Due Date: Monday, December 16, 2013 11:00 PM
Points: 40 points max
Turn In: The script and spool files turned in via the assignment drop box

General Directions

Use the a_xml database.

These tasks focus on the use the of XML techniques in MySQL. Consequently, you **must use XML techniques indicated in the task description** to solve the problems. You need to meaningfully use XPath expressions and methods to solve the tasks. If the task description says to use a particular technique, you must use that technique to get credit for the task.

All of these queries will use the ExtractValue function.

You may need to use additional techniques to complete a task but you may not simply shred the XML string into values and then do all of the work using the traditional SQL techniques. See the end of the assignment for an example of acceptable and not acceptable approaches.

The string used to store the xml data will follow the following rules. When I say an element has a value that means there is at least one character in the value. See the script for sample rows.

- The root element for each string will be <client>
- The <client> element will have an attribute for the cl_id that has a value.
- Each <client> element has a subelement for the client name that has a value.
- Each <client> element has a subelement <pets>. It is possible that the <pets> element has no subelements.
- The <pets> element may have multiple <animal> subelements.
- The <animal> element has one <an_id> subelement, one <an_type> subelement, one or more <an_name> subelements, and one <an_price> subelement. The an_id, an_type and an_price subelements always contain a value; the an_name element might be empty.
- The client id values will be unique; the animal id values will be unique

Preliminary Tasks

Create and populate a table named **a_xml.xml_animals** using the sql provided.

Tasks

Task 01: **Re-run the provided script to create and populate the table**. (15 point penalty for not following this task correctly.)

Task 02: The sql provided include data for several rows. Create at least five more rows that follow the rules given in the General Directions. Display the data values from the table after you have added your additional rows. Use extractvalue for each column, The table should not have multiple copies of the rows I provided. If it does, then reload the table from the provided script and run your script again.
These tasks should provide value for you in testing your queries for the rest of the tasks. Part of your grade for this task is based on the value of your insert in testing your queries. Include a comment for each insert as to how this insert will help.
Your script will include the inserts for the additional rows that you create.

Suggestion: Read through the tasks and try them before you decide on the new rows to insert. You might decide that certain types of data will help you test your queries more efficiently.

Sample display. The columns are the client name, animal type, animal name, and price. This is not the same data as in the supplied sql.

Client	TypeOfAnimal	AnimalName	AnimalPrice
Johnson	bird bird	ShowBoat Mr. Peanut	75 100
Nelson	cat	Ursula Buster Mittens	500
Wendell	ant caterpiller	Tiny Tim	1 1
Finckle			
Smythe	bird cat elephant	Archibald Smithers Smuthers Bossun	275 450 250

Task 03: Display the client name and the client id for clients who own a cat. Use a sql wildcard test to determine which clients have a cat.

Sample display

Client	ClientID
Anders	1234
Leeson	5678

Task 04: Display the client name and the client id for clients who own a cat. Use the XPath Count function to determine which clients have a cat.

Task 05: Display the client name and the client id for clients who own a bird that costs 250.

Task 06: Display the client name and the client id and the type and name(s) of the client's first animal. The animal type and name is concatenated as shown.

Sample display

Client	ClientID	Animal_First
Johnson	3344	turtle named Fluffy
Oliver	4433	bobcat named Whiskers Stalker

Task 07: Display the client name and the client id and the type and name of the client's first animal. If the animal has more than one name, display the animal's last name (the last name in the sequence). If the animal has only one name, then display that name. If the animal has no name or a blank name, then display the animal type and the message 'with no name'.

Client	ClientID	Animal_First
Johnson	3344	turtle named Fluffy
Oliver	4433	bobcat named Stalker
Madison	4444	bird with no name
Elise	1011	elephant named Yeller
Leeson	676	spider monkey named Mink

Task 08: Display the client name and the client id for clients who own at least two animals but no more than three animals. Do not use a count function for this.

Task 09: Display the client name and the client id for clients and the number of animals they have. Display the clients with the most animals first.

```
+---------+----------+-----------------+
| Client  | ClientID | NumberOfAnimals |
+---------+----------+-----------------+
| Albert  | 1111     | 6               |
| Boyd    | 1345     | 6               |
| Caley   | 4567     | 4               |
| WInters | 4876     | 0               |
```

Task 10: Display the client id as the first column
and use the descendant notation to display all the client data. Use \G to run the query.(see the text book for this technique)

Task 11: Display the client name as the first column
and the client id as the second column and
the first name of each of the client's birds in the third column.
For the fourth column, use the child notation to display all the animal data for birds.
If the client has no birds, the last two columns will be empty (a ZLS). Use \G to run the query.
Sample rows display

```
*************************** 1. row ***************************
    Client: Johnson
  ClientID: 8243
AnimalName: ShowBoat Mr. Peanut
AnimalData: 136 bird ShowBoat 75 137 bird Mr. Peanut 250
*************************** 2. row ***************************
    Client: Nelson
  ClientID: 3908
AnimalName:
AnimalData:
```

Task 12: Do the same display as for Task 11, but in the last two columns (animal name and animal data) display a null if the client has no birds. The null will display the way the client displays a null; do not create a literal 'NULL' in your query logic. Use \G to run the query.

Explanation for: **you must use XML techniques indicated in the task description**

The purpose of the assignment is to get you to use the XML techniques supplied by MySQL.

Suppose I asked you to use the xml_bk_1 table to get the average price of the DB books rounded to 0 digits after the decimal. And I said that you needed to use XPath techniques to locate the DB books.
This would be OK: You are using extractValue and XPath expressions with predicates and the XPath count function.
The round function is used only to improve the display, The average is not an XML technique.

```
select
Round(avg(extractValue(datax, '/book/price' )),0) as Price
from a_testbed.xml_bk_1
```

```
where extractValue(datax,'count(//categories/topic[self:text()="DB"])') > 0;
```

This would also be OK. The subquery does the required XML work and the outer query just does the average and rounding- which are not xml related techniques.

```
select round(avg(extPrice),0) as price
from
(
   select id, extractValue(datax, '/book/price' ) as extPrice
   from   a_testbed.xml_bk_1
   where  extractValue(datax,'count(//categories/topic[self:text()="DB"])')
> 0
)tbl;
```

This is *not* OK. The query does not use the XPath technique to locate the DB books. It simply gets the value of each of the attributes to be used and then uses wildcard techniques.

```
select round(avg(extPrice),0) as price
from
(
   select id
   , extractValue(datax, '/book/price' ) as extPrice
   , extractValue(datax, '//topic' ) as extTopics
   from   a_testbed.xml_bk_1
)tbl
where extTopics like '%DB%';
```