D205 Performance Assessment

Matthew E. Heino

D205 Data Acquisition

Introduction

In this assessment, the requirements are to use tables from the original database and the add-on CSV file to answer a research question that has a bearing on the operations of a business institution. The only requirement was to use only SQL and the Postgres database to extract, load, and then gather the required result – the answer to the research question. The research question will be discussed in the next section. At the end of this document, there is a full listing in the Code Appendix of the code that was required to meet the requirements of the assessment.

A: Research question

The research question that can be answered using SQL only is the following. What were the average responses for each of the items in the survey based on the states in which the survey was taken? This SQL research question can be further used to look for what matters to the patient and how can the organization use this information to better meet the needs of the patient.

A1: Identifying data

The data that is required came from three tables in the database. The first table is one that was in the original database with the assignment to the **patient** table. The next was a table that was in the original database – the **location** table. The final table that was required was created as an add-on for the assessment. The table that was created was **surveys_csv**.

The data that was required for processing the research question is the following. From the **patient** table the location_id, and the patient_id. From the **location** table the location_id

and state. From the surveys_csv table the eight items. These are item_1, item_2, item_3, item_4, item_5, item_6, item_7, and item_8.

The table below is a summary of the tables used, the data type, and the reason for choosing the column/data for this research question.

Table	Column Used	Data Type	Reason
patient	location_id*	text	This column is needed to perform the join between the location table and the patient table. It establishes the relation to be able to retrieve other data.
patient	patient_id*	text	This column is needed to perform the join between the patient table and the surveys_csv table. It establishes the relation to be able to retrieve other data.
location	location_id*	text	This column is used to establish a relationship between the patient table and the location table based on the value of the location of the patient.
location	state	text	This data was used to group the results based on which state or territory the survey was taken.
surveys_csv	item_1	integer	The first response item from where the average will be calculated.
surveys_csv	item_2	integer	Item 2 from the survey where the average will be calculated.
surveys_csv	item_3	integer	Item 3 from the survey where the

			average will be calculated.
surveys_csv	item_4	integer	Item 4 from the survey where the average will be calculated
surveys_csv	item_5	integer	Item 5 from the survey where the average will be calculated
surveys_csv	item_6	integer	Item 6 from the survey where the average will be calculated
surveys_csv	item_7	integer	Item 7 from the survey where the average will be calculated
surveys_csv	item_8	integer	Item 8 from the survey where the average will be calculated

Table 1. Tables, data types, and columns used

* These data items were not used in the calculation for the research question. They were needed to establish the relationship between the tables that are utilized in the final calculation and data output.

B: Entity relationship diagram

The diagram below shows how the entities are related to each other. Since there wasn't a common column between surveys_csv and Location it was required to use the patient table to establish this connection to retrieve the required data.

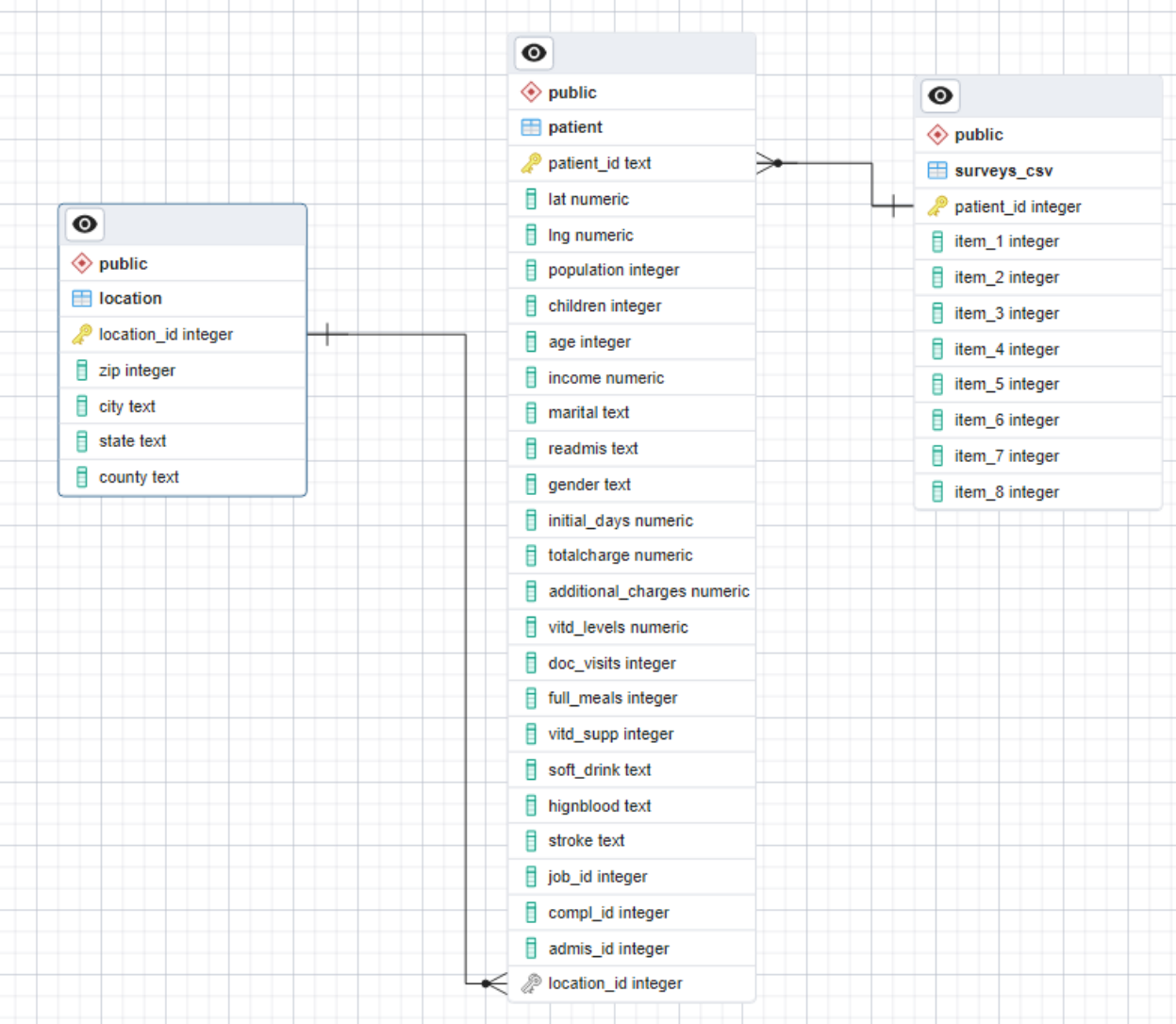

Figure 1 ERD diagram of the research question

B1: Code for the entity relationship diagram (ERD)

The code that was used to create the ERD that was discussed in the previous section is given below.

```
DROP TABLE IF EXISTS public.surveys_csv;

	CREATE TABLE IF NOT EXISTS public.surveys_csv
	(
```

```
    patient_id text COLLATE pg_catalog."default" NOT NULL,
    item_1 integer,
    item_2 integer,
    item_3 integer,
    item_4 integer,
    item_5 integer,
    item_6 integer,
    item_7 integer,
    item_8 integer,
    CONSTRAINT surveys_csv_pkey PRIMARY KEY (patient_id)
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public.surveys_csv
    OWNER to postgres;
```

Please note that the DROP clause is included. The DROP clause will not be needed if the table has not been created and will cause an error to be displayed about the non-existence of the table if issued in this manner. It was included here for completeness and can be commented out using "- - ".

B2L: Loading CSV data

After creating the table in the previous section the data can be loaded into it using the following SQL script.

```
copy public.surveys_csv (patient_id, item_1, item_2, item_3, item_4,
      item_5, item_6, item_7, item_8)
FROM 'C:/LabFiles/msurvey.csv'
        DELIMITER ',' CSV HEADER;
```

This will load the data that is found in the msurvey.csv. This file was provided in the assignment.

C: SQL Query

The SQL script below was created to answer the question that was stated in section "A: Research question."

```
SELECT
        loc.state,
        ROUND(AVG(surv.item_1), 2) AS ave_item_1,
        ROUND(AVG(surv.item_2), 2) AS ave_item_2,
        ROUND(AVG(surv.item_3),2) AS ave_item_3,
        ROUND(AVG(surv.item_4),2) AS ave_item_4,
        ROUND(AVG(surv.item_5),2) AS ave_item_5,
        ROUND(AVG(surv.item_6),2) AS ave_item_6,
        ROUND(AVG(surv.item_7),2) AS ave_item_7,
        ROUND(AVG(surv.item_8),2) AS ave_item_8
FROM
        patient AS pat
INNER JOIN location AS loc
        ON loc.location_id = pat.location_id
INNER JOIN surveys_csv as surv
        ON surv.patient_id = pat.patient_id
GROUP BY loc.state
ORDER BY loc.state DESC;
```

This code accomplishes the averaging of the responses and then groups them by the state or territory.

Below is the result of issuing the SQL script that was explained in the previous section.

	state text	ave_item_1 numeric	ave_item_2 numeric	ave_item_3 numeric	ave_item_4 numeric	ave_item_5 numeric	ave_item_6 numeric	ave_item_7 numeric	ave_item_8 numeric
1	WY	3.85	3.72	3.85	3.36	3.83	3.64	3.58	3.53
2	WV	3.56	3.52	3.54	3.58	3.53	3.62	3.63	3.55
3	WI	3.53	3.52	3.56	3.33	3.58	3.39	3.48	3.45
4	WA	3.47	3.56	3.62	3.48	3.54	3.44	3.37	3.38
5	VT	3.60	3.38	3.32	3.49	3.49	3.47	3.63	3.59
6	VA	3.55	3.51	3.43	3.60	3.44	3.56	3.43	3.56
7	UT	3.69	3.54	3.70	3.58	3.45	3.69	3.54	3.69
8	TX	3.50	3.50	3.51	3.58	3.50	3.54	3.55	3.50
9	TN	3.48	3.53	3.48	3.63	3.40	3.53	3.54	3.50
10	SD	3.50	3.57	3.50	3.52	3.62	3.40	3.50	3.51
11	SC	3.60	3.67	3.57	3.46	3.53	3.61	3.49	3.57
12	RI	3.47	3.40	3.33	3.13	3.47	4.00	4.07	3.60
13	PR	3.63	3.59	3.76	3.63	3.50	3.52	3.37	3.30
14	PA	3.47	3.44	3.50	3.52	3.48	3.49	3.48	3.44
15	OR	3.33	3.43	3.56	3.47	3.48	3.54	3.62	3.68
16	OK	3.38	3.37	3.43	3.46	3.47	3.38	3.39	3.60
17	OH	3.44	3.40	3.40	3.57	3.47	3.54	3.43	3.49
18	NY	3.53	3.43	3.49	3.55	3.46	3.54	3.51	3.47
19	NV	3.72	3.69	3.67	3.56	3.33	3.83	3.59	3.87
20	NM	3.54	3.48	3.55	3.54	3.36	3.77	3.52	3.51
21	NJ	3.48	3.45	3.38	3.50	3.47	3.59	3.47	3.47
22	NH	3.54	3.53	3.44	3.53	3.39	3.47	3.60	3.47
23	NE	3.53	3.55	3.45	3.43	3.46	3.59	3.53	3.54
24	ND	3.27	3.39	3.22	3.49	3.68	3.40	3.42	3.51
25	NC	3.49	3.50	3.55	3.43	3.56	3.54	3.51	3.61
26	MT	3.59	3.66	3.57	3.64	3.36	3.41	3.51	3.32
27	MS	3.67	3.60	3.54	3.53	3.57	3.48	3.48	3.64
28	MO	3.51	3.50	3.38	3.67	3.44	3.53	3.40	3.55
29	MN	3.59	3.53	3.60	3.46	3.55	3.63	3.46	3.51
30	MI	3.47	3.48	3.51	3.47	3.57	3.47	3.43	3.51

Figure 2: Part one of the result set

	state text	ave_item_1 numeric	ave_item_2 numeric	ave_item_3 numeric	ave_item_4 numeric	ave_item_5 numeric	ave_item_6 numeric	ave_item_7 numeric	ave_item_8 numeric
23	NE	3.53	3.55	3.45	3.43	3.46	3.59	3.53	3.54
24	ND	3.27	3.39	3.22	3.49	3.68	3.40	3.42	3.51
25	NC	3.49	3.50	3.55	3.43	3.56	3.54	3.51	3.61
26	MT	3.59	3.66	3.57	3.64	3.36	3.41	3.51	3.32
27	MS	3.67	3.60	3.54	3.53	3.57	3.48	3.48	3.64
28	MO	3.51	3.50	3.38	3.67	3.44	3.53	3.40	3.55
29	MN	3.59	3.53	3.60	3.46	3.55	3.63	3.46	3.51
30	MI	3.47	3.48	3.51	3.47	3.57	3.47	3.43	3.51
31	ME	3.40	3.52	3.43	3.46	3.60	3.45	3.34	3.34
32	MD	3.61	3.56	3.53	3.41	3.29	3.72	3.50	3.48
33	MA	3.46	3.52	3.42	3.52	3.53	3.64	3.54	3.45
34	LA	3.58	3.53	3.59	3.53	3.44	3.58	3.44	3.59
35	KY	3.48	3.47	3.43	3.37	3.52	3.42	3.44	3.45
36	KS	3.62	3.60	3.61	3.55	3.46	3.70	3.51	3.54
37	IN	3.50	3.64	3.64	3.41	3.58	3.55	3.61	3.60
38	IL	3.60	3.56	3.53	3.56	3.42	3.57	3.50	3.59
39	ID	3.70	3.59	3.72	3.44	3.50	3.54	3.53	3.68
40	IA	3.54	3.61	3.60	3.51	3.58	3.51	3.44	3.51
41	HI	3.50	3.63	3.53	3.73	3.37	3.30	3.70	3.73
42	GA	3.60	3.60	3.61	3.51	3.46	3.45	3.51	3.55
43	FL	3.58	3.51	3.53	3.52	3.49	3.51	3.54	3.44
44	DE	3.39	3.44	3.39	3.50	3.72	3.22	3.22	3.28
45	DC	3.50	3.60	3.40	3.20	3.30	3.20	3.80	3.80
46	CT	3.46	3.38	3.51	3.45	3.65	3.41	3.45	3.41
47	CO	3.51	3.45	3.39	3.38	3.58	3.51	3.49	3.39
48	CA	3.47	3.52	3.51	3.50	3.54	3.42	3.46	3.48
49	AZ	3.38	3.29	3.33	3.42	3.43	3.31	3.47	3.49
50	AR	3.44	3.29	3.52	3.48	3.63	3.38	3.55	3.53
51	AL	3.58	3.58	3.59	3.59	3.40	3.71	3.67	3.47
52	AK	3.62	3.58	3.58	3.74	3.45	3.45	3.48	3.40

Figure 3: Part two of the result set

There are 52 results in the set. There is a row for each of the fifty states plus a row for Washington DC and Puerto Rico.

C1: CSV files

The code to direct the results to a CSV file is given below.

```
COPY(
    SELECT
```

```
                loc.state,
                ROUND(AVG(surv.item_1), 2) AS ave_item_1,
                ROUND(AVG(surv.item_2), 2) AS ave_item_2,
                ROUND(AVG(surv.item_3),2) AS ave_item_3,
                ROUND(AVG(surv.item_4),2) AS ave_item_4,
                ROUND(AVG(surv.item_5),2) AS ave_item_5,
                ROUND(AVG(surv.item_6),2) AS ave_item_6,
                ROUND(AVG(surv.item_7),2) AS ave_item_7,
                ROUND(AVG(surv.item_8),2) AS ave_item_8
        FROM
                patient AS pat
        INNER JOIN location AS loc
                ON loc.location_id = pat.location_id
        INNER JOIN surveys_csv as surv
                ON surv.patient_id = pat.patient_id
        GROUP BY loc.state
        ORDER BY loc.state DESC
)
TO 'C:/LabFiles/results.csv'
DELIMITER ',' CSV HEADER;
```

The CSV file is submitted as a separate attachment. The title of the attachment is “results.csv”

D: Add-on File

The question as to when the surveys_csv and associated data should be updated is based on the need to know what the patient desires. It would be advantageous to review the data and update it once a business quarter.

D1: Explanation of the time period

The rationale behind the time frame stated in the previous section is that this is not a very pressing matter, but it should be reviewed to make sure that the important areas are being addressed and to receive insight into what could be improved from the patient experience perspective. It will help to have a grasp on what the patient deems most important. The time frame is good to see what items in the survey have changed over time, but this question is not for this assessment. This question could be addressed in later research questions.

E: Panopto Video of Code

The code for the Panopto video can be found here:

- https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=5822cdc8-ea62-4ce8-b298-b09e00c18f9a

References

F: Web sources

None – the only resources that were used for the creation of this assessment were the ones that were provided by the institution and previous knowledge of SQL.

G: Sources

None – no citations were used in answering the questions for this assessment.

Code Appendix

Code to create the table based on the ERD:

```
DROP TABLE IF EXISTS public.surveys_csv;


	CREATE TABLE IF NOT EXISTS public.surveys_csv
	(
	    patient_id text COLLATE pg_catalog."default" NOT NULL,
	    item_1 integer,
	    item_2 integer,
	    item_3 integer,
	    item_4 integer,
	    item_5 integer,
	    item_6 integer,
	    item_7 integer,
	    item_8 integer,
	    CONSTRAINT surveys_csv_pkey PRIMARY KEY (patient_id)
	)
	TABLESPACE pg_default;
	ALTER TABLE IF EXISTS public.surveys_csv
	    OWNER to postgres;
```

Code to load the data from the CSV file:

```
copy public.surveys_csv (patient_id, item_1, item_2, item_3, item_4, item_5,
item_6, item_7, item_8)
	FROM 'C:/LabFiles/msurvey.csv'
		DELIMITER ',' CSV HEADER;
```

Code for the research question:

```
SELECT
        loc.state,
        ROUND(AVG(surv.item_1), 2) AS ave_item_1,
        ROUND(AVG(surv.item_2), 2) AS ave_item_2,
        ROUND(AVG(surv.item_3),2) AS ave_item_3,
        ROUND(AVG(surv.item_4),2) AS ave_item_4,
        ROUND(AVG(surv.item_5),2) AS ave_item_5,
        ROUND(AVG(surv.item_6),2) AS ave_item_6,
        ROUND(AVG(surv.item_7),2) AS ave_item_7,
        ROUND(AVG(surv.item_8),2) AS ave_item_8
FROM
        patient AS pat
INNER JOIN location AS loc
        ON loc.location_id = pat.location_id
INNER JOIN surveys_csv as surv
        ON surv.patient_id = pat.patient_id
GROUP BY loc.state
ORDER BY loc.state DESC;
```

Code to save the results to CSV:

```
COPY(
        SELECT
                loc.state,
                ROUND(AVG(surv.item_1), 2) AS ave_item_1,
                ROUND(AVG(surv.item_2), 2) AS ave_item_2,
                ROUND(AVG(surv.item_3),2) AS ave_item_3,
                ROUND(AVG(surv.item_4),2) AS ave_item_4,
```

```
                ROUND(AVG(surv.item_5),2) AS ave_item_5,
                ROUND(AVG(surv.item_6),2) AS ave_item_6,
                ROUND(AVG(surv.item_7),2) AS ave_item_7,
                ROUND(AVG(surv.item_8),2) AS ave_item_8
        FROM
                patient AS pat
        INNER JOIN location AS loc
                ON loc.location_id = pat.location_id
        INNER JOIN surveys_csv as surv
                ON surv.patient_id = pat.patient_id
        GROUP BY loc.state
        ORDER BY loc.state DESC
)
TO 'C:/LabFiles/results.csv'
DELIMITER ',' CSV HEADER;
```