**Data Migration Scripts, Logs, and Validation Reports**

**Project:** Odoo ERP Integration for Safaricom Telecom  
**Version:** 1.0  
**Date:** [Insert Date]

**A. Purpose**

To provide a structured framework for developing, executing, and documenting data migration processes from legacy telecom systems into the Odoo ERP, ensuring accuracy, traceability, and accountability.

**B. Data Migration Scripts**

**1. Script Types**

| **Script Type** | **Description** | **Technologies / Tools** |
| --- | --- | --- |
| Extraction Scripts | Extract data from legacy OSS/BSS, billing, CRM systems | Python, SQL, Shell scripts |
| Transformation Scripts | Cleanse, format, and map data to Odoo-compatible structures | Python (Pandas), SQL |
| Load Scripts | Insert transformed data into Odoo database | Odoo ORM APIs, PostgreSQL SQL |
| Validation Scripts | Verify data completeness and integrity post-load | Python, SQL validation queries |

**2. Best Practices**

* Modular script design for reusability and maintainability.
* Parameterize scripts for environment and dataset flexibility.
* Logging incorporated for each step to capture successes and failures.
* Version control using Git or equivalent.

**C. Migration Logs**

**1. Log Contents**

* Start and end timestamps of each migration batch.
* Number of records processed, successfully migrated, and failed.
* Detailed error messages with context (record ID, field name).
* System resource usage and performance metrics.

**2. Log Management**

* Centralized log storage using ELK stack, Splunk, or cloud logging services.
* Automated alerts triggered on failures or threshold breaches.
* Regular review cycles during migration testing and production runs.

**D. Validation Reports**

**1. Report Types**

| **Report Type** | **Description** | **Frequency** | **Owner** |
| --- | --- | --- | --- |
| Record Count Reconciliation | Compare source vs target record counts | After each migration batch | Data Migration Lead |
| Data Quality Report | Identify missing, duplicate, or invalid data | After data cleansing and migration | QA Team |
| Referential Integrity Report | Verify relationships between tables/entities | After data load | QA Team |
| Performance Summary Report | Assess migration execution time and resource usage | After each migration run | Project Manager |

**2. Report Format**

* Summary dashboard with KPIs and trends.
* Detailed listings of discrepancies and remediation status.
* Recommendations for corrective actions.
* Sign-off section for validation by business and technical leads.

**E. Sample Script Snippet (Python)**

import logging

import psycopg2

# Setup logging

logging.basicConfig(filename='migration.log', level=logging.INFO, format='%(asctime)s %(levelname)s:%(message)s')

def extract\_data():

try:

# Connect to legacy database

conn = psycopg2.connect("dbname=legacy\_db user=admin password=secret")

cursor = conn.cursor()

cursor.execute("SELECT customer\_id, name, balance FROM customers WHERE active = TRUE")

data = cursor.fetchall()

logging.info(f"Extracted {len(data)} records")

return data

except Exception as e:

logging.error(f"Error during extraction: {e}")

raise

def transform\_data(records):

transformed = []

for r in records:

# Example transformation: Normalize names to uppercase

transformed.append((r[0], r[1].upper(), r[2]))

logging.info(f"Transformed {len(transformed)} records")

return transformed

def load\_data(records):

try:

conn = psycopg2.connect("dbname=odoo\_db user=odoo password=secret")

cursor = conn.cursor()

for rec in records:

cursor.execute("INSERT INTO res\_partner (id, name, credit) VALUES (%s, %s, %s) ON CONFLICT (id) DO UPDATE SET name = EXCLUDED.name, credit = EXCLUDED.credit", rec)

conn.commit()

logging.info(f"Loaded {len(records)} records into Odoo")

except Exception as e:

logging.error(f"Error during loading: {e}")

conn.rollback()

raise

if \_\_name\_\_ == "\_\_main\_\_":

data = extract\_data()

transformed\_data = transform\_data(data)

load\_data(transformed\_data)