WGU D326

Advanced Data Management

Stephanie Beal

06/28/2025

1. **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.**

Late Return Behavior Analysis

The business report I’ve chosen to create using the dvdrental database, is to evaluate the rental return behavior of customers in order to optimize late fee policies with a goal of maximizing revenue from late fees without reducing overall rentals. By analyzing the relevant data, we can gain insights on how to adjust rental durations in different categories to maximize profit, offer potential incentives to repeat offenders, creating late fee tiers based on category - new release vs out for a year, etc.

This data will allow stake holders to refine policies to increase profits while maintaining customer satisfaction and loyalty.

**A1. Identify the specific fields that will be included in the detailed table and the summary table of the report.**

the following fields will be included in the summary table:

customer\_id(int), customer\_name(VARCHAR, total\_rentals(int), late\_returns(int), avg\_days\_late(int), total\_estimated\_fees(NUM), late\_return\_rate (%)

The following fields will be included in the detailed table:

rental\_id(int), customer\_id(int), customer\_name(VARCHAR), film\_title(VARCHAR), rental\_date(TIMESTAMP), return\_date(TIMESTAMP), rental\_duration(int), expected\_return\_date(TIMESTAMP), days\_late(INT), estimated\_late\_fee(NUM)

**A2. Describe the types of data fields used for the report.**

Data types included:

INTEGER - includes number of rentals, customer id, rental id, total rentals

VARCHAR - text of customer name

TIMESTAMP - dates and times of rental/returns

I NTERVAL - used to calculate how many days late

NUMERIC - used for fees and percentages

**A3. Identify at least two specific table types from the given dataset that will provide the data necessary for the detailed table section and the summary table section of the report.**

The summary detail will pull from the detailed table.

The detailed table will pull from the customer table and the rental table.

**A4. Identify at least one field font he detailed table section that will require a custom transformation with a user-defined function and explain why it should be transformed.**

The user defined function (transformation) used will be to transform days\_late, which is currently stored as an interval, to numeric. This transformation into whole days late makes it easier to aggregate total and average days late across rows and to filter results.

**A5. Explain the different business uses of the detailed table section ands the summary table section of the report.**

The summary table shows overall trends and can be used to determine the effectiveness of current late return policies. This allows for a quick overview to see if policies should be looked at more closely based on month to month changes in data.

The detailed table allows for drilling down into customer habits enabling the business to identify repeat late offenders, determine the average days late. This data can be used to adjust current policies to identify trends in customer late returns in order to adjust policies in such a way as to prevent loss of customer base, while also increasing revenue.

**A6. Explain how often your report should be refreshed to make it relevant to stakeholders.**

Reports should be run monthly. This will create a baseline that will be able to show if certain months or seasons are more prone to late returns. Seasonal releases, common blockbuster release months, etc can affect the data making it important to be able to determine when policies may need to be adjusted permanently or temporarily.

**Code Starts here** (assignment section descriptions are included in comments):

**-- Common queries used while testing --**

--DELETE FROM late\_return\_detail;

--SELECT \* FROM late\_return\_detail;

--DELETE FROM late\_return\_summary;

--SELECT \* FROM late\_return\_summary;

--DROP TABLE late\_return\_summary;

--DROP TABLE late\_return\_detail;

--TRUNCATE late\_return\_summary;

--TRUNCATE late\_return\_detail;

**B: Provide Original code for function(s) in text format that perform the transformations(s) you identified in part A4:**

--B Transformation identified in A4 from interval to numeric--

-- Create function to transform --

CREATE OR REPLACE FUNCTION interval\_to\_days(input\_interval INTERVAL)

RETURNS INTEGER AS $$

SELECT FLOOR(EXTRACT(EPOCH FROM input\_interval) / 86400)::INT;

$$ LANGUAGE sql IMMUTABLE STRICT;

--test function--

SELECT interval\_to\_days(INTERVAL '4 days 8 hours'); --should return whole day only of 4)

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

**- - Create Detailed Table**

DROP TABLE IF EXISTS late\_return\_detail;

CREATE TABLE late\_return\_detail (

rental\_id INTEGER PRIMARY KEY,

customer\_id INTEGER,

customer\_name TEXT,

film\_title TEXT,

rental\_date TIMESTAMP,

return\_date TIMESTAMP,

rental\_duration INTEGER,

expected\_return\_date TIMESTAMP,

days\_late INTERVAL,

estimated\_late\_fee NUMERIC(5,2)

);

DROP TABLE IF EXISTS late\_return\_summary;

CREATE TABLE late\_return\_summary (

customer\_id INTEGER PRIMARY KEY,

customer\_name TEXT,

total\_rentals INTEGER,

late\_returns INTEGER,

avg\_days\_late NUMERIC(4,2),

total\_estimated\_fees NUMERIC(6,2),

late\_return\_rate\_percentage NUMERIC(5,2)

);

-- test code --

SELECT \* FROM late\_return\_detail

SELECT \* FROM late\_return\_summary

**D.  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.**

--D Query for raw data for detailed section of report —

INSERT INTO late\_return\_detail

SELECT

r.rental\_id,

c.customer\_id,

CONCAT(c.first\_name, ' ', c.last\_name) AS customer\_name,

f.title AS film\_title,

r.rental\_date,

r.return\_date,

f.rental\_duration,

(r.rental\_date + (f.rental\_duration || ' days')::interval) AS expected\_return\_date,

CASE

WHEN r.return\_date > (r.rental\_date + (f.rental\_duration || ' days')::interval)

THEN r.return\_date - (r.rental\_date + (f.rental\_duration || ' days')::interval)

ELSE INTERVAL '0 days'

END AS days\_late,

CASE

WHEN r.return\_date > (r.rental\_date + (f.rental\_duration || ' days')::interval)

THEN ROUND(EXTRACT(DAY FROM r.return\_date - (r.rental\_date + (f.rental\_duration || ' days')::interval)) \* 0.50, 2)

ELSE 0

END AS estimated\_late\_fee,

cat.name AS category\_name

FROM rental r

JOIN customer c ON r.customer\_id = c.customer\_id

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

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

JOIN film\_category fc ON f.film\_id = fc.film\_id

JOIN category cat ON fc.category\_id = cat.category\_id

WHERE r.return\_date IS NOT NULL;

-- test code --

SELECT \* FROM late\_return\_detail -- 15861 rows

SELECT \* FROM late\_return\_summary -- 599 rows

--add extra row to detailed table to test trigger

INSERT INTO late\_return\_detail

VALUES ('158655', 600, 'Jon Smyth', 'star wars');

SELECT \* FROM late\_return\_summary

**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.**

**--E Create trigger on Detailed Table to update Summary Table —**

CREATE OR REPLACE FUNCTION trg\_update\_late\_return\_summary()

RETURNS TRIGGER AS $$

BEGIN

-- If the customer already exists in the summary table, update their data --

IF EXISTS (

SELECT 1 FROM late\_return\_summary WHERE customer\_id = NEW.customer\_id

) THEN

UPDATE late\_return\_summary

SET

total\_rentals = total\_rentals + 1,

late\_returns = late\_returns + CASE WHEN get\_days\_late(NEW.days\_late) > 0 THEN 1 ELSE 0 END,

total\_estimated\_fees = total\_estimated\_fees + NEW.estimated\_late\_fee,

avg\_days\_late = ROUND((

((avg\_days\_late \* late\_returns) + CASE WHEN get\_days\_late(NEW.days\_late) > 0 THEN get\_days\_late(NEW.days\_late) ELSE 0 END)

/ GREATEST(late\_returns + CASE WHEN get\_days\_late(NEW.days\_late) > 0 THEN 1 ELSE 0 END, 1)

), 2),

late\_return\_rate\_percentage = ROUND(

100.0 \* (late\_returns + CASE WHEN get\_days\_late(NEW.days\_late) > 0 THEN 1 ELSE 0 END) / (total\_rentals + 1),

2

)

WHERE customer\_id = NEW.customer\_id;

ELSE

-- new summary row for the customer --

INSERT INTO late\_return\_summary (

customer\_id,

customer\_name,

total\_rentals,

late\_returns,

avg\_days\_late,

total\_estimated\_fees,

late\_return\_rate\_percentage

)

VALUES (

NEW.customer\_id,

NEW.customer\_name,

1,

CASE WHEN get\_days\_late(NEW.days\_late) > 0 THEN 1 ELSE 0 END,

CASE WHEN get\_days\_late(NEW.days\_late) > 0 THEN get\_days\_late(NEW.days\_late) ELSE 0 END,

NEW.estimated\_late\_fee,

CASE WHEN get\_days\_late(NEW.days\_late) > 0 THEN 100.00 ELSE 0.00 END

);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

- - **create and execute** - -

DROP TRIGGER IF EXISTS update\_late\_return\_summary ON late\_return\_detail;

CREATE TRIGGER updated\_late\_return\_summary

AFTER INSERT ON late\_return\_detail

FOR EACH ROW

EXECUTE FUNCTION trg\_update\_late\_return\_summary();

-- **test to make sure it works** —

INSERT INTO late\_return\_detail (

rental\_id, customer\_id, customer\_name, film\_title,

rental\_date, return\_date, rental\_duration, expected\_return\_date,

days\_late, estimated\_late\_fee, category\_name

)

VALUES (

98675, 1, 'Mary Smith', 'Star Wars',

'2025-06-01', '2025-06-05', 2, '2025-06-03',

INTERVAL '5 days', 1.00, 'Action'

);

**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 from part D.**

DROP PROCEDURE IF EXISTS refresh\_late\_return\_report();

-- Create the procedure

CREATE OR REPLACE PROCEDURE refresh\_late\_return\_report()

LANGUAGE plpgsql

AS $$

BEGIN

-- clear the report tables

DELETE FROM late\_return\_detail;

DELETE FROM late\_return\_summary;

-- re-populate detailed table

INSERT INTO late\_return\_detail

SELECT

r.rental\_id,

c.customer\_id,

CONCAT(c.first\_name, ' ', c.last\_name) AS customer\_name,

f.title AS film\_title,

r.rental\_date,

r.return\_date,

f.rental\_duration,

(r.rental\_date + (f.rental\_duration || ' days')::interval) AS expected\_return\_date,

CASE

WHEN r.return\_date > (r.rental\_date + (f.rental\_duration || ' days')::interval)

THEN r.return\_date - (r.rental\_date + (f.rental\_duration || ' days')::interval)

ELSE INTERVAL '0 days'

END AS days\_late,

CASE

WHEN r.return\_date > (r.rental\_date + (f.rental\_duration || ' days')::interval)

THEN ROUND(EXTRACT(DAY FROM r.return\_date - (r.rental\_date + (f.rental\_duration || ' days')::interval)) \* 0.50, 2)

ELSE 0

END AS estimated\_late\_fee

FROM rental r

JOIN customer c ON r.customer\_id = c.customer\_id

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

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

WHERE r.return\_date IS NOT NULL;

RETURN;

END;

$$;

**-- clear summary table —**

TRUNCATE TABLE late\_return\_summary;

**-- re-populate summary table —**

**— refresh report**

CALL refresh\_late\_return\_report();

SELECT \* FROM late\_return\_detail

SELECT \* FROM late\_return\_summary

**F1.  Identify a relevant job scheduling tool that can be used to automate the stored procedure.**

While there are other options that could be used sending on the situation, a good, relevant job scheduling tool for this stored procedure specifically, would be pgAgent because it is specifically for use with PostgreSQL, has graphic user interface integration with pgAdmin, and supports running stored procedures.

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

Link included:

**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.**

Utilized sources for the project include direct reference to the WGU D326 Task Overview questions and rubric as well as heavily utilizing <https://www.postgresql.org/> for functions and triggers in particular. I also referred to the D326 Course Guide.