**SECTION A:**

This project is about answering a proposed business question for a DVD rental business. The business question I chose is “How many DVD rentals have been sold by each employee?”. The answer to this question could help the DVD rental business with keeping track of which employees are selling the most, and which are selling the least. Having this information will help the business potentially reward employees who are doing a good job of selling, thus making them feel appreciated and increasing employee satisfaction, along with identifying which employees may need more help or guidance to help them sell more DVD rentals.

**A.1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable Name** | **Destination Table** | **Source Table** | **Datatype** | **Brief description** |
| rental\_id | detailed\_report | rental | INT | The ID of the dvd rental |
| staff\_id | detailed\_report & summary\_report | staff | INT | The ID of the employee |
| first\_name | detailed\_report & summary\_report | staff | VARCHAR(45) | The first name of the employee |
| last\_name | detailed\_report & summary\_report | staff | VARCHAR(45) | The last name of the employee |
| store\_id | detailed\_report | staff | SMALLINT | The ID of which dvd rental store the employee works at |

**A.2:**

The two specific tables providing the data necessary to answer the business question are the rental and staff data tables.

**A.3:**

The specific fields for the detailed\_report table are rental\_id, staff\_id, first\_name, last\_name, and store\_id.

The specific fields for the summary\_report table are staff\_id, full\_name, and rental\_total.

**A.4:**

One field in the detailed\_report table that will need a custom transformation is the staff\_id, which will be turned into a count of the total rentals sold by each employee. In addition, the first\_name and last\_name fields will also be transformed into a single field called full\_name for ease of reading.

**A.5:**

The business use of the detailed\_report table would be used when a stakeholder needs to access a more detailed report regarding a specific rental sale made by an employee. The summary\_report table would be used to very quickly identify the employee with the most and the employee with the fewest rental sales.

**A.6:**

The reports should be refreshed weekly in order to accurately keep track of employee rental sales for each work week.

**SECTION B:**

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

-- Creating the detailed report

DROP TABLE IF EXISTS detailed\_report;

CREATE TABLE detailed\_report(

rental\_id INT PRIMARY KEY,

staff\_id INT,

first\_name VARCHAR(45),

last\_name VARCHAR(45),

store\_id SMALLINT

);

-- Creating the summary report

DROP TABLE IF EXISTS summary\_report;

CREATE TABLE summary\_report(

staff\_id INT PRIMARY KEY,

full\_name VARCHAR(90),

rental\_total SMALLINT

);

**SECTION C:**

-- Section 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.

-- Extracting the data from the source database (rental and staff tables) to the detailed\_report table

INSERT INTO detailed\_report(rental\_id, staff\_id, first\_name, last\_name, store\_id)

SELECT r.rental\_id, s.staff\_id, s.first\_name, s.last\_name, s.store\_id

FROM rental AS r

INNER JOIN staff AS s ON s.staff\_id = r.staff\_id;

-- Verifying accuracy of the data from the detailed\_report table

SELECT \* FROM detailed\_report;

**SECTION D:**

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

-- Populating the summary\_report table and  transforming first\_name and last\_name into full\_name, and staff\_id into a count called rental\_total

INSERT INTO summary\_report(

SELECT staff\_id,

CONCAT(first\_name, ' ', last\_name) AS full\_name,

COUNT(staff\_id) AS rental\_total

FROM detailed\_report

GROUP BY staff\_id, first\_name, last\_name

ORDER BY rental\_total DESC);

**SECTION E:**

-- Section 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.

-- Creating a trigger function for what should happen, which would be deleting what’s in the summary table and then inserting new data back into it.

DROP FUNCTION IF EXISTS summary\_update();

CREATE FUNCTION summary\_update()

RETURNS TRIGGER AS $$

BEGIN

DELETE FROM summary\_report;

INSERT INTO summary\_report(

SELECT staff\_id,

CONCAT(first\_name, ' ', last\_name) AS full\_name,

COUNT(staff\_id) AS rental\_total

FROM detailed\_report

GROUP BY staff\_id, first\_name, last\_name

ORDER BY rental\_total DESC

);

RETURN NEW;

END; $$

LANGUAGE PLPGSQL;

-- Creating the actual trigger to call the function when the detailed\_report table updates on any row.

CREATE TRIGGER summary\_update\_trigger AFTER INSERT ON detailed\_report

FOR EACH STATEMENT EXECUTE PROCEDURE summary\_update();

**SECTION F:**

– Section 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.

CREATE OR REPLACE PROCEDURE summary\_detailed\_refresh()

LANGUAGE PLPGSQL

AS $$

BEGIN

DELETE FROM detailed\_report;

INSERT INTO detailed\_report(rental\_id, staff\_id, first\_name, last\_name, store\_id)

SELECT r.rental\_id, s.staff\_id, s.first\_name, s.last\_name, s.store\_id

FROM rental AS r

INNER JOIN staff AS s ON s.staff\_id = r.staff\_id;

DELETE FROM summary\_report;

INSERT INTO summary\_report(

SELECT staff\_id,

CONCAT(first\_name, ' ', last\_name) AS full\_name,

COUNT(staff\_id) AS rental\_total

FROM detailed\_report

GROUP BY staff\_id, first\_name, last\_name

ORDER BY rental\_total DESC);

END; $$

**F.1:**

This stored procedure should be executed on a weekly basis in order to guarantee the reports are up to date and accurate each work week for the stakeholders to look at. In order to schedule a stored procedure to be run on a schedule, I would use a job scheduling tool for PostgreSQL, such as pgAgent.

**SECTION H:**

The following web sources were used to help create code in this application:

<https://www.postgresql.org/docs/current/plpgsql-trigger.html>

<https://www.postgresql.org/docs/current/sql-createprocedure.html>

**SECTION I:**

*Create procedure*. PostgreSQL Documentation. (2022, November 10). Retrieved January 5, 2023, from https://www.postgresql.org/docs/current/sql-createprocedure.html

*Trigger functions*. PostgreSQL Documentation. (2022, November 10). Retrieved January 5, 2023, from https://www.postgresql.org/docs/current/plpgsql-trigger.html