🔥 6-Month Data Engineering Mastery Plan

Duration: 24 Weeks

🎯 Program Goal

Master Firestore, Firebase Security, GCP, ETL, BigQuery, Python automation, and technical writing at a level suitable for a Data Engineering role.

MONTH 1 — Firestore, Data Modeling & Security Mastery

WEEK 1 — Firestore Foundations

Learning Objectives:

- Understand Firestore architecture (NoSQL document model).
- Learn core components: documents, collections, subcollections.
- Understand scaling behavior of Firestore (reads, writes, listener costs).
- Explore Firestore console and SDK basics.

Reading Plan: Firestore overview, Data model, Structure data, Get data (queries basics).
Hands-on Work:

- Create 3 sample collections: Users, Products, Orders.
- Perform CRUD using Console + SDK.
- Write basic queries: filters, ordering, limits.
- Track read/write counts for each query.
- Experiment with nested data vs referenced data.

Deliverables:

- 2-page summary of Firestore basics.
- Collection + data structure diagrams (draw.io / Mermaid).
- A small practice dataset with at least 50 records.

WEEK 2 — Advanced Data Modeling & Indexing

Learning Objectives:

- Model real-world systems in Firestore.
- Avoid hotspots and contention.
- Implement composite indexes.
- Understand distributed queries.

Reading Plan: Data modeling, Firestore indexes, Data types, Best practices for scalability.
Hands-on Work:

- Design 3 end-to-end data models: E-commerce (Users, Products, Reviews), Learning platform, Ticketing system.
- Create ERD diagrams for each.

- Create composite indexes for complex queries.
- Test queries that require indexes.
- Simulate hotspot conditions (writing to same document path).

Deliverables: ERD diagrams (3 models), Indexes JSON export, Query performance comparison table.

WEEK 3 — Query Optimization, Scaling & Best Practices

Learning Objectives:

- Learn efficient querying patterns.
- Understand read-optimization and cost reduction.
- Implement real-time queries at scale.
- Learn advanced filtering and pagination techniques.

Reading Plan: get-data (advanced queries), Bundles, Firestore locations, Real-time queries at scale, Reads/writes performance.

Hands-on Work:

- Implement 20 different query scenarios.
- Build real-time listeners for 3 datasets.
- Optimize a slow query using composite indexes.
- Test pagination strategies: cursor-based vs offset.
- Bundle 100 documents and test load time.

Deliverables: Query optimization report, Bundle implementation example, Performance comparison charts.

WEEK 4 — Security Rules Mastery

Learning Objectives:

- Master rule syntax, expressions, conditions.
- Role-based access control and Field-level validation.
- Testing rules using Emulator Suite.

Reading Plan: Rules overview, Rules structure, Rules conditions, Rules queries & fields, Insecure rules pitfalls.

Hands-on Work:

- Write security rules for: Users, Products, Orders, Reviews.
- Implement: Only authenticated writes, Field validation conditions, Prevent delete after 10 minutes.
- Build 20 failing test cases and 20 passing test cases.

Deliverables: Full "Security Rules Specification" document, Emulator test suite, Role-mapping diagram.

MONTH 2 — Firebase Automation & Integrated Architecture

WEEK 5 — Cloud Functions Basics

Learning Objectives:

- Understand Cloud Functions lifecycle and execution model.
- Work with Firestore, Auth, and Storage triggers.
- Learn logging, monitoring, and debugging using Firebase console.

Reading Plan: Cloud Functions documentation (overview, trigger types, deployment basics).
Hands-on Work:

- Write 6 Cloud Functions:
  - `onCreate`: initialize user profile scaffold.
  - `onWrite`: log document changes to an audit collection.
  - `onDelete`: cleanup related documents.
  - Auth trigger: send welcome email on new user sign-up (mock email).
  - Scheduled function: heartbeat function that writes a timestamp document.
  - Callable function: simple calculation or echo service consumed by a client.
- Deploy functions and verify execution in logs.

Deliverables:

- Cloud Functions code repository with clear folder structure.
- Logging dashboard screenshots with at least 10 successful invocations.
- Short note on Cold Start and how it affects latency.

WEEK 6 — Advanced Cloud Functions + ETL

Learning Objectives:

- Implement batch operations and transactional workflows.
- Design scheduled background jobs for ETL.
- Handle retries and idempotency in functions.

Reading Plan: Bulk operations, Move data, Export-import, Transactions and contention.
Hands-on Work:

- Build a nightly data synchronization function that:
  - Reads active records from one collection.
  - Transforms fields (normalization, enrichment).
  - Writes aggregated data into a reporting collection.
- Build Firestore → Cloud Storage export using Cloud Functions.
- Implement retry logic (simple backoff) for transient failures.
- Build a data-cleaning ETL function that fixes inconsistent fields (e.g., missing timestamps).

Deliverables:

- ETL pipeline documentation (input, transformation, output).
- Flowchart showing ETL steps and triggers.
- Sample run log showing before and after data examples.

WEEK 7 — Firebase Extensions + Storage

Learning Objectives:

- Understand common Firebase Extensions and their use cases.
- Automate heavy workloads (image processing, export jobs).
- Secure Cloud Storage using granular rules.

Reading Plan: Firebase Extensions catalog, Storage security rules, Signed URLs.
Hands-on Work:

- Install and configure 3 extensions:
  - Image Resize.
  - Export Collections to BigQuery.
  - Auth user data sync to Firestore.
- Build upload → resize → store workflow:
  - Create an uploads bucket and `images` collection.
  - Upload original image and verify resized versions.
- Write Storage Security Rules for:
  - Allow only authenticated uploads.
  - Restrict reads to owner for certain folders.

Deliverables:

- Storage Rules document with explanation of each rule.
- Extensions usage guide with configuration screenshots.
- Example of a signed URL workflow (generate and consume).

WEEK 8 — Integration: Full Firebase Backend

Learning Objectives:

- Combine Firestore, Security Rules, Cloud Functions, Storage, and BigQuery.
- Design a cohesive backend for a realistic application.

Hands-on Work:

- Build a backend for a sample application (e.g., course platform or e-commerce):
  - Firestore data model with 4–5 main collections.
  - Security rules enforcing roles (admin, user) and data validation.
  - Cloud Functions implementing business logic (e.g., order creation, notifications).
  - Storage integration for user or product media.
  - BigQuery export enabled via extension.
- Simulate typical workflows end-to-end (user sign-up, data writes, analytics export).

Deliverables:

- Full mini-project repository with README.
- Architecture diagram showing all components and interactions.
- Short "Backend Overview" technical note (1–2 pages).

MONTH 3 — GCP Foundations & BigQuery Mastery

WEEK 9 — IAM + Service Accounts

Learning Objectives:

- Understand GCP IAM hierarchy: organization, folders, projects, resources.
- Design and apply least-privilege access policies.
- Use service accounts for workload identity.

Reading Plan: IAM overview, Roles vs Permissions, Service accounts best practices.
Hands-on Work:

- Create 4 service accounts: Functions, Cloud Run, BigQuery, Deployment.
- Assign least privilege roles to each (e.g., BigQuery Data Editor, Firestore User).
- Configure a service account key for local development (and then learn why to avoid keys in production).
- Document which components use which service account.

Deliverables:

- IAM architecture diagram (project-level).
- Access Matrix table (services vs roles vs service accounts).
- Short document on "Principle of Least Privilege in this Project".

WEEK 10 — Cloud Run

Learning Objectives:

- Deploy containerized services using Cloud Run.
- Connect Cloud Run with Firestore and BigQuery securely.
- Protect endpoints and manage revisions.

Reading Plan: Cloud Run overview, Deploying containers, Integrating with other GCP services.
Hands-on Work:

- Containerize a simple Python API using Flask or FastAPI.
- Deploy to Cloud Run with:
  - One endpoint reading from Firestore.
  - One endpoint running a BigQuery query.
- Configure IAM so only authenticated callers (service accounts) can invoke private endpoints.
- Test deployment using Postman and document request-response examples.

Deliverables:

- Cloud Run deployment guide (step-by-step).
- Postman collection with example calls.
- Screenshot of successful invocations and logs.

WEEK 11 — BigQuery Foundations

Learning Objectives:

- Create and manage BigQuery datasets and tables.
- Write foundational SQL for analytics.
- Use partitioning and clustering to optimize storage and cost.

Reading Plan: BigQuery basics, Loading data, Table partitioning and clustering, Pricing model.
Hands-on Work:

- Load Firestore-exported data into BigQuery.
- Write at least 50 SQL queries covering:
  - Filters, aggregations, grouping.
  - Joins between 2–3 tables.
  - Basic date and string functions.
- Create partitioned tables (e.g., by date) and compare query cost.
- Build a basic dashboard in Looker Studio using BigQuery as a source.

Deliverables:

- SQL practice file (queries with comments).
- Dashboard link or screenshots.
- Notes on how partitioning impacted cost and performance.

WEEK 12 — BigQuery Advanced

Learning Objectives:

- Use window functions for advanced analytics.
- Write stored procedures and user-defined functions.
- Analyze and optimize query performance.

Reading Plan: Window functions, Scripting and stored procedures, Query optimization best practices.

Hands-on Work:

- Write 30 advanced SQL queries including:
  - ROW_NUMBER, RANK, moving averages.
  - Cohort-style analysis for user activity.
- Create 3 stored procedures for recurring analytics tasks.
- Use `EXPLAIN` and query plan to analyze performance and tweak queries.

Deliverables:

- Advanced SQL report summarizing key patterns and queries.
- Stored procedure scripts with usage notes.
- Before/after comparison of at least one optimized query.

MONTH 4 — Python ETL & Automation

WEEK 13 — Python ETL Foundations

Learning Objectives:

- Build small ETL scripts using Python and Pandas.
- Perform data cleaning and transformation consistently.
- Write modular, reusable ETL functions.

Reading Plan: Pandas basics, Data cleaning patterns, Python packaging basics (modules and virtual environments).

Hands-on Work:

- Create 3 ETL scripts:
  - CSV → cleaned CSV.
  - JSON → flattened CSV or DataFrame.
  - Firestore export JSON → normalized table.
- Implement validation functions to check:
  - Null values.
  - Duplicates.
  - Invalid ranges (e.g., negative prices).
- Define a data dictionary describing fields and types.

Deliverables:

- ETL scripts stored in a structured repository.
- Data dictionary for at least one dataset.
- Short write-up on data quality checks implemented.

WEEK 14 — Firestore + Python + BigQuery

Learning Objectives:

- Connect Python applications to Firestore and BigQuery using official clients.
- Build end-to-end pipelines from Firestore to BigQuery.
- Add logging and basic error handling.

Reading Plan: Firestore Python client docs, BigQuery Python client docs, Logging best practices.

Hands-on Work:

- Extract collections from Firestore into Python.
- Transform data into analytics-friendly schemas (flatten nested fields, normalize keys).
- Load transformed data into BigQuery using Python client.
- Add structured logging (info, warning, error) and basic try-except error handling.

Deliverables:

- Complete ETL pipeline script (Firestore → Python → BigQuery).
- Deployment instructions for running this pipeline on Cloud Run or locally.
- Sample log output from at least one successful and one failed run (with explanation).

WEEK 15 — Playwright Automation

Learning Objectives:

- Use Playwright for browser automation and scraping.
- Extract structured data from websites.
- Clean and persist scraped data for analytics.

Reading Plan: Playwright Python docs (selectors, navigation, waits), Legal/ethical considerations of scraping.

Hands-on Work:

- Scrape 3 websites with different structures (list page, detail page, paginated listing).
- Extract structured data (e.g., products, articles, or events).
- Clean data (remove duplicates, normalize text, parse dates).
- Store cleaned results in:
  - CSV/Excel.
  - BigQuery table via ETL step.

Deliverables:

- Playwright automation scripts.
- Sample datasets generated from the runs.
- Brief document on the scraping approach, including selectors and pagination handling.

WEEK 16 — Automated Pipelines

Learning Objectives:

- Orchestrate multi-step pipelines using cloud schedulers and services.
- Implement retries, alerts, and failure notifications.
- Combine scraping, ETL, and loading into a single flow.

Reading Plan: Cloud Scheduler basics, Cloud Run + Scheduler integration, Monitoring and alerting introduction.

Hands-on Work:

- Create a pipeline: Playwright → ETL → BigQuery.
- Deploy the ETL/scraping service to Cloud Run.
- Schedule runs using Cloud Scheduler (e.g., daily at a fixed time).
- Add:
  - Retry policy for transient failures.
  - Basic alert (e.g., email or log-based) when a run fails.

Deliverables:

- Fully automated pipeline running on schedule.
- Pipeline architecture diagram (from trigger to data warehouse).
- "Runbook" document: what to check when the pipeline fails.

MONTH 5 — Full Data Engineering Projects

WEEK 17 — Project 1: Firebase Data Engineering System — Design & Setup

Learning Objectives:

- Apply all Firebase skills in a cohesive real-world project.
- Design a scalable backend for a specific business domain.

- Plan data flows, access patterns, and security upfront.

Project Scope: Build a complete backend system (e.g., analytics-enabled e-commerce or learning platform) using Firestore, Security Rules, Cloud Functions, Storage, and BigQuery.

Hands-on Work:

- Finalize project domain and requirements (features, entities, basic workflows).
- Design:
  - Firestore schema (collections, subcollections, indexes).
  - Security rules strategy (roles, validation logic).
  - Cloud Functions responsibilities (triggers and callable APIs).
  - Storage usage patterns (what is stored where, access rules).
  - BigQuery export strategy (which collections to export and how often).
- Set up:
  - Firebase project.
  - BigQuery datasets.
  - IAM roles and service accounts.

Deliverables:

- High-level design document (3–5 pages) with requirements, architecture, and tech stack.
- Firestore schema diagram (ERD-style) and index list.
- Initial repository structure (folders for functions, rules, docs).

WEEK 18 — Project 1: Implementation, Testing & Documentation

Learning Objectives:

- Implement the designed Firebase backend end-to-end.
- Add comprehensive testing and basic monitoring.
- Produce high-quality technical documentation.

Hands-on Work:

- Implement:
  - Firestore collections and seed data.
  - Security rules covering all critical operations.
  - Cloud Functions for business workflows (e.g., order creation, notifications, aggregates).
  - Storage flows for media uploads (if applicable).
  - BigQuery export via extension and verify exported schemas.
- Testing:
  - Write rule tests in Emulator Suite.
  - Test Cloud Functions with both success and failure paths.
  - Validate that BigQuery contains correct and timely data.
- Documentation:
  - Update design doc with actual implementation details.
  - Add “How to Run Locally” and “How to Deploy” sections.

Deliverables:

- Working Firebase backend project in GitHub.
- Technical design and implementation document (final version).
- Test summary (what was tested, how, and results).

WEEK 19 — Project 2: Python–GCP ETL System — Design & First Version

Learning Objectives:

- Design a production-style ETL system using Python and GCP.
- Integrate scraping or ingestion, transformation, and loading to BigQuery.
- Think in terms of data pipelines and modular stages.

Project Scope: Build an end-to-end pipeline (Scrape/Ingest → Clean → Store → BigQuery → Dashboard), including monitoring and logging.

Hands-on Work:

- Define the data source(s): public website, API, or synthetic data.
- Design:
  - Ingestion method (Playwright, requests, or API client).
  - Transformation steps (cleaning, normalizing, enriching).
  - Storage layout (intermediate files in Storage vs direct load).
  - BigQuery table schemas (raw and processed).
- Implement first pipeline version:
  - Ingest data and store raw dumps.
  - Clean and transform into final schema.
  - Load into BigQuery.

Deliverables:

- ETL project repository with clear module separation.
- Pipeline design document (components, flow, dependencies).
- Initial dashboard or query set demonstrating value from the data.

WEEK 20 — Project 2: Optimization, Pub/Sub & Async

Learning Objectives:

- Optimize ETL performance and reliability.
- Use Pub/Sub for event-driven processing.
- Apply asynchronous programming where appropriate.

Hands-on Work:

- Introduce Pub/Sub between ingestion and processing stages (optional but ideal):
  - Publish messages when new raw data is available.
  - Subscribe with a processing component that transforms and loads data.
- Optimize:
  - Use async scraping or parallel requests where safe.
  - Batch inserts into BigQuery instead of row-by-row.
- Add robustness:
  - Implement retry and dead-letter queue pattern (or simulate).
  - Improve logging with correlation IDs for tracing.

Deliverables:

- Optimized ETL pipeline version (tagged release in repository).
- Performance comparison (before vs after optimization).
- Updated documentation describing Pub/Sub and async enhancements.

MONTH 6 — System Design, Portfolio & Interviews

WEEK 21 — System Design (Firebase + GCP)

Learning Objectives:

- Design scalable, fault-tolerant data systems.
- Communicate architecture clearly using diagrams and narratives.
- Compare different design choices and trade-offs.

Hands-on Work:

- Create 3 full system design case studies:
  - High-scale event logging system with Firestore + BigQuery.
  - Real-time analytics dashboard using Firebase and BigQuery.
  - Data ingestion platform using Pub/Sub, Cloud Run, and BigQuery.
- For each case:
  - Draw architecture diagram.
  - Write a 1–2 page design explanation (requirements, components, data flow, scaling, failure handling).
  - Identify bottlenecks and monitoring points.

Deliverables:

- 3 system design diagrams.
- 3 accompanying design documents.

WEEK 22 — Advanced Data Engineering Concepts

Learning Objectives:

- Understand streaming vs batch pipelines conceptually.
- Explore Dataflow-style architectures and data governance basics.
- Connect your existing projects to these concepts.

Reading Plan: Dataflow concepts, Streaming vs batch processing, Data governance and quality at scale.

Hands-on Work:

- Design (and optionally implement a small part of) a Dataflow-like pipeline using your existing ETL as a base:
  - Identify which parts could be streaming.
  - Sketch a pipeline with Pub/Sub ingestion and processing stages.
- Define data quality and governance rules for one project:
  - Required fields, allowed ranges, retention policies.
  - Access and compliance considerations.
- If possible, build a small streaming-like simulation (e.g., repeated ingestion of changing data into BigQuery).

Deliverables:

- Dataflow-style pipeline design document.
- Data quality and governance checklist for one project.
- Optional mini-project or script simulating streaming updates.

WEEK 23 — Portfolio Building

Learning Objectives:

- Package your projects into a professional portfolio.
- Present your skills clearly for data engineering roles.
- Align documentation and GitHub structure with industry expectations.

Hands-on Work:

- Polish 2 main projects (Firebase backend + Python ETL):
  - Ensure README includes overview, tech stack, architecture diagram, setup steps.
  - Add links to any dashboards or demo endpoints.
- Create:
  - Portfolio folder summarizing all key work (PDF or Markdown).
  - Short project summaries (problem, solution, tools, impact).
- Update resume:
  - Add projects with measurable outcomes.
  - Highlight Firebase, GCP, BigQuery, ETL, and Python skills.

Deliverables:

- Polished GitHub repositories for at least 2 projects.
- Portfolio summary document.
- Resume Version 1 targeted at data engineer roles.

WEEK 24 — Final Revision, Mock Interviews & Wrap-Up

Learning Objectives:

- Consolidate knowledge across Firestore, Firebase, GCP, SQL, and Python.
- Practice explaining your work clearly in interviews.
- Identify and patch any remaining weak areas.

Hands-on Work:

- Revision:
  - Review notes from all previous months (Firestore, rules, Functions, BigQuery, ETL).
  - Revisit tricky topics (e.g., security rules edge cases, query optimization).
- Mock interviews:
  - Conduct at least 3 self or peer mock interviews:
    - One focused on Firestore and Firebase.
    - One on GCP, BigQuery, and SQL.
    - One purely on project discussion and system design.
- Final polish:
  - Fix any gaps found during mocks (missing diagrams, unclear docs).
  - Create a one-page "Profile Snapshot" summarizing skills, tools, and top 2 projects.

Deliverables:

- Final version of all project repositories and documentation.
- Notes from mock interviews and action items.
- One-page personal summary for recruiters or mentors.