WGU SQL Business Report — DVD Rental Database

Student: Debra J Viel

Course: Advanced Data Management (D326)

Task: Real-World Business Report

# A. Business Report

This business report provides insights into category-level revenue performance using data from the DVD Rental database. The goal is to enable fast and accurate reporting of total revenue, number of rentals, and average rental amount per category, supported by a detailed table for drill-down analysis.

## A1. Specific Fields

Detailed Table (detailed\_category\_revenue): rental\_id, rental\_date, customer\_id, film\_id, title, category, amount, duration\_label.

Summary Table (summary\_category\_revenue): category, total\_revenue, total\_rentals, avg\_rental.

## A2. Types of Data Fields

INTEGER: rental\_id, customer\_id, film\_id, total\_rentals

TIMESTAMP: rental\_date

TEXT: title, category, duration\_label

NUMERIC(5,2) or NUMERIC(10,2): amount, total\_revenue, avg\_rental

## A3. Source Tables

rental, payment, inventory, film, film\_category, category

## A4. Field Requiring Custom Transformation

Field: rental\_duration (from film)

Transformation: Convert integer days into Short / Medium / Long labels.

Why: Simplifies grouping and makes the data more readable for business users.

## A5. Business Uses

• Detailed table: Enables deeper analysis, customer lookups, and investigation of rental trends and outliers.

• Summary table: Supports fast reporting and category performance comparisons for promotional and operational decisions.

## A6. Report Refresh Frequency

A daily refresh is appropriate to ensure decision-makers always have current data on category-level revenue.

# B. Transformation Function

CREATE OR REPLACE FUNCTION duration\_label(days INTEGER)  
RETURNS TEXT AS $$  
BEGIN  
 IF days <= 3 THEN  
 RETURN 'Short';  
 ELSIF days BETWEEN 4 AND 6 THEN  
 RETURN 'Medium';  
 ELSE  
 RETURN 'Long';  
 END IF;  
END;  
$$ LANGUAGE plpgsql;

# C. Detailed and Summary Table Creation

CREATE TABLE IF NOT EXISTS detailed\_category\_revenue (  
 rental\_id INTEGER,  
 rental\_date TIMESTAMP,  
 customer\_id INTEGER,  
 film\_id INTEGER,  
 title TEXT,  
 category TEXT,  
 amount NUMERIC(5,2),  
 duration\_label TEXT  
);  
  
CREATE TABLE IF NOT EXISTS summary\_category\_revenue (  
 category TEXT,  
 total\_revenue NUMERIC(10,2),  
 total\_rentals INTEGER,  
 avg\_rental NUMERIC(10,2)  
);

# D. Raw Data Extraction Query

INSERT INTO detailed\_category\_revenue  
SELECT   
 r.rental\_id,  
 r.rental\_date,  
 r.customer\_id,  
 f.film\_id,  
 f.title,  
 c.name AS category,  
 COALESCE(p.amount, 0) AS amount,  
 duration\_label(f.rental\_duration) AS duration\_label  
FROM rental r  
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 c ON fc.category\_id = c.category\_id  
LEFT JOIN payment p ON p.rental\_id = r.rental\_id;

# E. Trigger to Update Summary

CREATE OR REPLACE FUNCTION update\_summary()  
RETURNS TRIGGER AS $$  
BEGIN  
 DELETE FROM summary\_category\_revenue;  
  
 INSERT INTO summary\_category\_revenue (category, total\_revenue, total\_rentals, avg\_rental)  
 SELECT category,  
 SUM(amount) AS total\_revenue,  
 COUNT(\*) AS total\_rentals,  
 AVG(amount) AS avg\_rental  
 FROM detailed\_category\_revenue  
 GROUP BY category;  
  
 RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
DROP TRIGGER IF EXISTS trg\_update\_summary ON detailed\_category\_revenue;  
  
CREATE TRIGGER trg\_update\_summary  
AFTER INSERT OR UPDATE OR DELETE  
ON detailed\_category\_revenue  
FOR EACH STATEMENT  
EXECUTE FUNCTION update\_summary();

# F. Stored Procedure (Revised — Fixed)

CREATE OR REPLACE PROCEDURE refresh\_category\_revenue()  
LANGUAGE plpgsql  
AS $$  
BEGIN  
 TRUNCATE TABLE summary\_category\_revenue;  
 TRUNCATE TABLE detailed\_category\_revenue;  
  
 INSERT INTO detailed\_category\_revenue  
 SELECT   
 r.rental\_id,  
 r.rental\_date,  
 r.customer\_id,  
 f.film\_id,  
 f.title,  
 c.name AS category,  
 COALESCE(p.amount, 0) AS amount,  
 duration\_label(f.rental\_duration) AS duration\_label  
 FROM rental r  
 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 c ON fc.category\_id = c.category\_id  
 LEFT JOIN payment p ON p.rental\_id = r.rental\_id;  
  
 -- Trigger rebuilds summary automatically  
END;  
$$;

# F1. Job Scheduling Tool

A scheduled job can be created in pgAgent to call the procedure automatically each day:

CALL refresh\_category\_revenue();

# G. Panopto Video Demonstration

Video Link: [Insert your Panopto video URL here]

The video demonstrates:

1. Running the original procedure and showing the duplicate summary issue.

2. Running the revised procedure.

3. Showing correct summary output after the fix.

# H. Sources

• WGU DVD Rental Database (Labs on Demand)  
• PostgreSQL Documentation

# I. Professional Communication

This submission clearly documents all code, the business purpose of the report, and the specific changes made to address evaluator feedback on Aspect F. The procedure was revised to rely on the trigger for summary table population, resolving the duplication issue.