**VDM1 – TASK 1: AUTOMATING DATA INTEGRATION**

**Advanced Data Management – D191**

**James Blankenship**

**#003462522**

**Value Statement:**

We at Albany DVD Rentals value our customers, and we

have asked you to find our top 10 customers that have spent the most cash on DVD rentals so we can send them a gift card to our store.

**Your business question must rely on the aggregation of data. In this assessment, the summary data will be automatically created from the detailed data.**

**Plan for and compose the sections of a real-world business report that can be created from the DVD Database on Labs on Demand (see link below), and demonstrate the functionality of the supporting SQL code by doing the following:**

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

**1.**  **Describe the data used for the report.**

The data used to answer the business question will include data about the customers, like their customer id, first name, last name, email address and creation date. There will also be film data, including the films rental rate. The data in the tables will be linked together by customer id, film id, and inventory id.

**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 tables that will provide the needed data will consist of the customer, rental, film, and inventory tables.

**3.  Identify the specific fields that will be included in the**

**detailed and the summary sections of the report.**

The detailed table will include the id, full name, amount spent, email address, and creation date of the customers. It will also include a column indicating if the customer is one of the top 10 spenders.

The summary table will include the full name, amount spent, and email address of the customers.

**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 fields of the top\_10 spenders’ column with the value ‘Y’ will be translated into ‘Yes’, and the fields with the value ‘N’ will be translated into ‘No’. Doing so will make the table easier to understand and draw information from. This will help the less experienced users.

**5.  Explain the different business uses of the detailed**

**and the summary sections of the report.**

The detailed section will contain information such as the customers id, total amount spent, the customer’s name and email address, creation date of the account, and whether the customer is a top spender. This information can be used to perform analytics on the data to foreshadow a trend between creation date and amount spent.

The summary section will contain only the simple information, like the customer’s name and email address, and their total amount spent. This information can be used to see who spent the most money so management can contact the customer and send them rewards via email. This can encourage valuable relationships between the management and their customers.

**6.  Explain how frequently your report should be**

**refreshed to remain relevant to stakeholders.**

The report should be updated when management feels the

need to grant rewards to top spenders. This can be done to

encourage customers to spend more at the store, or show

customers that the store’s management care about their

service. I recommend updating the report regularly if

management plans on rewarding customers in a timely fashion.

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

CREATE TABLE detailed (

Customer\_id INT PRIMARY KEY,

Full\_name VARCHAR(40),

Amount\_spent FLOAT,

Email\_address VARCHAR(40),

Creation\_date DATE,

Top\_10 VARCHAR(3)

);

CREATE TABLE summary (

Full\_name VARCHAR(40),

Amount\_spent FLOAT,

Email\_address VARCHAR(40)

);

1. **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.**

INSERT INTO detailed

SELECT c.customer\_id,

CONCAT(c.first\_name, ‘ ‘, c.last\_name)

AS full\_name,

SUM(f.rental\_rate) AS amount\_spent,

c.email AS email\_address,

c.create\_date AS creation\_date,

CASE

WHEN SUM(f.rental\_rate) > 118 THEN ‘Y’

ELSE ‘N’

END

AS top\_10

FROM customer c

JOIN rental r

ON r.customer\_id = c.customer\_id

JOIN inventory i

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

JOIN film f

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

GROUP BY c.customer\_id

ORDER BY amount\_spent DESC;

I verified the data’s accuracy by running the query below. I looked for any empty or invalid cells in the table, as well as spot checked the columns to make sure the rows data types matched the column’s data types.

SELECT \* FROM detailed;

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

CREATE FUNCTION char\_to\_str()

RETURNS VARCHAR(3) AS $str$

DECLARE str VARCHAR(3);

BEGIN

UPDATE detailed

SET top\_10 =

CASE

WHEN top\_10 = ‘Y’ THEN ‘Yes’

WHEN top\_10 = ‘N’ THEN ‘No’

WHEN top\_10 = ‘Yes’ THEN ‘Yes’

WHEN top\_10 = ‘No’ THEN ‘No’

END

WHERE amount\_spent > 0;

SELECT top\_10 INTO str FROM detailed;

RETURN str;

END; $str$

LANGUAGE PLPGSQL;

I verified the accuracy of the function by running these queries below.

SELECT char\_to\_str();

SELECT \* FROM detailed ORDER BY amount\_spent DESC;

1. **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 FUNCTION update\_summary()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO summary

VALUES (NEW.full\_name, NEW.amount\_spent,

NEW.email\_address);

RETURN NEW;

END; $$

LANGUAGE PLPGSQL;

CREATE TRIGGER update\_trigger

AFTER INSERT ON detailed

FOR EACH ROW

EXECUTE PROCEDURE update\_summary();

I verified that the trigger worked correctly by running the queries below.

INSERT INTO detailed

VALUES (‘777’, ‘John Doe’, 0.0, ‘j.d@gmail.com’, ‘2-19-2017’, ‘No’);

SELECT \* FROM 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 PROCEDURE refresh\_tables()

LANGUAGE SQL

AS $$

DELETE FROM detailed;

DELETE FROM summary;

INSERT INTO detailed

SELECT c.customer\_id,

CONCAT(c.first\_name, ‘ ‘, c.last\_name)

AS full\_name,

SUM(f.rental\_rate) AS amount\_spent,

c.email AS email\_address,

c.create\_date AS creation\_date,

CASE

WHEN SUM(f.rental\_rate) > 118 THEN ‘Y’

ELSE ‘N’

END

AS top\_10

FROM customer c

JOIN rental r

ON r.customer\_id = c.customer\_id

JOIN inventory i

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

JOIN film f

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

GROUP BY c.customer\_id

ORDER BY amount\_spent DESC;

SELECT char\_to\_str();

$$;

I verified that the stored procedure worked correctly by running the queries below.

CALL refresh\_tables();

SELECT \* FROM summary;

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

The stored procedure should be executed when management feels the need to grant rewards to top spenders. It needs to be executed before calculation of the top spenders, so the calculations are accurate and relevant. I recommend updating the report regularly every couple of weeks.

The stored procedure could be run on a schedule multiple ways, but the ways I recommend would be writing a trigger to call the stored procedure every 2 weeks or set up a schedule in windows scheduler to call the stored procedure every 2 weeks. This would allow the tables to remain up to date.

**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.**

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

No web sources or segments of third-party code were used for this project.