

Joseph Abbott III

Maura Valentino

Advanced Data Management - D326

24 January 2026

Submission

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.

'There is four months of usable rental data. Since it is less than a year, an annual report will not be possible. A month to month report is possible. Our monthly report will focus on total payments by month and inventory. This will gauge monthly profit and indicate what inventory items are more popular.'

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

The summary table will include:

amount (numeric 5,2)

payment_id (primary key / foreign key / int)

film_id (foreign key / int)

month (primary key / varchar(9))

The detailed summary will include:

amount (numeric 5,2)

payment_id (primary key / foreign key / int)

film_id (foreign key / int)

month (varchar(9))

staff_id (int)

customer_id (int)

2. Describe the types of data fields used for the report.

varchar

integer (int)

decimal (numeric 5,2)

3. 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.

The necessary data is coming from the payment table and the inventory table.

4. 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).

payment_date will need to be transformed from a datetime stamp to just a month value. This is for readability.

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

The summary table shows monthly income and which films are producing that income. The detailed table provides the same information as the summary table, however, it also provides insight on the productivity of the staff, the number of unique customers, the number of repeat customers, how much the repeat customers are spending, and what films customers are drawn to renting.

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

These reports need to be compiled monthly within the first business day of the new month. This guarantees the information is current and accessible in time to predict the current month trends.

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

```
CREATE OR REPLACE FUNCTION month_string(payment_date TIMESTAMP)
RETURNS VARCHAR(9)
LANGUAGE plpgsql AS $$
```

```
DECLARE month_return VARCHAR(9);

BEGIN

SELECT to_char(payment_date, 'Month') INTO month_return;

RETURN month_return;

END;

$$;
```

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

```
CREATE TABLE detailed (

    payment_id INT PRIMARY KEY,

    film_id INT,

    staff_id INT,

    customer_id INT,

    amount DECIMAL(5,2),

    month VARCHAR(9)

);
```

```
CREATE TABLE summary (

    payment_id INT,

    film_id INT,

    amount DECIMAL(5,2),
```

```
month VARCHAR(9),  
  
CONSTRAINT fk_summary_payment  
  
FOREIGN KEY (payment_id) REFERENCES detailed(payment_id)  
  
);
```

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.

```
INSERT INTO detailed (payment_id, film_id, staff_id, customer_id, amount, month)  
  
SELECT  
  
p.payment_id,  
  
i.film_id,  
  
p.staff_id,  
  
p.customer_id,  
  
p.amount  
  
to_char(p.payment_date, 'YYYY-MM') AS month  
  
FROM payment p  
  
JOIN rental r  
  
ON r.rental_id = p.rental_id  
  
JOIN inventory i  
  
ON i.inventory_id = r.inventory_id  
  
WHERE p.payment_date >= timestamp '2007-02-01 00:00:00'  
  
AND p.payment_date <= timestamp '2007-05-31 23:59:59';
```

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.

```
/* Trigger function to update SUMMARY table after insert on DETAILED table */
```

```
CREATE OR REPLACE FUNCTION insert_trigger_function()
```

```
RETURNS TRIGGER
```

```
LANGUAGE plpgsql
```

```
AS $$
```

```
BEGIN
```

```
TRUNCATE summary;
```

```
INSERT INTO summary(payment_id, film_id, amount, month_return)
```

```
SELECT payment_id, film_id, amount, month_return
```

```
FROM detailed;
```

```
RETURN NEW;
```

```
END;
```

```
$$;
```

```
/* Trigger to call the insert_trigger_function after insert on DETAILED table */
```

```
CREATE TRIGGER new_summary
```

```
AFTER INSERT
```

```
ON detailed
```

```
FOR EACH STATEMENT
```

```
EXECUTE FUNCTION insert_trigger_function();
```

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.

```
/* Procedure to refresh DETAILED table */

CREATE OR REPLACE PROCEDURE refresh_tables()

LANGUAGE plpgsql

AS $$

BEGIN

TRUNCATE detailed;

INSERT INTO detailed (payment_id, film_id, staff_id, customer_id, amount, month_return)

SELECT

    p.payment_id,

    i.film_id,

    p.staff_id,

    p.customer_id,

    p.amount,

    to_char(p.payment_date, 'YYYY-MM') AS month_return

FROM payment p

JOIN rental r

    ON r.rental_id = p.rental_id

JOIN inventory i
```

```

ON i.inventory_id = r.inventory_id

WHERE p.payment_date >= timestamp '2007-02-01 00:00:00'

AND p.payment_date <= timestamp '2007-05-31 23:59:59';

TRUNCATE summary;

INSERT INTO summary(payment_id, film_id, amount, month_return)

SELECT payment_id, film_id, amount, month_return

FROM detailed;

END;

$$;

/* Call the procedure to refresh tables and display contents of SUMMARY and DETAILED
tables */

CALL refresh_tables();

SELECT * FROM SUMMARY;

SELECT * FROM DETAILED;

```

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

I would install pgAgent to handle automated job scheduling and management on postgresSQL databases. I would schedule the jobs off hours at 0100 on the first of each month. This time prevents interference in daily activities while compiling necessary information as soon as possible.

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.

No sources were used for this submission.