VIA 504E – Big Data Technologies and Applications

Homework 1

528211011 – Hakan Can İpek

Q1 : Please discuss full stacked pipeline for Big Data Application Pipeline – **Recent big data tools** and **big data platforms comparision between on-prem and cloud – use code based statistical analysis**. Is there any increase wrt other? Please use advanced statistical interpretation for your findings?

A1 : Öncelikle tüm resme bakmadan bir pipeline nedir ondan başlayalım.

İsminden anlaşılacağı üzere bir boru hattı gibi düşünebiliriz. Bilgisayar kavramı olarak işlevi, birden çok process in sıra ile iş yapmasını sağlamak olarak diyebiliriz. Lisans zamanında bir hocam pipeline ile alakalı enteresan ama her zaman aklımın köşesinde kalan bir örnek vermişti. Pipeline ı bir sandviççiye benzetmişti. Ve sandviç hazırlamanın her aşamasından sorumlu birilerinin olduğunu düşünmemizi istemişti. İlk kişi(fetch) ekmeği ikiye bölüyor, ikinci kişi(decode) malzemeleri hazırlıyor, üçüncü kişi(execute) ekmek ve malzemeleri birleştiriyor ve son kişide(store) son ürünün paketlemesinden sorumlu oluyordu. Bu sayede her kişi kendi işiyle sorumlu ve birden fazla sipariş geldiği zaman her siparişin bu süreçten sıra ile geçmesi gerekiyor. İlginç bir örnekti.

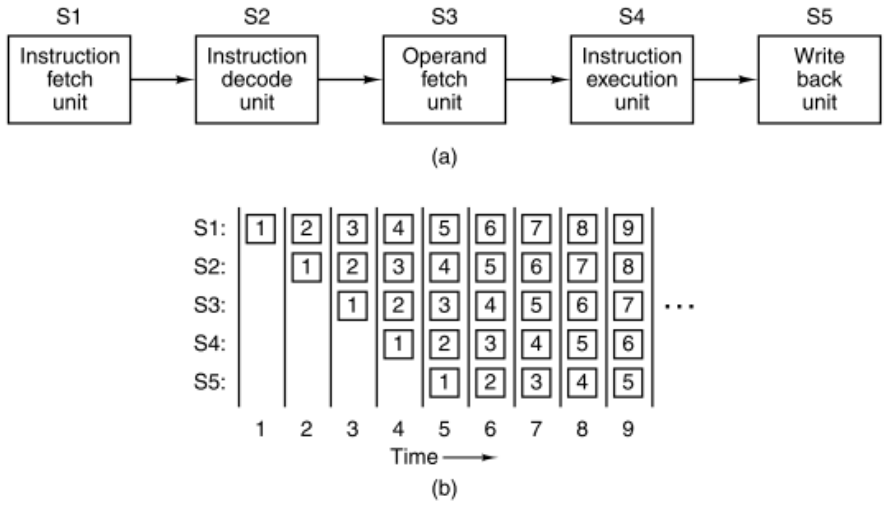
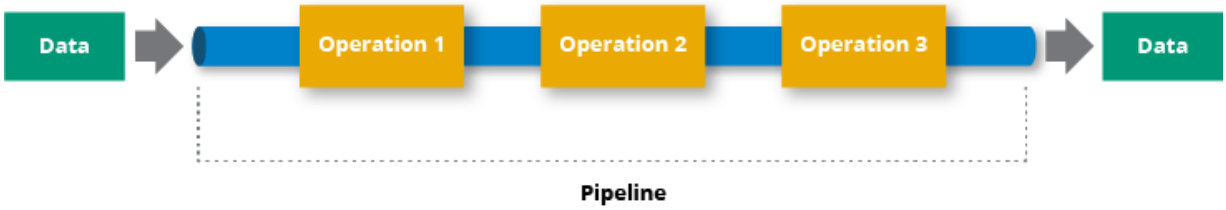


Figure 2-4. (a) A five-stage pipeline. (b) The state of each stage as a function of time. Nine clock cycles are illustrated.

Data Pipeline data işleme serilerine denir. Eğer data mevcutta data platformuna yüklenmedi ise, pipeline başlangıcından giriş yapar. Devamında her adım sıradaki adım için girdi olacak çıktılar üreterek devam eder. Bu süreç pipeline tamamlanana kadar devam eder. Bağımsız süreçlerin paralel olarak devam ettiği durumlar da mevcuttur. Data pipeline üç ana element içerir. Kaynak noktası, işlem adım veya adımları, hedef noktası. Data pipeline örnek olarak uygulamadan(application) veri deposuna(data warehouse), veri havuzundan analitik veritabanına veya bir ödeme işleme sistemine veri akışını sağlıyor olabilir. Aynı zamanda birden fazla data pipeline ı aynı kaynağa sahip olabilir ve tek görevi veri setini düzenlemek olabilir.

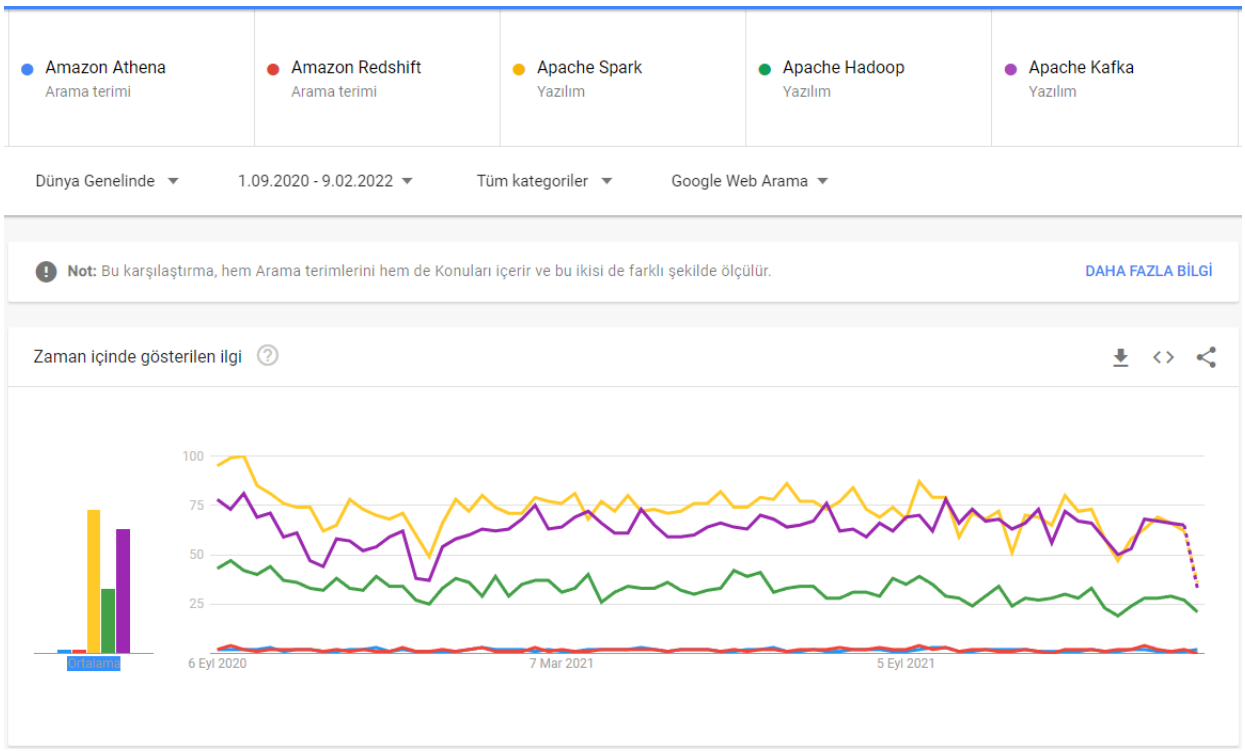
Big Data Pipeline ise standart 'data pipeline'ların büyük veriyi(big data) işlemek için gelişmiş halleridir. Büyük verinin 3 ana özelliğinin(Volume, Velocity, Variety) bir yada daha fazlasını karşılamak için geliştirilmişlerdir. Velocity, büyük veriler için data pipeline akışı oluşturmayı hedefler. Volume, 'data pipeline'ların ölçeklenebilmesini gerektirir. Variety, big data pipelineların çeşitli formatlardaki

veriyi(yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış) tanımasını ve işleyebilmesini hedefler.



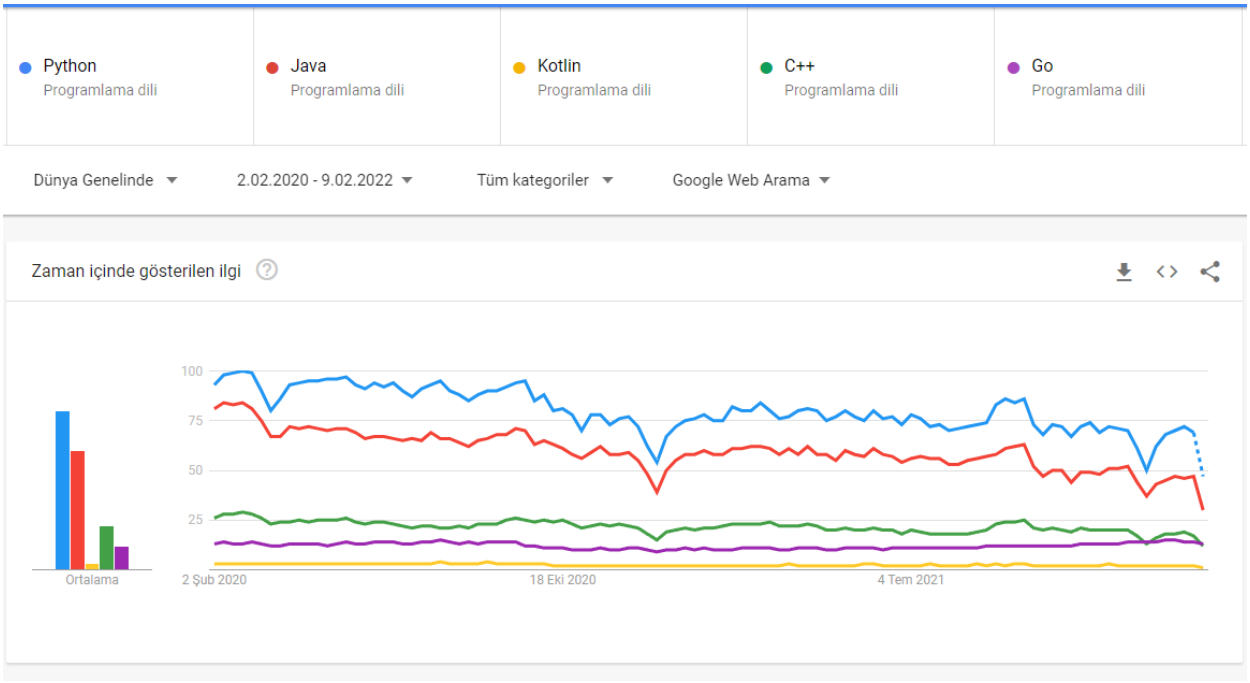
Pipeline

Data mühendisliğinde öne çıkan uygulamalar baktığımızda, Amazon Athena, Amazon Redshift, Apache Spark, Apache Hadoop, Apache Kafka gibi uygulamaları görüyoruz. Programlama dili olarak Python, sorgulama dili olarakta SQL in öne çıktığını görüyoruz.



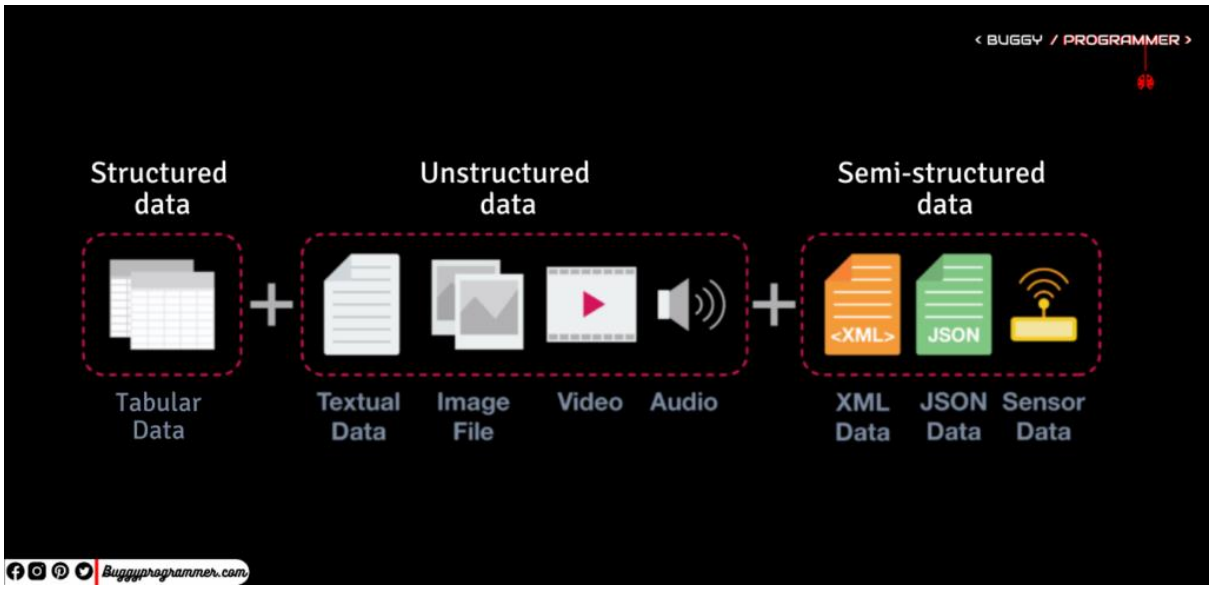
Amazon uygulamaları genellikle ücretli ve başka Amazon servislerine bağımlı oldukları için kısmen trendlerde daha düşük gözükmektedirler. Apache açık kaynaklı ve ücretsiz yazılımlar sundukları için firmalar tarafından daha çok tercih edilmektedirler.

Programlama Dilleri açısından bir kıyaslama yaparsak,

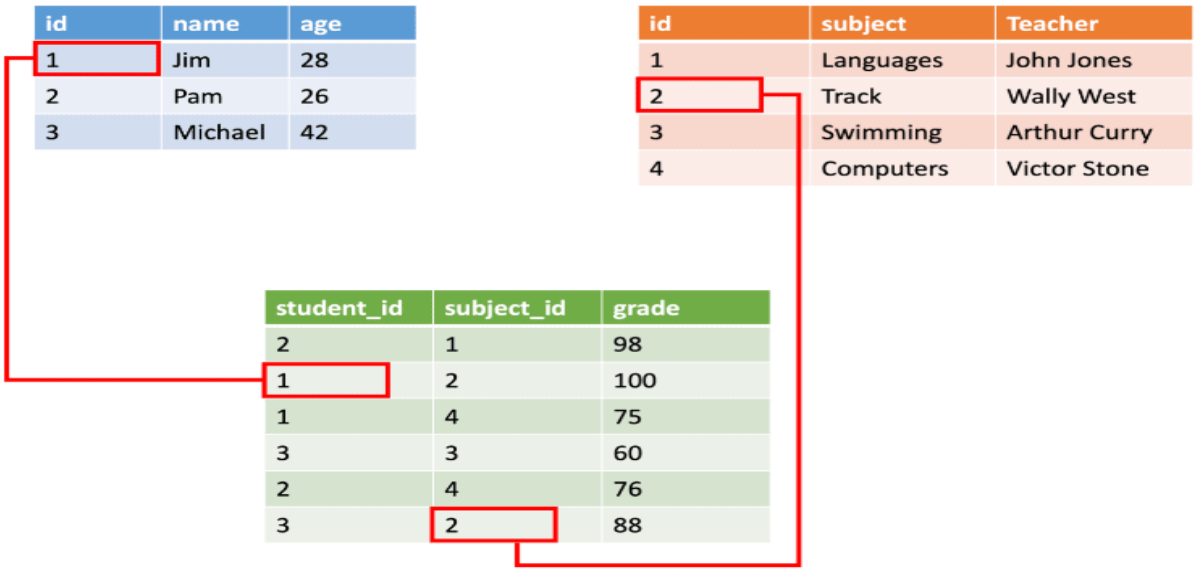


Son 2 sene içerisindeki grafiği incelediğimizde genel olarak programlama dili olarak Python un diğer dillere kıyasla daha çok ilgi gösterildiğini görebiliyoruz.

Q2: **Discuss data types and importance for big data small-and-wide data proposals** (structured, unstructured and semi structured nature), please draw a chart for all possible BD application chain – technical wrt programming language operation.



A2: ***Structured Data*** genellikle tablo şeklinde olan, satır ve sutunlar ile bir veri tabanında ifade edilebilir verilerdir. Bu verileri tutan veri tabanları genellikle İlişkisel Veri Tabanları(Relational Database) olarak adlandırılır, bunların yönetiminde kullanılan sistemlere de RDBMS(Relational Database Management System) denir. Buradaki ilişki iki veya daha fazla adet olan tablolar arasındaki bağlantıyı ifade eder, genellikle tabloların belirli kolonları arasında(çoğunlukla nümerik id kolonları) olmaktadır. Structered data ait tüm satırlar aynı tipteki nesneyi ifade eder ve her nesnenin aynı özellikleri(property) yani kolonları bulunmaktadır. Aynı zamanda bu veri gruplarının her kolonu kendi içinde aynı veri tipinde tanımlanır. Yani herhangi bir nesneye ait bir kolon her nesne için aynı veri tipi özelliğini gösterir(nümerik, karakter dizisi, zaman, ondalıklı sayı gibi).



Semi Structured data ise, bir structered yapı gibi kesin sınırlar içermez, fakat kendi içerisinde belli sınırları bulunur. Bu tarz veriler genellikle JSON(JavaScript Object Notation) formatında dökümanlardan oluşur(Mongo, Couchbase vb.). Ayrıca Anahtar-Değer(Key-Value) depoları(Redis) ve Graf(graph) veritabanları(Neo4j) da bunlara örnektir. Structured yapı ile aralarında bir ilişki kurmaya çalışırsak buradaki JSON Property diyebileceğimiz kavramlar kolonlara karşılık gelebilir. Ve bu property karşılığına gelen değerlerde bir kolon değeri gibidir. Buradaki hassas fark ise şu şekildedir, bir JSON içerisine ihtiyacımıza göre farklı propertyler ekleyebiliriz ve bunun için ekstra bir masraf yada çabaya ihtiyaç yoktur, ama bir structered yapıda, her bir kolon eklenmesi demek tabloda yer alan tüm verilere bir kolon eklenmesini gerektirir ve bu zahmetlidir, ayrıca tablonun boyutu ne kadar büyükse o kadar zaman gerektirir. Aynı şekilde bu property değerlerini belirli bir data tipi ile ilişkilendirmemiz gerekmez, bu tamamen bizim tercihimizle alakalıdır(fakat uygulamalarda çoğunlukla daha temiz bir kod yazımı ve daha iyi nesne yönetimi açısından aynı veri tipinde olmasına özen gösterilmesi daha doğrudur). Son olarak Structered yapılar genellikle Primitive diyebileceğimiz veri yapılarını

tutmak üzerine tasarlanmıştır ve wrapper(birden çok primitive ve/veya başka wrapperların içerisinde bulundurulduğu büyük nesne) diyebileceğimiz daha büyük nesneleri tutabilmek için çoğunlukla bir ilişki kurularak yeni bir tablo kullanımı ihtiyacı hissederler. Fakat bu tarz semi structered veriler için böyle bir ihtiyaca gerek yoktur, bir JSON objesi içerisinde ister primitive, ister wrapper her çeşit yapıyı bulundurabiliriz.

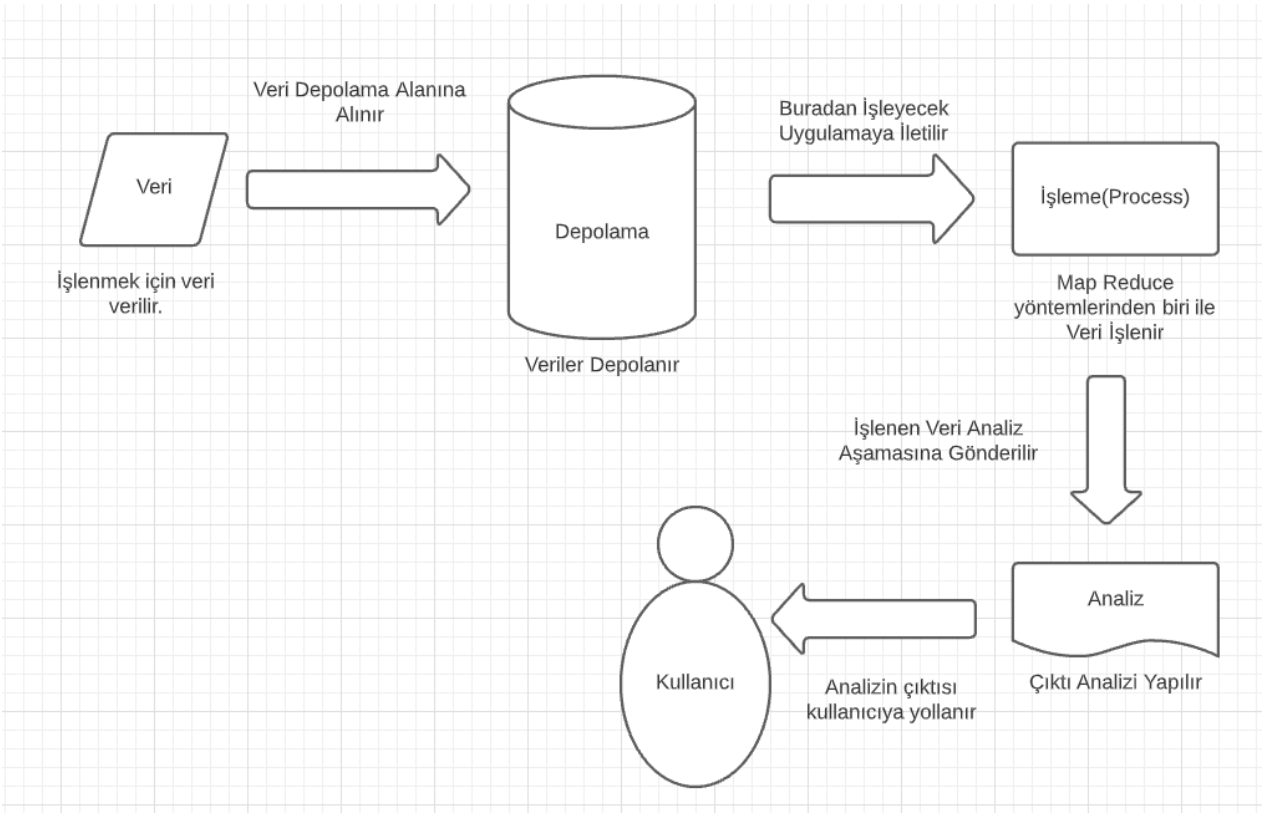
```
## Document 1 ##
{
  "customerID": "103248",
  "name":
  {
    "first": "AAA",
    "last": "BBB"
  },
  "address":
  {
    "street": "Main Street",
    "number": "101",
    "city": "Acity",
    "state": "NY"
  },
  "ccOnFile": "yes",
  "firstOrder": "02/28/2003"
}
```

Unstructured Data ise, önceden tanımlanmış bir modele sahip olmayan ya da önceden tanımlanabilecek davranışları olmayan verileri ifade eder. Unstructured bilgiler genellikle yazı ağırlıklı setlerdir ama numerik, zaman gibi değerlerde içerebilir. Videolar, sesler ve ikili veri dosyaları genellikle bu tür içerisinde yer alır.

Büyük veri uygulamalarında bu veri tipleri ile çalışırken kullanılan veri yapıları büyük önem taşır. Örneğin eğer kullanacağımız veriler structured ise bu bize uygulamalarımızda verileri daha kolay işlememize yardımcı olacaktır. Çünkü alacağımız verinin içeriğini önceden biliyor olacağız. Bu veride neler olacabileceği ve neler olamayacağı belli olduğu için uygulamalarımızda buna göre bir yaklaşım benimseriz ve daha hızlı ve daha kolay geliştirilebilir uygulamalar tasarlayabiliriz. Buradaki sıkıntı ise yaklaşımımız çoğunla daha statiktir, bu veri tipi genişledikçe bizimde uygulama içerisinde yaklaşımlarımı geliştirmemiz gerek, yeni gelen her özellik için yeni bir yaklaşım ve yeni bir işleme süreci eklememiz gerekecektir. Semi-

Structured yapılarla çalışırken ise elimizde verinin genel yapısını biliyor olup buna göre belirli değişimlere uyum sağlayabilecek bir esneklikte yaklaşımlar sergileriz. Bu yaklaşımlarda genelde ortak paylaşımda buluşulması hedeflenir. Verinin içeriğine daha genel bir bakış açısı ile bakılarak görece daha dinamik olmak gerekecektir. Dinamik yaklaşımlar genellikle tasarımı ve uygulanması daha uzun sürer, fakat daha uzun süreler sürdürülebilir ve bakımı görece daha kolaydır. Birçok işlemi halledebilen ufak parçalar ile büyük resime ulaşılmaya çalışılır.

Unstructural yapılarda ise bir bilinmezle uğraşılacaktır. Buradaki yaklaşımlarda genel çözümler veya statik çözümler genelde işlemez. Davranış odaklı gitmek gerekir, örneğin bir ses verisi ile uğraşılıyorsa bu sesin davranışları araştırılmalı ve bu araştırma sonucunda yaklaşımlar belirlenmelidir. Örneğin bir arama sonucu gelen yazılardan oluşan veri kümesini incelerken Regular Expression(düzenli ifade, RegEx) kullanılan yaklaşımlar benimsenebilir.


Veri
İşlenmek için veri verilir.
Veri Depolama Alanına Alınır
Depolama
Veriler Depolanır
Buradan İşleyecek Uygulamaya İletilir
İşleme(Process)
Map Reduce yöntemlerinden biri ile Veri İşlenir
İşlenen Veri Analiz Aşamasına Gönderilir
Analiz
Çıktı Analizi Yapılır
Analizin çıktısı kullanıcıya yollanır
Kullanıcı

Q3: **What are the highlighted big data projects in last two years?** As stated in the first bullet, please check details from trends data – i.e. using keyword for international airline companies . Is there any correlation wrt search keywords? What is popular tools (services) in recent years? How you quantify it? Please support your statements with your findings and discuss it in detail?

A3: Popüler araçlara baktığımızda karşımıza şu sonuçlar çıkmıştır.

- ***Snowflake***, bulut tabanlı bir veri deposu ve analitik servis sağlayıcısıdır. Çalışırken scale edilebilir, otomatik cluster edilebilir ve ODBC, JDBC, Javascript, Python, Spark, R ve Node js ile entegrasyon kolaylığı sağlar. 3 katmanlı bir mimariye sahiptir, bunlar Veritabanı Deposu, Sorgu İşleme ve Bulut servisleridir.

- ***Amazon Redshift***, petabyte seviyesinde scale edilebilir data warehouse çözümüdür. Veri bilimcileri, veri analistleri, veri yöneticileri ve yazılım geliştiriciler için tasarlanmıştır. Paralel işleme ve karşılaştırma algoritmaları kullanıcılar milyarlarca satır üzerinde operasyonlar yapmalarını sağlamaktadır ve işlem süresini ciddi oranda düşürmektedir. Günümüzün büyük miktarlardaki verilerini analiz etmek için mükemmel bir araçtır, çünkü birden fazla data warehouse kullanır.

- ***Hevo Data,*** kod içermeyen, yeni çağ işlerinin birden fazla kaynaktan gelen verilerini bir data warehouse entegre etmesini

sağlayan ve bu datanın herhangi bir BI aracında veya iş uygulamasında kullanılmasına yardımcı olan bir data pipeline platformudur. Bu platform, 30 dan fazla ülkedeki 100 den fazla veri odaklı çalışan organizasyonların güvenini kazanmış 100 den fazla kullanıma hazır entegrasyon içerir.

- ***Google BigQuery,*** analitikler için kurumsal düzeyde yönetilebilir ve sunucusuz bir veri ambarıdır. Günümüz veri analitikçilerinin ve veri bilimcilerinin verimli veri analizi yapma konusunda güçlendirir. Ana özellikleri BigQueryML, BigQuery GIS, BigQuery BI Engine ve connected sheet lerdir. İçgörüler, güçlü iş kararları, analitik çalıştırmalar ve petabyte seviyesinde SQL sorgu analizi için çok güçlü bir çözümdür. Dremel teknoloji üzerine kurulmuştur ve sunusuz bir mimariye sahiptir. Optimum bir performans için Bor, colossus, Jupiter ve Dremel gibi teknolojilerden yararlanır.

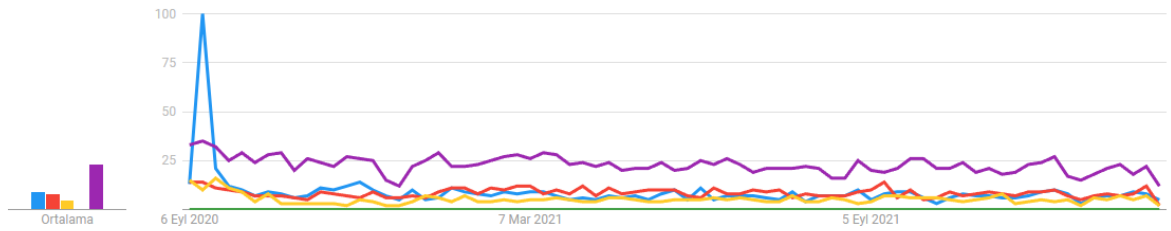
Şimdi bu teknolojilerin son iki yıldaki Google Trends durumlarına bakalım.

Snowflake	Redshift	BigQuery	Hevo Data	Apache Hadoop
Şirket	Arama terimi	Arama terimi	Arama terimi	Yazılım

Dünya Genelinde | 2.09.2020 - 9.02.2022 | Bilgisayarlar ve Elektronik Ür... | Google Web Arama

Not: Bu karşılaştırma, hem Arama terimlerini hem de Konuları içerir ve bu ikisi de farklı şekilde ölçülür. DAHA FAZLA BİLGİ

Zaman içinde gösterilen ilgi



Bölgeye göre karşılaştırmalı döküm

Bölge