Announcements

• Reminder: Lab 2 work next week
• Lab 2 specs & grading rubric: Monday
• Lab 2 setup instructions: http://tinyurl.com/hymam9a
• Format of final exam
Homework for Today

- Chapter 3 from the *Data Wrangling* book
Question 1

Which of the following data formats is not covered in the assigned chapter for today?

A. CSV
B. TSV
C. JSON
D. YAML
E. XML
Question 2

The sample data shown above is in ___ format:

A. CSV
B. TSV
C. JSON
D. SQL
E. XML
The sample data shown above is in ___ format:

A. CSV
B. TSV
C. JSON
D. SQL
E. XML
The sample data shown above is in ___ format.

A. CSV

B. TSV

C. JSON

D. SQL

E. XML
Integrating with Python

- Python support for MySQL not build it. To interact with MySQL from Python, use a library called a “connector”


- Install PyMySQL through pip: pip2 install pymysql

- Assumes existing Python 2.7 installation:
  
  ```
  python  -V
  pip2  -V
  ```
Connection Test

```python
import pymysql

try:
    connect = pymysql.connect(host="127.0.0.1",
                            user="root",
                            passwd="cs327e!",
                            db="utexas")  
    cur = connect.cursor()
    cur.execute("select count(*) from dual")
    print cur.fetchone()

except pymysql.Error as error:
    print "connect error: ", error

finally:
    connect.close()
```

(1,
[Finished in 0.9s]
What can go wrong

```python
import pymysql

try:
    connect = pymysql.connect(host="128.0.0.1",  
                               user="root",  
                               passwd="cs327e!",  
                               db="utexas")  

    cur = connect.cursor()
    cur.execute("select count(*) from dual")
    print cur.fetchone()

except pymysql.Error as error:
    print "connect error: ", error

finally:
    connect.close()
```
Concept Question 1

What caused this connection error?

A. Bad host or IP address
B. Bad username and/or password
C. Bad db name
D. Bad SQL query
E. Any of the above
import pymysql

def create_connection():
    try:
        connection = pymysql.connect(host="127.0.0.1", user="root", passwd="cs327el", db="utexas")
        return connection
    except pymysql.Error as error:
        print("connection error: ", error)

def insert():
    try:
        conn = create_connection()
        cur = conn.cursor()
        cur.execute("insert into Student (eid, first_name, last_name, age, dob) "+"values ('jpa45', 'Jon', 'Patel', 18, '1998-03-01')")
        conn.commit()
    except pymysql.Error as error:
        print("insert error: ", error)
    destroy_connection(conn)

    def destroy_connection(conn):
        conn.close()

insert()
What can go wrong

```python
import pymysql

def create_connection():
    try:
        connection = pymysql.connect(host="127.0.0.1", user="root", passwd="cs327e!", db="utexas")
        return connection
    except pymysql.Error as error:
        print("connection error: ", error)

def insert():
    try:
        conn = create_connection()
        cur = conn.cursor()
        cur.execute("insert into Student (eid, first_name, last_name, age, dob)" +
                     " values ('jpa45', 'Jon', 'Patel', 18, '03-01-1998')")
        conn.commit()
        destroy_connection(conn)
    except pymysql.Error as error:
        print("insert error: ", error)

def destroy_connection(conn):
    conn.close()

insert()
```

insert error: (1292, u"Incorrect date value: '03-01-1998' for column 'dob' at row 1")
Concept Question 2

What caused this insert to fail?

A. Duplicate record
B. Insufficient values
C. Invalid connection or cursor object
D. Internal MySQL error
E. None of the above
Multiple Inserts

def import_csv():
    insert_prefix = "insert into Student (first_name, last_name, eid, age, dob) values ("
    try:
        csvfile = open("student.csv", "rb")
        reader = csv.reader(csvfile)
        for i, row in enumerate(reader):
            if i == 0:
                continue
            insert_stmt = insert_prefix
            for j, val in enumerate(row):
                if j == 0 or j == 1 or j == 3:
                    insert_stmt += "" + val + "", "
                elif j == 2:
                    continue
                elif j == 4:
                    insert_stmt += val + ", "
                else:
                    insert_stmt += str_to_date("" + val + ", %m/%d/%Y")"
            insert_stmt += ")"
            run_insert(insert_stmt)
    except IOError as e:
        print "IO Error: " + e.strerror
What can go wrong

```python
def import_csv():
    insert_prefix = "insert into Student (first_name, last_name, eid, age, dob) values ("
    try:
        csvfile = open("student.csv", "rb")
        reader = csv.reader(csvfile)
        for i, row in enumerate(reader):
            if i == 0: continue
            insert_stmt = insert_prefix
            for j, val in enumerate(row):
                if j == 0 or j == 1 or j == 3:
                    insert_stmt += """ + val + """, "
                elif j == 2:
                    continue
                elif j == 4:
                    insert_stmt += val + ", "
                else:
                    insert_stmt += "str_to_date(" + val + ",'\%m/\%d/\%Y')"
            insert_stmt += ")"
            run_insert(insert_stmt)
```
Delete Statements

Option 1:

```
DELETE FROM T
```

e.g. `DELETE FROM Student`

Option 2:

```
DELETE FROM T WHERE c_0 = v_0
```

e.g. `DELETE FROM Student WHERE eid = 'mna34'`

Option 3:

```
DELETE FROM T WHERE (SELECT * FROM T')
```

e.g. `DELETE FROM Current_Student WHERE (SELECT * FROM Archived_Student)`

**Note:** $T <> T'$
Concept Question 3

Suppose we modify the PK in the Student table. Instead of the EID, we use an AUTO_INCREMENT column as the PK. What problem can arise from using a surrogate key?

A. Surrogate keys are less descriptive  
B. “Hidden” duplicate records  
C. Can’t reset an AUTO_INCREMENT column  
D. None of the above
Concept Question 4

Can we make this code run more efficiently? How so?

- A. Reuse the connection
- B. Commit inserts in batches
- C. Remove print statements
- D. All of the above
Inserts with FKS

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>First Name</td>
<td></td>
<td>EID</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Maria</td>
<td>Reid</td>
<td>Maria Reid</td>
<td>mna34</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Allison</td>
<td>Chantelle</td>
<td>Allison Chantelle</td>
<td>acr587</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Francis</td>
<td>Shi</td>
<td>Francis Shi</td>
<td>fos47</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>Oswald</td>
<td>Jia</td>
<td>Oswald Jia</td>
<td>jso3728</td>
<td>17</td>
</tr>
</tbody>
</table>

```python
def import_csv()
    def run_insert(insert_stmt):
        try:
            conn = create_connection()
            cur = conn.cursor()
            cur.execute(insert_stmt)
            conn.commit()
            destroy_connection(conn)
        except pymysql.Error as error:
            print "insert error: ", error
```
What can go wrong

```python
def import_csv():
    insert_prefix = "insert into Domestic_Student (eid, ssn, state) values ("

    try:
        csvfile = open("student_detail.csv", "rb")
        reader = csv.reader(csvfile)
        for i, row in enumerate(reader):
            if i == 0: continue
            insert_stmt = insert_prefix

            for j, val in enumerate(row):
                is_domestic_student = True

                if j == 0 or j == 1 or j == 2 or j == 4 or j == 5:
                    continue
                elif j == 6 and val == "":
                    is_domestic_student = False
                    break
                elif j == 3 or j == 6:
                    insert_stmt += "" + val + ","
                elif j == 7:
                    insert_stmt += "" + val + ","

            if is_domestic_student is True:
                insert_stmt += "")"
            run_insert(insert_stmt)
```

insert error: (1452, u'Cannot add or update a child row: a foreign key constraint fails | (\'utexas`\'\'domestic_student`, CONSTRAINT `domestic_student_ibfk_1` FOREIGN KEY (\'eid\') REFERENCES `student` (\'eid\')))
[Finished in 0.9s]
Final Remarks

• Avoid using surrogate keys. If you have no choice, check for duplicate records by manually inspecting the data. We will learn a more efficient way to do this when we cover the GROUP-BY clause.

• Read the API docs for PyMySQL:

• PyMySQL sample code available in our snippets repo

• Please setup your environment prior to Monday’s class (and if you get stuck, post the issue on Piazza)