**BSA\_Database Information on All Stored Procedures and Functions**

***Please NOTE***

*An ERD can be found at the end of this doc, which will provide a visual aid to clearly show where and what information is being stored as well as how the tables are connected.*

*Procedure are marked with GREEN*

*Functions are marked with YELLOW*

-------------------------------------------------------------------------------------------------------------------------------

*GetCustomersAll*

**How to Call:**

call BSA\_Database.GetCustomersAll ();

**What does it do?**

This procedure returns all the information that is stored in the customer table. It takes no parameters.

*GetCustomersByID*

**How to Call:**

call BSA\_Database.GetCustomersByID (*customer*\_*ID*);

**What does it do?**

This procedure looks up a customer by their unique customer\_ID and returns back all of their information (in the customer table) such as email and company\_name. It takes an INT as a parameter.

*GetCustomersByName*

**How to Call:**

call BSA\_Database.GetCustomersByName (*'company*\_*name'*);

**What does it do?**

This procedure works in a similar way to *GetCustomersByID,* however information is looked up and found via a customers company name, which is provided during the registration phase of this web application. Note this procedure takes one parameter a VARCHAR(100), a string that can be no greater than 100 characters.

*GetCustomersByEmail*

**How to Call:**

call BSA\_Database.GetCustomersByEmail (*'email'*);

**What does it do?**

Information returned is the same as *GetCustomersByName* and *GetCustomersByID,* only difference being information is found via a customer email which is provided at registration. It takes only one parameter a VARCHAR(100), a string that can be no greater than 100 characters.

*AddCompany*

**How to Call:**

call BSA\_Database.AddCompany (*'email’, ‘password’, ‘company*\_*name', ‘first\_name’ , ‘last\_name’*);

**What does it do?**

This procedure should only be called when customers have successfully filled of the required registration info and will input them into the database. It takes three parameters email, password, and company\_name, first\_name, last\_name. **\*NOTE\* Order is important and customer\_ID.**

**\*\*\*\*\*\*\*\* END OF customer TABLE PROCEDURES**

*GetSurveyByID*

**How to Call:**

call BSA\_Database.GetSurveyByID (*survey\_ID*);

**What does it do?**

This procedure will provide all information stored in the surveys created table about a registered customers existing survey via a unique survey\_ID number. It takes a single parameter of an INT.

*AddSurvey*

**How to Call:**

call BSA\_Database.AddSurvey (*customer\_ID, ‘survey*\_*name'*);

**What does it do?**

This procedure will add a survey given a customer\_ID, and survey\_name, it does not worry if a survey has questions, an empty survey can still be created and is valid. This procedure takes an INT customer\_ID, and VARCHAR(100) survey\_name. **\*NOTE\* Order is important.**

**\*\*\*\*\*\*\*\* END OF surveys\_created TABLE PROCEDURES**

*GetSurveyQuestions*

**How to Call:**

call BSA\_Database.GetSurveyQuestions (*survey\_ID*);

**What does it do?**

This procedure will give the caller all of the questions in a specific survey which is found via a unique survey\_ID. It takes only one parameter an INT.

*GetQuestionNumber*

**How to Call:**

call BSA\_Database.GetQuestionNumber (*question\_ID*);

**What does it do?**

This procedure will retrieve the order in which the question is supposed to appear via a question\_ID, for example if this procedure were to retrieve a 1 then the question is meant to be the first question. It takes a single parameter of an INT

*AddQuestions*

**How to Call:**

call BSA\_Database.AddQuestions (survey\_ID, question\_order, ‘question\_string’);

**What does it do?**

This procedure will input newly created question for a particular survey into the database. It takes three parameters; INT survey\_ID, INT question\_order, VARCHAR(250) question\_string . **\*NOTE\* Order is important.**

**\*\*\*\*\*\*\*\* END OF question TABLE PROCEDURES**

*GetAnswersByQuestionID*

**How to Call:**

call BSA\_Database.GetAnswersByQuestionID (*question\_ID*);

**What does it do?**

This procedure will get all answers that pertain to a specific and unique question via the question\_ID. It takes only one parameter; INT question\_ID.

*AddAnswer*

**How to Call:**

call BSA\_Database.AddAnswer(*question\_ID*, answer\_order, 'answer\_string');

**What does it do?**

This procedure will create a new answer choice for a specific and unique question. It takes three parameters; INT *question\_ID*, INT answer\_order, VARCHAR(100) 'answer\_string'). **\*NOTE\* Order is important.**

**\*\*\*\*\*\*\*\* END OF answers TABLE PROCEDURES**

*GetResultsFromTaker*

**How to Call:**

call BSA\_Database.GetResultsFromTaker (taker\_ID);

**What does it do?**

This procedure will retrieve all values in the survey\_results table given a taker\_ID which is an INT.

*AddResult*

**How to Call:**

call BSA\_Database.AddAnswer (*answer\_ID, survey\_ID, question\_ID*);

**What does it do?**

This procedure will create a new result entry in the survey results table all of the parameters are of type INT. **\*NOTE\* Order is important.**

**\*\*\*\*\*\*\*\* END OF survey\_results TABLE PROCEDURES**

**\*\*\*\*\*\*\*\* END PROCEDURES**

*ValidLogin*

**How to Call:**

select BSA\_Database.ValidLogin (‘*emai’l*);

**What does it do?**

This function will check if an email address exists in our database currently only takes in one parameter VARCHAR(100) email, this is subject to change though.

**What does it return?**

It returns a bool, if email is found will return true(1) if not false(0) **\*NOTE\* Full email address is required, ex. “test@test.com”.**

*FindQuestionID*

**How to Call:**

select BSA\_Database.ValidLogin (*answer\_ID*);

**What does it do?**

Finds the corresponding question\_ID given an INT answer\_ID, in other word finds out what question an answer belongs to.

**What does it return?**

It returns a INT, which is the question\_ID an answer belongs to.

*TotalQuestions*

**How to Call:**

select BSA\_Database.TotalQuestions (*survey\_ID*);

**What does it do?**

Find the total number of questions in a survey via the survey\_ID field.

**What does it return?**

It returns an INT, which is the total number of questions in a survey.

*IsSurveyFinished*

**How to Call:**

select select BSA\_Database.IsSurveyFinished (*question\_ID, survey\_ID);*

**What does it do?**

Determines if the survey taker has answered the last question via the parameters question\_ID and survey\_ID both INTs.

**What does it return?**

It returns a bool, if done will return true(1) if not false(0)

**\*\*\*\*\*\*\*\* Remarks**

* This doc is up to date as of July 8th, 2020.
* All procedures have been tested in the MySQL workbench to ensure that there are no syntax errors and data is gotten or added, from or to, the right tables and columns. Further testing will be provided through a shell script which will connect to the MySQL shell.
* Please note that the ‘Date\_Time’ or ‘Date\_Created’ fields in the ERD is automatically update with all of the Add/input procedures.
* None of the following procedure descriptions; *AddCompany, AddSurvey, AddQuestions, and Add Answer* mentioned that 1) customer \_ID is auto incremented and created only when the *AddCompany procedure is called, 2) survey\_ID is also* auto incremented and created only when the *AddSurvey procedure is called, 3) similarly the question\_ID is*  auto incremented and created only when the *AddQuestions procedure is called, 4) answer\_ID* is also auto incremented and created only when the *AddAnswer procedure is called.*
* *Storage capacity of VARCHAR’s found in the ERD do not reflect the actual storage capacity in the database as of now.*
* Lastly note that the password field found in the customer table is subject to be changed depending on Auth0 (password less sign in) implementation; implementation of this will be time boxed to mitigate risk, otherwise will remain the same.

**ERD Of CEN4020 Survey App**

A close up of text on a white background

Description automatically generated