#--------------------

#Python library files

#--------------------

from \_libraryAndDBConnection import \* #includes database connection and cursor setting strings

#--------------------

#custom library files

#--------------------

import \_master #for master table data handling

import \_edit #for updating tables data

import \_report #for selecting tables data

import \_transaction #for transaction table data handling

import \_dataAnalysisAndPlot #for data analysis & plotting

import \_database #for database management

import \_createMySQLTablesWithTestData #for creating MySQL tables with test data

import \_dataExportImport

import \_reportCustom

import \_dataAnalysisAndPlotCustom

import \_pdfFileGeneration

import \_about

#--------------------

def donothing():

pass

#--------------------

#--------------------

MENU function

#--------------------

**def menu(rootframe):**

menubar = Menu(rootframe)

**# Master Menu Items**

mastermenu = Menu(menubar, tearoff=0)

# master menu item 1

mastermenuitem1parameters = { 'table':[''],

'pk':[''],

'cbo':[''],

'autofill':[]}

mastermenu.add\_command( label="Menu Item 1 Label",

command=lambda:\_master.createInsertRootFrame(rootframe, mastermenuitem1parameters) )

# master menu item 2

:

# master menu item 3

:

…

# setting MASTER menu label on menubar

menubar.add\_cascade(label="MASTER", menu=mastermenu)

**# Edit Menu Item**

editmenu = Menu(menubar, tearoff=0)

# mention any one of the table's details as to set it open by default for update

# user can select other tables from the dropdown

editmenuparameters = {'table':[''],'pk':[''],'cbo':[''],'autofill':['']}

editmenu.add\_command( label="Update Database Table Data",

command=lambda: \_edit.createEditRootFrame(rootframe, editmenuparameters))

menubar.add\_cascade(label="EDIT", menu=editmenu)

**# Transaction Menu Items**

#Only the columns mentioned in the following parameters will be placed on the form controls.

transmenu = Menu(menubar, tearoff=0)

# trans menu item 1

transmenuitem1parameters = {

'table' : ['tblname'],

'pk' : ['colname'],

'dateColumn' : 'colname',

'pickup' : [

{

'masterCBO' : ['parentcbotblname.colname#childcbotblname.colname'],

'masterLookupItems' : ['tblname.colname'],

'getvalfromothertableonlookupcboevent' : [],

'masterPrimaryKeys' : ['tblname.colname'],

'masterAutofillValues' : ['tblname.colname'],

'condition' : [],

},

{

…

},

],

'transItems' : ['colname'],

'expressions' : ['textfieldname1 = expression']

'displayonlynoinsert' : ['colname = colname.get()],

'invisible' : ['colname = colname.get()',],

'prefilled' : ['textfieldname1 = val|expression',],

'masterUpdates' : ['tblname.colname1 = val|expression',],

}

transmenu.add\_command( label="Trans Menu Item 1 Label",

command=lambda: \_transaction.createTransRootFrame(rootframe, transmenuitem1parameters) )

# trans menu item 2

:

# trans menu item 3

:

…

# setting TRANSACTIONS menu label on menubar

menubar.add\_cascade(label="TRANSACTIONS", menu=transmenu)

**# REPORT Menu Items**

reportmenu = Menu(menubar, tearoff=0)

rptparam1 = {'table':['item'],'pk':['itemcode'],'cbo':['itemcategory.itemcategory']}

reportmenu.add\_command(label="Onscreen Reports",

command=lambda: \_report.createSelectRootFrame(rootframe, rptparam1))

reportmenu.add\_separator()

reportmenu.add\_command(label="Items Fixed",

command=lambda: \_reportCustom.rptItemDetail(rootframe))

reportmenu.add\_command(label="Items Filtered Combobox",

command=lambda: \_reportCustom.rptItemDetailConditionalPickupList(rootframe))

reportmenu.add\_command(label="Items Filtered Textbox",

command=lambda: \_reportCustom.rptItemDetailConditionalTextbox(rootframe))

reportmenu.add\_command(label="Generate PDF",

command=lambda: \_pdfFileGeneration.invoicePurchase(rootframe))

menubar.add\_cascade(label="REPORT", menu=reportmenu)

**# DATA ANALYSIS Menu Items**

damenu = Menu(menubar, tearoff=0)

daparam1 = {'table':['item'],'pk':['itemcode'],'cbo':['itemcategory.itemcategory']}

damenu.add\_command(label="Data Analysis - Complete",

command=lambda: \_dataAnalysisAndPlot.createDataAnalysisRootFrame(rootframe, daparam1))

damenu.add\_command(label="Data Analysis",

command=lambda: \_dataAnalysisAndPlotCustom.plotItemGSTRate(rootframe))

damenu.add\_command(label="Data Analysis - Plot", command=donothing)

damenu.add\_command(label="Data Analysis - Statistics", command=donothing)

#damenu.add\_command(label="Data Analysis - Filter (Boolean Indexing)",

command=lambda: dataAnalysisFilterData.dataAnalysis(db))

menubar.add\_cascade(label="DATA ANALYSIS", menu=damenu)

**# DATABASE Menu Items**

dbmenu = Menu(menubar, tearoff=0)

dbmenu.add\_command(label="Export-Import Data",

command=lambda: \_dataExportImport.index(rootframe))

#newdatabase = "d1234"

#dbmenu.add\_command(label="Create Database", command=lambda:

\_database.createNewDatabase(rootframe,newdatabase))

dbmenu.add\_command(label="Create Table", command=lambda:

\_createMySQLTablesWithTestData.createTablesWithTestData(rootframe))

dbmenu.add\_command(label="Backup Database", command=lambda:

\_database.backupDatabase(rootframe))

dbmenu.add\_command(label="Alter Table", command=lambda: \_database.alterTable(rootframe))

#dbmenu.add\_command(label="Restore Database", command=lambda:

\_database.restoreDatabase(rootframe))

#dbmenu.add\_command(label="Reset Database", command=lambda:

\_database.resetDatabase(rootframe))

#dbmenu.add\_command(label="Drop Table", command=donothing)

menubar.add\_cascade(label="DATABASE", menu=dbmenu)

**# HELP Menu Items**

helpmenu = Menu(menubar, tearoff=0)

helpmenu.add\_command(label="About", command= \_about.manual)

helpmenu.add\_command(label="Manual & Guide", command= \_about.manual)

helpmenu.add\_command(label="Help", command= \_about.help)

helpmenu.add\_command(label="Contact", command= \_about.contact)

menubar.add\_cascade(label="HELP", menu=helpmenu)

**# EXIT Menu Items**

def clearcursorandconnection():

#if cursor.open:

# cursor.close()

if conn.open:

conn.close() #it will close its dependent cursor on its own

def exitapp():

#rootframe.quit() #NOT RECOMMENDED

clearcursorandconnection()

rootframe.destroy()

exitmenu = Menu(menubar, tearoff=0)

exitmenu.add\_command(label="Close Cursor & Connection", command=clearcursorandconnection)

exitmenu.add\_command(label="Exit Application", command=exitapp) #rootframe.destroy

menubar.add\_cascade(label="Exit", menu=exitmenu)

**# menu function terminates**

return menubar

**NOTES:**

* Use different variable names for the dictionaries of parameters, otherwise, only the last most parameters dictionary will be used.
* Specify multiple values for a parameter by using ['p1','p2',…] format. Where p1 itself may be in the format 'tblname.colname'.
* Do not discard or change the 'keys' of the parameter dictionaries of Master, Transaction and other menu items.
* If any parameter is not required in any master menu item, place '' (empty single quotes) within [] for that parameter.
* If any parameter is not required in any transaction menu item, keep [] blank for that parameter. Do not use even empty single quotes ''.
* Do not name any date type column as 'date'. It should be named something as 'transdate', 'joindate' etc.
* Use 'lambda' keyword to pass arguments to the function of the same or a different file.

**DATE BASED EXPRESSIONS**

* 'datestobeitemed = pd.date\_range( roomitemingdateend.get(),

roomitemingdatestart.get()-timedelta(days=1),

freq="d"

).strftime("%Y%m%d").tolist()',

* 'staynumofdays = datetime.strptime(roomitemingdateend.get(),"%Y%m%d") –

datetime.strptime(roomitemingdatestart.get(),"%Y%m%d")'

* DATE arithmatics

from datetime import datetime, date, timedelta

today = datetime.today()

print(today + timedelta(days=1))

'returndate = datetime.strptime(str((datetime.strptime(issuedate.get(),"%Y%m%d") +

timedelta(days=int(numofdaysallowed.get()))).date()), "%Y-%m-%d").strftime("%Y%m%d")'

**Master Menu Item Parameters:**

* table name of the table which is associated with this menu item
* pk name of the 'primary key' column of the 'table'

Need not mention it explicitly, ID column 'tablenameid' being handled automatically.

e.g.

"itemid INT PRIMARY KEY AUTO\_INCREMENT" of the table 'item' creates and uses the primary key 'itemid' automatically.

'pk' parameter need not be the primary key of the table; it can be any other column in the table for reference.

* cbo ['tblname.colname',]

combo box to be filled with lookup values from a column of another table

* autofill

'autofill':['cbocolname#tblname.colname']

Mention the list of column names, if any, in autofill other than the pickup list column name itself.

To fill multiple textfileds with data values from other tables upon selecting a value in the combo box drop-down!

Representing the SQL query:

"sql = select "+colname+" from "+tblname+" where "+cbocolname+" =

'"+comboselectedval+"'"

e.g.

'autofill':['itemcategory#itemcategory.itemgstrate']

Represents SQL query:

Select itemgstrate from itemcategory where itemcategory=2

Example Master menu Items

masteritem = { 'table':['item'],

'pk':['itemid'],

'cbo':['itemcategory.itemcategory', 'itemunit.itemunit'],

'autofill':[]

}

mastermenu.add\_command( label="Item",

command=lambda: \_master.createInsertRootFrame(rootframe, newmnuitem)

)

**Transaction Menu Item Parameters:**

* table

name of the transaction table

e.g.

'table' : ['itemsale'],

* pk

name of the 'primary key' column of the 'table'

* dateColumn

name of the date column of the transaction 'table' for which calendar is to be used

e.g.

'dateColumn' : 'saledate',

* pickup

creates parent-child or independent/stand-alone pick-up lists.

'pickup' : [

{

'masterCBO' : ['parentcbotblname.colname#childcbotblname.colname'],

'masterLookupItems' : ['tblname.colname'],

'getvalfromothertableonlookupcboevent' : [],

'masterPrimaryKeys' : ['tblname.colname'],

'masterAutofillValues' : ['tblname.colname'],

'condition' : [],

},

…

],

where

* masterCBO the master combobox whose selection will reset the drop-down list of its child combobox.

mastercbotblname.colname#childcbotblname.colname

* masterLookupItems the combobox whose selection will fill the textfields with 'autofill' values. Leave any parameter including mastercbo blank by using '[],' expression.

Use

'masterLookupItems' : ['item.itemname']

so as to update the total quantity in items table in addition to the stock updation in itempurchase table

* condition SQL condition to filter the rows used to fill the combo box.

'condition' : ['tblname.colname < + val',],

where '+' separates the colname and the value of a form control for dynamic conditional checks

e.g.

'item.itemcode < + int(price.get())'

'tblname.colname < + '\"N\"'

Use escape char '\"' to have embedded single quote for generating SQL queries.

# CONDITION - To add new features in next version of this software:

['item.itemcode between + int(invoice.get() and int(price.get())'],

Or

['"item.itemcode between {} and {}".format(invoice.get(),price.get())'],

Or

['''"CONCAT(',', roomitemeddates, ',') like

'%,{},%'".format(roomitemingdatestart.get())'''],

Or

['"find\_in\_set({},roomitemeddates)<>0".format(roomitemingdatestart.get())'],

where roomitemeddates is a VARCHAR containg comma separated dates

e.g.

for SQL query:

select \* from shirts where find\_in\_set('34',size)<>0

use the 'condition' parameter as:

'condition' : ['"find\_in\_set({},roomitemeddates)=0"

.format(roomitemingdatestart.get())],

Or

'condition' : ['"find\_in\_set({},roomitemeddates)=0 and roomcategory={}"

.format(roomitemingdatestart.get(),roomcategory.get())'],

WORKING EXAMPLE:

'condition' : ['"find\_in\_set({},roomitemeddates)=0 and

roomcategory=\'{}\'".format(roomitemingdatestart.get(),roomcategory.get())'],

* transItems

names of the columns of the trans table which are to be filled-in by the user.

e.g.

'transItems' : ['colname'],

* expressions

'expressions' : [

'textfieldname1=int(textfieldname2.get())+int(textfieldname3.get())'

]

Use ONLY colname (without 'tblname.') on right hand side of an expression.

e.g.

'expressions':[

'amount = float(itemquantity.get())\*float(itemprice.get())',

'gst = round(float(amount.get())\*float(itemgstrate.get())/100.00,2)',

],

* displayonlynoinsert

'displayonlynoinsert' : ['colname = colname.get()],

Data from master or other tables to be displayed on the screen but not to be inserted into the transaction table.

* invisible

'invisible' : ['colname = colname.get()',],

The assignment which is to be saved in 'transaction' table BUT NOT to be displayed on the screen.

e.g. in an inventory system using AVG PRICE (instead of LIFO or FIFO), item table contains average item price which is to be updated after every purchase.

Also use it to get the values for the pickup combo boxes where there are no corresponding textfields for them on the form.

* prefilled

fill a textfield with some initial value

e.g.

'prefilled' : ['fright=0.00'],

* masterUpdates

'masterUpdates' : ['tblname.colname1 = 'val'|expression,],

e.g.

# to reset the stock in the master table 'item'

'item.itemtotalstock = float(itemtotalstock.get())+float(itemquantity.get())',

**EXAMPLE OF TRANSACTION MENU ITEM**

paramitemsale = {

'table' : ['itemsale'],

'pk' : ['saleid'],

'dateColumn' : 'sdate',

'prefilled' : [],

'pickup' : [

{

'masterCBO' : ['itempurchase.itemcategory#itempurchase.itemname'],

'masterLookupItems' : ['itempurchase.itemname'],

'getvalfromothertableonlookupcboevent' : [],

'masterPrimaryKeys' : ['itempurchase.itempurchaseid'],

'masterAutofillValues' : ['itempurchase.itemminsaleprice',

'itempurchase.itemcode','itempurchase.itemstock',

'itempurchase.itemunit','itempurchase.itemgstrate'],

'condition' : ['"itempurchase.itemstock > 0"'],

},

{

'masterCBO':['customer.customercategory#customer.customercode'],

'masterLookupItems' : ['customer.customercode'],

'getvalfromothertableonlookupcboevent' : [],

'masterPrimaryKeys' : ['customer.customerid'],

'masterAutofillValues':['customer.customername','customer.customermobile'],

'condition' : [],

},

],

'transItems' : ['invoice','itemquantity','fright','marginpercent'],

'expressions' : [

'itemprice = round(float(itemminsaleprice.get()) +

float(itemminsaleprice.get())\*float(marginpercent.get())/100,2)',

'amount = round(float(itemprice.get())\*float(itemquantity.get()),2)',

'netamount = float(amount.get())+float(fright.get())'

],

'displayonlynoinsert' : [],

'invisible' : [

'customercategory = customercategory.get()',

'customercode = customercode.get()',

'customername = customername.get()',

'customermobile = customermobile.get()',

'itemcategory = itemcategory.get()',

'itemcode = itemcode.get()',

'itemname = itemname.get()',

'itemunit = itemunit.get()',

],

'masterUpdates' : [

'itempurchase.itemstock= float(itemstock.get())-float(itemquantity.get())',

'item.itemtotalstock=float(itemtotalstock.get())-float(itemquantity.get())',

],

}

transmenu.add\_command(label="Item Sale",

command=lambda: \_transaction.createTransRootFrame(rootframe, paramitemsale))

menubar.add\_cascade(label="TRANSACTIONS", menu=transmenu)