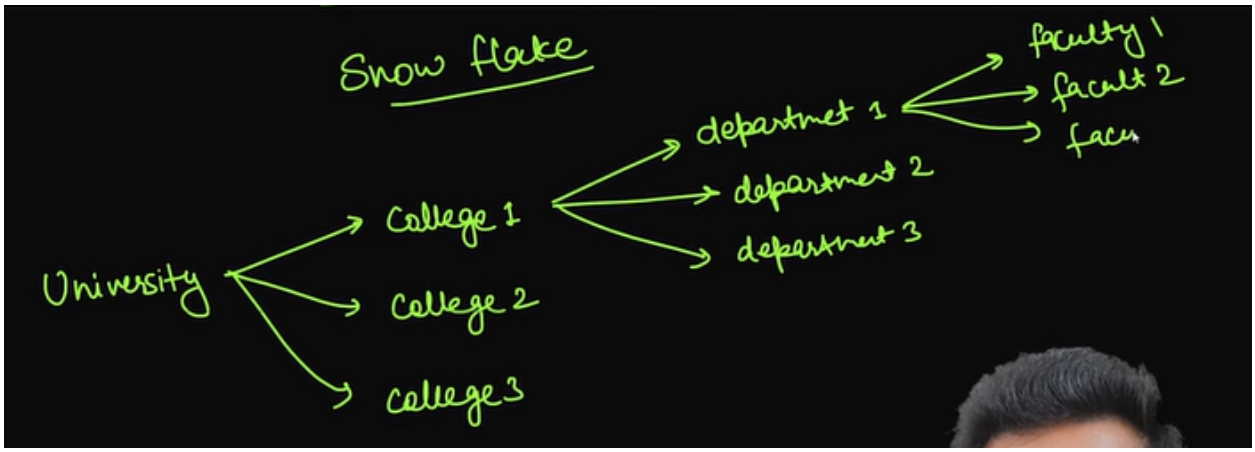
Product Dimension & Snowflake Schema (cont)

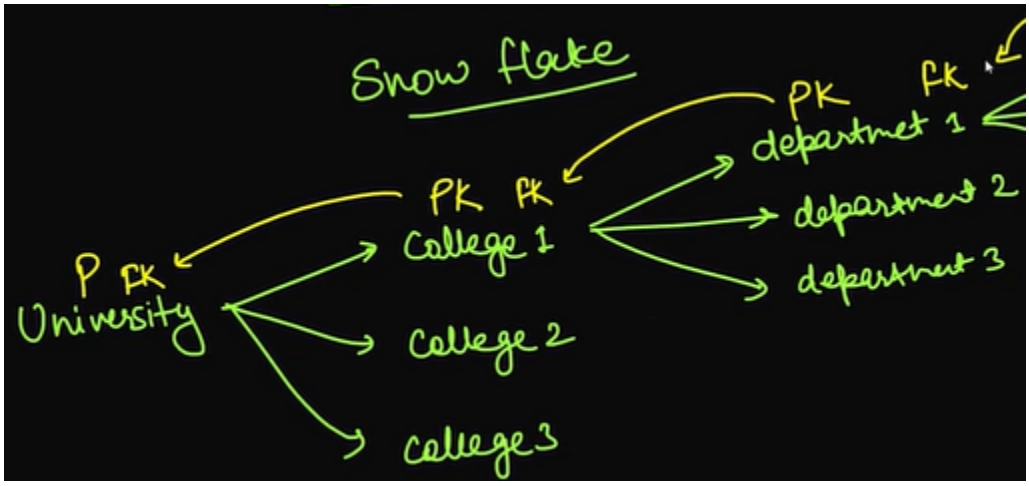
1. Another example of sf schema
2. Consider an example of university

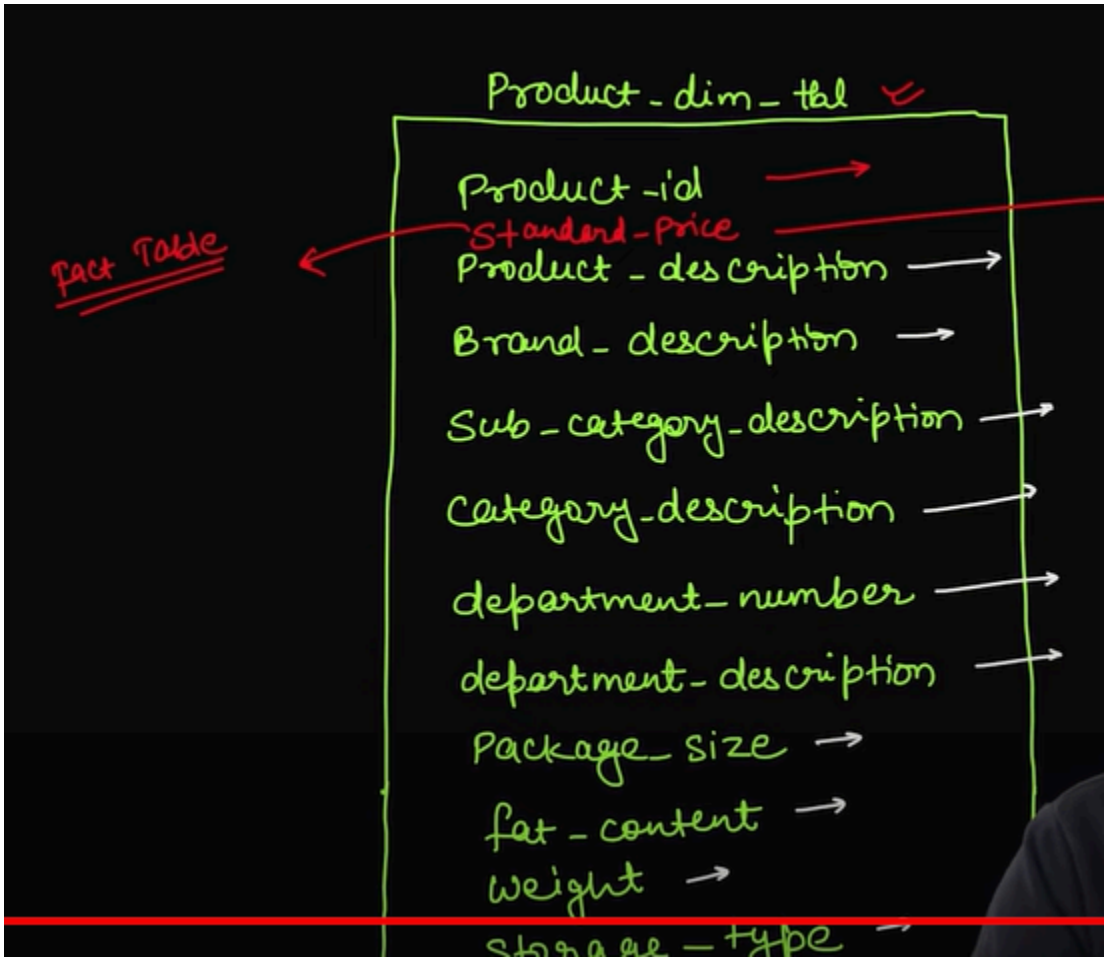


3. Here each uni has diff colleges and each college has many departments and each dept has
4. SO here this is one to many relation
5. This example in the table format

6. If we see here…lot of data in the columns such as college,university are redundant

7. So in snowflake schema we use foreign keys and primary keys

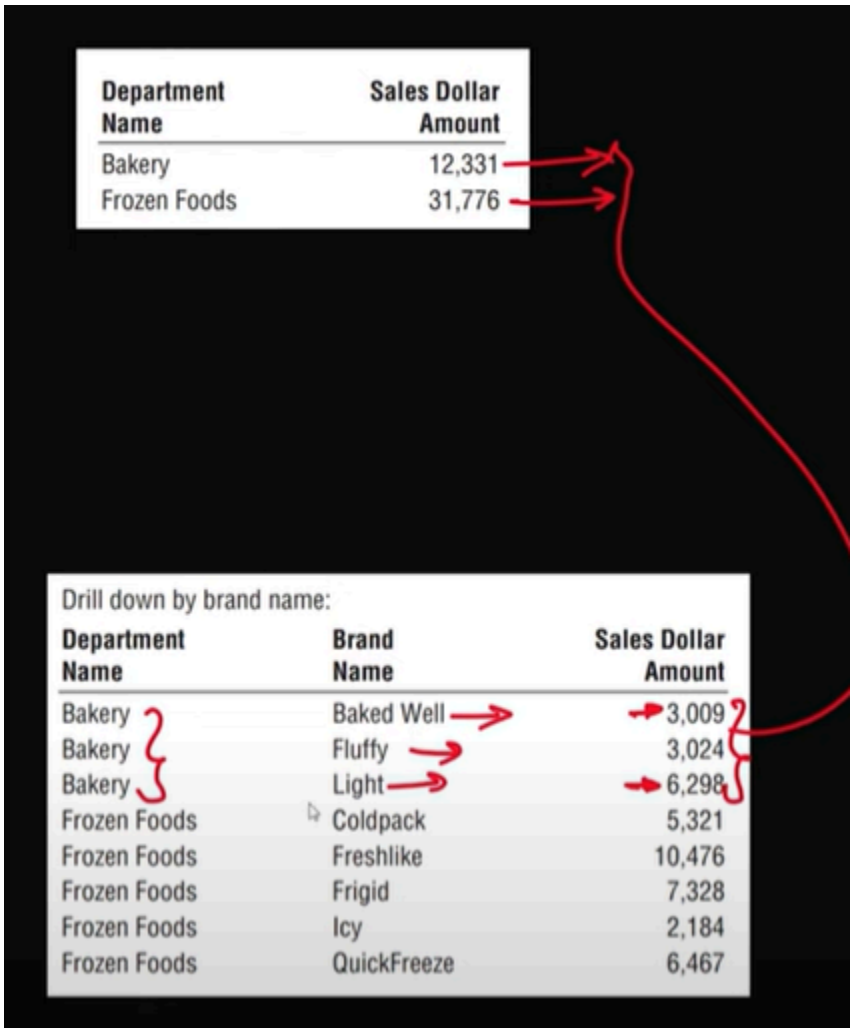


8. Measurement in dimensional table
9. 

10. SO here we have standard price col…which does not change over a period of time…
11. If the measurement col does not changes over time..then we can include that in dim table
12. And this col can be in both fact and dimension table

13. Lets understand this terms now

14. Lets take an example of this table



Department Name	Sales Dollar Amount
Bakery	12,331
Frozen Foods	31,776

Drill down by brand name:

Department Name	Brand Name	Sales Dollar Amount
Bakery	Baked Well	3,009
Bakery	Fluffy	3,024
Bakery	Light	6,298
Frozen Foods	Coldpack	5,321
Frozen Foods	Freshlike	10,476
Frozen Foods	Frigid	7,328
Frozen Foods	Icy	2,184
Frozen Foods	QuickFreeze	6,467

15. If we drill down(roll down) by brand name we get above result (just adding a grain level in table)

16. Or if we drill down or roll down by fat content

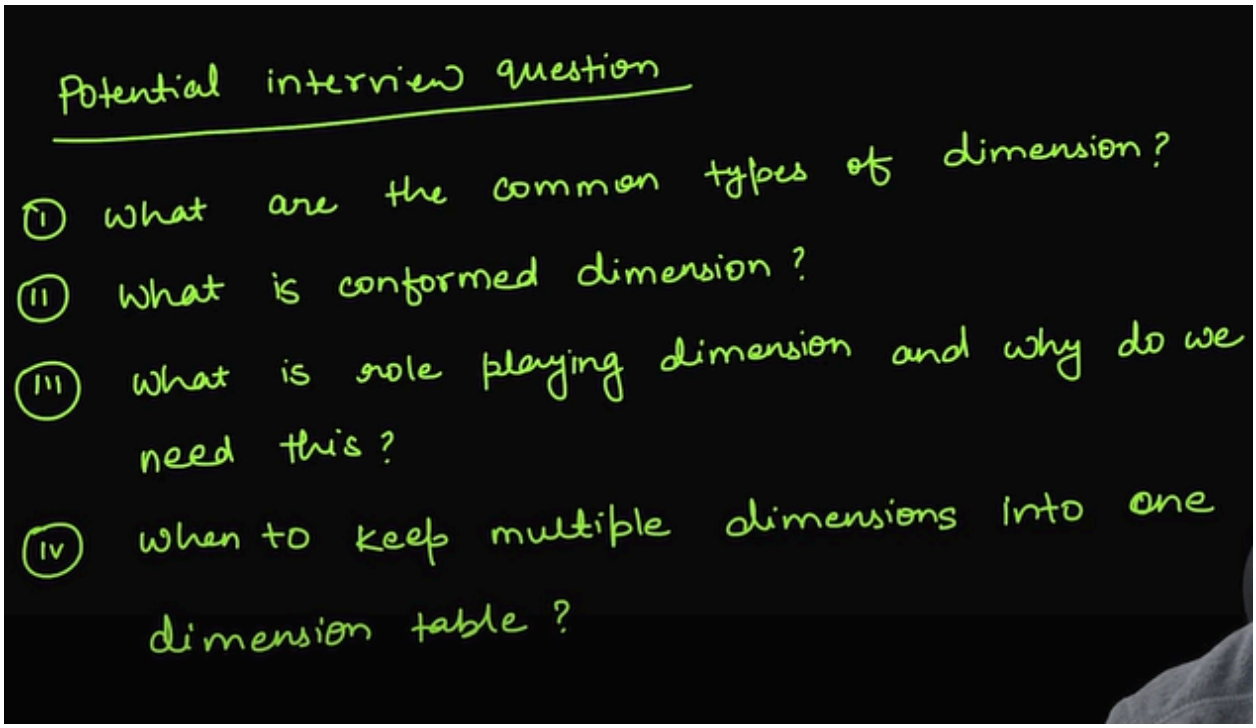
Or drill down by fat content:

Department Name	Fat Content	Sales Dollar Amount
Bakery	Nonfat	6,298
Bakery	Reduced fat	5,027
Bakery	Regular fat	1,006
Frozen Foods	Nonfat	5,321
Frozen Foods	Reduced fat	10,476
Frozen Foods	Regular fat	15,979

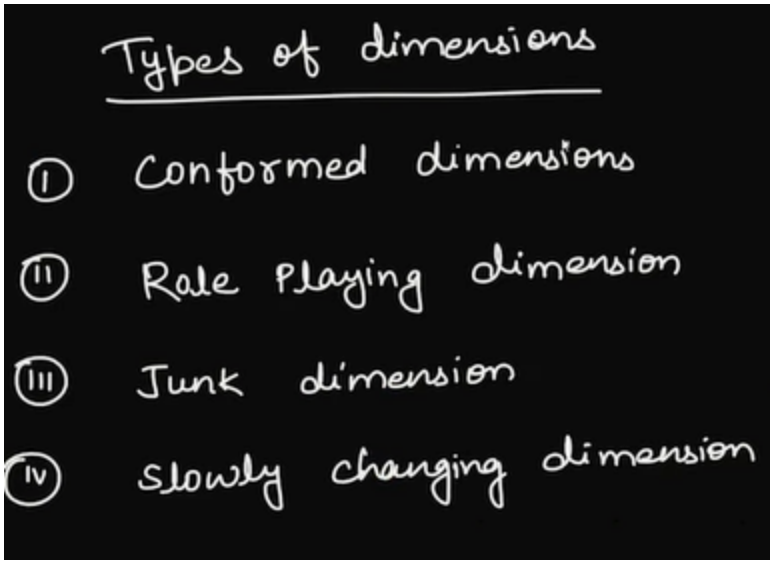
17. If we do the opp then it is roll up or drill up

Types of dimensions in DW

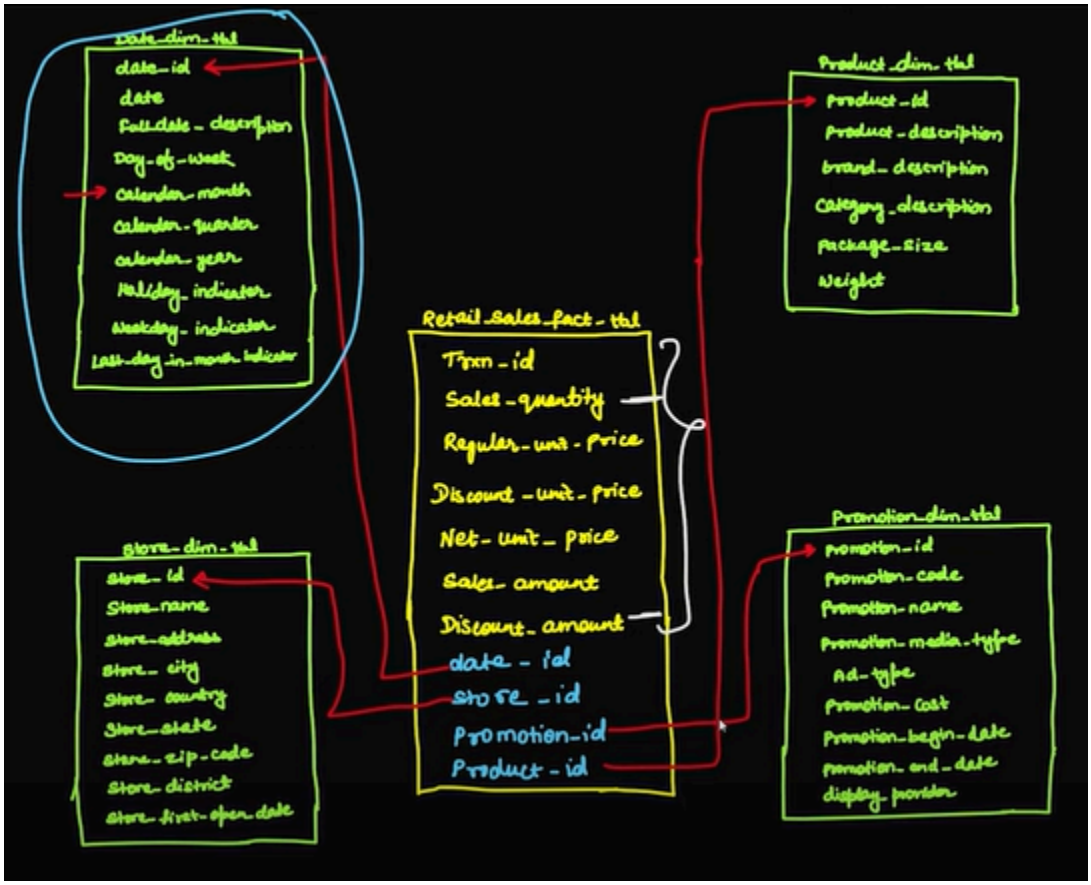
1. Potential Interview questions include



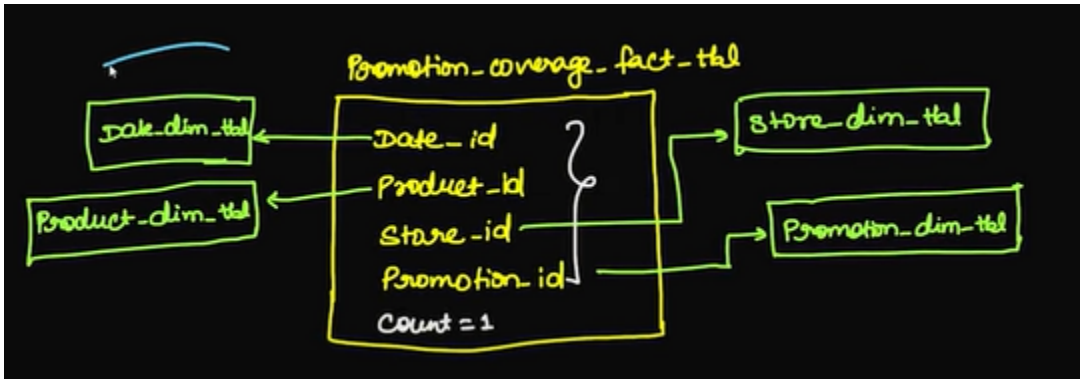
2. Dimensions and dim table are diff things

3. 

these are the most common dimensions
4. Conformed dimensions
5. Here one dim table will be used in multiple fact tables…then it is conformed dimensions
6. Example here we have data dim table



7. Here we have another fact table..with the same date dimtable



8. **Example:**

   Consider an e-commerce company with separate data marts for sales, marketing, and customer service. Traditionally, each data mart might have its own "Customer" dimension with slight variations in attributes or data formats.

   Conformed dimensions would involve creating a single "Customer" dimension table with standardized attributes like "Customer ID," "Name," "Email," "Address," etc. This central table would then be linked to the sales, marketing, and customer service fact tables. This approach allows the company to:

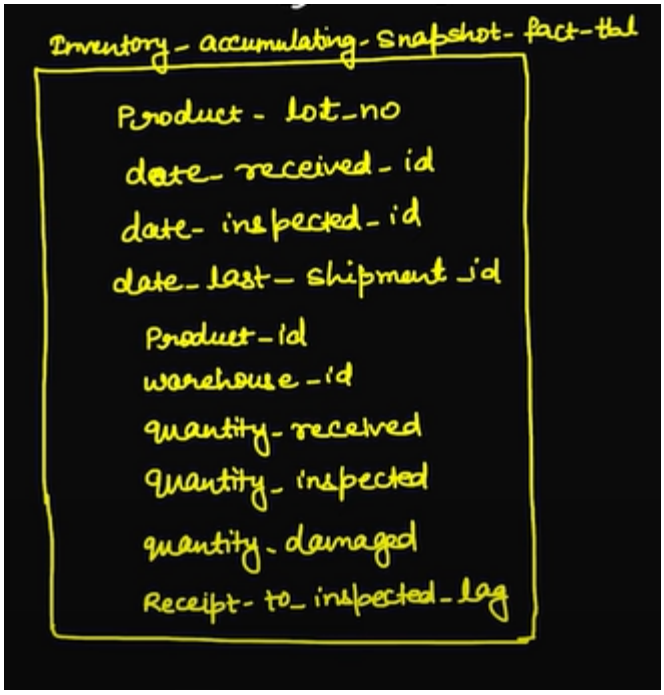
   - Analyze customer purchase behavior across different channels (e.g., website vs. email campaigns).
   - Identify customer segments for targeted marketing efforts based on purchase history.
   - Track customer interactions across sales and support touchpoints.

9. Advantages

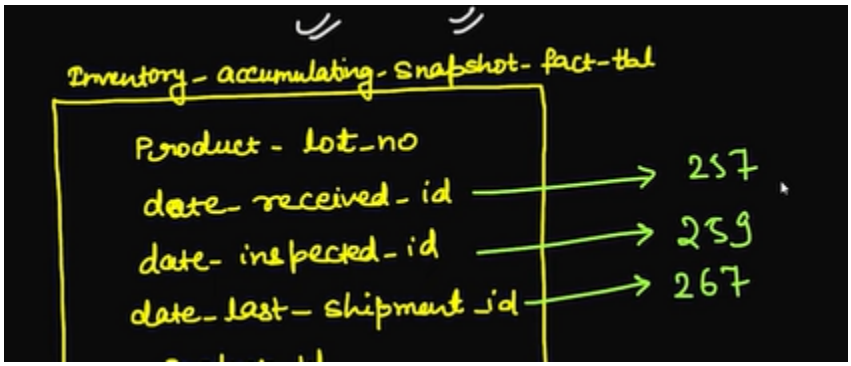
   Reduced Redundancy and Development Costs:

   - Conformed dimensions prevent the need to recreate the same dimension table structure for each fact table. This saves on storage space and simplifies development efforts. Instead of building separate "Customer" dimensions for sales and marketing data, you maintain a single, central "Customer" dimension that serves both purposes.

10. Role Playing Dimension

11. Lets have this example 

12. Here we have 3 diff date_id



13. So to avoid confusion while joining
14. We'll create date dim for order date and request date…which has similar schema to date dim table
15. Role playing dim explained : https://g.co/gemini/share/248c6bdcfe8a
16. Junk DImension
17. Lets consider this transaction fact_table..we'll add two more col

18. A junk dimension, unlike what the name might suggest, serves a valuable purpose in data warehousing. Here's how it works:

    Concept:

    A junk dimension is a single table that combines multiple low-cardinality attributes (attributes with few distinct values) from various sources. These attributes are typically flags, indicators, or codes that wouldn't justify creating individual dimension tables due to their limited number of values.

    Benefits:

    - **Reduced Complexity:** By grouping low-cardinality attributes into a single table, you simplify the dimensional model and avoid cluttering it with many small dimension tables. This makes the model easier to understand and manage.
    - **Improved Query Performance:** Junk dimensions can improve query performance by reducing the number of joins needed in complex queries. Instead of joining multiple small dimensions, you join a single, compact junk dimension.

19. example

    Imagine an e-commerce website tracks online orders. You have a fact table capturing data like "Order ID," "Product ID," "Customer ID," "Order Date," etc.

    There might also be separate flags for:

    - **Payment Method (Credit Card, Debit Card, etc.)**
    - **Order Status (New, Processing, Shipped, etc.)**
    - **Promotional Code Used (Yes/No)**

    These attributes have a limited number of possible values. Creating individual dimension tables for each wouldn't be efficient.

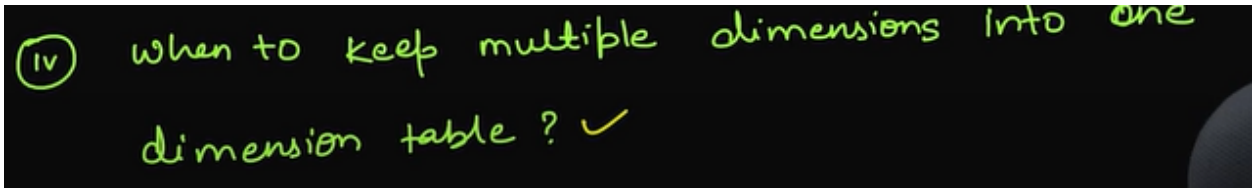
    Instead, you can create a junk dimension named "OrderDetails" with columns for:

    - **Payment Type (Credit Card, Debit Card, etc.)**
    - **Order Status (New, Processing, Shipped, etc.)**
    - **Promo Code Used (Yes/No)**

    The fact table would then join with the "OrderDetails" junk dimension to access these attributes. This reduces complexity and potentially improves query speed.

20. **Important Considerations:**

    - **Junk dimensions are most suitable for truly low-cardinality attributes.** If an attribute starts to have a significant number of values, it might be better suited for a separate dimension table.
    - **Proper naming conventions are crucial** within the junk dimension to avoid confusion about the meaning of each attribute.

21. 

    we use junk dimensions here