VDM1 Task 1: Automating Data Integration

Daniel Roberts

Student ID: #009198202

***A: Summarize***

1. This report is generated from the database “dvdrentals”. This report is focused on showing the top staff members and their total sales. This report will also show the store with the most total sales. This data will help the company determine which store and staff is performing at a higher rate. The Detailed Report will show each staff member’s ID, store, first and last names, email address, whether they are still employed with the store and show their total sales amount to date. The Detailed Report can be used by the company to provide rewards for staff and store based off their sale numbers. The Summary Report will focus primarily on each store. It will show the store’s ID, address, and total sales numbers. The company can utilize that report to see which store and location is generating the most revenue.
2. The tables needed for the report are:

* Staff
* Payment
* Store
* Address

1. The fields we will need:

|  |  |  |  |
| --- | --- | --- | --- |
| **Staff** | **Payment** | **Store** | **Address** |
| first\_name | staff\_id | manager\_staff\_id | address\_id |
| last\_name | amount | staff\_id | address |
| email | payment\_id | address\_id |  |
| active |  | store\_id |  |
| sales\_amount |  |  |  |

1. I am going to change the active field from “True” or “False” to “YES” or “NO” to make it more user-friendly. I will be using a CASE statement to make that transformation.
2. The Detailed Report will give the following data:

* staff\_id, first\_name, last\_name, email, sales\_amount and active

The Detailed Report will give the company an overall picture of which store and staff member is performing at a higher rate. It will also give them the first and last name of the staff member, their email address, their total sales and whether that staff member is still active or employed.

The Summary Report will give the following data:

* store\_id, address and sales\_amount

The Summary Report will focus on showing which store is outperforming the other. This report will provide the store\_id, the store address and the store’s total sales amount.

1. The report should be refreshed quarterly or every 3-4 months. This will give the company a frequent picture to see which store is performing at a higher rate, while also allowing the stores ample time to improve their sales numbers.

**B. SQL code that creates the table to hold the detail report:**

DROP TABLE IF EXISTS Detailed\_Report;

CREATE TABLE Detailed\_Report (

staff\_id integer,

store\_id integer,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100),

active VARCHAR(3),

sales\_amount DECIMAL

);

**SQL code that creates the table to hold the summary report:**

DROP TABLE IF EXISTS Summary\_Report;

CREATE TABLE Summary\_Report (

store\_id integer,

store\_address VARCHAR(100),

sales\_amount DECIMAL

);

**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.**

INSERT INTO Detailed\_Report (

SELECT STORE.store\_id,

STAFF.staff\_id,

first\_name,

last\_name,

email AS email\_address,

CASE active WHEN ‘True’ THEN ‘YES’ WHEN ‘False’ THEN ‘NO’ END,

SUM(amount) AS sales\_amount

FROM STAFF

INNER JOIN STORE

ON STAFF.store\_id = STORE.store\_id

INNER JOIN ADDRESS

ON STORE.address\_id = ADDRESS.address\_id

INNER JOIN PAYMENT

ON PAYMENT.staff\_id = STAFF.staff\_id

GROUP BY (STAFF.staff\_id, STORE.store\_id, ADDRESS.address\_id)

ORDER BY sales\_amount DESC

);

**QUERY:** SELECT \*

FROM Detailed\_Report;

**QUERY TO VERIFY ACCURACY:** SELECT SUM(amount)

FROM PAYMENT

WHERE staff\_id = 2;

**Code to insert data into Summary\_Report:**

INSERT INTO Summary\_Report (

SELECT

STORE.store\_id,

ADDRESS.address,

SUM(amount) AS sales\_amount

FROM STORE

INNER JOIN STAFF

ON STORE.store\_id = STAFF.store\_id

INNER JOIN ADDRESS

ON STORE.address\_id = ADDRESS.address\_id

INNER JOIN PAYMENT

ON PAYMENT.staff\_id = STAFF.staff\_id

GROUP BY (STAFF.staff\_id, STORE.store\_id, ADDRESS.address\_id)

ORDER BY sales\_amount DESC

);

**QUERY:** SELECT \*

FROM Summary\_Report;

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

- My transformation code was enacted within my “INSERT INTO” statement

**Code:**

* CASE active WHEN ‘True’ THEN ‘YES’ WHEN ‘False’ THEN ‘NO’ 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.**

**Code:**

CREATE OR REPLACE FUNCTION update\_summary\_report()

RETURNS TRIGGER

LANGUAGE PLPGSQL

AS  
$$

BEGIN

IF NEW.sales\_amount<>OLD.sales\_amount THEN

UPDATE Summary\_Report

SET sales\_amount = (SELECT SUM(sales\_amount)

FROM Detailed\_Report

WHERE store\_id = OLD.store\_id)

WHERE store\_id = OLD.store\_id;

END IF;

RETURN NEW;

END;

$$;

CREATE TRIGGER after\_sale\_trigger

AFTER UPDATE ON Detailed\_Report

FOR EACH ROW EXECUTE PROCEDURE update\_summary\_report();

**Query to update Detailed\_Report:**

UPDATE Detailed\_Report

SET sales\_amount = 9999

WHERE staff\_id = 2;

**Query to check accuracy on Summary\_Report:**

SELECT \*

FROM Summary\_Report;

**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.**

**Code:**

CREATE PROCEDURE refresh\_data()

AS $$

BEGIN

DELETE

FROM Detailed\_Report;

INSERT INTO Detailed\_Report (

SELECT

STORE.store\_id,

STAFF.staff\_id,

first\_name,

last\_name,

email AS email\_address,

CASE active WHEN ‘True’ THEN ‘YES’ WHEN ‘False’ THEN ‘NO’ END,

SUM(amount) AS sales\_amount

FROM STAFF

INNER JOIN STORE

ON STAFF.store\_id = STORE.store\_id

INNER JOIN ADDRESS

ON STORE.address\_id = ADDRESS.address\_id

INNER JOIN PAYMENT

ON PAYMENT.staff\_id = STAFF.staff\_id

GROUP BY (STAFF.staff\_id, STORE.store\_id, ADDRESS.address\_id)

ORDER BY sales\_amount DESC

);

DELETE

FROM Summary\_Report;

INSERT INTO Summary\_Report (

SELECT

STORE.store\_id,

ADDRESS.address,

SUM(amount) AS sales\_amount

FROM STORE

INNER JOIN STAFF

ON STORE.store\_id = STAFF.store\_id

INNER JOIN ADDRESS

ON STORE.address\_id = ADDRESS.address\_id

INNER JOIN PAYMENT

ON PAYMENT.staff\_id = STAFF.staff\_id

GROUP BY (STAFF.staff\_id, STORE.store\_id, ADDRESS.address\_id)

ORDER BY sales\_amount DESC

);

END;

$$ LANGUAGE PLPSQL;

CALL refresh\_data();

**Explanation of how stored procedure can be run on a schedule and ensure the data’s freshness:**

The procedure will drop both the Detailed and Summary reports from the database. The procedure then recreates both reports using the query. This procedure should be run quarterly. This will ensure that data is recent for the company, while also ensuring the stores and staff members have ample time to increase sale number. This automation can be accomplished by utilizing an Agent such as pgAgent to schedule a specific date every 3 months for the script to execute.

**Query to check accuracy of refresh\_data procedure:**

SELECT \*

FROM Detailed\_Report;

SELECT \*

FROM Summary\_Report;

**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 web sources or third-party code was used in my application.

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

* No content is quoted, paraphrased, or summarized from other sources.