Term Project: Is AI taking our jobs or transforming them?
Lana Geissinger
Bellevue University
DSC540_T303 Data Preparation (2257-1)
Professor Catherine Williams
Milestone 5
August 10 2025

# Merging the Data and Storing in a Database/Visualizing Data

```
import pandas as pd
import json
from dotenv import load_dotenv
import sqlite3
import duckdb
```

```
# Create a connection to SQLite database
conn = sqlite3.connect('../data/job_analysis.db')
```

Data Sources for Database Import:

## Milestone 2 Files

- **SOC_DB.csv**: Standard Occupational Classification database
- **skills_upd.csv**: O*NET skills database
- In Milestone 5, I shifted the focus from occupations to skills since the BLS API no longer provided the needed job data. This change allowed for a deeper look at how AI affects specific skills instead of broad job categories.
  - Source: O*NET Skills Database (https://www.onetcenter.org/dictionary/29.3/excel/skills.html)
  - New transformations applied: Cleaned column names by removing special characters and whitespace, and converting all names to lowercase.

## Milestone 3 Files

- **Growing_Declining.csv**:
  employment trends
- This file was created by combining and modifying two HTML tables: "Fastest declining occupations" table and "Fastest growing occupations" table.
  - Additional transformations applied (Missed in Milestone3): cleaned columns names by removing special characters and replacing commas with underscores, and removing "Total" summary row.

## Milestone 4 Files

- **occupation_details_upd.json**: Detailed occupation information

```
DATA_PATHS = {
    'soc': '../output/SOC_DB.csv', # flat file - Milestone 2
    'employment': '../output/Growing_Declining.csv', # csv created from HTML tables
    'occupation_details': '../api_responses/occupation_details_upd.json',
    'skills': '../output/skills_upd.csv' # flat file - Milestone 2 (added skills to M2 fro
m M5 for consistency in skill grouping)
}
```

```
# Function to create SQLite database with proper tables
def create_database(db_path='../data/job_analysis.db'):
    try:
        conn = sqlite3.connect(db_path)
        print("✅ Database connection established")
        return conn
    except sqlite3.Error as e:
        print(f"❌ Database connection failed: {e}")
        return None
```

```
# Function to load data into database
def load_data(conn, data_paths):
    try:
        soc_df = pd.read_csv(data_paths['soc'])
        employment_df = pd.read_csv(data_paths['employment'])
        skills_df = pd.read_csv(data_paths['skills'])

        with open(data_paths['occupation_details'], 'r') as f:
            occupation_data = json.load(f)
            occupation_df = pd.DataFrame(occupation_data['occupations'])

        # Create tables
        soc_df.to_sql('soc_codes', conn, if_exists='replace', index=False)
        employment_df.to_sql('employment_trends', conn, if_exists='replace', index=False)
        occupation_df.to_sql('occupation_details', conn, if_exists='replace', index=False)
        skills_df.to_sql('skills', conn, if_exists='replace', index=False)

        print("✅ All data loaded successfully")
        return True
    except Exception as e:
        print(f"❌ Error loading data: {e}")
        return False
```

```python
# Function to display column names for each table loaded in the database
def show_table_columns(conn):
    query = """
        SELECT name, type
        FROM sqlite_master
        WHERE type='table'
    """
    try:
        tables = pd.read_sql_query(query, conn)
        for table in tables['name']:
            print(f"\n📄 Columns in table '{table}':")
            df = pd.read_sql_query(f"SELECT * FROM {table} LIMIT 1", conn)
            print(df.columns.tolist())
    except Exception as e:
        print(f"❌ Error retrieving columns: {e}")
```

```python
# Call function to confirm connection, data load and check column names before joining dat
asets together
conn = create_database()
load_data(conn, DATA_PATHS)
show_table_columns(conn)
```

```
✅ Database connection established
✅ All data loaded successfully

📄 Columns in table 'occupation_skills':
['occupation_title', 'employment_trend', 'onet_soc_code', 'soc_code', 'occupa
tion_description', 'extracted_skills']

📄 Columns in table 'soc_codes':
['major_group', 'minor_group', 'broad_group', 'detailed_occupation', 'occupat
ion_title']

📄 Columns in table 'employment_trends':
['2023_national_employment_matrix_title', '2023_national_employment_matrix_co
de', 'employment_2023', 'employment_2033', 'employment_change_numeric_20233
3', 'employment_change_percent_202333', 'median_annual_wage_dollars_2024', 'g
rowth_status', 'annual_change_rate', 'occupation_category']

📄 Columns in table 'occupation_details':
['original_title', 'onet_code', 'description', 'job_family', '2023_national_e
mployment_matrix_title', 'growth_status', 'soc_code']

📄 Columns in table 'skills':
['onetsoc_code', 'title', 'element_id', 'element_name', 'scale_id', 'scale_na
me', 'data_value', 'n', 'standard_error', 'lower_ci_bound', 'upper_ci_bound',
'recommend_suppress', 'not_relevant', 'date', 'domain_source']
```

```python
# Add onet_code column to employment_trends table using ALTER TABLE
def add_column_to_table(conn):
    try:
        # Add the column
        conn.execute("""
        ALTER TABLE employment_trends
        ADD COLUMN onet_code TEXT;
        """)

        # Update the column with values from occupation_details
        conn.execute("""
        UPDATE employment_trends
        SET onet_code = (
            SELECT od.onet_code
            FROM occupation_details od
            WHERE od.soc_code = employment_trends."2023_national_employment_matrix_code"
        );
        """)

        conn.commit()
        print("✅ Added column successfully")
    except sqlite3.Error as e:
        print(f"❌ Error adding column: {e}")
```

```python
add_column_to_table(conn)
```

```
✅ Added column successfully
```

```
# Create joins between tables using SQL
def create_job_analysis_view(conn):
    create_view_sql = """
                      CREATE VIEW IF NOT EXISTS job_analysis_view AS
                      SELECT employment_trends."2023_national_employment_matrix_title" AS
occupation_title,
                             employment_trends."2023_national_employment_matrix_code",
                             employment_trends.onet_code,
                             employment_trends.growth_status,
                             employment_trends.employment_2023,
                             employment_trends.employment_2033,
                             employment_trends.median_annual_wage_dollars_2024         AS
median_annual_wage,
                             occupation_details.description,
                             soc_codes.major_group,
                             soc_codes.minor_group,
                             skills.element_name,
                             skills.scale_id,
                             skills.data_value
                      FROM employment_trends
                               LEFT JOIN occupation_details
                                         ON employment_trends."2023_national_employment_ma
trix_code" =
                                            occupation_details.soc_code
                             LEFT JOIN soc_codes
                                         ON employment_trends."2023_national_employment_ma
trix_code" =
                                            soc_codes.detailed_occupation
                             LEFT JOIN skills
                                         ON employment_trends.onet_code = skills.onetsoc_c
ode"""

    try:
        # Drop the view if it exists
        conn.execute("DROP VIEW IF EXISTS job_analysis_view")
        # Create the new view
        conn.execute(create_view_sql)
        conn.commit()
        print("✅ View created successfully")
        return True
    except sqlite3.Error as e:
        print(f"❌ Error creating view: {e}")
        return False


# Connect to database and create view
conn_sqlite = sqlite3.connect('../data/job_analysis.db')
create_job_analysis_view(conn_sqlite)

# Query the view
df = pd.read_sql_query("SELECT * FROM job_analysis_view", conn_sqlite)

# Use DuckDB to run SQL on DataFrames
con_duck = duckdb.connect()
con_duck.register('df', df)
```

```
df_sql1 = con_duck.execute("""
                           SELECT *
                           FROM df
                           WHERE median_annual_wage > 60000 LIMIT 10;
                           """).fetchdf()

print(df_sql1)

# Close connections
conn_sqlite.close()
con_duck.close()
```

✅ View created successfully

```
      occupation_title 2023_national_employment_matrix_code   onet_code  \
0  Roof Bolters, Mining                                47-5043  47-5043.00
1  Roof Bolters, Mining                                47-5043  47-5043.00
2  Roof Bolters, Mining                                47-5043  47-5043.00
3  Roof Bolters, Mining                                47-5043  47-5043.00
4  Roof Bolters, Mining                                47-5043  47-5043.00
5  Roof Bolters, Mining                                47-5043  47-5043.00
6  Roof Bolters, Mining                                47-5043  47-5043.00
7  Roof Bolters, Mining                                47-5043  47-5043.00
8  Roof Bolters, Mining                                47-5043  47-5043.00
9  Roof Bolters, Mining                                47-5043  47-5043.00

  growth_status  employment_2023  employment_2033  median_annual_wage  \
0     Declining              2.0              1.4             76640.0
1     Declining              2.0              1.4             76640.0
2     Declining              2.0              1.4             76640.0
3     Declining              2.0              1.4             76640.0
4     Declining              2.0              1.4             76640.0
5     Declining              2.0              1.4             76640.0
6     Declining              2.0              1.4             76640.0
7     Declining              2.0              1.4             76640.0
8     Declining              2.0              1.4             76640.0
9     Declining              2.0              1.4             76640.0

                                    description major_group minor_group
\
0  Operate machinery to install roof support bolt...     47-0000     47-5000
1  Operate machinery to install roof support bolt...     47-0000     47-5000
2  Operate machinery to install roof support bolt...     47-0000     47-5000
3  Operate machinery to install roof support bolt...     47-0000     47-5000
4  Operate machinery to install roof support bolt...     47-0000     47-5000
5  Operate machinery to install roof support bolt...     47-0000     47-5000
6  Operate machinery to install roof support bolt...     47-0000     47-5000
7  Operate machinery to install roof support bolt...     47-0000     47-5000
8  Operate machinery to install roof support bolt...     47-0000     47-5000
9  Operate machinery to install roof support bolt...     47-0000     47-5000

             element_name scale_id  data_value
0          Active Learning       IM        2.88
1          Active Learning       LV        2.88
2         Active Listening       IM        3.12
3         Active Listening       LV        2.88
4  Complex Problem Solving       IM        3.00
5  Complex Problem Solving       LV        2.38
6             Coordination       IM        3.00
7             Coordination       LV        3.50
8        Critical Thinking       IM        3.38
9        Critical Thinking       LV        3.00
```

```python
def validate_database_setup():

    print("🔍 Validating Database Setup\n")

    try:

        conn = sqlite3.connect('../data/job_analysis.db')
        print("✅ Database connection successful")


        required_tables = ['soc_codes', 'employment_trends', 'occupation_details', 'skill
s']
        cursor = conn.cursor()
        existing_tables = cursor.execute(
            "SELECT name FROM sqlite_master WHERE type='table'"
        ).fetchall()
        existing_tables = [table[0] for table in existing_tables]

        all_tables_exist = True
        for table in required_tables:
            if table in existing_tables:
                row_count = cursor.execute(f"SELECT COUNT(*) FROM {table}").fetchone()[0]
                print(f"✅ {table}: {row_count} records")
            else:
                print(f"❌ Missing table: {table}")
                all_tables_exist = False


        view_exists = cursor.execute(
            "SELECT COUNT(*) FROM sqlite_master WHERE type='view' AND name='job_analysis_v
iew'"
        ).fetchone()[0]

        if view_exists:
            print("✅ job_analysis_view exists")
        else:
            print("❌ job_analysis_view missing")
            all_tables_exist = False

        conn.close()
        return all_tables_exist

    except sqlite3.Error as e:
        print(f"❌ Database validation failed: {e}")
        return False
```

Main function for setting up and validating job analysis database:

1. Creates the database connection
2. Loads initial data from files
3. Adds the onet_code column to employment_trends
4. Updates onet_code values from occupation_details
5. Creates the analysis view
6. Validates the database setup
7. Shows sample data from the view

```
def main():
    print("🚀 Starting Job Analysis Database Setup\n")

    try:

        print("Step 1: Creating database connection...")
        conn = create_database()
        if not conn:
            raise Exception("Failed to create database connection")

        print("\nStep 2: Loading data into tables...")
        if not load_data(conn, DATA_PATHS):
            raise Exception("Failed to load data into tables")

        print("\nStep 3: Adding onet_code column...")
        try:
            conn.execute("""
                         ALTER TABLE employment_trends
                             ADD COLUMN onet_code TEXT;
                         """)
            print("✅ Added onet_code column")
        except sqlite3.OperationalError:
            print("ℹ️ onet_code column already exists")

        print("\nStep 4: Updating onet_code values...")
        update_query = """
                       UPDATE employment_trends
                       SET onet_code = (SELECT occupation_details.onet_code
                                        FROM occupation_details
                                        WHERE occupation_details.soc_code =
                                              employment_trends."2023_national_employment_
matrix_code"
                           LIMIT 1
                           ); \
                       """
        conn.execute(update_query)
        conn.commit()
        print("✅ Updated onet_code values")

        print("\nStep 5: Creating analysis view...")
        try:
            conn.execute("DROP VIEW IF EXISTS job_analysis_view")
            create_job_analysis_view(conn)
            print("✅ Created job_analysis_view")
        except sqlite3.Error as e:
            print(f"❌ Error creating view: {e}")
            raise e

        print("\nStep 6: Validating database setup...")
        success = validate_database_setup()
        if success:
            print("\n✨ Database setup is valid!")
        else:
            print("\n⚠️ Database setup needs attention!")

        print("\nStep 7: Showing sample data from view...")
```

```
        sample_query = """
                       SELECT occupation_title,
                              growth_status,
                              median_annual_wage,
                              major_group,
                              element_name,
                              scale_id,
                              data_value
                       FROM job_analysis_view LIMIT 5; \
                       """
        sample_data = pd.read_sql_query(sample_query, conn)
        print("\nSample data from job_analysis_view:")
        print(sample_data)

        print("\n✅ Database setup completed successfully!")

    except Exception as e:
        print(f"\n❌ Setup failed: {str(e)}")
    finally:
        if 'conn' in locals():
            conn.close()
            print("\nDatabase connection closed")


if __name__ == "__main__":
    main()
```

```
🚀 Starting Job Analysis Database Setup

Step 1: Creating database connection...
✅ Database connection established

Step 2: Loading data into tables...
✅ All data loaded successfully

Step 3: Adding onet_code column...
✅ Added onet_code column

Step 4: Updating onet_code values...
✅ Updated onet_code values

Step 5: Creating analysis view...
✅ View created successfully
✅ Created job_analysis_view

Step 6: Validating database setup...
🔍 Validating Database Setup

✅ Database connection successful
✅ soc_codes: 1444 records
✅ employment_trends: 61 records
✅ occupation_details: 81 records
✅ skills: 61530 records
✅ job_analysis_view exists

✨ Database setup is valid!

Step 7: Showing sample data from view...

Sample data from job_analysis_view:
              occupation_title growth_status  median_annual_wage major_group
\
0  Word Processors And Typists     Declining             47850.0     43-0000
1  Word Processors And Typists     Declining             47850.0     43-0000
2  Word Processors And Typists     Declining             47850.0     43-0000
3  Word Processors And Typists     Declining             47850.0     43-0000
4  Word Processors And Typists     Declining             47850.0     43-0000

              element_name scale_id  data_value
0          Active Learning       IM        2.25
1          Active Learning       LV        2.00
2         Active Listening       IM        3.25
3         Active Listening       LV        3.12
4  Complex Problem Solving       IM        2.00

✅ Database setup completed successfully!

Database connection closed
```

## Ethical Implications Of Merging the Data and Storing in a Database

While merging the data and storing it in a database, I performed the following steps:

- Created SQLite database
- Loaded data from the files into tables
- Combined tables into a single dataset
- Validated the database setup (schema, raw counts, and key fields)

**Ethical Implications:**

Data was collected from publicly available resources, BLS and O*NET,and usesd for research in line with their terms of use.
There is a small risk of losing context when removing special characters or footnotes.
Also, merging sources can create duplicate records, and reframing from occupation to skills may change interpretation. All changes were documented for future reference.