```
-- Enable UUID extension
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- Create user_profiles table
CREATE TABLE user_profiles (
   id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
   user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
   display_name TEXT,
   location_country TEXT,
   location_city TEXT,
   interests TEXT[],
   reading_level TEXT DEFAULT 'standard' CHECK (reading_level IN ('simple', 'standard',
'advanced')),
   created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
   updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
   UNIQUE(user_id)
);

-- Create news_articles table
CREATE TABLE news_articles (
   id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
   title TEXT NOT NULL,
   original_content TEXT,
   simplified_content TEXT,
   summary TEXT,
   source_name TEXT,
   source_url TEXT,
   author TEXT,
   published_at TIMESTAMP WITH TIME ZONE,
   category TEXT,
   country_code TEXT,
   city TEXT,
   latitude DECIMAL(10, 8),
   longitude DECIMAL(11, 8),
   sentiment_score DECIMAL(3, 2) CHECK (sentiment_score >= -1 AND sentiment_score
<= 1),
   importance_score INTEGER CHECK (importance_score >= 1 AND importance_score <=
10),
   view_count INTEGER DEFAULT 0,
   created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Create user_article_interactions table
CREATE TABLE user_article_interactions (
   id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
   user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
   article_id UUID REFERENCES news_articles(id) ON DELETE CASCADE,
```

```sql
    interaction_type TEXT CHECK (interaction_type IN ('view', 'save', 'share',
'quiz_completed')),
    quiz_score INTEGER,
    time_spent_seconds INTEGER,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    UNIQUE(user_id, article_id, interaction_type)
);

-- Create personal_impacts table
CREATE TABLE personal_impacts (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    article_id UUID REFERENCES news_articles(id) ON DELETE CASCADE,
    impact_type TEXT CHECK (impact_type IN ('financial', 'health', 'travel', 'work', 'lifestyle')),
    description TEXT,
    severity TEXT CHECK (severity IN ('low', 'medium', 'high')),
    affected_demographics TEXT[],
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Create bias_analysis table
CREATE TABLE bias_analysis (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    topic TEXT NOT NULL,
    analyzed_date DATE DEFAULT CURRENT_DATE,
    sources JSONB,
    consensus_summary TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Create api_usage_logs table
CREATE TABLE api_usage_logs (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    api_name TEXT,
    endpoint TEXT,
    requests_count INTEGER DEFAULT 1,
    date DATE DEFAULT CURRENT_DATE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    UNIQUE(api_name, endpoint, date)
);

-- Create indexes for performance
CREATE INDEX idx_articles_published ON news_articles(published_at DESC);
CREATE INDEX idx_articles_category ON news_articles(category);
CREATE INDEX idx_articles_location ON news_articles(country_code, city);
CREATE INDEX idx_user_interactions ON user_article_interactions(user_id, article_id);
CREATE INDEX idx_api_usage_date ON api_usage_logs(date, api_name);

-- Create updated_at trigger function
```

```
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = NOW();
    RETURN NEW;
END;
$$ language 'plpgsql';

-- Create trigger for user_profiles
CREATE TRIGGER update_user_profiles_updated_at BEFORE UPDATE ON user_profiles
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();


-- Enable RLS on all tables
ALTER TABLE user_profiles ENABLE ROW LEVEL SECURITY;
ALTER TABLE news_articles ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_article_interactions ENABLE ROW LEVEL SECURITY;
ALTER TABLE personal_impacts ENABLE ROW LEVEL SECURITY;
ALTER TABLE bias_analysis ENABLE ROW LEVEL SECURITY;
ALTER TABLE api_usage_logs ENABLE ROW LEVEL SECURITY;

-- User profiles policies
CREATE POLICY "Users can view their own profile" ON user_profiles
    FOR SELECT USING (auth.uid() = user_id);

CREATE POLICY "Users can update their own profile" ON user_profiles
    FOR UPDATE USING (auth.uid() = user_id);

CREATE POLICY "Users can insert their own profile" ON user_profiles
    FOR INSERT WITH CHECK (auth.uid() = user_id);

-- News articles policies (public read, admin write)
CREATE POLICY "Anyone can view articles" ON news_articles
    FOR SELECT USING (true);

-- User interactions policies
CREATE POLICY "Users can view their own interactions" ON user_article_interactions
    FOR SELECT USING (auth.uid() = user_id);

CREATE POLICY "Users can create their own interactions" ON user_article_interactions
    FOR INSERT WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Users can update their own interactions" ON user_article_interactions
    FOR UPDATE USING (auth.uid() = user_id);

-- Personal impacts policies (public read)
CREATE POLICY "Anyone can view impacts" ON personal_impacts
    FOR SELECT USING (true);
```

```
-- Bias analysis policies (public read)
CREATE POLICY "Anyone can view bias analysis" ON bias_analysis
   FOR SELECT USING (true);

-- API usage logs (admin only - no public policies)
```