Introduction to Applied Transaction Analytics

Table of Contents

Introduction to Applied Transaction Analytics

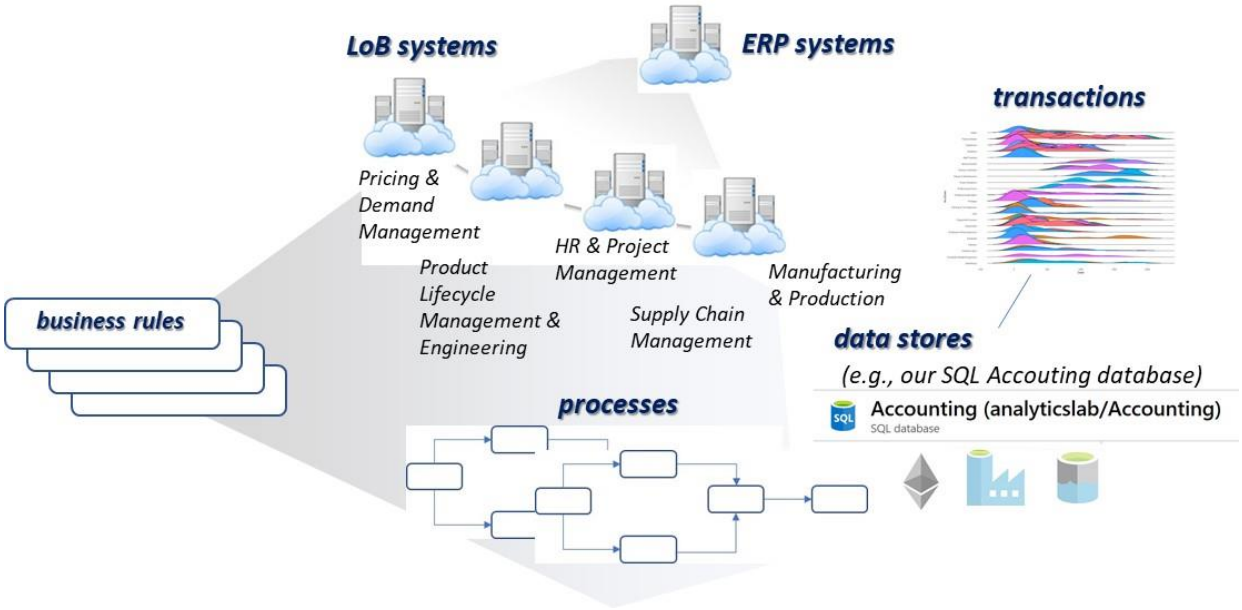
The term “Business Intelligence” was used in 1865 to describe the acquisition of banking information for competitive advantage. The term “Analytics” was used by Aristotle in 350 BC to describe a system for logical ***reasoning***. Analysts today have a lot more data to comprehend, so they utilize a range of tools and technologies to help. But these *tools just amplify the analysts’ reasoning abilities - or inabilities*.

Applied Transaction Analytics *(sections Acct 7373 & 7374)* focuses on the role of the ***analyst***, broadly defined to include corporate analysts, tax analysts, auditors and consultants. Across all of these roles, the technologies and methodologies are the same *(e.g., auditors may use projections, while corporate analysts use forecasts and consultants use predictions – past, present and future are all just values of a time variable – the methodologies and models are the same)*

These analysts work in ***transaction environments***, and apply an ***analytics process*** to measure business activity and performance, explain and confirm causes and drivers, identify opportunities and risks, plan and budget future periods, and render inferences and opinions about the state of the business. So, we begin this introduction by describing the transaction environment and the analytics process. We’ll conclude with an exercise that applies a familiar tool *(Excel with Power Pivot and Power BI)* to analysis of transaction data. The purpose is to give you a feel for the Analytics Process, get comfortable with querying relational data, and understand data description using distribution models. Later, we’ll take the “training wheels” off and dive into serious modeling.

Transaction Environments

Transactions are the footprint of business execution – the evidence. In mid-large corporations, transaction volume can be high *(a baseline requirement for many systems is 5,000/sec, and many companies have hundreds of business systems - seriously)*. The following briefly describes the typical transaction environment:



ERP Systems. ERP systems usually host the *"top level"* accounting processes, and often, functional processes *(functional processes those that support the "value-add" activities of a business – i.e., make stuff / sell stuff)*. ERP is an acronym for Enterprise Resource Planning – the benefit of ERP being that the system controls all the resources across the enterprise so business operations are fully integrated. This theoretically leads to efficiencies and *standardization* of processes.

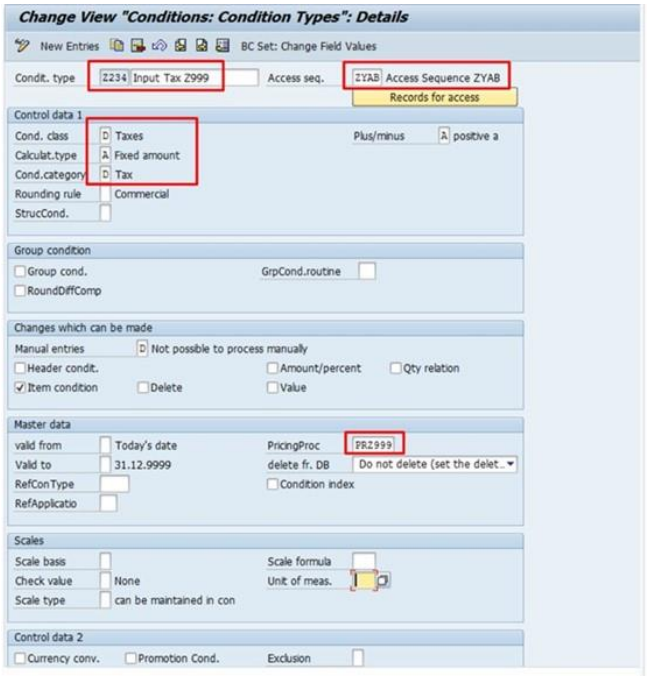
LoB Systems. *Standardization* of processes is a good news/bad news proposition. While there are efficiencies to be gained, there are differentiators to be lost. So, companies purchase and configure Line-of-Business *(LoB)* systems *(e.g., Emerson PlantWeb, Bloomberg trading platforms, Aegis FactoryLogix)*, or build completely custom systems *(e.g., JP Morgan's customer site)*.

Regardless of whether processes are hosted in the ERP or LoB systems, accounting transactions are summarized into journal interfaces and updated to the general ledger. These integration journals are highly aggregated, which means *source detail is lost at the GL and Subledger level*. So, analysts must link *(or "tie")* GL transactions to the original source transactions.

Processes can range from manual and improvisational *(e.g., customer sales meeting)*, to rigorous processes that are embedded in systems *(e.g., order entry in SAP)*. Analysts often focus on the End-to-End *(e2e)* processes that create business transactions *(e.g., Order-to-Cash, Procure-to-Pay)*. Many processes execute *across* systems *(project resources might be assigned from the HR system, scheduled in a project management system, and billed in the ERP).* Analysts should be able to trace an accounting transaction back to the originating process and understand the steps and authorizations that created it *(e.g., assessments and tests of controls in audit)*.

Business Rules (BRs) control the *execution of processes* and the *routing of transactions (e.g., you can't create an invoice in SAP if work wasn't approved in the Project system)*, as well as the *calculation of transaction amounts (e.g., a pricing condition in SAP applies discount and tax to a sale)*. In other words, BRs do a *lot* - and analysts need to understand what BRs are involved in a process, how they work, and who/what is authorized to run them.

BRs can be configured in simple screen - *e.g., pricing "condition" in SAP…*

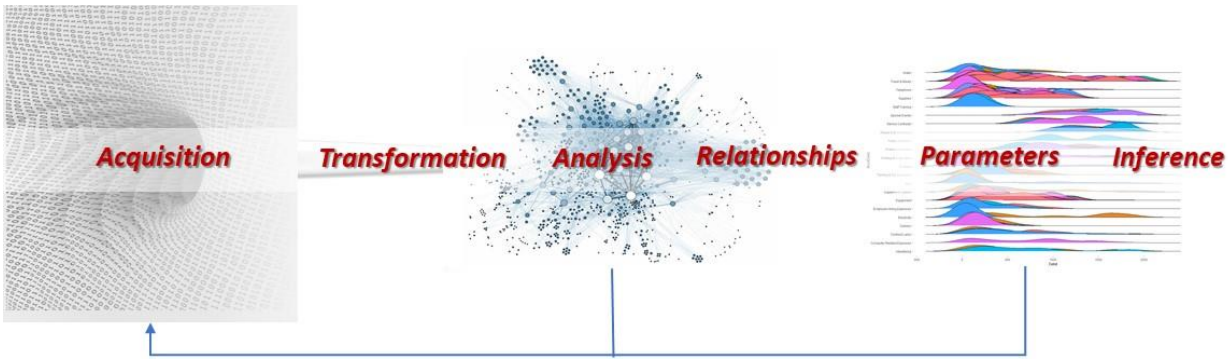


… or coded into custom processes and applications with thousands of lines of code. There are even *rules engines* for particularly dynamic, complex environments – especially those with *cross system processes.* Analysts should be able to trace accounting transaction amounts through the business rules, and explain the conditions that created them. In today's business environment, BRs encapsulate ***contracts*** *(contracts are not on paper anymore)*, and the logic and terms are complex and *dynamic*.

Data Stores. All of these ERP and LoB systems must store data somewhere. Overwhelmingly, data are stored in SQL databases, which are predominantly "SQL Server" and "Oracle" flavors *(SQL is an acronym for Structured Query Language, and it's pronounced "See-quel")*. SQL Server is a rich, mature, reliable, ***platform*** *(a platform is a cluster of technologies that provide a base on which to build applications)*, that is optimized for transaction processing *(note: There are other platforms that may be important to analysts' work, outlined in* ***appendix 1*** *– these are FYI - we will not be using these in the course).* We will focus on SQL databases– that will cover most of your data acquisition skills.

The Analytics Process

Now that we have an idea of where the Analyst works, let's turn our attention towards understanding the Analytics Process:



Note that the overall process is iterative – we often return to acquisition and analysis to add data, and reevaluate relationships and data structure.

Data Acquisition and Transformation

Data Acquisition and Transformation is ~80% of the effort in Analytics. The previous section introduced the transaction environment - but how do we get the data? Data is acquired by issuing a query to SQL Servers that generally follows this logic:

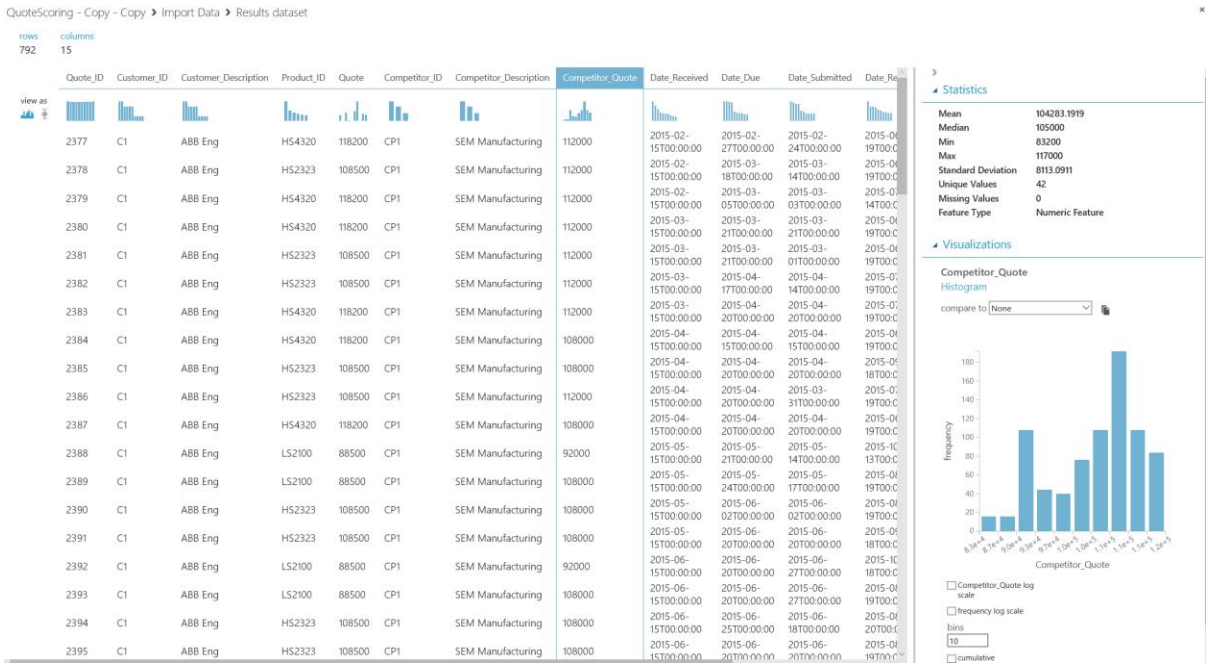
***SELECT** (data) **FROM** (data source**s**) **WHERE** (data is filtered based on some condition – e.g., x > y)*

Notice that the query references multiple data source***s***. Transaction systems are designed to optimize database performance, consistency and flexibility, so the data are often "normalized" to eliminate redundancy. The result is that data will often be "spread out" among multiple locations/tables with relationship links tying them together. We will spend *considerable* time understanding this ***relational model.***

Data usually has to be ***transformed*** to be useful: one department's Product 1 is another's Prod-A; one departments account FA20 is another's 10045; one department's period 1 is another's 2019-01-01. At the end of transformation - data are consistent, and values are not treated as variables *(e.g., values like Jan-19 or Revenues are not columns – the column variables would be "Date" and "Account")*. We often use the term *"**tidy data**"* for consistent data structure.

Data Analysis

We usually have to spend some time *"exploring"* the data and relationships, by sorting, summarizing, counting, comparing, visualizing… to understand it enough that we can describe it. The image below describes how an analytics environment might treat data coming in:



(the screen above is from Azure Machine Learning which provides a description of the data being consumed prior to modeling – it shows the data types, distribution visualizations, and distribution parameters: mean, median, min, max, variance…)

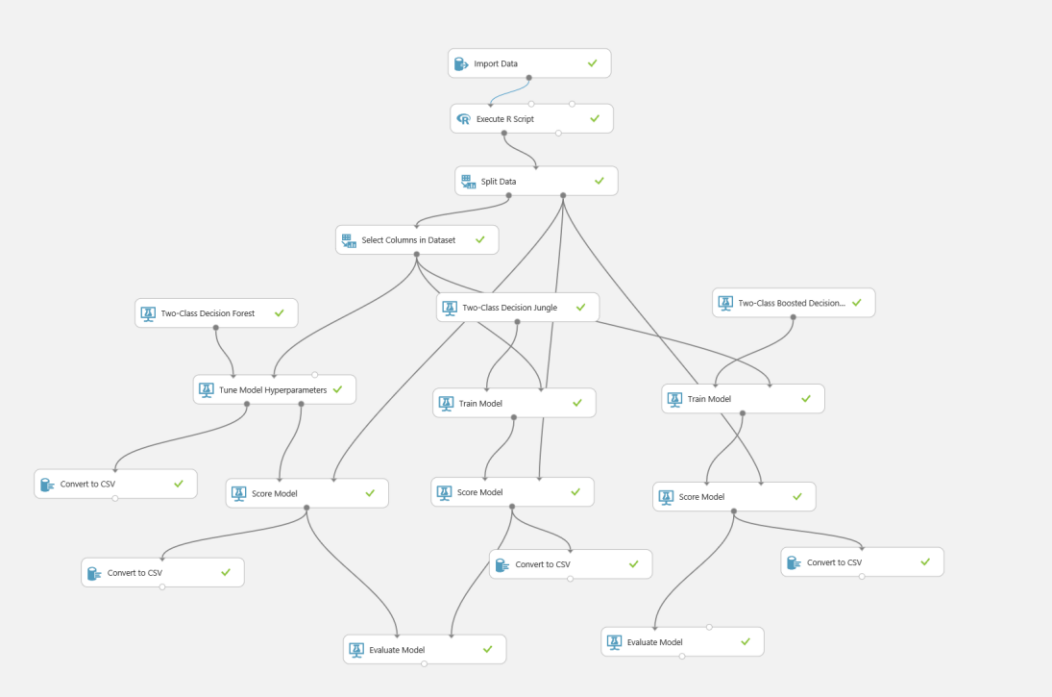
Notice that this analytics environment places a lot of emphasis on the distribution, by default. This the level of detail we need to make decisions about ***model selection and approach***.

Modeling

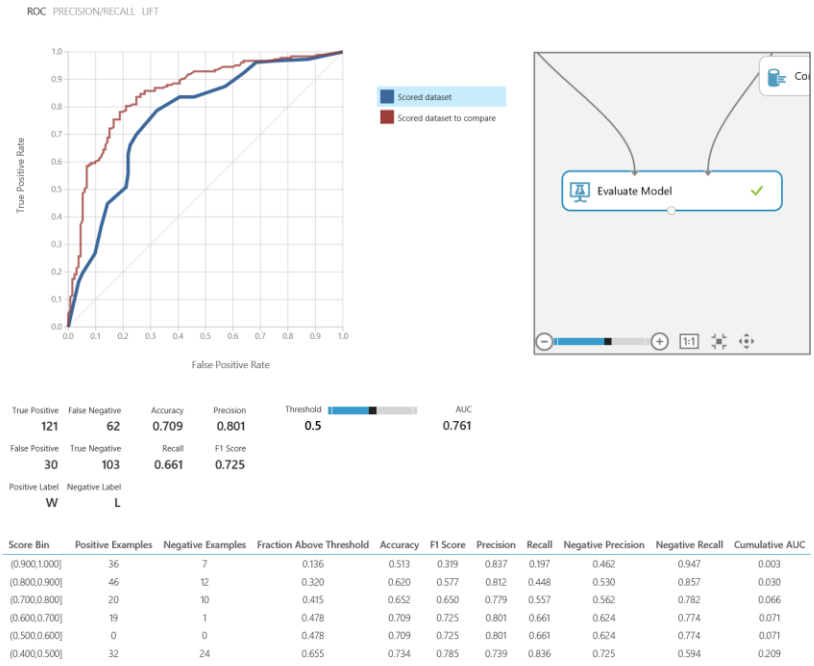
The core competency of analytics is ***modeling. Models are just equations that produce parameters that quantify data relationships***. We then use those parameters for inference – i.e., to *express opinions,* or *interpret evidence*, etc. Sometimes, those parameters are intractably complex *(in which case, we can utilize machine learning algorithms and AI to produce projections)*, and we call those ***non-parametric*** models *(we will see in DA2 that "non-parametric" is often a misnomer – there are actually LOTS of parameters)*. And sometimes the parameters are more easily defined and understood – ***parametric*** models.

Non-parametric and parametric models have strengths and weaknesses, and we will use both throughout these courses. In general, as models become more complex *(adding parameters)* they increase in *flexibility and precision,* but decrease in *interpretability (i.e., usefulness for explaining relationships and causation)*. Note: be sure and read chapter 2 of ISL. Most Analysts use a combination of parametric and non-parametric models on the same analysis – each perspective contributes to data description, model selection and tuning, and interpretation.

Modeling can be a very sophisticated process:



(The screen above shows a solution that simultaneously processes data through several models for a grid of hyper-parameter values with various resampling strategies…



Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	36	7	0.136	0.513	0.319	0.837	0.197	0.462	0.947	0.003
(0.800,0.900]	46	12	0.320	0.620	0.577	0.812	0.448	0.530	0.857	0.030
(0.700,0.800]	20	10	0.415	0.652	0.650	0.779	0.557	0.562	0.782	0.066
(0.600,0.700]	19	1	0.478	0.709	0.725	0.801	0.661	0.624	0.774	0.071
(0.500,0.600]	0	0	0.478	0.709	0.725	0.801	0.661	0.624	0.774	0.071
(0.400,0.500]	32	24	0.655	0.734	0.785	0.739	0.836	0.725	0.594	0.209

…and then analyzes model performance, parameters and accuracy)

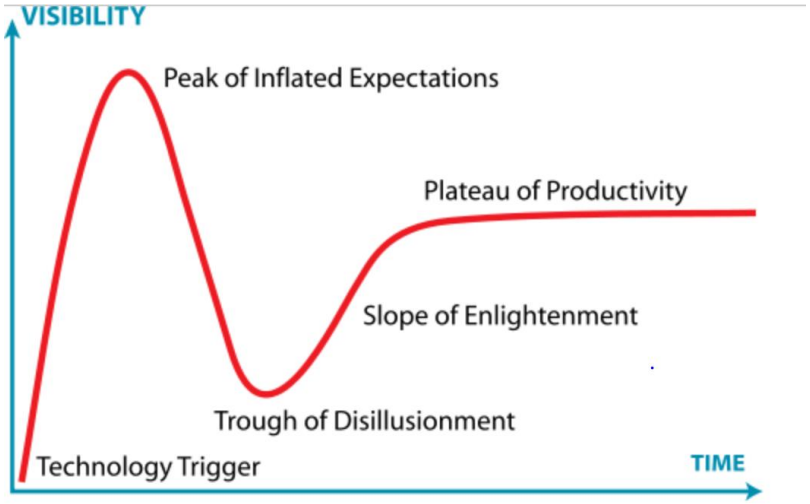
Or they can be simple *(e.g., you will use the Excel "trend line" feature to create a regression model in this exercise – this is as simple as it gets – it's also extremely limited)*.

Additional Thoughts and Expectations

As we study Applied Analytics, we will walk through the Analytics Process over and over again – adding complexity, tools and most importantly, understanding as we go. These process is based on ***statistical concepts*** *(note that one of your textbooks is titled "Introduction to Statistical Learning")*. We build on the basics that you had in Intro Statistics, but we will go well beyond those. It's necessary that you understand the statistical foundations to be able to design solutions and diagnose problems. ***There's no shortcut***. If you want to review statistical concepts, there are recommended texts in the "Resources" folder on Blackboard.

We will also cover some mathematical concepts. If you look up "Analytics" on Wikipedia, you'll see: "…there is extensive use of computer skills, ***mathematics and statistics***…" Our coverage of mathematics is not deep – we'll use linear algebra for understanding data analysis and regression modeling, vector, and light calculus *(integration for understanding distributions and differentiation for optimization)*. There are math cheat sheets in the Resources folder on Blackboard – a full review of algebra, linear algebra, geometry and calculus shouldn't be necessary.

Many business organizations are currently trying to digest Analytics. Like many technology cycles, there's a tendency to look for shortcuts, and to get excited before the technologies are truly understood. This happens so often that Gartner *(a prominent research firm)* came up with the "hype cycle" to describe technology adoption:



For many Analytics methodologies and technologies, we are comfortably in the "Plateau of Productivity" stage *(e.g., JP Morgan was building algorithmic trading applications in the 90's – and most of those are still optimal)*. Other technologies *(e.g., Blockchain, AI)* are in the "peak of inflated expectations" stage.

The final thought here is: Technologies and Tools won't solve your problems. Aristotle's first definition of Analytics prevails: it's a system of logical ***reasoning***. You'll have to think.

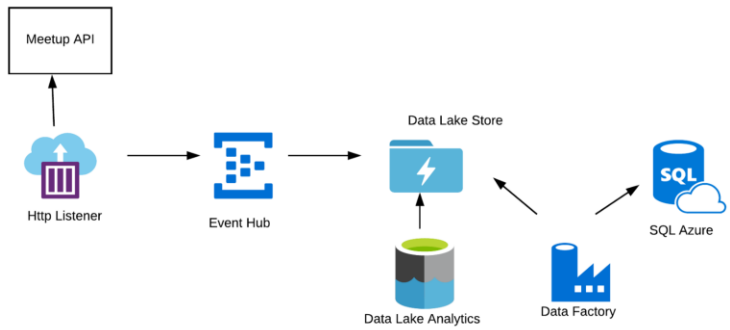
Appendices

Appendix 1 –Transaction Platforms

This introduction outlines some of the primary reasons we learn SQL data acquisition in our courses. There are ***other transaction platforms*** that are currently *(or soon to be)* important in analysts' work:

1. ***Data Lakes*** are databases that store really large unstructured data *(social media, unmanaged csv / Excel files, BOLBs…)*. Data Lakes can be accessed using SQL so *SQL is a good core skill* for Data Lakes too.

2. ***Event Hubs and Data Factories***. Data Lakes will often store device data *(e.g., onsite sensors)* that stream data over the internet. Event hubs receive the data and data factories ***route and transform*** that data to the appropriate destination *(often a table in SQL as shown here)*.



(example event hub / data factory architecture)

The reason analysts need to be aware of these platforms is because there are processes and rules that create transactions here. What kind of processes and rules? Inventory replenishment and relief, retail store checkout, oilfield production, ride sharing… these are the transactions analysts are expected to manage, and these types of applications are exploding.

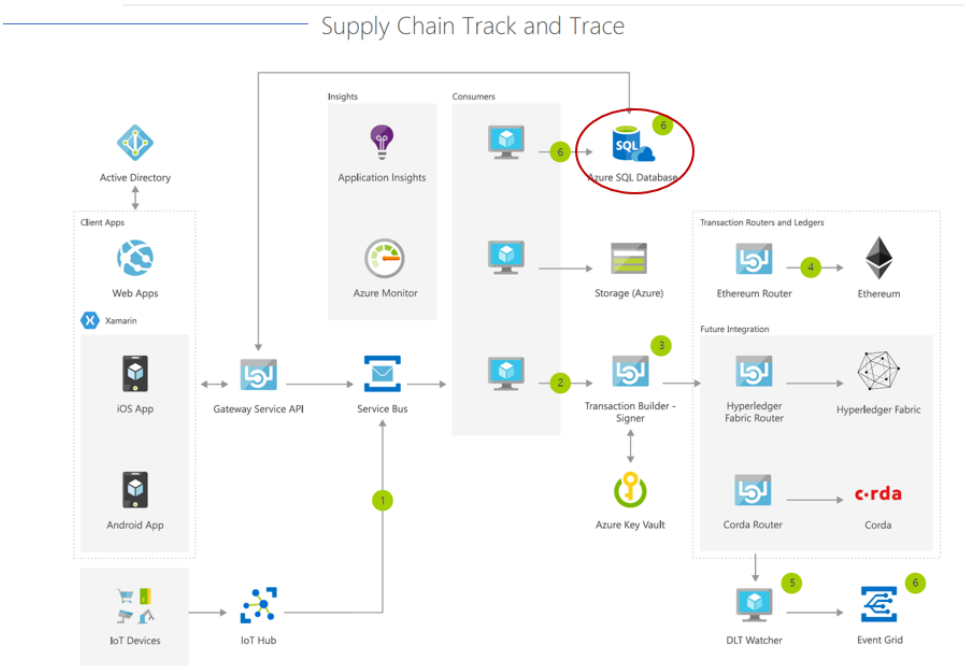
3. ***BlockChains*** are distributed transaction ledgers *(DTLs)* that have some unique capabilities, and transactions are now beginning to trickle into blockchain platforms. Adoption has been slow because there are significant barriers:

    a. *Performance*. Recall that we defined a transaction baseline of LoB systems as 5k/sec above? Well, Blockchains are struggling to exceed 1k/sec and the architecture is inherently limiting *(i.e., unlikely to improve - the process for committing transactions is really secure - and really slow)*
    b. *Extensibility*. Blockchain transactions are often controlled by "smart contracts" *(i.e., a special type of BR)*. Smart contracts are very limited in ability to host logic (the language is very primitive and the architecture restricts complexity). Also, much of the transaction logic is already working in other systems – so the business case to migrate is questionable.

There are benefits too:

a. *Reliability.* Blockchain transactions are immutable and ***atomic*** *(complete)*, so they are "vouched" as soon as they are created *(this is what gets auditors excited – unduly so)*.
b. *Security.* Blockchains and transaction commits are highly distributed (many nodes participate in transactions), so it's nearly impossible for a single party to alter or create fraudulent transactions *(they wouldn't be able to get enough nodes to "vote" on it)*.

Regardless of the barriers and benefits, there are some strong use cases that justify the effort. Governments (e.g., Russian and Estonia) are implementing blockchains to record business transactions and assess tax (reducing tax evasion). The reliability and security of blockchains is attracting some early applications in healthcare, and there the atomic characteristics encourage supply chain application, as illustrated below:



(example blockchain architecture)

The architecture above is for a pattern to ship refrigerated pharmaceuticals per the ***contract*** *in blockchain. If at any point, the temperature falls below the contract range (IOT device), an out-of-compliance transaction is created. You'll also notice that the architecture routes transactions to SQL Server (or the ERP), so again, SQL skills will give you access to this data (there are also SQL based tools for direct blockchain query).*

Appendix 2 –Analytics Platforms

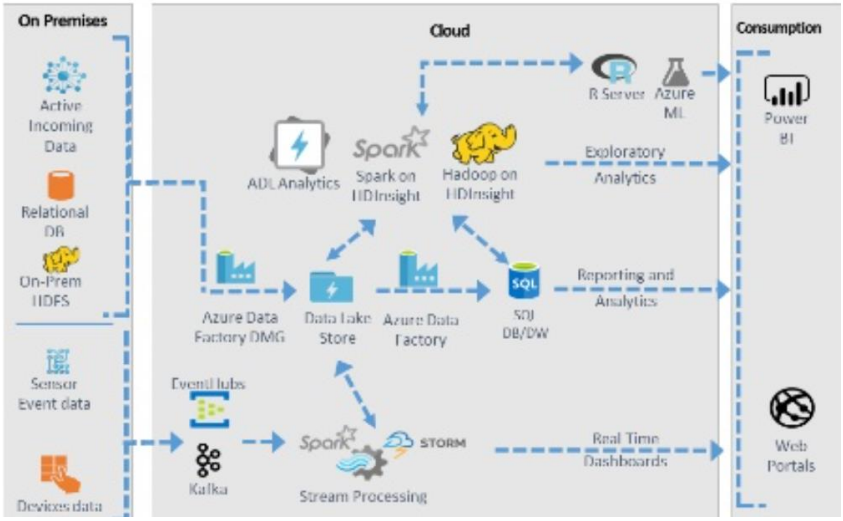
As previously mentioned, the exercise that follows asks you to build a model in Excel. However, Excel is not a modeling tool and very limited data management capabilities.

Enterprise scenarios cannot be modeled in Excel. Power BI and Tableau are not modeling tools either - they are visualization dashboards, which do not provide a basis for inference *(they have very limited value, and often than not, lead to misinterpretation)*.

So, if you want to parametrically describe data and relationships, and draw inferences from the data *(including audit analytical procedures, substantive testing and projection)*, you need a modeling tool. There are several modeling environments that have nice interfaces *(like Azure Machine Learning)*, but like all "user friendly" tools, they sacrifice functionality and extensibility to generalize and simplify. They are also poor learning tools, as the essence of modeling *(the equation)* is often hidden. In other words: they work great when you're in their sweet spot, but that sweet spot is not very big.

The other alternative is to learn an analytics language – of which there are 2: R and Python. We use R for our courses because it is easier for analysts to learn *(generalizing here: Python is more of a programming language where R is more of a statistical analysis language)* and it has a strong base of pedagogical material. Rstudio will support all of the material and problems that we will cover through our courses.

When analysts work in a production transaction environment, R Studio is not enough. Our version of R Server is scalable and can process a lot of data, but when we need to analyze massive amounts of structured and unstructured data, we run into problems: How do we store all that data, query all that data, and process all that data through models? The architectures to deal with these problems scale out the data processing using clusters *(clusters are many, many servers running in parallel)*, and scale out the model processing using Spark clusters *(same idea)*



(the architecture illustrated above is a basic analytics setup with Data Lake, Spark, Hadoop, R Server and Azure Machine Learning). A good