Due Date: Thursday 2013-07-25 11:00 p.m.
Points: 45 points max
Turn In: The zipped file containing the script and spool files.

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 non-blank 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.

Re-run the provided script to create and populate the table. If you do not have the original rows in the table at step 01, you will lose 10 points

Tasks

Task 01: The sql provided include data for 7 rows; my inserts use id values 1 through 7. Create at least five more rows that follow the rules given in the General Directions; your inserts must use id values 10 and higher. This should have at least 12 rows. Display the data values from the table after you have added your additional rows. Use extractvalue for each column. If the row count is wrong, then reload the xml_animals table from the provided script.
Your script will include the inserts for the additional rows that you create.
Read through the tasks and try them before you decide on the 5 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 id (the id is not part of the xml), client name, animal type, animal name, and price. This is not the same data as in the supplied sql.

```
+----+---------+--------------------+-----------------------------------+----------------+
| id | Client  | TypeOfAnimal       | AnimalName                        | AnimalPrice    |
+----+---------+--------------------+-----------------------------------+----------------+
|  1 | Johnson | bird bird          | ShowBoat Mr. Peanut               | 75 100         |
|  2 | Nelson  | cat                | Ursula Buster Mittens             | 500            |
|  3 | Wendell | ant caterpiller    | Tiny Tim                          | 1 1            |
|  4 | Finckle |                    |                                   |                |
|  5 | Smythe  | bird cat elephant  | Archibald Smithers Smuthers Bossun | 275 450 250    |
```

Task 02: Display the id and 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.

```
+----+---------+----------+
| id | Client  | ClientID |
+----+---------+----------+
|  2 | Marks   | 5784     |
|  5 | Wendell | 6895     |
|  7 | Clyde   | 1122     |
+----+---------+----------+
```

Task 03: Display the id and 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 04: Display the id and the client name and the client id for clients who own a bird that costs 250.

Task 05: Display the id and 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
+----+---------+----------+-----------------------------------+
| id | Client  | ClientID | Animal_First                      |
+----+---------+----------+-----------------------------------+
|  9 | Johnson | 3344     | turtle named Fluffy               |
| 11 | Oliver  | 4433     | bobcat named Whiskers Stalker     |
```

Task 06: Display the id and 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, then display the animal type and the message 'with no name'.

```
+----+---------+----------+--------------------------+
| id | Client  | ClientID | Animal_First             |
+----+---------+----------+--------------------------+
|  3 | Johnson | 3344     | turtle named Fluffy      |
|  6 | Oliver  | 4433     | bobcat named Stalker     |
|  7 | Madison | 4444     | bird with no name        |
| 12 | Elise   | 1011     | elephant named Yeller    |
| 14 | Leeson  | 676      | spider monkey named Mink |
```

Task 07: Display the id and 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 08: Display the id and the client name and the client id for clients and the number of animals they have. Display the clients with the most animals first.

```
+----+---------+----------+-----------------+
| id | Client  | ClientID | NumberOfAnimals |
+----+---------+----------+-----------------+
|  3 | Albert  | 1111     | 6               |
```

```
|  4 | Boyd    | 1345     | 6              |
| 12 | Caley   | 4567     | 4              |
| 17 | WInters | 4876     | 0              |
```

Task 09: 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 10: 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 (animal name and animal data) will display a null. 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.

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 11: Do the same display as for Task 10, 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%';
```

THE END