This is a starter notebook for the project, you'll have to import the libraries you'll need, you can find a list of the ones available in this workspace in the requirements.txt file in this workspace.

Installation

```
!pip install -r  requirements.txt
```

```
Collecting langchain==0.1.10 (from -r requirements.txt (line 1))
  Using cached langchain-0.1.10-py3-none-any.whl (806 kB)
Collecting langchain-community==0.0.25 (from -r requirements.txt (line 2))
  Using cached langchain_community-0.0.25-py3-none-any.whl (1.8 MB)
Collecting langchain-core==0.1.28 (from -r requirements.txt (line 3))
  Using cached langchain_core-0.1.28-py3-none-any.whl (252 kB)
Collecting langchain-experimental==0.0.53 (from -r requirements.txt (line 4))
  Using cached langchain_experimental-0.0.53-py3-none-any.whl (173 kB)
Requirement already satisfied: langchain-text-splitters==0.0.1 in /usr/local/
Requirement already satisfied: openai==0.28.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: pydantic==2.6.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pydantic_core==2.16.1 in /usr/local/lib/python
Requirement already satisfied: pytest>=7.4.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: sentence-transformers>=2.2.0 in /usr/local/lib
Requirement already satisfied: transformers>=4.31.0 in /usr/local/lib/python3
Requirement already satisfied: chromadb==0.4.24 in /usr/local/lib/python3.10/
Requirement already satisfied: jupyter==1.0.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: bitsandbytes in /usr/local/lib/python3.10/dist
Requirement already satisfied: diffusers in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: accelerate in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.1
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/lib
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3
Requirement already satisfied: langsmith<0.2.0,>=0.1.0 in /usr/local/lib/pyth
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/di
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in /usr/local/lib/pytho
Requirement already satisfied: anyio<5,>=3 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: packaging<24.0,>=23.2 in /usr/local/lib/python
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/pytho
Requirement already satisfied: typing-extensions>=4.6.1 in /usr/local/lib/pyt
Requirement already satisfied: build>=1.0.3 in /usr/local/lib/python3.10/dist
Requirement already satisfied: chroma-hnswlib==0.7.3 in /usr/local/lib/python
Requirement already satisfied: fastapi>=0.95.2 in /usr/local/lib/python3.10/d
Requirement already satisfied: uvicorn[standard]>=0.18.3 in /usr/local/lib/py
Requirement already satisfied: posthog>=2.4.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: pulsar-client>=3.1.0 in /usr/local/lib/python3
Requirement already satisfied: onnxruntime>=1.14.1 in /usr/local/lib/python3.
Requirement already satisfied: opentelemetry-api>=1.2.0 in /usr/local/lib/pyt
Requirement already satisfied: opentelemetry-exporter-otlp-proto-grpc>=1.2.0
Requirement already satisfied: opentelemetry-instrumentation-fastapi>=0.41b0
Requirement already satisfied: opentelemetry-sdk>=1.2.0 in /usr/local/lib/pyt
Requirement already satisfied: tokenizers>=0.13.2 in /usr/local/lib/python3.1
Requirement already satisfied: pypika>=0.48.9 in /usr/local/lib/python3.10/di
Requirement already satisfied: overrides>=7.3.1 in /usr/local/lib/python3.10/
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.
Requirement already satisfied: grpcio>=1.58.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: bcrypt>=4.0.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: typer>=0.9.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: kubernetes>=28.1.0 in /usr/local/lib/python3.1
Requirement already satisfied: mmh3>=4.0.1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: orjson>=3.9.12 in /usr/local/lib/python3.10/di
Requirement already satisfied: notebook in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: qtconsole in /usr/local/lib/python3.10/dist-pa
```

```
!pip install openai==0.28
```

```
Requirement already satisfied: openai==0.28 in /usr/local/lib/python3.10/dist
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/di
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/pyth
```

```python
from langchain.llms import OpenAI
```

```python
import torch
from langchain.chains import ConversationalRetrievalChain
from langchain.chains.question_answering import load_qa_chain
from langchain.document_loaders import DirectoryLoader
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.llms import HuggingFacePipeline
from langchain.memory import ConversationBufferMemory
from langchain.prompts import PromptTemplate
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import Chroma
from transformers import AutoTokenizer, GenerationConfig, TextStreamer, pipeli
```

Settings

```python
MODEL_NAME = 'gpt-4-turbo'
```

```
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'
IMAGES_DIR = 'images'

```

```
model = OpenAI(model_name=MODEL_NAME, api_key="sk-proj-YdcXMyjbGipRFHSJGYk4T3E

generation_prompt = """User
Generate exactly twenty real estate listings rows from USA locales. All the pr

Neighborhood: Malibu, California, USA
Price (USD): $1,200,000
Bedrooms: 4
Bathrooms: 3.5
House Size (sqft): 2,800
Description: Welcome to your coastal retreat in Malibu, California! This stunn
Neighborhood Description: Malibu, California, USA, epitomizes coastal luxury w

Neighborhood: Manhattan, New York City, USA
Price (USD): $2,500,000
Bedrooms: 3
Bathrooms: 2.5
House Size (sqft): 2,000
Description: Welcome to your urban oasis in the heart of Manhattan! This luxur
Neighborhood Description: Manhattan, New York City, USA, stands as the beating


Make sure the property description talks about why it's nice, like if it's goo

Format the CSV with clear titles for each part. Use the same style as the exam
"""

```

```
    /usr/local/lib/python3.10/dist-packages/langchain_community/llms/openai.py:24
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/langchain_community/llms/openai.py:10
      warnings.warn(
```

```
from typing import List
from langchain_core.pydantic_v1 import BaseModel

from langchain.output_parsers import PydanticOutputParser

class RealEstate(BaseModel):
    neighborhood: str
    price: str
    bedrooms: str
    bathrooms: str
    house_size: str
    description: str
    neighborhood_description: str

```

```
class ListingCollection(BaseModel):
    listings: List[RealEstate]


parser = PydanticOutputParser(pydantic_object=ListingCollection)

prompt = PromptTemplate(
    template="{instruction}\n{format_instructions}\n",
    input_variables=["instruction"],
    partial_variables={"format_instructions": parser.get_format_instructions},
)


```

```
chain = prompt | model | parser
result = chain.invoke({"instruction": generation_prompt})

```

```
from fastapi.encoders import jsonable_encoder

listings = jsonable_encoder(result.listings)

df = pd.DataFrame(listings)

df.to_csv("real_estate_listings.csv", index=True)
print("CSV file 'real_estate_listings.csv' has been created.")
```

```
CSV file 'real_estate_listings.csv' has been created.
```

```
import pandas as pd

df = pd.read_csv('real_estate_listings.csv')
df.head()
```

	Unnamed: 0	neighborhood	price	bedrooms	bathrooms	house_size	descript
0	0	Malibu, California, USA	$1,200,000	4	3.5	2,800 sqft	Welcom your coa retrea Malibu, C
1	1	Manhattan, New York City, USA	$2,500,000	3	2.5	2,000 sqft	Welcom your ur oasis in heart of M
2	2	Aspen, Colorado, USA	$3,400,000	5	4.0	3,500 sqft	Experie the ultin moun living in As

Next steps: Generate code with df View recommended plots

```
df.rename(columns={'Unnamed: 0': 'id'}, inplace=True)
```

```
df.head()
```

	id	neighborhood	price	bedrooms	bathrooms	house_size	description	ne
0	0	Malibu, California, USA	$1,200,000	4	3.5	2,800 sqft	Welcome to your coastal retreat in Malibu, Cal...	
1	1	Manhattan, New York City, USA	$2,500,000	3	2.5	2,000 sqft	Welcome to your urban oasis in the heart of Ma...	M
2	2	Aspen, Colorado, USA	$3,400,000	5	4.0	3,500 sqft	Experience the ultimate mountain living in Asp...	

Next steps: Generate code with df View recommended plots

```
from pathlib import Path


questions_dir = Path("chroma")
questions_dir.mkdir(exist_ok=True, parents=True)


def write_file(question, answer, file_path):
    text = f"""
Q: {question}
A: {answer}
""".strip()
    with Path(questions_dir / file_path).open("w") as text_file:
        text_file.write(text)
```

```
import os
from PIL import Image

property_texts = []

property_ids = [{'id': i} for i in range(len(df))]

property_template = """
Description: {}
```

```
Neighborhood Description: {}
Price: {}
Bedrooms: {}
Bathrooms: {}
House Size (sqft): {}
"""

for index, row in df.iterrows():
    property_texts.append(property_template.format(
        row['description'],
        row['neighborhood_description'],
        row['price'],
        row['bedrooms'],
        row['bathrooms'],
        row['house_size']
    ))

```

```
property_texts
```

```
['\nDescription: Welcome to your coastal retreat in Malibu, California! This
stunning residence offers 4 bedrooms, 3.5 bathrooms, and panoramic views of
the Pacific Ocean. With an open-concept living area, gourmet kitchen, and
expansive deck overlooking the ocean, it provides the perfect blend of
luxury and beachfront living. Step outside to enjoy direct access to the
sandy beach, or relax in the private hot tub while watching the sunset over
the water. This is coastal living at its finest.\nNeighborhood Description:
Malibu, California, USA, epitomizes coastal luxury with its pristine
beaches, rugged cliffs, and upscale amenities. Nestled along the iconic
Pacific Coast Highway, this affluent enclave offers a unique blend of
natural beauty, cultural attractions like the Getty Villa, and a laid-back
beach lifestyle. With its stunning vistas and commitment to environmental
stewardship, Malibu captivates residents and visitors alike with its
quintessential California charm.\nPrice: $1,200,000\nBedrooms: 4\nBathrooms:
3.5\nHouse Size (sqft): 2,800 sqft\n',
 "\nDescription: Welcome to your urban oasis in the heart of Manhattan! This
luxurious apartment offers 3 bedrooms, 2.5 bathrooms, and breathtaking views
of the city skyline. With floor-to-ceiling windows, a modern kitchen with
top-of-the-line appliances, and a spacious living area perfect for
entertaining, it epitomizes upscale city living. Step out onto the private
balcony to take in the bustling city below, or unwind in the building's
exclusive rooftop lounge while enjoying panoramic views of Manhattan. This
is urban living at its finest.\nNeighborhood Description: Manhattan, New
York City, USA, stands as the beating heart of one of the world's most
iconic urban landscapes. With its towering skyscrapers, bustling streets,
and diverse neighborhoods, Manhattan embodies the energy and dynamism of the
Big Apple. From the iconic landmarks of Times Square and Central Park to the
cultural hubs of Broadway and the Museum Mile, Manhattan offers a vibrant
tapestry of arts, entertainment, and culinary delights, attracting millions
of visitors from around the globe each year. In the midst of its fast-paced
lifestyle, Manhattan maintains a sense of community and resilience, with
neighborhoods like Greenwich Village, Harlem, and the Upper West Side each
contributing to the city's rich tapestry of culture and history.\nPrice:
$2,500,000\nBedrooms: 3\nBathrooms: 2.5\nHouse Size (sqft): 2,000 sqft\n",
 "\nDescription: Experience the ultimate mountain living in Aspen, Colorado!
 "\nDescription: Experience the ultimate mountain living in Aspen, Colorado!
This luxurious chalet features 5 bedrooms, 4 bathrooms, and breathtaking
views of the surrounding peaks. The home boasts a large living area with a
stone fireplace, a state-of-the-art kitchen, and a cozy dining area perfect
for après-ski gatherings. Enjoy the outdoor hot tub or explore the nearby
ski slopes and hiking trails. Aspen offers a perfect blend of adventure and
relaxation.\nNeighborhood Description: Aspen, Colorado, USA, is renowned for
its stunning mountain scenery, world-class skiing, and vibrant cultural
scene. This prestigious resort town offers a variety of outdoor activities
year-round, from skiing and snowboarding in the winter to hiking and biking
in the summer. Aspen's commitment to the environment is evident in its well-
maintained trails and open spaces. The town also boasts a lively downtown
area with upscale boutiques, fine dining, and art galleries, making it a
premier destination for visitors and residents alike.\nPrice:
$3,400,000\nBedrooms: 5\nBathrooms: 4.0\nHouse Size (sqft): 3,500 sqft\n",
 '\nDescription: Discover modern living in the heart of Silicon Valley with
this stunning Palo Alto home. Featuring 4 bedrooms, 3 bathrooms, and a
sleek, contemporary design, this property offers a spacious living area,
high-tech kitchen appliances, and a beautifully landscaped backyard. Ideal
for tech enthusiasts and families alike, this home is located near top-rated
schools and cutting-edge companies.\nNeighborhood Description: Palo Alto,
California, USA, is a global center for technology and innovation, home to
Stanford University and numerous tech companies. This vibrant community
offers a blend of suburban charm and high-tech sophistication. With its
```

```python
%pip install --upgrade --quiet  langchain-experimental
%pip install --upgrade --quiet  pillow open_clip_torch torch matplotlib
```

```python
from langchain_experimental.open_clip import OpenCLIPEmbeddings

db = Chroma(
    collection_name="listings", embedding_function=OpenCLIPEmbeddings()
)

```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: U
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings ta
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access p
  warnings.warn(
```

```python
db.add_texts(texts=property_texts, metadatas = property_ids)
```

```
['a85192d0-10bf-11ef-8ff1-0242ac1c000c',
 'a85194ec-10bf-11ef-8ff1-0242ac1c000c',
 'a85195d2-10bf-11ef-8ff1-0242ac1c000c',
 'a8519686-10bf-11ef-8ff1-0242ac1c000c',
 'a8519744-10bf-11ef-8ff1-0242ac1c000c',
 'a8519802-10bf-11ef-8ff1-0242ac1c000c',
 'a85198c0-10bf-11ef-8ff1-0242ac1c000c',
 'a8519974-10bf-11ef-8ff1-0242ac1c000c',
 'a8519a1e-10bf-11ef-8ff1-0242ac1c000c',
 'a8519ad2-10bf-11ef-8ff1-0242ac1c000c'.
 'a8519b72-10bf-11ef-8ff1-0242ac1c000c',
 'a8519c1c-10bf-11ef-8ff1-0242ac1c000c',
 'a8519cc6-10bf-11ef-8ff1-0242ac1c000c',
 'a8519d7a-10bf-11ef-8ff1-0242ac1c000c',
 'a8519e2e-10bf-11ef-8ff1-0242ac1c000c',
 'a8519ed8-10bf-11ef-8ff1-0242ac1c000c',
 'a8519f82-10bf-11ef-8ff1-0242ac1c000c',
 'a851a036-10bf-11ef-8ff1-0242ac1c000c',
 'a851a0e0-10bf-11ef-8ff1-0242ac1c000c',
 'a851a19e-10bf-11ef-8ff1-0242ac1c000c',
 'a851a284-10bf-11ef-8ff1-0242ac1c000c']
```

Process Images

```python
from diffusers import AutoPipelineForText2Image
pipeline = AutoPipelineForText2Image.from_pretrained("stabilityai/sdxl-turbo",
                                                     torch_dtype=torch.float16
                                                     variant="fp16").to(DEVICE

```

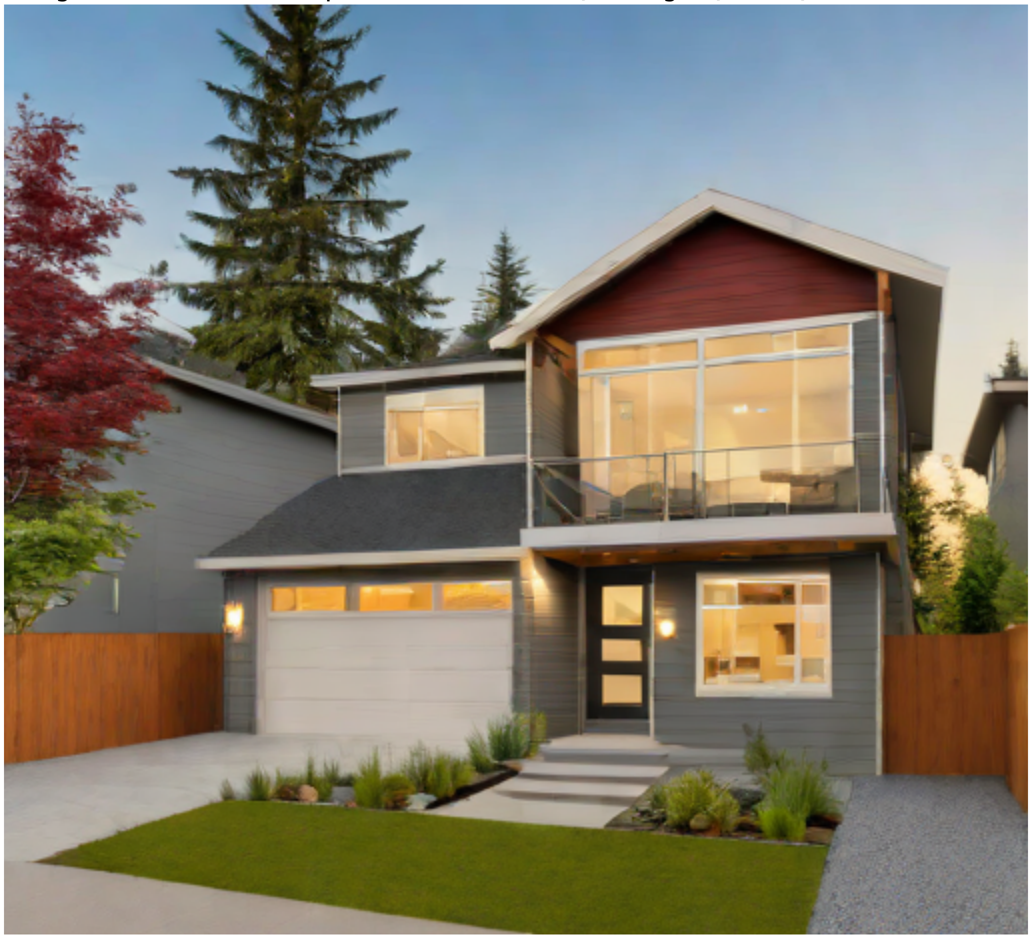
```
Loading pipeline components...: 100%                7/7 [00:05<00:00,  1.45s/
                                                                              it]
```

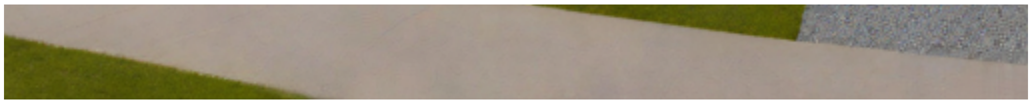
```python
listing_prompt = "An image illustrating the property and neighborhood descript
random_seed = torch.manual_seed(42)

listing_images = []
for _, listing_row in df.iterrows():
    custom_listing_prompt = listing_prompt.format(listing_row['description'],
    generated_image = pipeline(
        prompt=custom_listing_prompt,
        num_inference_steps=3,
        guidance_scale=1.0,
        negative_prompt=[],
        generator=random_seed
    ).images[0]
    listing_images.append(generated_image)

```

```
Token indices sequence length is longer than the specified maximum sequence l
The following part of your input was truncated because CLIP can only handle s
Token indices sequence length is longer than the specified maximum sequence l
The following part of your input was truncated because CLIP can only handle s
100%                                   3/3 [00:01<00:00,  2.19it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                   3/3 [00:00<00:00,  5.89it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  5.97it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  6.06it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  7.68it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  7.62it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  6.84it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  7.57it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  4.34it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  7.57it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  7.79it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  7.44it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  7.32it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  6.89it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  6.57it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                          3/3 [00:00<00:00,  6.10it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                                    3/3 [00:00<00:00,  5.09it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                                    3/3 [00:00<00:00,  7.52it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                                    3/3 [00:00<00:00,  6.96it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                                    3/3 [00:00<00:00,  7.43it/s]
The following part of your input was truncated because CLIP can only handle s
The following part of your input was truncated because CLIP can only handle s
100%                                                    3/3 [00:00<00:00,  7.44it/s]
```

```python
idx = 9
print('Description:', df.iloc[idx]['description'])
print('Neighborhood Description:', df.iloc[idx]['neighborhood_description'])
listing_images[idx]
```

```
Description: Embrace the vibrant lifestyle of Portland with this contemporary
Neighborhood Description: Portland, Oregon, USA, is celebrated for its progre
```





Generate
randomly select 5 items from a list
Close
Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
for i, image in enumerate(listing_images):
    filename = os.path.join('/content/chroma', str(i) + ".png")
    image.save(filename)
```

```
from PIL import Image
property_images = []
property_images_paths = []

for i in range(0,21):
    image = os.path.join('/content/chroma', str(i) + ".png")
    property_images.append(Image.open(image))
    property_images_paths.append(image)
```

```
db.add_images(uris=property_images_paths, metadatas = property_ids)
```

```
['d464797c-10c0-11ef-8ff1-0242ac1c000c',
 'd4663e24-10c0-11ef-8ff1-0242ac1c000c',
 'd4663f14-10c0-11ef-8ff1-0242ac1c000c',
 'd4663fc8-10c0-11ef-8ff1-0242ac1c000c',
 'd466407c-10c0-11ef-8ff1-0242ac1c000c',
 'd4664126-10c0-11ef-8ff1-0242ac1c000c',
 'd46641d0-10c0-11ef-8ff1-0242ac1c000c',
 'd466427a-10c0-11ef-8ff1-0242ac1c000c',
 'd466431a-10c0-11ef-8ff1-0242ac1c000c',
 'd46643c4-10c0-11ef-8ff1-0242ac1c000c',
 'd4664464-10c0-11ef-8ff1-0242ac1c000c',
 'd466450e-10c0-11ef-8ff1-0242ac1c000c',
 'd46645a4-10c0-11ef-8ff1-0242ac1c000c',
 'd466463a-10c0-11ef-8ff1-0242ac1c000c',
 'd46646e4-10c0-11ef-8ff1-0242ac1c000c',
 'd466478e-10c0-11ef-8ff1-0242ac1c000c',
 'd4664838-10c0-11ef-8ff1-0242ac1c000c',
 'd46648f6-10c0-11ef-8ff1-0242ac1c000c',
 'd46649a0-10c0-11ef-8ff1-0242ac1c000c',
 'd4664a40-10c0-11ef-8ff1-0242ac1c000c',
 'd4664aea-10c0-11ef-8ff1-0242ac1c000c']
```

Semantic Search and Response Generation

```
def get_similar_listings(user_preferences: List[str], top_k: int = 5) -> List[i
```

```

    combined_preferences = '\n'.join(user_preferences)
    results = db.similarity_search(combined_preferences, k=top_k * 2)

    return [result.metadata['id'] for result in results if result.metadata['id'

```

Generate 10 random numbers using numpy Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
instructions = 'Create a short description tailored to each listing, capturinc
```

```
class ListingSummary(BaseModel):
    listing_id: int
    summary_text: str

class SummaryCollection(BaseModel):
    listing_summaries: List[ListingSummary]

parser = PydanticOutputParser(pydantic_object=SummaryCollection)
prompt_template = PromptTemplate(
    template="{instruction}\nBuyer Preferences:\n{buyer_preferences}\nListings
    input_variables=["instruction", "buyer_preferences", "listings"],
    partial_variables={"format_instructions": parser.get_format_instructions},
)

```

```
cached_summaries = {}

def generate_customized_summaries(buyer_preferences: List[str], top_k: int = 5

    top_listings = get_similar_listings(buyer_preferences, top_k)

    listings = [('ID:' + str(listing_id), property_texts[listing_id]) for list

    query = prompt_template.format(
        instruction=instructions,
        buyer_preferences='\n'.join(buyer_preferences),
        listings='\n'.join([''.join(listing) for listing in listings])
    )

    response = []
    for summary in parser.parse(model(query)).listing_summaries:
        response.append((summary.listing_id, summary.summary_text))
        cached_summaries[summary.listing_id] = summary.summary_text

    return response

```

```
questions = ["What type of neighborhood are you aiming for?",
             "Which house size best suits your needs?",
             "How many bed do you want?",
             "How many bath do you want?",
             "What is the price range you want?"]

ans = [
        "California",
        "2000 sqft",
        "6",
        "4",
        ""
]

```

```
result = generate_customized_summaries(ans)
print(result)
```

```
[(5, 'Experience the pinnacle of luxury in this Beverly Hills estate, boastin
```

```
ans = [
        "New York",
        "1400 sqft",
        "3",
        "3",
        "2,000,000"
]

result = generate_customized_summaries(ans)
print(result)
```

```
[(1, 'Experience the pinnacle of Manhattan luxury with this 2,000 sqft apartm
```

```
!tar chvfz real_estate_udacity.tar.gz *
```

```
chroma/
chroma/15.png
chroma/16.png
chroma/6.png
chroma/13.png
chroma/0.png
chroma/11.png
chroma/20.png
chroma/1.png
chroma/3.png
chroma/10.png
chroma/2.png
chroma/9.png
chroma/7.png
chroma/8.png
```

```
chroma/12.png
chroma/18.png
chroma/5.png
chroma/14.png
chroma/4.png
chroma/19.png
chroma/17.png
real_estate_listings.csv
requirements.txt
sample_data/
sample_data/anscombe.json
sample_data/README.md
sample_data/mnist_train_small.csv
sample_data/mnist_test.csv
sample_data/california_housing_train.csv
sample_data/california_housing_test.csv
```

```
Start coding or generate with AI.
```