**Data Migration Specialist Technical Challenge**

You’re leading a high-volume migration from **Magento to Shopify Plus** with over **5M records**, a tight downtime window, and complex ERP/CRM integrations. This test assesses how you plan, clean, validate, and troubleshoot data migrations independently.

### **1. Migration Plan (Conceptual)**

**Task:** Outline your end-to-end plan for:

* Extracting data from Magento
* Reformatting for Shopify
* Loading into Shopify via API
* Syncing with ERP/CRM systems

#### Include tools, techniques, and key Python libraries or Excel/SQL strategies you'd use.

#### 

#### **Response:** Following is the comprehensive plan for migrating data from **Magento** to **Shopify Plus**:

#### **Discovery & Scoping**

|  |  |
| --- | --- |
| **What** | **Why** |
| Inventory all objects – products, variants, customers, orders, gift cards, discounts, meta fields, custom attributes. | Know data volumes, identify edge cases and any custom tables or fields. |
| Dependency mapping – Magento ↔ ERP ↔ CRM ↔ WMS, scheduled jobs, webhooks. | Clearly identify data ownership and determine the correct sequence for synchronization. |
| Downtime window negotiation (e.g., Fri 22:00 → Sat 04:00 UTC). | Critical for planning batch size and parallel execution strategies. |
| Success criteria – record counts ±0.1%, financial totals ±0.01%, SEO slugs preserved, no 5xx from Shopify API. | Provides clear and measurable objectives for migration success. |

#### **Environment & Tooling**

|  |  |
| --- | --- |
| **Layer** | **Stack / Tool** |
| Extraction | Python 3.11, MySQL connector Python, PyMongo, direct SQL dumps, Magento REST API for incremental changes. |
| Staging & Transform | PostgreSQL (RDS or Docker), dbt for repeatable SQL models, Pandas for data transformations, AWS S3 for file storage. |
| Validation | Great Expectations for automated data quality checks, pytest for unit tests, custom checksum scripts for validation. |
| Load | Shopify REST & GraphQL APIs, Bulk Operations GraphQL (large data uploads). |
| Scheduling / Orchestration | Prefect 2.0 or Apache Airflow for workflow automation, retries, and detailed logging. |
| ERP/CRM Sync | Native connectors if available; otherwise webhooks → AWS Lambda → ERP APIs; CDC tables for replayable sync. |
| Monitoring | Datadog/Grafana for real-time tracking, alerting via Slack for anomalies. |
| Rollback | Pre-migration Shopify export, RDS snapshots for staging database rollback points. |

#### **Detailed Workflow**

##### Pre-Cutover "Full" Migration (T - 7 days → T - 1 hours)

* **Extract:** Large tables pulled in manageable chunks via SQL, smaller datasets through Magento REST API.
* **Stage & Clean:** dbt transforms Magento's EAV schema into flat, Shopify-compatible structures; clean data using Pandas.
* **Enrich & Map:** Attribute mappings to Shopify product types, tags, and metafields. Generate SEO-friendly slugs and URL redirects.
* **Bulk Load:** Shopify sandbox bulk load using GraphQL Bulk Operation API.
* **QA:** Automated checks via Great Expectations, visual spot checks, SQL-based financial totals verification.

##### **Delta / Incremental Loads (T - 1 hr → T 0)**

* Capture incremental changes using timestamps and Magento webhooks.
* Execute streamlined ETL for incremental changes.

##### **Cutover (Downtime Window)**

* Magento maintenance mode activation; final data snapshot.
* Rapid delta uploads to Shopify with parallel bulk operations.
* DNS/CDN switch to Shopify; ERP/CRM synchronization triggered.

#### **ERP / CRM Synchronization**

* ID Mapping: Maintain cross-reference tables (Magento ↔ Shopify IDs).
* Inventory: Shopify webhooks trigger AWS Lambda functions updating ERP.
* Orders: Historical orders read-only; new orders sync in real-time via API.
* Customer Data: Scheduled nightly sync via Shopify GraphQL and ERP APIs.

#### **Validation & Sign-off**

* Automated scripts for record count, financial totals, and sample hashes comparison.
* Stakeholder-approved UAT checklist.
* Performance testing (500 simultaneous add-to-cart/checkouts).

#### **Rollback / Contingency**

* Bulk API errors >2% trigger pause and manual review.
* Delays trigger DNS rollback and Magento reactivation.
* Critical issues resolved via Shopify Store Import revert and immediate redeployment.

#### **Key Python / SQL / Excel Techniques**

* Pandas for efficient data manipulation; PyArrow for fast parquet handling.
* psycopg2 for PostgreSQL bulk data operations.
* SQL window functions for deduplication.
* Excel Power Query for flexible stakeholder data reviews.

#### **Summary / Conclusion**

This migration plan ensures a smooth transition from Magento to Shopify Plus by carefully extracting, transforming, and loading data while maintaining synchronization with ERP/CRM systems. The use of Python and relevant libraries provides flexibility and efficiency in handling large volumes of data.

### **2. Data Cleanup Plan (Practical Thinking)**

**Scenario:** You find duplicate customers, malformed SKUs, missing product descriptions, and blank metafields.

**Task:**

* List the **Python dependencies** you'd use
* Summarize what your **script would do** in 4–6 bullet points
* Write one **SQL query** that identifies duplicate customers. Use placeholder table header names. (use email or phone)

#### **Response:** Following is the **Data Cleanup Plan**

**Python Dependencies:** pandas, numpy, regex, phonenumbers, openpyxl/csv

**Script Actions:**

* Deduplicate customer records by email/phone.
* Correct SKU formats using regex.
* Populate missing product descriptions and metafields.
* Validate data types and Shopify constraints.
* Log and summarize changes made.

**SQL Duplicate Query:**

SELECT 'Email' AS DupField, email AS Identifier, COUNT(\*) AS Count

FROM customers

WHERE email IS NOT NULL AND email <> ''

GROUP BY email

HAVING COUNT (\*) > 1

UNION ALL

SELECT 'Phone', phone, COUNT (\*)

FROM customers

WHERE phone IS NOT NULL AND phone <> ''

GROUP BY phone

HAVING COUNT (\*) > 1;

### **3. Post-Migration QA & Debugging**

**Scenario:** After launch, customers report:

* Incomplete order history
* Mismatched inventory
* Price discrepancies

**Task:**

* Describe how you'd compare Shopify and Magento datasets for validation (tools/scripts)
* How would you trace and fix inventory sync issues?
* If product prices are wrong, how would you **roll back or patch prices fast**?

#### **Response:** Following are the details

* Compare datasets (Magento vs Shopify) using pandas DataFrames.
* Check detailed order histories, product listings, and customer data.
* Trace inventory mismatches via SKU-level comparisons; fix using targeted sync or manual updates.
* Correct pricing quickly via bulk CSV re-import or manual Shopify admin bulk edits.

### **4. High-Volume Optimization**

**Scenario:** You have 6 hours for cutover and face Shopify API rate limits.

**Task:**

* How would you optimize data uploads to stay within Shopify's API limits?
* Would you use Bulk APIs or parallel requests? Why?
* How would you log failed pushes and retry automatically?

#### **Response:** Following are the details

* Optimize uploads through batching and parallel requests within rate limits.
* Leverage Bulk Operations API primarily for scalability; use parallel requests sparingly for data unsupported by Bulk.
* Implement detailed logging and automatic retry mechanisms for robustness.

### 

### **5. Live Issue Response & Communication**

**Scenario:** A live store shows incorrect prices due to a migration issue. Customers are complaining.

**Task:**

* What’s your **immediate response plan**?
* Draft a short Slack message to internal stakeholders
* How would you restore pricing within 30 minutes?

#### **Response:** Following are the details

**Immediate Response:** Rapid assessment, temporary store lockdown if severe, quick fix deployment.

**Slack Communication:** "Hey team – we've identified incorrect prices on the live store. I'm actively correcting this and expect resolution within 30 mins. Updates to follow."

**Rapid Fix Strategy:** Bulk CSV upload for immediate price corrections, double-checking site, communication of resolution, documentation of lessons learned for future migration quality improvements

### 

### **6. My Migration**

**Task:** Share an example of a data migration or large dataset cleanup you’ve handled

* What was the problem?
* What tools and approacH did you use?
* What was the outcome?

#### **Response:** Following are the details

**Problem**

I recently worked on a research project involving the collection and analysis of clinical trial data from multiple online sources. The primary challenge was consolidating disparate, large-scale datasets that were inconsistent, incomplete, and scattered across various web resources. The data required significant cleaning, restructuring, and normalization to enable accurate, reliable analysis.

**Tools and Approach**

To handle this complex data migration and cleanup:

**Python Scripting**: I created automated scripts using Python, particularly leveraging libraries such as pandas, requests, BeautifulSoup, and regular expressions (re) to efficiently scrape, parse, and structure the data.

**Data Extraction and Integration**: I developed custom web scraping tools, exemplified by my GitHub project [WebMD Scraper](https://github.com/pranay-ratan/WebMD-Scraper/blob/main/webmdscraper_version2.py), which systematically retrieved structured information from web pages.

**Data Cleaning and Normalization**: The scripts incorporated methods to handle missing values, deduplicate records, standardize data formats (e.g., dates, numerical values), and ensure compliance with the required schema for further analysis.

**Validation and Verification**: Extensive validation checks using Python libraries like pandas ensured data integrity, correctness, and consistency post-migration.

**Outcome**

The comprehensive data migration and cleanup allowing to build a unified, high-quality dataset ready for advanced analytics. This significantly enhanced the accuracy of our analyses and facilitated meaningful insights into the clinical trials under study. By automating the extraction and cleaning processes, we saved considerable time and minimized manual errors, ultimately improving the overall efficiency and reliability of our research project.