

This is the JD for that job Analyze business requirements to understand data needs and relationships Design conceptual data models to represent high-level business entities and relationships Develop logical data models with detailed attributes, keys, and relationships Create physical data models optimized for specific database or data lake platforms Define and document data definitions, standards, and naming conventions Collaborate with data architects, engineers, and analysts to align models with technical and business needs Normalize or denormalize data structures based on performance and use-case requirements Map source systems to target models for ETL/ELT development Maintain data dictionaries and metadata repositories Ensure data models support data quality, integrity, and consistency Review and validate models with stakeholders and subject matter experts Update models based on evolving business requirements or system changes Support data lineage and impact analysis efforts Participate in data governance and stewardship initiatives Use modeling tools like ER/Studio, ERwin, or dbt for documentation and visualization Provide guidance on data modeling best practices to engineering teams Collaborate on schema design for data lake zones (raw, curated, trusted) Ensure models support scalability, performance, and compliance requirements Assist in reverse engineering models from existing databases or data lakes Contribute to training and onboarding materials related to data models

Data Modeller Interview Questions & Suggested Answers

1. Can you explain the difference between Conceptual, Logical, and Physical Data Models?

Sample Answer:

"Sure. A **Conceptual Data Model** provides a high-level view of business entities and their relationships, without technical details. It's mainly for business stakeholders.

The **Logical Data Model** adds details like attributes, primary and foreign keys, and relationships, but it's still independent of any database technology.

The **Physical Data Model** translates the logical model into an actual implementation, adding data types, indexes, partitions, and other technical details optimized for a specific database or data lake."

2. How do you decide when to normalize or denormalize a data model?

Sample Answer:

"I generally normalize data to eliminate redundancy, ensure data integrity, and simplify maintenance.

However, for performance-critical scenarios—especially in reporting, analytics, or data lakes—denormalization is applied to reduce complex joins and improve query performance. I decide based on the use-case, query patterns, and input from engineering teams."

3. How do you collaborate with data architects, engineers, and analysts?

Sample Answer:

"I work closely with data architects to align models with enterprise architecture and standards. With data engineers, I share source-to-target mappings and provide guidance on schema

design for databases or data lakes.

I also consult analysts to ensure models meet their reporting and analytical needs, adjusting structures if necessary."

4. What tools have you used for data modelling and documentation?

Sample Answer:

"I have experience with tools like **ERwin Data Modeler** for conceptual, logical, and physical modelling.

I've also used **dbt** for transforming data in the data warehouse and for maintaining documentation.

For visualizations or quick drafts, I use tools like Lucidchart.

I'm flexible and can quickly adapt to **ER/Studio** or other platforms your organization uses."

5. How do you ensure your data models support data quality, integrity, and consistency?

Sample Answer:

"I ensure integrity by defining primary and foreign keys, enforcing constraints, and aligning models with business rules.

I collaborate with data governance teams to maintain metadata repositories and data dictionaries.

In addition, I work with engineers to validate data pipelines against the model, and I support automated testing wherever possible."

6. How do you approach source-to-target mapping for ETL/ELT development?

Sample Answer:

"I start by thoroughly understanding the source systems and business requirements.

I map source fields to target entities and attributes in the logical or physical model.

This mapping serves as the foundation for ETL/ELT development and helps ensure data quality and completeness.

I document these mappings clearly for engineers and update them as systems evolve."

7. Have you worked on reverse engineering data models from existing databases or data lakes?

Sample Answer:

"Yes. Using tools like **ERwin** or **ER/Studio**, I've reverse-engineered databases to extract existing schema structures.

This helps in documentation, understanding legacy systems, and planning for improvements.

I also cross-reference these models with business requirements to identify gaps or inefficiencies."

8. What is your experience with schema design in a data lake environment?

Sample Answer:

"I've collaborated with data engineers to design schemas for different data lake zones—

Raw, Curated, and Trusted.

In raw zones, we store data as-is for traceability.

Curated zones involve applying business rules and transformations.

In trusted zones, data is clean, validated, and optimized for consumption by analysts or ML models.

I ensure data models align with these zones and support scalability and performance."

9. How do you support data lineage and impact analysis efforts?**Sample Answer:**

"I maintain comprehensive metadata and participate in documenting data lineage, showing how data flows from source to report.

For impact analysis, I use modelling tools to trace dependencies, so any schema change is carefully assessed for downstream effects.

This helps reduce risk and maintain data trustworthiness."

10. How do you stay updated on data modelling best practices?**Sample Answer:**

"I stay updated by following data management forums, reading whitepapers, and engaging in communities related to tools like ERwin and dbt.

I also actively participate in internal knowledge-sharing sessions and collaborate with architects to adopt emerging best practices."

✓ Bonus Technical Terms to Casually Mention in the Interview

- ✓ Data Integrity
 - ✓ Source-to-Target Mapping
 - ✓ Data Lake Zones (Raw, Curated, Trusted)
 - ✓ Data Governance & Stewardship
 - ✓ Normalization / Denormalization
 - ✓ Scalability & Performance Optimization
 - ✓ Data Lineage & Impact Analysis
 - ✓ Metadata Repositories
 - ✓ ETL/ELT Pipelines
-

One of the Interview question data modeller concept for swiggy/zomato

✓ How to Answer: Data Modelling Concept for Swiggy/Zomato**Step 1: Understand the Core Entities**

At a high level, key business entities could be:

- **User / Customer**
 - **Restaurant**
 - **Menu / Dish**
 - **Order**
 - **Delivery Partner**
 - **Payment**
 - **Rating & Review**
 - **Location**
-

Step 2: Conceptual Data Model (High-Level Explanation)

You can say:

"*At a conceptual level, the core entities are **Customer**, **Restaurant**, **Order**, **Delivery Partner**, and **Menu Item**. The relationships would be:*

- *A Customer places one or more Orders.*
 - *An Order contains one or more Menu Items.*
 - *A Restaurant provides Menu Items.*
 - *A Delivery Partner delivers the Order.*
 - *Customers and Delivery Partners are both associated with Location information.*"
-

Step 3: Logical Data Model (Key Tables & Relationships)

Table	Primary Fields	Relationships
Customer	CustomerID, Name, Email, Phone	-
Restaurant	RestaurantID, Name, LocationID	-
MenuItem	ItemID, Name, Price, RestaurantID	FK to Restaurant
Order	OrderID, CustomerID, RestaurantID, TotalAmount, Status	FK to Customer, Restaurant

Table	Primary Fields	Relationships
OrderItem	OrderItemID, OrderID, ItemID, Quantity, Price	FK to Order, MenuItem
DeliveryPartner	PartnerID, Name, Phone, VehicleDetails, LocationID	-
DeliveryAssignment	AssignmentID, OrderID, PartnerID, Status, AssignedTime	FK to Order, DeliveryPartner
Payment	PaymentID, OrderID, PaymentMethod, PaymentStatus	FK to Order
RatingReview	ReviewID, OrderID, CustomerID, Rating, Comment	FK to Order, Customer
Location	LocationID, Address, Latitude, Longitude	Shared by Restaurant, Customer, Delivery Partner

Step 4: Additional Considerations (Bonus Points in Interview)

- ✓ You can mention **Normalization** is applied to avoid redundancy.
- ✓ For performance, frequently accessed data like **Restaurant ratings** or **Menu popularity** may be **denormalized** into summary tables.
- ✓ You can support **data lineage** for tracing how Orders flow through the system.
- ✓ Data model should support **scalability** for peak hours and high-volume days like festivals.
- ✓ **Data lake zones** could store raw order events for analytics.

CT Data Modelerresume_OCR.pdf

Tailored Self-Introduction & Roles/Responsibilities for Interview

1. Quick Self-Introduction

"Hi, I'm Balakrishnan Muralidharan. I have over 14 years of experience in the IT industry, with the last 5+ years focused on Snowflake data warehousing and data modelling. I specialize in designing end-to-end data models—conceptual, logical, and physical—that support scalable, secure, and high-performance data solutions aligned with business requirements."

2. Specific Roles & Responsibilities Based on Your Resume

"In my most recent role at Intraedge Technologies, I was responsible for designing conceptual and logical data models for key entities such as users, call sessions, devices, and service plans in a cloud communication environment. I developed physical data models optimized for Snowflake to handle high-volume call detail records (CDRs) and real-time analytics.

I collaborated with product and engineering teams to define schema standards and ensured data structures were normalized and denormalized based on performance and analytical flexibility requirements. I worked extensively on source-to-target mapping from AWS S3 logs and legacy databases, supporting ELT development using DBT and Snowpipe. I also documented data definitions, relationships, and lineage through internal metadata repositories and participated in reverse engineering legacy schemas to modernize the data architecture."

"Prior to that, at Chettinad Morimura Semiconductor Materials, I designed and implemented Snowflake-based data models for procurement, inventory, supplier, and logistics operations. I integrated data from ERP and warehouse platforms and ensured consistency through proper data standards and naming conventions. I also performed reverse engineering, metadata management, and supported impact analysis for migration and optimization efforts."

3. Highlight Your Technical Strengths

"Across these roles, I've built strong expertise in Snowflake, SQL, ELT concepts, data modelling, and metadata management. I've also worked extensively with tools like SQL Developer, DBT, Microsoft Visio, and Miro Board for data design, documentation, and collaboration. I'm confident my background in both data architecture and hands-on modelling makes me well-suited for this role."

Optional Add-On (if you want to sound people-oriented)

"Apart from my technical experience, I've always enjoyed collaborating with cross-functional teams to ensure that data models not only meet technical standards but also directly support business goals and reporting needs."

based on my resume, could you please suggest me faq's on data modelling interview

Data Modelling Interview FAQs & Suggested Answers (Based on Your Resume)

1. Can you walk us through your data modelling experience with Snowflake?

Answer:

"In my recent role at Intraedge Technologies, I designed end-to-end data models for a cloud communication platform. This included conceptual, logical, and physical models to represent

entities like users, call sessions, devices, and service plans.

The physical models were optimized specifically for Snowflake to efficiently handle high-volume call detail records (CDRs) and support real-time analytics.

I also contributed to schema design for raw, curated, and trusted zones within Snowflake, ensuring the models were scalable and aligned with both technical and business requirements."

2. How did you handle source-to-target mapping for ELT pipelines in your recent projects?

Answer:

"I worked extensively on source-to-target mapping for ELT pipelines by integrating data from AWS S3 logs and legacy databases into Snowflake.

We used DBT and Snowpipe to automate and manage the ELT processes. I ensured clear documentation of source-to-target mappings to support accurate data integration and transformation, which directly supported reporting and analytics requirements."

3. What's your approach to normalization and denormalization, and when do you apply each?

Answer:

"My approach is based on business use cases and performance requirements. For transactional data and maintaining data integrity, I apply normalization to eliminate redundancy.

However, for analytical use cases, especially in Snowflake where performance is key, I denormalize certain structures—such as creating aggregated views—to optimize query performance and simplify reporting."

4. How do you document data models and ensure data lineage is maintained?

Answer:

"I document data models using DBT and internal metadata repositories. DBT helps maintain both data transformation logic and model documentation in one place.

We also maintain data dictionaries and metadata repositories to ensure consistent data definitions and lineage tracking across the data ecosystem."

5. Can you explain how you reverse engineered legacy databases for data modelling?

Answer:

"In both Intraedge and my previous role at Chettinad Morimura, I reverse engineered existing databases to derive logical models.

This involved analyzing legacy schemas, understanding entity relationships, and translating them into modern, optimized Snowflake models.

I also performed impact analysis to ensure smooth migration and that the new models supported existing reporting requirements."

6. How did you collaborate with cross-functional teams to align data models with business needs?**Answer:**

"I regularly collaborated with product teams, engineers, and business stakeholders to understand operational workflows and KPIs.

This ensured that my data models were not only technically sound but also met the reporting and analytical needs of the business.

I also participated in defining schema standards and aligning models with data governance requirements."

7. How do you ensure your data models meet compliance, security, and performance standards?**Answer:**

"In my recent projects, I ensured compliance by incorporating role-based access controls, data masking, and audit logging in Snowflake.

Performance was addressed through optimized physical modelling, appropriate partitioning, and denormalized views where necessary to support analytical queries efficiently."

8. What challenges have you faced in designing data models for real-time analytics?**Answer:**

"One challenge was efficiently handling high-volume CDRs for real-time analytics in Snowflake. We addressed this by designing performance-optimized physical models and leveraging Snowpipe for near real-time data ingestion.

Additionally, we applied denormalized views for faster querying while maintaining normalized structures for core transactional data."

9. How do you design data models to support data lake architecture, specifically in Snowflake?**Answer:**

"I've designed schemas for raw, curated, and trusted zones in Snowflake's data lake architecture.

The raw zone captures ingested data in its native format, curated applies necessary transformations, and the trusted zone provides clean, validated data for business consumption. This layered approach ensures scalability, flexibility, and alignment with analytical requirements."

10. Can you explain how you used DBT for data modelling and documentation?**Answer:**

"DBT played a crucial role in both data transformations and maintaining documentation.

I used DBT to build and manage transformation workflows, define data models, and maintain

up-to-date documentation of the data pipeline.
This ensured both transparency and efficiency in our Snowflake environment."

Bonus Snowflake-Specific Questions & Answers

What are your best practices for Snowflake schema design?

Answer:

"I follow best practices like choosing appropriate clustering keys, minimizing data redundancy, and designing schemas that align with Snowflake's architecture.
I also separate raw, curated, and trusted data zones to ensure data quality and scalability."

How do you optimize Snowflake for performance?

Answer:

"I leverage clustering, proper partitioning strategies, denormalized views for reporting, and Snowflake's automatic scaling features.
I also monitor query performance and adjust models or pipelines as needed to maintain efficiency."

How do you use Snowpipe in your data ingestion process?

Answer:

"Snowpipe enables continuous data ingestion from sources like AWS S3 into Snowflake.
In my projects, I configured Snowpipe to automate the loading of raw data, supporting near real-time ingestion, which is essential for our analytics use cases."

Explanation on Kimbal dimentional model Fact table Dimensional tables

Kimball Dimensional Model — Simple Explanation

The **Kimball Dimensional Model**, also known as the **Star Schema approach**, is a methodology for designing data warehouses optimized for easy reporting and analytical querying.

It structures the data into:

1. Fact Tables

2. Dimension Tables

1. Fact Table

The **Fact Table** contains **measurable, quantitative data** about business processes or events.

Characteristics of a Fact Table:

- ✓ Contains **metrics**, **measures**, or **facts** like sales amount, quantity, revenue, etc.
- ✓ Typically holds **foreign keys** referencing related **dimension tables**.
- ✓ Can be **large in volume**, containing millions or billions of records.
- ✓ Often includes **aggregations** like sum, average, count, etc.

Example: Sales Fact Table

Sales_ID	Date_Key	Product_Key	Customer_Key	Store_Key	Quantity	Total_Amount
1001	20240701	101	5001	301	2	500

2. Dimension Tables

Dimension Tables contain descriptive information, also known as **attributes**, that provide context to the facts.

Characteristics of Dimension Tables:

- ✓ Store **textual** or **categorical** data (e.g., Product Name, Customer Name, Store Location).
- ✓ Help answer **who**, **what**, **when**, **where**, **how** questions.
- ✓ Typically smaller in size compared to fact tables.
- ✓ Used for **filtering**, **grouping**, and **drill-down** in reports.

Example: Product Dimension Table

Product_Key	Product_Name	Category	Brand
101	iPhone 15	Mobile	Apple

Example: Date Dimension Table

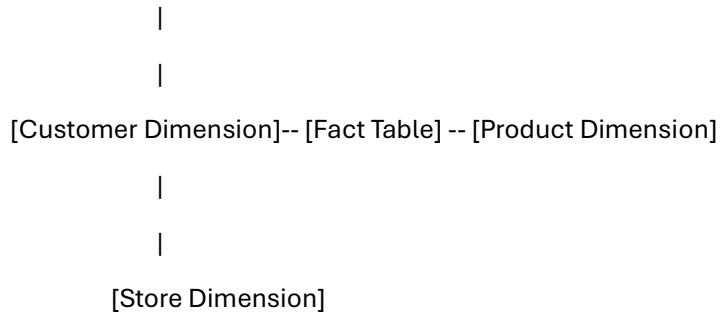
Date_Key	Full_Date	Day_Name	Month	Year
20240701	2024-07-01	Monday	July	2024

Kimball Star Schema Visual (Simple)

less

CopyEdit

[Date Dimension]



- **Fact Table** sits at the center.
 - Surrounded by connected **Dimension Tables**.
 - This forms the **Star Schema**, ideal for analytical queries.
-

Bonus: Types of Facts

- ✓ **Transactional Facts** — record individual business events (e.g., each sale).
 - ✓ **Snapshot Facts** — capture data at a specific point in time (e.g., daily stock levels).
 - ✓ **Accumulating Snapshot Facts** — track progress through a process (e.g., order fulfillment stages).
-

Summary

The **Kimball Dimensional Model** simplifies reporting and querying by:

- ✓ Structuring data into Fact and Dimension tables.
 - ✓ Enabling fast, intuitive BI and reporting.
 - ✓ Supporting performance tuning for analytical workloads.
-

Periodic Snapshot fact table
accumulating snapshot fact table
Factless fact table
Star schema
and Snowflake schema

1. Periodic Snapshot Fact Table

A **Periodic Snapshot Fact Table** captures **aggregated measures** at regular, consistent intervals, such as daily, weekly, or monthly.

When to Use:

- When you need to track performance over time (e.g., daily sales, monthly account balances).

Example: Daily Sales Snapshot

Snapshot_Date	Store_Key	Product_Key	Total_Sales	Total_Quantity
2024-07-04	301	101	50,000	120

- This table shows the total sales for each product at each store for that day.
 - Data is captured daily, regardless of whether sales happened every second.
-

2. Accumulating Snapshot Fact Table

An **Accumulating Snapshot Fact Table** tracks the progress of a process, with key dates or milestones updated over time.

When to Use:

- For processes with clearly defined start and end points (e.g., Order Fulfillment, Loan Application).

Example: Order Fulfillment Table

Order_ID	Order_Placed_Date	Packed_Date	Shipped_Date	Delivered_Date	Status
1001	2024-07-01	2024-07-02	2024-07-03	2024-07-04	Delivered

- Each row represents one order.
 - Dates get updated as the order progresses.
 - Ideal for SLA tracking and process performance.
-

3. Factless Fact Table

A **Factless Fact Table** records the **occurrence of an event** or the **absence of a measure**, but contains only keys — no numeric facts.

When to Use:

- Many-to-many relationships where the mere occurrence is important (e.g., student attendance, promotional eligibility).

Example: Student Attendance

Date_Key	Student_Key	Class_Key
2024-07-04	501	101

- This table records which students attended which class on which day.
- No numeric measures, just the event of attendance.

4. Star Schema vs. Snowflake Schema

Feature	Star Schema	Snowflake Schema
Structure	Fact table in center with de-normalized dimension tables	Dimensions are normalized, split into sub-tables
Query Performance	Faster due to fewer joins	Slower due to multiple joins
Storage Space	Requires more space (due to de-normalization)	More storage-efficient (normalized)
Design Complexity	Simple, easy to understand	More complex, normalized structure
Use Case	Ideal for BI tools, reporting, quick querying	Suitable for large, complex dimensions with hierarchical data

Star Schema Example:

less

CopyEdit

[Date Dimension]

|

[Customer Dim] — [Sales Fact Table] — [Product Dim]

|

[Store Dimension]

Snowflake Schema Example:

vbnnet

CopyEdit

[Date Dim]

|

[Customer Sub-Dim]—[Customer Dim]—[Sales Fact Table]—[Product Dim]—[Product Sub-Dim]

|

[Store Dim]

In Snowflake schema, dimension tables like **Product** or **Customer** are normalized into sub-tables.

Quick Conclusion

- ✓ **Star Schema:** Best for simplicity and fast querying — preferred in reporting and BI.
 - ✓ **Snowflake Schema:** Best for complex hierarchies, better storage efficiency, but more joins.
 - ✓ **Snapshot & Factless Fact Tables:** Solve specific modelling scenarios for tracking progress, periodic summaries, or event occurrence.
-

SCD's

Slowly Changing Dimensions (SCDs)

SCDs deal with how you handle **changes in dimension data over time** in a data warehouse.

For example, if a **Customer** changes their address or a **Product** gets rebranded, how do you capture that history?

Types of SCDs

SCD Type 0 - Fixed Dimension (Never Changes)

- No changes allowed after initial load.
 - Example: Date Dimension — historical dates never change.
-

SCD Type 1 - Overwrite (No History Preserved)

- Simply overwrite the old data with the new data.
- No history is kept.

Example:

Customer_ID	Name	City
1001	John Smith	Mumbai

If John moves to Bangalore:

Customer_ID	Name	City
1001	John Smith	Bangalore

Old city 'Mumbai' is lost.

SCD Type 2 - Add New Row (Full History Preserved)

- A new row is added for every change.
- Tracks history with:
 - Surrogate Key
 - Effective Date
 - Expiry Date or Current Flag

Example:

Surrogate_Ke y	Customer_I D	Nam e	City	Start_Dat e	End_Dat e	Current_Fla g
1	1001	John Smit h	Mumbai	2023-01- 01	2024-06- 30	N
2	1001	John Smith	Bangalor e	2024-07- 01	NULL	Y

SCD Type 3 - Add New Column (Partial History Preserved)

- Keep limited historical information in the same row using extra columns.

Example:

Customer_ID	Name	Current_City	Previous_City
1001	John Smith	Bangalore	Mumbai

- Only tracks **previous** value, not full history.
-

SCD Type 4 - Historical Table

- Maintain a separate **history table** to track all changes.
 - Main dimension table holds only current data.
-

SCD Type 6 - Hybrid (Combination of Type 1, 2, and 3)

- Combines overwriting, row versioning, and additional columns for flexible history tracking.
 - Common in complex scenarios.
-

Bonus: SCD in Snowflake

- ✓ In Snowflake, SCDs are implemented using SQL MERGE statements, Streams, Tasks, or DBT workflows.
 - ✓ You can automate SCD Type 2 handling using DBT **snapshots**, which track changes and preserve history.
-

Summary Table

SCD Type	Description	Tracks History?
0	Fixed, never changes	No
1	Overwrite existing data	No
2	Add new row per change	Yes (full history)
3	Add new column for change	Limited history
4	Separate history table	Yes

SCD Type	Description	Tracks History?
6	Hybrid of 1, 2, and 3	Yes (flexible)

natural key and surrogate key

Here's a clear explanation:

Natural Key vs Surrogate Key

Aspect	Natural Key	Surrogate Key
Definition	A key derived from the real-world business data	A system-generated, artificial key (usually numeric)
Source	Comes from source systems or business processes	Generated within the data warehouse (sequence, identity column)
Examples	Customer Email, Social Security Number, Product Code	Auto-incremented ID, UUID, Hash Key
Uniqueness	Should be unique based on business rules	Guaranteed unique by design
Stability	May change over time (e.g., product codes change)	Does not change once assigned
Usage	Often used for data integration or initial load	Used as the primary key in dimension tables to maintain referential integrity
Recommended for Data Warehouse	Not recommended as primary keys in DW, but stored for reference	Preferred for primary keys in DW, especially in SCD Type 2 scenarios

Example

Natural Key Example (Business-driven)

Product_Code	Product_Name
P001	iPhone 15
P002	Galaxy S23

- Product_Code is unique but could change if business rules change.
-

Surrogate Key Example (System-generated)

Product_SKey	Product_Code	Product_Name
101	P001	iPhone 15
102	P002	Galaxy S23

- Product_SKey is an artificial, numeric, system-generated key.
 - Used for joins and maintaining history.
-

Why Surrogate Keys Are Preferred in Data Warehousing

- ✓ They simplify joins between Fact and Dimension tables.
 - ✓ Help manage Slowly Changing Dimensions (SCD Type 2).
 - ✓ Avoid dependency on potentially changing business keys.
 - ✓ Provide guaranteed uniqueness within the warehouse.
-

Quick Summary

"Natural Keys come from the business world and might change; Surrogate Keys are generated by the system, stable, and ideal for maintaining referential integrity in data models."

How the data model behave when a sudden change is implemented (Eg a country changes its name)

How Data Model Behaves with Sudden Change (e.g., Country Name Change)

It Depends on How the Data Model is Designed:

Scenario 1: If Natural Key is used as Primary Key (Not Recommended)

Impact:

- If the country name *itself* is part of the key or used for joins, changing the country name creates data integrity issues.
- All fact tables and related dimension tables may break or require cascading updates.

Example:

If Country_Name = "Turkey" is used as the key, and the name changes to "Türkiye",

- All tables referencing "Turkey" would need updates.
- High risk of broken joins and referential integrity problems.

Scenario 2: If Surrogate Key is used as Primary Key (Recommended Approach)

Impact:

- The Surrogate Key remains unchanged.
- Only the **descriptive attribute** (country name) is updated in the dimension table.
- No impact on joins or relationships in the data model.
- Historical reporting depends on SCD implementation.

SCD Considerations:

SCD Type	Behavior for Country Name Change
Type 1 (Overwrite)	The country name is simply updated to the new name; no history is preserved.
Type 2 (New Row Added)	A new row is inserted with the new country name, preserving historical records with the old name.
Type 3 (Add Column)	A new column captures the previous country name, partial history is preserved.

 Recommended Best Practice in Data Modelling

- ✓ Use **Surrogate Keys** for dimension tables like Country.
 - ✓ If historical reporting is important, implement **SCD Type 2** so past reports show "Turkey" and future reports show "Türkiye" with clear historical distinction.
 - ✓ Ensure data dictionaries and metadata repositories are updated to reflect the change.
 - ✓ Communicate with business users to understand whether reports should reflect the new name retrospectively or only for future data.
-

Example Table Illustration (SCD Type 2)

Country_SKey	Country_Code	Country_Name	Start_Date	End_Date	Current_Flag
101	TR	Turkey	2000-01-01	2023-06-30	N
102	TR	Türkiye	2023-07-01	NULL	Y

Fact Table continues to reference the Surrogate Key, so relationships stay intact.

Summary Answer You Can Say in Interview

"If a sudden change like a country name update occurs, the impact on the data model depends on the use of keys and SCD strategy.

When designed properly using Surrogate Keys, the relationships remain unaffected.

The change is handled in the dimension table, and depending on the SCD approach, we can preserve full history, partial history, or overwrite the existing record to reflect the latest information."

working model for Ola/Uber Swiggy Zomato Netflix youtube dropbox Instagram

1. Ola/Uber (Cab Booking Platform) - Working Data Model

Key Entities (Dimensions):

- Customer
- Driver
- Vehicle
- Location
- Payment Method

Fact Table:

- **Trip Fact Table**
 - Trip_ID (PK)
 - Customer_ID (FK)
 - Driver_ID (FK)
 - Vehicle_ID (FK)
 - Pickup_Location_ID (FK)
 - Drop_Location_ID (FK)
 - Trip_Start_Time
 - Trip_End_Time
 - Distance
 - Fare_Amount
 - Payment_ID (FK)

 **2. Swiggy/Zomato (Food Delivery Platform) - Working Data Model**

Key Entities (Dimensions):

- Customer
- Restaurant
- Menu_Item
- Delivery_Partner
- Location
- Payment

Fact Table:

- **Order Fact Table**
 - Order_ID (PK)
 - Customer_ID (FK)
 - Restaurant_ID (FK)
 - Delivery_Partner_ID (FK)
 - Location_ID (FK)
 - Order_Time
 - Delivery_Time

- Total_Amount
 - Payment_ID (FK)
 - Order_Status
-

3. Netflix/YouTube (Streaming Platform) - Working Data Model

Key Entities (Dimensions):

- User
- Subscription Plan
- Content (Movie/Series/Video)
- Device
- Location

Fact Table:

- Viewing Activity Fact Table
 - View_ID (PK)
 - User_ID (FK)
 - Content_ID (FK)
 - Device_ID (FK)
 - Location_ID (FK)
 - Start_Time
 - End_Time
 - Duration
 - Completion_Percentage
 - Subscription_Plan_ID (FK)

4. Dropbox (Cloud Storage Platform) - Working Data Model

Key Entities (Dimensions):

- User
- File
- Folder
- Subscription Plan
- Device

Fact Table:

- **File Activity Fact Table**
 - Activity_ID (PK)
 - User_ID (FK)
 - File_ID (FK)
 - Folder_ID (FK)
 - Activity_Type (Upload, Download, Delete, Share)
 - Activity_Time
 - Device_ID (FK)
 - Subscription_Plan_ID (FK)
-

 **5. Instagram (Social Media Platform) - Working Data Model**

Key Entities (Dimensions):

- User
- Post
- Follower/Following
- Hashtag
- Location
- Device

Fact Tables:

- **Post Fact Table**
 - Post_ID (PK)
 - User_ID (FK)
 - Location_ID (FK)
 - Post_Time
 - Content_Type (Image/Video/Text)
- **Like/Comment Fact Table**
 - Interaction_ID (PK)
 - Post_ID (FK)
 - User_ID (FK)
 - Interaction_Type (Like/Comment)

- Interaction_Time
-

Bonus Notes for Interviews

- ✓ All these platforms rely on **fact tables** to track events (e.g., trips, orders, streams, uploads, posts).
 - ✓ **Dimension tables** provide descriptive context (e.g., user details, location, content).
 - ✓ Data Modelling focuses on scalability, performance, and supporting analytical/reporting use cases.
 - ✓ **SCDs** may apply to entities like User, Subscription Plans, or Location when details change.
 - ✓ Surrogate Keys are recommended for Dimension Tables to handle changes efficiently.
-

How to use DBT for documentation / visualization (also check about ER/Studio / ERwin)

1. DBT (Data Build Tool) for Documentation & Visualization

What is DBT?

DBT is primarily a tool for building, transforming, and documenting data models in modern data stacks like **Snowflake**, **BigQuery**, or **Redshift**, using SQL and version control.

How to Use DBT for Documentation:

✓ YAML Files for Model Documentation:

In your DBT project, you document models, columns, descriptions, and relationships in .yml files.

Example:

yaml

```
version: 2
models:
  - name: customer
    description: "Customer master table"
    columns:
      - name: customer_id
        description: "Unique ID for each customer"
      - name: customer_name
        description: "Full name of the customer"
```

✓ dbt Docs Generation:

Run the following command to generate interactive documentation:

```
dbt docs generate dbt docs serve
```

✓ Interactive Lineage Graph:

- DBT provides a **visual DAG (Directed Acyclic Graph)** showing how models depend on each other.
- You can click on each model to see its columns, descriptions, and relationships.
- Helps with understanding data lineage and model dependencies.

Where DBT Lacks:

- DBT is great for **transformations and lineage**, but it is **not a full-fledged conceptual or logical data modelling tool** like ERwin or ER/Studio.
 - It focuses on physical layer documentation in the modern data stack.
-

2. ER/Studio for Data Modelling & Documentation

Key Features:

- ✓ Comprehensive **conceptual, logical, and physical** data modelling.
- ✓ Supports **reverse engineering** from databases.
- ✓ Generates **ER diagrams**, relationship diagrams, and documentation.
- ✓ Can model for multiple platforms like Oracle, SQL Server, Snowflake, etc.
- ✓ Includes **data lineage** and **impact analysis** tools.
- ✓ Supports **naming standards, version control**, and collaboration.

When to Use:

- ✓ Ideal for enterprise-level, large-scale data modelling and metadata management.
 - ✓ When you need visual, printable ER diagrams for business and technical teams.
 - ✓ Useful for impact analysis during system changes.
-

3. ERwin Data Modeler for Data Modelling & Documentation

Key Features:

- ✓ One of the most popular enterprise data modelling tools.
 - ✓ Supports **conceptual, logical, and physical** data models.
 - ✓ Generates **database-specific DDL scripts**.
 - ✓ Visual ER diagrams and comprehensive metadata documentation.
 - ✓ Supports **SCDs, data dictionaries**, and data standards enforcement.
 - ✓ Reverse engineering capability to derive models from existing databases.
 - ✓ Supports collaborative model management and version control.
-

Quick Comparison Table

Tool	Strengths	Limitations
DBT	Modern ELT-focused, auto-generated docs, lineage graph	Not for conceptual/logical modelling; limited to physical layer

Tool	Strengths	Limitations
ER/Studio	Full-scale data modelling, reverse engineering, visualization	Commercial, can be expensive
ERwin	Industry-standard for data modelling, DDL generation, lineage	Commercial, some learning curve

Ideal Workflow for Data Modeller (Like You)

- ✓ Use **ERwin** or **ER/Studio** for conceptual, logical, and physical data modelling, documentation, and impact analysis.
 - ✓ Use **DBT** in modern data projects (e.g., Snowflake) for managing transformations and maintaining lineage/documentation of your data pipelines.
 - ✓ Combine both: ER/Studio/ERwin for initial design, DBT for implementation and ongoing pipeline documentation.
-