Coverage for **userdata_insertion.py** : 84%

126 statements | 106 run | 20 missing | 0 excluded

```
1   """Example for Mysql database connectivity with python """
2   from datetime import date
3   from datetime import datetime
4   import logging
5   import json
6   from dateutil import relativedelta
7   import mysql.connector
8
9   def connection_db():
10      """making database connection"""
11      mydb = mysql.connector.connect(user='root', password='root',
12                                     host='localhost',
13                                     database='db1')
14      return mydb
15
16  def logger_creation():
17      """Creating logger object"""
18      logger = logging.getLogger(__name__)
19      logger.setLevel(logging.INFO)
20      formatter = logging.Formatter('%(asctime)s:%(levelname)s:%(name)s:%(message)s')
21      file_handler = logging.FileHandler('logFile.log', mode='w')
22      file_handler.setFormatter(formatter)
23      logger.addHandler(file_handler)
24      return logger
25
26  def table_creation(mycursor):
27      '''Creating Request_Info and Response_Info tables'''
28      mycursor.execute("create table if not exists Request_Info(Request_Id int NOT NULL \
29              AUTO_INCREMENT,FirstName varchar(50),MiddleName varchar(50),LastName varchar(50),\
30              DOB date,Gender varchar(50),Nationality varchar(50),Current_City varchar(50),\
31              State varchar(50),Pin_Code int,Qualification varchar(50),Salary int,\
32              PAN varchar(50),Request_Date date,primary key(Request_Id))")
33
34      mycursor.execute("create table if not exists Response_Info(Response_Id int not null \
35              AUTO_INCREMENT,Request_Id int,Response varchar(255),primary key(Response_Id),\
36              foreign key(Request_Id) references Request_Info(Request_Id))")
37      mydb.commit()
38
39
40
41  class User:
42      """User class represents user typical structure"""
43      def __init__(self):
44          self.first_name = None
45          self.middle_name = None
46          self.last_name = None
47          self.dob = None
48          self.gender = None
49          self.nationality = None
50          self.current_city = None
51          self.state = None
```

```
52          self.pin_code = None
53          self.qualification = None
54          self.salary = None
55          self.pan = None
56          self.reason =""
57
58      def add_user(self, user_data):
59          """ Used for adding user data to Request_Info tabe"""
60          self.first_name = user_data[0]
61          self.middle_name = user_data[1]
62          self.last_name = user_data[2]
63          self.dob = datetime.strptime(user_data[3],'%Y-%m-%d').date()
64          self.gender = user_data[4]
65          self.nationality = user_data[5]
66          self.current_city = user_data[6]
67          self.state = user_data[7]
68          self.pin_code = int(user_data[8])
69          self.qualification = user_data[9]
70          self.salary = int(user_data[10])
71          self.pan = user_data[11]
72
73          sql1 = "insert into Request_Info(FirstName,MiddleName,LastName,DOB,Gender,Nationality,\
74          Current_City,State,Pin_Code,Qualification,Salary,PAN,Request_Date )\
75           values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
76          data = (self.first_name, self.middle_name, self.last_name, self.dob, self.gender,
77                  self.nationality,self.current_city,self.state, self.pin_code,
78                  self.qualification, self.salary, self.pan,datetime.now().date())
79          mycursor.execute(sql1, data)
80          mydb.commit()
81
82      def age_eligibility(self):
83          """Checks for age eligibility"""
84          diff = relativedelta.relativedelta(date.today(), self.dob)
85          if self.gender.lower() == "male":
86              if diff.years <= 21:
87                  self.reason = "age is less than expected"
88          elif self.gender.lower()== "female":
89              if diff.years <= 18:
90                  self.reason = "age is less than expected"
91          logger.info("age_eligibilty method executed")
92
93      def nationality_eligibility(self):
94          """Used for vaidating nationality"""
95          if self.nationality.lower() not in ['indian', 'american']:
96              self.reason = self.reason+", Nationality is not matched"
97          logger.info("nationality_eligibility method executed")
98
99      def state_eligibility(self):
100         """Validates state of user"""
101         state_list = ['andhra-pradesh', 'arunachal-pradesh', 'assam', 'bihar', 'chhattisgarh',\
102                       'karnataka','madhya-pradesh','odisha', 'tamil-nadu',\
103                       'telangana', 'west-bengal']
104         if self.state.lower() not in state_list:
105             self.reason = self.reason+", State is not matched in the list"
106         logger.info("state_eligibility method method executed")
107
108     def salary_eligibility(self):
```

```
109         '''Used for checking salary validaton'''
110         if self.salary < 10000 or self.salary > 90000:
111             self.reason = self.reason+", salary is not matching the range"
112         logger.info("salary_eligibility method executed")
113 
114     def request_eligibility(self):
115         '''Checks for request received within 5 days'''
116         data=(self.pan,)
117         print(self.pan)
118         sql1="select Request_Date from Request_Info where PAN=%s"
119         mycursor.execute(sql1,data)
120         request_date = mycursor.fetchone()
121         print(request_date[0])
122         diff=relativedelta.relativedelta(date.today(),request_date[0])
123         print(diff.days)
124         if diff.days <= 5 and diff.days !=0:
125             self.reason=self.reason+", Recently request is received within 5 days "
126         logger.info("request_eligibility method is executed")
127 
128     def add_response(self):
129         ''' Used for adding Json response to Response_Info table'''
130         if len(self.reason) == 0:
131             response = "Success"
132             json_obj = json.dumps({"Request_Id": mycursor.lastrowid, "Response": response})
133             sql1 = "insert into Response_Info(Request_Id,Response) values(%s,%s)"
134             data = (mycursor.lastrowid, json_obj)
135             mycursor.execute(sql1, data)
136             mydb.commit()
137             with open('response.txt', 'a',encoding='utf8') as outfile:
138                 json.dump({"Request_Id": mycursor.lastrowid, "Response": response}, outfile)
139                 outfile.write('\n')
140             logger.info("Response is added successfully")
141         else:
142             response = "Failure"
143             json_obj = json.dumps({"Request_Id": mycursor.lastrowid, "Response": response,\
144                                    "Reason": self.reason})
145             sql1 = "insert into Response_Info(Request_Id,Response) values(%s,%s)"
146             data = (mycursor.lastrowid, json_obj)
147             mycursor.execute(sql1, data)
148             mydb.commit()
149             with open('response.txt', 'a',encoding='utf8') as outfile:
150                 json.dump({"Request_Id": mycursor.lastrowid, "Response": response,\
151                            "Reason": self.reason}, outfile)
152                 outfile.write('\n')
153             logger.info("Response is added successfully")
154 
155 
156 
157 if __name__ == "__main__":
158     mydb=connection_db()
159     mycursor = mydb.cursor()
160     logger=logger_creation()
161     table_creation(mycursor)
162     try:
163         user_data = list(map(str, input("Enter firstname, middlename, lastname, DOB, gender,\
164          Nationality,Current-city,state, pin-code, Qualification, salary, PAN-number: ").split())
165 
```

```
166         user = User()
167         user.add_user(user_data)
168         user.age_eligibility()
169         user.nationality_eligibility()
170         user.state_eligibility()
171         user.salary_eligibility()
172         user.request_eligibility()
173         user.add_response()
174     except Exception:
175         logger.error("Error occured in main")
```

« index coverage.py v5.5, created at 2021-09-14 09:24 +0530