

I've imported pandas to store information regarding each table in the ERD, I plan on filling each Dataframe for each table. Display & HTML are purely for pretty printing in the ipynb file. I've used Faker to generate a realistic and random dataset.

```
In [ ]: ! pip install Faker

import pandas as pd
from IPython.display import display, HTML
from faker import Faker
```

I've made a dataframe for each table and initialized each column header according to the table's attributes. The reason I've used Panda's dataframe is because it is very easy to manipulate and pretty print especially when you're working with large datasets. It's also extremely easy to generate a CSV file using Panda's dataframe.

```
In [ ]: #Declare Dataframes for each table

adminDataFrame = pd.DataFrame(columns=['adminID', 'adminName', 'loginID', 'password', 'adminType'])
customerDataFrame = pd.DataFrame(columns=['custID', 'custName', 'customerType', 'emailID', 'address', 'provice', 'city', 'pincode', 'password'])
accountDataFrame = pd.DataFrame(columns=['accountID', 'custID', 'meterID', 'accountNumber', 'name', 'address', 'accountStatus'])
invoiceDataFrame = pd.DataFrame(columns=['invoiceID', 'meterID', 'accountNumber', 'tarriID', 'readingDate', 'dueDate', 'billNumber', 'presentReading', 'previousReading', 'consumptionUnit', 'fixedCharge', 'EnergyCharge', 'Tax', 'billAmount', 'interest', 'previousBalance', 'interestBalanceAmount', 'credit', 'netAmount', 'status'])
meterDataFrame = pd.DataFrame(columns=['meterID', 'meterType'])
tarriDataFrame = pd.DataFrame(columns=['tarriID', 'tarriType', 'tarriDescription'])
billingDataFrame = pd.DataFrame(columns=['billID', 'custID', 'invoiceID', 'accountID', 'paymentMode', 'paymentDate', 'paymentTime', 'billAmount', 'paidAmount', 'excessPaid'])
```

Following cells showcases all the functions that I've used or could've used in my data generation, The reason I chose Faker was because I wanted to generate extremely realistic datasets, note every function call to Faker produces an External API request, more on that later.

```
In [ ]: #Random data fetcher
fake = Faker()
print('\nRandom Name : \n'+fake.name())
print('\nRandom Address : \n'+fake.street_address())
print('\nRandom City : \n'+fake.city())
print('\nRandom Postal Code : \n'+fake.postcode())
print('\nRandom Phone Number : \n'+fake.phone_number())
print('\nRandom Date : \n'+fake.date(pattern='%Y-%m-%d', end_datetime=None))
print('\nRandom Time : \n'+fake.time(pattern='%H:%M:%S', end_datetime=None))
print('\nRandom Date Before : \n',fake.date_this_year(before_today=True, after_today=False).year)
print('\nRandom Date After : \n',fake.date_this_year(before_today=False, after_today=True).year)
print('\nRandom Integer : \n',fake.random_int(min=0, max=9999, step=1))
print('\nRandom Email : \n'+fake.profile(sex='F').get('mail'))
print('\nRandom Encrypted Password : \n'+fake.password())
print('\nRandom Description : \n'+fake.text())
```

There are certain attributes which need to be filled using business knowledge we gathered in the previous phase of the project, or that have to do with very specific details, I've declared those variables over here, and I'll be using these later on.

```
In [ ]: #fixed data
adminType = ['Super Admin', 'Basic Rights']
customerType = ['Residencial', 'Commercial']
accountStatus = ['active', 'in-active']
paymentMode = ['cash', 'bank transfer', 'credit']
tarrifType = ['Area factor', 'Time factor']
tarrifDescription=['Factor applied to different Areas depending on average Income', 'Factor applied at different intervals during the day']
meterType = ['Digital', 'Variable', 'Dial', 'Smart']
invoiceStatus = ['payed', 'due']
provinces = ['Sindh', 'Punjab', 'Balochistan', 'KPK', 'Giglit Baltistan']
```

certain tables in the datasets contain secondary information that gets referenced throughout the OLTP database, The reason to generate them separately is because their size is usually very restricted, and varies for each attribute.

```
In [ ]: #Generating Secondary tables
def generateMeterTable():
    for i in range(len(meterType)):
        meterDataFrame.loc[i] = [i,meterType[i]]

def generateTarriifTable():
    for i in range(len(tarriifType)):
        tarriifDataFrame.loc[i] = [i,tarriifType[i],tarriifDescription[i]]

def generateAdminTable():
    length = 100
    for i in range(length):
        adminDataFrame.loc[i] = [i,fake.name(),i ,fake.password() , adminType[i%2]]

generateMeterTable()
generateTarriifTable()
generateAdminTable()
display(HTML(meterDataFrame.to_html()))
display(HTML(tarriifDataFrame.to_html()))
display(HTML(adminDataFrame.to_html()))
```

This cell block generates the dataset, loop generates 30000 rows, the reason for such a small amount is that my obsession with generating realistic dataset introduced External API calls (Faker), Faker's backend has to handle these requests and so it can't just let us keep making 1000s of requests per seconds due to network traffic restrictions, Which is why they throttle the number of requests we can make, slowing the process of generating data exponentially. Although it allows us to make 100s of requests per second however, as the requests go up it throttles the number of requests even more so much so that once i've generated 20000 rows it limits us to 5 requests per second.

Generating these 30k rows on my PC took over 2 hours.

```
In [ ]: #Generating Primary tables
for i in range (30001):
    custProfile = fake.profile(sex='M')
    randomDate = fake.date(pattern='%Y-%m-%d', end_datetime=None)
    randomDate2 = fake.date(pattern='%Y-%m-%d', end_datetime=None)
    randomTime = fake.time(pattern='%H:%M:%S', end_datetime=None)
    randomInteger = fake.random_int(min=0, max=9999, step=1)

    customerDataFrame.loc[i] = [i,custProfile.get('name'),customerType[i%2],custProfile.get('mail'),fake.street_address(),provinces[i%5],fake.city(),fake.postcode(),fake.password()]
    billingDataFrame.loc[i] = [i,i,i,i,paymentMode[i%3],randomDate,randomTime,randomInteger,randomInteger+i,i]
    accountDataFrame.loc[i] = [i,i,i%len(meterType),i,custProfile.get('name'),fake.street_address(),accountStatus[i%2]]
    invoiceDataFrame.loc[i] = [i,i%len(meterType),i,i%len(tariffType),randomDate,randomDate2,i,randomInteger,randomInteger/2,"KWH",randomInteger/10,randomInteger/15,(randomInteger*2/7),(randomInteger+100*2),"12.5%",randomInteger/2,(randomInteger/2),randomInteger,randomInteger*5,invoiceStatus[i%2]]
    print(i)
```

```
In [ ]: display(HTML(customerDataFrame.to_html()))
display(HTML(billingDataFrame.to_html()))
display(HTML(accountDataFrame.to_html()))
display(HTML(invoiceDataFrame.to_html()))
```

```
In [ ]: #Saving Data frames (tables) as CSV Files
adminDataFrame.to_csv('adminTable.csv',index=False)
customerDataFrame.to_csv('customerTable.csv',index=False)
accountDataFrame.to_csv('accountTable.csv',index=False)
invoiceDataFrame.to_csv('invoiceTable.csv',index=False)
meterDataFrame.to_csv('meterTable.csv',index=False)
tariffDataFrame.to_csv('tariffTable.csv',index=False)
billingDataFrame.to_csv('billingTable.csv',index=False)
```

I'd like to apologize for the limited number of rows, Although I've used 30k rows they are nowhere near enough for big data. I hope the realistic dataset makes up for it, I would've completely turned it around but I was extremely busy with my Projects and Finals along with my Final year project assignments, and I had to leave karachi on 8th of January. Sorry.

```
In [ ]:
```