Project

Daxeshkumar Patel

041086185

Algonquin College of Applied Arts and Technology

24W\_CST2102\_300 Database Analytics

Professor Jim Myronyk

April 3, 2024

**TABLE OF CONTENT**

**PART 1: FOUR SQL QUERIES USING THE FEHILY DATABASE FROM CLASS**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Description** | **Page No.** |
| **1.1** | List the top 2 revenue generating for each publisher. | 3 |
| **1.2** | Use a subquery to count the number of books sold in the month closest to Christmas by each genre | 4 |
| **1.3** | Use the SUM(), RANK(), and LAG() analytic windowing functions with partitions to develop a meaningful query | 5 |
| **1.4** | Improve the performance of 1 of the above queries. Explain your approach and support the results with before and after explain plan results | 6 |

**PART 2: EXPORT ORACLE DATA FROM ORACLE TO MS POWER BI AND MYSQL**

|  |  |  |
| --- | --- | --- |
| **2.1** | Export the Fehily data from your user ‘dax’ schema and load it into MS Power BI. | 7 |
| **2.2** | Reproduce the query results from Part1 in PBI | 8 |
| **2.3** | Export the Fehily data from your user ‘dax’ schema and load it into MySQL | 10 |
| **2.4** | Reproduce 2 query results from Part1 in MySQL | 11 |

**PART 1:** **FOUR SQL QUERIES USING THE FEHILY DATABASE FROM CLASS**

**1. List the top 2 revenue generating authors (i.e., author id, author concatenated name, title\_id, book revenue, total author revenue) for each publisher.**

**EXPLANATION:-**

Main requirement our here is to show the top 2 revenue generating authors in the table having columns for author id, author concatenated name, title\_id, book revenue, total author revenue. For doing so I have calculated the revenue **(t.price \* t.sales) and alias it as a Revenue.** Then, to short revenue in **descending order and use Rank to give ranking** to each raw and use join to connect the tables. After that as required top 2 revenue generators for every publisher, I have partitioned the tanking by the pub\_id. To show the result in requested format use **SELECT SELECT au\_id, name, title\_id, pub\_id, Revenue, Rank and** use **WHERE** clause on **rank < =2 to filter the results further.**

**QUERY: -**

SELECT au\_id, name, title\_id, pub\_id, Revenue, Rank

FROM

(

SELECT

a.au\_id, a.fname || ' ' || a.lname AS name, at.title\_id, p.pub\_id,SUM(t.price \* t.sales) AS Revenue,

RANK() OVER (PARTITION BY p.pub\_id ORDER BY SUM(t.price \* t.sales) DESC) AS Rank

FROM authors a

JOIN author\_titles at ON a.au\_id = at.au\_id

JOIN titles t ON at.title\_id = t.title\_id

JOIN publishers p ON t.pub\_id = p.pub\_id

GROUP BY a.au\_id, a.fname, a.lname, at.title\_id, p.pub\_id

)

WHERE Rank <= 2;

A screenshot of a computer

Description automatically generated**OUTPUT: -**

**2.Use a subquery to count the number of books sold in the month closest to Christmas by each genre.**

**EXPLANATION:-**

To achieve desired result It is mandatory to ger the details of books sold and filter the month from the date format. Hence, extracted the Month from the Date format of TITLES table and consider the total count as books sold per month by using **COUNT(\*).** Then convert the months in to ascending order by using **EXTRACT(MONTH FROM MIN(PUBDATE)) ASC** for getting better understanding.

**QUERY:-**

SELECT

Genre, TO\_CHAR(PUBDATE,'MONTH') AS Month,

COUNT(\*) as Books\_Sold

FROM

TITLES t

WHERE

EXTRACT (MONTH FROM PUBDATE) = (

SELECT MAX(EXTRACT(MONTH FROM PUBDATE))

FROM TITLES

WHERE Genre = t.genre )

GROUP BY

Genre, TO\_CHAR(PUBDATE,'MONTH')

ORDER BY

EXTRACT(MONTH FROM MIN(PUBDATE)) ASC;

**OUTPUT: -**

A screenshot of a computer

Description automatically generated

**3. Use the SUM(), RANK(), and LAG() analytic windowing functions with partitions to develop a meaningful query**

**EXPLANATION:-**

* The **SUM(Sales)** OVER (PARTITION BY Genre) calculates the total sales within each genre and sums up the sales values within that partitions itself.
* The **RANK()** OVER (PARTITION BY Genre ORDER BY SUM(Sales) DESC) give rank to each title within its genre based on the total sales. The higher the total sales, lover the rank would be.
* The **LAG(Sales)** OVER (PARTITION BY Genre ORDER BY Title\_id) function retrieves the sales value of the previous title within the same genre.
* Belo query provides details of the total sales for each genre, rank titles based on their sales performance within each genre and compare the sales of each title with the previous titles within the same genre.

**QUERY: -**

SELECT

Title\_id, Title, Genre, SUM(Sales) OVER (PARTITION BY Genre) AS Total\_Sales,

RANK() OVER (PARTITION BY Genre ORDER BY SUM(Sales) DESC) AS Rank, Sales,

LAG(Sales) OVER (PARTITION BY Genre ORDER BY Title\_id) AS Previous\_Sales

FROM Titles

GROUP BY Title\_id, Title, Genre, Sales;

**OUTPUT: -**

A screenshot of a computer

Description automatically generated

**4. Improve the performance of 1 of the above queries. Explain your approach and support the results with before and after explain plan results.**

**Explanation: -**

**1.Subquery for Maximum genre per month:** In place of recalculating the maximum month for each genre in the WHERE clause, calculate it once for each genre by using a subquery (genre\_mx\_month).

**2.Join with Main Table:** Join the main TITLES table with the result of the subquery based on the genre. This has ensured that we only perform the max month calculation one time only.

**QUERY: -**

SELECT t.Genre, TO\_CHAR(t.PUBDATE, 'MONTH') AS Month, COUNT(\*) AS Books\_Sold

FROM TITLES t

JOIN (

SELECT Genre, MAX(EXTRACT(MONTH FROM PUBDATE)) AS Mxmonth

FROM TITLES

GROUP BY Genre )

genre\_mx\_month ON t.Genre = genre\_mx\_month.genre

WHERE EXTRACT(MONTH FROM t.PUBDATE) = genre\_mx\_month.Mxmonth

GROUP BY t.Genre, TO\_CHAR(t.PUBDATE, 'MONTH')

ORDER BY EXTRACT(MONTH FROM MIN(PUBDATE)) ASC;

**OUT** **PUT: -**

A screenshot of a computer

Description automatically generated

**PART 2:** **EXPORT ORACLE DATA FROM ORACLE TO MS POWER BI AND MYSQL**

**2.1. Export the Fehily data from your user ‘dax’ schema and load it into MS Power BI.**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**2.2. Reproduce the query results from Part1 in PBI**

**2.2.1.List the top 2 revenue generating authors (i.e., author id, author concatenated name, title\_id, book revenue, total author revenue) for each publisher.**

**OUTPUT: -**

A screenshot of a computer

Description automatically generated

**2.2.2.Use a subquery to count the number of books sold in the month closest to Christmas by each genre.**

**OUTPUT: -**

**A graph of a graph

Description automatically generated with medium confidence**

**2.2.3. Use the SUM(), RANK(), and LAG() analytic windowing functions with partitions to**

**develop a meaningful query**

**OUTPUT: -**

A screenshot of a computer

Description automatically generated

**2.3. Export the Fehily data from your user ‘dax’ schema and load it into MySQL.**

A screenshot of a computer

Description automatically generated

**2.4.Reproduce 2 query results from Part1 in MySQL.**

**2.4.1.Use a subquery to count the number of books sold in the month closest to Christmas by each genre.**

**QUERY:-**

SELECT

Genre, MONTHNAME(PUBDATE) AS Month,

COUNT(\*) as Books\_Sold

FROM

TITLES t

WHERE

MONTH(PUBDATE) =

(

SELECT MAX(MONTH(PUBDATE))

FROM TITLES

WHERE Genre = t.genre

)

GROUP BY

Genre, MONTHNAME(PUBDATE)

ORDER BY

MONTH(MIN(PUBDATE)) ASC;

**OUTPUT: -**

A screenshot of a computer

Description automatically generated

**2.4.2. Use the SUM(), RANK(), and LAG() analytic windowing functions with partitions to develop a meaningful query**

**QUERY: -**

SELECT

Title\_id, Title, Genre,

SUM(Sales) OVER (PARTITION BY Genre) AS Total\_Sales,

RANK() OVER (PARTITION BY Genre ORDER BY Sales DESC) AS `Rank`,

Sales,

LAG(Sales) OVER (PARTITION BY Genre ORDER BY Title\_id) AS Previous\_Sales

FROM Titles;

**OUTPUT: -**

A screenshot of a computer

Description automatically generated