Module 5: Debugging, Databases and Project Skeletons

Assignment

Version 2.0

1. Correct the below code so that the output should be the version number.

[Example: SQLite version: 3.6.21]

```
import sqlite3
con = sqlite3.connect('test.db')
with con:

    cur = con.cursor()
    cur.execute('SELECT xxxxx')

    data = cur.fetchone()

    print "SQLite version: %s" % data
```

2. Correct the below program so that it should display the last inserted row id.

[Expected output: The last Id of the inserted row is 4]

```
import sqlite3
con = sqlite3.connect('new_db')
with con:

    cur = con.cursor()
    cur.execute("CREATE TABLE Friends(Id INTEGER PRIMARY KEY, Name TEXT);")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Tom');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Rebecca');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Jim');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Robert');")

    print "The last Id of the inserted row is %d" %
```

3. Correct the below code so that it checks weather database exists or not.

```
import os
import sqlite3

db_filename = 'todo.db'

db_is_new = not xxxxxxx(db_filename)

conn = sqlite3.connect(db_filename)

if db_is_new:
   print 'Need to create schema'
   print 'Creating database'
else:
   print 'Database exists, assume schema does, too.'

conn.close()
```

4. If Car is a table already created. What is the key word in place of “XXXX” to be used to display the column names of Cars table?

```
import sqlite3 as lite
import sys
con = lite.connect('test.db')
with con:

   cur = con.cursor()
   cur.execute("SELECT * FROM Cars")
   for colinfo in cur.XXXX:
```

```
print colinfo
```

5. Below program is for creating car's table and inserting values. But some corrections are needed. Correct the errors and execute this code.

```
import sqlite3 as lite
cars = (
   (1, 'Audi', 52642),
   (2, 'Mercedes', 57127),
   (3, 'Skoda', 9000),
   (4, 'Volvo', 29000),
   (5, 'Bentley', 350000),
   (6, 'Hummer', 41400),
   (7, 'Volkswagen', 21600)
)
con = lite.connect('test.db')
with con:
   cur = con.cursor()
   cur.execute("DROP TABLE IF EXISTS Cars")
   cur.execute("CREATE TABLE Cars(Id INT, Name TEXT, Price INT)")
   cur.XXX("INSERT INTO Cars VALUES(?, ?, ?)", cars)
```

6. If the question 5 is successfully executed then retrieve the data by correcting the below code.

```
import sqlite3 as lite
con = lite.connect('test.db')
with con:

   cur = con.cursor()
   cur.execute("SELECT * FROM Cars")
```

```
rows = cur.xxxx()
for row in rows:
    print row
```

7. Correct the below code. [Note: Question 5 should be successfully executed]

```
import sqlite3 as lite
con = lite.connect('test.db')
with con:

    con.row_factory = lite.XXX
    cur = con.cursor()
    cur.execute("SELECT * FROM Cars")
    rows = cur.fetchall()
    for row in rows:
        print "%s %s %s" % (row["Id"], row["Name"], row["Price"])
```

8. Correct the below code and it should update the values.

```
import sqlite3 as lite
import sys
uId = 1
uPrice = 62300
con = lite.connect('test.db')
with con:
    cur = con.cursor()
    cur.execute("UPDATE Cars SET Price=? WHERE Id=?", (X, Y))
    con.commit()
```

```
print "Number of rows updated: %d" % cur.rowcount
```

9. Correct the below code so that it displays the metadata info of the cars table.

```
import sqlite3 as lite
con = lite.connect('test.db')
with con:

    cur = con.cursor()

    cur.execute('XXXXX table_info(Cars)')

    data = cur.fetchall()

    for d in data:
        print d[0], d[1], d[2]
```

10. Correct the below code so that it displays all the rows from the Cars table with their column names.

```
import sqlite3 as lite
con = lite.connect('test.db')
with con:

    cur = con.cursor()
    cur.execute('SELECT * FROM Cars')

    col_names = [cn[0] for cn in cur.XXXX]
```

```
rows = cur.XXXXl()


print "%s %-10s %s" % (col_names[0], col_names[1], col_names[2])
for row in rows:
   print "%2s %-10s %s" % row
```

11. Write python program which loads “sample-storedata.csv” file data into “store” table in sqlite3.

    “sample-storedata.csv” is supplied.

12. Fetch all the rows in store table created.

13. Fetch the column names of the store table created.