**1)** Provide the XSLT that would transform the source xml below into the destination xml. A few comments have been provided to assist you in the transformations, these comments do not need to be included in your working example.

Source XML

<customer>

<fname>Bob</fname>

<lname>Smith</lname>

<phone>702-555-1212</phone>

<dob\_y>1950</dob\_y>

<dob\_m>05</dob\_m>

<dob\_d>10</dob\_d>

<email>bob@smith.com</email>

<active>TRUE</active> <!--values: TRUE/FALSE -->

<type>WEEKLY</type>

</customer>

Destination XML

<application>

<first\_name>Bob</first\_name>

<last\_name>Smith</last\_name>

<phone1>702</phone1>

<phone2>555</phone2>

<phone3>1212</phone3>

<dob>05/10/1950</dob>

<email\_domain>smith.com</email\_domain>

<status>Active</status> <!--values: Active/Inactive -->

<repeat>W</repeat> <!--Grab first character of string, don't create XML node if the value is null or empty for <type>-->

</application>

**2)** You have a mysql table called "person", which contains several fields, the only one we care about for now is "ssn" which is a varchar(9). You must be able to search for people by the last 4 digits of their ssn, and still be able to return the full 9 digit ssn. Remember, you only have one field at your disposal, "ssn". You may not add a field that only stores the last 4. The table is expected to contain several million records, and will continue to grow over time. Performance and speed are important, it is not acceptable for it to take several seconds to return a response. Describe, in as much detail as necessary, how you would go about storing and querying the data.

**3)** You have a mysql tabe called "login", which contains a username and password. There is a form that collects these values from the user, and your job is to authenticate them against the "login" table. Provide a php function that accepts a username and password, and returns TRUE if the login is valid, FALSE if not. Don’t worry about providing the html for the form, let’s just assume someone else did that for you and all you have to deal with is the function to check the database.

**4)** Using the tables below, answer the following questions:

|  |  |  |
| --- | --- | --- |
| **emp** | | |
| **id** | **fname** | **lname** |
| 1 | andrew | m |
| 2 | mike | l |
| 3 | mike | g |
| 4 | justin | f |
| 5 | justin | s |
| **emp\_phone** | | |
| **id** | **phone** | |
| 1 | 702-123-4567 | |
| 3 | 603-123-4567 | |
| 5 | 814-123-4567 | |

a) Write a query to retrieve only the first names in emp that appear more than one time in the table.

b) Write a query to retrieve all the records from emp that have a matching record in emp\_phone.

c) Write ONE query to retrieve all the records from emp and their phone number from emp\_phone if they have one, or NULL if they don't.

d) Write a query to retrieve all records from emp that do NOT have a corresponding record in emp\_phone.

**5)** Write a command line program in PHP that simulates a single player game of bowling. Use the official scoring rules from <http://www.pba.com/Resources/Bowling101/>. Have the program bowl a full game and print out a score card. Randomly knock down 0 to 10 pins per roll. Please keep code maintainability in mind.

**6)** Encryption is a common task when working with data. Using PHP's openssl library, write a small snippet of code that will decode & decrypt the following encrypted string. Provide your php code along with the decrypted string. Hint: You will need to use the openssl\_private\_decrypt function.

*Base64 Encoded* Encrypted string: TfWJ3VMgayJaBWiIf16ctbe1N+msdWkMQyo381eRuglwLKLutKBj/Nn4DBxQlnp44FBs26RJuPdahRegovNov5jgZkNIpOFDf6qJPoGbwfBsZpnwVm7slo7peobEQi64o8hjXZAkePxvlExPNSZv6elUpzq1/D3YOr0ki9+54Lw=

Use the following private key to decrypt the above base64 encoded string.

-----BEGIN PRIVATE KEY-----

MIICdQIBADANBgkqhkiG9w0BAQEFAASCAl8wggJbAgEAAoGBAKFVDjZPSxRG3dsJ

f28pqV/4Q0SRVByz8aX+fO4G6iPsl7p9VYkIg+irhmMM8KJ5p7f4daULkMuogx5d

wOISZD/eTj1juULlLXDrOMRzoc9Yrrxc2L3osgPMHtoKDvGTk5/N2z/iSxXhHquz

IlSgqsDP2EFNBANJaj21GTikkXQlAgMBAAECgYBv7Q1mGk7RK3UhUA6L9ZBfV2J7

wINlQmXErrDHhh6Me8isBeYAotq44b7jGwgbAGGeXY5oyYRT9n245Hw7m1zWZIwB

YXIXQT0gWciISQDZKzUdcEikhhXiRG21ITHvovetQvokwZMVWpDsRoYgrk7Shl2m

aT04TxfGWUDKzpv5AQJBAMxJ/y8V2xBznqNm5Ss57Wi0R6j40zTDCa/emwE3ejY7

jn67zDFvYMh0fTsa0di4DUXNzzT/Wc5NW8neuvYQTMECQQDKK3FQ/ZoAijlNMSIu

UEGWWeNzlHyVe1BsauTQCR3YthEaSj1AZXlNiIBCj2xej1jjRuVBEmt3HV6vGwzV

cixlAkAcdN5IB8pZE1Hwvv+DMvGAGUS2I9r/yX9K8T40QC8U6NzjiHNcG4Cmy5s6

JXU/s/udUprfbgZrd1km2JDAf+rBAkAx98bMI8IafA9pmsk99SwgwxrKiFq6f34D

LfBb0sUDuQxFGTBGaE4w8Znx2Y0JWhi4I9+p06moCSRL1z22y79ZAkA0xMz7B9uH

f7k/ikRPTy4P94zMt8CscRAQ/wWVIHUnNOPNyzO1XDq22kp2w0Uo1L/UNmBlJIsn

diazJ02zGoL0

-----END PRIVATE KEY-----