Data Management Portfolio Project:

Corporate Workflow Management

Srushti Borkar

Shayana Reddy

Athul Rohan

Contents

1.Introduction

1.1 Project Overview

This project is based on a **simple workflow system** that shows how work is handled inside an organization. Every piece of work is treated as a **job case** with a unique job case ID, and each job case goes through different stages before it is completed.

The idea is to understand how work (that is shared by clients who are not included in this example) is **planned**, **executed**, and **checked for quality** (and delivered back), and how data from each stage can be stored and analyzed using MongoDB and Python.

1.2 Scenario Explanation

The workflow is divided into **three main stages**:

1. **Review Stage**

   In the review stage, the job case is first analyzed. Here, an estimate is made about how much time the job might take. This estimated effort helps in planning resources and setting expectations. A limited number of reviewers handle multiple job cases, which reflects a real workplace situation.

2. **Production Stage**

   In the production stage, the project is worked on. The system records when the work starts and when it was ended – which can be used to calculate time taken. Employees work on more than one task per day but in our example the sample chosen is one random employee on one day.

3. **Quality Control Stage**

   After the work is completed, the job case is checked. Each job case receives an error score that gives a quality rating. Also, a decision is made on if rework is necessary or not, usually based on error score. This stage helps ensure client satisfaction.

2. Data Model and Rationale

The data is organized into **three separate datasets**, one for each stage of the workflow:

- Review
- Production
- Quality control

All three datasets are linked using the foreign key **“job_case_id”** - the common foreign key for all tables.

Each dataset focuses on a specific part of the workflow:

- Review data focuses on planning and estimation
- Production data focuses on the execution
- Quality control data focuses on errors and rework decisions

2.1 Main Entities and Relationships

The Review entity represents the planning stage, where estimated effort and department details are recorded. The reviewer sets an estimated time for each case Id they review. This would be used to charge clients (in a real-life scenario) and used as a control to test production operator KPIs against.

The Production entity captures the execution stage, including who performed the work and how long it took.

The Quality Control entity stores the final evaluation quality level, error scores and whether rework was required or not.

All entities are connected using the job_case_id, creating a clear and linear workflow. Each job case appears once in each stage, allowing the full process to be tracked from start to finish.

2.2 Rationale Behind the Relationship Structure

Keeping track of all tables using one singular Job case ID helps ensure consistency and reliability of the database. In hypothetical negative feedback from the client’s side, the project can be traced easily all the way to the start to properly understand where what went wrong.

This relationship design also allows flexible analysis. Data can be grouped by department, employee, or quality outcome without changing the structure of the model. It also makes it easier to extend the system in the future.

Overall, the relationships are defined to keep the data connected, efficient, and ready for both analysis and future growth.

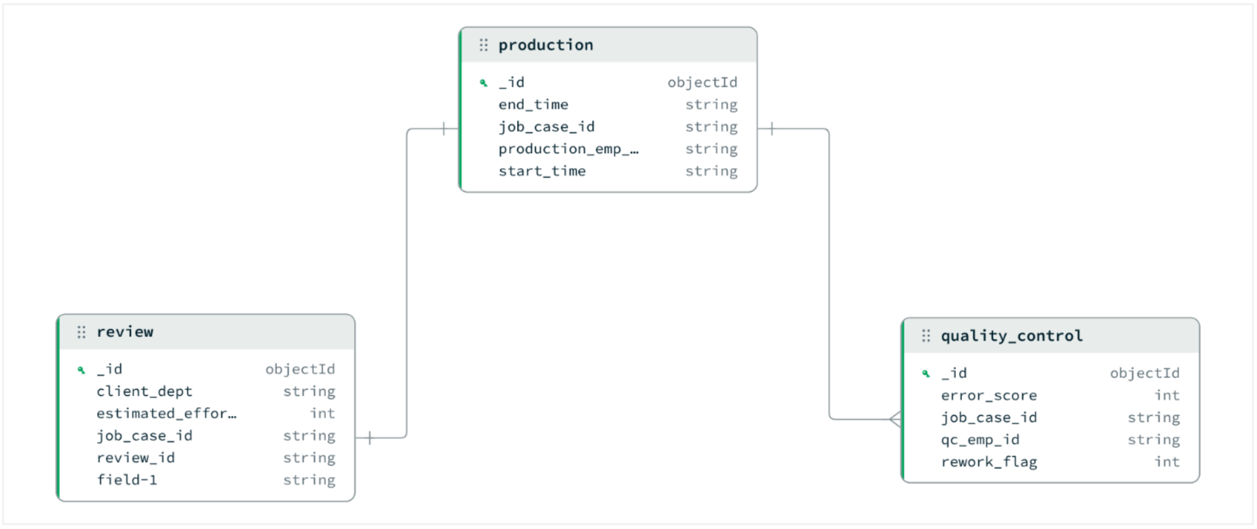
2.3 Reasoning behind table selection

These tables were chosen to organize the workflow in a structured and manageable way. Each table represents a distinct stage of the process, which helps keep related data grouped together. This separation supports clearer organization of information and prevents different types of data from being mixed within the same structure.

The table design also supports efficient analysis. Data related to a specific stage can be accessed and analyzed independently, reducing complexity during data processing and reporting.

Another reason for this approach is adaptability. As requirements change, individual stages can be modified or extended without requiring changes to the entire data structure. This helps maintain stability over time.

2.4 ER Diagram



The ER diagram represents a workflow with three main entities: Review, Production, and Quality Control. Each table has its own primary key (_id), and all tables are linked using the Job Case ID column, which represents a single unit of work.

The relationship between **Review and Production is one-to-one (1:1).** Each job case is reviewed once and then moves to exactly one production record. This reflects the planning-to-execution flow of the process.

The relationship between **Production and Quality Control is one-to-many (1:N)**. A single production record can have multiple quality control checks, for example if rework is required or if multiple inspections are performed.

This relationship design allows each job case to be tracked clearly across stages while supporting repeated quality checks without duplicating production or review data.

3. ETL Concept

3.1 ETL-in: Data Ingestion

In a real-world scenario, workflow data would be captured through internal operational systems used for review, production, and quality control activities, APIs would update the data directly into the MongoDB server. These systems ensure that relevant information is recorded as work progresses through each stage.

For this project, these inputs are represented using CSV files, which act as structured data exports and are loaded into MongoDB for further processing. Before analysis, the data is cleaned, formats are standardized, and basic transformations are applied, such as converting error scores into quality categories and rework indicators. The datasets are then linked using a common job case identifier.

While data would typically be updated regularly in practice, this project processes the data in batches, which is sufficient for demonstrating the ETL process and supporting analysis.

3.2 ETL-out: How data would be used or leave the solution

In this project, ETL-out is executed using Python after connecting it to MongoDB. The CSV files are first imported into MongoDB as separate collections, and MongoDB is then connected to the Python environment. From there, the data is extracted into Python for further processing.

Once available in Python, the data is used for analysis and visualization. Aggregations and transformations are performed to evaluate workflow performance, time usage, and quality outcomes. Python libraries are used to create visualizations that help identify patterns, trends, and issues across different stages of the workflow. The results of this analysis can also be exported as CSV or Excel files for reporting and documentation purposes.

This approach reflects a typical analytical workflow, where data is stored in a database, accessed through a tool like Python, and then analyzed and visualized to uncover insights that help with business related decisions.

In an ideal real-life severalhere would be be multiple ETL-Out pipelines, one could be a PowerBI or

Tableau dashboard viewed by the CEO along with his/her morning coffee, another would be the dozens of managers to connect to the server via Excel to get timely updates on internal performance and of course there would be several analysts who would probably use Python to help make strategic decisions.

The ETL-out process supports questions such as how actual production time compares with estimates, how frequently rework occurs, and where quality issues are most common. From this, indicators such as time variance, rework rate, and error levels can be derived to support monitoring and improvement decisions.

4. Implemented python part

4.1 Overview

The notebook connects to the workflow management database using the **pymongo library**. Clicking on "Run All" refreshes the database connection and reloads the data, ensuring that the notebook always works with the most up-to- date information available on the server.

Connecting directly to MongoDB instead of exporting a static CSV file and then loading it ensures that the analysis is dynamic. As workflow management is a "living" process, a live connection ensures the dashboard reflects the status of jobs, for instance, a job finished 5 minutes ago appears immediately upon "Run All"). To add to that, it prevents "version drift" where different managers might be looking at different versions of exported files.

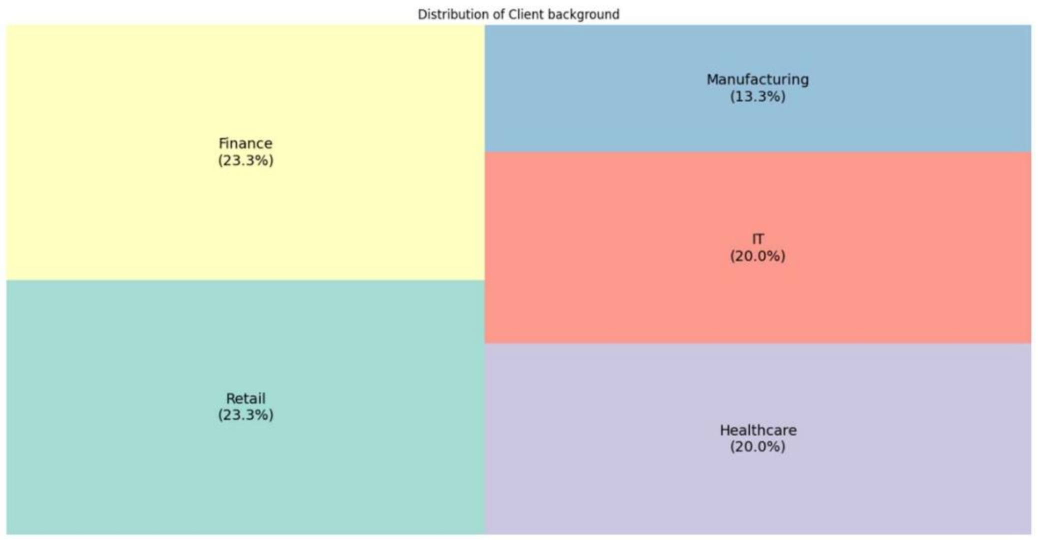
Once the connection is established, the next step is to r**ead the relevant database collections and assign them to separate variables** for further processing.

The three collections/tables used in this project - review, production, and quality control, are loaded into the notebook and are automatically updated.

The **"_id" field created by MongoDB is dropped**. At this stage, the system also **ensures that all date fields are correctly parsed** and converted into a consistent date format.

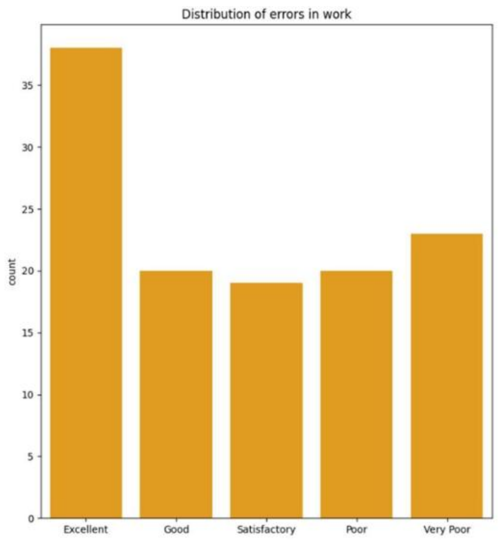
After the individual datasets are prepared, the review and production tables are merged using a **left outer join** with **job case id** as the key. The **quality control table** is then **joined using** the same **job case as a foreign key**. If a job has been completed in Production but hasn't reached Quality Control yet, the merged table will show "NaN" (empty) for the QC fields. This process results in a single consolidated table containing the complete workflow data for each job case.

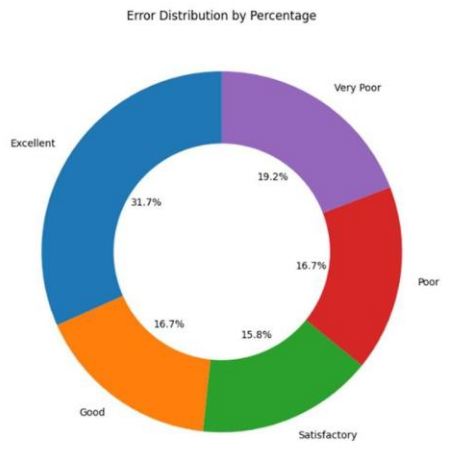
4.2 Data Analysis



Dominant Sectors: **Finance and Retail** represent the largest portion of the business, each accounting for **23.3%** of the client background.

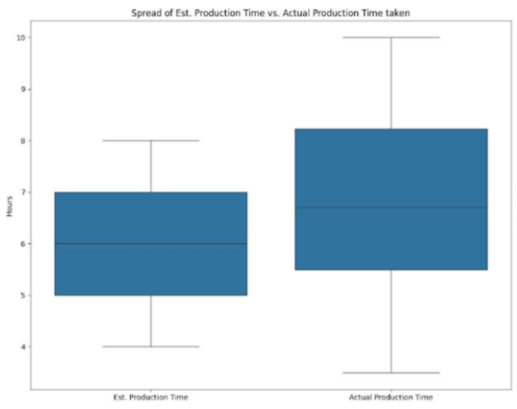
Supporting Sectors: **IT and Healthcare** each account for **20.0%,** while Manufacturing is the smallest client segment at 13.3%.

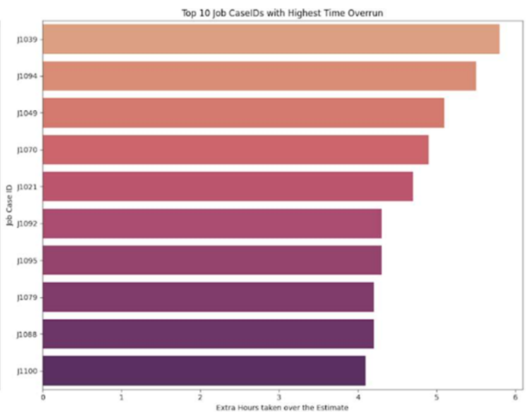




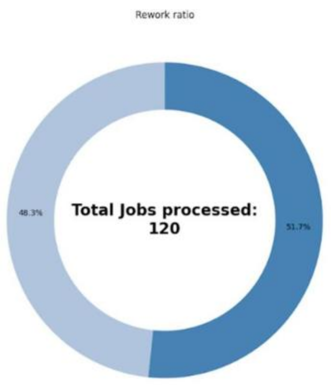
While **31.7%** of work is rated as “Excellent” a significant portion is struggling, with **16.2%** rated as “Very

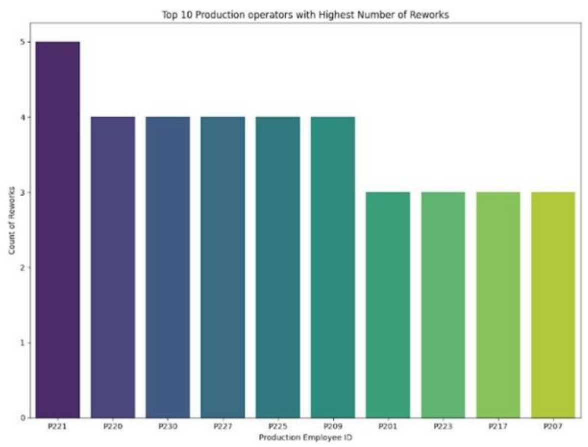
Poor” and **16.7%** rated as “Poor”.





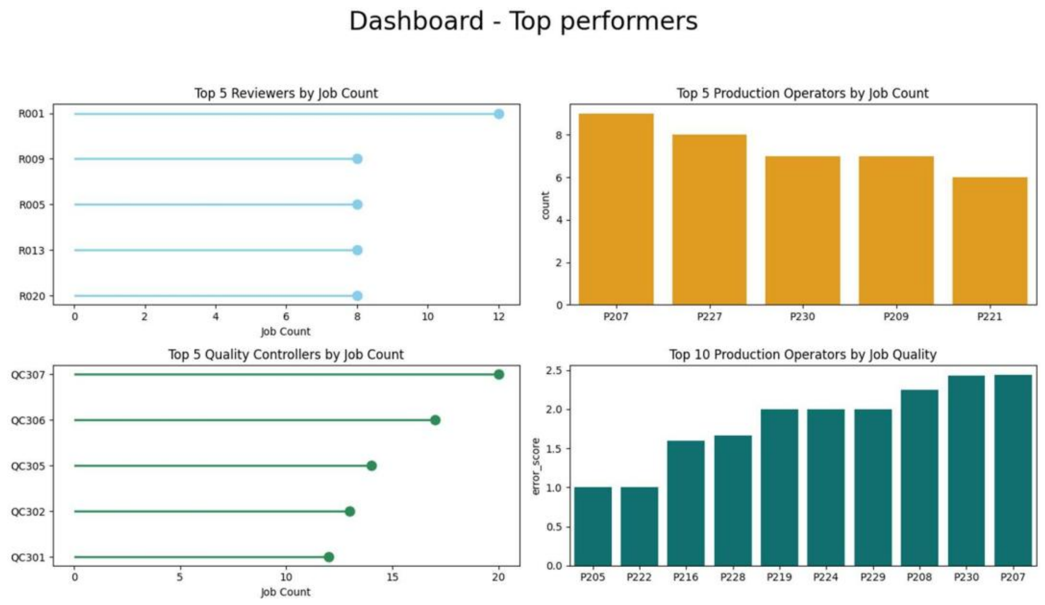
The median of **Actual Production Time** is nearly 7 hours which is consistently higher than the median **Estimated Production Time** of 6 hours. Certain jobs show extreme overruns, with **J1039** exceeding its estimate by nearly **6 hours**, followed by **J1094** with an overrun of approximately **5.5 hours**.





Out of **120 total jobs processed**, over half (**51.7%**) required rework. This suggests only about **48.3%** of jobs are “first-time right”.

Operator **P221** has the highest number of individual reworks of **5 cases**.



QC307 is the primary driver of throughput with **20 jobs** completed which significantly higher than their peers.

R001 leads the Reviewers with **12 jobs**, while the rest of the top five are tightly clustered at **8 jobs** each.

Operator **P207** leads production volume with **G jobs**.

5. Reflection

5.1 Strengths of the Design and Implementation

One thing that worked well was the overall structure of the workflow. Dividing the process into clear stages helped keep the data organized and made it easier to work with during analysis. At the same time, linking the stages through a common identifier allowed the full workflow to be analyzed without making the design complicated.

MongoDB as a database solution provides great schema flexibility if we need to scale-up the design. It allows data to be stored in a natural, readable format that is easy to work with which directly translates to quicker and simpler development. It handles large amounts of data well, can grow as the system grows, and continues working even if one part fails.

The combination of MongoDB and Python also worked effectively. MongoDB handled data storage and structure well, while Python made it easier to perform transformations, aggregations, and visual analysis. This separation of responsibilities helped keep the implementation simple and flexible.

The data preparation steps were another positive aspect. Converting raw values into more meaningful formats, such as quality categories and numeric indicators, made the data easier to interpret and use for reporting.

5.2 Aspects That Were More Challenging Than Expected

One challenge was designing the data model so that it stayed clear while still working well across all stages of the workflow. Making sure the relationships were simple and not overcomplicating everything took more effort than expected.

Another difficulty was preparing realistic data and working with Python to transform and align it correctly with MongoDB. Small issues during data handling and transformation required extra time to resolve.

5.3 Areas for Improvement with More Time

With more time, the data model could be expanded by adding a few more attributes to better describe how work moves through the workflow. For example, attributes related to intermediate status changes, priority levels, or time gaps between stages would help provide clearer insights into delays and overall process flow, while keeping the basic structure unchanged.

The data itself could also be made more realistic. Using larger datasets or simulating incremental updates instead of static samples would better reflect real operational conditions and allow performance and scalability to be tested more effectively.

From the Python side, the implementation could be improved by making the code more structured. Separating data extraction, transformation, and analysis into smaller, well-defined functions would make the code easier to maintain and reuse. Adding simple validation and logging would also help handle data inconsistencies in a more reliable way.

Overall, these changes would improve flexibility, scalability, and maintainability while preserving the core design of the solution.