```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("alloy_ml_dataset_elemental_features.csv")
df
```

	Solvent	Solute	Potential_ID	Alpha	Mu	Sigma	solvent_density	solvent_bulk	solvent_shear	solute_density	solute_bu
0	Ag	Al	1	0.0	-3	3	10.212	99.94	27.99	2.663	69
1	Ag	Au	2	0.8	-2	2	10.212	99.94	27.99	18.221	145
2	Ag	Au	3	0.0	0	2	10.212	99.94	27.99	18.221	145
3	Ag	Au	1	2.6	-2	5	10.212	99.94	27.99	18.221	145
4	Ag	Co	1	-0.2	3	12	10.212	99.94	27.99	8.900	185
...	...	...	...	...	...	...	...	...	...	...	
422	Zr	Pd	1	-1.2	4	16	6.510	95.00	33.00	11.576	175
423	Zr	Pt	1	-1.3	3	17	6.510	95.00	33.00	20.603	251
424	Zr	Ta	1	1.9	-7	13	6.510	95.00	33.00	16.690	190
425	Zr	Ti	1	0.0	-3	12	6.510	95.00	33.00	4.510	110
426	Zr	W	1	2.8	1	24	6.510	95.00	33.00	19.250	310

427 rows × 12 columns

Next steps: Generate code with df | New interactive sheet

```
from sklearn.preprocessing import LabelEncoder

# For Solvent
le_solvent = LabelEncoder()
df['Solvent_encoded'] = le_solvent.fit_transform(df['Solvent'])
# Show mapping
mapping_solvent = dict(zip(le_solvent.classes_, le_solvent.transform(le_solvent.classes_)))
print("Solvent mapping:", mapping_solvent)

# For Solute
le_solute = LabelEncoder()
df['Solute_encoded'] = le_solute.fit_transform(df['Solute'])
# Show mapping
mapping_solute = dict(zip(le_solute.classes_, le_solute.transform(le_solute.classes_)))
print("Solute mapping:", mapping_solute)
```

```
nt64(15), 'V': np.int64(16), 'W': np.int64(17), 'Zr': np.int64(18)}
64(15), 'Re': np.int64(16), 'Si': np.int64(17), 'Sm': np.int64(18), 'Ta': np.int64(19), 'Ti': np.int64(20), 'V': np.int64(21
```

```
df
```

	Potential_ID	Alpha	Mu	Sigma	solvent_density	solvent_bulk	solvent_shear	solute_density	solute_bulk	solute_shear
0	1	0.0	-3	3	10.212	99.94	27.99	2.663	69.93	16.76
1	2	0.8	-2	2	10.212	99.94	27.99	18.221	145.31	19.43
2	3	0.0	0	2	10.212	99.94	27.99	18.221	145.31	19.43
3	1	2.6	-2	5	10.212	99.94	27.99	18.221	145.31	19.43
4	1	-0.2	3	12	10.212	99.94	27.99	8.900	185.00	75.00
...	...	...	...	...	...	...	...	...	...	...
422	1	-1.2	4	16	6.510	95.00	33.00	11.576	175.97	51.58
423	1	-1.3	3	17	6.510	95.00	33.00	20.603	251.60	56.22
424	1	1.9	-7	13	6.510	95.00	33.00	16.690	190.00	75.00
425	1	0.0	-3	12	6.510	95.00	33.00	4.510	110.00	43.00
426	1	2.8	1	24	6.510	95.00	33.00	19.250	310.00	160.00

427 rows × 12 columns

Next steps: Generate code with df New interactive sheet

```
cols = ['Solute_encoded', 'Solvent_encoded'] + [c for c in df.columns
                                                if c not in ['Solute_encoded','Solvent_encoded']]
df = df[cols]
df
```

	Solute_encoded	Solvent_encoded	Potential_ID	Alpha	Mu	Sigma	solvent_density	solvent_bulk	solvent_shear	solute_d
0	1	0	1	0.0	-3	3	10.212	99.94	27.99	
1	2	0	2	0.8	-2	2	10.212	99.94	27.99	
2	2	0	3	0.0	0	2	10.212	99.94	27.99	
3	2	0	1	2.6	-2	5	10.212	99.94	27.99	
4	3	0	1	-0.2	3	12	10.212	99.94	27.99	
...	...	...	...	...	...	...	...	...	...	
422	14	18	1	-1.2	4	16	6.510	95.00	33.00	
423	15	18	1	-1.3	3	17	6.510	95.00	33.00	
424	19	18	1	1.9	-7	13	6.510	95.00	33.00	
425	20	18	1	0.0	-3	12	6.510	95.00	33.00	
426	22	18	1	2.8	1	24	6.510	95.00	33.00	

427 rows × 12 columns

Next steps: Generate code with df New interactive sheet

```
df.describe()
```

	Solute_encoded	Solvent_encoded	Potential_ID	Alpha	Mu	Sigma	solvent_density	solvent_bulk	solv
count	427.000000	427.000000	427.000000	427.000000	427.000000	427.000000	427.000000	427.000000	
mean	9.770492	7.812646	10.604215	-0.406557	-6.159251	20.615925	9.872267	155.284731	
std	7.172507	5.355225	15.760540	1.652059	20.898546	17.871412	5.398244	66.296154	
min	0.000000	0.000000	1.000000	-3.300000	-129.000000	1.000000	1.740000	35.000000	
25%	4.000000	3.000000	1.000000	-1.700000	-7.000000	9.000000	6.510000	99.940000	
50%	9.000000	7.000000	1.000000	-1.000000	-2.000000	16.000000	8.900000	160.000000	
75%	15.000000	12.000000	15.500000	1.000000	2.000000	27.000000	11.576000	180.000000	
max	24.000000	18.000000	57.000000	3.600000	88.000000	151.000000	21.040000	350.000000	

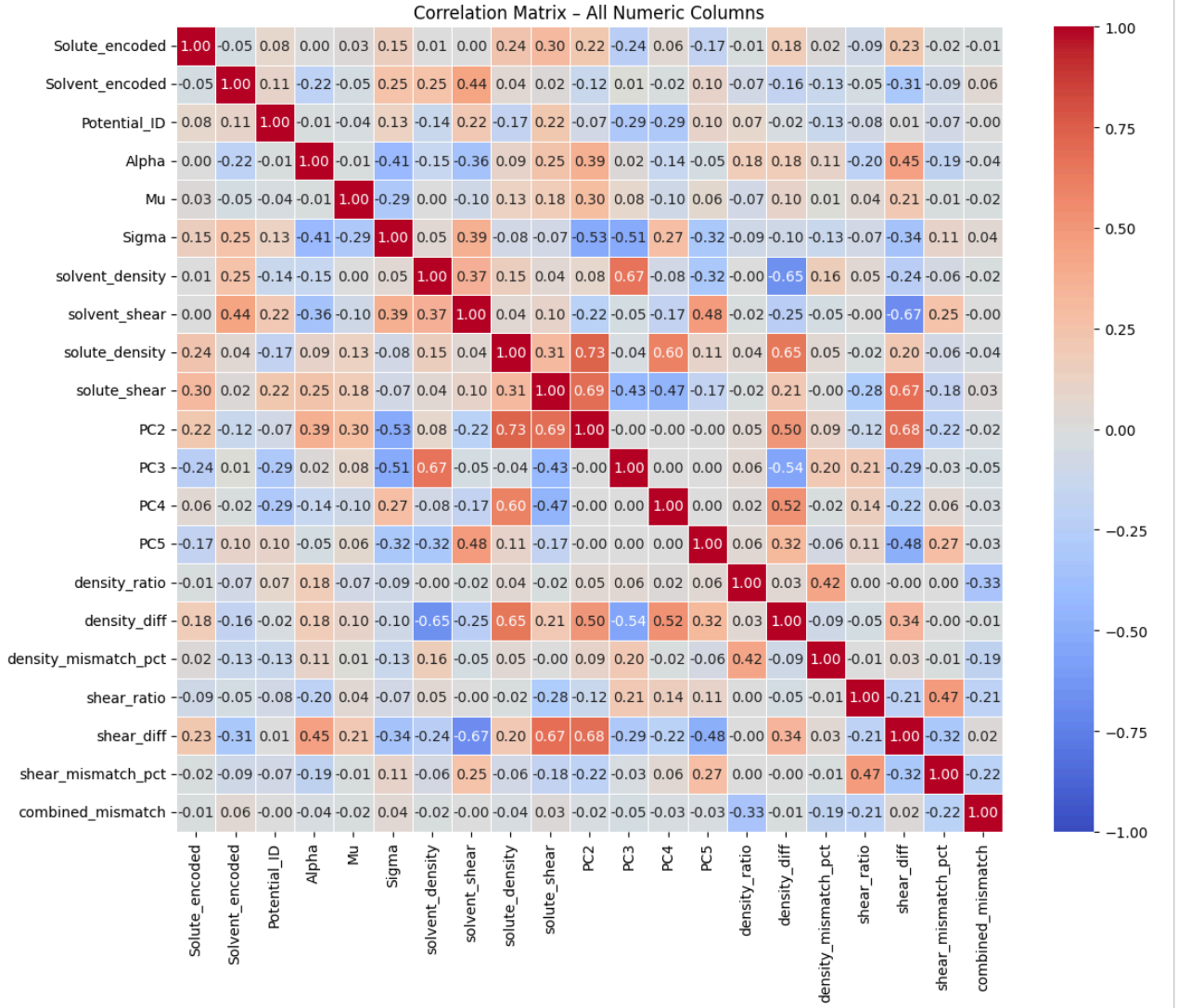
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


# Step 1: Ensure your encoded columns exist (e.g., solute_encoded, solvent_encoded)

# Step 2: Select only numeric columns
numeric_df = df.select_dtypes(include='number')

# Step 3: Compute correlation matrix
corr_matrix = numeric_df.corr()

# Step 4: Plot the heatmap
plt.figure(figsize=(12, 10))                    # adjust size as needed
sns.heatmap(
    corr_matrix,
    vmin=-1,
    vmax=1,
    center=0,
    cmap='coolwarm',                             # diverging palette
    annot=True,
    fmt=".2f",
    linewidths=0.5
)
plt.title("Correlation Matrix – All Numeric Columns")
plt.tight_layout()
plt.show()
```



```
from sklearn.preprocessing import StandardScaler

# columns to exclude from standardization
exclude_cols = ['Solute_encoded', 'Solvent_encoded', 'Potential_ID', 'Alpha', 'Mu']

# columns to scale
cols_to_scale = [col for col in df.columns if col not in exclude_cols]

# initialize scaler
scaler = StandardScaler()

# fit and transform only selected columns
df[cols_to_scale] = scaler.fit_transform(df[cols_to_scale])

# verify
df
```

	Solute_encoded	Solvent_encoded	Potential_ID	Alpha	Mu	Sigma	solvent_density	solvent_shear	solute_density	sol
0	1	0	1	0.0	-3	-0.986860	0.063008	-0.872721	-1.355699	
1	2	0	2	0.8	-2	-1.042881	0.063008	-0.872721	1.580576	
2	2	0	3	0.0	0	-1.042881	0.063008	-0.872721	1.580576	
3	2	0	1	2.6	-2	-0.874819	0.063008	-0.872721	1.580576	
4	3	0	1	-0.2	3	-0.482672	0.063008	-0.872721	-0.178585	
...	...	...	...	...	...	...	...	...	...	
422	14	18	1	-1.2	4	-0.258588	-0.623575	-0.739291	0.326459	
423	15	18	1	-1.3	3	-0.202567	-0.623575	-0.739291	2.030133	
424	19	18	1	1.9	-7	-0.426651	-0.623575	-0.739291	1.291629	
425	20	18	1	0.0	-3	-0.482672	-0.623575	-0.739291	-1.007113	
426	22	18	1	2.8	1	0.189579	-0.623575	-0.739291	1.774781	

427 rows × 10 columns

Next steps: Generate code with df | New interactive sheet

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# 1. Exclude non-numeric or categorical columns
exclude_cols = ['Solute_encoded', 'Solvent_encoded', 'Potential_ID', 'Alpha', 'Mu']
cols_to_scale = [col for col in df.columns if col not in exclude_cols]

# 2. Standardize the selected columns
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df[cols_to_scale])

# 3. Apply PCA (say, retain 95% of variance)
pca = PCA(n_components=0.95)
pca_features = pca.fit_transform(scaled_features)

# 4. Create a new dataframe with principal components
pca_df = pd.DataFrame(pca_features, columns=[f'PC{i+1}' for i in range(pca_features.shape[1])])

# 5. Add back excluded columns
final_df = pd.concat([df[exclude_cols].reset_index(drop=True), pca_df], axis=1)

# 6. Check results
print(final_df.head())
print(f"Explained variance ratio (sum): {pca.explained_variance_ratio_.sum():.2f}")
```

```
   Solute_encoded  Solvent_encoded  Potential_ID  Alpha  Mu       PC1  \
0               1                0             1    0.0  -3 -1.536528
1               2                0             2    0.8  -2 -0.832481
2               2                0             3    0.0   0 -0.832481
3               2                0             1    2.6  -2 -0.760849
4               3                0             1   -0.2   3 -0.704254

        PC2       PC3       PC4       PC5
0 -0.903554  1.181172 -0.486379 -0.105954
1  1.013623  1.063183  1.569850  0.359261
2  1.013623  1.063183  1.569850  0.359261
3  0.937648  0.973164  1.624331  0.280078
4  0.515778  0.170779 -0.337539 -0.533925
Explained variance ratio (sum): 1.00
```

```
df
```

	Solute_encoded	Solvent_encoded	Potential_ID	Alpha	Mu	Sigma	solvent_density	solvent_shear	solute_density	sol
0	1	0	1	0.0	-3	-0.986860	0.063008	-0.872721	-1.355699	
1	2	0	2	0.8	-2	-1.042881	0.063008	-0.872721	1.580576	
2	2	0	3	0.0	0	-1.042881	0.063008	-0.872721	1.580576	
3	2	0	1	2.6	-2	-0.874819	0.063008	-0.872721	1.580576	
4	3	0	1	-0.2	3	-0.482672	0.063008	-0.872721	-0.178585	
...	...	...	...	...	...	...	...	...	...	
422	14	18	1	-1.2	4	-0.258588	-0.623575	-0.739291	0.326459	
423	15	18	1	-1.3	3	-0.202567	-0.623575	-0.739291	2.030133	
424	19	18	1	1.9	-7	-0.426651	-0.623575	-0.739291	1.291629	
425	20	18	1	0.0	-3	-0.482672	-0.623575	-0.739291	-1.007113	
426	22	18	1	2.8	1	0.189579	-0.623575	-0.739291	1.774781	

427 rows × 15 columns

Next steps: Generate code with df | New interactive sheet

```
df.drop(columns=['PC1'], inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipython-input-2750509230.py in <cell line: 0>()
----> 1 df.drop(columns=['PC1'], inplace=True)
      2 df

 3 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
   7068         if mask.any():
   7069             if errors != "ignore":
-> 7070                 raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7071             indexer = indexer[~mask]
   7072         return self.delete(indexer)

KeyError: "['PC1'] not found in axis"
```

Next steps: Explain error

```
df
```

	Solute_encoded	Solvent_encoded	Potential_ID	Alpha	Mu	Sigma	solvent_density	solvent_shear	solute_density	sol
0	1	0	1	0.0	-3	-0.986860	0.063008	-0.872721	-1.355699	
1	2	0	2	0.8	-2	-1.042881	0.063008	-0.872721	1.580576	
2	2	0	3	0.0	0	-1.042881	0.063008	-0.872721	1.580576	
3	2	0	1	2.6	-2	-0.874819	0.063008	-0.872721	1.580576	
4	3	0	1	-0.2	3	-0.482672	0.063008	-0.872721	-0.178585	
...	...	...	...	...	...	...	...	...	...	
422	14	18	1	-1.2	4	-0.258588	-0.623575	-0.739291	0.326459	
423	15	18	1	-1.3	3	-0.202567	-0.623575	-0.739291	2.030133	
424	19	18	1	1.9	-7	-0.426651	-0.623575	-0.739291	1.291629	
425	20	18	1	0.0	-3	-0.482672	-0.623575	-0.739291	-1.007113	
426	22	18	1	2.8	1	0.189579	-0.623575	-0.739291	1.774781	

427 rows × 14 columns

Next steps: Generate code with df | New interactive sheet

Feature	Formula	Interpretation
Density ratio	`p_solute / p_solvent`	How dense the solute is compared to solvent (captures siz

Feature	Formula	Interpretation
Density difference	`ρ_solute - ρ_solvent`	Absolute mismatch in atomic packing density.
Density mismatch %	`abs(ρ_solute - ρ_solvent) / ρ_solvent`	Relative density mismatch — useful for segregation energy
Shear modulus ratio	`G_solute / G_solvent`	Indicates stiffness contrast — soft solutes in stiff matrices
Shear modulus difference	`G_solute - G_solvent`	Absolute elastic mismatch.
Shear modulus mismatch %	`abs(G_solute - G_solvent) / G_solvent`	Fractional stiffness mismatch — often correlates with inter
Elastic energy mismatch	`(G_solute - G_solvent)**2`	Quadratic mismatch — penalizes large differences.
Combined mismatch factor	`((ρ_solute - ρ_solvent)/ρ_solvent) * ((G_solute - G_solvent)/G_solvent)`	Captures combined structural and mechanical incompatib

```
df['density_ratio'] = df['solute_density'] / df['solvent_density']
df['density_diff'] = df['solute_density'] - df['solvent_density']
df['density_mismatch_pct'] = abs(df['solute_density'] - df['solvent_density']) / df['solvent_density']

df['shear_ratio'] = df['solute_shear'] / df['solvent_shear']
df['shear_diff'] = df['solute_shear'] - df['solvent_shear']
df['shear_mismatch_pct'] = abs(df['solute_shear'] - df['solvent_shear']) / df['solvent_shear']

df['combined_mismatch'] = ((df['solute_density'] - df['solvent_density']) / df['solvent_density']) * \
                          ((df['solute_shear'] - df['solvent_shear']) / df['solvent_shear'])
df
```

	Solute_encoded	Solvent_encoded	Potential_ID	Alpha	Mu	Sigma	solvent_density	solvent_shear	solute_density	sol
0	1	0	1	0.0	-3	-0.986860	0.063008	-0.872721	-1.355699	
1	2	0	2	0.8	-2	-1.042881	0.063008	-0.872721	1.580576	
2	2	0	3	0.0	0	-1.042881	0.063008	-0.872721	1.580576	
3	2	0	1	2.6	-2	-0.874819	0.063008	-0.872721	1.580576	
4	3	0	1	-0.2	3	-0.482672	0.063008	-0.872721	-0.178585	
...	...	...	...	...	...	...	...	...	...	
422	14	18	1	-1.2	4	-0.258588	-0.623575	-0.739291	0.326459	
423	15	18	1	-1.3	3	-0.202567	-0.623575	-0.739291	2.030133	
424	19	18	1	1.9	-7	-0.426651	-0.623575	-0.739291	1.291629	
425	20	18	1	0.0	-3	-0.482672	-0.623575	-0.739291	-1.007113	
426	22	18	1	2.8	1	0.189579	-0.623575	-0.739291	1.774781	

427 rows × 21 columns