Ejemplos comparando DuckDB SQL con Pandas.

Configuración inicial

DuckDB

```
import duckdb
import pandas as pd

# Crear conexión
conn = duckdb.connect()

# Crear datos de ejemplo
conn.execute("""
CREATE TABLE ventas (
    id INTEGER,
    vendedor VARCHAR,
    producto VARCHAR,
    categoria VARCHAR,
    cantidad INTEGER,
    precio DECIMAL(10,2),
    fecha DATE,
    region VARCHAR
)
""")

conn.execute("""
INSERT INTO ventas VALUES
(1, 'Ana', 'Laptop', 'Electrónicos', 2, 1200.00, '2024-01-15', 'Norte'),
(2, 'Carlos', 'Mouse', 'Electrónicos', 5, 25.50, '2024-01-16', 'Sur'),
(3, 'Ana', 'Teclado', 'Electrónicos', 3, 75.00, '2024-01-17', 'Norte'),
(4, 'María', 'Silla', 'Muebles', 1, 350.00, '2024-01-18', 'Centro'),
(5, 'Carlos', 'Mesa', 'Muebles', 2, 450.00, '2024-01-19', 'Sur'),
(6, 'Ana', 'Monitor', 'Electrónicos', 1, 300.00, '2024-01-20', 'Norte'),
(7, 'María', 'Lámpara', 'Muebles', 4, 80.00, '2024-01-21', 'Centro'),
(8, 'Pedro', 'Smartphone', 'Electrónicos', 1, 800.00, '2024-01-22', 'Este')
""")
```

Pandas

```
# Crear DataFrame equivalente
df = pd.DataFrame({
    'id': [1, 2, 3, 4, 5, 6, 7, 8],
    'vendedor': ['Ana', 'Carlos', 'Ana', 'María', 'Carlos', 'Ana', 'María', 'Pedro'],
    'producto': ['Laptop', 'Mouse', 'Teclado', 'Silla', 'Mesa', 'Monitor', 'Lámpara',
'Smartphone'],
    'categoria': ['Electrónicos', 'Electrónicos', 'Electrónicos', 'Muebles',
'Muebles', 'Electrónicos', 'Muebles', 'Electrónicos'],
    'cantidad': [2, 5, 3, 1, 2, 1, 4, 1],
    'precio': [1200.00, 25.50, 75.00, 350.00, 450.00, 300.00, 80.00, 800.00],
    'fecha': pd.to_datetime(['2024-01-15', '2024-01-16', '2024-01-17', '2024-01-18',
'2024-01-19', '2024-01-20', '2024-01-21', '2024-01-22']),
    'region': ['Norte', 'Sur', 'Norte', 'Centro', 'Sur', 'Norte', 'Centro', 'Este']
})
```

Ejemplos comparativos

1. SELECT básico con WHERE

DuckDB:

```sql
SELECT vendedor, producto, precio
FROM ventas
WHERE precio > 300;
```

Pandas:

```python
df[df['precio'] > 300][['vendedor', 'producto', 'precio']]
```

2. GROUP BY con agregaciones

DuckDB:

```sql
SELECT vendedor,
       COUNT(*) as total_ventas,
       SUM(cantidad) as total_cantidad,
       AVG(precio) as precio_promedio
FROM ventas
GROUP BY vendedor;
```

Pandas:

```python
df.groupby('vendedor').agg({
    'id': 'count',
    'cantidad': 'sum',
    'precio': 'mean'
}).rename(columns={
    'id': 'total_ventas',
    'cantidad': 'total_cantidad',
    'precio': 'precio_promedio'
})
```

3. WHERE con múltiples condiciones

DuckDB:

```sql
SELECT *
FROM ventas
WHERE categoria = 'Electrónicos'
  AND precio BETWEEN 100 AND 500;
```

Pandas:

```python
df[(df['categoria'] == 'Electrónicos') &
   (df['precio'].between(100, 500))]
```

4. GROUP BY con HAVING

DuckDB:

```sql
SELECT categoria,
       COUNT(*) as num_productos,
       SUM(precio * cantidad) as total_ingresos
FROM ventas
GROUP BY categoria
HAVING SUM(precio * cantidad) > 1000;
```

Pandas:

```python
resultado = df.groupby('categoria').agg({
    'id': 'count',
    'precio': lambda x: sum(x * df.loc[x.index, 'cantidad'])
}).rename(columns={
    'id': 'num_productos',
    'precio': 'total_ingresos'
})
resultado[resultado['total_ingresos'] > 1000]
```

5. ORDER BY

DuckDB:

```sql
SELECT vendedor, producto, precio
FROM ventas
ORDER BY precio DESC, vendedor ASC;
```

Pandas:

```python
df[['vendedor', 'producto', 'precio']].sort_values(
    ['precio', 'vendedor'],
    ascending=[False, True]
)
```

6. Filtros con IN y LIKE

DuckDB:

```sql
SELECT *
FROM ventas
WHERE vendedor IN ('Ana', 'Carlos')
  AND producto LIKE '%o%';
```

Pandas:

```python
df[(df['vendedor'].isin(['Ana', 'Carlos'])) &
   (df['producto'].str.contains('o'))]
```

7. Agregaciones por múltiples columnas

DuckDB:

```sql
SELECT region, categoria,
       COUNT(*) as ventas,
       AVG(precio) as precio_avg,
       MAX(cantidad) as max_cantidad
FROM ventas
GROUP BY region, categoria
ORDER BY region, categoria;
```

Pandas:

```python
df.groupby(['region', 'categoria']).agg({
    'id': 'count',
    'precio': 'mean',
    'cantidad': 'max'
}).rename(columns={
    'id': 'ventas',
    'precio': 'precio_avg',
    'cantidad': 'max_cantidad'
}).sort_index()
```

8. Columnas calculadas

DuckDB:

```sql
SELECT vendedor,
       producto,
       cantidad,
       precio,
       (cantidad * precio) as total,
       CASE
         WHEN precio > 500 THEN 'Alto'
         WHEN precio > 100 THEN 'Medio'
         ELSE 'Bajo'
       END as rango_precio
FROM ventas;
```

Pandas:

```
df_resultado = df.copy()
df_resultado['total'] = df_resultado['cantidad'] * df_resultado['precio']
df_resultado['rango_precio'] = pd.cut(
    df_resultado['precio'],
    bins=[0, 100, 500, float('inf')],
    labels=['Bajo', 'Medio', 'Alto'],
    right=False
)
df_resultado[['vendedor', 'producto', 'cantidad', 'precio', 'total', 'rango_precio']]
```

9. Filtros con fechas

DuckDB:

```
SELECT *
FROM ventas
WHERE fecha >= '2024-01-18'
  AND fecha <= '2024-01-21';
```

Pandas:

```
df[(df['fecha'] >= '2024-01-18') & (df['fecha'] <= '2024-01-21')]
```

10. TOP/LIMIT con condiciones

DuckDB:

```
SELECT TOP 3 vendedor, producto, precio
FROM ventas
WHERE categoria = 'Electrónicos'
ORDER BY precio DESC;
```

Pandas:

```
df[df['categoria'] == 'Electrónicos'][['vendedor', 'producto', 'precio']]\
    .nlargest(3, 'precio')
```

11. Estadísticas avanzadas por grupo

DuckDB:

```sql
SELECT vendedor,
       COUNT(*) as num_ventas,
       MIN(precio) as precio_min,
       MAX(precio) as precio_max,
       AVG(precio) as precio_promedio,
       STDDEV(precio) as precio_stddev
FROM ventas
GROUP BY vendedor
HAVING COUNT(*) > 1;
```

Pandas:

```python
stats = df.groupby('vendedor')['precio'].agg([
    'count', 'min', 'max', 'mean', 'std'
]).rename(columns={
    'count': 'num_ventas',
    'min': 'precio_min',
    'max': 'precio_max',
    'mean': 'precio_promedio',
    'std': 'precio_stddev'
})
stats[stats['num_ventas'] > 1]
```

12. Percentiles y cuartiles

DuckDB:

```sql
SELECT categoria,
       PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY precio) as q1,
       PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY precio) as mediana,
       PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY precio) as q3
FROM ventas
GROUP BY categoria;
```

Pandas:

```python
df.groupby('categoria')['precio'].quantile([0.25, 0.5, 0.75]).unstack()
```

13. Ejemplo complejo con subconsultas

DuckDB:

```sql
SELECT vendedor,
       COUNT(*) as num_ventas,
       AVG(precio) as precio_promedio
FROM ventas
WHERE precio > (SELECT AVG(precio) FROM ventas)
GROUP BY vendedor
ORDER BY num_ventas DESC;
```

Pandas:

```python
precio_promedio_global = df['precio'].mean()
df_filtrado = df[df['precio'] > precio_promedio_global]
resultado = df_filtrado.groupby('vendedor').agg({
    'id': 'count',
    'precio': 'mean'
}).rename(columns={
    'id': 'num_ventas',
    'precio': 'precio_promedio'
}).sort_values('num_ventas', ascending=False)
```

Estos ejemplos muestran las principales operaciones SQL y sus equivalentes en Pandas. DuckDB ofrece sintaxis SQL familiar, mientras que Pandas proporciona métodos específicos de DataFrame que pueden ser más expresivos para análisis de datos complejos.