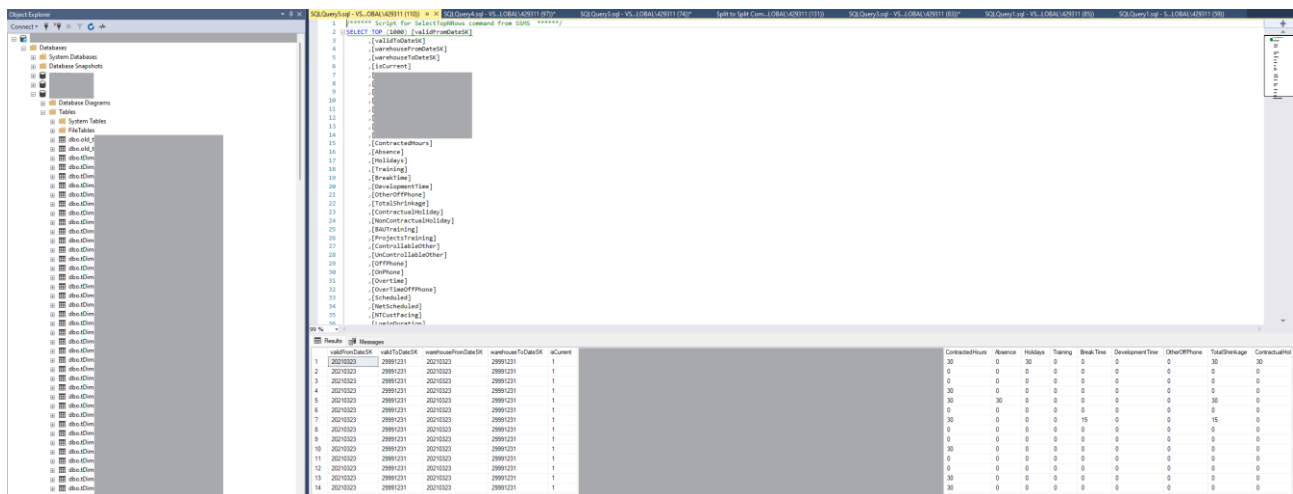


Portfolio 1:

My role in my department is a Data Engineer who contributes to an ETL that pulls data from various sources and transforms it into a star schema data warehouse that feeds BI to stakeholders.

1. Monitoring Data Stores for Performance and Availability

Within my role, the primary data stores we use are on premises, though various teams are piloting migration to cloud-based solutions. The primary data store method we use is SQL databases as the majority of data is simple text, numeric or boolean data relating to customer details, transaction and contact histories (*see fig 1*). We do use NoSQL databases for storing and accessing non-standard data such as images and audio files. Specialist data transfer services, such as Base24 are used for inter-bank communications (withdrawals, payments at POS) for their low latency and interoperability.



The screenshot displays a SQL Server Enterprise console. On the left, a tree view shows the database structure, including 'Staffing'. The main window shows a query window with a SELECT statement and a results grid. The query is a complex JOIN of multiple tables, including 'ContractedHours', 'Absence', 'Holiday', 'Training', 'BreakTime', 'DevelopmentTime', 'OtherOffHours', 'TotalOffHours', and 'ContractualRate'. The results grid shows 16 rows of data, with columns for 'ContractedHours', 'Absence', 'Holiday', 'Training', 'BreakTime', 'DevelopmentTime', 'OtherOffHours', 'TotalOffHours', and 'ContractualRate'.

ContractedHours	Absence	Holiday	Training	BreakTime	DevelopmentTime	OtherOffHours	TotalOffHours	ContractualRate
30	0	30	0	0	0	0	30	30
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
30	30	0	0	0	0	0	30	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Figure 1- SQL Database with staffing data

Data that is kept strictly for audit purposes, especially aged call recordings, are archived as there is no requirement for immediate retrieval of the data and this frees system resources for more immediate requirements.

We use remote storage providers such as Microsoft Sharepoint for documentation as performance is not critical for this task.

We also use distributed file systems for ETL package deployments, including version control (*see fig 2*). This is one of the more flawed tools and I believe we would be better served by migrating to Git for more robust CI/CD and version control.

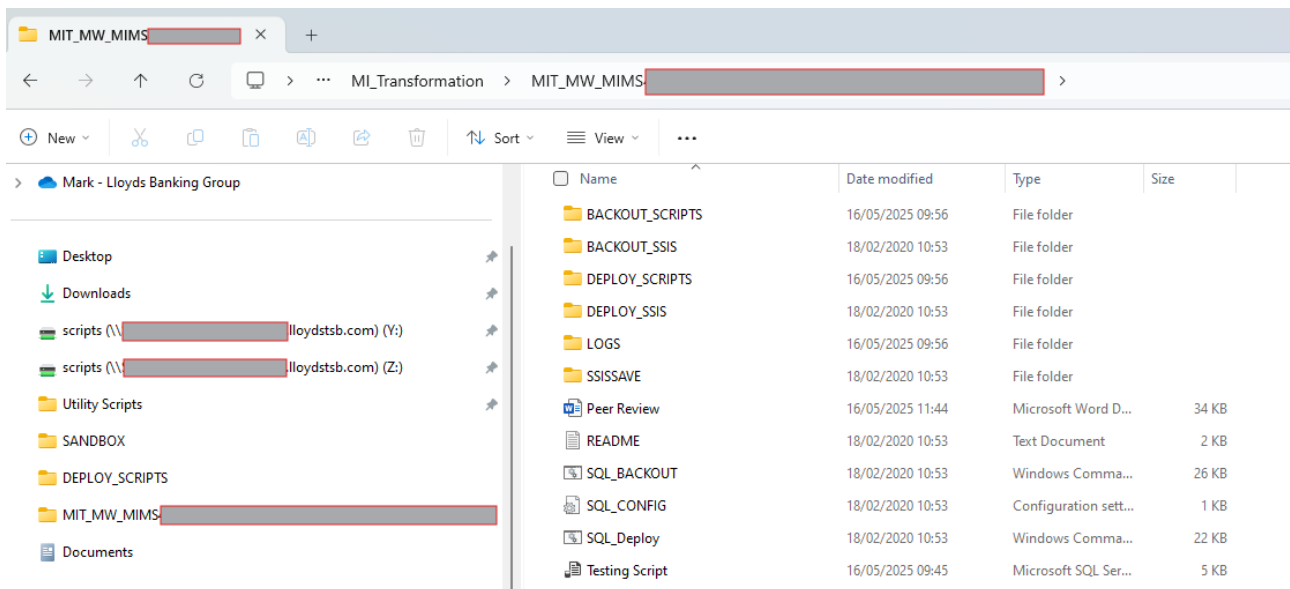


Figure 2 - Distributed File System in use for package deployment

2 . Data Normalisation

Data Normalisation is the process of organising data within a database to reduce redundancy and improve data integrity. It involves structuring data into multiple related tables according to a set of rules, typically following normal forms (from 1NF to 5NF). These forms define how data should be grouped, ensuring each piece of information is stored only once and in the most logical place.

Reducing data redundancy is achieved by eliminating repeated data. For example, rather than storing customer information in every order record, a normalised design stores it once in a customer table and references it via a key. This saves storage, avoids duplication, and reduces the risk of conflicting entries.

Protecting against inconsistent dependencies means ensuring relationships between data remain logically sound. In non-normalised databases, updates in one area (e.g., a customer's address) may not cascade correctly if data is duplicated. Normalisation ensures updates occur in a single place, avoiding errors.

Overall data integrity and quality benefit because normalisation enforces consistency, reduces anomalies in insertions, deletions, and updates, and simplifies validation. The result is a database that remains accurate, scalable, and easy to maintain.

An example from my business area is how we manage case data for quality testing of customer interactions. Rather than storing full details of the staff being tested (or the testers themselves) with each case, we use keys to link to a central StaffManager dimension (*see fig 3*). This contains shared attributes like competency levels, team structure, and manager (via another key).

Figure 3- Table structure showing Foreign and Primary Keys as part of Star Schema

The benefits of this are:

Redundancy: Each colleague only needs one entry in a single dimension table, rather than repeating that information in every row of the fact table.

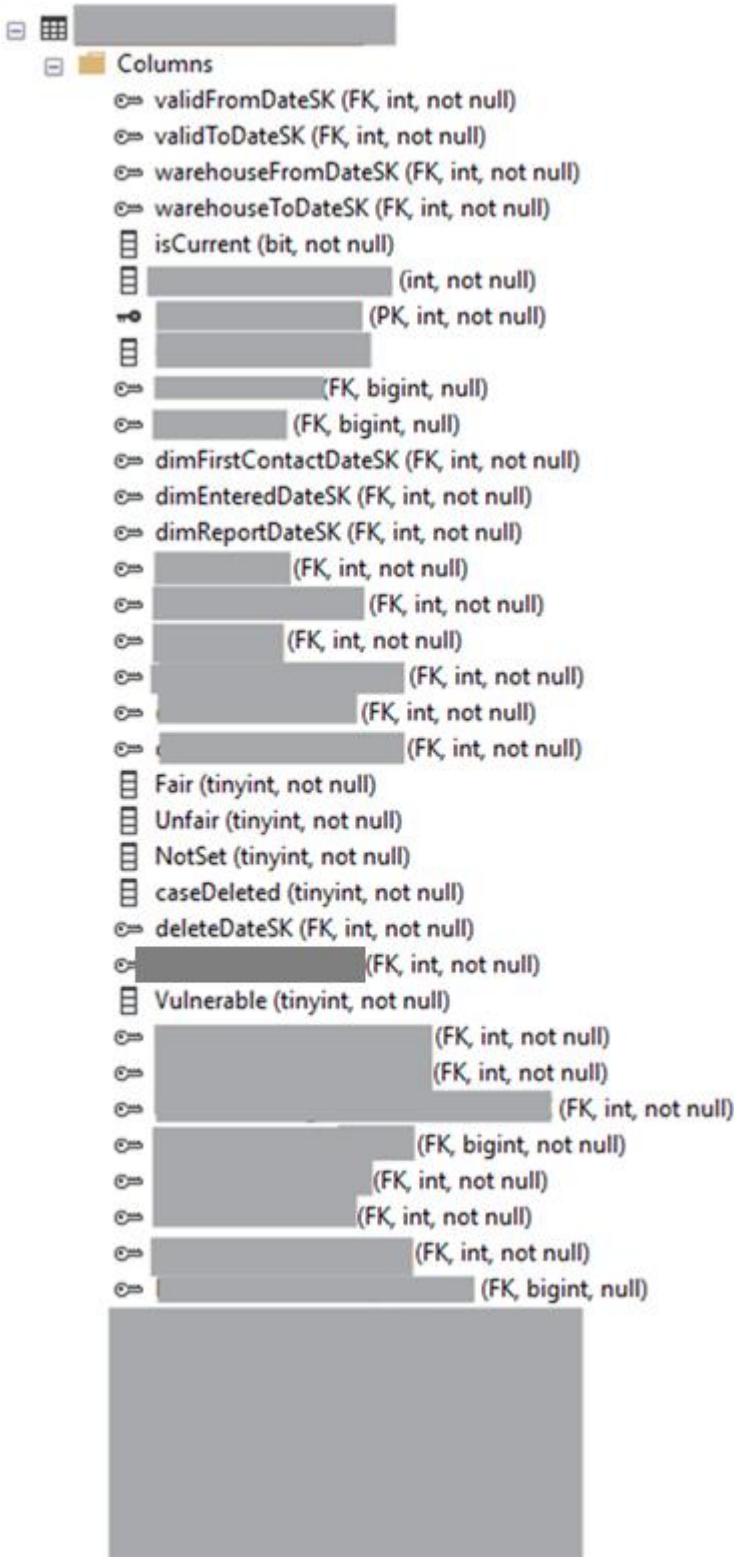
Inconsistent Dependencies: Updates to team structures or managers occur in one place, ensuring all linked records stay aligned.

Data Integrity and Quality: Data need only be checked in a single location and we can be sure it is correct across the entire database rather than having multiple copies that each need checking.

3. Data Quality and Risk Management

Data Protection/Security: Most of our data comes from automated logging (e.g., dateEntered, dateCreated), controlled selections (Yes/No, Card/Mortgage/Loan), or pattern-locked entries (account numbers, values). However, free-text fields can contain unpredictable content. To manage this, we categorise every field based on its maximum potential sensitivity (Open, Limited Internal, Limited External, Sensitive, Highly Sensitive). Any downstream use inherits this classification to ensure appropriate safeguards.

Data Drift: We recently audited our data mapping tables after identifying static reference tables as a source of silent failure. Many relied on manual stakeholder updates that were missed, resulting in data drift. We now conduct regular reviews, automate mappings where possible by linking directly to trusted raw sources, and challenge manual dependencies.



Incomplete or Inaccurate Data: We apply validation rules during ingestion to ensure required fields are populated, values fall within expected ranges, and formats (e.g., dates, IDs) match required patterns. We flag exceptions early to prevent propagation of poor data.

Human Error & Integration Failures: To guard against accidental corruption or loss during data transfer, we implement row-count checks and automated alerts for anomalies such as abnormally high or low intake volumes. Scheduled jobs are monitored, and fallback logic is in place to recover from interruptions.

Data Quality Practices: We profile incoming data to identify null rates, outliers, and duplicate records. Where issues are found, we apply data cleansing; standardising formats, correcting known errors, and isolating unreliable records. All transformations are logged with metadata, creating an audit trail (see fig 4).

Why it Matters: Decisions made on poor-quality data can have real-world consequences to the bank, our customers and our colleagues. Trustworthy pipelines enable accurate reporting, fair colleague evaluations, regulatory compliance, and customer outcomes.

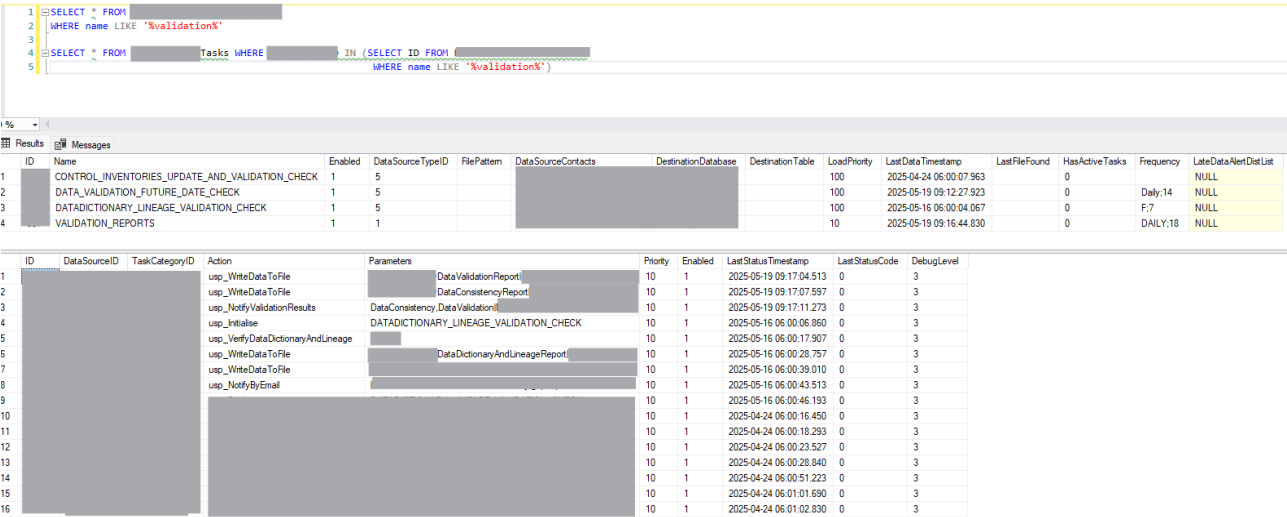


Figure 4- Data Validation and cleansing automated processes

4. Data Ingestion Frameworks and Optimisation

In my team, data ingestion is handled via overnight batch processing. Each night, our ETL pipeline pulls data from 23 RAW data sources and processes it through an ETL layer, then into a structured datamart with MRT (Modelled Reporting Tables) and REP (Reporting) layers. These support Power BI dashboards used by the business the following day.

We use batch processing because our data changes frequently before finalisation—for example, open cases may be updated multiple times before closing. Streaming or on-demand ingestion would risk surfacing incomplete or misleading information. Additionally, the data is not generated evenly throughout the day; activity largely ceases after business hours, making overnight processing efficient and minimally disruptive to shared server resources.

This method also supports complex transformations and joins that benefit from processing complete datasets rather than partial or constantly updating streams.

The trade-off is scalability and timeliness. Our current pipeline runs from midnight until around 9am. Any batch delay can postpone dashboard availability and affect business operations (*see fig 5*).

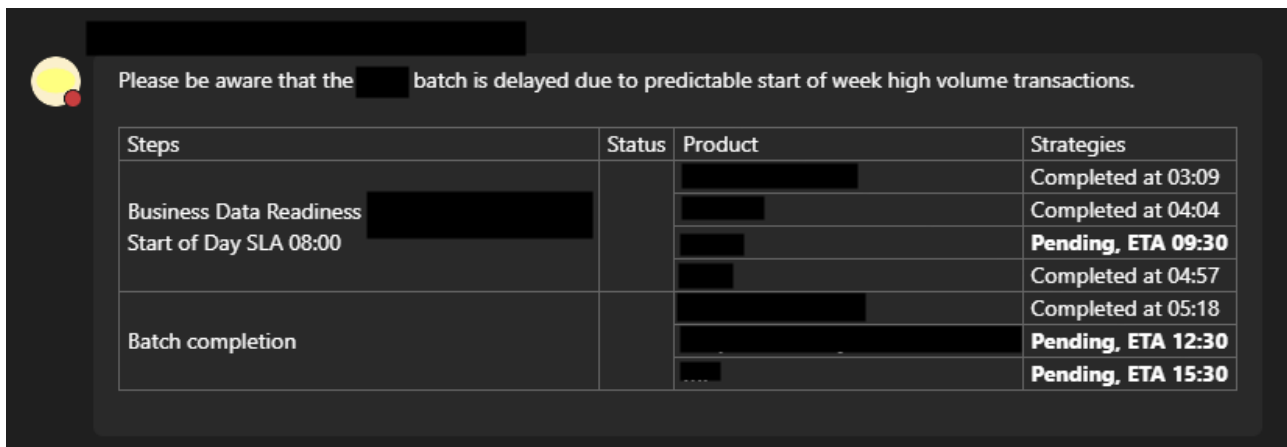


Figure 5- Batch delay caused by uneven resource demand (high start of week volumes)

We're currently piloting a migration to a cloud service provider, where elasticity in compute resources promises faster processing. In a cloud environment, resource-intensive ETL jobs can scale horizontally, completing faster without increasing total cost, provided usage is efficiently managed. However, cloud services introduce cost unpredictability; where in-house infrastructure has a fixed cost, cloud billing varies based on compute and storage used.

An example of efficiency improvement will be migration of our dataset from an internal SQL server to the cloud-based warehouse. By batching the move and leveraging parallel processing, we expect to improve reliability and support quicker turnaround for quality and performance reporting.

Extension: Descriptive, Predictive and Prescriptive Analytics

Descriptive analytics explains what has happened. It uses historical data, visualisations (e.g. charts, dashboards), and summary statistics to identify trends and performance patterns. It's used to monitor KPIs and understand past behaviours.

Predictive analytics forecasts what might happen. It applies techniques like regression analysis, classification, and machine learning models to identify likely future outcomes based on historical data. For example, predicting customer churn or future sales volumes.

Prescriptive analytics recommends what actions to take. It uses optimisation algorithms, simulations, and decision models to evaluate different scenarios and suggest the best course of action. This might include resource allocation or pricing strategies.

Descriptive analytics are the most commonly used data in my department as we provide BI visualisation via canvases to our stakeholders to describe performance against KPIs (*see fig 6*). This might be something like 'how many agents successfully identified a vulnerable customer'.

Predictive analytics are occasionally used to support planning and operational decision-making. One example is the forecasting of long-term transaction volumes at ATMs. By analysing historical usage patterns and identifying seasonal trends, we can predict when individual machines are likely to experience high or low demand. These insights are used to inform cash replenishment schedules,

helping to ensure machines are stocked ahead of busy periods and avoid over-servicing during quieter times. This improves both customer experience and operational efficiency by reducing unnecessary visits and minimising the risk of outages.

Prescriptive analytics are used in conjunction with the previous two to drive business decisions. An example would be the value, type and location of foreclosed houses to be assigned to specific agents to sell based on those agents' previous performance selling houses of similar types, values or locations. e.g. one agent may have been better at selling <£200k houses and another might have been better at selling houses in West Yorkshire.

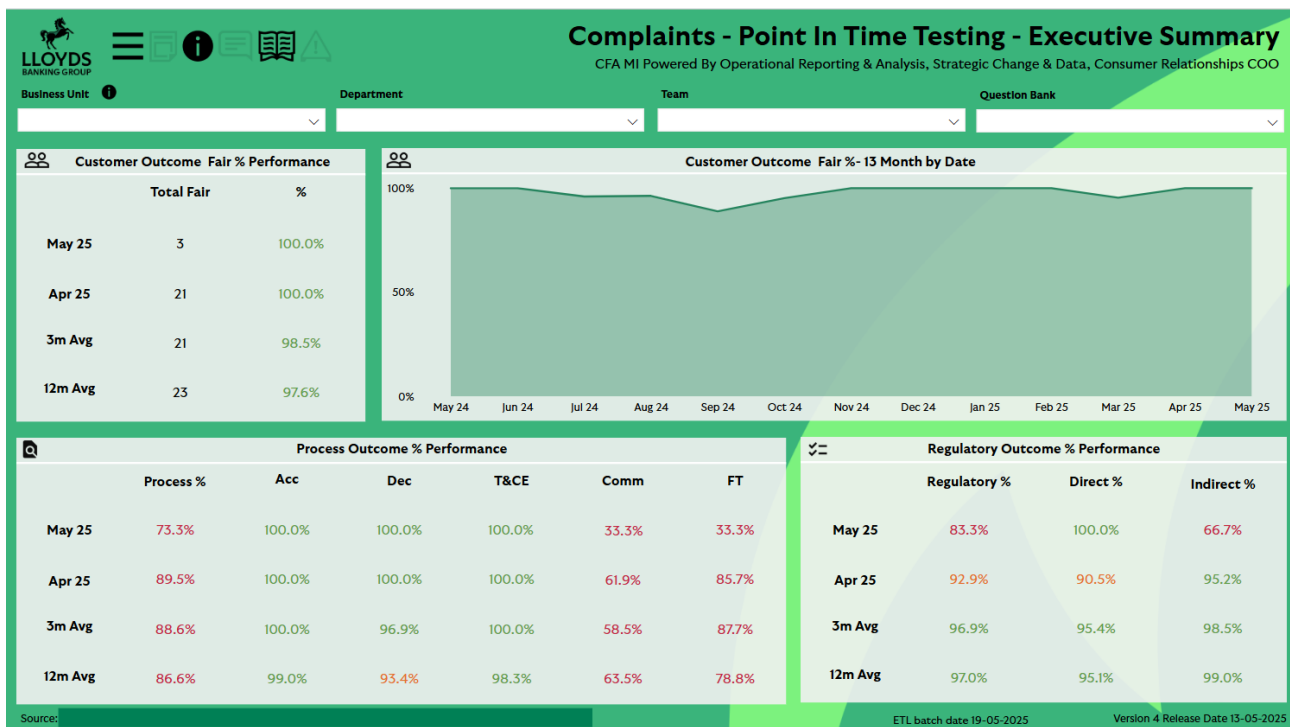


Figure 6 - Power BI Canvas showing performance against customer complaint KPIs