Data Acquisition Performance Assessment

Alfred Conrad Santos

Western Governors University

Data Acquisition - D205

William Sewell

8 Feb 2024

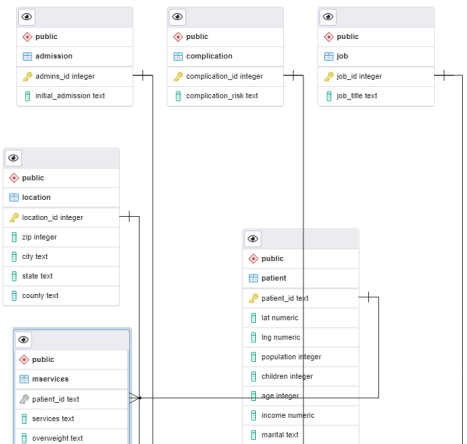
A: Research Question

My research question for this project is, "What is the average number of Vitamin-D levels for patients with anxiety?" This question is relevant to understanding the relationship between anxiety and Vitamin-D levels among patients in the hospital setting. By investigating this correlation, healthcare providers can gain insights into potential connections between mental health conditions and nutritional factors, which may inform treatment strategies and improve patient care.

A1: Identifying Data

So, to answer the question "What is the average number of Vitamin-D levels for patients with anxiety?" We need the "VitD_levels" column from the "patient" table in the original data set and the "Anxiety" column from the additional CSV file (mservices.csv), both matched by the patient_id column. These columns contain the necessary information to determine the average number of Vitamin-D levels for patients with anxiety. The "VitD_levels" column likely contains numerical data representing the levels of Vitamin-D in patients' blood samples. This column has an integer data type.The "anxiety" column contains textual data indicating whether a patient has anxiety or not. This column has a text data type .By properly identifying the data types of these columns, we ensure accurate data analysis and interpretation for addressing the research question.

B:



B1:

```
CREATE TABLE public.admission
(
    admins_id integer NOT NULL,
    initial_admission text,
    PRIMARY KEY (admins_id)
);
CREATE TABLE public.complication
(
    complication_id integer NOT NULL,
    complication_risk text,
    PRIMARY KEY (complication_id)
);
CREATE TABLE public.job
(
    job_id integer NOT NULL,
    job_title text,
    PRIMARY KEY (job_id)
);
CREATE TABLE public.location
(
    location_id integer NOT NULL,
    zip integer,
    city text,
    state text,
```

```
    county text,
    PRIMARY KEY (location_id)
);
CREATE TABLE public.patient
(
    patient_id text NOT NULL,
    lat numeric,
    lng numeric,
    population integer,
    children integer,
    age integer,
    income numeric,
    marital text,
    readmis text,
    gender text,
    initial_days numeric,
    totalcharge numeric,
    additional_charges numeric,
    vitd_levels numeric,
    doc_visits integer,
    full_meals integer,
    vitd_supp integer,
    soft_drink text,
    hignblood text,
    stroke text,
    job_id integer,
    compl_id integer,
    admis_id integer,
    location_id integer,
    PRIMARY KEY (patient_id)
);
ALTER TABLE public.patient
    ADD FOREIGN KEY (admis_id)
    REFERENCES public.admission (admins_id)
    NOT VALID;
ALTER TABLE public.patient
```

```
    ADD FOREIGN KEY (compl_id)
    REFERENCES public.complication (complication_id)
    NOT VALID;
ALTER TABLE public.patient
    ADD FOREIGN KEY (job_id)
    REFERENCES public.job (job_id)
    NOT VALID;
ALTER TABLE public.patient
    ADD FOREIGN KEY (location_id)
    REFERENCES public.location (location_id)
    NOT VALID;
ALTER TABLE public.mservices
    ADD FOREIGN KEY (patient_id)
    REFERENCES public.patient (patient_id)
    NOT VALID;
END;
```

The script above was generated from pgAdmin. I used the GUI tool to create the columns, select data types, etc. and then copied the SQL code that this process generated and executed.

B2:

```
CREATE TABLE public.mservices
(
    patient_id text NOT NULL,
    services text,
    overweight text,
    arthritis text,
    diabetes text,
    hyperlipidemia text,
    backpain text,
    anxiety text,
    allergic_rhinitis text,
    reflux_esophagitis text,
    asthma text
);
```

```
--command " "\\copy public.mservices (patient_id, services, overweight, arthritis, diabetes, hyperlipidemia, backpain, anxiety, allergic_rhinitis, reflux_esophagitis, asthma) FROM 'C:/LabFiles/Medical/MSERVI~1.CSV' DELIMITER ',' CSV HEADER QUOTE '\"' ESCAPE '"';""
```

Utilizing the PGADMIN Gui again.This code worked successfully to generate the table and allow me to insert all of the data from the CSV file. However, the text datatype was used to maintain consistency with the rest of the database, and such optimizations were outside the scope of this assignment.

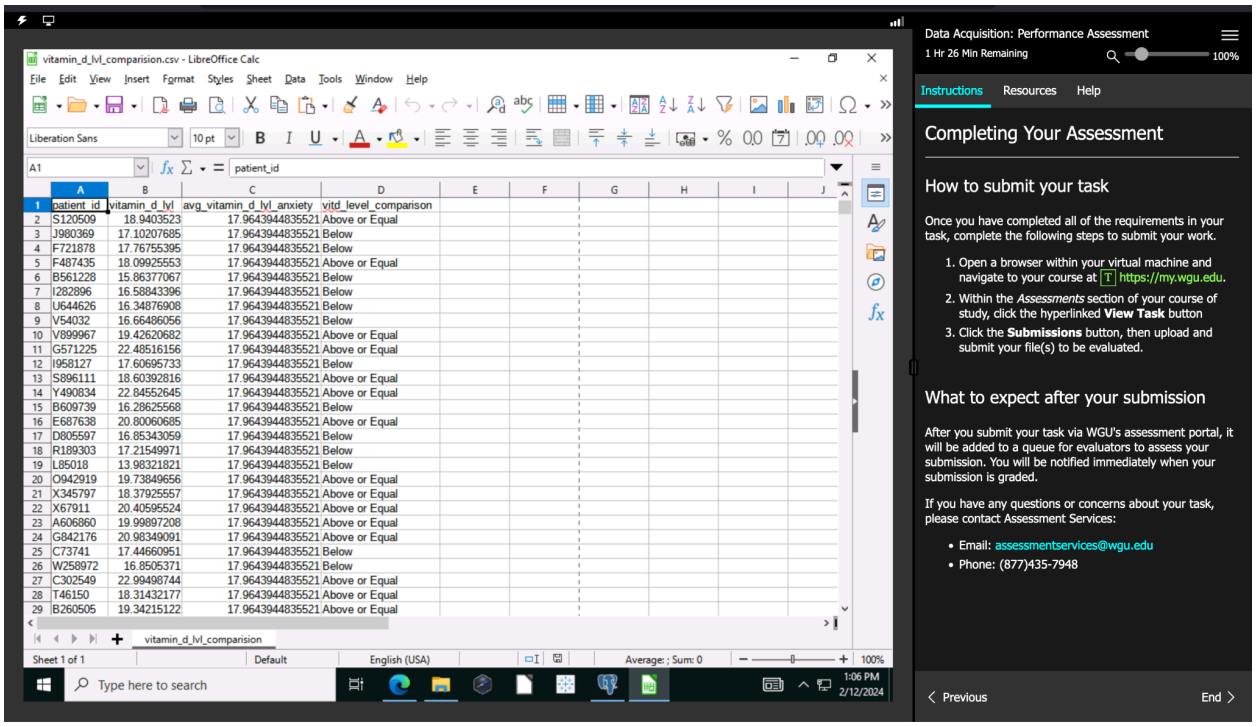
The following code afterwards is used to copy the CSV file into my newly created table.

C:

```
SELECT
	p.patient_id,
	p.vitd_levels AS vitamin_d_lvl,
	AVG(p.vitd_levels) OVER () AS avg_vitamin_d_lvl_anxiety,
	CASE
	WHEN p.vitd_levels >= AVG(p.vitd_levels) OVER () THEN 'Above or Equal'
	ELSE 'Below'
	END AS vitd_level_comparison
FROM
	patient p
JOIN
	mservices m ON p.patient_id = m.patient_id
WHERE
```

```
m.anxiety = 'Yes';
```

Here is a result of that query displayed as a CSV to allow more info



patient_id	vitamin_d_lvl	avg_vitamin_d_lvl_anxiety	vitd_level_comparison
S120509	18.9403523	17.9643944835521	Above or Equal
J980369	17.10207685	17.9643944835521	Below
F721878	17.76755395	17.9643944835521	Below
F487435	18.09925553	17.9643944835521	Above or Equal
B561228	15.86377067	17.9643944835521	Below
I282896	16.58843396	17.9643944835521	Below
U644626	16.34876908	17.9643944835521	Below
V54032	16.66486056	17.9643944835521	Below
V899967	19.42620682	17.9643944835521	Above or Equal
G571225	22.48516156	17.9643944835521	Above or Equal
I958127	17.60695733	17.9643944835521	Below
S896111	18.60392816	17.9643944835521	Above or Equal
Y490834	22.84552645	17.9643944835521	Above or Equal
B609739	16.28625568	17.9643944835521	Below
E687638	20.80060685	17.9643944835521	Above or Equal
D805597	16.85343059	17.9643944835521	Below
R189303	17.21549971	17.9643944835521	Below
L85018	13.98321821	17.9643944835521	Below
O942919	19.73849656	17.9643944835521	Above or Equal
X345797	18.37925557	17.9643944835521	Above or Equal
X67911	20.40595524	17.9643944835521	Above or Equal
A606860	19.99897208	17.9643944835521	Above or Equal
G842176	20.98349091	17.9643944835521	Above or Equal
C73741	17.44660951	17.9643944835521	Below
W258972	16.8505371	17.9643944835521	Below
C302549	22.99498744	17.9643944835521	Above or Equal
T46150	18.31432177	17.9643944835521	Above or Equal
B260505	19.34215122	17.9643944835521	Above or Equal

Let's break down the code step by step:

1. **SELECT**: This is the keyword used to select columns from the database.

2. **p.patient_id**: This selects the patient ID from the 'patient' table and assigns it the alias 'patient_id'.
3. **p.vitd_levels AS vitamin_d_lvl**: This selects the vitamin D levels of the patients from the 'patient' table and assigns it the alias 'vitamin_d_lvl'.
4. **AVG(p.vitd_levels) OVER () AS avg_vitamin_d_lvl_anxiety**: This calculates the average of the vitamin D levels for all patients who have anxiety (where m.anxiety = 'Yes'). The `OVER()` function is used to calculate an aggregate function like AVG() across all rows in the result set.
5. **CASE**: This is a conditional statement. It evaluates conditions and returns a value based on those conditions.
6. **WHEN p.vitd_levels >= AVG(p.vitd_levels) OVER () THEN 'Above or Equal' ELSE 'Below' END AS vitd_level_comparison**: This checks if the vitamin D level of a patient is greater than or equal to the average vitamin D level of patients with anxiety. If it is, it assigns the label 'Above or Equal'; otherwise, it assigns 'Below'.
7. **FROM patient p JOIN mservices m ON p.patient_id = m.patient_id**: This specifies which tables are being used for the query. It selects data from the 'patient' table (aliased as 'p') and joins it with the 'mservices' table (aliased as 'm') based on the patient ID.
8. **WHERE m.anxiety = 'Yes'**: This filters the data to only include patients who have anxiety. It checks the 'anxiety' column in the 'mservices' table.

In summary, this SQL query selects patient IDs and their corresponding vitamin D levels, calculates the average vitamin D level for patients with anxiety, compares each patient's vitamin D level to the average, and labels them accordingly ('Above or Equal' if the vitamin D level is greater than or equal to the average, 'Below' otherwise).

D :A specific time period for acquiring and refreshing the add-on file in the hospital database could be every month.

- Frequency: Monthly updates strike a balance between ensuring data relevance and feasibility. It provides healthcare providers with relatively current information without overwhelming the system with constant updates.
- Clinical Relevance: Many medical guidelines, protocols, and procedures are updated on a monthly basis or at regular intervals. Therefore, monthly updates can help ensure that healthcare providers have access to the latest evidence-based practices and guidelines.
-Feedback and Iteration: Monthly updates provide regular opportunities for feedback from healthcare providers and stakeholders. This feedback can be used to assess the effectiveness of the refresh period and make adjustments as needed to better meet the needs of the hospital and its staff.
-Overall, a monthly refresh period balances the need for data relevance, operational efficiency, and regulatory compliance, while also allowing for flexibility to adapt to changing clinical and organizational requirements.

F. No web sources were used to acquire data or segments of third-party code to support the application.

G. No sources, Nor in-text citations and references, for content that is quoted, paraphrased,

or summarized were used

H. Demonstrate professional communication in the content and presentation of your submission.