**INTRODUCTION**

The DVD Rental business would like to define groups of customers by the number of rentals each customer makes per month. The focus of these groups is to identify “Tier A” customers, who each rent **ten or more** DVDs per month. So, “**Can the company create "buckets” for customers, based on rentals per month, and how can that benefit the business?”** e.g., buckets would be grouped by 0 – 9 rentals, 10 – 20 rentals, and more than 20 rentals per month.

**SECTION A –** (Summarize one real-world written business report that can be created from the DVD Dataset from the “Labs on Demand Assessment Environment and DVD Database” attachment)

The business would like to thank customers by creating the following tiered loyalty rewards program. The program works by grouping customers into tiers based on metrics of rentals per month, and providing a reward to be used in the calendar month following the report month:

* 5% off discount for every customer with **at least** **one** rental in the report month (Standard)
* 10% off discount for every customer with **ten** rentals in the report month (Tier A)
* 25% off discount for every customer with **twenty** rentals in the report month (Tier A+)

This program will be communicated in-store and by email. The discount will be applied to the customer’s account for all rentals made during the month in which the reward is active.

Additionally, to entice inactive customers to rent again and drive those customers into return status, an additional reward of a free rental will be available. The reward is provided when a customer with “inactive” status rents and then returns a DVD. This reward differs from the grouping discount for active customers above, as it is a single-use promotion. (Tracking the use of the free-rental promotion and flagging the database is beyond the scope of this initial step in the project. For now, the focus is assigning the reward attribute to the customer account.)

These incentives should generate new interest and keep customers returning every month, with the intent of converting the active customers in the standard group into the “Tier A” group.

**SECTION A1** **–** (Identify the specific fields that will be included in the detailed table and the summary table of the report)

The report will include two sections; **Detailed** and **Summary.** Each section will call for fields from standard tables within the **dvdrental** database, as well as fields derived from a user made, Common Table Expression: **CustomerRentals.** (The tables and expressions are discussed throughout **Section A**)

The **Detailed** section of the report collects the total reward information for every customer, for each month on record, within the database. The **detailed** section will display the following fields:

* **Discount –** A text field displaying one of four options concerning discount information:  
   “25% OFF” | “10% OFF” | “5% OFF” | “Free Return”  
  The values above are assigned with a **case** statement, discussed further in **Section A2**
* **customer\_id** **–** The internally produced customer id, taken from the **customer** table
* **active –** Binary displaying 1 for active, 0 for inactive (assumed), taken from the **customer** table
* **email –** Customer's last known email address, taken from the **customer** table
* **rental\_month –** The ***text*** value of the month in which the rentals are being totaled. This field uses the **TO\_****CHAR()** function, which extracts the text name of the month from the ***timestamp*** value of **rental\_date,** which is taken from the **CustomerRentals** CTE, which originally took the value from the **rental** table
* **rental\_count –** The total rentals made by the customer in the report’s **rental\_month**
* **Tier A Rewards –** A count of the total number of times customers in the **CustomerRentals** CTE, who have **ten or more** rentals in a given month, determined by the **rental\_month** field above

The **summary** section is used to limit the amount of results from the **detailed** section and simplify the readability and display for stakeholders and employees. The **summary** section will be constrained to a single month of results using a **WHERE** clause, discussed further in **SECTION A5.**   
The following fields will be displayed in the **summary** section of the report:

* **“Discount” –** A ***text*** field displaying one of four options concerning discount information:   
  “25% OFF” | “10% OFF” | “5% OFF” | “Free Return”   
  The values above are assigned with a case statement, discussed further in **Section A2**
* **customer\_id –** The internally produced customer id, taken from the **customer** table
* **email –** Customer's last known email address, taken from the **customer** table
* **rental\_month –** The ***text*** value of the month in which the rentals are being totaled. This field uses the **TO\_****CHAR()** function, which extracts the text name of the month from the ***timestamp*** value of **rental\_date**, which is taken from the **CustomerRentals** CTE, which originally took the value from the **rental** table

**SECTION A2 –** (Describe the types of data fields used for the report)

Before the **SELECT** statement can be ran for the data fields in the display tables, a **Common Table Expression** called **CustomerRentals** is created using the following data types:

* **customer\_id – *smallint* –** internally assigned number for tracking customers, taken from the **customer** table
* **rental\_month: TO\_****CHAR(rental\_date, ‘Month’) AS rental\_month – *text* –** This function extracts the month from the **rental\_date** ***timestamp*** field in the **rental** table, and displays it as ***text***
* **rental\_count:** **COUNT(rental\_id) AS rental\_count – *bigint******–*** This aggregate function uses the **GROUP BY** clause with the **customer\_id** and **rental\_month** data fields from the **rental** table, to total the rentals per customer, per month

\*After the CTE **CustomerRentals** **(cr)** is created, the **customer (c)** table is joined by the **customer\_id** field. Disambiguation of the following fields is necessary. See **SECTION A3** for details.

The display data fields in the **Detailed** and **Summary** sections will be of the following types:

* **“Discount” – *text* –** Derived from a case statement based on values of the **cr.rental\_count** and/or **c.active** fields, from the **CustomerRentals** CTE and **customer** tables respectively(See **SECTION A4** for details)
* **cr.customer\_id** **– *smallint –*** Internally assigned number for tracking customers, the display tables use the **‘cr.’** disambiguation
* **c.active – *integer –*** Binary status of customer: 1 for active, 0 for inactive (assumed). The display tables use the **‘c.’** disambiguation
* **c.email – *character varying (50)* -** last known email of the customer, the display tables use the **‘c.’** disambiguation
* **cr.rental\_month – *text –*** This data field is a result of a **TO\_****CHAR()** function used on the ***timestamp*** data field, **rental\_date,** using the keyword argument of **Month.** The display tables use the **‘cr.’** disambiguation
* **cr.rental\_count – *bigint* –** This aggregate function uses the **GROUP BY** clause with the **customer\_id** and **rental\_month** data fields to total the rentals per customer, per month. The display tables use the **‘cr.’** disambiguation
* **“A Tier Rewards”: (SELECT** **COUNT(DISTINCT customer\_id) FROM CustomerRentals WHERE rental\_count > 9) AS "Tier A Rewards" – *bigint –*** This aggregate function is called as a sub-statement using the field **customer\_id** from the **CustomerRentals** Common Table Expression, to total the number of customers with **ten or more** rentals in the report timeframe

**SECTION A3 –** (Identify at least two specific tables from the given dataset that will provide the data necessary for the detailed table section and the summary table section of the report)

The business report uses a **Common Table Expression** that is later joined with a standard table from the provided **‘dvdrental’** database. The **summary** section of the reportuses the same data fields as the **detailed** section, but is adjusted for timeframe and displays fewer fields. All the tables used in the report are listed below without disambiguation:

* **‘CustomerRentals’ –** A Common Table Expression used to first gather aggregate information about the company’s rental statistics, before applying discount attributes
* **‘rental'** table **–** This table provides the rental data for the **CustomerRentals** CTE
* **‘customer’** table **–** This table provides the data fields for the customer and is used in a **join** with the **CustomerRentals** CTE

**SECTION A4 –** (Identify at least one field in the detailed table section that will require a custom transformation with a user-defined function and explain why it should be transformed (e.g., you might translate a field with a value of N to No and Y to Yes))

* The **‘Discount’** field used in the **detailed** and **summary** sections of the report is determined in the following case statement:  
  CASE  
   WHEN cr.rental\_count BETWEEN 9 AND 19 THEN '10% OFF'  
   WHEN cr.rental\_count > 19 THEN '25% OFF'  
   WHEN c.active = 0 THEN 'Free Return'  
   ELSE '5% OFF'  
  END AS "Discount"

This field is transformed into a readable text statement, from either an aggregate **COUNT()** function of the **rental.rental\_id** ***integer*** in three cases, or the **customer.active *integer*** in another case. The transformation is necessary, so the applied discount can be clearly communicated. The result would otherwise be a ***smallint*** value representing the number of rentals in the period per customer. The employee would then have to mentally convert that number to the appropriate discount.

* The **‘rental\_month’** field used in the **detailed** and **summary** sections of the report is extracted and transformed into a ***text*** field, from the month data in the ***timestamp*** field of **rental\_date*.***

This field is transformed into the ***text*** version and readable month instead of using the full data in the ***timestamp*** field. The transformation is necessary to easily identify the month in which the report takes place.

**SECTION A5 –** (Explain the different business uses of the detailed table section and the summary table section of the report)

The full business report includes a **detailed** and **summary** section. Although these sections share the same data points, the two sections do provide unique insight.

The **Detailed** section will allow the stakeholders to view a customer’s rental habits, broken down by month, which can lead to various other business uses. For example, further analytics will detail trends in a customer’s rental behavior and allow a more customized promotion strategy. The main users of the **Detailed** section of the report will be the analytics, planning, and strategy teams, as it provides a longer-term view of a customer’s monthly habits.

The **Summary** section of the report limits the timeframe of the **Detailed** section to a one-month period and restricts the data fields visible to **“Discount****”, customer\_id, email,** and **rental\_month**. This monthly data is used to determine discounts applicable to the customer’s account for the current rental month. The users of this **summary** section will generally be customer facing employees, who will need basic information to provide to the customer.

**SECTION A6 –** (Explain how frequently your report should be refreshed to remain relevant to stakeholders)

This report will have two effective windows of operation, with opportunities for ad-hoc or other scheduled use.

The marketing and customer service team running the monthly rewards program will need the **summary** section of the report to be generated **at 12:01 A.M., on the** **first day of the month, every month.** The report has the functionality to change the requested month automatically, with the following **WHERE clause**:  
**WHERE rental\_date >= CURRENT\_DATE - INTERVAL '1 month' AND rental\_date < CURRENT\_DATE - INTERVAL '1 day'**

The analytics and management teams will need the **detailed** section of the report at the end of each year, end of each quarter, and at any time a request is submitted. The teams will use the larger dataset to strategize, plan and trend the information for revenue growth. The **WHERE clause** mentioned above will not be used in the **detailed** section of the report.

**SECTION B –** (Provide original code for function(s) in text format that perform the transformation(s) you identified in part A4)

The fields identified in **SECTION A4** (**“Discount”** & **rental\_month**), are transformed “on the fly” within the actual reporting procedure. This is done with the combination of a **Common Table Expression** and **INNER JOIN** statement, using a **case** statement and **TO\_CHAR()** function respectively as follows:

-- This query is used to demonstrate the “on the fly” transformation -- of data within the actual reporting stored procedure  
ROLLBACK; -- Assure no transactions in progress  
WITH CustomerRentals AS (  
 SELECT  
 customer\_id,  
 TO\_CHAR(rental\_date, 'Month') AS **rental\_month**,  
 COUNT(rental\_id) AS rental\_count  
 FROM  
 rental  
GROUP BY  
 customer\_id, rental\_month  
)  
SELECT   
 cr.rental\_month,  
 CASE  
 WHEN cr.rental\_count BETWEEN 9 AND 19 THEN '10% OFF'  
 WHEN cr.rental\_count > 19 THEN '25% OFF'  
 WHEN c.active = 0 THEN 'Free Return'  
 ELSE '5% OFF'  
END AS **"Discount"**   
FROM CustomerRentals cr  
INNER JOIN customer c ON c.customer\_id = cr.customer\_id;

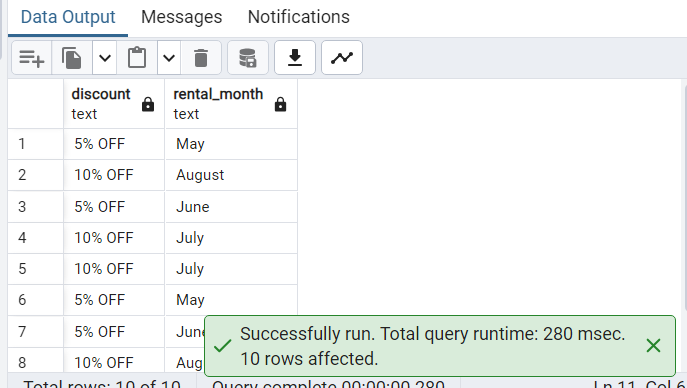
For the purposes of this section, the changes will also be demonstrated within a **User Defined Function** as follows:

-- Drop the function if it exists  
DROP FUNCTION IF EXISTS transform\_data();

-- Create the transform\_data() function  
CREATE OR REPLACE FUNCTION transform\_data()  
RETURNS TABLE (  
 discount text,  
 rental\_month text  
)  
AS $$  
BEGIN  
 RETURN QUERY  
 WITH CustomerRentals AS ( -- CTE made to count rentals/month for each customer  
 SELECT  
 customer\_id,  
 TO\_CHAR(rental\_date, 'Month') AS rental\_month,  
 COUNT(rental\_id) AS rental\_count  
 FROM  
 rental  
 GROUP BY  
 customer\_id, rental\_month  
 )  
 SELECT  
 CASE -- This allows the results of the CTE to be compared for the appropriate discount  
 WHEN cr.rental\_count BETWEEN 9 AND 19 THEN '10% OFF'  
 WHEN cr.rental\_count > 19 THEN '25% OFF'  
 WHEN c.active = 0 THEN 'Free Return'  
 ELSE '5% OFF'  
 END AS discount,   
 cr.rental\_month  
 FROM  
 CustomerRentals cr  
 INNER JOIN customer c ON cr.customer\_id = c.customer\_id  
 ORDER BY c.customer\_id  
 LIMIT 10; -- Limit the results for display purposes  
END;  
$$ LANGUAGE plpgsql;

-- Call the function and display the results  
SELECT \* FROM transform\_data();

The results of the **user defined function, transform\_data():**  
\*note the results of the transformation function have been limited to 10 rows for demonstration purposes



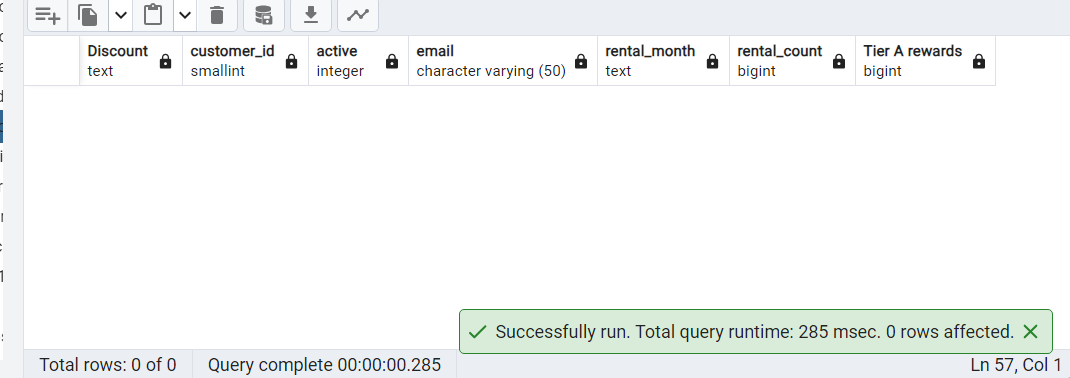
**SECTION C –** (Provide original SQL code in a text format that creates the detailed and summary tables to hold your report table sections)

The following code will create two empty tables for the **detailed** and **summary** sections of the report. The tables will be filled with data retrieved from stored procedure(s) found in **Sections D, E,** and **F** of this document:

SQL used to create the **detailed** section/table:  
-- This function creates the "Detailed" section of the business report -- and calls the empty table to display the structure

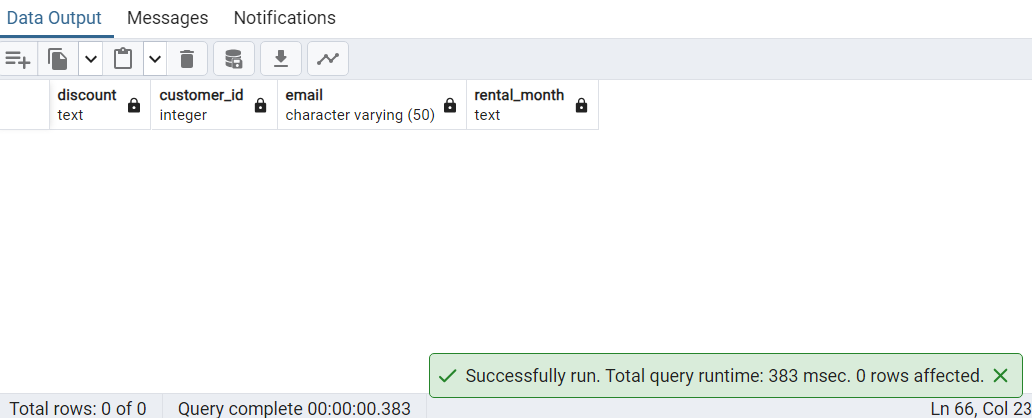
DROP TABLE IF EXISTS detailed;  
CREATE TABLE detailed (  
 "Discount" text,  
 customer\_id smallint,  
 active integer,  
 email character varying (50),  
 rental\_month text,  
 rental\_count bigint,  
 "Tier A rewards" bigint  
);

-- Call the empty table for display of structure  
SELECT \* FROM detailed;

Screenshot of empty **detailed** table:  


SQL used to create the **summary** section/table:  
-- This function creates the "summary" section of the business report -- and calls the empty table  
DROP TABLE IF EXISTS summary;   
CREATE TABLE summary (  
 Discount text,  
 customer\_id integer,  
 email character varying (50),  
 rental\_month text  
);

-- Call the empty table for display of structure  
SELECT \* FROM summary;

Screenshot of empty **summary** table:  


**SECTION D –** (Provide an original SQL query in a text format that will extract the raw data needed for the detailed section of your report from the source database)

The following code will create the **detailed** section of the report in the first function, and then extract the raw data from the **dvdrental** database, transforming some fields, insert the data into the **detailed** table, and call the **detailed** table to display the successful and expected results:

ROLLBACK; -- To ensure no running queries

-- This program will create and fill the "detailed" section of the report

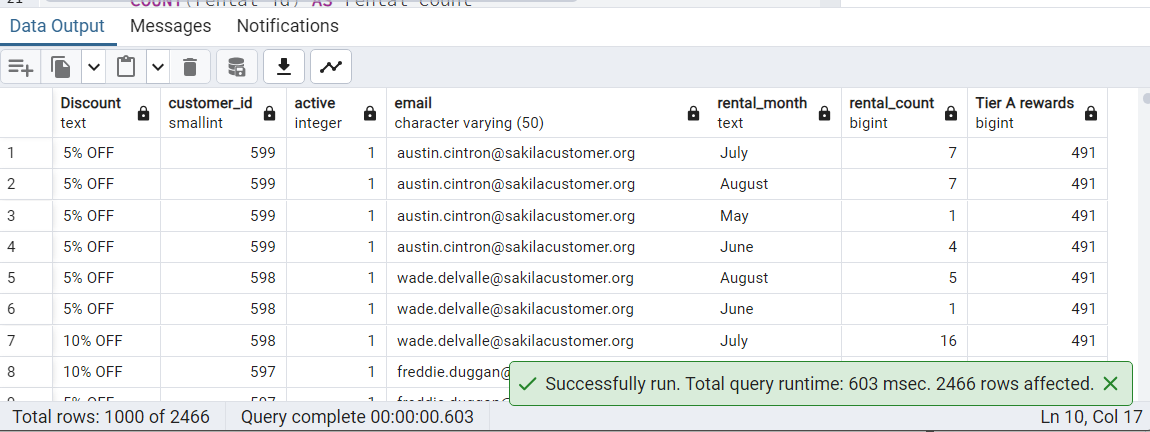
-- This function creates the "detailed" section of the business report  
DROP TABLE IF EXISTS detailed;  
CREATE TABLE detailed (  
 "Discount" text,  
 customer\_id smallint,  
 active integer,  
 email character varying (50),  
 rental\_month text,  
 rental\_count bigint,  
 "Tier A rewards" bigint  
);

WITH CustomerRentals AS (  
 SELECT  
 customer\_id,  
 TO\_CHAR(rental\_date, 'Month') AS rental\_month,  
 COUNT(rental\_id) AS rental\_count   
 FROM  
 rental  
 GROUP BY  
 customer\_id, rental\_month  
)

INSERT INTO detailed (  
 "Discount",  
 customer\_id,  
 active,  
 email,  
 rental\_month,  
 rental\_count,  
 "Tier A rewards"  
)  
SELECT  
 CASE   
 WHEN cr.rental\_count BETWEEN 9 AND 19 THEN '10% OFF'  
 WHEN cr.rental\_count > 19 THEN '25% OFF'  
 WHEN c.active = 0 THEN 'Free Return'  
 ELSE '5% OFF'  
 END AS "Discount",  
cr.customer\_id,  
c.active,  
c.email,  
cr.rental\_month,  
cr.rental\_count,  
(SELECT COUNT(DISTINCT customer\_id) FROM CustomerRentals WHERE rental\_count > 9) AS "Tier A Rewards"  
FROM  
 CustomerRentals cr  
INNER JOIN customer c ON cr.customer\_id = c.customer\_id  
ORDER BY 2 desc;

-- Call the table to show results, verify data

SELECT \* FROM detailed;

Screenshot of **detailed** table with data retrieved from the provided query:   


**SECTION E** –(Provide original SQL code in a text format 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)

The function called **update\_summary()** uses an **INSERT SELECT** statement to fill the **summary** table any time the **detailed** table receives an **INSERT** action. For production purposes, the conditional where clause: **WHERE rental\_month LIKE '%' || TO\_CHAR(CURRENT\_DATE - INTERVAL '1 month', 'Month') || '%';** could be used to base the report on the prior calendar month to today’s date.

This code creates the function and trigger used to fill the summary table:  
-- Drop any existing trigger or function for this section  
DROP TRIGGER IF EXISTS update\_summary ON detailed;  
DROP FUNCTION IF EXISTS update\_summary();

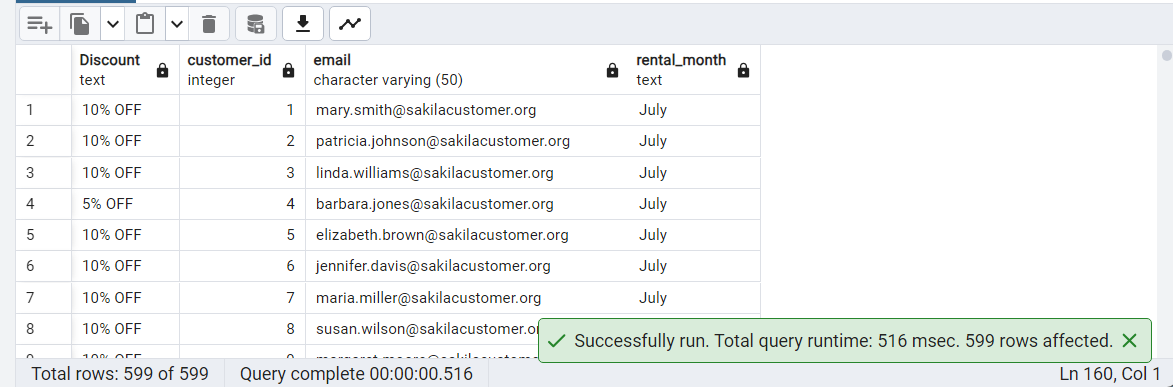
-- The update\_summary() function used to fill the summary table  
CREATE OR REPLACE FUNCTION update\_summary()  
 RETURNS TRIGGER  
 LANGUAGE PLPGSQL  
AS $$  
BEGIN  
 DELETE FROM summary; -- Clears out summary table for refresh  
 INSERT INTO summary (  
 "Discount",  
 customer\_id,  
 email,  
 rental\_month  
 )  
 -- Pull the data from the "detailed" section of the report.  
 SELECT "Discount", customer\_id, email, rental\_month FROM detailed  
 WHERE rental\_month LIKE '%July%'; -- (EXAMPLE ONLY)  
 -- WHERE rental\_month LIKE '%' || TO\_CHAR(CURRENT\_DATE - INTERVAL -- '1 month', 'Month') || '%';  
 RETURN NULL;  
END;  
$$;

-- The update\_summary TRIGGER that will be used any time the detailed table has an INSERT performed  
CREATE TRIGGER update\_summary  
AFTER INSERT ON detailed  
FOR EACH STATEMENT  
EXECUTE FUNCTION update\_summary();

The trigger fires with the following test **INSERT INTO detailed** statement:  
-- Use Insert statement to trigger update\_summary TRIGGER and   
-- FUNCTION

INSERT INTO detailed (  
 "Discount",  
 customer\_id,  
 active,  
 email,  
 rental\_month,  
 rental\_count,  
 "Tier A rewards"  
)  
VALUES  
(NULL, NULL, NULL, NULL, NULL, NULL, NULL);

--Display results AND VERIFY DATA  
SELECT\* FROM summary  
ORDER BY customer\_id asc;

Screenshot of **summary** table with accurate month:  


**SECTION F –** (Provide an original stored procedure in a text format that can be used to refresh the data in *both* the detailed table and summary table. The procedure should clear the contents of the detailed table and summary table and perform the raw data extraction from part D)

The following is a stored procedure that can update the **detailed** table of the report with fresh data, which will trigger the **update\_summary** trigger:

-- \*\*\*\*\*\*\*\*\*\* FULL STORED PROCEDURE FOR REPORT \*\*\*\*\*\*\*\*\*\*

CREATE OR REPLACE PROCEDURE fill\_report()  
LANGUAGE plpgsql  
AS $$  
BEGIN

-- \*\*\*\*\*\*\*\*\*\* CREATE EMPTY TABLES (DROP IF THEY EXIST ALREADY)\*\*\*\*\*\*\*\*\*\*

DROP TABLE IF EXISTS detailed; -- DETAILED  
CREATE TABLE detailed (  
 "Discount" text,  
 customer\_id smallint,  
 active integer,  
 email character varying (50),  
 rental\_month text,  
 rental\_count bigint,  
 "Tier A rewards" bigint  
);  
DROP TABLE IF EXISTS summary; -- SUMMARY  
CREATE TABLE summary (  
 "Discount" text,  
 customer\_id integer,  
 email character varying (50),  
 rental\_month text  
);

-- \*\*\*\*\*\*\*\*\*\* CREATE TRIGGER AND FUNCTION FOR REPORT DATA \*\*\*\*\*\*\*\*\*\*

DROP TRIGGER IF EXISTS update\_summary ON detailed;  
DROP FUNCTION IF EXISTS update\_summary();

-- update\_summary() function used to fill the summary table

CREATE OR REPLACE FUNCTION update\_summary()  
 RETURNS TRIGGER  
 LANGUAGE PLPGSQL  
AS $inner$  
BEGIN  
 DELETE FROM summary; -- Clears out summary table for refresh  
 INSERT INTO summary (  
 "Discount",  
 customer\_id,  
 email,  
 rental\_month  
 )  
 -- Pull the data from the "detailed" section of the report.  
 SELECT "Discount", customer\_id, email, rental\_month FROM  
detailed  
 WHERE rental\_month LIKE '%July%'; -- (EXAMPLE ONLY)  
 -- WHERE rental\_month LIKE '%' || TO\_CHAR(CURRENT\_DATE – INTERVAL  
'1 month', 'Month') || '%';  
 RETURN NULL;  
END;  
$inner$;  
   
-- update\_summary TRIGGER fired with INSERT on detailed table  
CREATE TRIGGER update\_summary  
AFTER INSERT ON detailed  
FOR EACH STATEMENT  
EXECUTE FUNCTION update\_summary();

-- \*\*\*\*\*\*\*\*\*\* FILL DETAILED TABLE WITH RAW DATA \*\*\*\*\*\*\*\*\*\*

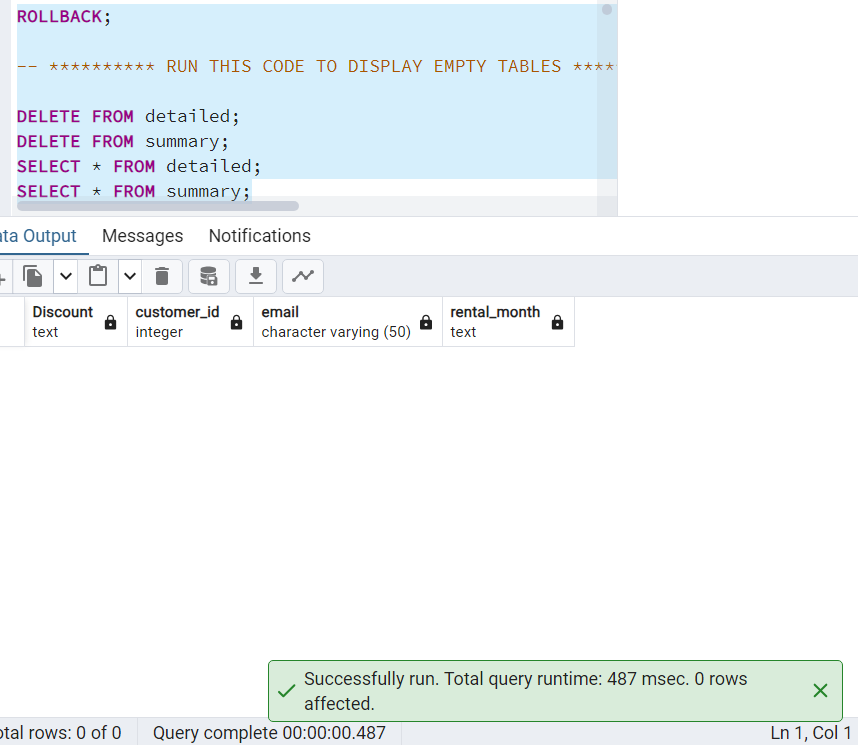
DELETE FROM detailed; -- Refresh the data and removes anything that has been erased from 'dvdrental' database

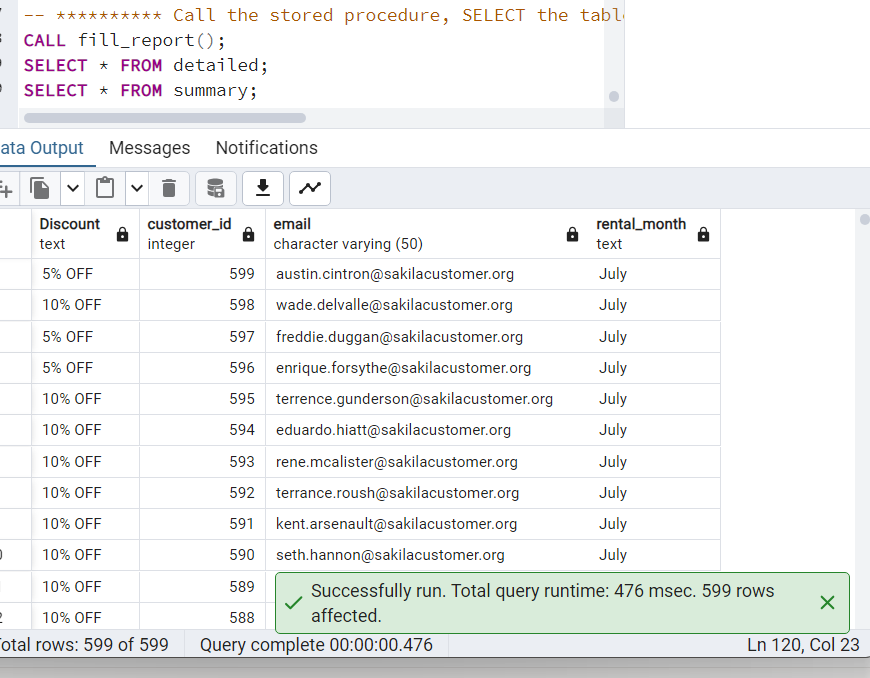
-- Create Common Table Expression to gather rental statistics for Rewards program

WITH CustomerRentals AS (  
 SELECT  
 customer\_id,  
 TO\_CHAR(rental\_date, 'Month') AS rental\_month,  
 COUNT(rental\_id) AS rental\_count  
 FROM  
 rental  
 GROUP BY  
 customer\_id, rental\_month  
)  
  
INSERT INTO detailed (  
 "Discount",  
 customer\_id,  
 active,  
 email,  
 rental\_month,  
 rental\_count,  
 "Tier A rewards"  
)  
SELECT  
 CASE  
 WHEN cr.rental\_count BETWEEN 9 AND 19 THEN '10% OFF'  
 WHEN cr.rental\_count > 19 THEN '25% OFF'  
 WHEN c.active = 0 THEN 'Free Return'  
 ELSE '5% OFF'  
 END AS "Discount",  
 cr.customer\_id,  
 c.active,  
 c.email,  
 cr.rental\_month,  
 cr.rental\_count,  
 -- The sub-statement below filters the rental\_count field for a  
number 10 or higher  
 (SELECT COUNT(DISTINCT customer\_id) FROM CustomerRentals WHERE rental\_count > 9) AS "Tier A Rewards"  
FROM  
 CustomerRentals cr  
 INNER JOIN customer c ON cr.customer\_id = c.customer\_id  
 -- WHERE c.active = 0 -- (Filters for active patients)  
ORDER BY  
 2 desc;  
END;  
$$;

-- \*\*\*\*\*\*\*\*\*\* Call the stored procedure, SELECT the tables and verify the data

CALL fill\_report();

Screenshot of empty tables prior to stored procedure call:  


Screenshot of successful stored procedure call:  


**SECTION F1 –** (Identify a relevant job scheduling tool that can be used to automate the stored procedure)

Postgresql requires an external tool in order to automate a scheduled test. This stored procedure should be called on the first day of the month, every month, to collect statistics from the prior month and reward the customers accordingly.

I used **pgAdmin 4** to create this report, so the adimn of this company should do the same and install the scheduling agent called pgAgent. The agent’s configuration is stored in the postgres database and will run as a daemon, calling the **fill\_report()** stored procedure at 12:01 A.M. on the first of each month.

**SECTION G –** (Provide a Panopto video recording that includes the presenter and a vocalized demonstration of the functionality of the code used for the analysis)

Please see the Panopto URL: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=922e2455-a43f-482d-8e1f-b0eb00671aa7>

**SECTION H –** (Acknowledge all utilized sources, including any sources of third-party code, using in-text citations and references. If no sources are used, clearly declare that no sources were used to support your submission.)

No outside sources were cited in this document or used in the creation of any code.