Snowflake + dbt + Airflow + PostgreSQL + Airbyte + Superset Financial Data Platform POC (10-Day Plan)

1. Background & Objective

I am a Principal Data Engineer who has recently completed internal technical training and must build a hands-on Proof of Concept (POC) before client onboarding (Morgan Stanley).

Primary Evaluation Goals:

- Demonstrate end-to-end modern data platform with source systems, ingestion, transformation, and visualization
- Show production-realistic patterns with governance, security, and operational excellence
- Prove effective integration of diverse technologies while maintaining simplicity
- Model financial data safely and realistically with clear business value
- Handle real-world data engineering concerns (transactional sources, CDC, cost optimization)

Key Differentiator: This POC demonstrates a complete data platform from source to dashboard with justification for each technology choice.

2. Prerequisites & Setup Requirements

Local Software (Windows/Mac)

#	Software	Purpose	Installation
1	Docker Desktop	Runs Airflow, Airbyte, PostgreSQL, Superset, dbt containers	Download from docker.com
2	Python 3.10	Core programming language (optional - for local scripts)	Download from python.org
3	Git	Version control	Download from git-scm.com
4	VS Code	Code editor	Download from code.visualstudio.com
5	DBeaver	Multi-database GUI client	Download from dbeaver.io

Cloud Accounts (Free Tier)

#	Account	Purpose	Credits/Limits
1	GCP Free Tier	Cloud Storage, Cloud SQL, BigQuery (staging)	$300 credits, 90 days
2	Snowflake Trial #1	Producer account (data engineering)	$400 credits, 30 days
3	Snowflake Trial #2	Consumer account (analytics)	$400 credits, 30 days

Note: All components except Snowflake and GCS run locally via Docker. PostgreSQL simulates source system, keeping costs at zero. Total setup time approximately 4-5 hours.

3. Core Technology Stack & Explicit Decisions

Technology Stack & Rationale:

Technology	Purpose	Why Chosen
PostgreSQL (Docker)	Source transactional database	Simulates enterprise OLTP system, enables CDC demonstration
Airbyte (Docker)	ELT ingestion from PostgreSQL to Snowflake	Demonstrates modern data integration, handles schema evolution
GCP Cloud Storage	Landing zone for external data	Free tier available, enterprise-grade object storage
Snowflake	Enterprise data warehouse & transformation	Core evaluation focus, handles structured/semi-structured data
dbt Core (Docker)	Analytics engineering & transformation	Containerized for consistency, easy version management
Apache Airflow (Docker)	Orchestration & scheduling	Production-grade workflow management, rich ecosystem
Superset (Docker)	Business intelligence & visualization	Open-source BI, integrates with Snowflake via SQLAlchemy

Architectural Decisions with Justification:

Decision	Justification	Production Consideration
Everything in Docker	Consistent environment, easy setup/teardown, no	Production would use managed

Decision	Justification	Production Consideration
except GCP/Snowflake	local dependency conflicts	services for reliability
PostgreSQL locally via Docker	Simulates source system without cloud costs. Enables CDC demonstration via logical replication	Production would use managed PostgreSQL (Cloud SQL, RDS)
Airbyte for ingestion	Handles schema evolution, incremental syncs, and error handling out-of-the-box	Production would use Airbyte Cloud or enterprise deployment
GCP over AWS	GCP's $300 free tier offers more services. BigQuery available for staging layer option	Both are enterprise-viable. GCP's data ecosystem integrates well
Snowflake as central warehouse	Required for evaluation. Superior performance for analytics workloads	Production would leverage Enterprise edition
dbt in Docker	Consistent runtime environment, easy to share across team members	Production would use dbt Cloud for CI/CD and collaboration
Two Snowflake Trial Accounts	Models producer/consumer data mesh. Enables secure sharing demonstration	Enterprise would use multiple accounts for compliance

4. Data Domain & Business Context

Domain: Mutual Funds + Investor Transactions + Market Data

Finance Knowledge Constraint Acknowledgement:

- I am new to finance domain
- Financial logic must be: Simple, Explainable, Realistic
- NAV and prices are consumed, not derived from scratch

Business Metrics Focus (Simple but Meaningful):

- Daily Fund Performance (NAV % change day-over-day)
- Assets Under Management (AUM) by fund (simplified calculation)
- Investor Transaction Trends (weekly summary, deposits vs withdrawals)
- Fund Manager Performance (historical analysis)
- Customer Segmentation by Investment Behavior

5. Data Sources & Ingestion Patterns (Enhanced)

Source	Format	Ingestion Pattern	Business Purpose
PostgreSQL Transactions	Tables	Airbyte CDC (Logical Replication)	Investor activity simulation, real transactional data
Mutual Fund NAV	CSV	GCS → Snowpipe	Core pricing data, updates daily
Fund Metadata	CSV	Airbyte batch sync	Fund characteristics, slow-changing
Market Prices (Index)	JSON (Mock API)	Python → GCS → Snowpipe	Benchmark comparison data
Fund Manager Info	CSV	Airbyte with SCD Type 2	Slowly changing dimension demo
Reference Data	Static CSV	dbt Seeds	Calendar, Currency codes
External Ratings Data	JSON	Airbyte API connector (simulated)	Fund ratings from external source

Ingestion Patterns Demonstrated:

- CDC from PostgreSQL (change data capture)
- Batch synchronization (full/incremental)
- API ingestion (simulated via Airbyte)
- Event-driven micro-batch (Snowpipe)
- Idempotent processing patterns

6. Architecture & Data Flow

End-to-End Flow:

```text
```

```
PostgreSQL (Docker - Source)
  → Airbyte (Docker - CDC/Batch ELT)
  → GCS (Landing Zone)
  → Snowflake RAW (via Snowpipe)
  → dbt (Docker - Transformation: Staging → Intermediate → Marts)
  → Snowflake Data Sharing
  → Consumer Account
  → Superset (Docker - Dashboards)
  ↑
Apache Airflow (Docker - Orchestration)
```

7. PostgreSQL Source System Design

Simulated Banking Schema:

- Core transactional tables: investors, accounts, transactions
- Logical replication enabled for CDC capability
- Realistic financial data model with referential integrity

8. Airbyte Implementation Strategy

Connection Configuration:

- **Source:** PostgreSQL (with logical replication for CDC)
- **Destination:** Snowflake RAW layer
- **Sync Mode:**
  - Full refresh for static tables
  - Incremental (append) for transaction tables
  - CDC for investor/account dimensions

Airbyte Features Demonstrated:

- Schema evolution handling
- Normalization to Snowflake native tables
- Error handling and automatic retries
- Connection scheduling via Airflow API
- Data type mapping between PostgreSQL and Snowflake

9. dbt Implementation Strategy (Dockerized)

Models Architecture:

```text
models/
├── staging/
│   ├── postgres/         # PostgreSQL source models
│   ├── csv_sources/      # CSV/File source models
│   └── api_sources/      # API source models
├── intermediate/         # Business transformations, reusable
├── marts/                # Consumer-ready data products
│   ├── finance/          # Fund performance, AUM
│   ├── investor/         # Investor behavior analytics
│   └── reporting/        # Aggregated metrics for dashboards
└── seeds/                # Reference data
```

Advanced Features Demonstrated:

- **Tests:** not_null, unique, relationships, custom data quality tests
- **Snapshots:** SCD Type 2 for fund metadata and manager info
- **Macros:** Reusable SQL for financial calculations
- **Packages:** dbt_utils, dbt_expectations in packages.yml
- **Exposures:** Superset dashboard connections
- **Documentation:** Complete data dictionary with lineage
- **Meta Tags:** Data classification (pii: true, business_critical: true)

10. Airflow Orchestration Pattern

Design Philosophy: Airflow as pure orchestrator, triggers all Docker services, manages dependencies.

Enhanced DAG Structure:

- Trigger Airbyte syncs for PostgreSQL to Snowflake
- Check GCS for new files using sensors
- Trigger Snowpipe auto-ingest from GCS
- Run dbt transformations (staging then marts)
- Refresh Superset cache for updated dashboards
- Manage dependencies and orchestration flow

11. Superset Dashboard Strategy

Dashboard Design:

- **Fund Performance Dashboard:** NAV trends, comparison to benchmarks
- **Investor Analytics Dashboard:** Transaction patterns, AUM growth
- **Data Quality Dashboard:** Pipeline health, data freshness
- **Operational Dashboard:** Pipeline run times, error rates

Integration Pattern:

- Direct Snowflake connection from Superset using SQLAlchemy
- Semantic layer definitions in Superset
- Scheduled cache refresh for performance
- Row-level security via Superset roles mapped to Snowflake roles

12. Snowflake Features Demonstrated

Feature	Demonstration Purpose	Implementation
External Stages (GCS)	Landing zone pattern	GCS integration with HMAC keys
Snowpipe Auto-ingest	Event-driven micro-batch	GCS notifications to Snowpipe
Streams	CDC for incremental processing	PostgreSQL change tracking
Secure Data Sharing	Data product distribution	Share marts to consumer account
Time Travel	Safe development, error recovery	1-day time travel for debugging
Warehouse Management	Size scaling, auto-suspend	Separate load/transform/query WH
Tasks	Internal scheduling	Daily aggregation tasks
Materialized Views	Performance optimization	Pre-aggregated investor metrics
Dynamic Data Masking	Column-level security	Mask PII in consumer account
NEW: External Functions	Call external APIs	Integrate with rating service

13. GCP Infrastructure Setup

Minimal GCP Resources:

- **Cloud Storage Bucket:** fin-data-landing-zone
- **Folders:** nav-data/, market-data/, airbyte-staging/
- **Service Account:** snowflake-ingester with HMAC keys
- **IAM Roles:** Storage Object Admin for Snowpipe

GCP Setup via Web Console (No gcloud needed):

1. Create project via web console
2. Enable Cloud Storage API
3. Create bucket via UI
4. Generate HMAC keys for Snowflake
5. Set up Pub/Sub notifications for Snowpipe

Cost Control Measures:

- Use only us-central1 region (lower costs)
- Set up budget alerts at $50
- Enable object lifecycle rules (delete after 30 days)

14. Data Governance & Security Layer

Classification & Tagging:

- Data classification tags applied in Producer account
- PII identifiers for sensitive tables
- Business domain and cost center tagging for governance

Security Implementation:

- **PostgreSQL:** Row-level security for source data
- **Airbyte:** Encrypted connections, no data persistence
- **GCP:** Service accounts with least privilege
- **Snowflake Producer:** Role hierarchy (loader, transformer, viewer)
- **Snowflake Consumer:** Read-only role + dynamic masking

- **Superset:** Role-based access to dashboards

End-to-End Lineage:

- Airbyte connection catalog → dbt documentation → Superset datasets
- Complete visibility from PostgreSQL to dashboard

15. Observability & Operations Strategy

Four-Layer Observability:

1. **Source System:** PostgreSQL replication lag, row counts
2. **Ingestion:** Airbyte sync success/failure, record counts
3. **Transformation:** dbt run results, test failures, model timing
4. **Consumption:** Superset query performance, dashboard load times

Monitoring Dashboard (Superset):

- Pipeline health metrics
- Data freshness by source
- Snowflake credit consumption
- Data quality test results

Centralized Logging:

- JSON file logging driver for all services
- Log rotation with size limits (10MB max per file)
- Retention policy (3 files maximum)

16. Execution Timeline (10 Days - Enhanced)

Day	Focus Area	Key Deliverables	Success Criteria
1	Docker Foundation	Docker Compose setup, PostgreSQL schema, GCP web setup	All containers running, GCS bucket created
2	Source & Ingestion	PostgreSQL data simulation, Airbyte connections to Snowflake	CDC working, data flowing to Snowflake RAW
3	External Data	GCS web setup, Snowpipe configuration, file ingestion	Snowpipe auto-ingest working

Day	Focus Area	Key Deliverables	Success Criteria
4	dbt Staging Layer	Dockerized dbt, source models, staging transformations	Clean, tested staging models
5	dbt Business Logic	Intermediate models, financial calculations, reusable macros	Business logic implemented and tested
6	Data Products	Marts creation, star schemas, data sharing setup	Consumer account can query shared data
7	Orchestration	Airflow DAGs, Airbyte integration, dependency management	End-to-end pipeline orchestrated
8	Visualization	Superset setup, dashboard creation, semantic layer	Business dashboards operational
9	Governance & Security	Tagging, masking policies, row-level security, monitoring	Security controls implemented
10	Polish & Narrative	Documentation, walkthrough prep, optimization	Compelling story from PostgreSQL to dashboard

17. Success Criteria & Evaluation Narrative

The POC is successful if it demonstrates I can:

1. **Design & Implement** a complete modern data platform from source to insight
2. **Integrate Diverse Technologies** seamlessly with clear handoffs
3. **Handle Realistic Data Patterns** including CDC, schema evolution, and incremental processing
4. **Apply Production-Ready Patterns** for security, governance, and observability
5. **Optimize for Cost & Performance** across all layers
6. **Communicate Business Value** through tangible dashboards and metrics
7. **Justify Architecture Decisions** with clear trade-offs and migration paths

Walkthrough Narrative Structure:

1. Start with the business dashboard: "Show me yesterday's fund performance and investor activity"
2. Trace backward through: Superset → Snowflake Consumer → Shared Data → Curated Marts → Staging → Raw Ingestion → Airbyte → PostgreSQL
3. Highlight key decisions at each layer with "why" explanations
4. Demonstrate handling of a real-world scenario (e.g., late-arriving transaction)

5. Show observability across the entire stack
6. End with production migration considerations for each component

18. Production Migration Considerations

Component	POC Implementation	Production Recommendation
PostgreSQL	Docker local	GCP Cloud SQL or AlloyDB with HA
Airbyte	Docker local	Airbyte Cloud or self-hosted K8s
GCS	Free tier bucket	Multi-region with lifecycle policies
Snowflake	Trial accounts	Enterprise with Business Critical
dbt	Docker container	dbt Cloud with CI/CD
Airflow	Docker local	Cloud Composer or Astronomer
Superset	Docker local	Managed instance or Kubernetes
Monitoring	Basic logging	Datadog/Splunk integration

This enhanced POC demonstrates a complete, production-realistic data platform while maintaining focus on Snowflake capabilities as the central evaluation component. All components are containerized for consistency and ease of setup.