Performance Assessment: Advanced Data Management (CSN1)

Ralph Moore

College of Information Technology, Western Governors University

D326: Advanced Data Management

Instructor Jan Flory

December 11, 2023

Contents

[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. 3](#_Toc154241913)

[B. Provide original code for function(s) in text format that perform the transformation(s) you identified in part A4. 6](#_Toc154241914)

[C. Provide original SQL code in a text format that creates the detailed and summary tables to hold your report table sections. 10](#_Toc154241915)

[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. 13](#_Toc154241916)

[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 fom part D. 16](#_Toc154241917)

[G. Provide a Panopto video recording that includes the presenter and a vocalized demonstration of the functionality of the code used for the analysis. 18](#_Toc154241918)

[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. 18](#_Toc154241919)

[References 19](#_Toc154241920)

[Raw Code 20](#_Toc154241921)

Performance Assessment: Advanced Data Management (CSN1)

Data analysts frequently transform data within a database so that it can be used for analysis and so that the data will be easier for nontechnical users to use and understand. You will emulate such a process in this task by choosing your own business question to analyze, creating tables and queries to use as a business report, and streamlining your analysis by writing your own SQL functions, triggers, and stored procedures. This task defines a report as a collection of data that answers a real-world business question. Your report will have two distinct sections (SQL tables that you will create) that differ in the granularity of the data they present and how directly they support the answering of the business question you choose. The detailed table should contain all data that informs the answer to the question at a very granular level, and the summary table should contain aggregated data that provide a direct answer to the business question.

## 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 real-world business report that could be created from the DVD Dataset is analysis of sales per store by month. This information could be utilized for many business purposes, such as:

1. Make personalized offers
   1. Targeted marketing based on location
   2. Customer loyalty programs
2. The company could use the information for many business purposes
   1. Adjustments to inventory based on location
   2. Understand customer behavior
   3. Improving customer experience
   4. Understand customer trends
   5. Improve marketing

The business report will consist of two tables created in the DVD Dataset database, detailed report and summary report. The detailed report will include rentals, film names, genres, rental dates, and store id. The summary report will include total rentals by store by month.

1. **Identify the specific fields that will be included in the detailed table and the summary table of the report.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable name | Table Going in to | Database table the data will come from | Datatype of the field | Description or definition of the data field |
| Rental\_id | detailed table | rental | INT | Unique rental IDs from the rental table, will be counted by date and store to update the summary report. |
| Store\_id | detailed table | store | SMALLINT | Single digit, there are only two store locations, store 1 and store 2. |
| Film\_title | detailed table | film | Varchar(255) | Title of the film provided in the event subsequent reports will be run on this table. |
| Film\_genre | detailed table | category | Varchar(50) | Genre of film provided in the event subsequent reports will be run on this table. |
| Rental\_date | detailed table | rental | DATE | Date of rental transaction, this data will be transformed to get rentals by month for the summary table and business report. |
| Rental\_month | summary table | rental | VARCHAR | Produced by transformation on rental\_date, extracts the month from a date as a string. |
| Rental\_year | summary table | rental | INT | Produced by transformation on rental\_date, extracts the year from the date as an integer. |
| Total\_rentals | summary table | rental | BIGINT | Aggregate column, count of all unique rental\_ids for transactions in the rental table. |
| Store\_id | summary table | store | SMALLINT | Single digit, there are only two store locations, store 1 and store 2. |

1. **Describe the types of data fields used for the report.**

**The detailed table will include:**

* rental\_id INT – Unique rental IDs from the rental table, will be counted by date and store to update the summary report.
* store\_id SMALLINT - Single digit, there are only two store locations, store 1 and store 2.
* film\_title VARCHAR(255) – Title of the film provided in the event subsequent reports will be run on this table.
* film\_genre VARCHAR(50) – Genre of film provided in the event subsequent reports will be run on this table.
* rental\_date DATE – Date of rental transaction, this data will be transformed to get rentals by month for the summary table and business report.

**The summary table will include:**

* rental\_month VARCHAR –Produced by transformation on rental\_date, extracts the month from a date as a string.
* rental\_year INT –Produced by transformation on rental\_date, extracts the year from the date as an integer.
* total\_rentals BIGINT –Aggregate column, count of all unique rental\_ids for transactions in the rental table.
* store\_id SMALLINT – Single digit, there are only two store locations, store 1 and store 2.

1. **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.**

For the detailed table, data will be provided from the rental, inventory, film, film category, and category tables. For the summary report, data will be provided from the rental date and inventory tables.

|  |  |  |
| --- | --- | --- |
| Variable name | Table Going in to | Database table the data will come from |
| Rental\_id | detailed table | rental |
| Store\_id | detailed table | store |
| Film\_title | detailed table | film |
| Film\_genre | detailed table | category |
| Rental\_date | detailed table | rental |
| Variable name | **Table Going in to** | **Database table the data will come from** |
| Rental\_year | summary table | rental |
| Total\_rentals | summary table | Rental Aggregate column, count of all unique rental\_ids |
| Store\_id | summary table | store |

1. **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 rental\_date field in the detailed table will utilize a a custom transformation with a user-defined function by extracting the name of the month as a string from this field. The DVD Dataset business report is an analysis of sales per store by month. Having the month spelled out will make the report easier to read and understand.

1. **Explain the different business uses of the detailed table section and the summary table section of the report.**

The detailed report table contains individual rental id, the store id, film title, film genre and rental date records. The summary report will be created off the detailed report table and will be an overview containing rental year, total rentals, and store id. The main business use for these reports is comparing the rentals at the two different locations.

Understanding the difference in rentals by location will enable the business to make better business decisions and utilize location intelligence. Some possible benefits of the data produced by these reports could include:

* Location selection of future stores
* Optimizing marketing campaigns
* Targeted customer marketing campaigns
* Enhanced understanding of customer base
* Ensure staffing reflects the customer demand

1. **Explain how frequently your report should be refreshed to remain relevant to stakeholders.**

It is recommended that the report be refreshed monthly on the first of each month. The main use case of the report is to compare rentals of store one vs store two monthly. Refreshing monthly will allow the management to compare the monthly rentals of store one compared to store two. Based on the analysis of store one vs store two, management will be able to use the information to gain knowledge and wisdom to apply business decisions that are best for the business. Logical businesses decisions that could be drawn from the data include inventory adjustments, targeted marketing by location, deeper understanding of customer base by location and adequately staffing each location.

## Provide original code for function(s) in text format that perform the transformation(s) you identified in part A4.

**First Function:**

The rental\_date field in the detailed table will utilize a custom transformation with a user-defined function by extracting the name of the month as a string from this field. The DVD Dataset business report is an analysis of sales per store by month. Having the month spelled out will make the report easier to read and understand.

**Second Function:**

The rental\_date field in the detailed table will utilize a custom transformation with a user-defined function by extracting the year as an integer. Having the year available in the report will ensure that the analysis is conducted on the correct month and not a previous year.

A screenshot of a computer

Description automatically generated

**--RETURN MONTH AS STRING--**

CREATE OR REPLACE FUNCTION get\_month\_string(rental\_date DATE)

RETURNS TEXT

LANGUAGE plpgsql

AS $$

DECLARE month\_string TEXT;

BEGIN

SELECT TO\_CHAR (rental\_date, 'Month')

INTO month\_string;

RETURN month\_string;

END; $$;

A screenshot of a computer

Description automatically generated

**--RETURN YEAR AS INT--**

CREATE OR REPLACE FUNCTION get\_year(rental\_date DATE)

RETURNS INT

LANGUAGE plpgsql

AS $$

DECLARE sales\_year INT;

BEGIN

SELECT EXTRACT (YEAR from rental\_date)

INTO sales\_year;

RETURN sales\_year;

END; $$;

**Data verification: Output values from functions to verify data in the summary table.**

A screenshot of a computer program

Description automatically generated

**--CREATE RENTALS BY MONTH TABLE--**

**--Data verification for summary report table--**

CREATE TABLE rentals\_by\_month(

rental\_month VARCHAR(25),

rental\_year INT,

store\_id SMALLINT,

rental\_id INT

);

INSERT INTO rentals\_by\_month (

rental\_month,

rental\_year,

store\_id,

rental\_id

)

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

store\_id,

rental\_id

FROM detailed\_report;

A screenshot of a computer

Description automatically generated

**--DATA VERIFICATION QUERY TOTALS BOTH STORES RENTALS IN SUMMARY REPORT--**

SELECT rental\_month,

COUNT(rental\_id) AS total\_rentals

FROM rentals\_by\_month

GROUP BY rental\_month

ORDER BY total\_rentals;

**--RESULTS FROM VERIFICATION QUERY—**

A screenshot of a computer

Description automatically generated

**--SELECTING ALL FROM SUMMARY\_REPORT—**

SELECT \* FROM summary\_report

ORDER BY rental\_month;

A screenshot of a computer

Description automatically generated

Adding store 1 and store 2 August rentals produces 5,686 which equal to the total\_rentals column in the rentals\_by\_month table. Data validated.

## Provide original SQL code in a text format that creates the detailed and summary tables to hold your report table sections.

A screenshot of a computer

Description automatically generated

**--CREATE DETAILED REPORT TABLE--**

CREATE TABLE detailed\_report (

rental\_id INT,

store\_id SMALLINT,

film\_title VARCHAR(255),

film\_genre VARCHAR(50),

rental\_date DATE

);

**--CREATE SUMMARY REPORT TABLE--**

CREATE TABLE summary\_report (

rental\_month VARCHAR(25),

rental\_year INT,

total\_rentals BIGINT,

store\_id SMALLINT

);

1. 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.

A screenshot of a computer screen

Description automatically generated

**--POPULATE DETAILED REPORT TABLE--**

INSERT INTO detailed\_report (

rental\_id,

store\_id,

film\_title,

film\_genre,

rental\_date

)

SELECT

r.rental\_id,

i.store\_id,

f.title as film\_title,

cat.name as film\_genre,

r.rental\_date

FROM rental AS r

INNER JOIN inventory AS i ON i.inventory\_id = r.inventory\_id

INNER JOIN film AS f ON f.film\_id = i.film\_id

INNER JOIN film\_category as fcat ON fcat.film\_id = f.film\_id

INNER JOIN category as cat ON cat.category\_id = fcat.category\_id;

**Data population of the summary report**

A screenshot of a computer screen

Description automatically generated

INSERT INTO summary\_report(

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

## 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.

A screenshot of a computer

Description automatically generated

**--TRIGGER CREATION BETWEEN 2 TABLES—**

--updates the summary table when data is added to the detailed table--

CREATE OR REPLACE FUNCTION summary\_updater\_function()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM summary\_report;

INSERT INTO summary\_report

(SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

RETURN NEW;

END; $$;

CREATE TRIGGER summary\_updater\_trigger

AFTER INSERT OR UPDATE OR DELETE ON detailed\_report

FOR EACH STATEMENT

EXECUTE PROCEDURE summary\_updater\_function();

**Verifying trigger**

A screenshot of a computer

Description automatically generated

--VERIFYING TRIGGER ON DETAILED\_REPORT--

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16044—

A screenshot of a computer

Description automatically generated

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 10—

A screenshot of a computer

Description automatically generated

INSERT INTO detailed\_report VALUES (46000, 2, 'Rocky', 'Boxing', '01-01-1978')

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16045—

A screenshot of a computer

Description automatically generated

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 11—

A screenshot of a computer

Description automatically generated

DELETE FROM detailed\_report WHERE rental\_id = 46000 RETURNING \*;

--this will show the exact row that is deleted--

A screenshot of a computer

Description automatically generated

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16044—

A screenshot of a computer

Description automatically generated

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 10—

A screenshot of a computer

Description automatically generated

--RESULT IS 10, delete was successful--

## 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.

A screenshot of a computer

Description automatically generated

**--STORED PROCEDURE REFESHES REPORTS--**

CREATE OR REPLACE PROCEDURE report\_refresher()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM detailed\_report;

DELETE FROM summary\_report;

INSERT INTO detailed\_report (

SELECT

r.rental\_id,

i.store\_id,

f.title as film\_title,

cat.name as film\_genre,

r.rental\_date

FROM rental AS r

INNER JOIN inventory AS i ON i.inventory\_id = r.inventory\_id

INNER JOIN film AS f ON f.film\_id = i.film\_id

INNER JOIN film\_category as fcat ON fcat.film\_id = f.film\_id

INNER JOIN category as cat ON cat.category\_id = fcat.category\_id);

INSERT INTO summary\_report (

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

RETURN;

END; $$;

**--TO CALL PROCEDURE--**

CALL report\_refresher();

A screenshot of a phone

Description automatically generated

1. **Identify a relevant job scheduling tool that can be used to automate the stored procedure.**

A scheduling tool that can be used to automate the stored procedure is pgAgent. “pgAgent is a scheduling agent that runs and manages jobs; each job consists of steps and schedules”. For instructions, please visit [Creating a pgAgent Job — pgAdmin 4 7.8 documentation](http://127.0.0.1:12068/help/help/pgagent_jobs.html) (pgAdmin 4 7.8 documentation, 2023)

## Provide a Panopto video recording that includes the presenter and a vocalized demonstration of the functionality of the code used for the analysis.

## 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.

See references.

## References

*pgAdmin 4 7.8 documentation*. (2023, December 23). Retrieved from Creating a pgAgent Job: http://127.0.0.1:12068/help/help/pgagent\_jobs.html

## Raw Code

**--CREATE DETAILED REPORT TABLE--**

CREATE TABLE detailed\_report (

rental\_id INT,

store\_id SMALLINT,

film\_title VARCHAR(255),

film\_genre VARCHAR(50),

rental\_date DATE

);

**--POPULATE DETAILED REPORT TABLE--**

INSERT INTO detailed\_report (

rental\_id,

store\_id,

film\_title,

film\_genre,

rental\_date

)

SELECT

r.rental\_id,

i.store\_id,

f.title as film\_title,

cat.name as film\_genre,

r.rental\_date

FROM rental AS r

INNER JOIN inventory AS i ON i.inventory\_id = r.inventory\_id

INNER JOIN film AS f ON f.film\_id = i.film\_id

INNER JOIN film\_category as fcat ON fcat.film\_id = f.film\_id

INNER JOIN category as cat ON cat.category\_id = fcat.category\_id;

**--CREATE FUNCTIONS TO TRANSFORM MONTH AND YEAR--**

**--RETURN MONTH AS INT--**

CREATE OR REPLACE FUNCTION get\_month\_int(rental\_date DATE)

RETURNS INT

LANGUAGE plpgsql

AS $$

DECLARE sales\_month INT;

BEGIN

SELECT EXTRACT(MONTH FROM rental\_date)

INTO sales\_month;

RETURN sales\_month;

END; $$;

**--RETURN MONTH AS STRING--**

CREATE OR REPLACE FUNCTION get\_month\_string(rental\_date DATE)

RETURNS TEXT

LANGUAGE plpgsql

AS $$

DECLARE month\_string TEXT;

BEGIN

SELECT TO\_CHAR (rental\_date, 'Month')

INTO month\_string;

RETURN month\_string;

END; $$;

**--RETURN YEAR AS INT--**

CREATE OR REPLACE FUNCTION get\_year(rental\_date DATE)

RETURNS INT

LANGUAGE plpgsql

AS $$

DECLARE sales\_year INT;

BEGIN

SELECT EXTRACT (YEAR from rental\_date)

INTO sales\_year;

RETURN sales\_year;

END; $$;

**--CREATE RENTALS BY MONTH TABLE--**

--Data verification for summary report table--

CREATE TABLE rentals\_by\_month(

rental\_month VARCHAR(25),

rental\_year INT,

store\_id SMALLINT,

rental\_id INT

);

INSERT INTO rentals\_by\_month (

rental\_month,

rental\_year,

store\_id,

rental\_id

)

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

store\_id,

rental\_id

FROM detailed\_report;

**--DATA VERIFICATION QUERY TOTALS BOTH STORES RENTALS IN SUMMARY REPORT--**

SELECT rental\_month,

COUNT(rental\_id) AS total\_rentals

FROM rentals\_by\_month

GROUP BY rental\_month

ORDER BY total\_rentals;

**--CREATE SUMMARY REPORT TABLE--**

CREATE TABLE summary\_report (

rental\_month VARCHAR(25),

rental\_year INT,

total\_rentals BIGINT,

store\_id SMALLINT

);

INSERT INTO summary\_report(

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

**--TRIGGER CREATION BETWEEN 2 TABLES--**

**--updates the summary table when data is added to the detailed table--**

CREATE OR REPLACE FUNCTION summary\_updater\_function()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM summary\_report;

INSERT INTO summary\_report

(SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

RETURN NEW;

END; $$;

CREATE TRIGGER summary\_updater\_trigger

AFTER INSERT OR UPDATE OR DELETE ON detailed\_report

FOR EACH STATEMENT

EXECUTE PROCEDURE summary\_updater\_function();

**--VERIFYING TRIGGER ON DETAILED\_REPORT--**

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16044--

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 10--

INSERT INTO detailed\_report VALUES (46000, 2, 'Rocky', 'Boxing', '01-01-1978')

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16045--

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 11--

DELETE FROM detailed\_report WHERE rental\_id = 20000 RETURNING \*;

--Shows row that is deleted--

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16044--

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 10, delete was successful--

**--STORED PROCEDURE REFESHES REPORTS--**

CREATE OR REPLACE PROCEDURE report\_refresher()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM detailed\_report;

DELETE FROM summary\_report;

INSERT INTO detailed\_report (

SELECT

r.rental\_id,

i.store\_id,

f.title as film\_title,

cat.name as film\_genre,

r.rental\_date

FROM rental AS r

INNER JOIN inventory AS i ON i.inventory\_id = r.inventory\_id

INNER JOIN film AS f ON f.film\_id = i.film\_id

INNER JOIN film\_category as fcat ON fcat.film\_id = f.film\_id

INNER JOIN category as cat ON cat.category\_id = fcat.category\_id);

INSERT INTO summary\_report (

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

get\_year(rental\_date) AS rental\_year,

COUNT(rental\_id) AS total\_rentals,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

RETURN;

END; $$;

**--TO CALL PROCEDURE--**

CALL report\_refresher();