|  |  |
| --- | --- |
| 1. Sample random rows | df.sample() # Default sample  df.sample(20) # Sample 20 rows |
| 1. Number of missing values in each column | df.isna().sum() # Count per column  (df.isna().sum() / len(df))\  .sort\_values(ascending=False) # % missing per column |
| 1. Descriptive statistics | df.describe() # Basic stats (25/50/75 %)  df.describe(percentiles=[.10, .50, .90]) # Custom percentiles |
| 1. Correlation matrix | numeric\_corr = df.select\_dtypes(include="number").corr() # Correlate numerics  sns.heatmap(numeric\_corr, annot=True) # Visualize correlations |
| 1. Frequency tables | df["col\_1"].value\_counts() # Frequency of each category  pd.crosstab(df["col\_1"], df["col\_2"],  normalize="all", margins=True) # Two-way proportion tabledf["col\_1"].value\_counts() # Frequency of each category  pd.crosstab(df["col\_1"], df["col\_2"],  normalize="all", margins=True) # Two-way proportion table |
| 1. Resampling | df\_ts = df.set\_index("date") # Set datetime index  df\_ts.resample("M").sum().head() # Monthly totals  (df\_ts.resample("H").ffill() / 24).head(72) # Hourly forward-fill then scale |
| 1. Index number by group | df["group\_id"] = df.groupby("group\_col").cumcount() + 1 # 1-based counter |
| 1. Leading and lagging variables | df["lag\_1"] = df["value"].shift(1) # Previous value  df["lead\_1"] = df["value"].shift(-1) # Next value  df["pct\_change\_%"] = df["value"].pct\_change() \* 100 # % change |
| 1. Rolling and cumulative aggregations | df["rolling\_mean\_7"] = df["value"].rolling(7).mean() # 7-period mean  df["cum\_sum"] = df["value"].cumsum() # Running total  df["cum\_mean"] = df["value"].expanding().mean() # Running average  df["rolling\_mean\_7"] = df["value"].rolling(7,  min\_periods=1).mean() # Rolling w/ min |
| 1. Conditionally format plots | sns.scatterplot(data=df, x="col\_1", y="col\_2",  hue="group\_col", alpha=.7) # Scatter w/ hue  sns.barplot(data=df\_mean, x="cat\_col", y="metric",  palette="Blues\_r") # Mean bar chart |
| 1. Pairplot | NEED TO FINALIZE |
| 1. Jitterplot | sns.stripplot(data=df, x="cat\_col\_1", y="metric") # Simple strip  sns.stripplot(data=df, x="cat\_col\_1", y="metric",  hue="cat\_col\_2", jitter=.2) # Strip w/ jitter + hue  plt.title("Metric by Category 1 and Category 2") # Title  plt.legend(title="Category 2", bbox\_to\_anchor=(1.05, 1),  loc="upper left") |
| 1. Pairplot | g = sns.FacetGrid(data=df, col="facet\_col") # Column facets  g.map(sns.scatterplot, "col\_1", "col\_2") # Map scatter  g = sns.FacetGrid(data=df, col="facet\_col",  row="row\_col", height=3) # Row + col facets  g.map(sns.boxplot, "col\_1") # Map boxplot  g.set\_titles(col\_template="{col\_name}",  row\_template="{row\_name}") # Custom titles  g.set\_axis\_labels("", "col\_1") # Y-label only  g.fig.set\_size\_inches(12, 6) # Figure size  g.fig.subplots\_adjust(wspace=0.2, hspace=0.3) # Spacing |
| 1. Jointplot | Finalize this later |
| 1. Bubbleplot | sns.scatterplot(data=df, x="col\_1", y="metric",  hue="group\_col", size="size\_col", alpha=.6) # Bubble scatter  sns.regplot(data=df, x="col\_1", y="metric",  scatter=False, color="black") # Trend line  plt.title("Metric vs col\_1 by Group & Size", pad=20) # Plot title  plt.xlabel("col\_1") # X-axis label  plt.ylabel("metric") |