**A.**

One business report idea would describe which category of DVDs bring in the most revenue. This would give an excellent view of what customers are looking for and can help inform managers on how to choose new products that customers will enjoy.

**A1.**

The summary table will include the name field from the category table and the amount field from the payment table.

The detailed table will include the payment_id field from the payment table, the name field from the category table, the title field from the film table, the amount field from the payment table, and the payment_date field from the payment table.

**A2.**

The category field is from the name field from the category table, uses a character varying (25) data type, will be called "category", and will be used in both the summary table and the detailed table. In the summary table it will be used to lay eat each individual category so that it can be matched with the total revenue associated with it. In the detailed table it will describe a specific rental's category.

The amount field is from the payment table, uses a money data type, will be called "amount", and will be used in both the summary and detailed tables. In the summary table it will show the total revenue made from rentals in a specific category, and in the detailed table it will show how much was paid for a specific rental.

The payment_id field is from the payment table, uses an integer data type, will be called "payment_id", and will only be used in the detailed table to identify the specific payment of a rental.

The title field is from the film table, uses a character varying (255) data type, will be called "title", and will only be used in the detailed table to identify a specific film being rented.

The payment_date field is from the payment table, uses a date data type, will be called "payment_date", and will only be used in the detailed table to determine the date a DVD rental was paid for.

**A3.**

The category table contains the names of film categories that can be used in both the summary and detailed tables. Also, the payment table contains the amount of money paid for each rental and can also be used in both the detailed and summary tables.

**A4.**

The payment_date field would be more useful if it only had the date within it instead of both the date and time. It is not necessary for this table since I am only concerned with the point within a year that a specific DVD was rented and not the point within a day.

**A5.**

The summary table section can be used to determine what category of DVDs people are buying and can inform managers on how to focus their efforts. For example, they could market themselves considering their more popular categories of movies as well as use this table to inform decisions on future acquisition of merchandise.

The detailed table view can be used to find exactly which DVD was purchased at a specific point in time. This can be useful for determining trends during certain parts of the year. For example, a manager could view this table and see that during the Christmas season more movies within the classic, comedy, and family categories are purchased.

**A6.**

The main goal of this report would be to assist in making decisions regarding marketing and merchandising and needs to be up-to-date whenever these kinds of decisions are being made. Therefore, I believe a quarterly update of these tables would be an appropriate time to refresh this report.

**B.**

```
CREATE OR REPLACE FUNCTION to_date(old_timestamp TIMESTAMP)
    RETURNS DATE
    LANGUAGE plpgsql
AS
$$
DECLARE new_date DATE;
BEGIN
    SELECT old_timestamp::DATE INTO new_date;
    RETURN new_date;
END;
$$;
```

C.

```
CREATE TABLE summary_table (
      category VARCHAR(25),
      amount MONEY
);

CREATE TABLE detailed_table (
      payment_id INTEGER,
      category VARCHAR(25),
      title VARCHAR (255),
      amount MONEY,
      payment_date DATE
);
```

D.

```
INSERT INTO detailed_table

SELECT payment.payment_id, category.name, film.title, payment.amount,
      to_date(payment.payment_date)
FROM payment

LEFT JOIN rental ON payment.rental_id = rental.rental_id
LEFT JOIN inventory ON rental.inventory_id = inventory.inventory_id
LEFT JOIN film ON inventory.film_id = film.film_id
LEFT JOIN film_category ON film.film_id = film_category.film_id
LEFT JOIN category ON film_category.category_id = category.category_id

ORDER BY payment.payment_date DESC;
```

E.

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

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

F.

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

    INSERT INTO detailed_table

    SELECT payment.payment_id, category.name, film.title, payment.amount,
        to_date(payment.payment_date)
    FROM payment

    LEFT JOIN rental ON payment.rental_id = rental.rental_id
    LEFT JOIN inventory ON rental.inventory_id = inventory.inventory_id
    LEFT JOIN film ON inventory.film_id = film.film_id
    LEFT JOIN film_category ON film.film_id = film_category.film_id
    LEFT JOIN category ON film_category.category_id = category.category_id

    ORDER BY payment.payment_date DESC;
END;
$$;
```

F1.

One job scheduling tool that can be used to automate this stored procedure is pgAgent.

H.

No sources were used to support this submission.