DSC540-T301_2237-1_Samanta_Rajib_week_9&10

August 12, 2023

[64]:
```
# Class : DSC540-T301 Data Preparation (2237-1)
# Name : Rajib Samanta
# Assignment : Week 9 & 10 Excercises
## Assignment :  Data Wrangling with Python: Activity 11, page 320
## Activity 11: Retrieving Data Correctly From Databases

import numpy as np
import pandas as pd
```

[65]:
```
# Connect to the supplied petsDB database

import sqlite3
conn = sqlite3.connect("petsdb_rajib")
```

[66]:
```
## Function to check if the connection can be establish or not
def is_opened(conn):
    try:
        conn.execute("SELECT * FROM persons LIMIT 1")
        return True
    except sqlite3.ProgrammingError as e:
        print("Connection closed {}".format(e))
        return False
print(is_opened(conn))
conn.close() ## Close the connection
```

```
True
```

[67]:
```
conn = sqlite3.connect("petsdb_rajib") # Setup new connection for Database query
c = conn.cursor()

# Execute a query to fetch all rows from the table
c.execute(f"SELECT * FROM persons LIMIT 10")

# Fetch the column names from the cursor description
column_names = [description[0] for description in c.description]

# Fetch all rows from the query result
rows = c.fetchall()
```

```
# Print column names
print("Column Names:", column_names)

# Print column values
for row in rows:
    print("Row Values:", row)

# Close the cursor and connection
#cursor.close()
```

```
Column Names: ['id', 'first_name', 'last_name', 'age', 'city', 'zip_code']
Row Values: (1, 'Erica', None, 22, 'south port', 2345678)
Row Values: (2, 'Jordi', None, 73, 'east port', 123456)
Row Values: (3, 'Chasity', None, 70, 'new port', 76856785)
Row Values: (4, 'Gregg', None, 31, 'new port', 76856785)
Row Values: (5, 'Tony', 'Lindgren', 7, 'west port', 2345678)
Row Values: (6, 'Cary', None, 73, 'new port', 76856785)
Row Values: (7, 'Gabe', 'Thompson', 54, 'new port', 9756543)
Row Values: (8, 'Francisca', None, 14, 'west port', 123456)
Row Values: (9, 'Katelyn', 'Torphy', 49, 'east port', 9756543)
Row Values: (10, 'Raleigh', None, 68, 'new port', 2345678)
```

[68]:
```
# 2. Find out the different age groups are in the persons database.

# select age and count using group by sql function
print("  Age     Person Count ")
print("------------------------------")
for ppl, age in c.execute("SELECT count(*), age FROM persons GROUP BY age"):
    print("   {}             {}".format(ppl, age))
```

```
  Age     Person Count
------------------------------
   2             5
   1             6
   1             7
   3             8
   1             9
   2             11
   3             12
   1             13
   4             14
   2             16
   2             17
   3             18
   1             19
   3             22
   2             23
```

```
3               24
2               25
1               27
1               30
3               31
1               32
1               33
2               34
3               35
3               36
1               37
2               39
1               40
1               42
2               44
2               48
1               49
1               50
2               51
2               52
2               53
2               54
1               58
1               59
1               60
1               61
2               62
1               63
2               65
2               66
1               67
3               68
1               69
1               70
4               71
1               72
5               73
3               74
```

[69]:
```
# 3. find out which age group has the highest number of people

## Use group by function with descending order and print only first row to get␣
 ↪the highest row count
for ppl, age in c.execute(    "SELECT count(*), age FROM persons GROUP BY age␣
 ↪ORDER BY count(*) DESC"):
    print("Highest number of people is {} and came from {} age group".
 ↪format(ppl, age))
```

```
    break
```

Highest number of people is 5 and came from 73 age group

[70]:
```
# 4. To find out how many people do not have a full name (the last name is␣
 ↪blank/null)
## Use group by function where last_name is null, it will return only single row
for p_count in c.execute(    "SELECT count(*) FROM persons where last_name IS␣
 ↪null"):
    print("No. of People dont have last Name is :  " + str(p_count[0])) # first␣
 ↪value
    break
```

No. of People dont have last Name is : 60

[71]:
```
## Explore pet table
# Execute a query to fetch all rows from the table
c.execute(f"SELECT * FROM pets  LIMIT 10")

# Fetch the column names from the cursor description
column_names = [description[0] for description in c.description]

# Fetch all rows from the query result
rows = c.fetchall()

# Print column names
print("Column Names:", column_names)

# Print column values
for row in rows:
    print("Row Values:", row)

# Close the cursor and connection
#cursor.close()
```

```
Column Names: ['owner_id', 'pet_name', 'pet_type', 'treatment_done']
Row Values: (57, 'mani', 1.0, 0)
Row Values: (80, 'tamari', None, 0)
Row Values: (25, 'raba', None, 0)
Row Values: (27, 'olga', None, 0)
Row Values: (60, 'raba', None, 0)
Row Values: (37, 'dara', 1.0, 0)
Row Values: (33, 'chegal', 1.0, 0)
Row Values: (16, 'dara', None, 0)
Row Values: (100, 'chegal', None, 0)
Row Values: (46, 'raba', None, 1)
```

[72]:
```
# 5 To find out how many people have more than one pet
# use inline view for grouping and having clause
c.execute("SELECT count(*) FROM (SELECT count(owner_id) FROM pets GROUP BY␣
 ↪owner_id HAVING count(owner_id) >1)")
# Fetch the result using fetchone()Pet
p_count = c.fetchone()[0]

# Print the count
print("No. of people who have more than one pet :  {}".format(p_count))
```

```
No. of people who have more than one pet :  43
```

[73]:
```
# 6. Find out how many Pets have recived Treatment
# use where caluse ,treatment_done should not be 0
c.execute("SELECT count(1) FROM pets where treatment_done >0")
# Fetch the result using fetchone()
p_count = c.fetchone()[0]

# Print the count
print("No. of Pet who have recived Treatment :  {}".format(p_count))
```

```
No. of Pet who have recived Treatment :  36
```

[74]:
```
# 7. Find out how many Pets have recived Treatment and type is known
# use where caluse ,treatment_done should not be 0 and type is not null
c.execute("SELECT count(1) FROM pets where treatment_done >0 and pet_type IS␣
 ↪NOT null")
# Fetch the result using fetchone()
p_count = c.fetchone()[0]

# Print the count
print("No. of Pet who have recived Treatment and type is known :  {}".
 ↪format(p_count))
```

```
No. of Pet who have recived Treatment and type is known :  16
```

[75]:
```
# 8. Find out how many Pets from city called 'east port'
# use where caluse persons.city='east port' where person live, need to do inner␣
 ↪join with person and pet table
c.execute("SELECT count(*) FROM pets JOIN persons ON pets.owner_id = persons.id␣
 ↪WHERE persons.city='east port'")

# Fetch the result using fetchone()
p_count = c.fetchone()[0]

# Print the count
print("No. of Pets from city called 'east port' :  {}".format(p_count))
```

```
No. of Pets from city called 'east port' :  49
```

[76]:
```
# 9. Find out how many Pets from city called 'east port' and recieved treatment
# use where caluse persons.city='east port' where person live, need to do inner␣
 ↪join with person and pet table
c.execute("SELECT count(*) FROM pets JOIN persons ON pets.owner_id = persons.id␣
 ↪WHERE persons.city='east port' and pets.treatment_done >0")

# Fetch the result using fetchone()
p_count = c.fetchone()[0]

# Print the count
print("No. of Pets from city called 'east port' and  recieved treatment:  {}".
 ↪format(p_count))
```

```
No. of Pets from city called 'east port' and  recieved treatment:  11
```

[77]:
```
conn.close() ## Close the connection
```

[ ]:
```

```