**Data Modeling Documentation:**

1. **Star Schema:**
   1. Facts Table:
      1. **Transaction**
         1. Primary Key: Transaction\_ID (each transaction is assigned to an ID)
         2. Foreign Keys: Supplier\_ID, Product\_ID, Store\_ID, Customer\_ID, Sales\_Staff\_ID, Promotion\_ID (each foreign key is linked to its dimension table)
         3. Relationship: many-to-1 Dimension tables
   2. Dimension Tables:
      1. **Supplier**
         1. Primary Key: Supplier\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Facts Transaction
         4. Normalization: avoid redundancy, when it’s the same supplier for many transactions and avoid long characters of the name to save storage space
      2. **Product**
         1. Primary Key: Product\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Facts Transaction,
         4. Normalization: avoid redundancy of a products name and save storage space
      3. **Store**
         1. Primary Key: Store\_ID
         2. Foreign Key: District\_ID
         3. Relationship: 1-to-many Facts Transaction, many-to-1 Dimension Table District
         4. Normalization: avoid redundancy of the store’s name such as Fünf Höfe or Riem Arcaden and save storage space
      4. **District**
         1. Primary Key: District\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Dim Store
         4. Normalization: avoid redundancy, thus many stores can be located in the same district such as Altstadt-Lehel or Au-Haidhausen, save storage space
      5. **Customer**
         1. Primary Key: Customer\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Facts Transaction
         4. Normalization: avoid redundancy, if one person places many orders and prevent misunderstandings, e.g. the names are the same but different persons
      6. **Sales Staff**
         1. Primary Key: Sales\_staff\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Facts Transaction
         4. Normalization: avoid redundancy, since one employee can be responsible for many transactions and save storage space
      7. **Promotion**
         1. Primary Key: Promotion\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Facts Transaction
         4. Normalization: avoid redundancy such as Promotion 2 and save storage space
   3. Performance implications
      1. **Disadvantages**
         1. Very high-level data redundancy
         2. Takes up a lot of storage space
         3. Maintenance is slower
      2. **Advantages** 
         1. Fast implementation of the Star schema
         2. Clear structure due to simple Architecture
         3. Easy to understand
         4. Query processing faster due to less joins
2. **Snowflake Schema:**
   1. Facts Table:
      1. **Transaction**
         1. Primary Key: Transaction\_ID (each transaction is assigned to an ID)
         2. Foreign Keys: Supplier\_ID, Product\_ID, Store\_ID, Customer\_ID, Sales\_Staff\_ID, Promotion\_ID (each foreign key is linked to its dimension table)
         3. Relationship: many-to-1 Dimension tables
   2. Dimension Tables:
      1. **Product**
         1. Primary Key: Product\_ID
         2. Foreign Key: Product\_category\_ID, Product\_subcategory\_ID, Product\_brand\_ID, Product\_department\_ID, Payment\_method\_ID
         3. Relationship: 1-to-many Facts Transaction, many-to-1 Dimension tables listed in foreign key
         4. Normalization: avoid redundancy of a products name and save storage space
         5. Historicization: recording when a change was made, in the case of a new product is added or the name is changed
      2. **Product category**
         1. Primary Key: Product\_category\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Dim Product
         4. Normalization: avoid redundancy such as Beauty and save storage space, thus long categorical names like Miscellaneous can take up a lot of storage space for more than 35.000 entries, it is easier and faster to make changes e.g. name, historization
         5. Historicization: recording when a change was made, e.g. a products category is changed
      3. **Product subcategory** 
         1. Primary Key: Product\_subcategory\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Dim Product
         4. Normalization: avoid redundancy such as Makeup and long categorical names such as Accessories to save storage space, a change of a product’s subcategory is possible in the future
         5. Historicization: recording when a change was made, e.g. a products subcategory is changed
      4. **Product brand**
         1. Primary Key: Product\_brand\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Dim Product
         4. Normalization: avoid redundancy such as Siemens and save storage space, less likely to have a name change for a brand, but it is still possible.
         5. Historicization: recording when a change was made, e.g. a products brand changes its name
      5. **Product Department** 
         1. Primary Key: Product\_department\_ID
         2. Foreign Key: not applicable
         3. Relationship: 1-to-many Dim Product
         4. Normalization: avoid redundancy such as Health & Beauty and save storage space, change from e.g. home & living to electronics possible in the future
         5. Historicization: recording when a change was made, e.g. a products department is changed
      6. **Store**
         1. Primary Key: Store\_ID
         2. Foreign Key: Store\_location\_ID, Store\_type\_ID, District\_ID
         3. Relationship: 1-to-many Facts Transaction, many-to-1 Dimension Tables District, Store type and Store location
         4. Normalization: avoid redundancy such as Fünf Höfe or Riem Arcaden and save storage space
         5. Historicization: recording when a change was made, e.g. the owner and the name of a store changes
      7. **Store location**
3. Primary Key: Store\_location\_ID
4. Foreign Key: not applicable
5. Relationship: 1-to-many Dim Store
6. Normalization: avoid redundancy such as Kaufingerstraße 9 or Sendlinger Straße 10 and save storage space
7. Historicization: recording when a change was made, such as when the store’s location/street name is changed
   * 1. **Store type**
        1. Primary Key: Store\_type\_ID
        2. Foreign Key: not applicable
        3. Relationship: 1-to-many Dim Store type
        4. Normalization: avoid redundancy such as City Center or Luxury Mall and save storage space
        5. Historicization: recording when a change was made, in the case of renaming a store type
     2. **District**
        1. Primary Key: District\_ID
        2. Foreign Key: not applicable
        3. Relationship: 1-to-many Dim Store
        4. Normalization: avoid redundancy, thus many stores can be located in the same district such as Altstadt-Lehel or Au-Haidhausen, save storage space
        5. Historicization: recording when a change was made, or a new district is added
     3. **Payment method**
        1. Primary Key: Payment\_method\_ID
        2. Foreign Key: not applicable
        3. Relationship: 1-to-many Facts Transaction
        4. Normalization: avoid redundancy such as Bargeld and save storage space
        5. Historicization: recording when a change was made, such as renaming a certain payment method
     4. **Promotion**
        1. Primary Key: Promotion\_ID
        2. Foreign Key: not applicable
        3. Relationship: 1-to-many Facts Transaction
        4. Normalization: avoid redundancy such as Promotion 2 and save storage space
        5. Historicization: recording when a change was made, such as renaming the promotions differently
     5. **Sales Staff**
        1. Primary Key: Sales\_staff\_ID
        2. Foreign Key: not applicable
        3. Relationship: 1-to-many Facts Transaction
        4. Normalization: avoid redundancy, since one employee can be responsible for many transactions and save storage space
        5. Historicization: recording when a change was made, such as staff’s last name or when there is a notice of termination
     6. **Customer**
        1. Primary Key: Customer\_ID
        2. Foreign Key: Customer\_loyalty\_status\_ID
        3. Relationship: 1-to-many Facts Transaction, many-to-1 Dim Customer loyalty status
        4. Normalization: avoid redundancy, if one person places many orders and prevent misunderstandings, if the names are the same but different persons
        5. Historicization: recording when a change was made, such as customer’s surname
     7. **Customer loyalty status**
        1. Primary Key: Customer\_loyalty\_status\_ID
        2. Foreign Key: not applicable
        3. Relationship: 1-to-many Dim Customer
        4. Normalization: avoid redundancy such as Silber and save storage space
        5. Historicization: recording when a change was made, such as a new loyalty status or a renaming
     8. **Supplier**
        1. Primary Key: Supplier\_ID
        2. Foreign Key: not applicable
        3. Relationship: 1-to-many Facts Transaction
        4. Normalization: avoid redundancy, when it’s the same supplier for many transactions and avoid long characters of the name to save storage space
        5. Historicization: recording when a change was made, such as a new loyalty status or a renaming
   1. Performance implications
      1. **Disadvantages**
         1. Time-consuming implementation of Snowflake architecture
         2. Complex architecture due to many normalizations
         3. Query processing slower due to many joins
      2. **Advantages**
         1. Normalized Data Structure 🡪 Data split into different dimension tables
         2. Very low-level data redundancy
         3. Requires less storage space
         4. Maintenance is faster due to normalization