In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
# getting carparks dataset as a dataframe
hdbcarparks = pd.read_excel("hdbcarparks(converted).xlsx")
hdbcarparks.head()
```

Out[2]:

	Unnamed: 0	car_park_no	address	x_coord	y_coord	car_park_type	type_of_parking_system
0	0	ACB	BLK 270/271 ALBERT CENTRE BASEMENT CAR PARK	30314.7936	31490.4942	BASEMENT CAR PARK	ELECTRONIC PARKING
1	1	ACM	BLK 98A ALJUNIED CRESCENT	33758.4143	33695.5198	MULTI-STOREY CAR PARK	ELECTRONIC PARKING
2	2	AH1	BLK 101 JALAN DUSUN	29257.7203	34500.3599	SURFACE CAR PARK	ELECTRONIC PARKING
3	3	AK19	BLOCK 253 ANG MO KIO STREET 21	28185.4359	39012.6664	SURFACE CAR PARK	COUPON PARKING
4	4	AK31	BLK 302/348 ANG MO KIO ST 31	29482.0290	38684.1754	SURFACE CAR PARK	COUPON PARKING

In [3]:
```
# getting sports facility dataset as a dataframe
sportsfacilities = pd.read_csv("sportsfacilities.csv")
sportsfacilities.head()
```

Out[3]:

	X	Y	gid	Name	description	FACILITIES	ROAD_NAME	CO
0	103.951881	1.374282	2	ActiveSG Pasir Ris Sport Centre	NaN	Swimming Complex/Sports Hall/Stadium/Tennis Ce...	Pasir Ris Central	65835 H
1	103.802553	1.296177	3	Queenstown ActiveSG Swimming Complex/Stadium	NaN	Swimming Complex/Stadium	Stirling Road	Compl
2	103.874920	1.356413	5	Serangoon Swimming Complex/ Stadium	NaN	Swimming Complex/ Stadium	Yio Chu Kang Road	Compl
3	103.780196	1.434791	7	Woodlands Sports Centre	NaN	Swimming Complex/Sports Hall/Stadium/Gym	Woodlands Street 12	

	X	Y	gid	Name	description	FACILITIES	ROAD_NAME	CO
								62694 H
4	103.861536	1.325444	8	St Wilfred ActiveSG Sports Centre	NaN	Field/Tennis Centre	St Wilfred Road	I Cer Centre

In [4]:
```
# Renaming the column names
hdbcarparks=hdbcarparks.rename(columns = {'y':'lat','x':'lon'})
sportsfacilities=sportsfacilities.rename(columns = {'X':'lat','Y':'lon'})
# To make sure that there are no null values and All are either integers/ Float valu
hdbcarparks.info()
print('\n XXXXXXXXXXXXXXXXXXXXXXX\n')
sportsfacilities.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2192 entries, 0 to 2191
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Unnamed: 0              2192 non-null   int64  
 1   car_park_no             2192 non-null   object 
 2   address                 2192 non-null   object 
 3   x_coord                 2192 non-null   float64
 4   y_coord                 2192 non-null   float64
 5   car_park_type           2192 non-null   object 
 6   type_of_parking_system  2192 non-null   object 
 7   short_term_parking      2192 non-null   object 
 8   free_parking            2192 non-null   object 
 9   night_parking           2192 non-null   object 
 10  car_park_decks          2192 non-null   int64  
 11  gantry_height           2192 non-null   float64
 12  car_park_basement       2192 non-null   object 
 13  lon                     2192 non-null   float64
 14  lat                     2192 non-null   float64
dtypes: float64(5), int64(2), object(8)
memory usage: 257.0+ KB

 XXXXXXXXXXXXXXXXXXXXXXX

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   lat          35 non-null     float64
 1   lon          35 non-null     float64
 2   gid          35 non-null     int64  
 3   Name         35 non-null     object 
 4   description  0 non-null      float64
 5   FACILITIES   35 non-null     object 
 6   ROAD_NAME    35 non-null     object 
 7   CONTACT_NO   35 non-null     object 
 8   GYM          17 non-null     float64
 9   INC_CRC      35 non-null     object 
 10  FMEL_UPD_D   35 non-null     float64
dtypes: float64(5), int64(1), object(5)
memory usage: 3.1+ KB
```

In [5]:
```
# Harversine equation - helps to find out closest carparks
from math import radians, cos, sin, asin, sqrt
def dist(lat1, long1, lat2, long2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lat1, long1, lat2, long2 = map(radians, [lat1, long1, lat2, long2])
    # haversine formula
    dlon = long2 - long1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371* c
    return km
```

In [6]:
```
# functions to find the nearest 3 carparks and their carpark number and distance (in

def find_nearest1(lat, long):
    distances = hdbcarparks.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    return hdbcarparks.loc[distances.idxmin(), 'address']

def find_cpno1(lat, long):
    distances = hdbcarparks.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    return hdbcarparks.loc[distances.idxmin(), 'car_park_no']

def find_dist1(lat, long):
    distances = hdbcarparks.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    return int(distances.min()*1000)

def find_nearest2(lat, long):
    distances = hdbcarparks.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    distances = distances.drop(distances.idxmin())
    return hdbcarparks.loc[distances.idxmin(), 'address']

def find_cpno2(lat, long):
    distances = hdbcarparks.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    distances = distances.drop(distances.idxmin())
    return hdbcarparks.loc[distances.idxmin(), 'car_park_no']

def find_dist2(lat, long):
    distances = hdbcarparks.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    distances = distances.drop(distances.idxmin())
    return int(distances.min()*1000)

def find_nearest3(lat, long):
    distances = hdbcarparks.apply(
```

```
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    distances = distances.drop(distances.idxmin())
    distances = distances.drop(distances.idxmin())
    return hdbcarparks.loc[distances.idxmin(), 'address']

def find_cpno3(lat, long):
    distances = hdbcarparks.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    distances = distances.drop(distances.idxmin())
    distances = distances.drop(distances.idxmin())
    return hdbcarparks.loc[distances.idxmin(), 'car_park_no']

def find_dist3(lat, long):
    distances = hdbcarparks.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    distances = distances.drop(distances.idxmin())
    distances = distances.drop(distances.idxmin())
    return int(distances.min()*1000)
```

In [7]:

```
sportsfacilities['address1'] = sportsfacilities.apply(
    lambda row: find_nearest1(row['lat'], row['lon']),
    axis=1)
sportsfacilities['cpno1'] = sportsfacilities.apply(
    lambda row: find_cpno1(row['lat'], row['lon']),
    axis=1)
sportsfacilities['dist1'] = sportsfacilities.apply(
    lambda row: find_dist1(row['lat'], row['lon']),
    axis=1)
sportsfacilities['address2'] = sportsfacilities.apply(
    lambda row: find_nearest2(row['lat'], row['lon']),
    axis=1)
sportsfacilities['cpno2'] = sportsfacilities.apply(
    lambda row: find_cpno2(row['lat'], row['lon']),
    axis=1)
sportsfacilities['dist2'] = sportsfacilities.apply(
    lambda row: find_dist2(row['lat'], row['lon']),
    axis=1)
sportsfacilities['address3'] = sportsfacilities.apply(
    lambda row: find_nearest3(row['lat'], row['lon']),
    axis=1)
sportsfacilities['cpno3'] = sportsfacilities.apply(
    lambda row: find_cpno3(row['lat'], row['lon']),
    axis=1)
sportsfacilities['dist3'] = sportsfacilities.apply(
    lambda row: find_dist3(row['lat'], row['lon']),
    axis=1)
# To check the data frame if it has a new column of hotel name (for each and every m
sportsfacilities.head()
```

Out[7]:

	lat	lon	gid	Name	description	FACILITIES	ROAD_NAME	CO
0	103.951881	1.374282	2	ActiveSG Pasir Ris Sport Centre	NaN	Swimming Complex/Sports Hall/Stadium/Tennis Ce...	Pasir Ris Central	65835 H
1	103.802553	1.296177	3	Queenstown ActiveSG Swimming Complex/Stadium	NaN	Swimming Complex/Stadium	Stirling Road	Compl

	lat	lon	gid	Name	description	FACILITIES	ROAD_NAME	CO
2	103.874920	1.356413	5	Serangoon Swimming Complex/ Stadium	NaN	Swimming Complex/ Stadium	Yio Chu Kang Road	Compl
3	103.780196	1.434791	7	Woodlands Sports Centre	NaN	Swimming Complex/Sports Hall/Stadium/Gym	Woodlands Street 12	62694 H
4	103.861536	1.325444	8	St Wilfred ActiveSG Sports Centre	NaN	Field/Tennis Centre	St Wilfred Road	Cer Centre

In [8]:
```
# drop some useless columns
sportsfacilities=sportsfacilities.drop(columns=['gid', 'description','INC_CRC','FMEL
sportsfacilities.head()
```

Out[8]:

	lat	lon	Name	FACILITIES	ROAD_NAME	CONTACT_NO	GYM
0	103.951881	1.374282	ActiveSG Pasir Ris Sport Centre	Swimming Complex/Sports Hall/Stadium/Tennis Ce...	Pasir Ris Central	Swimming Complex: 65835523; Sports Hall: 65838...	1.0
1	103.802553	1.296177	Queenstown ActiveSG Swimming Complex/Stadium	Swimming Complex/Stadium	Stirling Road	Swimming Complex/Stadium: 64737269	NaN
2	103.874920	1.356413	Serangoon Swimming Complex/ Stadium	Swimming Complex/ Stadium	Yio Chu Kang Road	Swimming Complex/Stadium: 62884606	NaN
3	103.780196	1.434791	Woodlands Sports Centre	Swimming Complex/Sports Hall/Stadium/Gym	Woodlands Street 12	Swimming Complex: 62694192; Sports Hall: 63652...	1.0
4	103.861536	1.325444	St Wilfred ActiveSG Sports Centre	Field/Tennis Centre	St Wilfred Road	Field/Tennis Centre/Squash Centre: 62933452	NaN

In [9]:
```
# export to excel
sportsfacilities.to_excel("nearestCarparks.xlsx")
```