```
!pip install dask hvplot --quiet

# Import libraries
from dask.distributed import Client
import dask.dataframe as dd
import hvplot.dask
import pandas as pd
import hvplot.pandas
```

```
# Start local Dask client
client = Client()
client  # will show dashboard link
```

```
INFO:distributed.http.proxy:To route to workers diagnostics web server please install jupyter-server-proxy: python -m pip install jupyte
INFO:distributed.scheduler:State start
INFO:distributed.scheduler:  Scheduler at:     tcp://127.0.0.1:33931
INFO:distributed.scheduler:  dashboard at:  http://127.0.0.1:8787/status
INFO:distributed.scheduler:Registering Worker plugin shuffle
INFO:distributed.nanny:        Start Nanny at: 'tcp://127.0.0.1:41019'
INFO:distributed.nanny:        Start Nanny at: 'tcp://127.0.0.1:38323'
INFO:distributed.scheduler:Register worker addr: tcp://127.0.0.1:45129 name: 0
INFO:distributed.scheduler:Starting worker compute stream, tcp://127.0.0.1:45129
INFO:distributed.core:Starting established connection to tcp://127.0.0.1:58622
INFO:distributed.scheduler:Register worker addr: tcp://127.0.0.1:44071 name: 1
INFO:distributed.scheduler:Starting worker compute stream, tcp://127.0.0.1:44071
INFO:distributed.core:Starting established connection to tcp://127.0.0.1:58632
INFO:distributed.scheduler:Receive client connection: Client-8cb8ff44-8adf-11f0-8184-0242ac1c000c
INFO:distributed.core:Starting established connection to tcp://127.0.0.1:58642
```

Client

Client-8cb8ff44-8adf-11f0-8184-0242ac1c000c

Connection method: Cluster object **Cluster type:** distributed.LocalCluster

Dashboard: http://127.0.0.1:8787/status

▸ **Cluster Info**

```
# Upload Blinkit dataset (CSV)
from google.colab import files
uploaded = files.upload()
```

Choose Files BlinkIT Grocery Data.xlsx

BlinkIT Grocery Data.xlsx(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 596867 bytes, last modified: 9/6/2025 - 100% done

```
Saving BlinkIT Grocery Data.xlsx to BlinkIT Grocery Data.xlsx
```

```
# Example dtype fixes (adjust according to your Blinkit dataset columns)
dtype_fix = {
    "Order_ID": "object",
    "Product": "object",
    "Category": "object",
    "Quantity": "int64",
    "Price": "float64",
    "Revenue": "float64",
    "Date": "object",  # will convert later
}
```

```
import pandas as pd

# Load Excel with pandas
blinkit_excel = pd.read_excel("BlinkIT Grocery Data.xlsx")

# Save to CSV
blinkit_excel.to_csv("blinkit.csv", index=False)

# Now load with Dask
import dask.dataframe as dd

ddf = dd.read_csv("blinkit.csv", assume_missing=True)
ddf.head()
```

	dddddddD	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility	Item Weight	Sales	Rating
0	Regular	FDX32	Fruits and Vegetables	2012.0	OUT049	Tier 1	Medium	Supermarket Type1	0.100014	15.10	145.4786	5.0
1	Low Fat	NCB42	Health and Hygiene	2022.0	OUT018	Tier 3	Medium	Supermarket Type2	0.008596	11.80	115.3492	5.0
2	Regular	FDR28	Frozen Foods	2016.0	OUT046	Tier 1	Small	Supermarket Type1	0.025896	13.85	165.0210	5.0
3	Regular	FDL50	Canned	2014.0	OUT013	Tier 3	High	Supermarket Type1	0.042278	12.15	126.5046	5.0
4	Low Fat	DRI25	Soft Drinks	2015.0	OUT045	Tier 2	Small	Supermarket	0.033970	19.60	55.1614	5.0

```
# Load dataset
ddf = dd.read_csv("blinkit.csv", dtype=dtype_fix, assume_missing=True)


# Preview dataset
ddf.head()
```

	dddddddD	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility	Item Weight	Sales	Rating
0	Regular	FDX32	Fruits and Vegetables	2012.0	OUT049	Tier 1	Medium	Supermarket Type1	0.100014	15.10	145.4786	5.0
1	Low Fat	NCB42	Health and Hygiene	2022.0	OUT018	Tier 3	Medium	Supermarket Type2	0.008596	11.80	115.3492	5.0
2	Regular	FDR28	Frozen Foods	2016.0	OUT046	Tier 1	Small	Supermarket Type1	0.025896	13.85	165.0210	5.0
3	Regular	FDL50	Canned	2014.0	OUT013	Tier 3	High	Supermarket Type1	0.042278	12.15	126.5046	5.0
4	Low Fat	DRI25	Soft Drinks	2015.0	OUT045	Tier 2	Small	Supermarket	0.033970	19.60	55.1614	5.0

```
# Dataset Info
ddf.info()
```

```
<class 'dask.dataframe.dask_expr.DataFrame'>
Columns: 12 entries, dddddddD to Rating
dtypes: float64(5), string(7)
```

```
# Columns
print(ddf.columns)
```

```
Index(['dddddddD', 'Item Identifier', 'Item Type', 'Outlet Establishment Year',
       'Outlet Identifier', 'Outlet Location Type', 'Outlet Size',
       'Outlet Type', 'Item Visibility', 'Item Weight', 'Sales', 'Rating'],
      dtype='object')
```

```
# Number of partitions
print("Partitions:", ddf.npartitions)
```

```
Partitions: 1
```

```
# Convert sample to pandas for preview
df_sample = ddf.head(10)
df_sample
```

	dddddddD	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility	Item Weight	Sales	Rating
0	Regular	FDX32	Fruits and Vegetables	2012.0	OUT049	Tier 1	Medium	Supermarket Type1	0.100014	15.10	145.4786	5.0
1	Low Fat	NCB42	Health and Hygiene	2022.0	OUT018	Tier 3	Medium	Supermarket Type2	0.008596	11.80	115.3492	5.0
2	Regular	FDR28	Frozen Foods	2016.0	OUT046	Tier 1	Small	Supermarket Type1	0.025896	13.85	165.0210	5.0
3	Regular	FDL50	Canned	2014.0	OUT013	Tier 3	High	Supermarket Type1	0.042278	12.15	126.5046	5.0
4	Low Fat	DRI25	Soft Drinks	2015.0	OUT045	Tier 2	Small	Supermarket Type1	0.033970	19.60	55.1614	5.0
5	low fat	FDS52	Frozen Foods	2020.0	OUT017	Tier 2	Small	Supermarket Type1	0.005505	8.89	102.4016	5.0
6	Low Fat	NCU05	Health and Hygiene	2011.0	OUT010	Tier 3	Small	Grocery Store	0.098312	11.80	81.4618	5.0
7	Low Fat	NCD30	Household	2015.0	OUT045	Tier 2	Small	Supermarket Type1	0.026904	19.70	96.0726	5.0
8	Low Fat	FDW20	Fruits and Vegetables	2014.0	OUT013	Tier 3	High	Supermarket Type1	0.024129	20.75	124.1730	5.0
9	Low Fat	FDX25	Canned	2018.0	OUT027	Tier 3	Medium	Supermarket	0.101562	NaN	181.9292	5.0

Next steps: Generate code with df_sample | View recommended plots | New interactive sheet

```
# Describe statistics
ddf.describe().compute()
```

	Outlet Establishment Year	Item Visibility	Item Weight	Sales	Rating
count	8523.000000	8523.000000	7060.000000	8523.000000	8523.000000
mean	2016.450546	0.066132	12.857645	140.992783	3.965857
std	3.189396	0.051598	4.643456	62.275067	0.605651
min	2011.000000	0.000000	4.555000	31.290000	1.000000
25%	2014.000000	0.026989	8.773750	93.826500	4.000000
50%	2016.000000	0.053931	12.600000	143.012800	4.000000
75%	2018.000000	0.094585	16.850000	185.643700	4.200000
max	2022.000000	0.328391	21.350000	266.888400	5.000000

```
# Count rows
print("Total Rows:", ddf.shape[0].compute())
```

```
Total Rows: 8523
```

```
# Count null values
ddf.isnull().sum().compute()
```

	0
ddddddddD	0
Item Identifier	0
Item Type	0
Outlet Establishment Year	0
Outlet Identifier	0
Outlet Location Type	0
Outlet Size	0
Outlet Type	0
Item Visibility	0
Item Weight	1463
Sales	0
Rating	0

dtype: int64

**

DATA ANALYSIS

**

```
# Rename the column properly
ddf = ddf.rename(columns={"ddddddddD": "Date"})

# Now convert it to datetime
ddf["Date"] = dd.to_datetime(ddf["Date"], errors="coerce")



print(ddf.head())  # Check your first rows
```

```
  Date Item Identifier              Item Type  Outlet Establishment Year  \
0  NaT           FDX32  Fruits and Vegetables                     2012.0
1  NaT           NCB42     Health and Hygiene                     2022.0
2  NaT           FDR28           Frozen Foods                     2016.0
3  NaT           FDL50                 Canned                     2014.0
4  NaT           DRI25            Soft Drinks                     2015.0

  Outlet Identifier Outlet Location Type Outlet Size        Outlet Type  \
0            OUT049               Tier 1      Medium  Supermarket Type1
1            OUT018               Tier 3      Medium  Supermarket Type2
2            OUT046               Tier 1       Small  Supermarket Type1
3            OUT013               Tier 3        High  Supermarket Type1
4            OUT045               Tier 2       Small  Supermarket Type1

   Item Visibility  Item Weight     Sales  Rating
0         0.100014        15.10  145.4786     5.0
1         0.008596        11.80  115.3492     5.0
2         0.025896        13.85  165.0210     5.0
3         0.042278        12.15  126.5046     5.0
4         0.033970        19.60   55.1614     5.0
```

```
# --- Exploratory Analysis ---

# 1. Total sales per year
sales_per_year = ddf.groupby("Outlet Establishment Year")["Sales"].sum().compute()

# Convert Series → DataFrame
sales_per_year = sales_per_year.reset_index()

# Now hvplot works
sales_per_year.hvplot.bar(
    x="Outlet Establishment Year", y="Sales",
    title="Total Sales per Establishment Year",
    xlabel="Year", ylabel="Total Sales",
```

```
    width=800, height=400
)
```

```
# 2. Sales by Item Type
sales_by_type = ddf.groupby("Item Type")["Sales"].sum().compute()
print(sales_by_type)

sales_by_type.hvplot.bar(
    title="Sales by Item Type",
    xlabel="Item Type", ylabel="Sales",
    rot=45, width=800, height=400
)
```

```
Item Type
Baking Goods              81894.7364
Breads                    35379.1198
Breakfast                 15596.6966
Canned                    90706.7290
Dairy                    101276.4616
Frozen Foods             118558.8814
Fruits and Vegetables    178124.0810
Hard Drinks               29334.6806
Health and Hygiene        68025.8388
Household                135976.5254
Meat                      59449.8638
Others                    22451.8916
Seafood                    9077.8700
Snack Foods              175433.9224
Soft Drinks               58514.1670
Starchy Foods             21880.0274
Name: Sales, dtype: float64
```

```
# 3. Average sales by Outlet Size
avg_sales_size = ddf.groupby("Outlet Size")["Sales"].mean().compute()
print(avg_sales_size)

avg_sales_size.hvplot.bar(
    title="Average Sales by Outlet Size",
    xlabel="Outlet Size", ylabel="Average Sales"
)
```

```
Outlet Size
High      142.037414
Medium    139.877647
Small     141.699322
Name: Sales, dtype: float64
```

```
# 4. Rating vs Sales correlation
# Drop NaN and compute directly with pandas (avoids empty partition issue)
df_corr = ddf[["Sales", "Rating"]].dropna().compute()

# Correlation matrix
correlation = df_corr.corr()
print("Correlation between Sales & Rating:")
print(correlation)
```

```
Correlation between Sales & Rating:
            Sales    Rating
Sales   1.000000  0.011329
Rating  0.011329  1.000000
```

Correlation between Sales & Rating: 0.011 which means no meaningful relationship.

```
df_corr.hvplot.scatter(
    x="Rating", y="Sales",
    title="Sales vs Rating Scatterplot",
    alpha=0.5, size=5, color="blue"
)
```

```
# 5. Save smaller sample
ddf_sample = ddf.head(20000, compute=True)
ddf_sample.to_csv("blinkit_small.csv", index=False)
```

```
# Reload small dataset
ddf_small = dd.read_csv("blinkit_small.csv")
print(ddf_small.head())
```

```
    Date Item Identifier              Item Type  Outlet Establishment Year  \
0   NaN           FDX32  Fruits and Vegetables                      2012.0
1   NaN           NCB42     Health and Hygiene                      2022.0
2   NaN           FDR28           Frozen Foods                      2016.0
3   NaN           FDL50                 Canned                      2014.0
4   NaN           DRI25            Soft Drinks                      2015.0

  Outlet Identifier Outlet Location Type Outlet Size        Outlet Type  \
0            OUT049               Tier 1      Medium  Supermarket Type1
1            OUT018               Tier 3      Medium  Supermarket Type2
2            OUT046               Tier 1       Small  Supermarket Type1
3            OUT013               Tier 3        High  Supermarket Type1
4            OUT045               Tier 2       Small  Supermarket Type1

   Item Visibility  Item Weight     Sales  Rating
0         0.100014        15.10  145.4786     5.0
1         0.008596        11.80  115.3492     5.0
2         0.025896        13.85  165.0210     5.0
3         0.042278        12.15  126.5046     5.0
4         0.033970        19.60   55.1614     5.0
```

```
# 6. Final Visualization: Sales & Rating per Item Type
# Example for sales by item type
sales_by_type = ddf.groupby("Item Type")["Sales"].sum().compute()

sales_by_type = sales_by_type.reset_index()  # make it a clean Pandas DF

sales_by_type.hvplot.bar(
    x="Item Type",
    y="Sales",
    title="Total Sales by Item Type",
    rot=45
)
```