Business Report for “The Best Ever DVD Store” Summary

**SECTION A: BUSINESS REPORT**

At “Best Ever DVD Store” the primary goal is finding out who we can send promotions to **every 3 months** if they are one of the **top 100 customers** who have rented from us. It is also to encourage the other customers to purchase more rentals resulting in more profit for the company. **Every 3 months** the database will update the customers’ **total number of rentals** they have and for a more detailed amount, how much they have each spent on rentals for future analysis. So, the question we want to answer is: **What are the names and emails of the top 100 DVD renters & how much did we profit from them each?** Answering this question will also help in examining the popularity of the store and how many people will utilize the promotions & special discounts.

**SECTION A1 SPECIFIC FIELDS:**

**Detailed Table**

**Table name:**

**customer\_details**

1. **customer\_id int \*** Unique to one customer
2. **first\_name varchar(50) \*** Customers first name
3. **last\_name varchar(50) \*** Customers last name
4. **email varchar(50) \*** Customers email address
5. **payment\_date timestamp \*** Unique date and time of payment
6. **amount numeric(5,2) \*** Amount paid by customer

**Summary Table**

**Table name:**

**summary**

1. **customer\_name varchar(95) \*** Full name of customer
2. **email varchar(90) \*** Email address of customer
3. **total\_rentals\_bought int \*** Total rentals customer bought
4. **total\_spent** **numeric(5,2) \*** Total amount of money customer spent

**SECTION A2-A5:**

I use the **customer** and **payment** tables **from the provided DVD Rental database**. I cross reference the two by using the **Customer\_id** that each of the two tables contain. Using the two tables we create our **summary** and **customer\_details table**. At this point we can insert all our data listed in **A1** into the newly created tables.

After evaluating our tables, we know that there is a **unique time and date** for each purchase, we can usethe **Summary** **table** to **create a function** to calculate the **total number of rentals** using **customer\_id** to reveal the **number of rentals** bought by that unique person to see which customers brings in the most revenue.

Alternatively, using the **customer\_details** **table** shareholders can access the **payment\_id** and **amount** they made from each of the customer’s purchases. This can also determine the amount of money each customer spends on rentals

Creating a trigger function **refreshing\_summary()** can update the **summary table** **every 3 months**. The reason to update on a **3-month** interval is to avoid the risk of spamming customers. Spamming could push away customers by reducing sales, so it is best to avoid sending out too many promotions every month to prevent potential loss of profit.

Creating a trigger **refreshing\_summary** after insert on the **detailed table** will continually update the **summary table** as data is added to the **customer\_details** table.

**SECTION B Function for Transformation:**

CREATE OR REPLACE FUNCTION refreshing\_summary()

RETURNS TRIGGER

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM summary ;

INSERT INTO summary (

SELECT Concat(first\_name,' , ',last\_name) AS customer\_name, email, COUNT(customer\_id), SUM(amount)

FROM customer\_details

GROUP BY customer\_name,email

ORDER BY Count(customer\_id) DESC

LIMIT 100

);

RETURN NEW ;

END;

**SECTION C DETAILED AND SUMMARY TABLES:**

DROP TABLE IF EXISTS customer\_details;

CREATE TABLE customer\_details (

customer\_id int

,first\_name varchar(50)

,last\_name varchar(50)

,email varchar(50)

,payment\_date timestamp

,payment\_id int

,amount numeric(5,2)

);

DROP TABLE IF EXISTS summary;

CREATE TABLE summary (

customer\_name varchar(95)

,email varchar(90)

,total\_rentals\_bought integer

,total\_spent numeric(5,2)

);

**SECTION D SQL QUERY:**

INSERT INTO customer\_details (customer\_id,first\_name,last\_name, email, payment\_date, payment\_id,amount)

SELECT cust.customer\_id,cust.first\_name,cust.last\_name,cust.email,pay.payment\_date,pay.payment\_id,pay.amount

FROM customer cust

INNER JOIN payment pay ON cust.customer\_id = pay.customer\_id;

**SECTION E TRIGGERS:**

CREATE TRIGGER refreshing\_summary

AFTER INSERT ON customer\_details

FOR EACH STATEMENT

EXECUTE PROCEDURE refreshing\_summary();

**SECTION F STORED PROCEDURES:**

CREATE OR REPLACE PROCEDURE table\_refresh()

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM customer\_details; --empties the customer\_details

-- Re-inserts new data to customers details table --

INSERT INTO customer\_details (

customer\_id

,first\_name

,last\_name

,email

,payment\_date

,payment\_id

,amount )

SELECT cust.customer\_id,cust.first\_name,cust.last\_name,cust.email,pay.payment\_date,pay.payment\_id,pay.amount

FROM customer cust

INNER JOIN payment pay ON cust.customer\_id = pay.customer\_id;

END;

$$

CALL table\_refresh(); -- to call the stored procedure

**SECTION F1 JOB SCHEDULING TOOL:**

USING PgAgent scheduling tool

The stored procedure is run every 3 months, running it in that interval ensures the data is accurate as new rentals are being bought before the upcoming promotion.

**Testing out our work**

--SELECT \* FROM customer\_details;

--SELECT first\_name,last\_name, Count(customer\_id), sum(amount)

--FROM customer\_details

--Group by first\_name, last\_name

--Order By Count(customer\_id) DESC

--LIMIT 100;

--SELECT \* FROM summary;

--INSERT INTO customer\_details (customer\_id,first\_name,last\_name,email,payment\_date,payment\_id,amount)

--VALUES (341,'Peter','Menard','peter.menard@sakilacustomer.org','2023-07-16 22:25:46.996577',32099,5.99);

**SECTION G PANOPTO VIDEO:**

**VIDEO LINK:** [Advanced Data Management D191 | D326 (Student Creators) [assignments] Wednesday, July 19, 2023 at 9:21:25 PM (panopto.com)](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=82bce705-098c-497e-9798-b045002c8fcc)

**SECTION H:**

**No sources were used.**