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

--CREATE DETAILED REPORT TABLE--

CREATE TABLE detailed\_report (

rental\_id INT,

store\_id SMALLINT,

film\_title VARCHAR(255),

film\_genre VARCHAR(50),

rental\_date DATE

);

--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.--

--POPULATE DETAILED REPORT TABLE--

INSERT INTO detailed\_report (

rental\_id,

store\_id,

film\_title,

film\_genre,

rental\_date

)

SELECT

t.rental\_id,

y.store\_id,

f.title as film\_title,

cat.name as film\_genre,

t.rental\_date

FROM rental AS t

INNER JOIN inventory AS y ON y.inventory\_id = t.inventory\_id

INNER JOIN film AS f ON f.film\_id = y.film\_id

INNER JOIN film\_category as fcat ON fcat.film\_id = f.film\_id

INNER JOIN category as cat ON cat.category\_id = fcat.category\_id;

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

--FUNCTIONS - TRANSFORM MONTH AND YEAR--

--MONTH AS INT--

CREATE OR REPLACE FUNCTION get\_month\_int(rental\_date DATE)

RETURNS INT

LANGUAGE plpgsql

AS $$

DECLARE month\_of\_sale INT;

BEGIN

SELECT EXTRACT(MONTH FROM rental\_date)

INTO month\_of\_sale;

RETURN month\_of\_sale;

END; $$;

--MONTH AS STRING--

CREATE OR REPLACE FUNCTION get\_month\_string(rental\_date DATE)

RETURNS TEXT

LANGUAGE plpgsql

AS $$

DECLARE month\_as\_string TEXT;

BEGIN

SELECT TO\_CHAR (rental\_date, 'Month')

INTO month\_as\_string;

RETURN month\_as\_string;

END; $$;

--YEAR AS INT--

CREATE OR REPLACE FUNCTION year\_obtained(rental\_date DATE)

RETURNS INT

LANGUAGE plpgsql

AS $$

DECLARE year\_of\_sale INT;

BEGIN

SELECT EXTRACT (YEAR from rental\_date)

INTO year\_of\_sale;

RETURN year\_of\_sale;

END; $$;

--Data verification for summary report table--

CREATE TABLE rentals\_by\_month(

rental\_month VARCHAR(25),

rental\_year INT,

store\_id SMALLINT,

rental\_id INT

);

INSERT INTO rentals\_by\_month (

rental\_month,

rental\_year,

store\_id,

rental\_id

)

SELECT

get\_month\_string(rental\_date) AS rental\_month,

year\_obtained(rental\_date) AS rental\_year,

store\_id,

rental\_id

FROM detailed\_report;

--DATA VERIFICATION - TOTALS BOTH STORES RENTALS IN SUMMARY REPORT--

SELECT rental\_month,

COUNT(rental\_id) AS rentals\_sum

FROM rentals\_by\_month

GROUP BY rental\_month

ORDER BY rentals\_sum;

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

--SUMMARY REPORT TABLE--

CREATE TABLE summary\_report (

rental\_month VARCHAR(25),

rental\_year INT,

rentals\_sum BIGINT,

store\_id SMALLINT

);

INSERT INTO summary\_report(

SELECT

get\_month\_string(rental\_date) AS rental\_month,

year\_obtained(rental\_date) AS rental\_year,

COUNT(rental\_id) AS rentals\_sum,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

year\_obtained(rental\_date) AS rental\_year,

COUNT(rental\_id) AS rentals\_sum,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_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.

--TRIGGER CREATION--

--summary table updates based on entries to detailed table--

CREATE OR REPLACE FUNCTION Update\_summary\_function()

RETURNS TRIGGER

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM summary\_report;

INSERT INTO summary\_report

(SELECT

get\_month\_string(rental\_date) AS rental\_month,

year\_obtained(rental\_date) AS rental\_year,

COUNT(rental\_id) AS rentals\_sum,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

year\_obtained(rental\_date) AS rental\_year,

COUNT(rental\_id) AS rentals\_sum,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

RETURN NEW;

END; $$;

--CREATE TRIGGER--

CREATE TRIGGER update\_summary\_trigger

AFTER INSERT OR UPDATE OR DELETE ON detailed\_report

FOR EACH STATEMENT

EXECUTE PROCEDURE Update\_summary\_function();

--VERIFYING TRIGGER--

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16044--

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 10--

INSERT INTO detailed\_report VALUES (46000, 2, 'Rocky', 'Boxing', '01-01-1978')

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16045--

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 11--

DELETE FROM detailed\_report WHERE rental\_id = 46000 RETURNING \*;

--Shows row that is deleted--

SELECT COUNT(\*) FROM detailed\_report

--RESULT IS 16044--

SELECT COUNT(\*) FROM summary\_report

--RESULT IS 10, delete was successful--

--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

--STORED PROCEDURE REFESHES REPORTS--

CREATE OR REPLACE PROCEDURE sp\_refresh\_data()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM detailed\_report;

DELETE FROM summary\_report;

INSERT INTO detailed\_report (

SELECT

t.rental\_id,

y.store\_id,

f.title as film\_title,

cat.name as film\_genre,

t.rental\_date

FROM rental AS t

INNER JOIN inventory AS y ON y.inventory\_id = t.inventory\_id

INNER JOIN film AS f ON f.film\_id = y.film\_id

INNER JOIN film\_category as fcat ON fcat.film\_id = f.film\_id

INNER JOIN category as cat ON cat.category\_id = fcat.category\_id);

INSERT INTO summary\_report (

SELECT

get\_month\_string(rental\_date) AS rental\_month,

year\_obtained(rental\_date) AS rental\_year,

COUNT(rental\_id) AS rentals\_sum,

store\_id

FROM detailed\_report

WHERE store\_id =1

GROUP BY store\_id, rental\_month, rental\_year)

UNION ALL ((

SELECT

get\_month\_string(rental\_date) AS rental\_month,

year\_obtained(rental\_date) AS rental\_year,

COUNT(rental\_id) AS rentals\_sum,

store\_id

FROM detailed\_report

WHERE store\_id =2

GROUP BY store\_id, rental\_month, rental\_year

))

ORDER BY rental\_year, rental\_month, store\_id;

RETURN;

END; $$;

--CALL STORED PROCEDURE--

CALL sp\_refresh\_data();