**VDM2 — VDM2 Task 1: Data Analysis**

**A.** The real world business report that I would be creating would be a report that shows a list of how many customers the DVD Rental business has per country in their database. I would order the report with the highest customers per country at the top, descending downward.

**1.** The **detailed** table will consist of “customer\_id”, “first\_name”, “last\_name”, “address”, “address2”, “city”, “postal\_code, and “country”. The **summarized** table will consist of “country” and a custom field called “num\_customers”.

**2.** The **detailed** tables data fields will consist of the data of all customers in the database relevant to where they live. The **summarized** table will be the name of every country and the number of customers that are from that particular country.

**3.** The tables being used from the dvdrental dataset are ”**customer**”, “**city**”, “**address**”, and “**country**”.

**4.** From the **detailed** table, I would create a function that would concatenate “address” and “address2” into only one field . I would make this transformation in order to clean up the detailed table and only have ONE address field for easier reading and information navigation.

**5.** The summary table would be able to show this business exactly how their customer base is spread out around the world. If customer count per country is put into descending order, with highest count at the top, the business can then see which country most of their customers reside in. This would allow them to really refine their current marketing and advertising strategies to get the most out of that country’s customer base. This table would also allow them to find and focus on countries where business is not so popular in order to spread their customer base or to “tap into” certain countries possible customer bases. The detailed table could be used to focus on more specific areas of certain countries in order to really pinpoint marketing and advertising strategies even in particular postal codes if needed. It could also be used to see information like names and addresses about any and all customers that reside in said particular area.

**6.**  In order to stay most relevant to stakeholders, I would refresh the reports every time a new customer is added to the database as long as it didn’t bog down the database too much. If the company became very popular like Netflix and thousands of new customers were added each day, I would most likely have them refresh the reports only once a day.

**B.**

create function combine\_address(address text, address2 text)

returns text

language plpgsql

as

$$

declare

new\_address text;

begin

-- Set new Address

new\_address = address || ' ' || address2;

-- return it

return new\_address;

end;

$$;

**C:**

create table detailed\_report (

customer\_id INT UNIQUE NOT NULL,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

address VARCHAR(50) NOT NULL,

address2 VARCHAR(50),

city VARCHAR(50) NOT NULL,

postal\_code VARCHAR(10),

country VARCHAR (50) NOT NULL

);

create table summarized\_report (

num\_customers int NOT NULL,

country VARCHAR (50) NOT NULL

);

**D:**

insert into detailed\_report

select customer.customer\_id, customer.first\_name, customer.last\_name, address.address, address.address2, city.city, address.postal\_code, country.country

from customer, address, city, country

where customer.address\_id = address.address\_id

and address.city\_id = city.city\_id

and city.country\_id = country.country\_id

order by country.country\_id;

**E:**

create function update\_summarized\_table()

returns trigger

language plpgsql

as

$$

begin

-- Delete all summarized table info..

delete from summarized\_report;

-- Redo table with new counts..

insert into summarized\_report

select count(detailed\_report.country) as numCustomers, detailed\_report.country

from detailed\_report

group by detailed\_report.country

order by numCustomers desc;

return new;

end;

$$;

create trigger update\_summarized\_table\_trigger

after insert

on detailed\_report

for each statement

execute procedure update\_summarized\_table()

**F:**

create procedure refresh\_reports()

language plpgsql

as $$

begin

-- clear both tables..

delete from detailed\_report;

delete from summarized\_report;

-- Add data into detailed report..

insert into detailed\_report

select customer.customer\_id, customer.first\_name, customer.last\_name, address.address, address.address2, city.city, address.postal\_code, country.country

from customer, address, city, country

where customer.address\_id = address.address\_id

and address.city\_id = city.city\_id

and city.country\_id = country.country\_id

order by country.country\_id;

-- now tranform into summarized report..

insert into summarized\_report

select count(detailed\_report.country) as numCustomers, detailed\_report.country

from detailed\_report

group by detailed\_report.country

order by numCustomers desc;

-- Commit

commit;

end;

$$;

**1.** To automate stored procedures with PGAdmin4 you could use an extension like pgAgent or pg\_cron.

**H.** Sources:

- **Site:** <https://www.postgresqltutorial.com/>

- **Site:** <https://severalnines.com/>

- **Book:** SQL for Data Analytics : Perform Fast and Efficient Data Analysis with the Power of SQL by Upom Malik, , Matt Goldwasser, and Benjamin Johnston