```
# 1. Install dependencies
!pip install fsspec s3fs pyarrow pandas fastparquet

# 2. Import libraries
import pandas as pd

# 3. Define the dataset path (from FSQ OS Places)
# Categories dataset (latest snapshot)
path = "s3://fsq-os-places-us-east-1/release/dt=2025-08-07/categories/parquet/"

# 4. Load FULL dataset (not just head)
df = pd.read_parquet(path, engine="pyarrow")

# 5. Print dataset info
print("Shape of dataset:", df.shape)
print("Columns:", df.columns.tolist())

# 6. Store in GPU RAM (optional check)
# Colab will store it in RAM, not disk by default.
df.head()
```

```
from google.colab import data_table

data_table.enable_dataframe_formatter()

df  # just run this cell
```

1 to 25 of 1245 entries Filter

index	category_id	category_level	category_name	category_label	level1_category_id	level1_category_name	level2_ca
0	63be6904847c3692a84b9c0c	3	Hockey Club	Sports and Recreation > Hockey > Hockey Club	4f4528bc4b90abdf24c9de85	Sports and Recreation	63be6904847c
1	52960eda3cf9994f4e043acc	5	Indonesian Meatball Restaurant	Dining and Drinking > Restaurant > Asian Restaurant > Indonesian Restaurant > Indonesian Meatball Restaurant	63be6904847c3692a84b9bb5	Dining and Drinking	4d4b7105d754
2	4bf58dd8d48988d117951735	3	Candy Store	Retail > Food and Beverage Retail > Candy Store	4d4b7105d754a06378d81259	Retail	4bf58dd8d489
3	56aa371be4b08b9a8d57350e	4	Satay Restaurant	Dining and Drinking > Restaurant > Asian Restaurant > Satay Restaurant	63be6904847c3692a84b9bb5	Dining and Drinking	4d4b7105d754
4	4bf58dd8d48988d189941735	3	Football Stadium	Arts and Entertainment > Stadium > Football Stadium	4d4b7104d754a06370d81259	Arts and Entertainment	4bf58dd8d489
5	56aa371be4b08b9a8d573566	3	Ski Store	Retail > Sporting Goods Retail > Ski Store	4d4b7105d754a06378d81259	Retail	4bf58dd8d489
6	63be6904847c3692a84b9b31	3	Transmissions Shop	Business and Professional Services > Automotive Service > Transmissions	4d4b7105d754a06375d81259	Business and Professional Services	63be6904847c

```
pd.set_option("display.max_rows", 1245)   # show up to 200 rows
pd.set_option("display.max_columns", None)  # show all columns
df
# Save the DataFrame to a CSV file
```

```
df.to_csv('your_data.csv', index=False)

# Download the file to your local machine
from google.colab import files
files.download('your_data.csv')
```

				Medicine > Physician > Geriatric Doctor			
12	4bf58dd8d48988d1fa941735	4	College Lab	Community and Government > Education > College and University > College Lab	63be6904847c3692a84b9b9a	Community and Government	4bf58dd8d4898
13	4bf58dd8d48988d1f06951735	3	Men's Store	Retail > Fashion Retail > Men's Store	4d4b7105d754a06378d81259	Retail	63be6904847c
14	63be6904847c3692a84b9bdd	2	Podiatrist	Health and Medicine > Physician > Podiatrist	63be6904847c3692a84b9bb9	Health and Medicine	63be6904847c
15	4bf58dd8d48988d1ee934735	3	Hostel	Travel and Transportation > Lodging > Hostel	4d4b7105d754a06379d81259	Travel and Transportation	63be6904847c
16	4d954b0ea243a5684a65b473	2	Convenience Store	Retail > Convenience Store	4d4b7105d754a06378d81259	Retail	4d954b0ea243
17	5267e4d9e4b0ec79466e48c5	3	State Park	Travel and Transportation > Outdoors and Recreation > State Park	4d4b7105d754a06373d81259	Travel and Transportation	63be6904847c
18	4bf58dd8d48988d1e4941735	2	Bowling Alley	Arts and Entertainment > Entertainment > Bowling Alley	4d4b7104d754a06370d81259	Arts and Entertainment	4bf58dd8d4898
19	50328912291d4c4b30a586d68	2	Assisted Living	Community and Government > Assisted Living	63be6904847c3692a84b9b9a	Community and Government	50328912291d4
20	4bf58dd8d48988d1f4d941735	4	Paella Restaurant	Dining and Drinking > Restaurant > Spanish Restaurant > Paella Restaurant	63be6904847c3692a84b9bb5	Dining and Drinking	4d4b7105d754
21	4bf58dd8d48988d1f9e941735	4	College Engineering Building	Community and Government > Education > College and University > College Engineering Building	63be6904847c3692a84b9b9a	Community and Government	4bf58dd8d489

```
!pip install transformers torch
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.56.0)
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.8.0+cu126)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.19.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.34
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.22.
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.7
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch) (3.4.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0,>=0.34.
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->transformers)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2.5.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2025
```

```
# ==========================================
# PROPER WORKING CLASSIFICATION CODE
# ==========================================

import pandas as pd
import numpy as np

# ==========================================
# DIVERSE EMOTION MAPPING FOR PLACES
# ==========================================

PLACE_EMOTIONS = {
    # Excitement & Thrill
    'excitement': [
        'Sports and Recreation', 'Event', 'Nightlife Spot',
        'stadium', 'arena', 'sports', 'hockey', 'football', 'baseball', 'basketball',
        'soccer', 'racing', 'gym', 'fitness', 'competition', 'tournament', 'championship',
        'amusement park', 'theme park', 'roller coaster', 'arcade', 'gaming',
        'speedway', 'drag strip', 'motocross', 'skate park', 'bmx track', 'zip line',
        'bungee jumping', 'skydiving', 'parasailing', 'extreme sports', 'action sports','Harbor', 'Port', 'Boat or Ferry', 'Cru
    ],

    # Joy & Happiness
    'joy, happy': [
        'Arts and Entertainment', 'Event',
        'candy', 'ice cream', 'toy', 'playground', 'carnival', 'circus', 'festival',
        'celebration', 'party', 'wedding', 'comedy', 'theater', 'entertainment',
```

```
        'beach', 'water park', 'vacation', 'resort', 'holiday',
        'petting zoo', 'fair', 'boardwalk', 'pier', 'ferris wheel', 'bounce house',
        'birthday party', 'graduation', 'baby shower', 'reunion', 'picnic', 'barbecue','Bowling Alley', 'Laser Tag Center', 'Es
        'Scuba Diving Instructor', 'Swimming', 'Rafting Spot', 'Surf Spot', 'Outdoor Sculpture',
        'Scenic Lookout', 'Running and Track', 'College Track', 'College Cricket Pitch',
        'Jazz and Blues Venue', 'Hot Dog Joint', 'Snack Place', 'Cafe', 'Café', 'Gozleme Place',
        'Wagashi Place', 'Chaat Place', 'Takoyaki Place', 'Lyonese Bouchon', 'Trattoria', 'Pizzeria',
        'Buffet', 'Wings Joint', 'BBQ Joint', 'Creperie', 'Empada House', 'Manti Place', 'Bratwurst Joint',
        'Currywurst Joint', 'Dizi Place', 'Cha Chaan Teng', 'Taverna', 'Pastelaria', 'Botenero','Harbor', 'Port', 'Boat or Ferr
    ],

    # Comfort & Warmth
    'comfort': [
        'Dining and Drinking', 'Community and Government',
        'restaurant', 'cafe', 'bakery', 'diner', 'bistro', 'meatball', 'satay',
        'chinese', 'italian', 'indonesian', 'food', 'home', 'family', 'traditional',
        'cozy', 'neighborhood', 'local', 'comfort food', 'inn', 'cottage',
        'bed and breakfast', 'farmhouse', 'kitchen', 'fireplace', 'porch',
        'grandma house', 'family restaurant', 'mom and pop', 'homestyle'
    ],

    # Relaxation & Peace
    'relaxation,calm': [
        'Landmarks and Outdoors', 'Health and Medicine',
        'spa', 'massage', 'wellness', 'meditation', 'yoga', 'retreat', 'garden',
        'park', 'nature', 'forest', 'lake', 'peaceful', 'quiet', 'serene',
        'library', 'bookstore', 'tea house', 'lounge', 'monastery', 'temple',
        'church', 'chapel', 'sanctuary', 'farm',
        'zen garden', 'botanical garden', 'arboretum', 'hammock', 'cabin',
        'lakeside', 'riverside', 'meadow', 'prairie', 'countryside', 'ashram', 'Spa', 'Sauna', 'Hot Spring', 'Bath House', 'Hot
        'Spiritual Center', 'Botanical Garden', 'Waterfall', 'Island', 'Canal', 'Canal Lock', 'Tree',
        'Hill', 'Lighthouse', 'Lodge', 'Agriturismo', 'Art School', 'Creative Service'
    ],

    # Stress & Anxiety
    'stress': [
        'Health and Medicine', 'Community and Government', 'Business and Professional Services',
        'hospital', 'emergency', 'dental', 'medical', 'surgery', 'clinic', 'doctor',
        'dermatologist', 'urgent care', 'police', 'court', 'legal', 'tax office',
        'government', 'repair', 'mechanic', 'transmission', 'garage', 'crisis',
        'DMV', 'IRS', 'bankruptcy', 'divorce lawyer', 'probation office', 'jail',
        'prison', 'psychiatric ward', 'intensive care', 'waiting room', 'audit'
    ],

    # Social & Connection
    'social': [
        'Dining and Drinking', 'Nightlife Spot', 'Community and Government', 'Arts and Entertainment',
        'bar', 'pub', 'club', 'social', 'community center', 'meeting place',
        'gathering', 'networking', 'coworking', 'plaza', 'town square',
        'market', 'event space', 'conference', 'nightclub', 'karaoke',
        'mixer', 'meetup', 'church social', 'block party', 'neighborhood watch',
        'book club', 'support group', 'social club', 'fraternity', 'sorority'
    ],

    # Adventure & Discovery
    'adventure': [
        'Travel and Transportation', 'Landmarks and Outdoors', 'Arts and Entertainment',
        'museum', 'gallery', 'exhibition', 'discovery', 'exploration', 'tour',
        'travel', 'hiking', 'mountain', 'trail', 'expedition', 'safari',
        'archaeological', 'historic', 'cultural', 'learning', 'educational',
        'ski', 'skiing', 'climbing', 'adventure',
        'cave exploring', 'spelunking', 'wilderness', 'jungle', 'desert',
        'volcano', 'glacier', 'national park', 'observatory', 'planetarium'
    ],

    # Romance & Intimacy
    'romance': [
        'Dining and Drinking',
        'romantic', 'wine bar', 'rooftop', 'sunset', 'candlelight', 'fine dining',
        'intimate', 'date', 'valentine', 'honeymoon', 'boutique hotel', 'vineyard',
        'couples retreat', 'love hotel', 'romantic getaway', 'secluded cabin',
        'moonlight dinner', 'private dining', 'champagne bar', 'jazz lounge'
    ],

    # Luxury & Indulgence
    'luxury': [
        'Dining and Drinking', 'Health and Medicine', 'Retail',
```

```
        'luxury', 'premium', 'high-end', 'exclusive', 'VIP', 'five-star',
        'gourmet', 'boutique', 'designer', 'yacht', 'country club', 'penthouse',
        'michelin star', 'private jet', 'limousine', 'mansion', 'castle',
        'resort spa', 'private island', 'butler service', 'first class'
    ],

    # Shopping & Retail
    'shopping': [
        'Retail',
        'store', 'shop', 'market', 'mall', 'outlet', 'retail', 'boutique',
        'shopping center', 'department store', 'supermarket',
        'farmers market', 'flea market', 'antique shop', 'thrift store',
        'online shopping', 'black friday', 'clearance sale', 'showroom'
    ],

    # Nostalgia & Memory
    'nostalgia': [
        'Landmarks and Outdoors', 'Arts and Entertainment', 'Community and Government',
        'vintage', 'retro', 'classic', 'old-fashioned', 'traditional', 'historic',
        'heritage', 'antique', 'memorial', 'childhood', 'hometown', 'landmark',
        'old school', 'throwback', 'bygone era', 'time capsule', 'museum piece',
        'family photo', 'scrapbook', 'memory lane', 'reunion', 'anniversary'
    ],

    # Energy & Vitality
    'energy': [
        'Sports and Recreation', 'Nightlife Spot', 'Arts and Entertainment',
        'dance', 'music venue', 'concert', 'live music', 'disco', 'rave',
        'workout', 'crossfit', 'spinning', 'aerobics', 'zumba',
        'rock concert', 'mosh pit', 'festival stage', 'nightclub', 'dj booth',
        'dance floor', 'pump up', 'adrenaline', 'high energy', 'electric'
    ],

    # Professional & Achievement
    'professional': [
        'Business and Professional Services', 'Event',
        'office', 'conference room', 'meeting', 'presentation', 'boardroom',
        'networking event', 'business lunch', 'corporate', 'professional',
        'career', 'work', 'success', 'achievement', 'promotion', 'deal closing'
    ],

    # Convenience & Routine
    'convenience': [
        'Travel and Transportation', 'Business and Professional Services', 'Retail',
        'commute', 'daily routine', 'errand', 'necessity', 'practical',
        'efficient', 'quick stop', 'grab and go', 'one-stop shop',
        'convenience store', 'gas station', 'ATM', 'pharmacy', 'post office'
    ],

    # Civic Duty & Responsibility
    'civic': [
        'Community and Government',
        'voting', 'town hall', 'public service', 'citizenship', 'civic duty',
        'community involvement', 'public meeting', 'volunteer', 'charity',
        'social responsibility', 'municipal', 'public works', 'city hall'
    ],

    # Health & Wellness
    'wellness': [
        'Health and Medicine', 'Sports and Recreation',
        'fitness', 'health', 'medical care', 'therapy', 'rehabilitation',
        'prevention', 'check-up', 'treatment', 'healing', 'recovery',
        'mental health', 'physical therapy', 'nutrition', 'diet', 'exercise'
    ],

    # Entertainment & Leisure
    'entertainment': [
        'Arts and Entertainment', 'Event', 'Nightlife Spot',
        'movies', 'shows', 'performance', 'concerts', 'plays', 'exhibitions',
        'entertainment', 'leisure', 'fun', 'recreation', 'cultural events',
        'nightlife', 'dancing', 'music', 'comedy shows', 'festivals'
    ],

    # Exploration & Travel
    'exploration': [
        'Travel and Transportation', 'Landmarks and Outdoors',
        'journey', 'destination', 'sightseeing', 'tourism', 'exploration',
```

```
            'discovery', 'new places', 'adventure travel', 'scenic route',
            'road trip', 'vacation', 'getaway', 'wanderlust', 'exploration'
        ],

        # Fear & Terror
        'fear': [
            'haunted house', 'horror movie', 'scary', 'dark alley', 'cemetery',
            'graveyard', 'abandoned building', 'ghost town', 'creepy',
            'nightmare', 'phobia', 'spider web', 'snake pit', 'cliff edge',
            'deep water', 'storm', 'tornado', 'earthquake zone', 'crime scene'
        ],

        # Melancholy & Sadness
        'melancholy': [
            'funeral home', 'memorial service', 'graveyard', 'rain', 'empty house',
            'abandoned place', 'lonely bench', 'hospital room', 'goodbye',
            'departure lounge', 'train station farewell', 'empty playground',
            'closed business', 'demolition site', 'last call', 'final sale'
        ],

        # Anger & Frustration
        'anger': [
            'traffic jam', 'DMV line', 'customer service', 'complaint department',
            'broken elevator', 'crowded subway', 'parking ticket', 'road rage',
            'protest', 'argument', 'confrontation', 'boxing gym', 'debate stage',
            'complaint booth', 'refund counter', 'cancelled flight', 'overbooked'
        ],

        # Wonder & Awe
        'wonder': [
            'Landmarks and Outdoors', 'Arts and Entertainment',
            'cathedral', 'grand canyon', 'mountain peak', 'ocean view', 'starry sky',
            'northern lights', 'redwood forest', 'ancient ruins', 'architectural marvel',
            'natural wonder', 'breathtaking view', 'majestic', 'spectacular',
            'miraculous', 'inspiring', 'jaw-dropping', 'mind-blowing', 'stunning'
        ],

        # Creativity & Inspiration
        'creativity': [
            'Arts and Entertainment',
            'art studio', 'workshop', 'maker space', 'pottery wheel', 'easel',
            'music studio', 'recording booth', 'rehearsal space', 'writers retreat',
            'brainstorming room', 'innovation lab', 'design studio', 'craft room',
            'inspiration', 'muse', 'creative zone', 'artistic expression', 'gallery opening'
        ],

        # Competition & Rivalry
        'competition': [
            'Sports and Recreation', 'Event',
            'battleground', 'arena fight', 'championship ring', 'playoff game',
            'final match', 'showdown', 'face-off', 'rivalry', 'tournament bracket',
            'elimination round', 'sudden death', 'overtime', 'tiebreaker',
            'head to head', 'challenge', 'contest', 'battle of wills'
        ],

        # Spirituality & Transcendence
        'spirituality': [
            'Community and Government', 'Landmarks and Outdoors',
            'cathedral', 'mosque', 'synagogue', 'temple', 'shrine', 'altar',
            'prayer room', 'meditation hall', 'spiritual retreat', 'pilgrimage',
            'sacred ground', 'holy site', 'enlightenment', 'transcendence',
            'divine', 'mystical', 'otherworldly', 'ethereal', 'cosmic'
        ],

        'education': [
        'College Lab', 'College Engineering Building', 'College Academic Building', 'College Communications Building',
        'College History Building', 'College Theater', 'College Science Building', 'College Library',
        'College Arts Building', 'College Technology Building', 'College Administrative Building',
        'Primary and Secondary School', 'Middle School', 'High School', 'University', 'Community College',
        'Language School', 'Trade School', 'Tutoring Service', 'Laboratory', 'Research Laboratory', 'Medical School'
    ],
        'healthcare': [
        'Dermatologist', 'Geriatric Doctor', 'Podiatrist', 'Mental Health Clinic', 'Obstetrician Gynecologist (Ob-gyn)',
        'Physician', 'Respiratory Doctor', 'Psychiatrist', 'Healthcare Clinic', 'Oncologist', 'Internal Medicine Doctor',
        'Womens Health Clinic', 'Addiction Treatment Center', 'Hospice', 'Nursing Home', 'Emergency Room',
        'Urgent Care Center', 'Acupuncture Clinic', 'Physical Therapy Clinic', 'Medical Lab'
    ],
```

```
        'food': [
        'Indonesian Meatball Restaurant', 'Satay Restaurant', 'Paella Restaurant', 'Pelmeni House', 'Shandong Restaurant',
        'Padangnese Restaurant', 'Colombian Restaurant', 'Turkish Restaurant', 'Kale Pache Place', 'Jiangxi Restaurant',
        'Anhui Restaurant', 'Kokoreç Restaurant', 'Punjabi Restaurant', 'Provençal Restaurant', 'Norman Restaurant',
        'Multicuisine Indian Restaurant', 'Shanxi Restaurant', 'Ligurian Restaurant', 'Donburi Restaurant',
        'Mamak Restaurant', 'Belarusian Restaurant', 'Ouzeri', 'Doner Restaurant', 'Swabian Restaurant',
        'Slovak Restaurant', 'Basque Restaurant', 'Cajun and Creole Restaurant', 'German Restaurant',
        'Southern Brazilian Restaurant', 'Blini House', 'Turkish Home Cooking Restaurant', 'Rajasthani Restaurant',
        'Ukrainian Restaurant', 'Chti Restaurant', 'Tuscan Restaurant', 'Chettinad Restaurant', 'Goiano Restaurant',
        'New American Restaurant', 'Bosnian Restaurant', 'Venezuelan Restaurant', 'Teishoku Restaurant',
        'Italian Restaurant', 'Alsatian Restaurant', 'Macanese Restaurant', 'Moroccan Restaurant', 'Sri Lankan Restaurant',
        'Samgyetang Restaurant', 'Guizhou Restaurant', 'Arepa Restaurant', 'Northeast Indian Restaurant', 'South Indian Restaurant'
        'Japanese Family Restaurant', 'American Restaurant', 'Molise Restaurant', 'Abruzzo Restaurant', 'Basilicata Restaurant',
        'Vegan and Vegetarian Restaurant', 'Poutine Restaurant', 'Udon Restaurant', 'Sardinian Restaurant', 'Breton Restaurant',
        'Palatine Restaurant', 'Campanian Restaurant', 'Calabria Restaurant', 'Lombard Restaurant', 'Fondue Restaurant',
        'Udupi Restaurant', 'Maharashtrian Restaurant', 'Franconian Restaurant', 'Dosa Place', 'Kumru Restaurant',
        'South American Restaurant', 'Empanada Restaurant', 'Beijing Restaurant', 'Eastern European Restaurant',
        'Çöp Şiş Place', 'Yoshoku Restaurant', 'Indonesian Restaurant', 'Huaiyang Restaurant', 'Gukbap Restaurant',
        'Gluten-Free Restaurant', 'Corsican Restaurant', 'Japanese Restaurant', 'Sushi Restaurant', 'Fish Taverna',
        'Russian Restaurant', 'Nabe Restaurant', 'Friterie', 'Taiwanese Restaurant', 'Cuban Restaurant', 'Pachinko Parlor',
        'Tantuni Restaurant', 'Mac and Cheese Joint', 'Mineiro Restaurant', 'Molise Restaurant', 'Japanese Curry Restaurant',
        'Thai Restaurant', 'Ramen Restaurant', 'Burrito Restaurant', 'Poke Restaurant', 'Shabu-Shabu Restaurant',
        'Peking Duck Restaurant', 'Pilavcı', 'Peruvian Roast Chicken Joint', 'Lebanese Restaurant', 'New American Restaurant', 'caf
    ],
        'retail': [
        'Candy Store', 'Men\'s Store', 'Convenience Store', 'Video Store', 'Outlet Mall', 'Bakery', 'Baggage Locker',
        'Farmers Market', 'Swimwear Store', 'Kitchen Supply Store', 'Jewelry Store', 'Fashion Accessories Store',
        'Clothing Store', 'Gift Store', 'Toy Store', 'Organic Grocery', 'Supermarket', 'Furniture and Home Store',
        'Stationery Store', 'Grocery Store', 'Bookstore', 'Souvenir Store', 'Butcher', 'Deli', 'Pet Supplies Store',
        'Hardware Store', 'Appliance Store', 'Liquor Store', 'Ice Cream Parlor', 'Cupcake Shop', 'Pastry Shop',
        'Smoothie Shop', 'Coffee Shop', 'Tea Room', 'Bakery', 'Discount Store', 'Waffle Shop', 'Bagel Shop',
        'Cheese Store', 'Chocolate Store', 'Specialty Grocery', 'Farmers Market'
    ],
        'outdoors': [
        'Hockey Club', 'Football Stadium', 'Baseball', 'Farm', 'College Stadium', 'Mountain', 'Mountain Hut', 'Urban Park',
        'Waterfront', 'National Park', 'Zoo', 'Outdoor Gym', 'Forest', 'Lake', 'Cave', 'Reservoir', 'Volcano',
        'Trail', 'Beach', 'River', 'Playground', 'Stadium', 'Sports Club', 'Golf Course', 'Golf Club', 'Batting Cages',
        'Swimming Pool', 'Ski Resort and Area', 'Ski Trail', 'Ski Lodge', 'Ski Chairlift', 'Basketball', 'Soccer', 'Tennis',
        'Volleyball Court', 'Rugby', 'Hiking Trail', 'Surfing', 'Camping', 'Campground', 'Boat Launch', 'Canoe and Kayak Rental'
    ],
        'entertainment': [
        'Speakeasy', 'Bowling Alley', 'Amphitheater', 'Event Space', 'Indie Movie Theater', 'General Entertainment',
        'Circus School', 'Circus', 'Arcade', 'Music Festival', 'Concert Hall', 'Dance Hall', 'VR Cafe', 'Comedy Club',
        'Multiplex', 'Performing Arts Venue', 'Night Club', 'Theme Restaurant', 'Opera House', 'Club House', 'Ballroom'
    ], 'civic_government': [
        'Town Hall', 'County', 'Community and Government', 'Capitol Building', 'States and Municipalities', 'City Hall',
        'Government Building', 'Government Department', 'Public and Social Service', 'Police Station', 'Courthouse',
        'Embassy or Consulate', 'Voting', 'Town', 'Municipal'
    ], 'fear': [
            'Dentist', 'Cardiologist', 'Pediatrician', 'Neurologist', 'Oral Surgeon', 'Urologist',
            'General Surgeon', 'Anesthesiologist', 'Pathologist', 'Orthopedic Surgeon', 'Plastic Surgeon',
            'Gastroenterologist', 'Radiologist', 'Psychologist', 'Fire Station', 'Ambulance Service',
            'Police Station', 'Immigration Attorney', 'Blood Bank', 'Rescue Service', 'Disabled Persons Service',
            'AIDS Resource', 'Polling Place', 'Military Base'
        ],


    }

    # ==========================================
    # CLASSIFICATION FUNCTIONS
    # ==========================================

    def extract_category_text(row):
        """Extract the best category text from the row"""
        # Column priority order
        columns_to_check = [
            'level6_category_name', 'level5_category_name', 'level4_category_name',
            'level3_category_name', 'level2_category_name', 'level1_category_name'
        ]

        for col in columns_to_check:
            if col in row.index and pd.notna(row[col]) and str(row[col]).strip():
                return str(row[col]).strip()

        return "Unknown"
```

```
def classify_emotion(text):
    """Classify the emotional association of a place"""
    if pd.isna(text) or text == "Unknown" or not text.strip():
        return "neutral"

    text_lower = text.lower().strip()

    # Calculate scores for each emotion
    emotion_scores = {}

    for emotion, keywords in PLACE_EMOTIONS.items():
        score = 0
        for keyword in keywords:
            if keyword.lower() in text_lower:
                # Give more weight to longer, more specific matches
                score += len(keyword.split())
        emotion_scores[emotion] = score

    # Find the emotion with the highest score
    max_score = max(emotion_scores.values())

    if max_score > 0:
        # Return the emotion with highest score
        best_emotion = max(emotion_scores, key=emotion_scores.get)
        return best_emotion

    return "neutral"

# ========================================
# MAIN CLASSIFICATION FUNCTION
# ========================================

def classify_places_with_emotions(dataframe):
    """
    Main function to classify places with emotions
    Returns the modified dataframe
    """

    print("🚀 STARTING PLACE EMOTION CLASSIFICATION")
    print("=" * 60)

    # Make a copy to avoid modifying original
    df_work = dataframe.copy()

    # Step 1: Extract category text
    print("📝 Step 1: Extracting category text from hierarchical columns...")
    df_work['category_text'] = df_work.apply(extract_category_text, axis=1)

    # Show sample of extracted text
    print("Sample extracted texts:")
    sample_texts = df_work['category_text'].head(10).tolist()
    for i, text in enumerate(sample_texts, 1):
        print(f"  {i}. {text}")

    # Step 2: Classify emotions
    print(f"\n🎯 Step 2: Classifying emotions for {len(df_work)} places...")
    df_work['place_emotion'] = df_work['category_text'].apply(classify_emotion)

    # Step 3: Show results
    print(f"\n✅ Classification complete!")

    return df_work

def show_classification_results(df, num_rows=50):
    """Display classification results"""

    if 'place_emotion' not in df.columns:
        print("❌ No 'place_emotion' column found. Run classification first!")
        return

    print(f"\n📊 SHOWING FIRST {num_rows} CLASSIFICATION RESULTS")
    print("=" * 80)

    # Display results
    for i in range(min(num_rows, len(df))):
        category = df.iloc[i]['category_text']
        emotion = df.iloc[i]['place_emotion']
```

```
            print(f"{i+1:3d}. {category:<45} → {emotion:>15}")

        # Show emotion distribution
        print(f"\n📈 EMOTION DISTRIBUTION ANALYSIS")
        print("=" * 50)

        emotion_counts = df['place_emotion'].value_counts()
        total = len(df)

        for emotion, count in emotion_counts.items():
            percentage = (count / total) * 100
            bar = "█" * int(percentage / 2)  # Visual bar
            print(f"{emotion:15s}: {count:4d} ({percentage:5.1f}%) {bar}")

        print(f"\nTotal places classified: {total:,}")

        # Show some interesting examples
        print(f"\n🎯 INTERESTING EXAMPLES BY EMOTION:")
        print("-" * 40)

        for emotion in emotion_counts.head(8).index:  # Top 8 emotions
            examples = df[df['place_emotion'] == emotion]['category_text'].head(3).tolist()
            print(f"{emotion.upper()}:")
            for example in examples:
                print(f"  • {example}")
            print()

    # ========================================
    # EXECUTE EVERYTHING
    # ========================================

    print("🎉 PLACE EMOTION CLASSIFIER READY!")
    print("=" * 60)
    print("This will:")
    print("1. Extract category text from your hierarchical columns")
    print("2. Classify each place with an emotional association")
    print("3. Show you the first 50 results")
    print("4. Display emotion distribution statistics")
    print("\nRunning classification now...")
    print("-" * 60)

    # Run the classification
    try:
        # Check if df exists
        if 'df' in locals() or 'df' in globals():
            print("✅ Found dataframe 'df'")

            # Run classification
            df_classified = classify_places_with_emotions(df)

            # Show results
            show_classification_results(df_classified, num_rows=50)

            # Store result
            df_result = df_classified

            print(f"\n🎊 SUCCESS!")
            print("Your classified dataframe is stored in 'df_result'")
            print("New columns added:")
            print("  • 'category_text' - extracted place names")
            print("  • 'place_emotion' - emotional classifications")

        else:
            print("⚠️  Dataframe 'df' not found in current environment")
            print("Please run: df_classified = classify_places_with_emotions(your_dataframe)")

    except Exception as e:
        print(f"❌ Error: {e}")
        print("\nTo run manually:")
        print("df_classified = classify_places_with_emotions(df)")
        print("show_classification_results(df_classified)")

    print(f"\n🔄 Available emotions: {list(PLACE_EMOTIONS.keys())}")
```

```
🎉 PLACE EMOTION CLASSIFIER READY!
============================================================
This will:
```

```
1. Extract category text from your hierarchical columns
2. Classify each place with an emotional association
3. Show you the first 50 results
4. Display emotion distribution statistics

Running classification now...
------------------------------------------------------------
✅ Found dataframe 'df'
🚀 STARTING PLACE EMOTION CLASSIFICATION
============================================================
📝 Step 1: Extracting category text from hierarchical columns...
Sample extracted texts:
  1. Hockey Club
  2. Indonesian Meatball Restaurant
  3. Candy Store
  4. Satay Restaurant
  5. Football Stadium
  6. Ski Store
  7. Transmissions Shop
  8. Dermatologist
  9. Farm
  10. Baseball

🎯 Step 2: Classifying emotions for 1245 places...

✅ Classification complete!

📊 SHOWING FIRST 50 CLASSIFICATION RESULTS
================================================================================
  1. Hockey Club                                → outdoors
  2. Indonesian Meatball Restaurant             → comfort
  3. Candy Store                                → retail
  4. Satay Restaurant                           → comfort
  5. Football Stadium                           → outdoors
  6. Ski Store                                  → adventure
  7. Transmissions Shop                         → stress
  8. Dermatologist                              → stress
  9. Farm                                       → relaxation,calm
 10. Baseball                                   → excitement
 11. Geriatric Doctor                           → healthcare
 12. Speakeasy                                  → entertainment
 13. College Lab                                → education
 14. Men's Store                                → retail
 15. Podiatrist                                 → healthcare
 16. Hostel                                     → neutral
 17. Convenience Store                          → convenience
 18. Street Fair                                → joy, happy
 19. Bowling Alley                              → joy, happy
 20. Assisted Living                            → neutral
 21. Paella Restaurant                          → food
 22. College Engineering Building               → education
 23. Video Store                                → retail
 24. Amphitheater                               → joy, happy
 25. Town Hall                                  → civic government
```

```
# ========================================
# DISPLAY RESULTS IN TABULAR FORMAT
# ========================================

import pandas as pd

def display_results_table(df, num_rows=50, columns_to_show=None):
    """Display classification results in a clean tabular format"""

    if 'place_emotion' not in df.columns:
        print("❌ No emotion classification found. Run classification first!")
        return

    # Default columns to display
    if columns_to_show is None:
        columns_to_show = ['category_text', 'place_emotion']

        # Add confidence columns if they exist
        if 'emotion_confidence' in df.columns:
            columns_to_show.append('emotion_confidence')
        if 'emotion_reasoning' in df.columns:
            columns_to_show.append('emotion_reasoning')

    # Select the data to display
    display_df = df[columns_to_show].head(num_rows).copy()
```

```
        # Clean up the display
        display_df.index = range(1, len(display_df) + 1)  # Start index from 1

        # Set pandas display options for better formatting
        pd.set_option('display.max_columns', None)
        pd.set_option('display.width', None)
        pd.set_option('display.max_colwidth', 50)
        pd.set_option('display.expand_frame_repr', False)

        print("=" * 100)
        print(f"PLACE EMOTION CLASSIFICATION RESULTS - FIRST {num_rows} ROWS")
        print("=" * 100)

        # Display the table
        print(display_df.to_string())

        print("=" * 100)

        return display_df

    def display_full_results_table(df):
        """Display ALL results in tabular format"""

        if 'place_emotion' not in df.columns:
            print("❌ No emotion classification found. Run classification first!")
            return

        # Select key columns
        display_df = df[['category_text', 'place_emotion']].copy()
        display_df.index = range(1, len(display_df) + 1)

        # Set display options to show all rows
        with pd.option_context('display.max_rows', None,
                               'display.max_columns', None,
                               'display.width', None,
                               'display.max_colwidth', 50):
            print("=" * 80)
            print(f"COMPLETE PLACE EMOTION CLASSIFICATION RESULTS ({len(df)} ROWS)")
            print("=" * 80)
            print(display_df.to_string())
            print("=" * 80)

        return display_df

    def display_by_emotion_table(df, emotion_filter):
        """Display results filtered by specific emotion in table format"""

        if 'place_emotion' not in df.columns:
            print("❌ No emotion classification found. Run classification first!")
            return

        # Filter by emotion
        filtered_df = df[df['place_emotion'] == emotion_filter][['category_text', 'place_emotion']].copy()

        if len(filtered_df) == 0:
            print(f"❌ No places found with emotion: {emotion_filter}")
            available_emotions = df['place_emotion'].unique()
            print(f"Available emotions: {sorted(available_emotions)}")
            return

        filtered_df.index = range(1, len(filtered_df) + 1)

        print("=" * 80)
        print(f"PLACES CLASSIFIED AS '{emotion_filter.upper()}' ({len(filtered_df)} results)")
        print("=" * 80)
        print(filtered_df.to_string())
        print("=" * 80)

        return filtered_df

    def display_emotion_summary_table(df):
        """Display emotion distribution summary in table format"""

        if 'place_emotion' not in df.columns:
            print("❌ No emotion classification found. Run classification first!")
            return
```

```
        # Create summary statistics
        emotion_counts = df['place_emotion'].value_counts()
        total = len(df)

        # Create summary dataframe
        summary_df = pd.DataFrame({
            'Emotion': emotion_counts.index,
            'Count': emotion_counts.values,
            'Percentage': [f"{(count/total)*100:.1f}%" for count in emotion_counts.values]
        })

        summary_df.index = range(1, len(summary_df) + 1)

        print("=" * 50)
        print("EMOTION DISTRIBUTION SUMMARY")
        print("=" * 50)
        print(summary_df.to_string())
        print("=" * 50)
        print(f"Total Places: {total:,}")

        return summary_df

    def display_comprehensive_table(df, num_rows=50):
        """Display comprehensive results with all available information"""

        if 'place_emotion' not in df.columns:
            print("❌ No emotion classification found. Run classification first!")
            return

        # Get all relevant columns
        display_columns = ['category_text', 'place_emotion']

        # Add additional columns if they exist
        optional_columns = ['emotion_confidence', 'emotion_reasoning', 'recommendation_score']
        for col in optional_columns:
            if col in df.columns:
                display_columns.append(col)

        # Also include some original columns for context
        original_columns = ['level6_category_name', 'level5_category_name', 'level4_category_name']
        for col in original_columns:
            if col in df.columns:
                display_columns.append(col)
                break  # Just add one for context

        # Create display dataframe
        display_df = df[display_columns].head(num_rows).copy()
        display_df.index = range(1, len(display_df) + 1)

        # Set pandas options for wide tables
        pd.set_option('display.max_columns', None)
        pd.set_option('display.width', None)
        pd.set_option('display.max_colwidth', 30)  # Shorter column width for wide tables

        print("=" * 120)
        print(f"COMPREHENSIVE CLASSIFICATION RESULTS - FIRST {num_rows} ROWS")
        print("=" * 120)
        print(display_df.to_string())
        print("=" * 120)

        return display_df

    # ========================================
    # EASY-TO-USE FUNCTIONS
    # ========================================

    def print_results_table(df, rows=50):
        """Simple function to print results in table format"""
        return display_results_table(df, num_rows=rows)

    def print_all_results(df):
        """Print ALL results (warning: might be very long!)"""
        return display_full_results_table(df)

    def print_emotion_summary(df):
        """Print emotion distribution summary"""
        return display_emotion_summary_table(df)
```

```
def print_by_emotion(df, emotion):
    """Print all places with specific emotion"""
    return display_by_emotion_table(df, emotion)

# ========================================
# USAGE EXAMPLES
# ========================================

print("=" * 60)
print("TABULAR DISPLAY FUNCTIONS READY!")
print("=" * 60)
print("Available display functions:")
print()
print("1. Basic table (first 50 rows):")
print("   print_results_table(df_result)")
print()
print("2. Custom number of rows:")
print("   print_results_table(df_result, rows=100)")
print()
print("3. All results (might be very long!):")
print("   print_all_results(df_result)")
print()
print("4. Emotion distribution summary:")
print("   print_emotion_summary(df_result)")
print()
print("5. Filter by specific emotion:")
print("   print_by_emotion(df_result, 'excitement')")
print("   print_by_emotion(df_result, 'comfort')")
print()
print("6. Comprehensive view with all columns:")
print("   display_comprehensive_table(df_result)")

# ========================================
# AUTO-EXECUTE IF df_result EXISTS
# ========================================

try:
    if 'df_result' in locals() or 'df_result' in globals():
        print("\n" + "=" * 60)
        print("FOUND df_result - DISPLAYING RESULTS TABLE")
        print("=" * 60)

        # Show first 50 results in table format
        print_results_table(df_result, rows=50)

        # Show emotion summary
        print_emotion_summary(df_result)

    else:
        print("\n⚠️  df_result not found. Run classification first, then use:")
        print("print_results_table(df_result)")

except NameError:
    print("\n⚠️  df_result not available. After classification, run:")
    print("print_results_table(df_result)")
```

```
============================================================
TABULAR DISPLAY FUNCTIONS READY!
============================================================
Available display functions:

1. Basic table (first 50 rows):
   print_results_table(df_result)

2. Custom number of rows:
   print_results_table(df_result, rows=100)

3. All results (might be very long!):
   print_all_results(df_result)

4. Emotion distribution summary:
   print_emotion_summary(df_result)

5. Filter by specific emotion:
   print_by_emotion(df_result, 'excitement')
   print_by_emotion(df_result, 'comfort')

6. Comprehensive view with all columns:
```

```
    display_comprehensive_table(df_result)

============================================================
FOUND df_result - DISPLAYING RESULTS TABLE
============================================================
================================================================================================================
PLACE EMOTION CLASSIFICATION RESULTS - FIRST 50 ROWS
================================================================================================================
                     category_text     place_emotion
1                      Hockey Club          outdoors
2   Indonesian Meatball Restaurant           comfort
3                      Candy Store            retail
4                  Satay Restaurant           comfort
5                  Football Stadium          outdoors
6                         Ski Store         adventure
7               Transmissions Shop            stress
8                    Dermatologist            stress
9                             Farm   relaxation,calm
10                        Baseball        excitement
11                Geriatric Doctor        healthcare
12                       Speakeasy     entertainment
13                    College Lab         education
14                    Men's Store            retail
15                     Podiatrist        healthcare
16                         Hostel           neutral
17                Convenience Store       convenience
18                      Street Fair        joy, happy
19                    Bowling Alley        joy, happy
20                  Assisted Living           neutral
21                Paella Restaurant              food
22    College Engineering Building         education
23                    Video Store            retail
24                   Amphitheater        joy, happy
25                      Town Hall  civic_government
26                         County  civic_government
27         Comfort Food Restaurant           comfort
```

```
df_result.to_csv('df_result.csv', index=False)

# Download the file to your local machine
from google.colab import files
files.download('df_result.csv')
```

```
neutral_rows = df_result[df_result['place_emotion'] == 'neutral']
print(neutral_rows)
```

```
                  category_id  category_level                                category_name
15    4bf58dd8d48988d1ee931735               3                                       Hostel       Travel and Transportat
19    5032891291d4c4b30a586d68               2                              Assisted Living         Community and Governn
31    63be6904847c3692a84b9b66               2                        Human Resources Agency  Business and Professional Se
33    5032781d91d4c4b30a586d5b               2                                       Tailor       Business and Professio
46    5032850891d4c4b30a586d62               4                                  Credit Union  Business and Professional Se
57    63be6904847c3692a84b9b8a               2                       Renewable Energy Service  Business and Professional Se
58    4bf58dd8d48988d1fa931735               3                                         Hotel       Travel and Transporta
63    63be6904847c3692a84b9b4e               3                 Carpet and Flooring Contractor  Business and Professional Se
65    53fca564498e1a175f32528b               3                                    Taxi Stand  Travel and Transportation >
85    522e32fae4b09b556e370f19               2                                    Optometrist                 Health and M
89    4bf58dd8d48988d129951735               3                                   Rail Station  Travel and Transportation >
92    4bf58dd8d48988d1dd931735               3                                      Soup Spot      Dining and Drinking >
97    63be6904847c3692a84b9b91               3                                Software Company  Business and Professional Se
99    63be6904847c3692a84b9b43               3                                Financial Planner  Business and Professional Se
107   530e33ccbcbc57f1066bbff9               3                                        Village  Landmarks and Outdoors > Sta
111   4bf58dd8d48988d1a9941735               4                             College Rec Center  Community and Government >
113   63be6904847c3692a84b9b73               2                                Metals Supplier  Business and Professional Se
120   5744ccdfe4b0c0459246b4be               3                                Currency Exchange  Business and Professional Se
121   52e81612bcbc57f1066b7a11               2                                        Gun Range                  Sports and
124   4eb1baf03b7b2c5b1d4306ca               2                                          Stable                   Landmarks
125   4bf58dd8d48988d143941735               2                                   Breakfast Spot              Dining and Drin
130   4bf58dd8d48988d12d941735               2                                         Monument                   Landmarks an
132   63be6904847c3692a84b9b4d               3                                        Carpenter  Business and Professional Se
137   52e81612bcbc57f1066b7a26               2                               Recreation Center          Sports and Recreatio
138   63be6904847c3692a84b9b90               2                              Technology Business  Business and Professional Se
142   4bf58dd8d48988d16c941735               3                                   Burger Joint    Dining and Drinking > Rest
143   5032897c91d4c4b30a586d69               2                                    Pet Service   Business and Professional S
150   57558b36e4b065ecebd306b4               4                                      Estaminet  Dining and Drinking > Restau
151   4bf58dd8d48988d1c5941735               3                                  Sandwich Spot   Dining and Drinking > Resta
159   56aa371be4b08b9a8d5734f9               2                                   Samba School              Arts and Entert
160   52f2ab2ebcbc57f1066b8b36               3                                     IT Service  Business and Professional Se
164   4eb1c1623b7b52c0e1adc2ec               3                                Car Dealership        Retail > Automotive Re
165   52e81612bcbc57f1066b7a42               3                                Driving School  Community and Government >
```

166	4bf58dd8d48988d1cc941735	3	Steakhouse	Dining and Drinking > Re
169	4bf58dd8d48988d1fd931735	3	Metro Station	Travel and Transportation >
173	5f2c1af1b6d05514c704319d	3	Marine Terminal	Travel and Transportation >
187	56aa371be4b08b9a8d5734d7	2	Industrial Estate	Business and Professional Se
189	63be6904847c3692a84b9b9e	2	Dump	Community
199	5e189d71eee47d000759b7e2	2	Meadery	Dining a
203	63be6904847c3692a84b9b5d	3	Painter	Business and Professional Se
214	4bf58dd8d48988d162941735	2	Other Great Outdoors	Landmarks and Outdoors
236	52e81612bcbc57f1066b7a2f	2	Bowling Green	Sports and Recre
238	4f4531504b9074f6e4fb0102	2	Platform	Travel and Tra
241	52f2ab2ebcbc57f1066b8b50	2	Cable Car	Travel and Tran
248	4f04b10d2fb6e1c99f3db0be	3	Music School	Community and Government >
249	63be6904847c3692a84b9b30	3	Towing Service	Business and Professional Se
250	4bf58dd8d48988d106941735	3	Track	Sports and Recreation > Run
251	5032833091d4c4b30a586d60	3	Motorcycle Dealership	Retail > Automotive Retail
256	63be6904847c3692a84b9c2b	3	Charter Bus	Travel and Transportation >
258	58daa1558bbb0b01f18ec1dc	4	Söğüş Place	Dining and Drinking > Restau
267	4bf58dd8d48988d13b941735	2	Education	Community and
276	52939a7d3cf9994f4e043a34	6	Tapiocaria	Dining and Drinking > Restau
280	63be6904847c3692a84b9ba6	3	Military	Community and Government >
284	63be6904847c3692a84b9b98	2	Wholesaler	Business and Professional
287	63be6904847c3692a84b9b3e	3	Accounting and Bookkeeping Service	Business and Professional Se
290	63be6904847c3692a84b9b4a	3	Hair Removal Service	Business and Professional Se

```
# Save the DataFrame to a CSV file
neutral_rows.to_csv('your_data.csv', index=False)

# Download the file to your local machine
from google.colab import files
files.download('your_data.csv')
```

```
print(df_result)
```

	category_id	category_level	category_name	
0	63be6904847c3692a84b9c0c	3	Hockey Club	Sports and Recreation
1	52960eda3cf9994f4e043acc	5	Indonesian Meatball Restaurant	Dining and Drinking > Restau
2	4bf58dd8d48988d117951735	3	Candy Store	Retail > Food and Beverage
3	56aa371be4b08b9a8d57350e	4	Satay Restaurant	Dining and Drinking > Restau
4	4bf58dd8d48988d189941735	3	Football Stadium	Arts and Entertainment > Sta
5	56aa371be4b08b9a8d573566	3	Ski Store	Retail > Sporting Goo
6	63be6904847c3692a84b9b31	3	Transmissions Shop	Business and Professional Se
7	63be6904847c3692a84b9bc9	3	Dermatologist	Health and Medicine > Phy
8	4bf58dd8d48988d15b941735	2	Farm	Landmar
9	63be6904847c3692a84b9bfe	2	Baseball	Sports and
10	63be6904847c3692a84b9bce	3	Geriatric Doctor	Health and Medicine > Physi
11	4bf58dd8d48988d1d4941735	3	Speakeasy	Dining and Drin
12	4bf58dd8d48988d1a5941735	4	College Lab	Community and Government >
13	4bf58dd8d48988d106951735	3	Men's Store	Retail > Fashion
14	63be6904847c3692a84b9bdd	2	Podiatrist	Health and
15	4bf58dd8d48988d1ee931735	3	Hostel	Travel and Transportat
16	4d954b0ea243a5684a65b473	2	Convenience Store	Reta
17	5267e4d8e4b0ec79466e48c5	3	Street Fair	Event > Mar
18	4bf58dd8d48988d1e4931735	2	Bowling Alley	Arts and Enterta
19	5032891291d4c4b30a586d68	2	Assisted Living	Community and Governm
20	4bf58dd8d48988d14d941735	4	Paella Restaurant	Dining and Drinking > Restau
21	4bf58dd8d48988d19e941735	4	College Engineering Building	Community and Government >
22	4bf58dd8d48988d126951735	2	Video Store	
23	56aa371be4b08b9a8d5734db	3	Amphitheater	Arts and Entertainment > Pe
24	52e81612bcbc57f1066b7a38	2	Town Hall	Community and
25	5345731ebcbc57f1066c39b2	3	County	Landmarks and Outdoors > Sta
26	52e81612bcbc57f1066b7a00	3	Comfort Food Restaurant	Dining and Drinking > Restau
27	52e81612bcbc57f1066b7a39	3	Mental Health Clinic	Health and Medicine > Menta
28	63be6904847c3692a84b9be9	2	Cannabis Store	Re
29	52e81612bcbc57f1066b7a3f	3	Hindu Temple	Community and Government > S
30	52e928d0bcbc57f1066b7e9c	4	Pelmeni House	Dining and Drinking > Restau
31	63be6904847c3692a84b9b66	2	Human Resources Agency	Business and Professional Se
32	4bf58dd8d48988d1b4941735	4	College Stadium	Community and Government >
33	5032781d91d4c4b30a586d5b	2	Tailor	Business and Professio
34	4f4534884b9074f6e4fb0174	2	Funeral Home	Business and Professional Se
35	5665ef1d498ec706735f0e59	3	Corporate Amenity	Business and Professional Se
36	4bf58dd8d48988d109951735	3	Lingerie Store	Retail > Fashion Re
37	52af3b513cf9994f4e043bff	5	Shandong Restaurant	Dining and Drinking > Restau
38	4bf58dd8d48988d1ed941735	3	Spa	Business and Professional Se
39	4bf58dd8d48988d1f1931735	2	General Entertainment	Arts and Entertainment >
40	52f2ab2ebcbc57f1066b8b3c	3	Massage Clinic	Business and Professional Se
41	4bf58dd8d48988d171941735	2	Event Space	Business and Professional S
42	56aa371be4b08b9a8d5734d3	2	Auto Workshop	
43	4eb1d4d54b900d56c88a45fc	2	Mountain	Landmarks a
44	52960eda3cf9994f4e043ac5	5	Padangnese Restaurant	Dining and Drinking > Restau
45	63be6904847c3692a84b9b76	3	Office Building	Business and Professional Se

46	5032850891d4c4b30a586d62	4	Credit Union	Business and Professional Se
47	58daa1558bbb0b01f18ec1f4	5	Colombian Restaurant	Dining and Drinking > Restau
48	52e81612bcbc57f1066b7a43	3	Circus School	Community and Government >
49	4bf58dd8d48988d1f1941735	2	Board Store	
50	55a5a1ebe4b013909087cb77	2	Mountain Hut	Landmarks and Ou
51	4bf58dd8d48988d103941735	3	Home (private)	Community and Government >
52	4bf58dd8d48988d112941735	2	Juice Bar	Dining and
53	5744ccdfe4b0c0459246b4a6	5	Kale Pache Place	Dining and Drinking > Restau
54	4f04b08c2fb6e1c99f3db0bd	2	Travel Agency	Travel and Transport
55	52af3b293cf9994f4e043bfb	5	Jiangxi Restaurant	Dining and Drinking > Restau

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score
import joblib

# ==============================
# 1. Load your dataset
# ==============================
# df_result should already be loaded with your columns:
# level1_category_name ... level6_category_name, place_emotion

# ==============================
# 2. Combine hierarchical categories into one text feature
# ==============================
df_result['category_combined'] = df_result[
    ['level1_category_name', 'level2_category_name', 'level3_category_name',
     'level4_category_name', 'level5_category_name', 'level6_category_name']
].fillna('').agg(' '.join, axis=1)

# ==============================
# 3. Encode target emotion
# ==============================
le_emotion = LabelEncoder()
df_result['place_emotion_enc'] = le_emotion.fit_transform(df_result['place_emotion'])

# ==============================
# 4. TF-IDF vectorization
# ==============================
X_text = df_result['category_combined']
y = df_result['place_emotion_enc']

tfidf_vectorizer = TfidfVectorizer(
    max_features=1000,  # reduce sparsity
    ngram_range=(1,2)
)
X = tfidf_vectorizer.fit_transform(X_text)

# ==============================
# 5. Train-test split
# ==============================
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ==============================
# 6. Train Random Forest classifier with class balancing
# ==============================
rf_model = RandomForestClassifier(
    n_estimators=500,
    random_state=42,
    class_weight='balanced'
)
rf_model.fit(X_train, y_train)

# ==============================
# 7. Evaluate the model
# ==============================
y_pred = rf_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))

# Avoid mismatch error: only show classes present in y_test
present_classes = sorted(list(set(y_test)))
target_names = [le_emotion.classes_[i] for i in present_classes]
```

```
print(classification_report(y_test, y_pred, labels=present_classes, target_names=target_names))

# ============================
# 8. Save model and vectorizer
# ============================
joblib.dump(rf_model, 'place_emotion_rf_model.pkl')
joblib.dump(tfidf_vectorizer, 'tfidf_vectorizer.pkl')
joblib.dump(le_emotion, 'emotion_encoder.pkl')

print("Model and encoders saved successfully!")
```

```
Accuracy: 0.46586345381526106
                  precision    recall  f1-score   support

       adventure       0.67      0.33      0.44         6
civic_government       0.00      0.00      0.00         1
         comfort       0.50      0.40      0.44        40
     convenience       0.00      0.00      0.00         1
      creativity       0.00      0.00      0.00         2
       education       0.67      0.29      0.40         7
          energy       0.00      0.00      0.00         1
   entertainment       0.14      0.33      0.20         3
      excitement       1.00      0.50      0.67        16
            fear       0.50      0.40      0.44         5
            food       0.36      0.48      0.41        21
      healthcare       0.33      0.33      0.33         3
      joy, happy       0.25      0.18      0.21        17
          luxury       0.00      0.00      0.00         1
         neutral       0.51      0.71      0.60        55
       nostalgia       0.00      0.00      0.00         1
        outdoors       0.33      0.33      0.33         3
    professional       0.00      0.00      0.00         2
 relaxation,calm       0.56      0.31      0.40        16
          retail       0.50      0.11      0.18         9
         romance       0.00      0.00      0.00         1
        shopping       0.64      0.80      0.71        20
          social       0.56      0.62      0.59         8
          stress       0.38      0.38      0.38         8
        wellness       0.50      0.50      0.50         2

       micro avg       0.48      0.47      0.47       249
       macro avg       0.34      0.28      0.29       249
    weighted avg       0.50      0.47      0.46       249

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
Model and encoders saved successfully!
```

```
# Install sentence-transformers if not installed
# !pip install sentence-transformers

import pandas as pd
from sentence_transformers import SentenceTransformer, util
import torch

# =========================
# 1. Load your dataset
# =========================
# df_result is your DataFrame with 'category_combined', 'category_cleaned', etc.

# We'll use 'category_cleaned' if it exists; otherwise 'category_combined'
text_column = 'category_cleaned' if 'category_cleaned' in df_result.columns else 'category_combined'
place_texts = df_result[text_column].tolist()

# =========================
# 2. Load pre-trained transformer
# =========================
model = SentenceTransformer('all-MiniLM-L6-v2')  # Fast, good quality embeddings

# =========================
# 3. Encode all places
# =========================
print("Encoding place texts...")
```

```
place_embeddings = model.encode(place_texts, convert_to_tensor=True)

# =========================
# 4. Function to recommend places
# =========================
def recommend_places(user_input, top_k=10):
    # Encode user input
    user_emb = model.encode(user_input, convert_to_tensor=True)

    # Compute cosine similarity
    cosine_scores = util.cos_sim(user_emb, place_embeddings)[0]  # shape: [num_places]

    # Get top-k indices
    top_results = torch.topk(cosine_scores, k=top_k)
    top_indices = top_results.indices.tolist()  # <-- convert tensor to list of ints

    recommended_places = []
    for score, idx in zip(top_results.values, top_indices):
        recommended_places.append({
            "place": df_result.iloc[idx][text_column],
            "emotion": df_result.iloc[idx]['place_emotion'],
            "score": score.item()
        })

    return recommended_places


# =========================
# 5. Example usage
# =========================
user_input = "happy, joy, outdoors"
top_places = recommend_places(user_input, top_k=10)

for i, place in enumerate(top_places, start=1):
    print(f"{i}. Place: {place['place']}, Emotion: {place['emotion']}, Score: {place['score']:.4f}")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100%  349/349 [00:00<00:00, 31.7kB/s]
config_sentence_transformers.json: 100%  116/116 [00:00<00:00, 13.3kB/s]
README.md: 10.5k/? [00:00<00:00, 1.02MB/s]
sentence_bert_config.json: 100%  53.0/53.0 [00:00<00:00, 6.02kB/s]
config.json: 100%  612/612 [00:00<00:00, 72.3kB/s]
model.safetensors: 100%  90.9M/90.9M [00:01<00:00, 113MB/s]
tokenizer_config.json: 100%  350/350 [00:00<00:00, 15.8kB/s]
vocab.txt: 232k/? [00:00<00:00, 7.50MB/s]
tokenizer.json: 466k/? [00:00<00:00, 14.6MB/s]
special_tokens_map.json: 100%  112/112 [00:00<00:00, 7.65kB/s]
config.json: 100%  190/190 [00:00<00:00, 9.93kB/s]
Encoding place texts...
1. Place: Landmarks and Outdoors Other Great Outdoors    , Emotion: neutral, Score: 0.4707
2. Place: Landmarks and Outdoors     , Emotion: nostalgia, Score: 0.4453
3. Place: Landmarks and Outdoors Well    , Emotion: relaxation,calm, Score: 0.4295
4. Place: Landmarks and Outdoors Field    , Emotion: neutral, Score: 0.4265
5. Place: Landmarks and Outdoors Garden    , Emotion: relaxation,calm, Score: 0.4198
6. Place: Landmarks and Outdoors Waterfront    , Emotion: outdoors, Score: 0.4175
7. Place: Landmarks and Outdoors Plaza    , Emotion: social, Score: 0.4160
8. Place: Landmarks and Outdoors Hot Spring    , Emotion: relaxation,calm, Score: 0.4141
9. Place: Landmarks and Outdoors Nature Preserve    , Emotion: relaxation,calm, Score: 0.4101
10. Place: Landmarks and Outdoors Picnic Shelter    , Emotion: joy, happy, Score: 0.4057
```

```
df_result.to_csv('df_result.csv', index=False)

# Download the file to your local machine
from google.colab import files
```

```
files.download('df_result.csv')
```

```
# ================================================
# 1. Install dependencies
# ================================================
!pip install -q sentence-transformers torch scikit-learn

# ================================================
# 2. Imports
# ================================================
import pandas as pd
import torch
from sentence_transformers import SentenceTransformer, InputExample, models
from torch.utils.data import DataLoader
from sklearn.preprocessing import LabelEncoder
import torch.nn as nn
import pickle

# ================================================
# 3. Prepare dataset
# ================================================
text_column = "category_combined"   # ✅ best text column
label_column = "place_emotion"

df = df_result[[text_column, label_column]].dropna().reset_index(drop=True)

print("Using text column:", text_column)
print("Number of rows:", len(df))
print("Sample rows:")
print(df.head())

# Encode labels
le = LabelEncoder()
df["label_encoded"] = le.fit_transform(df[label_column])

print("\nLabel classes:", list(le.classes_))

# Mapping dictionaries
id2label = {i: label for i, label in enumerate(le.classes_)}
label2id = {label: i for i, label in enumerate(le.classes_)}

# Convert into InputExamples
train_examples = [
    InputExample(texts=[str(row[text_column])], label=int(row["label_encoded"]))
    for _, row in df.iterrows()
]

# ================================================
# 4. Build custom classifier model
# ================================================
word_embedding_model = models.Transformer("sentence-transformers/all-MiniLM-L6-v2")
pooling_model = models.Pooling(word_embedding_model.get_word_embedding_dimension())

dense = models.Dense(
    in_features=pooling_model.get_sentence_embedding_dimension(),
    out_features=len(le.classes_),
    activation_function=nn.Identity()   # logits only
)

model = SentenceTransformer(modules=[word_embedding_model, pooling_model, dense])

# ================================================
# 5. Training setup
# ================================================
train_dataloader = DataLoader(train_examples, shuffle=True, batch_size=16)

class CrossEntropyLossWrapper(nn.Module):
    def __init__(self, model):
        super().__init__()
        self.model = model
        self.loss_fct = nn.CrossEntropyLoss()

    def forward(self, sentence_features, labels):
        features = sentence_features[0]
```

```
            logits = self.model(features)["sentence_embedding"]
            return self.loss_fct(logits, labels)

    train_loss = CrossEntropyLossWrapper(model)

    # ================================================
    # 6. Train
    # ================================================
    model.fit(
        train_objectives=[(train_dataloader, train_loss)],
        epochs=5,
        warmup_steps=int(0.1 * len(train_dataloader)),
        show_progress_bar=True
    )

    # ================================================
    # 7. Save model + label encoder
    # ================================================
    output_path = "fine_tuned_emotion_model"
    model.save(output_path)

    with open(f"{output_path}/label_encoder.pkl", "wb") as f:
        pickle.dump(le, f)

    print("✅ Fine-tuned classification model saved at", output_path)

    # ================================================
    # 8. Prediction function
    # ================================================
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model.to(device)

    def predict_emotion(text):
        features = model.tokenize([text])
        features = {k: v.to(device) for k, v in features.items()}

        with torch.no_grad():
            logits = model(features)["sentence_embedding"]
            probs = torch.softmax(logits, dim=1)
            pred_idx = torch.argmax(probs, dim=1).item()

        return id2label[pred_idx], probs[0][pred_idx].item()

    # ================================================
    # 9. Example prediction
    # ================================================
    print("\nPrediction example:")
    place = "Hockey Club"
    emotion, confidence = predict_emotion(place)
    print(f"Place: {place} → Emotion: {emotion} (confidence: {confidence:.4f})")
```

```
Using text column: category_combined
Number of rows: 1245
Sample rows:
                                category_combined place_emotion
0        Sports and Recreation Hockey Hockey Club         outdoors
1  Dining and Drinking Restaurant Asian Restauran...       comfort
2     Retail Food and Beverage Retail Candy Store           retail
3  Dining and Drinking Restaurant Asian Restauran...       comfort
4  Arts and Entertainment Stadium Football Stadiu...      outdoors
```

```
print("\nPrediction example:")
place = "li creste cafe"
emotion, confidence = predict_emotion(place)
print(f"Place: {place} → Emotion: {emotion} (confidence: {confidence:.4f})")
```

```
wandb: Paste an API key from your profile and hit enter: ··········
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the command line.
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Prediction example:
Place: li creste cafe → Emotion: shopping (confidence: 0.1258)
```

```
output_path = "place2mood_model"
model.save(output_path)

import pickle
with open(f"{output_path}/label_encoder.pkl", "wb") as f:
    pickle.dump(le, f)
```

```
  warnings.warn(warn_msg)
```

```
from google.colab import files
import shutil

# Make a zip file from the saved folder
shutil.make_archive("place2mood_model", 'zip', "place2mood_model")

# Download the zip file to your computer
files.download("place2mood_model.zip")
```