Tyler Meester
D191 - Advanced Data Management

A. Business Problem

Which 10 customers have rented the most DVDs from the business?

There are many people who may find themselves browsing through a DVD rental store, some who are loyal customers that have poured many dollars into the business. For such a business it's important to know who those customers are to offer rewards and encourage further business. Such customers are likely to recommend the business to their friends and family, providing for exceptional organic marketing and growth.

A1. Describe the data used: The data used in this report will come from the provided 'dvdrental' database. This database includes a great deal of information about the business's operations including rental inventory, customer information, payment/transaction information, and much more. The data from the database that is relevant to this problem is the Rental and Customer data. The rental data shows all the store's rental records which includes the rental id, rental date, inventory id, customer id, return date, staff id, and last update. The customer data shows all the customer records which includes customer id, store id, first name, last name, email, address, activity, creation date, last update, and active. To answer this business question we will need the a list of all the rental records from the rental table and the customers' first name and last name from the customer table. The data used will be of type int and varchar(255).

A2. Identify two or more specific tables to include: The report will utilize data specific to the 'rental' table as well as the 'customer' table in the 'dvdrental' database.

A3. Identify the specific fields that will be used:

rental.rental_id,
customer.customer_id,
customer.first_name,
customer.last_name

A4. Identify one field that will need to be transformed: customer.first_name and customer.last_name will be concatenated into one field, 'customer_name'.

A5. Explain different business uses:
This information can be used to reward customer loyalty and encourage organic growth for the business via customer referrals to friends and family. Organic marketing is known to be the most effective way to grow a business, and this report will prove to be very valuable for enabling effective organic marketing.

6. Explain how often report should be refreshed:
This report should be refreshed at the end of each business day in order to maintain data freshness.

Tyler Meester
D191 - Advanced Data Management

B. Write query that creates table and holds report sections

```
---------- Section B starts here----------
--Creating the detailed table
DROP TABLE IF EXISTS detailed_table;
CREATE TABLE detailed_table (
rental_id int PRIMARY KEY,
customer_id int,
first_name varchar(255),
last_name varchar(255)
);

--Validating the detailed_table
SELECT * FROM detailed_table;

--Creating the summary table
DROP TABLE IF EXISTS summary_table;
CREATE TABLE summary_table (
customer_id int,
customer_name varchar(255),
rental_count int
);

--Validating the summary_table
SELECT * FROM summary_table;
```

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

```
----------Section C starts here----------
-- Extracts the raw data needed from the customer and rental tables and inserts it into the
-- detailed table.
INSERT INTO detailed_table (
rental_id,
customer_id,
first_name,
last_name)

SELECT rental.rental_id,
customer.customer_id,
customer.first_name,
customer.last_name
FROM rental
INNER JOIN customer on customer.customer_id = rental.customer_id;
-- Verifies detailed table data accuracy
SELECT *
FROM detailed_table;
```

Tyler Meester
D191 - Advanced Data Management

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

----------Section D starts here----------
```
-- Function that populates the summary table and performs the required transformation that was identified
-- in Part A4. The transformation is a concatenation of first_name and last_name into one field,
customer_name.
DROP FUNCTION IF EXISTS populate_summary_table()
CREATE FUNCTION populate_summary_table()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN

DELETE FROM summary_table;

INSERT INTO summary_table(
customer_id,
customer_name,
rental_count)
SELECT
customer_id,
CONCAT(first_name,' ', last_name) AS customer_name,
COUNT(customer_id) AS rental_count
FROM detailed_table
GROUP BY customer_id, customer_name
ORDER BY rental_count DESC
LIMIT 10;
RETURN NEW;
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.

----------Section E starts here----------
```
-- Trigger that calls the populate_summary_table() function and updates/populates the
-- summary table whenever data is added to the detailed table.
CREATE TRIGGER summary_update
AFTER INSERT ON detailed_table
FOR EACH STATEMENT
EXECUTE PROCEDURE populate_summary_table();
```

Tyler Meester
D191 - Advanced Data Management

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.

1. **This procedure should be run everyday at the end of the day to maintain data freshness. This can be setup by using an external scheduler such as pgAgent. To setup a scheduled procedure using pgAgent the user will first need to install pgAgent. Then, in pgAdmin, the user can create a pgAgent job. During the pgAgent job creation the user can customize the details of the job by calling the stored procedure in the code tab of the steps section, setting the schedule in the schedules section (this is where the user decides when the procedure will be called), giving the job a name, etc.**

```
----------Section F starts here----------
-- This procedure refreshes the data in both the detailed table and the summary table.
-- The contents of the tables are cleared and then the procedure performs the ETL process from
-- Part C to upload the updated data into the tables. This procedure should be run everyday at the end of
-- the day to maintain data freshness.

DROP PROCEDURE update_tables()
CREATE PROCEDURE update_tables()
LANGUAGE plpgsql
AS $$
BEGIN

-- Clearing and updating detailed table
DELETE FROM detailed_table;
INSERT INTO detailed_table(
rental_id,
customer_id,
first_name,
last_name)

SELECT
rental.rental_id,
customer.customer_id,
customer.first_name,
customer.last_name
FROM rental
INNER JOIN customer on customer.customer_id = rental.customer_id

-- Clearing and updating the summary table
DELETE FROM summary_table;
INSERT INTO summary_table(
customer_id,
customer_name,
rental_count)
SELECT
customer_id,
CONCAT(first_name,' ', last_name) AS customer_name,
COUNT(customer_id) AS rental_count
FROM detailed_table
```

```
GROUP BY customer_id, customer_name
ORDER BY rental_count DESC
LIMIT 10;

END;
$$
```

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.

See assignment submission

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.

No sources outside of the provided materials were required for the completion of this report.

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

No sources outside of the provided materials were required for the completion of this report.

Tyler Meester
D191 - Advanced Data Management

Full Collection of SQL Querries

```
---------- Section B starts here----------
--Creating the detailed table
DROP TABLE IF EXISTS detailed_table;
CREATE TABLE detailed_table (
rental_id int PRIMARY KEY,
customer_id int,
first_name varchar(255),
last_name varchar(255)
);

--Validating the detailed_table
SELECT * FROM detailed_table;

--Creating the summary table
DROP TABLE IF EXISTS summary_table;
CREATE TABLE summary_table (
customer_id int,
customer_name varchar(255),
rental_count int
);

--Validating the summary_table
SELECT * FROM summary_table;


----------Section C starts here----------
-- Extracts the raw data needed from the customer and rental tables and inserts it into the
-- detailed table.
INSERT INTO detailed_table (
rental_id,
customer_id,
first_name,
last_name)

SELECT rental.rental_id,
customer.customer_id,
customer.first_name,
customer.last_name
FROM rental
INNER JOIN customer on customer.customer_id = rental.customer_id;
-- Verifies detailed table data accuracy
SELECT *
FROM detailed_table;
```

Tyler Meester
D191 - Advanced Data Management

----------Section D starts here----------
-- Function that populates the summary table and performs the required transformation that was identified
-- in Part A4. The transformation is a concatenation of first_name and last_name into one field,
customer_name.
DROP FUNCTION IF EXISTS populate_summary_table()
CREATE FUNCTION populate_summary_table()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN

DELETE FROM summary_table;

INSERT INTO summary_table(
customer_id,
customer_name,
rental_count)
SELECT
customer_id,
CONCAT(first_name,' ', last_name) AS customer_name,
COUNT(customer_id) AS rental_count
FROM detailed_table
GROUP BY customer_id, customer_name
ORDER BY rental_count DESC
LIMIT 10;
RETURN NEW;
END;
$$

----------Section E starts here----------
-- Trigger that calls the populate_summary_table() function and updates/populates the
-- summary table whenever data is added to the detailed table.
CREATE TRIGGER summary_update
AFTER INSERT ON detailed_table
FOR EACH STATEMENT
EXECUTE PROCEDURE populate_summary_table();

----------Section F starts here----------
-- This procedure refreshes the data in both the detailed table and the summary table. The contents of the
-- tables are cleared and then the procedure performs the ETL process from Part C to upload the updated data
-- into the tables. This procedure should be run everyday at the end of the day to maintain data freshness.

DROP PROCEDURE update_tables()
CREATE PROCEDURE update_tables()
LANGUAGE plpgsql
AS $$
BEGIN

-- Clearing and updating detailed table
DELETE FROM detailed_table;

```
INSERT INTO detailed_table(
rental_id,
customer_id,
first_name,
last_name
)

SELECT
rental.rental_id,
customer.customer_id,
customer.first_name,
customer.last_name
FROM rental
INNER JOIN customer on customer.customer_id = rental.customer_id;

-- Clearing and updating the summary table
DELETE FROM summary_table;
INSERT INTO summary_table(
customer_id,
customer_name,
rental_count)
SELECT
customer_id,
CONCAT(first_name,' ', last_name) AS customer_name,
COUNT(customer_id) AS rental_count
FROM detailed_table
GROUP BY customer_id, customer_name
ORDER BY rental_count DESC
LIMIT 10;

END;
$$
```