**Assignment**

* Create a table in hourly base Granularity and by using python read that table.
* How to Handle Missing values with Statistical Method (mean, median , mode)
* Create Visualization using pandas

Granularity means lowest level of detail that you can find in table.

We write a python program by using the **Psycopg2 module** to connect to the PostgreSQL database and then perform some database or SQL transactions such as creating a table and then insert data to the table, updating a table data and delete records from the table using python code. We shall also see how to fetch table data from PostgreSQL database to python program.

**To Install Psycopg2 module:**

For Windows,

**pip install psycopg2**

For Linux/Mac,

**pip3 install psycopg2**

Make sure to use an up-to-date version of **pip** (you can upgrade it using something like pip install

-U pip).

Now Once you installed the module the next thing is to identify all the different credentials that you need to connect with your Database i.e..., Hostname,Database,Username,Password,Port-Id you can get most of the information from PG-Admin. Just go to your PG\_Admin Right Click on the Server name go to properties and go to Connections

* We Use Connection and Cursor Functions.
* We Place Our Database Connection Command inside Try and Except Block incase if the Data base connection Fails it does not Effect our Program it throw an exception.

Exception block capturing the Error and Print the Error.

* Cursor helps you to perform SQL operations.
* To Create a Cursor We Use a variable **cur = conn.cursor()**
* At the End of the Program we need to End close the Cursor **cur.close()**

**Same as conn.close()**

* Initially, we assign none for both conn and cur variables
* We Use Finally block to close connection and cursor if and only if the value of conn and cur is not none .
* If your Database connection never succeeded then conn variable would never have the connection object and never have close method it throws an error. In order to avoid we use none value for both conn and cur variable in the beginning.
* In Order to commit the transaction we use conn.commit( ) . it save the transactions into the database

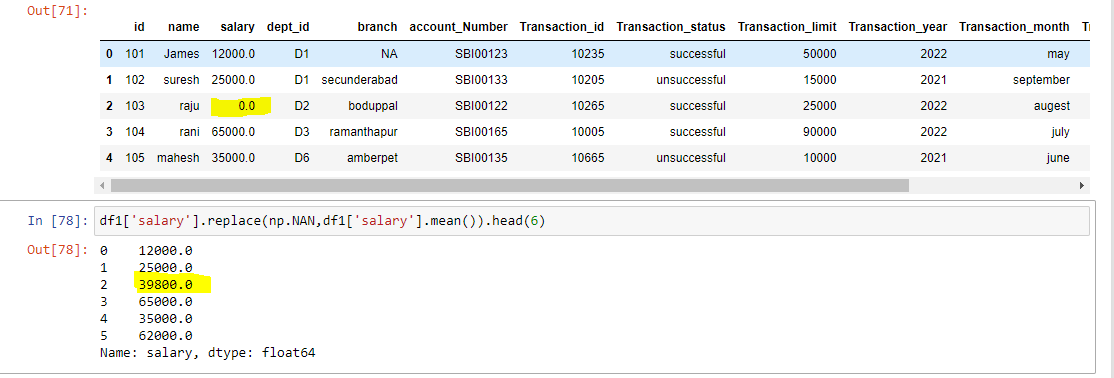
Syntax will be same as SQL command. We just need to place it in python program.

To execute the commands we use cur.execute( ) function.

We can fetch the data from database by using cur.fetchall( )

**How to Handle Missing values with Statistical Method (mean, median , mode)**

This strategy can be applied on a feature which has numeric data like the age of a person. We can calculate the mean, median or mode of the feature and replace it with the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns. Replacing with the above three approximations are a statistical approach of handling the missing values. This method is also called as leaking the data while training. Another way is to approximate it with the deviation of neighbouring values. This works better if the data is linear.



Pros:

* This is a better approach when the data size is small
* It can prevent data loss which results in removal of the rows and columns

Cons:

* Imputing the approximations add variance and bias
* Works poorly compared to other multiple-imputations method

**For Visualization we use pandas**

For Windows,

**pip install pandas**

For Linux/Mac,

**pip3 install pandas**

Chart

Description automatically generatedScatter plot can be created using the **DataFrame.plot.scatter()** methods.

**import psycopg2**

**import pandas as pd**

**hostname = 'localhost'**

**database = 'DataEngineering'**

**username = 'postgres'**

**pwd = 'Sadhan@1'**

**port\_id = 5432**

**conn = None**

**cur = None**

**try:**

**conn = psycopg2.connect(**

**host = hostname,**

**dbname = database,**

**user = username,**

**password = pwd,**

**port = port\_id)**

**cur = conn.cursor()**

**cur.execute('DROP TABLE IF EXISTS employee1')**

**create\_script = '''CREATE TABLE if not exists employee1(id int ,**

**name varchar(40),**

**salary int,**

**dept\_id varchar(30),**

**branch varchar(32),**

**account\_Number varchar(32),**

**Transaction\_id int,**

**Transaction\_status varchar(42),**

**Transaction\_limit int,**

**Transaction\_year int,**

**Transaction\_month varchar(21),**

**Transaction\_day varchar(32),**

**Transaction\_hour int) '''**

**cur.execute(create\_script)**

**insert\_script = '''insert into employee1(id,name,salary,dept\_id,branch,account\_Number,Transaction\_id,Transaction\_status,Transaction\_limit,Transaction\_year,Transaction\_month,Transaction\_day,Transaction\_hour)**

**values (101,'James', 12000, 'D1',NULL,'SBI00123',10235,'successful',50000,2022,'may','monday',10),**

**(102,'suresh', 25000, 'D1','secunderabad','SBI00133',10205,'unsuccessful',15000,2021,'september','tuesday',06),**

**(103,'raju', Null, 'D2','boduppal','SBI00122',10265,'successful',25000,2022,'augest','sunday',15),**

**(104,'rani', 65000, 'D3','ramanthapur','SBI00165',10005,'successful',90000,2022,'july','wednesday',18),**

**(105,'mahesh', 35000, 'D6','amberpet','SBI00135',10665,'unsuccessful',10000,2021,'june','monday',Null),**

**(106,'ganesh', 62000, Null,'begumpet','SBI00178',10465,'None',30000,2022,'april','friday',02)'''**

**cur.execute(insert\_script)**

**cur.execute('select \* from employee1')**

**data = cur.fetchall()**

**df1 = pd.DataFrame(data=data,columns=['id','name','salary','dept\_id','branch','account\_Number','Transaction\_id','Transaction\_status','Transaction\_limit','Transaction\_year','Transaction\_month','Transaction\_day','Transaction\_hour'])**

**#print(df1)**

**print(df1.isna().sum())**

**print(df1.isna().mean())**

**cur.execute('''SELECT avg(Transaction\_hour),Transaction\_day FROM employee1**

**group by Transaction\_day''')**

**data = cur.fetchall()**

**conn.commit()**

**df = pd.DataFrame(data=data,columns=['hour','day'])**

**print(df)**

**except Exception as error:**

**print(error)**

**finally:**

**if cur is not None:**

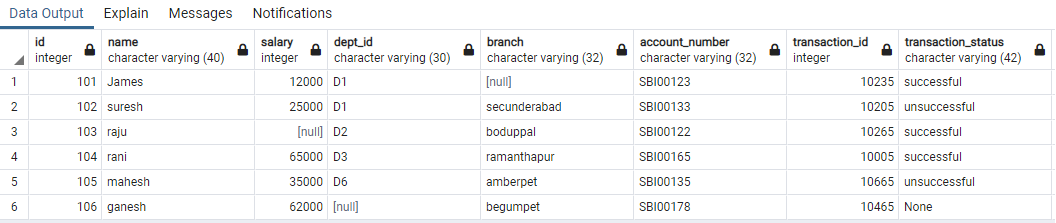
**cur.close()**

**if conn is not None:**

**conn.close()**

Table

Description automatically generated with medium confidence



**df1.info()**

Graphical user interface, application

Description automatically generated

**subset = ['salary', 'Transaction\_hour']**

**df1.loc[:, subset] = df1.loc[:, subset].fillna(0)**

Graphical user interface

Description automatically generated**df1**

**df1.fillna("NA",inplace=True)**

**df1.head()**

Graphical user interface

Description automatically generated

**df1['salary'].replace(np.NAN,df1['salary'].mean()).head(6)**

Graphical user interface, text, application, email

Description automatically generated

**df = pd.DataFrame(data=data,columns=['Transaction\_hours','Transaction\_day'])**

Graphical user interface, application

Description automatically generated**df.plot.scatter(x='Transaction\_hours', y='Transaction\_day')**