Performance Assessment: Advanced Data Management - VDM1

Andrea Hayes

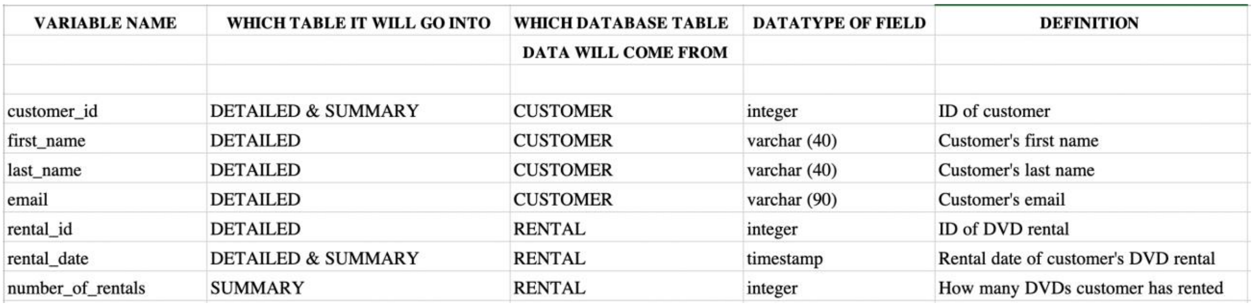
Western Governors University Advanced Data Management D191

April 28th, 2023

A. A. Summarize one real-world business report that can be created from the attached Data Sets and Associated Dictionaries.

The attached Data Sets referencing the DVD Rental Database could be used to answer a plethora of different business inquiries. For the sake of this business report, we will narrow the focus to include the extraction of one simple metric: how many rentals has each customer made? With the data uncovered by this question, stakeholders of the DVD rental business would have the ability to implement various promotional offers and to track their efficacy. As a suggestion, this DVD Rental business could initiate a ‘Rent Nine get one Rental Free’ promotion and track the rentals each month to determine if the promotion had a statistically significant increase on how many DVDs each customer rented.

A1. Describe the data used for the report.



VARIABLE NAME	WHICH TABLE IT WILL GO INTO	WHICH DATABASE TABLE	DATATYPE OF FIELD	DEFINITION
		DATA WILL COME FROM		
customer_id	DETAILED & SUMMARY	CUSTOMER	integer	ID of customer
first_name	DETAILED	CUSTOMER	varchar (40)	Customer's first name
last_name	DETAILED	CUSTOMER	varchar (40)	Customer's last name
email	DETAILED	CUSTOMER	varchar (90)	Customer's email
rental_id	DETAILED	RENTAL	integer	ID of DVD rental
rental_date	DETAILED & SUMMARY	RENTAL	timestamp	Rental date of customer's DVD rental
number_of_rentals	SUMMARY	RENTAL	integer	How many DVDs customer has rented

To determine the effectiveness of the suggested promotion, stakeholders of the DVD Rental Business would need access to both data about the customers and the rentals. Having both data subsets would give insight into how many times each customer rented a video over a specified timeframe. With this data, at the start of the promotion, the business could establish a baseline for how frequently a customer rented a video, and later compare that baseline to data gathered each incremental month. Tracking only customers who participate in the promotion, this would yield metrics that would reveal the percentage of increased DVD rentals in relation to the promotion. While this is beyond the scope of the present report, a separate table would need to be generated that would extract the customers who participated in the promotion from the customers who did *not* participate in the promotion, isolating this specific data set for analysis.

A2. Identify two or more specific tables from the given dataset that will provide the data necessary for the detailed and the summary sections of the report.

The data used for this report would include data from both the customer table and the rental table within the DVD rental database. For the detailed report, this data would be used to identify the customers' full name, email, and how many rentals they accrued over the course of the promotion. For the summary report, this information would be condensed so to quickly assess how many rentals each customer made month over month.

A3. Identify the specific fields that will be included in the detailed and the summary sections of the report.

For the *detailed section* of the report, stakeholders of the DVD Rental Business would need access to these specific fields:

- customer_id
- first_name
- last_name
- email
- rental_id
- rental_date

For the *summary section* of the report, stakeholders of the DVD Rental Business would need access to these specific fields:

- customer_id
- rental_date
- number_of_rentals

A4. Identify one field in the detailed section that will require a custom transformation and explain why it should be transformed. For example, you might translate a field with a value of 'N' to 'No' and 'Y' to 'Yes'.

For the sake of tracking the data related to the promotion, a custom function will need to be created that converts the customers' rental dates into the actual number of rentals made. In the current database, only the rental date and return date is available. For the sake of tracking the efficacy of the promotion, the rental dates column will need to be counted and converted into the number of rentals per customer. This will enable the stakeholders to determine the quantitative number of rentals made over a pre-determined timeframe, so to measure whether the number of rentals increase in any statistically significant way.

A5. Explain the different business uses of the detailed and the summary sections of the report.

The *detailed report* would be used to store every customer transaction over the span of that month. This would serve to reveal how many DVDs each customer rented per month. If desired, stakeholders could perform a deeper analysis to identify the customers whose rentals increased more than others and isolate which circumstances influenced those customers to rent more. Customer emails would be made available to send reminders of the promotion or to notify customers of new promotions the business might wish to push. To determine the effectiveness of the promotion, it would be necessary to track whether the customers' rentals increased in any statistically significant way over a pre-determined time frame. The metrics produced by such an inquiry would reveal whether the promotion should be continued, modified, or discontinued.

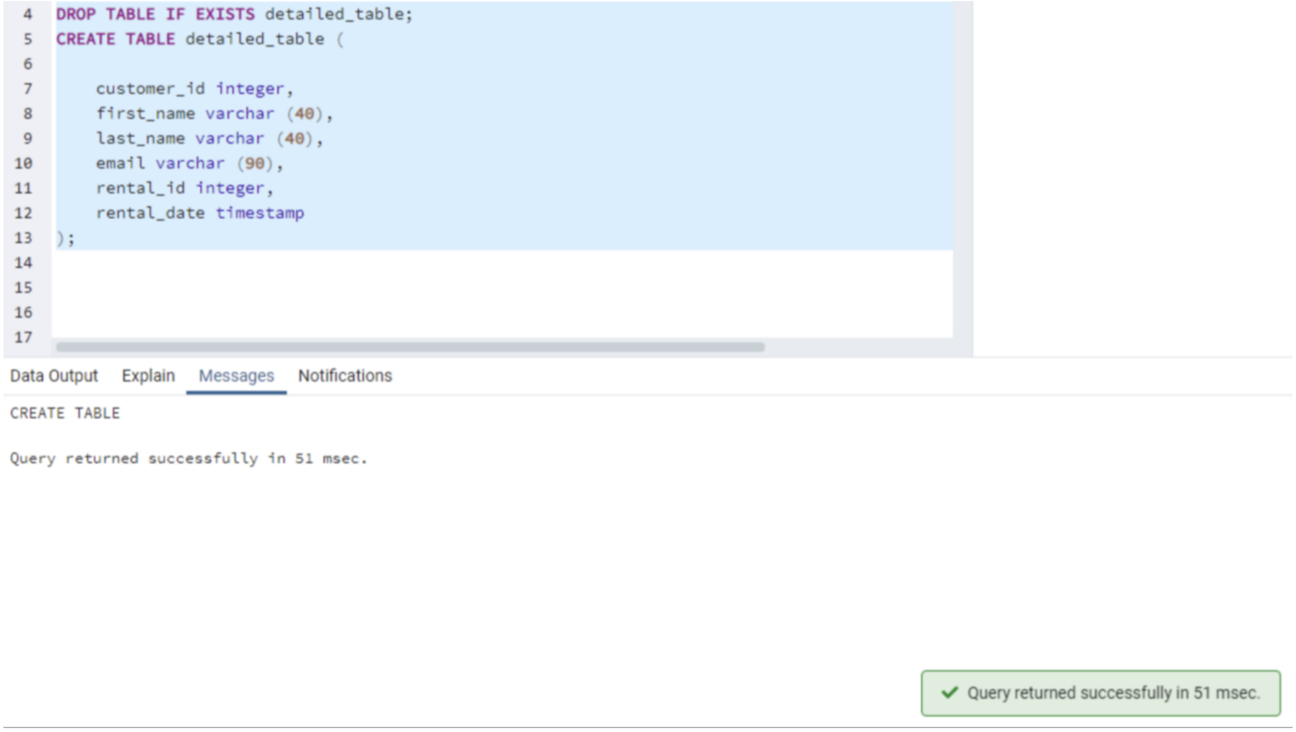
The *summary report* would be used to compare essential findings with the baseline from the previous month, to produce a quick month-over-month review of how many DVDs each customer rented.

A6. Explain how frequently your report should be refreshed to remain relevant to stakeholders.

For optimal tracking of the data, I would suggest refreshing this report monthly. Prior to running the stored procedure, the baseline data from the previous month should be highlighted and presented. That way, the data from the *prior* month can be benchmarked against the data from the *current* month, to determine if the promotion resulted in a statistically significant increase in DVD rentals.

B. Write a SQL code that creates the tables to hold your report sections.

```
--Section B starts here:
--Create Detailed Report:
```

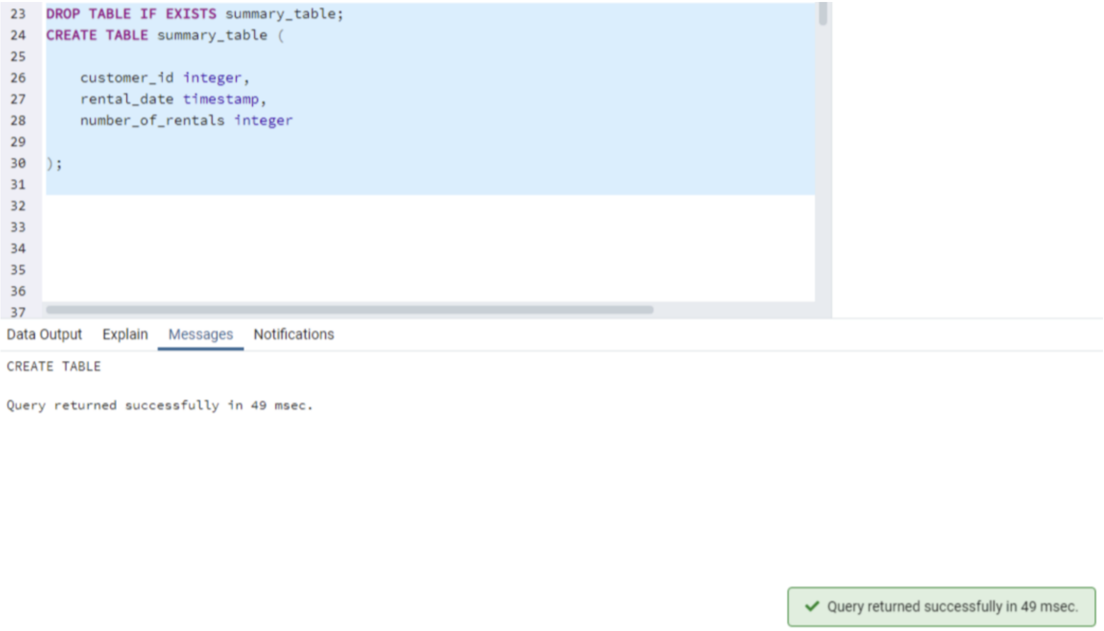


```
DROP TABLE IF EXISTS detailed_table;
CREATE TABLE detailed_table (

    customer_id integer,
    first_name varchar (40),
    last_name varchar (40),
    email varchar (90),
    rental_id integer,
    rental_date timestamp
);
```

```
CREATE TABLE

Query returned successfully in 51 msec.
```

✓ Query returned successfully in 51 msec.

```
--Create Summary Report:
```



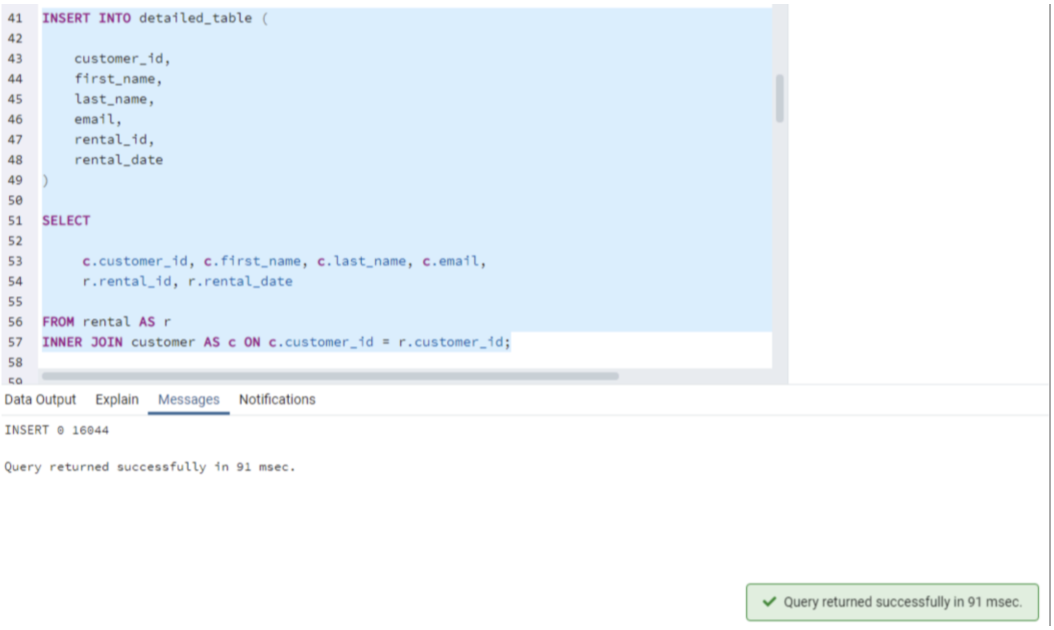
```sql
DROP TABLE IF EXISTS summary_table;
CREATE TABLE summary_table (

    customer_id integer,
    rental_date timestamp,
    number_of_rentals integer

);
```

```
CREATE TABLE

Query returned successfully in 49 msec.
```

✓ Query returned successfully in 49 msec.

C. Write a SQL query that will extract the raw data needed for the Detailed section of your report from the DVD rental database and verify the data's accuracy.

```
--Section C starts here:

/* Extract raw data from Detailed Section of the report from the DVD rental
database into detailed_table */
```



```sql
INSERT INTO detailed_table (

    customer_id,
    first_name,
    last_name,
    email,
    rental_id,
    rental_date
)

SELECT

     c.customer_id, c.first_name, c.last_name, c.email,
     r.rental_id, r.rental_date

FROM rental AS r
INNER JOIN customer AS c ON c.customer_id = r.customer_id;
```

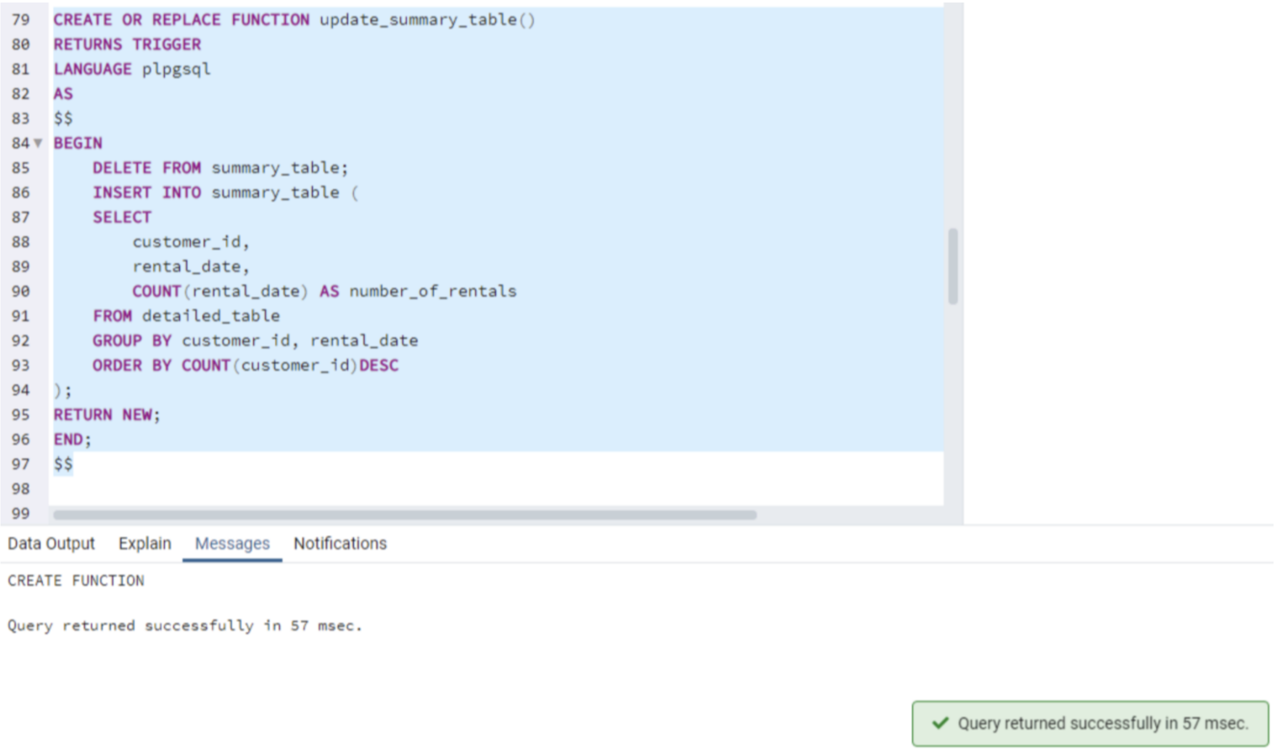
```
INSERT 0 16044

Query returned successfully in 91 msec.
```

✓ Query returned successfully in 91 msec.

```
/* This will extract the raw data from the customer table and rental
table and insert it into the detailed_table. */

/* Verify the data’s accuracy by juxtaposing detailed_table to the raw
data extracted from the rental table and customer table. */

/* Verify all the variables from the extracted raw data are present in
the output and in the correct order. To do this, perform a SELECT *
FROM query on the detailed_table to acknowledge that the datatypes in
the fields are correct. */

/* Verify that the data in the detailed_table is the correct match to
the category of data in rental table and customer table. To do this,
perform a SELECT * FROM query on the detailed_table to acknowledge
that the datatypes in the fields are correct. */
```

D. Write code for function(s) that perform the transformation(s) you identified in part A4.

```
--Section D starts here:
--Create function:
```



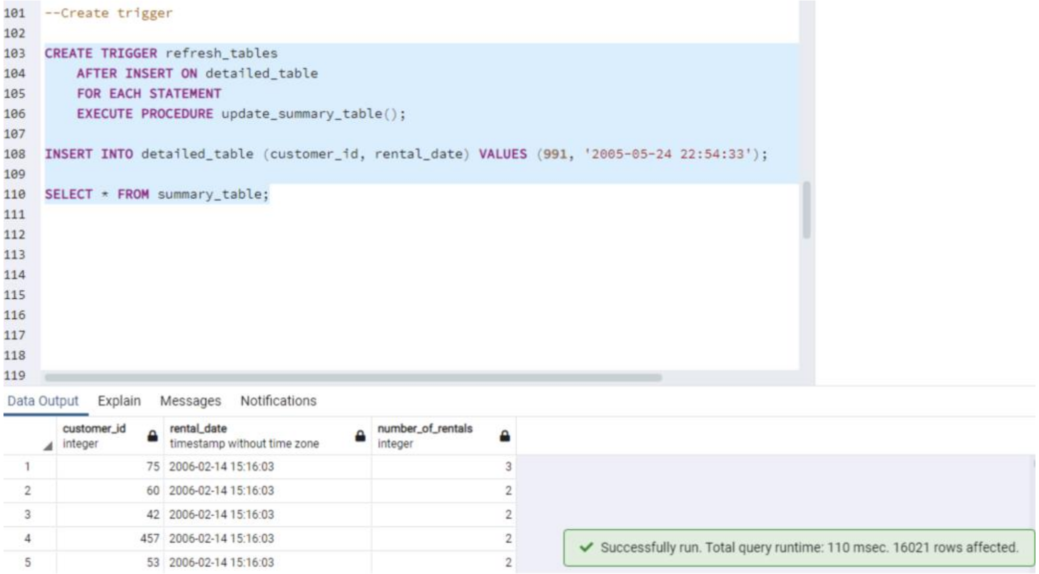
```
CREATE OR REPLACE FUNCTION update_summary_table()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN
    DELETE FROM summary_table;
    INSERT INTO summary_table (
    SELECT
        customer_id,
        rental_date,
        COUNT(rental_date) AS number_of_rentals
    FROM detailed_table
    GROUP BY customer_id, rental_date
    ORDER BY COUNT(customer_id)DESC
);
RETURN NEW;
END;
$$
```

```
CREATE FUNCTION

Query returned successfully in 57 msec.
```

✓ Query returned successfully in 57 msec.

E. Write a SQL code that creates a trigger on the detailed table of the report that will continually update the summary table as data is added to the detailed table.

`--Section E starts here:`



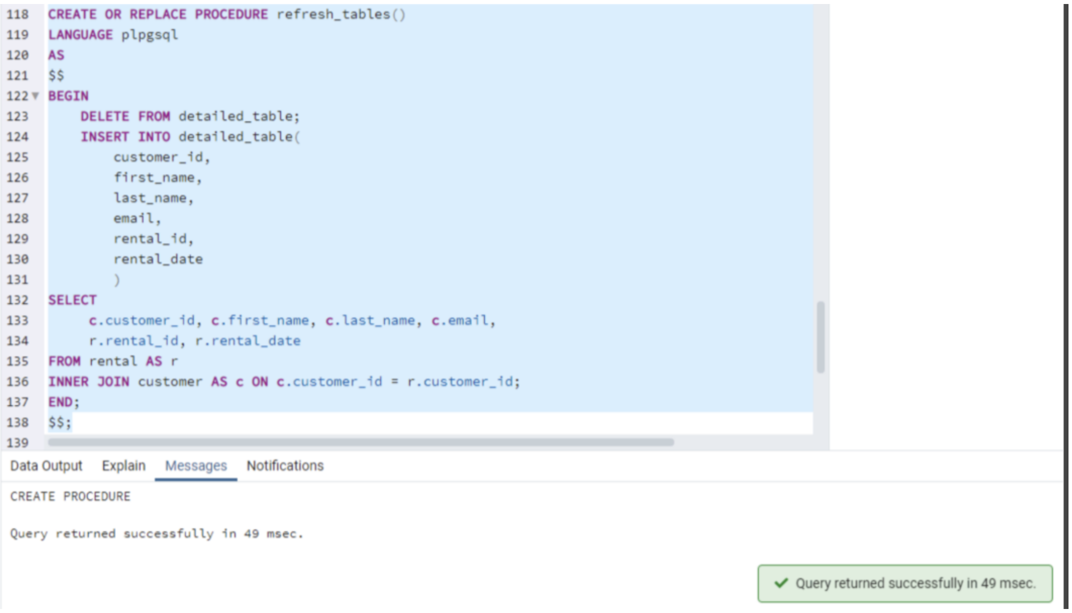
```sql
--Create trigger

CREATE TRIGGER refresh_tables
    AFTER INSERT ON detailed_table
    FOR EACH STATEMENT
    EXECUTE PROCEDURE update_summary_table();

INSERT INTO detailed_table (customer_id, rental_date) VALUES (991, '2005-05-24 22:54:33');

SELECT * FROM summary_table;
```

	customer_id integer	rental_date timestamp without time zone	number_of_rentals integer
1	75	2006-02-14 15:16:03	3
2	60	2006-02-14 15:16:03	2
3	42	2006-02-14 15:16:03	2
4	457	2006-02-14 15:16:03	2
5	53	2006-02-14 15:16:03	2

✓ Successfully run. Total query runtime: 110 msec. 16021 rows affected.

F. Create a stored procedure that can be used to refresh the data in *both* your detailed and summary tables. The procedure should clear the contents of the detailed and summary tables and perform the ETL load process from part C and include comments that identify how often the stored procedure should be executed.

`--Section F starts here:`



```sql
CREATE OR REPLACE PROCEDURE refresh_tables()
LANGUAGE plpgsql
AS
$$
BEGIN
    DELETE FROM detailed_table;
    INSERT INTO detailed_table(
        customer_id,
        first_name,
        last_name,
        email,
        rental_id,
        rental_date
        )
SELECT
      c.customer_id, c.first_name, c.last_name, c.email,
      r.rental_id, r.rental_date
FROM rental AS r
INNER JOIN customer AS c ON c.customer_id = r.customer_id;
END;
$$;
```

```
CREATE PROCEDURE

Query returned successfully in 49 msec.
```

✓ Query returned successfully in 49 msec.

To ensure data freshness and for the sake of tracking metrics, it is suggested that the stored procedure be run monthly on a repeating date. This would be large enough a timespan to gauge whether the 'Rent Nine get one Rental Free' promotion was significantly increasing DVD rentals.

To execute the stored procedure, the pgAgent job scheduling tool could be utilized. pgAgent is a job scheduling agent for Postgres databases, managed by pgAdmin. It runs as a daemon on Linux systems, systematically connecting to the database to check for procedures that need to execute.

G. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

Panopto video recording:

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6e286e95-cbbc-4f70-b8e2-aff10108cfdc

The operating system I'm using is a MacOS Monterey, Version 12.6.3 with an Apple M1 chip. I am using Labs on Demand Assessment Environment and DVD Database, which I can access through a virtual machine, opened through the Mozilla Firefox web browser. To create the queries and access the DVD Database, I am using PGAdmin 4, which is a web-based Graphical User Interface application that is used to communicate with PostgreSQL. PGAdmin 4 is accessing the relational DVD Rental Database off a remote server.