Section A – lay out what you are doing in the project. Pretend you are an owner of the DVD rental store and what are questions you might want to ask?

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.

*This project will be used to determine future rental options, you need to see which actors’ movies are most popular. I will be asking the question, who are the top ten actors with the highest sales. This helps you narrow down which film to order because the one with a top actor is most likely going to sell more copies. This will span over June, July, and August. The DVD business will benefit from this because they maximize specific rentals that they know will sell better.*

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

*Summary Table: Name VARCHAR(255) – Full first and last name (transformed in the detailed table). Total\_sales (TEXT) – transformed from numeric 80000 to 80,000.*

*Detailed Table: actor\_id(INT), first\_name (VARCHAR 45), last\_name (VARCHAR 45), amount (numeric), movie\_title (VARCHAR 255) (first\_name and Last\_name will be transformed into one column named ‘name VARCHAR(255)’, date(rental\_date transformed into MDY)(VARCHAR (12)).*

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

*Data fields used will be VARCHAR (This will contain names, dates, movie titles), INTEGER(will be used with the actor\_id), and TEXT(transformed numeric data for summary 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.

*Data from the summary tables will be compiled from the detailed table (name(transformed data from part 4 in the detailed table), SUM(amount) that has been transformed from 80000 to 80,000).*

*The detailed table will pull from Payment (amount), Rental (rental\_date), Inventory (just used to join to have access to the film table), Film(title), Film\_actor(actor\_id), and Actor (first\_name, last\_name) tables.*

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 will be transforming the first\_name and last\_name columns into one cohesive column for easier readability. I will also transform the rental\_date timestamp to MDY.*

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

*The summary table is for high level executives, they want to see the final picture, an overview of a conclusion. They can then have a focus point on which movie titles to order based on top actors. The detailed table is for the analysts who are there to spot trends and need more detail to identify those trends. They will be able to break it down further and look at which titles are bringing in the most dollars instead of focusing just on which actor brings in the most dollars. They can also narrow down by month and see which months have top actors.*

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

*The report should be refreshed on the 1st of the quarter to see if your top ten actors are still the same or if someone new is generating more money and look into ordering those films with the specific actor. You could do monthly, but I feel a better determination of a trend would be a compilation of a three-month period.*

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

*CREATE OR REPLACE FUNCTION full\_name(first\_name VARCHAR(45), last\_name VARCHAR(45))*

*RETURNS VARCHAR(255)*

*LANGUAGE plpgsql*

*AS*

*$$*

*BEGIN*

*RETURN first\_name || ' ' || last\_name;*

*END;*

*$$;*

*CREATE OR REPLACE FUNCTION Month(rental\_date TIMESTAMP)*

*RETURNS VARCHAR(12)*

*LANGUAGE plpgsql*

*AS $$*

*DECLARE RentalDate VARCHAR(12);*

*BEGIN*

*SELECT TO\_CHAR(rental\_date, 'FMMon DD, YYYY') INTO RentalDate;*

*RETURN RentalDate;*

*END;*

*$$;*

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

***Summary Tables****:*

*CREATE TABLE top\_ten (*

*Name VARCHAR(255),*

*Total\_sales TEXT*

*);*

***Detailed Tables:***

*CREATE TABLE actor\_dollars\_detail (*

*actor\_id INT,*

*name VARCHAR(255),*

*amount NUMERIC(5,2),*

*movie\_title VARCHAR(255),*

*date VARCHAR(12)*

*);*

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 actor\_dollars\_detail*

*SELECT actor.actor\_id, full\_name(first\_name, last\_name), amount, title, Month(rental\_date)*

*FROM payment*

*INNER JOIN rental*

*ON rental.customer\_id = payment.customer\_id*

*INNER JOIN inventory*

*ON inventory.inventory\_id = rental.inventory\_id*

*INNER JOIN film*

*ON film.film\_id = inventory.film\_id*

*INNER JOIN film\_actor*

*ON film\_actor.film\_id = film.film\_id*

*INNER JOIN actor*

*ON actor.actor\_id = film\_actor.actor\_id*

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.

*CREATE OR REPLACE FUNCTION insert\_trigger\_function()*

*RETURNS TRIGGER*

*LANGUAGE plpgsql*

*AS*

*$$*

*BEGIN*

*DELETE FROM top\_ten;*

*INSERT INTO top\_ten*

*SELECT name, TO\_CHAR(SUM(amount), 'FM999,999,999') AS total\_sales*

*FROM actor\_dollars\_detail*

*GROUP BY name*

*ORDER BY SUM(amount) DESC*

*LIMIT 10;*

*RETURN NEW;*

*END;*

*$$;*

*--added the delete so the table would update properly when practicing inserts and deletes.*

*CREATE TRIGGER quarterly\_update*

*AFTER INSERT OR DELETE*

*ON actor\_dollars\_detail*

*FOR EACH STATEMENT*

*EXECUTE PROCEDURE 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.

*CREATE OR REPLACE PROCEDURE refresh\_table()*

*LANGUAGE plpgsql*

*AS $$*

*BEGIN*

*DELETE FROM actor\_dollars\_detail;*

*DELETE FROM top\_ten;*

*INSERT INTO actor\_dollars\_detail*

*SELECT actor.actor\_id, Full\_name(first\_name, last\_name) AS name, amount, title, Month(rental\_date) AS date*

*FROM payment*

*INNER JOIN rental*

*ON rental.customer\_id = payment.customer\_id*

*INNER JOIN inventory*

*ON inventory.inventory\_id = rental.inventory\_id*

*INNER JOIN film*

*ON film.film\_id = inventory.film\_id*

*INNER JOIN film\_actor*

*ON film\_actor.film\_id = film.film\_id*

*INNER JOIN actor*

*ON actor.actor\_id = film\_actor.actor\_id*

*WHERE rental\_date BETWEEN '2005-06-01 00:00:00' AND '2005-08-31 00:00:00';*

*INSERT INTO top\_ten*

*SELECT name, TO\_CHAR(SUM(amount), 'FM999,999,999') AS total\_sales*

*FROM actor\_dollars\_detail*

*GROUP BY name*

*ORDER BY SUM(amount) DESC*

*LIMIT 10;*

*RETURN;*

*END;*

*$$;*

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

*Depending on what the company is using for SQL it’ll depend which scheduling tool to use. Since we used pgAdmin I would use the pgAgent tool. I picked this because there is a GUI that allows an easier functionality. I would refresh on the 1st of the month at midnight. This way on the 1st of every month analysts can come in and review the previous months data and make an ordering decision.*

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=0c420ee1-cf7f-4811-94da-b1a100d54690.

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 third-party sources used. Just the class materials*