A.

My project is about the DVD rental business' total number of rentals. The business question I would ask is: Which DVD rental store rented the most movies in 2005? This report will show every rental that has occurred in each store by month, as well as how many rentals each store has in total for the year. The DVD business will benefit from this report because knowing which store had the most rentals will give them insight into which store has greater profitability. Knowing this can help them allocate more resources to that store to increase profit even further while experimenting at the lower-performing store.

A1.

The fields that will be included in the detailed table are month VARCHAR(9), store_id INT, film_title VARCHAR(255), and rental_id INT.
The fields that will be included in the summary table are store_id INT and rental_count INT.

A2.

The month field is a variable-length string with a maximum of 9 characters containing the month that the rental occurred in. The store_id field is an integer that identifies the specific store the rental happened at. The film_title field is a variable-length string with a maximum of 255 characters that contains the title of the film that has been rented. The rental_id field is an integer that identifies each specific rental. The rental_count field is an integer containing the number of total rentals for each store.

A3.

The tables that will provide the data necessary for the detailed and summary tables are the inventory, film, and rental tables. The month field, found in the detailed summary report, will be pulled from the rental_date field in the rental table. The store_id field, found in both detailed and summary tables, will be pulled from the store_id field in the inventory table. The film_title field, found in the detailed table, will be pulled from the title field of the film table. The rental_id field, found in the detailed table, will be pulled from the rental table. The rental_count field, found in the summary table, will be calculated by counting the number of rental_id fields found in the rental table.

A4.

One field in the detailed table that will require a custom transformation with a user-defined function is the month field. This field originates from the rental_date field in the rental table. It will transform the two-digit month value in the TIMESTAMP data type into a VARCHAR data type that will state the name of the month the rental occurred. This transformation is beneficial because it will be much easier to read. Instead of looking at a TIMESTAMP field, which may be difficult to read and keep track of for some, it will simply state a month. This will make it easier

to see the month each rental took place, rather than looking for the two-digit month value in each row.

A5.

The detailed table will show each rental that occurred in both DVD rental stores by month in 2005. The stakeholders will be able to see which specific movies are being rented each month. With this information, they will be able to decide if certain movies are being rented often enough to keep them in their inventory.
The summary table will show the total number of rentals each store has received in 2005. Stakeholders will be able to use this table to see which store has the most rentals, as well as determine if each store is meeting its sales goals. Each store may have different goals to hit depending on things like location and inventory, so their total numbers may differ from one another. The stakeholders will be able to assess what resources can be allocated to further increase profit at the higher-performing store and improve performance at the lower-performing store, while also identifying the strategies that aren't working.

A6.

To keep my report relevant to stakeholders, it should be refreshed monthly. This will keep the information up to date in both tables by listing out the movies that have been rented in the last month, while also adding those rentals to the yearly total.

B.

```
CREATE OR REPLACE FUNCTION get_month(rental_date TIMESTAMP)
RETURNS VARCHAR(9)
LANGUAGE plpgsql
AS
$$
DECLARE rental_month VARCHAR(9);
BEGIN
        SELECT TO_CHAR(rental_date, 'Month') INTO rental_month;
        RETURN rental_month;
END;
$$;
```

B1.

```
CREATE TABLE detailed_table (
        store_id INT,
        month VARCHAR(9),
        rental_id INT,
```

```
        film_title VARCHAR(255)
);

CREATE TABLE summary_table (
        store_id INT,
        rental_count INT,
);
```

C.

```
INSERT INTO detailed_table
        SELECT i.store_id, get_month(r.rental_date), r.rental_id, f.title
        FROM rental r
        JOIN inventory i
        ON i.inventory_id = r.inventory_id
        JOIN film f
        ON f.film_id = i.film_id
        WHERE r.rental_date BETWEEN '2005-01-01' AND '2005-12-31'
        ORDER BY i.store_id, r.rental_date;
```

D.

```
CREATE OR REPLACE FUNCTION insert_trigger_function()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
DELETE FROM summary_table;
INSERT INTO summary_table
SELECT store_id, COUNT(rental_id)
FROM detailed_table
GROUP BY store_id
ORDER BY store_id;
RETURN NEW;
END;
$$;
```

E.

```
CREATE TRIGGER new_detailed_table
    AFTER INSERT
    ON detailed_table
    FOR EACH STATEMENT
    EXECUTE PROCEDURE insert_trigger_function();
```

F.

```
CREATE OR REPLACE PROCEDURE refresh_tables()
LANGUAGE plpgsql
AS $$
DELETE FROM detailed_table;
DELETE FROM summary_table;

INSERT INTO detailed_table
        SELECT i.store_id, get_month(r.rental_date), r.rental_id, f.title
        FROM rental r
        JOIN inventory i
        ON i.inventory_id = r.inventory_id
        JOIN film f
        ON f.film_id = i.film_id
        WHERE r.rental_date BETWEEN '2005-01-01' AND '2005-12-31'
        ORDER BY i.store_id, r.rental_date;
```

F1.
A relevant job scheduling tool that can be used to automate the stored procedure is pgAgent.