DATA WAREHOUSE DIMENSIONS

Lecture 5

DIMENSION TABLES

Always has primary key (PK)

Poduct ID	Name	Category
P001	Sun Glases TR-7	Assecoirs
P002	Chocolate bar 70% cacao	Sweets
P003	Oat meal biscuits	Sweets

Produkt_PK	Name	Category
1	Sun Glases TR-7	Assecoirs
2	Chocolate bar 70% cacao	Sweets
3	Oat meal biscuits	Sweets

Use of Surrogate Keys

Produkt_PK	Product_ID
1	P001
2	P002
3	P003

- **Dimension Tables**
  - Function: Used to slice and dice data
- **Properties of a Dimension Table**
  - Always needs a primary key
    - Natural key from the source system is available
    - Natural key is not the best choice for a primary key
    - Recommendation: Use surrogate keys
      - Usually an incrementing integer
      - Surrogate keys have more benefits

DIMENSION TABELS

Always has primary key (PK)

Poduct ID	Name	Category
P001	Sun Glases TR-7	Assecoirs
P002	Chocolate bar 70% cacao	Sweets
P003	Oat meal biscuits	Sweets

Produkt_PK	Name	Category
1	Sun Glases TR-7	Assecoirs
2	Chocolate bar 70% cacao	Sweets
3	Oat meal biscuits	Sweets

Use of Surrogate Keys

Produkt_ PK	Product_ID
1	P001
2	P002
3	P003

Use of Surrogate Keys

- Question: Do we need to keep the natural keys?
  - Answer: We can, but it is often not necessary
- Recommendation: Use a lookup table
  - Reference: Links the created surrogate key with the natural key

Implementing a Lookup Table

- SQL query: Retrieve distinct values of the product ID (natural key)
- Add sequence: Populate a sequence next to these values
  - Easily done in SQL or other ETL tools

HOW TO CREATE CORRECT REFERENCES FROM FACT TABLE TO DIMENSION TABLE USING SURROGATE KEYS?

Always has primary key (PK)

Poduct ID	Name	Category
P001	Sun Glases TR-7	Assecoirs
P002	Chocolate bar 70% cacao	Sweets
P003	Oat meal biscuits	Sweets

Produkt_PK	Name	Category
1	Sun Glases TR-7	Assecoirs
2	Chocolate bar 70% cacao	Sweets
3	Oat meal biscuits	Sweets

Use of Surrogate Keys

Produkt_ PK	Product_ID
1	P001
2	P002
3	P003

- **Typical Example Scenario: Fact Table**
  - Origin: Fact table comes from the source system.
  - Issue: Fact table still contains the natural key.

  - **Solution: Using a Lookup Table or Joins**

  - **Lookup Table Approach**
    - Use the lookup table to map natural keys to surrogate keys.

  - **Join Approach**
    - Retain both surrogate and natural keys in the table.
    - Create a join between fact and dimension tables.

HOW TO CREATE CORRE
FROM FACT TABLE TO DI
USING SURROGA

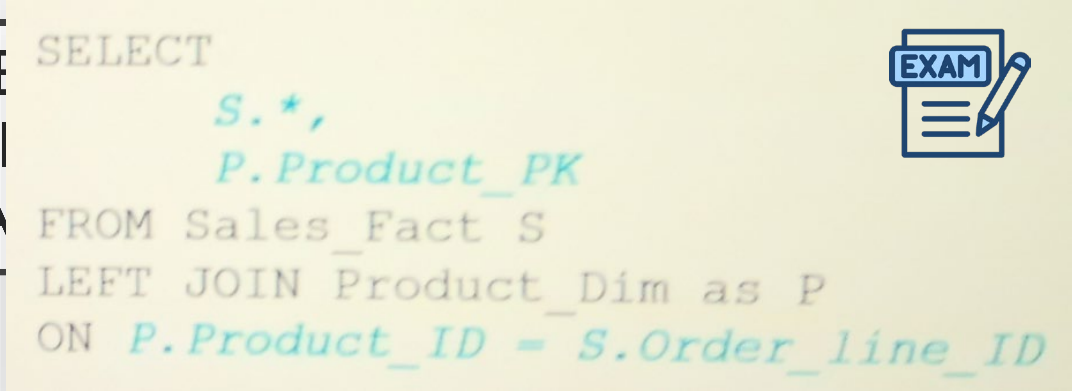
Always has primary key (PK)

Poduct ID	Name	Category
P00I	Sun Glases TR-7	Assecoirs
P002	Chocolate bar 70% cacao	Sweets
P003	Oat meal biscuits	Sweets

Produkt_PK	Name	Category
I	Sun Glases TR-7	Assecoirs
2	Chocolate bar 70% cacao	Sweets
3	Oat meal biscuits	Sweets

Use of Surrogate Keys

Produkt_PK	Product_ID
I	P00I
2	P002
3	P003



```
SELECT
        S.*,
        P.Product_PK
FROM Sales_Fact S
LEFT JOIN Product_Dim as P
ON P.Product_ID = S.Order_line_ID
```

- **Implementing a Join in SQL**
- Start with the original fact table.
  - Includes all attributes already created and transformed.
  - Replacing the natural key is usually the last step after transformations.
- **Example of SQL Join**
  - Use a left join from the sales fact table to the product dimension table.
  - Aliases for Tables: P for product dimension, S for sales fact.
  - Join Columns: Use product ID from the lookup or dimension table and order line ID from the sales table.
- **Process**
  - Add the value from the product table using the join.
  - Replace natural key values with surrogate key values.
  - Example: Replace P034 with 34 from the product or lookup table.

HOW TO CREATE CORRECT REFERENCES FROM FACT TABLE TO DIMENSION TABLE USING SURROGATE KEYS?

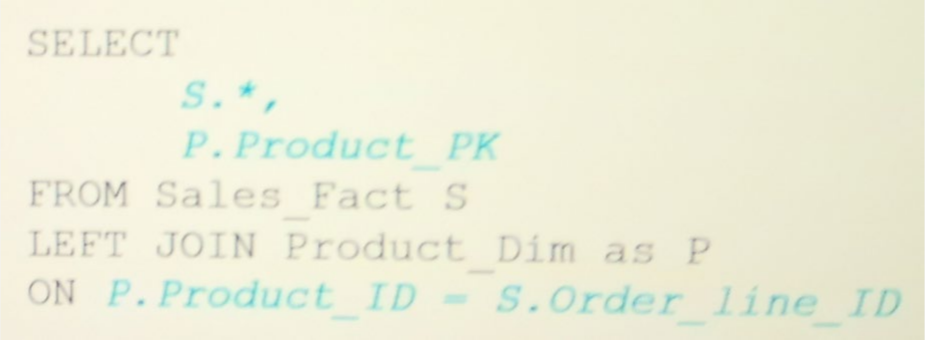
Always has primary key (PK)

Poduct ID	Name	Category
P001	Sun Glases TR-7	Assecoirs
P002	Chocolate bar 70% cacao	Sweets
P003	Oat meal biscuits	Sweets

Produkt_PK	Name	Category
1	Sun Glases TR-7	Assecoirs
2	Chocolate bar 70% cacao	Sweets
3	Oat meal biscuits	Sweets

Use of Surrogate Keys

Produkt_PK	Product_ID
1	P001
2	P002
3	P003



```
SELECT
      S.*,
      P.Product_PK
FROM Sales_Fact S
LEFT JOIN Product_Dim as P
ON P.Product_ID = S.Order_line_ID
```

- **Alternative Methods**
  - Other methods can be used, but joins are common.
  - Example: Removing prefix (e.g., P) and transforming the key to an integer.
  - Common method: Left join to ensure all matches.

- **Final Result**
  - Achieve correct references to the dimension table.
  - Ensure accurate and efficient data slicing and dicing.

HOW TO CREATE CORRECT REFERENCES FROM FACT TABLE TO DIMENSION TABLE USING SURROGATE KEYS?

Always has primary key (PK)

Produkt _PK	Name	Category
1	Sun Glases TR-7	Assecoirs
2	Chocolate bar 70% cacao	Sweets
3	Oat meal biscuits	Sweets

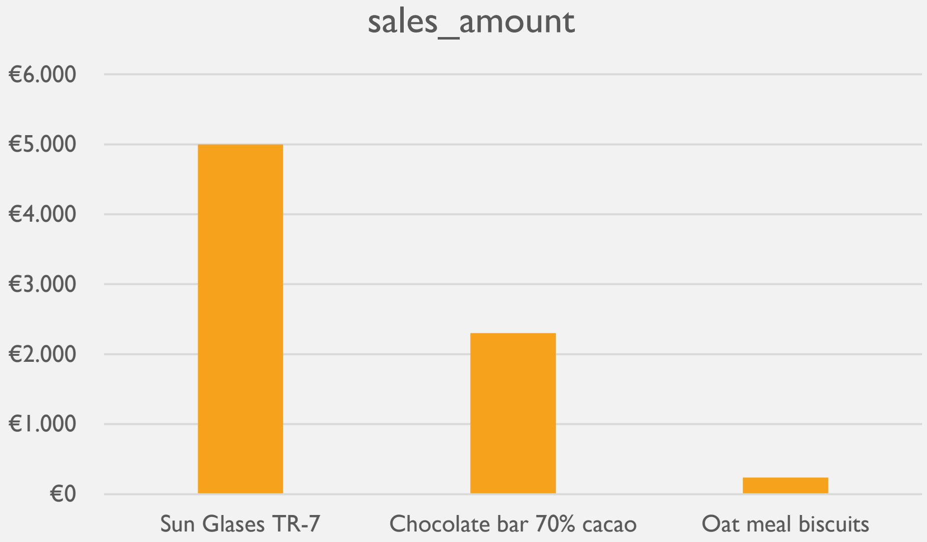
Cusomer_ ID	Customer_Na me	Order_ ID	Order_Line_ID	Order_Line_ FK	Quanti ty	UnitPri ce	Disc oujt _Pri ce
312	Johannes Strass	2345	Sun Glases TR-7	1	2	23.99	23.9 9
312	Johannes Strass	2314	Chocolate bar 70% cacao	156	3	8.99	8.99
312	Johannes Strass	2314	Oat meal biscuits	643	1	16.99	16.9 9

Relatively few rows / many columns With deseriptive attribute

- **Dimension Table Utilization**
  - Create a correct reference from the product primary key in the fact table.
  - Ensure it refers to the correct values.
- **Example Scenario: Sunglasses TR7**
  - Order line one references:
    - Correct product name (e.g., Sunglasses TR7)
    - Related attributes (e.g., category, subcategory)
- **Optimization**
  - Remove product name and attributes from the fact table.
  - Use the dimension table for all descriptive attributes.
- **Dimension Table Characteristics**
  - Typically has fewer rows.
  - Wide table with many columns for different descriptive attributes.

USE OF DIMENSION IN ANLYSIS

Order_Line_name	sales_amount
Sun Glases TR-7	5.000 €
Chocolate bar 70% cacao	2.300 €
Oat meal biscuits	234 €



- **Learning Outcome**
  - Dimension tables are used to group and filter data.
  - Known as "slicing and dicing" the data.
- **Functionality**
  - Use attributes of the dimension table for grouping data.
  - Example: Group data by product name.
- **Data Analysis**
  - Dimension tables serve as the entry point for data analysis.
  - Crucial for effective data analysis due to their structure and functionality.
- **Significance of Dimensions**
  - Essential for organizing and interpreting data.
  - Enhance the ability to analyze and draw insights from data.

DATE DIMENSION

- **Common Usage**
  - Most commonly used dimension in data processes.
  - Almost always available in data warehouses.

- **Importance in Analysis**
  - Crucial for measuring performance over time.
  - Key aspect of dimensional analysis.

Date Dimension

- One of the most common & most important dimensions
- Contains date related features
  - • Year, Month (name & number), Day, Quarter, Week,
  - Weekday (name & number)
- Meaningful surrogate key
  - For example 2022-04-02 20220402
- Extra row for no date/null (source) 1900-01-01 (dim)

DATE DIMENSION
ATTRIBUTES OF THE DATE DIMENSION

- **Basic Attributes**
  - Year
  - Month (both name and number)
  - Day in the month
  - Quarter
  - Week
  - Day of the week (both as text and number)
- **Examples**
  - Monday = 1, Tuesday = 2, etc.

Date Dimension

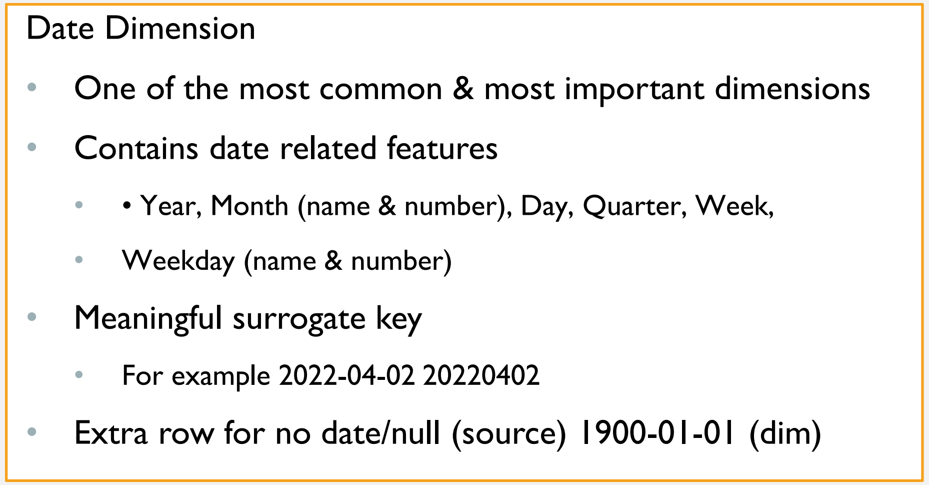
- One of the most common & most important dimensions
- Contains date related features
  - • Year, Month (name & number), Day, Quarter, Week,
  - Weekday (name & number)
- Meaningful surrogate key
  - For example 2022-04-02 20220402
- Extra row for no date/null (source) 1900-01-01 (dim)

DATE DIMENSION
Surrogate Key in the Date Dimension

- **Meaningful Surrogate Key**
  - Not a simple incrementing number.
  - Consists of year, month, and day (e.g., 20220402 for April 2, 2022).
  - Serves as the primary key in the date dimension.

- **Handling Missing Dates**

- **Dummy Value Row**
  - Extra row representing a dummy value for missing dates.
  - Ensures no null values in foreign keys within the fact table.
  - Common dummy date: January 1, 1900.



Date Dimension

- One of the most common & most important dimensions
- Contains date related features
  - • Year, Month (name & number), Day, Quarter, Week,
  - Weekday (name & number)
- Meaningful surrogate key
  - For example 2022-04-02 20220402
- Extra row for no date/null (source) 1900-01-01 (dim)

DATE DIMENSION
Ensuring Referential Integrity

- **Replacing Null Values**
  - Replace null or missing date values with the dummy foreign key.
  - Maintains referential integrity and relationships.

- **Conclusion**
  - Proper setup of the date dimension is crucial for accurate and efficient data analysis.

Date Dimension

- One of the most common & most important dimensions
- Contains date related features
  - • Year, Month (name & number), Day, Quarter, Week,
  - Weekday (name & number)
- Meaningful surrogate key
  - For example 2022-04-02 20220402
- Extra row for no date/null (source) 1900-01-01 (dim)

ADDING TIME GRANULARITY AND ENHANCING THE DATE DIMENSION

Separate Time Dimension

- **Importance of Time Aspect**
  - If timestamps are important, separate time dimension is needed.
  - Granularity of time should be considered separately from the date.
- **Time Dimension**
  - Create a separate dimension for time when necessary.

Date Dimension Characteristics

- **Predictability and Calculation**
  - Date dimension is predictable and can be populated in advance.
  - Can include future dates not yet in the fact table.
- **Future Dates**
  - Populate the date dimension for upcoming years.

Date Dimension

- Time is usually a separate dimension
- Can be populated in advance (e.g. for next 5 or 10 years)
  - Numbers & Text (e.g. January, 1)
  - Long & Abreviated (Jan, January— Mon, Monday)
  - Combinations of attributes (Q1, 2022-Q1)
  - Fiscal dates (Fiscal Year etc.)
  - Flags (Weekend, company holidays etc.)

ADDING TIME GRANULARITY AND ENHANCING THE DATE DIMENSION

- **Components of a Date Dimension**
  - **Attributes to Include**
    - Both numbers and text (e.g., January and 1 for the first month).
    - Long and abbreviated names (e.g., January and Jan).
  - **Business Use Case**
    - Include attributes based on business requirements.
    - Abbreviated names for reporting purposes.

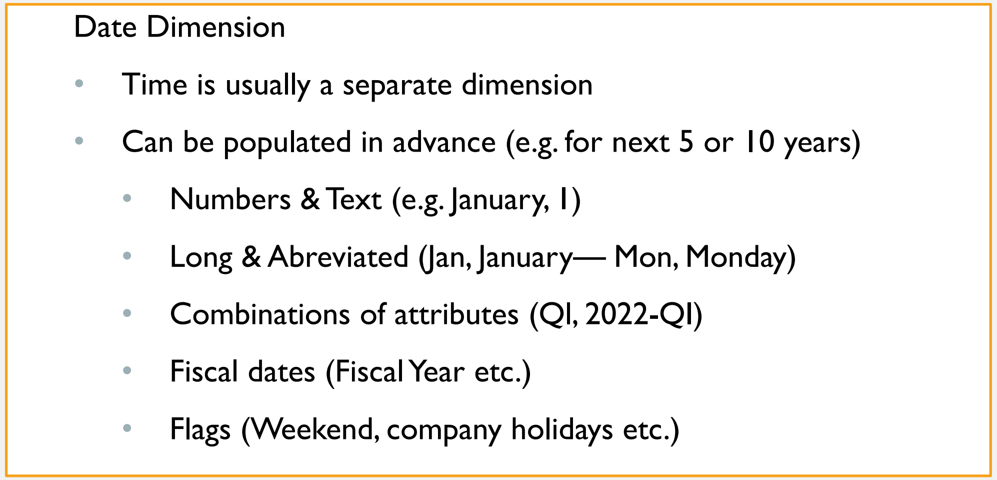
Date Dimension

- Time is usually a separate dimension
- Can be populated in advance (e.g. for next 5 or 10 years)
  - Numbers & Text (e.g. January, 1)
  - Long & Abreviated (Jan, January— Mon, Monday)
  - Combinations of attributes (Ql, 2022-Ql)
  - Fiscal dates (Fiscal Year etc.)
  - Flags (Weekend, company holidays etc.)

ADDING TIME GRANULARITY AND ENHANCING THE DATE DIMENSION
Combination Attribute

- **Examples**
  - Quarter and year combination (e.g., 2022 Q1).
  - Simplifies data grouping for end users.

- **Fiscal Dates**
  - Include fiscal year information.



Date Dimension

- Time is usually a separate dimension
- Can be populated in advance (e.g. for next 5 or 10 years)
  - Numbers & Text (e.g. January, 1)
  - Long & Abreviated (Jan, January— Mon, Monday)
  - Combinations of attributes (Q1, 2022-Q1)
  - Fiscal dates (Fiscal Year etc.)
  - Flags (Weekend, company holidays etc.)

ADDING TIME GRANULARITY AND ENHANCING THE DATE DIMENSION
Flags and Indicators

- **Usage**
  - Flags indicate if a condition is true or not.
  - Examples: Is it a weekend? Is it a holiday?

- **Analysis**
  - Use flags for detailed data analysis.

- **Conclusion**
  - A well-structured date dimension enhances data analysis capabilities.
  - Including comprehensive and relevant attributes facilitates easier and more effective reporting.

Date Dimension

- Time is usually a separate dimension
- Can be populated in advance (e.g. for next 5 or 10 years)
  - Numbers & Text (e.g. January, 1)
  - Long & Abreviated (Jan, January— Mon, Monday)
  - Combinations of attributes (Q1, 2022-Q1)
  - Fiscal dates (Fiscal Year etc.)
  - Flags (Weekend, company holidays etc.)

EXAMPLE OF A DATE TABLE

- **Primary Key**
  - The first and most important column in the date table.
- **Original Date**
  - The base date value, typically always available.
  - Stored as a date data type.
- **Note on Formatting**
  - Different formats (hyphens, backslashes) are handled in the BI application, not in the database.

Date Dimension

- Time is usually a separate dimension
- Can be populated in advance (e.g. for next 5 or 10 years)
  - Numbers & Text (e.g. January, 1)
  - Long & Abreviated (Jan, January— Mon, Monday)
  - Combinations of attributes (Q1, 2022-Q1)
  - Fiscal dates (Fiscal Year etc.)
  - Flags (Weekend, company holidays etc.)

Date_FK	Date	Month	Short_Month	Year-Quarter	Year	Weekend	Is_Week
20240101	2022-01-01	January	Jan	2024-Q1	2024	Saturday	1
20240102	2022-01-02	January	Jan	2024-Q1	2024	Sunday	1
20240103	2022-01-03	January	Jan	2024-Q1	2024	Monday	0

EXAMPLE OF A DATE TABLE

- **Combination and Abbreviated Values**
  - Include both long and abbreviated forms of dates.
  - Example: January (full name) and Jan (abbreviation).

- **Flags and Indicators**
- **Weekend Indicator**
  - Flag to indicate if the date is a weekend.
  - Saturday and Sunday are weekends (value = 1).
  - Monday through Friday are weekdays (value = 0).

- **Binary Flag Usage**
  - Advantages: Can be aggregated, summed, and used in calculations.
  - Disadvantages: Less user-friendly.
  - Alternative: Use understandable text (e.g., "weekend" or "weekday").

Date Dimension

- Time is usually a separate dimension
- Can be populated in advance (e.g. for next 5 or 10 years)
  - Numbers & Text (e.g. January, 1)
  - Long & Abreviated (Jan, January— Mon, Monday)
  - Combinations of attributes (Q1, 2022-Q1)
  - Fiscal dates (Fiscal Year etc.)
  - Flags (Weekend, company holidays etc.)

Date_FK	Date	Month	Short_Month	Year-Quarter	Year	Weekend	Is_Week
20240101	2022-01-01	January	Jan	2024-Q1	2024	Saturday	1
20240102	2022-01-02	January	Jan	2024-Q1	2024	Sunday	1
20240103	2022-01-03	January	Jan	2024-Q1	2024	Monday	0

EXAMPLE OF A DATE TABLE

- **User Considerations**
  - Choose flag representations based on business user needs and reporting requirements.

- **Conclusion**
  - The date dimension is crucial and must be designed with attention to detail, considering both functionality and user-friendliness.

Date Dimension

- Time is usually a separate dimension
- Can be populated in advance (e.g. for next 5 or 10 years)
  - Numbers & Text (e.g. January, 1)
  - Long & Abreviated (Jan, January— Mon, Monday)
  - Combinations of attributes (QI, 2022-QI)
  - Fiscal dates (Fiscal Year etc.)
  - Flags (Weekend, company holidays etc.)

Date_FK	Date	Month	Short_Month	Year-Quarter	Year	Weekend	Is_Week
20240101	2022-01-01	January	Jan	2024-Q1	2024	Saturday	1
20240102	2022-01-02	January	Jan	2024-Q1	2024	Sunday	1
20240103	2022-01-03	January	Jan	2024-Q1	2024	Monday	0

NULL IN DIMENSTIONS
RECAP ABOUT NULLS

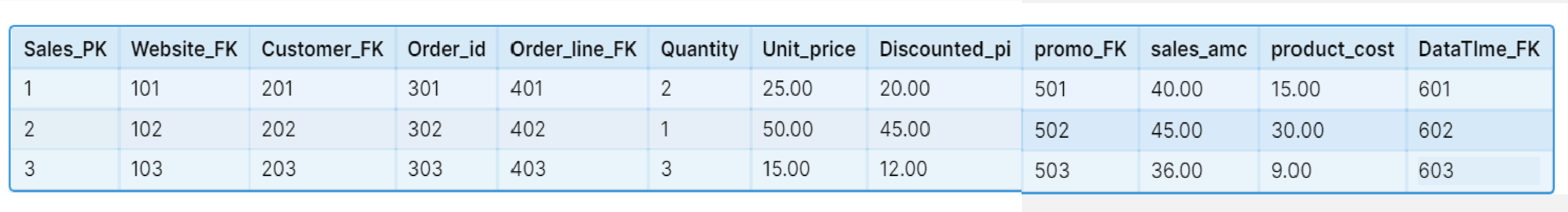
- **Nulls in Foreign Keys**
  - Must be avoided to maintain referential integrity.
  - Replace nulls with a dummy value (e.g., -1).

- **Consequences of Nulls**
  - Nulls in foreign keys break joins and relationships.
  - Results in loss of values during joins with dimension tables.
  - Example: Grouping by promotion type would miss values with nulls.

- **Using Dummy Values**
  - Replace nulls with a dummy value (e.g., -1) to preserve data in joins.
  - Include a corresponding row in the dimension table (e.g., "No Promo" for promotion type).

Promo _PK	Social media prom
Newsletter	1
Website ads	2
No promo	3
Promo name	-1

Date_PK	Date
20220101	1/1/2022
20220102	1/2/2022
20220103	1/3/2022
19000101	1/1/1900



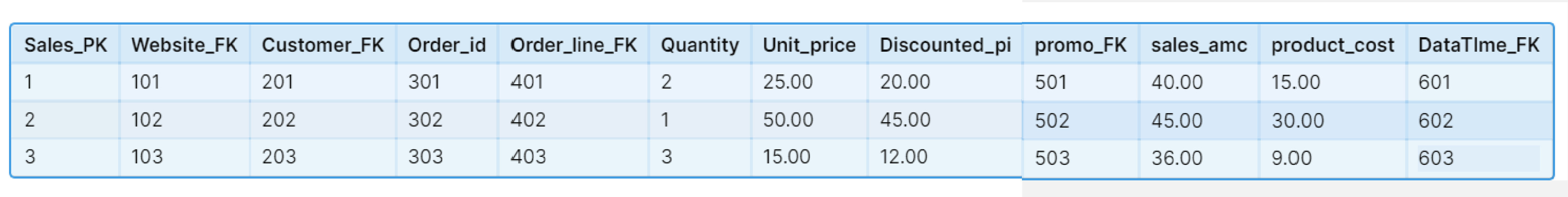
Sales_PK	Website_FK	Customer_FK	Order_id	Order_line_FK	Quantity	Unit_price	Discounted_pi	promo_FK	sales_amc	product_cost	DataTIme_FK
1	101	201	301	401	2	25.00	20.00	501	40.00	15.00	601
2	102	202	302	402	1	50.00	45.00	502	45.00	30.00	602
3	103	203	303	403	3	15.00	12.00	503	36.00	9.00	603

NULLS IN FACT TABLES

Promo_PKz	Social media prom
Newsletter	1
Website ads	2
No promo	3
Promo name	-1

Date_PK	Date
20220101	1/1/2022
20220102	1/2/2022
20220103	1/3/2022
19000101	1/1/1900

- **Nulls in Fact Tables**
  - **Appropriate Use**
    - Nulls can be present in fact tables for valid scenarios.
    - Example: No sales on weekends if stores are closed.
    - Using zero instead of null could distort averages and sums.
  - **Aggregation Compatibility**
    - Nulls work well with aggregations like sum and average.
    - Do not necessarily need to replace nulls in fact tables.



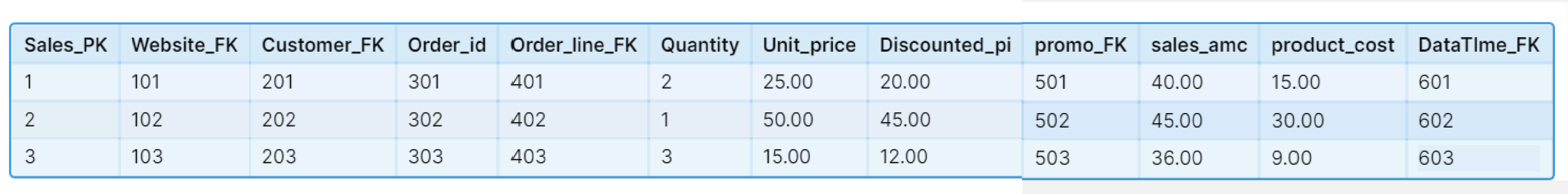
Sales_PK	Website_FK	Customer_FK	Order_id	Order_line_FK	Quantity	Unit_price	Discounted_pi	promo_FK	sales_amc	product_cost	DataTIme_FK
1	101	201	301	401	2	25.00	20.00	501	40.00	15.00	601
2	102	202	302	402	1	50.00	45.00	502	45.00	30.00	602
3	103	203	303	403	3	15.00	12.00	503	36.00	9.00	603

NULLS IN FACT TABLES

Promo_PKz	Social media prom
Newsletter	1
Website ads	2
No promo	3
Promo name	-1

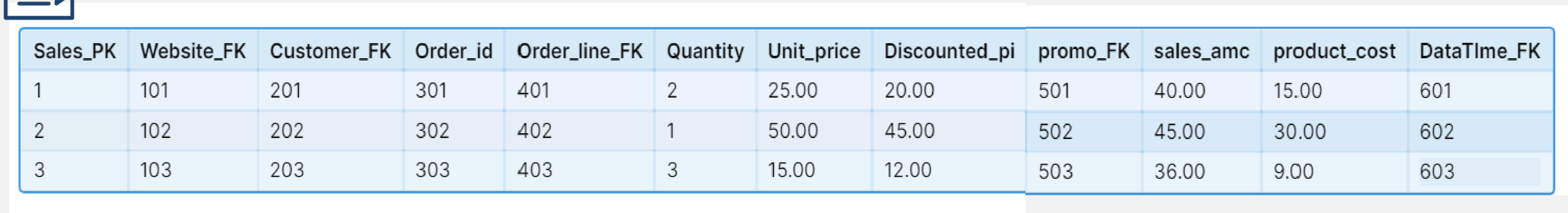
Date_PK	Date
20220101	1/1/2022
20220102	1/2/2022
20220103	1/3/2022
19000101	1/1/1900

- **Key Takeaways**
  - **Foreign Keys**
    - Avoid nulls in foreign keys; use dummy values.
    - Ensure dummy values are properly represented in dimension tables.
  - **Fact Tables**
    - Nulls can be acceptable and useful for certain aggregations.
    - Replace nulls with meaningful values only when necessary to maintain data accuracy.



Sales_PK	Website_FK	Customer_FK	Order_id	Order_line_FK	Quantity	Unit_price	Discounted_pi	promo_FK	sales_amc	product_cost	DataTIme_FK
1	101	201	301	401	2	25.00	20.00	501	40.00	15.00	601
2	102	202	302	402	1	50.00	45.00	502	45.00	30.00	602
3	103	203	303	403	3	15.00	12.00	503	36.00	9.00	603

NULLS IN DIMENSION TABLES



Sales_PK	Website_FK	Customer_FK	Order_id	Order_line_FK	Quantity	Unit_price	Discounted_pi	promo_FK	sales_amc	product_cost	DataTIme_FK
1	101	201	301	401	2	25.00	20.00	501	40.00	15.00	601
2	102	202	302	402	1	50.00	45.00	502	45.00	30.00	602
3	103	203	303	403	3	15.00	12.00	503	36.00	9.00	603

Nulls must be avoided in FKs

- **Maintaining Referential Integrity**
  - Dimension tables should include a row for the dummy value.
  - Example for Date Dimension: Use 1st January 1900 as a dummy date.

Promo_PK	Social media prom
Newsletter	1
Website ads	2
No promo	3
Promo name	-1

Date_PK	Date
20220101	1/1/2022
20220102	1/2/2022
20220103	1/3/2022
19000101	1/1/1900

NULLS IN DIMENSION

- **Replacing Nulls**
  - Replace nulls in dimensions with descriptive values.
  - Example: "No promotion available", "No category available".
  - Use dummy values like 1st January 1900 for dates.

Promo_PK	Social media prom
Newsletter	1
Website ads	2
No promo	3
Promo name	-1

Date_PK	Date
20220101	1/1/2022
20220102	1/2/2022
20220103	1/3/2022
19000101	1/1/1900

- **Reasoning Behind Replacement**
  - Improve clarity for business users.
  - Nulls lack descriptive context, potentially confusing.
  - Descriptive values help users understand the absence of data.

- Replace nulls With descriptive values
  - More understable for business users
  - Values appear in aggregations in BI tools

IMPLEMENTATION STRATEGY

- **User Flexibility and Understanding**
  - Allow users to decide inclusion in aggregations and visualizations.
  - Null values default to exclusion in BI tool graphs.

- **Benefits of Descriptive Values**
  - Provide more options for end users.
  - Enhance comprehensibility of data in dimensions.

Promo_PK	Social media prom
Newsletter	1
Website ads	2
No promo	3
Promo name	-1

Date_PK	Date
20220101	1/1/2022
20220102	1/2/2022
20220103	1/3/2022
19000101	1/1/1900

- **Replace nulls With descriptive values**
  - More understable for business users
  - Values appear in aggregations in BI tools

IMPLEMENTATION STRATEGY

- **Dimensional Null Attributes**
  - Replace nulls with dummy or descriptive values.
  - Maintain consistency and clarity across reporting and analysis.

- **Integration with Fact Tables**
  - If nulls originate from fact tables, ensure consistency in handling across dimensions.
  - Utilize consistent approaches for maintaining data integrity.

Promo_PK	Social media prom
Newsletter	1
Website ads	2
No promo	3
Promo name	-1

Date_PK	Date
20220101	1/1/2022
20220102	1/2/2022
20220103	1/3/2022
19000101	1/1/1900

- Replace nulls With descriptive values
  - More understable for business users
  - Values appear in aggregations in BI tools

HIERACHIE IN DIMENSIONS
INTRODUCTION TO DIMENSION HIERARCHIES

- **Purpose and Importance**
  - Dimensions often include hierarchies that organize data logically.
  - Understanding how to manage hierarchies effectively is crucial.
- **Source Data Normalization**
- **Transactional Data Nature**
  - Source data is typically normalized for transactional processing.
  - Example: Product and category stored in separate tables for efficiency.

Often normalized

Product name	Category_id
Milk	1
Printer	2
Red Towel	3
Green Towel	3
Blue Towel	3

Category_id	Category
1	Groceries
2	Electronics
3	Houshold

Customer_id	Customer name	Order id	Order_line_r	Product_FK	Category_FK
312	Franklin Miller	2314	Sunglases S	34	2

Snowflaked schema (should be avoided)

HIERARCHIE IN DIMENSIONS
CHALLENGES IN ANALYTICAL PROCESSING

- **Requirements for Data Warehouse**
  - Analytical processing requires high performance and usability.
  - Denormalization of data enhances read performance and usability.

Often normalized

Product name	Category_id
Milk	1
Printer	2
Red Towel	3
Green Towel	3
Blue Towel	3

Category_id	Category
1	Groceries
2	Electronics
3	Houshold

Customer_id	Customer name	Order id	Order_line_r	Product_FK	Category_FK
312	Franklin Miller	2314	Sunglases S	34	2

Snowflaked schema (should be avoided)

HIERARCHIE IN DIMENSIONS
PITFALLS OF SNOWFLAKING

- **Snowflaked Schema**
  - Over-normalization leads to a snowflaked schema.
  - Example: Product name linked to category via separate tables, creating multiple foreign keys.
  - Result: Complicated schema structure resembling a snowflake.
- **Issues with Snowflaking**
  - **Performance Impact**
    - Increases complexity and widens fact tables unnecessarily.
    - Decreases query performance due to multiple joins.
  - **Usability Concerns**
    - Reduced usability with complex schema
    - Difficulties in data retrieval and interpretation.

Often normalized

Product name	Category_id
Milk	1
Printer	2
Red Towel	3
Green Towel	3
Blue Towel	3

Category_id	Category
1	Groceries
2	Electronics
3	Houshold

Customer_id	Customer name	Order id	Order_line_r	Product_FK	Category_FK
312	Franklin Miller	2314	Sunglases S	34	2

Snowflaked schema (should be avoided)

HIERARCHIE IN DIMENSIONS
BEST PRACTICES FOR HIERARCHIES

- **Avoid Snowflaking**
  - Minimize unnecessary normalization in dimensions.
  - Keep hierarchies straightforward and intuitive.
- **Optimize for Performance and Usability**
  - Denormalize dimensions when beneficial for analytical queries.
  - Maintain a balance between normalization and denormalization based on performance needs.

Often normalized

Product name	Category_id
Milk	1
Printer	2
Red Towel	3
Green Towel	3
Blue Towel	3

Category_id	Category
1	Groceries
2	Electronics
3	Houshold

Customer_id	Customer name	Order id	Order_line_r	Product_FK	Category_FK
312	Franklin Miller	2314	Sunglases S	34	2

Snowflaked schema (should be avoided)

HIERARCHIE IN DIMENSIONS
CONCLUSION

Often normalized

Product name	Category_id
Milk	1
Printer	2
Red Towel	3
Green Towel	3
Blue Towel	3

Category_id	Category
1	Groceries
2	Electronics
3	Houshold

Conclusion

- **Balancing Act**
    - Strive for a well-structured schema that balances normalization with usability.
    - Enhance data warehouse performance and usability by optimizing dimension hierarchies effectively.

Customer_id	Customer name	Order id	Order_line_r	Product_FK	Category_FK
312	Franklin Miller	2314	Sunglases S	34	2

Snowflaked schema (should be avoided)

HIERARCHIE IN DIMENSIONS

DEALING WITH DATA NORMALIZATION IN DIMENSIONS

Normative Practices in Data Modeling

- **Normalization Habits**
  - Experienced data modelers often prioritize normalization.
  - Normalize data even if it's denormalized in the source for certain scenarios.

Challenges in Data Warehousing

- **Data Warehouse Requirements**
  - Unlike operational databases, data warehouses prioritize usability and performance.
  - Normalization can hinder these goals in analytical environments.
- **Resistance to Normalization in Data Warehouses**
- **Usability Concerns**
  - Normalization can complicate schema and reduce usability.
  - Avoid normalization to enhance data warehouse usability.

Often normalized

Product name	Category_id
Milk	1
Printer	2
Red Towel	3
Green Towel	3
Blue Towel	3

Category_id	Category
1	Groceries
2	Electronics
3	Houshold

- Some professional ave the habit to normalize data
- Bad for usability & performance!
- We should not do that!

HIERARCHIE IN DIMENSIONS
DEALING WITH DATA NORMALIZATION IN DIMENSIONS

- **Embracing Denormalization**
- **Strategy for Dimensions**
  - Flatten or denormalize data in dimensions for better performance.
  - Example: Create a single table with joined attributes like Category_id and category name.
- **Benefits of Denormalization**
- **Enhanced Performance**
  - Simplifies querying and improves query performance.
  - Reduces complexity in schema design.
- **Conclusion**
- **Adopting Best Practices**
  - Prioritize denormalization over normalization in data warehouse dimensions.
  - Optimize schema for usability and query performance to support analytical needs effectively.

Flattened dimension

Product name	Category_id
Milk	1
Printer	2
Red Towel	3
Green Towel	3
Blue Towel	3

Category_id	Category
1	Groceries
2	Electronics
3	Houshold

Product_ID	Product name	Category
1	Milk	Groceries
2	Printer	Electronics
3	Red Towel	Houshold
4	Green Towel	Houshold
5	Blue Towel	Houshold

Flattened dimension

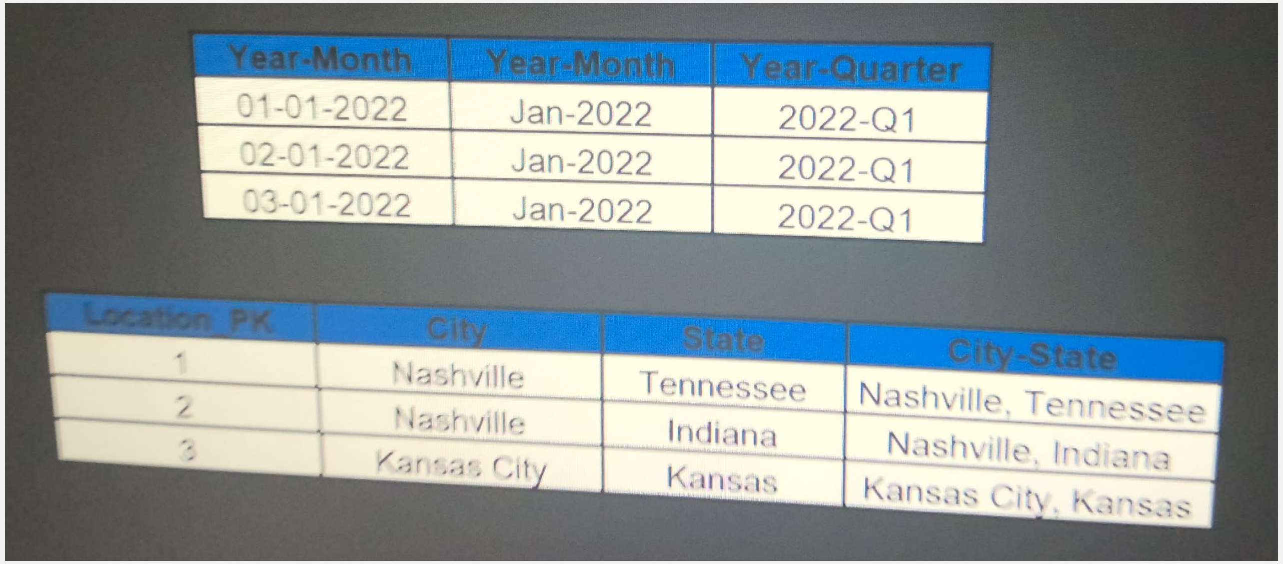
MANAGING HIERARCHIES IN DIMENSIONS

Combining Attributes Across Hierarchies

- **User-Friendly Approach**
  - Combine attributes from different hierarchy levels into a single column.
  - Example: Year and quarter combined for direct usability.

Handling Duplicate Values

- **Example: City Names**
  - Cities like Nashville may exist in multiple states.
  - Create pre-calculated columns (e.g., City_State) for clarity and usability.



Year-Month	Year-Month	Year-Quarter
01-01-2022	Jan-2022	2022-Q1
02-01-2022	Jan-2022	2022-Q1
03-01-2022	Jan-2022	2022-Q1

Location_PK	City	State	City-State
1	Nashville	Tennessee	Nashville, Tennessee
2	Nashville	Indiana	Nashville, Indiana
3	Kansas City	Kansas	Kansas City, Kansas

MANAGING HIERARCHIES IN DIMENSIONS CONCLUSION

- **Enhancing User Friendliness**
- **Simplifying Complex Hierarchies**
  - Simplify user experience by pre-combining commonly used attribute combinations.
  - Improve usability and accessibility of data.
- **Practical Considerations**
- **Reducing Dimension Complexity**
  - Collapse hierarchies into fewer tables for easier management.
  - Optimize dimension structure for enhanced query performance and usability.
- **Optimizing Dimension Design**
  - Focus on combining hierarchies effectively to streamline data access.
  - Prioritize user friendliness and performance in dimension management strategies.