Assignment2_python

October 31, 2021

DATA ENGINEERING PLATFORMS (MSCA 31012)

File: PythonMySQL

Desc: Connecting to MySQL from Jupyter Notebook

Auth: Fiona Fei

Date: 10/25/2021

ALL RIGHTS RESERVED | DO NOT DISTRIBUTE

[2]:
```
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
# Install packages
!pip3 install pymysql
!pip3 install plotly
!pip3 install cufflinks
```

```
Requirement already satisfied: pymysql in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(1.0.2)
Requirement already satisfied: plotly in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(5.3.1)
Requirement already satisfied: tenacity>=6.2.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from plotly) (8.0.1)
Requirement already satisfied: six in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from plotly) (1.16.0)
Requirement already satisfied: cufflinks in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(0.17.3)
Requirement already satisfied: numpy>=1.9.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cufflinks) (1.21.3)
```

```
Requirement already satisfied: pandas>=0.19.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cufflinks) (1.3.4)
Requirement already satisfied: plotly>=4.1.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cufflinks) (5.3.1)
Requirement already satisfied: six>=1.9.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cufflinks) (1.16.0)
Requirement already satisfied: colorlover>=0.2.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cufflinks) (0.3.0)
Requirement already satisfied: setuptools>=34.4.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cufflinks) (47.1.0)
Requirement already satisfied: ipython>=5.3.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cufflinks) (7.28.0)
Requirement already satisfied: ipywidgets>=7.0.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cufflinks) (7.6.5)
Requirement already satisfied: appnope in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (0.1.2)
Requirement already satisfied: traitlets>=4.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (5.1.0)
Requirement already satisfied: pexpect>4.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (4.8.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (3.0.20)
Requirement already satisfied: pickleshare in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (0.7.5)
Requirement already satisfied: jedi>=0.16 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (0.18.0)
Requirement already satisfied: backcall in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (0.2.0)
Requirement already satisfied: decorator in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (5.1.0)
Requirement already satisfied: matplotlib-inline in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (0.1.3)
```

```
Requirement already satisfied: pygments in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipython>=5.3.0->cufflinks) (2.10.0)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipywidgets>=7.0.0->cufflinks) (1.0.2)
Requirement already satisfied: widgetsnbextension~=3.5.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipywidgets>=7.0.0->cufflinks) (3.5.1)
Requirement already satisfied: ipython-genutils~=0.2.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipywidgets>=7.0.0->cufflinks) (0.2.0)
Requirement already satisfied: ipykernel>=4.5.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipywidgets>=7.0.0->cufflinks) (6.4.2)
Requirement already satisfied: nbformat>=4.2.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipywidgets>=7.0.0->cufflinks) (5.1.3)
Requirement already satisfied: python-dateutil>=2.7.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from pandas>=0.19.2->cufflinks) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from pandas>=0.19.2->cufflinks) (2021.3)
Requirement already satisfied: tenacity>=6.2.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from plotly>=4.1.1->cufflinks) (8.0.1)
Requirement already satisfied: debugpy<2.0,>=1.0.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (1.5.1)
Requirement already satisfied: jupyter-client<8.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (7.0.6)
Requirement already satisfied: tornado<7.0,>=4.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (6.1)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from jedi>=0.16->ipython>=5.3.0->cufflinks) (0.8.2)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks) (4.1.2)
Requirement already satisfied: jupyter-core in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks) (4.8.1)
Requirement already satisfied: ptyprocess>=0.5 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from pexpect>4.3->ipython>=5.3.0->cufflinks) (0.7.0)
```

```
Requirement already satisfied: wcwidth in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=5.3.0->cufflinks)
(0.2.5)
Requirement already satisfied: notebook>=4.4.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (6.4.5)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks)
(0.18.0)
Requirement already satisfied: attrs>=17.4.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks)
(21.2.0)
Requirement already satisfied: entrypoints in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from jupyter-client<8.0->ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (0.3)
Requirement already satisfied: pyzmq>=13 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from jupyter-client<8.0->ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks)
(22.3.0)
Requirement already satisfied: nest-asyncio>=1.5 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from jupyter-client<8.0->ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks)
(1.5.1)
Requirement already satisfied: prometheus-client in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks)
(0.11.0)
Requirement already satisfied: argon2-cffi in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks)
(21.1.0)
Requirement already satisfied: Send2Trash>=1.5.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks)
(1.8.0)
Requirement already satisfied: nbconvert in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks)
(6.2.0)
Requirement already satisfied: jinja2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks)
(3.0.2)
Requirement already satisfied: terminado>=0.8.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
```

```
(from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks)
(0.12.1)
Requirement already satisfied: cffi>=1.0.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0
->cufflinks) (1.15.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from jinja2->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cuf
flinks) (2.0.1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->
cufflinks) (0.5.4)
Requirement already satisfied: defusedxml in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->
cufflinks) (0.7.1)
Requirement already satisfied: pandocfilters>=1.4.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->
cufflinks) (1.5.0)
Requirement already satisfied: bleach in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->
cufflinks) (4.1.0)
Requirement already satisfied: mistune<2,>=0.8.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->
cufflinks) (0.8.4)
Requirement already satisfied: jupyterlab-pygments in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->
cufflinks) (0.1.2)
Requirement already satisfied: testpath in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->
cufflinks) (0.5.0)
Requirement already satisfied: pycparser in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cffi>=1.0.0->argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipyw
idgets>=7.0.0->cufflinks) (2.20)
Requirement already satisfied: webencodings in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>
=7.0.0->cufflinks) (0.5.1)
Requirement already satisfied: packaging in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
```

```
(from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>
=7.0.0->cufflinks) (21.0)
Requirement already satisfied: pyparsing>=2.0.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from packaging->bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->
ipywidgets>=7.0.0->cufflinks) (2.4.7)
```

[4]:
```
#import statements
import pymysql
import pandas as pd
```

[168]:
```
######## QUESTION 1 ######## - { 10 Points }
# a) Show the list of databases.

# Open database connection
db = pymysql.connect("localhost","root","rootroot","classicmodels" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "show databases;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")
```

[169]:
```
df = pd.DataFrame( [[ij for ij in i] for i in rows] )
```

[170]:
```
df.head()
```

[170]:
```
                    0
0        classicmodels
1   information_schema
2                mysql
3   performance_schema
4               sakila
```

[171]:
```
# b) Select sakila database.
# prepare a cursor object using cursor() method
```

```
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "USE sakila;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")
df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[171]:
```
Empty DataFrame
Columns: []
Index: []
```

[172]:
```
# c) Show all tables in the sakila database.
# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "show tables; "

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")
```

[173]:
```
df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[173]:
```
                   0
0              actor
1         actor_info
2  actors_portfolia
```

```
3  actors_portfolio
4           address
```

[174]:
```
# d) Show each of the columns along with their data types for the actor table
# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SHOW COLUMNS FROM `actor`; "

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[174]:
```
             0                  1   2    3                  4  \
0     actor_id  smallint unsigned  NO  PRI               None
1   first_name        varchar(45)  NO                    None
2    last_name        varchar(45)  NO  MUL               None
3  last_update          timestamp  NO       CURRENT_TIMESTAMP

                                             5
0                               auto_increment
1
2
3  DEFAULT_GENERATED on update CURRENT_TIMESTAMP
```

[175]:
```
# e) Show the total number of records in the actor table.
# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT count(actor_id) as 'count' FROM actor;"

try:
   # Execute the SQL command
   cursor.execute(sql)
```

```
    # Fetch all the rows in a list of lists.
    rows = cursor.fetchall()
except Exception as e:
     print (e)
     print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[175]:
```
     0
0  200
```

[176]:
```
# f) What is the first name and last name of all the actors in the actor table ?
# actor: first_name, last_name
# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select first_name,last_name from actor;"

try:
    # Execute the SQL command
    cursor.execute(sql)

    # Fetch all the rows in a list of lists.
    rows = cursor.fetchall()
except Exception as e:
     print (e)
     print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[176]:
```
          0            1
0  PENELOPE      GUINESS
1      NICK     WAHLBERG
2        ED        CHASE
3  JENNIFER        DAVIS
4    JOHNNY  LOLLOBRIGIDA
```

[177]:
```
# g) Insert your first name (in first name column) and middle initial ( in the␣
 ↪last name column ) into the actors table.
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
```

```
sql = "insert into actor(first_name,last_name) values ('FIONA','F');"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
```

[178]:
```
# h) Update your middle initial with your last name (in last name column) in␣
 ↪the actors table.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "update actor \
        set \
        last_name = 'FEI' \
        where \
        first_name = 'FIONA';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
```

[179]:
```
# i) Delete the new record from the actor table where the first name and last␣
 ↪name matches yours.
# in the actors table.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
```

```
sql = "DELETE FROM actor \
WHERE first_name = 'FIONA';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
```

```
[180]: # j) Create a table payment_type with the following specifications and␣
        ↪appropriate data types

       # Table Name : "Payment_type"
       # Primary Key: "payment_type_id"
       # Column: "Type"

       cursor = db.cursor()

       # Prepare SQL query to INSERT a record into the database.

       sql = "create table Payment_type( \
          payment_type_id INT NOT NULL AUTO_INCREMENT, \
          Type VARCHAR(100) NOT NULL, \
          PRIMARY KEY ( payment_type_id ));"

       try:
          # Execute the SQL command
          cursor.execute(sql)

          # Fetch all the rows in a list of lists.
          rows = cursor.fetchall()
       except Exception as e:
           print (e)
           print ("Error: unable to fetch data")

       df = pd.DataFrame( [[ij for ij in i] for i in rows] )
       df.head()
```

```
[180]: Empty DataFrame
       Columns: []
```

```
Index: []
```

[181]:
```
# Insert following rows in to the table:
# 1, "Credit Card" ; 2, "Cash"; 3, "Paypal" ; 4 , "Cheque"
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "INSERT INTO Payment_type(payment_type_id,Type) \
VALUES(1,'Credit Card'), \
        (2,'Cash'), \
        (3,'Paypal'), \
        (4,'Cheque');"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
```

[182]:
```
# k) Rename table payment_type to payment_types.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "RENAME TABLE Payment_type TO payment_types;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[182]: Empty DataFrame
       Columns: []
       Index: []
```

```
[183]: # l) Drop the table payment_types.
       cursor = db.cursor()

       # Prepare SQL query to INSERT a record into the database.

       sql = "DROP TABLE payment_types;"

       try:
          # Execute the SQL command
          cursor.execute(sql)

          # Fetch all the rows in a list of lists.
          rows = cursor.fetchall()
       except Exception as e:
           print (e)
           print ("Error: unable to fetch data")

       df = pd.DataFrame( [[ij for ij in i] for i in rows] )
       df.head()
```

```
[183]: Empty DataFrame
       Columns: []
       Index: []
```

```
[184]: ######## QUESTION 2 ######## - { 10 Points }
       # a) List all the movies ( title & description ) that are rated PG-13 ?

       cursor = db.cursor()

       # Prepare SQL query to INSERT a record into the database.

       sql = "select distinct title,description \
       from film \
       where rating = 'PG-13';"

       try:
          # Execute the SQL command
          cursor.execute(sql)

          # Fetch all the rows in a list of lists.
          rows = cursor.fetchall()
       except Exception as e:
           print (e)
```

```
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[184]:                   0                                                   1
       0  AIRPLANE SIERRA  A Touching Saga of a Hunter And a Butler who m…
       1    ALABAMA DEVIL  A Thoughtful Panorama of a Database Administra…
       2    ALTER VICTORY  A Thoughtful Drama of a Composer And a Feminis…
       3      ANTHEM LUKE  A Touching Panorama of a Waitress And a Woman …
       4      APOLLO TEEN  A Action-Packed Reflection of a Crocodile And …
```

```
[185]: # b) List all movies that are either PG OR PG-13 using IN operator ?
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT * \
FROM film \
WHERE rating IN ('PG-13', 'PG');"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[185]:     0                  1                                                  2  \
       0   1   ACADEMY DINOSAUR  A Epic Drama of a Feminist And a Mad Scientist…
       1   6       AGENT TRUMAN  A Intrepid Panorama of a Robot And a Boy who m…
       2   7    AIRPLANE SIERRA  A Touching Saga of a Hunter And a Butler who m…
       3   9      ALABAMA DEVIL  A Thoughtful Panorama of a Database Administra…
       4  12     ALASKA PHANTOM  A Fanciful Saga of a Hunter And a Pastry Chef …

             3  4     5  6     7    8      9     10  \
       0  2006  6  None  6  0.99   86  20.99     PG
       1  2006  1  None  3  2.99  169  17.99     PG
       2  2006  1  None  6  4.99   62  28.99  PG-13
       3  2006  1  None  3  2.99  114  21.99  PG-13
       4  2006  1  None  6  0.99  136  22.99     PG
```

```
                                    11                  12
0  Deleted Scenes,Behind the Scenes 2021-10-28 23:28:28
1                    Deleted Scenes 2006-02-15 05:03:42
2           Trailers,Deleted Scenes 2006-02-15 05:03:42
3           Trailers,Deleted Scenes 2006-02-15 05:03:42
4      Commentaries,Deleted Scenes 2006-02-15 05:03:42
```

[186]:
```
# c) Report all payments greater than and equal to 2$ and Less than equal to 7$␣
 ↪?
# Note : write 2 separate queries conditional operator and BETWEEN keyword

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select * \
FROM payment \
WHERE amount between 2 and 7;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[186]:
```
   0  1  2       3     4                    5                    6
0  1  1  1    76.0  2.99 2005-05-25 11:30:37 2006-02-15 22:12:30
1  3  1  1  1185.0  5.99 2005-06-15 00:54:12 2006-02-15 22:12:30
2  6  1  1  1725.0  4.99 2005-06-16 15:18:57 2006-02-15 22:12:30
3  7  1  1  2308.0  4.99 2005-06-18 08:41:48 2006-02-15 22:12:30
4  9  1  1  3284.0  3.99 2005-06-21 06:24:45 2006-02-15 22:12:30
```

[187]:
```
# c) Report all payments greater than and equal to 2$ and Less than equal to 7$␣
 ↪?
# Note : write 2 separate queries conditional operator and BETWEEN keyword

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
```

```
sql = "SELECT * \
FROM payment \
WHERE amount >= 2 \
AND amount <= 7;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[187]:
```
   0  1  2       3     4                    5                    6
0  1  1  1    76.0  2.99 2005-05-25 11:30:37 2006-02-15 22:12:30
1  3  1  1  1185.0  5.99 2005-06-15 00:54:12 2006-02-15 22:12:30
2  6  1  1  1725.0  4.99 2005-06-16 15:18:57 2006-02-15 22:12:30
3  7  1  1  2308.0  4.99 2005-06-18 08:41:48 2006-02-15 22:12:30
4  9  1  1  3284.0  3.99 2005-06-21 06:24:45 2006-02-15 22:12:30
```

[188]:
```
# d) List all addresses that have phone number that contain digits 589.
#A separate query for phone numbers that start with 140, and a third query
#that ends with 589
# Note : write 3 different queries

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT * \
from address \
WHERE phone like '%589%';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[188]:      0                     1     2              3    4      5             6  \
       0    4  1411 Lillydale Drive  None            QLD  576          6172235589
       1  153      782 Mosul Street        Massachusetts   94  25545  885899703621
       2  333      1860 Taguig Loop            West Java  119  59550   38158430589
       3  388  368 Hunuco Boulevard               Namibe  360  17165  106439158941
       4  492     185 Mannheim Lane            Stavropol  408  23661  589377568313

                                                       7                    8
       0  b'\x00\x00\x00\x00\x01\x01\x00\x00\x00[\r\xe44… 2014-09-25 22:30:09
       1  b'\x00\x00\x00\x00\x01\x01\x00\x00\x00\xe9\xc4… 2014-09-25 22:33:46
       2  b'\x00\x00\x00\x00\x01\x01\x00\x00\x00B\xac\xa… 2014-09-25 22:31:32
       3  b'\x00\x00\x00\x00\x01\x01\x00\x00\x00\xb5\x85… 2014-09-25 22:30:03
       4  b'\x00\x00\x00\x00\x01\x01\x00\x00\x003>\x82\x… 2014-09-25 22:32:56
```

```
[189]: # d) List all addresses that have phone number that contain digits 589.
       #A separate query for phone numbers that start with 140, and a third query
       #that ends with 589
       # Note : write 3 different queries

       cursor = db.cursor()

       # Prepare SQL query to INSERT a record into the database.

       sql = "SELECT * \
       from address \
       WHERE phone like '140%';"

       try:
          # Execute the SQL command
          cursor.execute(sql)

          # Fetch all the rows in a list of lists.
          rows = cursor.fetchall()
       except Exception as e:
           print (e)
           print ("Error: unable to fetch data")

       df = pd.DataFrame( [[ij for ij in i] for i in rows] )
       df.head()
```

```
[189]:    0                  1     2        3    4 5            6  \
       0  3  23 Workhaven Lane  None  Alberta  300    14033335568
```

```
                                                   7                    8
0  b'\x00\x00\x00\x00\x01\x01\x00\x00\x00\xcd\xc4… 2014-09-25 22:30:27
```

[190]:
```
# d) List all addresses that have phone number that contain digits 589.
#A separate query for phone numbers that start with 140, and a third query
#that ends with 589
# Note : write 3 different queries

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT * \
from address \
WHERE phone like '%589';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[190]:
```
     0                      1     2          3    4      5            6  \
0    4  1411 Lillydale Drive  None        QLD  576              6172235589
1  333      1860 Taguig Loop         West Java  119  59550  38158430589

                                                   7                    8
0  b'\x00\x00\x00\x00\x01\x01\x00\x00\x00[\r\xe44… 2014-09-25 22:30:09
1  b'\x00\x00\x00\x00\x01\x01\x00\x00\x00B\xac\xa… 2014-09-25 22:31:32
```

[191]:
```
# e) List all staff members ( first name, last name, email ) whose password is␣
 ↪NULL ?

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT first_name, last_name, email \
FROM staff \
WHERE password is NULL;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[191]:     0         1                             2
       0  Jon  Stephens  Jon.Stephens@sakilastaff.com
```

```
[192]: # f) Select all films that have title names like ZOO and rental duration
       # greater than or equal to 4

       cursor = db.cursor()

       # Prepare SQL query to INSERT a record into the database.

       sql = "select * \
       from film \
       where title like '%ZOO%' \
       AND rental_duration >= 4;"

       try:
          # Execute the SQL command
          cursor.execute(sql)

          # Fetch all the rows in a list of lists.
          rows = cursor.fetchall()
       except Exception as e:
           print (e)
           print ("Error: unable to fetch data")

       df = pd.DataFrame( [[ij for ij in i] for i in rows] )
       df.head()
```

```
[192]:     0                    1  \
       0  568     MEMENTO ZOOLANDER
       1  924  UNFORGIVEN ZOOLANDER
       2  999     ZOOLANDER FICTION
```

```
                                                     2     3  4     5  6     7  \
0  A Touching Epistle of a Squirrel And a Explore…  2006  1  None  4  4.99
1  A Taut Epistle of a Monkey And a Sumo Wrestler…  2006  1  None  7  0.99
2  A Fateful Reflection of a Waitress And a Boat …  2006  1  None  5  2.99

     8      9     10                                        11  \
0   77  11.99  NC-17                          Behind the Scenes
1  129  15.99     PG  Trailers,Commentaries,Behind the Scenes
2  101  28.99      R                  Trailers,Deleted Scenes

                    12
0 2006-02-15 05:03:42
1 2006-02-15 05:03:42
2 2006-02-15 05:03:42
```

[193]:
```python
# g) What is the cost of renting the movie ACADEMY DINOSAUR for 2 weeks ?
# Note : use of column alias and watch for rental_duration value

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select (14/rental_duration)*rental_rate AS cost \
from film \
where title = 'ACADEMY DINOSAUR';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[193]:
```
          0
0  2.310000
```

[194]:
```python
# h) List all unique districts where the customers, staff, and stores are␣
 ↪located
# Note : check for NOT NULL values
```

```
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select distinct address_id from customer \
union select distinct address_id from staff \
union select distinct address_id from store \
WHERE \
  address_id IS NOT NULL;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[194]:    0
       0  5
       1  6
       2  7
       3  8
       4  9
```

```
[195]: # i) List the top 10 newest customers across all stores based on customer_id



cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select * \
from customer \
order by customer_id desc \
limit 10;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
```

```
    rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[195]:
```
     0  1          2           3                                      4    5  6  \
0  599  2     AUSTIN     CINTRON      AUSTIN.CINTRON@sakilacustomer.org  605  1
1  598  1       WADE    DELVALLE       WADE.DELVALLE@sakilacustomer.org  604  1
2  597  1    FREDDIE      DUGGAN      FREDDIE.DUGGAN@sakilacustomer.org  603  1
3  596  1    ENRIQUE    FORSYTHE    ENRIQUE.FORSYTHE@sakilacustomer.org  602  1
4  595  1   TERRENCE   GUNDERSON  TERRENCE.GUNDERSON@sakilacustomer.org  601  1

                    7                    8
0 2006-02-14 22:04:37 2006-02-15 04:57:20
1 2006-02-14 22:04:37 2006-02-15 04:57:20
2 2006-02-14 22:04:37 2006-02-15 04:57:20
3 2006-02-14 22:04:37 2006-02-15 04:57:20
4 2006-02-14 22:04:37 2006-02-15 04:57:20
```

[196]:
```
######## QUESTION 3 ######## - { 10 Points }
# a) Show total number of movies

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select count(DISTINCT(title)) \
from film;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[196]:
```
      0
0  1000
```

[197]:
```
# b) What is the minimum payment received and max payment received across all␣
 ↪transactions ?


cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select min(amount)\
from payment;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[197]:
```
      0
0  0.00
```

[198]:
```
# b) What is the minimum payment received and max payment received across all␣
 ↪transactions ?

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select max(amount)\
from payment;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")
```

```
df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[198]:
```
       0
0  11.99
```

[199]:
```
# c) Number of customers that rented movies between Feb-2005 & May-2005
# ( based on paymentDate ).
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select count(distinct customer_id) \
from payment \
where payment_date >= '2005-02-01' and payment_date <= '2005-05-31 23:59:59';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[199]:
```
     0
0  520
```

[200]:
```
# d) List all movies where replacement_cost is greater than 15$ or␣
 ↪rental_duration is
# between 6 & 10 days
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select title \
from film \
where replacement_cost > 150 or rental_duration between 6 and 10;"

try:
   # Execute the SQL command
   cursor.execute(sql)
```

```
    # Fetch all the rows in a list of lists.
    rows = cursor.fetchall()
except Exception as e:
     print (e)
     print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[200]:
```
                   0
0  ACADEMY DINOSAUR
1  ADAPTATION HOLES
2        AFRICAN EGG
3    AIRPLANE SIERRA
4    AIRPORT POLLOCK
```

[201]:
```
# e) What is the total amount spent by customers for movies in the year 2005 ?



cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select sum(amount) \
from payment;"

try:
    # Execute the SQL command
    cursor.execute(sql)

    # Fetch all the rows in a list of lists.
    rows = cursor.fetchall()
except Exception as e:
     print (e)
     print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[201]:
```
          0
0  67416.51
```

[202]:
```
# f) What is the average replacement cost across all movies ?



cursor = db.cursor()
```

```
# Prepare SQL query to INSERT a record into the database.

sql = "select avg(replacement_cost) 'Average Replacement Cost' \
from film;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[202]:
```
           0
0  19.984000
```

[203]:
```
# g) What is the standard deviation of rental rate across all movies ?



cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select std(rental_rate) 'SD of Rental Rate' \
from film;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[203]:
```
         0
0  1.64557
```

[204]:
```
# h) What is the midrange of the rental duration for all movies

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT( MAX(rental_duration) + MIN(rental_duration) ) / 2 FROM film;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[204]:
```
        0
0  5.0000
```

[205]:
```
######## QUESTION 4 ######## - { 10 Points }
# a) Customers sorted by first Name and last name in ascending order.
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT \
   * \
FROM \
   customer \
ORDER BY \
   first_name ASC, \
   last_name ASC;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")
```

```
df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[205]:      0  1       2       3                                 4    5  6  \
       0  375  2   AARON   SELBY   AARON.SELBY@sakilacustomer.org  380  1
       1  367  1    ADAM   GOOCH    ADAM.GOOCH@sakilacustomer.org  372  1
       2  525  2  ADRIAN   CLARY  ADRIAN.CLARY@sakilacustomer.org  531  1
       3  217  2   AGNES  BISHOP  AGNES.BISHOP@sakilacustomer.org  221  1
       4  389  1    ALAN    KAHN     ALAN.KAHN@sakilacustomer.org  394  1

                           7                    8
       0 2006-02-14 22:04:37 2006-02-15 04:57:20
       1 2006-02-14 22:04:37 2006-02-15 04:57:20
       2 2006-02-14 22:04:37 2006-02-15 04:57:20
       3 2006-02-14 22:04:36 2006-02-15 04:57:20
       4 2006-02-14 22:04:37 2006-02-15 04:57:20
```

```
[206]: # b) Count of movies that are either G/NC-17/PG-13/PG/R grouped by rating.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select rating, count(rating) AS 'number' \
from film \
group by rating;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[206]:        0    1
       0     PG  194
       1      G  178
       2  NC-17  210
       3  PG-13  223
       4      R  195
```

[207]:
```
# c) Number of addresses in each district.
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select district, count(district) AS 'number' \
from address \
group by district;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[207]:
```
            0  1
0     Alberta  2
1         QLD  2
2    Nagasaki  1
3  California  9
4      Attika  1
```

[208]:
```
# d) Find the movies where rental rate is greater than 1$ and
# order result set by descending order.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select title, rental_rate \
from film \
where rental_rate > 1 \
order by rental_rate desc;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
```

```
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[208]:
```
                  0     1
0     ACE GOLDFINGER  4.99
1    AIRPLANE SIERRA  4.99
2    AIRPORT POLLOCK  4.99
3   ALADDIN CALENDAR  4.99
4        ALI FOREVER  4.99
```

[209]:
```
# e) Top 2 movies that are rated R with the highest replacement cost ?

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select title \
from film \
where rating = 'R' \
order by replacement_cost desc \
limit 2;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[209]:
```
                     0
0  CHARIOTS CONSPIRACY
1     CUPBOARD SINNERS
```

[210]:
```
# f) Find the most frequently occurring (mode) rental rate across products.

cursor = db.cursor()
```

```
# Prepare SQL query to INSERT a record into the database.

sql = "select count(distinct rental_rate) as 'occur time' \
from film \
limit 1;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[210]:
```
   0
0  3
```

[211]:
```
# g) Find the 2 longest movies with movie length greater than 50mins
# and which has commentaries as a special features.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select title \
from film \
where special_features like '%commentaries%' \
AND length > 50 \
order by length desc \
limit 2;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
```

```
df.head()
```

[211]:
```
                0
0  CONTROL ANTHEM
1       HOME PITY
```

[212]:
```
# h) List the years which has more than 2 movies released.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT release_year,COUNT(*) \
FROM film \
GROUP BY release_year \
ORDER BY COUNT(*) DESC;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[212]:
```
      0     1
0  2006  1000
```

1 Part C (Individual): Combining Data, Nested Queries, Views and Indexes, Transforming Data

[213]:
```
######## QUESTION 1 ######## - { 20 Points }
# a) List the actors (firstName, lastName) who acted in more then 25 movies.
# Note: Also show the count of movies against each actor
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT first_name, last_name,COUNT(*) \
FROM actor \
GROUP BY actor_id \
ORDER BY COUNT(*) DESC;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[213]:
```
          0             1  2
0  PENELOPE       GUINESS  1
1      NICK      WAHLBERG  1
2        ED         CHASE  1
3  JENNIFER         DAVIS  1
4    JOHNNY  LOLLOBRIGIDA  1
```

[214]:
```
# b) List the actors who have worked in the German language movies.
# Note: Please execute the below SQL before answering this question.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select distinct \
        actor.first_name, actor.last_name \
from actor \
inner join film_actor on actor.actor_id = film_actor.actor_id \
inner join film on film_actor.actor_id = film.film_id \
inner join language on film.language_id = language.language_id \
where language.name = 'German';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")
```

```
df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[214]:
```
          0        1
0  PENELOPE  GUINESS
```

[215]:
```
# c) List the actors who acted in horror movies.
# Note: Show the count of movies against each actor in the result set.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select distinct actor.first_name, actor.last_name, count(*) as 'count of␣
 ↪movies' \
from actor \
inner join film_actor on actor.actor_id = film_actor.actor_id \
inner join film on film_actor.actor_id = film.film_id \
inner join film_category on film.film_id = film_category.film_id \
inner join category on film_category.category_id = category.category_id \
where category.name = 'Horror' \
group by actor.actor_id;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[215]:
```
          0        1   2
0    ANGELA   HUDSON  34
1    AUDREY  OLIVIER  25
2   CAMERON   STREEP  24
3  JENNIFER    DAVIS  22
4       JOE    SWANK  25
```

[216]:
```
# d) List all customers who rented more than 3 horror movies.

cursor = db.cursor()
```

```
# Prepare SQL query to INSERT a record into the database.

sql = "select distinct customer.first_name, customer.last_name, count(*) as␣
 ↪'Number of Horror Movies' \
from customer \
inner join rental on customer.customer_id = rental.customer_id \
inner join inventory on rental.inventory_id = inventory.inventory_id \
inner join film on inventory.film_id = film.film_id \
inner join film_category on film.film_id = film_category.film_id \
inner join category on film_category.category_id = category.category_id \
where category.name = 'Horror' \
group by customer.customer_id \
having count(*) > 3 \
order by count(*) desc;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[216]:
```
       0        1  2
0    ANA  BRADLEY  5
1  DUANE    TUBBS  5
2   EMMA     BOYD  5
3   KARL     SEAL  5
4    KEN  PREWITT  5
```

[217]:
```
# e) List all customers who rented the movie which starred SCARLETT BENING

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select distinct customer.first_name, customer.last_name \
from customer \
inner join rental on customer.customer_id = rental.customer_id \
inner join inventory on rental.inventory_id = inventory.inventory_id \
inner join film on inventory.film_id = film.film_id \
inner join film_actor on  film.film_id = film_actor.actor_id \
inner join actor on film_actor.actor_id = actor.actor_id \
where actor.first_name = 'SCARLETT' and actor.last_name = 'BENING';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[217]:
```
        0           1
0  ASHLEY  RICHARDSON
1    JOHN  FARNSWORTH
2   LARRY    THRASHER
3   BETTY       WHITE
4  LONNIE      TIRADO
```

[218]:
```
# f) Which customers residing at postal code 62703 rented movies that were␣
 ↪Documentaries.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select distinct customer.first_name, customer.last_name \
from customer \
inner join rental on customer.customer_id = rental.customer_id \
inner join inventory on rental.inventory_id = inventory.inventory_id \
inner join film on inventory.film_id = film.film_id \
inner join film_category on film.film_id = film_category.film_id \
inner join category on film_category.category_id = category.category_id \
inner join address on customer.address_id = address.address_id \
where address.postal_code = '62703' and category.name = 'Documentary';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[218]:
```
      0        1
0  ANDY  VANHORN
```

[219]:
```
## g) Find all the addresses where the second address line is not empty (i.e.,␣
 ↪contains some
#text), and return these second addresses sorted.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select address, address2 \
from address \
where address2 is not null \
order by address2 asc;"
try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[219]:
```
                   0 1
0     1913 Hanoi Way
1   1121 Loja Avenue
2  692 Joliet Street
3   1566 Inegl Manor
4    53 Idfu Parkway
```

[220]:
```
# h) How many films involve a "Crocodile" and a "Shark" based on film␣
 ↪description ?



cursor = db.cursor()
```

```
# Prepare SQL query to INSERT a record into the database.

sql = "select count(film_id) \
from film \
where description like '%Crocodile%' or '%Shark%';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[220]:
```
    0
0  99
```

[221]:
```
# i) List the actors who played in a film involving a "Crocodile" and a␣
 ↪"Shark", along with
#the release year of the movie, sorted by the actors' last names.



cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select distinct actor.first_name, actor.last_name, film.release_year \
from actor \
inner join film_actor on actor.actor_id = film_actor.actor_id \
inner join film on film_actor.actor_id = film.film_id \
where film.description like '%Crocodile%' or '%Shark%' \
order by actor.last_name asc;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")
```

```
df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[221]:          0          1     2
       0   JESSICA     BAILEY  2006
       1  HARRISON       BALE  2006
       2  SCARLETT     BENING  2006
       3      CUBA      BIRCH  2006
       4      NICK  DEGENERES  2006
```

```
[222]: # j) Find all the film categories in which there are between 55 and 65 films.␣
        ↪Return the
       #names of categories and the number of films per category, sorted from highest␣
        ↪to lowest by
       #the number of films.

       cursor = db.cursor()

       # Prepare SQL query to INSERT a record into the database.

       sql = "select category.name, count(*) \
       from category \
       inner join film_category on category.category_id = film_category.category_id \
       inner join film on film_category.film_id = film.film_id \
       group by category.name \
       having count(*) between 55 and 65 \
       order by count(*) desc;"

       try:
          # Execute the SQL command
          cursor.execute(sql)

          # Fetch all the rows in a list of lists.
          rows = cursor.fetchall()
       except Exception as e:
           print (e)
           print ("Error: unable to fetch data")

       df = pd.DataFrame( [[ij for ij in i] for i in rows] )
       df.head()
```

```
[222]:        0   1
       0  Action  64
       1     New  63
       2   Drama  62
       3   Games  61
```

```
4  Sci-Fi  61
```

[223]:
```
# k) In which of the film categories is the average difference between the film␣
 ↪replacement
#cost and the rental rate larger than 17$?

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select category.name, avg(film.replacement_cost - film.rental_rate) dif \
from category \
inner join film_category on category.category_id = film_category.category_id \
inner join film on film_category.film_id = film.film_id \
group by category.category_id \
having dif > 17;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[223]:
```
           0          1
0     Action  18.265625
1  Animation  17.318182
2   Children  17.166667
3   Classics  18.263158
4      Drama  18.064516
```

[224]:
```
# l) Many DVD stores produce a daily list of overdue rentals so that customers␣
 ↪can be
#contacted and asked to return their overdue DVDs. To create such a list,␣
 ↪search the rental
#table for films with a return date that is NULL and where the rental date is␣
 ↪further in the
#past than the rental duration specified in the film table. If so, the film is␣
 ↪overdue and we
#should produce the name of the film along with the customer name and phone␣
 ↪number.
```

```
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select concat(customer.last_name, ', ', customer.first_name) as␣
 ↪customer, address.phone, film.title \
from rental \
inner join customer on rental.customer_id = customer.customer_id \
inner join address on customer.address_id = address.address_id \
inner join inventory on rental.inventory_id = inventory.inventory_id \
inner join film on inventory.film_id = film.film_id \
where rental.return_date is null \
and rental_date + interval film.rental_duration day < current_date() \
order by title;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.rename(columns={0: 'customer', 1: 'phone', 2: 'title'}, inplace=True);

df.head()
```

[224]:
```
          customer         phone             title
0   OLVERA, DWAYNE   62127829280  ACADEMY DINOSAUR
1    HUEY, BRANDON   99883471275    ACE GOLDFINGER
2    OWENS, CARMEN  272234298332  AFFAIR PREJUDICE
3     HANNON, SETH  864392582257        AFRICAN EGG
4      COLE, TRACY  371490777743       ALI FOREVER
```

[225]:
```
# m) Find the list of all customers and staff given a store id
# Note : use a set operator, do not remove duplicates

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select first_name, last_name \
from customer \
where store_id = 1 \
union \
select first_name, last_name \
from staff \
where store_id = 1;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[225]:

```
           0         1
0       MARY     SMITH
1   PATRICIA   JOHNSON
2      LINDA  WILLIAMS
3  ELIZABETH     BROWN
4      MARIA    MILLER
```

[226]:

```
######## QUESTION 2 ######## - { 10 Points }
# a) List actors and customers whose first name is the same as
# the first name of the actor with ID 8.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT s.first_name,s.last_name  \
FROM ( \
  SELECT customer.first_name,customer.last_name  \
  FROM customer \
  UNION ALL \
  SELECT actor.first_name,actor.last_name  \
  FROM actor \
  WHERE actor.actor_id != 8 \
) as s \
  JOIN actor i8 ON i8.first_name = s.first_name \
WHERE i8.actor_id=8;"
```

```
try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[226]:

```
         0       1
0  MATTHEW   MAHAN
1  MATTHEW   LEIGH
2  MATTHEW  CARREY
```

[227]:

```
# b) List customers and payment amounts, with payments greater
# than average the payment amount

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT \
    customer_id, amount \
FROM \
    payment \
WHERE \
    amount > (SELECT AVG(amount) \
        FROM payment);"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[227]:
```
   0     1
0  1  5.99
1  1  9.99
2  1  4.99
3  1  4.99
4  1  5.99
```

[228]:
```
# c) List customers who have rented movies at least once
# Note: use IN clause

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select distinct first_name, last_name \
from customer \
where customer_id in (select customer_id from rental);"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[228]:
```
           0         1
0       MARY     SMITH
1   PATRICIA   JOHNSON
2      LINDA  WILLIAMS
3    BARBARA     JONES
4  ELIZABETH     BROWN
```

[229]:
```
# d) Find the floor of the maximum, minimum and average payment amount


cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select floor(max(amount)) \
from payment;"
```

```
try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[229]:
```
    0
0  11
```

[230]:
```
# d) Find the floor of the maximum, minimum and average payment amount


cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select floor(min(amount)) \
from payment;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[230]:
```
   0
0  0
```

[231]:
```
# d) Find the floor of the maximum, minimum and average payment amount


cursor = db.cursor()
```

```
# Prepare SQL query to INSERT a record into the database.

sql = "select floor(avg(amount)) \
from payment;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[231]:
```
   0
0  4
```

[233]:
```
######## QUESTION 3 ######## - { 5 Points }
# a) Create a view called actors_portfolio which contains information about␣
 ↪actors and
#films ( including titles and category).

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "create view actors_portfolio as \
select actor.first_name, actor.last_name, film.title, category.name \
from actor \
inner join film_actor on actor.actor_id = film_actor.actor_id \
inner join film on film_actor.actor_id = film.film_id \
inner join film_category on film.film_id = film_category.film_id \
inner join category on film_category.category_id = category.category_id;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[233]:
```
Empty DataFrame
Columns: []
Index: []
```

[234]:
```
# b) Describe the structure of the view and query the view to get information␣
↪on the actor
# ADAM GRANT


cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "describe actors_portfolio;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[234]:
```
            0             1   2 3     4 5
0  first_name   varchar(45)  NO    None
1   last_name   varchar(45)  NO    None
2       title  varchar(128)  NO    None
3        name   varchar(25)  NO    None
```

[235]:
```
# b) Describe the structure of the view and query the view to get information␣
↪on the actor
# ADAM GRANT


cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
```

```
sql = "select distinct * \
from actors_portfolio \
where \
first_name = 'ADAM' and last_name = 'GRANT';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[235]:

```
      0      1                2       3
0  ADAM  GRANT  BILKO ANONYMOUS  Family
```

[236]:

```
# c) Insert a new movie titled Data Hero in Sci-Fi Category starring ADAM GRANT
# Note: this is feasible

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select distinct customer.first_name, customer.last_name \
from customer \
inner join rental on customer.customer_id = rental.customer_id \
inner join inventory on rental.inventory_id = inventory.inventory_id \
inner join film on inventory.film_id = film.film_id \
inner join film_category on film.film_id = film_category.film_id \
inner join category on film_category.category_id = category.category_id \
inner join address on customer.address_id = address.address_id \
where address.postal_code = '62703' and category.name = 'Documentary';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[236]:
```
      0        1
0  ANDY  VANHORN
```

[237]:
```
######## QUESTION 4 ######## - { 5 Points }
# a) Extract the street number ( characters 1 through 4 ) from customer␣
 ↪addressLine1
# Note: this is a compound query


cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql="select REGEXP_SUBSTR(address,'[0-9]+') as address from address;"


try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[237]:
```
      0
0    47
1    28
2    23
3  1411
4  1913
```

[238]:
```
# b) Find out actors whose last name starts with character A, B or C.

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
```

```
sql = "SELECT * FROM actor WHERE last_name LIKE 'A%' OR \
last_name LIKE 'B%' OR last_name LIKE 'C%';"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

```
[238]:      0          1       2                   3
       0   58  CHRISTIAN  AKROYD 2006-02-15 04:34:33
       1   92    KIRSTEN  AKROYD 2006-02-15 04:34:33
       2  182     DEBBIE  AKROYD 2006-02-15 04:34:33
       3  118       CUBA   ALLEN 2006-02-15 04:34:33
       4  145        KIM   ALLEN 2006-02-15 04:34:33
```

```
[239]: # c) Find film titles that contains exactly 10 characters

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select title \
from film \
where char_length(title) = 10;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[239]:
```
             0
0   ALONE TRIP
1   BASIC EASY
2   BUGSY SONG
3   CAUSE DATE
4   CHILL LUCK
```

[240]:
```
# d) Format a payment_date using the following format e.g "22/1/2016"

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT DATE_FORMAT(payment_date, '%d/%m/%Y') FROM payment;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[240]:
```
             0
0   25/05/2005
1   28/05/2005
2   15/06/2005
3   15/06/2005
4   15/06/2005
```

[241]:
```
# e) Find the number of days between two date values rental_date & return_date


cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT DATEDIFF(return_date,rental_date) AS days \
FROM rental;"

try:
   # Execute the SQL command
```

```
    cursor.execute(sql)

    # Fetch all the rows in a list of lists.
    rows = cursor.fetchall()
except Exception as e:
     print (e)
     print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.head()
```

[241]:
```
      0
0   2.0
1   4.0
2   8.0
3  10.0
4   9.0
```

[242]:
```
######## QUESTION 5 ######## - { 20 Points }
# Provide 5 additional queries, data visualizations and indicate the business␣
 ↪use
#cases/insights they address. Please refer to the in class exercises relating␣
 ↪to Python Jupyter
#Notebook with the SQL/Plotly code
#Note: Insights should not be a flavor of the previously addressed queries␣
 ↪within
#Assignment 2.
```

[243]:
```
# Question 1
# What are the top five popular film category?
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select category.name, count(film.film_id) as count \
from category  \
inner join film_category \
on category.category_id = film_category.category_id \
inner join film \
on film_category.film_id = film.film_id \
group by category.category_id, category.name \
order by count desc ;"

try:
    # Execute the SQL command
    cursor.execute(sql)
    # Fetch all the rows in a list of lists.
    rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.rename(columns={0: 'Category Name', 1: 'Count'}, inplace=True);

df.head()
```

[243]:
```
  Category Name  Count
0        Sports     74
1       Foreign     73
2        Family     69
3   Documentary     68
4     Animation     66
```

[244]:
```
# Visualize Data
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objects as go
import plotly.graph_objects as go


labels=['sports', 'Foreign', 'Family','Dcoumentary', 'Animation' ]
values=[74,73,69,68,66]

trace=go.Pie(labels=labels,values=values,␣
↪marker=dict(colors=['green']),hoverinfo="value")
data = [trace]
layout = go.Layout(title="Pie Chart - Top 5 Popular Film Categories")
fig = go.Figure(data = data,layout = layout)

iplot(fig)
```

[245]:
```
# Question 2
# What are the top five popular district in customer's address?
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select district, count(district) as count \
from address \
group by district \
order by count desc;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.rename(columns={0: 'Category Name', 1: 'Count'}, inplace=True);

df.head()
```

[245]:

```
     Category Name  Count
0     Buenos Aires     10
1       California      9
2     West Bengali      9
3         Shandong      9
4    Uttar Pradesh      8
```

[246]:

```
# Visualize Data
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objects as go
import plotly.graph_objects as go


labels=['Buenos Aires', 'California', 'West Bengali','Shandong', 'Uttar␣
 ↪Pradesh' ]
values=[10,9,9,9,8]

trace=go.Pie(labels=labels,values=values,␣
 ↪marker=dict(colors=['yellow']),hoverinfo="value")
data = [trace]
layout = go.Layout(title="Pie Chart - Top 5 Popular District in Customer's␣
 ↪Address")
fig = go.Figure(data = data,layout = layout)

iplot(fig)
```

[247]:

```
#Question 3 What is the top 5 popular rating in all movies?
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "select rating, count(rating) as count \
from film \
group by rating \
order by count desc;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.rename(columns={0: 'Category Name', 1: 'Count'}, inplace=True);

df.head()
```

[247]:

```
  Category Name  Count
0         PG-13    223
1         NC-17    210
2             R    195
3            PG    194
4             G    178
```

[248]:

```
# Visualize Data
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objects as go
import plotly.graph_objects as go


labels=['PG-13','NC-17','R','PG','G' ]
values=[233,210,195,194,178]

trace=go.Pie(labels=labels,values=values,␣
 ↪marker=dict(colors=['pink']),hoverinfo="value")
data = [trace]
layout = go.Layout(title="Pie Chart - Top 5 Popular Rating in ALl Movies")
fig = go.Figure(data = data,layout = layout)

iplot(fig)
```

[249]:

```
#Question 4 What is the top 5 longest trailer movies and their length?

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
```

```
sql = "select title, length \
from film \
where special_features like '%Trailers%' \
order by length desc \
limit 5;"

try:
    # Execute the SQL command
    cursor.execute(sql)

    # Fetch all the rows in a list of lists.
    rows = cursor.fetchall()
except Exception as e:
     print (e)
     print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.rename(columns={0: 'Category Name', 1: 'Count'}, inplace=True);

df.head()
```

[249]:

```
        Category Name  Count
0           HOME PITY    185
1        POND SEATTLE    185
2  SOLDIERS EVOLUTION    185
3    CRYSTAL BREAKING    184
4      KING EVOLUTION    184
```

[250]:

```
# Visualize Data
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objects as go
import plotly.graph_objects as go


labels=['HOME PITY','POND SEATTLE','SOLDIERS EVOLUTION','CRYSTAL␣
↪BREAKING','KING EVOLUTION']
values=[185,185,185,184,184]

trace=go.Pie(labels=labels,values=values,␣
↪marker=dict(colors=['cyan']),hoverinfo="value")
data = [trace]
layout = go.Layout(title="Pie Chart - Top 5 Longest Movies and Their Length")
fig = go.Figure(data = data,layout = layout)

iplot(fig)
```

```
[251]: #Question 5 What are the movies with more than 12 actors among all movies?
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT f.title AS 'Film Title', COUNT(fa.actor_id) AS count \
FROM film_actor fa \
INNER JOIN film f  \
ON fa.film_id= f.film_id \
GROUP BY f.title \
having count > 12 \
order by count desc;"

try:
   # Execute the SQL command
   cursor.execute(sql)

   # Fetch all the rows in a list of lists.
   rows = cursor.fetchall()
except Exception as e:
    print (e)
    print ("Error: unable to fetch data")

df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.rename(columns={0: 'Film Name', 1: 'Count'}, inplace=True);

df.head()
```

```
[251]:            Film Name  Count
       0   LAMBS CINCINATTI     15
       1  BOONDOCK BALLROOM     13
       2        CHITTY LOCK     13
       3         CRAZY HOME     13
       4    DRACULA CRYSTAL     13
```

```
[252]: # Visualize Data
from plotly.offline import init_notebook_mode,iplot
import plotly.graph_objects as go
import plotly.graph_objects as go


labels=['LAMBS CINCINATTI','BOONDOCK BALLROOM','CHITTY LOCK','CRAZY␣
 ↪HOME','DRACULA CRYSTAL' ]
values=[15,13,13,13,13]

trace=go.Pie(labels=labels,values=values,␣
 ↪marker=dict(colors=['grey']),hoverinfo="value")
```

```
data = [trace]
layout = go.Layout(title="Pie Chart - Movies with More than 12 Actors")
fig = go.Figure(data = data,layout = layout)

iplot(fig)
```

[ ]:

```

```

[ ]:

```

```