**ST1501 CA2 Group Tasks**

**Class: DAAA/2B/23**

**Group No: 05**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **Student No** | **Name** | **Team Lead (Y/N)** |
| 2317425 | Adam | Y |
| 2317649 | Ziven |  |
| 2341475 | Caleb |  |
| 2317959 | Nicholas |  |

**Group Tasks Solution**

1. Submit OLTP\_insert.sql including SQL queries that insert data into the OLTP tables.

Identify data quality issues in dataset provided. List down details in the table below.

|  |  |  |
| --- | --- | --- |
| Issue No. | Description of issue (which file, which line/part, what is the problem) | Solution (how do you resolve the issue) |
| 1 | modeling.csv – the file is in excel form with multiple sheets within it and not in csv format. BULK INSERT is also not able to pull from a specific sheet for example, dataset in sheet 1. | We split up the sheets into their own separate csv file. |
| 2 | employee.csv – The Gender column in the CREATE TABLE statement is defined as CHAR (1) while the gender in the employee.docx has gender as ‘Male’ and ‘Female. | We changed the Gender column definition to VARCHAR (6) to allow for "Male" and "Female" strings. |
| 3 | order.csv – The columns are not in the same order as the columns from the CREATE TABLE for order. | We changed the headings of CREATE TABLE to match the order in the csv file. |
| 4 | order.csv – There is an error in the data in line 2722 where customerID is c0068xxxxxxxx. The CREATE TABLE does not allow this to be inserted as it is CustomerID VARCHAR(10) NOT NULL. | We fixed it by removing the extra 8 ‘x’ from the ID. |
| 5 | Order.csv - The dates are in the form DD/MM/YYYY where it is using the ‘/’ as a divider. The CREATE TABLE date format only accepts YYYY-MM-DD with the ‘-’ as a divider. | We fixed it using bulk insert as with bulk insert, the data types and formats are often handled more explicitly based on the file format and the table schema, reducing the likelihood of format-related errors during insertion. |
| 6 | Order.csv - There is a missing customer ID in row 857. This causes issues when inserting into the DW orderfacts. | We fixed it by dropping/removing the order from the order.csv. This is because we do want to assume who the customer was that ordered that order. Dropping it was the best option. |

Explain how you check if your table creation and data insertion are correct.

|  |
| --- |
| 1. **Verify Table Creation:**  * Check if the table schema (columns, data types, constraints) matches your requirements. You can do this by:   + Using SQL commands like DESCRIBE, SHOW CREATE TABLE, or querying metadata tables (INFORMATION\_SCHEMA in SQL Server) to view the table structure.   + Comparing the output with your intended schema to ensure all columns are present with the correct data types and constraints.   SELECT COLUMN\_NAME, DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.COLUMNS  WHERE TABLE\_NAME = 'Customer';  SELECT COLUMN\_NAME, DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.COLUMNS  WHERE TABLE\_NAME = 'Employee';  For example, as shown below:  A screenshot of a data  Description automatically generated   1. **Confirm Data Insertion:**  * After inserting data into the table, verify that the data has been correctly inserted. Here’s how:   + Run a SELECT query to retrieve data from the table (SELECT \* FROM your\_table\_name).   + Compare the retrieved data with the data you intended to insert. Check for:     - Correct values in each column.     - Integrity constraints (like primary keys, foreign keys) are maintained.     - Data formats (dates, numbers, strings) are as expected.   For example: Looking at Customer and Employee using Select \*  A screenshot of a computer  Description automatically generated   1. **Check for Errors and Warnings:**  * Look for any error messages or warnings during table creation and data insertion. Common issues include:   + Syntax errors in SQL statements.   + Constraint violations (e.g., attempting to insert a duplicate primary key).   + Data truncation (inserting data that exceeds column size limits). * Address any errors or warnings to ensure data integrity.   For example, issue number 4:  A close-up of a white background  Description automatically generated |

1. Paste your data warehouse design here (database diagram). Write a short description to explain your design such as choice of measurement and levels of details.

|  |  |
| --- | --- |
| DW design | Show all details of the DB diagram. Format your layout and size of tables to ensure nothing is hidden. |
| Explain your design | We have decided to do a snowflake schema for our data warehouse because of several reasons:  **Why did we choose Snowflake over Star schema:**   * Normalization and redundancy   + It reduces the amount of redundant data.   + Saves storage space and maintain data integrity. * Query performance/complexity   + While normalized tables can lead to slower query performance due to the need for multiple inner joins than star schema, snowflake schema can handle more complex queries where detailed relationships between dimensions are needed.   + Snowflake schema is better for complex queries compared to star schema. * Scalability and maintenance:   + Snowflake schema is easier to maintain and update due to its ability for more granular updates and changes. While star schema is simpler due to the denormalized nature, in the long run it would be worse as data red redundancy increases.   **Choice of Measurements:**   * Price from the OrderFacts table is the choice of measurements   + The Price measure is crucial for financial analysis, allowing the business to track revenue, identify profitable customers, understand cost structures, and perform profitability analysis.     - Sum: Total revenue or cost across all orders.     - Average: Average order value.     - Minimum and Maximum: Identifying the lowest and highest order values.     - Group By: Aggregating by different dimensions like time, customer, employee, model, and dataset to derive insights.   **Levels of details:**   * Each dimension provides detailed context for the orders, enabling analysis across different perspectives (time, customer, model, dataset, employee). * OrderDIM:   + Details: Includes OrderID , RequiredAcc (the required accuracy for the order) and CompletionDate (when the order was completed). This helps in understanding the quality requirements and timelines of orders. * CustomerDIM:   + Details: Contains customer-specific information such as CustomerID, FirstName, LastName, CompanyName, and Contact. This dimension allows for analysis based on customer demographics, company associations, and contact information, which can be useful for customer segmentation and personalized marketing. * ModelTypeLookup:   + Details: Maps ModelCode to ModelType, providing a classification of models. This lookup table supports understanding the types of models used in orders, facilitating analysis of model performance and trends. * ModelDIM:   + Details: Includes ModelID, ModelCode, TrainingDate, and Accuracy. This dimension allows for analysis based on the model's training details and accuracy, helping to evaluate model performance and improvements over time. * DatasetDIM:   + Details: Captures DatasetID, DatasetName, providing information on the datasets used in model training and evaluation. This dimension supports analysis of dataset usage and its impact on model performance. * EmployeeDIM:   + Details: Contains EmployeeID, FirstName, LastName, Contact, and Gender of employees. This dimension allows for analysis of employee contributions to orders, helping in performance evaluations and resource management. * TimeDIM:   + Details: Includes Date, Year, Quarter, Month, and DayOfMonth. This dimension is critical for time-based analysis, such as tracking orders over time, identifying seasonal trends, and performing year-over-year comparisons.   The OrderFacts table is designed at the order level, allowing detailed analysis and aggregation across various dimensions.  Each dimension table contains specific attributes relevant to its entity (e.g., customer, model, employee), supporting comprehensive and detailed analysis.  The ModelTypeLookup table separates model types from model details, providing flexibility and ease of updating model classifications. |
|  |  |

1. Implement the data warehouse you designed in b) using MS SQL server.

Submit DW\_create.sql with all the SQL statements that creates DW tables, and DW\_insert.sql with all the SQL statements that query data from OLTP tables and insert into DW tables.

Briefly explain how you verify that your data warehouse is setup correctly, and that the data is inserted correctly.

|  |
| --- |
| To verify that our data warehouse is set up correctly and data is inserted properly:   * We first ensured all tables exist and have the correct structure, including appropriate columns and data types. * We also checked that tables contain the expected number of rows and reviewed a sample of data to confirm its accuracy. * Validate data integrity by looking for missing values and performing basic aggregations. * Verify relationships between tables, monitor recent data insertions, and ensure that indexes and constraints are correctly applied. |

1. Implement the queries and explain your findings to the 3 questions. You can list no more than 3 findings/queries for each question below. Modify the template accordingly.

* Q1:

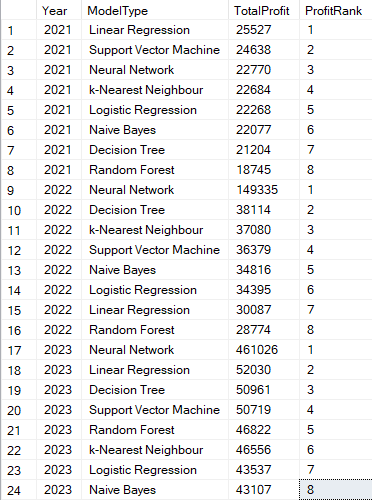
What do you want to find out?

|  |
| --- |
| Which Model Type generates the greatest number of profits obtained every year? |

Insert your query here:

|  |
| --- |
| WITH ProfitRanks AS (  SELECT  t.Year,  mtl.ModelType,  SUM(order\_facts.Price) AS TotalProfit,  ROW\_NUMBER() OVER (PARTITION BY t.Year ORDER BY SUM(order\_facts.Price) DESC) AS ProfitRank  FROM  SPAIDW2B2305..OrderFacts AS order\_facts  JOIN  SPAIDW2B2305..TimeDIM AS t ON order\_facts.TimeKey = t.TimeKey  JOIN  SPAIDW2B2305..ModelDIM AS modelDIM ON order\_facts.ModelKey = modelDIM.ModelKey  JOIN  SPAIDW2B2305..ModelTypeLookup AS mtl ON modelDIM.ModelCode = mtl.ModelCode  GROUP BY  t.Year, mtl.ModelType  )  SELECT  Year,  ModelType,  TotalProfit,  ProfitRank  FROM  ProfitRanks  ORDER BY  Year ASC, ProfitRank ASC; |

Insert your results here:



Explain what you find based on the results:

|  |
| --- |
| * **Top Model Types by Year:** * The query identifies the most profitable model types for each year by ranking them based on total profit. This shows which models contribute the most to revenue annually. * **Profit Trends Over Time:** * By examining the Year and TotalProfit fields, you can identify trends in profitability for different model types over the years. For instance, you might observe that certain models are consistently top performers, while others may have fluctuating profitability. * **Emerging and Declining Models:** * The rankings (ProfitRank) help highlight which model types are emerging as top performers and which are declining in profitability. This is crucial for strategic planning and resource allocation. |

* Q2:

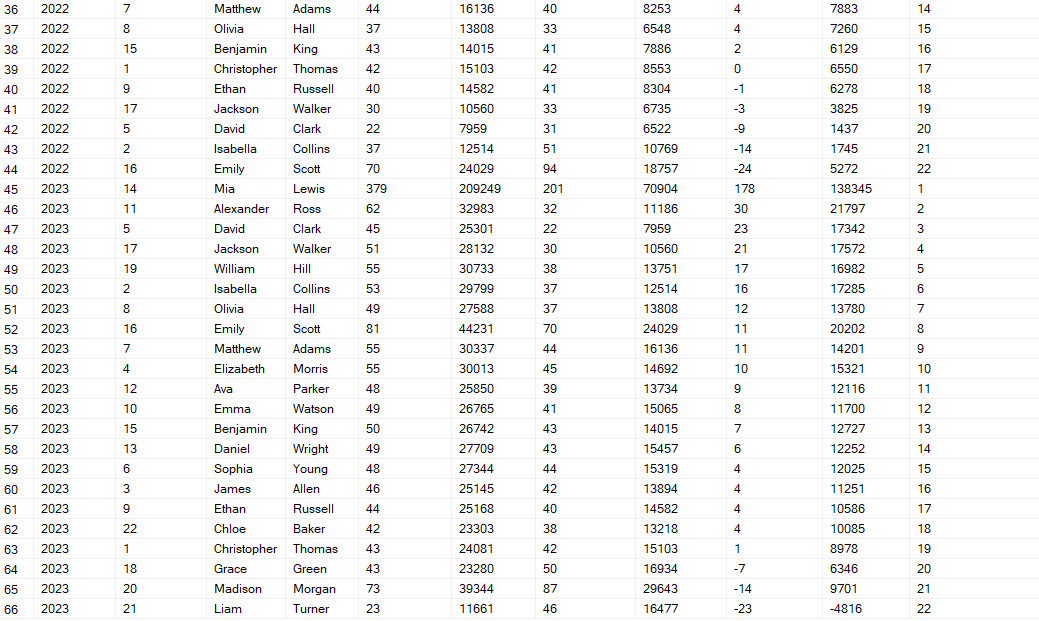
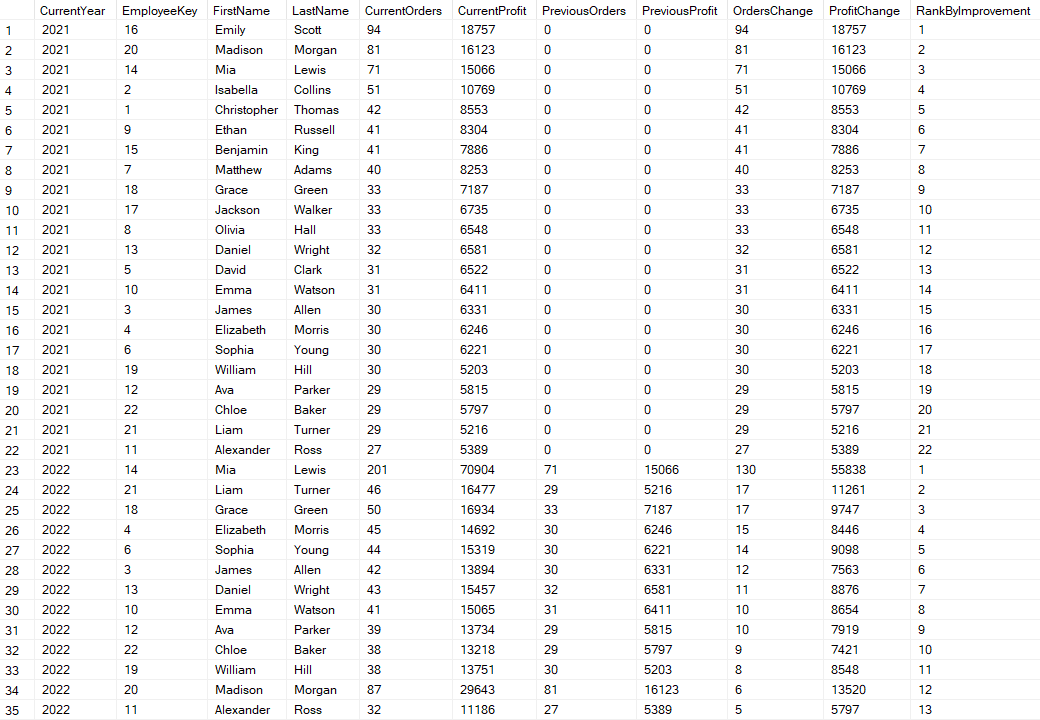
What do you want to find out?

|  |
| --- |
| Yearly number of orders completed, and profits generated by employee |

Insert your query here:

|  |
| --- |
| WITH YearlyPerformance AS (  SELECT  t.Year,  e.EmployeeKey,  e.FirstName,  e.LastName,  COUNT(ofct.OrderKey) AS TotalOrders,  SUM(ofct.Price) AS TotalProfit  FROM  OrderFacts ofct  JOIN  TimeDIM t ON ofct.TimeKey = t.TimeKey  JOIN  EmployeeDIM e ON ofct.EmployeeKey = e.EmployeeKey  GROUP BY  t.Year, e.EmployeeKey, e.FirstName, e.LastName  ),  PerformanceChanges AS (  SELECT  yp1.EmployeeKey,  yp1.FirstName,  yp1.LastName,  yp1.Year AS CurrentYear,  yp1.TotalOrders AS CurrentOrders,  yp1.TotalProfit AS CurrentProfit,  COALESCE(yp2.TotalOrders, 0) AS PreviousOrders,  COALESCE(yp2.TotalProfit, 0) AS PreviousProfit,  (yp1.TotalOrders - COALESCE(yp2.TotalOrders, 0)) AS OrdersChange,  (yp1.TotalProfit - COALESCE(yp2.TotalProfit, 0)) AS ProfitChange  FROM  YearlyPerformance yp1  LEFT JOIN  YearlyPerformance yp2  ON  yp1.EmployeeKey = yp2.EmployeeKey  AND yp1.Year = yp2.Year + 1  )  SELECT  CurrentYear,  EmployeeKey,  FirstName,  LastName,  CurrentOrders,  CurrentProfit,  PreviousOrders,  PreviousProfit,  OrdersChange,  ProfitChange,  ROW\_NUMBER() OVER (PARTITION BY CurrentYear ORDER BY OrdersChange DESC, ProfitChange DESC) AS RankByImprovement  FROM  PerformanceChanges  ORDER BY  CurrentYear, RankByImprovement; |

Insert your results here:



Explain what you find based on the results:

|  |
| --- |
| * **Improvement in Performance:** This query helps identify employees who have improved their performance in terms of both the number of orders and profit. * **Rank by Year:** Employees are ranked within each year based on their improvement, which highlights those who have shown the most significant positive changes. * How this helps the owner of SPAI is by highlighting better performing employees to consider for a pay-rise or promotion. |

* Q3:

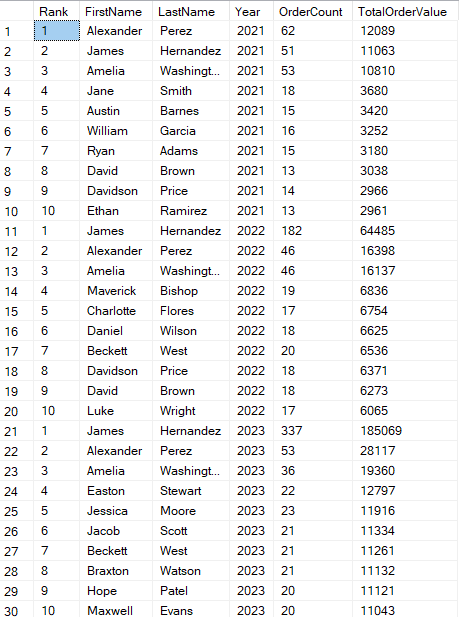
What do you want to find out?

|  |
| --- |
| identify the top 10 customers who have contributed the most to the total order value for each year and rank them by contribution |

Insert your query here:

|  |
| --- |
| WITH TopCustomers AS (  SELECT  customerDIM.CustomerKey,  customerDIM.FirstName,  customerDIM.LastName,  SUM(order\_facts.Price) AS TotalOrderValue  FROM  OrderFacts AS order\_facts  JOIN  CustomerDIM AS customerDIM ON order\_facts.CustomerKey = customerDIM.CustomerKey  GROUP BY  customerDIM.CustomerKey, customerDIM.FirstName, customerDIM.LastName  ),  YearlyOrderDistribution AS (  SELECT  tc.CustomerKey,  tc.FirstName,  tc.LastName,  timeDIM.Year,  COUNT(\*) AS OrderCount,  SUM(order\_facts.Price) AS TotalOrderValue  FROM  OrderFacts AS order\_facts  JOIN  TimeDIM AS timeDIM ON order\_facts.TimeKey = timeDIM.TimeKey  JOIN  TopCustomers AS tc ON order\_facts.CustomerKey = tc.CustomerKey  GROUP BY  tc.CustomerKey, tc.FirstName, tc.LastName, timeDIM.Year  ),  RankedYearlyDistribution AS (  SELECT  FirstName,  LastName,  Year,  OrderCount,  TotalOrderValue,  ROW\_NUMBER() OVER (PARTITION BY Year ORDER BY TotalOrderValue DESC) AS Rank  FROM  YearlyOrderDistribution  )  SELECT  Rank,  FirstName,  LastName,  Year,  OrderCount,  TotalOrderValue  FROM  RankedYearlyDistribution  WHERE  Rank <= 10  ORDER BY  Year, Rank; |

Insert your results here:



Explain what you find based on the results:

|  |
| --- |
| * **Customer Loyalty Programs:** SPAI can develop loyalty programs to reward these top customers during their peak purchasing years. * **Personalized Marketing:** Understanding the order patterns helps SPAI tailor personalized marketing efforts to boost sales during off-peak years. * **Resource Allocation:** By knowing when high-value customers are most active, SPAI can allocate resources more efficiently to maximize customer satisfaction and sales. |