dennislamcv1 last year

897 lines (897 loc) · 52.9 KB

Preview Code Blame

Database Engineer Capstone Week 3

Import Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random

import datetime
from datetime import datetime, timedelta
import scipy.stats


import mysql.connector as connector

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
#sets the default autosave frequency in seconds
%autosave 60
sns.set_style('dark')
sns.set(font_scale=1.2)
```

```
plt.rc('axes', titlesize=9)
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=12)
plt.rc('ytick', labelsize=12)

pd.set_option('display.max_columns',None)
#pd.set_option('display.max_rows',None)
pd.set_option('display.width', 1000)
pd.option_context('float_format','{:.2f}'.format)

random.seed(0)
np.random.seed(0)
np.set_printoptions(suppress=True)
```

```
Autosaving every 60 seconds
```

Load Data from SQL database

Your first step is to import the connector module, enter your user details and connect with the database (Hint: you can use an alias when importing the module).

MySQL

In [2]:
```
#Create a connection

mydb = connector.connect(
  user="root",
  passwd="password",
  database="littlelemondb",
  auth_plugin='mysql_native_password'
)
```

In [3]:
```
print(mydb)
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x00
0002884E500490>
```

In [4]:
```
mycursor = mydb.cursor()
```

In [5]:
```
mycursor
```

Out[5]:
```
<mysql.connector.cursor_cext.CMySQLCursor at 0x2884e500880>
```

In [6]:
```
mydb.reconnect() #Reconnect cursor
```

In this second task, you now need to query the database to show all tables within the database.

In [6]:
```
pd.read_sql_query('SHOW tables', mydb)
```

Out[6]:

	Tables_in_littlelemondb
0	booking
1	customer
2	deliverystatus
3	menu
4	orders
5	ordersview
6	ordersview2
7	staff

In [8]:
```
pd.read_sql_query("SHOW columns FROM booking", mydb)
```

Out[8]:

	Field	Type	Null	Key	Default	Extra
0	BookingID	b'int'	NO	PRI	None	
1	BookingDate	b'datetime'	NO		None	
2	TableNumber	b'int'	NO		None	

An alternate way to learn the same information would be to use the DESCRIBE function. The syntax is:

In [9]:
```
pd.read_sql_query("DESCRIBE booking", mydb)
```

Out[9]:

	Field	Type	Null	Key	Default	Extra
0	BookingID	b'int'	NO	PRI	None	
1	BookingDate	b'datetime'	NO		None	
2	TableNumber	b'int'	NO		None	

For the third and final task, Little Lemon need you to return specific details from your database. They require the full name and contact details for every customer that has placed an order greater than $60 for

a promotional campaign.

In [10]:
```
pd.read_sql_query("SELECT * FROM customer", mydb)
```

Out[10]:

CustomerID	FullName	ContactNumber	Email

In [11]:
```
pd.read_sql_query("SELECT * FROM orders", mydb)
```

Out[11]:

OrderID	Date	Quantity	TotalCost	Booking_BookingID	Customer_Cu

In [7]:
```
pd.read_sql_query("""SELECT customer.FullName, customer.Contact
                     INNER JOIN orders
                     ON customer.CustomerID = orders.Customer_Cu
                     WHERE orders.TotalCost > 60
                     """, mydb)
```

Out[7]:

FullName	ContactNumber

The cloned repository contains a procedure called GetMaxQuantity(). Call this procedure and verify

In [13]:
```
pd.read_sql_query("""CALL GetMaxQuantity();""", mydb)
```

Out[13]:

	MAX(orders.Quantity)
0	None

Call the ManageBooking() procedure by passing the appropriate parameters. First with an available table number, then with one that has already been reserved.

In [14]:
```
pd.read_sql_query("""CALL CheckBooking("2022-01-01", 5);""", my
```

```
---------------------------------------------------------------
-----------
OperationalError                          Traceback (most recent
call last)
Input In [14], in <cell line: 1>()
----> 1 pd.read_sql_query("""CALL CheckBooking("2022-01-01",
5);""", mydb)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.p
y:399, in read_sql_query(sql, con, index_col, coerce_float, para
ms, parse_dates, chunksize, dtype)
```

```
ms, parse_dates, chunksize, dtype)
    341 """
    342 Read SQL query into a DataFrame.
    343
   (...)
    396 parameter will be converted to UTC.
    397 """
    398 pandas_sql = pandasSQL_builder(con)
--> 399 return pandas_sql.read_query(
    400     sql,
    401     index_col=index_col,
    402     params=params,
    403     coerce_float=coerce_float,
    404     parse_dates=parse_dates,
    405     chunksize=chunksize,
    406     dtype=dtype,
    407 )

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.py:2080, in SQLiteDatabase.read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize, dtype)
   2068 def read_query(
   2069     self,
   2070     sql,
   (...)
   2076     dtype: DtypeArg | None = None,
   2077 ):
   2079     args = _convert_params(sql, params)
-> 2080     cursor = self.execute(*args)
   2081     columns = [col_desc[0] for col_desc in cursor.description]
   2083     if chunksize is not None:

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.py:2018, in SQLiteDatabase.execute(self, *args, **kwargs)
   2017 def execute(self, *args, **kwargs):
-> 2018     cur = self.con.cursor()
   2019     try:
   2020         cur.execute(*args, **kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\connection_cext.py:675, in CMySQLConnection.cursor(self, buffered, raw, prepared, cursor_class, dictionary, named_tuple)
    673 self.handle_unread_result(prepared)
    674 if not self.is_connected():
--> 675     raise OperationalError("MySQL Connection not available.")
    676 if cursor_class is not None:
    677     if not issubclass(cursor_class, CMySQLCursor):

OperationalError: MySQL Connection not available.
```

Create an SQL statement that calls the AddBooking() procedure

Call the cursor .execute() method using the above parameters. Print out the result using the cursor .fetchall() method.

In [15]:
```
pd.read_sql_query("""CALL AddBooking(99, 99, 99, "2022-12-10")
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
Input In [15], in <cell line: 1>()
----> 1 pd.read_sql_query("""CALL AddBooking(99, 99, 99, "2022-12-10");""", mydb)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.py:399, in read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize, dtype)
    341 """
    342 Read SQL query into a DataFrame.
    343
   (...)
    396 parameter will be converted to UTC.
    397 """
    398 pandas_sql = pandasSQL_builder(con)
--> 399 return pandas_sql.read_query(
    400     sql,
    401     index_col=index_col,
    402     params=params,
    403     coerce_float=coerce_float,
    404     parse_dates=parse_dates,
    405     chunksize=chunksize,
    406     dtype=dtype,
    407 )

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.py:2080, in SQLiteDatabase.read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize, dtype)
   2068 def read_query(
   2069     self,
   2070     sql,
   (...)
   2076     dtype: DtypeArg | None = None,
   2077 ):
   2079     args = _convert_params(sql, params)
-> 2080     cursor = self.execute(*args)
   2081     columns = [col_desc[0] for col_desc in cursor.description]
   2083     if chunksize is not None:

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.py:2018, in SQLiteDatabase.execute(self, *args, **kwargs)
   2017 def execute(self, *args, **kwargs):
-> 2018     cur = self.con.cursor()
   2019     try:
   2020         cur.execute(*args, **kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\connection_cext.py:675, in CMySQLConnection.cursor(self, buffered, raw, prepared, cursor_class, dictionary, named_tuple)
```

```
    673 self.handle_unread_result(prepared)
    674 if not self.is_connected():
--> 675     raise OperationalError("MySQL Connection not availab
le.")
    676 if cursor_class is not None:
    677     if not issubclass(cursor_class, CMySQLCursor):

OperationalError: MySQL Connection not available.
```

In [16]:

```
mycursor.fetchall()
```

```
---------------------------------------------------------------------
-----------
InterfaceError                            Traceback (most recent
call last)
Input In [16], in <cell line: 1>()
----> 1 mycursor.fetchall()

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\
cursor_cext.py:609, in CMySQLCursor.fetchall(self)
    603 def fetchall(self) -> List[RowType]:
    604     """Return all rows of a query result set.
    605
    606     Returns:
    607         list: A list of tuples with all rows of a query
result set.
    608     """
--> 609     self._check_executed()
    610     if not self._cnx.unread_result:
    611         return []

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\
cursor_cext.py:172, in CMySQLCursor._check_executed(self)
    167 """Check if the statement has been executed.
    168
    169 Raises an error if the statement has not been executed.
    170 """
    171 if self._executed is None:
--> 172     raise InterfaceError(ERR_NO_RESULT_TO_FETCH)

InterfaceError: No result set to fetch from
```

Create a SQL statement that calls the UpdateBooking() procedure

In [17]:

```
pd.read_sql_query("""CALL UpdateBooking(99, "2022-01-10");""",
```

```
---------------------------------------------------------------------
-----------
OperationalError                          Traceback (most recent
call last)
Input In [17], in <cell line: 1>()
----> 1 pd.read_sql_query("""CALL UpdateBooking(99, "2022-01-1
0");""", mydb)
```

```
File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.py:399, in read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize, dtype)
    341 """
    342 Read SQL query into a DataFrame.
    343
   (...)
    396 parameter will be converted to UTC.
    397 """
    398 pandas_sql = pandasSQL_builder(con)
--> 399 return pandas_sql.read_query(
    400     sql,
    401     index_col=index_col,
    402     params=params,
    403     coerce_float=coerce_float,
    404     parse_dates=parse_dates,
    405     chunksize=chunksize,
    406     dtype=dtype,
    407 )

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.py:2080, in SQLiteDatabase.read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize, dtype)
   2068 def read_query(
   2069     self,
   2070     sql,
   (...)
   2076     dtype: DtypeArg | None = None,
   2077 ):
   2079     args = _convert_params(sql, params)
-> 2080     cursor = self.execute(*args)
   2081     columns = [col_desc[0] for col_desc in cursor.description]
   2083     if chunksize is not None:

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.py:2018, in SQLiteDatabase.execute(self, *args, **kwargs)
   2017 def execute(self, *args, **kwargs):
-> 2018     cur = self.con.cursor()
   2019     try:
   2020         cur.execute(*args, **kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\connection_cext.py:675, in CMySQLConnection.cursor(self, buffered, raw, prepared, cursor_class, dictionary, named_tuple)
    673 self.handle_unread_result(prepared)
    674 if not self.is_connected():
--> 675     raise OperationalError("MySQL Connection not available.")
    676 if cursor_class is not None:
    677     if not issubclass(cursor_class, CMySQLCursor):

OperationalError: MySQL Connection not available.
```

In [18]:

```
mycursor.fetchall()
```

```
---------------------------------------------------------------
-----------
InterfaceError                            Traceback (most recent
call last)
Input In [18], in <cell line: 1>()
----> 1 mycursor.fetchall()

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\
cursor_cext.py:609, in CMySQLCursor.fetchall(self)
    603 def fetchall(self) -> List[RowType]:
    604     """Return all rows of a query result set.
    605
    606     Returns:
    607         list: A list of tuples with all rows of a query
result set.
    608     """
--> 609     self._check_executed()
    610     if not self._cnx.unread_result:
    611         return []

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\
cursor_cext.py:172, in CMySQLCursor._check_executed(self)
    167 """Check if the statement has been executed.
    168
    169 Raises an error if the statement has not been executed.
    170 """
    171 if self._executed is None:
--> 172     raise InterfaceError(ERR_NO_RESULT_TO_FETCH)

InterfaceError: No result set to fetch from
```

Create a SQL statement that calls the CancelBooking() procedure

In [19]:
```
pd.read_sql_query("""CALL CancelBooking(99);""", mydb)
```

```
---------------------------------------------------------------
-----------
OperationalError                          Traceback (most recent
call last)
Input In [19], in <cell line: 1>()
----> 1 pd.read_sql_query("""CALL CancelBooking(99);""", mydb)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.p
y:399, in read_sql_query(sql, con, index_col, coerce_float, para
ms, parse_dates, chunksize, dtype)
    341 """
    342 Read SQL query into a DataFrame.
    343
   (...)
    396 parameter will be converted to UTC.
    397 """
    398 pandas_sql = pandasSQL_builder(con)
--> 399 return pandas_sql.read_query(
    400     sql,
    401     index_col=index_col,
```

```
    402     params=params,
    403     coerce_float=coerce_float,
    404     parse_dates=parse_dates,
    405     chunksize=chunksize,
    406     dtype=dtype,
    407 )

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.p
y:2080, in SQLiteDatabase.read_query(self, sql, index_col, coerc
e_float, params, parse_dates, chunksize, dtype)
   2068 def read_query(
   2069     self,
   2070     sql,
   (...)
   2076     dtype: DtypeArg | None = None,
   2077 ):
   2079     args = _convert_params(sql, params)
-> 2080     cursor = self.execute(*args)
   2081     columns = [col_desc[0] for col_desc in cursor.descri
ption]
   2083     if chunksize is not None:

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\sql.p
y:2018, in SQLiteDatabase.execute(self, *args, **kwargs)
   2017 def execute(self, *args, **kwargs):
-> 2018     cur = self.con.cursor()
   2019     try:
   2020         cur.execute(*args, **kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\
connection_cext.py:675, in CMySQLConnection.cursor(self, buffere
d, raw, prepared, cursor_class, dictionary, named_tuple)
    673 self.handle_unread_result(prepared)
    674 if not self.is_connected():
--> 675     raise OperationalError("MySQL Connection not availab
le.")
    676 if cursor_class is not None:
    677     if not issubclass(cursor_class, CMySQLCursor):

OperationalError: MySQL Connection not available.
```

In [20]:

```
mycursor.fetchall()
```

```
---------------------------------------------------------------
-----------
InterfaceError                            Traceback (most recent
call last)
Input In [20], in <cell line: 1>()
----> 1 mycursor.fetchall()

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\
cursor_cext.py:609, in CMySQLCursor.fetchall(self)
    603 def fetchall(self) -> List[RowType]:
    604     """Return all rows of a query result set.
    605
    606     Returns:
```

```
    607         list: A list of tuples with all rows of a query
result set.
    608     """
--> 609     self._check_executed()
    610     if not self._cnx.unread_result:
    611         return []

File C:\ProgramData\Anaconda3\lib\site-packages\mysql\connector\
cursor_cext.py:172, in CMySQLCursor._check_executed(self)
    167 """Check if the statement has been executed.
    168
    169 Raises an error if the statement has not been executed.
    170 """
    171 if self._executed is None:
--> 172     raise InterfaceError(ERR_NO_RESULT_TO_FETCH)

InterfaceError: No result set to fetch from
```

Python code done by Dennis Lam