Performance Assessment: Advanced Data Management (CSN1)

Ralph Moore

Student ID: 001199141

D326: Advanced Data Management

Instructor Jan Flory

December 29, 2023

## Summarize one real-world written business report

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

* Make personalized offers
  + Targeted marketing based on location
  + Customer loyalty programs
* The company could use the information for many business purposes
  + Adjustments to inventory based on location
  + Understand customer behavior
  + Improving customer experience
  + Understand customer trends
  + Improve marketing

The business report will consist of two tables created in the DVD Dataset database, a detailed report and summary report. The detailed report will include:

* Rentals
* film names
* genres
* rental dates
* store id

The summary report will include total rentals by store by month.

1. **Identify the specific fields**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 for reports run on this table. |
| film\_genre | detailed table | category | Varchar(50) | Genre of film for reports run on this table. |
| rental\_date | detailed table | rental | DATE | Date of rental will be changed to obtain rentals by month for the detailed and summary table |
| rental\_month | summary table | rental | VARCHAR | Produced by changing rental date, changes the month from date to string. |
| rental\_year | summary table | rental | INT | Produced by changing rental date, changes year from date to integer. |
| count\_of\_rentals | summary table | rental | BIGINT | Aggregate column, count of all unique rental\_ids 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**

**The detailed table:**

* 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 for reports run on this table.
* film\_genre VARCHAR(50) – Genre of film for reports run on this table.
* rental\_date DATE – Date of rental will be changed to obtain rentals by month for the detailed and summary table

**The summary table:**

* rental\_month VARCHAR – Produced by changing rental date, changes the month from date to string.
* rental\_year INT – Changing rental date, changes year from date to integer.
* rentals\_sum - BIGINT –Aggregate column, count of all unique rental\_ids for rentals 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**

.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable name | Data Type | Table Going in | Table Coming From | Description |
| Rental\_id | INT | detailed table | rental | Unique rental IDs from the rental table, will be counted by date and store to update the summary report. |
| Store\_id | SMALLINT | detailed table | store | Single digit, there are only two store locations, store 1 and store 2. |
| Film\_title | VARCHAR | detailed table | film | Title of the film for reports run on this table. |
| Film\_genre | VARCHAR | detailed table | category | Genre of film for reports run on this table. |
| Rental\_date | DATE | detailed table | rental | Date of rental will be changed to obtain rentals by month for the detailed and summary table |
| Variable name |  | **Table Going in** | Table Coming From |  |
| rental\_month | VARCHAR | summary table | rental | Produced by changing rental date, changes the month from date to string. |
| rental\_year | INT | summary table | rental | Changing rental date, changes year from date to integer. |
| rentals\_sum | BIGINT | summary table | Rental Aggregate - count of all unique rental\_ids | Aggregate column, count of all unique rental\_ids for rentals in the rental table. |
| Store\_id | SMALLINT | summary table | store | Single digit, there are only two store locations, store 1 and store 2. |

1. **Identify at least one field in the detailed table**

Date of rental (Rental\_date) will be changed to month in order to obtain rentals by month for the detailed and summary table by utilizing a user-defined function.

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 reports**

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 a summary containing rental month and year, total rentals, and store id. The main business use for these reports is comparing the rentals at the two different locations by month.

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 runs**

The report be refreshed on the first of each month. The main use case of the report is to compare rentals of store one vs store two monthly. 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.

## Function in text format

A screenshot of a computer

Description automatically generated

**--MONTH AS STRING--**

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

RETURNS TEXT

LANGUAGE plpgsql

AS $$

DECLARE month\_as\_string TEXT;

BEGIN

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

INTO month\_as\_string;

RETURN month\_as\_string;

END; $$;

## Creates the tables

A screenshot of a computer

Description automatically generatedA 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

);

**--SUMMARY REPORT TABLE--**

CREATE TABLE summary\_report (

rental\_month VARCHAR(25),

rental\_year INT,

rentals\_sum BIGINT,

store\_id SMALLINT

);

## Extract the raw data

A screenshot of a computer

Description automatically generated

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

INSERT INTO detailed\_report (

rental\_id,

store\_id,

film\_title,

film\_genre,

rental\_date

)

SELECT

t.rental\_id,

y.store\_id,

f.title as film\_title,

cat.name as film\_genre,

t.rental\_date

FROM rental AS t

INNER JOIN inventory AS y ON y.inventory\_id = t.inventory\_id

INNER JOIN film AS f ON f.film\_id = y.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;

## Trigger

## A screenshot of a computer Description automatically generated

**--TRIGGER CREATION--**

--summary table updates based on entries to detailed table--

CREATE OR REPLACE FUNCTION Update\_summary\_function()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM summary\_report;

INSERT INTO summary\_report

(SELECT

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

year\_obtained(rental\_date) AS rental\_year,

COUNT(rental\_id) AS rentals\_sum,

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,

year\_obtained(rental\_date) AS rental\_year,

COUNT(rental\_id) AS rentals\_sum,

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; $$;

A screenshot of a computer

Description automatically generated

**--CREATE TRIGGER--**

CREATE TRIGGER update\_summary\_trigger

AFTER INSERT OR UPDATE OR DELETE ON detailed\_report

FOR EACH STATEMENT

EXECUTE PROCEDURE Update\_summary\_function();

## Stored procedure

A screenshot of a computer

Description automatically generated

## --STORED PROCEDURE REFESHES REPORTS--

## CREATE OR REPLACE PROCEDURE sp\_refresh\_data()

## LANGUAGE plpgsql

## AS $$

## BEGIN

## DELETE FROM detailed\_report;

## DELETE FROM summary\_report;

## INSERT INTO detailed\_report (

## SELECT

## t.rental\_id,

## y.store\_id,

## f.title as film\_title,

## cat.name as film\_genre,

## t.rental\_date

## FROM rental AS t

## INNER JOIN inventory AS y ON y.inventory\_id = t.inventory\_id

## INNER JOIN film AS f ON f.film\_id = y.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,

## year\_obtained(rental\_date) AS rental\_year,

## COUNT(rental\_id) AS rentals\_sum,

## 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,

## year\_obtained(rental\_date) AS rental\_year,

## COUNT(rental\_id) AS rentals\_sum,

## 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; $$;

## A screenshot of a computer Description automatically generated

## --CALL STORED PROCEDURE--

## CALL sp\_refresh\_data();

F1. pgAgent would be my choice of a scheduling agent. “pgAgent is a job scheduling agent for Postgres databases, capable of running multi-step batch or shell scripts and SQL tasks on complex schedules” (The pgAdmin Development Team, 2023).

## Panopto video

Included as a link: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=9615f0e8-16d9-4031-a4ab-b0e8013cf462>

## Acknowledge all utilized sources.

See references.

# References

The pgAdmin Development Team. (2023). *pgAgent*. Retrieved December 31, 2023, from pgAdmin 4 7.8 documentation: http://127.0.0.1:17767/help/help/pgagent.html