Edition 2026:1

Monday Cup

By Suzanne Medes

January 12, 2026

7 minutes of Best Practices and Solutions.

With AI applications

01.12.26 Problem:Tracking too much detail

When you break down tasks, avoid going so deep that your workplan becomes a task list. Those are two separate tools.

A WBS, workplan, or work breakdown structure exists best to categorize deliverables and break them down “enough” so they can be 1) linked to goals, and 2) applied to resources and a time line.

A best practice workplan covers the overall scope of the project and should only contain enough detail to assign the work, move the work forward, and measure the progress *and* shifts as the work is delivered/billed. A WBS tis not a checklist.

*KEY TERM: Deliverable – A completed body of work in the form of a tangible capability or document that can be handed over to a client (internal/external) and move the goals of the project forward.

Solution: Turn every tangible goal on a project into a list of deliverables, then *break each deliverable down only to the point where an item can be assigned.*

Method: The Rule of 3

Topics: Project Management, WBS, Agile, Sprints, Jira, Strategy, Planning, Success Rate, ROI

AI APPLICATION: Automation of workflows, backlogs, reporting, dashboards, metrics

EXAMPLE:

NEXT PAGE

AI EDW AUTOMATION PROJECT
AI-Enabled Data Cleaning and Automation for Enterprise Data Warehouse

EXAMPLES: Two methods of breaking down project tasks on an AI EDW project: one weak, one strong.

By Suzanne Medes, WORD MECCA Monday, January 12th 2026

Project Summary

AI-Enabled Data Cleaning and Automation for Enterprise Data Warehouse (EDW) Project Overview: This initiative focuses on enhancing the company's backend EDW by cleaning and preparing data for AI-driven automation and real-time synchronicity. The goal is to address common data quality issues such as duplicates, inconsistencies, and incompleteness, enabling seamless integration with AI models to automate processes, improve decision-making, and ensure data reliability across systems.

Objectives

- Assess current EDW data quality and define cleaning priorities.
- Execute data cleansing to create a robust, standardized dataset.
- Implement AI tools for ongoing automation and synchronization, reducing manual interventions and operational inefficiencies.

Scope

The example project encompasses data profiling, cleaning workflows, AI model development, and integration testing. It excludes hardware upgrades or non-EDW data sources. Estimated duration: 6-9 months, with a cross-functional team including data engineers, AI specialists, and IT stakeholders. Expected Outcomes: A cleaned EDW with AI-automated maintenance, leading to 30-50% faster data processing, reduced errors, and enhanced analytics capabilities for business units.

WBS EXAMPLES

Weak Task Breakdown Structure

This example is overly detailed and disorganized, resembling an exhaustive to-do list rather than a high-level project work plan. It includes unnecessary sub-tasks, procedural minutiae, timelines, dependencies, and resource specifics that belong in detailed execution plans/tools used by the TEAM to track their own backlog, or a billing system if contract language requires that detail for invoicing.

Those details are not suited for overall scope roadmaps/managing projects; the granularity makes it hard to track at a managerial level, overwhelms stakeholders, and turns the plan into a micromanagement document instead of a strategic overview and work delivery tool. **WASTED TIME & FOCUS.** *Admin heavy.*

- Phase 1: Initial Data Assessment and Planning
    - Sub-phase 1.1: Gather requirements from stakeholders
        - Task 1.1.1: Schedule meetings with IT team, data analysts, and business units; prepare agenda including discussion on current EDW issues like duplicate records, inconsistent formats, and missing values; send invites via email/calendar; RSVPs.
        - Task 1.1.2: Conduct interviews; record notes on data quality problems such as outdated schemas, integration gaps with legacy systems; use tools like Zoom for virtual sessions or in-person if possible; transcribe recordings using Otter.ai.
        - Task 1.1.3: Compile requirements document; include sections on data volume (e.g., 500TB), variety (structured/unstructured), velocity; prioritize based on business impact using a scoring system from 1-10; get approvals from managers.
    - Sub-phase 1.2: Audit existing EDW data
        - Task 1.2.1: Access EDW using SQL queries to sample data; run SELECT statements on key tables like customer data, sales history; identify anomalies such as NULL values in 15% of rows or mismatched data types.
        - Task 1.2.2: Use data profiling tools like Talend or Informatica; generate reports on metrics including completeness (85%), accuracy (92%), timeliness; export to PDF and share via SharePoint.
        - Task 1.2.3: Document findings in a spreadsheet; categorize issues by severity (high: data loss risks; medium: performance drags; low: cosmetic); include screenshots of query results and tool outputs.
        - Task 1.2.4: Estimate cleaning effort; calculate hours based on data size, e.g., 2 hours per GB for manual review.
    - Sub-phase 1.3: Define project scope and timeline
        - Etc. (continues with even more details...)

- Phase 2: Data Cleaning and Preparation for AI
  - Sub-phase 2.1: Develop cleaning scripts and processes
    - Task 2.1.1: Install Python libraries like Pandas, NumPy on development environment; write functions for deduplication using fuzzy matching algorithms; test on sample dataset of 10,000 rows.
    - Task 2.1.2: Handle missing data; implement imputation methods such as mean/median fill or KNN; validate against original data to ensure no bias introduction.
    - Task 2.1.3: Standardize formats; convert dates to ISO, currencies to USD; write regex patterns for string cleaning; run batch processes overnight to avoid downtime.
    - Task 2.1.4: Integrate with ETL tools like Apache NiFi; set up pipelines for real-time cleaning.
    - And so on…

This pattern continues across multiple phases with excessive layers, making it impractical for high-level tracking, resource planning, dependency alignment, milestone expectations, and overall delivery.

SOLUTION:

Strong Task Breakdown Structure

The Rule of 3:

This next example follows a clean, high-level structure: 3 main work sections, each with exactly 3 deliverables, and each deliverable broken into exactly 3 concise tasks. Tasks are actionable, assignable to resources (e.g., a data engineer or project manager), and trackable via milestones or status updates. This keeps the plan strategic, focused on scope management, and suitable for an IT project's overall work plan without drowning in execution details.

STREAMLINED & EFFECTIVE. *Useful. AI-Ready. Predictable. Measurable. Adjustable. Billable.*

Work Section	Deliverable	Tasks
1. Data Assessment	1.1 Requirements Report	• Collect stakeholder input on EDW issues • Profile data for quality metrics • Prioritize cleaning needs
	1.2 Audit Summary	• Sample EDW datasets • Identify key anomalies • Estimate cleaning scope
	1.3 Project Roadmap	• Define success criteria • Outline timelines • Assign initial roles
2. Data Cleaning	2.1 Cleaning Framework	• Develop deduplication scripts • Implement data standardization rules • Set up validation checks
	2.2 Cleaned Dataset	• Execute batch cleaning processes • Handle missing/inconsistent data • Verify data integrity
	2.3 Automation Pipeline	• Integrate ETL tools • Enable real-time syncing • Test pipeline performance
3. AI Implementation	3.1 AI Model Prototype	• Select AI algorithms for automation • Train model on cleaned data • Evaluate initial accuracy
	3.2 Integration Plan	• Design EDW-AI interfaces • Configure synchronicity features • Document deployment steps
	3.3 Testing and Rollout	• Conduct end-to-end tests • Gather user feedback • Finalize production rollout

I'm Suzanne Medes, Technologist & Writer and this is your **Monday Cup** of tips to start your business week armed with information that matters.

FOLLOW ME for one of a kind content, videos, eBooks, toolkits and more!

I have 30 years in the IT game and a powerful strategy for the next 30. In the first chapter of my career I knew enough to be dangerous. This time, *I am dangerous*.

Use my knowledge to gain a competitive advantage during the AI Shift.

BOOKMARK SuzanneMedes.com if you want more.

Top it off for Techies

Deduplication Scripts

Deduplication scripts are automated routines, often written in programming languages like Python, SQL, or R that identify and remove duplicate records from a dataset. In the context of data management, especially for projects like the AI-Enabled Data Warehouse (EDW) described in this document example, deduplication is crucial for ensuring data quality and reliability.

How Deduplication Scripts Work

- ✓ **Scan Data:** The script reviews the dataset for records that are identical or nearly identical (using techniques like fuzzy matching)
- ✓ **Identify Duplicates:** It flags records that match on key fields (e.g. customer ID, email address, transaction)
- ✓ **Remove or Merge:** The script either deletes the duplicates or merges them, depending on business rules
- ✓ **Report:** Often, the script generates a summary of what was removed or merged for auditing purposes
- ✓ **Why Are Deduplication Scripts Important?**
  - They help maintain a clean, reliable dataset, essential for accurate analytics/AI model training
  - They reduce storage costs and improve processing speed
  - They prevent errors in reporting and decision-making caused by duplicate data

Example in this Project Context

Under the "**Data Cleaning**" section, one of the tasks is to "Develop deduplication scripts." This means your team would create automated routines to clean the EDW by removing duplicate entries, ensuring that subsequent AI automation works with high-quality, unique data.

This sample script generated by Grok 4.0 uses the popular pandas library to remove duplicate rows from a dataset (for example, a CSV file containing EDW records):

```
import pandas as pd

# Load your data (replace 'your_data.csv' with your actual file)
df = pd.read_csv('your_data.csv')

# Display the number of rows before deduplication
print(f"Rows before deduplication: {len(df)}")

# Remove duplicate rows based on all columns
df_deduped = df.drop_duplicates()

# If you want to deduplicate based on specific columns (e.g.,
'customer_id' and 'email'):
# df_deduped = df.drop_duplicates(subset=['customer_id',
'email'])

# Display the number of rows after deduplication
print(f"Rows after deduplication: {len(df_deduped)}")

# Save the cleaned data to a new file
df_deduped.to_csv('your_data_deduped.csv', index=False)
```

- ➢ **Reads** your dataset into a Data Frame
- ➢ **Removes** duplicate rows (either across all columns or just specific ones)
- ➢ **Saves** the deduplicated data for further processing