1. **Summarize one real-world business report that can be created from the attached Data Sets and Associated Dictionaries.** What types of films by category are rented the most?
   1. The data that will be used for the report includes how many videos were rented and which category of movie was rented the most (e.g., Action, Comedy, Sci-Fi). Based on what category of movie was rented the most, the rental store can make inventory decisions. For instance, if action films were rented the most, then the rental store should order more action films.
   2. The data tables that provide the data necessary for this report are as follows: *rental, inventory, film, film\_category, and category*
   3. The detailed section(i), and the summary section (ii) will include the following fields
      1. rental\_id, inventory\_id, film\_id, title, category\_id, name
      2. category\_id, name(from category), units\_rented, percentage\_rented
   4. The field in the detailed section that will need transformation is the name column. There is a category named ‘New’ that is not a genre of film and as such is not of any value to the business report. A more useful title for this category of film is ‘unspecified’. ‘New’ does not tell us anything about the kind of film it is nor does it help make decisions about which films to buy.
   5. The detailed section will give those in charge of buying inventory the ability to see what kinds of films are being rented. It will also let them see rentals attached to specific film titles, rather than just an inventory\_id, which gives no specific information about what film is being rented. The summary section will give them an idea of what category of movie is being rented the most by unit and percentage. This will give them some data to go off when they buy new inventory.
   6. I think the data should be refreshed monthly to reflect how often new movies are released. Then, with at least a year's worth of data, trends can be found based on quarter and holiday season. For instance, we could test a hypothesis that during tax season patrons will rent more and adjust inventory accordingly. During the cold season, customers might watch more dramas, classics, and romance titles. So, we could keep up with the anticipated inventory needs by month and then by extension any insights derived from the exercises.
2. **Write a SQL code that creates the tables to hold your report sections**

-- create detailed table

DROP TABLE IF EXISTS detailed\_section;

CREATE TABLE detailed\_section

(

rental\_id integer,

inventory\_id integer,

film\_id smallint,

title varchar(255),

name varchar(25),

category\_id smallint

);

INSERT INTO detailed\_section

(

rental\_id,

inventory\_id,

film\_id,

title,

name,

category\_id

)

SELECT

r.rental\_id,

r.inventory\_id,

i.film\_id,

f.title,

c.name,

cat.category\_id

FROM rental AS r

INNER JOIN inventory AS i

ON r.inventory\_id = i.inventory\_id

INNER JOIN film AS f

ON i.film\_id = f.film\_id

INNER JOIN film\_category as cat

ON f.film\_id = cat.film\_id

INNER JOIN category as c

ON cat.category\_id = c.category\_id;

-- create summary section

DROP TABLE IF EXISTS summary\_section;

CREATE TABLE summary\_section

AS

SELECT \* FROM category;

ALTER TABLE summary\_section

DROP COLUMN last\_update;

ALTER TABLE summary\_section

ADD COLUMN units\_rented INT,

ADD COLUMN percentage\_rented DECIMAL(5,2);

1. **Write a SQL query that will extract the raw data needed for the Detailed section of your report from source database and verify the data’s accuracy.**

-- create detailed table

DROP TABLE IF EXISTS detailed\_section;

CREATE TABLE detailed\_section

(

rental\_id integer,

inventory\_id integer,

film\_id smallint,

title varchar(255),

name varchar(25),

category\_id smallint

);

INSERT INTO detailed\_section

(

rental\_id,

inventory\_id,

film\_id,

title,

name,

category\_id

)

SELECT

r.rental\_id,

r.inventory\_id,

i.film\_id,

f.title,

c.name,

cat.category\_id

FROM rental AS r

INNER JOIN inventory AS i

ON r.inventory\_id = i.inventory\_id

INNER JOIN film AS f

ON i.film\_id = f.film\_id

INNER JOIN film\_category as cat

ON f.film\_id = cat.film\_id

INNER JOIN category as c

ON cat.category\_id = c.category\_id;

-- function to transform 'New' category to 'unspecified'

CREATE OR REPLACE FUNCTION transform\_new()

RETURNS varchar(25)

language plpgsql

AS

$$

DECLARE

unspecified varchar(25) = 'unspecified';

BEGIN

return unspecified;

END;

$$;

-- update query using function

UPDATE detailed\_section

SET name = (SELECT transform\_new())

WHERE name = 'New';

-- create summary section

DROP TABLE IF EXISTS summary\_section;

CREATE TABLE summary\_section

AS

SELECT \* FROM category;

ALTER TABLE summary\_section

DROP COLUMN last\_update;

ALTER TABLE summary\_section

ADD COLUMN units\_rented INT,

ADD COLUMN percentage\_rented DECIMAL(5,2);

-- function to get the count per film category

CREATE OR REPLACE FUNCTION get\_count

(

cat\_id INT

)

RETURNS numeric

language plpgsql

AS

$$

DECLARE

film\_count numeric;

BEGIN

SELECT COUNT(category\_id)

INTO film\_count

FROM detailed\_section

WHERE category\_id = cat\_id;

return film\_count;

END;

$$;

-- populate units\_rented column with rent counts

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(1))

WHERE category\_id = 1;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(2))

WHERE category\_id = 2;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(3))

WHERE category\_id = 3;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(4))

WHERE category\_id = 4;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(5))

WHERE category\_id = 5;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(6))

WHERE category\_id = 6;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(7))

WHERE category\_id = 7;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(8))

WHERE category\_id = 8;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(9))

WHERE category\_id = 9;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(10))

WHERE category\_id = 10;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(11))

WHERE category\_id = 11;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(12))

WHERE category\_id = 12;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(13))

WHERE category\_id = 13;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(14))

WHERE category\_id = 14;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(15))

WHERE category\_id = 15;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(16))

WHERE category\_id = 16;

-- function to get count of total rentals

CREATE OR REPLACE FUNCTION get\_rental\_count()

RETURNS numeric

language plpgsql

AS

$$

DECLARE

rentals numeric;

BEGIN

SELECT SUM(units\_rented)

INTO rentals

FROM summary\_section;

RETURN rentals;

END;

$$;

-- function to get count of rentals by category

CREATE OR REPLACE FUNCTION get\_count\_by\_type(cat\_id integer)

RETURNS numeric

language plpgsql

AS

$$

DECLARE

rentals numeric;

BEGIN

SELECT SUM(units\_rented)

INTO rentals

FROM summary\_section

WHERE category\_id = cat\_id;

RETURN rentals;

END;

$$;

-- insert the percentages of films rented in summary section

-- using the scalar functions to divide

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(1)/get\_rental\_count() \* 100)

WHERE category\_id = 1;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(2)/get\_rental\_count() \* 100)

WHERE category\_id = 2;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(3)/get\_rental\_count() \* 100)

WHERE category\_id = 3;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(4)/get\_rental\_count() \* 100)

WHERE category\_id = 4;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(5)/get\_rental\_count() \* 100)

WHERE category\_id = 5;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(6)/get\_rental\_count() \* 100)

WHERE category\_id = 6;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(7)/get\_rental\_count() \* 100)

WHERE category\_id = 7;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(8)/get\_rental\_count() \* 100)

WHERE category\_id = 8;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(9)/get\_rental\_count() \* 100)

WHERE category\_id = 9;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(10)/get\_rental\_count() \* 100)

WHERE category\_id = 10;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(11)/get\_rental\_count() \* 100)

WHERE category\_id = 11;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(12)/get\_rental\_count() \* 100)

WHERE category\_id = 12;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(13)/get\_rental\_count() \* 100)

WHERE category\_id = 13;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(14)/get\_rental\_count() \* 100)

WHERE category\_id = 14;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(15)/get\_rental\_count() \* 100)

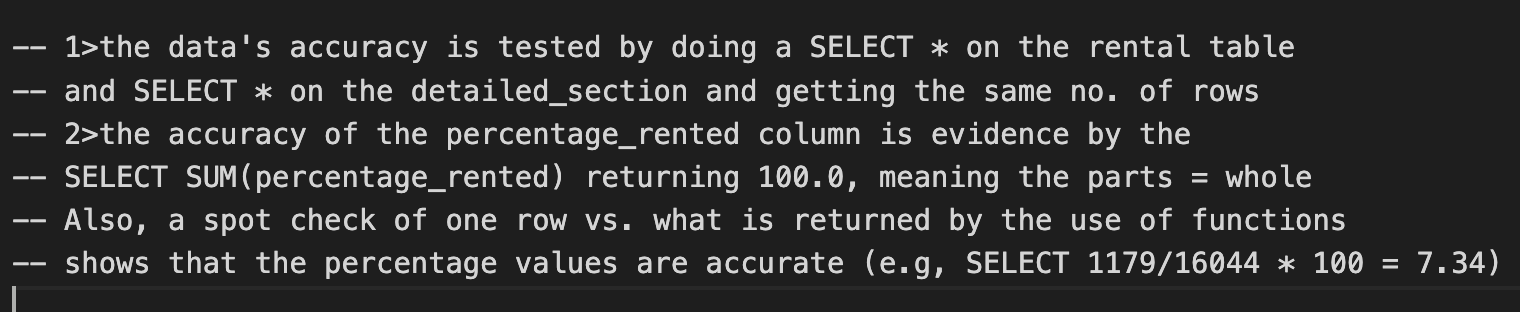
WHERE category\_id = 15;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(16)/get\_rental\_count() \* 100)

WHERE category\_id = 16;



1. **Write code for function(s) that perform the transformation(s) you identified in part A4.**

CREATE OR REPLACE FUNCTION transform\_new()

RETURNS varchar(25)

language plpgsql

AS

$$

DECLARE

unspecified varchar(25) = 'unspecified';

BEGIN

return unspecified;

END;

$$;

UPDATE summary\_section

SET name = (SELECT transform\_new())

WHERE name = 'New';

1. **Write a SQL code that creates a trigger on the detailed table of the report that will continually update the summary table as the data is added to the detailed table.**

-- create function that will be invoked when trigger is -- called

CREATE OR REPLACE FUNCTION update\_detailed\_sec()

RETURNS TRIGGER

language plpgsql

AS

$$

BEGIN

-- delete contents of detailed table

DELETE FROM detailed\_section;

-- re-insert into table with new row/s

INSERT INTO detailed\_section

(

rental\_id,

inventory\_id,

film\_id,

title,

name,

category\_id

)

SELECT

r.rental\_id,

r.inventory\_id,

i.film\_id,

f.title,

c.name,

cat.category\_id

FROM rental AS r

INNER JOIN inventory AS i

ON r.inventory\_id = i.inventory\_id

INNER JOIN film AS f

ON i.film\_id = f.film\_id

INNER JOIN film\_category as cat

ON f.film\_id = cat.film\_id

INNER JOIN category as c

ON cat.category\_id = c.category\_id;

-- Apply transformation to 'New' column

UPDATE detailed\_section

SET name = (SELECT transform\_new())

WHERE name = 'New';

-- DELETE contents of summary\_section

DELETE FROM summary\_section;

-- INSERT DATA BACK INTO summary\_section

INSERT INTO summary\_section

(category\_id, name)

(SELECT category\_id, name from category);

-- populate units\_rented column with rent counts

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(1))

WHERE category\_id = 1;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(2))

WHERE category\_id = 2;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(3))

WHERE category\_id = 3;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(4))

WHERE category\_id = 4;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(5))

WHERE category\_id = 5;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(6))

WHERE category\_id = 6;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(7))

WHERE category\_id = 7;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(8))

WHERE category\_id = 8;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(9))

WHERE category\_id = 9;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(10))

WHERE category\_id = 10;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(11))

WHERE category\_id = 11;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(12))

WHERE category\_id = 12;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(13))

WHERE category\_id = 13;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(14))

WHERE category\_id = 14;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(15))

WHERE category\_id = 15;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(16))

WHERE category\_id = 16;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(1)/get\_rental\_count() \* 100)

WHERE category\_id = 1;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(2)/get\_rental\_count() \* 100)

WHERE category\_id = 2;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(3)/get\_rental\_count() \* 100)

WHERE category\_id = 3;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(4)/get\_rental\_count() \* 100)

WHERE category\_id = 4;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(5)/get\_rental\_count() \* 100)

WHERE category\_id = 5;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(6)/get\_rental\_count() \* 100)

WHERE category\_id = 6;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(7)/get\_rental\_count() \* 100)

WHERE category\_id = 7;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(8)/get\_rental\_count() \* 100)

WHERE category\_id = 8;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(9)/get\_rental\_count() \* 100)

WHERE category\_id = 9;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(10)/get\_rental\_count() \* 100)

WHERE category\_id = 10;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(11)/get\_rental\_count() \* 100)

WHERE category\_id = 11;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(12)/get\_rental\_count() \* 100)

WHERE category\_id = 12;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(13)/get\_rental\_count() \* 100)

WHERE category\_id = 13;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(14)/get\_rental\_count() \* 100)

WHERE category\_id = 14;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(15)/get\_rental\_count() \* 100)

WHERE category\_id = 15;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(16)/get\_rental\_count() \* 100)

WHERE category\_id = 16;

RETURN NULL;

END;

$$;

-- create trigger

CREATE OR REPLACE TRIGGER refresh\_tables

AFTER INSERT

ON rental

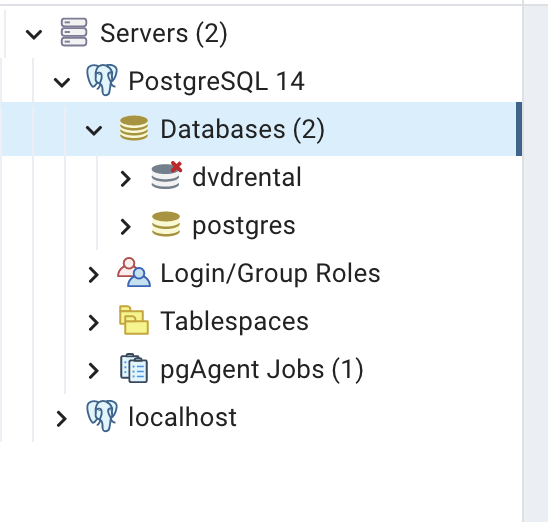
FOR EACH STATEMENT

EXECUTE PROCEDURE update\_detailed\_sec();

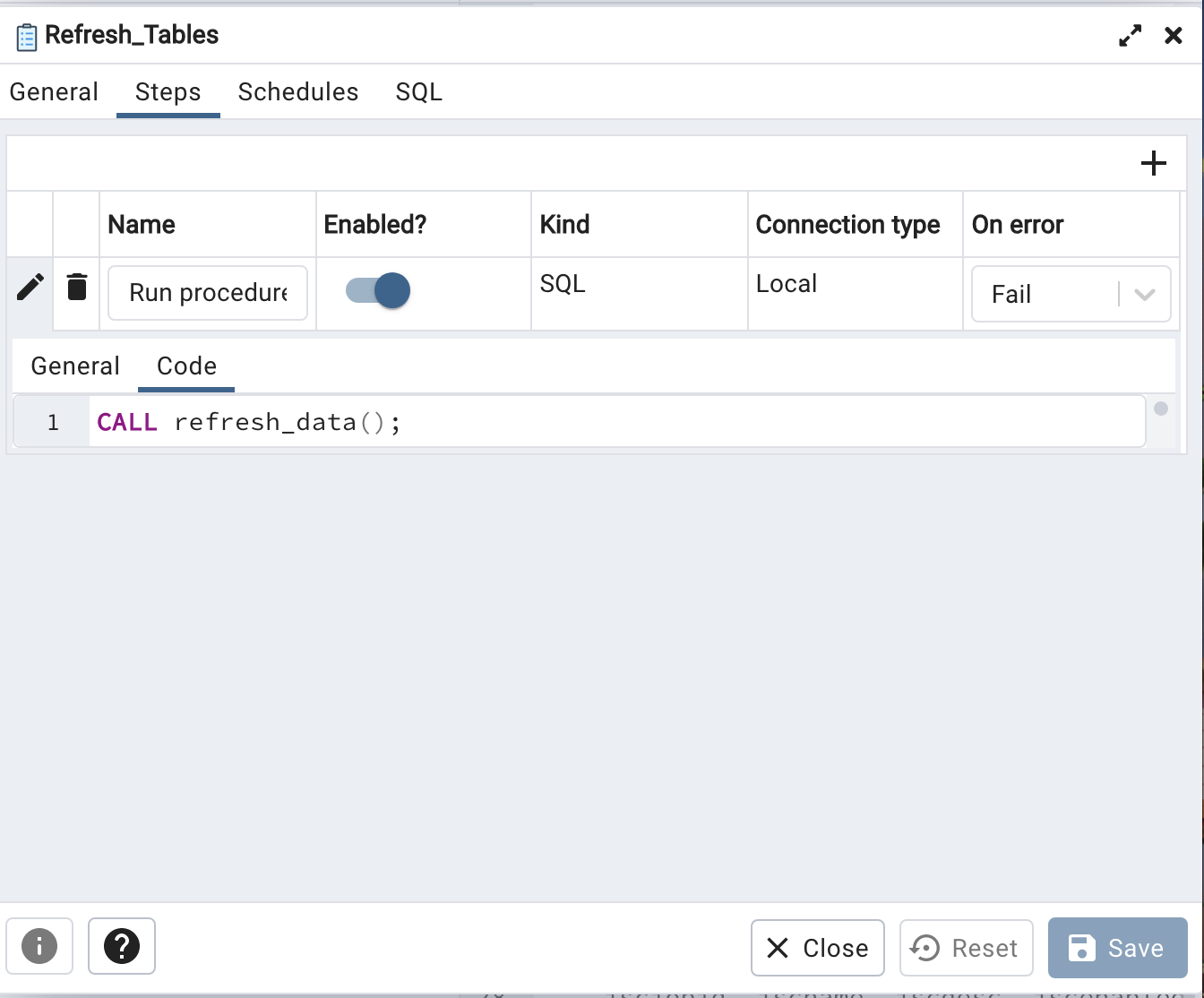
1. **Create a stored procedure that can be used to refresh the data in *both* your detailed and summary tables. The procedure should clear the contents of the detailed and summary tables and perform the ETL load process from part C and include comments that identify how often the stored procedure should be executed.**

1. **Explain how the stored procedure can be run on a schedule to ensure data freshness.**

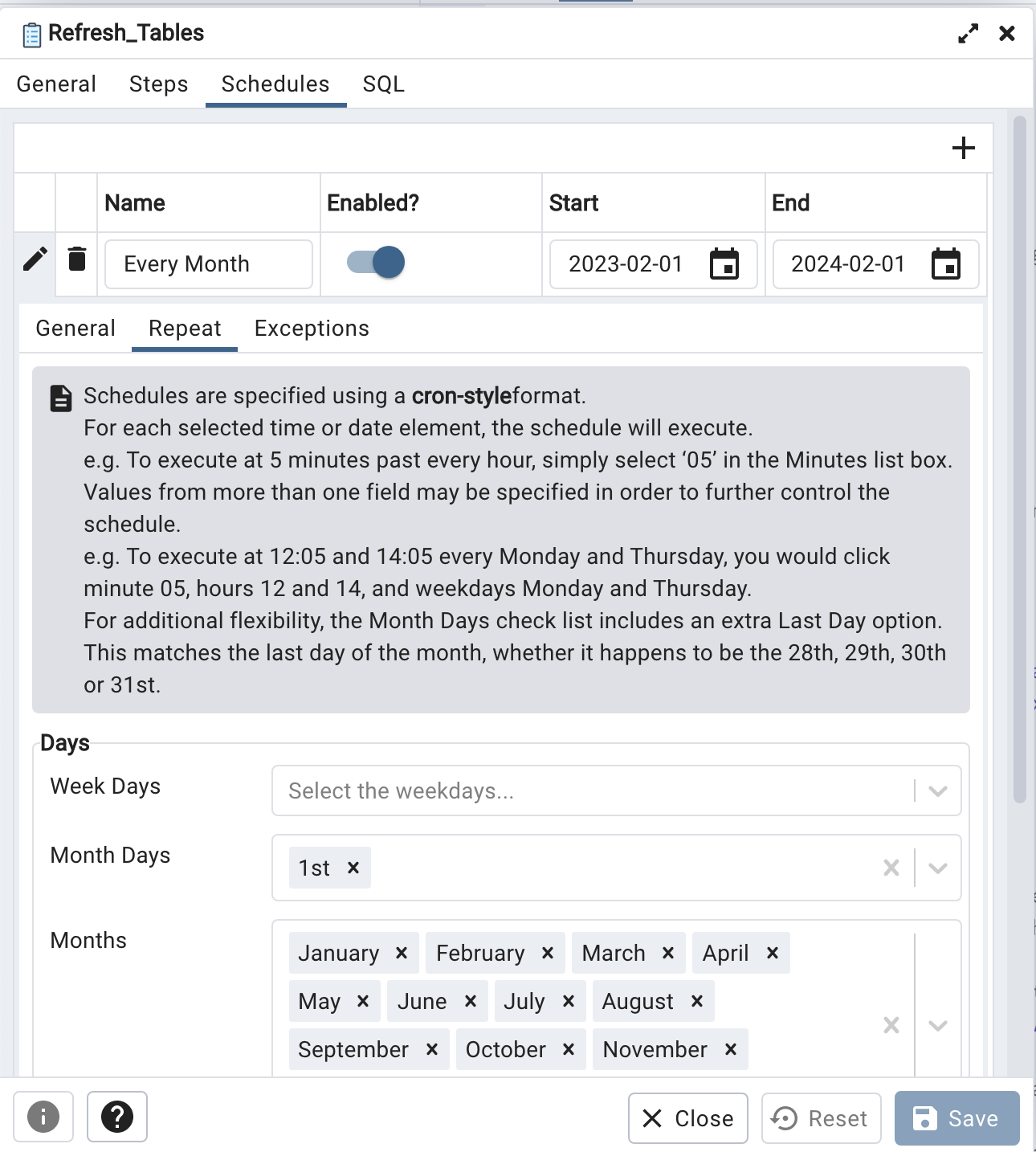
The stored procedure can be run on a schedule using pgAgent, the scheduling app that comes with PostgreSQL. A "job" can be created and the procedure can be called when the job is created.



The job is set up to call the procedure I have written called *refresh\_data.*



Then the job is set to run monthly per my recommendation in A.6.



CREATE OR REPLACE PROCEDURE refresh\_data()

LANGUAGE plpgsql

AS $$

BEGIN

DROP TABLE IF EXISTS detailed\_section;

CREATE TABLE detailed\_section

(

rental\_id integer,

inventory\_id integer,

film\_id smallint,

title varchar(255),

name varchar(25),

category\_id smallint

);

INSERT INTO detailed\_section

(

rental\_id,

inventory\_id,

film\_id,

title,

name,

category\_id

)

SELECT

r.rental\_id,

r.inventory\_id,

i.film\_id,

f.title,

c.name,

cat.category\_id

FROM rental AS r

INNER JOIN inventory AS i

ON r.inventory\_id = i.inventory\_id

INNER JOIN film AS f

ON i.film\_id = f.film\_id

INNER JOIN film\_category as cat

ON f.film\_id = cat.film\_id

INNER JOIN category as c

ON cat.category\_id = c.category\_id;

UPDATE detailed\_section

SET name = (SELECT transform\_new())

WHERE name = 'New';

DROP TABLE IF EXISTS summary\_section;

CREATE TABLE summary\_section

AS

SELECT \* FROM category;

ALTER TABLE summary\_section

DROP COLUMN last\_update;

ALTER TABLE summary\_section

ADD COLUMN units\_rented INT,

ADD COLUMN percentage\_rented DECIMAL(5,2);

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(1))

WHERE category\_id = 1;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(2))

WHERE category\_id = 2;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(3))

WHERE category\_id = 3;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(4))

WHERE category\_id = 4;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(5))

WHERE category\_id = 5;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(6))

WHERE category\_id = 6;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(7))

WHERE category\_id = 7;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(8))

WHERE category\_id = 8;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(9))

WHERE category\_id = 9;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(10))

WHERE category\_id = 10;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(11))

WHERE category\_id = 11;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(12))

WHERE category\_id = 12;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(13))

WHERE category\_id = 13;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(14))

WHERE category\_id = 14;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(15))

WHERE category\_id = 15;

UPDATE summary\_section

SET units\_rented =

(SELECT get\_count(16))

WHERE category\_id = 16;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(1)/get\_rental\_count() \* 100)

WHERE category\_id = 1;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(2)/get\_rental\_count() \* 100)

WHERE category\_id = 2;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(3)/get\_rental\_count() \* 100)

WHERE category\_id = 3;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(4)/get\_rental\_count() \* 100)

WHERE category\_id = 4;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(5)/get\_rental\_count() \* 100)

WHERE category\_id = 5;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(6)/get\_rental\_count() \* 100)

WHERE category\_id = 6;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(7)/get\_rental\_count() \* 100)

WHERE category\_id = 7;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(8)/get\_rental\_count() \* 100)

WHERE category\_id = 8;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(9)/get\_rental\_count() \* 100)

WHERE category\_id = 9;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(10)/get\_rental\_count() \* 100)

WHERE category\_id = 10;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(11)/get\_rental\_count() \* 100)

WHERE category\_id = 11;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(12)/get\_rental\_count() \* 100)

WHERE category\_id = 12;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(13)/get\_rental\_count() \* 100)

WHERE category\_id = 13;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(14)/get\_rental\_count() \* 100)

WHERE category\_id = 14;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(15)/get\_rental\_count() \* 100)

WHERE category\_id = 15;

UPDATE summary\_section

SET percentage\_rented =

(SELECT get\_count\_by\_type(16)/get\_rental\_count() \* 100)

WHERE category\_id = 16;

RETURN;

END;

$$;

***G. Video URL is attached to submission***

**H. Record the web sources you used to acquire data or segments of third-party code to support the application if applicable. Be sure the web sources are reliable.**

This is where I downloaded the dvd rental database and postgresql applications pgAdmin4 and pgAgent

https://www.postgresqltutorial.com/postgresql-getting-started/postgresql-sample-database/

https://www.postgresql.org/download/

**I.**  **Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.**

I did not use any citations or references in the production of this project.