**ST1501 CA2 Group Task**

**Class:** DAAA/FT/2A/01

**Group No**: Team Bryan

**Group Members:**

|  |  |
| --- | --- |
| **AdmNo** | **Name** |
| 2214449 | Tan Wen Tao Bryan |
| 2201861 | Zachary Leong |
| 2214452 | Ryan Yeo |
| 2214241 | Ethan Pang |

**Group Task**

**OLTP database Diagram**

|  |
| --- |
| **A screenshot of a computer  Description automatically generated** |

**Data Warehouse Schema**

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

**Queries explanation**

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Explanation | SQL script | SQL Results |
| 1 | Identify the top 3 earners and lowest 3 earners based on the total sales amount, retrieve percentage of sales for each employee as 2 separate subqueries. Results of 2 subqueries are joined with UNION ALL | SELECT  'Top Earner' AS Earner\_Type,  q1.Staff\_Name,  q1.Total\_Sales\_Amount,  q1.Percentage\_of\_Total\_Sales  FROM  (SELECT  e.First\_Name + ' ' + e.Last\_Name AS Staff\_Name,  SUM(sf.Unit\_Price \* sf.Quantity) AS Total\_Sales\_Amount,  CAST((SUM(sf.Unit\_Price \* sf.Quantity) \* 100.0 / t.Total\_Sales) AS decimal(7, 2)) AS Percentage\_of\_Total\_Sales  FROM  Sales\_Fact sf  INNER JOIN Employees\_DIM e ON sf.Employee\_ID = e.Employee\_ID  CROSS JOIN (  SELECT SUM(sf2.Unit\_Price \* sf2.Quantity) AS Total\_Sales  FROM Sales\_Fact sf2  ) t  GROUP BY  e.First\_Name,  e.Last\_Name,  t.Total\_Sales  ORDER BY  Total\_Sales\_Amount DESC  OFFSET 0 ROWS  FETCH NEXT 3 ROWS ONLY) AS q1  UNION ALL  SELECT  'Lowest Earner' AS Earner\_Type,  q2.Staff\_Name,  q2.Total\_Sales\_Amount,  q2.Percentage\_of\_Total\_Sales  FROM  (SELECT  e.First\_Name + ' ' + e.Last\_Name AS Staff\_Name,  SUM(sf.Unit\_Price \* sf.Quantity) AS Total\_Sales\_Amount,  CAST((SUM(sf.Unit\_Price \* sf.Quantity) \* 100.0 / t.Total\_Sales) AS decimal(7, 2)) AS Percentage\_of\_Total\_Sales  FROM  Sales\_Fact sf  INNER JOIN Employees\_DIM e ON sf.Employee\_ID = e.Employee\_ID  CROSS JOIN (  SELECT SUM(sf2.Unit\_Price \* sf2.Quantity) AS Total\_Sales  FROM Sales\_Fact sf2  ) t  GROUP BY  e.First\_Name,  e.Last\_Name,  t.Total\_Sales  ORDER BY  Total\_Sales\_Amount ASC  OFFSET 0 ROWS  FETCH NEXT 3 ROWS ONLY) AS q2; | A screenshot of a computer  Description automatically generated  Insight: Top Earner is Freya Gomez who made sales over $800000 which made up 15.32% of the total sales, lowest earner is Sophia Reynolds who made sales of $336727 which is 0.64% of the total sales. |
| 2 | Retrieves the total yearly earnings, identifies the best and worst sales month with the respective sales amount for each year | SELECT  Sales\_Year,  Total\_Yearly\_Earnings,  DATENAME(MONTH, DATEADD(MONTH, Best\_Sales\_Month - 1, '2000-01-01')) AS Best\_Sales\_Month,  Best\_Month\_Sales\_Amount,  DATENAME(MONTH, DATEADD(MONTH, Worst\_Sales\_Month - 1, '2000-01-01')) AS Worst\_Sales\_Month,  Worst\_Month\_Sales\_Amount  FROM (  SELECT  Sales\_Year,  SUM(Monthly\_Sales\_Amount) AS Total\_Yearly\_Earnings,  MAX(CASE WHEN Monthly\_Sales\_Amount = MaxSalesAmount THEN Sales\_Month END) AS Best\_Sales\_Month,  MAX(Monthly\_Sales\_Amount) AS Best\_Month\_Sales\_Amount,  MAX(CASE WHEN Monthly\_Sales\_Amount = MinSalesAmount THEN Sales\_Month END) AS Worst\_Sales\_Month,  MIN(Monthly\_Sales\_Amount) AS Worst\_Month\_Sales\_Amount  FROM (  SELECT  YEAR(t.Date) AS Sales\_Year,  MONTH(t.Date) AS Sales\_Month,  SUM(sf.Unit\_Price \* sf.Quantity) AS Monthly\_Sales\_Amount,  MAX(SUM(sf.Unit\_Price \* sf.Quantity)) OVER (PARTITION BY YEAR(t.Date)) AS MaxSalesAmount,  MIN(SUM(sf.Unit\_Price \* sf.Quantity)) OVER (PARTITION BY YEAR(t.Date)) AS MinSalesAmount  FROM  Sales\_Fact sf  INNER JOIN Time\_DIM t ON sf.Time\_ID = t.Time\_ID  GROUP BY  YEAR(t.Date), MONTH(t.Date)  ) AS MonthlySales  GROUP BY Sales\_Year  ) AS YearlySales  ORDER BY Sales\_Year; | Insight: Best sales month for the year 2015 is October and worst sales month for 2015 is June. |
| 3 | Retrieves the details of the pending orders with the earliest order date and provide info about the products in the orders.  Including the total sales amount for each product and the quantity sold.  Sorted by earliest order date and sales revenue in descending order | SELECT o.Order\_ID, p.Product\_Name, p.Description, SUM(sf.Unit\_Price \* sf.Quantity) AS Product\_Sales\_Amount,  SUM(sf.Quantity) AS Amount\_Sold, o.Status  FROM  Orders\_DIM o  INNER JOIN Sales\_Fact sf ON sf.Order\_ID = o.Order\_ID  INNER JOIN Products\_DIM p ON sf.Product\_ID = p.Product\_ID  WHERE  o.Status = 'Pending' and  o.Order\_Date = (  SELECT MIN(Order\_Date)  FROM Orders\_DIM  WHERE Status = 'PENDING')  GROUP BY  o.Order\_ID, p.Product\_Name, p.Description, o.Order\_Date, o.Status  ORDER BY  o.Order\_Date, SUM(sf.Unit\_Price \* sf.Quantity) DESC; | A screenshot of a computer code  Description automatically generated  Insight: Order\_ID of 90 is the earliest order which is still pending, and the product that was ordered from that order was the Intel Xeon E5-2690 (OEM/Tray) |
| 4 | For each category, retrieve the top 3 best selling products, and query the customers name and address | WITH RankedProducts AS (  SELECT  ca.Category\_Name, p.Product\_Name, cu.CustName AS Customer\_Name, cu.Address ,SUM(sf.Quantity \* sf.Unit\_Price) AS Sales\_Revenue,  ROW\_NUMBER() OVER (PARTITION BY ca.Category\_Name ORDER BY SUM(sf.Quantity \* sf.Unit\_Price) DESC) AS Rank  FROM Sales\_Fact sf  INNER JOIN Customers\_DIM cu ON cu.Customer\_ID = sf.Customer\_ID  INNER JOIN Products\_DIM p ON sf.Product\_ID = p.Product\_ID  INNER JOIN Categories\_DIM ca ON ca.Category\_ID = p.Category\_ID  GROUP BY  ca.Category\_Name,  p.Product\_Name,  cu.CustName,  cu.Address  )  SELECT Category\_Name, Product\_Name, Customer\_Name, Address, Sales\_Revenue  FROM RankedProducts  WHERE Rank <=3  ORDER BY Category\_Name, Rank; | A screenshot of a computer  Description automatically generated  Insight: No product from the RAM category was being sold at all, Intel XEON E5-2683 V4 bought by Emmerson Electric made the best sales revenue for the CPU category. |
| 5 | Retrieves sales and profitability trend for the top 3 products based on the profit.  Provide info about each product name, number of inventory stock, total sales, selling price, cost price, profit and sales trend. | --Common table expression  WITH ProfitInfo AS (  SELECT  t.Year,  DateName(month , DateAdd(month , t.Month , 0) - 1 ) AS 'Month',  t.day,  p.Product\_Name,  s.Stocks,  sf.Quantity AS TotalSales,  sf.Unit\_Price AS Selling\_Price,  sf.Standard\_Cost AS Cost\_Price,  (sf.Unit\_Price \* sf.Quantity) - (sf.Standard\_Cost \* s.Stocks) AS Profit,  t.Date,  --Row Number assigns row numbers within each product group based on the date  ROW\_NUMBER() OVER (PARTITION BY p.Product\_ID ORDER BY t.Date) AS RowNum  FROM  Products\_DIM p  INNER JOIN  Stocks s ON p.Product\_ID = s.Product\_ID  INNER JOIN  Sales\_Fact sf ON p.Product\_ID = sf.Product\_ID  INNER JOIN  Time\_DIM t ON sf.Time\_ID = t.Time\_ID  )  --Main Query  --Extract the top 3 products based on profit  SELECT  Product\_Name,  Year,  Month,  Day,  Stocks AS 'Stock of Components',  TotalSales AS 'Sales',  Selling\_Price AS 'Selling Price',  Cost\_Price AS 'Cost Price',  Profit,  --LAG Window Function: difference between total sales of current row and total sales of previous row  TotalSales - LAG(TotalSales, 1, 0) OVER (PARTITION BY Product\_Name ORDER BY Date) AS 'Sales Trend'  FROM  ProfitInfo  WHERE  Product\_Name IN (  SELECT TOP 3 Product\_Name  FROM ProfitInfo  WHERE RowNum > 1 -- Excluding the first row to calculate the sales trend correctly  GROUP BY Product\_Name  ORDER BY SUM(Profit) DESC  )  ORDER BY  Product\_Name, Year, MONTH(Month + '1,1'); | A screenshot of a computer  Description automatically generated  Insight: Generally, based on the top 3 products cumulative profit, ASRock X99 Extreme11 and Asus KGPE-D16 have an increase in profit from $12310 to $41388 and -$1395 to $9447 respectively due to an increase in sales from 70 to 132 and 45 to 71 respectively. Supermicro MDB-X10DAX has a decrease in profit from $24396 to $6233 due to a decrease in sales from 102 to 61. Profit is directly related to sales specifically as it is the only change with time. |