—--README FILE—---

Your Rental Buddy: Gathering, Scraping, Munging, and Cleaning Data

Estimate of total data: around 2000 records

Step 1: Found Sources of data to scrape from
JSON, HTML websites

Step 2: Gathered real-world data for multiple databases which will aid international students with finding accommodation
Data Sources:
 *Scraped data from the JuneHomes website
 *Google forms for real-time updates about the needs of students when it comes to finding accommodation

Step 3: Cleaned the data
Data cleaning methods used:
 *Added data to data frames
 *Removed null values as per the percentage
 *Filled the remaining null values with mean or median
 *Removed outliers
 *Removed redundant data
 *Normalized the data

Step 4: Created tables corresponding to the data found

Step 5: Created use cases and joined the relevant tables for retrieving required information for the rentee/renter

—---INSERT QUERIES—---

```
INSERT INTO JuneApartments(
id,
Apt_id ,
url ,
address,
Beds ,
Bath ,
Price ,
BedArea ,
Availablefrom ,
Availabletill ,
Description
)
VALUES
        ( '1001',
        '78',
        'https://junehomes.com/residences/boston-ma/mission-hill/1068-mission-hill/3141',
        'Park Drive',
                    '2',
                    '2',
                    '1100',
                    '69',
                    '2022-12-05',
                    '2023-01-04',
'This comfy, cute and charming room is available'
                         );

INSERT INTO JuneAmenities(
id,
Amenities
)
VALUES
          (
   '6811',
          'Wifi'
                         );

INSERT INTO Junetransport(
id,
Trans_id,
stations,
color,
walktime,
description)
VALUES
          (
                             '9',
                              '2',
                               'Blue Line',
                              '#2040AA',
                              '10',
                             'Boston Landing'
                         );

INSERT INTO SubleaseSpot(

  `Name`,
  `PhoneNumber`,
  `Email`,
  `Gender`,
  `Address`,
  `ProxToUni`,
  `Brokerage`,
  `LeaseSpotType`,
  `BedroomCount`,
  `BathroomCount`,
  `Rent`,
  `DietaryPref`,
  `GenderPref`,
  `Amenities`,
  `AvailSpot`,
  `PrefMoveInDate`,
  `AvailSpotNum`
  )
VALUES

                            (

                            'Vaishali Mhatre',
                              '8573286790',
                              'veenam45@gmail.com',
                                                                               'Female',
                             'Park Drive',
                                                                              '0.5',
                                         '700',
                             'OnLease',
                                         '2',
                             '3',
                                                                            '1100',
                           'Vegetarian',
                           'Female',
                           'Dishwasher',
                           '1',
                           '2023-01-01',
                    '1'
                             );



INSERT INTO SubleaseRoommate(
  `Name`,
  `Gender`,
   `PhoneNumber`,
  `Email`,
  `Budget`,
  `RoommateDietaryPref`,
  `RoommateGenderPref`,
  `Amenities`,
  `PrefModeofTravel`,
  `TypeOfSpot`,
  `PrefMoveInDate`,
  `NoOfRoommates`
  )
VALUES

                            (
                            'Shalini Pawar',
                            'Female',
                              '8573286790',
                              'veenam45@gmail.com',
                                                                                    '700',
                             'Vegetarian',
                                          'Female',
                             'Wifi',
                                                                                 'Green Line',
                           'OnLease',
                           '2023-01-01',
                           '2'
                             );

INSERT INTO TemporarySpot(
  `Name`,
   `PhoneNumber`,
     `Email`,
      `Gender`,
  `Address`,
  `ProximityToUni`,
  `BedroomCount`,
  `BathroomCount`,
  `TempRent`,
  `DietaryPref`,
  `GenderPref`,
  `Amenities`,
  `AvailableSpot`,
  `PrefMoveInDate`,
   `PrefMoveOutDate`,
   `AvailSpotNum`
  )
VALUES

                            (
                            'Sayak Hande',
                           '8573286790',
                           'veenam45@gmail.com',
                             'Male',
                             'J Vue at the LMA',
                                                                                    '0.5',
                              '2',
                              '2',
                               '10',
                              'Vegetarian',
                                            'Female',
                              'INHouseLoundry',
                                                                                                  '2',
                            '2023-01-01',
                            '2023-02-05',
                            '2'

);


INSERT INTO TemporaryRoommate(`RoommateId`,
  `Name`,
  `Gender`,
   `PhoneNumber`,
  `Email`,
  `Budget`,
  `DietaryPref`,
  `GenderPref`,
  `Amenities`,
  `PrefModeofTravel`,
  `TypeOfSpot`,
  `PrefMoveInDate`,
  `NoOfRoommates`
  )
VALUES

                             ( '1',
                             'Shalini Pawar',
                             'Female',
                               '8573286790',
                               'veenam45@gmail.com',
                                                                                                     '700',
                              'Vegetarian',
                                            'Female',
                              'Wifi',
                                                                                                  'Green Line',
                            'OnLease',
                            '2023-01-01',
                            '2'
                              );


—------------------------------------------CREATE QUERIES—---------------------------------------------

CREATE TABLE JuneApartments
(
id int not NULL PRIMARY KEY,
Apt_id int,
url varchar(300),
address varchar(100),
Beds decimal,
Bath decimal,
Price decimal,
BedArea int,
Availablefrom date,
Availabletill date,
Description varchar(500));


CREATE TABLE JuneAmenities
(
id int not NULL ,
Amenities varchar(100));


CREATE TABLE Junetransport
(
id int not NULL ,
Trans_id int,
stations varchar(100),
color varchar(100),
walktime int,
description varchar(100));

CREATE TABLE `SubleaseSpot` (
  `SpotID` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(45),
  `PhoneNumber` VARCHAR(10),
  `Email` VARCHAR(45),
  `Gender` VARCHAR(45) ,
  `Address` VARCHAR(200),
  `ProxToUni` FLOAT,
  `Brokerage` INT,
  `LeaseSpotType` VARCHAR(100),
  `BedroomCount` INT,
  `BathroomCount` INT,
  `Rent` INT,
  `DietaryPref` VARCHAR(45),
  `GenderPref` VARCHAR(45),
  `Amenities` VARCHAR(200),
  `AvailSpot` VARCHAR(45),
  `PrefMoveInDate` DATE,
  `AvailSpotNum` INT,
  PRIMARY KEY (`SpotID`),
  UNIQUE INDEX `SpotID_UNIQUE` (`SpotID` ASC) VISIBLE);


  CREATE TABLE `SubleaseRoommate` (
  `RoommateID` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(45) ,
  `Gender` VARCHAR(45),
  `PhoneNumber` varchar(10) ,
  `Email` VARCHAR(45),
  `Budget` INT,
  `RoommateDietaryPref` VARCHAR(45),
  `RoommateGenderPref` VARCHAR(45),
  `Amenities` VARCHAR(45),
  `PrefModeofTravel` VARCHAR(45),
  `TypeOfSpot` VARCHAR(45),
  `PrefMoveInDate` DATE,
  `NoOfRoommates` INT,
  PRIMARY KEY (`RoommateID`),
  UNIQUE INDEX `RoommateID_UNIQUE` (`RoommateID` ASC) VISIBLE);

CREATE TABLE TemporarySpot
(SpotID int NOT NULL PRIMARY KEY,
Name VARCHAR(45),
PhoneNumber int(10),
Email VARCHAR(45),
Gender VARCHAR(45),
Address VARCHAR(200),
ProximityToUni float,
BedroomCount int,
BathroomCount int,
TempRent int,
DietaryPref VARCHAR(45),
GenderPref VARCHAR(45),
Amenities VARCHAR(45),
AvailableSpot VARCHAR(45),
PrefMoveInDate date,
PrefMoveOutDate date,
AvailSpotNum int);

CREATE TABLE TemporaryRoommate
(RoommateId int NOT NULL PRIMARY KEY,
Name varchar(45),
Gender varchar(45),
PhoneNumber varchar(10),
Email varchar(45),
Budget varchar(45),
DietaryPref varchar(45),
GenderPref varchar(45),
Amenities varchar(45),
PrefModeofTravel varchar(45),
TypeOfSpot varchar(45),
PrefMoveInDate date,
NoOfRoommates int);
```

```
import tweepy
import pandas as pd
from sqlalchemy import create_engine
import mysql.connector as msql
from mysql.connector import Error
import re
df_apt = pd.read_csv(r'C:\Users\amey8\Downloads\Assignment3\JuneApt.csv')
df_amenities = pd.read_csv(r'C:\Users\amey8\Downloads\Assignment3\JuneAmenities.csv')
df_transport = pd.read_csv(r'C:\Users\amey8\Downloads\Assignment3\Transport.csv')
try:

   conn = msql.connect(host='localhost', database='rental_buddydb', user='root',
password='amey@1105',auth_plugin='mysql_native_password')

   if conn.is_connected():
      cursor = conn.cursor()

      for i,row in df_apt.iterrows():
         t=[]
         i=0
         for j in row:
            if(i!=0):
               t.append(j)
            i=i+1
         sql = "INSERT INTO
JuneApartments(id,Apt_id,url,Address,Beds,Bath,Price,BedArea,Availablefrom,Availabletill,D
escription) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
         #print(sql)
         cursor.execute(sql, tuple(t))
         conn.commit()

      for i,row in df_amenities.iterrows():
         t=[]
         i=0
         for j in row:
            if(i!=0):
               t.append(j)
            i=i+1
         sql = "INSERT INTO JuneAmenitites(id,Amenities) VALUES (%s,%s)"
         #print(sql)
```

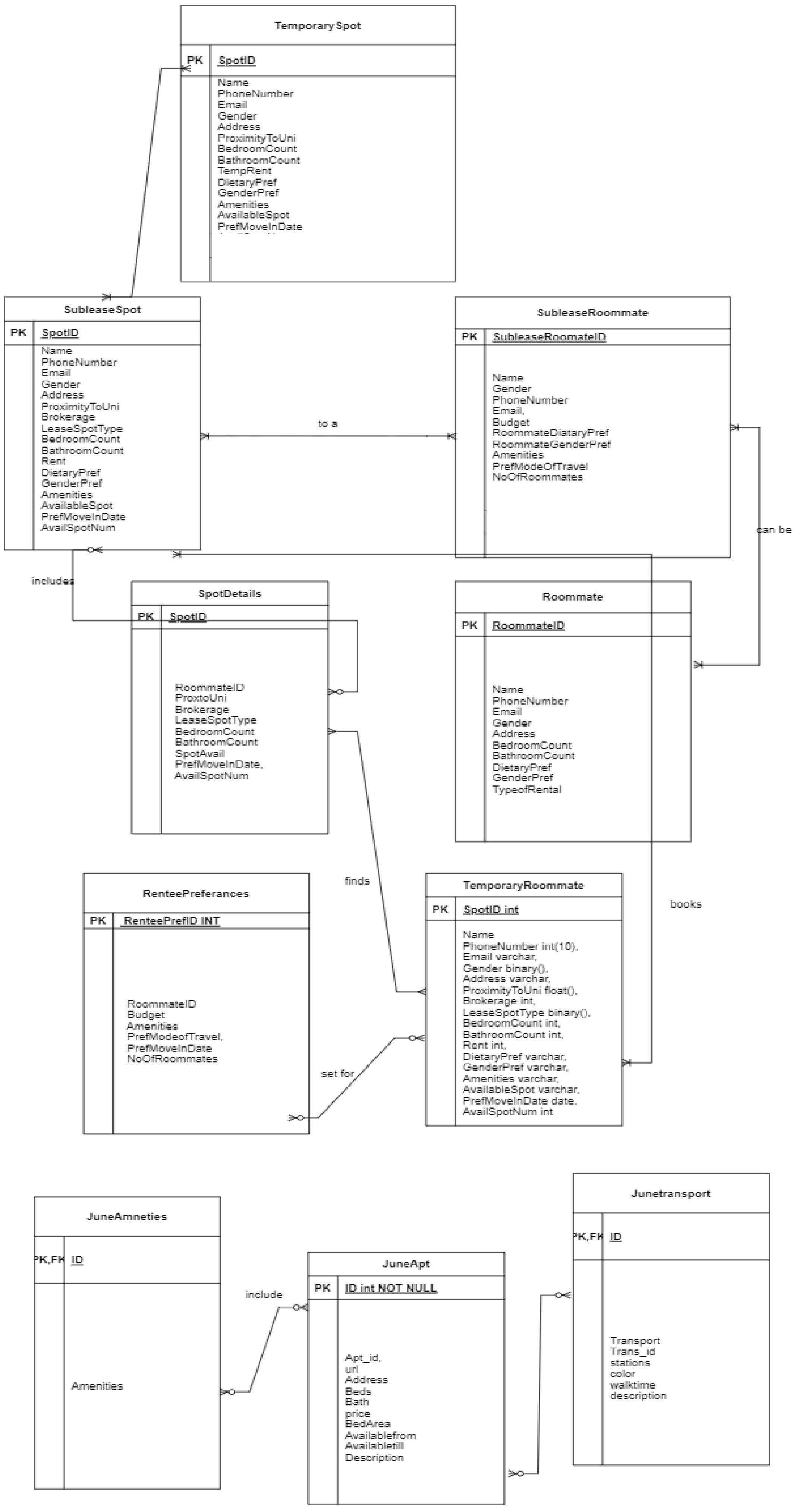
```
            cursor.execute(sql, tuple(t))
            conn.commit()

        for i,row in df_transport.iterrows():
            t=[]
            i=0
            for j in row:
                if(i!=0):
                    t.append(j)
                i=i+1
            sql = "INSERT INTO Junetransport(id,Trans_id,stations,color,walktime,description)
VALUES (%s,%s,%s,%s,%s,%s)"
            #print(sql)
            cursor.execute(sql, tuple(t))
            conn.commit()

        cursor.close()
        conn.close()
except Error as e:
    print("Error while connecting to MySQL", e)
    if cursor and conn:
        cursor.close()
        conn.close()
```

——

```
# -*- coding: utf-8 -*-
"""Assignment3.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/11bezCB5qmRU57Ml3BPR12XVuY5z1-s-R
"""

#filtering unavailable rooms
def findidbed(l_s):
  tidx_l=l_s.find('ListingCard_badgeTooltip__INLZY')+33
  tidx_u = l_s[tidx_l:].find('<')
  #print (l_s[tidx_l:tidx_u+tidx_l-1 ])
  return l_s[tidx_l:tidx_u+tidx_l-1 ]

#Code to scrap data from website
from bs4 import BeautifulSoup
import requests
from csv import writer
import csv
import json
city='boston-ma'
main_apt=[]
main_transport=[]
main_amenities=[]
apt_url=[]
apt_add=[]
apt_amenities=[]
apt_bed_area=[]
apt_room_id=[]
apt_price=[]
apt_transport=[]
apt_avail_from=[]
apt_avail_till=[]
apt_bedroom=[]
apt_bathroom=[]
apt_desc=[]
apt_size=[]
id=0
header=['Apt_id','Type','url','Address','Beds','Bath','AptArea','BedArea','Amenities']
NoneType = type(None)
url=[]
for i in range(1,22):
  if i == 1:
    url.append("https://junehomes.com/residences/boston-ma?count=50")
  else:
    url.append(f"https://junehomes.com/residences/boston-ma?count=50&page={i}")

i=0
for ur in url:
  page = requests.get(ur)
  soup = BeautifulSoup(page.content, 'html.parser')
  aptlist = [soup.find_all('a', class_="ListingCard_root__dWXKe") ,soup.find_all('div',
class_="ListingCard_badgeTooltip__INLZY")]
  print(ur)
  k=0

  for apt in aptlist[0]:
    apt_s=str(apt)
    #print(i)

    if len(aptlist[1])<=k :
      break
    if findidbed( str( aptlist[1][k]))!= 'Bedroom':
      break
    i=i+1
    k=k+1
    #print(k)
    href_index = apt_s.find('href')+6  #apartment details on specific apt list
    turl="https://junehomes.com"+apt_s[href_index:apt_s.find('>')-1]
    #print (turl)
    apt_url.append(turl)
    page1 = requests.get(turl)
    soup_apt = BeautifulSoup(page1.content, 'html.parser')
    apt_bed_area.append(str(soup_apt.find('span', class_="Typography_p1-500__fXf6d
charcoal-800").text))
    #apt_amenities.append(list(str(soup_apt.find_all('span',
class_="Typography_p1-500__fXf6d FeaturesList_label__b8j4n")).replace('<span
class="Typography_p1-500__fXf6d
FeaturesList_label__b8j4n">','').replace('</span>','')[1:-1].split(',')))
    apt_amenities.append(str(soup_apt.find_all('span', class_="Typography_p1-500__fXf6d
FeaturesList_label__b8j4n")).replace('<span class="Typography_p1-500__fXf6d
FeaturesList_label__b8j4n">','').replace('</span>','')[1:-1])
    tidx=turl.find(city)+ len(city)+1
    turl=turl[tidx:]
    apt_add.append(turl)
    rent=soup_apt.find_all('script',id="__NEXT_DATA__")
    s=str(rent).replace('<script id="__NEXT_DATA__"
type="application/json">','').replace('</script>','')
    y=json.loads(s[1:-1])
    if str(y["props"]["pageProps"]).find("room")>3:
      apt_room_id.append("N/A")
      apt_price.append("N/A")
      apt_transport.append("N/A")
      apt_avail_from.append("N/A")
      apt_avail_till.append("N/A")
      apt_avail_from.append("N/A")
      apt_bedroom.append("N/A")
      apt_bathroom.append("N/A")
      apt_desc.append("N/A")
    else:
      apt_room_id.append(y["props"]["pageProps"]["room"]["id"])
      apt_price.append(y["props"]["pageProps"]["room"]["price"])
      apt_transport.append(y["props"]["pageProps"]["room"]["transport"])
      #if y["props"]["pageProps"]["room"]["availability"][0] is not None:
      #print(str(y["props"]["pageProps"]["room"]).find("availability"))
      if y["props"]["pageProps"]["room"]["available"]==False:
        apt_avail_from.append("N/A")
        apt_avail_till.append("N/A")
      else:
        apt_avail_from.append(y["props"]["pageProps"]["room"]["availability"][0])
        apt_avail_till.append(y["props"]["pageProps"]["room"]["availability"][1])
      apt_bedroom.append(y["props"]["pageProps"]["room"]["homeBedrooms"])
      apt_bathroom.append(y["props"]["pageProps"]["room"]["homeBathrooms"])
      apt_desc.append(y["props"]["pageProps"]["room"]["description"])
```

```
#
header=['id','Apt_id','url','Address','Beds','Bath','Price','BedArea','Amenities','Availablefrom','A
vailabletill','Transport','Description']
# for j in range(0,i):
#
main_apt.append([j,apt_room_id[j],apt_url[j],apt_add[j],apt_bedroom[j],apt_bathroom[j],apt_p
rice[j],apt_bed_area[j],apt_amenities[j],apt_avail_from[j],apt_avail_till[j],apt_desc[j]])

header_transport=['id','Transport']
header_amenities=['id','Amenities']
header_apt=['id','Apt_id','url','Address','Beds','Bath','Price','BedArea','Availablefrom','Available
till','Description']
for j in range(0,i):
  main_transport.append([j,apt_transport[j]])
  # for am in apt_amenities[j]:
  #   main_amenities.append([j,am])
  main_amenities.append([j,apt_amenities[j]])

main_apt.append([j,apt_room_id[j],apt_url[j],apt_add[j],apt_bedroom[j],apt_bathroom[j],apt_p
rice[j],apt_bed_area[j],apt_avail_from[j],apt_avail_till[j],apt_desc[j]])

#Putting into dataframe
import pandas as pd
df_apt = pd.DataFrame (main_apt, columns=
['id','Apt_id','url','Address','Beds','Bath','Price','BedArea','Availablefrom','Availabletill','Descripti
on'])
df_transport=pd.DataFrame (main_transport, columns= ['id','Transport'])
df_amenities=pd.DataFrame (main_amenities, columns= ['id','Amenities'])
df_apt.to_csv('RawJuneApt.csv')
df_transport.to_csv('RawJuneTransport.csv')
df_amenities.to_csv('RawJuneAmenities.csv')

#display Apartment
main_apt[0]

#display Transport from Apartment
main_transport[0]

#display Apartment Amenities
main_amenities[0][1]

"""#Data Cleaning Data from Web Scraping

"""

cleaned_transport=[]

for i in range(0,len(main_transport)):
  for j in range(0, len(main_transport[i][1])):
    tstr=""
    if (str(main_transport[i][1][j]).find("id")==2):
    #print(i,j,main_transport[i][1][j]["stations"])
      for s in main_transport[i][1][j]["stations"]:
        tstr=tstr+","+str(s)

cleaned_transport.append([main_transport[i][0],main_transport[i][1][j]["id"],tstr,main_transport
[i][1][j]["color"],main_transport[i][1][j]["walktime"],main_transport[i][1][j]["description"]])

df_transport=pd.DataFrame (cleaned_transport, columns=
['id','Trans_id','stations','color','walktime','description'])

cleaned_amenities=[]

for i in main_amenities:

  for j in i[1].split(", "):
     cleaned_amenities.append([i[0],j])
  #print(i[1].split(", "))

df_amenities=pd.DataFrame (cleaned_amenities, columns= ['id','Amenities'])

df_apt.head()

df_transport.head()

df_amenities.head()

#fetching number of rows and columns
print("Apt:",df_apt.describe())
print("\nTransport:",df_transport.describe())
print("\nAmenities:",df_amenities.describe())

#fetching number of rows and columns
print("Apt:",df_apt.shape)
print("Transport:",df_transport.shape)
print("Amenities:",df_amenities.shape)

print("Apt:")
print (f'id: {df_apt.id.count()}' )
print (f'Apt_id: {df_apt.Apt_id.count()}' )
print (f'url: {df_apt.url.count()}' )
print (f'Address: {df_apt.Address.count()}' )
print (f'Beds: {df_apt.Beds.count()}' )
print (f'Bath: {df_apt.Bath.count()}' )
print (f'Price: {df_apt.Price.count()}' )
print (f'BedArea: {df_apt.BedArea.count()}' )
print (f'Availablefrom: {df_apt.Availablefrom.count()}' )
print (f'Availabletill      : {df_apt.Availabletill   .count()}' )
print (f'Description: {df_apt.Description.count()}' )

print("\nTransport:")
print (f'Trans_id: {df_transport.Trans_id.count()}' )
print (f'Station: {df_transport.stations.count()}' )
print (f'id: {df_transport.id.count()}' )
print (f'walktime: {df_transport.walktime.count()}' )
print (f'description: {df_transport.description.count()}' )

#fetching unique values in each column
print("Apt:",df_apt.nunique())
print("\n\nTransport:",df_transport.nunique())
print("\n\nAmenities:",df_amenities.nunique())

#Searching for Duplicate Entries
print("Apt:",df_apt.duplicated().sum())
print("\n\nTransport:",df_transport.duplicated().sum())
print("\n\nAmenities:",df_amenities.duplicated().sum())

#Replacing 'N/A' value with null value
df_apt= df_apt.replace('N/A',None)
df_apt = df_apt.where((pd.notnull(df_apt)),None)

df_transport= df_transport.replace('N/A',None)
df_transport = df_transport.where((pd.notnull(df_transport)),None)

df_amenities= df_amenities.replace('N/A',None)
df_amenities = df_amenities.where((pd.notnull(df_amenities)),None)

#Searching for null values in Dataframe

print("Apt:\n",df_apt.isnull().sum())
print("\n\nTransport:\n",df_transport.isnull().sum())
print("\n\nAmenities:\n",df_amenities.isnull().sum())

#Boundaries
import numpy as np

print(df_apt.Price.min())
print(df_apt.Price.max())

print(df_apt.Price.quantile(.25))
print(df_apt.Price.quantile(.50))

print(df_apt.Price.quantile (.75))
print(df_apt.Price.mean())

print(df_apt.Price.median())
print(df_apt.Price.mode())

print(df_apt.Price.std())

"""#**Loading cleaned data into excel**"""

#Putting into dataframe

df_apt.to_csv('JuneApt.csv')
df_transport.to_csv('Transport.csv')
df_amenities.to_csv('JuneAmenities.csv')

import pandas as pd
from sqlalchemy import create_engine
import mysql.connector as msql
from mysql.connector import Error
import re
df_apt['Availablefrom'] = df_apt['Availablefrom'].apply(lambda a: pd.to_datetime(a))
df_apt['Availabletill'] = df_apt['Availabletill'].apply(lambda a: pd.to_datetime(a))

"""## **Realtime Data collected from Google forms**"""

import pandas as pd
from sqlalchemy import create_engine
import mysql.connector as msql
from mysql.connector import Error
import re

df_subspot = pd.read_csv('SubletyourSpot.csv')
df_tempspot = pd.read_csv('TemporarySpotSublet.csv')

df_subspot.head()

df_apt['Availabletill'] = df_apt['Availabletill'].apply(lambda a: pd.to_datetime(a))

df_tempspot.head()

df_subspot.pop('Timestamp')
df_tempspot.pop('Timestamp')

#df_subspot.head()
df_subspot.head()

print("\nSubleasedSpot:",df_subspot.describe())
print("\nTemporarySpot:",df_tempspot.describe())
print("\nSubleasedSpot:",df_subspot.shape)
print("\nTemporarySpot:",df_tempspot.shape)

#fetching unique values in each column
print("SubleasedSpot:",df_subspot.nunique())
print("\nTemporarySpot:",df_tempspot.nunique())

# #Replacing 'N/A' value with null value
# df_subspot= df_subspot.replace('N/A',None)
# df_subspot = df_subspot.where((pd.notnull(df_subspot)),None)

# #Replacing 'N/A' value with null value
# df_subspot= df_tempspot.replace('N/A',None)
# df_subspot = df_tempspot.where((pd.notnull(df_tempspot)),None)

#Searching for null values in Dataframe

print("SubleasedSpot:\n",df_subspot.isnull().sum())
print("\nTemporarySpot:",df_subspot.isnull().sum())

#df_subspot.pop('Available Spot')

#Searching for null values in Dataframe

print("SubleasedSpot:\n",df_subspot.isnull().sum())
print("\nTemporarySpot:\n",df_tempspot.isnull().sum())

#Boundaries
import numpy as np

print(df_subspot.Rent.min())
print(df_subspot.Rent.max())

df_subspot.to_csv('SubleaseSpot.csv')
df_tempspot.to_csv('TemporarySpot.csv')
```

—---ER DIAGRAM-------—---------------------------------------


Temporary Spot
PK SpotID
Name
PhoneNumber
Email
Gender
Address
ProximityToUni
BedroomCount
BathroomCount
TempRent
DietaryPref
GenderPref
Amenities
AvailableSpot
PrefMoveInDate
Sublease Spot
PK SpotID
Name
PhoneNumber
Email
Gender
Address
ProximityToUni
Brokerage
LeaseSpotType
BedroomCount
BathroomCount
Rent
DietaryPref
GenderPref
Amenities
AvailableSpot
PrefMoveInDate
AvailSpotNum
to a
SubleaseRoommate
PK SubleaseRoomateID
Name
Gender
PhoneNumber
Email,
Budget
RoommateDiataryPref
RoommateGenderPref
Amenities
PrefModeOfTravel
NoOfRoommates
can be
includes
SpotDetails
PK SpotID
RoommateID
ProxtoUni
Brokerage
LeaseSpotType
BedroomCount
BathroomCount
SpotAvail
PrefMoveInDate,
AvailSpotNum
Roommate
PK RoommateID
Name
PhoneNumber
Email
Gender
Address
BedroomCount
BathroomCount
DietaryPref
GenderPref
TypeofRental
finds
books
RenteePreferances
PK RenteePrefID INT
RoommateID
Budget
Amenities
PrefModeofTravel,
PrefMoveInDate
NoOfRoommates
set for
TemporaryRoommate
PK SpotID int
Name
PhoneNumber int(10),
Email varchar,
Gender binary(),
Address varchar,
ProximityToUni float(),
Brokerage int,
LeaseSpotType binary(),
BedroomCount int,
BathroomCount int,
Rent int,
DietaryPref varchar,
GenderPref varchar,
Amenities varchar,
AvailableSpot varchar,
PrefMoveInDate date,
AvailSpotNum int
JuneAmneties
PK,FK ID
Amenities
include
JuneApt
PK ID int NOT NULL
Apt_id,
url
Address
Beds
Bath
price
BedArea
Availablefrom
Availabletill
Description
Junetransport
PK,FK ID
Transport
Trans_id
stations
color
walktime
description