[TECHNOLOGY 2](#_Toc17468461)

[AWS 2](#_Toc17468462)

[HADOOPP 3](#_Toc17468463)

[STREAM 3](#_Toc17468464)

[SPARK 4](#_Toc17468465)

[Core 4](#_Toc17468466)

[SQL 4](#_Toc17468467)

[Streaming 4](#_Toc17468468)

[Mlib Machine Learning Library 4](#_Toc17468469)

[GraphX 4](#_Toc17468470)

[APACHE KAFKA 4](#_Toc17468471)

[DJANGO 5](#_Toc17468472)

[SQL 5](#_Toc17468473)

[Oracle PL/SQL 5](#_Toc17468474)

[T-SQL 6](#_Toc17468475)

[INNER JOIN 7](#_Toc17468476)

[LEFT OUTER JOIN 7](#_Toc17468477)

[FULL OUTER JOIN 7](#_Toc17468478)

[CARTESIAN PRODUCT 7](#_Toc17468479)

[LATERAL JOIN (CROSS APPLY) 7](#_Toc17468480)

[UNION ALL 7](#_Toc17468481)

[UNION 7](#_Toc17468482)

[INTERSECT 7](#_Toc17468483)

[EXCEPT 7](#_Toc17468484)

[PIVOT 8](#_Toc17468485)

[OVER() 8](#_Toc17468486)

[SQL CLR 9](#_Toc17468487)

[TABLEAU 9](#_Toc17468488)

[Workbook Components 9](#_Toc17468489)

[Tableau Interface 10](#_Toc17468490)

[POWER BI 10](#_Toc17468491)

[MONGODB 10](#_Toc17468492)

[Microservices 11](#_Toc17468493)

[API 11](#_Toc17468494)

[12 FACTORS 11](#_Toc17468495)

[REST 12](#_Toc17468496)

[DOCKER 12](#_Toc17468497)

[KUBERNETES 12](#_Toc17468498)

[PYTHON 13](#_Toc17468499)

[LUIGI 13](#_Toc17468500)

[DASK 13](#_Toc17468501)

[PANDAS 13](#_Toc17468502)

[SQL ALCHEMY 13](#_Toc17468503)

[R 13](#_Toc17468504)

[Data Structure 13](#_Toc17468505)

[JAVASCRIPT 13](#_Toc17468506)

[SCIKIT-LEARN 13](#_Toc17468507)

[TENSORFLOW 13](#_Toc17468508)

[PYTORCH 13](#_Toc17468509)

[MACHINE LEARNING 13](#_Toc17468510)

[Regression 13](#_Toc17468511)

[Boost 13](#_Toc17468512)

[Deep Learning 13](#_Toc17468513)

[Feature Engineering 13](#_Toc17468514)

[CIBC Control 14](#_Toc17468515)

[CIBC 20 Services (Financial) 14](#_Toc17468516)

[CIBC Processes (FCU) 14](#_Toc17468517)

[CIBC 26 Processes (OPC) – 113 Sub-processes 14](#_Toc17468518)

[AGILE Audit Lifecycle 16](#_Toc17468519)

[AGILE AUDIT 16](#_Toc17468520)

[APPLICATION DEVELOPMENT AGILE 18](#_Toc17468521)

[CLEARWATER 19](#_Toc17468522)

[NOVA Target State 19](#_Toc17468523)

[IDR Data Governance 20](#_Toc17468524)

[IDR Data Quality Lifecycle 20](#_Toc17468525)

[IDR State Architecture 21](#_Toc17468526)

[NON-CLEARWATER 22](#_Toc17468527)

[CLEARWATER 25](#_Toc17468528)

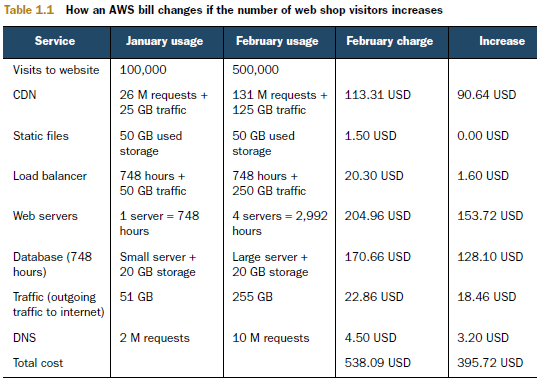
# TECHNOLOGY

## AWS

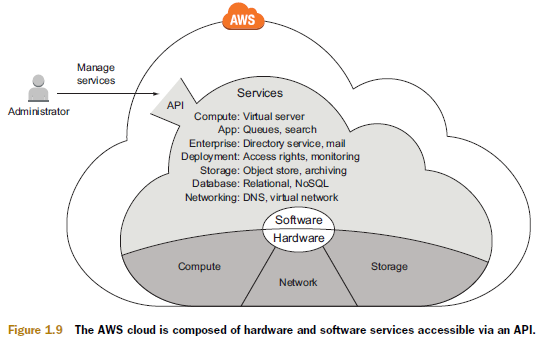
•**Elastic Compute Cloud (EC2),** a service for provisioning computing resources on

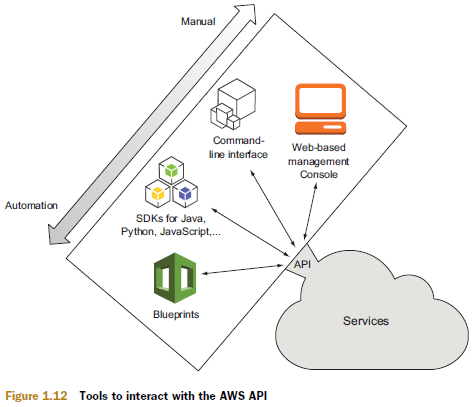
demand •**Elastic Load Balancing (ELB)** distributes traffic to a bunch of servers behind it. Highly available by default.•**Simple Storage Service (S3),** online storage for opaque data •**Elastic Block Store (EBS),** persistent disk-like storage for EC2 instances, in 2008 •**Elastic MapReduce (EMR),** a service providing Hadoop-like clusters for running MapReduce (and later Apache Hive and Apache Pig) jobs, in 2009 •**Relational Database Service (RDS),** a service for managing relational database server instances running in AWS, also in 2009 ⬩**Instance types:** heavy compute capability, vast storage, economy, or simply general-purpose use ⬩**Availability zones** independent within a region, but faster interconnections ⬩**Temporary instance** can disappear after some time ⬩**Images** what instances are running: operating system type and version, the software packages that are available, and applications that are installed. These considerations are all bundled up into images ⬩**Security groups** AWS service to control network traffic like a firewall. Security groups can be attached to services like ELB, EC2, and RDS. With security groups, configure load balancer so that it only accepts requests on port 80 from the internet, web servers only accept connections on port 80 from the load balancer, and MySQL only accepts connections on port 3306 from the web servers. If you want to log in to your web servers via SSH, you must also open port 22. ⬩**CIDR** (Classless Inter-Domain Routing)

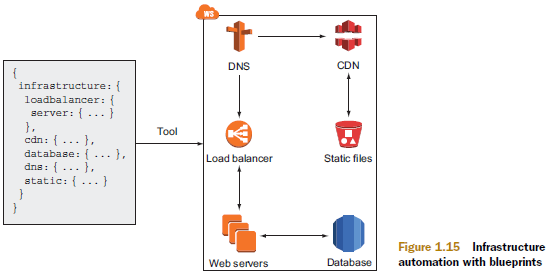
**AWS in Action**

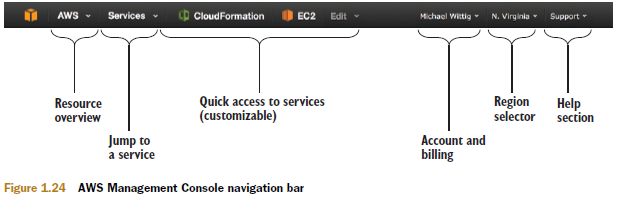
****

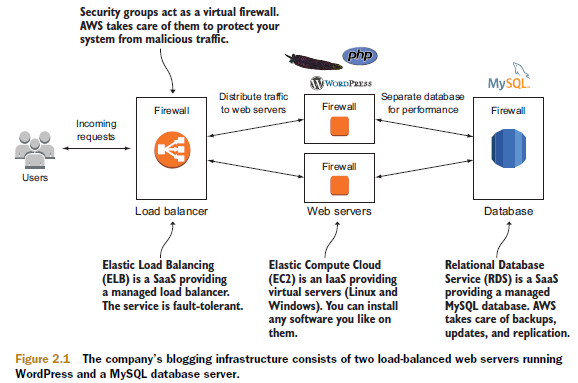
|  |  |
| --- | --- |
|  |  |

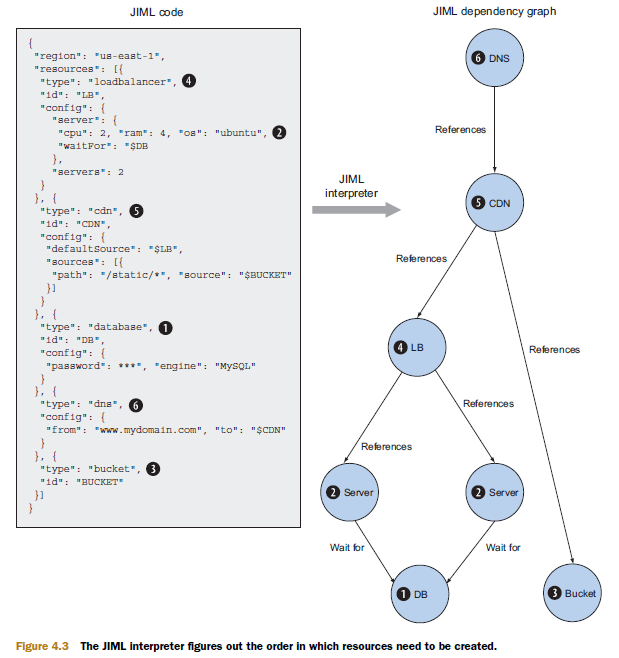


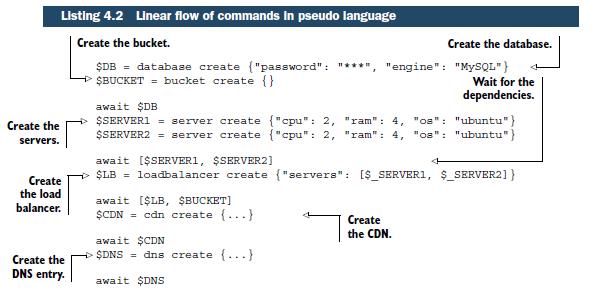


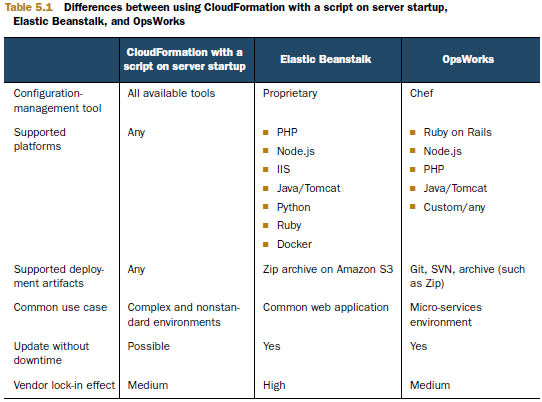


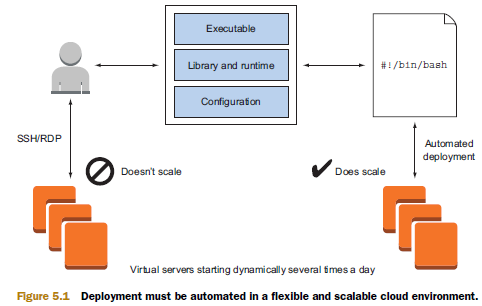


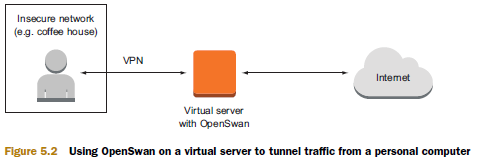


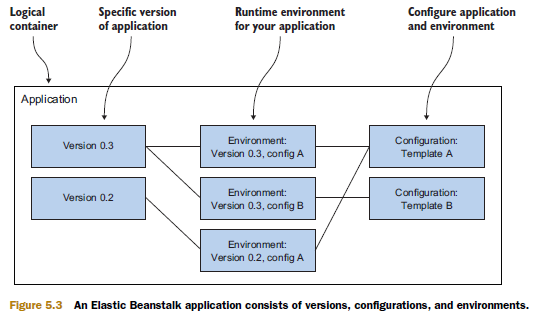


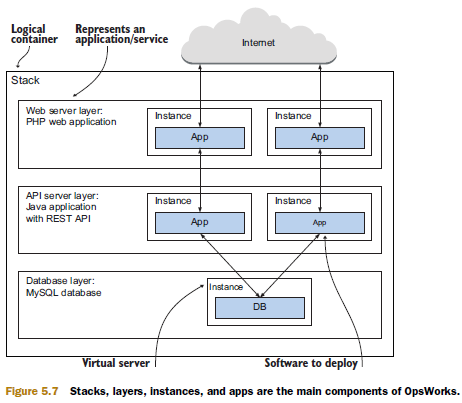












## HADOOPP

Key Features: ➊**HDFS** (Hadoop Distributed File System) adapted to work with huge or large scale bandwidth ➋**MapReduce**: Set up model for the processing of Big Data ➌**YARN**: resource scheduler or assistant for Hadoop resource management ➍**Hadoop Libraries**: enables third party programs to work with Hadoop

## STREAM

Emerging use cases for Spark and Kafka: •**ETL and data engineering**: Data preparation for all analytics •**AI and machine learning**: Massively scalable, parallel processing •**Business Intelligence**: Next-generation business intelligence with big and fast data •**Streaming apps**: Real-time processing of streaming data for the internet of things, artificial intelligence (AI) and natural language processing (NLP).

## SPARK

#### Core

Spark Core is the foundation of the overall project. It provides distributed task dispatching, scheduling, and basic [I/O](https://en.wikipedia.org/wiki/I/O_interface) functionalities, exposed through an application programming interface (for [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)), and [R](https://en.wikipedia.org/wiki/R_(programming_language))) centered on the RDD [abstraction](https://en.wikipedia.org/wiki/Abstraction_(computer_science)) (the Java API is available for other JVM languages, but is also usable for some other non-JVM languages that can connect to the JVM, such as [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language))[[16]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-16)). This interface mirrors a [functional](https://en.wikipedia.org/wiki/Functional_programming)/[higher-order](https://en.wikipedia.org/wiki/Higher-order_programming) model of programming: a "driver" program invokes parallel operations such as map, [filter](https://en.wikipedia.org/wiki/Filter_(computer_science)) or reduce on an RDD by passing a function to Spark, which then schedules the function's execution in parallel on the cluster.[[2]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-hc10-2) These operations, and additional ones such as [joins](https://en.wikipedia.org/wiki/Join_(database)), take RDDs as input and produce new RDDs. RDDs are [immutable](https://en.wikipedia.org/wiki/Immutable_object) and their operations are [lazy](https://en.wikipedia.org/wiki/Lazy_evaluation); fault-tolerance is achieved by keeping track of the "lineage" of each RDD (the sequence of operations that produced it) so that it can be reconstructed in the case of data loss. RDDs can contain any type of Python, Java, or Scala objects.

Besides the RDD-oriented functional style of programming, Spark provides two restricted forms of shared variables: *broadcast variables* reference read-only data that needs to be available on all nodes, while *accumulators* can be used to program reductions in an [imperative](https://en.wikipedia.org/wiki/Imperative_programming) style.[[2]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-hc10-2) A typical example of RDD-centric functional programming is the following Scala program that computes the frequencies of all words occurring in a set of text files and prints the most common ones. Each map, flatMap (a variant of map) and reduceByKey takes an [anonymous function](https://en.wikipedia.org/wiki/Anonymous_function) that performs a simple operation on a single data item (or a pair of items), and applies its argument to transform an RDD into a new RDD.

**val** conf **=** **new** **SparkConf**().setAppName("wiki\_test") // create a spark config object

**val** sc **=** **new** **SparkContext**(conf) // Create a spark context

**val** data **=** sc.textFile("/path/to/somedir") // Read files from "somedir" into an RDD of (filename, content) pairs.

**val** tokens **=** data.flatMap(**\_**.split(" ")) // Split each file into a list of tokens (words).

**val** wordFreq **=** tokens.map((**\_**, **1**)).reduceByKey(**\_** + **\_**) // Add a count of one to each token, then sum the counts per word type.

wordFreq.sortBy(s **=>** -s.\_2).map(x **=>** (x.\_2, x.\_1)).top(**10**) // Get the top 10 words. Swap word and count to sort by count.

#### SQL

Spark [SQL](https://en.wikipedia.org/wiki/SQL) is a component on top of Spark Core that introduced a data abstraction called DataFrames,[[a]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-18) which provides support for structured and [semi-structured data](https://en.wikipedia.org/wiki/Semi-structured_data). Spark SQL provides a [domain-specific language](https://en.wikipedia.org/wiki/Domain-specific_language) (DSL) to manipulate DataFrames in [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), or [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). It also provides SQL language support, with [command-line interfaces](https://en.wikipedia.org/wiki/Command-line_interface) and [ODBC](https://en.wikipedia.org/wiki/Open_Database_Connectivity)/[JDBC](https://en.wikipedia.org/wiki/Java_Database_Connectivity) server. Although DataFrames lack the compile-time type-checking afforded by RDDs, as of Spark 2.0, the strongly typed DataSet is fully supported by Spark SQL as well.

**import** **org.apache.spark.sql.SparkSession**

**val** url **=** "jdbc:mysql://yourIP:yourPort/test?user=yourUsername;password=yourPassword" // URL for your database server.

**val** spark **=** **SparkSession**.builder().getOrCreate() // Create a Spark session object

**val** df **=** spark

.read

.format("jdbc")

.option("url", url)

.option("dbtable", "people")

.load()

df.printSchema() // Looks the schema of this DataFrame.

**val** countsByAge **=** df.groupBy("age").count() // Counts people by age

//or alternatively via SQL:

//df.createOrReplaceTempView("people")

//val countsByAge = spark.sql("SELECT age, count(\*) FROM people GROUP BY age")

#### Streaming

park Streaming uses Spark Core's fast scheduling capability to perform [streaming analytics](https://en.wikipedia.org/wiki/Event_stream_processing). It ingests data in mini-batches and performs RDD transformations on those mini-batches of data. This design enables the same set of application code written for batch analytics to be used in streaming analytics, thus facilitating easy implementation of [lambda architecture](https://en.wikipedia.org/wiki/Lambda_architecture). However, this convenience comes with the penalty of latency equal to the mini-batch duration. Other streaming data engines that process event by event rather than in mini-batches include [Storm](https://en.wikipedia.org/wiki/Storm_(event_processor)) and the streaming component of [Flink](https://en.wikipedia.org/wiki/Apache_Flink).[[20]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-21) Spark Streaming has support built-in to consume from [Kafka](https://en.wikipedia.org/wiki/Apache_Kafka), [Flume](https://en.wikipedia.org/wiki/Apache_Flume), [Twitter](https://en.wikipedia.org/wiki/Twitter#Implementation), [ZeroMQ](https://en.wikipedia.org/wiki/ZeroMQ), [Kinesis](https://en.wikipedia.org/wiki/Amazon_Web_Services#Database), and [TCP/IP sockets](https://en.wikipedia.org/wiki/Network_socket).[[21]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-22) In Spark 2.x, a separate technology based on Datasets, called Structured Streaming, that has a higher-level interface is also provided to support streaming.[[22]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-23)

#### Mlib Machine Learning Library

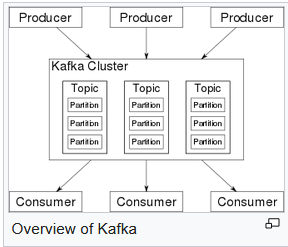
Spark MLlib is a [distributed](https://en.wikipedia.org/wiki/Distributed_computing) machine-learning framework on top of Spark Core that, due in large part to the distributed memory-based Spark architecture, is as much as nine times as fast as the disk-based implementation used by [Apache Mahout](https://en.wikipedia.org/wiki/Apache_Mahout) (according to benchmarks done by the MLlib developers against the [alternating least squares](https://en.wikipedia.org/wiki/Linear_regression) (ALS) implementations, and before Mahout itself gained a Spark interface), and [scales](https://en.wikipedia.org/wiki/Scale_(computing)) better than [Vowpal Wabbit](https://en.wikipedia.org/wiki/Vowpal_Wabbit).[[23]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-24) An overview of Spark MLlib is exist.[[24]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-25) Many common machine learning and statistical algorithms have been implemented and are shipped with MLlib which simplifies large scale machine learning [pipelines](https://en.wikipedia.org/wiki/Pipeline_(software)), including:

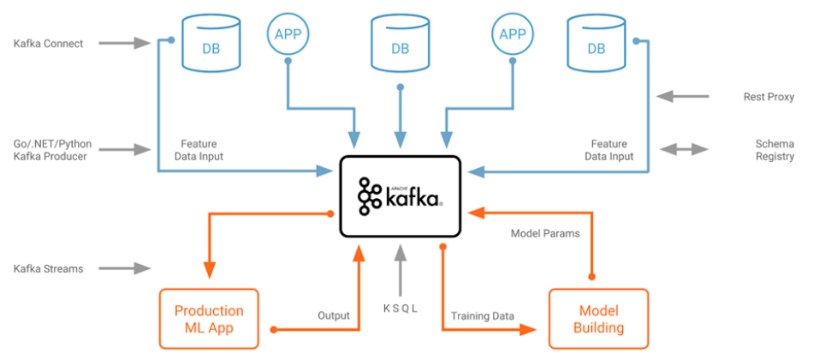
* [summary statistics](https://en.wikipedia.org/wiki/Summary_statistics), [correlations](https://en.wikipedia.org/wiki/Correlation_and_dependence), [stratified sampling](https://en.wikipedia.org/wiki/Stratified_sampling), [hypothesis testing](https://en.wikipedia.org/wiki/Hypothesis_testing), random data generation[[25]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-26)
* [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis): [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machines), [logistic regression](https://en.wikipedia.org/wiki/Logistic_regression), [linear regression](https://en.wikipedia.org/wiki/Linear_regression), decision trees, [naive Bayes classification](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)
* [collaborative filtering](https://en.wikipedia.org/wiki/Collaborative_filtering) techniques including alternating least squares (ALS)
* [cluster analysis methods](https://en.wikipedia.org/wiki/Cluster_analysis) including [k-means](https://en.wikipedia.org/wiki/K-means_clustering), and [latent Dirichlet allocation](https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation) (LDA)
* [dimensionality reduction techniques](https://en.wikipedia.org/wiki/Dimensionality_reduction) such as [singular value decomposition](https://en.wikipedia.org/wiki/Singular_value_decomposition) (SVD), and [principal component analysis](https://en.wikipedia.org/wiki/Principal_component_analysis) (PCA)
* [feature extraction](https://en.wikipedia.org/wiki/Feature_extraction) and [transformation](https://en.wikipedia.org/wiki/Data_transformation_(statistics)) functions
* [optimization](https://en.wikipedia.org/wiki/Optimization_(mathematics)) algorithms such as [stochastic gradient descent](https://en.wikipedia.org/wiki/Stochastic_gradient_descent), [limited-memory BFGS](https://en.wikipedia.org/wiki/Limited-memory_BFGS) (L-BFGS)

#### GraphX

GraphX is a distributed [graph-processing](https://en.wikipedia.org/wiki/Graph_(abstract_data_type)) framework on top of Apache Spark. Because it is based on RDDs, which are immutable, graphs are immutable and thus GraphX is unsuitable for graphs that need to be updated, let alone in a transactional manner like a [graph database](https://en.wikipedia.org/wiki/Graph_database).[[26]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-27) GraphX provides two separate APIs for implementation of massively parallel algorithms (such as [PageRank](https://en.wikipedia.org/wiki/PageRank)): a [Pregel](https://en.wikipedia.org/wiki/Graph_database#Distributed_processing) abstraction, and a more general MapReduce-style API.[[27]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-28) Unlike its predecessor Bagel, which was formally deprecated in Spark 1.6, GraphX has full support for property graphs (graphs where properties can be attached to edges and vertices).[[28]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-29) GraphX can be viewed as being the Spark in-memory version of [Apache Giraph](https://en.wikipedia.org/wiki/Apache_Giraph), which utilized Hadoop disk-based MapReduce.[[29]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-30) Like Apache Spark, GraphX initially started as a research project at UC Berkeley's AMPLab and Databricks, and was later donated to the Apache Software Foundation and the Spark project.[[30]](https://en.wikipedia.org/wiki/Apache_Spark#cite_note-31)

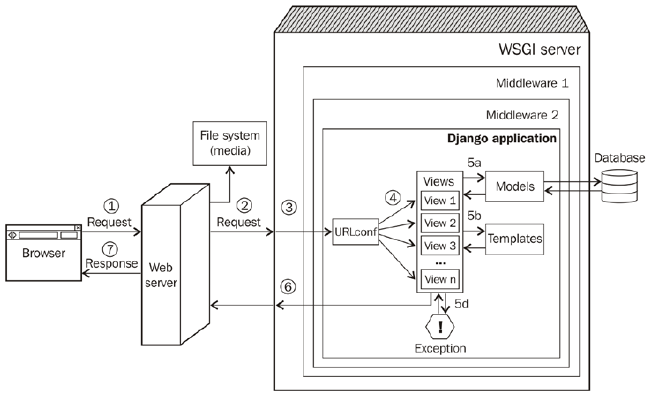
## APACHE KAFKA

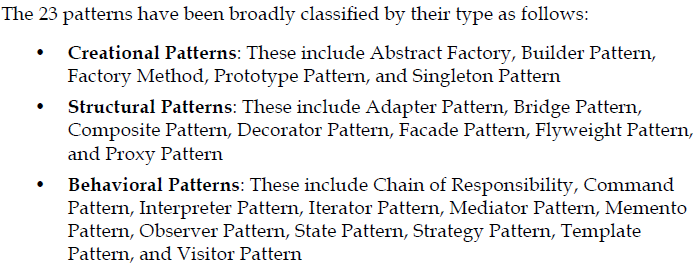


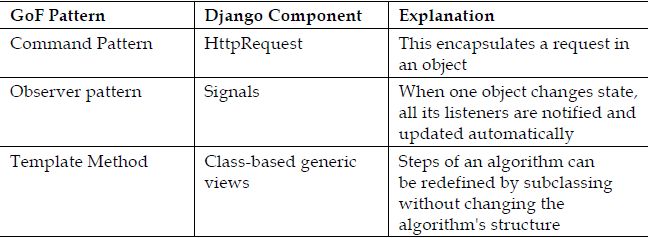


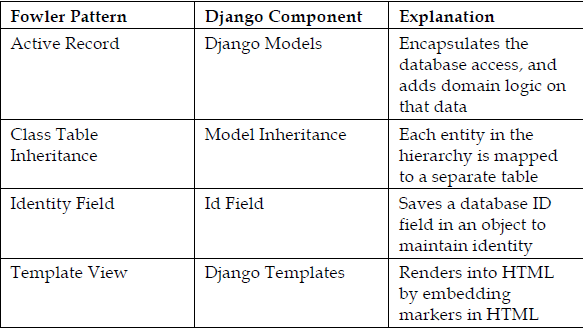
Kafka stores key-value messages that come from arbitrarily many processes called *producers*. The data can be partitioned into different "partitions" within different "topics". Within a partition, messages are strictly ordered by their offsets (the position of a message within a partition), and indexed and stored together with a timestamp. Other processes called "consumers" can read messages from partitions. For stream processing, Kafka offers the Streams API that allows writing Java applications that consume data from Kafka and write results back to Kafka. Apache Kafka also works with external stream processing systems such as [Apache Apex](https://en.wikipedia.org/wiki/Apache_Apex), [Apache Flink](https://en.wikipedia.org/wiki/Apache_Flink), [Apache Spark](https://en.wikipedia.org/wiki/Apache_Spark), and [Apache Storm](https://en.wikipedia.org/wiki/Apache_Storm). Kafka runs on a cluster of one or more servers (called brokers), and the partitions of all topics are distributed across the cluster nodes. Additionally, partitions are replicated to multiple brokers. This architecture allows Kafka to deliver massive streams of messages in a fault-tolerant fashion and has allowed it to replace some of the conventional messaging systems like [Java Message Service](https://en.wikipedia.org/wiki/Java_Message_Service) (JMS), [Advanced Message Queuing Protocol](https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol) (AMQP), etc. Since the 0.11.0.0 release, Kafka offers *transactional writes*, which provide exactly-once stream processing using the Streams API. Kafka supports two types of topics: ➊*Regular topics* can be configured with a retention time or a space bound. If there are records that are older than the specified retention time or if the space bound is exceeded for a partition, Kafka is allowed to delete old data to free storage space. By default, topics are configured with a retention time of 7 days, but it's also possible to store data indefinitely. ➋For *compacted topics*, records don't expire based on time or space bounds. Instead, Kafka treats later messages as updates to older message with the same key and guarantees never to delete the latest message per key. Users can delete messages entirely by writing a so-called tombstone message with null-value for a specific key. Four major APIs: ➊ **Producer API** Permits application to publish streams of records ➋ **Consumer API** Permits application to subscribe to topics and processes streams of records ➌ **Connector API** Executes the reusable producer and consumer APIs that can link the topics to the existing applications ➍ **Streams API** Converts the input streams to output and produces the result. The consumer and producer APIs build on top of the Kafka messaging protocol and offer a reference implementation for Kafka consumer and producer clients in Java. The underlying messaging protocol is a [binary protocol](https://en.wikipedia.org/wiki/Binary_protocol) that developers can use to write their own consumer or producer clients in any programming language. This unlocks Kafka from the [Java Virtual Machine](https://en.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) eco-system. A list of available non-Java clients is maintained in the Apache Kafka wiki.

## DJANGO









**Structural patterns**: ⬩normalized models ⬩model mixins ⬩user profiles ⬩service objects

**Retrieval patterns**: ⬩property field ⬩custom model managers **View patterns**: ⬩acces controlled ⬩context enhancers ⬩services **Template patterns**: ⬩inheritance ⬩active link

## SQL

REPL (Read-Eval-Print-Loop) instructions

## Oracle PL/SQL

##### PL/SQL anonymous block

The basic unit of a PL/SQL (Procedural Language for SQL) source program is the block, which groups together related declarations and statements. A PL/SQL block is defined by the keywords DECLARE, BEGIN, EXCEPTION, and END. These keywords divide the block into a declarative part, an executable part, and an exception-handling part. The declaration section is optional and may be used to define and initialize constants and variables. If a variable is not initialized then it defaults to [NULL](https://en.wikipedia.org/wiki/Null_(SQL)) value. The optional exception-handling part is used to handle run time errors. Only the executable part is required. A block can have a label.

**<<label>>** -- this is optional

**DECLARE**

-- this section is optional

number1 NUMBER(**2**);

number2 number1%TYPE := **17**; -- value default

text1 VARCHAR2(**12**) := 'Hello world';

text2 DATE := SYSDATE; -- current date and time

**BEGIN**

-- this section is mandatory, must contain at least one executable statement

**SELECT** street\_number

**INTO** number1

**FROM** address

**WHERE** **name** = 'INU';

**EXCEPTION**

-- this section is optional

**WHEN** OTHERS **THEN**

DBMS\_OUTPUT**.**PUT\_LINE('Error Code is ' || TO\_CHAR(sqlcode));

DBMS\_OUTPUT**.**PUT\_LINE('Error Message is ' || sqlerrm);

**END**;

The symbol := functions as an [assignment operator](https://en.wikipedia.org/wiki/Assignment_operator) to store a value in a variable.

Blocks can be nested – i.e., because a block is an executable statement, it can appear in another block wherever an executable statement is allowed. A block can be submitted to an interactive tool (such as SQL\*Plus) or embedded within an Oracle Precompiler or [OCI](https://en.wikipedia.org/wiki/Oracle_Call_Interface) program. The interactive tool or program runs the block once. The block is not stored in the database, and for that reason, it is called an anonymous block (even if it has a label).

##### Function

PL/SQL function is generally used to compute and return a single value. This returned value may be a single scalar value (such as a number, date or character string) or a single collection (such as a nested table or array). User-defined functions supplement the built-in functions provided by Oracle Corporation.

**CREATE** **OR** **REPLACE** **FUNCTION** <function\_name> [(**input**/output variable declarations)] **RETURN** return\_type

[AUTHID <**CURRENT\_USER** | **DEFINER**>] <**IS**|**AS**> -- heading part

amount number; -- declaration block

**BEGIN** -- executable part

<PL/SQL block **with** **return** **statement**>

**RETURN** <return\_value>;

[**Exception**

**none**]

**RETURN** <return\_value>;

**END**;

Pipe-lined table functions return collections[[4]](https://en.wikipedia.org/wiki/PL/SQL#cite_note-4) and take the form:

**CREATE** **OR** **REPLACE** **FUNCTION** <function\_name> [(**input**/output variable declarations)] **RETURN** return\_type

[AUTHID <**CURRENT\_USER** | **DEFINER**>] [<**AGGREGATE** | PIPELINED>] <**IS**|**USING**>

[declaration block]

**BEGIN**

<PL/SQL block **with** **return** **statement**>

PIPE **ROW** <**return** **type**>;

**RETURN**;

[**Exception**

**exception** block]

PIPE **ROW** <**return** **type**>;

**RETURN**;

**END**;

A function should only use the default IN type of parameter. The only out value from the function should be the value it returns.

##### Procedure

Like functions, procedures are named program units that can be invoked repeatedly. The primary difference is that **functions can be used in a SQL statement whereas procedures cannot**. Another difference is that the procedure can return multiple values whereas a function should only return a single value.

The procedure begins with a mandatory heading part to hold the procedure name and optionally the procedure parameter list. Next come the declarative, executable and exception-handling parts, as in the PL/SQL Anonymous Block.

**CREATE** **PROCEDURE** create\_email\_address ( -- Procedure heading part begins

name1 VARCHAR2,

name2 VARCHAR2,

company VARCHAR2,

email **OUT** VARCHAR2

) -- Procedure heading part ends

**AS**

-- Declarative part begins (optional)

error\_message VARCHAR2(**30**) := 'Email address is too long.';

**BEGIN** -- Executable part begins (mandatory)

email := name1 || '.' || name2 || '@' || company;

**EXCEPTION** -- Exception-handling part begins (optional)

**WHEN** VALUE\_ERROR **THEN**

DBMS\_OUTPUT**.**PUT\_LINE(error\_message);

**END** create\_email\_address;

The example above shows a standalone procedure - this type of procedure is created and stored in a database schema using the CREATE PROCEDURE statement. A procedure may also be created in a PL/SQL package - this is called a Package Procedure. A procedure created in a PL/SQL anonymous block is called a nested procedure. The standalone or package procedures, stored in the database, are referred to as "[stored procedures](https://en.wikipedia.org/wiki/Stored_procedure)". Procedures can have three types of parameters: IN, OUT and IN OUT.

1. An IN parameter is used as input only. An IN parameter is passed by reference, though it can be changed by the inactive program.
2. An OUT parameter is initially NULL. The program assigns the parameter value and that value is returned to the calling program.
3. An IN OUT parameter may or may not have an initial value. That initial value may or may not be modified by the called program. Any changes made to the parameter are returned to the calling program by default by copying but - with the NO-COPY hint - may be passed [by reference](https://en.wikipedia.org/wiki/Call_by_reference).

PL/SQL also supports external procedures via the Oracle database's standard ext-proc process. [[5]](https://en.wikipedia.org/wiki/PL/SQL#cite_note-5)

##### Package

Packages are groups of conceptually linked functions, procedures, variables, PL/SQL table and record TYPE statements, constants, cursors, etc. The use of packages promotes re-use of code. Packages are composed of the package specification and an optional package body. The specification is the interface to the application; it declares the types, variables, constants, exceptions, cursors, and subprograms available. The body fully defines cursors and subprograms, and so implements the specification. Two advantages of packages are:

1. Modular approach, encapsulation/hiding of business logic, security, performance improvement, re-usability. They support [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) features like function overloading and encapsulation.
2. Using package variables one can declare session level (scoped) variables since variables declared in the package specification have a session scope.

##### Trigger

A [database trigger](https://en.wikipedia.org/wiki/Database_trigger) is like a stored procedure that Oracle Database invokes automatically whenever a specified event occurs. It is a named PL/SQL unit that is stored in the database and can be invoked repeatedly. Unlike a stored procedure, you can enable and disable a trigger, but you cannot explicitly invoke it. While a trigger is enabled, the database automatically invokes it—that is, the trigger fires—whenever its triggering event occurs. While a trigger is disabled, it does not fire. You create a trigger with the CREATE TRIGGER statement. You specify the triggering event in terms of triggering statements, and the item they act on. The trigger is said to be created on or defined on the item—which is either a table, a view, a schema, or the database. You also specify the timing point, which determines whether the trigger fires before or after the triggering statement runs and whether it fires for each row that the triggering statement affects. If the trigger is created on a table or view, then the triggering event is composed of DML statements, and the trigger is called a DML trigger. If the trigger is created on a schema or the database, then the triggering event is composed of either DDL or database operation statements, and the trigger is called a system trigger. An INSTEAD OF trigger is either: A DML trigger created on a view or a system trigger defined on a CREATE statement. The database fires the INSTEAD OF trigger instead of running the triggering statement.

###### Purpose of triggers

⬩Generating derived column values automatically ⬩Enforcing referential integrity ⬩Event logging/ storing information on table access ⬩Auditing ⬩Synchronous replication of tables ⬩Security authorizations ⬩Preventing invalid transactions

##### Array handling

PL/SQL refers to [arrays](https://en.wikipedia.org/wiki/Array_data_type) as "collections". Three types: ➊ [Associative arrays](https://en.wikipedia.org/wiki/Associative_array) (Index-by tables) ➋Nested tables ➌Varrays (variable-size arrays). Programmers must specify an upper limit for varrays, but need not for index-by tables or for nested tables. The language includes several collection [methods](https://en.wikipedia.org/wiki/Method_(computer_science)) used to manipulate collection elements f. ex. FIRST, LAST, NEXT, PRIOR, EXTEND, TRIM, DELETE, etc. Index-by tables can be used to simulate associative arrays, as in this [example of a memo function for Ackermann's function in PL/SQL](https://en.wikipedia.org/wiki/Ackermann_function#cite_note-10).

###### Associative arrays (index-by tables)

Index-by tables can be indexed by numbers or strings. It parallels a [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) *map*, which comprises key-value pairs. There is only one dimension and is unbounded.

###### Nested tables

With nested tables, needs to understand what is nested. Here, a new type is created that may be composed of a number of components. That type can then be used to make a column in a table, and nested within that column are those components.

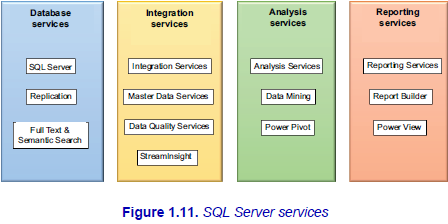
###### Varrays (variable-size arrays)

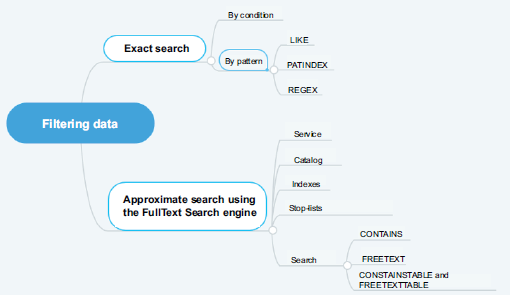
In Varrays, "variable" in "variable-size arrays" doesn't apply to the size of the array in the way you might think that it would. The size the array is declared with is in fact fixed. The number of elements in the array is variable up to the declared size. Arguably then, variable-sized arrays aren't that variable in size.

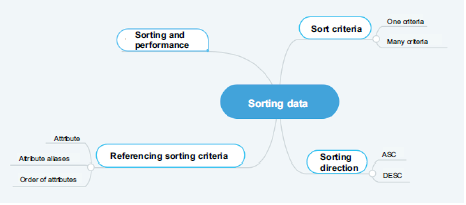
##### Cursors

A [cursor](https://en.wikipedia.org/wiki/Cursor_(databases)) is a mechanism, pointer to a private SQL area that stores information coming from a SELECT or data manipulation language (DML) statement (INSERT, UPDATE, DELETE, or MERGE). A [cursor](https://en.wikipedia.org/wiki/Cursor_(databases)) holds the rows (one or more) returned by a SQL statement. The set of rows the [cursor](https://en.wikipedia.org/wiki/Cursor_(databases)) holds is referred to as the active set.[[7]](https://en.wikipedia.org/wiki/PL/SQL#cite_note-7) A [cursor](https://en.wikipedia.org/wiki/Cursor_(databases)) can be explicit or implicit. In a FOR loop, an explicit cursor shall be used if the query will be reused, otherwise an implicit cursor is preferred. If using a cursor inside a loop, use a FETCH is recommended when needing to bulk collect or when needing dynamic SQL.

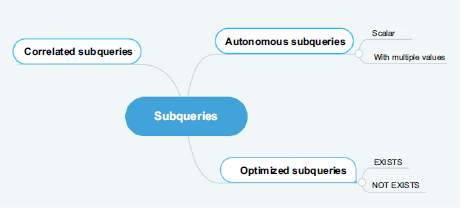
## T-SQL

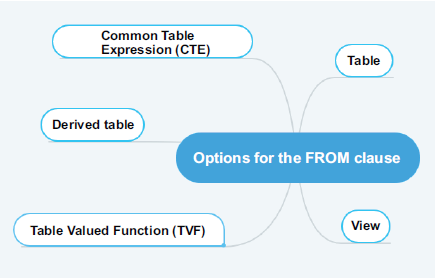


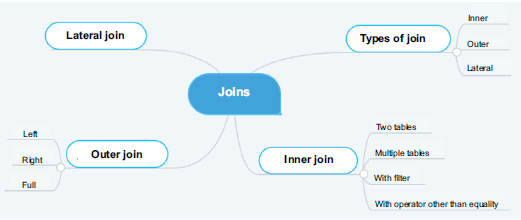












#### INNER JOIN

**USE** AdventureWorks2014

**GO**

**SELECT** P.Name **AS** Product,

SC.Name **AS** Subcategory

**FROM** Production.Product **AS** P

**JOIN** Production**.**ProductSubcategory **AS** SC

**ON** (P**.**ProductSubcategoryID **=** SC**.**ProductSubcategoryID);

**GO**

#### LEFT OUTER JOIN

**USE** AdventureWorks2014;

**GO**

**SELECT** P.Name **AS** Product,

SC.Name **AS** SubCategory

**FROM** Production.Product **AS** P

**LEFT OUTER JOIN** Production**.**ProductSubcategory **AS** SC

**ON (**P**.**ProductSubcategoryID **=** SC**.**ProductSubcategoryID);

**GO**

#### FULL OUTER JOIN

**USE** AdventureWorks2014;

**GO**

**SELECT** P.Name **AS** Product,

SC.Name **AS** SubCategory

**FROM** Production.Product **AS** P

**FULL OUTER JOIN** Production**.**ProductSubcategory **AS** SC

**ON (**P**.**ProductSubcategoryID **=** SC**.**ProductSubcategoryID);

**GO**

#### CARTESIAN PRODUCT

**USE** AdventureWorks2014

**SELECT** P.Name **AS** Product,

SC.Name **AS** SubCategory

**FROM** Production.Product **AS** P

**CROSS JOIN** Production**.**ProductSubcategory **AS** SC

#### LATERAL JOIN (CROSS APPLY)

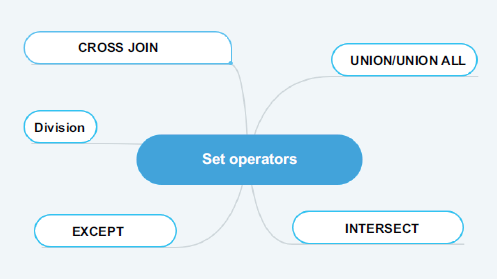
**SELECT** P.Name **AS** Product,

F.Name **AS** SubCategory

**FROM** Production**.**ProductSubcategory **AS** SC

**CROSS APPLY** UF\_PRD **(**sc**.**ProductSubcategoryID**) AS** F

**GO**



#### UNION ALL

**-- Display of all employees and salespeople.**

**SELECT** P.FirstName**,** P.MiddleName**,**P.LastName

**FROM** HumanResources**.**Employee **AS** E

**JOIN** Person**.**Person **AS** P

**ON (**E**.**BusinessEntityID **=** P.BusinessEntityID**)**

**-- The first request returns 290 employees.**

**UNION ALL**

**SELECT** P.FirstName**,** P.MiddleName**,** P.LastName

**FROM** Sales**.**SalesPerson **AS** SP

**JOIN** Person**.**Person **AS** P

**ON (**SP.BusinessEntityID **=** P.BusinessEntityID**)**

**-- The second request returns 17 salespeople.**

**GO**

**-- The union with duplicates returns 307 lines**

**-- (employees and salespeople).**

#### UNION

**-- Display of all employees and salespeople.**

**SELECT** P.FirstName**,** P.MiddleName**,** P.LastName

**FROM** HumanResources**.**Employee **AS** E

**JOIN** Person**.**Person **AS** P

**ON (**E**.**BusinessEntityID **=** P.BusinessEntityID**)**

**-- The first request returns 290 employees.**

**UNION**

**SELECT** P.FirstName**,** P.MiddleName**,** P.LastName

**FROM** Sales**.**SalesPerson **AS** SP

**JOIN** Person**.**Person **AS** P

**ON (**SP.BusinessEntityID **=** P.BusinessEntityID**)**

**-- The second request returns 17 salespeople.**

**GO**

**-- The union without duplicates returns 290 employees**

**-- because the salespeople are already employees.**

#### INTERSECT

**-- Display of all sales employees.**

**SELECT** P.FirstName**,** P.MiddleName**,** P.LastName

**FROM** HumanResource.Employee **AS** E

**JOIN** Person**.**Person **AS** P

**ON (**E**.** BusinessEntityID **=** P.BusinessEntityID**)**

**-- The first query returns 290 employees.**

**INTERSECT**

**SELECT** P.FirstName**,** P.MiddleName**,** P.LastName

**FROM** Sales**.**SalesPerson **AS** SP

**JOIN** Person**.**Person **AS** P

**ON (**SP.BusinessEntityID **=** P.BusinessEntityID**)**

**-- The second query returns 17 sellers.**

**GO**

**-- The intersection returns 17 sales employees**

**-- because all salespeople are already employees.**

#### EXCEPT

**-- Display non-sales employees.**

**SELECT** P.FirstName**,** P.MiddleName**,** P.LastName

**FROM** HumanResources**.**Employee **AS** E

**JOIN** Person.Person **AS** P

**ON (**E**.** BusinessEntityID **=** P.BusinessEntityID**)**

**-- The first query returns 290 employees.**

**EXCEPT**

**SELECT** P.FirstName**,** P.MiddleName**,** P.LastName

**FROM** Sales**.**SalesPerson **AS** SP

**JOIN** Person.Person **AS** P

**ON (**SP.BusinessEntityID **=** P.BusinessEntityID**)**

**-- The second query returns 17 salespeople.**

**GO**

**-- The difference returns 273 employees.**

#### PIVOT

##### Basic

USE AdventureWorks2014 ;

GO

SELECT DaysToManufacture, AVG(StandardCost) AS AverageCost

FROM Production.Product

GROUP BY DaysToManufacture;

DaysToManufacture AverageCost

----------------- -----------

0 5.0885

1 223.88

2 359.1082

4 949.4105

-- Pivot table with one row and five columns

SELECT 'AverageCost' AS Cost\_Sorted\_By\_Production\_Days,

[0], [1], [2], [3], [4]

FROM

(SELECT DaysToManufacture, StandardCost

FROM Production.Product) AS SourceTable

PIVOT

(

AVG(StandardCost)

FOR DaysToManufacture IN ([0], [1], [2], [3], [4])

) AS PivotTable;

Cost\_Sorted\_By\_Production\_Days 0 1 2 3 4

------------------------------ ----------- ----------- ----------- ----------- -----------

AverageCost 5.0885 223.88 359.1082 NULL 949.4105

##### Complex

USE AdventureWorks2014;

GO

SELECT VendorID, [250] AS Emp1, [251] AS Emp2, [256] AS Emp3, [257] AS Emp4, [260] AS Emp5

FROM

(SELECT PurchaseOrderID, EmployeeID, VendorID

FROM Purchasing.PurchaseOrderHeader) p

PIVOT

(

COUNT (PurchaseOrderID)

FOR EmployeeID IN

( [250], [251], [256], [257], [260] )

) AS pvt

ORDER BY pvt.VendorID;

VendorID Emp1 Emp2 Emp3 Emp4 Emp5

----------- ----------- ----------- ----------- ----------- -----------

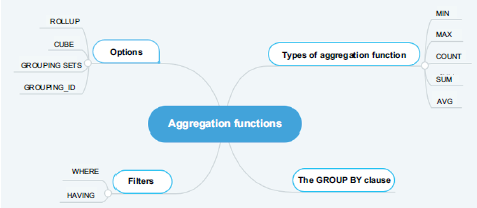
1492 2 5 4 4 4

1494 2 5 4 5 4

1496 2 4 4 5 5

1498 2 5 4 4 4

1500 3 4 4 5 4



##### GROUP BY ROLLUP

SELECT

    warehouse, SUM(quantity)

FROM

    inventory

GROUP BY warehouse;

SQL ROLLUP with one column rollup example

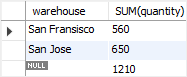
SELECT

    warehouse, SUM(quantity)

FROM

    inventory

GROUP BY ROLLUP (warehouse);



##### CUBE v ROLLUP

The **GROUP BY** clause is used to group the results of aggregate functions according to a specified column. However, the GROUP BY clause doesn’t perform aggregate operations on multiple levels of a hierarchy. F. ex., you can calculate the total of all employee salaries for each department in a company (one level of hierarchy) but you cannot calculate the total salary of all employees regardless of the department they work in (two levels of hierarchy). **ROLLUP** operators let you extend the functionality of GROUP BY clauses by calculating subtotals and grand totals for a set of columns. The **CUBE** operator is similar in functionality to the ROLLUP operator; however, the CUBE operator can calculate subtotals and grand totals for all permutations of the columns specified in it.

**ROLLUP**

1-Department and Gender

2-Department

3-Grand Total

We do not have salary grouped by Gender only. This is because gender is lowest in hierarchy.

**CUBE**

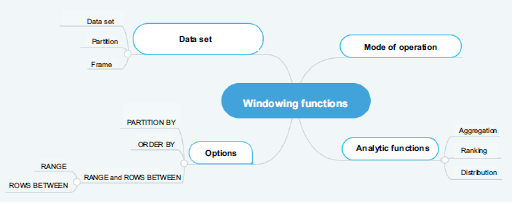
All four possible combinations:

1- Department and Gender

2- Department only

3- Gender Only

4- Grand Total.



#### OVER()

AVG() window function to calculate the average sales for employees in Q1:

select emp\_name, dealer\_id, sales, avg(sales) over() as avgsales from q1\_sales;

+-----------------+------------+--------+-----------+

| emp\_name | dealer\_id | sales | avgsales |

+-----------------+------------+--------+-----------+

| Beverly Lang | 2 | 16233 | 13631 |

| Kameko French | 2 | 16233 | 13631 |

| Ursa George | 3 | 15427 | 13631 |

| Ferris Brown | 1 | 19745 | 13631 |

| Noel Meyer | 1 | 19745 | 13631 |

| Abel Kim | 3 | 12369 | 13631 |

| Raphael Hull | 1 | 8227 | 13631 |

| Jack Salazar | 1 | 9710 | 13631 |

| May Stout | 3 | 9308 | 13631 |

| Haviva Montoya | 2 | 9308 | 13631 |

+-----------------+------------+--------+-----------+

AVG() window function with the PARTITION BY clause to determine the average car sales for each dealer in Q1:

select emp\_name, dealer\_id, sales, avg(sales) over (partition by dealer\_id) as avgsales from q1\_sales;

+-----------------+------------+--------+-----------+

| emp\_name | dealer\_id | sales | avgsales |

+-----------------+------------+--------+-----------+

| Ferris Brown | 1 | 19745 | 14357 |

| Noel Meyer | 1 | 19745 | 14357 |

| Raphael Hull | 1 | 8227 | 14357 |

| Jack Salazar | 1 | 9710 | 14357 |

| Beverly Lang | 2 | 16233 | 13925 |

| Kameko French | 2 | 16233 | 13925 |

| Haviva Montoya | 2 | 9308 | 13925 |

| Ursa George | 3 | 15427 | 12368 |

| Abel Kim | 3 | 12369 | 12368 |

| May Stout | 3 | 9308 | 12368 |

+-----------------+------------+--------+-----------+

AVG() and ROW\_NUM() window functions to determine average car sales for each dealer in Q1 and assign row number to each row in partition:

select dealer\_id, sales, emp\_name,row\_number() over (partition by dealer\_id order by sales) as `row`,avg(sales) over (partition by dealer\_id) as avgsales from q1\_sales;

+------------+--------+-----------------+------+---------------+

| dealer\_id | sales | emp\_name | row | avgsales |

+------------+--------+-----------------+------+---------------+

| 1 | 8227 | Raphael Hull | 1 | 14356 |

| 1 | 9710 | Jack Salazar | 2 | 14356 |

| 1 | 19745 | Ferris Brown | 3 | 14356 |

| 1 | 19745 | Noel Meyer | 4 | 14356 |

| 2 | 9308 | Haviva Montoya | 1 | 13924 |

| 2 | 16233 | Beverly Lang | 2 | 13924 |

| 2 | 16233 | Kameko French | 3 | 13924 |

| 3 | 9308 | May Stout | 1 | 12368 |

| 3 | 12369 | Abel Kim | 2 | 12368 |

| 3 | 15427 | Ursa George | 3 | 12368 |

+------------+--------+-----------------+------+---------------+

## SQL CLR

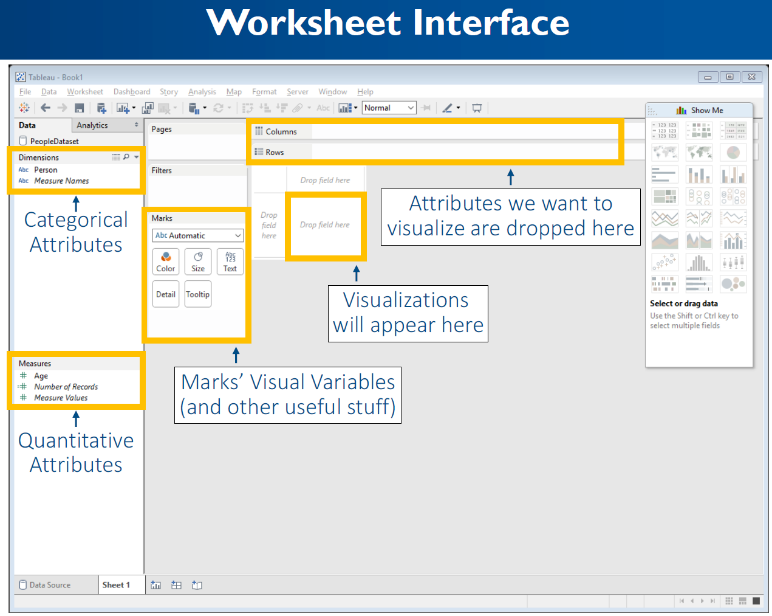
**SQL CLR** or **SQLCLR** ([SQL](https://en.wikipedia.org/wiki/SQL) [Common Language Runtime](https://en.wikipedia.org/wiki/Common_Language_Runtime)) is technology for hosting of the Microsoft .NET common language runtime engine within SQL Server. The SQLCLR allows [managed code](https://en.wikipedia.org/wiki/Managed_code) to be hosted by, and run in, the [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) environment.

This technology, introduced in 2005, allow users for example to create the following types of managed code objects in SQL Server in .NET languages such as [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)) or [VB.NET](https://en.wikipedia.org/wiki/VB.NET).

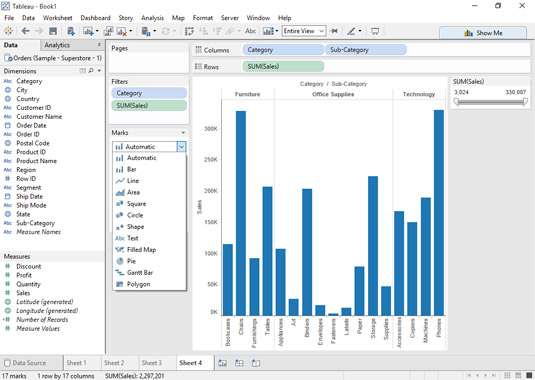
* [Stored procedures](https://en.wikipedia.org/wiki/Stored_procedure) (SPs) which are analogous to *procedures* or *void functions* in procedural languages like VB or C,
* [Triggers](https://en.wikipedia.org/wiki/Database_trigger) which are stored procedures that fire in response to [Data Manipulation Language](https://en.wikipedia.org/wiki/Data_Manipulation_Language) (DML) or [Data Definition Language](https://en.wikipedia.org/wiki/Data_Definition_Language) (DDL) events,
* [User-defined functions](https://en.wikipedia.org/wiki/User-defined_function) (UDFs) which are analogous to functions in procedural languages,
* [User-defined aggregates](https://en.wikipedia.org/w/index.php?title=User-defined_aggregate&action=edit&redlink=1) (UDAs) which allow developers to create custom aggregates that act on sets of data instead of one row at a time,
* [User-defined types](https://en.wikipedia.org/wiki/User-defined_type) (UDTs) that allow users to create simple or complex data types which can be serialized and deserialized within the database.

The SQL CLR relies on the creation, deployment, and registration of [CLI assemblies](https://en.wikipedia.org/wiki/Assembly_(CLI)), which are physically stored in managed code dynamic load libraries (DLLs). These assemblies may contain CLI namespaces, classes, functions and properties.

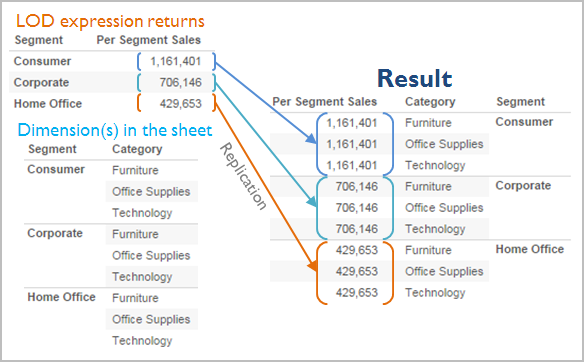
## TABLEAU



🞟**Marks card** gives control over how the data is displayed in the view. The options allow change of detail level, appearances, etc.



🞟**Level of Detail (LOD**) expressions are used to run complex queries involving many dimensions at the data source level instead of bringing all the data to Tableau interface. A simple example is adding dimension to an already calculated aggregate value.



An expression has a finer level of detail than the view when it references a superset of the dimensions in the view. When you use such an expression in the view, Tableau will aggregate results up to the view level. Example: The following level of detail expression references two dimensions:

{FIXED [Segment], [Category] : SUM([Sales])}

When this expression is used in a view that has only [Segment] as its level of detail, the values must be aggregated. Here’s what you would see if you dragged that expression to a shelf:

AVG([{FIXED [Segment]], [Category]] : SUM([Sales]])}])

An aggregation—in this case, average—is automatically assigned by Tableau. You can change the aggregation as needed.

#### Workbook Components

Sheet: A sheet is a singular chart or map in Tableau. Symbol: 

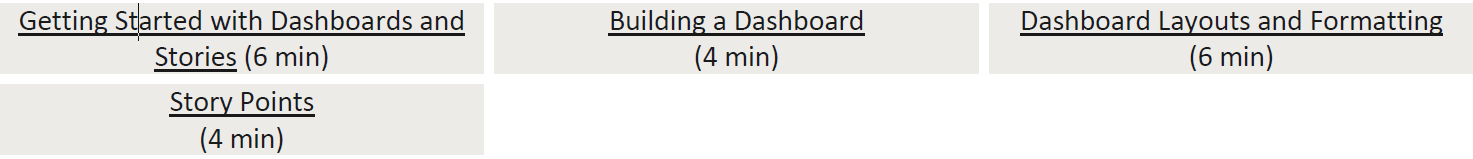
Dashboard: A dashboard is a canvas for displaying multiple sheets at a time and allowing them to interact with each other. Symbol: 

Container: A container is a layout frame on a dashboard that can house sheets, images, filters/parameters, and text boxes. Containers can be horizontal (objects placed go side-by-side) or vertical (objects placed are on top of one another). Double-click any sheet on a dashboard by the center “grip” marks to select the container that the sheet sits in.

Story: A story is a viewing portal that contains a sequence of worksheets or dashboards that work together to convey information. Each individual sheet in a story is called a story point. Symbol: 

Workbook: A workbook is the entire Tableau file containing your sheets and dashboards.

Packaged Workbook: A single zip file with a .twbx extension that contains a workbook along with any supporting local file data sources and background images. Use this format to package your work for sharing with others who don’t have access to the data.



#### Tableau Interface

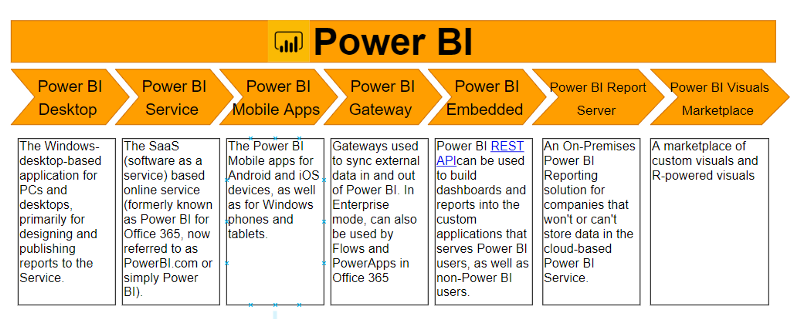
Data Pane: The default left pane that lists your open data sources and the dimensions and measures contained in the selected data sources. Sets and Parameters are also listed here.

Analytics Pane: Clicking the Analytics tab on the left pane will display available analyses for the data displayed on your sheet. Inapplicable analyses will be grayed out. Analyses include adding constant lines, box plots, trend lines, forecasts, and reference bands.

Marks Card: The Marks card is the tool used to create a sheet that controls most of the visual elements in a sheet. Using the Marks card, you can switch between different chart types (bar, line, symbol, filled map, and so on), change colors and sizes, add labels, change the level of detail, and edit the tool tips.

Rows and Columns Shelves: The Rows shelf and the Columns shelf is where you determine which variables will go on what axis. Put data you want displayed along the X-axis on the Columns shelf and data you want displayed on the Y-axis on the Rows shelf.

## POWER BI



## MONGODB

##### Terminology and Concepts

The following table presents the various SQL terminology and concepts and the corresponding MongoDB terminology and concepts.

|  |  |
| --- | --- |
| SQL Terms/Concepts | MongoDB Terms/Concepts |
| database | [database](https://docs.mongodb.com/manual/reference/glossary/#term-database) |
| table | [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) |
| row | [document](https://docs.mongodb.com/manual/reference/glossary/#term-document) or [BSON](https://docs.mongodb.com/manual/reference/glossary/#term-bson) document |
| column | [field](https://docs.mongodb.com/manual/reference/glossary/#term-field) |
| index | [index](https://docs.mongodb.com/manual/reference/glossary/#term-index) |
| table joins | [$lookup](https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/#pipe._S_lookup), embedded documents |
| primary key  Specify any unique column or column combination as primary key. | [primary key](https://docs.mongodb.com/manual/reference/glossary/#term-primary-key)  In MongoDB, the primary key is automatically set to the [\_id](https://docs.mongodb.com/manual/reference/glossary/#term-id) field. |
| aggregation (e.g. group by) | aggregation pipeline  See the [SQL to Aggregation Mapping Chart](https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/). |
| transactions | [transactions](https://docs.mongodb.com/manual/core/transactions/)  For many scenarios, the [denormalized data model (embedded documents and arrays)](https://docs.mongodb.com/manual/core/data-model-design/#data-modeling-embedding) will continue to be optimal for your data and use cases instead of multi-document transactions. That is, for many scenarios, modeling your data appropriately will minimize the need for multi-document transactions. |

##### Executables

The following table presents some database executables and the corresponding MongoDB executables. This table is *not* meant to be exhaustive.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MongoDB | MySQL | Oracle | Informix | DB2 |
| Database Server | [mongod](https://docs.mongodb.com/manual/reference/program/mongod/#bin.mongod) | mysqld | oracle | IDS | DB2 Server |
| Database Client | [mongo](https://docs.mongodb.com/manual/reference/program/mongo/#bin.mongo) | mysql | sqlplus | DB-Access | DB2 Client |

##### Examples

The following table presents the various SQL statements and the corresponding MongoDB statements. The examples in the table assume the following conditions:

* The SQL examples assume a table named people.
* The MongoDB examples assume a collection named people that contain documents of the following prototype:

{

\_id: ObjectId("509a8fb2f3f4948bd2f983a0"),

user\_id: "abc123",

age: 55,

status: 'A'

}

###### Create and Alter

The following table presents the various SQL statements related to table-level actions and the corresponding MongoDB statements.

|  |  |
| --- | --- |
| SQL Schema Statements | MongoDB Schema Statements |
| CREATE TABLE people (  id MEDIUMINT NOT NULL  AUTO\_INCREMENT,  user\_id Varchar(30),  age Number,  status char(1),  PRIMARY KEY (id)  ) | Implicitly created on first [insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne) or [insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany) operation. The primary key \_id is automatically added if \_id field is not specified.  db.people.insertOne( {  user\_id: "abc123",  age: 55,  status: "A"  } )  However, you can also explicitly create a collection:  db.createCollection("people") |
| ALTER TABLE people  ADD join\_date DATETIME | Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.  However, at the document level, [updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany) operations can add fields to existing documents using the [$set](https://docs.mongodb.com/manual/reference/operator/update/set/#up._S_set) operator.  db.people.updateMany(  { },  { $set: { join\_date: new Date() } }  ) |
| ALTER TABLE people  DROP COLUMN join\_date | Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.  However, at the document level, [updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany) operations can remove fields from documents using the [$unset](https://docs.mongodb.com/manual/reference/operator/update/unset/#up._S_unset) operator.  db.people.updateMany(  { },  { $unset: { "join\_date": "" } }  ) |
| CREATE INDEX idx\_user\_id\_asc  ON people(user\_id | db.people.createIndex( { user\_id: 1 } ) |
| CREATE INDEX  idx\_user\_id\_asc\_age\_desc  ON people(user\_id, age DESC) | db.people.createIndex( { user\_id: 1, age: -1 } ) |
| DROP TABLE people | db.people.drop() |

For more information on the methods and operators used, see:

|  |  |  |
| --- | --- | --- |
| [db.collection.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne)  [db.collection.insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany)  [db.createCollection()](https://docs.mongodb.com/manual/reference/method/db.createCollection/#db.createCollection) | [db.collection.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany)  [db.collection.createIndex()](https://docs.mongodb.com/manual/reference/method/db.collection.createIndex/#db.collection.createIndex)  [db.collection.drop()](https://docs.mongodb.com/manual/reference/method/db.collection.drop/#db.collection.drop) | [$set](https://docs.mongodb.com/manual/reference/operator/update/set/#up._S_set)  [$unset](https://docs.mongodb.com/manual/reference/operator/update/unset/#up._S_unset) |

See also: ⬩[Databases and Collections](https://docs.mongodb.com/manual/core/databases-and-collections/) ⬩[Documents](https://docs.mongodb.com/manual/core/document/) ⬩[Indexes](https://docs.mongodb.com/manual/indexes/) ⬩[Data Modeling Concepts](https://docs.mongodb.com/manual/core/data-models/).

###### Insert

The following table presents the various SQL statements related to inserting records into tables and the corresponding MongoDB statements.

|  |  |
| --- | --- |
| SQL INSERT Statements | MongoDB insertOne() Statements |
| INSERT INTO people(user\_id,  age,  status)  VALUES ("bcd001",  45,  "A") | db.people.insertOne(  { user\_id: "bcd001", age: 45, status: "A" }  ) |

For more information, see [db.collection.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne).

See also ⬩[Insert Documents](https://docs.mongodb.com/manual/tutorial/insert-documents/) ⬩[db.collection.insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany) ⬩[Databases and Collections](https://docs.mongodb.com/manual/core/databases-and-collections/) ⬩[Documents](https://docs.mongodb.com/manual/core/document/)

###### Select

The following table presents the various SQL statements related to reading records from tables and the corresponding MongoDB statements. **Note:** The [find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find) method always includes the \_id field in the returned documents unless specifically excluded through [projection](https://docs.mongodb.com/manual/tutorial/project-fields-from-query-results/#projection). Some of the SQL queries below may include an \_id field to reflect this, even if the field is not included in the corresponding [find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find) query.

|  |  |
| --- | --- |
| SQL SELECT Statements | MongoDB find() Statements |
| SELECT \*  FROM people | db.people.find() |
| SELECT id,  user\_id,  status  FROM people | db.people.find(  { },  { user\_id: 1, status: 1 }  ) |
| SELECT user\_id, status  FROM people | db.people.find(  { },  { user\_id: 1, status: 1, \_id: 0 }  ) |
| SELECT \*  FROM people  WHERE status = "A" | db.people.find(  { status: "A" }  ) |
| SELECT user\_id, status  FROM people  WHERE status = "A" | db.people.find(  { status: "A" },  { user\_id: 1, status: 1, \_id: 0 }  ) |
| SELECT \*  FROM people  WHERE status != "A" | db.people.find(  { status: { $ne: "A" } }  ) |
| SELECT \*  FROM people  WHERE status = "A"  AND age = 50 | db.people.find(  { status: "A",  age: 50 }  ) |
| SELECT \*  FROM people  WHERE status = "A"  OR age = 50 | db.people.find(  { $or: [ { status: "A" } , { age: 50 } ] }  ) |
| SELECT \*  FROM people  WHERE age > 25 | db.people.find(  { age: { $gt: 25 } }  ) |
| SELECT \*  FROM people  WHERE age < 25 | db.people.find(  { age: { $lt: 25 } }  ) |
| SELECT \*  FROM people  WHERE age > 25  AND age <= 50 | db.people.find(  { age: { $gt: 25, $lte: 50 } }  ) |
| SELECT \*  FROM people  WHERE user\_id like "%bc%" | db.people.find( { user\_id: /bc/ } )  -or-  db.people.find( { user\_id: { $regex: /bc/ } } ) |
| SELECT \*  FROM people  WHERE user\_id like "bc%" | db.people.find( { user\_id: /^bc/ } )  -or-  db.people.find( { user\_id: { $regex: /^bc/ } } ) |
| SELECT \*  FROM people  WHERE status = "A"  ORDER BY user\_id ASC | db.people.find( { status: "A" } ).sort( { user\_id: 1 } ) |
| SELECT \*  FROM people  WHERE status = "A"  ORDER BY user\_id DESC | db.people.find( { status: "A" } ).sort( { user\_id: -1 } ) |
| SELECT COUNT(\*)  FROM people | db.people.count()  *or*  db.people.find().count() |
| SELECT COUNT(user\_id)  FROM people | db.people.count( { user\_id: { $exists: true } } )  *or*  db.people.find( { user\_id: { $exists: true } } ).count() |
| SELECT COUNT(\*)  FROM people  WHERE age > 30 | db.people.count( { age: { $gt: 30 } } )  *or*  db.people.find( { age: { $gt: 30 } } ).count() |
| SELECT DISTINCT(status)  FROM people | db.people.aggregate( [ { $group : { \_id : "$status" } } ] )  or, for distinct value sets that do not exceed the [BSON size limit](https://docs.mongodb.com/manual/reference/limits/#limit-bson-document-size)  db.people.distinct( "status" ) |
| SELECT \*  FROM people  LIMIT 1 | db.people.findOne()  *or*  db.people.find().limit(1) |
| SELECT \*  FROM people  LIMIT 5  SKIP 10 | db.people.find().limit(5).skip(10) |
| EXPLAIN SELECT \*  FROM people  WHERE status = "A" | db.people.find( { status: "A" } ).explain() |

For more information on the methods and operators used, see

|  |  |
| --- | --- |
| [db.collection.find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)  [db.collection.distinct()](https://docs.mongodb.com/manual/reference/method/db.collection.distinct/#db.collection.distinct)  [db.collection.findOne()](https://docs.mongodb.com/manual/reference/method/db.collection.findOne/#db.collection.findOne)  [limit()](https://docs.mongodb.com/manual/reference/method/cursor.limit/#cursor.limit)  [skip()](https://docs.mongodb.com/manual/reference/method/cursor.skip/#cursor.skip)  [explain()](https://docs.mongodb.com/manual/reference/method/cursor.explain/#cursor.explain)  [sort()](https://docs.mongodb.com/manual/reference/method/cursor.sort/#cursor.sort)  [count()](https://docs.mongodb.com/manual/reference/method/cursor.count/#cursor.count) | [$ne](https://docs.mongodb.com/manual/reference/operator/query/ne/#op._S_ne)  [$and](https://docs.mongodb.com/manual/reference/operator/query/and/#op._S_and)  [$or](https://docs.mongodb.com/manual/reference/operator/query/or/#op._S_or)  [$gt](https://docs.mongodb.com/manual/reference/operator/query/gt/#op._S_gt)  [$lt](https://docs.mongodb.com/manual/reference/operator/query/lt/#op._S_lt)  [$exists](https://docs.mongodb.com/manual/reference/operator/query/exists/#op._S_exists)  [$lte](https://docs.mongodb.com/manual/reference/operator/query/lte/#op._S_lte)  [$regex](https://docs.mongodb.com/manual/reference/operator/query/regex/#op._S_regex) |

See also: ⬩[Query Documents](https://docs.mongodb.com/manual/tutorial/query-documents/) ⬩[Query and Projection Operators](https://docs.mongodb.com/manual/reference/operator/query/) ⬩[mongo Shell Methods](https://docs.mongodb.com/manual/reference/method/)

###### Update Records

The following table presents the various SQL statements related to updating existing records in tables and the corresponding MongoDB statements.

|  |  |
| --- | --- |
| SQL Update Statements | MongoDB updateMany() Statements |
| UPDATE people  SET status = "C"  WHERE age > 25 | db.people.updateMany(  { age: { $gt: 25 } },  { $set: { status: "C" } }  ) |
| UPDATE people  SET age = age + 3  WHERE status = "A" | db.people.updateMany(  { status: "A" } ,  { $inc: { age: 3 } }  ) |

For more information on the method and operators used in the examples:

⬩[db.collection.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany) ⬩[$gt](https://docs.mongodb.com/manual/reference/operator/query/gt/#op._S_gt) ⬩[$set](https://docs.mongodb.com/manual/reference/operator/update/set/#up._S_set) ⬩[$inc](https://docs.mongodb.com/manual/reference/operator/update/inc/#up._S_inc) See also: ⬩[Update Documents](https://docs.mongodb.com/manual/tutorial/update-documents/) ⬩[Update Operators](https://docs.mongodb.com/manual/reference/operator/update/) ⬩[db.collection.updateOne()](https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#db.collection.updateOne) ⬩[db.collection.replaceOne()](https://docs.mongodb.com/manual/reference/method/db.collection.replaceOne/#db.collection.replaceOne)

###### Delete Records

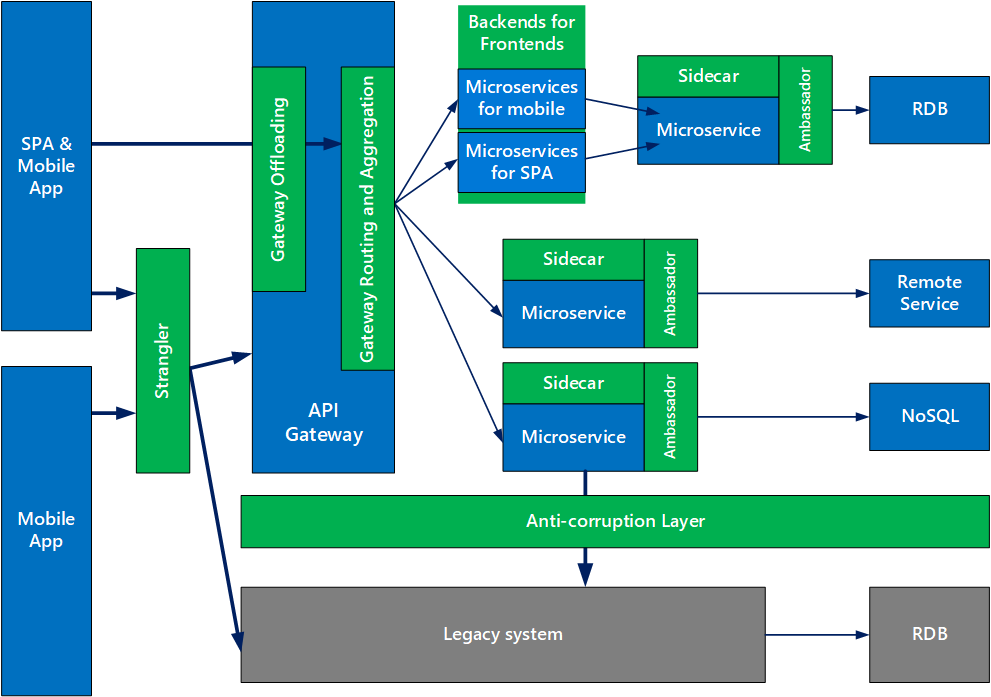
The following table presents the various SQL statements related to deleting records from tables and the corresponding MongoDB statements.

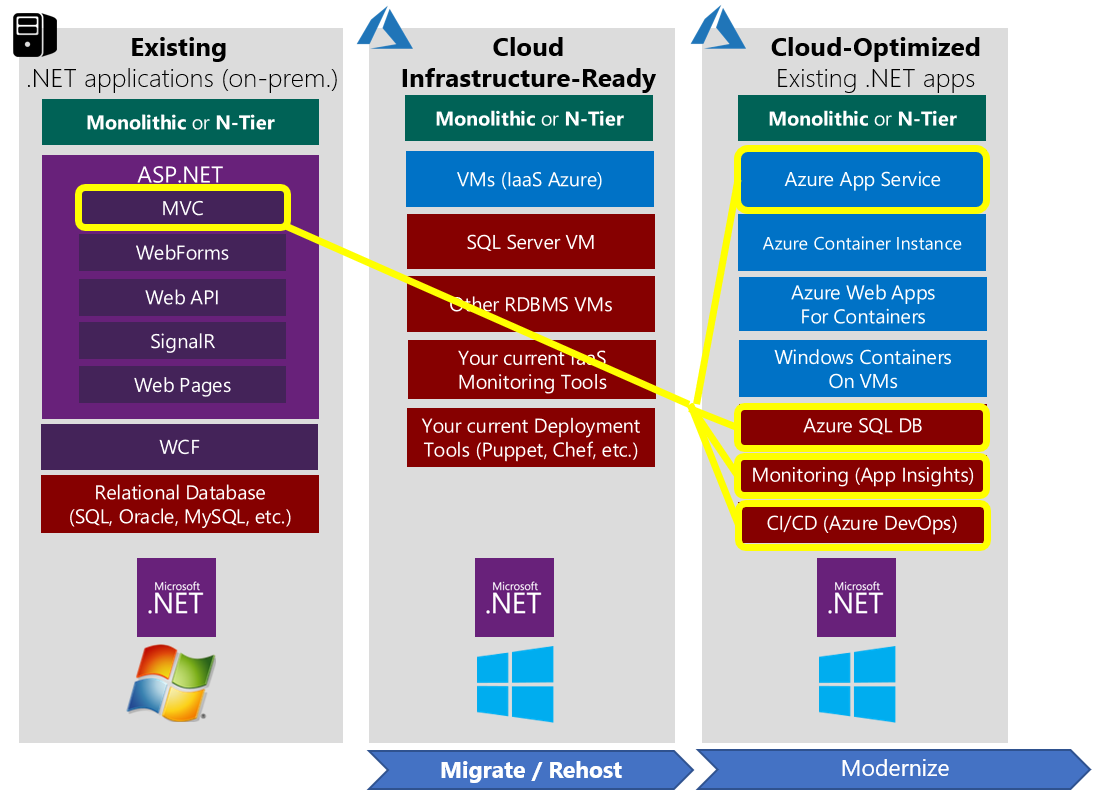
|  |  |
| --- | --- |
| SQL Delete Statements | MongoDB deleteMany() Statements |
| DELETE FROM people  WHERE status = "D" | db.people.deleteMany( { status: "D" } ) |
| DELETE FROM people | db.people.deleteMany({}) |

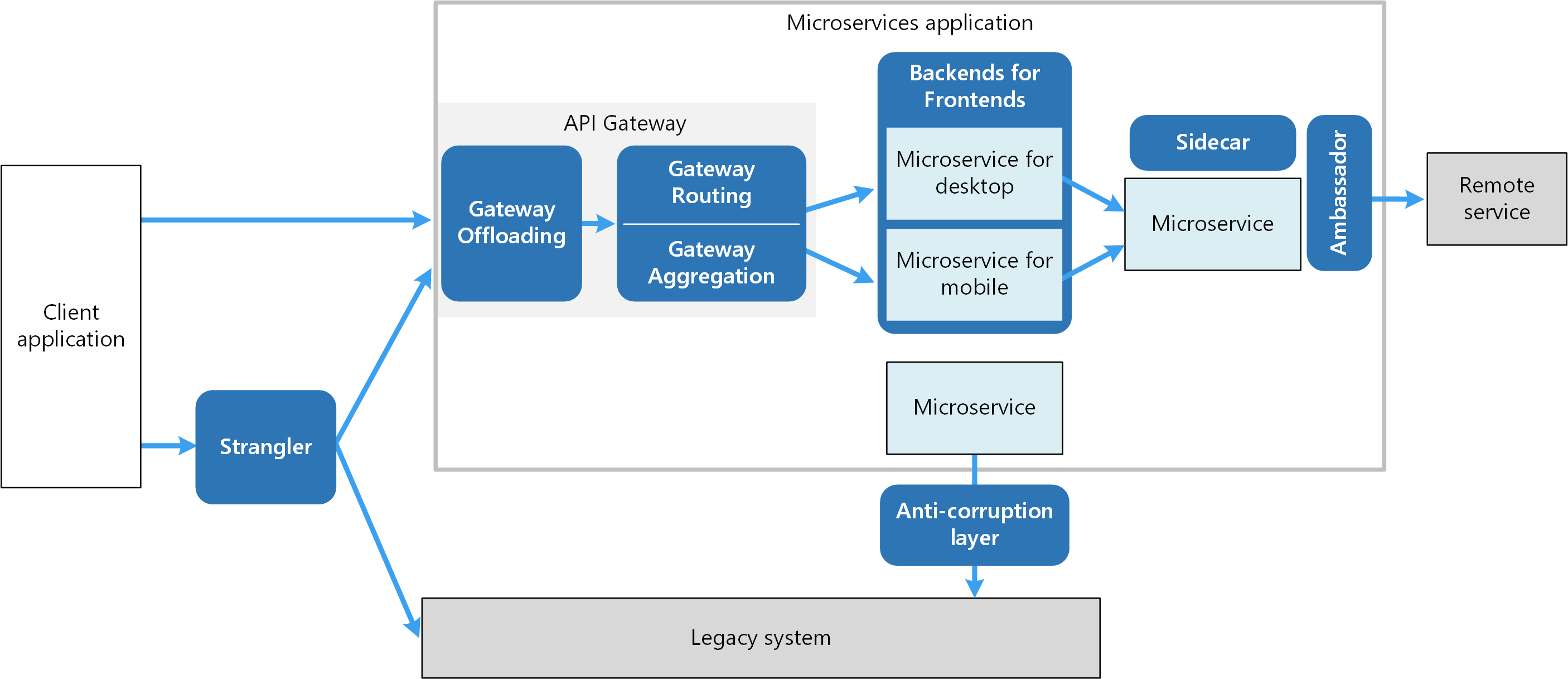
For more information, see [db.collection.deleteMany()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/#db.collection.deleteMany). See also: ⬩[Delete Documents](https://docs.mongodb.com/manual/tutorial/remove-documents/) ⬩

[db.collection.deleteOne()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteOne/#db.collection.deleteOne)

## Microservices







* ⬩[**Ambassador**](https://docs.microsoft.com/azure/architecture/patterns/ambassador) to offload common client connectivity tasks such as monitoring, logging, routing, and security (such as TLS) in a language agnostic way ⬩[**Anti-corruption layer**](https://docs.microsoft.com/azure/architecture/patterns/anti-corruption-layer) implements a façade between new and legacy applications, to ensure that the design of a new application is not limited by dependencies on legacy systems ⬩[**Backends for Front-ends**](https://docs.microsoft.com/azure/architecture/patterns/backends-for-frontends) creates separate backend services for different types of clients (desktop, mobile) ⇨ A single backend service doesn’t need to handle the conflicting requirements of various client types. This pattern can help keep each microservice simple, by separating client-specific concerns ⬩[**Bulkhead**](https://docs.microsoft.com/azure/architecture/patterns/bulkhead) isolates critical resources (connection pool, memory, CPU) for each workload or service. A single workload (or service) can’t consume all of the resources, starving others. This pattern increases the resiliency of the system by preventing cascading failures caused by one service ⬩[**Gateway Aggregation**](https://docs.microsoft.com/azure/architecture/patterns/gateway-aggregation) aggregates requests to multiple individual microservices into a single request, reducing chattiness between consumers and services ⬩[**Gateway Offloading**](https://docs.microsoft.com/azure/architecture/patterns/gateway-offloading) enables each microservice to offload shared service functionality, such as the use of SSL certificates, to an API gateway ⬩[**Gateway Routing**](https://docs.microsoft.com/azure/architecture/patterns/gateway-routing) routes requests to multiple microservices using a single endpoint, so that consumers don't need to manage many separate endpoints ⬩[**Sidecar**](https://docs.microsoft.com/azure/architecture/patterns/sidecar) deploys helper components of an application as a separate container or process to provide isolation and encapsulation ⬩[**Strangler**](https://docs.microsoft.com/azure/architecture/patterns/strangler) supports incremental migration by gradually replacing specific pieces of functionality with new services.

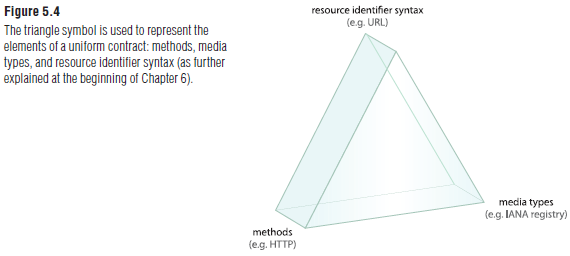
#### API

#### 12 FACTORS

Cloud-friendly applications embrace elastic scalability, ephemeral filesystems, statelessness, and treating everything as a service ⇨ Can scale and deploy rapidly ⬩**Codebase** One codebase tracked in revision control, many deploys ⬩**Dependencies**. Explicitly declare and isolate dependencies ⬩**Configuration**: Store configuration in the environment ⬩**Backing Services**: Treat backing services as attached resources ⬩**Build, release, run:** Separate build and run stages ⬩**Processes**: Execute app as one or more stateless processes ⬩**Port binding:** Export services via port binding ⬩**Concurrency**: Scale out via the process model

⬩**Disposability**: Maximize robustness with fast startup and graceful shutdown ⬩**Dev/prod parity:** Keep development, staging, and production as similar as possible ⬩**Logs**: Treat logs as event streams ⬩**Admin processes**: Run admin/management tasks as one-off processes

#### REST

REST constraints are design rules that are applied to establish the distinct characteristics

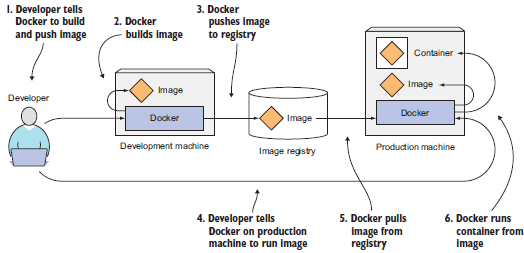
of the REST architectural style. Each constraint is a pre-determined design decision that can have both positive and negative impacts. The intent is for the positives of each constraint to balance out the negatives to produce an overall architecture that resembles the best features of the Web. Formal REST constraints: ⬩**Client-Server** {393} requires that a service offer one or more capabilities and listen for requests on these capabilities. A consumer invokes a capability by sending the corresponding request message, and the service either rejects the request or performs the requested task before sending a response message back to the consumer. Exceptions that prevent the task from proceeding are raised back to the consumer, and the consumer is responsible for taking corrective action ⬩**Stateless** {395} The communication between service consumer (client) and service (server) must be stateless between requests. This means that each request from a service consumer should contain all the necessary information for the service to understand the meaning of the request, and all session state data should then be returned to the service consumer at the end of each request. Statelessness is one of the primary influences over service contract design in REST-style architecture. It imposes significant restrictions on the kinds of communication allowed between services and their consumers in order to achieve its design goals. The application of the Cache {398} and Layered System {404} constraints helps to compensate for limitations resulting from Stateless {395} ⬩**Cache** {398} Response messages from the service to its consumers are explicitly labeled as cacheable or non-cacheable. This way, the service, the consumer, or one of the intermediary middleware components can cache the response for reuse in later requests. The Cache {398} constraint builds upon Client-Server {393} and Stateless {395} with a requirement that responses are implicitly or explicitly labeled as cacheable or noncacheable. Requests are passed through a cache component, which may reuse previous responses to partially or completely eliminate some interactions over the network. This form of elimination can improve efficiency and scalability, and can further improve user-perceived performance by reducing the average latency during a series of interactions. However, a common reason for incorporating caching as a native part of a REST architecture is as a counterbalance to some of the negative impacts of applying the Stateless {395} constraint ⬩**Interface/Uniform Contract** {400} The Interface {400} constraint (also known as “Uniform Interface”) states that all services and service consumers within a REST-compliant architecture must share a single, overarching technical interface. As the primary constraint that distinguishes REST from other architecture types, Interface {400} is generally applied using the methods and media types provided by HTTP and other Internet standards ⬩**Code-On-Demand** {407} This optional constraint is primarily intended to allow logic within clients (such as Web browsers) to be updated independently from server-side logic. Code-On-Demand {407} typically relies on the use of Web-based technologies, such as Web browser plug-ins, applets, or client-side scripting languages (i.e. JavaScript). Code-On-Demand {407} can further be applied to services and service consumers. For example, a service can be designed to dynamically defer portions of logic to service consumer programs. For example, this type of functionality can be used in support of Stateless {395}, which dictates when session state should be deferred back to the service consumer. Code-On-Demand {407} can also build upon this by further deferring the processing effort. This approach may be justifiable when service logic can be executed by the consumer more efficiently or effectively

#### DOCKER

Docker = platform for packaging, distributing, and running applications. Allows you to package your application together with its whole environment (libraries that the app requires or files usually available on the filesystem of an installed operating system). Docker makes it possible to transfer this package to a central repository from which it can then be transferred to any computer running Docker and executed there. 3 concepts: ➊**Images:** Docker-based container image is something you package your application and its environment into. It contains the filesystem that will be available to the application and other metadata, such as the path to the executable that should be executed when the image is run. ➋**Registries**: A Docker Registry is a repository that stores your Docker images and facilitates easy sharing of those images between different people and computers. When you build your image, you can either run it on the computer you’ve built it on, or you can push (upload) the image to a registry and then pull (download) it on another computer and run it there. Certain registries are public, allowing anyone to pull images from it, while others are private, only accessible

to certain people or machines. ➌**Containers**: Docker-based container is a regular Linux container created from a Docker-based container image. A running container is a process running on the host running Docker, but it’s completely isolated from both the host and all

other processes running on it. The process is also resource-constrained, meaning it can only access and use the amount of resources (CPU, RAM, and so on) that are allocated to it.

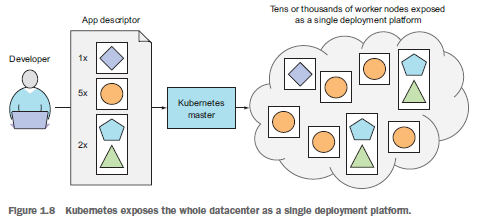


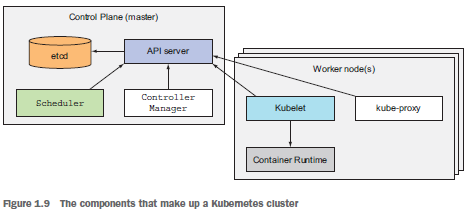
#### KUBERNETES

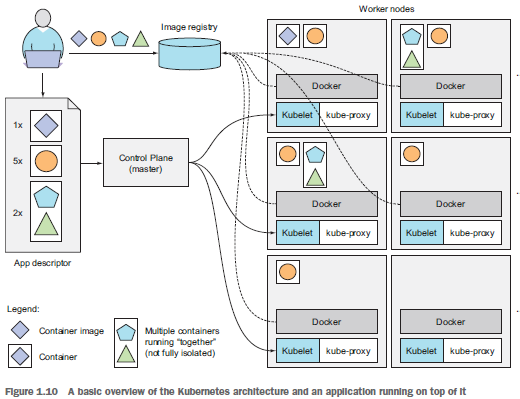
Kubernetes is a software system that allows you to easily deploy and manage containerized

applications on top of it. It relies on the features of Linux containers to run heterogeneous

applications without having to know any internal details of these applications and without having to manually deploy these applications on each host. Because these apps run in containers, they don’t affect other apps running on the same server, which is critical when you run applications for completely different organizations on the same hardware. Kubernetes enables you to run your software applications on thousands of computer nodes as if all those nodes were a single, enormous computer. It abstracts away the underlying infrastructure and, by doing so, simplifies development, deployment, and management for both development and the operations teams. Deploying applications through Kubernetes is always the same, whether your cluster contains only a couple of nodes or thousands of them. The size of the cluster makes no difference at all. Additional cluster nodes simply represent an additional amount of resources available to deployed apps.







## PYTHON

#### LUIGI

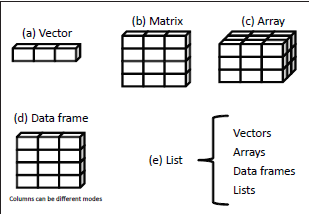
#### DASK

#### PANDAS

#### SQL ALCHEMY

## R

#### Data Structure



## JAVASCRIPT

## SCIKIT-LEARN

## TENSORFLOW

## PYTORCH

## MACHINE LEARNING

**⬩Regression questions**: ‘How much’ and ‘how many’. For example, how much will my car be worth in two years? ⬩**Classification questions**: such as ‘Type of object’. For example, what to class does this object belong? ⬩**Clustering or grouping questions**. For example, what are the different clusters for this particular set of objects? ⬩**Abnormality detection questions**. For example, is this object abnormal based on what is defined as normal?

#### Regression

#### Boost

#### Deep Learning

## Feature Engineering

➊**Resampling Imbalanced Data**: Balanced Accuracy, Precision-Recall Curves, F1-score, SMOTE (Synthetic Minority Oversampling Technique) ➋**Creating New Features**: DFS (Deep Feature Synthesis) ➌**Handling Missing Values**: Iterative Imputer (R imputation packages missForest, mi, mice, etc.) ➍**Outlier Detection**: Isolation Forest

# CIBC Control

## CIBC 20 Services (Financial)

|  |  |
| --- | --- |
| AUDIT | Admin of Non-Core loans. |
| FINANCE | Advertisement Costs |
| GLOBAP OPS | AR, AP |
| HR | Business Analysis |
| LEGAL | Call Centre Supports products for Commercial Banking |
| MARKETING | Compliance |
| RETAIL | Fees (Directors, OSFI) |
| RISK | Financial Analysis |
| TECH SERV | Financial Ombudsman |
| WORLD MARKETS | Financial Risk Support |
| WEALTH | HR - Compensation |
|  | HR - Compliance |
|  | Management Costs |
|  | Project Management |
|  | Resource Centre - reports (M&A, Green sheets, Prospectus') and internet searches |
|  | Stock services |
|  | TI/TS Application Support Cost |
|  | TI/TS Technology Services Cost |

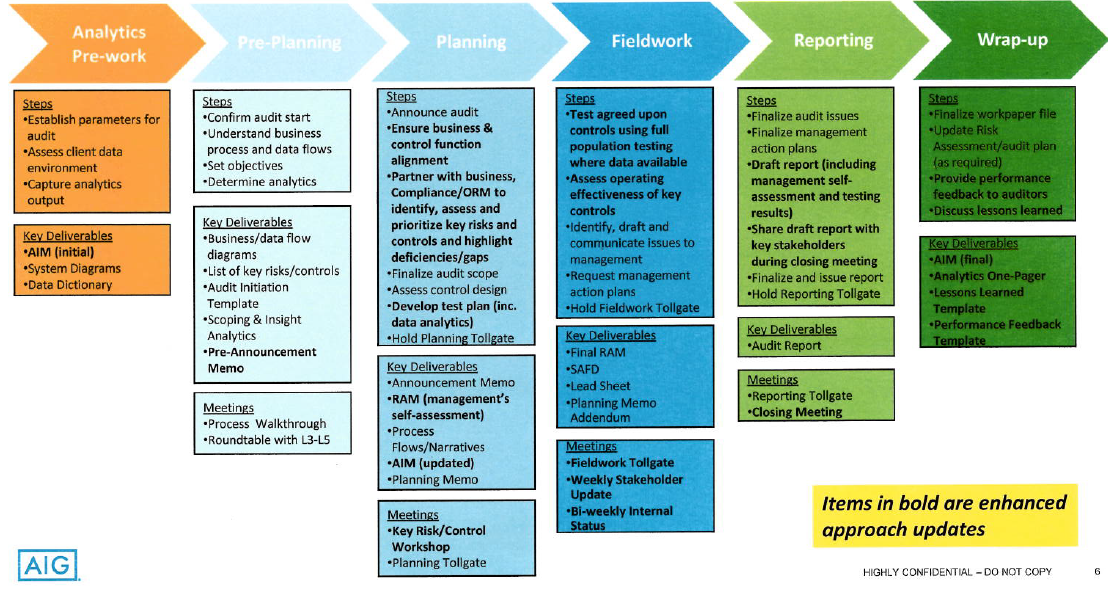
## CIBC Processes (FCU)

|  |  |
| --- | --- |
| BUSINESS\_PROCESS | SUB\_PROCESS |
| A/P | Accrual |
| Interco loan | Account for loan payable to treasury |
| Outstanding Cheques Clearing | Accounting Outstanding Cheques |
| Accrue Liabilities | ID significant individual liabilities |
| Accrue Obligations related to Securities | Record Repos Position |
| Record Securities Sold Short Position |
| Calculate/ Collect Mortgage Income | Originate a mortgage - recording of acquisition costs on mortgage origination |
| Income Taxes Note Disclosure | Compilation of Note Disclosure |
| Note Disclosure Aging of Deposits | Demand, Notice & Term Deposits |
| Note Disclosure IR Sensitivity | Loans & Deposits Aging & yields |
| Note Disclosure Mortgage and customer Loans | Mortgages & Consumer Loans |
| Note Disclosure Segment info | Establish customer CIF (name, address, & permanent information) |
| Defer Acquisition Cost on Mortgages | Calculate/invoice acquisition cost |
| Prepare amortization schedule |
| Defer Payments to Loblaws | Defer Acquisition Cost of acquiring credit products and points |
| EUC Applications | General Controls |
| Financial Statements Preparation | Compilation of Notes to the Financial Statements |
| Get a mortgage loan on the books | Funding Mortgages |
| Get the Loans on the books | Attach credit - PLC |
|  | Disburse Funds for Personal Loans |
| GL/source system balancing | Automatically compare ICBS and GL:M balances |
| HR | Bi-weekly review of payroll register (Including New Hire, Transfers, and Terminations) |
| ICBS Application Controls | AS400 Recovery |
|  | Change Management |
|  | ICBS Incident & Problem Management Process |
| ICBS Information Security | Security Administration |
| Maintain customer demand (chequing) deposit | Calculate and accrue daily interest |
| Maintain interest rates |
| Transaction Cheque Clearing |
| Transaction processing - EFT |
| Transaction processing - POS, ABM, Internet, TB - on Tandem |
| Maintain customer loan | Maintain interest rates in ICBS |
|  | Recognize interest calc & accrual |
|  | Transaction Processing - Payments or PLC cheques |
| Maintain customer notice (RSP) deposits | RSP Renewal |
| Maintain customer notice(savings) deposit | Transaction processing - Internet, ABM, TB (transfers only) - on Tandem |
| Maintain residential mortgages | All sub-process |
|  | Apply payments to Int. income and principal / Accrue Int. at month-end |
|  | Determine mortgage interest rates |
| Manage Bank Accounts | Balance & Settle A/P Bank Account |
|  | Balance & Settle ABM Unpostable, All EFT Return Bank Accounts |
|  | Balance & Settle ABM, POS, RB, SCD, Plus, Outbound EFT Bank Accounts |
|  | Balance & Settle Cheque Clearing |
|  | Balance & Settle EFT Bank Account |
|  | Balance & Settle General Operating, Treasury, Mortgage, EFT, USD Bank |
|  | Balance & Settle Guarantee Payments Bank A/C Drafts & MO |
|  | Balance & Settle Guarantee Payments, Cheq Clr Bank A/C Loans |
|  | Balance & Settle Payroll Bank |
|  | Balance & Settle Treasury Bank A/C |
| Manage Suspense Accounts | Manage Operating Suspense A/C |
| Other Misc Suspense Accts | Accounting Items in Suspense A/C |
| Purchase & pay for non interest expenses | Pay Outside Services (Amortized Trailer fees / Commissions) |
| Pay Other Misc Expenses |
| Purchase & pay other expenses | Pay Other Expenses |
| Recognize deferred taxes | Book Monthly Tax Recovery |
|  | Determine monthly tax rate - Acct |
| Recognize fee income | Recognize Amicus ABM Surcharge |
|  | Recognize Interac Charges (convenience fee auto charged for each customer txn) |
|  | Recognize Returned Cheque Fees |
| Recognize FOREX non-trading income | Recognize other income |

## CIBC 26 Processes (OPC) – 113 Sub-processes

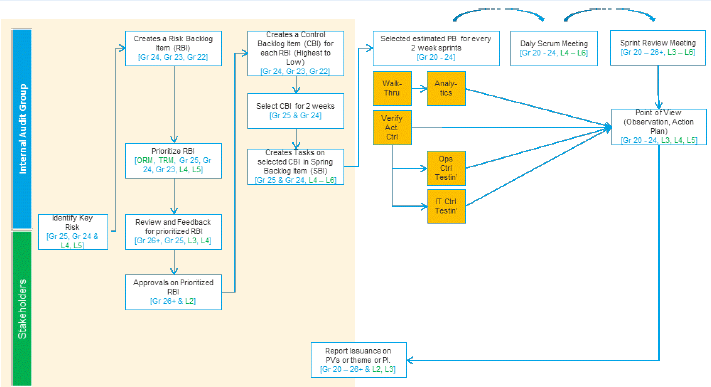
|  |  |
| --- | --- |
| Process | Sub Process |
| Brokerage & Trade | Broker Services - Cash Processing |
| Broker Services- Collateral Management |
| Cash Management |
| Cash Management - Collection of Foreign Cheques |
| Cash Management-Cheque Issuance |
| Cash Management-Incoming wire payments/Cheque deposits |
| Cash Management-ISI Liasion Desk/Bank Reconciliation-Break Resolution |
| CP Issuance - Billing |
| CP Issuance - Book Based Maturity |
| CP Issuance - Physical Maturity |
| CP Issuance-DCS Settlement |
| CP Issuance-Physical Settlement |
| Domestic Equity /Bonds Settlements |
| Equity Arbitrage |
| Futures & Options Settlements |
| GIC Settlements |
| Institutional Equity Settlements - Equity Arbitrage |
| International Settlements |
| Money Market |
| Money Market DTC/FED Settlements |
| Money Market US Settlements- Physical Trades |
| Over The Counter Receipt of Securities |
| Over The Counter/Branch Receipt of Securities |
| Safekeeping |
| Security Lending and Borrowing |
| Segregation Management |
| Stock Transfers |
| UK Securities Lending |
| Compliance | COB Disclosure |
| Credit Mgt | Monitor Credit |
| Customer Satisfaction | Customer Complaints Management |
| Customer Restitution |
| Derivatives Settlement Operations | Confirmations |
| Post-Settlement Investigations |
| Pre-Settlement Investigations |
| Settlements |
| Foreign Exchange | Booking |
| Maintenance | Account Information Maintenance |
|  | Customer Information Maintenance |
|  | Operator Profile Maintenance |
|  | Suspense Account Maintenance |
| Manage and Monitor the Imperial vehicles | Execute Transactions |
| Identify Substitute and Replacement Assets |
| Reporting |
| Management Processes | Investments |
| Lending |
| Procedures Information |
| Regulatory Compliance |
| Sales Management |
| Origination | Adjudication |
|  | Application Processing |
|  | Funding & Disbursement |
| Origination (Commercial) | Adjudication (Commercial) |
| Funding & Disbursement (Commercial) |
| Outsourcing | Outsourcing - ADP |
| Payments Processing | Cash Settlements |
| Credit Administration |
| Investigations |
| Reports Balancing |
| Sanction Filtering |
| Validation & Message Repair |
| Portfolio Management | Credit Derivative Hedging |
| Credit Derivative Trading |
| Establish Portfolio Strategy |
| Hedging |
| Portfolio Management |
| Proprietary Products | Account Maintenance (CM, Talvest and SI only) |
| Account Opening (PPS) |
| Account Opening/Closing/Transfers (CM/Talvest only) |
| Client Tax Reporting / Tax filing |
| Financial Transactions/ Adjustments |
| Trust Accounting |
| Registered Products | 3rd Party Settlements - Brokerage |
| 3rd Party Settlements-Fixed Term |
| Account Transfers (Internal)-Fixed Term |
| Adjustments - Brokerage |
| Adjustments-Fixed Term |
| Client Support-Fixed Term |
| Deposits-Fixed Term |
| GL Reporting-Fixed Term |
| Monitoring & Compliance - Brokerage |
| Tax Reporting - Brokerage |
| Tax Reporting (GIC Withdrawals)-Fixed Term |
| Transfers - Brokerage |
| Withdrawals- Brokerage |
| Withdrawals-Fixed Term |
| Sales Fulfillment | Lending - Personal Loan Products |
|  | Lending - Small Business Loan Products |
|  | Small Business Account |
| Sales Origination | Account Open - Personal Deposits |
|  | Account Open - Small Business Deposits |
| Security &Control | CSP Application Processing |
|  | Database Control |
| Service - Inventory Control | Ordering |
| Servicing | Annual Statement |
| Call Center |
| Discharge |
| Early Renewals |
| Product Changes |
| Renewals |
| Taxes |
| Transaction Processing |
| Servicing (Commercial) | Annual Portfolio Review |
| Renewals (Commercial) |
| Transaction Processing (Commercial) |
| Technology Mgt | IT Access Control |
| 3rd Party Mutual Funds | Processing |
| Trade Finance | Documentation Verification |
| Transaction Processing | Centralized Instruments Processing |
| Cheque Processing |
| Deposit Processing |
| Inter Branch Payments (IBP) Processing |
| Withdrawal Processing |
|  |  |

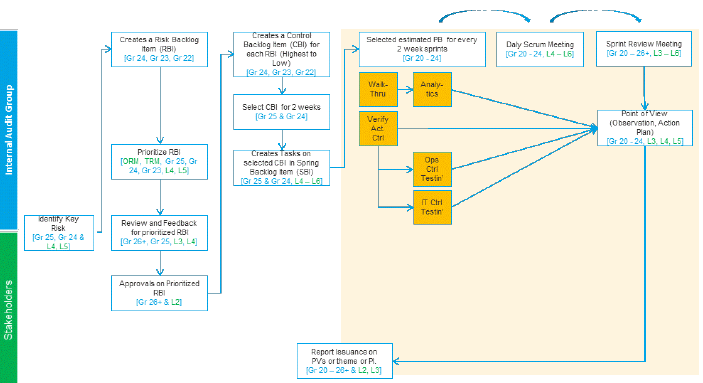
# AGILE Audit Lifecycle



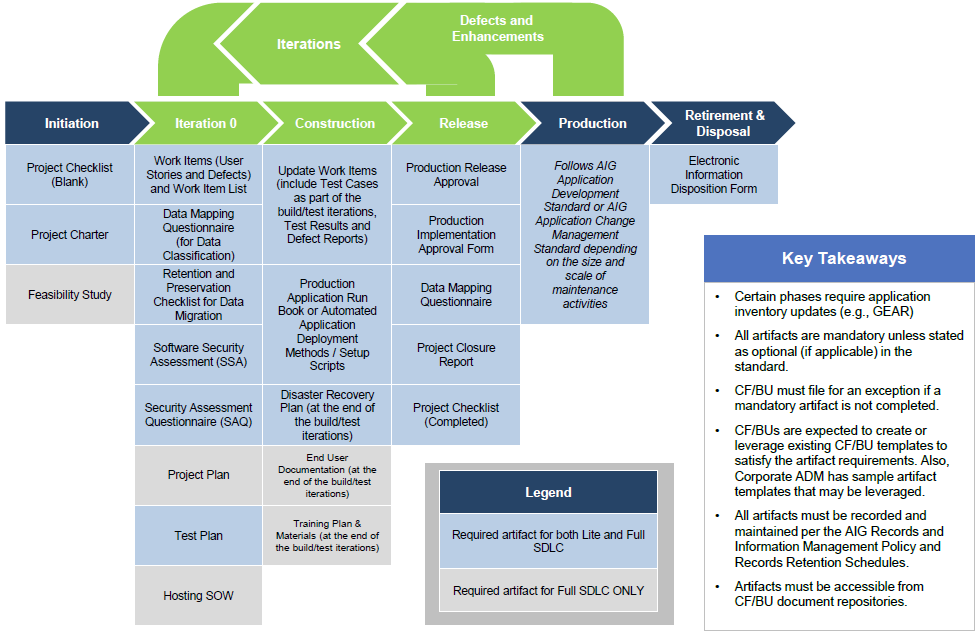
## AGILE AUDIT

Agile concepts: •Audit Increment planning” build a backlog of key risks and controls •Execute each sprint (2 week intervals) •After each sprint have a sprint review meeting with L4 to discuss results and initiate. After each sprint have tollgate to discuss stopping or continuing with audit •After each sprint and before next Sprint have Lessons learned session to discuss went well in sprint and what needs enhancements from next sprint •Holding daily scrum meetings (10 minutes) to discuss progress from yesterday, plan for current day and if any escalation is required



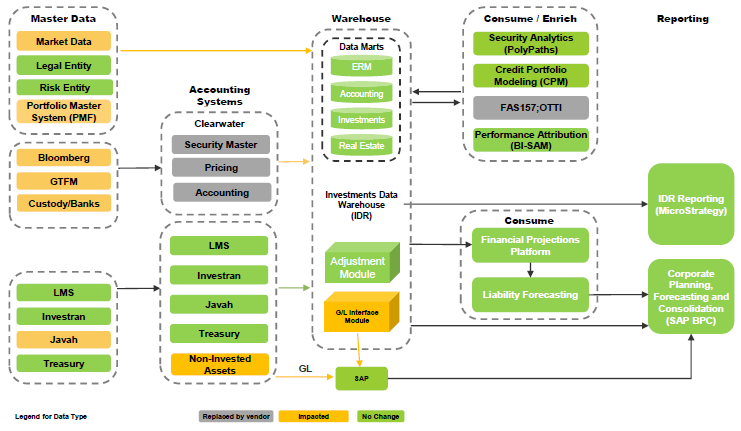


## APPLICATION DEVELOPMENT AGILE



# CLEARWATER

## NOVA Target State



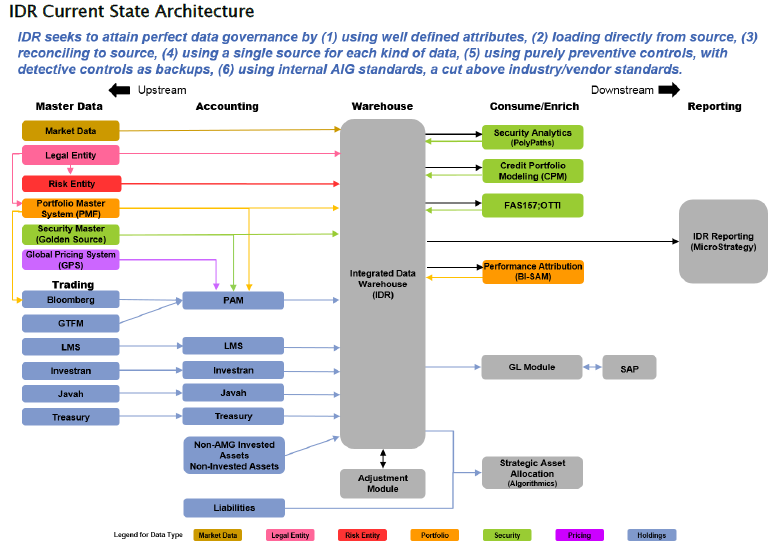
## IDR Data Governance



## IDR Data Quality Lifecycle

|  |  |
| --- | --- |
|  |  |

## IDR State Architecture



|  |  |
| --- | --- |
|  |  |

## NON-CLEARWATER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **\*IT Dependency Name** | **\*IT Dependency Type** | **\*Application 1** | **Associated Business Process(es)** | **FCU Contact** |
| Microstrategy - IDR Holdings Report OLD Microstrategy - IDR Holdings (TAX) Report NEW | Key Report | Microstrategy | Investment Accounting / Investment Operations / OTTI | Anthony/Jasmin |
| Level Change Report (2016) | Key Report | IDR | Financial Reporting (Leveling) | Cassie |
| Rule View Compare Report (2016) | Key Report | IDR | Financial Reporting (Leveling) | Cassie |
| Clearwater Multisource pricing file | Key Report | Clearwater | Clearwater | Cassie |
| Balance Sheet Queries / Price Exception Report | Key Report | IDR | Clearwater | Cassie |
| Private Placement Valuation PPV to Clearwater(CW) & Clearwater(CW) to PPV Interface Private Placement Valuation Yield to Price Calculation & Price to Yield Calculation | Key Interface | PPV/Clearwater | Clearwater | Cassie |
| Argus Quarterly Valuation Comparison Report (2016) | Key Report | Argus Enterprises | Global Real Estate (GRE) | Gene |
| GAAP Variance Analysis report | Key Report | BPC | Cap Corp | Gene |
| IDR Screening Report for Embedded Derivative Purchases (2016) | Key Report | Microstrategy | Fair Value Option | Gene |
| Interface SAP ECC to BPC | Interface | FW and SAP ECC | Cap Corp | Gene |
| Investran - Holdings Report (2016) | Key Report | Investran | Alternative Investments / Investment Accounting / OTTI | Gene |
| Investran Cash Activity Report (2016) | Key Report | Investran | Alternative Investments | Gene |
| Investran Valuations Report (2016) | Key Report | Investran | Alternative Investments | Gene |
| Javah to SAP Interface (2016) | Interface | Javah | Derivatives | Gene |
| LMS to IDR (2016) | Interface | LMS | CML | Gene |
| Microstrategy - Tax IDR Transactions Detail OLD Microstrategy - (GAAP/STAT) IDR Transactions Detail - **Purchases & Sales** | Key Report | Microstrategy | Alternative Investments / Investment Accounting | Gene |
| Microstrategy - IDR Transactions Detail Report - **Paydowns & Other Transactions** | Key Report | Microstrategy | Alternative Investments / Investment Accounting | Gene |
| R100 Validation Report (BPC TM1) (2016) | Key Report | BPC and IDR via TM1 | Investment Accounting | Gene |
| Schedules 46 & 60 Consolidated Report (2016) | Key Report | Javah | DAG | Gene |
| IDR DB Flow Report Daily | Key Report | IDR | Alternative Investments | Gene |
| IDR DB Flow Report Monthly | Key Report | IDR | Alternative Investments | Gene |
| IDR Income Earned (Tax Basis) Report OLD  Income Earned (GAAP/STAT) Report NEW | Key Report | Microstrategy | Investment Tax | Gene |
| Polypaths Cash Flow File / Securities of Interest List / FAS-91 | Key Report | IDR | Clearwater | Gene/Cassie |
| AIG Debt Repurchases Report (2016) | Key Report | JAVAH | Long-term debt | James |
| Autocheck GIC Draws/Deposits (2016) | Key Report | Autocheck | Derivatives | James |
| BizTalk to SWIFT | Interface | Biztalk | GCM Operations | James |
| Bloomberg to TSA Interface (2016) OLD Bloomberg to JAVAH Interface (2016) NEW | Interface | Bloomberg | AIGFP Investments / GCM Ops / Investment Operations | James |
| CA Change Report | Key Report | SEC Haircut | Derivatives | James |
| Cash Flow Oracle Schedules (2016) | Key Report | Oracle | AIGFP Financial Reporting/Intercompany | James |
| JAVAH to AutoCheck Interface | Interface | Javah | Derivatives | James |
| Javah to SACM (2016) | Interface | Javah | Multiple processes GCM Operations | James |
| JAVAH to TRS Interface (2016) | Interface | JAVAH | Derivatives | James |
| LMS Alter Due Audit Report (2016) | Key Report | LMS | CML | James |
| LMS Daily Cash Detail Report (2016) | Key Report | LMS | CML | James |
| LMS Loan Terms Audit Report (2016) | Key Report | LMS | CML | James |
| LMS Trial Balance Report (2016) | Key Report | LMS | CML | James |
| MAT (Manual Adjustment Table) Change Report | Key Report | SCA | Derivatives | James |
| Quick CA Report | Key Report | SEC Haircut | Derivatives | James |
| Rate Reset Report (2016) | Key Report | PROF | Derivatives | James |
| SACM to Biztalk | Interface | Biztalk | GCM Operations | James |
| SACM to DTDF (2016) | Interface | SACM | GCM Operations | James |
| SCA Daily Call Activity Report | Key Report | SCA | Derivatives | James |
| SSRS (SMF Change Report) | Key Report | JAVAH | Derivatives | James |
| Survelliance Report (2016) (renaming to watchlist report) | Key Report | LMS | CML | James |
| FAS 157 leveling to Clearwater (nightly Extract of Xnet ID, BAC, FAS157 and AIG derived rating from IDR datamart to CW) | Key Interface | IDR/Clearwater | Clearwater | James |
| Form B Transfer Trans Report (Asset Transfer Population Report) | Key Report | Clearwater | Clearwater | James |
| Asset Transfer Transaction Report | Key Report | Clearwater | Clearwater | James |
| Cross Trades Report | Key Report | Clearwater | Clearwater | James |
| Banks to OpenLink (SWIFT/Biztalk to Openlink) | Interface | OpenLink | Corporate Treasury | Jasmin |
| FNSCS Non-Statutory Inventory Report and LOC | Key Report |  |  | Jasmin |
| OL Make/Checker - Payee Setups | AAC | OpenLink | Corporate Treasury | Jasmin |
| OL SOD Wire - Enter/Modify/Release (2016) | AAC | OpenLink | Corporate Treasury | Jasmin |
| OLE Balancing Report | Key Report | OpenLink | Corporate Treasury | Jasmin |
| OLE Bank Transaction Report (2016 new) | Key Report | OLE | Corporate Treasury | Jasmin |
| OLE Daily Adjustments (Back Value) Report | Key Report | OLE | Corporate Treasury | Jasmin |
| OLE Remote Wire Upload Report | Key Report | OLE | Corporate Treasury | Jasmin |
| OpenLink maker/checker - Template Setups | AAC | OpenLink | Corporate Treasury | Jasmin |
| OpenLink to Biztalk | Interface | OpenLink | Corporate Treasury | Jasmin |
| SAP CD - OL | Interface | SAPCD | Corporate Treasury | Jasmin |
| OLE Remote Wire Status Report | Key Report | OpenLink | Corporate Treasury | Jasmin |
| Source systems to CDCS | Interface | CDCS | Corporate Treasury | Jasmin |
| Stat Security Change Report/LOC Statutory Inventory Report | Key Report | LOC | Corporate Treasury | Jasmin |
| WDS to OpenLink | Interface | WDS | Corporate Treasury | Jasmin |
| WDS to ORD/Trust | Interface | WDS | Corporate Treasury | Jasmin |
| Ariba is updated daily, via automatic feeds, with data from WDS. This includes changes in employment status  Any issues related  to the data feeds are addressed Ariba Systems and Operations in a timely basis. | Interface | WDS | Other General Operating Expenses | Jasmin |
| Fieldglass is updated daily, via automatic feeds, with data from WDS. This is an Auto Feed, Daily Load, which includes adding / removing users. In-coming feeds Auto-Monitored by Off-Shore – Shared Services Team and Accenture Team. | Interface | WDS | Other General Operating Expenses | Jasmin |
| SAP EP1 to Biztalk for wire payments | Interface | SAP Ep1 | AP (Other General Operating Expenses) and Treasury | Jasmin |
| Same individual is unable to both setup and release purchase order | AAC | Ariba | Other General Operating Expenses | Jasmin |
| Approver limits are checked by the system. | AAC | Ariba | Other General Operating Expenses | Jasmin |
| Fiscal authority limits is automatically updated in Fieldglass from WDS | AAC | Fieldglass | Other General Operating Expenses | Jasmin |
| Requisition/Work Order is automatically routed to the appropriate individuals for approval based on fiscal limits, and the SOW is created based on the approved terms and amounts. | AAC | Fieldglass | Other General Operating Expenses | Jasmin |
| SAP is configured to perform a three way match between the Purchase Order, Invoice, and Receiving Documents.  SAP is set up to reject Invoices (block the payment) if an item does not match. | AAC | SAP AP | Other General Operating Expenses | Jasmin |
| Invoices submitted on an SOW that cause the total amount paid on the SOW to exceed the total amount approved for the SOW are rejected by Fieldglass and will not be sent for payment. | AAC | Fieldglass | Other General Operating Expenses | Jasmin |
| When invoices are created in SAP VIMS, automated controls ensure that key mandatory fields (e.g., vendor, bank, company code, invoice reference field) are populated and then these key mandatory fields are validated against the Master SAP Data. | AAC | SAP AP | Other General Operating Expenses | Jasmin |
| System prevents the Payment Proposer and Payment Executor from modifying the payee and amount on any invoice.  Payment method and paying bank account can be changed. | AAC | SAP AP | Other General Operating Expenses | Jasmin |
| Once the vendor master data is approved upon data entry, systematic validation is performed to identify any errors on the specific fields on the vendor information.  The request cannot be saved until all mandatory fields by account group are entered.  Any errors identified are manually re-routed to the appropriate party for resolution. | AAC | Vendor Master (SAP) | Other General Operating Expenses | Jasmin |
| Any attempt to create or modify sensitive fields automatically results in a temporary vendor payment block and sensitive reviewer must approve the sensitive fields to unblock. System automatically prevents the same ID from Vendor Master Data Maintainer and the Sensitive Field Approver to confirm. Sensitive data is masked to avoid violation of Privacy Laws. | AAC | Vendor Master (SAP) | Other General Operating Expenses | Jasmin |
| COA change report | Key Report | SAP EP1/DOA (WDS) | Other General Operating Expenses | Jasmin |
| SE16N (report\_LFA1) | Key Report | SAP EP1 | Other General Operating Expenses | Jasmin |
| SAP- CD to Biztalk | Key Report | SAP - CD | Treasury | Jasmin |
| Appian - SOD Wire Enter/Modify/Release | AAC | Appian-Wire | Corporate Treasury | Jasmin |
| Appian to OpenLink Interface | Interface | Appian-Wire | Corporate Treasury | Jasmin |
| ACH 824 Acknowledgement Reconciliation Report | Key Report | CDCS | Corporate Treasury | Jasmin/Allen |
| ACH 997 Acknowledgement Reconciliation Report | Key Report | CDCS | Corporate Treasury | Jasmin/Allen |
| CDCS Automated Balancing - Automated balancing routines are in place to ensure that transmitted files to CDCS are accurate and complete. (Source systems to CDCS). | AAC | CDCS | Corporate Treasury | Jasmin/Allen |
| CDCS Interface Balance Report (CDCS auto balancing routine reports)(CDCS to Emdeon) | Key Report | CDCS | Corporate Treasury | Jasmin/Allen |
| IDR to Revport Interface | Interface | Revport | AMG US |  |

## CLEARWATER

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Control Name** | **Report Name** | **Description of Nature/Usage** | **Input Source  (i.e. system feeding this report)** | **Describe How Report is Generated(i.e. Microstrategy, Canned Report, Query, etc.)** | **RA Comments** | | |
| Reconcile consolidated price exception report to validation results | Clearwater Multisource pricing file | File contains alternate prices for the bonds and equity portfolio for any given as-of date. This is used as an input to the Excel EUC (Exception Generation tool) as well as for the stored procedures. | Clearwater | CW sent through SFTP or IVFA Downloaded from the Clearwater Portal | In Scope | | |
| Private Placements | Amortization schedules from Clearwater for WAL calculation | (Anthony Versaci) File received from Clearwater for Amortization schedules. This file is sent via SFTP and Analytics IT has a process to copy from IT into our production folder for load into Oracle. | Clearwater | CW sent through SFTP and and Analytics IT has a process to copy from IT into our production folder for load into Oracle. | AL Comparison Report for Q1. Different amortzation schedules are used by investment analytics to recalculate a WAL then the AL comparison report compares mgmt's WAL, vs investment analytics WAL and also shows the change quarter over quarter in WAL. | | |
| Reconcile consolidated price exception report to validation results | Balance Sheet Queries /  / Price Exception Report | Report run out of Microstrategy to obtain MVs of the full population of securities that make up the fair valued bond and equity balance sheet line items of AIG's GAAP consolidated balance sheet. | IDR | Microstrategy | In Scope | | |
| The "Securities of Interest List" key report that existed pre-CW is no longer used | Polypaths Cash Flow File / Securities of Interest List / FAS-91 | The investment analytics team now uses a cash flow file from Polypaths. Stephen DeTommaso and Anthony Versaci are the contacts. | IDR | Microstrategy | In Scope | | |
| Review Asset Transfer reconciliation | Form B Transfer Trans Report (Asset Transfer Population Report) | Report run from the Clearwater website to show all asset transfers processed during a specified time period. | Clearwater | Run from the Clearwater website | In Scope | | |
| Review Asset Transfer reconciliation | Asset Transfer Transaction Report | Used to ensure that all asset transfers are executed at Clearwater | Clearwater | Run from the Clearwater website | In Scope | | |
| Review report identifying cross trades | Cross Trades Report | To identify buy/sell trades entered by the front office that should have been booked as interportfolio transfers. | Clearwater | Run from the Clearwater website | In Scope | | |
| Review and approve Price Override Report | Price Override Report | Clearwater process treats manually instructed prices as overrides. Dublin Operations is planning to generate and periodically review the Clearwater override report.  As of 11/9 Reliance on Clearwater for complete and accurate recording of manual prices | Clearwater | Run from the Clearwater website | Per FCU, the report is not used by the business. Confirmed with Core | | |
| Identifies updated SMF records in which the tax fields are updated based on instructions from ITG. | Clearwater New Positions Report | Appian Confirmations Report (Appian to ITG Tax Class Reconciliation) No longer used. Investments Tax Group (ITG) will periodically download a CW new positions report and review the new SMF tax attributes | Clearwater | Run from the Clearwater website | Per FCU, the report is not used by the business. Confirmed with Core | | |
| Approve transactions entered in Bloomberg by NY Front Office | Trade Blotter Report from CW | PM's review and approval of trades listed on the blotter. | Clearwater | TSA/Dart Custom Report | Testing Blotter - Structured NY - Year to Date - IA | | |
| Review and approve JAVAH transaction trade blotters | Trade blotter from Javah | To identify and evidence approval of all trades that occur on a given day. | JAVAH | Canned Report | No testing required | | |
| Resolve exceptions for DQA reports | Data quality exception reports | Used to verify data quality of Microstrategy reports | Javah, LMS, CW, Investran | For the sub-ledger feed Clearwater will replace PAM | Meeting Requested | | |
| Review and sign off on final OTTI details for equity securities population | PAM report of equity securities in an unrealized loss position | Used to assess securities for write downs | Clearwater | Feed from CW | Per FCU, the report is not used by the business. Confirmed with Core | | |
| Validate completeness of OTTI population | OTTI Module Rule-Set Report (OTTI Evaluation Portfolio Listing) (i.e. Listing of in-scope portfolios)) | Used to determine the in-scope population for OTTI | Microstrategy | Feed from CW | Per FCU, the report is not used by the business. Confirmed with Core | | |
| Review RML cash reconciliation | Clearwater Cash Rec Report (RML) | Identifies reconciling items resulting from the RML Cash reconciliation performed by Clearwater. | Clearwater | Downloaded from the Clearwater Portal | Confirmed with RML, Non-Key | | |
| Review RML Trial Balance Reconciliation | Clearwater GAAP Base Balance Sheet by Lot Report (RML Holdings Report) | Used as an input for the RML outstanding principal balance reconciliation. | Clearwater | Downloaded from the Clearwater Portal | Confirmed with RML, Non-Key | | |
| Review and approve changes to FAS 157 leveling rules | FAS 157 leveling to Clearwater (nightly Extract of Xnet ID, BAC, FAS157 and AIG derived rating from IDR datamart to CW) | IDR to CW | Clearwater | Global Operations - Dublin Operations | Real feed or extract sent via email? We test 2 reports here Level Change Report & Rule View Compare Report.  TEST: Rule View Compare Report | | |
| Review and approve Private Placement valuation | Calculated prices or yields from Clearwater back to PPV | CW to PPV | Clearwater | AIGI Front Office - Private Placements | Peter - Completeness + Calcs | | |
| Review and approve JAVAH transaction trade blotters | Trade details of trades executed on Bloomberg to Javah | BBG to JAVAH | Clearwater | AIGI Front Office - GCM | Non-Key. Removed. FCU tests Recs | | |
| Review and approve Private Placement valuation | Prices or yields from PPV including calculation + PPV to Clearwater, Clearwater to PPV (yield to price) | PPV to CW, CW to PPV | Clearwater | Price calculation for private placement securities will be performed in Clearwater. The PPV data is sent to and received from Clearwater via web service connection. | Peter - Completeness + Calcs | | |
| Review Asset Transfer reconciliation | Report from PAM Direct detailing the transfers | Used to ensure that all asset transfers are executed in PAM | Clearwater | Downloaded from the Clearwater Portal |  | | |
| Review and approve asset reconciliations | Reconciling Items Aging Reports | To age reconciling items. | Clearwater | Oversight control to be implemented by Dublin Investment Ops for review and monitoring of aging items.  Per Dublin Investment Ops, the report is not yet available as of 10/24/18. | |  | |
| Review and approve cash reconciliations | Reconciling Items Ageing Reports | To age reconciling items. | Clearwater | Oversight control to be implemented by Dublin Investment Ops for review and monitoring of aging items. Per Dublin Investment Ops, the report is not yet available as of 10/24/18. | | |  | |
| Income analyzer (is this used as a monitoring control?) | | |  | Non-Key |  | | |
| Monitoring of aging of reconciliation aging (what additional controls is management planning to perform with this report?) | | |  |  |  | | |