



**UNIVERSITY
OF MALAYA**

WIE3007 DATA MINING AND WAREHOUSING

Semester 2 Session 2022/23

INDIVIDUAL ASSIGNMENT

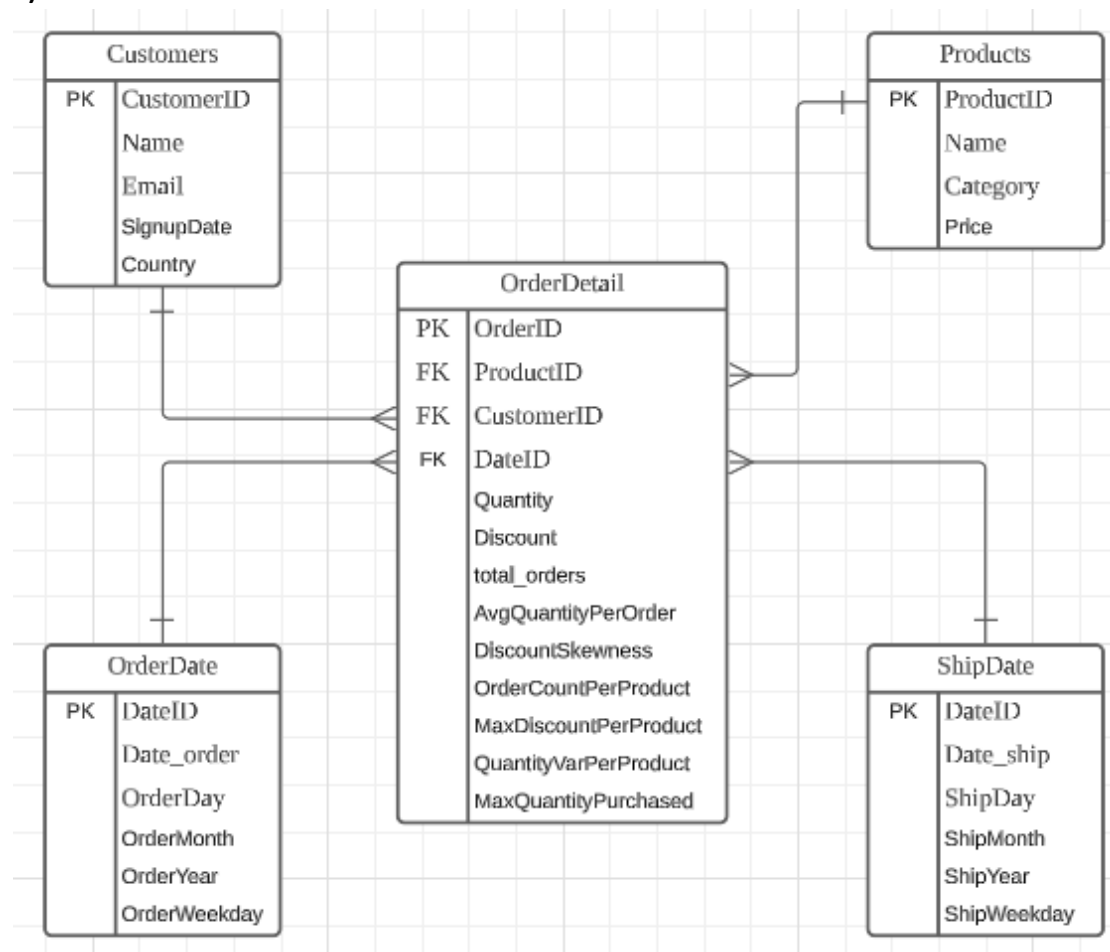
NAME	WAN SURAYA BINTI WAN MOHD LOTFI
MATRIC NO	U2005345/1
SUPERVISOR NAME	PROFESSOR DR. TEH YING WAH

Star Schema Diagram

Business Objective:

1. **Customer Segmentation:** Identify customer behavior patterns to enable effective customer segmentation.
2. **Product Performance Analysis:** Analyze trends within products to enhance product performance strategies.
3. **Time-Based Analysis:** Explore temporal patterns, such as day of the week or month, to understand the impact on sales and identify trends.

1) Star Schema



2) Keys in Star Schema:

Fact Table: OrderDetail

- Primary Key: OrderID
- Foreign Keys:
 - OrderID
 - ProductID
 - CustomerID
 - DateID
- Attributes
 - Quantity
 - Discount
- Aggregated Features:
 - total_orders
 - AvgQuantityPerOrder
 - DiscountSkewness
 - OrderCountPerProduct
 - MaxDiscountPerProduct
 - QuantityVarPerProduct
 - MaxQuantityPurchased

Dimension Tables:

1) Customers

- Primary Key
 - CustomerID
- Attributes:
 - Name
 - Email
 - SignupDate
 - Country

2) Products

- ProductID
- Name
- Category
- Price

3) OrderDate

- Primary Key
 - DateID
- Attributes
 - Date_order
- Temporal Features:
 - OrderDay
 - OrderMonth
 - OrderYear
 - OrderWeekday

4) ShipDate

- Primary Key
 - DateID
- Attributes
 - Date_ship
- Temporal Features:
 - ShipDay
 - ShipMonth
 - ShipYear
 - ShipWeekday

3) Star Schema Model Hierarchy

Fact table: OrderDetail

- 1) Primary Key
 - a. OrderID
- 2) Foreign Key
 - a. ProductID
 - b. CustomerID
 - c. DateID
- 3) Attributes
 - a. Quantity
 - b. Discount
- 4) Aggregated features
 - a. total_orders
 - b. AvgQuantityPerOrder
 - c. DiscountSkewness
 - d. MaxQuantityPurchased
 - e. OrderCountPerProduct
 - f. MaxDiscountPerProduct
 - g. QuantityVarPerProduct

Dimension table: Product

- 1) Primary Key: ProductID
- 2) Attributes:
 - a. Name
 - b. Category
 - c. Price

Dimension table: Customers

- 1) Primary Key: CustomerID
- 2) Attributes:
 - a. Name
 - b. Email
 - c. SignupDate
 - d. Country

Dimension table: OrderDate

- 1) PrimaryKey: DateID
- 2) Attributes:Date_order
- 3) Temporal features:
 - a. OrderDay
 - b. OrderMonth
 - c. OrderYear
 - d. OrderWeekday

Dimension table: ShipDate

- 1) PrimaryKey: DateID
- 2) Attributes:Date_ship
- 3) Temporal features:
 - a. ShipDay
 - b. ShipMonth
 - c. ShipYear
 - d. ShipWeekday

4) Star Schema Relationships

Dimension Table: Product and Fact table: OrderDetail

Relationship: One-to-Many

Key: ProductID in Product table link to ProductID in OrderDetail table

Dimension Table: Customers and Fact table: OrderDetail

Relationship: One-to-Many

Key: CustomerID in Customer table link to CustomerID in OrderDetail table

Dimension Table: OrderDate and Fact table: OrderDetail

Relationship: One-to-Many

Key: DateID in OrderDate table link to DateID in OrderDetail table

Dimension Table: ShipDate and Fact table: OrderDetail

Relationship: One-to-Many

Key: DateID in ShipDate table link to DateID in OrderDetail table

Data Dictionary

Fact Table: Orders

- **OrderID**
 - Data Type: Integer
 - Constraints: Primary Key
- **ProductID**
 - Data Type: Integer
 - Constraints: Foreign Key (references Products(ProductID))
- **CustomerID**
 - Data Type: Integer
 - Constraints: Foreign Key (references Customers(CustomerID))
- **DateID**
 - Data Type: Integer
 - Constraints: Foreign Key (references OrderTime(DateID))
- **Quantity**
 - Data Type: Integer
 - Constraints: Not null, non-negative values
- **Discount**
 - Data Type: Float
 - Constraints: Not null, range [0, 1]
- **total_orders**
 - Data Type: Integer
 - Constraints: Not null. non-negative values
 - The count of orders for each customer
- **AvgQuantityPerOrder (New feature from MEAN(order_details.Discount) from customer entity retrieved by Deep Feature Synthesis)**
 - Data Type: Float
 - Constraints: Not null
 - The average quantity of items purchased in each order for each customer.
- **DiscountSkewness (New feature from SKEW(order_details.Discount) from customer entity retrieved by Deep Feature Synthesis)**
 - Data Type: Float
 - Constraints: Not null
 - The skewness of the distribution of discounts in each customer's orders
- **OrderCountPerProduct (New feature from COUNT(order_details) from product entity retrieved by Deep Feature Synthesis)**
 - Data Type: Integer
 - Constraints: Not null, non-negative values
 - Quantity of each product sold.
- **MaxDiscountPerProduct (New feature from MAX(order_details.Discount) from product entity retrieved by Deep Feature Synthesis)**
 - Data Type: Float
 - Constraints: Not null
 - The maximum discount a product has received.

-
- **QuantityVarPerProduct (New feature from STD(order_details.Quantity): from product entity retrieved by Deep Feature Synthesis)**
 - Data Type: Float
 - Constraints: Not null
 - Measures the variability in customer purchasing behaviour for a product,
- **MaxQuantityPurchased (New feature from MAX(order_details.Quantity) from customer entity retrieved by Deep Feature Synthesis)**
 - Data Type: Integer
 - Constraints: Not null, non-negative values
 - The maximum quantity of items purchased in any order for each customer.

Dimension Table: Customers

- **CustomerID**
 - Data Type: Integer
 - Constraints: Primary Key
- **Name**
 - Data Type: String
 - Constraints: Not null
- **Email**
 - Data Type: String
 - Constraints: Not null, unique
- **SignupDate**
 - Data Type: Date
 - Constraints: Not null

Dimension Table: Products

- **ProductID**
 - Data Type: Integer
 - Constraints: Primary Key
- **Name**
 - Data Type: String
 - Constraints: Not null
- **Category**
 - Data Type: String
 - Constraints: Not null
- **Price**
 - Data Type: Float
 - Constraints: Not null, non-negative values

Dimension Table: ShipDate

- **DateID**
 - Data Type: Integer
 - Constraints: Primary Key
- **Date_ship**
 - Data Type: Date
 - Constraints: Not null
- **ShipDay (New feature from deep feature synthesis)**
 - Data Type: Integer
 - Constraints: Not null, range [1, 31]
- **ShipMonth (New feature from deep feature synthesis)**
 - Data Type: Integer

- Constraints: Not null, range [1, 12]
- **ShipYear(New feature from deep feature synthesis)**
 - Data Type: Integer
 - Constraints: Not null
- **ShipWeekday(New feature from deep feature synthesis)**
 - Data Type: Integer
 - Constraints: Not null

Dimension Table: OrderDate

- **DateID**
 - Data Type: Integer
 - Constraints: Primary Key
- **Date_order**
 - Data Type: Date
 - Constraints: Not null
- **OrderDay(New feature from deep feature synthesis)**
 - Data Type: Integer
 - Constraints: Not null, range [1, 31]
- **OrderMonth(New feature from deep feature synthesis)**
 - Data Type: Integer
 - Constraints: Not null, range [1, 12]
- **OrderYear(New feature from deep feature synthesis)**
 - Data Type: Integer
 - Constraints: Not null
- **OrderWeekday(New feature from deep feature synthesis)**
 - Data Type: Integer
 - Constraints: Not null

Insight Report

The dataset used for featuretools:

Customers:

CustomerID: [101, 102, 103]

Name: ['John Doe', 'Jane Smith', 'Mike Jordan']

Email: ['john.doe@example.com', 'jane.smith@example.com', 'mike.jordan@example.com']

SignupDate: ['2023-01-10', '2023-01-15', '2023-01-20']

Products:

ProductID: [201, 202, 203]

Name: ['Laptop', 'Tablet', 'Smartphone']

Category: ['Electronics', 'Electronics', 'Electronics']

Price: [1000, 500, 800]

Orders:

OrderID: [301, 302, 303]

CustomerID: [101, 102, 103]

OrderDate: ['2023-02-01', '2023-02-05', '2023-02-10']

ShipDate: ['2023-02-03', '2023-02-07', '2023-02-12']

OrderDetails:

OrderID: [301, 302, 303]

ProductID: [201, 202, 203]

Quantity: [1, 2, 1]

Discount: [0, 0.1, 0]

Chosen Columns and Reasoning

Business Objective 1 - Customer Segmentation: Identify customer behaviour patterns to enable effective customer segmentation.

In order to do customer segmentation, it is important to get attributes that shows behaviours of customers, as we typically want to identify features that capture meaningful differences among customers. Feature tools have provided several possible attributes to achieve this business objective and from the output, insights were made and utilized in developing the star schema.

Features taken from Customers entity.

```
[ ] feature_matrix, feature_defs = ft.dfs(entityset=es, target_dataframe_name="customers")
feature_matrix
```

	COUNT(orders)	COUNT(order_details)	MAX(order_details.Discount)	MAX(order_details.Quantity)	MEAN(order_details.Discount)	MEAN(order_details.Quantity)	MIN(order_details)
CustomerID							
101	4	4	0.10	3.0	0.050000	2.250000	
102	6	6	0.10	5.0	0.033333	2.166667	
103	4	4	0.20	5.0	0.137500	3.500000	
104	3	3	0.10	3.0	0.033333	2.000000	
105	2	2	0.20	4.0	0.175000	2.500000	

1. COUNT(orders): The count of orders for each customer

Example business solution: The business can identify and prioritize customers with a high number of orders. These customers may be targeted for loyalty programs, exclusive offers, or personalized marketing campaigns to encourage them to continue making frequent purchases.

Star Schema component: OrderDetail Fact Table - total_orders

Utilization: *The count of orders for each customer can be a fact in the fact table related to the Customer dimension table. This can help analyse each customer's total number of orders placed by customers and identify customers who are frequent buyers versus those who make fewer purchases.*

2. MEAN(order_details.Quantity): The average quantity of items purchased in each order for each customer. This could highlight segments of customers who consistently buy in large or small quantities.

Example business solution: Understanding the average quantity of items purchased by different customer segments can help the business optimize inventory management. For customers who consistently buy in large quantities, the business might offer bulk discounts or suggest related products. Conversely, for those who prefer smaller quantities, the focus could be on smaller, more frequent purchases.

Star Schema Component: OrderDetail Fact Table - AvgQuantityPerOrder

Utilization: *Similar to the discount, include the average quantity as a measure in the fact table. This allows for analysis of the quantity preferences of different customer segments.*

3. **SKEW(order_details.Discount):** The skewness of the distribution of discounts in each customer's orders. Positive skewness may indicate a preference for discounted items.

Example business solution: The business can leverage this insight to create targeted marketing campaigns promoting discounted products to specific customer segments, potentially increasing sales in these categories.

Star Schema Component: OrderDetail Fact Table - DiscountSkewness

Utilization: *Insert the skewness measure into the fact table and connect it with the customer dimension table to capture the distribution of discounts for each customer. This can help in segmenting customers based on their preference for discounted items*

4. **MAX(order_details.Quantity) – The maximum quantity of items purchased in any order for each customer.**

Example business solution: For customers identified with occasional large purchases, the business can tailor specific strategies like personalized recommendations.

Recommending complementary items or products frequently purchased together can enhance the overall shopping experience and encourage additional spending.

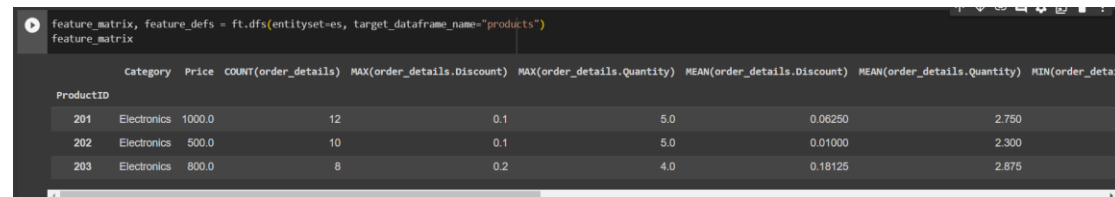
Star Schema Component: OrderDetail Fact Table - MaxQuantityPurchased

Utilization: *Insert the maximum quantity of items in any order of each customer into the fact table and connect it to the customer dimension table. This way, we can identify customers who occasionally make large purchases.*

Business Objective 2 - Product Performance Analysis: Analyze trends within products to enhance product performance strategies.

In order for a business to enhance their product performance strategies, it is important to acquire attributes that effectively capture the dynamics of product behaviours. With the help of featuretools, a range of attributes that offer insights into product trends were shown.

Features taken from Products entity:



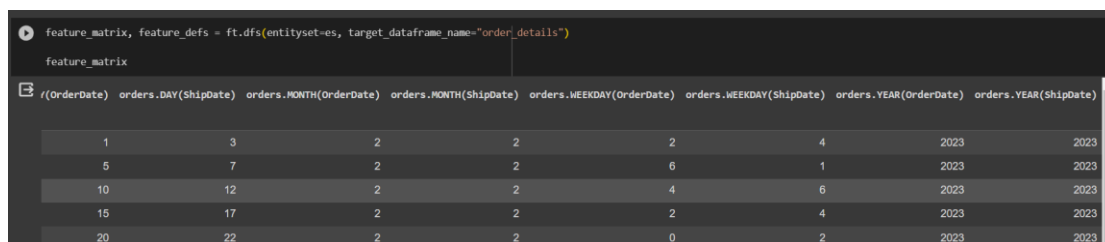
ProductID	Category	Price	COUNT(order_details)	MAX(order_details.Discount)	MAX(order_details.Quantity)	MEAN(order_details.Discount)	MEAN(order_details.Quantity)	MIN(order_details.Quantity)
201	Electronics	1000.0	12	0.1	5.0	0.06250	2.750	1
202	Electronics	500.0	10	0.1	5.0	0.01000	2.300	1
203	Electronics	800.0	8	0.2	4.0	0.18125	2.875	1

- COUNT(order_details) per product:** Provides information on the quantity of each product sold.
Example Business solution: Can help with inventory planning. By understanding the COUNT(order_details) per product, the business can optimize inventory planning for high-demand products, ensuring that popular items are adequately stocked to meet customer demand.
Star Schema Component: OrderDetail Fact Table - OrderCountPerProduct
Utilization: *The measure of number of times the product has been ordered shall be in the fact table and connected to the Product dimension table. This way, we can assess the popularity of products.*
- MAX(order_details.Discount):** Provides insights into the maximum discount a product has received.
Example business solution: Analyzing MAX(order_details.Discount) can inform promotional strategies. Products with a history of responding well to discounts may be targeted in flash sales or seasonal promotions.
Star Schema Component: OrderDetail Fact Table - MaxDiscountPerProduct
Utilization: *The measure of maximum discount applied to a product can be inserted into a fact table and connected to the Product dimension table to gain insight into the maximum discount a product has received. This information can be useful in understanding how price sensitivity affects the performance of different products.*
- STD(order_details.Quantity):** Measures the variability in customer purchasing behaviour for a product, aiding in inventory management and understanding demand consistency. High standard deviation means high variability in the number of units ordered for that particular product.
Example business solution: Monitoring STD(order_details.Quantity) can help the business identify products with inconsistent demand. This information can be used to implement quality control measures and ensure customer satisfaction by avoiding stockouts or overstock situations.
Star Schema Component: OrderDetail Fact Table - QuantityVarPerProduct
Utilization: *The measure of standard deviation of quantities for the product can be inserted into the fact table and connect it with the Product dimension table to gain insight on variability of product demand.*

Business Objective 3 - Time-Based Analysis: Explore temporal patterns, such as day of the week or month, to understand the impact on sales and identify trends.

The business can further extend its capabilities with a distinct focus on time-based analysis. In the pursuit of refining time-based analysis, it becomes crucial to acquire attributes that capture dynamic aspects of temporal behaviours. Featuretools has provided time-related attributes from the order_details entity and provided a comprehensive view of the temporal aspects associated with customer orders dates and shipments dates.

Features taken from order_details entity:



(OrderDate)	orders.DAY(ShipDate)	orders.MONTH(OrderDate)	orders.MONTH(ShipDate)	orders.WEEKDAY(OrderDate)	orders.WEEKDAY(ShipDate)	orders.YEAR(OrderDate)	orders.YEAR(ShipDate)
1	3	2	2	2	4	2023	2023
5	7	2	2	6	1	2023	2023
10	12	2	2	4	6	2023	2023
15	17	2	2	2	4	2023	2023
20	22	2	2	0	2	2023	2023

1. Transforming Orders entity into a Time Dimension Table:

- Original entity:
Orders:
 - a. OrderID
 - b. CustomerID
 - c. OrderDate
 - d. ShipDate
- Utilize features like:
 1. orders.DAY(OrderDate)
 2. orders.DAY(ShipDate)
 3. orders.MONTH(OrderDate)
 4. orders.MONTH(ShipDate)
 5. orders.WEEKDAY(OrderDate)
 6. orders.WEEKDAY(ShipDate)
 7. orders.YEAR(OrderDate)
 8. orders.YEAR(ShipDate)
- The separation of day, month, weekday, and year provides a more granular view of temporal patterns within the data. Each of the components offers unique insights and facilitates specific types of analysis:
 1. Day
 - Usage: Analysing data based on the day of the month helps identify patterns related to specific days, such as the beginning or end of the month.
 - Example: Identifying whether certain products or promotions are more popular on specific days of the month
 2. Month
 - Usage: Monthly analysis allows observation of trends and seasonality that occur over longer periods

- Example: Identifying if there are months with higher sales, indicating seasonality or special events

3. Weekday

- Usage: Understand patterns based on weekdays helps capture variations related to different days of the week
- Example: Identifying if there are specific weekdays when certain products are more frequently purchased, indicating potential promotional opportunities.

4. Year

- Usage: Analysing data on a yearly basis helps identify long-term trends and provides insights into annual variations
- Example: Observing whether there is consistent growth in sales over the years or detecting any anomalies or exceptional years.

- Time-based analysis's division of day, month, weekday, and year offers a focused and detailed perspective of temporal patterns, enabling nuanced insights into daily, monthly, weekly, and yearly trends for better decision-making. This generates a comprehensive time dimension table, which facilitates effective trend analysis.

2. Separation of Orders table into ShipDate and OrderDate:

The decision to separate shipDate and OrderDate into two-dimension tables in the star schema is driven by a strategic optimization for efficient time-based analysis. The feature tools provided distinct date-related attributes for both shipping and ordering events, thus the separation provides granularity and specificity in the time-based analysis.

The primary use cases involve analyzing either ordering or shipping events independently, and the related date-related attributes are significantly different for each, then keeping them separate offers more flexibility and clarity in this analysis.

This separation aims to enhance query performance by simplifying and focusing queries related to either shipping or ordering processes, reducing join complexity and streamlining aggregations. It provides a modular and adaptable structure, aligning with a natural organizational hierarchy in the data. The design choice is strategic, showing a deliberate and forward-looking approach to the whole database architecture and seeking to satisfy present analytical demands while predicting future requirements.

3. The inclusion of this time dimension tables (ShipDate and OrderDate) in the star schema enables each measure in the fact table to be associated with specific temporal information. This dimension allows measures to specify the time at which they typically occur, whether it be on a particular day, month, year, or weekday. With this, it achieves the business objective to explore patterns over time, understand their impact on sales and identify trends that contribute to more informed decision-making.
4. **Example business solution:** By connecting the OrderDate dimension table with MAX(order_details.Quantity) - MaxQuantityPurchased attribute in the Orders Fact Table, customers with occasional large purchases may represent a different segment. The business can develop strategies to encourage and capitalize on these sporadic high-volume transactions at a particular time, such as offering exclusive deals during peak purchasing times.

Summary:

In a star schema, the central fact table is surrounded by dimension tables. The dimensions provide context to the measures in the fact table, allowing for easy and efficient querying and analysis.

In this context, the central fact table chosen is from the order_detail entity due to it containing measures and numerical values like Quantity and Discount. After performing featuretools, I have added more features as measures into the fact table in order to have a more quality star schema that achieves the business objectives. The features gained from feature tools are:

1) From customer entity:

1. COUNT(orders)
2. MEAN(order_details.Quantity)
3. SKEW(order_details.Discount)
4. MAX(order_details.Quantity)

2) From products entity:

1. COUNT(order_details)
2. MAX(order_details.Discount)
3. STD(order_details.Quantity)

In the context of dimension tables, it is often preferable to include descriptive attributes such as temporal features. In this case, the original Orders table, which included ShipDate and OrderDate, has been restructured into two distinct time dimension tables: ShipDate and OrderDate. These dimension tables are linked to a DateID primary key in the fact table, facilitating a more organized and efficient representation of temporal information in the overall data model.

3) From orders entity:

1. orders.DAY(OrderDate)
2. orders.DAY(ShipDate)
3. orders.MONTH(OrderDate)
4. orders.MONTH(ShipDate)
5. orders.WEEKDAY(OrderDate)
6. orders.WEEKDAY(ShipDate)
7. orders.YEAR(OrderDate)
8. orders.YEAR(ShipDate)

The star schema design enables the business to perform complex queries and aggregations efficiently. For example, it allows us to easily answer questions like "What is the average discount preference for customers in a specific region?" or "Which customer segment makes occasional large purchases and at what month?"

Python code

Installation

```
!pip install featuretools
import featuretools as ft
import pandas as pd
```

Creating dataframe

Note: More rows were added in the code to further understand the features provided by featuretools.

```
import pandas as pd

# Create DataFrames for each entity
customers_df = pd.DataFrame({
    'CustomerID': [101, 102, 103, 104, 105, 106, 107, 108, 109,
110, 111, 112, 113],
    'Name': ['John Doe', 'Jane Smith', 'Mike Jordan', 'Chris
Evans', 'Emily Taylor', 'David Johnson', 'Samantha White', 'Kevin
Brown',
            'Laura Davis', 'Alex Turner', 'Eva Martinez', 'Brian
Miller', 'Olivia Wilson'],
    'Email': ['john.doe@example.com', 'jane.smith@example.com',
'mike.jordan@example.com', 'chris.evans@example.com',
'emily.taylor@example.com', 'david.johnson@example.com',
            'samantha.white@example.com',
'kevin.brown@example.com', 'laura.davis@example.com',
            'alex.turner@example.com',
'eva.martinez@example.com', 'brian.miller@example.com',
            'olivia.wilson@example.com'],
    'SignupDate': ['2023-01-10', '2023-01-15', '2023-01-20',
'2023-02-05', '2023-02-10', '2023-02-15', '2023-02-20', '2023-02-
25',
                '2023-03-01', '2023-03-05', '2023-03-10',
'2023-03-15', '2023-03-20']
})

# Convert data types for customers_df
customers_df['CustomerID'] =
customers_df['CustomerID'].astype('int')
customers_df['Name'] = customers_df['Name'].astype('str')
customers_df['Email'] = customers_df['Email'].astype('string')
customers_df['SignupDate'] =
pd.to_datetime(customers_df['SignupDate'])

products_df = pd.DataFrame({
    'ProductID': [201, 202, 203],
    'Name': ['Laptop', 'Tablet', 'Smartphone'],
```

```

        'Category': ['Electronics', 'Electronics', 'Electronics'],
        'Price': [1000, 500, 800]
    })

# Convert data types for products_df
products_df['ProductID'] = products_df['ProductID'].astype('int')
products_df['Name'] = products_df['Name'].astype('str')
products_df['Category'] =
products_df['Category'].astype('category')
products_df['Price'] = products_df['Price'].astype('float')

orders_df = pd.DataFrame({
    'OrderID': [301, 302, 303, 304, 305, 306, 307, 308, 309, 310,
311, 312, 313, 314, 315, 316, 317, 318, 319, 320,
321, 322, 323, 324, 325, 326, 327, 328, 329, 330
],
    'CustomerID': [101, 102, 103, 101, 102, 103, 109, 104, 105,
102, 101, 102, 103, 101, 102, 103, 109, 104, 105, 102,
108, 110, 111, 113, 112, 107, 106, 113, 111,
104],
    'OrderDate': ['2023-02-01', '2023-02-05', '2023-02-10',
'2023-02-15', '2023-02-20', '2023-02-25', '2023-03-01', '2023-03-
05', '2023-03-10', '2023-03-15', '2023-03-20', '2023-03-25',
'2023-03-30', '2023-04-01', '2023-04-05',
'2023-04-10', '2023-04-15', '2023-04-20',
'2023-04-25', '2023-04-30',
'2023-05-01', '2023-05-05', '2023-05-10',
'2023-05-15', '2023-05-20',
'2023-05-25', '2023-05-30', '2023-06-01',
'2023-06-05', '2023-06-10'],
    'ShipDate': ['2023-02-03', '2023-02-07', '2023-02-12', '2023-
02-17', '2023-02-22', '2023-02-27', '2023-03-03', '2023-03-07',
'2023-03-12', '2023-03-17', '2023-03-22', '2023-03-27', '2023-04-
01', '2023-04-03', '2023-04-07',
'2023-04-12', '2023-04-17', '2023-04-22', '2023-
04-27', '2023-05-02',
'2023-05-03', '2023-05-07', '2023-05-12', '2023-
05-17', '2023-05-22',
'2023-05-27', '2023-06-01', '2023-06-03', '2023-
06-07', '2023-06-12']
})

# Convert data types for orders_df
orders_df['OrderID'] = orders_df['OrderID'].astype('int')
orders_df['CustomerID'] = orders_df['CustomerID'].astype('int')
orders_df['OrderDate'] = pd.to_datetime(orders_df['OrderDate'])
orders_df['ShipDate'] = pd.to_datetime(orders_df['ShipDate'])

```

```

order_details_df = pd.DataFrame({
    'OrderID': [301, 302, 303, 304, 305, 306, 307, 308, 309, 310,
311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323,
324, 325, 326, 327, 328, 329, 330],
    'ProductID': [201, 202, 202, 201, 202, 203, 201, 202, 203,
201, 201, 202, 203, 201, 202, 203, 201, 202, 203, 201,201, 202,
203, 201, 202, 203, 201, 202, 203, 201 ],
    'Quantity': [1, 2, 5, 3, 1, 4, 2, 3, 1, 5, 3, 1, 4, 2, 3, 1,
5, 2, 4, 1, 3, 1, 4, 2, 3, 1, 5, 2, 4, 1],
    'Discount': [0, 0.1, 0, 0.05, 0, 0.2, 0.1, 0, 0.15, 0, 0.05,
0, 0.2, 0.1, 0, 0.15, 0.05, 0, 0.2, 0.1,0.05, 0, 0.2, 0.1, 0,
0.15, 0.05, 0, 0.2, 0.1]
})

# Convert data types for order_details_df
order_details_df['OrderID'] =
order_details_df['OrderID'].astype('int')
order_details_df['ProductID'] =
order_details_df['ProductID'].astype('int')
order_details_df['Quantity'] =
order_details_df['Quantity'].astype('int')
order_details_df['Discount'] =
order_details_df['Discount'].astype('float')

```

Creating an EntitySet and adding the dataframes

```

es = ft.EntitySet(id="the_entityset")
from woodwork.logical_types import Categorical, PostalCode

# Adding customers dataframe
es = es.add_dataframe(
    dataframe_name="customers",
    dataframe=customers_df,
    index="CustomerID",
    time_index="SignupDate",
)

# Adding products dataframe
es = es.add_dataframe(
    dataframe_name="products",
    dataframe=products_df,
    index="ProductID",
)

# Adding orders dataframe
es = es.add_dataframe(
    dataframe_name="orders",
    dataframe=orders_df,
    index="OrderID",
)

```

```

        time_index="OrderDate",
    )

# Adding order_details dataframe
es = es.add_dataframe(
    dataframe_name="order_details",
    dataframe=order_details_df,
    index='new_index',
    make_index=True,
)

# Displaying the EntitySet
es

```

Output:

```

➡ Entityset: the_entityset
  DataFrames:
    customers [Rows: 13, Columns: 4]
    products [Rows: 3, Columns: 4]
    orders [Rows: 30, Columns: 4]
    order_details [Rows: 30, Columns: 5]
  Relationships:
    No relationships

```

Adding relationships

```

es = es.add_relationship("customers", "CustomerID", "orders",
    "CustomerID")
es
es = es.add_relationship("products", "ProductID",
    "order_details", "ProductID")
es
es.add_relationship("orders", "OrderID", "order_details",
    "OrderID")
es

```

Output:

```

Entityset: the_entityset
DataFrames:
  customers [Rows: 13, Columns: 4]
  products [Rows: 3, Columns: 4]
  orders [Rows: 30, Columns: 4]
  order_details [Rows: 30, Columns: 5]
Relationships:
  orders.CustomerID -> customers.CustomerID
  order_details.ProductID -> products.ProductID
  order_details.OrderID -> orders.OrderID

```

Performing feature_matrix on 'customers' entity or dataframe

```

feature_matrix, feature_defs = ft.dfs(entityset=es,
target_dataframe_name="customers")
feature_matrix

```

Output:

CustomerID	COUNT(orders)	COUNT(order_details)	MAX(order_details.Discount)	MAX(order_details.Quantity)	MEAN(order_details.Discount)	MEAN(order_details.Quantity)	MIN(order_details)
101	4	4	0.10	3.0	0.050000	2.250000	
102	6	6	0.10	5.0	0.033333	2.166667	
103	4	4	0.20	5.0	0.137500	3.500000	
104	3	3	0.10	3.0	0.033333	2.000000	
105	2	2	0.20	4.0	0.175000	2.500000	
106	1	1	0.05	5.0	0.050000	5.000000	
107	1	1	0.15	1.0	0.150000	1.000000	
108	1	1	0.05	3.0	0.050000	3.000000	
109	2	2	0.10	5.0	0.075000	3.500000	
110	1	1	0.00	1.0	0.000000	1.000000	
111	2	2	0.20	4.0	0.200000	4.000000	
112	1	1	0.00	3.0	0.000000	3.000000	
113	2	2	0.10	2.0	0.050000	2.000000	

13 rows x 101 columns

Performing feature_matrix on 'products' entity or dataframe

```

feature_matrix, feature_defs = ft.dfs(entityset=es,
target_dataframe_name="products")
feature_matrix

```

Output:

ProductID	Category	Price	COUNT(order_details)	MAX(order_details.Discount)	MAX(order_details.Quantity)	MEAN(order_details.Discount)	MEAN(order_details.Quantity)	MIN(order_details)
201	Electronics	1000.0	12	0.1	5.0	0.06250	2.750	
202	Electronics	500.0	10	0.1	5.0	0.01000	2.300	
203	Electronics	800.0	8	0.2	4.0	0.18125	2.875	

Performing feature_matrix on 'orders' entity or dataframe

```

feature_matrix, feature_defs = ft.dfs(entityset=es,
target_dataframe_name="orders")
feature_matrix

```

Output:

	CustomerID	COUNT(order_details)	MAX(order_details.Discount)	MAX(order_details.Quantity)	MEAN(order_details.Discount)	MEAN(order_details.Quantity)	MIN(order_details.Discount)
OrderID							
301	101	1	0.00	1.0	0.00	1.0	
302	102	1	0.10	2.0	0.10	2.0	
303	103	1	0.00	5.0	0.00	5.0	
304	101	1	0.05	3.0	0.05	3.0	
305	102	1	0.00	1.0	0.00	1.0	
306	103	1	0.20	4.0	0.20	4.0	
307	109	1	0.10	2.0	0.10	2.0	
308	104	1	0.00	3.0	0.00	3.0	
309	105	1	0.15	1.0	0.15	1.0	
310	102	1	0.00	5.0	0.00	5.0	

Performing feature_matrix on 'order_detail' entity or dataframe

```
feature_matrix, feature_defs = ft.dfs(entityset=es,  
target_dataframe_name="order_details")  
feature_matrix
```

Output:

	OrderID	ProductID	Quantity	Discount	products.Category	products.Price	orders.CustomerID	products.COUNT(order_details)	products.MAX(order_details.Discount)	products
new_index										
0	301	201	1	0.00	Electronics	1000.0	101	12	0.1	
1	302	202	2	0.10	Electronics	500.0	102	10	0.1	
2	303	202	5	0.00	Electronics	500.0	103	10	0.1	
3	304	201	3	0.05	Electronics	1000.0	101	12	0.1	
4	305	202	1	0.00	Electronics	500.0	102	10	0.1	
5	306	203	4	0.20	Electronics	800.0	103	8	0.2	
6	307	201	2	0.10	Electronics	1000.0	109	12	0.1	
7	308	202	3	0.00	Electronics	500.0	104	10	0.1	
8	309	203	1	0.15	Electronics	800.0	105	8	0.2	
9	310	201	5	0.00	Electronics	1000.0	102	12	0.1	
10	311	201	3	0.05	Electronics	1000.0	101	12	0.1	
11	312	202	1	0.00	Electronics	500.0	102	10	0.1	
12	313	203	4	0.20	Electronics	800.0	103	8	0.2	

Link to google collab:

https://colab.research.google.com/drive/14S3_7frqluDS9AxRKBOOLGGBNxCcspq?usp=sharing