2022

Advanced Data Management – D191

VDM1

DANIEL LAFORCE

A. Summarize one real-world business report that can be created from the attached Data Sets and Associated Dictionaries.

The attached data sets can be used to pull revenue reports for the DVD rental store. This will allow them to can get the full picture of how each of their retail locations is doing daily.

For this report, we need to pull each location's sales data. To do this we will pull each of the DVD rentals including the rental, payment date, customer information, and store rented from. From this information, we can create revenue reports.

1. Describe the Data used for the report

We will be utilizing the store's sales, customer information, inventory, and payment data.

2. Identify two or more specific tables from the given dataset that will provide the data necessary for the detailed and the summary sections of the report.

The following are the tables that will provide necessary information for the detailed elaboration of the report:

a) Rental – Table that includes all the DVD store rentals and when rented

b) Inventory – Table that includes each item available for rent

c) Customer – Table that includes each customer and their local store

d) Payment – Table that includes all payments made in both stores between 2/14/2007 and 05/14/2007

3. Identify the specific fields that will be included in the detailed and the summary sections of the report.

Details Report:

pid- Unique payment ID generated at the time of sale

rid – Unique rental ID generated at the time of sale

amount – Amount paid for the payment

date – Date/Time stamp of the rental

cid - Customer ID associated with the rental/payment

fname - Full Name concatenated from customer first and last name

email – Email of the customer

Summary Report:

sid - Summary ID identifier for each record in the summary report

date – Date the summary of sales for the shop applies to sales – Total sales made for the shop for that day

shop_id – Identifier of the shop sales are summarized for

4. Identify one field in the detailed section that will require a custom transformation and explain why it should be transformed. For example, you might translate a field with a value of 'N' to 'No' and 'Y' to 'Yes'.

The first_name and last_name fields in the customer table will need custom transformation to allow for their name to be within one field. Creating a concat field will help reduce the number of fields needed for the same information, while improving the readability.

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

The detailed report will give an in-depth look at the individual sales, including the customer, their rental purchase, the amount of their purchase and the date of their purchase. This report can be used to get in touch with what the customers enjoy watching, or which customers they may wish to reward brand loyalty.

The summary report will give a high-level summary of how each store did each day. This report can be utilized for incentivizing store competitions to increase revenue, determine sales trends based upon the day or the month. A similar report can help determine specific days of the week that the store rents out more DVDs.

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

Due to the nature of the report, daily updates will be needed to keep data fresh. Being said, the data should be refreshed at the very least once a week before the stakeholder's meetings.

B. Write a SQL code that creates the tables to hold your report sections.

Table for holding detailed report section:

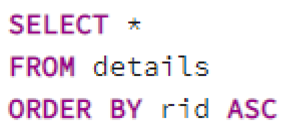
```
DROP TABLE IF EXISTS public.details;
CREATE TABLE IF NOT EXISTS public.details
(
    pid integer NOT NULL primary key,
    rid integer NOT NULL,
    amount numeric(5,2) NOT NULL,
    date timestamp without time zone NOT NULL,
    cid integer NOT NULL,
    fname character varying(100) COLLATE pg_catalog."default" NOT NULL,
    email character varying(300) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT fk_name FOREIGN KEY (cid)
        REFERENCES public.customer (customer_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
);
```

Table for holding summary of the business:

```
DROP TABLE IF EXISTS public.summary;
create table public.summary (
    sId serial primary key,
    date date not null,
    sales float not null,
    shop_id int not null,
    constraint Fk_shopId Foreign Key(shop_id) References store(store_id)
);
```

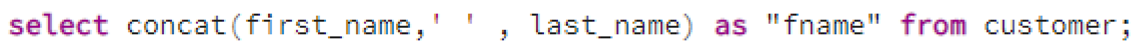
C. Write a SQL query that will extract the raw data needed for the Detailed section of your report from the source database and verify the data’s accuracy.

SQL Query:



```
SELECT *
FROM details
ORDER BY rid ASC
```

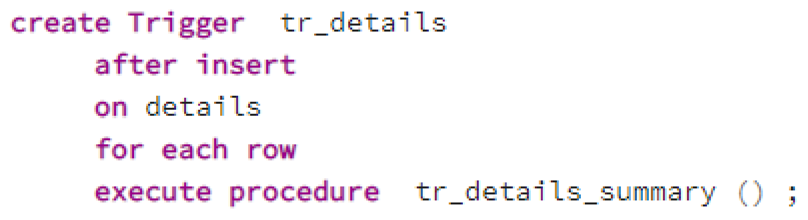
D. Write code for function(s) that perform the transformation(s) you identified in part A4.



```
select concat(first_name,' ' , last_name) as "fname" from customer;
```

The CONCAT function is the function that we used for performing the transformation I identified in the A4 section. This function is used to join two or more columns together.

E. Write a SQL code 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.



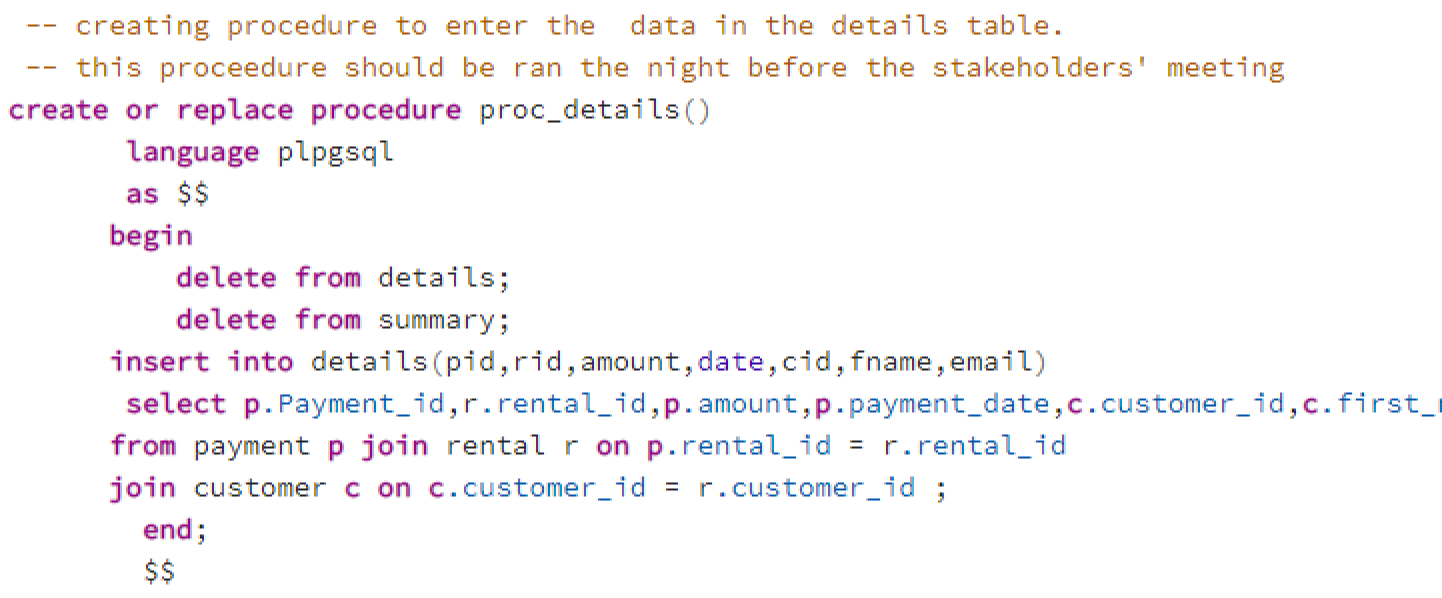
```
create Trigger  tr_details
     after insert
     on details
     for each row
     execute procedure  tr_details_summary () ;
```

F. 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.

```
create or replace function  tr_details_summary ()
returns trigger
language plpgsql
AS
$$
BEGIN

-- if the store id is present then update the amount if store id is not present in the
-- this proceedure should be ran the night before the stakeholders' meeting
IF (select count(*) from summary where shop_id =(
           select distinct store_id
    from details inner join rental on details.rid = rental.rental_id
    join inventory on inventory.inventory_id = rental.inventory_id
    where rid = New.rid) and date = date(New.date)) = 0   Then

    Insert into Summary(date,sales,shop_id) values (date(New.date),New.amount,(select
    from details inner join rental on details.rid = rental.rental_id
    join inventory on inventory.inventory_id = rental.inventory_id
    where rid = New.rid));
       else
    update summary set sales = sales+New.amount where date = date(New.date) and shop_
    from details inner join rental on details.rid = rental.rental_id
    join inventory on inventory.inventory_id = rental.inventory_id
    where rid = New.rid);
    END if;
         return New;
   END;
     $$
```



```
 -- creating procedure to enter the  data in the details table.
 -- this proceedure should be ran the night before the stakeholders' meeting
create or replace procedure proc_details()
       language plpgsql
       as $$
      begin
          delete from details;
          delete from summary;
      insert into details(pid,rid,amount,date,cid,fname,email)
       select p.Payment_id,r.rental_id,p.amount,p.payment_date,c.customer_id,c.first_
      from payment p join rental r on p.rental_id = r.rental_id
      join customer c on c.customer_id = r.customer_id ;
        end;
        $$
```

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

The stored procedure can be run nightly or weekly to ensure it is ready for the stakeholders' meeting to discuss the business revenues. To do this we can use an external utility such as pgAgent (https://www.pgadmin.org/docs/pgadmin4/development/pgagent_jobs.html) or run a cron job to set to run automatically on a schedule.

Sources

Blessy Varghese. (n.d.). *Triggers in postgresql - youtube*. YouTube.com. Retrieved March 30, 2022, from https://www.youtube.com/watch?v=9vn8cRDpEi4

Malik, U., Goldwasser, M., & Johnston, B. (2019). *Sql For Data Analysis: Perform fast and efficient data analysis with the power of Sql*. Packt Publishing.

Page, D. (n.d.). *PgAgent*¶. pgAgent - pgAdmin 4 6.7 documentation. Retrieved March 29, 2022, from https://www.pgadmin.org/docs/pgadmin4/development/pgagent.html

PostgreSQLTutorial.com. (n.d.). PostgreSQL Tutorial. Retrieved March 29, 2022, from https://www.postgresqltutorial.com/

PostgreSQLTutorial.com. (n.d.). *PostgreSQL sample database*. PostgreSQL Tutorial. Retrieved March 29, 2022, from https://www.postgresqltutorial.com/postgresql-sample-database/