A case study to introduce Microsoft Data Mining in the database course

Mohammad Dadashzadeh
Oakland University

ABSTRACT

The content of the database management systems course in the business curriculum has remained stable covering conceptual data modeling, relational database design and implementation, structured query language (SQL), application development, and database administration. Given the breadth and the depth of needed coverage, there is little opportunity left for the instructor to introduce data warehousing concepts in any depth let alone to cover predictive analytics. This paper presents a market basket analysis case study that successfully leverages SQL coverage in the course to introduce students to Microsoft data mining algorithms for predictive analytics. The phased presentation of the case study and the pedagogical opportunities it affords are discussed.

Keywords: MIS Curriculum, Database Course Content, Microsoft Data Mining, Market Basket Analysis

INTRODUCTION

As real-world predictive analytic applications such as detecting fraud, predicting sales, customer segmentation, and social media sentiment analysis gain increasing momentum, there is a greater urgency to introduce information system students to the algorithms used for machine learning. Given the role that data plays in building predictive models, it makes sense to introduce predictive analytics in the database course. However, the breadth and the depth of needed coverage of standard topics in the database course (Mannino, 2019; Topi et. al, 2010) including conceptual data modeling, relational database design and implementation, structured query language (SQL), application development, and database administration, leave little opportunity for the instructor to introduce data warehousing concepts in any depth let alone to cover predictive analytics.

Amongst the many algorithms used for machine learning such as regression, classification, and clustering, the Apriori algorithm (Agrawal & Srikant, 1994) used for association analysis lends itself naturally to introduction in the database course. The Apriori algorithm is designed to operate on databases containing transactions such as collection of items bought together by a customer in a store visit. The conceptual modeling of the database needed for this scenario is a part of typical coverage in a database course as depicted in Figure 1.

Given the students' familiarity with this database and SQL, a case study can be gradually presented to discover what categories of products are bought together and how such data analysis can lead to formulation of association rules that can inform marketing promotions and recommendation systems while supporting data-driven managerial decision making. The case study provides a vehicle to teach students more advanced SQL queries both in Microsoft Access and SQL Server, as well as how to use Microsoft's implementation of Apriori algorithm and interpret the resulting association rules. The next section of the paper introduces the gradual development of the case study and the pedagogical opportunities it presents.

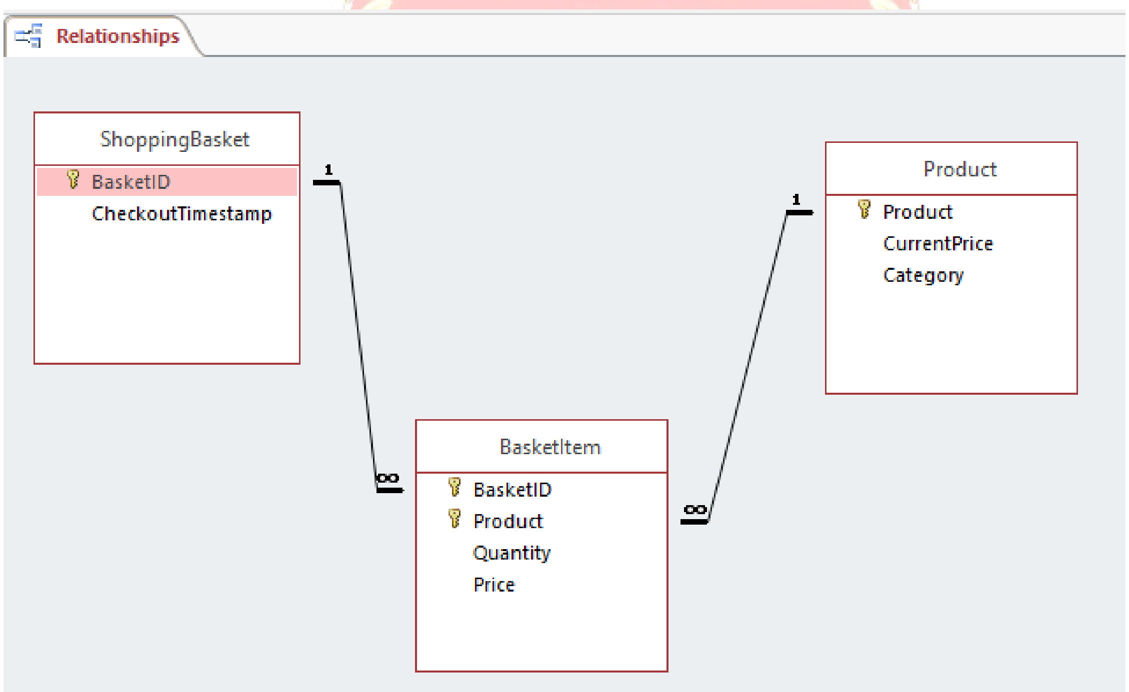


Figure 1. Conceptual Data Model for Shopping Basket Analysis Database

THE SHOPPING BASKET ANALYSIS CASE STUDY: PEDAGOGY

Given the database design shown in Figure 1, the following pedagogical opportunities can be taken advantage of to present the case study in a phased manner:

The impact of time on database design.

Given that when a shopping basket is created in the database the price of each item in the basket is obtained from the Product table, why is there a need to repeat that price in the Basket_Item table? Discussing this question leads to a better understanding of the importance of careful naming of attributes, e.g., Current_Price versus Price, and the fact that a database must capture transaction information in such a way that it can be re-produced with integrity at any time in the future.

Visual Basic for Applications (VBA) database programming.

A Microsoft Access copy of the database with the Checkout_Timestamp and Quantity data fields set to null is given to the students asking them to write (or complete) VBA code that would randomly assign a date and time to each shopping basket and similarly assign a random quantity to each basket item. Although, minor in programming scope and difficulty (see Appendix), this opportunity to learn Access VBA programming is valuable for test data generation in the students' database application development term project.

Query By Example (QBE) listing of pairs of product categories most bought together.

Students are asked to develop a query to identify top 5 pairs of product categories most bought together as shown in Figure 2. This non-trivial self-join query is especially valuable in teaching students how to overcome Microsoft Access' restriction in not supporting Count Distinct operation and how an imposed ordering (e.g., Category#1 < Category#2) in such queries can remove the output of redundant pairs (see Appendix).

Category#1	Category#2	Times Bought Together
Helmets	Tires and Tubes	1617
Helmets	Road Bikes	805
Bottles and Cages	Helmets	715
Mountain Bikes	Tires and Tubes	569
Bottles and Cages	Mountain Bikes	563

Figure 2. Top 5 Pairs of Product Categories Bought Together

QBE query to list product category most bought along with a pair of product categories.

This extension of the product category association to three items (see Figure 3 and Appendix) shows the students the essential approach needed for discovering association rules and the advantages that a tool such as Microsoft Data Mining offers to perform association analysis.

Query2

Category#1	Category#2	Category	Number of Times Bought Together
Helmets	Jerseys	Tires and Tubes	179
Bottles and Cages	Helmets	Mountain Bikes	172
Gloves	Helmets	Tires and Tubes	160
Fenders	Helmets	Mountain Bikes	131
Bottles and Cages	Helmets	Road Bikes	130
Caps	Helmets	Tires and Tubes	127
Helmets	Mountain Bikes	Tires and Tubes	122
Helmets	Road Bikes	Tires and Tubes	118
Bottles and Cages	Fenders	Mountain Bikes	114
Fenders	Jerseys	Mountain Bikes	102

Figure 3. Top 10 Product Categories Most Bought Together Along with Another Pair

Importing an Access database into SQL Server.

Using SQL Server Import and Export Wizard students import the Microsoft Access tables into a new database in SQL Server (see Appendix).

Defining the relationships between tables in SQL Server.

The Import and Export Wizard does not always transfer the relationships between tables and they must be defined by creating a Database Diagram in SQL Server (see Appendix). This provides an instructional opportunity to point out differences between the two database management systems in data types that are supported.

SQL query to list pairs of product categories most bought together.

With the database imported into SQL Server, students are asked to use SQL Server Management Studio's Query Editor to design the SQL query to list pairs of product categories most bought together (see Figure 4 and Appendix).

```
SELECT      TOP (5) Product.Category AS Category#1, Product_1.Category AS Category#2,
            COUNT(DISTINCT BasketItem_1.BasketID) AS [Number of Times Bought Together]
FROM        BasketItem AS BasketItem_1 INNER JOIN Product AS Product_1
                ON BasketItem_1.Product = Product_1.Product
                INNER JOIN Product
                            INNER JOIN BasketItem
                            ON Product.Product = BasketItem.Product
                ON BasketItem_1.BasketID = BasketItem.BasketID AND Product_1.Category > Product.Category
GROUP BY    Product.Category, Product_1.Category
ORDER BY    [Number of Times Bought Together] DESC
```

100 %

Results | Messages

	Category#1	Category#2	Number of Times Bought Together
1	Helmets	Tires and Tubes	1617
2	Helmets	Road Bikes	805
3	Bottles and Cages	Helmets	715
4	Mountain Bikes	Tires and Tubes	569
5	Bottles and Cages	Mountain Bikes	563

Figure 4. Top 5 Pairs of Product Categories Bought Together in SQL Server

Creating a data mining solution project using Visual Studio.

Microsoft data mining algorithms (Microsoft, 2018) are employed by creating a SQL Server Analysis Services Multidimensional and Data Mining project using Visual Studio. The solution steps required (see Figure 5) are to identify the data source (e.g., our SQL Server database), the data source view (i.e., tables and views from the data source we need), and to define the data mining structure.

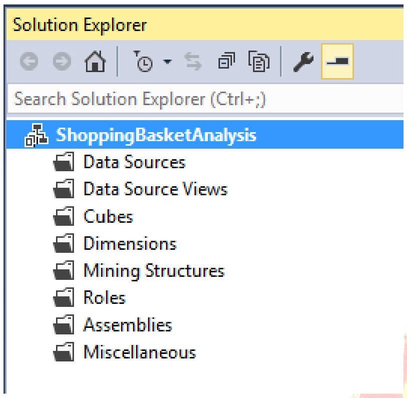


Figure 5. An Empty Data Mining Solution Folder in Visual Studio

Creating a named query in the data source view.

Since we intend to discover product categories that are bought together, we will need each row in the Basket Item table to include the purchased product's category. This could have been accomplished by creating a view in our SQL Server database. However, it is more appropriate to create this one-shot view in the data source view of our project as a new named query (see Appendix). This provides an instructional opportunity to discuss why views supporting multiple queries need to be created in the database and when an in-line view (i.e., in the FROM clause of SQL SELECT statement) to support a single query becomes more appropriate.

Creating a data mining structure.

Microsoft Data Mining Wizard guides the user in creating a data mining model. The Association Rules algorithm is one of nine algorithms supported and requires specification of the "case" table (i.e., the ShoppingBasket table and its primary key BasketID) as well as the "nested" table (i.e., the named query BasketItemWithCategory listing each shopping basket item's product category) (see Appendix). The algorithm also requires specifying the input column to be used (i.e., category) and the predictable column (i.e., also category) to discover associations as shown in Figure 6.

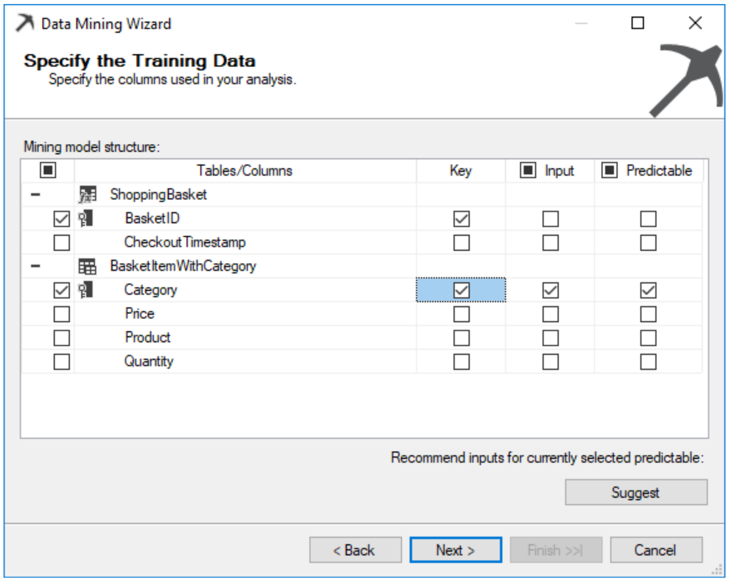


Figure 6. Specifying the Training Data for Microsoft Association Rules Algorithm

Specifying data set to be reserved for testing the predictive model.

Setting aside a percentage of data for testing the association rules to be discovered will mean that the results produced will be different for each student. To avoid that randomness, students are asked to specify zero as percentage of data for testing as shown in Figure 7.

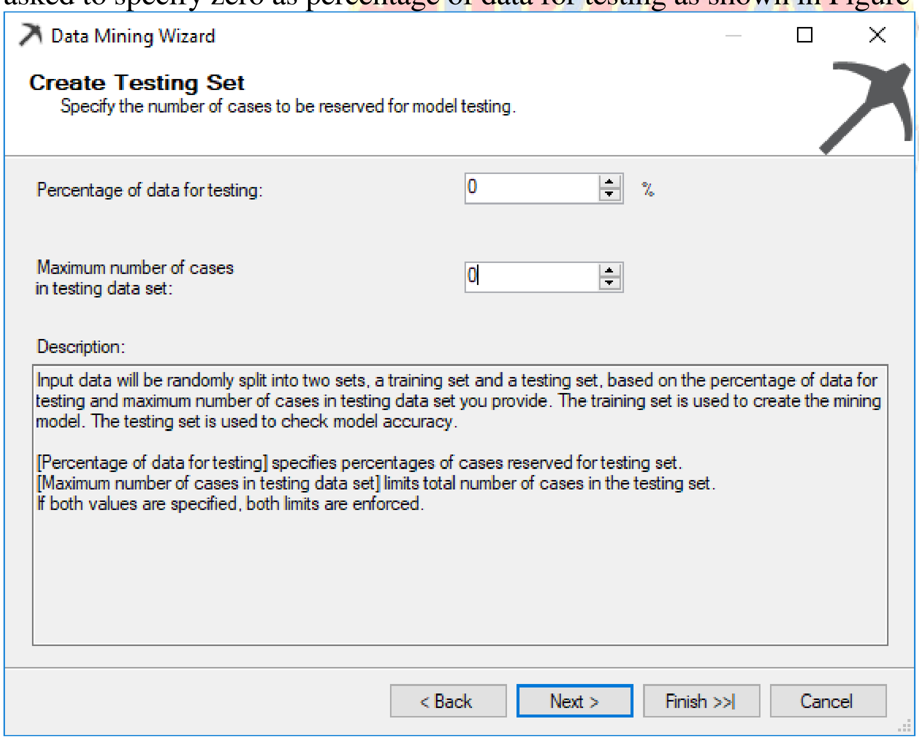


Figure 7. Specifying Percentage of Data Reserved for Testing

Interpreting the association model's results.

Processing the defined data mining model will discover the association rules and display them as shown in Figure 8. The rules are listed along with the "Probability" and "Importance" of the rule. The first association rule in Figure 8 indicates that Bike Stands, Road Bikes → Tires and Tubes, signifying that a product of category Tires and Tubes is most often bought together with products of category Bike Stands and Road Bikes. Indeed, the "probability" or "confidence" value of this rule indicates certainty. That is,

Probability(Tires and Tubes | Bike Stands, Road Bikes) = 100%

Or, equivalently,

P(Tires and Tubes, Bike Stands, Road Bikes) / P(Bike Stands, Road Bikes) = 1

Students are asked to develop SQL queries to verify the above or any of the other rules' confidence value (see Appendix).

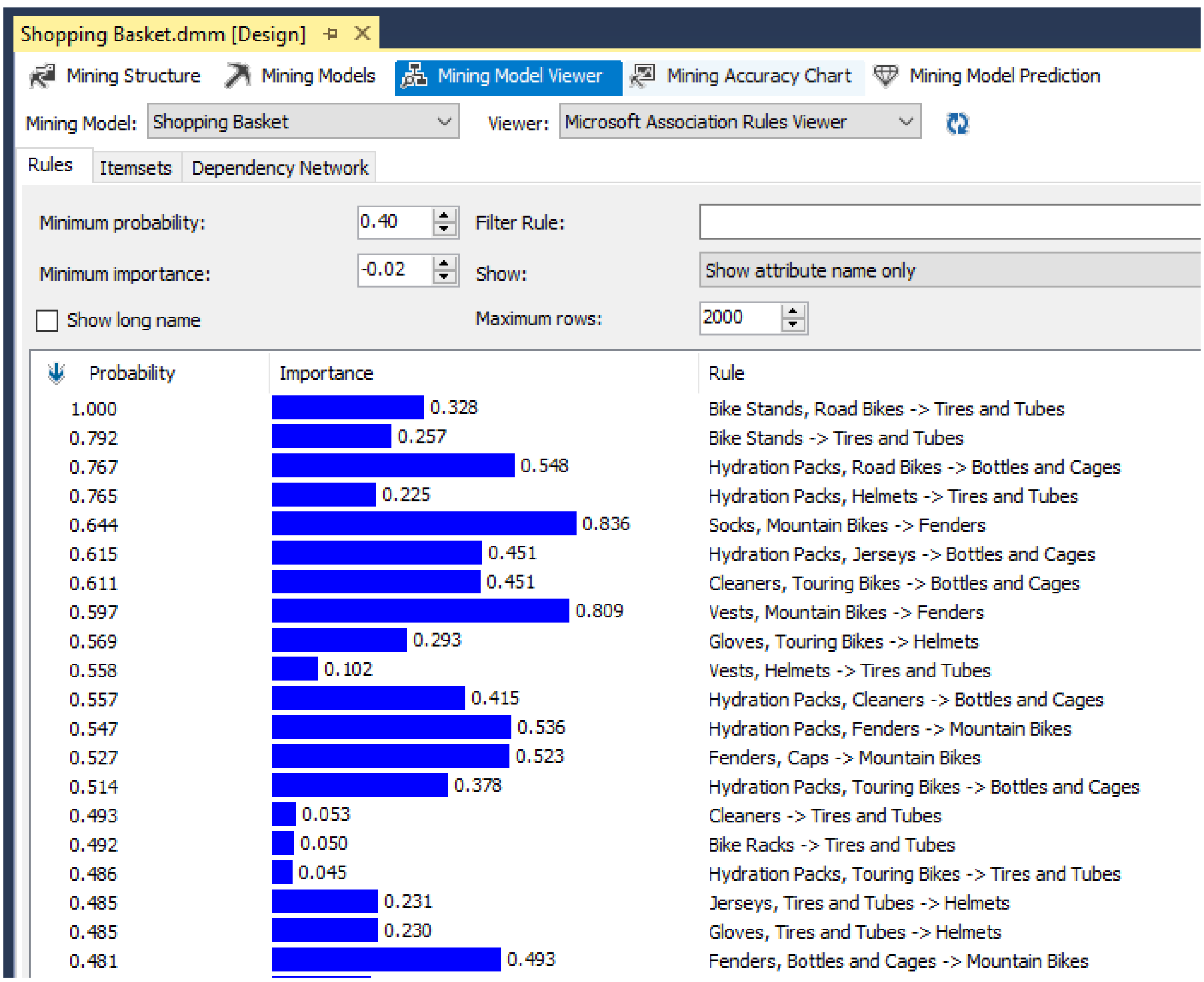


Figure 8. Association Rules Discovered Sorted by Conditional Probability

Interpreting the "importance" value.

In an association model, a strong rule, or one that has high confidence, might not necessarily be interesting because it does not provide new information. In our example, the Bike Stands, Road Bikes → Tires and Tubes association rule has a confidence value of 1, but only an "importance" value of 0.328. The importance of this rule is related to the extent that probability of finding a product of category Tires and Tubes in the shopping basket is increased (or lifted) when there are also products of categories Bike Stands, and Road Bikes present versus when they are not. The logarithm of this ratio is shown as the "importance" value:

Importance(Bike Stands, Road Bikes → Tires and Tubes) =

Log (P(Tires and Tubes | <Bike Stands, Road Bikes>) / P(Tires and Tubes | NOT <Bike Stands, Road Bikes>))

Since, the base 10 logarithm of 2.12 is 0.328, we see that the probability of finding a product of category Tires and Tubes in the shopping basket is 2.12 times higher when there are also products of categories Bike Stands, and Road Bikes present versus when they are not.

The highest importance value of 0.836 shown in Figure 8 is for the association rule: Socks, Mountain Bikes → Fenders. This signifies that the probability of finding a fender in the shopping basket is 6.85 (the base 10 logarithm of 6.85 is 0.836) times higher when there are also products of categories Socks, and Mountain Bikes present versus when they are not. Writing SQL queries to verify importance values is another non-trivial problem-solving learning opportunity (see Appendix).

Data driven decision making.

Microsoft's Association Rules algorithm also provides its Dependency Network view of the association rules discovered as shown in Figure 9. This provides a good starting place for leading class discussions on how the links shown can lead to managerial action.

Figure 9. The Dependency Network for Product Category Associations

Association analysis based on actual products.

Having guided and presented the case study in stages, an appropriate follow-up assignment is to ask the students to build a model to discover association rules based on the actual products as opposed to product categories (see Appendix).

SUMMARY AND CONCLUSIONS

Adoption of data-driven decision making as an essential business skill and a learning objective in AACSB accredited business school's curricular efforts has resulted in predictive analytics being taught/used in courses from every department in a business school. Given the role that data plays in building predictive models, the database course should not be left behind. However, the breadth and the depth of needed topic coverage in the database course leaves little room for introduction of data mining theory and practice. This is also complicated by the proliferation of software tools for building predictive models (including open source, commercial, and free for academic use) in that students need to learn the mechanics of different software packages. Since 2005, Microsoft has introduced nine data mining algorithms for predictive analytics built into its SQL Server Analysis Services (SSAS) which is available to universities as a part of Microsoft Developer Network Academic Alliance (MSDNAA). The case study presented in this paper leverages students' familiarity with SQL, Microsoft Access and SQL Server, to introduce Microsoft Data Mining in the database course.

REFERENCES

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, 487-499.

Mannino, M. (2019). Database Design, Application Development & Administration, 7^{th} Edition. Chicago: Chicago Business Press.

Microsoft. (2018). Data Mining Algorithms. Retrieved from https://docs.microsoft.com/en-us/sql/analysis-services/data-mining/data-mining-algorithms-analysis-services-data-mining?view=sql-analysis-services-2017 on May 4, 2018.

Topi, H., Valacich, J., Wright, R., Kaiser, K., Nunamaker, J., Sipior, J., & de Vreede, G. (2010). IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. Communications of the Association for Information Systems, 26(18).

APPENDIX

Answers and Supporting Screenshots for Case Study Steps

This appendix provides supplemental materials that aid in completing the case study in a hand-on manner. A copy of the Access database with completed queries, equivalent SQL Server queries, and Visual Studio data mining project folder are also available from the author.

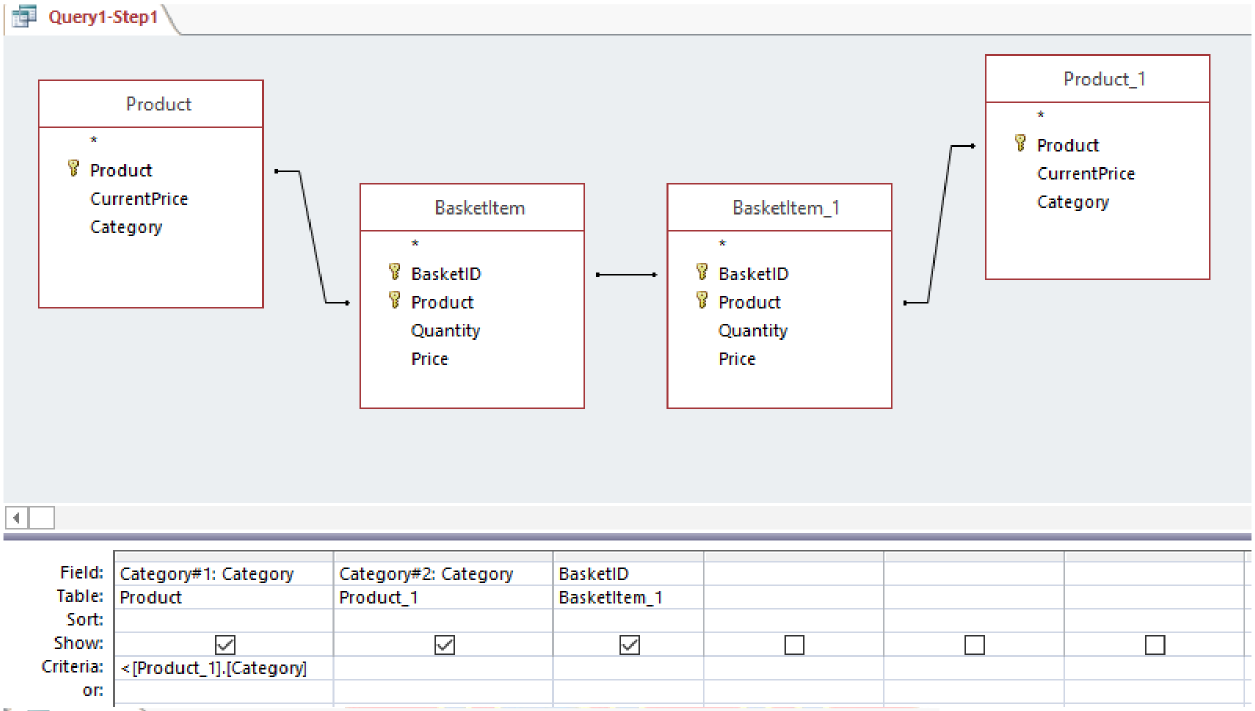
A. Microsoft Access VBA code

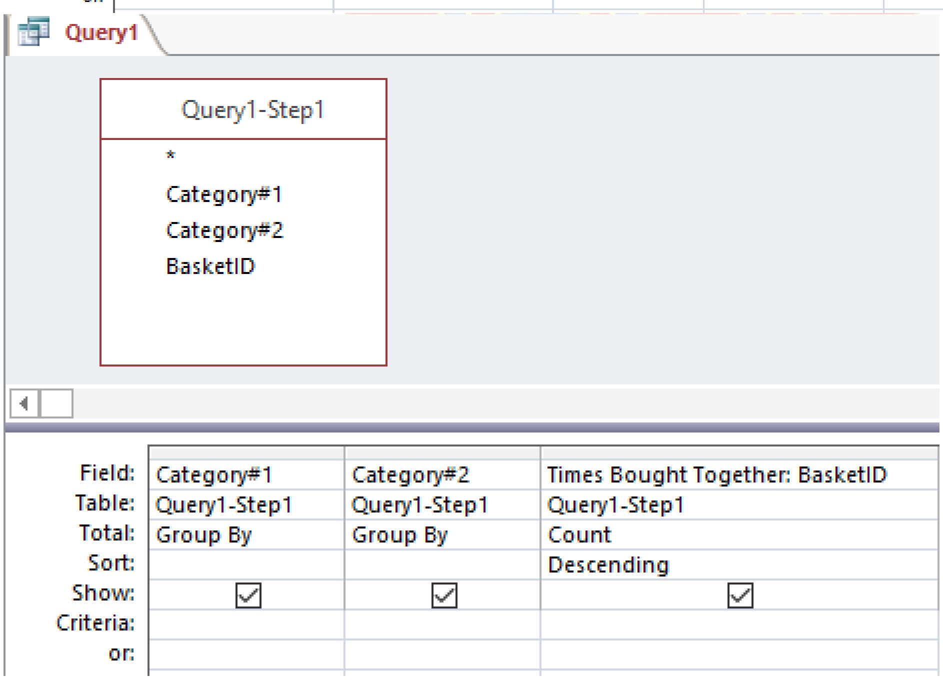
```
Sub AddQuantity()
'Adds a random quantity value of 1 or 2 to each row in BasketItem table ...
'7% of records are assigned quantity value of 2 ...

Dim rs As Recordset, Qty As Integer

Set rs = CurrentDb.OpenRecordset("BasketItem")

Randomize

Do While Not rs.EOF

    Qty = 1
    If Rnd < 0.07 Then 'Only 7% of time ...
        Qty = 2
    End If

    rs.Edit
        rs("Quantity") = Qty
    rs.Update

    rs.MoveNext

Loop

rs.Close

End Sub
```

```
Sub AddCheckoutTimestamp()
'Adds a random Checkout_Timestamp value to each row in ShoppingBasket table ...
'Generate random date in April 2018 ...
'Generate random time bwteen 9:00 and 20:59 ...

Dim rs As Recordset
Dim myDay As Integer, myDate As Date, Time1 As Date, Time2 As Date, myTime As Date
Dim myTimestamp As Date

Set rs = CurrentDb.OpenRecordset("ShoppingBasket")

Randomize

Do While Not rs.EOF

    myDay = Int(Rnd * 30) + 1 'Between 1 and 30 ...
    myDate = DateSerial(2018, 4, myDay) 'April 2018 ...

    Time1 = "09:00:00 AM"
    Time2 = "08:59:59 PM"

    myTime = TimeValue(Time1 + Rnd() * (Time2 - Time1))
    myTimestamp = myDate & " " & myTime

    rs.Edit
        rs("CheckoutTimestamp") = myTimestamp
    rs.Update

    rs.MoveNext

Loop

rs.Close

End Sub
```

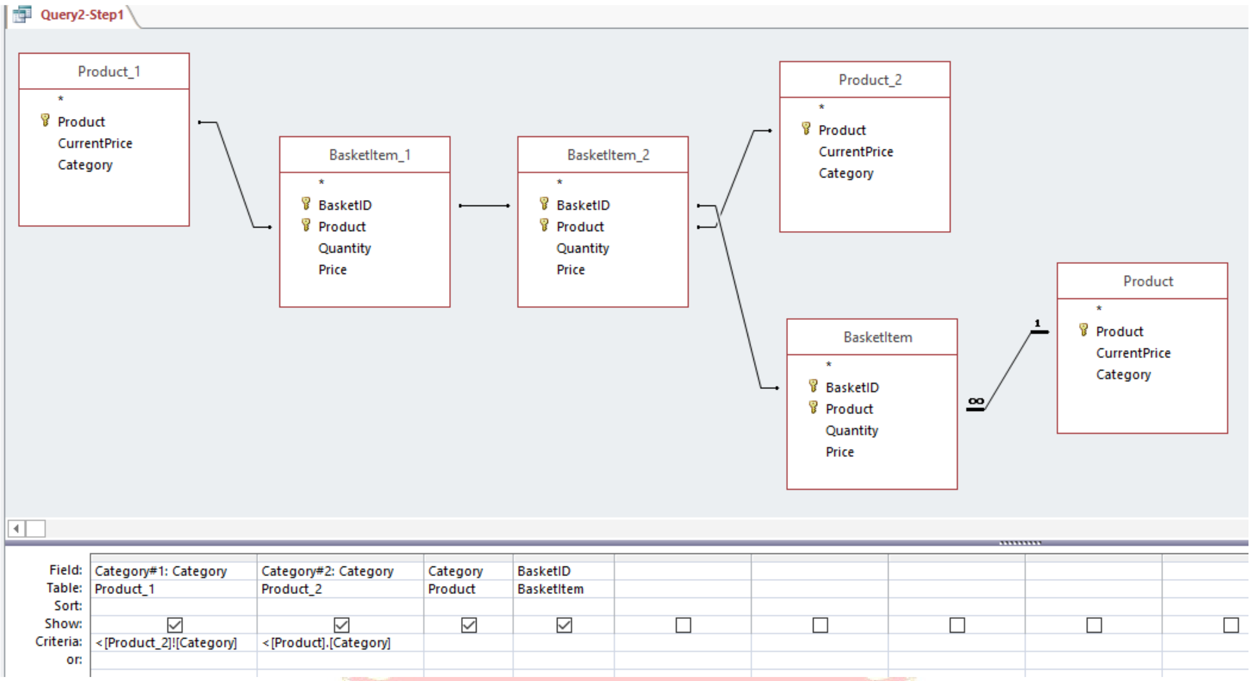
B. QBE query to list pairs of product categories most bought together

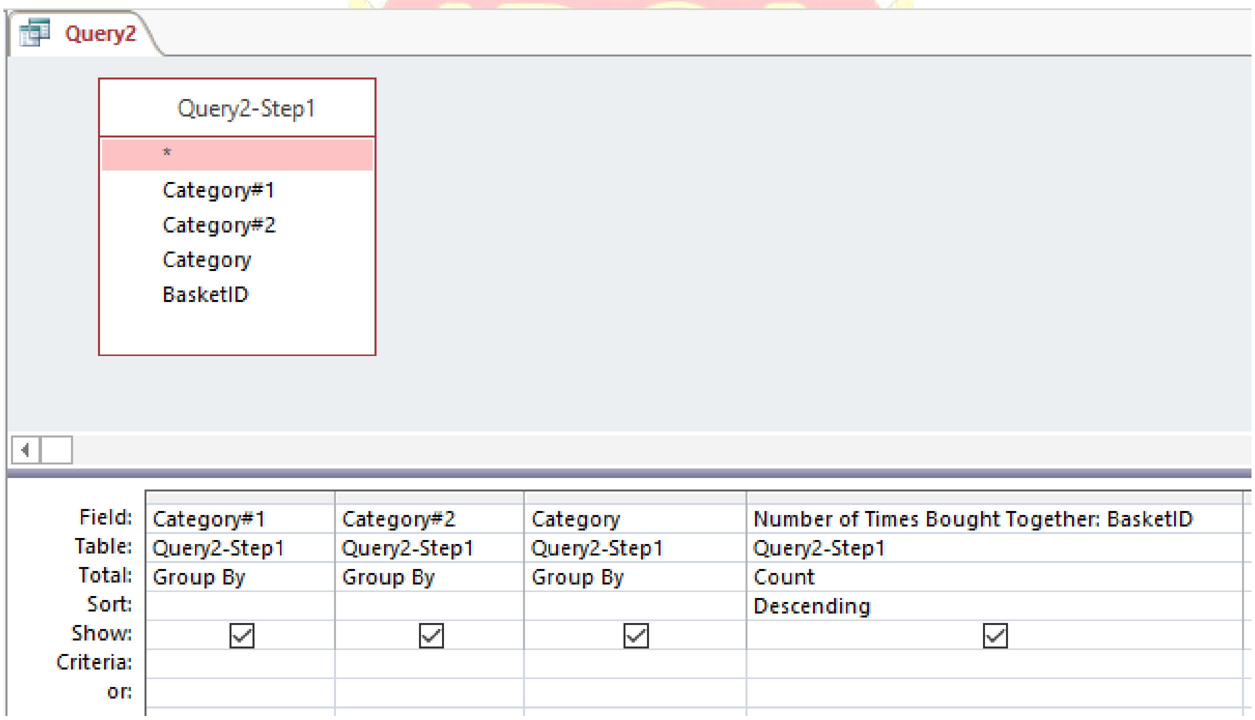
Since Microsoft Access does <u>not</u> support **Count Distinct** aggregation, a helping query utilizing **Select Distinct** is first used:



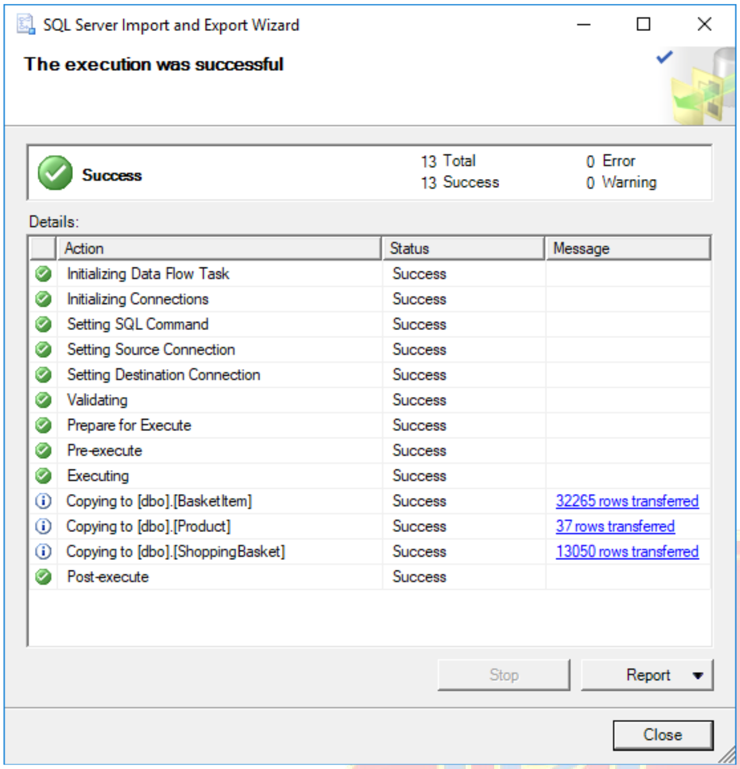


C. QBE query to list product category most bought along with a pair of product categories

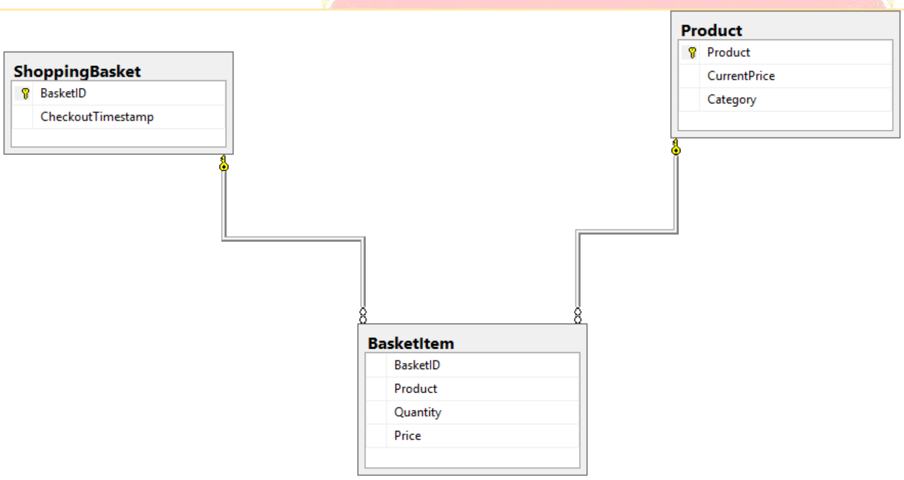




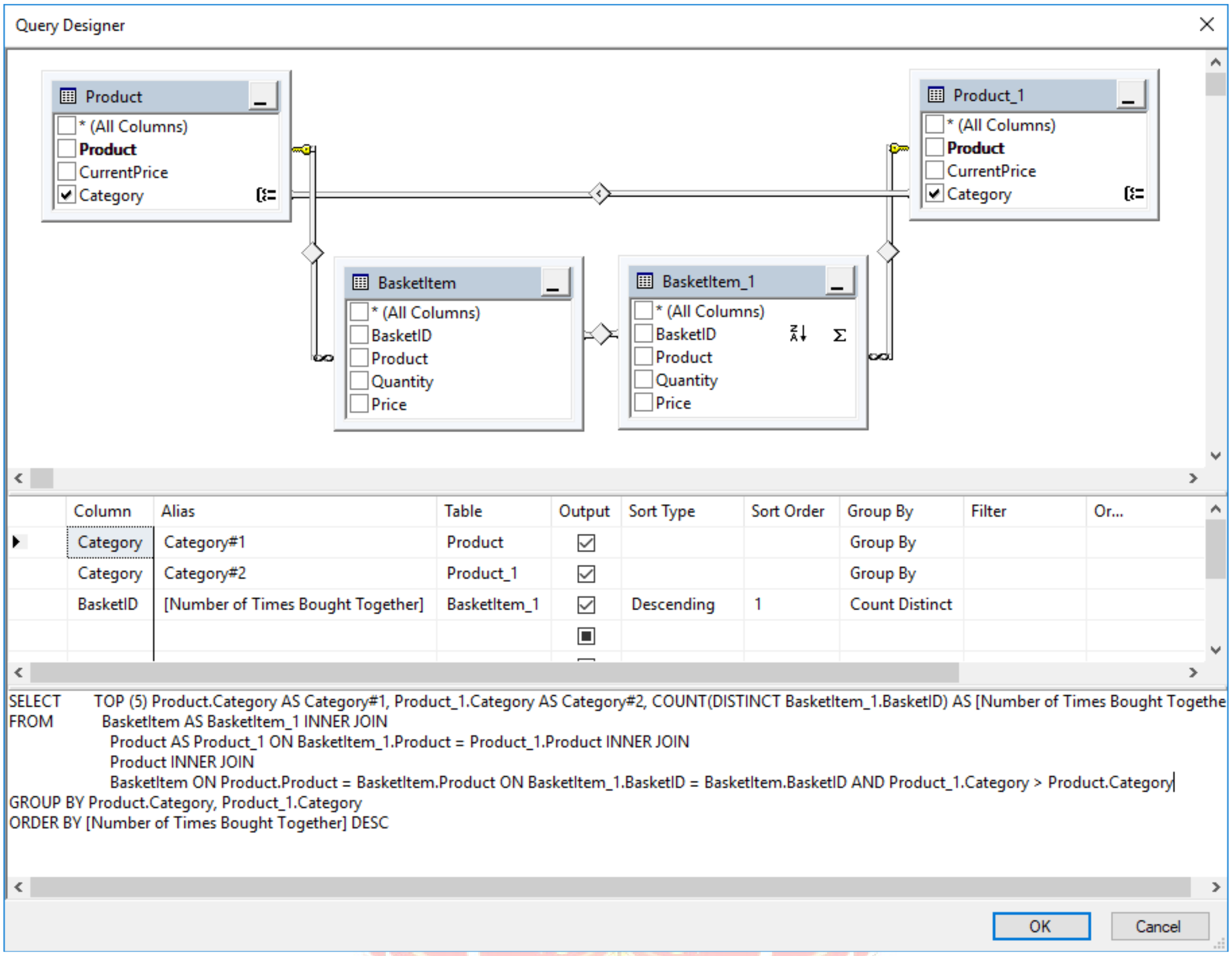
D. Importing an Access database into SQL Server



E. Defining the relationships between tables in SQL Server

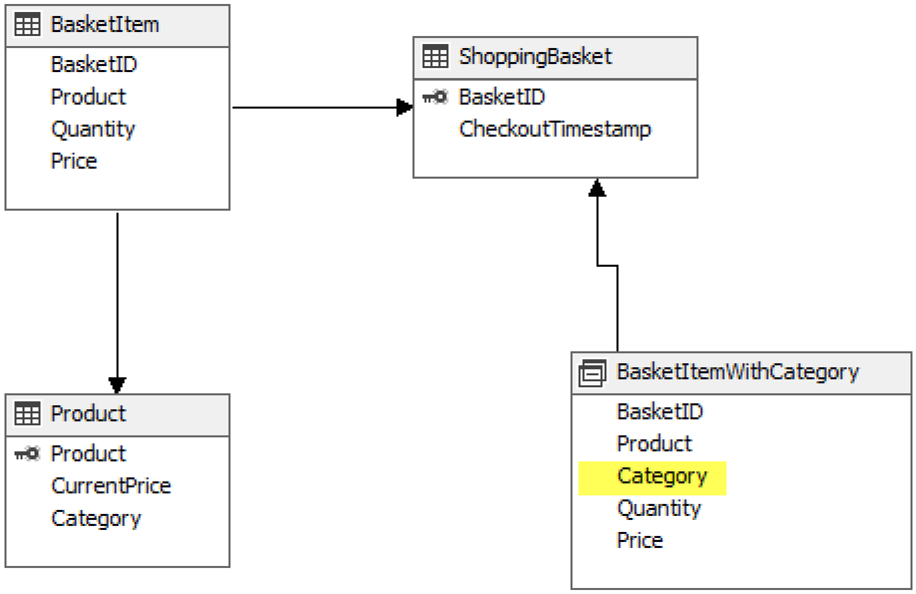


F. SQL query to list pairs of product categories most bought together

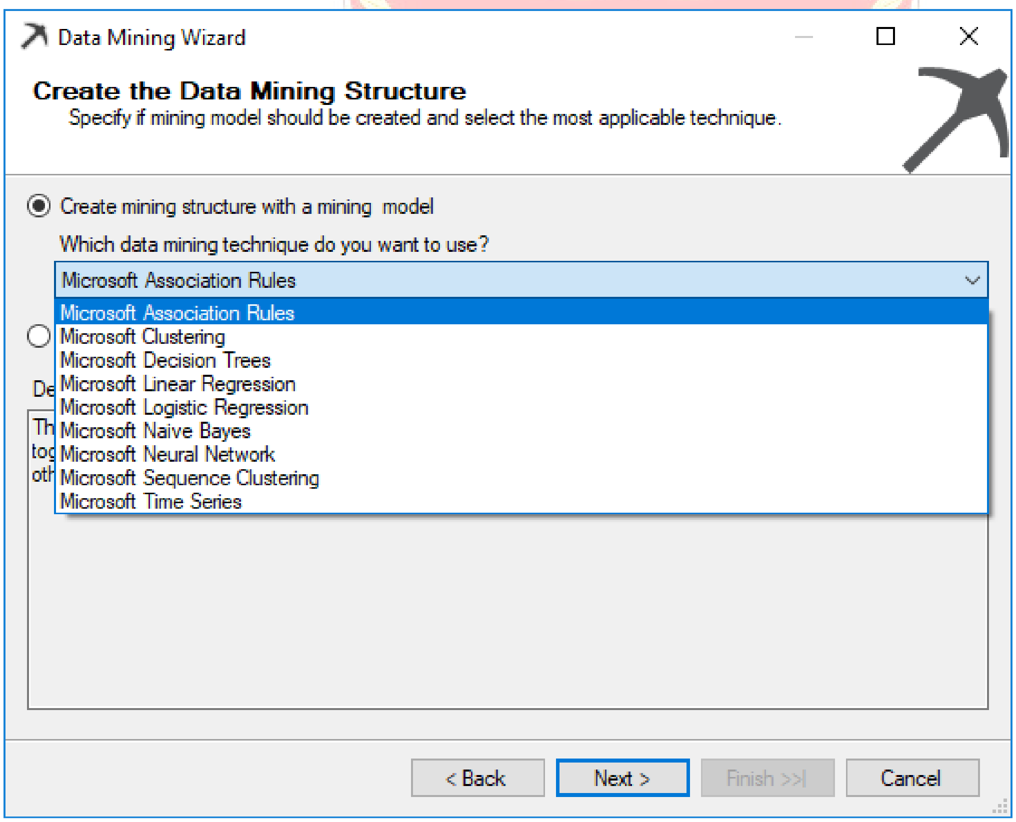



```
SELECT      TOP (5) Product.Category AS Category#1, Product_1.Category AS Category#2,
            COUNT(DISTINCT BasketItem_1.BasketID) AS [Number of Times Bought Together]
FROM        BasketItem AS BasketItem_1 INNER JOIN Product AS Product_1
                ON BasketItem_1.Product = Product_1.Product
                INNER JOIN Product
                    INNER JOIN BasketItem
                    ON Product.Product = BasketItem.Product
                ON BasketItem_1.BasketID = BasketItem.BasketID
                    AND Product_1.Category > Product.Category
GROUP BY    Product.Category, Product_1.Category
ORDER BY    [Number of Times Bought Together] DESC
```

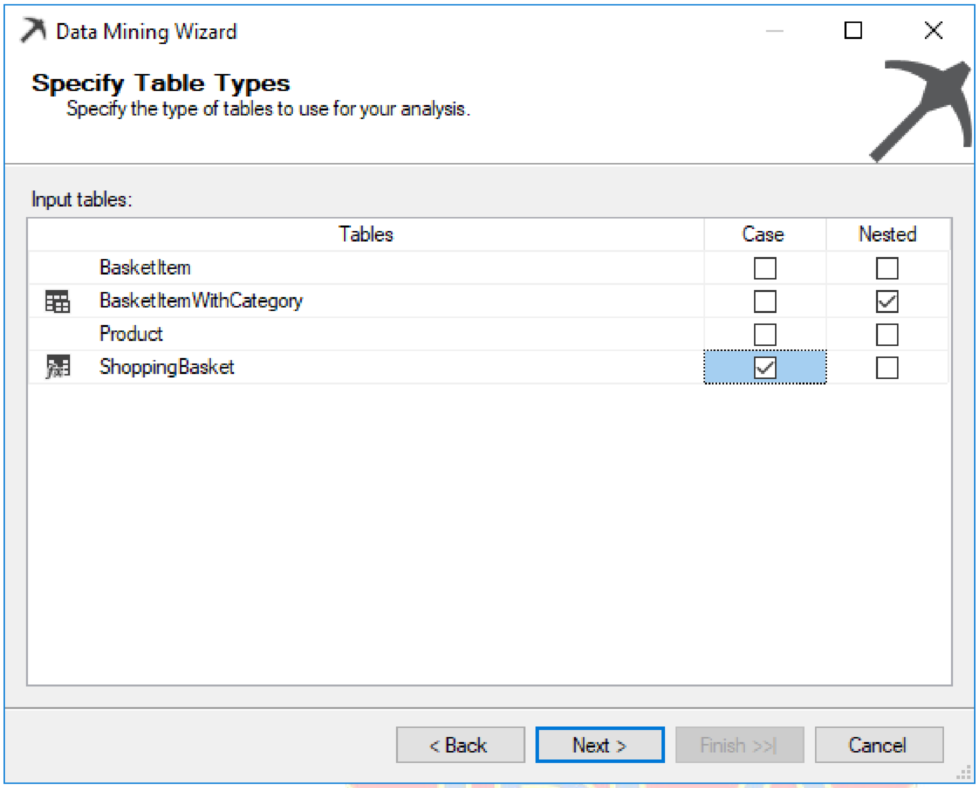
G. Creating a new named query (BasektItemWithCategory) in the data source view



H. Selecting the data mining algorithm to use



I. Specifying tables to be used by the data mining algorithm



J. SQL queries to show Probability(Tires and Tubes | Bike Stands, Road Bikes) = 1

The following SQL query returns 13 as number of baskets with a product in each category: Tires and Tubes, Bike Stands, and Road Bikes:

```
SELECT      COUNT(DISTINCT BasketID) AS [Number of Baskets]
FROM        ShoppingBasket AS X
WHERE       EXISTS
            (SELECT     *
            FROM        BasketItem INNER JOIN Product
                            ON BasketItem.Product = Product.Product
            WHERE       BasketItem.BasketID = X.BasketID
                        AND Product.Category = N'Tires and Tubes')

            AND EXISTS
            (SELECT     *
            FROM        BasketItem INNER JOIN Product
                            ON BasketItem.Product = Product.Product
            WHERE       BasketItem.BasketID = X.BasketID
                        AND Product.Category = N'Bike Stands')

            AND EXISTS
            (SELECT     *
            FROM        BasketItem INNER JOIN Product
                            ON BasketItem.Product = Product.Product
            WHERE       BasketItem.BasketID = X.BasketID
                        AND Product.Category = N'Road Bikes')
```

And, the following SQL query also returns 13 as number of baskets with a product in each category: Bike Stands, and Road Bikes:

```
SELECT      COUNT(DISTINCT BasketID) AS [Number of Baskets]
FROM        ShoppingBasket AS X
WHERE       EXISTS
            (SELECT     *
            FROM        BasketItem INNER JOIN Product
                                ON BasketItem.Product = Product.Product
            WHERE       BasketItem.BasketID = X.BasketID
                        AND Product.Category = N'Bike Stands')

            AND EXISTS
            (SELECT     *
            FROM        BasketItem INNER JOIN Product
                                ON BasketItem.Product = Product.Product
            WHERE       BasketItem.BasketID = X.BasketID
                        AND Product.Category = N'Road Bikes')
```

Therefore,
Probability(Tires and Tubes | Bike Stands, Road Bikes) =
P(Tires and Tubes, Bike Stands, Road Bikes) / P(Bike Stands, Road Bikes) =
Number of baskets with Tires and Tubes, Bike Stands, and Road Bikes /
Number of baskets with Bike Stands and Road Bikes = 13 / 13 = 100%

K. SQL queries to show Importance

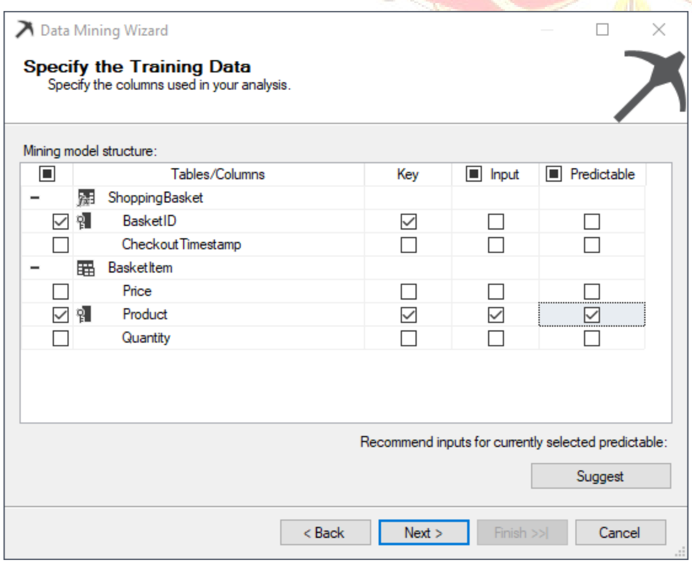
(Bike Stands, Road Bikes → Tires and Tubes) = 0.328
Importance(Bike Stands, Road Bikes → Tires and Tubes) =
Log (P(Tires and Tubes | <Bike Stands, Road Bikes>) / P(Tires and Tubes | NOT <Bike Stands, Road Bikes>))
= Log(1 / P(Tires and Tubes | NOT <Bike Stands, Road Bikes>))
We have,
P(Tires and Tubes | NOT <Bike Stands, Road Bikes>) =
P(Tires and Tubes and NOT <Bike Stands, Road Bikes>) /
P(NOT <Bike Stands, Road Bikes>)
The numerator,
# of baskets with Tires and Tubes and NOT < Bike Stands, Road Bikes > =
# of baskets with Tires and Tubes and No Bike Stands and No Road Bikes +
# of baskets with Tires and Tubes and Bike Stands and No Road Bikes +
# of baskets with Tires and Tubes and No Bike Stands and Road Bikes
The following SQL query gives the first count as 5,158:

```
SELECT      COUNT(DISTINCT BasketID) AS [Number of Baskets]
FROM        ShoppingBasket AS X
WHERE       EXISTS
            (SELECT     *
            FROM        BasketItem INNER JOIN Product
                                ON BasketItem.Product = Product.Product
            WHERE       BasketItem.BasketID = X.BasketID
                        AND Product.Category = N'Tires and Tubes')

            AND NOT EXISTS
            (SELECT     *
```

```
FROM        BasketItem INNER JOIN Product
                ON BasketItem.Product = Product.Product
WHERE       BasketItem.BasketID = X.BasketID
            AND Product.Category = N'Bike Stands')

AND NOT EXISTS
(SELECT     *
FROM        BasketItem INNER JOIN Product
                ON BasketItem.Product = Product.Product
WHERE       BasketItem.BasketID = X.BasketID
            AND Product.Category = N'Road Bikes')
```

Similar queries produce:
# of baskets with Tires and Tubes and NOT < Bike Stands, Road Bikes > =
5,158 + 90 + 473 = 5,721
The denominator,
# of baskets with NOT < Bike Stands, Road Bikes > =
# of baskets with No Bike Stands and No Road Bikes +
# of baskets with Bike Stands and No Road Bikes +
# of baskets with No Bike Stands and Road Bikes =
10,564 + 117 + 2,356 = 13,037
Therefore,
P(Tires and Tubes | NOT <Bike Stands, Road Bikes>) = 5,721 / 13,037 = 0.438
And,
Importance(Bike Stands, Road Bikes → Tires and Tubes) =
Log (P(Tires and Tubes | <Bike Stands, Road Bikes>) / P(Tires and Tubes | NOT <Bike Stands, Road Bikes>))
= Log(1 / P(Tires and Tubes | NOT <Bike Stands, Road Bikes>))
= Log(1 / 0.438) = Log(2.2788) = 0.357
which is close to 0.328!

L. Developing a data mining model for predicting products that are bought together



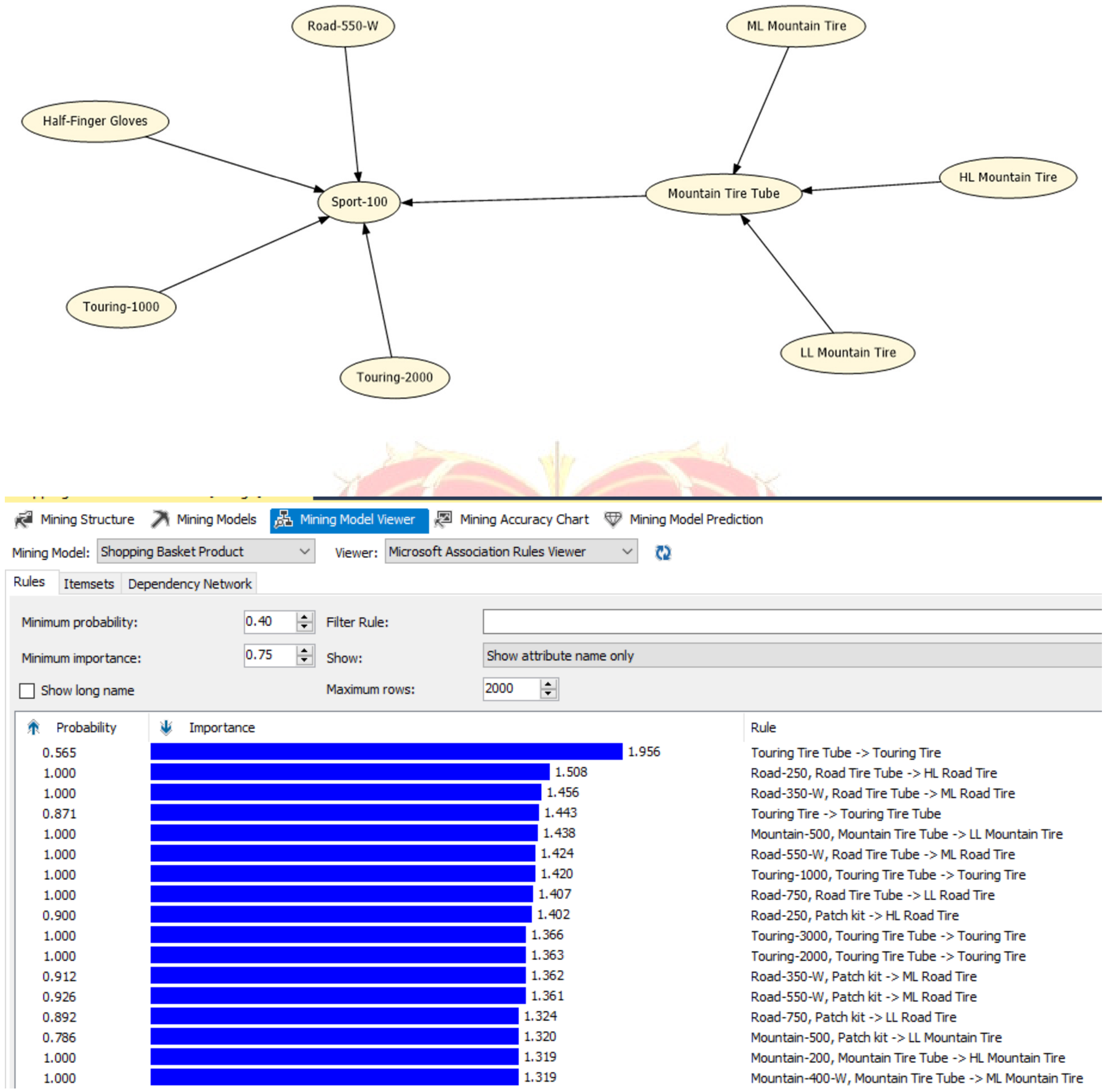


Figure A. Association Rules of Products Bought Together