

Group Project: Healthcare Data Engineering

Updated Project Status & Team Tasks

Outstanding Progress Update

Team Lead Christian has successfully implemented the core infrastructure and ETL pipeline! The foundation is solid and ready for the final components.

Completed Components (Great work!)

1. Infrastructure & Environment Setup

- **Docker Environment:** PostgreSQL + pgAdmin fully configured
- **Python Environment:** All dependencies installed and working
- **Cross-platform Compatibility:** Works on both Linux and Windows

2. Data Cleaning Pipeline (100% Complete)

- Successfully processes all data formats (CSV, JSON, SQLite)
- Standardizes dates, phone numbers, and zip codes
- Handles missing values intelligently (861 missing value_text entries managed)
- Generates comprehensive quality reports

3. Database Implementation

- **Schema:** Professionally designed PostgreSQL schema with 7 tables
- **Security:** Role-based access control, audit logging, GDPR compliance framework
- **Performance:** Indexes, views, and optimized queries
- **Data Loading:** Successfully loaded 1,762 healthcare records:
 - Patients: 150 records
 - Encounters: 291 records
 - Diagnoses: 221 records
 - Medications: 160 records
 - Procedures: 54 records
 - Observations: 886 records

Tasks for Team Completion

Kian - Database Specialist Tasks

Your expertise is needed to bring the query implementation to life! The SQL queries are written, but we need your skills to make them shine.

Task 1: Complete Query Runner Implementation (Priority: HIGH)

File: `scripts/query_runner.py`

The query methods are stubbed out but need your implementation:

1. **Test each query** in `sql/queries.sql` directly in PostgreSQL first
2. **Implement the query methods** in `query_runner.py`
3. **Add performance timing** to each query
4. **Format output** for clear presentation

Specific queries to implement:

- ☒ Query 1: Average age and age range (REQUIRED)
- ☒ Query 2: Average patients per physician (REQUIRED)
- ☒ Query 3: Diagnosis analysis by age group
- ☒ Query 4: Lab results trends
- ☒ Query 5: Care continuity patterns

Task 2: Performance Testing & Optimization

1. **Run EXPLAIN ANALYZE** on each query
2. **Document query execution times**
3. **Optimize slow queries** (if any > 1 second)
4. **Create performance report** with:
 - Query execution times
 - Index usage statistics
 - Optimization recommendations

Task 3: Database Comparison Analysis (For Final Report)

Create a comparison of PostgreSQL vs 2 other databases:

1. **MongoDB** (NoSQL option)
2. **Cassandra** (Distributed option)

Compare on:

- Healthcare data structure fit
- Performance characteristics
- Scalability potential
- Security features

Deliverable Format: 1-2 page analysis in `docs/database_comparison.md`

Kiana - Data Quality Analyst Tasks

Your analytical skills will ensure our data meets healthcare standards! The cleaning is done, now we need deep quality insights.

Task 1: Advanced Data Quality Analysis

Create: `scripts/data_quality_analyzer.py`

Implement comprehensive quality metrics:

1. Completeness Analysis

- Field-by-field completion rates
- Required vs optional field analysis
- Pattern detection in missing data

2. Validity Analysis

- Data type consistency
- Format validation (dates, IDs, codes)
- Business rule compliance

3. Consistency Analysis

- Cross-table consistency checks
- Temporal consistency (dates make sense)
- Duplicate detection

4. Accuracy Analysis (if reference data available)

- Compare diagnosis codes to ICD-10 standards
- Validate medication names against drug databases

Task 2: Data Quality Visualization

Create visualizations showing:

1. **Quality scores by dataset** (bar chart)
2. **Missing data heatmap**
3. **Data distribution plots** for key fields
4. **Anomaly detection results**

Tools: Use matplotlib/seaborn or create an HTML dashboard

Task 3: Quality Improvement Recommendations

Document in `docs/data_quality_report.md`:

1. **Critical Issues Found** (if any)
2. **Data Quality Score Summary**
3. **Recommendations for Production**
4. **Monitoring Strategy**

Task 4: Test Case Development

Create `tests/test_data_quality.py` with:

1. Unit tests for validation rules
 2. Integration tests for data flow
 3. Edge case handling tests
-

Christian - Final Integration Tasks

Excellent work on the pipeline! Just a few finishing touches needed:

Task 1: Security Documentation

Complete `docs/security_policy.md`:

1. **Data Access Controls** (using the roles created)
2. **Encryption Strategy** (column-level for PII)
3. **GDPR Compliance Checklist**
4. **Backup & Recovery Procedures**
5. **Audit Log Monitoring**

Task 2: System Architecture Documentation

Create `docs/architecture.md` with:

1. **System Overview Diagram**
2. **Data Flow Diagram**
3. **Technology Stack Justification**
4. **Scalability Considerations**

Task 3: Final Integration

1. **Integrate Kian's queries** into main pipeline
 2. **Add Kiana's quality checks** to the flow
 3. **Create final execution script** that runs everything
 4. **Generate comprehensive final report**
-

Timeline & Milestones

Week 1 (Remaining Tasks)







- **Day 1-2:** Kian completes query implementation
- **Day 2-3:** Kiana implements quality analysis
- **Day 4:** Integration and testing
- **Day 5:** Documentation completion

Week 2 (Final Week)






- **Day 1-2:** Performance testing and optimization
 - **Day 3:** Final report preparation
 - **Day 4:** Presentation preparation
 - **Day 5:** Project submission
-

Definition of Done

For Each Component:

-  Code is tested and working
-  Performance meets requirements (< 5 sec for queries)
-  Documentation is complete
-  Code follows PEP 8 standards
-  Logging is comprehensive
-  Error handling is robust

For Final Submission:

-  All 5 queries execute successfully
 -  Data quality report shows > 90% quality score
 -  Security policy is comprehensive
 -  Performance report included
 -  GitHub repository is clean and organized
-

Quick Start for Teammates

bash

1. Clone the repository

```
git clone git@github.com:SnakeyRoad/healthcare-data-engineering-project.git
cd healthcare-data-engineering-project
```

2. Start Docker services

```
docker-compose up -d
```

3. Set up Python environment

```
python3 -m venv venv
source venv/bin/activate # Windows: venv\Scripts\activate
pip install -r requirements.txt
```


4. Run the pipeline (data is already cleaned and loaded!)

```
python scripts/main.py --step queries # For Kian
python scripts/data_quality_analyzer.py # For Kiana (create this)
```

5. Access the database

```
# pgAdmin: http://localhost:5050
# Email: admin@healthcare.com
# Password: admin_password_2024
```

Final Deliverables Checklist

- ☐ Working ETL Pipeline  (Complete!)
 - ☐ 5 Cross-Dataset Queries (Kian - in progress)
 - ☐ Data Quality Analysis (Kiana - in progress)
 - ☐ Database Comparison (Kian)
 - ☐ Security Documentation (Christian)
 - ☐ Architecture Documentation (Christian)
 - ☐ Performance Report (Team)
 - ☐ Final Presentation (Team)
-

Communication

- **Daily Standup:** Share progress in team chat
 - **Blockers:** Reach out immediately if stuck
 - **Code Reviews:** Review each other's PRs before merging
 - **Final Review:** Thursday before submission
-

Remember: We've already accomplished the hardest parts! The foundation is solid, and we just need to add the finishing touches. Each of your contributions is crucial for project success. Let's bring this home! 🚀

Last Updated: June 23, 2025