**HEALTH SURVEY SYSTEM:**

Health Researchers and Patients Survey Tool

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 17-February-2019 | 1.0 | Initial Design | Dhanya Jose |

# Table of Contents

* **Table of Contents 3**
* **Introduction 4**
* **System Design 5**
* **Data Models 6**
* **Entity Relation Diagrams 10**
* **RESPONDENT USER Implementation 11**
* **Sequence Flow Diagram RESPONDENT 12**
* **ADMIN USER Implementation 13**
* **Sequence Flow Diagram ADMIN USER 14**
* **Use Cases 15**

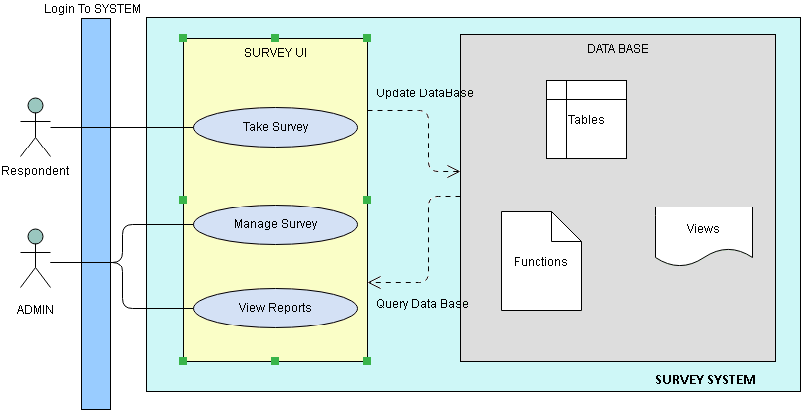
# Introduction

This document includes the design and implementation of health survey system. The health survey system will be a survey tool for health researchers and patients. There will be two types user: respondents who answer questions, and administrators who define the questions and when they should appear to respondents, and who view reports on the respondent’s answers. It contains basic design of data models (Tables, Views, Constraints and Functions), Entity Relation Diagrams, Sequence flow Diagrams and also descriptions and examples of how each use case specified above can be achieved.

Pre Requisite: PostgreSQL.

# System Design

The health survey system has 2 types of user Respondent and Admin. System provides UI to login to System with a username and password and perform operations such as Take Survey, Manage Survey and View Report according to logged in user type by interacting with the Database for respective tables, views and functions and responds with desired result.



# DATA MODELS

Survey system mainly consist of 7 tables listed as below

1. ref\_survey\_questions
2. ref\_survey\_result
3. ref\_hie\_access\_level
4. respondent\_master
5. admin\_master
6. login\_details
7. survey\_result
8. **ref\_survey\_questions TABLE**

The ref\_survey\_questions table contains all questions present in the survey and the details associated with question such as id and priority details. Table structure is give as below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL NO** | **COLUMN NAME** | **TYPE** | **CONSTRAINTS** | **DEFAULT** | **REMARKS** |
| 1 | q\_id | integer | serial, PRIMARY KEY |  | system generated serial primary key |
| 2 | q\_description | VARCHAR (200) | NOT NULL |  | description of question |
| 3 | q\_priority | BOOLEAN | NOT NULL | FALSE | Priority as set by admin user. By default setting to false |

1. **ref\_survey\_result TABLE**

The ref\_survey\_result table contains the pre-defined answer options in the survey and the details associated with answers such as id and description and score for each result. Table structure is give as below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL NO** | **COLUMN NAME** | **TYPE** | **CONSTRAINTS** | **DEFAULT** | **REMARKS** |
| 1 | result\_id | INTEGER | serial, PRIMARY KEY |  | system generated serial primary key |
| 2 | result\_description | VARCHAR (25) | NOT NULL |  | description of result |
| 3 | result\_score | INTEGER | NOT NULL | 0 | Pre-defined result weightage as given in GAD-7 Questionnaire. By default setting to 0 |

1. **ref\_hie\_access\_level TABLE**

The ref\_hie\_access\_level table contains hierarchical access level details which can be used to establish hierarchy among admin users and restrict them to view reports based on levels of respondent. Table contains different levels of admin users and the top access level each level can access. Table structure is give as below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL NO** | **COLUMN NAME** | **TYPE** | **CONSTRAINTS** | **DEFAULT** | **REMARKS** |
| 1 | hie\_id | INTEGER | serial, PRIMARY KEY |  | system generated serial primary key |
| 2 | hie\_level | VARCHAR (4) | UNIQUE,NOT NULL |  | hierarchy level |
| 3 | hie\_level\_desc | VARCHAR (20) | UNIQUE, NOT NULL |  | hierarchy level description |
| 4 | hie\_top\_access | INTEGER |  | 0 | Top access that corresponding level can view |

1. **respondent\_master TABLE**

The respondent\_master table contains all the details of each respondent such as name, postcode, email and etc. It is the master table for respondent and the details added at the time of signup and can be updated as well. Table structure is give as below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL NO** | **COLUMN NAME** | **TYPE** | **CONSTRAINTS** | **DEFAULT** | **REMARKS** |
| 1 | user\_id | INTEGER | serial, PRIMARY KEY |  | system generated serial primary key |
| 2 | first\_name | VARCHAR (100) | NOT NULL |  | First Name |
| 3 | last\_name | VARCHAR (100) | NOT NULL |  | Last Name |
| 4 | gender | VARCHAR (5) | NOT NULL |  | gender |
| 5 | postcode | VARCHAR (10) | NOT NULL |  | postcode |
| 6 | email | VARCHAR (350) | UNIQUE, NOT NULL |  | email |
| 7 | dob | DATE | NOT NULL |  | Date Of Birth |
| 8 | hie\_access | INTEGER |  | 0 | hierarchy access to restrict admin |

1. **admin\_master TABLE**

The admin\_master table contains all the details of each admin user such as name, email and etc. It is the master table for admin user. Table structure is give as below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL NO** | **COLUMN NAME** | **TYPE** | **CONSTRAINTS** | **DEFAULT** | **REMARKS** |
| 1 | admin\_id | INTEGER | serial, PRIMARY KEY |  | system generated serial primary key |
| 2 | first\_name | VARCHAR (100) | NOT NULL |  | First Name |
| 3 | last\_name | VARCHAR (100) | NOT NULL |  | Last Name |
| 4 | email | VARCHAR (350) | UNIQUE, NOT NULL |  | email |
| 5 | hie\_level | VARCHAR(10) | FOREIGN KEY |  | foreign key reference on ref\_hie\_access\_level table |

1. **login\_details TABLE**

The login\_details table contains username and password for all respondent and admin users. It also has a reference to respondent and admin master table for respective admin and respondent user login. Table structure is give as below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL NO** | **COLUMN NAME** | **TYPE** | **CONSTRAINTS** | **DEFAULT** | **REMARKS** |
| 1 | username | VARCHAR (350) | PRIMARY KEY |  | user name |
| 2 | password | VARCHAR (350) | NOT NULL |  | password |
| 3 | ref\_respondent\_id | INTEGER | FOREIGN KEY |  | refernces respondent\_master. Null for admin user |
| 4 | ref\_admin\_id | INTEGER | FOREIGN KEY |  | refernces admin\_master. Null for respondent user |
| 5 | last\_login | TIMESTAMP | NOT NULL |  | Time of last login |

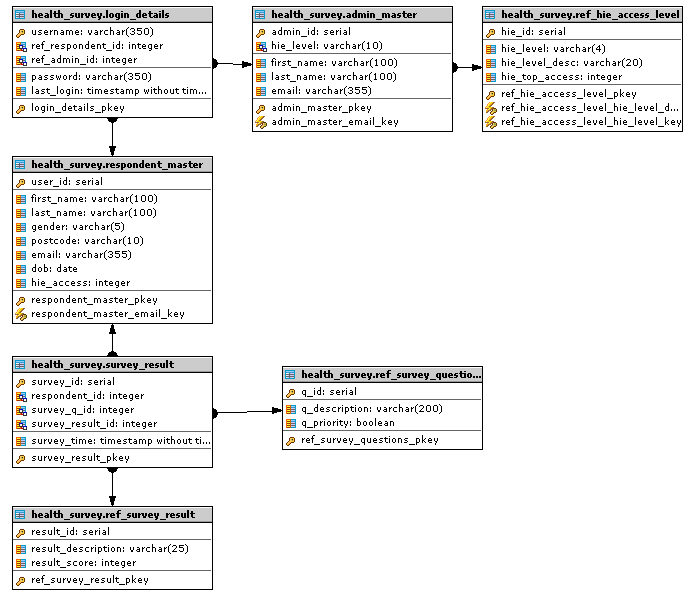
1. **survey\_result TABLE**

The survey\_result table basically contains result of each survey for the respective respondent along with time of the survey. It refers question, result and respondent master tables. Table structure is give as below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL NO** | **COLUMN NAME** | **TYPE** | **CONSTRAINTS** | **DEFAULT** | **REMARKS** |
| 1 | survey\_id | INTEGER | serial, PRIMARY KEY |  | system generated serial primary key |
| 2 | respondent\_id | INTEGER | FOREIGN KEY |  | references respondent\_master |
| 3 | survey\_q\_id | INTEGER | FOREIGN KEY |  | references ref\_survey\_questions |
| 4 | survey\_result\_id | INTEGER | FOREIGN KEY |  | references ref\_survey\_result |
| 5 | survey\_time | TIMESTAMP | NOT NULL |  | Time of survey |

# Entity Relation Diagrams

Entity relation diagram for the health survey system is given as below.



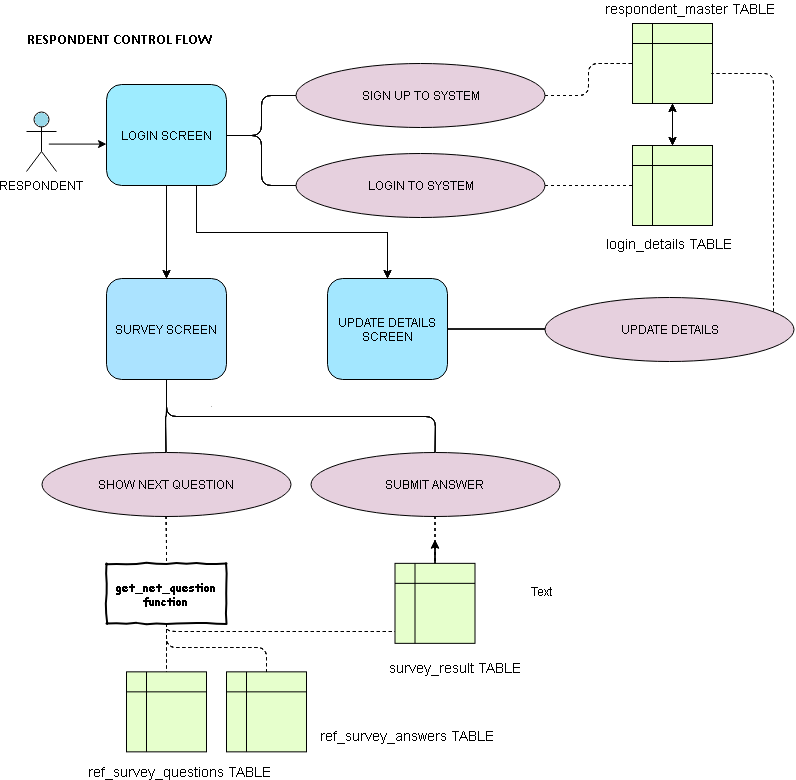
# RESPONDENT user Implementation

A respondent user can either sign up if new user or login for already existing user. For new user register with name, gender, post code, date of birth and email as all mandatory fields. On registration raise exception while missing mandatory fields else records will be added for *respondent\_master* table for corresponding respondent. A user name and password will be provided for each correspondent. User name can be taken as email or system generated username as needed and password can be set on sign up screen. And the user name and password will be saved to *login\_details* table with password as encrypted to ensure security. And upon login match username and password with *login\_details* table and raise exception on unmatched details and for expired password using *last\_login* column from *login\_details* table if needed.

After Login there will be update details and survey screen. Update details screen will fetch the details of the respondent on load from and *respondent\_master* table and allows to edit details except for emailed *survey\_result* if email id is considered as username. This screen allows changing password. Raise exception for missing mandatory fields on submit. And on successful submit update record in *respondent\_master* table for that respondent.

On Survey screen it allows to take the health survey on load and on submit next question will be queried as per required criteria, which has been implemented by function for this design which will be explained in use cases section in detail. Answer list will be fetched from *ref\_survey\_result* as it has pre-defined result set. On submit record will be added to table. At any given time respondent can logout from system

# Sequence Flow Diagram RESPONDENT



# ADMIN USER Implementation

Admin User will have only login option and no signup option to ensure security and login works similar to respondent login option which works using *login\_details* table

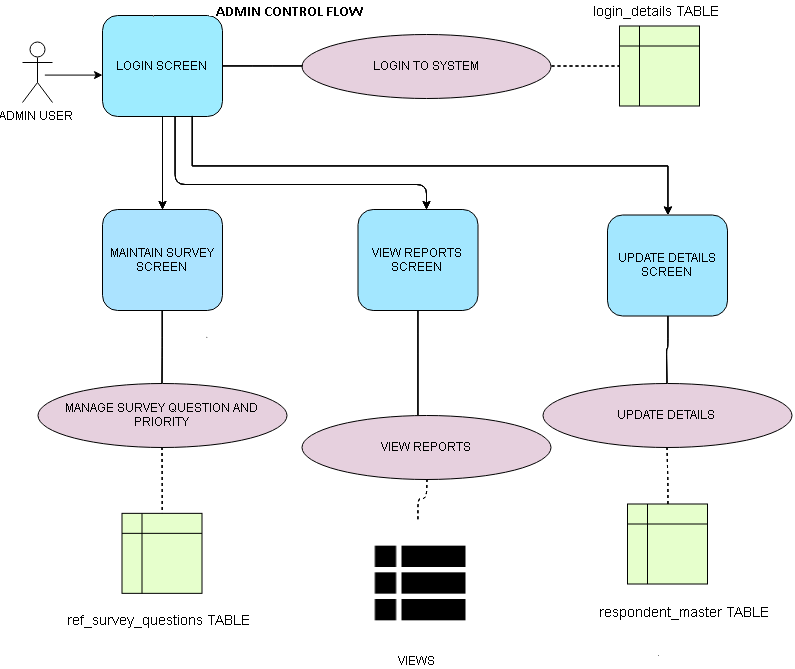
Upon login admin user have update details similar to respondent update details screen which fetches and updates *admin\_master* table details.

Admin user have admin master screen if needed for adding new admins and assigning hierarchy level so that that particular admin users can have restricted view of respondent. Admins can be categorized as Super admin, branch admin, user admin etc. These types of hierarchy can be defined *ref\_hie\_access\_level* table and corresponding id can be maintained as *hie\_level* in *admin\_master* table and for each hierarchy type a *top\_access* value is maintained in *ref\_hie\_access\_level* table and *hie\_access* value is maintained for each respondent in *respondent\_master* table. And each admin user can only view those respondent data whose *hie\_access* mentioned in *respondent\_master* tableis less than the *top\_access* of each admin user hierarchy type. This will restrict the admin access and can be used to enhance confidentiality of respondents. This is an extended implementation

Admin user have survey master screen which allows adding, deleting and modifying survey as needed and as well as set priority for each question. On load survey master will fetch all records from *ref\_survey\_questions* table and give options for adding, deleting and modifying question. On submit added, modified records will insert and update into table and deleted questions will delete from table.

Next functionality is view both demographic and statistical reports which will be explained in detail in use case section. Viewing reports can be implemented in many ways using functions or create by creating views and fetching data from it or even as jobs which run only when the request is made. In use case section detailed explanation of using functions and views are given.

# Sequence Flow Diagram ADMIN USER



# USE CASES AND EXAMPLES

Respondent use cases such as register respondent, submit survey and Admin user use cases survey maintenance covered in implementation section. Remaining use case such as retrieving next question for respondent and viewing reports can be achieved by using function or creating views depends on use case.

1. **RESPONDENT USE CASE 2**

Implementation of this use case has done through function. Provided the criteria

1. The system should first select questions (at random) that are marked as ‘priority’ and have not already been answered by the respondent at least once in the current week (beginning on Monday).
2. Once all ‘priority’ questions have been answered at least once in the current week, a ‘non-priority’ question may be randomly chosen that has not already been answered by the respondent at least once in the same week.
3. Once all available questions have been answered at least once in the current week, either a ‘priority’ or ‘non-priority’ question may be randomly chosen next, as long as it has the least number of answers from the user in that week.

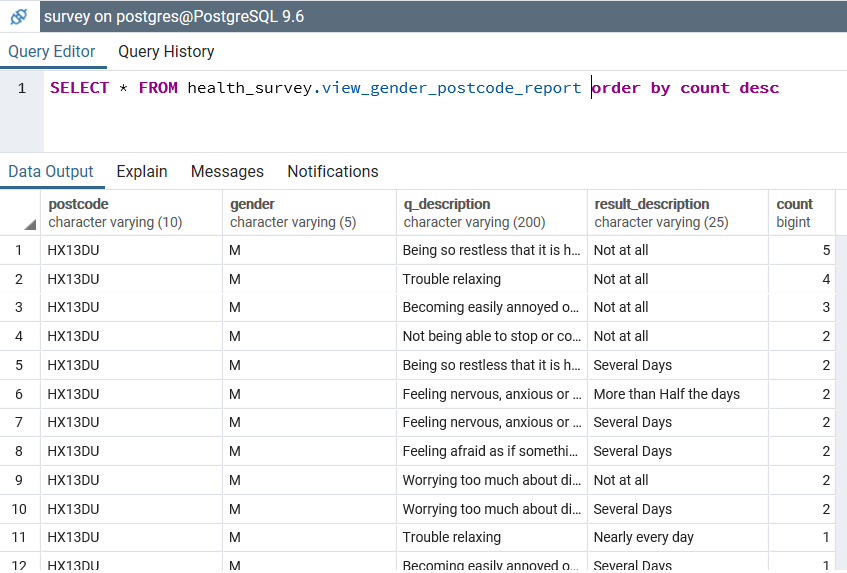
Function fetches the unanswered priority question based on priority flag maintained in survey questions table using date constraint. If it fetches no record then it fetches unanswered priority question and if that too fetches no record least answered question id is fetched by ordering them in ascending order for that week starting Monday.

Note: refer function ***health\_survey.get\_next\_question(resp\_id integer)*** for function definition.

1. **ADMIN USE CASE 3 AND 4**

Admin use cases to view reports are implemented by creating views.

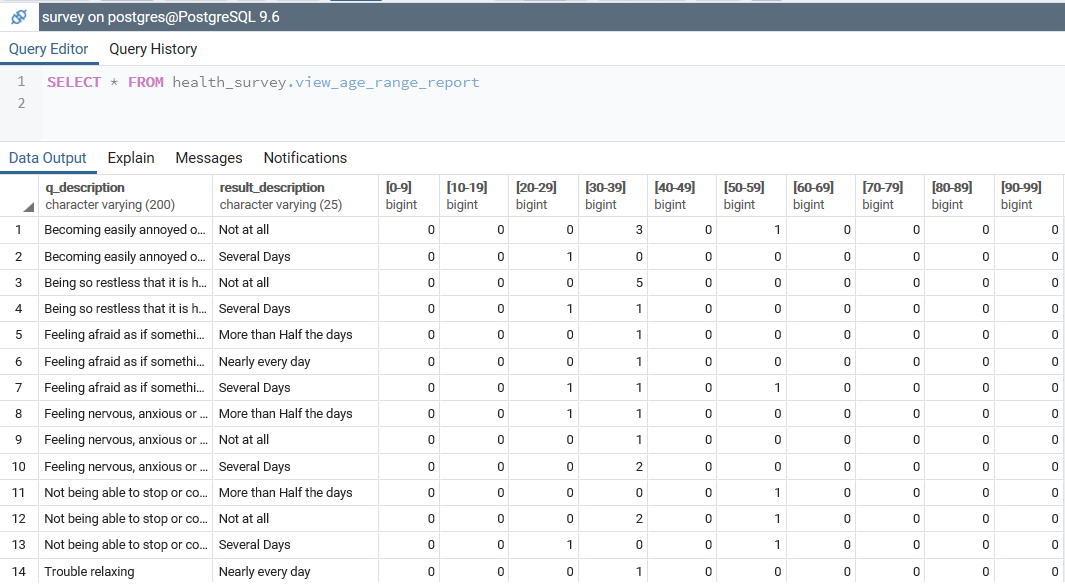
1. **View created for use case 3.a GENDER POSTCODE report**

****

Note: refer function ***health\_survey.view\_gender\_postcode\_report*** for definition.

1. **View created for Admin use case 3.b AGE RANGE report**

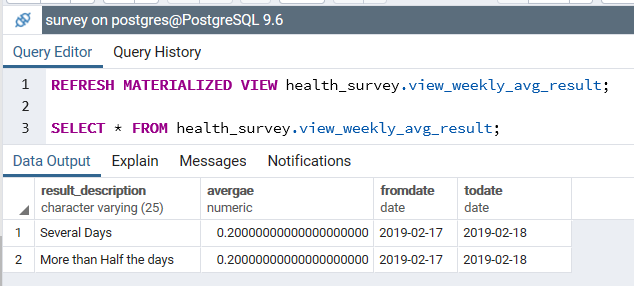
Note: refer function ***health\_survey.view\_age\_range\_report*** for definition.



1. **View created for Admin use case 4.a WEEKLY AVERAGE report**

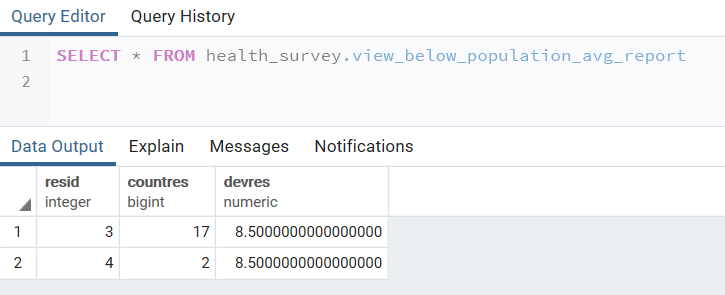
Weekly average report have created using materialized view and data can be populated only when we query for the same. Which ensures fast recovery of data.

Note: refer function ***health\_survey.view\_weekly\_avg\_result*** report for definition.

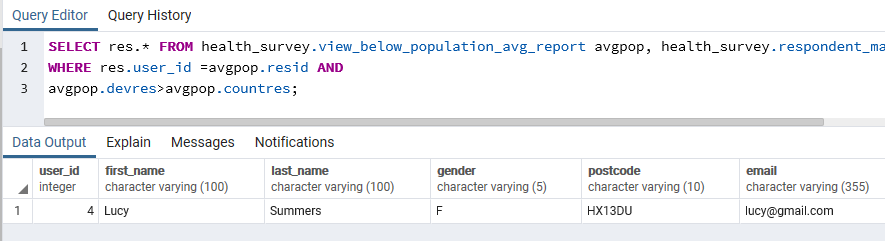


1. **View created for Admin use case 4.b BELOW POPULATION AVERAGE report**

View is created with response count and the value which is below 1 standard deviation from the population average

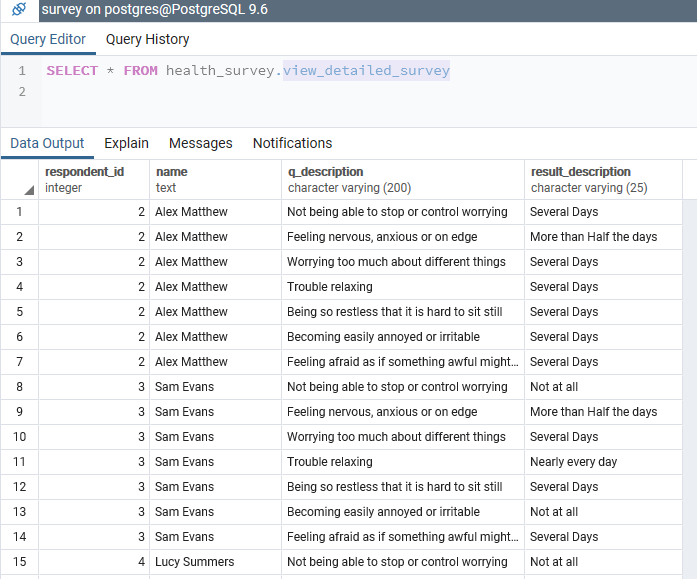


Report can be viewed as follows



Note: refer function ***health\_survey.view\_below\_population\_avg\_report*** for definition.

Also created view complete survey as questions and answers and respective respondent id and name as the survey\_result table has reference ids to respective tables.



Note: refer function ***health\_survey.*** ***view\_detailed\_survey*** for definition.

--------------------------------------**END OF DOCUMENT**--------------------------------