PROBLEM STATEMENT

The script we have to generate here should focus on customer's notification alert preferences. This is for stock recommendation application. Here as input we will get customer's preferences for type of alerts like "sms", "email" and "telegram". At output we should get the valid of their preferences. Meaning if customer wants "sms" as an alert then our database should have their contact number. If not then the alert will not be created.

as our previous script, here also we had followed same step like importing liabraries, connecting to the database, inserting dummy values for our references, generating methods using class in python.

Input/ output file format: JSON Purpose: To set alert preference based upon the available information of customers from our database.

In [1]:

```
import mysql.connector
import pandas as pd
import json
from sqlalchemy import create_engine

engine=create_engine("mysql+pymysql://root:120450109009@localhost:3306/df")

conn = mysql.connector.connect(host="localhost", user="root", passwd="120450109009", database="df" ,charset="utf8")
cur= conn.cursor(dictionary= True)

cur.execute("CREATE TABLE np_tab(ip_id BIGINT, notif_type_id BIGINT, notif_pref VARCHAR(50), dt VARCHAR(50))")
cur.execute("CREATE TABLE e_tab(ip_id BIGINT, eml VARCHAR(50), st_dt VARCHAR(50), e_dt VARCHAR(50))")
cur.execute("CREATE TABLE ph_tab(ip_id BIGINT, ph BIGINT, st_dt VARCHAR(50), e_dt VARCHAR(50))")
cur.execute("CREATE TABLE tg_tab(ip_id BIGINT, tg VARCHAR(50), st_dt VARCHAR(50), e_dt VARCHAR(50))")

cur.execute("insert into e_tab values(1,NULL,'1-1-2021','1-1-2021')")
cur.execute("insert into e_tab values(2,'xy@gmail.com','1-1-2021','1-1-2021')")
cur.execute("insert into e_tab values(3,'xz@gmail.com','1-1-2021','1-1-2021')")
cur.execute("insert into e_tab values(4,'yz@gmail.com','1-1-2021','1-1-2021')")
conn.commit()

cur.execute("insert into ph_tab values(1,9876543210,'1-1-2021','1-1-2021')")
cur.execute("insert into ph_tab values(2,9876543210,'1-1-2021','1-1-2021')")
cur.execute("insert into ph_tab values(3,9876543210,'1-1-2021','1-1-2021')")
cur.execute("insert into ph_tab values(4,9876543210,'1-1-2021','1-1-2021')")
conn.commit()

cur.execute("insert into tg_tab values(1,'xyz','1-1-2021','1-1-2021')")
cur.execute("insert into tg_tab values(2,'xyz','1-1-2021','1-1-2021')")
cur.execute("insert into tg_tab values(3,'xyz','1-1-2021','1-1-2021')")
cur.execute("insert into tg_tab values(4,'xyz','1-1-2021','1-1-2021')")
conn.commit()
```

In [2]:

```
class alert:
    def __init__(self, data):
        self.data = data
    def output(self, data):
        dtfm = pd.DataFrame(dt["data"]['notif_pref'])
        for i in range(len(dtfm)):
            dtfm['notif_pref'][i]= ", ".join(dt['data']['notif_pref'][i]['notif_pref'])
        dtfm['ip_id'] = dt["data"]["ip_id"]
        dtfm['dt'] = dt["data"]['date']
        dtfm.to_sql(name= "np_tab", con= engine,if_exists = "replace", index=False)
        l=[]
        m=[]
        ip_id = dt["data"]["ip_id"]
        date = dt["data"]['date']

        for i in range(len(dtfm)):
            if "sms" in dtfm['notif_pref'][i]:
                cur.execute(f"update ph_tab set e_dt='{date}' where ip_id='{ip_id}'")
                conn.commit()
                q= pd.read_sql(f"SELECT ph from ph_tab where ip_id='{ip_id}'", engine)
                if q.ph[0] != None:
                    l.append("sms")
                else:
                    next
            if "eml" in dtfm['notif_pref'][i]:
                cur.execute(f"update e_tab set e_dt='{date}' where ip_id='{ip_id}'")
                conn.commit()
                q= pd.read_sql(f"SELECT eml from e_tab where ip_id='{ip_id}'", engine)
                if q.eml[0] != None:
                    l.append("eml")
                else:
                    next
            if "tg" in dtfm['notif_pref'][i]:
                cur.execute(f"update tg_tab set e_dt='{date}' where ip_id='{ip_id}'")
                conn.commit()
                q= pd.read_sql(f"SELECT tg from tg_tab where ip_id='{ip_id}'",engine)
                if q.tg[0] != None:
                    l.append("tg")
                else:
                    next
            m.append(l)
            l=[]


        for i in range(len(m)):
            m[i]= ", ".join(m[i])

        fd= pd.DataFrame(m, columns=['notif_pref'])
        for j in range(len(fd)):
            fd["notif_pref"][j]= list(fd['notif_pref'][j].split(", "))
        fd['notif_type_id']= dtfm['notif_type_id']
        final = {}
        final['ip_id'] =  dt['data']['ip_id']
        final['date'] =  dt['data']['date']
        final['notif_pref'] = fd.to_dict(orient="records")
        with open("sample_alert.json", "w") as outfile:
            return json.dump(final, outfile)
```

In [4]:

```
with open("alert.json") as file:
    dt= json.load(file)
at= alert(dt)
at.output(dt)
```

```
<ipython-input-2-d046cbf321ec>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  dtfm['notif_pref'][i]= ", ".join(dt['data']['notif_pref'][i]['notif_pref'])
```

In [ ]: