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.

Business Question:
"Which customers have late rentals and what is their contact information?"

This report will give a detailed summary of customers that have not yet returned a rental after a seven-day rental period. The purpose of this report is to provide the customer's name, email, and phone so that the business can send reminders to customers who have a rental that needs to be returned. It will also show how late the rentals are and how many rentals a customer has for each transaction. The summary table will return the number of customers who have late rentals. This could be used to see if late rentals are trending up or down.

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

Detailed Table:

- customer_id
- customer_name
- email
- phone
- day_late
- movies_late

Summary Table:

- today
- late_rentals

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

Detailed Table:

Column Name:	**Data Type:**	**Constraints:**	**Description:**
customer_id	INT	PRIMARY KEY, FOREIGN KEY	Customer's identification number
customer_name	VARCHAR(50)		Customer's first and last name
email	VARCHAR(50)		Customer's email
phone	VARCHAR(20)		Customer's phone number
days_late	INTERVAL		Calculates how many days past the seven-day rental period a rental is
movies	INT		Counts how many rentals a customer has

Summary Table:

Column Name:	**Data Type:**	**Constraints:**	**Description:**
today	DATE	PRIMARY KEY	Today's date
late_rentals	INT		Total number of customers with late rentals from Detailed Table

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.

I have used the customer, rental, and address tables to fill the detailed table. The summary table will take count of the dataset in the detailed 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).

I chose to transform the customer's first and last name fields. Using a user-defined function I combined the customer's first name and last name so that it would be easier to read and use when addressing the customer for rental return purposes.

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

   The detailed table section would be used to find the name, email, and phone of the customers who have withstanding rentals. The contact information in the detailed table section can be used to send reminders to customers who need to return rentals.

   The summary table would be used as a quick reference to see how many customers have withstanding rentals. This count can be used to see if withstanding rentals are trending up or down from previous months.

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

   The table should be updated at the beginning of each day so that reminders can be sent to customers who have not returned their rental. In this report I went with seven days to determine if a rental is late, although, that would be determined by the company's policy.

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

```
-- B: Transformation Function
-- Combines first and last name to fullname
CREATE OR REPLACE FUNCTION
name_concat(first_name VARCHAR(50), last_name VARCHAR(50))
RETURNS VARCHAR(50)
LANGUAGE plpgsql
AS $$
BEGIN
        RETURN first_name || ' ' || last_name;
END;
$$;
```

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

```
-- Clear Tables
DROP TABLE IF EXISTS late_detailed;
DROP TABLE IF EXISTS late_summary;

-- C: Create Detailed and Summary Tables
-- Create Detailed Table
CREATE TABLE IF NOT EXISTS late_detailed (
customer_id INT,
customer_name VARCHAR(50),
email VARCHAR(50),
```

```
phone VARCHAR(20),
days_late INTERVAL,
movies_late INT,
PRIMARY KEY(customer_id),
FOREIGN KEY(customer_id) REFERENCES customer(customer_id));

-- Create Summary Table
CREATE TABLE IF NOT EXISTS late_summary (
today DATE,
late_rentals INT,
PRIMARY KEY(today));
```

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: Inserts Data Into Detailed Table
/* Data Returned: Contact information from customers who have
not returned their rental after the 7 day rental period.
Also returns how many days past the rental period they are. */
INSERT INTO late_detailed
SELECT c.customer_id, name_concat(first_name, last_name), c.email, a.phone, now() -
r.rental_date - interval '7 days' AS days_late, COUNT(*) AS movies_late
FROM rental r
INNER JOIN customer c ON
c.customer_id = r.customer_id
INNER JOIN address a ON
a.address_id = c.address_id
WHERE now() - r.rental_date - interval '7 days' > interval '0 days' AND return_date IS null
GROUP BY c.customer_id, a.phone, r.rental_date
ORDER BY days_late DESC;
```

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: Update Trigger For Summary Table On Detailed Table Insert
-- Will trigger after insert into Detailed Table
CREATE OR REPLACE FUNCTION late_summary_update()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
        TRUNCATE late_summary;
        INSERT INTO late_summary
        SELECT CURRENT_DATE AS today, COUNT(*) as late_rentals
        FROM late_detailed;
```

```
        RETURN NEW;
END; $$;

CREATE TRIGGER late_summary_trigger
AFTER INSERT ON late_detailed
FOR EACH ROW
EXECUTE FUNCTION late_summary_update();
```

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.

```
-- F: Stored Precedure To Refresh Detailed & Summary Table
-- Procedure will delete data then repopulate
CREATE OR REPLACE PROCEDURE table_refresh()
LANGUAGE plpgsql
AS $$
BEGIN
        TRUNCATE late_detailed;
        INSERT INTO late_detailed
        SELECT c.customer_id, name_concat(first_name, last_name), c.email, a.phone, now() -
        r.rental_date - interval '7 days' AS days_late, COUNT(*) AS movies_late
        FROM rental r
        INNER JOIN customer c ON
        c.customer_id = r.customer_id
        INNER JOIN address a ON
        a.address_id = c.address_id
        WHERE now() - r.rental_date - interval '7 days' > interval '0 days' AND return_date IS
        null
        GROUP BY c.customer_id, a.phone, r.rental_date
        ORDER BY days_late DESC;
END; $$;

-- Call procedure
CALL table_refresh();

-- Show data from Summary and Detailed Tables
SELECT * FROM late_summary;
SELECT * FROM late_detailed;
```

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

The job scheduling tool I would use for this report is pgAgent. Its scheduling is integrated in PgAdmin 4 so it can be easily managed. There are options for specific days as well as times of the day down to the minutes. I could use scheduling tool to make a report at the beginning of each

workday so that employees could contact customers who have not yet returned their late rentals.

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

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=a442fee7-c983-4c05-b428-b176010d1edf

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.

Using "An Overview of Job Scheduling Tools for PostgreSQL" I concluded that pgAgent would be the best scheduling tool to use in my report (Dias, 2020). I used multiple examples on various pages of PostgreSQL Tutorial to help write my own code (PostgreSQL Tutorial, 2011)./

References

Dias, H. (2020, February 3). *An Overview of Job Scheduling Tools for PostgreSQL*. Retrieved from Severalnines: https://severalnines.com/blog/overview-job-scheduling-tools-postgresql/

PostgreSQL Tutorial. (2011). *PostgreSQL Create Function Statement*. Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-plpgsql/postgresql-create-function/

PostgreSQL Tutorial. (2011). *PostgreSQL Create Procedure*. Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-plpgsql/postgresql-create-procedure/

PostgreSQL Tutorial. (2011). *PostgreSQL Functions*. Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-functions/

PostgreSQL Tutorial. (2011). *PostgreSQL Triggers*. Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-triggers/

PostgreSQL Tutorial. (2011). *PostgreSQL Truncate Table*. Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-truncate-table/

PostgreSQL Tutorial. (2011). *PostgreSQL Tutorial*. Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/

ProstgreSQL Tutorial. (2011). *ProstgreSQL Interval Data Type*. Retrieved from ProstgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-interval/

I) Demonstrate professional communication in the content and presentation of your submission.