

CPSC 304 Project Cover Page

Milestone #: 4

Date: Aug 9, 2022

Group Number: 14

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Patrick Conner	13781075	w3y6	pconner@student.ubc.ca
James Reinhardt	96645619	i5u3b	james.reinhardt222@gmail.com
Akriti Sharma	79884136	r0s7c	9akriti@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Repository Link:

https://github.students.cs.ubc.ca/CPSC304-2022S-T2/project_i5u3b_r0s7c_w3y6

1. Please see the first page.
2. Please see separate SQL script.
3. A PDF file containing:
 - a. A short description of the final project, and what it accomplished.

The final project is a web application built using PHP. Its GUI enables users to populate and query a database containing information on the software development job application process. This includes creating, editing, deleting, and querying Users, as well as querying various Applications, Questions, and Companies.

- b. A description of how your final schema differed from the schema you turned in.

- i. If the final schema differed, explain why. Note that turning in a final schema that's different from what you planned is fine, we just want to know what changed and why.

Only one change was made to the schema, that the Feedback2 table was not implemented. This is because the Feedback2 table only existed for normalization purposes and was unnecessary for the implementation.

Additionally, due to time constraints we were not able to fully actualize the GUI as originally intended. Instead, our focus was on actualizing the required SQL queries. Fully realizing the GUI is something that could be accomplished in our personal time, as much of it would be repetition of existing code.

c. List of Queries to Implement:

Query & Requirements:	Description, SQL, and GUI input/output:																					
<p>INSERT</p> <p>Provide an interface for the user to specify some input for the insert operation.</p>	<p>Add a new user - provide an interface to insert all info required to generate a tuple for the User table.</p> <pre>executeBoundSQL("insert into users values (:bind1, :bind2, :bind3, :bind4, :bind5, :bind6)", \$alltuples);</pre> <div><div><p>Insert a New User</p><p>ID: <input type="text" value="1234"/></p><p>Name: <input type="text" value="Insert Test"/></p><p>Age: <input type="text" value="24"/></p><p>Experience: <input type="text" value="0"/></p><p>Education: <input type="text" value="UBC"/></p><p>About: <input type="text" value="This is a test."/></p><p><input type="button" value="Insert"/></p></div><div><p>Results</p><p>Selected User Attributes:</p><table><thead><tr><th>userID</th><th>username</th><th>about</th></tr></thead><tbody><tr><td>1</td><td>Patrick</td><td></td></tr><tr><td>2</td><td>James</td><td></td></tr><tr><td>3</td><td>Akriti</td><td></td></tr><tr><td>4</td><td>Erika</td><td>Hello</td></tr><tr><td>5</td><td>Fred</td><td>I am Fred</td></tr><tr><td>1234</td><td>Insert Test</td><td>This is a test.</td></tr></tbody></table></div></div>	userID	username	about	1	Patrick		2	James		3	Akriti		4	Erika	Hello	5	Fred	I am Fred	1234	Insert Test	This is a test.
userID	username	about																				
1	Patrick																					
2	James																					
3	Akriti																					
4	Erika	Hello																				
5	Fred	I am Fred																				
1234	Insert Test	This is a test.																				
<p>DELETE</p> <p>Implement a cascade-on-delete situation (or an alternative that was agreed to by the TA if the DB system doesn't provide this). Provide an interface for the user to specify some input for the deletion operation.</p>	<p>Provide an interface option to delete a user.</p> <pre>executeBoundSQL("Delete from users where userID = :bind1", \$alltuples);</pre> <div><div><p>Delete a user from the UserTable</p><p>UserID: <input type="text" value="1234"/></p><p><input type="button" value="Submit"/></p></div><div><p>Results</p><p>These are the registered users.:</p><table><thead><tr><th>ID</th><th>Name</th><th>About</th></tr></thead><tbody><tr><td>1</td><td>Patrick</td><td></td></tr><tr><td>2</td><td>James</td><td></td></tr><tr><td>3</td><td>Akriti</td><td></td></tr><tr><td>4</td><td>Erika</td><td>Hello</td></tr><tr><td>5</td><td>Fred</td><td>I am Fred</td></tr></tbody></table></div></div>	ID	Name	About	1	Patrick		2	James		3	Akriti		4	Erika	Hello	5	Fred	I am Fred			
ID	Name	About																				
1	Patrick																					
2	James																					
3	Akriti																					
4	Erika	Hello																				
5	Fred	I am Fred																				
<p>UPDATE</p> <p>Provide an interface for the user to specify some input for the update operation.</p>	<p>Update User attributes.</p> <pre>executePlainSQL("UPDATE Users SET username=' " . \$newName . "', age=' " . \$newAge . "', experience=' " . \$newExp . "', education=' " . \$newEdu . "', about=' " . \$newAbout . "' WHERE userID=' " . \$userID . " ");</pre>																					

	<div><div><div>Update User attributes</div><div>The values are case sensitive and if you enter in the wrong case, the update statement will not do anything.</div><div>User ID: <input type="text" value="1234"/></div><div>New Name: <input type="text" value="Update Test"/></div><div>New Age: <input type="text" value="20"/></div><div>New Experience: <input type="text" value="4"/></div><div>New Education: <input type="text" value="SFU"/></div><div>New About: <input type="text" value="Hello"/></div><div><input type="button" value="Update"/></div></div><div><div>Results</div><div>Selected User Attributes:</div><table><thead><tr><th>username</th><th>education</th><th>about</th></tr></thead><tbody><tr><td>Patrick</td><td>UBC</td><td></td></tr><tr><td>James</td><td>UBC</td><td></td></tr><tr><td>Akriti</td><td>UBC</td><td></td></tr><tr><td>Erika</td><td>UBC</td><td>Hello</td></tr><tr><td>Fred</td><td>UBC</td><td>I am Fred</td></tr><tr><td>Update Test</td><td>SFU</td><td>Hello</td></tr></tbody></table></div></div>	username	education	about	Patrick	UBC		James	UBC		Akriti	UBC		Erika	UBC	Hello	Fred	UBC	I am Fred	Update Test	SFU	Hello
username	education	about																				
Patrick	UBC																					
James	UBC																					
Akriti	UBC																					
Erika	UBC	Hello																				
Fred	UBC	I am Fred																				
Update Test	SFU	Hello																				
<div>SELECTION</div> <div>Create one query of this category and provide an interface for the user to specify the values of the selection conditions to be returned. Example:</div> <div>SELECT ... FROM ... WHERE Field1 = :Var1 AND Field2 > :Var20 = Incorrect or missing</div>	<div>Select from one of two tables, select attributes to be displayed, and filter to apply.</div> <div><pre>executePlainSQL("SELECT \$attr FROM \$tbl WHERE \$var=' ' . \$filter . ' '");</pre></div> <div><div><div>View a list of interview questions(Selection Query)</div><div>Choose Question Type: <input type="text" value="coding"/></div><div>Select Attributes for viewing: <input type="text" value="Question Title"/></div><div>Choose a filter: <input type="text" value="easy (only select for coding)"/> <input type="button" value="Submit"/></div></div><div><div>Results</div><div>Retrieved data from 'coding':</div><div>Selected Attribute: 'qType'</div><div>sorting</div><div>linked list</div><div>dynamic programming</div></div></div>																					
<div>PROJECTION</div> <div>Create one query of this category, with 3-5 attributes in the projection condition, but not SELECT *. If you wish, you can have the user select the attributes from a drop- down list, but we will accept a hard-coded SELECT statement for the table(s) in question. This can be combined with another step if there are 3-5 attributes in the projection (but not SELECT *).</div>	<div>Select 3 User attributes to view.</div> <div><pre>executePlainSQL("SELECT \$first, \$sec, \$third FROM Users");</pre></div> <div><div><div>View a selection of User attributes (Projection Query)</div><div>Select User attributes: <input type="text" value="Name"/> <input type="text" value="Age"/> <input type="text" value="Experience"/> <input type="button" value="Submit"/></div></div><div><div>Results</div><div>Selected User Attributes:</div><table><thead><tr><th>username</th><th>age</th><th>experience</th></tr></thead><tbody><tr><td>Patrick</td><td>0</td><td></td></tr><tr><td>James</td><td>2</td><td></td></tr><tr><td>Akriti</td><td>1</td><td></td></tr><tr><td>Erika</td><td>21</td><td>0</td></tr><tr><td>Fred</td><td>22</td><td>0</td></tr><tr><td>Insert Test</td><td>24</td><td>0</td></tr></tbody></table></div></div>	username	age	experience	Patrick	0		James	2		Akriti	1		Erika	21	0	Fred	22	0	Insert Test	24	0
username	age	experience																				
Patrick	0																					
James	2																					
Akriti	1																					
Erika	21	0																				
Fred	22	0																				
Insert Test	24	0																				
<div>JOIN</div> <div>Create one query in this category, which joins at least 2 tables and</div>	<div>Select one of the companies below to view their coding questions:</div> <div><pre>executePlainSQL("SELECT DISTINCT coding.qTitle,company.name From coding</pre></div>																					

performs a meaningful query, and provide an interface for the user to execute this query. The user must provide at least one value to qualify in the WHERE clause (e.g. join the Customer and the Transaction table to find the names and phone numbers of all customers who have purchased a specific item).

```
INNER JOIN interviewcoding ON
coding.codingQID =
interviewcoding.codingQID
INNER JOIN interview ON
interviewcoding.intID = interview.intID
INNER JOIN applications ON interview.appID
= applications.appID
INNER JOIN position ON applications.posID =
position.posID
INNER JOIN company ON position.comID =
company.comID
Where company.name = " . $company . "');
```

View a list of coding questions based on Company (Join Query)

Choose a company:

Results

Retrieved Coding Questions asked at selected company:

Question Title	Company
rotting oranges	Google
Maximum Subarray	Google
reverse a linked list	Google
find the lowest common ancestor	Google

AGGREGATION with GROUP BY

Create one query that requires the use of aggregation (min, max, average, or count are all fine), and provide an interface (e.g., HTML button/dropdown, etc.) for the user to execute this query.

View average salary for each Company.

```
executePlainSQL("SELECT avg(salary)
FROM Position p, Company c
WHERE c.comID = p.comID and c.name = " .
$company . " '
GROUP BY c.name");
```

Results

View average salary for each Company (Aggregation with Group By)

Choose a company:

Average Salary:
Google
75000

AGGREGATION with HAVING

Create one meaningful query that requires the use of a HAVING clause, and provide an interface (e.g., HTML button/dropdown,

View number of applications to each company's positions.

```
executePlainSQL("SELECT c.name, p.title,
count(a.appID)
from Position p, Company c, Applications a
where a.posID = p.posID
```

<p>etc.) for the user to execute this query.</p>	<pre>and p.comID = c.comID GROUP BY c.name, p.title HAVING c.name = '\$bind1'");</pre> <p>View number of applications per company (Aggregation with Group By and Having)</p> <p>Company Name: <input type="text" value="Google"/></p> <p><input type="button" value="Submit"/></p> <p>Results</p> <p>Number of applications for given company is:</p> <table> <thead> <tr> <th>Age</th> <th>Experience</th> </tr> </thead> <tbody> <tr> <td>Google Junior Software Developer 2</td> <td></td> </tr> <tr> <td>Google Software Developer</td> <td>3</td> </tr> </tbody> </table>	Age	Experience	Google Junior Software Developer 2		Google Software Developer	3
Age	Experience						
Google Junior Software Developer 2							
Google Software Developer	3						
<p>NESTED AGGREGATION with GROUP BY</p> <p>Create one query that finds some aggregated value for each group (e.g., use a nested subquery, such as finding the average number of items purchased per customer, subject to some constraint). Some examples for the Sailors table are given in the project specs. Note the difference between this query and the above Aggregation Query. You must use separate distinct queries for this criterion and the Aggregation Query (i.e., do not double dip).</p> <p>It is fine to use a view to get the desired behaviour.</p>	<p>View average experience for each age with above average experience.</p> <pre>executePlainSQL("SELECT age, avg(experience) FROM Users GROUP BY age Having avg(experience) > (select avg(experience) From Users)");</pre> <p>View average experience for each age with above average experience (Nested Aggregation with Group By)</p> <p><input type="button" value="Submit"/></p> <p>Results</p> <p>Average experience for each age (with above average experience):</p> <table> <thead> <tr> <th>Age</th> <th>Experience</th> </tr> </thead> <tbody> <tr> <td>20</td> <td>4</td> </tr> </tbody> </table>	Age	Experience	20	4		
Age	Experience						
20	4						
<p>DIVISION</p> <p>Create one query of this category and provide an interface (i.e., HTML button, etc.) for the user to execute this query (e.g., find all the customers who bought all the items).</p>	<p>Display users that applied to every position.</p> <pre>executePlainSQL("SELECT u.username from Users u WHERE NOT EXISTS (select p.posID from Position p WHERE NOT EXISTS</pre>						

	<pre>(SELECT * from Applications a where a.userID = u.userID and p.posID = a.posID) ") ;</pre>						
	<div>Display the Users who applied to every job (Division)</div> <div>Submit</div>	<div>Results</div> <div>Users who applied to every job</div> <table><thead><tr><th>ID</th><th>Name</th><th>About</th></tr></thead><tbody><tr><td>Patrick</td><td></td><td></td></tr></tbody></table>	ID	Name	About	Patrick	
ID	Name	About					
Patrick							

- d. Screenshots of the sample output of the queries using the GUI:
Please see the query table above.
4. Please see separate README.txt file.