Meesho Data Engineer Interview Guide – Experienced

Overview of the Interview Process

Meesho's interview process is comprehensive, focusing on technical, system design, and managerial skills. Below is a high-level breakdown of the rounds:

- **Round 1:** Preliminary Coding Assessment (Hackerrank)
- **Round 2:** Technical Interview 1 (Coding & Data Structures)
- **Round 3:** Technical Interview 2 (SQL, Big Data Concepts, and DSA)
- **Round 4:** System Design & ETL Design Round
- **Round 5:** Techno-Managerial Round
- **Round 6:** HR Interview

Round 1: Preliminary Round (Coding Assessment Round)

This round serves as the initial screening for the SDE-2 Data Engineering role at Meesho. The assessment is conducted on the Hackerrank platform and evaluates your problem-solving skills in multiple key areas. Here's a detailed breakdown of the round:

1. Test Duration and Structure:

- **Total Duration**: 1 hour
- **Sections**:
    1. **SQL Coding Questions**: Focus on SQL query formulation and manipulation.
    2. **Data Structure Question**: A coding problem based on data structures, typically involving arrays, lists, or trees.
    3. **Algorithm Question**: A problem based on algorithmic thinking, requiring you to implement a solution that is both efficient and correct.

Each section is individually timed, and it's crucial to manage your time well to complete all the questions. This round is designed to test your ability to write code under pressure while applying core data engineering concepts.

2. Detailed Breakdown of the Sections:

A. SQL Coding Questions:

- **Types of SQL Questions**:
    - Queries involving window functions (e.g., ROW_NUMBER(), RANK(), DENSE_RANK()), joins (INNER, LEFT, RIGHT, etc.), and subqueries.
    - Manipulating data using GROUP BY and HAVING clauses.

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

    - Common tasks such as aggregate functions (e.g., COUNT(), SUM(), AVG()) and working with date/time functions.
- **Key Focus**:
    - Window functions: Meesho's data engineering role likely requires strong SQL skills for large-scale data manipulation, and window functions are essential in these scenarios.
    - Aggregation and filtering: Understanding how to efficiently process and summarize large datasets is crucial.
    - Optimizing queries: Demonstrating that you can write efficient SQL queries (in terms of time and space complexity) will be a big plus.

B. Data Structure Question:

- **Types of Problems**:
    - **Array manipulations** (e.g., rotating an array, finding pairs with a given sum).
    - **Linked Lists** (e.g., reversing a linked list, detecting cycles).
    - **Trees**, specifically **Binary Search Trees (BSTs)** (e.g., checking if a tree is balanced, finding the lowest common ancestor).
    - **Stacks and Queues** (e.g., implementing a queue using stacks, evaluating expressions).
- **Key Focus**:
    - **Binary Search Trees**: The interviewer might test your understanding of common tree operations like traversals (preorder, inorder, postorder), balancing, and searching.
    - **Efficiency**: Focus on writing efficient solutions with an optimal time and space complexity.

C. Algorithm Question:

- **Types of Problems**:
    - **Sorting and searching algorithms** (e.g., merge sort, quicksort, binary search).
    - **Dynamic Programming** (e.g., solving problems like knapsack, longest common subsequence).
    - **Graph algorithms** (e.g., depth-first search, breadth-first search, shortest path algorithms like Dijkstra).
- **Key Focus**:
    - **Problem-solving approach**: Focus on breaking down the problem into smaller sub-problems and selecting the right algorithm to solve each one efficiently.
    - **Time and space complexity**: Be sure to analyze your solution and discuss its efficiency, as interviewers will often be looking for optimized approaches.

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Round 2: Technical Interview 1 (Coding & DSA Round)

This round was focused on coding and data structures (DSA) and lasted for 1 hour and 30 minutes. It was conducted on the Bar Raiser Interview Platform and involved solving two medium to hard-level DSA problems. Here's a detailed breakdown of the types of problems you encountered and tips to approach them:

Problem 1: Trapping Rain Water (Array-based problem)

Problem Statement: The problem requires calculating the amount of water that can be trapped between elements in an array after raining. The elements of the array represent elevations, and the water is trapped in the valleys formed by higher elevations on both sides.

Example**:**

Input: `arr[] = {2, 0, 2}`
Output: `2`

Explanation: The structure forms a valley in the middle, where 2 units of water can be trapped.

Approach:

1. Two-Pointer Technique: This is a popular solution for the Trapping Rain Water problem. You maintain two pointers (left and right) at the two ends of the array.
2. Track Max Heights: Use two arrays to store the maximum heights from the left and right sides. For each element, the trapped water is determined by the minimum of the max heights from both sides minus the current height.

   Steps:

   - Initialize two pointers left and right at the start and end of the array, respectively.
   - Maintain two variables left_max and right_max to store the maximum heights encountered from the left and right, respectively.
   - Traverse the array by moving the pointers towards each other, calculating the trapped water at each step.

Time Complexity: O(n), as you traverse the array once. Space Complexity: O(1) if you only use two variables for tracking max heights.

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Example Solution (Python):

```
def trap(height):
    left, right = 0, len(height) - 1
    left_max, right_max = 0, 0
    water_trapped = 0

    while left <= right:
        if height[left] <= height[right]:
            if height[left] >= left_max:
                left_max = height[left]
            else:
                water_trapped += left_max - height[left]
            left += 1
        else:
            if height[right] >= right_max:
                right_max = height[right]
            else:
                water_trapped += right_max - height[right]
            right -= 1

    return water_trapped
```

Problem 2: SQL Query to Find Average Sessions per User (SQL Problem)

Problem Statement: The problem involves calculating the average number of sessions per user within a 30-day period, ending on a specific date (2019-07-27), using a table Activity with columns:

- user_id: User identifier
- session_id: Session identifier
- activity_date: Date of the activity
- activity_type: Type of the activity (enum values: open_session, end_session, scroll_down, send_message)

The goal is to find the average number of sessions per user that had at least one activity in the 30-day window.

Approach:

1. Subquery: We can use a subquery to filter the activities for the last 30 days ending on 2019-07-27.
2. GROUP BY: Group by user_id to calculate the number of sessions for each user in the given time period.
3. COUNT DISTINCT: Use COUNT(DISTINCT session_id) to count the number of unique sessions per user.
4. Final Calculation: Find the average of the session counts per user.

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

SQL Query

```
SELECT ROUND(AVG(session_count), 2) AS avg_sessions_per_user
FROM (
    SELECT user_id, COUNT(DISTINCT session_id) AS session_count
    FROM Activity
    WHERE activity_date BETWEEN '2019-07-27' - INTERVAL 30 DAY AND '2019-
07-27'
    GROUP BY user_id
) AS sessions_per_user;
```

Explanation:

- The subquery filters activities between 2019-06-27 and 2019-07-27.
- The COUNT(DISTINCT session_id) ensures that we only count unique sessions per user.
- The outer query calculates the average session count and rounds it to 2 decimal places.

Problem 3: Zigzag Order Traversal of a Binary Tree (Tree-based Problem)

Problem Statement: Given a binary tree, you are required to print the nodes in a zigzag order. The traversal starts from the root, and in each level, nodes alternate between left-to-right and right-to-left.

For example, for the tree:

```
markdown
Copy code
       1
     /   \
    2     3
   / \   / \
  4   5 6   7
```

The zigzag traversal would be: `1 3 2 4 5 6 7`.

Approach:

1. Level Order Traversal: Use a queue to implement level-order traversal (BFS).
2. Track Level Direction: Maintain a flag (left_to_right) to toggle the direction for each level.
3. Alternate Direction: After processing each level, reverse the order of traversal for the next level.

Steps:

- Use a queue to perform a BFS traversal.
- For each level, check the direction of traversal (left_to_right flag).
- Reverse the order of traversal after each level.

Time Complexity: O(n), where n is the number of nodes in the tree. Space Complexity: O(n), for the space used by the queue.

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Example Solution (Python):

```
from collections import deque

def zigzagLevelOrder(root):
    if not root:
        return []

    result = []
    queue = deque([root])
    left_to_right = True

    while queue:
        level = []
        level_size = len(queue)

        for _ in range(level_size):
            node = queue.popleft()
            level.append(node.val)

            if node.left:
                queue.append(node.left)
            if node.right:
                queue.append(node.right)

        if not left_to_right:
            level.reverse()

        result.append(level)
        left_to_right = not left_to_right

    return result
```

Explanation:

- A deque is used for efficient popping from the front and adding to the back of the queue.
- After processing each level, if left_to_right is False, the order of the level list is reversed.
- The flag left_to_right toggles after each level to alternate the direction.

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Round 3: Technical Interview 1 (SQL Coding/DSA Round & Big Data Concepts)

This round was a mix of **SQL** (focused on window functions and joins), **Data Structures** (stack-based problem), and **Big Data concepts** (Spark, Databricks, and MapReduce). Here's a detailed breakdown of the questions and tips on how to approach each of them:

Part 1: SQL Coding Question

Question: Find All Numbers that Appear at Least Three Times Consecutively

Problem Statement: You are given a table Logs with the following structure:

You need to find all numbers that appear at least three times consecutively.

Example Table:

```
+-----+------+
| Id  | Num  |
+-----+------+
| 1   | 1    |
| 2   | 1    |
| 3   | 1    |
| 4   | 2    |
| 5   | 1    |
| 6   | 2    |
| 7   | 2    |
+-----+------+
```

Expected Output:

```
+----------------+
| ConsecutiveNums |
+----------------+
| 1              |
+----------------+
```

Approach: This problem can be solved using **window functions** (specifically LAG or LEAD). You need to compare the current num with the previous two rows' num values to identify the numbers that appear consecutively.

SQL Solution:

```
SELECT DISTINCT num AS ConsecutiveNums
FROM (
    SELECT num,
           LEAD(num, 1) OVER (ORDER BY id) AS next_num,
           LEAD(num, 2) OVER (ORDER BY id) AS next_to_next_num
    FROM Logs
) AS temp
WHERE num = next_num AND num = next_to_next_num;
```

Explanation:

- LEAD(num, 1): This function returns the next value of num for each row.
- LEAD(num, 2): This function returns the value after the next one.
- You can then check if the current num, next_num, and next_to_next_num are the same to determine consecutive numbers.

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Part 2: Data Structure Problem (Stack)

Question: The Stock Span Problem

Problem Statement: The Stock Span Problem involves finding the **span of stock's price** for each day. The span is the number of consecutive days before the current day's price that are less than or equal to the current day's price.

Example:

Input: `[100, 80, 60, 70, 60, 75, 85]`
Output: `[1, 1, 1, 2, 1, 4, 6]`

Approach:

1. Use a **stack** to store indices of the array elements.
2. For each element, pop from the stack until you find an element greater than or equal to the current element.
3. The span of the current element will be the difference between the current index and the index of the last element in the stack.

Time Complexity: O(n), as each element is pushed and popped from the stack once.

Example Solution (Python):

```
def stockSpan(prices):
    span = []
    stack = []

    for i, price in enumerate(prices):
        while stack and prices[stack[-1]] <= price:
            stack.pop()
        span.append(i + 1 if not stack else i - stack[-1])
        stack.append(i)

    return span
```

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Part 3: Big Data Concepts

1. Handling Skewness in Data:

- **Skewness** happens when one or more values dominate a data partition, leading to an imbalance in processing.
- **Salting Mechanism**: A technique where a random value (salt) is added to skewed keys to distribute them evenly across partitions.
- **Broadcast Join**: Involves sending the smaller dataset to all nodes, avoiding shuffling of large datasets, and improving join performance.

Example Explanation:

- **Salting**: If the dataset contains highly skewed data (like a few users with a disproportionate number of transactions), you add a random "salt" value to the user ID to ensure that transactions are evenly distributed across partitions.
- **Broadcast Join**: For a small dataset (e.g., a lookup table), you can broadcast it to all nodes in the cluster to perform a local join on each node, reducing the need for data shuffling.

2. Spark Read Operation (Delta vs Parquet):

Question: Explain how spark.read.format("delta").load() **works.**

- Delta is an ACID-compliant storage layer on top of Apache Parquet. It supports features like ACID transactions, time travel, and schema evolution, making it suitable for large-scale data lakes.
- Parquet is a columnar storage format optimized for read-heavy operations, providing compression and efficient querying for big data workloads.
- Delta extends Parquet by adding transaction logs, allowing you to perform operations like upserts and deletes.

3. Mappers & Combiner in MapReduce:

Question: How to optimize mappers using properties in MapReduce?

- Mappers are optimized by tuning properties such as:
    - MapReduce Input Splits: Configuring the size of input splits can optimize the number of mappers.
    - Combiner: A local reducer that runs on the mapper side to reduce the amount of data shuffled over the network. It performs the same operation as the reducer but at a local level.

4. Managed vs Unmanaged Tables:

- **Managed Tables**: These tables are managed by Databricks or Spark. When you delete a managed table, both the schema and data are deleted.
- **Unmanaged Tables**: Only the schema is managed, while the data remains in the location specified by the user.

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

5. Databricks Cluster Management:

- **Databricks Clusters**: Databricks provides a managed environment to run Spark jobs, but clusters can also be in YARN mode or standalone mode. Databricks provides cluster orchestration with automatic scaling.
- **Cluster Mode**: Clusters can either run in standalone mode (where Databricks manages all resources) or YARN mode (where YARN manages resource allocation).

6. Spark Executor Management (Databricks):

Given the scenario:

- **10 workers, 100GB RAM, 25 cores**:

  i) Number of Executors: The number of executors depends on how you configure Spark. For example, if each executor has 4 cores, you could have 25/4 = **6 executors** (rounding down).

  ii) Size of Each Executor: This depends on the configuration. If you assign 10GB RAM per executor, you will allocate **10GB of memory** to each executor.

  iii) Out of Memory (OOM) in Spark Driver: OOM errors happen when the Spark driver doesn't have enough memory to handle the metadata and task scheduling. This could occur if the driver is handling too much data or if the memory is not properly configured.

7. Java Code Using File Input Stream:

You were asked to write Java code to read a file using FileInputStream. Here's a basic approach:

```
import java.io.FileInputStream;
import java.io.IOException;

public class FileReader {
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("path/to/file.txt");
        int data;
        while ((data = fis.read()) != -1) {
            System.out.print((char) data);
        }
        fis.close();
    }
}
```

8. Sqoop Command for Importing Multiple Tables:

To import multiple tables using **Sqoop**, you can use the following commands:

```
sqoop import --connect jdbc:mysql://localhost/db --username user --password
pass --table table1 --target-dir /data/table1
sqoop import --connect jdbc:mysql://localhost/db --username user --password
pass --table table2 --target-dir /data/table2
# Repeat for all 5 tables
```

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Round 4: System Design & ETL Design Round

1. System Design: E-commerce Platform (Flipkart)

Q1. Design an e-commerce platform like Flipkart:

- What are the key components you would use to build the system (e.g., web server, database, load balancer, caching)?
- How would you ensure the system can handle millions of concurrent users?
- What database would you choose for handling transactional and non-transactional data? Why?
- How would you design the architecture to handle high availability and scalability?
- How would you handle user sessions in a distributed environment?
- What role would Kafka or similar event-driven platforms play in your architecture?
- How would you implement caching to reduce load on the database and improve performance?
- What monitoring tools would you use for performance and fault tolerance?

2. Kafka and Spark Streaming (ETL Pipeline)

Q2. Design an ETL pipeline using Kafka and Spark Streaming:

- Given a streaming dataset from Kafka, how would you ingest the data in real-time using Spark?
- How would you clean the data by filtering out records with null values in specific fields, such as user_id?
- After cleaning, how would you store the transformed data into a Delta Lake (on S3 or HDFS)?
- How would you manage the streaming data schema and handle schema evolution in Delta Lake?
- Explain how to handle checkpointing in Spark Streaming and why it's important for fault tolerance.
- How would you ensure exactly-once processing for Kafka consumers in your Spark job?

Q3. How would you optimize your Spark Streaming ETL pipeline for high throughput and low latency?

- What strategies would you use for batch interval tuning and parallelism?
- How would you handle backpressure in Spark Streaming when the data volume exceeds the processing capacity?
- How would you scale your pipeline to handle millions of events per second?

3. Big Data Concepts & Streaming Architecture

Q4. Explain the architecture of Kafka:

- What are Kafka Producers, Consumers, and Brokers? How do they work together in Kafka's ecosystem?
- Explain the concepts of Topics, Partitions, and Replication in Kafka.
- How does message delivery work in Kafka (at least once, at most once, exactly once)?
- How does Kafka handle fault tolerance and high availability?

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Q5. Explain the architecture of Spark Streaming:

- What are DStreams and RDDs in Spark Streaming? How does Spark Streaming handle micro-batch processing?
- How does Structured Streaming differ from traditional Spark Streaming?
- How would you use Kafka as a source in Spark Streaming?
- How does Spark ensure fault tolerance during streaming (checkpointing, data recovery)?

Q6. Kafka vs. Spark Streaming:

- Compare Kafka Streams and Spark Structured Streaming for real-time processing.
- Which would you prefer for your ETL pipeline, and why?

4. Data Lake and Delta Lake

Q7. Explain the concept of a Data Lake and how Delta Lake works:

- What is the difference between a Data Lake and a Data Warehouse?
- How does Delta Lake provide ACID transactions on top of your Data Lake?
- How does Delta Lake handle schema evolution and time travel (data versioning)?
- What are the advantages of storing data in Parquet vs. CSV or JSON formats in Delta Lake?

Q8. How do you store streaming data in Delta Lake and handle schema evolution?

- If you have a continuous stream of data coming from Kafka into Delta Lake, how would you handle schema changes over time (e.g., adding a new column)?
- How would you handle data deduplication and late-arriving data in a real-time stream?

5. Spark Memory Management and Optimization

Q9. How would you handle memory management in Spark?

- What are the memory management settings in Spark (e.g., Spark executor memory, driver memory)?
- How do you handle Out of Memory (OOM) errors in Spark jobs, particularly in streaming jobs?
- How would you adjust parallelism and partitioning to optimize performance for large datasets in Spark?

Q10. What strategies would you use to optimize Spark jobs for both performance and cost on cloud platforms like AWS?

- How would you manage resource allocation for Spark jobs in a cloud environment (e.g., EC2 instances, S3 storage)?
- How do you manage the cost when running Spark jobs on AWS (e.g., using spot instances, autoscaling)?

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

6. Databricks and Cluster Management

Q11. Databricks:

- How does Databricks create clusters for running Spark jobs? Explain the different types of clusters (e.g., interactive clusters, job clusters).
- What is the difference between managed and unmanaged tables in Databricks? When would you use each type?
- Are Databricks clusters standalone or YARN mode clusters? Explain why or why not.
- How would you configure a Databricks Lakehouse architecture for ETL and real-time analytics?

7. Spark Resource Management and Optimization (Advanced)

Q12. Resource Management in Spark:

- You are given 10 worker machines with 100 GB RAM and 25 CPU cores. How would you determine the number of executors and the size of each executor?
- What happens if there is an Out of Memory (OOM) error in Spark, particularly in the Driver?

8. Java, File Handling, and OOPS

Q13. Java File Input Handling:

- Write a Java program using FileInputStream and BufferedReader to read data from a local file and print the output to the console.

Q14. Object-Oriented Programming (OOP) Concepts:

- Can you explain the concept of polymorphism and inheritance in Java with examples?
- What are the key differences between interfaces and abstract classes in Java?
- How do you handle dependency injection and inversion of control in a Spring-based system?

9. Sqoop

Q15. Sqoop Data Import:

- Suppose you need to import 5 tables from an external RDBMS (like MySQL) into Hadoop HDFS. Write the Sqoop command to import these tables.
- How would you handle data type mismatches when using Sqoop for importing data?

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

Round 5: Techno-Managerial Round

1. Introduction and Discussion on Past Projects

Q1. Introduce Yourself:

- Briefly summarize your background and experience in data engineering.
- Focus on your relevant technical skills, the tools you've worked with, and your accomplishments.

Q2. ZS Projects Discussion:

- Describe the ZS projects you worked on, your roles, and responsibilities.
- Focus on the tools, platforms, and technologies used (e.g., Databricks, Snowflake, etc.) and how they were applied in your projects.

Q3. Morgan Stanley Experience:

- Discuss the tech stacks and responsibilities you had while working at Morgan Stanley.
- Highlight your experience in migrating on-premise ETL pipelines to cloud services (Databricks, Azure, Snowflake).

2. ETL Design and Big Data Concepts

Q4. Data Lakehouse vs. Delta Lake vs. Data Warehouse:

- What is the difference between Data Lakehouse, Delta Lake, and a Data Warehouse?
- Explain how a Data Lakehouse combines the benefits of both a data lake and a data warehouse.

Q5. Data Compaction in Delta Lake:

- Have you worked with data compaction in Delta Lake?
- What is the Delta log and how does it differ from log files in Delta Lake?

Q6. Transaction History in Delta Lake:

- How does Delta Lake store the transaction history in S3 buckets?
- Explain the concept of ACID transactions in Delta Lake.

Q7. Hive Metastore Architecture:

- Explain the architecture and role of the Hive Metastore in a data pipeline.
- How does it work with Hive, Spark, and other big data tools?

Q8. Data Governance and Data Catalog:

- Write a step-by-step process on how you would design data governance policies.
- How would you implement a data catalog system from scratch to manage metadata and ensure compliance?

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

3. API Design and Database Interaction

Q9. Custom API Design:

- Design a Custom API that can query a backend server (running on Kubernetes or Docker) and return customer data such as the number of orders placed by a user based on their user ID.
- The database could be PostgreSQL or a Delta Lake table.
- What would be the architecture of this system and how would the API communicate with the database?

Q10. Presto and Data Catalog:

- How does Presto fetch data from a data catalog?
- Explain why Presto is particularly effective at handling complex queries over large datasets.

4. Data Modeling and SQL Queries

Q11. ETL Pipeline Data Model Design:

- Share your screen and design the data model for an ETL pipeline that ingests data from a database and loads it into a data warehouse like Snowflake.
- Define the schema for sample tables that could be used in your project at Morgan Stanley.

Q12. Normalization and SQL Query:

- Explain normalization in databases and its importance.
- Write an SQL query to handle SCD-1 (Slowly Changing Dimension Type 1) or SCD-3 (Slowly Changing Dimension Type 3).

5. Leadership and Team Management

Q13. Managing Team Challenges:

- How would you handle a situation where your team is struggling to keep up with the challenges of a project?
- How do you manage differences in skill levels or work pace within your team?

Q14. Tools for Alerting Mechanism (PagerDuty):

- Have you worked with PagerDuty or other alerting mechanisms?
- How would you set up an alert system to monitor your ETL pipeline or data infrastructure for failures or performance issues?

Q15. Job Scheduling and SLA Misses:

- How do you manage Job Scheduling and SLAs in an ETL pipeline?
- What would you do if a job misses its SLA (Service Level Agreement)? How would you handle such situations?

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

6. Achievements and Closing

Q16. Achievements:

- What are your **key achievements** in your career so far?
- The interviewer seemed particularly interested in your **University achievements** (e.g., being the **University topper** in your B.Tech course). Be prepared to elaborate on this.

Q17. Joining Time:

- How soon could you join **Meesho** if you are selected?
- This question typically assesses your availability and how soon you could contribute to the team.

Round 6: HR Interview – Detailed Question Set

The HR Interview was focused on understanding your personal background, experiences, motivations, and cultural fit for the organization. Here's a breakdown of the key questions asked:

1. Personal and Background Questions

Q1. Big Data Project Experience:

- Can you elaborate on your Big Data project experience?
- What technologies and platforms did you work with (e.g., Databricks, Snowflake, Delta Lake, Kafka)?
- Describe the challenges you faced in these projects and how you overcame them.

Q2. Hobbies and Interests:

- What are your hobbies or activities you enjoy outside of work?
- How do your hobbies align with your personal or professional growth?

Q3. Strengths & Weaknesses:

- What are your key strengths?
- Can you share an example of how you've leveraged these strengths in your previous roles?
- What are your weaknesses and how are you working to improve them?

Q4. Family Background:

- Tell me about your family background.
- How has your family influenced your personal and professional journey?

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar

2. Previous Interview Experiences

Q5. Interview Experiences at Other Companies:

- You mentioned that you have previously interviewed with companies like Amazon, Mastercard, Deutsche Bank, and Paytm. Can you share the interview experiences with these companies?
- What were the most challenging aspects of those interviews?
- How did you prepare for them and what did you learn from these experiences?

3. Motivations and Career Goals

Q6. Ultimate Goal in Life:

- What is your ultimate career goal or life goal?
- How do you plan to achieve this goal, and how does this role align with that vision?

4. Conclusion

Q7. Fitment for Meesho:

- Why do you think Meesho is the right fit for you?
- How do you envision your future with Meesho and contributing to the organization's goals?

Glassdoor Meesho Review –

https://www.glassdoor.co.in/Overview/Working-at-Meesho-EI_IE1503776.11,17.htm

Meesho Careers –

https://www.meesho.io/jobs

Subscribe to my YouTube Channel for Free Data Engineering Content –

https://www.youtube.com/@shubhamwadekar27

Connect with me here –

https://bento.me/shubhamwadekar

Checkout more Interview Preparation Material on –

https://topmate.io/shubham_wadekar

© Shubham Wadekar

For personal use only. Redistribution or resale is prohibited. © Shubham Wadekar