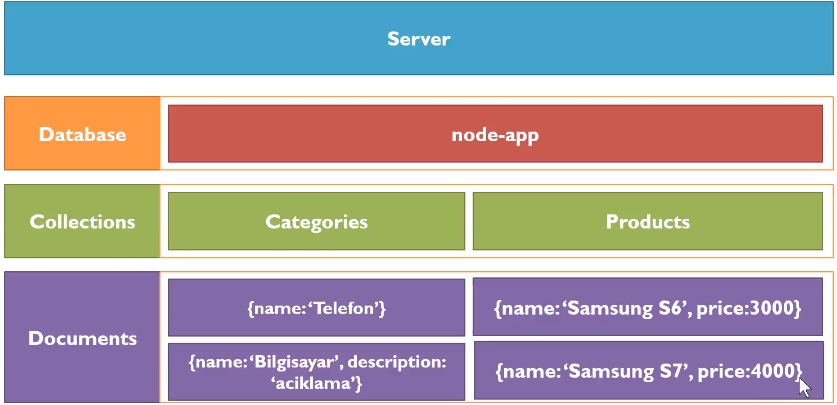
SQL’deki table bilgileri, NoSQL’deki “collections”’lara karşılık gelmekte olup, rowlar “documents”, kolonlar ise “fields” olarak adlandırılır. “documents” kısmında dictionary benzeri bir yapı bulunmaktadır.





Big data’da mongoDb tarzındaki NoSQL sistemi document yapısı sayesinde SQL’e göre daha performanslı çalışmaktadır. Farklı tablolar arasında ortak veriler birleşimi sağlar ve bu sebeple ekstradan tabloları birleştirme işlemine gerek kalmaz, bu işlem özgürlüğü sistemin SQL’e göre daha performanslı çalışmasını sağlar.

“import pymongo” ve pymongo server’ını kullanabilmek için “pymongo” ve “pymongo[srv]” paketlerinin yüklenmesi gerekir.



Yerel bir host’a bağlanmak için gerekli python kodu yukarıdaki gibidir, “myclient[]” komutu istenilen database’e ulaşımı sağlar, eğer öyle bir şey yok ise oluşturulur fakat oluşturulabilmesi için database içeriğinde “collection” barınması gerektiğinden onu da kod olarak girmemiz gerekir aksi takdirde yukarıdaki kod, olmayan bir “node-app” birimi için error verecektir.

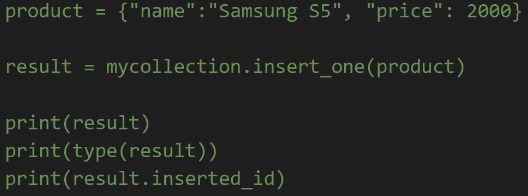
Server’ımıza bağlanmak için ise “MongoClient” içeriğindeki parantezi sistem tarafından oluşturulmuş link ile değiştirmemiz gerekiyorken, bu linki yazarken “<password>” ve “test” kısımların belirlenen şifre ve database ismi ile değiştirmemiz gerekir.



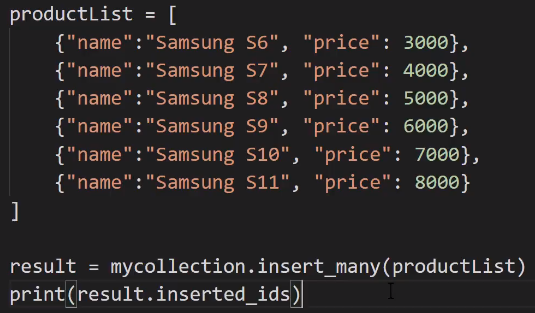
“mydb” komutu ile bir collection’a giriş yapılır, bu isimde collection bulunamazsa oluşturulur.



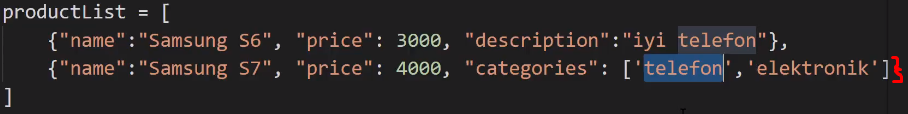
Collection isimlerini listelemek için gerekli komut.



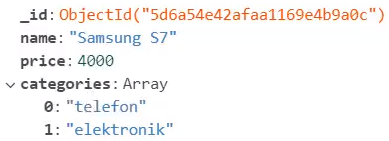
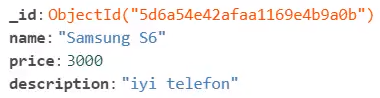
Dictionary verisi şeklinde belirlediğimiz “product” parametresini “.insert\_one()” komutu ile koleksiyona entegre ettik (çoklu değerler için “insert\_many”). Bu sayede yazılan veriyi MongoDb içerisine kaydetmiş olduk. Print komutlarıyla result parametresini, type’ını ve veri için atanan id değerini görüntüledik.

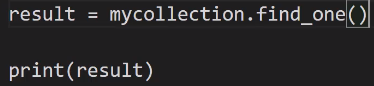


Dictionary içerisine “{“\_id”: 1}” şeklinde değer atayarak verilerin id bilgisini belirleyebiliriz fakat bu işlemi sistemin kendisine bırakmak daha mantıklıdır.



Şeklinde oluşturacağımız bir datalist sistemde şu şekilde gözükür:





Şeklinde yazılacak komut ile “mycollection” tablosundaki ilk veriyi görüntüleyebiliriz.



Şeklinde tüm kayıtları getirebiliriz, bu kayıtlar for döngüsü içerisinde alt alta yazdırılır.

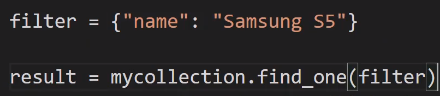


Şeklinde yazılan kodda kategorilere “0” değeri atandığı takdirde bu kategoriler gözükmez, “1” atanırsa gözükürler. Bu şekilde belirtmediğimiz kategoriler de gözükmeyecektir.

Aşağıdaki gibi bir yol izlediğimiz takdirde (yalnızca 0’lardan oluşan bir düzen) belirlediğimiz kategoriler gözükmeyecek, belirlemediklerimiz gözükecektir. “\_id” kategorisini 0 olarak belirlemediğimiz takdirde her durumda print edilir.



Yukarıdaki örnekler “find()” komutunun kategori kısmına değiniyorken (ikinci süslü parantez) şimdi yapılacaklar verilere değinecek (ilk süslü parantez).



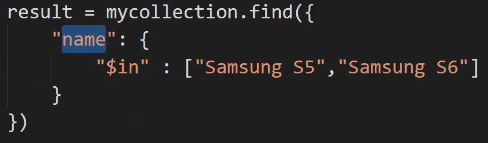
Şeklinde belirli markanın telefonundaki veriye ulaşabiliriz, bu veri bir adet olmalıdır. Aksi takdirde “find()” komutunu kullanarak bir for döngüsü oluşturmamız gerekir.

Herhangi bir verinin id değerini kullanarak veri çekmek istersek ayrıca bir kütüphaneyi import etmemiz gerekir.

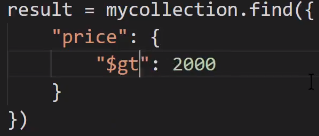




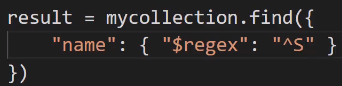
Yukarıdaki kütüphaneyi ekledikten sonra id değerimizi “ObjectId()” ile çevreleyerek belirli veriye ulaşabiliriz.



Şeklinde bir query ifade ettiğimiz zaman “$in” komutuyla belirli kategoride istenilen değer(ler)i getirme komutu yukarıdaki gibidir. Örnekte “name” kategorisinde Samsung S5/S6 içeren verileri karşımıza getir ifadesi yer almaktadır.



“$gt” (greater than) komutuyla örnekte 2000’in üzerindeki değerleri alacak bir kod yazdık. “$gt” komutunu “$gte” (greater than or equal) olarak kullandığımız takdirde 2000 değeri de scope’a dahil edilecektir. “$eq” ise eşitliğe karşılık gelmektedir (equal to). “$lt” ile de küçük sayıları alabiliriz.



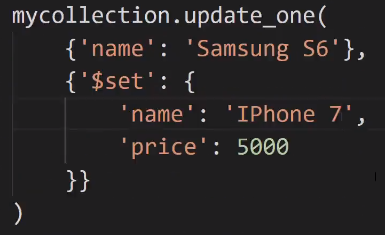
Bu komutla ismi “S” ile başlayan tüm kayıtları getirebiliriz. Anladığım kadarıyla “S” in sağında üs olması ve üs işaretlerinin arasına

Daha fazla operatör için 🡪 [**Kaynak**](https://docs.mongodb.com/manual/reference/operator/query/)

“sort(“category)” komutu ile belirlediğimiz kategoriden alfabetik veya sayısal bir sıralama işlemi yapabiliriz. Kategoriyi “name” olarak belirlediğimiz takdirde veriler ürün isimlerine göre alfabetik olarak sıralanır. “sort(“category, 1)” şeklinde sayı belirlersek kategorilendirme ascending yani artan bir şekilde gerçekleşirken “-1” değeri girersek descending yani azalan şekilde vuku bulur. “1” değeri default olarak ayarlıdır.

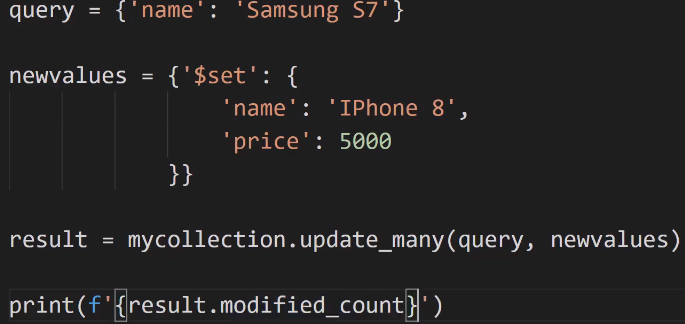


“sort()” fonksiyonunu örnekteki gibi uygulamamız halinde sistem sıralamayı ilk olarak isme göre yaparken, aynı isme sahip verilerin de fiyat değerlerine göre azalan bir sıralama sunacaktır.



Yukarıdaki şemada ilk parametrede “name”’i ifade ederek veriyi belirledik ve “$set” fonksiyonuyla güncellenecek değişkenleri yazdık. Veri tablomuzda birden fazla “Samsung S6” olması halinde “.update\_one()” komudunu kullandığımız için yalnızca ilk değer güncellenecektir. Tüm değerlerin güncellenmesi için “update\_many()” komutunu kullanmamız gerekir.

Aynı işlemi yapmanın bir diğer yolu ise örnekte gösterilmiştir:



“.modified\_count” metodu ile kaç verinin güncellendiği bilgisini alabiliriz.



İlk satırda ismi “IPhone 8” olan ilk verinin silinmesi, ikinci satırda ismi “S” ile başlayan bütün verilerin silinmesi kodlanmıştır.

“.deleted\_count” komutuyla silinen veri miktarını elde edebiliriz.