

ANÁLISIS ENTREGA VENTAS

Por:Victoria Abendaño
Freddy Curicama

INTRODUCCIÓN

Este proyecto tiene como finalidad aplicar técnicas de análisis de datos utilizando la plataforma JupyterLab, procesando información de ventas y entregas de productos tecnológicos. El objetivo es evaluar el cumplimiento logístico, el desempeño comercial y encontrar patrones de comportamiento relevantes.



TÉCNOLOGIAS USADAS



- **JupyterLab:** entorno interactivo de notebooks ideal para ciencia de datos.
- **Pandas:** librería para manipulación y análisis de datos estructurados.
- **SQLite3:** motor de base de datos ligera y embebida, usado para consultar datos desde archivos ` `.db` .
- **Matplotlib:** herramienta de visualización para gráficos estáticos.
- **Bokeh:** librería interactiva para gráficos modernos, útiles en análisis exploratorios.
- **PyGWalker:** interfaz gráfica que permite analizar datasets de forma visual como si fuera una herramienta tipo Tableau.

DATASET UTILIZADOS

Dataset 1: `ventas_tecnologia_10000.csv`

Este dataset contiene 10,000 registros de ventas de productos tecnológicos, con atributos como producto, fecha de venta, cantidad, precio unitario, canal de ventas , identificación del cliente, simulando una fuente de datos de un sistema de ventas de una empresa.

	producto	fecha_venta	cantidad	precio_unitario	canal_venta	cliente_id	ciudad
0	CPU Intel i5	2024-07-14	38	818.45	Online	CL-0009	Loja
1	Laptop Lenovo	2024-09-26	18	308.13	Tienda Física	CL-0043	Loja
2	Monitor Samsung	2024-08-20	9	531.46	Online	CL-0097	Ambato
3	Laptop HP	2024-12-29	44	1212.05	Tienda Física	CL-0097	Ambato
4	Impresora HP	2024-05-21	37	171.28	Online	CL-0039	Quito

DATASET UTILIZADOS

Dataset 2: `entregas_tecnologia_10000.db` (tabla: `entregas`)

Este dataset contiene 10,000 registros de entregas de productos tecnológicos, con atributos como producto, fecha de entrega, cantidad entregada, ciudad de entrega y tipo de transporte, representando datos relacionados con la distribución de los productos.

	producto	fecha_entrega	cantidad_entregada	bodega_origen	ciudad_entrega	tipo_transporte
0	Impresora Epson	2024-07-20	21	Bodega Sur	Loja	Aéreo
1	Impresora Epson	2024-02-13	1	Bodega Sur	Cuenca	Terrestre
2	CPU AMD Ryzen	2024-01-08	38	Bodega Sur	Quito	Terrestre
3	Laptop HP	2024-05-01	40	Bodega Norte	Cuenca	Aéreo
4	Laptop HP	2024-11-04	1	Bodega Sur	Quito	Aéreo

FUSIÓN DE DATASETS

usando `pd.merge()` por `producto` y `ciudad`.

```
#Unión (merge) de los dos datasets usando las columnas producto y ciudad como claves,Solo se conservan las filas que coincidan e
merged_df = pd.merge(
    ventas_df,
    entregas_df,
    left_on=['producto', 'ciudad'],
    right_on=['producto', 'ciudad_entrega'],
    how='inner'
)
# creación de nuevas columnas
merged_df['ingreso'] = merged_df['cantidad'] * merged_df['precio_unitario'] #ingreso: total generado por cada venta (cantidad x
merged_df['diferencia_cantidadades'] = merged_df['cantidad'] - merged_df['cantidad_entregada'] #diferencia_cantidadades: diferencia

merged_df.head()
```

	producto	fecha_venta	cantidad	precio_unitario	canal_venta	cliente_id	ciudad	fecha_entrega	cantidad_entregada	bodega_origen	ciudad_
0	CPU Intel i5	2024-07-14	38	818.45	Online	CL-0009	Loja	2024-01-01		29	Bodega Sur
1	CPU Intel i5	2024-07-14	38	818.45	Online	CL-0009	Loja	2024-01-28		38	Bodega Norte
2	CPU Intel i5	2024-07-14	38	818.45	Online	CL-0009	Loja	2024-03-11		39	Bodega Sur
3	CPU Intel i5	2024-07-14	38	818.45	Online	CL-0009	Loja	2024-07-02		28	Bodega Norte
4	CPU Intel i5	2024-07-14	38	818.45	Online	CL-0009	Loja	2024-06-13		6	Bodega Sur

GRÁFICO MATPLOTLIB

```
# Gráfico 1 con Matplotlib: Muestra la diferencia de equipos Vendidos y Entregados

# Agrupar por producto y sumar la diferencia
resumen_diferencia = merged_df.groupby('producto')['diferencia_cantidades'].sum().sort_values()

plt.figure(figsize=(12, 5)) # Crea una nueva figura para el gráfico de tamaño 12 de ancho por 5 de alto en pulgadas.
resumen_diferencia.plot(kind='bar', color='skyblue') # Dibuja un gráfico de barras verticales (kind='bar') con los datos agrupados.

plt.axhline(0, color='gray', linestyle='--') # Línea horizontal en 0 para visualizar fácilmente positivos vs negativos

plt.title("Diferencia Total Entregada vs Vendida por Producto") #Establece el título del gráfico.
plt.ylabel("Unidades de Diferencia (Vendidas - Entregadas)") # Etiqueta del eje Y indicando claramente el significado de los valores.
plt.xticks(rotation=90) #Rota las etiquetas del eje X (nombres de productos) 90° para evitar que se encimen y mejorar la legibilidad.
plt.tight_layout() # Ajusta automáticamente el espacio entre los elementos del gráfico para que no se corten al mostrarse.
plt.show() #Muestra el gráfico en pantalla.
```

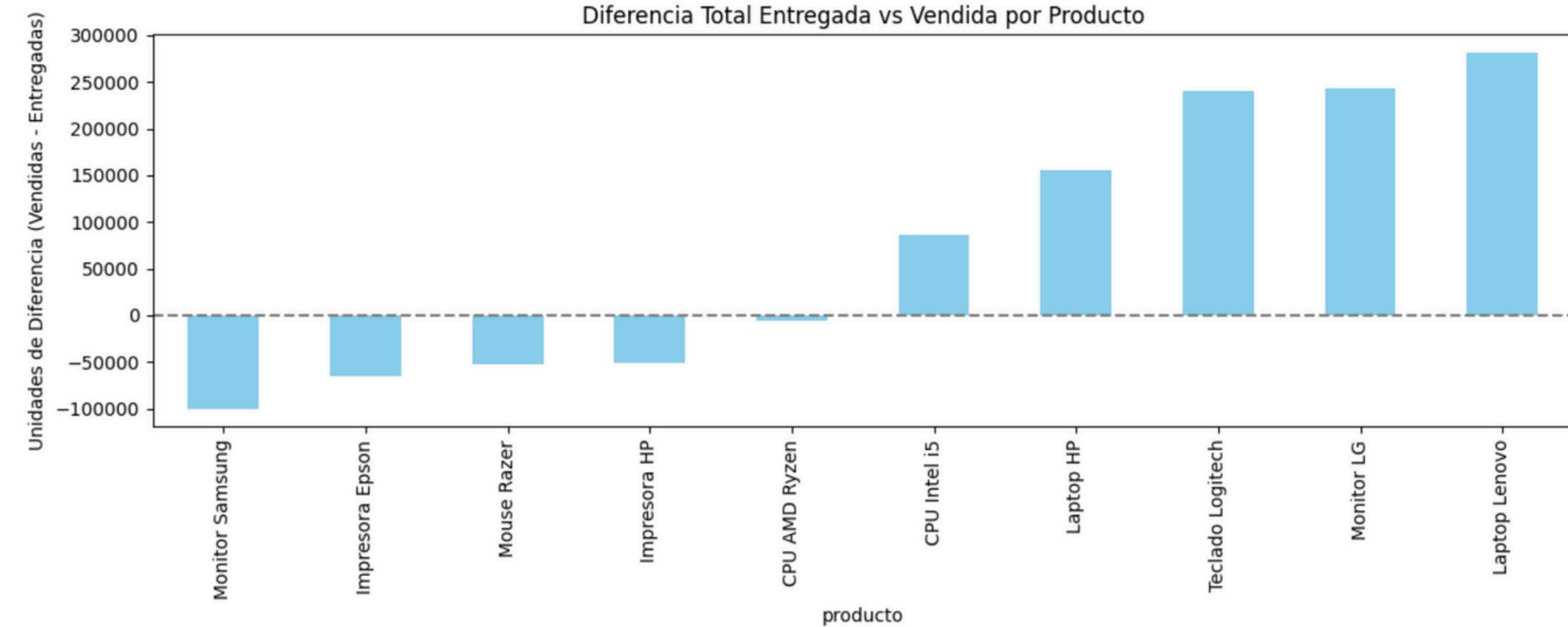


GRÁFICO MATPLOTLIB

```
# Gráfico 2 con Matplotlib: Muestra la diferencia de equipos Vendidos y Entregados
```

```
ciudad_df = merged_df.groupby("ciudad")[["cantidad", "cantidad_entregada"]].sum().reset_index() #Agrupa el DataFrame merged_df por ciudad y summa las columnas cantidad y cantidad_entregada. Luego se resetea el index para que sea una columna nueva.
```

```
# Datos
```

```
labels = ciudad_df["ciudad"] #Extrae la lista de nombres de ciudades para el eje X.
```

```
cantidad_vendida = ciudad_df["cantidad"] #Obtiene el total de unidades vendidas por ciudad.
```

```
cantidad_entregada = ciudad_df["cantidad_entregada"] #Obtiene el total de unidades entregadas por ciudad.
```

```
x = np.arange(len(labels)) # Crea un arreglo con las posiciones (índices) de cada ciudad, necesario para alinear las barras.
```

```
width = 0.35 # Define el ancho que tendrá cada barra (vendida o entregada).
```

```
# Crear gráfico
```

```
fig, ax = plt.subplots(figsize=(12, 6))# Crea la figura y el eje para el gráfico, con tamaño personalizado de 12x6 pulgadas.
```

```
bars1 = ax.bar(x - width/2, cantidad_vendida, width, label='Cantidad Vendida')# Dibuja las barras de cantidad vendida, desplazadas a la izquierda.
```

```
bars2 = ax.bar(x + width/2, cantidad_entregada, width, label='Cantidad Entregada') #Dibuja las barras de cantidad entregada, desplazadas a la derecha.
```

```
# Personalización
```

```
ax.set_xlabel('Ciudad') #Etiqueta del eje X.
```

```
ax.set_ylabel('Cantidad') #Etiqueta del eje Y.
```

```
ax.set_title('Cantidad Vendida vs Entregada por Ciudad') #Título del gráfico.
```

```
ax.set_xticks(x) #Establece las posiciones para cada etiqueta del eje X (ciudades).
```

```
ax.set_xticklabels(labels, rotation=45)# Asigna los nombres de las ciudades a las posiciones y las rota 45° para evitar que se overlappen.
```

```
ax.legend() # Agrega una leyenda para identificar qué color corresponde a cada variable (vendida vs entregada).
```

```
plt.tight_layout() # Ajusta automáticamente los márgenes para que todo se vea bien sin montarse.
```

```
plt.show() #Muestra el gráfico en pantalla.
```

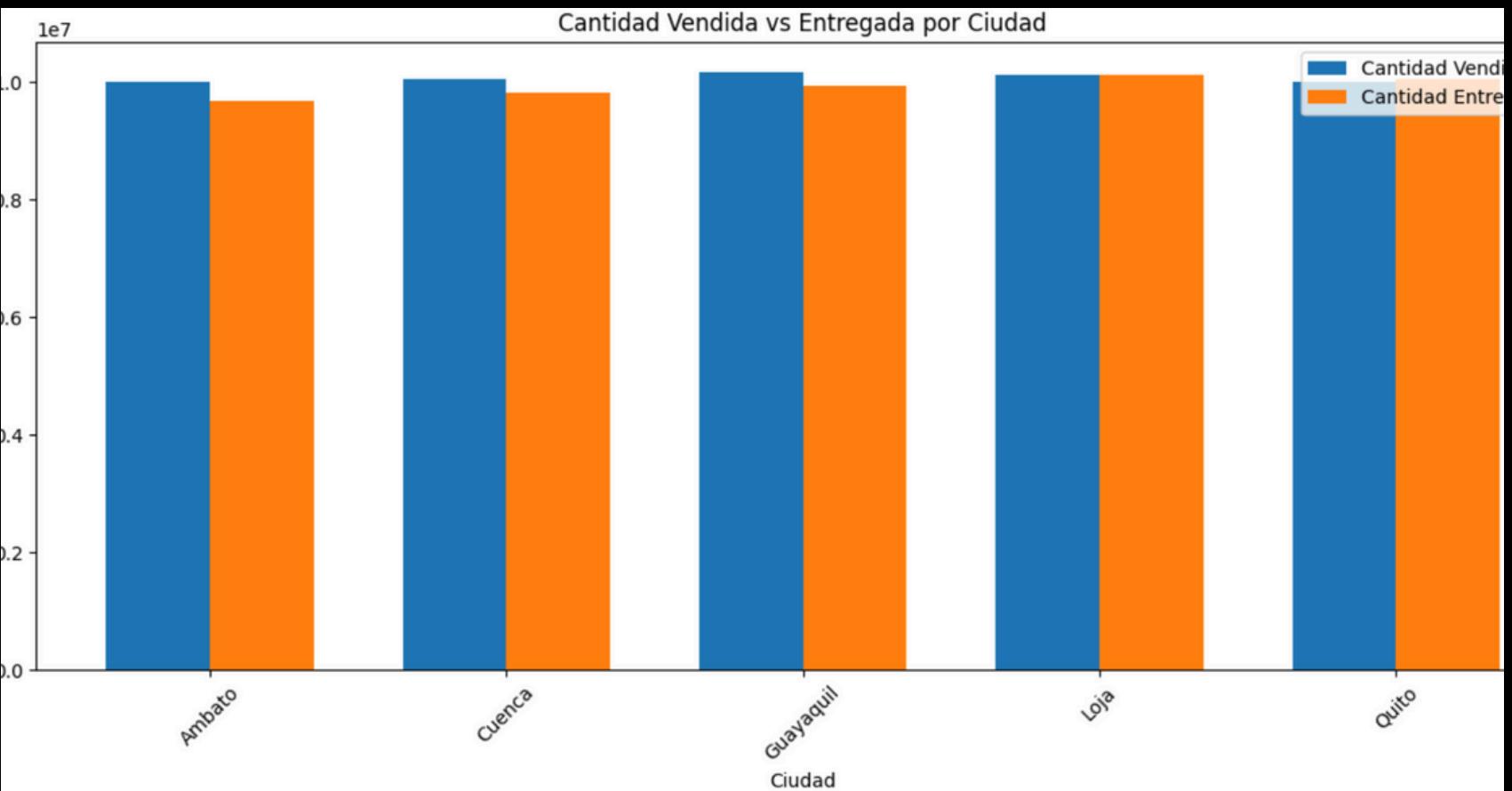


GRÁFICO BOKEH

```
# Gráfico 1 con Bokeh: Muestra la la suma de los ingresos por ciudad

ingresos_ciudad = merged_df.groupby("ciudad")["ingreso"].sum().reset_index()# Agrupa el DataFrame merged_df por la columna ciudad
source = ColumnDataSource(ingresos_ciudad) #Se crea un objeto fuente de datos de Bokeh (ColumnDataSource) con el DataFrame ingresos_ciudad

#Creación del gráfico Bokeh:Se define un gráfico de barras verticales (vbar) para mostrar ingresos por ciudad.
p = figure(x_range=ingresos_ciudad["ciudad"], title="Ingresos Totales por Ciudad", height=350)# Crea una figura Bokeh:
p.vbar(x='ciudad', top='ingreso', width=0.7, source=source) # Dibuja las barras verticales (vbar) del gráfico:
p.xaxis.major_label_orientation = "vertical" # p.xaxis.major_label_orientation = "vertical"
p.yaxis.axis_label = "Ingresos" #Establece el nombre del eje Y como Ingresos.
p.xaxis.axis_label = "Ciudad" #Establece el nombre del eje X como Ciudad.

show(p) #Muestra el gráfico en pantalla.
```

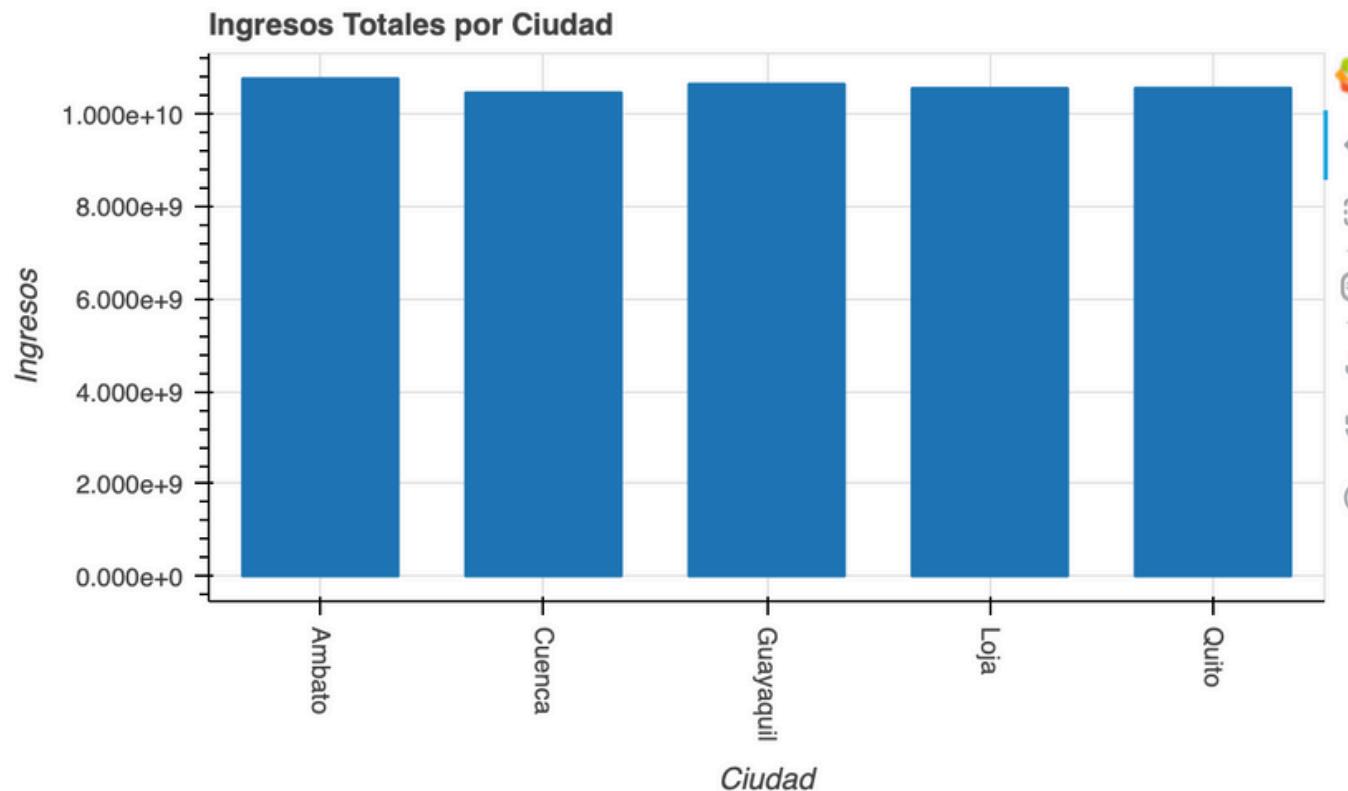


GRÁFICO BOKEH

#Gráfico 2 con Bokeh:Muestra productos de mayor a menor ingreso por producto.

```
ingreso_producto = merged_df.groupby("producto")["ingreso"].sum().reset_index()#Agrupa el DataFrame merged_df por la columna producto
ingreso_producto = ingreso_producto.sort_values("ingreso", ascending=False)# Ordena el DataFrame ingreso_producto en orden descendente
# Crear fuente de datos
source = ColumnDataSource(ingreso_producto)# Convierte el DataFrame ingreso_producto en una fuente de datos que Bokeh puede utilizar
#Crea una figura (figure) de Bokeh:
p = figure(y_range=ingreso_producto["producto"], #define el rango del eje Y como la lista de productos.
           title="Ingreso Total por Producto", # título del gráfico.
           width=800, height=600) #dimensiones del gráfico.

p.hbar(y="producto", right="ingreso", height=0.6, source=source, color="#0072B2") #Dibuja barras horizontales (hbar) en el gráfico
p.xaxis.axis_label = "Ingreso" #Etiqueta del eje X con el texto "Ingreso".
p.yaxis.axis_label = "Producto" #Etiqueta del eje Y con el texto "Producto".
p.y_range.range_padding = 0.1 #Añade un pequeño espacio arriba y abajo del rango del eje Y para que las barras no queden pegadas
p.ygrid.grid_line_color = None #Elimina las líneas de la cuadrícula del eje Y para un aspecto visual más limpio.

show(p) #Muestra el gráfico generado en pantalla
```

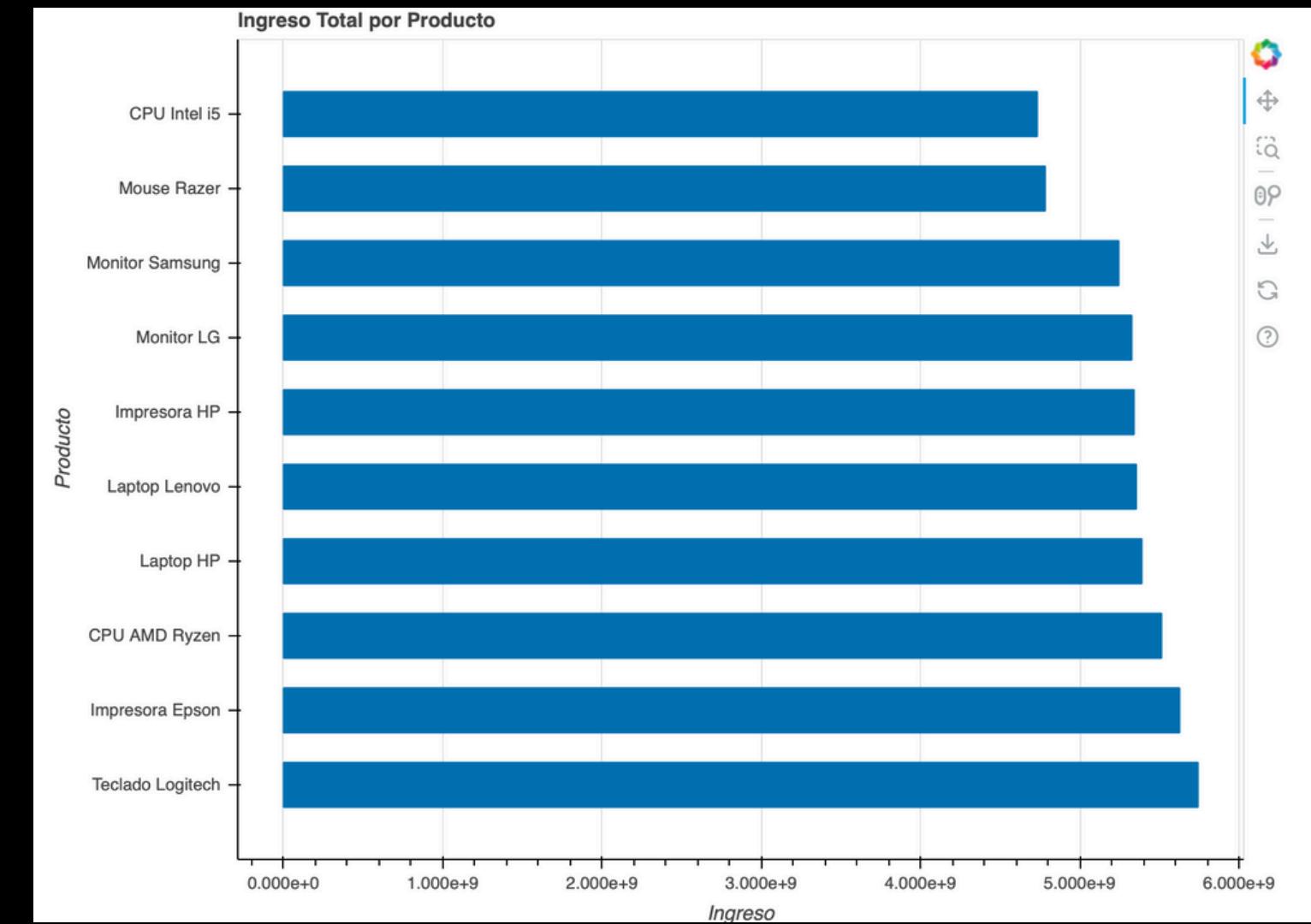


GRÁFICO BOKEH

#Gráfico 3 con Bokeh:Muestra Tipo de Transporte utilizado.

```
# Agrupar tipo de transporte
data = entregas_df["tipo_transporte"].value_counts().reset_index()
data.columns = ['tipo_transporte', 'conteo']
data['angle'] = data['conteo'] / data['conteo'].sum() * 2 * pi

# Asignar colores de forma segura (aunque haya menos de 3 o más de 20 elementos)
from itertools import cycle
palette = cycle(Category20c[20]) # Reutiliza colores si hay más de 20
data['color'] = [next(palette) for _ in range(len(data))]

# Crea un gráfico Bokeh
p = figure(height=400, title="Distribución de Entregas (Donut Chart)",
           toolbar_location=None, tools="hover", tooltips="@tipo_transporte: @conteo", x_range=(-0.5, 1.0))

#Dibuja los "segmentos" del gráfico tipo donut
p.annular_wedge(x=0, y=1, inner_radius=0.2, outer_radius=0.4, #posición del centro del gráfico, define el tamaño del hueco interno
                 start_angle=cumsum('angle', include_zero=True), end_angle=cumsum('angle'), #definen el ángulo de cada cuña usando la función cumsum
                 line_color="white", fill_color='color', legend_field='tipo_transporte', #cada segmento se colorea con los valores correspondientes
                 source=ColumnDataSource(data)) # usa como datos el ColumnDataSource(data)

p.axis.visible = False #Oculta los ejes (no se necesitan en gráficos circulares).
p.grid.grid_line_color = None #Elimina las líneas de la cuadricula de fondo para un diseño más limpio

show(p) #Muestra el gráfico generado en pantalla
```

Distribución de Entregas (Donut Chart)

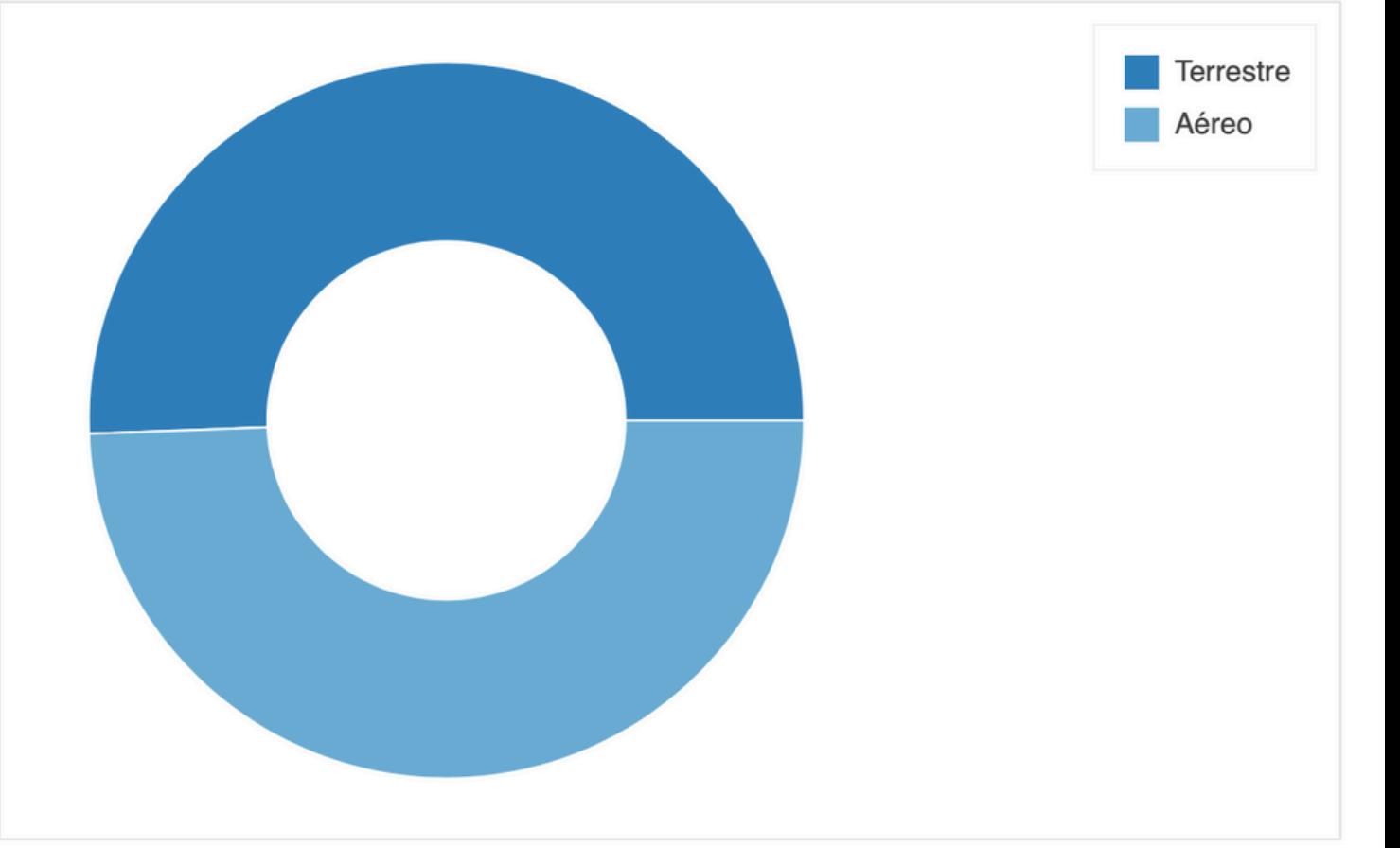


GRÁFICO PYGWALKER

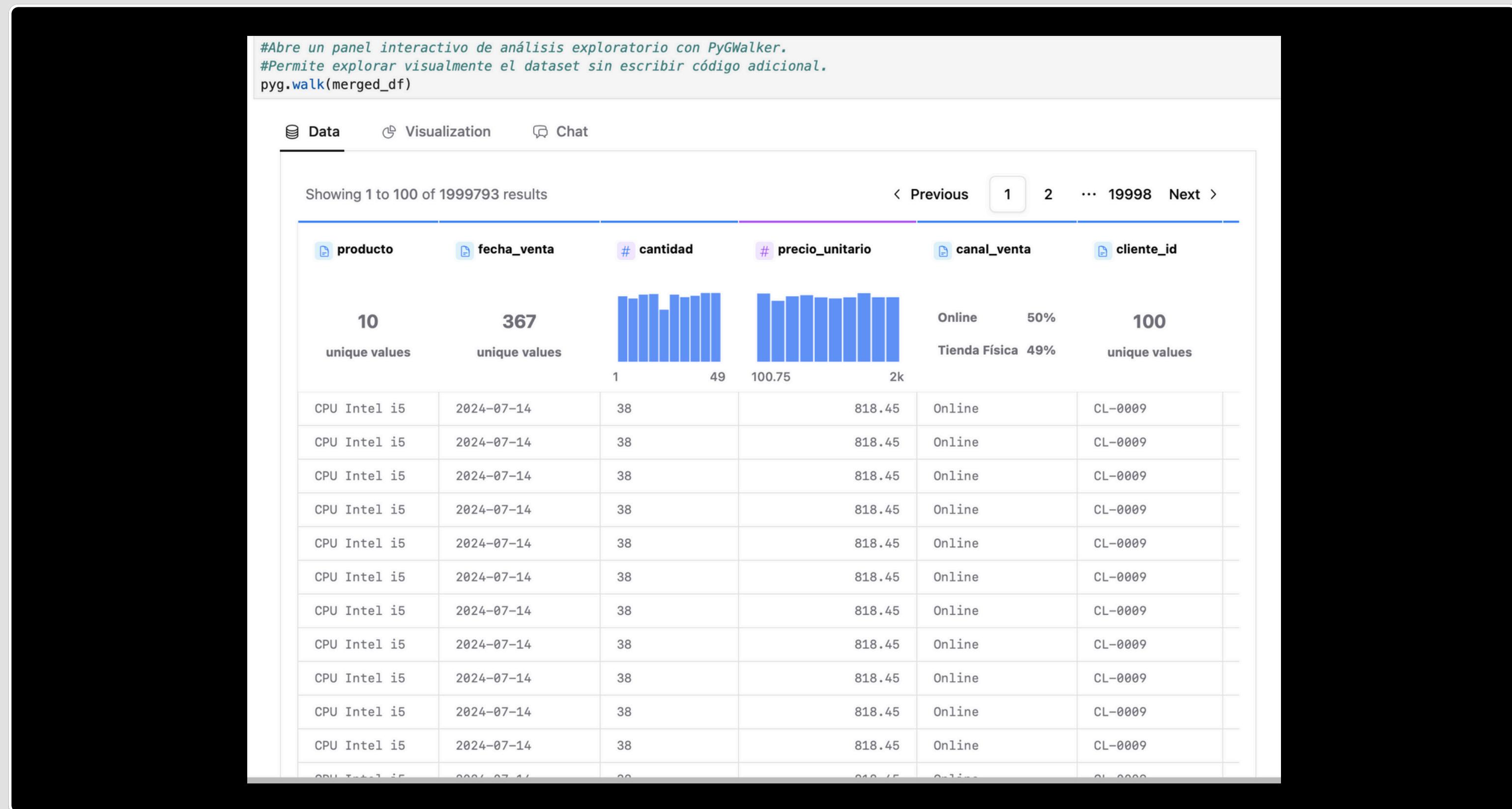
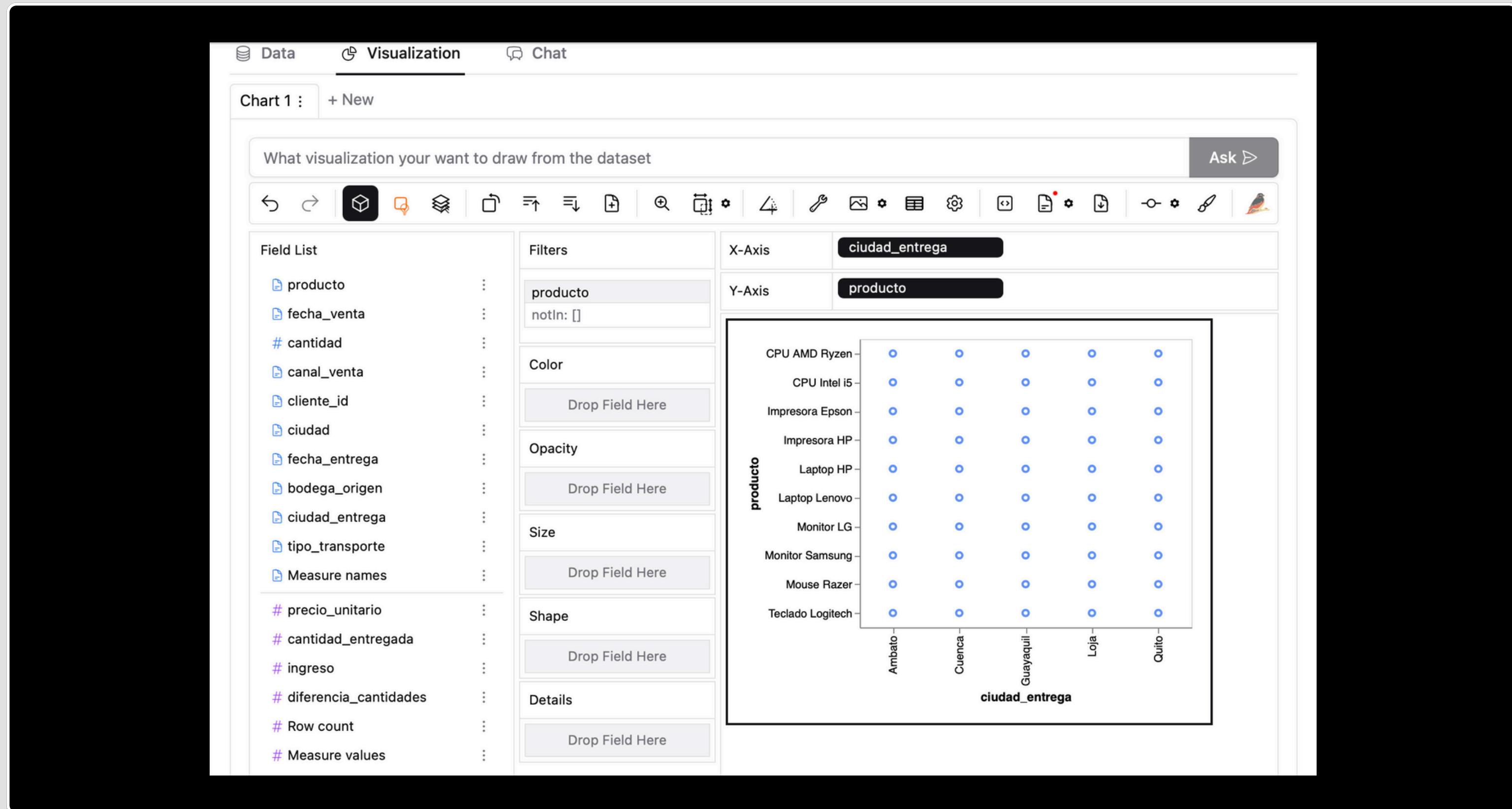


GRÁFICO PYGWALKER



CONCLUSIONES

- Se detectaron diferencias importantes entre lo vendido y lo entregado en ciertos productos.
- La ciudad con mayores ingresos fue claramente identificada gracias a los gráficos de Bokeh.
- La visualización de tipo de transporte permitió conocer las rutas logísticas más frecuentes.
- Las herramientas utilizadas permitieron automatizar el análisis y facilitar la visualización de hallazgos clave.





¡MUCHAS GRACIAS!

Herramientas para la IA