Top 51 Data Warehouse Interview Questions and Answers for 2025

Ace your upcoming interview with this list of data warehouse interview questions curated by a data engineer!

Dec 4, 2024 · 15 min read

CONTENTS

So, you've found your dream job in data and are preparing to ace the data warehousing part of the interview process. Having been on both sides of the table—as a candidate and as an interviewer—I know exactly how daunting the experience can be.

Over the years, I've conducted hundreds of technical interviews for data engineering roles and seen firsthand what separates a strong candidate from the rest: Good preparation and the ability to communicate effectively.

For data professionals, the interview process typically involves several stages. Some focus on theoretical concepts, others on live coding or take-home tests, and some assess your design and architecture skills. At the heart of it all, though, is data warehousing—your ability to understand, design, and optimize it.

This guide is here to help you navigate those stages with confidence. Whether brushing up on fundamental concepts, practicing scenario-based questions, or preparing to showcase your coding skills, you'll find everything you need to succeed. Let's dive right in!

Beginner-Level Data Warehouse Interview Questions

All interviews I conducted started with a few basic questions, even if it was for a senior position. It's a good practice to take the candidate progressively from the fundamentals and up. So, make sure your foundations are in great shape.

1. What is a data warehouse, and why is it used?

A data warehouse is a centralized repository that stores structured data from various sources. It is used primarily for reporting and data analysis, offering a unified, historical view of a company's data.

Read more in the What is a Data Warehouse blog post.

2. Can you explain the differences between OLAP and OLTP?

Understanding the difference between OLAP and OLTP is very important because they serve distinct purposes in data systems.

- **OLAP (online analytical processing)** is optimized for complex queries and historical data analysis. It is designed for read-heavy operations such as generating reports, visualizations, and trend analysis.
- **OLTP (online transaction processing)** Focuses on real-time transaction management, such as processing orders or recording customer payments. It is optimized for fast, write-heavy operations.

Feature	OLAP	OLTP
Purpose	Analyzing historical data	Managing transactional operations
Data volume	Large datasets	Small, real-time transactions
Query type	Complex, read-heavy queries	Simple, write-heavy queries
Schema design	Star or snowflake schema	Normalized schema
Examples	Dashboards, trend analysis	Banking transactions, order entry

Read more in the What is OLAP blog post.

3. What is a dimension table and a fact table?

Dimension tables and fact tables are the building blocks of a data warehouse schema. They work together to organize and represent data to facilitate meaningful analysis.

- **Dimension tables** contain descriptive attributes, such as customer names or product categories, that provide context to the data. They help answer questions such as "who," "what," "where," and "when."
- **Fact tables** contain quantitative data, such as sales figures or transaction amounts, which are the focus of analysis. Fact tables often reference dimension tables to provide a deeper understanding of metrics.

4. What are the stages of ETL in data warehousing?

The ETL process is fundamental to any data warehouse project. It transforms raw data into a structured, ready-to-analyze format and is necessary to ensure that the data warehouse is accurate and reliable.

- **Extract:** Data is gathered from multiple sources, such as relational databases, APIs, or flat files.
- **Transform:** The data is cleaned, formatted, and reshaped to match the data warehouse schema. This step may include removing duplicates, calculating new fields, or applying business rules.
- **Load:** The processed data is loaded into the data warehouse, where it becomes accessible for querying and analysis.

A more modern approach is ELT, where raw data is loaded as is, and the transformation process happens in the data warehouse.

To gain hands-on experience with ETL and ELT processes, check out ETL and ELT in Python, which guides you through implementing these steps programmatically.

5. Describe the star schema and snowflake schema. Which is better, and why?

Schemas provide a framework for organizing data in a data warehouse.

- **Star schema:** In this design, a central fact table is surrounded by denormalized dimension tables. It is simple, intuitive, and optimized for quick queries, making it suitable for most business intelligence use cases.
- **Snowflake schema:** A normalized version of the star schema, where dimension tables are split into additional tables to reduce redundancy. While it saves storage space, it can complicate queries and slow performance.

Feature	Star schema	Snowflake schema
Structure	Central fact table with denormalized dimension tables	Central fact table with normalized dimension tables
Complexity	Simple, fewer joins	Complex, more joins
Storage space	Uses more storage	Optimized for storage
Query performance	Faster, fewer joins	Slower, due to more joins
Use case	Simple reporting needs	Scenarios requiring minimal redundancy

The choice depends on the use case. Star schemas are better for simplicity and faster queries, while snowflake schemas are ideal for situations where minimizing redundancy is critical.

Thalia Barrera
Senior Data Science Editor at DataCamp. She enjoys simplifying concepts for engineers and data scientists through blogs, tutorials, and courses.

TOPICS

Data Engineering
Business Intelligence

The Top 39 Data Engineering Interview Questions and Answers in 2025

Top 51 Data Architect Interview Questions and How To Answer Them

Top 30 Cloud Computing Interview Questions and Answers (2025)

Top 30 Database Administrator Interview Questions for 2025

Intermediate-Level Data Warehouse Interview Questions

After the interviewer has made sure you have a solid foundation in data warehousing, they may move on to intermediate-level questions. This is where things start to get interesting.

6. What are slowly changing dimensions (SCD), and how do you handle them?

Slowly changing dimensions (SCD) refer to data in dimension tables that evolve gradually over time. For example, a customer's address may change, but the historical data needs to be preserved for accurate reporting.

There are three main types of SCD:

- **Type 1:** Overwrite the old data with new data (e.g., update the address directly).
- **Type 2:** Maintain historical data by adding a new record with a start and end date.
- **Type 3:** Keep limited historical data by adding new fields for the old and current values.

Type	Description	Example use case	Implementation approach
SCD type 1	Overwrite old data with new data	Correcting a typo in customer name	Update operation
SCD type 2	Maintain historical data by adding new records	Tracking changes in customer address over time	Insert new row with start and end dates
SCD type 3	Keep limited historical data using additional columns	Tracking "previous" and "current" department for an employee	Add columns for old and new values

Understanding these types is relevant for designing a data warehouse that supports current and historical reporting needs.

Read more in the Mastering Slowly Changing Dimensions blog post.

7. Can you describe your experience with ETL tools like Informatica, Talend, or AWS Glue?

Interviewers often look for hands-on experience with ETL tools, as they play an important role in data warehousing projects. Share specific examples, such as:

- How you used AWS Glue to automate ETL pipelines and process large volumes of data from S3 to Redshift.
- A project where you utilized Talend to extract and transform data from disparate sources, ensuring consistent formats.
- Your experience with Informatica in creating reusable workflows and monitoring ETL jobs for enterprise-scale data systems.

This is your opportunity to shine by sharing your real-life experience.

8. Explain the concept of partitioning in data warehousing.

Partitioning is a technique that improves query performance and manageability in a data warehouse by dividing large datasets into smaller, more manageable segments.

There are two common types of partitioning:

- **Horizontal partitioning:** Splits data across rows, such as dividing sales data by region or date.
- **Vertical partitioning:** Splits data across columns, often based on usage patterns.

Partitioning reduces query processing time and improves resource efficiency, especially for large datasets.

Partitioning can be related to indexing. Here are their differences:

Feature	Partitioning	Indexing
Purpose	Divides data into smaller segments for improved query performance	Provides fast access to specific rows
Granularity	Based on rows (e.g., by date)	Based on columns
Impact on storage	May increase storage slightly	Minimal impact
Use case	Large datasets with predictable query patterns	Queries filtering on indexed columns

9. What is a surrogate key, and why is it used?

A surrogate key is a unique identifier for each record in a table, typically a sequentially generated number. It is used in dimension tables as a substitute for natural keys (like customer ID or product code) to:

- Ensure uniqueness, even if natural keys change.
- Maintain consistent relationships between fact and dimension tables.
- Simplify join operations and improve query performance.

Surrogate keys are handy when dealing with complex schemas, where stable relationships are important.

Advanced-Level Data Warehouse Interview Questions

When moving on to more advanced-level questions, the interviewer expects to check your theoretical knowledge and previous experience handling more complex scenarios. Draw on your own expertise to answer these questions, as things may become more ambiguous.

10. How would you design a data warehouse for a large-scale organization?

Designing a data warehouse for a large organization requires careful planning to accommodate scalability, performance, and specific business needs. The process typically involves:

- **Requirement gathering:** Understanding business objectives, key performance indicators (KPIs), and data sources.
- **Data modeling:** Choosing a schema design (e.g., star, snowflake) based on reporting needs and data relationships.
- **Technology stack:** Selecting tools and platforms, such as Snowflake, Redshift, or BigQuery, that align with scalability and budget requirements.
- **ETL/ELT processes:** Designing pipelines to handle high volumes of data while ensuring data quality.
- **Performance optimization:** Implementing indexing, partitioning, and caching strategies for fast query execution.

This question evaluates your ability to handle end-to-end data warehouse design.

11. How do you maintain data quality in a data warehouse?

Poor data quality can lead to incorrect analyses and decisions, so it's important to implement good measures, which include:

- Validating data during the ETL process to check for errors or inconsistencies.
- Implementing data profiling to understand data patterns and identify anomalies.
- Setting up automated monitoring and alerts for data discrepancies.
- Regularly cleaning and de-duplicating data to increase accuracy and consistency.

Data quality is difficult to tackle and sometimes overlooked in practice. To better understand these concepts, I recommend taking the Introduction to Data Quality course.

12. Can you optimize query performance in a data warehouse? How?

Optimizing query performance is a common task for increasing efficiency and usability in a data warehouse. Some effective techniques include:

- **Indexing:** Create indexes on frequently queried columns to speed up lookups.
- **Partitioning:** Split large datasets into smaller segments for faster data retrieval.
- **Materialized views:** Pre-compute and store query results to reduce execution time for repetitive queries.
- **Denormalization:** Reduce joins by consolidating tables, particularly in reporting layers.
- **Query optimization:** Rewrite complex queries for better execution plans, leveraging database-specific features.

I recommend providing real-world examples of how you have applied these techniques to

strengthen your answer.

13. Explain the role of materialized views in data warehousing.

Materialized views are pre-computed query results stored for future use, significantly improving performance for recurring and complex queries. Unlike regular views, materialized views:

- Store results physically, eliminating the need to recompute them every time.
- Can be refreshed incrementally or periodically to maintain up-to-date data.
- Reduce the load on underlying tables and databases.

For instance, a materialized view might pre-aggregate daily sales data in a sales reporting system, allowing for faster analysis during peak reporting hours.

14. How do you approach incremental loading in ETL processes?

Incremental loading is a technique to update a data warehouse by loading only new or changed data, reducing processing time and resource usage. Common approaches include:

- **Timestamps:** Use a "last modified" timestamp column to identify new or updated records.
- **Change data capture (CDC):** Detect and extract changes directly from source systems, often through database logs or triggers.
- **Snapshot comparison:** Compare current data with previously loaded data to identify changes.

Incremental loading is especially important in large-scale data warehouses where full reloads would be impractical.

15. Discuss best practices for scalability in a data warehouse.

Scalability ensures that a data warehouse can handle growing data volumes and user demands without performance degradation. Best practices include:

- **Cloud-native solutions:** Use platforms like Snowflake, Redshift, or BigQuery that offer auto-scaling features.
- **Data partitioning:** Split data based on criteria like time or region to enable parallel processing.
- **Columnar storage:** Leverage columnar data storage for faster analytics and reduced storage costs.
- **Workload management:** Prioritize and allocate resources based on query complexity and user roles.
- **Regular maintenance:** Optimize database performance through periodic index rebuilding, statistics updates, and query audits.

Providing examples of implementing these practices in your previous roles demonstrates expertise in handling large-scale systems.

16. How do you handle schema changes in a data warehouse?

Schema changes are inevitable in data warehousing! Handling them efficiently minimizes disruptions and enhances data integrity. Strategies include:

- **Schema versioning:** Maintain multiple schema versions and migrate data incrementally to avoid impacting ongoing operations.
- **Backward compatibility:** Ensure new schema changes do not break existing queries by keeping legacy fields or creating views.
- **Automation tools:** Use tools like dbt or Liquibase to automate schema migration and rollback processes.
- **Impact analysis:** Identify dependencies such as queries, reports, or downstream systems that might be affected by schema changes and update them accordingly.
- **Testing:** Validate schema changes in a staging environment before deploying them to production.

For instance, when adding a new column to a fact table, you might initially populate it with default values to prevent errors in existing queries.

Code-Based Data Warehouse Interview Questions (SQL)

An interview for a data role will almost always include a SQL stage. It was certainly part of all the interviews I conducted because, let's be honest, data and analytics engineers need to have an advanced understanding of it. These SQL questions are specially tailored to data warehousing.

17. Write a SQL query to calculate the total sales for each product category in the past month.

This question evaluates your SQL skills and understanding of time-based filters. Here's a sample solution:

```
SELECT
    category_name,
    SUM(sales_amount) AS total_sales
FROM
    sales_fact_table
JOIN
    category_dimension_table
ON
    sales_fact_table.category_id = category_dimension_table.category_id
WHERE
    sales_date >= DATEADD(MONTH, -1, GETDATE())
GROUP BY
    category_name;
```


Follow-up questions include optimizing this query for a large dataset or explaining how indexes can improve performance.

18. How would you implement incremental data loading for a fact table?

This question checks your understanding of ETL/ELT processes. Provide a high-level explanation and code if relevant:

- **Identify new or changed data:** Use timestamps or change data capture (CDC).
- **Extract new data:** For example, use a SQL query:

```
SELECT *
FROM source_table
WHERE last_modified >= (SELECT MAX(last_loaded_time) FROM load_metadata);
```


- **Load into staging area:** Write the data to a staging table for validation.
- **Merge into fact table:** Use an `UPSERT` or `MERGE` operation to add new rows and update existing ones:

```
MERGE INTO fact_table AS target
USING staging_table AS source
ON target.id = source.id
WHEN MATCHED THEN
    UPDATE SET target.value = source.value
WHEN NOT MATCHED THEN
    INSERT (id, value) VALUES (source.id, source.value);
```


19. Write a query to find the top 3 customers by revenue for each product category.

This question tests your ability to use window functions. Example:

```
WITH ranked_customers AS (
    SELECT
        category_name,
        customer_id,
        SUM(sales_amount) AS total_revenue,
        RANK() OVER (PARTITION BY category_name ORDER BY SUM(sales_amount) DESC)
    FROM
        sales_fact_table
    JOIN
        category_dimension_table
    ON
        sales_fact_table.category_id = category_dimension_table.category_id
    GROUP BY
```

```
        category_name, customer_id
)
SELECT
    category_name, customer_id, total_revenue
FROM
    ranked_customers
WHERE
    rank <= 3;
```

POWERED BY datalab

20. How would you optimize a query that takes too long to execute?

This question combines coding and problem-solving. Steps to optimize:

- **Rewrite query:** Use efficient joins, avoid unnecessary subqueries, and apply proper indexing.
- **Use EXPLAIN plan:** Analyze the query execution plan to identify bottlenecks.
- **Partitioning:** If the table is large, use partitioning to reduce the data scanned.
- **Materialized views:** Pre-compute and cache expensive operations.

Example query improvement:

Before:

```
SELECT *
FROM orders
WHERE YEAR(order_date) = 2023;
```

POWERED BY datalab

Optimized:

```
SELECT *
FROM orders
WHERE order_date >= '2023-01-01' AND order_date < '2024-01-01';
```

POWERED BY datalab

You can further improve it by selecting only the necessary columns.

21. How would you design a schema for a star schema data warehouse with sales data?

This question involves conceptual design and implementation details. Provide a high-level overview:

- **Fact table:** Contains quantitative data (e.g., sales amount, quantity sold) with foreign keys to dimension tables. Example:

```
CREATE TABLE sales_fact (
        sale_id INT PRIMARY KEY,
        product_id INT,
        customer_id INT,
        store_id INT,
        time_id INT,
        sales_amount DECIMAL(10, 2),
        quantity_sold INT
    );
```

POWERED BY datalab

Dimension tables: Contain descriptive attributes for analysis. Example:

```
CREATE TABLE product_dimension (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    category_name VARCHAR(50)
);
```

POWERED BY datalab

22. Write a query to detect duplicate records in a table.

This question tests data quality validation skills.

```
SELECT
    id, COUNT(*) AS duplicate_count
FROM
    some_table
GROUP BY
    id
HAVING
    COUNT(*) > 1;
```

POWERED BY datalab

Follow-up: Explain how to remove duplicates:

```
DELETE
FROM some_table
WHERE id IN (
    SELECT id
    FROM (
        SELECT id, ROW_NUMBER() OVER (PARTITION BY id ORDER BY created_at) AS row
        FROM some_table
    ) AS duplicates
    WHERE row_num > 1
);
```

POWERED BY datalab

23. How would you implement a slowly changing dimension (SCD) of Type 2?

Type 2 SCD tracks historical changes by adding a new row for each change. Example implementation:

Check for existing records:

```
SELECT *
FROM dimension_table
WHERE business_key = 'some_key' AND current_flag = TRUE;
```

POWERED BY datalab

Insert new record and expire old one:

```
UPDATE dimension_table
SET current_flag = FALSE, end_date = GETDATE()
WHERE business_key = 'some_key' AND current_flag = TRUE;

INSERT INTO dimension_table (business_key, attribute, current_flag, start_date, e
VALUES ('some_key', 'new_value', TRUE, GETDATE(), NULL);
```

POWERED BY datalab

To prepare for this stage of the interview, check out these highly-rated courses to build a strong foundation and advance your knowledge:

- **Introduction to SQL:** Perfect for beginners, this course covers the essentials of SQL syntax and querying databases.
- **Intermediate SQL:** Level up your skills with advanced techniques like joins, subqueries, and window functions.
- **Data Manipulation in SQL:** Learn how to clean, aggregate, and analyze data directly in SQL.

Scenario-Based Data Warehouse Interview Questions

Your interview may also include a few scenario-based questions. For example, a whole stage could be reserved for you to solve a specific design problem. The key here is that there are no right or wrong answers, only guidelines on approaching these questions effectively.

24. How would you design a data warehouse for an e-commerce business?

This scenario tests your ability to tailor a data warehouse to a specific business domain. For an e-commerce business, the design might include:

- **Data sources:** Integrate data from transactional databases, web analytics platforms, customer relationship management (CRM) systems, and inventory systems.
- **Schema design:** Use a star schema with fact tables for sales transactions and dimensions for customers, products, and time.
- **ETL process:** Develop pipelines to handle large volumes of data, including incremental loading for transaction updates.
- **Performance optimization:** Partition the sales fact table by date to improve query performance and use materialized views for commonly used aggregations like daily revenue or top-selling products.
- **Analytics and reporting:** Ensure the warehouse supports dashboards for metrics like sales trends, customer retention, and inventory levels.

This question evaluates your ability to think holistically about data modeling, ETL, and

business needs.

25. How would you handle a situation where the data volume suddenly increases significantly?

This scenario checks your ability to manage scalability challenges. Steps could include:

- **Scaling infrastructure:** For cloud-based systems like Snowflake or Redshift, adjust compute resources to handle the increased load. For on-premises systems, ensure adequate storage and processing capacity.
- **Partitioning and indexing:** Reassess partitioning and indexing strategies to optimize performance for larger datasets.
- **ETL optimization:** Review ETL jobs to identify bottlenecks and improve efficiency, such as switching to incremental loading or parallel processing.
- **Query optimization:** Work with analysts to rewrite heavy queries and use materialized views or pre-aggregations.

These situations are common, so providing an example of a similar situation you've handled in the past can make your answer more compelling.

26. What would you do if you discovered data discrepancies in the warehouse?

This scenario tests your troubleshooting skills and attention to detail. Steps might include:

- **Identify the source:** Trace the data back through the ETL pipeline to pinpoint where the discrepancy originated.
- **Verify data:** Compare the warehouse data with the source systems to validate accuracy.
- **Fix the issue:** Update the ETL process to resolve the root cause, such as incorrect transformation logic or missing data.
- **Communicate:** Inform stakeholders of the issue and steps taken to address it.
- **Monitor:** Implement automated data validation checks to prevent similar issues in the future.

A structured approach like this shows your ability to maintain data quality and instill confidence in your data warehousing processes.

27. How would you migrate a data warehouse from an on-premises solution to the cloud?

Migrating to the cloud is a common challenge in modern data warehousing. Your response might include:

- **Assessment:** Evaluate the current on-premises system, identifying data volume, dependencies, and use cases.
- **Cloud platform selection:** Based on scalability, cost, and performance needs, choose a platform like Snowflake, Redshift, or BigQuery.
- **Data migration:** Use tools for bulk data transfer, such as AWS DMS or Snowpipe, and implement incremental loading to keep data current.
- **Schema and query optimization:** Adapt schemas and queries to use cloud-native features like columnar storage and serverless computing.
- **Testing and validation:** Validate data integrity and performance in the cloud environment before decommissioning the on-premises system.

This question assesses your ability to manage complex migration projects while minimizing downtime and data loss.

28. What strategies would you use to handle high data latency in a data warehouse?

High data latency can impact decision-making by delaying the availability of up-to-date data. To address this:

- **Optimize ETL pipelines:** Reduce latency by switching to ELT processes where transformations occur directly in the data warehouse.
- **Stream processing:** Integrating streaming solutions like Apache Kafka or AWS Kinesis for near-real-time data.
- **Batch window tuning:** Adjust batch processing schedules to reduce the delay in data availability.
- **Database optimization:** Use partitioning, indexing, and materialized views to speed up data access and processing.
- **Hybrid architecture:** Implement a mix of real-time and batch processing for different data needs, ensuring critical data is updated more frequently.

Such answers demonstrate your ability to handle performance challenges in dynamic data environments.

Technology-Specific Data Warehouse Interview Questions

Every data team has a specific stack they work with, and normally, they tailor their interviews to ask about those specific technologies. However, I think it's important to be at least aware of the different stacks and tools, so it doesn't hurt to review them at a high level.If the job you're applying to requires specific knowledge, you may need to research it further.

Snowflake

29. What features of Snowflake make it different from traditional data warehouses?

Snowflake stands out due to its unique architecture and features:

- **Separation of compute and storage:** Compute and storage scale independently, allowing for cost optimization and flexibility.
- **Built-in performance features:** Automatically manages tasks like clustering, indexing, and query optimization.
- **Time travel:** Allows users to query historical data and recover deleted data for up to 90 days.
- **Zero-copy cloning:** Enables instant creation of database clones without duplicating data.

30. How does Snowflake handle concurrency issues?

Snowflake's multi-cluster architecture supports high concurrency by automatically spinning up additional compute clusters during peak demand.

I recommend taking the Introduction to Snowflake course to learn more about it and get hands-on practice.

Amazon Redshift

31. How does Redshift differ from traditional relational databases?

Redshift particularly stands out for the following reasons:

- **Columnar storage:** Optimized for analytical queries by storing data in columns instead of rows, reducing I/O.
- **Massively parallel processing (MPP):** Distributes queries across multiple nodes to handle large datasets efficiently.
- **Materialized views and result caching:** Improves query performance by precomputing and reusing results.

32. What strategies can you use to improve query performance in Redshift?

These are some strategies you should apply if using Redshift:

- Use **sort keys** and **distribution keys** to optimize data placement and access.
- Analyze and vacuum tables to maintain table health and remove unused disk space.
- Use compression encoding to reduce storage and improve I/O efficiency.

33. Redshift vs. Snowflake: Which would you recommend for a small team with limited resources?

Snowflake is often better for small teams because it is a fully managed, serverless model that requires minimal administrative overhead. Redshift may require more configuration and tuning but can be more cost-effective for predictable workloads.

I recommend taking the Introduction to Redshift course to gain hands-on experience with this powerful data warehousing solution.

Google BigQuery

34. What is unique about BigQuery's architecture?

BigQuery stands out for the following features:

- **Serverless architecture:** Automatically handles resource allocation and scaling, allowing users to focus on queries rather than infrastructure.
- **Query pricing model:** Charges based on the amount of data processed rather than the infrastructure used.
- **Built-in machine learning (BigQuery ML):** Allows users to create and deploy ML models using SQL.

35. How does BigQuery handle partitioning and clustering?

BigQuery works as follows:

- **Partitioning:** Divides tables into segments based on criteria like date, optimizing query performance.
- **Clustering:** Organizes data within a partition based on columns, improving query performance for specific access patterns.

I recommend exploring the **Introduction to BigQuery** course for hands-on practice.

Databricks

36. How does Databricks differ from traditional data warehouses?

Databricks combines data warehousing and data lake capabilities with its **Lakehouse architecture**, providing:

- Unified data storage for structured and unstructured data.
- Built-in support for advanced analytics, including machine learning and AI.
- Delta Lake, a storage layer that ensures data reliability with ACID transactions.

37. What is Delta Lake, and why is it important?

Delta Lake is an open-source storage layer that:

- Ensures data consistency with ACID transactions.
- Supports schema enforcement and evolution.
- Handles data versioning, making it easier to audit and rollback changes.

I recommend taking the **Introduction to Databricks** course to explore how to unify data engineering, analytics, and machine learning on one platform.

dbt (Data Build Tool)

38. What is dbt, and how is it used in data warehousing?

dbt (Data Build Tool) is a transformation tool that enables analysts and engineers to write, test, and document data transformations directly in SQL. It integrates with modern data warehouses like Snowflake, Redshift, and BigQuery. Its functionalities include:

- **Modeling:** Writing reusable SQL transformations using modular models.
- **Version control:** Integrating with Git for versioning and collaboration.
- **Testing:** Implementing tests to ensure data integrity.
- **Documentation:** Generating interactive documentation for a better understanding of data lineage.

39. How does dbt handle dependencies between models?

dbt uses a DAG (Directed Acyclic Graph) to manage dependencies between models. When running a transformation, dbt ensures that dependent models are executed in the correct order. This helps data consistency and eliminates the need for manual dependency management.

40. What are the benefits of using dbt for data transformations in data warehousing?

dbt has the following advantages:

- **Simplified transformation:** Enables SQL-based data transformations, reducing the need for complex ETL pipelines.
- **Collaboration:** Allows teams to work together using version control and standardized SQL.
- **Automation:** Automates dependency management and execution, making workflows more efficient.
- **Integration:** Works natively with modern data warehouses, leveraging their compute power.

I recommend the **Introduction to dbt** course to help you master its modeling capabilities as dbt is adopted by more and more data teams.

Apache Airflow

41. What is Apache Airflow, and how is it used in data warehousing?

Apache Airflow is an orchestration tool used to programmatically author, schedule, and monitor workflows, making it essential for managing ETL/ELT processes in data warehousing. Typical use cases include:

- Automating data ingestion pipelines.
- Managing complex dependencies in ETL processes.
- Scheduling regular updates to data models in a data warehouse.

42. How does Airflow handle dependencies in workflows?

Airflow uses a DAG (Directed Acyclic Graph) to define workflows. Each task in the DAG represents an operation (e.g., loading data, running transformations), and dependencies between tasks are defined explicitly.

- Airflow ensures tasks are executed in the correct order based on these dependencies.
- It also supports retries, backfilling, and triggering workflows conditionally.

43. What are some best practices for using Airflow in a data warehouse project?

Some best practices for Airflow include:

- **Use clear task names:** Ensure tasks are named descriptively to make DAGs easier to understand.
- **Optimize task granularity:** Avoid creating overly granular tasks that could slow down execution or complicate debugging.
- **Leverage XComs:** Use XComs (cross-communication) to pass small amounts of data between tasks.
- **Monitor performance:** Use Airflow's monitoring features to identify and address bottlenecks.
- **Modularize DAGs:** Keep DAG definitions modular and reusable to reduce maintenance overhead.

44. How would you use Airflow and dbt together in a data warehouse project?

Airflow and dbt complement each other by integrating orchestration and transformation:

- Use Airflow to schedule and trigger dbt runs as part of larger workflows.
- Airflow can manage upstream processes like data ingestion and downstream processes like report generation, while dbt handles the transformation logic within the data warehouse.

Example: Create an Airflow DAG that ingests raw data, triggers dbt to transform it, and then notifies stakeholders once the data is ready for reporting.

I recommend taking the **Introduction to Airflow** course to learn how to orchestrate data pipelines effectively. The knowledge is transferable to other orchestration tools.

Behavioral Data Warehouse Interview Questions

Behavioral questions are often reserved for senior or manager positions, but you could face them at any level. These questions are not as technical, and their objective is to check how you handle complex situations, teamwork, pressure, etc. This is when you need to bring your previous experience stories.

45. Can you share an example of a challenging data warehousing project you worked on and how you approached it?

This question evaluates your problem-solving skills and ability to handle complex challenges. You could frame your answer using the STAR method:Start by describing the project context (e.g., building a data warehouse for a new product launch with tight deadlines). Then, explain your role and responsibilities, detailing your steps, such as collaborating with stakeholders, designing the schema, and implementing ETL pipelines. Finally, highlight the outcome, like meeting the deadline or enabling actionable insights.

46. How do you handle conflicting requirements from stakeholders during data warehouse design?

Conflicting requirements can arise in any collaborative project. To address them, begin by conducting one-on-one sessions to clarify priorities and objectives. Use frameworks like MoSCoW (Must Have, Should Have, Could Have, Won't Have) to rank the requirements. Suggest compromises, such as phased implementations or intermediate data models, and explain how your design aligns with business goals. Clear and transparent communication is essential to gain stakeholder buy-in.

47. Describe a situation where you had to optimize an underperforming data pipeline. What did you do?

Start by identifying the bottleneck using monitoring tools or analyzing logs. Then, take specific actions, such as rewriting inefficient SQL queries, implementing incremental loading, or parallelizing tasks. Validate the pipeline after optimization to ensure improved performance. Share measurable improvements, like cutting processing time in half or increasing pipeline reliability.

48. How do you enhance collaboration with cross-functional teams on a data warehousing project?

Explain how you establish communication channels and schedule regular meetings to align

goals with teams like engineering, analytics, and business units. Then, document processes, such as data models and ETL pipelines, to provide transparency. Use tools like Jira or Slack to track progress and resolve conflicts when they arise. Express how you can help balance priorities and ensure alignment by acting as a mediator.

49. How have you handled a situation where a critical ETL job failed during peak business hours?

Begin by describing the immediate response: quickly identify the root cause of the failure using monitoring tools and logs. Implement a quick fix or rerun the ETL job to restore functionality. Communicate with stakeholders about the issue, estimated resolution time, and potential impact. Finally, discuss how you conducted a root cause analysis and implemented preventive measures, such as enhanced monitoring or failover mechanisms, to avoid future disruptions.

50. How do you stay current with emerging trends and technologies in data warehousing?

Mention the industry resources you regularly follow, such as blogs, webinars, or certifications (e.g., AWS, Snowflake). Highlight your engagement with professional communities by attending meetups, contributing to forums, or joining LinkedIn groups. Additionally, explain how you experiment with new tools and techniques in side projects or proof-of-concept implementations to stay ahead in the field.

51. Can you describe a time when you improved a process or introduced an innovation in a data warehousing system?

Start by identifying the problem, such as slow query performance or data quality issues. Then, explain your innovative solution, like introducing materialized views, automating validation scripts, or integrating a new tool. Describe how you implemented and tested the improvement with your team, and share measurable outcomes, such as reduced query times or increased user satisfaction.

Tips to Ace Your Data Warehousing Interview

Preparing for a data warehousing interview requires combining technical expertise, problem-solving skills, and storytelling ability. Here are some actionable tips to help you succeed:

Refresh your basic concepts

Even if you have years of experience, revisiting the fundamentals can help you answer questions confidently. Key areas to focus on include:

- Understanding ETL/ELT processes.
- The differences between OLAP and OLTP.
- Schema designs like star and snowflake schemas.
- Data quality and consistency best practices.

Brush up on these concepts to ensure you can articulate them clearly, especially in beginner-level questions.

Collect previous experiences and create compelling stories

Interviewers love to hear about real-world examples. Spend time reflecting on past projects and challenges you've faced in your career. Structure your stories using the STAR method (Situation, Task, Action, Result) to provide a clear and engaging narrative. For example:

- A time you optimized a slow-running query or pipeline.
- How you handled a schema change that affected downstream analytics.
- A project where you successfully implemented a data warehouse for a specific business case.

These stories demonstrate your hands-on experience and problem-solving skills.

Practice coding and problem-solving

Expect questions that require writing SQL queries or solving technical problems. Practice on DataCamp, focusing on SQL challenges. Revisit topics like:

- Writing efficient joins, window functions, and subqueries.
- Detecting duplicates or identifying outliers in data.
- Optimizing queries for large datasets.

Practice with real projects

Hands-on experience is crucial for data warehousing roles. If you're not currently working on projects, create your own by:

- Building a small data warehouse using cloud platforms like Snowflake, Redshift, or BigQuery.
- Writing SQL queries to solve common analytical problems.
- Designing a data pipeline that integrates batch and real-time data processing.

Document your projects to show tangible results during the interview, and use them as discussion points.

Study interview questions thoroughly

Go through common interview questions to identify areas for further preparation. This blog post itself is an excellent resource! Reviewing a comprehensive set of questions ensures you're not caught off guard.

Be ready with questions for the interviewer

Demonstrate your interest in the role and the company by asking thoughtful questions, such as:

- "What challenges is the team currently facing in managing the data warehouse?"
- "How does the company handle schema evolution and data quality at scale?"
- "What tools or technologies are you planning to adopt in the future?"

This helps you gauge if the role aligns with your career goals while leaving a positive impression. When I was an interviewer, I would rate a candidate higher if they asked good questions. Interviewing is a two-way process!

Conclusion

Preparing for a data warehousing interview can seem daunting, but you can confidently tackle it with the right approach and resources. By refreshing your basic concepts, practicing with real-world scenarios, and studying the right set of questions (like the ones in this blog post!), you'll be well-equipped to showcase your skills.

To further enhance your preparation, here are some excellent resources:

- Data Warehousing Concepts: Master the fundamentals of data warehousing.
- Introduction to Snowflake: Learn about one of the most popular cloud-based data warehousing platforms.
- Introduction to Databricks: Dive into Databricks and explore its unified data analytics capabilities.
- ETL and ELT in Python: Gain hands-on experience building data pipelines using Python.
- Introduction to dbt: Learn how to transform data in your warehouse with dbt's powerful modeling tools.

AUTHOR
Thalia Barrera

Thalia Barrera is a Senior Data Science Editor at DataCamp with a master's in Computer Science and over a decade of experience in software and data engineering. Thalia enjoys simplifying tech concepts for engineers and data scientists through blog posts, tutorials, and video courses.

TOPICS
Data Engineering | Business Intelligence

Learn more about data engineering with these courses!

TRACK
Data Engineer in Python
0 min
Gain in-demand skills to efficiently ingest, clean, manage data, and schedule and monitor pipelines, setting you apart in the data engineering field.
See Details → Start Course

COURSE
Introduction to Data Engineering
4 hr 120.1K
Learn about the world of data engineering in this short course, covering tools and topics like ETL and cloud computing.
See Details → Start Course

COURSE
Data Warehousing Concepts
4 hr 32.7K
This introductory and conceptual course will help you understand the fundamentals of data warehousing.
See Details → Start Course

See More →

Related

BLOG

Top 51 Data Architect Interview Questions and How To Answer...

Prepare to excel in your next data architect interview with this comprehensive guide, which includes top...

Fotos Morina • 15 min

BLOG

Top 30 Cloud Computing Interview Questions and...

Explore key cloud computing interview questions and answers, from basic to advanced, to help you prepare for cloud-...

Marie Fayard • 15 min

BLOG

Top 30 Database Administrator Interview Questions for 2025

This guide covers the top database administrator interview questions, from basic to advanced topics, helping you...

Kurtis Pykes • 15 min

BLOG

Top 27 Azure Data Factory Interview Questions and...

Prepare for your upcoming data engineering interview with this guide to answering the most frequently asked...

Kurtis Pykes • 15 min

BLOG

Top 30 SQL Server Interview Questions (2025)

This comprehensive guide provides a curated list of SQL Server interview questions and answers, covering topics...

Kevin Babitz • 14 min

See More →

Grow your data skills with DataCamp for Mobile

Make progress on the go with our mobile courses and daily 5-minute coding challenges.

Download on the App Store | GET IT ON Google Play

LEARN

- Learn Python
- Learn AI
- Learn Power BI
- Learn Data Engineering
- Assessments
- Career Tracks
- Skill Tracks
- Courses
- Data Science Roadmap

DATA COURSES

- Python Courses
- R Courses
- SQL Courses
- Power BI Courses
- Tableau Courses
- Alteryx Courses
- Azure Courses
- AWS Courses
- Google Sheets Courses
- Excel Courses
- AI Courses
- Data Analysis Courses
- Data Visualization Courses
- Machine Learning Courses
- Data Engineering Courses
- Probability & Statistics Courses

DATALAB

- Get Started
- Pricing
- Security
- Documentation

CERTIFICATION

- Certifications
- Data Scientist
- Data Analyst
- Data Engineer
- SQL Associate
- Power BI Data Analyst
- Tableau Certified Data Analyst
- Azure Fundamentals
- AI Fundamentals

RESOURCES

- Resource Center
- Upcoming Events
- Blog
- Code-Alongs
- Tutorials
- Docs
- Open Source
- RDocumentation
- Book a Demo with DataCamp for Business
- Data Portfolio

PLANS

- Pricing
- For Students
- For Business
- For Universities
- Discounts, Promos & Sales
- Expense DataCamp
- DataCamp Donates

FOR BUSINESS

- Business Pricing
- Teams Plan
- Data & AI Unlimited Plan
- Customer Stories
- Partner Program

ABOUT

- About Us
- Learner Stories
- Careers
- Become an Instructor
- Press
- Leadership
- Contact Us
- DataCamp Español
- DataCamp Português
- DataCamp Deutsch
- DataCamp Français

SUPPORT

- Help Center
- Become an Affiliate

clear as data