For this Project I am assuming the role as an owner of a DVD rental Shop. Since this is a business, the profitability and sales of the business provides an upper-level view of the business. However, due to pending financial data, I am limiting to reviewing business performance based on ***Sales made by Location***

**Sales made by location**

As a business owner, one of the most crucial decisions that an owner can make is to decide if the location provides a necessary stream of revenue and one of the factors that can be determined to cause that is location. Locations determine whether or not a business can be profitable or not simply by the people living at that location. So, if a location has people who are mostly in the age category who are 50 years old or older, then most likely the outcome is that they would prefer films who are classics and would prefer less Science Fiction films. This can result in decisions made to adjust the inventory of films that are less science fiction genre.

Another use of this table is to see the Sales by Film Genre and the dates. We can see which location has which Genre of Films that are being rented out. Certain dates can tell when renters are willing to rent out and with that a pattern can be determined for having appropriate inventory of films and if demand for certain Genre on certain days.

The Two tables being created are below:

***sales\_location\_detailed***: which provides the sales data in detailed by Genre and location where the individual sales by location which uses two fields “city.city” and “country.country” columns concatenated with spaces. The columns are taken from the City table and the Country table.

The table is made from 11 tables using a JOIN statement with the main table being the “Payment” Table, the other tables that are joined are from the “Rental”, “Inventory”, ”Film”, ”Film\_Category”, “Category”, “Staff”, “Store”, “Address”, “City” and “Country” Tables which have been joined using the respective Primary and Foreign keys (Please see code).

The main fields that are being utilized are the following:

* City (City Table) Concatenated with “, “ and Country (Country table) columns named as *city\_address*
* *Payment\_ID* (Payment Table)
* *Rental\_ID* (Payment Table)
* *Payment\_Date* (Payment Table) that using CAST() has been converted to a DATE Format making it easier to read what date the payment was made instead of a TIMESTAMP format
* *Inventory\_ID* (Rental Table)
* Name (Category Table) which was renamed as “Genre” for it to be easier to identify respective film category as *Genre*
* *Title* (Film Table) which is the name of the Films
* *Sale* (Payment Table) which was using the CAST() function converted to MONEY as currency format is a financial view that is easy to understand revenue to stakeholders and when presenting to view at a high level view having a money format helps in reviewing numbers clearly

***sales\_location\_summary****:* The Summary view provides the business owners a quick glance of how much revenue each location made. As a business owner I would use such a report first and compare location revenue to see under or over performing and based on that next decision to review further can be taken. All the information is pulled from the “***sales\_location\_detailed***” table summarizes the data to be presentable to the owner

This view only requires two Fields

* *Location* which takes in the city\_address from sales\_by\_location table
* *Sales* (Payment Table Column) using the SUM() function location was summed up and then using the CAST() was converted to MONEY Format as the “$” makes it easier financially to identify money related transactions

In both tables the main field that is modified is the Sale(s) Field since numbers represented in Money format provide the Owner and stakeholders an easy view to differentiate between numbers that are related to Money and that are not.

The Second field that is also found in both tables is the City and Country fields that are concatenated together with a comma as a delimiter. Having the city and Country together provides the stakeholders with an idea of which city and country in case some are not aware where it is located and is easier to understand.

The Detailed table provides the data of all rental payments and transactions made by film Genre and date as well as the location. This provides the stakeholders especially the owner the ability to see which location rental is being carried out the most and the Genre. Certain locations would provide the owner an idea what the customers by location prefer to watch and accordingly inventory of certain films can then be adjusted to meet the needs of the location as well as the date provides when films are rented out the most days of the week with the date.

The Summary Table provides a quick view which usually what as a business owner would want since it helps make a decision if more stores need to be added or closed to a certain location. Based on that further review of the data can then be made to take appropriate actions.

This report should be refreshed at least once every month to see the performance of the location and based on that actions can be take if certain genre of films are not performing well, then do they require promotions or advertisement of films to attract new customers or existing customers to try new genre’s.

The Store procedure for this data can be run on adhoc basis, which can be minimum on a monthly basis or whenever urgently required. However, in order to provide a more accurate up to date information, the old data needs to be removed and replaced with the new data. Before presenting it to the stakeholders, the information can be updated for both tables to ensure up-to-date and accurate data.

1. -- Create tables

2. -- Here we create the detailed table

3. DROP TABLE IF EXISTS Sales\_location\_Detailed;

4. CREATE TABLE IF NOT EXISTS Sales\_location\_Detailed (

5. detail\_id SERIAL PRIMARY KEY,

6. location VARCHAR(255),

7. payment\_id integer,

8. rental\_id integer,

9. payment\_date timestamp,

10. inventory\_id integer,

11. genre VARCHAR(255),

12. title TEXT,

13. Sale numeric

14. );

15. -- NOW TO CREATE THE SUMMARY TABLE OF THE SALES BY LOCATION

16.

17. CREATE TABLE IF NOT EXISTS Sales\_Location\_Summary (

18. summary\_ID SERIAL PRIMARY KEY,

19. location VARCHAR(255),

20. Sales numeric

21. );

22. -- NOW WE INSERT DATA INTO THE SALES BY LOCATION DETAILED TABLE

23.

24. INSERT INTO Sales\_Location\_Detailed (

25. location, -- THIS IS A COMBINATION OF CITY AND COUNTRY TABLE

26. payment\_id, -- This is the Payment Table

27. rental\_id, -- This is from the rental table

28. payment\_date, -- This is from the Payment Table

29. inventory\_id, -- This is from the Rental Table

30. genre, -- This is from the Category Table

31. title, -- This is from the Film Table

32. Sale -- This is from the Rental Table

33. )

34. SELECT

35. city.city||', '||country.country AS city\_address,

36. payment.payment\_id,

37. payment.rental\_id,

38. CAST(payment.payment\_date as DATE),

39. rental.inventory\_id,

40. category.name AS Genre,

41. film.title,

42. CAST(payment.amount AS MONEY)

43. FROM payment

44. INNER JOIN rental ON rental.rental\_id = payment.rental\_id

45. INNER JOIN inventory ON rental.inventory\_id= inventory.inventory\_id

46. INNER JOIN film ON inventory.film\_id = film.film\_id

47. INNER JOIN film\_category ON film\_category.film\_id = film.film\_id

48. INNER JOIN category ON category.category\_id = film\_category.category\_id

49. INNER JOIN staff ON payment.staff\_id = staff.staff\_id

50. INNER JOIN store ON payment.staff\_id = store.manager\_staff\_id

51. INNER JOIN address ON store.address\_id=address.address\_id

52. INNER JOIN city ON city.city\_id =address.city\_id

53. INNER JOIN country ON city.country\_id = country.country\_id

54. GROUP BY category.name, payment.payment\_id, rental.inventory\_id, city.city, country.country, film.title

55. ORDER BY city\_address DESC;

56. -- This information is a combination from Rental, Inventory, Film, Film\_Category, Category, Staff, Store, City and Country Table

57. -- With this a combined view information is retrieved and added into the table

58.

59. -- NOW WE CAN VIEW THE Sales\_location\_Detailed table WITH THE DATA ADDED

60. CREATE FUNCTION summary\_data\_refresh()

61. RETURNS TRIGGER

62. LANGUAGE plpgsql

63. AS $$

64. BEGIN

65. -- OLD DATA NEEDS TO BE CLEARED OUT SO DATA IS FIRST CLEARED OUT

66. DELETE FROM Sales\_Location\_Summary;

67. -- NEW DATA IS THEN INSERTED

68. INSERT INTO Sales\_Location\_Summary (

69. location,

70. Sales

71. )

72. SELECT

73. c.location,

74. SUM(c.sale)

75. FROM sales\_location\_detailed AS c

76. GROUP BY c.location;

77. RETURN NEW;

78. END

79. $$;

80. -- WE NOW CREATE THE TRIGGER FUNCTION

81. CREATE TRIGGER refreshing\_data

82. AFTER INSERT ON Sales\_location\_Detailed

83. FOR EACH STATEMENT

84. EXECUTE PROCEDURE summary\_data\_refresh();

85.

86. -- PROCEDURE NEEDS TO BE CREATED TO REFRESH THE DETAILED TABLE AND THUS ALSO REFRESHING THE SUMMARY TABLE

87.

88. CREATE PROCEDURE refresh\_data()

89. LANGUAGE plpgsql

90. AS $$

91. BEGIN

92. -- We need to renew the data so first we empty the existing data in the table

93. DELETE FROM sales\_location\_detailed;

94. -- WE HAVE TO RE-DO THE DATA INSERTS INTO THE DETAILED TABLE TO HAVE THE DETA PRESENTED IN THE SUMMARY TABLE

95. INSERT INTO Sales\_Location\_Detailed (

96. location, -- THIS IS A COMBINATION OF CITY AND COUNTRY TABLE

97. payment\_id, -- This is the Payment Table

98. rental\_id, -- This is from the rental table

99. payment\_date, -- This is from the Payment Table

100. inventory\_id, -- This is from the Rental Table

101. genre, -- This is from the Category Table

102. title, -- This is from the Film Table

103. Sale -- This is from the Rental Table

104. )

105. SELECT

106. city.city||', '||country.country AS city\_address,

107. payment.payment\_id,

108. payment.rental\_id,

109. CAST(payment.payment\_date as DATE),

110. rental.inventory\_id,

111. category.name AS Genre,

112. film.title,

113. CAST(payment.amount AS MONEY)

114. FROM payment

115. INNER JOIN rental ON rental.rental\_id = payment.rental\_id

116. INNER JOIN inventory ON rental.inventory\_id= inventory.inventory\_id

117. INNER JOIN film ON inventory.film\_id = film.film\_id

118. INNER JOIN film\_category ON film\_category.film\_id = film.film\_id

119. INNER JOIN category ON category.category\_id = film\_category.category\_id

120. INNER JOIN staff ON payment.staff\_id = staff.staff\_id

121. INNER JOIN store ON payment.staff\_id = store.manager\_staff\_id

122. INNER JOIN address ON store.address\_id=address.address\_id

123. INNER JOIN city ON city.city\_id =address.city\_id

124. INNER JOIN country ON city.country\_id = country.country\_id

125. GROUP BY category.name, payment.payment\_id, rental.inventory\_id, city.city, country.country, film.title

126. ORDER BY city\_address DESC;

127.

128. END$$;

129.

130. -- THIS PROCEDURE WILL THEN LEAD TO THE DATA BEING REFRESHED AND WILL TRIGGER THE OTHER FUNCTION

131.

132. -- THIS WILL BE CALLING THE MAIN TABLE

133. CALL refresh\_data();

134.

135. -- NOW YOU CAN VIEW THE NEW DATA HERE

136. SELECT \* FROM Sales\_Location\_Detailed;

137.

138. -- THE SUMMARY DATA CAN BE VIEWED HERE

139. SELECT \* FROM Sales\_Location\_Summary;

140.

141.

142.