### Describe The Dataset

We have a dvdrental database. Inside of this database we 15 tables providing us information on customers, staff, stores, dvds etc… I will be using tables “film”, “rental”, “inventory”. I am going to aggregate the information from these tables to show the following.

* Most Popular Overall DVD
* Least Popular Overall DVD
* Rental Trend

I will be converting the field rental.rental\_date from a 24hr format to a 12hr format.

### Business Use Cases

Using this the detailed section of the report we will be able to make better educated decisions on which dvds to stock the inventory. Using the summary section of the report they will see the most popular dvd which will let them stock there inventory with the most popular dvd thus increasing rentals.

### Refreshing the Report

The report should be refreshed weekly.

### Real World Report

Now that we have enough information about our data, I have decided to create a summary report of the top 10 performing rentals of the month and if they are family friendly or not. The company will now know what kind of inventory to keep in stock. To achieve this report, I will need to merge data from the Inventory, Rental, and Film tables into a new table called all\_rentals, this will be our detailed table. I will also create an additional table called top\_rentals which will grab data from the detailed table and create an summary report.

### Report SQL

**---CREATE TABLES-------------------------------------------------------------------------------------------------------------**

**----------------------------------------------CREATE DETAILED TABLE ------------------------------------------------------------**

**DROP TABLE IF EXISTS detailedreport;**

**CREATE TABLE detailedreport (**

**customer\_id integer,**

**first\_name varchar (45),**

**last\_name varchar (45),**

**email varchar(90),**

**rental\_id integer,**

**rental\_date timestamp,**

**return\_date timestamp,**

**staff\_id integer**

**);**

**-- This creates an empty table titled detailed that will hold the information for the**

**-- detailed section of the business report**

**SELECT \***

**FROM detailedreport;**

**-- To veiw the empty detailedreport table**

**----------------------------------------------CREATE EMPTY SUMMARY TABLE--------------------------------------------------**

**CREATE TABLE summaryreport(**

**customer\_name varchar(95),**

**email varchar(90),**

**customer\_count integer**

**);**

**-- This creates an empty table titled summary that will hold the information for the**

**-- summary section of the business report**

**SELECT \***

**FROM detailedreport;**

**-- To veiw the emty detailedreport table**

**------ INSERTING DATA INTO THE DETAILED TABLE --------------------------------------------------------------------------------------------**

**INSERT INTO detailedreport(**

**customer\_id, -- customer**

**first\_name, -- customer**

**last\_name, -- customer**

**email, -- customer**

**rental\_id, -- rental**

**rental\_date, -- rental**

**return\_date, -- rental**

**staff\_id --rental**

**)**

**SELECT**

**c.customer\_id, c.first\_name, c.last\_name, c.email,**

**r.rental\_id, r.rental\_date, r.return\_date, r.staff\_id**

**FROM rental AS r**

**INNER JOIN customer AS c ON c.customer\_id = r.customer\_id;**

**--- This will load the information from the customer and rental tables into the detailed**

**-- table,**

**SELECT \***

**FROM detailedreport;**

**-- To view the now filled detailed table**

**-- CREATING THE FUNCTION a-------------------------------------------------------------------------------------------------**

**---------------------Updating the summary table-----------------------------------------------------------------**

**CREATE FUNCTION summary\_refresh\_fun()**

**RETURNS TRIGGER**

**LANGUAGE plpgsql**

**AS $$**

**BEGIN**

**DELETE FROM summaryreport;**

**-- this will empty the summaryreport table.**

**INSERT INTO summaryreport (**

**SELECT**

**concat\_ws (', ', last\_name, first\_name) AS customer\_name,**

**email,**

**COUNT(customer\_id)**

**FROM detailedreport**

**GROUP BY customer\_id, customer\_name, email**

**-- HAVING count(customer\_id) > 30**

**ORDER BY count(customer\_id)DESC**

**LIMIT 100**

**);**

**RETURN NEW;**

**END; $$**

**--This function will refresh the summary table with data taken from the detailed table**

**-- CREATING THE TRIGGER---------------------------------------------------------------------------------------------**

**CREATE TRIGGER summary\_refresh**

**AFTER INSERT ON detailed**

**FOR EACH STATEMENT**

**EXECUTE PROCEDURE summary\_refresh\_fun();**

**--This trigger will execute the function that I just created summary\_refresh\_fun()**

**-- when there is an insert into the detailed function.**

**--Lastly, Let's create the procedure.**

**--CREATING THE PROCEDURE -----------------------------------------------------------------------------------------**

**CREATE PROCEDURE refresh\_tables()**

**LANGUAGE plpgsql**

**AS $$**

**BEGIN**

**DELETE FROM detailedreport; -- this will empty the detailed table on any existing info.**

**INSERT INTO detailedreport(**

**customer\_id, -- customer**

**first\_name, -- customer**

**last\_name, -- customer**

**email, -- customer**

**rental\_id, -- rental**

**rental\_date, -- rental**

**return\_date, -- rental**

**staff\_id --rental**

**)**

**SELECT**

**c.customer\_id, c.first\_name, c.last\_name, c.email,**

**r.rental\_id, r.rental\_date, r.return\_date, r.staff\_id**

**FROM rental AS r**

**INNER JOIN customer AS c ON c.customer\_id = r.customer\_id;**

**--reinserting the new data into the detailed table**

**END;$$**

**-- To call stored procedure**

**-- CALL refresh\_tables();**

**-- To view results**

**-- SELECT \***

**FROM detailedreport;**

**-- SELECT \***

**FROM summaryreport;**