

Task 1 & 2: Load CSV File and Inspect Data

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings("ignore")

# Set display options
pd.set_option('display.max_columns', 200)
pd.set_option('display.max_rows', 100)
plt.ioctl set 'backend':'Agg')
sns.set_palette("muted")

print("Libraries imported successfully")

libraries imported successfully!

In [3]: # Load the CSV file
csv_path = ('/KCU_2022_1117_1219 DataWithToolInfo.csv',
            'KCU_2022_1118_1217 DataWithToolInfo.csv')

df_list = []
for path in csv_paths:
    df_part = pd.read_csv(path,
                           sep=';',
                           parse_dates=['timestamp_skv', 'timestamp_sensor', 'timestamp', 'timestamp.1', 'timestamp.2'],
                           dtype={'timestamp':True})
    df_list.append(df_part)
df = pd.concat(df_list, ignore_index=True)
print(f"Data loaded successfully!")
print(f"Shape: {df.shape(0)}, rows > {df.shape(1)} columns")
print(f"Columns: {df.columns}")
df.head()
✓ Dataset loaded successfully!
Shape: 173,863 rows x 165 columns
First few rows:
```

	timestamp_skv	timestamp_skf	timestamp_sensor	timestamp_sensorf	SynchroM	ST_CH1_P_F0	ST_CH1_P_F1	ST_CH1_P_F2	ST_CH1_P_F3	ST_CH1_P_F4	ST_CH1_P_F5	ST_CH1_P_F6	ST_CH1_P_F7	ST_CH1_P_F8	ST_CH1_P_F9	ST_CH1_P_F10	ST_CH1_P_F11	ST_CH1_P_Time	ST_CH1_P_F0	ST_CH1_P_F2	ST_CH1_P_F3	ST_CH1_P_F4	ST_CH1_P_F5	ST_CH1_P_F6	ST_CH1_P_F7	ST_CH1_P_F8	ST_CH1_P_F9	ST_CH1_P_F10	ST_CH1_P_F11	ST_C
0	2022-11-17 16:33:49.400	0.0	2022-11-17 16:33:49.506	0.0	0.106	0.001282	0.000004e+00	9.528109e-08	0.067829	0.020091	1.327992e-04	0.000017	0.000033	0.001752	0.000030	0.077239	2.465726e-13	-37.505	0.009124	0.000846	0.000010	1.0	0.020091	0.432020	2.937199e-07	0.001948	0.007690	0.001772	0.063266	1.486056e-11
1	2022-11-17 16:33:50.400	1.0	2022-11-17 16:33:51.006	1.5	0.806	0.001606	4.291534e-06	1.413739e-07	0.061529	0.006128	1.924036e-04	0.000017	0.000035	0.047299	0.000022	0.094005	3.772125e-13	-37.100	0.003805	0.000737	0.000015	1.0	0.036128	0.386472	2.912199e-07	0.002240	0.008178	0.001878	0.070914	1.609171e-11
2	2022-11-17 16:33:52.400	2.0	2022-11-17 16:33:52.506	1.5	0.106	0.001331	4.768372e-07	1.085789e-07	0.060714	0.022012	7.152505e-07	0.000017	0.000035	0.047876	0.000024	0.078773	3.283998e-13	-37.263	0.003268	0.004395	0.000011	1.0	0.022012	0.379498	2.893588e-07	0.002067	0.008178	0.001983	0.067239	1.395800e-11
3	2022-11-17 16:33:53.400	1.0	2022-11-17 16:33:54.006	1.5	0.806	0.001462	0.000004e+00	1.137699e-07	0.053222	0.024878	3.924371e-04	0.000017	0.000035	0.001774	0.000025	0.079028	2.119988e-13	-37.391	0.003290	0.004333	0.000011	1.0	0.024878	0.445509	2.940264e-07	0.002135	0.007690	0.001740	0.064132	1.699517e-11
4	2022-11-17 16:33:55.400	2.0	2022-11-17 16:33:55.506	1.5	0.106	0.001347	0.000004e+00	1.128285e-07	0.062160	0.020152	4.768371e-07	0.000017	0.000037	0.047519	0.000031	0.078518	3.642246e-13	-37.273	0.003251	0.003479	0.000011	1.0	0.020152	0.385418	2.965023e-07	0.002067	0.047543	0.001840	0.063172	1.827556e-11

```
In [4]: # Examine column names
print("Column Names:")
print(len(df.columns))
print(f"Total columns: {len(df.columns)}")

# Group columns by type
timestamp_cols = [col for col in df.columns if 'timestamp' in col.lower()]
sound_ch1_cols = [col for col in df.columns if col.startswith('ST_CH1')]
sound_ch2_cols = [col for col in df.columns if col.startswith('ST_CH2')]
camera_cols = [col for col in df.columns if 'cam' in col]
sensor_cols = [col for col in df.columns if 'sensor' in col]
tool_cols = [col for col in df.columns if col in ['pieces', 'pieces.1', 'pieces.2']]

print(f"Timestamp columns: {len(timestamp_cols)}")
print(f"Sound features CH1: {len(sound_ch1_cols)}")
print(f"Sound features CH2: {len(sound_ch2_cols)}")
print(f"Camera features: {len(camera_cols)}")
print(f"Environmental sensor features: {len(sensor_cols)}")
print(f"Tool data columns: {len(tool_cols)}")

print(f"n = {n}")
print(f"Sample column names by category:")
print(f"Timestamps (first 5): {timestamp_cols[:5]}")
print(f"Sound CH1 (first 10): {sound_ch1_cols[:10]}")
print(f"Sound CH2 (first 10): {sound_ch2_cols[:10]}")
print(f"Camera CH1 (first 10): {camera_cols[:10]}")
print(f"Camera CH2 (first 10): {camera_cols[:10]}")
print(f"Environmental sensor (first 10): {sensor_cols[:10]}")
print(f"Tool data (first 10): {tool_cols[:10]}")

Column Names:
=====
Total columns: 165

Timestamp columns: 5
Sound features CH1: 52
Sound features CH2: 52
Camera features: 48
Environmental sensor features: 3
Tool data columns: 3

Sample column names by category:
Timestamps (first 5): ['timestamp_skv', 'timestamp_sensor', 'timestamp', 'timestamp.1', 'timestamp.2']
Sound CH1 (first 10): ['ST_CH1_P_F0', 'ST_CH1_P_F1', 'ST_CH1_P_F2', 'ST_CH1_P_F3', 'ST_CH1_P_F4', 'ST_CH1_P_F5', 'ST_CH1_P_F6', 'ST_CH1_P_F7', 'ST_CH1_P_F8', 'ST_CH1_P_F9']
Sound CH2 (first 10): ['ST_CH2_P_F0', 'ST_CH2_P_F1', 'ST_CH2_P_F2', 'ST_CH2_P_F3', 'ST_CH2_P_F4', 'ST_CH2_P_F5', 'ST_CH2_P_F6', 'ST_CH2_P_F7', 'ST_CH2_P_F8', 'ST_CH2_P_F9']
Camera (first 10): ['4.313 mm SVW1', '10.604 mm SVW1', '10.604 mm SVW1', '5.74 mm SVW1', '10.74 mm SVW1', '4.36 mm SVW1', '4.40 mm SVW1', '5.36 mm SVW1', '5.32 mm SVW1', '5m assess SVW1']
Sensors: ['Beacon INT0028 - B8E240_R0', 'Beacon INT0028 - B8F240_R0', 'Beacon INT0028 - B8F240_T0P0']
Tool data: ['pieces']

In [5]: df[timestamp_cols]
```

	timestamp_skv	timestamp_sensor	timestamp	timestamp.1	timestamp.2
0	2022-11-17 16:33:49.400	2022-11-17 16:33:49.506	2022-11-17 04:31:39	2022-11-17 04:31:39	2022-11-17 04:31:39
1	2022-11-17 16:33:50.400	2022-11-17 16:33:51.006	2022-11-17 04:31:39	2022-11-17 04:31:39	2022-11-17 04:31:39
2	2022-11-17 16:33:52.400	2022-11-17 16:33:52.506	2022-11-17 04:31:39	2022-11-17 04:31:39	2022-11-17 04:31:39
3	2022-11-17 16:33:53.400	2022-11-17 16:33:54.006	2022-11-17 04:31:39	2022-11-17 04:31:39	2022-11-17 04:31:39
4	2022-11-17 16:33:55.400	2022-11-17 16:33:55.506	2022-11-17 04:31:39	2022-11-17 04:31:39	2022-11-17 04:31:39
...
173858	2023-01-27 17:09:24.400	2023-01-27 17:09:25.261	2023-01-27 05:08:27	2023-01-27 05:08:27	2023-01-27 05:08:27
173859	2023-01-27 17:09:26.400	2023-01-27 17:09:26.625	2023-01-27 05:08:27	2023-01-27 05:08:27	2023-01-27 05:08:27
173860	2023-01-27 17:09:27.400	2023-01-27 17:09:27.968	2023-01-27 05:08:27	2023-01-27 05:08:27	2023-01-27 05:08:27
173861	2023-01-27 17:09:28.400	2023-01-27 17:09:29.352	2023-01-27 05:08:28	2023-01-27 05:08:28	2023-01-27 05:08:28
173862	2023-01-27 17:09:30.400	2023-01-27 17:09:30.796	2023-01-27 05:08:28	2023-01-27 05:08:28	2023-01-27 05:08:28
173863	rows > 5 columns				

```
In [6]: # Check data types and missing values
print("Data Info:")
print(len(df))
print(df.info())

print(f"n = {n}")
print(f"Missing Values Summary:")
missing = df.isnull().sum()
missing_not = missing / len(df) * 100
missing_df = pd.DataFrame({
    'Missing Count': missing[missing > 0],
    'Percentage': missing_not[missing > 0]
})
df[missing_df.columns] = missing_df

if len(missing_df) > 0:
    print(f"Columns with missing values: {len(missing_df)}")
    print(missing_df.head(20))
else:
    print(f"n = {n} No missing values found")

Data Info:
=====
Columns 'pandas.core.frame.DataFrame':
RangeIndex: 173863 entries, 0 to 173862
Columns: 165 entries, timestamp_skv to tool.10
dtypes: datetime64[ns]: 5, float64: 159, int64: 2
memory usage: 218.3 MB
None

Missing Values Summary:
Columns with missing values: 35
Missing Count Percentage
4.36 SVW2 14 0.009203
4.40 SVW2 14 0.009203
5.36 SVW2 14 0.009203
5m assess SVW2 14 0.009203
5.74 SVW2 14 0.009203
5.74 mm SVW2 14 0.009203
0.604 mm SVW2 14 0.009203
0.604 mm SVW2 14 0.009203
4.313 SVW2 14 0.009203
4.36 SVW2 13 0.007477
4.40 SVW3 13 0.007477
4.74 mm SVW3 13 0.007477
4.74 mm SVW3 13 0.007477
0.604 mm SVW3 13 0.007477
0.604 mm SVW3 13 0.007477
5m assess SVW3 13 0.007477
5m assess SVW3 13 0.007477
```

Task 3: Check Timestamps

Firstly check whether the timestamps are equidistant – reflecting a constant rate of production – or whether there are "pause"-intervals! If you find such intervals try to formulate hypotheses to explain these "pauses"!

```
In [7]: timestamp_cols

Out[7]: ['timestamp_skv',
'timestamp_sensor',
'timestamp',
'timestamp.1',
'timestamp.2']
```

```
In [8]: df[timestamp_cols]
```

	timestamp_skv	timestamp_sensor	timestamp	timestamp.1	timestamp.2
0	2022-11-17 16:33:49.400				
1	2022-11-17 16:33:50.400				
2	2022-11-17 16:33:52.400				
3	2022-11-17 16:33:53.400				
4	2022-11-17 16:33:55.400				
...
173858	2023-01-27 17:09:24.400				
173859	2023-01-27 17:09:26.400				
173860	2023-01-27 17:09:27.400				
173861	2023-01-27 17:09:28.400				
173862	2023-01-27 17:09:30.400				
...
173863	rows > 5 columns				

```
In [10]: df[timestamp_cols].plot()
```



```
In [17]: df[timestamp_cols].diff().dt.days

Out[17]: 0 days 13:32:39
0 days 11:13:42
0 days 09:24:24
0 days 10:04:44
0 days 10:23:02
0 days 10:07:08
0 days 10:44:26
0 days 17:04:08
0 days 01:20:56
0 days 01:24:01
0 days 01:20:25
0 days 02:22:30
0 days 01:19:13
0 days 01:02:49
0 days 02:23:07
0 days 14:29:29
0 days 02:46:18
0 days 08:39:23
0 days 01:43:52
0 days 08:35:40
0 days 01:28:21
0 days 20:11:21
0 days 01:56:37
0 days 02:28:44
0 days 01:04:41
0 days 01:09:40
0 days 01:20:07
0 days 01:48:17
0 days 01:21:22
0 days 01:14:01
Name: timestamp_skv, dtype: timedelta64[ns]
```

```
In [18]: df[timestamp_cols].diff().dt.days

Out[18]: Index: [ 8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393],
dtype: timedelta64[ns]
```

```
In [19]: df[timestamp_cols].diff().dt.days

Out[19]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [20]: df[timestamp_cols].diff().dt.days

Out[20]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [21]: df[timestamp_cols].diff().dt.days

Out[21]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [22]: df[timestamp_cols].diff().dt.days

Out[22]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [23]: df[timestamp_cols].diff().dt.days

Out[23]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [24]: df[timestamp_cols].diff().dt.days

Out[24]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [25]: df[timestamp_cols].diff().dt.days

Out[25]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [26]: df[timestamp_cols].diff().dt.days

Out[26]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [27]: df[timestamp_cols].diff().dt.days

Out[27]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [28]: df[timestamp_cols].diff().dt.days

Out[28]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [29]: df[timestamp_cols].diff().dt.days

Out[29]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [30]: df[timestamp_cols].diff().dt.days

Out[30]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [31]: df[timestamp_cols].diff().dt.days

Out[31]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [32]: df[timestamp_cols].diff().dt.days

Out[32]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [33]: df[timestamp_cols].diff().dt.days

Out[33]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 94119, 107140,
115956, 124872, 130932, 133076, 135052, 136461, 138074, 139581, 139878,
168532, 171750, 173393]]]timestamp_skv]
```

```
In [34]: df[timestamp_cols].diff().dt.days

Out[34]: Index: [8432, 10884, 20841, 25833, 35156, 43584, 51793, 56784, 61110,
64623, 69802, 78038, 80289, 85604, 80244, 81457, 9411
```

