Intern Training Plan (4 Weeks) – Based on Data Engineering

Prerequisites

- Understanding of programming Python & SQL
- Fundamental database concepts
- AWS Free Tier account created

NYC yellow Trips Taxi Data

- Yellow Trips Data Dictionary
- Taxi Zone Lookup Table (CSV)
- Yellow Taxi Trip Records (PARQUET)

- **Day 4: Master Data Management - Foundation & Strategy**
  - MDM Concepts & Business Alignment
    - What is MDM and why it matters to the business (not just IT)
    - Master data domains: Customer, Product, Location, Vendor
      - For each domain: Who cares? What decisions depend on it? What breaks if it's wrong?
    - MDM implementation styles with real-world scenarios**:**
      - Registry: We need a single view but systems must stay independent
      - Consolidation: We need read-only golden records for analytics
      - Coexistence: We need bidirectional sync with operational systems
      - Centralized: Master data is created and managed in one system
  - MDM Governance Model
    - MDM Roles & Responsibilities:
      - Data Owner: Accountable for domain (usually business VP)
      - Data Steward: Day-to-day quality & change management (usually business analyst)
      - Data Custodian: Technical implementation (data engineer - that's you!)
      - Data Consumer: Uses data (analysts, apps, ML models)
    - MDM Operating Model:
      - How are golden records created? (Application form? Automated match?)
      - Who approves changes? (Steward? Owner? Automated rules?)
      - What's the escalation path for conflicts?
      - How are exceptions handled?
  - Golden records and data consolidation
  - Match and merge strategies with governance implications
  - Data quality dimensions: accuracy, completeness, consistency, timeliness, validity, uniqueness
  - MDM components with governance lens:
    - Repository (storage)
    - Integration layer (publish/subscribe)

        - Quality engine (rules as governance artifacts)
        - Workflow engine (approval workflows)
        - Audit & lineage (regulatory compliance)
    - Hands-on:
        - Identify master data domains in NYC taxi dataset (zones, vendors, rate codes)
        - For each domain, document:
            - Business owner (simulated)
            - Business justification (why is this master data?)
            - Update frequency and change triggers
            - Quality requirements and thresholds
            - Approval workflow (who can create/update/retire?)
        - Create simple matching algorithm for duplicate detection
        - Design golden record strategy for taxi zones with survivorship rules documented
        - Build master data tables in RDS with audit columns (created_by, updated_by, approved_by, version)

- **Day 5: CI/CD & Governance as Code**
    - Technical Content
        - Git workflows: branching, PRs, code reviews
        - CI/CD concepts and benefits
        - Git Actions for data pipelines
        - Automated testing in CI/CD
        - Terraform basics: providers, resources, state
        - CloudFormation overview
        - Environment management (dev/staging/prod)
    - Governance-as-Code
        - **Version-controlled governance artifacts**:
            - Data quality rules (Great Expectations YAML)
            - Match/merge logic (fuzzy matching thresholds)
            - Access policies (IAM policies, bucket policies)
            - Metadata schemas (JSON Schema)
            - Approval workflows (YAML configs)
    - **Hands-on:**
        - Create Terraform templates for S3, Lambda, RDS
        - Deploy infrastructure through CI/CD
        - Create version-controlled governance configs:
            - Data quality rules file (YAML)
            - Access policy templates (JSON)
            - Metadata schema definitions
            - Deployment pipeline with approval gates for production

Week 1 Deliverable

- Document "Week 1 Learnings" in wiki
- Data processing application with CI/CD
- Create RDS database with master data schema including governance metadata columns
- Complete MDM governance framework document with operating model
- **Governance Charter document**:
    - Roles and responsibilities (RACI)
    - Decision rights framework
    - Escalation paths
    - MDM domains with business owners identified

Week 2: Data Lakes, ETL & Advanced MDM

Goal : Master AWS S3 and storage strategies, understand data lake architecture with governance zones, learn ETL concepts with quality gates, and implement governed MDM.

- **Day 6: Data Lake Concepts & Modern Storage**
    - Data lake vs warehouse vs data mart
    - Architecture patterns: raw/processed/curated zones
    - AWS Lake Formation basics
    - Metadata management and cataloging
    - Master data layer in data lakes
    - Parquet, ORC, Avro comparison
    - Delta Lake: ACID transactions, time travel, schema evolution
    - Apache Iceberg & Hudi overview
    - Lakehouse architecture
    - Master data zone in data lake (separate from raw/processed/curated)
    - **Governance zones**:
        - Landing/Raw: No governance (original source state)
        - Validated: Basic quality checks passed
        - Curated: Business rules applied, enriched with master data
        - Master: Golden records only, strictly governed
        - Archive: Compliance retention, immutable
    - Access patterns by zone: Different governance for different zones
    - Data lineage as governance requirement: Where did this come from? How was it transformed?
    - **Hands-on Lab**:
        - Design data lake architecture with Draw.io showing governance boundaries
        - Implement data lake zones in S3 with appropriate tagging and access controls
        - Convert data to Delta Lake format

        - Use Delta Lake time travel features for audit trail compliance

- **Day 7: Introduction to Apache Spark and AWS Glue**
    - Spark is, its lazy evaluation architecture, and its main components
    - AWS Glue Data Catalog - Central metadata repository for all your data and integrations
    - Glue Job types: Spark (standard), Python Shell, Streaming ETL with DPUs (Data Processing Units) ,Built-in transformations and data quality rules
    - Glue Crawlers Configure data sources (S3, RDS, etc.)**,** Schedule crawlers to run automatically**,** Handle schema evolution and partitioning
    - Glue Data Catalog as governance tool:
        - Not just technical metadata (schema, partitions)
        - Business metadata (definitions, ownership)
        - Operational metadata (quality scores, lineage)
    - Data discovery for governance: How do users find trustworthy data?

    - **Hands-on:**
        - Write PySpark scripts for NYC taxi transformations
        - Implement master data lookups/enrichment reference data validation
        - Optimize Spark jobs
        - Enhance Glue Catalog with governance metadata
            - Add custom properties: data_owner, domain, classification
            - Create crawler with metadata extraction rules
            - Document data lineage in catalog

- **Day 8: AWS Glue & Data Quality**
    - Glue Data Catalog setup , Glue Crawlers: schema discovery, scheduling
    - Glue ETL jobs: visual and script-based
    - AWS Glue Data Quality rules
        - Great Expectations framework
        - Data profiling and validation
        - Quality monitoring and alerting
    - Quality rules as governance artifacts:
        - Who defines quality rules? (Stewards + Owners)
        - How are rules versioned and approved?
        - What happens when quality fails? (Block? Alert? Manual review?)
    - MDM in ETL:
        - Reference data validation
        - Master data enrichment
        - Orphan detection
        - SCD implementation in Glue
    - Quality dimensions mapped to business impact:

        - Completeness → Can we fulfill orders?
        - Accuracy → Are we billing correctly?
        - Timeliness → Are decisions based on current data?
        - Consistency → Do systems agree on customer address?

    - **Hands-on:**
        - Create Glue crawlers and catalog
        - Build ETL job: CSV to Parquet with quality gates
        - Implement data quality checks with business-rule documentation
        - Validate transaction data against master tables (referential integrity)
        - Create quality scorecard with governance context:
            - Quality dimension → Business metric → Owner → Threshold → Action on failure

- **Day 9-10: Advanced MDM - Matching, Deduplication & Lifecycle Governance**
    - Data profiling techniques
    - Fuzzy matching algorithms: Levenshtein, Jaro-Winkler
    - Probabilistic matching
    - ML-based matching
    - Survivorship rules
    - Master Data Lifecycle States:
        - Proposed: New record submitted, awaiting review
        - Active: Approved golden record, in use
        - Deprecated: Marked for retirement, still quarriable
        - Retired: No longer valid, archived for compliance
    - Change Management Process:
        - Who can propose new master data?
        - What approval workflow applies? (Auto-approve? Steward review? Owner sign-off?)
        - How are conflicting updates resolved? (Last write wins? Steward arbitration? Automated rules?)
        - How long is change history retained?
    - Match Confidence & Governance:
        - High confidence (>95%): Auto-merge with audit log
        - Medium confidence (80-95%): Flag for steward review
        - Low confidence (<80%): Require manual resolution
    - Document match logic as versioned rules (governance artifact)
    - Change Data Capture (CDC) for master data with audit trail
- **Hands-on:**
    - Implement fuzzy matching with Python (fuzzywuzzy, recordlinkage)

- Build deduplication pipeline for vendors confidence scores and steward review queue
- Create data quality scorecard with business impact mapping

Week 2 Deliverable

- Document "Week 2 Learnings"
- Build data lake with Delta Lake format and **governance zones**
- Implement Glue ETL with data quality checks **tied to business rules**
- Deploy MDM with:
  - Matching/deduplication engine with confidence thresholds
  - Lifecycle state management
  - Role-based access control
  - Audit trail and change history
- **Governance Dashboard**:
  - Quality metrics by domain
  - Master data approval queue
  - Orphan transactions report
  - Steward activity log

Week 3: Orchestration, Transformation & Data Warehousing

Goal : Master AWS Glue ETL jobs, Learn Apache Spark fundamentals , Understand data transformation patterns

- **Day 11 : Data Pipeline Orchestration & Monitoring**
  - AWS Step Functions, Event Bridge, workflow orchestration, CloudWatch, logging, cost optimization
  - Review AWS Well-Architected Framework for data
  - cost optimization techniques and logging strategies
  - Build a complete workflow pipeline
  - **Audit trail**: Every pipeline run logged with:
    - Who triggered it? (User, schedule, event)
    - What data was processed? (Lineage)
    - What quality checks passed/failed?
    - What master data was used? (Version snapshot)
  - **Hands-On Lab**
    - Create Step Functions state machines to Orchestrate multiple AWS services (Glue, lambda etc.)
    - Implement error handling and retries & Set up CloudWatch alarms and dashboards
    - Add governance checkpoints:
      - Quality validation step (fail pipeline if below threshold)
      - Master data freshness check (alert if reference data stale)
      - Approval notification (SNS to steward for manual review)
      - Audit log publication (to dedicated audit table)

- **Day 12-13: Advanced SQL Transformations with Governance & Version Control**
  - Complex SQL patterns for data transformations
  - CTEs (Common Table Expressions) for modularity
  - SQL functions and stored procedures
  - Materialized views for performance
  - SQL testing strategies
  - Version controlling SQL scripts
  - Data validation patterns in SQL
  - Implementing data quality checks
  - Creating reusable SQL quality functions
  - Scheduling SQL transformations with Glue
  - Documenting SQL transformations
  - **SQL transformation metadata**:
    - Author: Who wrote this transformation?
    - Owner: Who is accountable for business logic?
    - Purpose: What business question does this answer?
    - Dependencies: What master data is required?
    - Quality expectations: What output quality is guaranteed?
  - Hands-on:
    - Create modular SQL transformation scripts
    - Build data quality validation queries
    - Implement SQL-based tests
    - Create documentation for transformations
    - Implement SQL workflows in Glue
    - Version control SQL scripts in Git with governance metadata in headers

- **Day 14-15: Slowly Changing Dimensions & Master Data Versioning**
  - Technical Content
    - SCD Types 0-6 detailed
    - Temporal tables and bitemporal modeling
    - Surrogate keys vs natural keys
    - Implementation patterns
    - Version control for master data
    - Audit trails and change tracking
    - Point-in-time queries
    - Rollback strategies
  - Why SCD matters for governance:
    - Regulatory compliance: "Show me customer address on date of transaction" (GDPR, audit)
    - Dispute resolution: "What was product price when order placed?" (legal)
    - Impact analysis: "How many customers affected by this vendor change?" (risk)
    - Rollback capability: "Undo this batch of incorrect updates" (data quality)
  - SCD Type Selection as Governance Decision:

        - Type 1 (overwrite): For reference data where history not needed (e.g., typo fixes)
        - Type 2 (history tracking): For regulated data where history mandatory (e.g., customer segments)
        - Type 3 (limited history): For business-driven "current vs prior" (e.g., territory assignments)
    - Temporal queries for governance use cases:
        - Point-in-time reporting for compliance
        - "As of" joins for accurate historical analysis
        - Change frequency analytics for stewardship
    - Master Data Versioning
        - Version metadata for golden records:
            - version_number, effective_from, effective_to
            - created_by, approved_by, approval_date
            - change_reason, change_source
            - is_current_flag
        - Versioning policies:
            - How long is history retained? (Regulatory requirement vs operational need)
            - When are versions purged? (Never? After 7 years?)
            - How is "truth" determined when conflicts exist?
    - Hands-on:
        - Implement SCD Type 2 in PostgreSQL RDS with metadata columns
        - Write stored procedures for SCD operations
        - Create PySpark script for SCD with Delta Lake leveraging time travel for rollback
        - Implement in SQL with version control
        - Create version management procedures:
        - approve_version(record_id, approver, reason)
        - rollback_version(record_id, target_version, reason)
        - audit_version_history(record_id, date_range)

Week 3 Deliverable

- Document "Week 3 Learnings"
- Create end-to-end orchestrated pipeline with Step Functions including governance gates
- Implement SQL transformations with:
    - Quality tests
    - Version control
    - Governance metadata documentation
    - Steward approval workflow simulation
- Governance Compliance Report:
    - All transformations have documented owners
    - All master data has versioning enabled
    - All quality rules are version controlled

- Audit trail demonstrates complete lineage

Week 4 – Data Warehousing, Analytics & Enterprise Governance

Goal: Learn data warehouse design techniques with dimensional modeling, understand MDM's role in analytics, implement consumption governance, and demonstrate production-ready system

- **Day 16: Amazon Redshift & Dimensional Modeling**
  - Technical Content
    - Redshift clusters, nodes, slices
    - Distribution styles: KEY, ALL, EVEN, AUTO
    - Sort keys and compression
    - COPY commands for bulk loading
    - Redshift Spectrum for S3
    - Star schema and snowflake schema design
    - Fact and dimension tables
    - Conformed dimensions
    - Aggregate tables
  - MDM vs Analytics Modeling
    - Golden Record (MDM): Single source of truth, operational system of record
    - Dimension (Analytics): Denormalized, SCD-tracked, optimized for queries
    - Relationship: Dimensions are derived from golden records
    - Conformed Dimensions as Governance Artifacts:
      - Shared across data marts to ensure consistent definitions
      - Published by MDM team, consumed by analytics teams
  - Master Data Metrics for Analytics
    - Beyond data quality, track MDM effectiveness:
      - Duplicate rate: % of potential duplicates detected
      - Match confidence distribution: Are stewards spending time on low confidence matches?
      - Orphan rate: % transactions referencing non-existent master data
      - Stewardship velocity: Average time from proposed → approved
      - Golden record completeness: % required attributes populated
      - Cross-reference accuracy: Do external IDs map correctly?
  - Hands-on:
    - Provision Redshift cluster
    - Design dimensional model for NYC taxi clear MDM sourcing
      - zone_dim derived from taxi_zones golden records
      - vendor_dim derived from vendor master data
      - Document the derivation logic as governance artifact
    - Load data from S3 using COPY
    - Optimize with distribution/sort keys

        - Run analytical queries

- **Day 17 : Athena & Serverless Analytics**
    - Technical Content
        - Serverless querying with Presto/Trino
        - Creating external tables
        - Partitioning strategies
        - Query optimization and cost management
        - Integration with Glue Catalog
        - Query performance tuning
        - Columnar format optimization
        - AWS Quick Sight for visualization
        - Dashboard design best practices
    - **Self-service with guardrails**:
        - Catalog with certified datasets (steward-approved)
        - Pre-built queries/views (prevent accidental PII exposure)
        - Query cost limits (prevent runaway queries)
    - Hands-on:
        - Deploy dimensional model in Redshift with SCD Type 2, and create analytics dashboard in Quick Sight with governance indicators
            - Quality score indicators ("99.2% complete")
            - Ownership labels ("Owned by Operations Team")

- **Day 18 : Monitoring, Observability & Governance Metrics**
    - Technical Content
        - CloudWatch: logs, metrics, alarms, dashboards
        - CloudTrail for audit logging
        - AWS X-Ray for distributed tracing
        - Data lineage tracking
        - Data observability best practices
        - Incident response for data issues
        - Query optimization and EXPLAIN plans
        - Spark optimization: partitioning, broadcast joins, AQE
        - Storage optimization: file sizing, compaction, Z-ordering
        - Troubleshooting: OOM errors, data skew, slow queries
    - Governance Observability
        - Technical monitoring (traditional):
            - Pipeline success/failure rates
            - Query performance
            - Storage costs
        - Governance monitoring (new focus):
            - Data quality trends: Are we improving or degrading?
            - Stewardship activity: Are proposed changes being reviewed promptly?

        - Governance incident response:
            - Data breach detection (unusual access patterns)
            - Quality degradation alerts (sudden drop below threshold)
            - Orphan surge alerts (spike in transactions without master data)
    - Hands-on:
        - Set up CloudWatch dashboards with technical and governance metrics
        - Create alarms for pipeline failures and governance violations
        - Implement data lineage visualization

- **Day 19 : Security, Compliance & End-to-End Governance**
    - Technical Content
        - AWS KMS and Secrets Manager
        - IAM policies and least privilege
        - PII detection and masking (AWS Macie)
    - Complete governance framework
        - Multi-Domain MDM Architecture
            - Centralized governance with federated stewardship
            - Master data hub pattern (publish golden records to consumers)
            - Event-driven updates (CDC from MDM to downstream)
        - MDM Tools & Patterns
            - AWS-native solutions (Glue, Step Functions, RDS, API Gateway)
        - Governance Maturity Assessment
    - Hands-on:
        - Implement encryption for S3 and RDS key rotation policies
        - Configure VPC with private subnets for **data isolation**
        - Set up IAM with least privilege to governance roles
        - Implement PII masking with column-level security policies
        - Design multi-domain MDM architecture with clear ownership boundaries

- **Day 20 (Final Demo Day)**
    - Architecture Overview
        - Present comprehensive architecture diagram showing
        - Ingestion: batch (S3)
        - Storage: S3 data lake with zones (raw/processed/curated/master)
        - Processing: Spark/Glue/Lambda with Delta Lake
        - Transformation: SQL scripts with version control
        - Serving: Redshift dimensional model + Athena
        - Orchestration: AWS Step functions
        - Master Data Management layer
        - Monitoring: CloudWatch dashboards
        - CI/CD: Cloud formation pipeline
    - Live demo of working system (20 min):

- Master data operations
- Data quality monitoring
- Analytics dashboard
- **Working Demonstrations:**
  - Batch ETL with AWS Step functions orchestration
  - SQL transformations with data quality tests and version control
  - Master Data Management:
    - Deduplication and matching engine
    - SCD Type 2 implementation
    - Data quality dashboard
  - Analytics dashboard in QuickSight
  - CI/CD deployment in action
  - Monitoring and alerting