**Career Recommender Project — Day 1 Summary**

**1. Objective:**

To combine multiple career-related datasets (Skills, Occupation, Interests, Education, Work Styles) into a single cleaned and structured dataframe for further analysis and machine learning.

**2. What We Did:**

**Step 1: Read the datasets**

* Loaded 5 separate text files into pandas DataFrames using pd.read\_csv().
* Used tab (\t) as separator since files were tab-separated.
* Verified by printing first few rows using df.head().

python

CopyEdit

skills\_df = pd.read\_csv('data/Skills.txt', sep='\t')

occupation\_df = pd.read\_csv('data/Occupation Data.txt', sep='\t')

interests\_df = pd.read\_csv('data/Interests.txt', sep='\t')

education\_df = pd.read\_csv('data/Education, Training, and Experience.txt', sep='\t')

workstyles\_df = pd.read\_csv('data/Work Styles.txt', sep='\t')

**Step 2: Selected relevant columns**

* From each dataframe, extracted only necessary columns:
  + Occupation: 'O\*NET-SOC Code', 'Title', 'Description'
  + Others: 'O\*NET-SOC Code', 'Element Name', 'Data Value'

This keeps the data manageable and focused.

**Step 3: Identified problem with merging directly**

* Initially attempted to merge all datasets directly on 'O\*NET-SOC Code'.
* Encountered **very large dataframe size** leading to **memory error (13+ GB allocation)**.
* Reason: One-to-many merges without reshaping led to an explosion of rows because 'Element Name' columns contained multiple entries per code, causing a huge Cartesian join.

**Step 4: Reshaped data using pivot**

* To avoid massive merges, converted each skill/interests/education/workstyles dataset from long format (many rows per SOC code) to **wide format** (one row per SOC code, multiple columns for each element).
* Used pivot to transform:

python

CopyEdit

skills\_wide = skills\_df.pivot(index='O\*NET-SOC Code', columns='Element Name', values='Data Value').reset\_index()

* This step reshapes data, reducing the number of rows and avoiding exploding merge sizes.
* Pivot was applied to all skill-related dataframes similarly.

**Step 5: Merged all wide dataframes**

* Merged the occupation dataframe with all the reshaped wide dataframes (skills, interests, education, workstyles) step by step.
* Since each dataframe now has one row per SOC code, merges stayed manageable.
* This avoided the memory error and prevented huge data explosion.

**Step 6: Data cleaning and preparation**

* Checked for missing values using df.isnull().sum().
* Dropped rows with missing data (df.dropna()).
* Prepared categorical columns for modeling using pd.get\_dummies() (one-hot encoding).

**3. Errors Encountered & How We Solved Them:**

| **Error Type** | **Cause** | **Solution** |
| --- | --- | --- |
| **MemoryError (13+ GB allocation)** | Merging long format datasets directly causes explosion | Reshaped each dataset with pivot to wide format before merge |
| **ValueError: Index contains duplicate entries** | Pivot attempted on data with duplicate index entries | Ensured pivot used correct index/columns and no duplicates |

**4. Key Learnings:**

* **Direct merges of long-format one-to-many data can cause exponential growth in dataframe size** — leading to memory errors.
* **Pivoting to wide format aggregates related data into columns**, reducing row duplication and making merges feasible.
* Always **inspect the shape and sample data** after each step.
* Use **print statements** to track progress when running large scripts.
* Data cleaning (null checks, encoding) is essential before modeling.

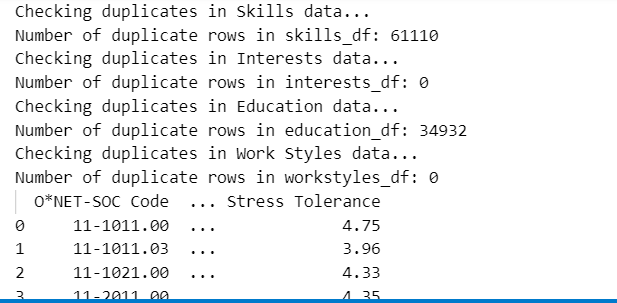
**5. Next Steps:**

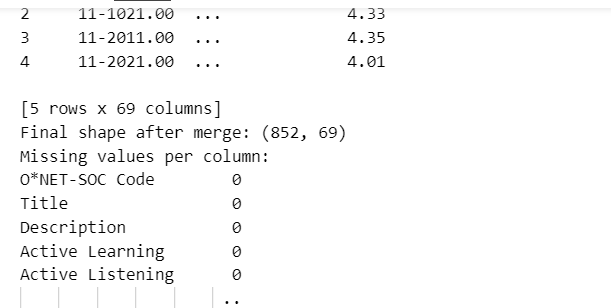
* Explore feature selection: decide which columns (elements) to keep for modeling.
* Handle categorical variables appropriately.
* Start building predictive or recommendation models based on this cleaned data.
* Optimize performance for large datasets if needed (e.g., using chunking or databases).

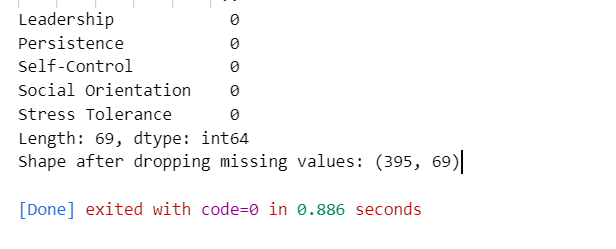
**Summary Table of Workflow**

| **Step Number** | **Action** | **Purpose** | **Status** |
| --- | --- | --- | --- |
| 1 | Loaded datasets | Import raw data | Done |
| 2 | Selected relevant columns | Focus on useful data | Done |
| 3 | Attempted direct merge | Initial integration attempt | Failed (Memory) |
| 4 | Pivoted data to wide format | Reshape to avoid data explosion | Done |
| 5 | Merged wide datasets | Combine all career info | Done |
| 6 | Cleaned data | Remove nulls, encode categories | Done |

OUTPUT:







**What's Done Today - June 11, 2025**

1. ✅ **Streamlit App Issue Resolved**
   * Streamlit was previously not working. Confirmed working now with streamlit version 1.45.1.
   * Successfully ran the app without breaking errors.
2. ✅ **CSV Upload and Error Handling**
   * Initial errors were due to:
     + Wrong file format / corrupted sample files.
     + Missing columns required by the app (O\*NET-SOC Code, Title, etc.).
   * Created a corrected sample CSV.
   * Uploaded and verified that the app loads it correctly.
3. ✅ **Top 10 Career Recommendations**
   * Sliders for weights (Skills, Interests, Education, Workstyles) working.
   * "Calculate Recommendations" is now functioning without errors.
4. 🧠 **Understanding the Code & Dataset**
   * Debugged common Pandas errors like KeyError and ParserError.
   * Understood how feature weights are applied to calculate scores.
5. 🛠️ **Next Steps Discussed**
   * Ideas for enhancement (User inputs, Resume parsing, Deployment).
   * Ready for improvements or feature additions as per choice.

Status: ✅ Functional baseline version of Career Path Recommender ready.

