Name: TJ Kapil
Student ID: 010102369
Subject: D191 Advanced Data Management
Title: Performance Assessment

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

After analyzing all of the data tables, I noticed that the staff tables and rental tables could be joined to show how many sales a staff member makes to customers. This could show us which staff member is making the most sales. The DVD rental company management could then decide to reward the staff member that makes the highest sales. This would also motivate the other staff members to make more sales so they could also be rewarded. This would lead to more company profit overall, because more sales directly correlates with more income.

1. Describe the data used for the report.

For this report we would need the staff data and the DVD rental data. Joining the data from these two tables together would let us see which staff sold to which customer. If we can determine the total sales the different staff members made, we can see which staff member scored the highest in sales. We will use email data to send the staff members rewards, for example maybe giftcards or something else.

2. 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.

We will need two tables to generate this report. The first table is the staff table, and the second table is the rental table.

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

Detailed table: staff_id, first_name, last_name, address_id, email, store_id, customer_id, rental_date, rental_id

Summary table: complete_name, staff_id, email, all_rentals

4. 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'.

One field that will require a custom transformation will be the first_name and last_name fields. Initially these two fields will be separate. Concatenating these two fields together will let store

management know clearly who the employee is without having to refer to two columns. This will improve the overall readability of the table and make it more clear. The total number of sales a staff member made will also be recorded and grouped together based on the staff members name, since there are only two staff members.

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

The summary section of the report will provide us with information regarding which staff member made the most number of sales. This will help the management staff in understanding who they should reward. The detailed section of the report will provide us with information regarding which customer a staff member sold to. This way we can figure out if the staff member has a good relationship with a customer and they are bringing them back. So the detailed table will be an excellent source to get information on which customers regularly come back. You can also check if certain customers always go back to the same staff member.

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

This report can be refreshed every month or it can be refreshed whenever the management staff feels necessary, because the management staff can decide how often they want to give their sales employee's a reward. For example, they can decide to refresh this list every 6 months. Then they will receive information about which staff member made the most sales and can reward them based on that. It's important to refresh the list because it's possible another staff member made more sales than the one last time. Also a database administrator does not need to refresh the tables manually each time. They can set some sort of script so that it can be refreshed automatically at some time and date.

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

```
/* Creating the detailed table */
/* This will show all the rentals that an employee has been a part of */
CREATE TABLE DETAILED (
        rental_id INT PRIMARY KEY,
        last_name VARCHAR(50),
        first_name VARCHAR(50),
        staff_id INT,
        rental_date TIMESTAMP,
        email VARCHAR(50),
        address_id VARCHAR(90)
        customer_id INT,
        store_id INT
);
/* Check if detailed table exists */
SELECT *
FROM detailed

/* Creating summary table */
/* This table is a more condensed version of the detailed table and will show the employee's full
name, their staff id, their email to send rewards, and all_rentals

CREATE TABLE SUMMARY (
        staff_id INT PRIMARY KEY,
        complete_name VARCHAR(90),
        email VARCHAR(90),
        all_rentals INT
);
/* Check if summary table exists */
SELECT *
FROM summary
```

C. EXTRACT DATA INTO NEW TABLES (DETAILED, SUMMARY):

/* INSERTING DATA FROM THE GIVEN TABLES INTO DETAILED SECTION OF REPORT */

```
INSERT INTO DETAILED(rental_id, last_name, first_name, staff_id , rental_date, email,
address_id,  customer_id, store_id)
SELECT r.rental_id, s.last_name, s.first_name, s.staff_id, r.rental_date , s.email, s. address_id,
r.customer_id, s.store_id
FROM rental AS r
JOIN staff as S ON r.staff_id = s.staff_id;
```

/* CHECK IF INFORMATION HAS BEEN STORED IN DETAILED TABLE */

```
SELECT * FROM DETAILED;
```

/* INSERT INTO summary table INFORMATION FROM THE DETAILED TABLE THAT IS NEEDED */

```
INSERT INTO SUMMARY (
SELECT staff_id,
CONCAT(last_name,' ', first_name) AS complete_name,
email,
COUNT(staff_id)
FROM DETAILED
GROUP BY staff_id, complete_name, email
ORDER BY COUNT(staff_id) DESC
);
```

/* CHECK IF INFORMATION HAS BEEN STORES IN SUMMARY TABLE */

```
SELECT* FROM SUMMARY;
```

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

/* I will create a function that will be able to clear out the old summary table and insert new data into it*/
/* The transformation are concating the last and first name into on complete name, and adding up the the total rental sales a staff member makes to a customer using COUNT*/

```
CREATE FUNCTION reset_summary_table_function()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN

/* DELETE FROM OLD SUMMARY TABLE */
DELETE FROM SUMMARY;

/* INSERTING NEW DATA INTO SUMMARY TABLE */

INSERT INTO SUMMARY (
SELECT staff_id,
SELECT CONCAT(last_name,'  ', first_name) AS complete_name,
email,
COUNT(staff_id)
FROM DETAILED
GROUP BY staff_id, complete_name, email
ORDER BY COUNT(staff_id)  DESC
);

/* RETURN THE NEW SUMMARY TABLE WITH NEW DATA*/
RETURN NEW;
/* END THE FUNCTION*/RETURN
END;$$
```

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.

```
/* This will create a trigger which will constantly update the summary table as new data is added to the detailed table, it will use the reset summary table function */
CREATE TRIGGER reset_summary_table_trigger
AFTER INSERT ON DETAILED
FOR EACH STATEMENT
EXECUTE PROCEDURE reset_summary_table_function();
```

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.

/* The following procedure can run whenever the DVD rental store management feels like giving out rewards, it could be after a month or 6 months etc. It is also possible to write a script so the function can be run automatically instead of manually each time by a database administrator */

```
CREATE PROCEDURE reset_tables()
LANGUAGE plpgsql
AS $$
BEGIN

/* OLD DATA DELETED FROM DETAILED TABLE*/

DELETE FROM DETAILED;

INSERT INTO DETAILED(rental_id, last_name, first_name, staff_id , rental_date, email,
address_id, customer_id, store_id)
SELECT r.rental_id, s.last_name, s.first_name, s.staff_id, r.rental_date , s.email, s.address_id,
r.customer_id, s.store_id
FROM rental AS r
JOIN staff as s ON r.staff_id = s.staff_id;

/* OLD DATA DELETED FROM SUMMARY TABLE*/
DELETE FROM SUMMARY;

INSERT INTO SUMMARY (
SELECT staff_id,
CONCAT(last_name,' ', first_name) AS complete_name,
email,
COUNT(staff_id)
FROM DETAILED
GROUP BY staff_id, complete_name, email
ORDER BY COUNT(staff_id)  DESC
);

END; $$

/* TO CALL THE STORED PROCEDURE*/
```

CALL reset_tables()

SELECT * FROM SUMMARY

1. Explain how the stored procedure can be run on a schedule to ensure data freshness.

Data can be reset in the summary and detailed tables each month or whenever the store management feels would be necessary. When the data is reset would be dependent on when the store wants to give the employees rewards. In PgAgent GUI a job can be scheduled for a date and time to have it be automatic.

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.

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=337c297b-9292-4787-83a9-af0d013dd779

H. Record the web sources you used to acquire data or segments of third-party code to support the application if applicable. Be sure the web sources are reliable.

https://www.postgresqltutorial.com/postgresql-getting-started/postgresql-sample-database/
https://www.postgresqltutorial.com/postgresql-plpgsql/postgresql-create-procedure/
https://www.postgresqltutorial.com/postgresql-triggers/
https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-create-table/
https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-joins/
https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-select-into/
https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-insert/
https://www.postgresqltutorial.com/postgresql-aggregate-functions/

I. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Not used.