JP Morgan & Chase Data Engineer Interview Guide – Experienced

Overview of the Interview Process

JP Morgan's interview process is thorough and evaluates candidates across multiple domains. Here's a high-level breakdown of the rounds:

- **Round 1:** Screening Call
- **Round 2:** Core Java, Concurrency, and System Design
- **Round 3:** SQL, Python, and AWS
- **Round 4:** Onsite Interviews (Coding Challenges, System Design, Technology Discussion)
- **Round 5:** Executive Director Round

Round 1: Screening Call (30 Minutes)

This initial round was with a VP-level manager. It primarily focused on assessing my motivations, experience, and high-level technical knowledge.

Topics Covered:

- **Experience:** Discussed my work with cloud platforms, data pipelines, databases, and programming languages.
- **Motivation:** Why JP Morgan and what I hoped to achieve in the role.
- **Technical Questions:**
    - Versions of Java, wrapper classes, and data types.
    - My experience using tools like Terraform and Jenkins.
    - CI/CD pipelines and deployment strategies.

Tip: Be prepared to explain how your past experience aligns with the role and show enthusiasm for joining the company. Brush up on your basic Java concepts and DevOps tools.

Round 2: Core Java, Multi-threading, and Big Data Architecture (1 Hour 45 Minutes)

Section 1: Terraform, Jenkins, and Deployments

The interview began with discussions on **infrastructure as code** (IaC) and CI/CD pipelines. Here's a breakdown of the questions and my responses:

Questions:

1. **Write Terraform configurations for configuring an EC2 machine.**
    - I outlined a basic Terraform configuration that sets up an EC2 instance, specifying the provider, resource blocks, AMI ID, and instance type. I emphasized best practices like using variables and output blocks.
2. **Explain the Terraform lifecycle for deploying a new cluster on AWS.**
    - I explained the lifecycle, using a **CloudFormation** stack as an example for deploying a Presto cluster. I walked through Terraform commands (init, plan, apply) and discussed how state files are managed.
3. **Describe how you deploy code to a production environment using Jenkins.**
    - I detailed a CI/CD pipeline:
        - **Trigger:** Changes pushed to GitHub trigger a webhook to Jenkins.
        - **Jenkins Job:** The job pulls the code, runs build and test stages, and deploys to production if successful.
        - **Deployment:** Automated deployment using tools like Ansible or Kubernetes for containerized applications.

Section 2: Core Java Concepts

The next segment focused on Java fundamentals and core concepts.

Questions:

1. **When were lambda expressions introduced in Java?**
    - Lambda expressions were introduced in **Java 8** as part of functional programming support.
2. **What is the default value for float and Float in Java?**
    - The default value for the **float** primitive is 0.0f.
    - For the **Float** wrapper class, the default value is null.
3. **Write a Singleton class implementation.**
    - I wrote a thread-safe Singleton class using the **double-checked locking pattern**:

```java
public class Singleton {
    private static volatile Singleton instance;

    private Singleton() { }

    public static Singleton getInstance() {
        if (instance == null) {
            synchronized (Singleton.class) {
                if (instance == null) {
                    instance = new Singleton();
                }
            }
        }
        return instance;
    }
}
```

4. **Explain the internal working of a HashMap.**
    - I described how HashMap uses an array of buckets and a linked list or balanced tree (from Java 8 onwards) for collision handling. I explained hashing, resizing, and how keys are mapped to indices using the hash function.
5. **What happens if the run() method in a Thread class is not overridden?**
    - If the run() method is not overridden, the thread will execute the Thread class's default run() method, which does nothing. The custom logic needs to be in a subclass or a Runnable implementation.
6. **Difference between stubs and skeletons in RMI (Remote Method Invocation):**
    - A **stub** is a client-side proxy that communicates with the server.
    - A **skeleton** is a server-side component that listens for client requests and invokes the appropriate method.
7. **Write code using Java's concurrent API (forEach, forEachEntry, forEachKey):**
    - I demonstrated using a ConcurrentHashMap to perform concurrent operations safely:

```
ConcurrentHashMap<Integer, String> map = new ConcurrentHashMap<>();
map.put(1, "A");
map.put(2, "B");

map.forEach((key, value) -> System.out.println(key + ":" + value));
map.forEachEntry(1, entry -> System.out.println(entry.getKey() + "=" + entry.getValue()));
```

Section 3: Big Data Concepts (Hadoop, Spark, and Databricks)

The final part focused on my experience with big data tools like **Spark**, **Hadoop**, and **Databricks**.

Questions:

1. **Bloom Filters in Spark projects:**
    - I shared a **spell-checker** example where Bloom Filters efficiently determine if a word exists in a large dataset without loading the entire dictionary into memory. Bloom Filters reduce memory usage but allow for a small false-positive rate.
2. **Optimizing Spark jobs:**
    - Techniques included:
        - **Partitioning** and **coalescing**.
        - **Caching** frequently accessed data.
        - Using **broadcast variables** for small datasets.
        - Leveraging **data locality** to minimize network I/O.
3. **Data locality in Hadoop:**
    - Data locality ensures that computation happens on the same node where the data resides, reducing network overhead and improving performance.
4. **Databricks Job Cluster and SQL Endpoint:**
    - I discussed how **Photon** enhances performance in Databricks SQL Warehouses by leveraging vectorized query execution.
5. **Calculating Databricks costs:**
    - I explained **DBU (Databricks Unit)** and how to estimate costs based on cluster size, workload duration, and tier (Standard vs Premium).
6. **Handling custom data types in Spark:**
    - I suggested implementing **custom encoders** using Spark's Encoder API to work with non-standard types.
7. **Controlling mappers in MapReduce:**
    - Mapper count is influenced by input split size and can be controlled by adjusting the **mapreduce.input.fileinputformat.split.maxsize** property.
8. **Dataframe join and filtering scenario:**

Problem Statement:

- Given two dataframes (df1: id, name and df2: id, country, address, city, count), join them, filter for rows where country = 'Singapore', and pivot the output. Sort cities in descending order of population count.

Solution Approach:

- I explained using Spark's **DataFrame API** with a **window function** for ranking cities:

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions._

val joinedDF = df1.join(df2, "id").filter($"country" === "Singapore")

val windowSpec = Window.partitionBy("country").orderBy(desc("count"))

val resultDF = joinedDF.withColumn("rank", rank().over(windowSpec))
                       .filter($"rank" === 1)
                       .select("city", "count")
                       .orderBy(desc("count"))
```

Round 3: SQL, Python, AWS, and General CS Insights (90 mins)

SQL Challenges

SQL was a significant focus in this round. The questions tested both conceptual clarity and practical problem-solving abilities.

Example Problems:

1. **Finding Consecutive Numbers Appearing Thrice:**
    - I was asked to write a query that identifies numbers appearing at least three times consecutively without interruption. This required a combination of window functions and logic for detecting patterns within rows.
2. **Hierarchical Employee-Manager Query:**
    - The task involved a typical **self-referential hierarchy** scenario:
        - **Table structure**:
        - I needed to write a query that navigates the hierarchy, with the head of the company identified by employee_id = 1. This problem emphasized understanding of **recursive CTEs** or alternative hierarchical querying techniques.

Python Challenges

Python questions tested both core language knowledge and practical coding skills. The tasks were designed to measure proficiency in data manipulation and algorithmic thinking.

Key Problems:

1. **Character Replacement in a List:**
    - Write a function that replaces all characters in a list except for a given character. This was a straightforward problem, but the key was handling edge cases efficiently.

2. **Finding Complete String Pairs:**
    - This problem required identifying pairs of strings that, when concatenated, contain all 26 English alphabets. The task involved iterating over two sets of strings and checking completeness—a great test of string manipulation and set operations.
3. **Data Handling with Pandas:**
    a. I was asked to:
        i. Read data from three files into a Pandas DataFrame.
        ii. Perform various transformations, such as removing specific columns, filtering rows, and searching for strings within the data.
    b. This task emphasized the importance of data wrangling and familiarity with the Pandas library for efficient data manipulation.

General Computer Science Questions

One specific problem focused on **tree data structures**:

1. **Binary Search Tree to Skewed Tree Conversion:**
    - I was asked to convert a Binary Search Tree (BST) into a skewed tree in either increasing or decreasing order. This task required an understanding of in-order traversal and maintaining tree properties during transformations.

Round 4: Techno-Managerial Interview

This 45-minute session was with a VP from my potential team, focusing on how I would handle real-world data engineering challenges and my technical expertise.

Key Topics Covered:

- **Real-Time Data Processing:**
  - I discussed Spark Streaming and Structured Streaming, explaining their differences with a real-world example.
  - Highlighted latency differences, micro-batch vs. continuous processing, and when to choose each.
- **Data Consumption in E-Commerce:**
  - I shared an end-to-end approach:
    - Expose APIs → Capture event data → Send to Mixpanel → Forward to Kafka Streams.
    - From Kafka, run ETL pipelines on Databricks and dump the data into Delta Lake for querying.
- **Datahub Spline Integration:**
  - Mapped the data lineage tracking using Spark listeners with Datahub Spline to capture both upstream and downstream metadata.
  - Explained capturing lineage at runtime and how it integrates with downstream processes.
- **ETL Pipeline for Mixed Data Streams:**
  - I described designing an ETL pipeline for both Kafka events and RDBMS data using Kafka Connect for change data capture (CDC).
- **Delta Lake Upserts:**
  - Explained how MERGE INTO in Delta Lake supports upserts and shared best practices for handling late-arriving data.
- **AWS Cost Optimization:**
  - Provided strategies:
    - S3 Tiering to archive infrequent data.
    - Switching from on-demand to spot instances.
    - Using IAM policies to restrict unnecessary resources.

Round 5: Executive Director Interview

This 30-minute session with an Executive Director focused more on leadership, strategic thinking, and project insights.

Key Questions and My Responses:

- **Kubernetes Cluster Usage:**
  - Discussed how I used Kubernetes for managing containerized workloads in my previous projects, ensuring scalability and fault tolerance.
- **AWS Services Experience:**
  - Shared my hands-on experience with:
    - S3, Presto, EMR, Glue, Redshift, RDS, Load Balancers, and strategies for cost control and scaling.
- **Spark vs. Hadoop:**
  - Outlined differences:
    - Spark's in-memory processing vs. Hadoop's disk-based architecture.
    - Spark's better suitability for iterative algorithms and machine learning workloads.
- **Migrating On-Prem to AWS:**
  - Explained the step-by-step approach:
    - Data migration with AWS Snowball.
    - Rewriting ETL jobs using AWS Glue and EMR.
    - Leveraging AWS native tools for seamless transition.
- **Scenario-Based Questions:**
  - **Adding Non-Value Data:**
    - Suggested balancing business needs with performance, offering a sandbox report for non-critical data requests to mitigate impact.
  - **Handling Workload Prioritization:**
    - Shared a strategy for evaluating task urgency and keeping stakeholders informed. Explained how I would communicate updates to managers.

Glassdoor JPMorganChase Review –

https://www.glassdoor.co.in/Reviews/JPMorganChase-Reviews-E5224839.htm

JPMorganChase Careers –

https://www.jpmorganchase.com/careers

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