

# SQL with Python

OOPs concepts are extensively used to increase reusability of code

Every Step is logged into a file using custom logging class

## Data Description

Data Set used: Carbon Nanotubes

Data Set URL: <https://archive.ics.uci.edu/ml/datasets/Carbon+Nanotubes>

Number of Instances: 10721

By Arjun Panwar

Follow mw on LinkedIn:

<https://www.linkedin.com/in/arjun-panwar/>

## Logging Class

```
In [4]: from datetime import datetime #importing DateTime package
class App_Logger:
    """
    It is used save logs into a file

    Parameters
    -----
    file: log file name Default is logfile.log

    """
    def __init__(self, file="logfile.log"):
        self.f_name = file
    def log(self, log_type, log_msg):
        """
        Function log to save logs and log type in file

        Parameters
        -----
        log_type: Type of log-info,error,warning etc
        log_msg: Log to be saved(message)

        """
        now = datetime.now() #current time
        current_time = now.strftime("%d-%m-%Y %H:%M:%S") #changing time format
        f = open(self.f_name, "a+") #opening file in append + mode
        f.write(current_time+" "+log_type+" "+log_msg+"\n") #writing log type
        f.close() #closing log file
```

## SQL Class

```
In [5]: import mysql.connector as connection #importing MYSQL connector
```

```

import pandas as pd #importing pandas
class sql:
    """
    SQL class through with we can perform most of the SQL tasks using python

    Parameters
    -----
    host: host URL of MySQL server
    user: user name
    passwd: password
    db: database name- default empty string (")
    """

    def __init__(self,host,user,passwd,db=""):
        """
        init function of sql class
        """
        self.host=host
        self.user=user
        self.passwd=passwd
        self.db=db
        self.logger=App_Logger("logfile.txt") #creating App_Logger object
        self.logger.log("info", "SQL object created") #logging

    def conn(self):
        """
        Function conn is used to make connection to SQL server

        Parameters
        -----
        """
        try:
            if self.db=="":
                #connection without db
                return connection.connect(host=self.host,user=self.user,passwd=self.passwd)
            else:
                # connection with db
                return connection.connect(host=self.host,user=self.user, database=self.db)
        except Exception as e:
            self.logger.log("error", f"connection error : {str(e)}") #logging
            print(str(e))

    def db_list(self):
        """
        Function db_list is used to show databases list

        Parameters
        -----
        """
        try:
            conn=self.conn() #making connection
            cursor = conn.cursor() #create a cursor to execute queries
            q="SHOW DATABASES" #query
            cursor.execute(q) #executing Query
            print(cursor.fetchall()) #printing result
            conn.close() #connection closed
            self.logger.log("info", "DB list displayed") #logged
        except Exception as e:
            conn.close() #connection closed
            print(str(e))
            self.logger.log("error", f"db list error : {str(e)}") #logging

    def create_db(self,db_name):
        """
        Function create_db is used to create a new database
    """

```

```

Parameters
-----
db_name: database name
'''

try:
    conn=self.conn() #making connection
    cursor = conn.cursor() #create a cursor to execute queries
    cursor.execute(f"create database {db_name}") #executing Query
    self.db=db_name #Initializing database name to class variable so
    conn.close()#connection closed
    self.logger.log("info", f"{db_name} DB created") #logging
except Exception as e:
    conn.close()#connection closed
    print(str(e))
    self.logger.log("error", f"db not created error : {str(e)}") #log

def drop_db(self,db_name):
    '''
    Function drop_db is used to drop a database

    Parameters
    -----
    db_name: database name
    '''

    try:
        conn=self.conn() #making connection
        cursor = conn.cursor() #create a cursor to execute queries
        cursor.execute(f"drop database {db_name}") #executing Query
        conn.close() #connection closed
        self.logger.log("info", f"{db_name} DB dropped") #logging
    except Exception as e:
        conn.close()#connection closed
        print(str(e))
        self.logger.log("error", f"db not Dropped error : {str(e)}") #log

def create_table(self,table_name,columns):
    '''
    Function create_table is used to create a new table

    Parameters
    -----
    table_name: table name
    columns: columns names with data type and other discription in SQL fo
    '''

    try:
        conn=self.conn() #making connection
        cursor = conn.cursor() #create a cursor to execute queries
        cursor.execute(f"CREATE TABLE {table_name} ({columns})") #executi
        conn.close() #connection closed
        self.logger.log("info", f"{table_name} table created with columns
    except Exception as e:
        conn.close() #connection closed
        print(str(e))
        self.logger.log("error", f"table not created error : {str(e)}") #

def insert(self,table_name,data):
    '''
    Function insert is used to insert value in table

    Parameters
    -----
    table_name: table name
    data: values to be inserted

```

```

'''
try:
    conn=self.conn() #making connection
    cursor = conn.cursor() #create a cursor to execute queries

    cursor.execute(f"INSERT INTO {table_name} VALUES ({data})") #exec
    conn.commit() #commiting the query
    conn.close() #connection closed
except Exception as e:
    conn.close() #connection closed
    self.logger.log("error", f"insert error : {str(e)}") #logging

def dump_file(self,f_name,t_name,columns, csv=True):
    '''
    Function dump_file is used to dump a csv into a table

    Parameters
    -----
    f_name: file name
    t_name: table name
    columns: columns names with data type and other discription in SQL t
    csv: True if csv file is comma separated otherwise False if csv file
    '''
    try:
        f=open(f_name,"r") #opening file in read mode
        f.readline() #reading first line to skip columns line in file
        self.create_table(t_name,columns) #creating table

        for line in f.readlines(): #reading file line by line
            if csv:
                data="\'"+line[:-1].replace(",","\'","\'")+\'\' # data for
                print(data)
            else:
                data="\'"+line[:-1].replace(";","\'","\'")+\'\' # data form
                self.insert(t_name,data) #inserting data

        self.logger.log("info", f"{f_name} file data dumped to {t_name} t

    except Exception as e:
        print(str(e))
        self.logger.log("error", f"file dump error : {str(e)}") #logging

def select_db(self,db_name):
    '''
    Function select_db is used to select a database

    Parameters
    -----
    db_name: database name
    '''
    self.db=db_name #Initializing database name to class variable so tha
    self.logger.log("info", f"{db_name} DB selected") #logging

def columns(self,t_name):
    '''
    Function columns is used to print columns names

    Parameters
    -----
    t_name: table name
    '''
    try:

```

```

conn=self.conn() #making connection
cursor = conn.cursor() #create a cursor to execute queries
cursor.execute(f"SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE 1")
for result in cursor.fetchall(): #printing result
    print(result[3],end=",")
conn.close() #connection closed
self.logger.log("info", f"columns names displayed") #logging
except Exception as e:
    conn.close() #connection closed
    self.logger.log("error", f"columns name not displayed : {str(e)}")

def query(self,query):
    """
    Function query is used to run a SQL query

    Parameters
    -----
    query: SQL query
    """
    try:
        conn=self.conn() #making connection
        cursor = conn.cursor() #create a cursor to execute queries
        cursor.execute(query) #executing Query
        for result in cursor.fetchall(): #printing result line by line
            print(result)

        conn.close() #connection closed
        self.logger.log("info", f"Query is performed : {query} ") #logging
    except Exception as e:
        conn.close() #connection closed
        self.logger.log("error", f"Query not performed : {query} : {str(e)}")

def pd_query(self,query,h=5,t=5):
    """
    Function pd_query is used to run a SQL query using pandas
    I used it here to get better output format of table

    Parameters
    -----
    query: SQL query
    h: no. of results from head - default=5
    t: no. of results from tail - default=5
    """
    try:
        df=pd.read_sql_query(query,self.conn()) #executing Query using pandas
        print("Head") #printing head of dataframe
        print(df.head(h))
        if t>0: #printing Tail of dataframe if tail count is more than 0
            print("Tail")
            print(df.tail(t))

        self.logger.log("info", f"Query is performed : {query} ") #logging
    except Exception as e:
        print(str(e))
        self.logger.log("error", f"Query not performed : {query} : {str(e)}")

```

## Creating Class Object by passing HOST,USER,PASSWORD

```
In [6]: ob=sql("localhost","root","2001")
```

## Selecting DB UCI

```
In [7]: ob.select_db("UCI")
```

## Dumping CSV File

```
In [ ]: columns="CI_n INT(2),CI_m INT(2),IAC_u VARCHAR(10),IAC_v VARCHAR(10),IAC_w VA
ob.dump_file("carbon_nanotubes.csv","test5",columns)
```

```
In [8]: # columns names
ob.columns("test5")
```

CAC\_u,CAC\_v,CAC\_w,CI\_m,CI\_n,email,IAC\_u,IAC\_v,IAC\_w,

## The SQL SELECT QUERY

```
In [9]: #Select all columns
ob.pd_query("Select * from test5")
```

Head

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
0	2	1	0,679005	0,701318	0,017033	0,721039	0,730232	0,017014	
1	2	1	0,717298	0,642129	0,231319	0,738414	0,65675	0,232369	
2	2	1	0,489336	0,303751	0,088462	0,477676	0,263221	0,088712	
3	2	1	0,413957	0,632996	0,040843	0,408823	0,657897	0,039796	
4	2	1	0,334292	0,543401	0,15989	0,303349	0,558807	0,157373	

email

0 None  
1 None  
2 None  
3 None  
4 None

Tail

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
10716	12	6	0,834201	0,399891	0,89127	0,841858	0,405882	0,891356	
10717	12	6	0,698374	0,24471	0,962699	0,706555	0,248416	0,962833	
10718	12	6	0,923823	0,568913	0,819842	0,929403	0,576284	0,819879	
10719	12	6	0,934978	0,602319	0,938889	0,941844	0,610608	0,938755	
10720	12	6	0,953664	0,698374	0,962699	0,961243	0,707812	0,962605	

email

10716 None  
10717 None  
10718 None  
10719 None  
10720 None

```
In [10]: #Select IAC_u column
ob.pd_query("Select IAC_u from test5")
```

Head

	IAC_u
0	0,679005
1	0,717298
2	0,489336

```

3  0,413957
4  0,334292
Tail
      IAC_u
10716 0,834201
10717 0,698374
10718 0,923823
10719 0,934978
10720 0,953664

```

```

In [11]: #The SQL SELECT DISTINCT
ob.pd_query("Select Distinct CI_m from test5")

```

```

Head
  CI_m
0     1
1     2
2     3
3     4
4     5
Tail
  CI_m
1     2
2     3
3     4
4     5
5     6

```

```

In [12]: #The SQL WHERE CLAUSE
ob.pd_query("Select CI_n,IAC_u from test5 where CI_n=2")

```

```

Head
  CI_n  IAC_u
0     2 0,679005
1     2 0,717298
2     2 0,489336
3     2 0,413957
4     2 0,334292
Tail
  CI_n  IAC_u
23    2 0,287448
24    2 0,717298
25    2 0,489336
26    2 0,586043
27    2 0,394742

```

```

In [13]: #A WHERE clause with AND
ob.pd_query("Select CI_n,CI_m,IAC_u from test5 where CI_n=2 and CI_m=1")

```

```

Head
  CI_n  CI_m  IAC_u
0     2     1 0,679005
1     2     1 0,717298
2     2     1 0,489336
3     2     1 0,413957
4     2     1 0,334292
Tail
  CI_n  CI_m  IAC_u
23    2     1 0,287448
24    2     1 0,717298
25    2     1 0,489336
26    2     1 0,586043
27    2     1 0,394742

```

```

In [14]: #A WHERE clause with OR:
ob.pd_query("Select CI_n,IAC_u from test5 where CI_n=2 or CI_n=3")

```

```

Head
  CI_n    IAC_u
0     2  0,679005
1     2  0,717298
2     2  0,489336
3     2  0,413957
4     2  0,334292
Tail
  CI_n    IAC_u
151    3  0,209665
152    3  0,782959
153    3  0,767589
154    3  0,790611
155    3  0,453671

```

```

In [15]: #A WHERE clause with NOT
ob.pd_query("Select CI_n,CI_m,IAC_u from test5 where NOT CI_n=4 and CI_m=3")

```

```

Head
  CI_n  CI_m    IAC_u
0     5     3  0,745844
1     5     3  0,586209
2     5     3  0,518112
3     5     3  0,815513
4     5     3  0,777181
Tail
  CI_n  CI_m    IAC_u
1386   12     3  0,651999
1387   12     3  0,910487
1388   12     3  0,917929
1389   12     3  0,882125
1390   12     3  0,939273

```

```

In [16]: #Order By in Descending order
ob.pd_query("Select CI_n,CI_m,IAC_u from test5 ORDER BY IAC_u DESC")

```

```

Head
  CI_n  CI_m    IAC_u
0    12     6  0,954851
1    12     6  0,954851
2    12     6  0,953664
3    12     6  0,953664
4    12     6  0,953495
Tail
  CI_n  CI_m    IAC_u
10716   12     6  0,046505
10717   12     6  0,046336
10718   12     6  0,046336
10719   12     6  0,045149
10720   12     6  0,045149

```

```

In [17]: #Order By in Ascending order
ob.pd_query("Select CI_n,CI_m,IAC_u from test5 ORDER BY IAC_u ASC")

```

```

Head
  CI_n  CI_m    IAC_u
0    12     6  0,045149
1    12     6  0,045149
2    12     6  0,046336
3    12     6  0,046336
4    12     6  0,046505
Tail
  CI_n  CI_m    IAC_u
10716   12     6  0,953495
10717   12     6  0,953664

```



```

10718    12      6  0,953664
10719    12      6  0,954851
10720    12      6  0,954851

```

## The SQL MIN() AND MAX() FUNCTION

```

In [18]: ob.query("Select MIN(IAC_u) from test5")

('0,045149',)

```

```

In [19]: ob.query("Select MAX(IAC_u) from test5")

('0,954851',)

```

## The SQL COUNT(), AVG() AND SUM() FUNCTION

```

In [20]: ob.query("Select count(IAC_u) from test5")

(10721,)

```

```

In [21]: ob.query("Select AVG(CI_m) from test5")

(Decimal('3.3372'),)

```

```

In [22]: ob.query("Select sum(CI_m) from test5")

(Decimal('35778'),)

```

## The SQL LIKE-OPERATOR

```

In [23]: # IAC_u starting with 0,05
ob.pd_query("Select * from test5 WHERE IAC_u LIKE '0,05%'")

```

Head

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
0	10	6	0,058609	0,303689	0,048204	0,05414	0,301778	0,048155	
1	10	6	0,058632	0,254618	0,218272	0,053762	0,252121	0,21841	
2	10	6	0,057712	0,278699	0,299904	0,054158	0,276809	0,300036	
3	10	6	0,059773	0,24295	0,344122	0,056029	0,240607	0,344272	
4	10	6	0,058609	0,303689	0,381537	0,056117	0,302546	0,381544	

email

```

0 None
1 None
2 None
3 None
4 None

```

Tail

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
30	12	6	0,056805	0,189791	0,438889	0,055081	0,182476	0,438845	
31	12	6	0,0563	0,364847	0,39127	0,050443	0,357052	0,391238	
32	12	6	0,05006	0,332769	0,510318	0,044671	0,324699	0,510211	
33	12	6	0,0563	0,364847	0,724604	0,050196	0,356793	0,724611	
34	12	6	0,050398	0,215518	0,653175	0,046787	0,207383	0,653093	

email

```

30 None
31 None
32 None
33 None
34 None

```

```

In [24]:

```

```
# IAC_u ending with 1105
ob.pd_query("Select * from test5 WHERE IAC_u LIKE '%1105'")
```

Head

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
0	8	5	0,121105	0,154701	0,009571	0,11627	0,150668	0,009618	
1	8	5	0,121105	0,154701	0,342904	0,11627	0,150668	0,342952	
2	8	5	0,121105	0,154701	0,676237	0,11627	0,150668	0,676285	
3	10	1	0,331105	0,091347	0,010314	0,323769	0,079936	0,010593	
4	10	1	0,331105	0,091347	0,343648	0,323769	0,079936	0,343926	

email

0 None  
1 None  
2 None  
3 None  
4 None

Tail

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
1	8	5	0,121105	0,154701	0,342904	0,11627	0,150668	0,342952	
2	8	5	0,121105	0,154701	0,676237	0,11627	0,150668	0,676285	
3	10	1	0,331105	0,091347	0,010314	0,323769	0,079936	0,010593	
4	10	1	0,331105	0,091347	0,343648	0,323769	0,079936	0,343926	
5	10	1	0,331105	0,091347	0,676981	0,323769	0,079936	0,677259	

email

1 None  
2 None  
3 None  
4 None  
5 None

In [25]:

```
# IAC_u that have "345" in any position.
ob.pd_query("Select * from test5 WHERE IAC_u LIKE '%345%'")
```

Head

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
0	5	4	0,345522	0,12863	0,575547	0,342738	0,122228	0,57552	
1	5	4	0,345522	0,12863	0,90888	0,34165	0,120009	0,908953	
2	6	1	0,345989	0,694692	0,273276	0,343994	0,706144	0,273133	
3	6	1	0,405345	0,161314	0,2694	0,402438	0,156649	0,26943	
4	6	1	0,345989	0,694692	0,606609	0,336665	0,695821	0,607081	

email

0 None  
1 None  
2 None  
3 None  
4 None

Tail

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
44	11	2	0,073345	0,275766	0,84635	0,067427	0,271994	0,846455	
45	12	2	0,078345	0,383216	0,063437	0,071642	0,380444	0,063451	
46	12	2	0,345943	0,070284	0,078941	0,343653	0,065212	0,079095	
47	12	2	0,345943	0,070284	0,412274	0,343533	0,062967	0,412151	
48	12	2	0,078345	0,383216	0,730104	0,074681	0,381255	0,730075	

email

44 None  
45 None  
46 None  
47 None  
48 None

In [26]:

```
# IAC_u that starts with "0,44" and ends with "5":
ob.pd_query("Select * from test5 WHERE IAC_u LIKE '0,44%5'")
```

```

Head
  CI_n  CI_m    IAC_u    IAC_v    IAC_w    CAC_u    CAC_v    CAC_w  \
0     5     4  0,448065  0,794013  0,264071  0,449231  0,798812  0,263991
1     5     4  0,448065  0,794013  0,597404  0,447709  0,797716  0,59743
2     8     4  0,445855  0,825961  0,193956  0,447673  0,82842  0,194033
3     8     4  0,445855  0,825961  0,52729  0,447956  0,82808  0,52723
4     9     4  0,444105  0,833602  0,366292  0,442813  0,833973  0,366345

email
0 None
1 None
2 None
3 None
4 None
Tail
  CI_n  CI_m    IAC_u    IAC_v    IAC_w    CAC_u    CAC_v    CAC_w  \
1     5     4  0,448065  0,794013  0,597404  0,447709  0,797716  0,59743
2     8     4  0,445855  0,825961  0,193956  0,447673  0,82842  0,194033
3     8     4  0,445855  0,825961  0,52729  0,447956  0,82808  0,52723
4     9     4  0,444105  0,833602  0,366292  0,442813  0,833973  0,366345
5     9     4  0,444105  0,833602  0,699626  0,443531  0,835625  0,699442

email
1 None
2 None
3 None
4 None
5 None

```

## The SQL IN AND NOT IN OPERATORS

```
In [27]: ob.pd_query("Select * from test5 WHERE CI_n IN (5,1)")
```

```

Head
  CI_n  CI_m    IAC_u    IAC_v    IAC_w    CAC_u    CAC_v    CAC_w  \
0     5     1  0,680778  0,825935  0,026903  0,68039  0,83773  0,027645
1     5     1  0,335424  0,662004  0,010773  0,319948  0,658136  0,011449
2     5     1    0,493  0,779265  0,048408  0,483656  0,780342  0,049284
3     5     1  0,460003  0,760702  0,107548  0,449358  0,756888  0,108202
4     5     1  0,790058    0,775  0,064537  0,798133  0,788111  0,065134

email
0 None
1 None
2 None
3 None
4 None
Tail
  CI_n  CI_m    IAC_u    IAC_v    IAC_w    CAC_u    CAC_v    CAC_w
\
714    5     4  0,871532  0,656045  0,846038  0,878005  0,658528  0,846017
715    5     4  0,869273  0,638391  0,919809  0,87764  0,64117  0,919679
716    5     4  0,742121  0,375208  0,952596  0,744456  0,373794  0,952638
717    5     4  0,795071  0,449772  0,990847  0,798836  0,450544  0,990861
718    5     4  0,866034  0,620371  0,993579  0,874819  0,623677  0,993609

email
714 None
715 None
716 None
717 None
718 None

```

```
In [28]: ob.pd_query("Select * from test5 WHERE CI_n between 4 and 5")
```

Head

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
0	4	1	0,798626	0,65477	0,039988	0,812277	0,658326	0,040916	
1	4	1	0,794352	0,691116	0,111416	0,807185	0,697932	0,111388	
2	4	1	0,212784	0,427407	0,063797	0,198483	0,426673	0,063134	
3	4	1	0,560308	0,783502	0,016178	0,565293	0,801366	0,016747	
4	4	1	0,516033	0,766319	0,087607	0,518262	0,783893	0,087303	

email  
 0 None  
 1 None  
 2 None  
 3 None  
 4 None

Tail

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
1002	5	4	0,871532	0,656045	0,846038	0,878005	0,658528	0,846017	
1003	5	4	0,869273	0,638391	0,919809	0,87764	0,64117	0,919679	
1004	5	4	0,742121	0,375208	0,952596	0,744456	0,373794	0,952638	
1005	5	4	0,795071	0,449772	0,990847	0,798836	0,450544	0,990861	
1006	5	4	0,866034	0,620371	0,993579	0,874819	0,623677	0,993609	

email  
 1002 None  
 1003 None  
 1004 None  
 1005 None  
 1006 None

## The SQL ALIAS

In [29]: `ob.pd_query("Select CI_n as CI,IAC_u as IAC from test5")`

Head

	CI	IAC
0	2	0,679005
1	2	0,717298
2	2	0,489336
3	2	0,413957
4	2	0,334292

Tail

	CI	IAC
10716	12	0,834201
10717	12	0,698374
10718	12	0,923823
10719	12	0,934978
10720	12	0,953664

In [30]: `ob.pd_query("Select IAC_u from test5 Group By IAC_u ")`

Head

	IAC_u
0	0,679005
1	0,717298
2	0,489336
3	0,413957
4	0,334292

Tail

	IAC_u
4825	0,89448
4826	0,876454
4827	0,757164
4828	0,9437
4829	0,934978

In [31]:

```
ob.pd_query("Select count(IAC_v) from test5 Group By IAC_v ")
```

```
Head
count(IAC_v)
0           2
1           2
2           2
3           2
4           2
Tail
count(IAC_v)
4809        2
4810        2
4811        2
4812        2
4813        2
```

```
In [32]: ob.pd_query("Select IAC_v from test5 Group By IAC_v having count(IAC_v)>3 ")
```

```
Head
IAC_v
0  0,6371
1  0,850063
2  0,411791
3  0,149937
4  0,588209
Tail
IAC_v
25  0,309734
26  0,066833
27  0,690266
28  0,911781
29  0,088219
```

## The SQL CREATE DATABASE

```
In [33]: ob.create_db("Test_DB")
```

## DROP DATABASE

```
In [34]: ob.drop_db("Test_DB")
```

## check NOT NuLL

```
In [36]: ob.select_db("UCI")
ob.pd_query("SELECT * FROM test5 WHERE IAC_u IS NOT NULL")
```

```
Head
CI_n  CI_m  IAC_u  IAC_v  IAC_w  CAC_u  CAC_v  CAC_w  \
0     2     1  0,679005  0,701318  0,017033  0,721039  0,730232  0,017014
1     2     1  0,717298  0,642129  0,231319  0,738414  0,65675  0,232369
2     2     1  0,489336  0,303751  0,088462  0,477676  0,263221  0,088712
3     2     1  0,413957  0,632996  0,040843  0,408823  0,657897  0,039796
4     2     1  0,334292  0,543401  0,15989  0,303349  0,558807  0,157373

email
0  None
1  None
2  None
3  None
4  None
Tail
CI_n  CI_m  IAC_u  IAC_v  IAC_w  CAC_u  CAC_v  CAC_w
```

```

\
10716 12 6 0,834201 0,399891 0,89127 0,841858 0,405882 0,891356
10717 12 6 0,698374 0,24471 0,962699 0,706555 0,248416 0,962833
10718 12 6 0,923823 0,568913 0,819842 0,929403 0,576284 0,819879
10719 12 6 0,934978 0,602319 0,938889 0,941844 0,610608 0,938755
10720 12 6 0,953664 0,698374 0,962699 0,961243 0,707812 0,962605

email
10716 None
10717 None
10718 None
10719 None
10720 None

```

## ALTER TABLE -ADD and DROP COLUMN

```
In [37]: ob.pd_query("ALTER TABLE test5 ADD email varchar(25)")
ob.pd_query("Select * from test5",5,0)
```

Execution failed on sql 'ALTER TABLE test5 ADD email varchar(25)': 1060 (42S21): Duplicate column name 'email'

Head

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w	\
0	2	1	0,679005	0,701318	0,017033	0,721039	0,730232	0,017014	
1	2	1	0,717298	0,642129	0,231319	0,738414	0,65675	0,232369	
2	2	1	0,489336	0,303751	0,088462	0,477676	0,263221	0,088712	
3	2	1	0,413957	0,632996	0,040843	0,408823	0,657897	0,039796	
4	2	1	0,334292	0,543401	0,15989	0,303349	0,558807	0,157373	

```

email
0 None
1 None
2 None
3 None
4 None

```

```
In [38]: ob.pd_query("ALTER TABLE test5 DROP email")
ob.pd_query("Select * from test5",5,0)
```

'NoneType' object is not iterable

Head

	CI_n	CI_m	IAC_u	IAC_v	IAC_w	CAC_u	CAC_v	CAC_w
0	2	1	0,679005	0,701318	0,017033	0,721039	0,730232	0,017014
1	2	1	0,717298	0,642129	0,231319	0,738414	0,65675	0,232369
2	2	1	0,489336	0,303751	0,088462	0,477676	0,263221	0,088712
3	2	1	0,413957	0,632996	0,040843	0,408823	0,657897	0,039796
4	2	1	0,334292	0,543401	0,15989	0,303349	0,558807	0,157373

## ALTER modify COLUMN

```
In [39]: ob.pd_query("ALTER TABLE test5 MODIFY CI_n int NOT NULL")
ob.pd_query("desc test5",5,0)
```

'NoneType' object is not iterable

Head

	Field	Type	Null	Key	Default	Extra
0	CI_n	b'int'	NO		None	
1	CI_m	b'int'	YES		None	
2	IAC_u	b'varchar(10)'	YES		None	
3	IAC_v	b'varchar(10)'	YES		None	
4	IAC_w	b'varchar(10)'	YES		None	

## SQL CHECK on CREATE TABLE

```
In [40]: ob.pd_query("CREATE TABLE Persons (ID int NOT NULL, LastName varchar(255) NOT NULL,
ob.pd_query("desc Persons",5,0)
```

Execution failed on sql 'CREATE TABLE Persons (ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Age int, CHECK (Age>=18))': 1050 (42S01): Table 'Persons' already exists

Head

	Field	Type	Null	Key	Default	Extra
0	ID	b'int'	NO		None	
1	LastName	b'varchar(255)'	NO	MUL	None	
2	FirstName	b'varchar(255)'	YES		None	
3	Age	b'int'	YES		None	

```
In [41]: #drop check
ob.pd_query("ALTER TABLE Persons DROP CHECK Age")
ob.pd_query("desc Persons",5,0)
```

Execution failed on sql 'ALTER TABLE Persons DROP CHECK Age': 3821 (HY000): Check constraint 'Age' is not found in the table.

Head

	Field	Type	Null	Key	Default	Extra
0	ID	b'int'	NO		None	
1	LastName	b'varchar(255)'	NO	MUL	None	
2	FirstName	b'varchar(255)'	YES		None	
3	Age	b'int'	YES		None	

## SQL DEFAULT on CREATE TABLE

```
In [42]: ob.pd_query("CREATE TABLE Person1 (ID int NOT NULL, LastName varchar(255) NOT NULL,
ob.pd_query("desc Person1",5,0)
```

Execution failed on sql 'CREATE TABLE Person1 (ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Age int, City varchar(255) DEFAULT 'Sandnes')': 1050 (42S01): Table 'Person1' already exists

Head

	Field	Type	Null	Key	Default	Extra
0	ID	b'int'	NO	PRI	None	
1	LastName	b'varchar(255)'	NO		None	
2	FirstName	b'varchar(255)'	YES		None	
3	Age	b'int'	YES		None	
4	City	b'varchar(255)'	YES		None	

## DROP A DEFAULT CONSTRAINT

```
In [43]: ob.pd_query("ALTER TABLE Person1 ALTER City DROP DEFAULT")
ob.pd_query("desc Person1",5,0)
```

'NoneType' object is not iterable

Head

	Field	Type	Null	Key	Default	Extra
0	ID	b'int'	NO	PRI	None	
1	LastName	b'varchar(255)'	NO		None	
2	FirstName	b'varchar(255)'	YES		None	
3	Age	b'int'	YES		None	
4	City	b'varchar(255)'	YES		None	

## CREATE INDEX

```
In [44]: ob.pd_query("CREATE INDEX idx_lastname on Person1 (LastName)")
ob.pd_query("desc Person1",5,0)
```

'NoneType' object is not iterable

Head

	Field	Type	Null	Key	Default	Extra
--	-------	------	------	-----	---------	-------

	ID	b'int'	NO	PRI	None
1	LastName	b'varchar(255)'	NO	MUL	None
2	FirstName	b'varchar(255)'	YES		None
3	Age	b'int'	YES		None
4	City	b'varchar(255)'	YES		None

## CREATE UNIQUE INDEX

In [45]:

```
ob.pd_query("Create UNIQUE INDEX id on Person1 (ID)")
ob.pd_query("desc Person1",5,0)
```

Execution failed on sql 'Create UNIQUE INDEX id on Person1 (ID)': 1061 (4200 0): Duplicate key name 'id'

Head

	Field	Type	Null	Key	Default	Extra
0	ID	b'int'	NO	PRI	None	
1	LastName	b'varchar(255)'	NO	MUL	None	
2	FirstName	b'varchar(255)'	YES		None	
3	Age	b'int'	YES		None	
4	City	b'varchar(255)'	YES		None	

In [46]:

```
#drop Index
ob.pd_query("ALTER TABLE Person1 DROP INDEX index_lastname")
ob.pd_query("desc Person1",5,0)
```

'NoneType' object is not iterable

Head

	Field	Type	Null	Key	Default	Extra
0	ID	b'int'	NO	PRI	None	
1	LastName	b'varchar(255)'	NO		None	
2	FirstName	b'varchar(255)'	YES		None	
3	Age	b'int'	YES		None	
4	City	b'varchar(255)'	YES		None	

## CREATE VIEW Syntax

In [47]:

```
ob.pd_query("CREATE VIEW test_view AS SELECT CI_n,IAC_u,IAC_v FROM test5")
ob.pd_query("Select * from test_view")
```

'NoneType' object is not iterable

Head

	CI_n	IAC_u	IAC_v
0	2	0,679005	0,701318
1	2	0,717298	0,642129
2	2	0,489336	0,303751
3	2	0,413957	0,632996
4	2	0,334292	0,543401

Tail

	CI_n	IAC_u	IAC_v
10716	12	0,834201	0,399891
10717	12	0,698374	0,24471
10718	12	0,923823	0,568913
10719	12	0,934978	0,602319
10720	12	0,953664	0,698374

## DROPPING VIEWS

In [48]:

```
ob.pd_query("DROP VIEW test_view")
ob.pd_query("Select * from test_view")
```

'NoneType' object is not iterable



Execution failed on sql 'Select \* from test\_view': 1146 (42S02): Table 'UCI.test\_view' doesn't exist

## IF Statement in MySQL

```
In [49]: ob.pd_query("Select CI_n, IF(CI_n>4, 'MORE', 'LESS') from test5")
```

Head

	CI_n	IF(CI_n>4, 'MORE', 'LESS')
0	2	LESS
1	2	LESS
2	2	LESS
3	2	LESS
4	2	LESS

Tail

	CI_n	IF(CI_n>4, 'MORE', 'LESS')
10716	12	MORE
10717	12	MORE
10718	12	MORE
10719	12	MORE
10720	12	MORE

```
In [ ]:
```