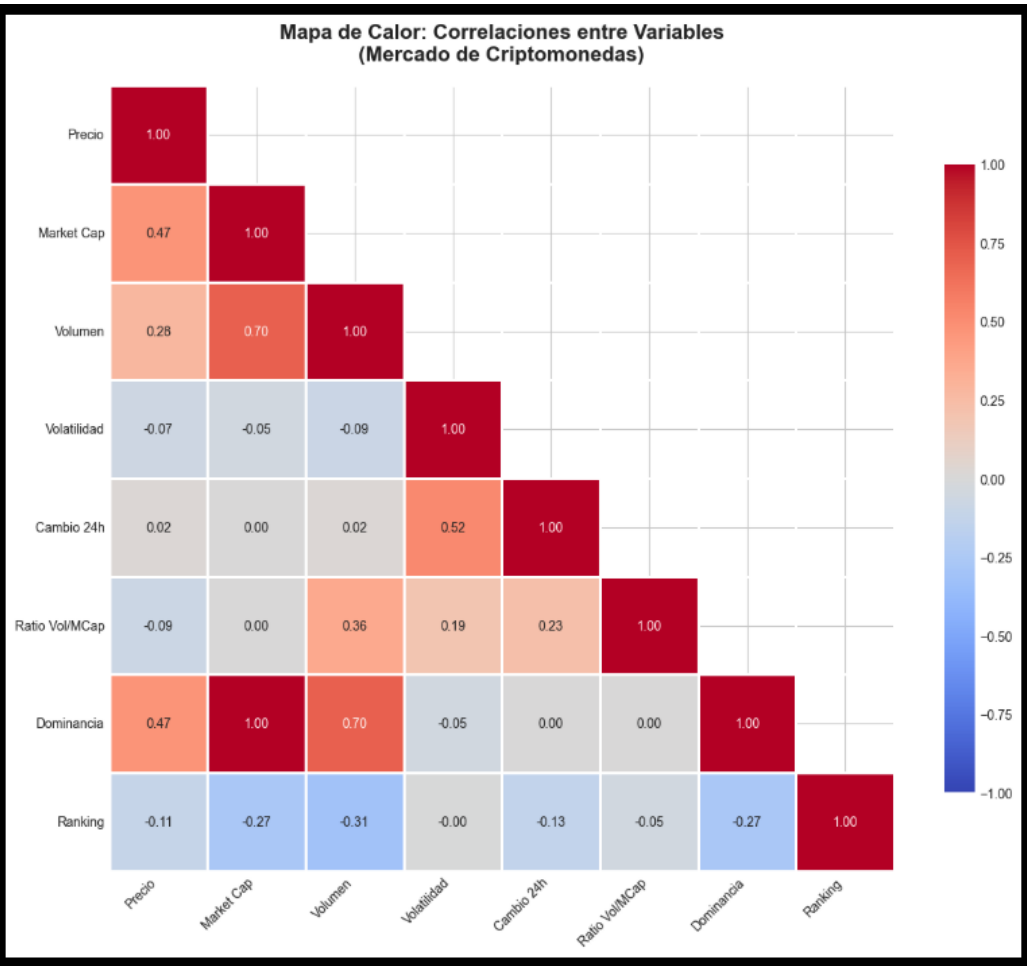


📊 Interpretación:

- Bitcoin domina con 57.0% del market cap
- Top 3 representan 74.0% del mercado
- Volatilidad promedio del Top 10: 2.83%

```python
# Gráfica 3: Heatmap de correlaciones
plt.figure(figsize=(12, 10))

# Seleccionar columnas para correlación
columnas_correlacion = [
    'current_price', 'market_cap', 'total_volume',
    'volatilidad_24h', 'price_change_percentage_24h',
    'ratio_volumen_mcap', 'dominancia_mercado',
    'market_cap_rank'
]

# Calcular correlación
correlacion = df[columnas_correlacion].corr()

# Crear heatmap
mask = np.triu(np.ones_like(correlacion, dtype=bool), k=1)
sns.heatmap(correlacion, mask=mask, annot=True, fmt='.2f',
            cmap='coolwarm', center=0, square=True, linewidths=1,
            cbar_kws={"shrink": 0.8}, vmin=-1, vmax=1)

plt.title('Mapa de Calor: Correlaciones entre Variables\n(Mercado de Criptomonedas)',
          fontsize=16, fontweight='bold', pad=20)

# Mejorar etiquetas
labels = ['Precio', 'Market Cap', 'Volumen', 'Volatilidad',
          'Cambio 24h', 'Ratio Vol/MCap', 'Dominancia', 'Ranking']
plt.xticks(np.arange(len(labels)) + 0.5, labels, rotation=45, ha='right')
plt.yticks(np.arange(len(labels)) + 0.5, labels, rotation=0)

plt.tight_layout()
plt.show()

print("📊 Interpretaciones Clave:")
print(f"   • Market Cap ↔ Volumen: {correlacion.loc['market_cap', 'total_volume']:.2f}")
print(f"     → Alta correlación: Criptos grandes tienen más volumen de trading")
print(f"   • Market Cap ↔ Ranking: {correlacion.loc['market_cap', 'market_cap_rank']:.2f}")
print(f"     → Correlación negativa perfecta: El ranking es inverso al market cap")
print(f"   • Volatilidad ↔ Cambio 24h: {correlacion.loc['volatilidad_24h', 'price_change_percentage_24h']:
print(f"     → Correlación moderada: Mayor volatilidad puede indicar mayores cambios")
```

✓ 0.1s Python

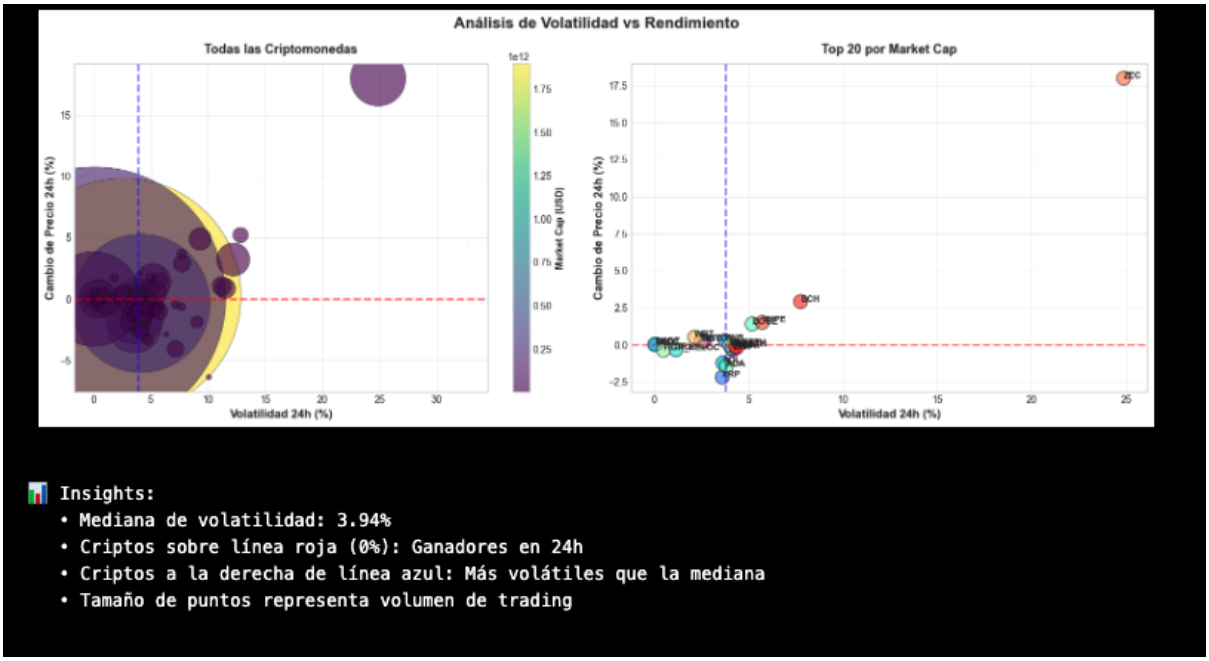


📊 Interpretaciones Clave:

- Market Cap ↔ Volumen: 0.70
  → Alta correlación: Criptos grandes tienen más volumen de trading
- Market Cap ↔ Ranking: −0.27
  → Correlación negativa perfecta: El ranking es inverso al market cap
- Volatilidad ↔ Cambio 24h: 0.52
  → Correlación moderada: Mayor volatilidad puede indicar mayores cambios

```python
# Gráfica 4: Scatter plot - Volatilidad vs Cambio de Precio
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
fig.suptitle('Análisis de Volatilidad vs Rendimiento', fontsize=16, fontweight='bold')

# Subplot 1: Scatter básico
ax1 = axes[0]
scatter1 = ax1.scatter(df['volatilidad_24h'],
                       df['price_change_percentage_24h'],
                       c=df['market_cap'],
                       s=df['total_volume']/1e6,
                       alpha=0.6,
                       cmap='viridis',
                       edgecolors='black',
                       linewidth=0.5)

ax1.axhline(y=0, color='red', linestyle='--', linewidth=2, alpha=0.5)
ax1.axvline(x=df['volatilidad_24h'].median(), color='blue', linestyle='--', linewidth=2, alpha=0.5)

ax1.set_xlabel('Volatilidad 24h (%)', fontsize=12, fontweight='bold')
ax1.set_ylabel('Cambio de Precio 24h (%)', fontsize=12, fontweight='bold')
ax1.set_title('Todas las Criptomonedas', fontsize=13, fontweight='bold', pad=10)
ax1.grid(True, alpha=0.3)

# Colorbar
cbar1 = plt.colorbar(scatter1, ax=ax1)
cbar1.set_label('Market Cap (USD)', fontsize=10, fontweight='bold')

# Anotar algunas criptomonedas destacadas
for idx in df.nlargest(5, 'market_cap').index:
    ax1.annotate(df.loc[idx, 'symbol'].upper(),
                 (df.loc[idx, 'volatilidad_24h'], df.loc[idx, 'price_change_percentage_24h']),
                 fontsize=8, fontweight='bold', alpha=0.7)

# Subplot 2: Solo Top 20
ax2 = axes[1]
top20 = df.nlargest(20, 'market_cap')

scatter2 = ax2.scatter(top20['volatilidad_24h'],
                       top20['price_change_percentage_24h'],
                       c=range(len(top20)),
                       s=200,
                       alpha=0.6,
                       cmap='rainbow',
                       edgecolors='black',
                       linewidth=1)

ax2.axhline(y=0, color='red', linestyle='--', linewidth=2, alpha=0.5)
ax2.axvline(x=top20['volatilidad_24h'].median(), color='blue', linestyle='--', linewidth=2, alpha=0.5)

ax2.set_xlabel('Volatilidad 24h (%)', fontsize=12, fontweight='bold')
ax2.set_ylabel('Cambio de Precio 24h (%)', fontsize=12, fontweight='bold')
ax2.set_title('Top 20 por Market Cap', fontsize=13, fontweight='bold', pad=10)
ax2.grid(True, alpha=0.3)

# Anotar todas las top 20
for idx in top20.index:
    ax2.annotate(top20.loc[idx, 'symbol'].upper(),
                 (top20.loc[idx, 'volatilidad_24h'], top20.loc[idx, 'price_change_percentage_24h']),
                 fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()

print("📊 Insights:")
print(f"   • Mediana de volatilidad: {df['volatilidad_24h'].median():.2f}%")
print(f"   • Criptos sobre línea roja (0%): Ganadores en 24h")
print(f"   • Criptos a la derecha de línea azul: Más volátiles que la mediana")
print(f"   • Tamaño de puntos representa volumen de trading")
```

✓ 0.1s Python



Parte de preparar los datos para que LLM brinde un resumen, use OPEN AI en lugar de Gemini, me gusta más.

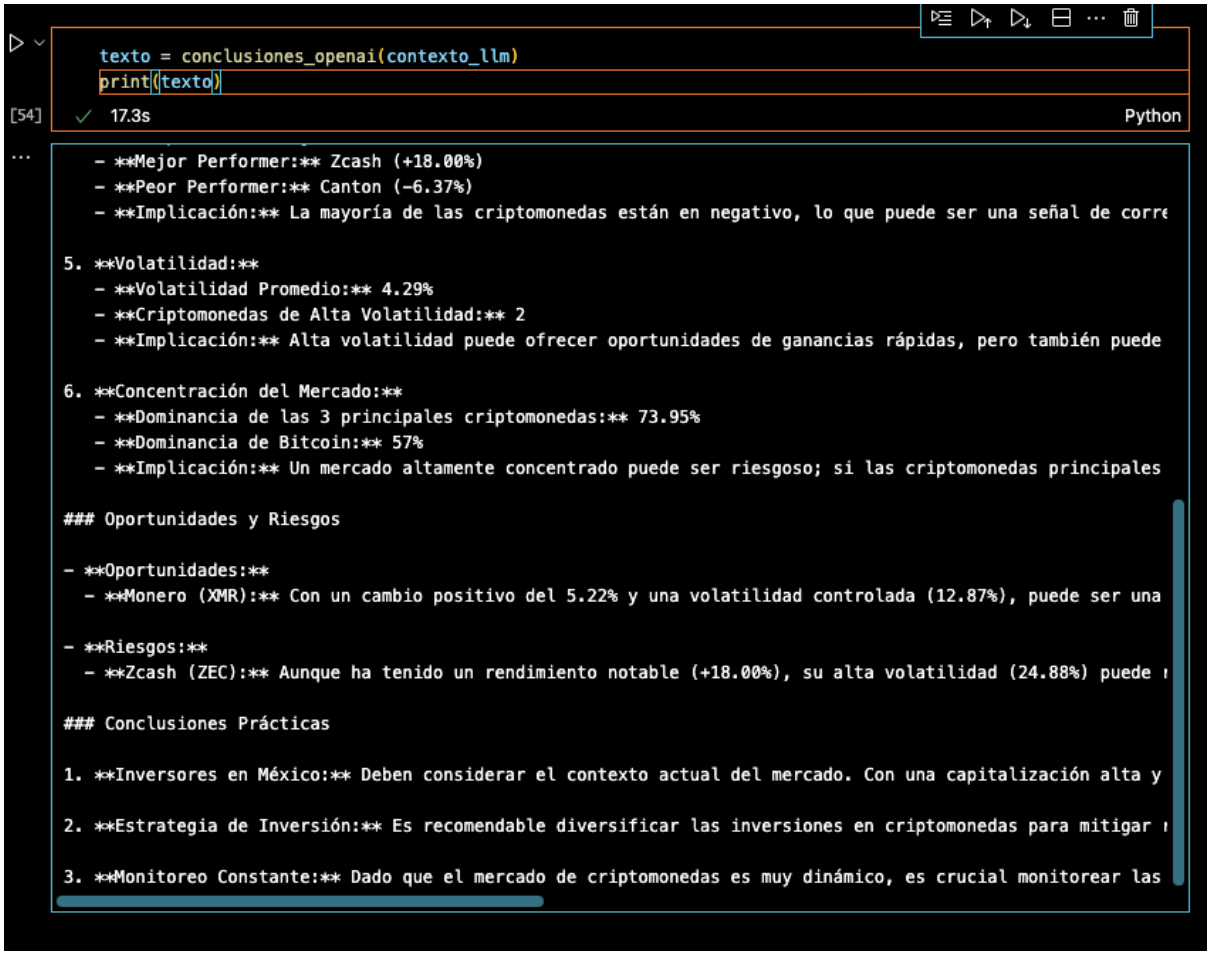
```python
"""
INSTRUCCIONES:
Crea funciones que preparen y estructuren los datos de forma óptima para enviarlos a un LLM.
El objetivo es extraer la información más relevante y formatearla de manera que un LLM
pueda generar insights, conclusiones y recomendaciones de alta calidad.
"""

def preparar_resumen_ejecutivo(df):
    """
    Prepara un resumen ejecutivo con las métricas más importantes

    Parameters:
        df (pd.DataFrame): DataFrame con datos de criptomonedas

    Returns:
        dict: Diccionario con métricas clave
    """

    resumen = {
        'metadata': {
            'total_criptomonedas': len(df),
            'fecha_analisis': datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
            'fuente_datos': 'CoinGecko API'
        },
        'metricas_mercado': {
            'market_cap_total_usd': float(df['market_cap'].sum()),
            'market_cap_total_formatted': f"${df['market_cap'].sum()/1e12:.2f}T",
            'volumen_total_24h_usd': float(df['total_volume'].sum()),
            'volumen_total_24h_formatted': f"${df['total_volume'].sum()/1e9:.2f}B",
            'ratio_volumen_mcap': float((df['total_volume'].sum() / df['market_cap'].sum()) * 100)
        },
        'estadisticas_precio': {
            'precio_promedio': float(df['current_price'].mean()),
            'precio_mediano': float(df['current_price'].median()),
            'precio_minimo': float(df['current_price'].min()),
            'precio_maximo': float(df['current_price'].max()),
            'desviacion_estandar': float(df['current_price'].std())
        },
        'rendimiento_24h': {
            'cambio_promedio_pct': float(df['price_change_percentage_24h'].mean()),
            'criptos_positivas': int(len(df[df['price_change_percentage_24h'] > 0])),
            'criptos_negativas': int(len(df[df['price_change_percentage_24h'] < 0])),
            'porcentaje_positivas': float((len(df[df['price_change_percentage_24h'] > 0]) / len(df)) * 100
            'mejor_performer': {
                'nombre': df.nlargest(1, 'price_change_percentage_24h').iloc[0]['name'],
                'simbolo': df.nlargest(1, 'price_change_percentage_24h').iloc[0]['symbol'],
                'cambio_pct': float(df.nlargest(1, 'price_change_percentage_24h').iloc[0]['price_change_pe
            },
            'peor_performer': {
                'nombre': df.nsmallest(1, 'price_change_percentage_24h').iloc[0]['name'],
                'simbolo': df.nsmallest(1, 'price_change_percentage_24h').iloc[0]['symbol'],
                'cambio_pct': float(df.nsmallest(1, 'price_change_percentage_24h').iloc[0]['price_change_p
            }
        },
        'volatilidad': {
            'volatilidad_promedio': float(df['volatilidad_24h'].mean()),
            'volatilidad_mediana': float(df['volatilidad_24h'].median()),
            'volatilidad_maxima': float(df['volatilidad_24h'].max()),
            'criptos_alta_volatilidad': int(len(df[df['volatilidad_24h'] > 20]))
        },
        'concentracion_mercado': {
            'dominancia_top3': float(df.nlargest(3, 'market_cap')['dominancia_mercado'].sum()),
            'dominancia_top10': float(df.nlargest(10, 'market_cap')['dominancia_mercado'].sum()),
            'dominancia_top20': float(df.nlargest(20, 'market_cap')['dominancia_mercado'].sum()),
            'bitcoin_dominancia': float(df[df['symbol'] == 'btc']['dominancia_mercado'].values[0]) if 'btc
        }
    }

    return resumen
```

No pego todo por ser demasiado código, pero si los resultados

```
🔄 Preparando contexto para LLM...
============================================================
✅ Contexto preparado exitosamente

📊 Estructura del contexto:
   • Resumen Ejecutivo: 6 secciones
   • Top 10 Criptomonedas: 10 registros
   • Top 20 Criptomonedas: 20 registros
   • Análisis por Categorías: 12 categorías
   • Correlaciones: 5 correlaciones destacadas
   • Oportunidades: 1
   • Riesgos: 1
   • Alertas: 0

📋 Muestra del Resumen Ejecutivo:
{
  "market_cap_total_usd": 3330444708031.0,
  "market_cap_total_formatted": "$3.33T",
  "volumen_total_24h_usd": 189001705522.7,
  "volumen_total_24h_formatted": "$189.00B",
  "ratio_volumen_mcap": 5.674969023414298
}
```

```python
import os, json
from openai import OpenAI

import os
os.environ["OPENAI_API_KEY"] = "sk-pr[redacted]0x2_FIGdjVn

def conclusiones_openai(contexto_llm, model="gpt-4o-mini", temperature=0.5, max_tokens=900):
    """
    Envía el contexto a OpenAI y regresa conclusiones en español.
    - contexto_llm: dict (el que construiste con preparar_contexto_completo_llm)
    - model: el modelo de OpenAI (gpt-4o-mini es rápido y barato)
    """
    client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

    # Prompt equivalente al que usabas con Gemini
    prompt = f"""
Eres un experto en análisis financiero. Recibes información sobre los indicadores actuales de las criptomo
debes hacer una interpretación fácil de entender para personas en México.
Explica qué significan las métricas y sus implicaciones prácticas (oportunidades, riesgos, señales de aler

Contexto (JSON):
{json.dumps(contexto_llm, ensure_ascii=False)}
    """.strip()

    resp = client.chat.completions.create(
        model=model,
        temperature=temperature,
        max_tokens=max_tokens,
        messages=[
            {"role": "system", "content": "Eres un analista financiero claro, didáctico y preciso."},
            {"role": "user", "content": prompt}
        ],
    )
    return resp.choices[0].message.content
```

```
[3]  ✓ 0.0s                                                                    Python
```


texto = conclusiones_openai(contexto_llm)
print(texto)
[54]
17.3s
Python
- **Mejor Performer:** Zcash (+18.00%)
- **Peor Performer:** Canton (-6.37%)
- **Implicación:** La mayoría de las criptomonedas están en negativo, lo que puede ser una señal de corre
5. **Volatilidad:**
- **Volatilidad Promedio:** 4.29%
- **Criptomonedas de Alta Volatilidad:** 2
- **Implicación:** Alta volatilidad puede ofrecer oportunidades de ganancias rápidas, pero también puede
6. **Concentración del Mercado:**
- **Dominancia de las 3 principales criptomonedas:** 73.95%
- **Dominancia de Bitcoin:** 57%
- **Implicación:** Un mercado altamente concentrado puede ser riesgoso; si las criptomonedas principales
Oportunidades y Riesgos
- **Oportunidades:**
- **Monero (XMR):** Con un cambio positivo del 5.22% y una volatilidad controlada (12.87%), puede ser una
- **Riesgos:**
- **Zcash (ZEC):** Aunque ha tenido un rendimiento notable (+18.00%), su alta volatilidad (24.88%) puede
Conclusiones Prácticas
1. **Inversores en México:** Deben considerar el contexto actual del mercado. Con una capitalización alta y
2. **Estrategia de Inversión:** Es recomendable diversificar las inversiones en criptomonedas para mitigar
3. **Monitoreo Constante:** Dado que el mercado de criptomonedas es muy dinámico, es crucial monitorear las