# SocioEconAdopt: SocioEcon-Aware Dog Adoption Recommender

## Milestone 2: Project Outline

Lufei Chen, Sophie Shi, Xilin Chen, Jier Yin

### 1. Motivation for the idea/description of the problem the application solves

Many families choose to adopt pets from shelters rather than buy them from breeders in the U.S. However, adoption decisions are often driven by emotional impulses or appearance rather than data-driven reasoning. Different dog breeds vary widely in size, diet, and costs. For example, large dogs require more food and exercise, leading to higher expenses, while small dogs eat less and cost less overall. Long-haired breeds need regular grooming, whereas short-haired breeds are easier to maintain. These differences raise different breeds and require different levels of financial support, but many adopters are unaware of these gaps before adoption. Some face unexpected burdens or even abandon their pets when costs become unsustainable.

Currently, existing adoption platforms often focus on appearance or basic breed information, while overlooking the "fit" between people and their environment. For example, the ideal breed for urban residents may differ greatly from that for rural families. To address this issue, we propose SocioEconAdopt, a data-driven platform that combines dog adoption records with U.S. Census data. The system analyzes breed traits (size, temperament, coat type) and community socioeconomic factors (income, family size, housing type) to recommend dog breeds suited to each user's living environment. It also provides an interactive map for users, shelters, and policymakers to explore adoption trends and understand how socioeconomic conditions shape adoption behavior. We believe that by integrating breed characteristics with regional socioeconomic data, a data-driven recommendation system can help adopters make choices that better align with their living conditions, thereby improving adoption matches and long-term success rates.

### 2. List of features you will definitely implement in the application

- Dog Recommendation System:

Suggests suitable dog breeds for users based on household and socioeconomic characteristics. The system will integrate adoption data (breed, size, coat, temperament) with Census variables (income, family size, housing type) to calculate and display top matching breeds.

- City-Based Breed Explorer:

An interactive search interface that allows users to explore the most common and popular dog breeds in their selected city or county. It will display local demographic statistics alongside adoption trends, helping users understand how breed preferences vary by region.

- Demographic and Breed Map Visualization:

Interactive U.S. map displaying: The most common dog breeds by county; Demographic and socioeconomic indicators (income, family size, urbanity); Correlations between local conditions and adoption characteristics; Users can hover over a city or county to see breed frequency, demographic statistics, and adoption insights.

- Database Integration and Query Implementation:

An SQL database combining the dog adoption and Census datasets. We will implement complex queries involving joins, aggregations, subqueries, and ranking to support analytical tasks such as identifying the most adoptable breeds in high-income areas or computing adoption rates per 1,000 households.

3. **List of features you might implement in the application, given enough time**

- User Profile and Preference System:

Allows users to log in and out, create personal accounts, save their basic and socioeconomic information, and store previously recommended breeds. This would make the system more personalized and allow users to revisit or update their preferences easily.

- Mobile-Friendly Interface:

Develop a responsive or mobile app version to allow users to browse recommendations and local adoption data on their phones.

- Adoption Cost Calculator

Estimate first-year and lifetime costs (food, grooming, vet, supplies) for each breed based on regional price indices and household income.

- “Compare Breeds” Dashboard

Interactive side-by-side comparison of up to three breeds by cost, size, grooming, lifespan, and popularity in the user's region.

- Local Shelter Finder / Integration

Use the Petfinder API or local shelter CSVs to show real adoptable dogs of recommended breeds near the user’s ZIP code.

4. **List of pages the application will have and a 1-2 sentence description of each page. We expect that the functionality of each page will be meaningfully different from the functionality of the other pages.**

- Home Page

Serves as the introduction to this application, outlining the purpose and mission to promote responsible, data-driven pet adoption. The page includes a brief overview and a search bar that

allows users to quickly navigate to the breed recommendation page.

- Dog Breed Recommendation Page

Provides an interactive interface where users can enter their household and socioeconomic information, such as location, income, and housing type. After submitting their inputs, users receive a list of recommended dog breeds that best fit their lifestyle, along with brief breed descriptions and suitability scores.

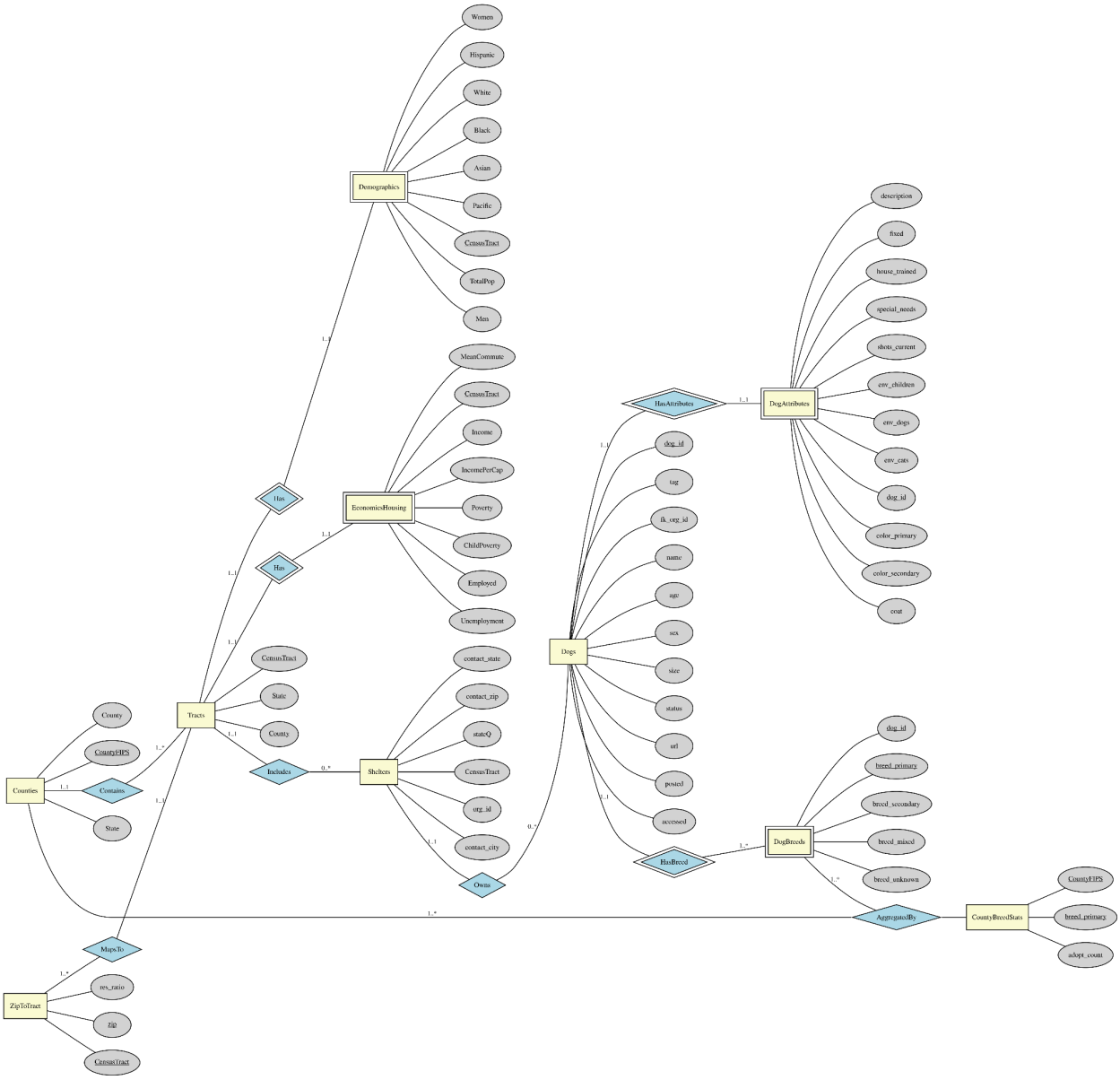
- City-Based  Breed Explorer Page

Allows users to search for a specific city or county to explore local dog adoption trends. The page displays the most commonly adopted breeds in that region along with key demographic and socioeconomic statistics, helping users understand how breed preferences vary across communities.

- Interactive Map Visualization Page

Presents an interactive U.S. map that visualizes adoption and demographic data by county. Users can hover over or click on regions to view detailed statistics such as dominant breeds, median income, and adoption rates, providing a nationwide overview of socioeconomic patterns in dog adoption.

## 5. Relational schema as an ER diagram




## 6. SQL DDL for creating the database

```
CREATE TABLE Tracts (
    CensusTract CHAR(11) PRIMARY KEY,
    State VARCHAR(64),
    County VARCHAR(128)
);

CREATE TABLE Demographics (
```

```
    CensusTract CHAR(11) PRIMARY KEY,
    TotalPop INT,
    Men INT,
    Women INT,
    Hispanic DECIMAL(6,3),
    White DECIMAL(6,3),
    Black DECIMAL(6,3),
    Asian DECIMAL(6,3),
    Pacific DECIMAL(6,3),
    FOREIGN KEY (CensusTract) REFERENCES Tracts(CensusTract)
);

CREATE TABLE EconomicsHousing (
    CensusTract CHAR(11) PRIMARY KEY,
    Income INT,
    IncomePerCap INT,
    Poverty DECIMAL(6,3),
    ChildPoverty DECIMAL(6,3),
    Employed INT,
    Unemployment DECIMAL(6,3),
    MeanCommute DECIMAL(6,3),
    FOREIGN KEY (CensusTract) REFERENCES Tracts(CensusTract)
);

CREATE TABLE Shelters (
    org_id VARCHAR(32) PRIMARY KEY,
    contact_city VARCHAR(100),
    contact_state VARCHAR(10),
    contact_zip VARCHAR(20),
    stateQ VARCHAR(10),
    CensusTract CHAR(11),
```

```
    FOREIGN KEY (CensusTract) REFERENCES Tracts (CensusTract)
);

CREATE TABLE Dogs (
    dog_id BIGINT PRIMARY KEY,
    org_id VARCHAR(32),
    name VARCHAR(255),
    age VARCHAR(50),
    sex VARCHAR(20),
    size VARCHAR(20),
    url VARCHAR(512),
    posted TIMESTAMP,
    FOREIGN KEY (org_id) REFERENCES Shelters(org_id)
);

CREATE TABLE DogBreeds (
    dog_id BIGINT PRIMARY KEY,
    breed_primary VARCHAR(100),
    breed_secondary VARCHAR(100),
    breed_mixed BOOLEAN,
    FOREIGN KEY (dog_id) REFERENCES Dogs(dog_id)
);

CREATE TABLE DogAttributes (
    dog_id BIGINT PRIMARY KEY,
    color_primary VARCHAR(50),
    color_secondary VARCHAR(50),
    coat VARCHAR(50),
    description TEXT,
    fixed BOOLEAN,
    house_trained BOOLEAN,
```

```
    special_needs BOOLEAN,
    shots_current BOOLEAN,
    env_children BOOLEAN,
    FOREIGN KEY (dog_id) REFERENCES Dogs(dog_id)
);
```

7. **Explanation of how you will clean and pre-process the data. This tutorial demonstrates how to do simple pre-processing in Python.**

Our project uses two datasets:
(1) allDogDescriptions.csv, which contains detailed adoption listings and breed attributes from Petfinder; and
(2) acs2017_census_tract_data.csv, which provides socioeconomic indicators from the American Community Survey (ACS) at the census-tract level.

We will perform the following data-cleaning and integration steps before building the database:

1. Clean and Aggregate ACS 2017 Census Data

- Convert data types: Ensure all numeric fields such as Income, Poverty, and Unemployment are correctly parsed as numeric and handle missing values with median imputation.
- Extract county codes: Derive a 5-digit county_fips code from TractId by taking the first five digits, which identify each county.
- Aggregate to county level:
    - Sum population-based measures (TotalPop, Employed, VotingAgeCitizen).
    - Compute population-weighted averages for percentage variables (e.g., Poverty, Income, Unemployment).
- Standardize geography: Keep consistent identifiers (State, County, county_fips) for joining with the adoption data later.

2. Clean Adoption and Breed Data

- Standardize categorical variables:
    - Normalize fields such as age, sex, size, and coat into controlled vocabularies (e.g., Baby/Young/Adult/Senior for age).
    - Map boolean attributes like fixed, house_trained, special_needs, and shots_current to True/False values.
- Clean breed information:
    - Create a unified Breed table using unique breed_primary values.

    - Include secondary breeds when available and mark listings with breed_mixed or breed_unknown flags.
- Parse and format dates: Convert posted and accessed columns into standardized date fields for temporal analysis.
- Clean textual fields: Remove extra whitespace, limit excessively long descriptions, and ensure consistent casing for names and cities.

3. Handling Missing Values

- Numeric fields: Replace missing numeric values in ACS (e.g., Income, Poverty) with state-level medians to preserve realistic variability.
- Categorical fields: Fill missing categorical values (e.g., size, coat, sex) with an "Unknown" category rather than dropping records.
- Boolean fields: Treat missing booleans as False when absence implies "not specified" or "no."
- Geographic gaps: For adoption listings without valid ZIP or county mappings, attempt state-level assignment; if unsuccessful, exclude from spatial analyses but retain for aggregate statistics.
- Documentation: All imputation rules will be recorded in a data-dictionary to maintain transparency.

4. Geographic Linking and Integration

- Map locations: Use contact_zip and contact_state from adoption listings to infer the corresponding county_fips using a ZIP-to-county crosswalk.
- Merge socioeconomic context: Join cleaned adoption data with county-level ACS indicators so that each adoption listing inherits relevant regional features such as Income, Poverty, and Urbanicity.

5. Data Validation and Loading

- Check for missing or invalid county_fips codes and drop rows that cannot be mapped reliably.
- Validate counts (e.g., total population per county) against expected ranges to ensure consistency.
- Load the cleaned data into MySQL tables following the relational schema, preserving referential integrity through foreign keys.

8. **List of technologies you will use. You must use some kind of SQL database. We recommend using MySQL specifically because you will use MySQL in HW2, and we will provide guidance for setting up a MySQL database.**

| Category | Tool / Technology | Purpose |
|---|---|---|
| Database | MySQL | Primary relational database for all joins and queries |
| Frontend | React / Mapbox | Build the user-facing interface and map visualizations |
| Data Processing | Python (Pandas, MySQL) | Data cleaning, ETL, and analysis |
| Visualization | Plotly.js | Trend plots and dashboards |
| Version Control | GitHub | Source control and collaboration |

**9. Description of what each group member will be responsible for**

| Member | Role | Responsibility |
|---|---|---|
| Lufei Chen | Data Engineering | Clean, preprocess, and integrate acs2017_census_tract_data.csv and allDogDescriptions.csv; create SQL database and populate tables. |
| Sophie Shi | Frontend | Design and implement Home, Explorer, and Map pages. |
| Xilin Chen | Backend | Implement endpoints for recommendations and queries; connect backend to MySQL; handle user input and output JSON. |
| Jier Yin | Project Lead & Analysis | Coordinate overall workflow, define recommendation scoring logic, validate results, and prepare final presentation and documentation. |