

US-TCS-604 – Big Data Analytics  
Practical 3 – Operators in MongoDB

**Create a MongoDB schema for blog website which will have the collection as post, comments and tags list. This website should follow these requirements:**

- Every post must have a unique id, title, description and url.
- Every post can have one or more tags.
- Every post can have one or more comments given by users, along with their names, message, date and likes.
- Every post has name of the publisher and total number of likes.

**Creating database blog.**

> use blog

switched to db blog

**Adding data to blog.**

```
> db.blog.insertMany([{"id": 1, "title": "How to Get Away with Murder", "desc": "you can't", "postby": "Krishi", "url": "www.blogone.in/htgawm", "likes": 100, "comments": [{"user": "ok", "message": "nice", "date": "12-12-2022", "likes": 10}], "tags": [{"user": "sabahat", "message": "knives out", "likes": 4}]},  
... {"id": 2, "title": "How to Bake a Brownie", "desc": "you shouldn't", "postby": "Vaibhav", "url": "www.blogone.in/htbab", "likes": 101, "comments": [{"user": "disha", "message": "very helpful", "date": "15-12-2022", "likes": 1}], "tags": [{"user": "sagar", "message": "eat", "likes": 3}]},  
... {"id": 3, "title": "Necessities of Suffering in Life", "desc": "1) so you can appreciate good things in life", "postby": "Imaad", "url": "www.blogone.in/nosil", "likes": 200, "comments": [{"user": "krishi", "message": "noice", "date": "05-12-2022", "likes": 30}], "tags": [{"user": "vaibhav", "message": "sad", "likes": 2}]},  
... {"id": 4, "title": "The Perfect Outfit", "desc": "goth", "postby": "Sabahat", "url": "www.blogone.in/tpo", "likes": 250, "comments": [{"user": "krishi", "message": "great tip", "date": "14-12-2022", "likes": 20}], "tags": [{"user": "krasia", "message": "agreed", "likes": 7}]},  
... {"id": 5, "title": "Dancing 101", "desc": "just do it", "postby": "Disha", "url": "www.blogone.in/d101", "likes": 150, "comments": [{"user": "ansh", "message": "true", "date": "20-11-2022", "likes": 30}], "tags": [{"user": "sabahat", "message": "great", "likes": 4}]})  
... )  
{  
    "acknowledged" : true,  
    "insertedIds" : [  
        ObjectId("639817bb8a1473e91a6fe8dc"),  
        ObjectId("639817bb8a1473e91a6fe8dd"),  
        ObjectId("639817bb8a1473e91a6fe8de"),  
        ObjectId("639817bb8a1473e91a6fe8df"),  
        ObjectId("639817bb8a1473e91a6fe8e0")  
    ]  
}
```

**Displaying the content of blog.**

> db.blog.find()

```
{
  "_id": ObjectId("639817bb8a1473e91a6fe8dc"), "id": 1, "title": "How to Get Away with Murder", "desc": "you can't", "postby": "Krishi", "url": "www.blogone.in/htgawm", "likes": 100, "comments": [ { "user": "ok", "message": "nice", "date": "12-12-2022", "likes": 10 } ], "tags": [ { "user": "sabahat", "message": "knives out", "likes": 4 } ] }
{ "_id": ObjectId("639817bb8a1473e91a6fe8dd"), "id": 2, "title": "How to Bake a Brownie", "desc": "you shouldn't", "postby": "Vaibhav", "url": "www.blogone.in/htbab", "likes": 101, "comments": [ { "user": "disha", "message": "very helpful", "date": "15-12-2022", "likes": 1 } ], "tags": [ { "user": "sagar", "message": "eat", "likes": 3 } ] }
{ "_id": ObjectId("639817bb8a1473e91a6fe8de"), "id": 3, "title": "Necessities of Suffering in Life", "desc": "1) so you can appreciate good things in life", "postby": "Imaad", "url": "www.blogone.in/nosil", "likes": 200, "comments": [ { "user": "krishi", "message": "noice", "date": "05-12-2022", "likes": 30 } ], "tags": [ { "user": "vaibhav", "message": "sad", "likes": 2 } ] }
{ "_id": ObjectId("639817bb8a1473e91a6fe8df"), "id": 4, "title": "The Perfect Outfit", "desc": "goth", "postby": "Sabahat", "url": "www.blogone.in/tpo", "likes": 250, "comments": [ { "user": "krishi", "message": "great tip", "date": "14-12-2022", "likes": 20 } ], "tags": [ { "user": "krasia", "message": "agreed", "likes": 7 } ] }
{ "_id": ObjectId("639817bb8a1473e91a6fe8e0"), "id": 5, "title": "Dancing 101", "desc": "just do it", "postby": "Disha", "url": "www.blogone.in/d101", "likes": 150, "comments": [ { "user": "ansh", "message": "true", "date": "20-11-2022", "likes": 30 } ], "tags": [ { "user": "sabahat", "message": "great", "likes": 4 } ] }
```

### **Displaying formatted content of blog.**

```
> db.blog.find().pretty()
{
  "_id": ObjectId("639817bb8a1473e91a6fe8dc"),
  "id": 1,
  "title": "How to Get Away with Murder",
  "desc": "you can't",
  "postby": "Krishi",
  "url": "www.blogone.in/htgawm",
  "likes": 100,
  "comments": [
    {
      "user": "ok",
      "message": "nice",
      "date": "12-12-2022",
      "likes": 10
    }
  ],
  "tags": [
    {
      "user": "sabahat",
      "message": "knives out",
      "likes": 4
    }
  ]
}
{
  "_id": ObjectId("639817bb8a1473e91a6fe8dd"),
```

```

"id" : 2,
"title" : "How to Bake a Brownie",
"desc" : "you shouldn't",
"postby" : "Vaibhav",
"url" : "www.blogone.in/htbab",
"likes" : 101,
"comments" : [
    {
        "user" : "disha",
        "message" : "very helpful",
        "date" : "15-12-2022",
        "likes" : 1
    }
],
"tags" : [
    {
        "user" : "sagar",
        "message" : "eat",
        "likes" : 3
    }
]
}
{
"_id" : ObjectId("639817bb8a1473e91a6fe8de"),
"id" : 3,
"title" : "Necessities of Suffering in Life",
"desc" : "1) so you can appreciate good things in life",
"postby" : "Imaad",
"url" : "www.blogone.in/nosil",
"likes" : 200,
"comments" : [
    {
        "user" : "krishi",
        "message" : "noice",
        "date" : "05-12-2022",
        "likes" : 30
    }
],
"tags" : [
    {
        "user" : "vaibhav",
        "message" : "sad",
        "likes" : 2
    }
]
}
{
"_id" : ObjectId("639817bb8a1473e91a6fe8df"),
"id" : 4,
"title" : "The Perfect Outfit",

```

```

"desc" : "goth",
"postby" : "Sabahat",
"url" : "www.blogone.in/tpo",
"likes" : 250,
"comments" : [
    {
        "user" : "krishi",
        "message" : "great tip",
        "date" : "14-12-2022",
        "likes" : 20
    }
],
"tags" : [
    {
        "user" : "krasia",
        "message" : "agreed",
        "likes" : 7
    }
]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8e0"),
    "id" : 5,
    "title" : "Dancing 101",
    "desc" : "just do it",
    "postby" : "Disha",
    "url" : "www.blogone.in/d101",
    "likes" : 150,
    "comments" : [
        {
            "user" : "ansh",
            "message" : "true",
            "date" : "20-11-2022",
            "likes" : 30
        }
    ],
    "tags" : [
        {
            "user" : "sabahat",
            "message" : "great",
            "likes" : 4
        }
    ]
}

```

### **Displaying posts with likes greater than or equal to 200.**

```
> db.blog.find({likes: {$gte:200}}).pretty()
```

```
{
```

```

        "_id" : ObjectId("639817bb8a1473e91a6fe8de"),
        "id" : 3,
        "title" : "Necessities of Suffering in Life",
        "desc" : "1) so you can appreciate good things in life",
        "postby" : "Imaad",
        "url" : "www.blogone.in/nosil",
        "likes" : 200,
        "comments" : [
            {
                "user" : "krishi",
                "message" : "noice",
                "date" : "05-12-2022",
                "likes" : 30
            }
        ],
        "tags" : [
            {
                "user" : "vaibhav",
                "message" : "sad",
                "likes" : 2
            }
        ]
    }
}

{
        "_id" : ObjectId("639817bb8a1473e91a6fe8df"),
        "id" : 4,
        "title" : "The Perfect Outfit",
        "desc" : "goth",
        "postby" : "Sabahat",
        "url" : "www.blogone.in/tpo",
        "likes" : 250,
        "comments" : [
            {
                "user" : "krishi",
                "message" : "great tip",
                "date" : "14-12-2022",
                "likes" : 20
            }
        ],
        "tags" : [
            {
                "user" : "krasia",
                "message" : "agreed",
                "likes" : 7
            }
        ]
}

```

## OR OPERATOR

**Displaying posts either with likes = 150 or made by Sabahat.**

```
> db.blog.find({$or:[{likes:150}, {postby:"Sabahat"}]}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8df"),
    "id" : 4,
    "title" : "The Perfect Outfit",
    "desc" : "goth",
    "postby" : "Sabahat",
    "url" : "www.blogone.in/tpo",
    "likes" : 250,
    "comments" : [
        {
            "user" : "krishi",
            "message" : "great tip",
            "date" : "14-12-2022",
            "likes" : 20
        }
    ],
    "tags" : [
        {
            "user" : "krasia",
            "message" : "agreed",
            "likes" : 7
        }
    ]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8e0"),
    "id" : 5,
    "title" : "Dancing 101",
    "desc" : "just do it",
    "postby" : "Disha",
    "url" : "www.blogone.in/d101",
    "likes" : 150,
    "comments" : [
        {
            "user" : "ansh",
            "message" : "true",
            "date" : "20-11-2022",
            "likes" : 30
        }
    ],
    "tags" : [
        {
            "user" : "sabahat",
            "message" : "great",
            "likes" : 4
        }
    ]
}
```

## AND OPERATOR

**Displaying post made by Disha and having likes less than 200.**

```
> db.blog.find({$and:[{likes:{$lt:200}}, {postby:"Disha"}]}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8e0"),
    "id" : 5,
    "title" : "Dancing 101",
    "desc" : "just do it",
    "postby" : "Disha",
    "url" : "www.blogone.in/d101",
    "likes" : 150,
    "comments" : [
        {
            "user" : "ansh",
            "message" : "true",
            "date" : "20-11-2022",
            "likes" : 30
        }
    ],
    "tags" : [
        {
            "user" : "sabahat",
            "message" : "great",
            "likes" : 4
        }
    ]
}
```

## NOT OPERATOR

**Displaying likes NOT less than 150.**

```
> db.blog.find({likes:{'$not':{$lt:150}}}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8de"),
    "id" : 3,
    "title" : "Necessities of Suffering in Life",
    "desc" : "1) so you can appreciate good things in life",
    "postby" : "Imaad",
    "url" : "www.blogone.in/nosil",
    "likes" : 200,
    "comments" : [
        {
            "user" : "krishi",
            "message" : "noice",
            "date" : "05-12-2022",
            "likes" : 30
        }
    ],
}
```

```

"tags" : [
    {
        "user" : "vaibhav",
        "message" : "sad",
        "likes" : 2
    }
]
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8df"),
    "id" : 4,
    "title" : "The Perfect Outfit",
    "desc" : "goth",
    "postby" : "Sabahat",
    "url" : "www.blogone.in/tpo",
    "likes" : 250,
    "comments" : [
        {
            "user" : "krishi",
            "message" : "great tip",
            "date" : "14-12-2022",
            "likes" : 20
        }
    ],
    "tags" : [
        {
            "user" : "krasia",
            "message" : "agreed",
            "likes" : 7
        }
    ]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8e0"),
    "id" : 5,
    "title" : "Dancing 101",
    "desc" : "just do it",
    "postby" : "Disha",
    "url" : "www.blogone.in/d101",
    "likes" : 150,
    "comments" : [
        {
            "user" : "ansh",
            "message" : "true",
            "date" : "20-11-2022",
            "likes" : 30
        }
    ],
    "tags" : [
        {

```

```

        "user" : "sabahat",
        "message" : "great",
        "likes" : 4
    }
]
}

```

## NOR OPERATOR

**Displaying all data except where likes is less than 100 or greater than 175.**

```
> db.blog.find({$nor:[{likes:{$lt:100}}, {likes:{$gt:175}}]}).pretty()
```

```
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dc"),
    "id" : 1,
    "title" : "How to Get Away with Murder",
    "desc" : "you can't",
    "postby" : "Krishi",
    "url" : "www.blogone.in/htgawm",
    "likes" : 100,
    "comments" : [
        {
            "user" : "ok",
            "message" : "nice",
            "date" : "12-12-2022",
            "likes" : 10
        }
    ],
    "tags" : [
        {
            "user" : "sabahat",
            "message" : "knives out",
            "likes" : 4
        }
    ]
}

{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dd"),
    "id" : 2,
    "title" : "How to Bake a Brownie",
    "desc" : "you shouldn't",
    "postby" : "Vaibhav",
    "url" : "www.blogone.in/htbab",
    "likes" : 101,
    "comments" : [
        {
            "user" : "disha",
            "message" : "very helpful",
            "date" : "15-12-2022",
            "likes" : 1
        }
    ]
}
```

```

        ],
        "tags" : [
            {
                "user" : "sagar",
                "message" : "eat",
                "likes" : 3
            }
        ]
    }
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8e0"),
    "id" : 5,
    "title" : "Dancing 101",
    "desc" : "just do it",
    "postby" : "Disha",
    "url" : "www.blogone.in/d101",
    "likes" : 150,
    "comments" : [
        {
            "user" : "ansh",
            "message" : "true",
            "date" : "20-11-2022",
            "likes" : 30
        }
    ],
    "tags" : [
        {
            "user" : "sabahat",
            "message" : "great",
            "likes" : 4
        }
    ]
}

```

## IN OPERATOR

**Display posts made by people listed in an array.**

```

> db.blog.find({postby:{ $in:["Krishi", "Disha"] }}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dc"),
    "id" : 1,
    "title" : "How to Get Away with Murder",
    "desc" : "you can't",
    "postby" : "Krishi",
    "url" : "www.blogone.in/htgawm",
    "likes" : 100,
    "comments" : [
        {
            "user" : "ok",
            "message" : "nice",

```

```

        "date" : "12-12-2022",
        "likes" : 10
    }
],
"tags" : [
{
    "user" : "sabahat",
    "message" : "knives out",
    "likes" : 4
}
]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8e0"),
    "id" : 5,
    "title" : "Dancing 101",
    "desc" : "just do it",
    "postby" : "Disha",
    "url" : "www.blogone.in/d101",
    "likes" : 150,
    "comments" : [
{
    "user" : "ansh",
    "message" : "true",
    "date" : "20-11-2022",
    "likes" : 30
}
],
"tags" : [
{
    "user" : "sabahat",
    "message" : "great",
    "likes" : 4
}
]
}
}

```

**Display all posts except those made by Krish and Disha.**

```

> db.blog.find({postby:{$not:{$in:["Krishi", "Disha"]}}}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dd"),
    "id" : 2,
    "title" : "How to Bake a Brownie",
    "desc" : "you shouldn't",
    "postby" : "Vaibhav",
    "url" : "www.blogone.in/htbab",
    "likes" : 101,
    "comments" : [
{
    "user" : "disha",

```

```

        "message" : "very helpful",
        "date" : "15-12-2022",
        "likes" : 1
    }
],
"tags" : [
    {
        "user" : "sagar",
        "message" : "eat",
        "likes" : 3
    }
]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8de"),
    "id" : 3,
    "title" : "Necessities of Suffering in Life",
    "desc" : "1) so you can appreciate good things in life",
    "postby" : "Imaad",
    "url" : "www.blogone.in/nosil",
    "likes" : 200,
    "comments" : [
        {
            "user" : "krishi",
            "message" : "noice",
            "date" : "05-12-2022",
            "likes" : 30
        }
    ],
    "tags" : [
        {
            "user" : "vaibhav",
            "message" : "sad",
            "likes" : 2
        }
    ]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8df"),
    "id" : 4,
    "title" : "The Perfect Outfit",
    "desc" : "goth",
    "postby" : "Sabahat",
    "url" : "www.blogone.in/tpo",
    "likes" : 250,
    "comments" : [
        {
            "user" : "krishi",
            "message" : "great tip",
            "date" : "14-12-2022",
        }
    ]
}

```

```

        "likes" : 20
    }
],
"tags" : [
{
    "user" : "krasia",
    "message" : "agreed",
    "likes" : 7
}
]
}
```

## NIN OPERATOR

### Short form for not in.

```
> db.blog.find({postby:{$nin:["Krishi", "Disha"]}}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dd"),
    "id" : 2,
    "title" : "How to Bake a Brownie",
    "desc" : "you shouldn't",
    "postby" : "Vaibhav",
    "url" : "www.blogone.in/htbab",
    "likes" : 101,
    "comments" : [
        {
            "user" : "disha",
            "message" : "very helpful",
            "date" : "15-12-2022",
            "likes" : 1
        }
    ],
    "tags" : [
        {
            "user" : "sagar",
            "message" : "eat",
            "likes" : 3
        }
    ]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8de"),
    "id" : 3,
    "title" : "Necessities of Suffering in Life",
    "desc" : "1) so you can appreciate good things in life",
    "postby" : "Imaad",
    "url" : "www.blogone.in/nosil",
    "likes" : 200,
    "comments" : [
        {

```

```

        "user" : "krishi",
        "message" : "noice",
        "date" : "05-12-2022",
        "likes" : 30
    }
],
"tags" : [
{
    "user" : "vaibhav",
    "message" : "sad",
    "likes" : 2
}
]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8df"),
    "id" : 4,
    "title" : "The Perfect Outfit",
    "desc" : "goth",
    "postby" : "Sabahat",
    "url" : "www.blogone.in/tpo",
    "likes" : 250,
    "comments" : [
    {
        "user" : "krishi",
        "message" : "great tip",
        "date" : "14-12-2022",
        "likes" : 20
    }
],
"tags" : [
{
    "user" : "krasia",
    "message" : "agreed",
    "likes" : 7
}
]
}

```

## Regular Expressions

### Starts With:

```

> db.blog.find({title:{$regex:/^H/}}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dc"),
    "id" : 1,
    "title" : "How to Get Away with Murder",
    "desc" : "you can't",
    "postby" : "Krishi",

```

```

"url" : "www.blogone.in/htgawm",
"likes" : 100,
"comments" : [
    {
        "user" : "ok",
        "message" : "nice",
        "date" : "12-12-2022",
        "likes" : 10
    }
],
"tags" : [
    {
        "user" : "sabahat",
        "message" : "knives out",
        "likes" : 4
    }
]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dd"),
    "id" : 2,
    "title" : "How to Bake a Brownie",
    "desc" : "you shouldn't",
    "postby" : "Vaibhav",
    "url" : "www.blogone.in/htbab",
    "likes" : 101,
    "comments" : [
        {
            "user" : "disha",
            "message" : "very helpful",
            "date" : "15-12-2022",
            "likes" : 1
        }
    ],
    "tags" : [
        {
            "user" : "sagar",
            "message" : "eat",
            "likes" : 3
        }
    ]
}

```

## ALL OPERATOR

```

> db.blog.find({postby:{$all:["Krishi", "Imaad"]}}).pretty()
> db.blog.find({postby:{$all:["Krishi"]}}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dc"),
    "id" : 1,

```

```

"title" : "How to Get Away with Murder",
"desc" : "you can't",
"postby" : "Krishi",
"url" : "www.blogone.in/htgawm",
"likes" : 100,
"comments" : [
    {
        "user" : "ok",
        "message" : "nice",
        "date" : "12-12-2022",
        "likes" : 10
    }
],
"tags" : [
    {
        "user" : "sabahat",
        "message" : "knives out",
        "likes" : 4
    }
]
}

> db.blog.insertOne({postby:["Krishi", "Imaad", "Hanif"]})
{
    "acknowledged" : true,
    "insertedId" : ObjectId("6398296d8a1473e91a6fe8e1")
}
> db.blog.find({postby:{$all:["Krishi", "Imaad", "Hanif"]}}).pretty()
{
    "_id" : ObjectId("6398296d8a1473e91a6fe8e1"),
    "postby" : [
        "Krishi",
        "Imaad",
        "Hanif"
    ]
}
> db.blog.find({postby:{$all:["Krishi", "Imaad"]}}).pretty()
{
    "_id" : ObjectId("6398296d8a1473e91a6fe8e1"),
    "postby" : [
        "Krishi",
        "Imaad",
        "Hanif"
    ]
}
> db.blog.find({postby:{$all:["Krishi"]}}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dc"),
    "id" : 1,

```

```

"title" : "How to Get Away with Murder",
"desc" : "you can't",
"postby" : "Krishi",
"url" : "www.blogone.in/htgawm",
"likes" : 100,
"comments" : [
    {
        "user" : "ok",
        "message" : "nice",
        "date" : "12-12-2022",
        "likes" : 10
    }
],
"tags" : [
    {
        "user" : "sabahat",
        "message" : "knives out",
        "likes" : 4
    }
]
}
{
    "_id" : ObjectId("6398296d8a1473e91a6fe8e1"),
    "postby" : [
        "Krishi",
        "Imaad",
        "Hanif"
    ]
}

> db.blog.find({postby:{$all:["Krishi", "Imaad"]}}).pretty()
{
    "_id" : ObjectId("6398296d8a1473e91a6fe8e1"),
    "postby" : [
        "Krishi",
        "Imaad",
        "Hanif"
    ]
}

> db.blog.find({postby:["Krishi", "Imaad"]}).pretty()
> db.blog.find({postby:{'$in':["Krishi", "Imaad"]}}).pretty()
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8dc"),
    "id" : 1,
    "title" : "How to Get Away with Murder",
    "desc" : "you can't",
    "postby" : "Krishi",
    "url" : "www.blogone.in/htgawm",
    "likes" : 100,
}

```

```

"comments" : [
    {
        "user" : "ok",
        "message" : "nice",
        "date" : "12-12-2022",
        "likes" : 10
    }
],
"tags" : [
    {
        "user" : "sabahat",
        "message" : "knives out",
        "likes" : 4
    }
]
}
{
    "_id" : ObjectId("639817bb8a1473e91a6fe8de"),
    "id" : 3,
    "title" : "Necessities of Suffering in Life",
    "desc" : "1) so you can appreciate good things in life",
    "postby" : "Imaad",
    "url" : "www.blogone.in/nosil",
    "likes" : 200,
    "comments" : [
        {
            "user" : "krishi",
            "message" : "noice",
            "date" : "05-12-2022",
            "likes" : 30
        }
    ],
    "tags" : [
        {
            "user" : "vaibhav",
            "message" : "sad",
            "likes" : 2
        }
    ]
}
{
    "_id" : ObjectId("6398296d8a1473e91a6fe8e1"),
    "postby" : [
        "Krishi",
        "Imaad",
        "Hanif"
    ]
}

```

# PRACTICAL 9

## Cassandra

### STEPS:

Perquisites:

1. Python 2.7
2. Java jdk 1.8

```
PS C:\Users\Mallika Sharma> java -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
PS C:\Users\Mallika Sharma> python --version
Python 2.7.18
PS C:\Users\Mallika Sharma>
```

Add directory path in user variables.

1. C:\Program Files\Java\jdk1.8.0\_121\bin

Download the latest Apache Cassandra 3.11 release:

Latest release on 2022-10-23

Maintained until 4.2.0 release (May-July 2023)

3.11.14

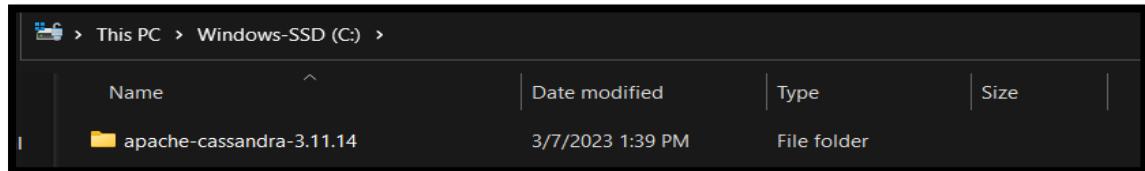
([pgp](#), [sha256](#) and [sha512](#))

2. C:\Python27

[https://cassandra.apache.org/\\_/download.html](https://cassandra.apache.org/_/download.html)

 apache-cassandra-3.11.14-bin.tar	2/3/2023 4:40 PM	WinRAR archive	31,343 KB
 python-2.7.amd64	2/3/2023 4:56 PM	Windows Installer ...	15,867 KB

Compress the tar file. Unzip it into the C drive



C:\apache-cassandra-3.11.14\bin

Add the above path to user variables as well.

### Type: Cassandra -f in command prompt

A screenshot of a Windows PowerShell window titled 'Select Administrator: Windows PowerShell'. The command 'cassandra -f' is typed into the prompt. The output shows a warning message: 'WARNING! Automatic page file configuration detected. It is recommended that you disable swap when running Cassandra for performance and stability reasons.' The command prompt ends with three asterisks (\*-----\*).

A screenshot of a terminal window showing the Cassandra server starting up. The log output includes: 'INFO [main] 2023-02-03 17:59:19,097 Server.java:159 - Starting listening for CQL clients on localhost/127.0.0.1:9042 (unencrypted).. INFO [main] 2023-02-03 17:59:19,302 CassandraDaemon.java:564 - Not starting RPC server as requested. Use JMX (StorageService-&gt;startJMX) if needed. INFO [main] 2023-02-03 17:59:19,302 CassandraDaemon.java:650 - Startup complete'. The command prompt ends with a single asterisk (\*).

Now keep the current prompt on, while opening a new command prompt.

### Type: cqlsh

Microsoft Windows [Version 10.0.22621.1265]

(c) Microsoft Corporation. All rights reserved.

C:\Users\disha>cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.

If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.

[cqlsh 5.0.1 | Cassandra 3.11.14 | CQL spec 3.4.4 | Native protocol v4]

Use HELP for help.

WARNING: pyreadline dependency missing. Install to enable tab completion.

cqlsh>

## **1. Keyspace Operations:**

### **Create Keyspace:**

```
cqlsh> CREATE KEYSPACE tutorialspoint WITH replication = {'class':'SimpleStrategy',  
'replication_factor' : 1};  
  
cqlsh> CREATE KEYSPACE dishaWITH replication = {'class':'SimpleStrategy',  
'replication_factor' : 1};
```

### **Verification:**

```
cqlsh> DESCRIBE KEYSPACES;  
  
tutorialspoint system_auth system_distributed disha  
system_schema system system_traces
```

---

### **Durable\_writes:**

By default, the durable\_writes properties of a table is set to true, however it can be set to false.

```
cqlsh> CREATE KEYSPACE deep WITH replication = {'class':'SimpleStrategy',  
'replication_factor' : 1} AND DURABLE_WRITES = false;
```

### **Verification:**

```
cqlsh> select * from system_schema.keyspaces;  
  
keyspace_name | durable_writes | replication  
-----+-----+  
system_auth | True | {'class': 'org.apache.cassandra.locator.SimpleStrategy',  
'replication_factor': '1'}  
  
system_schema | True | {'class':  
'org.apache.cassandra.locator.LocalStrategy'}  
  
deep | False | {'class': 'org.apache.cassandra.locator.SimpleStrategy',  
'replication_factor': '1'}  
  
disha| True | {'class': 'org.apache.cassandra.locator.SimpleStrategy',  
'replication_factor': '1'}
```

```

tutorialspoint |      True | {'class': 'org.apache.cassandra.locator.SimpleStrategy',
'replication_factor': '1'}

system_distributed |      True | {'class': 'org.apache.cassandra.locator.SimpleStrategy',
'replication_factor': '3'}

system |      True | {'class':
'org.apache.cassandra.locator.LocalStrategy'}

system_traces |      True | {'class': 'org.apache.cassandra.locator.SimpleStrategy',
'replication_factor': '2'}

```

(8 rows)

---

### **Using a Keyspace:**

```
cqlsh> USE disha;
cqlsh:disha>
```

### **Altering a KeySpace:**

```
cqlsh:disha> ALTER KEYSPACE disha WITH replication =
{'class':'NetworkTopologyStrategy'};

cqlsh:disha> ALTER KEYSPACE deep WITH replication =
{'class':'NetworkTopologyStrategy'} AND DURABLE_WRITES = true;
```

### **Verification:**

```
cqlsh> SELECT * FROM system_schema.keyspaces;
```

keyspace_name	durable_writes	replication
system_auth	True	{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}

system_auth	True	{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
-------------	------	---

```

system_schema |      True |          {'class':
'org.apache.cassandra.locator.LocalStrategy'}

      deep |      True |          {'class':
'org.apache.cassandra.locator.NetworkTopologyStrategy'}

      disha|      True |          {'class':
'org.apache.cassandra.locator.NetworkTopologyStrategy'}

tutorialspoint |      True | {'class': 'org.apache.cassandra.locator.SimpleStrategy',
'replication_factor': '1'}

system_distributed |      True | {'class': 'org.apache.cassandra.locator.SimpleStrategy',
'replication_factor': '3'}

      system |      True |          {'class':
'org.apache.cassandra.locator.LocalStrategy'}

      system_traces |      True | {'class': 'org.apache.cassandra.locator.SimpleStrategy',
'replication_factor': '2'}

```

(8 rows)

### **Dropping a Keyspace:**

cqlsh> DESCRIBE keyspaces;

```

tutorialspoint  system_auth  deep          system_traces
system_schema   system       system_distributed  disha

```

### **Verification:**

cqlsh> DROP keyspace deep;

cqlsh> DESCRIBE keyspaces;

```

tutorialspoint  system_auth  system_distributed  disha
system_schema   system       system_traces

```

## 2. Table Operations:

### Creating a Table:

```
cqlsh> DESCRIBE keyspaces
```

```
tutorialspoint system_auth system_distributed disha  
system_schema system system_traces
```

```
cqlsh> use disha;
```

```
cqlsh:disha> CREATE TABLE report(
```

```
... sub_id int PRIMARY KEY,  
... sub_name text,  
... marks varint,  
... grade text  
... );
```

```
cqlsh:disha>
```

### Verification:

```
cqlsh:disha> select * from report;
```

sub_id	grade	marks	sub_name
(0 rows)			

### Altering a Table:

```
cqlsh:disha> ALTER TABLE report
```

```
... ADD remark text;
```

### Verification:

```
cqlsh:disha> select * from report;
```

sub_id	grade	marks	remark	sub_name
(0 rows)				

### **Dropping a Column:**

```
cqlsh:disha> ALTER TABLE report
```

```
... DROP remark;
```

### **3. CRUD Operations:**

#### **Creating Data in Table:**

```
cqlsh:disha> INSERT INTO report (sub_id, grade, marks, sub_name) VALUES (36, 'O', 98, 'Big Data Analysis');
```

```
cqlsh:disha> INSERT INTO report (sub_id, grade, marks, sub_name) VALUES (55, 'O', 96, 'Cyber Security and Forensics');
```

```
cqlsh:disha> INSERT INTO report (sub_id, grade, marks, sub_name) VALUES (602, 'A', 88, 'Machine Learning');
```

```
cqlsh:disha> INSERT INTO report (sub_id, grade, marks, sub_name) VALUES (636, 'B', 82, 'Cloud Computing');
```

```
cqlsh:disha> INSERT INTO report (sub_id, grade, marks, sub_name) VALUES (101, 'A', 96, 'Wireless Sensor Networks');
```

#### **Verification:**

```
cqlsh:disha> select * from report;
```

sub_id	grade	marks	sub_name
55	O	96	Cyber Security and Forensics
602	A	88	Machine Learning
636	B	82	Cloud Computing
36	O	98	Big Data Analysis
101	A	89	Wireless Sensor Networks

(5 rows)

#### **Updating Data in a Table:**

```
cqlsh:disha> UPDATE report SET grade = 'C', marks=78 WHERE sub_id=636;
```

### **Verification:**

```
cqlsh:disha> select * from report;
```

sub_id	grade	marks	sub_name
55	O	96	Cyber Securit and Forensics
602	A	88	Machine Learning
636	C	78	Cloud Computing
36	O	98	Big Data Analysis
101	A	89	Wireless Sensor Networks

(5 rows)

### **Reading Required Columns:**

```
cqlsh:disha> SELECT sub_id, sub_name from report;
```

sub_id	sub_name
55	Cyber Securit and Forensics
602	Machine Learning
636	Cloud Computing
36	Big Data Analysis
101	Wireless Sensor Networks

(5 rows)

**Or**

```
cqlsh:disha> SELECT * from report WHERE grade = 'A' ALLOW FILTERING;
```

sub_id	grade	marks	sub_name
602	A	88	Machine Learning
101	A	89	Wireless Sensor Networks

(2 rows)

**Or**

```
SELECT sub_id, sub_name from report WHERE grade = 'A' ALLOW FILTERING;
```

sub_id	sub_name
602	Machine Learning
101	Wireless Sensor Networks

(2 rows)

### **Deleting Datafrom a Table:**

```
cqlsh:disha> DELETE marks from report WHERE sub_id=636;
```

**Verification:**

```
cqlsh:disha> select * from report;
```

sub_id	grade	marks	sub_name
55	O	96	Cyber Securit and Forensics
602	A	88	Machine Learning
636	C	null	Cloud Computing
36	O	98	Big Data Analysis
101	A	89	Wireless Sensor Networks

(5 rows)

**Deleting an Entire Row:**

```
cqlsh:disha> DELETE from report WHERE sub_id=636;
```

**Verification**

```
cqlsh:disha> select * from report;
```

sub_id	grade	marks	sub_name
55	O	96	Cyber Securit and Forensics
602	A	88	Machine Learning
36	O	98	Big Data Analysis
101	A	89	Wireless Sensor Networks

(4 rows)

\*\*\*\*\*

## PRACTICAL 10

### Hadoop Installation and Word Count in Java and Python

(Java in Local and HDFS MODE, Python in Local Mode)

**Note:** All files to be created wherever the command prompt opens, or open the command prompt where the files are stored.

**Note:** Start the PowerShell in administrator mode if any issues arise in yarn or namenode since elevated privileges are required in some scenarios.

Steps:

**1. Create a file which will be used for the word count:**

```
C:\Users\prana>type wordcount.txt
big data analytics involves techniques of analytics to analyze the big data to obtain insights from the same.
There are 4 types of big data analytics such as descriptive, diagnostic, prescriptive and predictive.
C:\Users\prana>
```

**2. Create the mapper.py file**

```
C:\Users\disha>type mapper.py
#!/usr/bin/env python

# import sys because we need to read and write data to STDIN and STDOUT
import sys

# reading entire line from STDIN (standard input)
for line in sys.stdin:

    # to remove leading and trailing whitespace
    line = line.strip()

    # split the line into words
    words = line.split()

    # we are looping over the words array and printing the word
    # with the count of 1 to the STDOUT
    for word in words:

        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        print('%s\t%s' % (word, 1))

C:\Users\disha>
```

**3. Execute the mapper.py and check the output**

```
C:\Users\disha>type wordcount.txt | python mapper.py  
big    1  
data   1  
analytics  1  
involves  1  
techniques 1  
of     1  
analytics 1  
to     1  
analyze 1  
the    1  
big    1  
data   1  
to     1  
obtain 1  
insights 1  
from   1  
the    1  
same.  1
```

There 1

```
are   1  
4    1  
types 1  
of   1  
big  1
```

```
data    1
analytics    1
such    1
as    1
descriptive, 1
diagnostic, 1
prescriptive 1
and    1
predictive. 1
```

#### 4. Create the reducer.py and check the output

```
#!/usr/bin/env python

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# read the entire line from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # splitting the data on the basis of tab we have provided in mapper.py
    word, count = line.split("\t", 1)

    # convert count (currently a string) to int
    try:
```

```

count = int(count)

except ValueError:

    # count was not a number, so silently
    # ignore/discard this line

    continue

# this IF-switch only works because Hadoop sorts map output
# by key (here: word) before it is passed to the reducer

if current_word == word:

    current_count += count

else:

    if current_word:

        # write result to STDOUT

        print '%s\t%s' % (current_word, current_count)

    current_count = count

    current_word = word

# do not forget to output the last word if needed!

if current_word == word:

    print '%s\t%s' % (current_word, current_count)

```

**5. Check execution of mapper and then reducer to confirm whether everything is working or not**

PS C:\Users\disha> type wordcount.txt | python mapper.py | sort | python reducer.py

```

4      1
analytics    3
analyze 1
and      1

```

are 1  
as 1  
big 3  
data 3  
descriptive, 1  
diagnostic, 1  
from 1  
insights 1  
involves 1  
obtain 1  
of 2  
predictive. 1  
prescriptive 1  
same. 1  
such 1  
techniques 1  
the 2  
There 1  
to 2  
types 1

## 6. Create a directory hdfs and then copy the wordcount in that folder

```
C:\Users\disha>hdfs dfs -mkdir /wordcount  
'C:\Program' is not recognized as an internal or external command,  
operable program or batch file.
```

```
C:\Users\disha>hdfs dfs -copyFromLocal wordcount.txt /wordcount
```

'C:\Program' is not recognized as an internal or external command,  
operable program or batch file.

**7. Check the contents of the hdfs**

**8.**

C:\Users\disha>hdfs dfs -ls /

'C:\Program' is not recognized as an internal or external command,  
operable program or batch file.

Found 4 items

drwxr-xr-x - disha supergroup	0 2023-03-10 10:04 /Pig_Data
drwx-wx-wx - disha supergroup	0 2023-03-09 16:09 /tmp
drwxr-xr-x - disha supergroup	0 2023-04-07 15:10 /wordcount
drwxr-xr-x - disha supergroup	0 2023-04-07 15:05 /wordcount.txt

C:\Users\disha>hdfs dfs -ls /wordcount

'C:\Program' is not recognized as an internal or external command,  
operable program or batch file.

Found 1 items

-rw-r--r-- 1 disha supergroup	212 2023-04-07 15:10 /wordcount/wordcount.txt
-------------------------------	---

**9. Download the hadoop streaming jar from: <https://jar-download.com/artifacts/org.apache.hadoop/hadoop-streaming/2.7.3/source-code>.  
Also place the jar in a folder where the mapper and reducer are placed**

**Execute the following command:**

PS C:\Users\disha> hadoop jar hadoop-streaming-2.7.3.jar -input  
/wordcount/wordcount.txt -output /wordcount/output -mapper /mapper.py -reducer  
/reducer.py

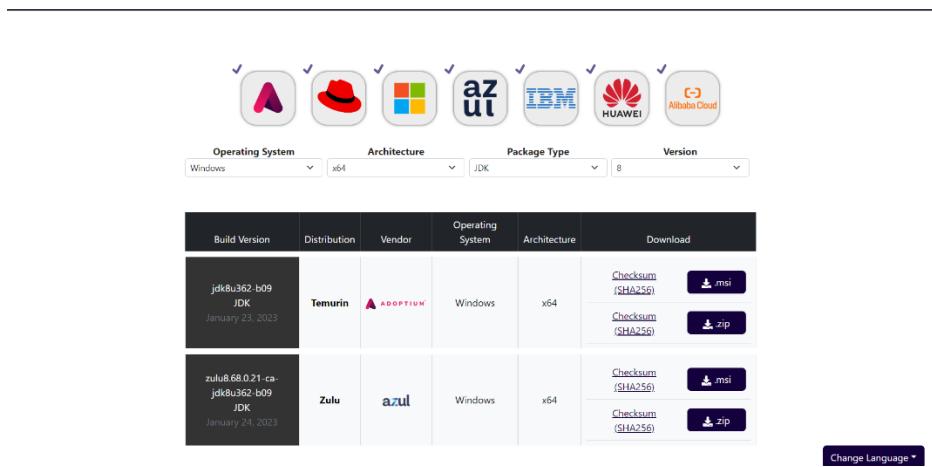
**10. Check the output by the following command**

hdfs dfs -cat /wordcount/output/part-00000

## **PRACTICAL 11**

# Hive Installation and Queries

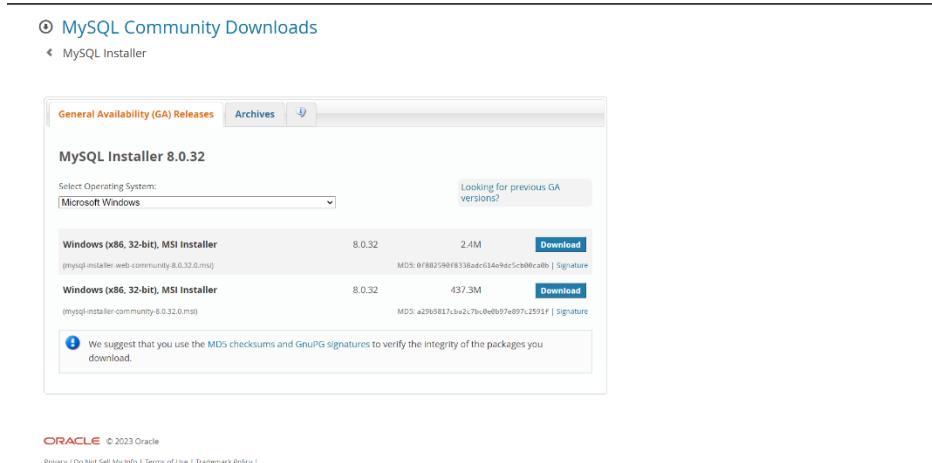
1. Download and install Java Development Kit (JDK) version 8.  
<https://adoptium.net/marketplace/?version=8>



The screenshot shows the Adoptium Marketplace interface for Java JDK version 8. At the top, there are filters for Operating System (Windows), Architecture (x64), Package Type (JDK), and Version (8). Below the filters, two JDK distributions are listed: Temurin and Zulu. For each distribution, there are two vendor options: ADOPTIUM and azul. Each vendor entry includes the build version, distribution name, and release date. To the right of each entry are download links for 'Checksum (SHA256)' and 'Checksum (SHA128)' in both MSI and ZIP formats. A 'Change Language' button is located at the bottom right of the grid.

Build Version	Distribution	Vendor	Operating System	Architecture	Download
jdk8u362-b09 JDK January 23, 2023	Temurin	ADOPTIUM	Windows	x64	<a href="#">Checksum (SHA256)</a> <a href="#">Checksum (SHA128)</a>
zulu8.68.0.21-ca-jdk8u362-b09 JDK January 24, 2023	Zulu	azul	Windows	x64	<a href="#">Checksum (SHA256)</a> <a href="#">Checksum (SHA128)</a>

2. Download and install MySQL database server and MySQL Workbench.  
<https://dev.mysql.com/downloads/installer/>



The screenshot shows the MySQL Community Downloads page for MySQL Installer 8.0.32. The page has tabs for 'General Availability (GA) Releases' (which is selected), 'Archives', and a search bar. It displays two download options for Windows (x86, 32-bit) MSI installers. Each option includes the file size, MD5 checksum, and a 'Download' button. A note at the bottom encourages users to verify the integrity of the packages using MD5 checksums and GnuPG signatures. The footer contains the Oracle logo and links to Privacy, Do Not Sell My Info, Terms of Use, and Trademark Policy.

MySQL Installer 8.0.32

Select Operating System: Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.32.0.msi)	8.0.32	2.4M	<a href="#">Download</a>
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.32.0.msi)	8.0.32	437.3M	<a href="#">Download</a>

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

ORACLE © 2023 Oracle  
Privacy | Do Not Sell My Info | Terms of Use | Trademark Policy |

3. The latest stable version of Apache Hive from the official website.  
<https://dlcdn.apache.org/hive/>

You can install version 3.x and later.

Install bin.tar.gz

---

#### Index of /hive

Name	Last modified	Size	Description
Parent Directory		-	
hive-1.2.2/	2022-06-17 12:34	-	
hive-2.3.9/	2022-06-17 12:34	-	
hive-3.1.3/	2022-06-17 12:34	-	
hive-3.1.3-bin/	2022-06-17 12:34	-	
hive-3.1.3-bin/	2022-06-17 12:34	-	
hive-3.1.3-bin/	2022-06-17 12:34	-	
hive-3.1.3-bin/	2022-06-17 12:34	-	
hive-3.1.3-bin/	2022-06-17 12:34	-	
hive-3.1.3-bin/	2022-06-17 12:34	-	
hive-standalone-metastore-3.0.0/	2022-06-17 12:34	-	
hive-storage-2.7.3/	2022-06-17 12:34	-	
hive-storage-2.8.1/	2022-06-17 12:34	-	
stablez-2/	2022-06-17 12:34	-	
KEYS	2022-08-24 17:35	102K	

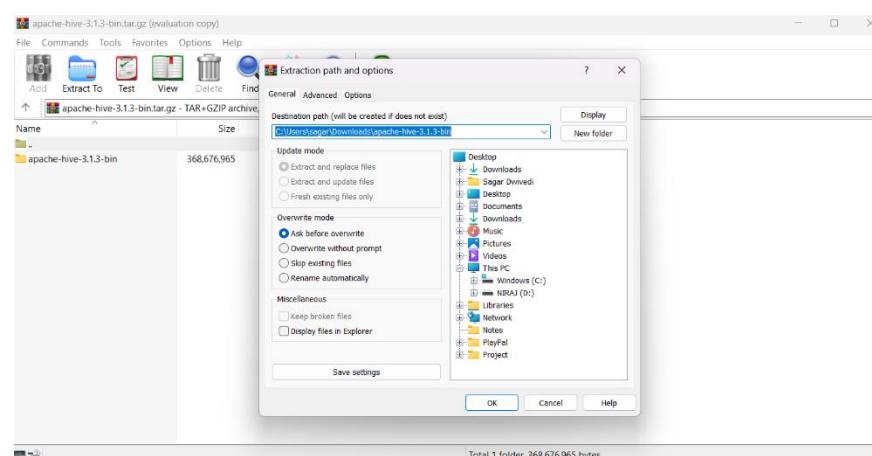
---

#### Index of /hive/hive-3.1.3

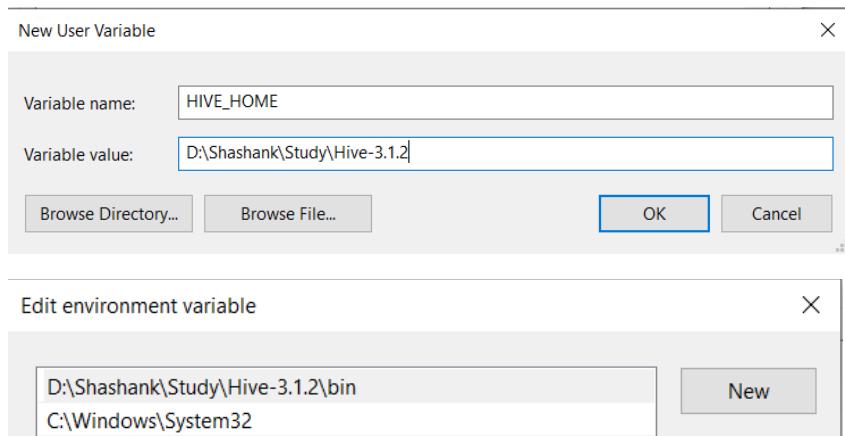
Name	Last modified	Size	Description
Parent Directory		-	
apache-hive-3.1.3-bin.tar.gz	2022-04-08 17:42	312M	
apache-hive-3.1.3-bin.tar.gz.asc	2022-04-08 17:42	488	
apache-hive-3.1.3-bin.tar.gz.sha256	2022-04-08 17:42	95	
apache-hive-3.1.3-src.tar.gz	2022-04-08 17:42	25M	
apache-hive-3.1.3-src.tar.gz.asc	2022-04-08 17:42	488	
apache-hive-3.1.3-src.tar.gz.sha256	2022-04-08 17:42	95	

---

4. Extract the downloaded file to a directory of your choice.



5. Install bin folder from <https://github.com/HadiFadl/Hive-cmd> and replace the bin in the hive folder.
6. Set the HADOOP\_HOME and HIVE\_HOME environment variables to the path where you extracted the downloaded files.
  - a. Add %HADOOP\_HOME%\bin and %HIVE\_HOME%\bin to the PATH environment variable.



7. Open MySQL Workbench and create a new schema for Hive metastore database. Note down the schema name, username, and password.

Name	Date modified	Type	Size
hive-schema-0.10.0.mysql	24-10-2019 08:47	SQL Text File	32 KB
hive-schema-0.11.0.mysql	24-10-2019 08:47	SQL Text File	32 KB
hive-schema-0.12.0.mysql	24-10-2019 08:47	SQL Text File	33 KB
hive-schema-0.13.0.mysql	24-10-2019 08:47	SQL Text File	37 KB
hive-schema-0.14.0.mysql	24-10-2019 08:47	SQL Text File	36 KB
hive-schema-1.1.0.mysql	24-10-2019 08:47	SQL Text File	35 KB
hive-schema-1.2.0.mysql	24-10-2019 08:47	SQL Text File	35 KB
hive-schema-1.3.0.mysql	24-10-2019 08:47	SQL Text File	35 KB
hive-schema-2.0.0.mysql	24-10-2019 08:47	SQL Text File	35 KB
hive-schema-2.1.0.mysql	24-10-2019 08:47	SQL Text File	35 KB
hive-schema-2.2.0.mysql	05-09-2020 00:39	SQL Text File	35 KB
hive-schema-2.3.0.mysql	05-09-2020 00:39	SQL Text File	35 KB
hive-schema-3.0.0.mysql	29-03-2022 01:09	SQL Text File	47 KB
hive-schema-3.1.0.mysql	29-03-2022 01:09	SQL Text File	48 KB
hive-txn-schema-0.13.0.mysql	24-10-2019 08:47	SQL Text File	3 KB
hive-txn-schema-0.14.0.mysql	24-10-2019 08:47	SQL Text File	3 KB
hive-txn-schema-1.3.0.mysql	24-10-2019 08:47	SQL Text File	5 KB
hive-txn-schema-2.0.0.mysql	24-10-2019 08:47	SQL Text File	4 KB
hive-txn-schema-2.1.0.mysql	24-10-2019 08:47	SQL Text File	5 KB

a. Open the hive-schema-3.1.0 in the workbench

The screenshot shows the MySQL Workbench interface. In the 'Query Editor' tab, two SQL commands are run:

```

Query 1 [hive-schema-3.1.0 mysql]
1 • | show databases;
2 • | use demo;
3

```

In the 'Result Grid' tab, the output of the first command is shown, listing various databases including 'demo'. The second command creates the 'demo' database.

Database
demo
Hive
information_schema
mysql
performance_schema
saikat
sys
test_bigdata

The 'Action Output' tab at the bottom shows the execution log with the following entries:

#	Time	Action	Message
1	12:29:27	create database demo	1 row(s) affected
2	12:29:35	create database demo	Error Code: 1007: Can't create database 'demo'; database exists
3	12:29:49	create database demo	Error Code: 1007: Can't create database 'demo'; database exists
4	12:29:59	show databases	9 row(s) returned
5	12:30:08	show databases	9 row(s) returned
6	12:30:08	use demo	0 row(s) affected

Create database and initiate it first before running the hive-schema script.

8. In the `hive-site.xml` file located in `hive` folder, update the following properties:  
Creating File `Hive-site.xml`

Now we need to create the `Hive-site.xml` file in `hive` for configuring it :-

(We can find these files in Hive -> conf -> hive-default.xml.template)

We need to copy the hive-default.xml.template file and paste it in the same location and rename it to hive-site.xml. This will act as our main Config file for Hive.

PC > Shashank (D:) > Shashank > Study > Hive-3.1.2 > conf				
Name	Date modified	Type	Size	
beeline-log4j2.properties.template	23-08-2019 03:14	TEMPLATE File	2 KB	
hive-default.xml.template	23-08-2019 03:31	TEMPLATE File	294 KB	
hive-env.sh.template	23-08-2019 03:14	TEMPLATE File	3 KB	
hive-exec-log4j2.properties.template	23-08-2019 03:15	TEMPLATE File	3 KB	
hive-log4j2.properties.template	23-08-2019 03:14	TEMPLATE File	4 KB	
hive-site	04-04-2021 20:14	XML Document	295 KB	
ivysettings	23-08-2019 03:14	XML Document	3 KB	
llap-cli-log4j2.properties.template	23-08-2019 03:14	TEMPLATE File	4 KB	
llap-daemon-log4j2.properties.template	23-08-2019 03:14	TEMPLATE File	7 KB	
parquet-logging.properties	23-08-2019 03:14	PROPERTIES File	3 KB	

## 9. Editing the hive-site.xml

- Reove &#8 from the hive-site

```
• <property>
  <name>hive.querylog.location</name>
  <value>$HIVE_HOME/iotmp</value>
```

```
• <description>Location of Hive run time structured log  file</description>
</property><property>
  <name>hive.exec.local.scratchdir</name>
  <value>$HIVE_HOME/iotmp</value>
  <description>Local scratch space for Hive jobs</description>
</property><property>
  <name>hive.downloaded.resources.dir</name>
  <value>$HIVE_HOME/iotmp</value>
  <description>Temporary local directory for added resources in the remote file
system.</description>
</property>
```

```
• <property>
<name>hive.metastore.uris</name>
<value>thrift://<Your IP Address>:9083</value>
</property>
<property>
<name>javax.jdo.option.ConnectionUserName</name>
```

```

<value>hive</value>
<description>Username to use against metastore database</description>
</property>

<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://localhost:3306/<Your
Database>?createDatabaseIfNotExist=true</value>
<description>
JDBC connect string for a JDBC metastore.
To use SSL to encrypt/authenticate the connection, provide database-specific SSL flag in
the connection URL.
For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
</description>
</property>

<property>
<name>hive.metastore.warehouse.dir</name>
<value>hdfs://localhost:9000/user/hive/warehouse</value>
<description>location of default database for the warehouse</description>
</property>

<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value><Hive Password></value>
<description>password to use against metastore database</description>
</property>

<property>
<name>datanucleus.schema.autoCreateSchema</name>
<value>true</value>
</property>
<property>
<name>datanucleus.schema.autoCreateTables</name>
<value>True</value>
</property>

<property>
<name>datanucleus.schema.validateTables</name>
<value>true</value>
<description>validates existing schema against code. turn this on if you want to verify
existing schema</description>
</property>
```

```

<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
    <description>Driver class name for a JDBC metastore</description>
</property>

```

Install the .jar file from <https://mvnrepository.com/artifact/com.mysql/mysql-connector-j/8.0.32> and store it in the lib folder.

Open Powershell in administrative mode and run

*Start-all to run the Hadoop servers.*

```

PowerShell 7.3.3
PS C:\Users\sagar> start-all
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
Starting yarn services
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
PS C:\Users\sagar> |

```

Start the Hive metastore service by running the following command in the command prompt: "hive --service metastore"

```

PowerShell 7.3.3
PS C:\Users\sagar> hive --service metastore
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/hadoop-3.3.4/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/HIVE-3.1.3/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
2023-03-17 13:03:41,108 INFO conf.HiveConf: Found configuration file file:/C:/HIVE-3.1.3/conf/hive-site.xml
2023-03-17 13:03:42,915 INFO server.HiveServer2: STARTUP_MSG:
*****
STARTUP_MSG: Starting HiveServer2
STARTUP_MSG: host = sagar/172.27.32.1
STARTUP_MSG: args = []
STARTUP_MSG: version = 3.1.3
STARTUP_MSG: classpath = C:\hadoop-3.3.4\etc\hadoop;C:\hadoop-3.3.4\share\hadoop\common;C:\hadoop-3.3.4\share\hadoop\common\lib\accessors-smart-2.4.7.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\animal-sniffer-annotations-1.17.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\asm-5.0.4.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\audience-annotations-0.5.0.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\avro-1.7.7.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\checker-qual-2.5.2.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-beanutils-1.9.4.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-cli-1.2.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-codec-1.15.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-collections-3.2.2.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-compress-1.21.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-configuration2-2.1.1.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-daemon-1.0.15.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-io-2.8.0.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-lang3-3.12.0.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-logging-1.1.3.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-math3-3.1.1.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-net-3.6.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\commons-text-1.4.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\curator-client-4.2.0.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\curator-framework-4.2.0.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\curator-recipes-4.2.0.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\dnsjava-2.1.7.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\failureaccess-1.0.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\gsone-2.8.9.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\guava-27.0-jre.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\hadoop-annotations-3.3.4.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\hadoop-auth-3.3.4.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\hadoop-shaded-guava-1.1.1.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\hadoop-shaded protobuf_3_7-1.1.1.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\httpclient-4.5.13.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\httpcore-4.4.13.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\j2objc-annotations-1.1.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jackson-annotations-2.12.7.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jackson-core-asl-1.9.13.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jackson-databind-2.12.7.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jackson-core-2.12.7.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jackson-jaxrs-1.9.13.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jackson-mapper-asl-1.9.13.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jackson-xc-1.9.13.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jakarta-activation-api-1.2.1.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jaxb-impl-2.2.3-1.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jcip-annotations-1.0-1.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jersey-core-1.19.jar;C:\hadoop-3.3.4\share\hadoop\common\lib\jersey-json-1.19.jar;C:\hadoop-3.3.4\share\hadoop\c

```

Start the Hive server by running the following command in the command prompt: "hive --service hiveserver2"

Then run hive.

```
Administrator: PowerShell x Administrator: PowerShell x Administrator: PowerShell x Administrator: PowerShell x + - 
PowerShell 7.3.3
PS C:\Users\sagar> hive
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/hadoop-3.3.4/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/HIVE-3.1.3/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
2023-03-17 13:03:50,490 INFO conf.HiveConf: Found configuration file file:/C:/HIVE-3.1.3/conf/hive-site.xml
Hive Session ID = 67e74f26-bddc-48f9-8de7-b11fad7cfce2
2023-03-17 13:03:52,259 INFO SessionState: Hive Session ID = 67e74f26-bddc-48f9-8de7-b11fad7cfce2

Logging initialized using configuration in jar:file:/C:/HIVE-3.1.3/lib/hive-common-3.1.3.jar!/hive-log4j2.properties Async: true
2023-03-17 13:03:52,524 INFO SessionState:
Logging initialized using configuration in jar:file:/C:/HIVE-3.1.3/lib/hive-common-3.1.3.jar!/hive-log4j2.properties Async: true
2023-03-17 13:03:53,558 INFO session.SessionState: Created HDFS directory: /tmp/hive/sagar/67e74f26-bddc-48f9-8de7-b11fad7cfce2
2023-03-17 13:03:53,564 INFO session.SessionState: Created local directory: C:/HIVE-3.1.3/iotmp/67e74f26-bddc-48f9-8de7-b11fad7cfce2
2023-03-17 13:03:53,570 INFO session.SessionState: Created HDFS directory: /tmp/hive/sagar/67e74f26-bddc-48f9-8de7-b11fad7cfce2/tmp_space.db
2023-03-17 13:03:53,587 INFO conf.HiveConf: Using the default value passed in for log id: 67e74f26-bddc-48f9-8de7-b11fad7cfce2
2023-03-17 13:03:53,587 INFO session.SessionState: Updating thread name to 67e74f26-bddc-48f9-8de7-b11fad7cfce2 main
2023-03-17 13:03:55,000 INFO metastore.HiveMetaStore: 0: Opening raw store with implementation class:org.apache.hadoop.hive.metastore.ObjectStore
2023-03-17 13:03:55,034 WARN metastore.ObjectStore: datanucleus.autoStartMechanismMode is set to unsupported value null . Setting it to value: ignored
2023-03-17 13:03:55,040 INFO metastore.ObjectStore: ObjectStore, initialize called
2023-03-17 13:03:55,042 INFO conf.MetastoreConf: Found configuration file file:/C:/HIVE-3.1.3/conf/hive-site.xml
2023-03-17 13:03:55,044 INFO conf.MetastoreConf: Unable to find config file hivemetastore-site.xml
2023-03-17 13:03:55,044 INFO conf.MetastoreConf: Found configuration file null
2023-03-17 13:03:55,045 INFO conf.MetastoreConf: Unable to find config file metastore-site.xml
2023-03-17 13:03:55,045 INFO conf.MetastoreConf: Found configuration file null
2023-03-17 13:03:55,293 INFO DataNucleus.Persistence: Property datanucleus.cache.level2 unknown - will be ignored
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
2023-03-17 13:03:55,756 INFO hikari.HikariDataSource: HikariPool-1 - Starting...
2023-03-17 13:03:55,775 WARN util.DriverDataSource: Registered driver with driverClassName=com.mysql.jdbc.Driver was not found, trying direct instantiation.
2023-03-17 13:03:56,465 INFO hikari.HikariDataSource: HikariPool-1 - Start completed.
```

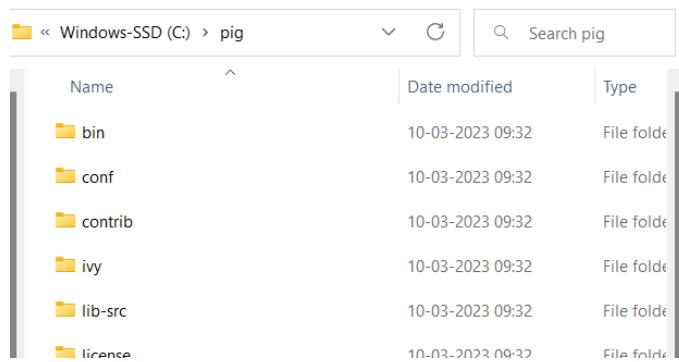
## PRACTICAL 12

## Pig Installation and Queries

## Steps for installation:

1. Download pig from official page. Download **pig-0.17.0.tar.gz** file:  
<https://downloads.apache.org/pig/pig-0.17.0/>

## 2. Unzip file



## 3. Set environment variables

Path	C:\Program Files\apache-cassandra-3.11.14\bin;C:\Program Fil...
PIG_HOME	C:\pig
PyCharm Community Editi...	C:\Program Files\JetBrains\PyCharm Community Edition 2022....
TEMP	C:\Users\prana\AppData\Local\Temp
TMP	C:\Users\prana\AppData\Local\Temp

%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\hive\bin
C:\pig\bin

## 4. Check pig version

```
C:\Users\prana>pig -version
'C:\hadoop-3.3.1\bin\hadoop-config.cmd' is not recognized as an internal or external command,
operable program or batch file.
'-Xmx1000M' is not recognized as an internal or external command,
operable program or batch file.
```

## 5. If received above error, solve by changing pig.cmd file as below

```
40 setlocal enabledelayedexpansion  
41  
42 set HADOOP_BIN_PATH=%HADOOP_HOME%\libexec  
43
```

## 6. Try pig -version command again.

```
PS C:\Users\Admin> cd D:/  
PS D:\> pig -version  
Apache Pig version 0.17.0 (r1797386)  
compiled Jun 02 2017, 15:41:58
```

```
C:\Users\prana>pig -version  
'C:\Program' is not recognized as an internal or external command,  
operable program or batch file.  
Apache Pig version 0.17.0 (r1797386)  
compiled Jun 02 2017, 15:41:58
```

## 7. Try running pig in local mode or in the hdfs mode

### a. Local mode:

```
C:\Users\prana>pig -x local  
'C:\Program' is not recognized as an internal or external command,  
operable program or batch file.  
2023-03-10 09:48:56,048 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL  
2023-03-10 09:48:56,049 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType  
2023-03-10 09:48:56,596 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 1  
:41:58  
2023-03-10 09:48:56,596 [main] INFO org.apache.pig.Main - Logging error messages to: C:\hadoop-3.3.1\logs\pig_16784219  
6584.log  
2023-03-10 09:48:56,667 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file C:\Users\prana/.pigbootup not  
found  
2023-03-10 09:48:56,930 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecate  
. Instead, use mapreduce.jobtracker.address  
2023-03-10 09:48:56,935 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to had  
oop file system at: file:///br/>2023-03-10 09:48:57,507 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprec  
ated. Instead, use dfs.bytes-per-checksum  
2023-03-10 09:48:57,567 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-bcd2bc7b-208  
-4d2b-99fa-9428bad14192  
2023-03-10 09:48:57,567 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set  
to false  
grunt>
```

```

PS C:\Users\Admin> cd D:/
PS D:> pig -version
Apache Pig version 0.17.0 (r1797386)
compiled Jun 02 2017, 15:41:58
PS D:> pig -x local
2023-04-12 12:29:29,567 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2023-04-12 12:29:29,567 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2023-04-12 12:29:29,842 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2023-04-12 12:29:29,842 [main] INFO org.apache.pig.Main - Logging error messages to: D:\hadoop-env\hadoop-3.3.0\logs\pig_1681282769828.log
2023-04-12 12:29:29,874 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file C:\Users\Admin/.pigbootup not found
2023-04-12 12:29:29,974 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2023-04-12 12:29:29,976 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
2023-04-12 12:29:30,119 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2023-04-12 12:29:30,135 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-ad6278fe-e8e4-434d-86a7-c7d1985f9f6b
2023-04-12 12:29:30,135 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> |

```

## Pig commands:

1. **Load:** Loads the data from the file system.

```

grunt> a = load 'data.txt' using PigStorage(',') as (lines:int);
2023-04-10 18:05:58,358 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
|grunt>

```

2. **Dump :** Executes the relation. Used to display the output

```

grunt> a = load 'data.txt' as (lines);
2023-04-10 18:08:02,078 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump a;

```

```

(1,2,3,4)
(7,8,9,4)
(3,5,7,1)
(1,5,9,7)
|grunt>

```

3. **Describe:** To describe the relation

```
grunt> describe a;
a: {lines: bytearray}
grunt>
```

4. **Pig -x local file.pig :** To execute the pig script

```
a = load 'data.txt' as (lines);
dump a;
describe a;
```

```
C:\Users\prana>pig -x local commands.pig
```

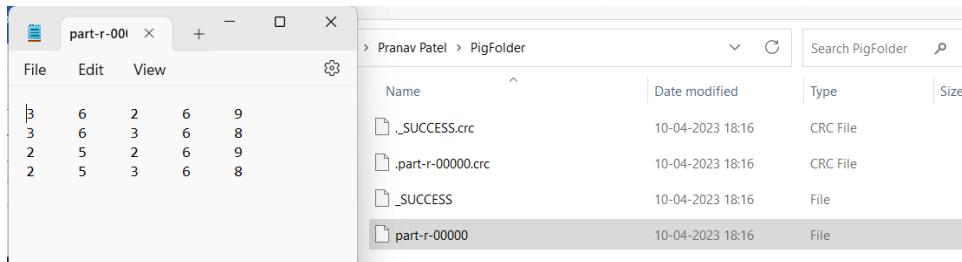
```
(1,2,3,4)
(7,8,9,4)
(3,5,7,1)
(1,5,9,7)
a: {lines: bytearray}
2023-04-10 18:11:33,175 [main] INFO
nds and 950 milliseconds (3950 ms)
```

5. **Cross :** To perform the cross product of 2 or more relations. It is an expensive operation. Displays the combined result of the relations.

```
grunt> cat pcross1.txt
2,5
3,6
4,7
5,8
6,9
7,10
8,11
9,12
10,13
11,14
12,15
13,16
14,17
15,18
16,19
17,20
18,21
19,22
20,23
21,24
22,25
23,26
24,27
25,28
26,29
27,30
28,31
29,32
30,33
31,34
32,35
33,36
34,37
35,38
36,39
37,40
38,41
39,42
40,43
41,44
42,45
43,46
44,47
45,48
46,49
47,50
48,51
49,52
50,53
51,54
52,55
53,56
54,57
55,58
56,59
57,60
58,61
59,62
60,63
61,64
62,65
63,66
64,67
65,68
66,69
67,70
68,71
69,72
70,73
71,74
72,75
73,76
74,77
75,78
76,79
77,80
78,81
79,82
80,83
81,84
82,85
83,86
84,87
85,88
86,89
87,90
88,91
89,92
90,93
91,94
92,95
93,96
94,97
95,98
96,99
97,100
98,101
99,102
100,103
101,104
102,105
103,106
104,107
105,108
106,109
107,110
108,111
109,112
110,113
111,114
112,115
113,116
114,117
115,118
116,119
117,120
118,121
119,122
120,123
121,124
122,125
123,126
124,127
125,128
126,129
127,130
128,131
129,132
130,133
131,134
132,135
133,136
134,137
135,138
136,139
137,140
138,141
139,142
140,143
141,144
142,145
143,146
144,147
145,148
146,149
147,150
148,151
149,152
150,153
151,154
152,155
153,156
154,157
155,158
156,159
157,160
158,161
159,162
160,163
161,164
162,165
163,166
164,167
165,168
166,169
167,170
168,171
169,172
170,173
171,174
172,175
173,176
174,177
175,178
176,179
177,180
178,181
179,182
180,183
181,184
182,185
183,186
184,187
185,188
186,189
187,190
188,191
189,192
190,193
191,194
192,195
193,196
194,197
195,198
196,199
197,200
198,201
199,202
200,203
201,204
202,205
203,206
204,207
205,208
206,209
207,210
208,211
209,212
210,213
211,214
212,215
213,216
214,217
215,218
216,219
217,220
218,221
219,222
220,223
221,224
222,225
223,226
224,227
225,228
226,229
227,230
228,231
229,232
230,233
231,234
232,235
233,236
234,237
235,238
236,239
237,240
238,241
239,242
240,243
241,244
242,245
243,246
244,247
245,248
246,249
247,250
248,251
249,252
250,253
251,254
252,255
253,256
254,257
255,258
256,259
257,260
258,261
259,262
260,263
261,264
262,265
263,266
264,267
265,268
266,269
267,270
268,271
269,272
270,273
271,274
272,275
273,276
274,277
275,278
276,279
277,280
278,281
279,282
280,283
281,284
282,285
283,286
284,287
285,288
286,289
287,290
288,291
289,292
290,293
291,294
292,295
293,296
294,297
295,298
296,299
297,300
298,301
299,302
300,303
301,304
302,305
303,306
304,307
305,308
306,309
307,310
308,311
309,312
310,313
311,314
312,315
313,316
314,317
315,318
316,319
317,320
318,321
319,322
320,323
321,324
322,325
323,326
324,327
325,328
326,329
327,330
328,331
329,332
330,333
331,334
332,335
333,336
334,337
335,338
336,339
337,340
338,341
339,342
340,343
341,344
342,345
343,346
344,347
345,348
346,349
347,350
348,351
349,352
350,353
351,354
352,355
353,356
354,357
355,358
356,359
357,360
358,361
359,362
360,363
361,364
362,365
363,366
364,367
365,368
366,369
367,370
368,371
369,372
370,373
371,374
372,375
373,376
374,377
375,378
376,379
377,380
378,381
379,382
380,383
381,384
382,385
383,386
384,387
385,388
386,389
387,390
388,391
389,392
390,393
391,394
392,395
393,396
394,397
395,398
396,399
397,400
398,401
399,402
399,403
400,404
401,405
402,406
403,407
404,408
405,409
406,410
407,411
408,412
409,413
410,414
411,415
412,416
413,417
414,418
415,419
416,420
417,421
418,422
419,423
420,424
421,425
422,426
423,427
424,428
425,429
426,430
427,431
428,432
429,433
430,434
431,435
432,436
433,437
434,438
435,439
436,440
437,441
438,442
439,443
440,444
441,445
442,446
443,447
444,448
445,449
446,450
447,451
448,452
449,453
450,454
451,455
452,456
453,457
454,458
455,459
456,460
457,461
458,462
459,463
460,464
461,465
462,466
463,467
464,468
465,469
466,470
467,471
468,472
469,473
470,474
471,475
472,476
473,477
474,478
475,479
476,480
477,481
478,482
479,483
480,484
481,485
482,486
483,487
484,488
485,489
486,490
487,491
488,492
489,493
490,494
491,495
492,496
493,497
494,498
495,499
496,500
497,501
498,502
499,503
500,504
501,505
502,506
503,507
504,508
505,509
506,510
507,511
508,512
509,513
510,514
511,515
512,516
513,517
514,518
515,519
516,520
517,521
518,522
519,523
520,524
521,525
522,526
523,527
524,528
525,529
526,530
527,531
528,532
529,533
530,534
531,535
532,536
533,537
534,538
535,539
536,540
537,541
538,542
539,543
540,544
541,545
542,546
543,547
544,548
545,549
546,550
547,551
548,552
549,553
550,554
551,555
552,556
553,557
554,558
555,559
556,560
557,561
558,562
559,563
560,564
561,565
562,566
563,567
564,568
565,569
566,570
567,571
568,572
569,573
570,574
571,575
572,576
573,577
574,578
575,579
576,580
577,581
578,582
579,583
580,584
581,585
582,586
583,587
584,588
585,589
586,590
587,591
588,592
589,593
590,594
591,595
592,596
593,597
594,598
595,599
596,600
597,601
598,602
599,603
600,604
601,605
602,606
603,607
604,608
605,609
606,610
607,611
608,612
609,613
610,614
611,615
612,616
613,617
614,618
615,619
616,620
617,621
618,622
619,623
620,624
621,625
622,626
623,627
624,628
625,629
626,630
627,631
628,632
629,633
630,634
631,635
632,636
633,637
634,638
635,639
636,640
637,641
638,642
639,643
640,644
641,645
642,646
643,647
644,648
645,649
646,650
647,651
648,652
649,653
650,654
651,655
652,656
653,657
654,658
655,659
656,660
657,661
658,662
659,663
660,664
661,665
662,666
663,667
664,668
665,669
666,670
667,671
668,672
669,673
670,674
671,675
672,676
673,677
674,678
675,679
676,680
677,681
678,682
679,683
680,684
681,685
682,686
683,687
684,688
685,689
686,690
687,691
688,692
689,693
690,694
691,695
692,696
693,697
694,698
695,699
696,700
697,701
698,702
699,703
700,704
701,705
702,706
703,707
704,708
705,709
706,710
707,711
708,712
709,713
710,714
711,715
712,716
713,717
714,718
715,719
716,720
717,721
718,722
719,723
720,724
721,725
722,726
723,727
724,728
725,729
726,730
727,731
728,732
729,733
730,734
731,735
732,736
733,737
734,738
735,739
736,740
737,741
738,742
739,743
740,744
741,745
742,746
743,747
744,748
745,749
746,750
747,751
748,752
749,753
750,754
751,755
752,756
753,757
754,758
755,759
756,760
757,761
758,762
759,763
760,764
761,765
762,766
763,767
764,768
765,769
766,770
767,771
768,772
769,773
770,774
771,775
772,776
773,777
774,778
775,779
776,780
777,781
778,782
779,783
780,784
781,785
782,786
783,787
784,788
785,789
786,790
787,791
788,792
789,793
790,794
791,795
792,796
793,797
794,798
795,799
796,800
797,801
798,802
799,803
800,804
801,805
802,806
803,807
804,808
805,809
806,810
807,811
808,812
809,813
810,814
811,815
812,816
813,817
814,818
815,819
816,820
817,821
818,822
819,823
820,824
821,825
822,826
823,827
824,828
825,829
826,830
827,831
828,832
829,833
830,834
831,835
832,836
833,837
834,838
835,839
836,840
837,841
838,842
839,843
840,844
841,845
842,846
843,847
844,848
845,849
846,850
847,851
848,852
849,853
850,854
851,855
852,856
853,857
854,858
855,859
856,860
857,861
858,862
859,863
860,864
861,865
862,866
863,867
864,868
865,869
866,870
867,871
868,872
869,873
870,874
871,875
872,876
873,877
874,878
875,879
876,880
877,881
878,882
879,883
880,884
881,885
882,886
883,887
884,888
885,889
886,890
887,891
888,892
889,893
890,894
891,895
892,896
893,897
894,898
895,899
896,900
897,901
898,902
899,903
900,904
901,905
902,906
903,907
904,908
905,909
906,910
907,911
908,912
909,913
910,914
911,915
912,916
913,917
914,918
915,919
916,920
917,921
918,922
919,923
920,924
921,925
922,926
923,927
924,928
925,929
926,930
927,931
928,932
929,933
930,934
931,935
932,936
933,937
934,938
935,939
936,940
937,941
938,942
939,943
940,944
941,945
942,946
943,947
944,948
945,949
946,950
947,951
948,952
949,953
950,954
951,955
952,956
953,957
954,958
955,959
956,960
957,961
958,962
959,963
960,964
961,965
962,966
963,967
964,968
965,969
966,970
967,971
968,972
969,973
970,974
971,975
972,976
973,977
974,978
975,979
976,980
977,981
978,982
979,983
980,984
981,985
982,986
983,987
984,988
985,989
986,990
987,991
988,992
989,993
990,994
991,995
992,996
993,997
994,998
995,999
996,1000
997,1001
998,1002
999,1003
1000,1004
1001,1005
1002,1006
1003,1007
1004,1008
1005,1009
1006,1010
1007,1011
1008,1012
1009,1013
1010,1014
1011,1015
1012,1016
1013,1017
1014,1018
1015,1019
1016,1020
1017,1021
1018,1022
1019,1023
1020,1024
1021,1025
1022,1026
1023,1027
1024,1028
1025,1029
1026,1030
1027,1031
1028,1032
1029,1033
1030,1034
1031,1035
1032,1036
1033,1037
1034,1038
1035,1039
1036,1040
1037,1041
1038,1042
1039,1043
1040,1044
1041,1045
1042,1046
1043,1047
1044,1048
1045,1049
1046,1050
1047,1051
1048,1052
1049,1053
1050,1054
1051,1055
1052,1056
1053,1057
1054,1058
1055,1059
1056,1060
1057,1061
1058,1062
1059,1063
1060,1064
1061,1065
1062,1066
1063,1067
1064,1068
1065,1069
1066,1070
1067,1071
1068,1072
1069,1073
1070,1074
1071,1075
1072,1076
1073,1077
1074,1078
1075,1079
1076,1080
1077,1081
1078,1082
1079,1083
1080,1084
1081,1085
1082,1086
1083,1087
1084,1088
1085,1089
1086,1090
1087,1091
1088,1092
1089,1093
1090,1094
1091,1095
1092,1096
1093,1097
1094,1098
1095,1099
1096,1100
1097,1101
1098,1102
1099,1103
1100,1104
1101,1105
1102,1106
1103,1107
1104,1108
1105,1109
1106,1110
1107,1111
1108,1112
1109,1113
1110,1114
1111,1115
1112,1116
1113,1117
1114,1118
1115,1119
1116,1120
1117,1121
1118,1122
1119,1123
1120,1124
1121,1125
1122,1126
1123,1127
1124,1128
1125,1129
1126,1130
1127,1131
1128,1132
1129,1133
1130,1134
1131,1135
1132,1136
1133,1137
1134,1138
1135,1139
1136,1140
1137,1141
1138,1142
1139,1143
1140,1144
1141,1145
1142,1146
1143,1147
1144,1148
1145,1149
1146,1150
1147,1151
1148,1152
1149,1153
1150,1154
1151,1155
1152,1156
1153,1157
1154,1158
1155,1159
1156,1160
1157,1161
1158,1162
1159,1163
1160,1164
1161,1165
1162,1166
1163,1167
1164,1168
1165,1169
1166,1170
1167,1171
1168,1172
1169,1173
1170,1174
1171,1175
1172,1176
1173,1177
1174,1178
1175,1179
1176,1180
1177,1181
1178,1182
1179,1183
1180,1184
1181,1185
1182,1186
1183,1187
1184,1188
1185,1189
1186,1190
1187,1191
1188,1192
1189,1193
1190,1194
1191,1195
1192,1196
1193,1197
1194,1198
1195,1199
1196,1200
1197,1201
1198,1202
1199,1203
1200,1204
1201,1205
1202,1206
1203,1207
1204,1208
1205,1209
1206,1210
1207,1211
1208,1212
1209,1213
1210,1214
1211,1215
1212,1216
1213,1217
1214,1218
1215,1219
1216,1220
1217,1221
1218,1222
1219,1223
1220,1224
1221,1225
1222,1226
1223,1227
1224,1228
1225,1229
1226,1230
1227,1231
1228,1232
1229,1233
1230,1234
1231,1235
1232,1236
1233,1237
1234,1238
1235,1239
1236,1240
1237,1241
1238,1242
1239,1243
1240,1244
1241,1245
1242,1246
1243,1247
1244,1248
1245,1249
1246,1250
1247,1251
1248,1252
1249,1253
1250,1254
1251,1255
1252,1256
1253,1257
1254,1258
1255,1259
1256,1260
1257,1261
1258,1262
1259,1263
1260,1264
1261,1265
1262,1266
1263,1267
1264,1268
1265,1269
1266,1270
1267,1271
1268,1272
1269,1273
1270,1274
1271,1275
1272,1276
1273,1277
1274,1278
1275,1279
1276,1280
1277,1281
1278,1282
1279,1283
1280,1284
1281,1285
1282,1286
1283,1287
1284,1288
1285,1289
1286,1290
1287,1291
1288,1292
1289,1293
1290,1294
1291,1295
1292,
```

6. **Store:** To store the output relation into a folder. Folder will contain another file where the result will be stored

```
grunt> store result into 'PigFolder';
```



7. **Filter :** Will filter input based on a criteria provided.

```
grunt> a = load 'filter.txt' using PigStorage(',') as (a1:int,a2:int);
2023-04-10 18:19:10,168 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - i
o.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2023-04-10 18:19:10,202 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - i
o.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump a;
```

```
(1,2)
(2,8)
(4,5)
(3,2)
(9,1)
(5,2)
(1,0)
```

```
grunt> result = filter a by a2==2;
grunt> dump result;
```

```
apRedU
(1,2)
(3,2)
(5,2)
```

8. **Foreach :** Generates data transformations based on columns of data.

```
grunt> a = load 'data.txt' using PigStorage(',') as (a1:int,a2:int,a3:int,a4:int);
2023-04-11 14:51:56,168 [main] INFO org.apache.hadoop.conf.Configuration.deprecation
ted. Instead, use dfs.bytes-per-checksum
grunt> dump a;
```

```
(1,2,3,4)
(7,8,9,4)
(3,5,7,1)
(1,5,9,7)
grunt> 
```

```
grunt> fe = foreach a generate a1,a2;
grunt> dump fe;
```

```
(1,2)
(7,8)
(3,5)
(1,5)
grunt> 
```

9. **Group:** Group operator is used to group the data in one or more relations. It groups the tuples that contain a similar group key. If the group key has more than one field, it treats it as tuple otherwise it will be the same type as that of the group key. In a result, it provides a relation that contains one tuple per group.

Data:

```
Jason,Roy,1
Jim,Halpert,2
Michael,Scott,3
Jake,Peralta,4
Jason,Bourne,5
Michael,Scarn,6
```

```
grunt> a = load 'input.txt' using PigStorage(',') as (fname:chararray, lname:chararray, id:int);
2023-04-11 15:00:10,068 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.ted. Instead, use dfs.bytes-per-checksum
grunt> dump a;
```

```
(Jason,Roy,1)
(Jim,Halpert,2)
(Michael,Scott,3)
(Jake,Peralta,4)
(Jason,Bourne,5)
(Michael,Scarn,6)
grunt> 
```

```
grunt> groupfname = group a by fname;
grunt> dump groupfname;
```

```
(Jim,{{(Jim,Halpert,2)})})
(Jake,{{(Jake,Peralta,4)})})
(Jason,{{(Jason,Bourne,5),(Jason,Roy,1)})})
(Michael,{{(Michael,Scarn,6),(Michael,Scott,3)})})
grunt>
```

#### 10. Limit : Limit no of output tuples

```
grunt> a = load 'data.txt' using PigStorage(',') as (a1:int,a2:int);
2023-04-11 16:37:45,569 [main] INFO org.apache.hadoop.conf.Configuration
.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump a;
```

```
Total input paths to process : 1
(1,2)
(7,8)
(3,5)
(1,5)
grunt> result = limit a 2;
grunt> dump result;
```

```
Total input paths to process : 1
(1,2)
(7,8)
grunt>
```

#### 11. Order by : The Apache Pig ORDER BY operator sorts a relation based on one or more fields. It maintains the order of tuples

```
grunt> a = load 'data.txt' using PigStorage(',') as (a1:int,a2:int);
2023-04-11 16:37:45,569 [main] INFO org.apache.hadoop.conf.Configuration
.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump a;
```

```
(1,2)
(7,8)
(3,5)
(1,5)
grunt> result = order a by a1 desc;
grunt> dump result;
```

```
(7,8)
(3,5)
(1,5)
(1,2)
grunt>
```

12. **Split** : The Apache Pig SPLIT operator breaks the relation into two or more relations according to the provided expression. Here, a tuple may or may not be assigned to one or more than one relation.

```
grunt> a = load 'data.txt' using PigStorage(',') as (a1:int,a2:int);
2023-04-11 16:37:45,569 [main] INFO org.apache.hadoop.conf.Configuration
.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump a;
```

```
Total input paths to process : 1
(1,2)
(7,8)
(3,5)
(1,5)
grunt> split a into x if a1<=2, y if a1>2;
grunt> dump x
```

```
Total input paths to process : 1
(1,2)
(1,5)
grunt> dump y;
```

```
Total input
(7,8)
(3,5)
grunt>
```

13. **Distinct** : Used to remove duplicate tuples in a relation. Pig sorts the data and then eliminated duplicates

```
grunt> a = load 'distinct.txt' using PigStorage(',') as (a1:int,a2:int,a3:int);
2023-04-11 16:43:55,759 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> dump a;
```

```
(1,3,5)
(2,1,4)
(1,3,5)
(1,4,2)
(2,1,4)
grunt> result = distinct a;
grunt> dump result;
```

```
(1,3,5)
(1,4,2)
(2,1,4)
grunt>
```

## 14. Join

The JOIN operator is used to combine records from two or more relations. While performing a join operation, we declare one (or a group of) tuple(s) from each relation, as keys. When these keys match, the two particular tuples are matched, else the records are dropped. Joins can be of the following types:

- Self Join :** Used to join a table with itself, as if there are 2 relations, temporarily renaming at least 1 relation

```
grunt> cat orders.txt
102,2009-10-08 00:00:00,3,3000
100,2009-10-08 00:00:00,3,1500
101,2009-11-20 00:00:00,2,1560
103,2008-05-20 00:00:00,4,2060
105,2010-03-29 00:00:00,7,4500
```

```
grunt> cat customers.txt
1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00
3,Kaushik,23,Kota,2000.00
4,Bharat,25,Mumbai,6500.00
5,Sachin,27,Bhopal,8500.00
6,Komal,22,MP,4500.0
7,Muffy,24,Indore,10000.0
grunt>
```

```

grunt> a = load 'customers.txt' using PigStorage(',') as (id:int,name:chararray,age:int,address:chararray,salary:int);
2023-04-11 16:50:54,017 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> b = load 'customers.txt' using PigStorage(',') as (id:int,name:chararray,age:int,address:chararray,salary:int);
2023-04-11 16:51:47,877 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> result = join a by id, b by id;
grunt> dump result;

```

```

Total input paths to process : 1
(1,Ramesh,32,Ahmedabad,2000,1,Ramesh,32,Ahmedabad,2000)
(2,Khilan,25,Delhi,1500,2,Khilan,25,Delhi,1500)
(3,Kaushik,23,Kota,2000,3,Kaushik,23,Kota,2000)
(4,Bharat,25,Mumbai,6500,4,Bharat,25,Mumbai,6500)
(5,Sachin,27,Bhopal,8500,5,Sachin,27,Bhopal,8500)
(6,Komal,22,MP,4500,6,Komal,22,MP,4500)
(7,Muffy,24,Indore,10000,7,Muffy,24,Indore,10000)

```

- b. **Inner Join :** Also referred to as Equijoin. Returns rows if there is a match in both tables.

```

grunt> a = load 'customers.txt' using PigStorage(',') as (id:int,name:chararray,age:int,address:chararray,salary:int);
2023-04-11 16:50:54,017 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum

```

```

grunt> b = load 'orders.txt' using PigStorage(',') as (oid:int,date:chararray,customer_id:int,amount:int);
2023-04-11 16:53:31,889 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum

```

```

grunt> c = join a by id,b by customer_id;
grunt> dump c;

```

```

Total input paths to process : 1
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,Kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,Kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Bharat,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(7,Muffy,24,Indore,10000,105,2010-03-29 00:00:00,7,4500)
grunt>

```

- c. **Outer Join :** Returns all rows from at least one of the relations. Can be Left outer join, Right outer join, or full outer join.

i. **Left outer join :** Returns all rows from the left table, even if there are no matches in the right relation.

ii. **Right outer join :** Returns all rows from the right table, even if there are no matches in the left table

iii. **Full Outer Join :** Returns all rows when there is a match in match in one of the relations.

```
grunt> d = join a by id LEFT OUTER, b by customer_id;
grunt> dump d;
```

```
Total input paths to process : 1
(1,Ramesh,32,Ahemdabad,2000,,,)
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,Kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,Kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Bharat,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(5,Sachin,27,Bhopal,8500,,,)
(6,Komal,22,MP,4500,,,)
(7,Muffy,24,Indore,10000,105,2010-03-29 00:00:00,7,4500)
grunt>
```

```
grunt> e = join a by id Right, b by customer_id;
grunt> dump e;
```

```
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,Kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,Kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Bharat,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(7,Muffy,24,Indore,10000,105,2010-03-29 00:00:00,7,4500)
grunt>
```

```
grunt> f = join a by id full outer, b by customer_id;
grunt> dump f;
```

```
(1,Ramesh,32,Ahemdabad,2000,,,)
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,Kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,Kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Bharat,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(5,Sachin,27,Bhopal,8500,,,)
(6,Komal,22,MP,4500,,,)
(7,Muffy,24,Indore,10000,105,2010-03-29 00:00:00,7,4500)
grunt>
```

## 15. Creating the csv file containing sales records

Data:

```
'12-03-2020','shirts','250','Cash','Joe','New York'
'01-12-2022','trousers','450','Card','John','Los Angeles'
'19-07-2021','backpack','1000','Cash','John','Pennsylvania'
'25-09-2023','suitcase','2500','Card','Neo','New York'
'21-10-2019','bottle','100','Card','Michael','Colorado'
'31-01-2017','soap','50','Cash','Jim','Scranton'
```

```
'28-02-2022','telephone','500','Cash','Calvin','Dallas'
'09-11-2018','batteries','40','Cash','Dwayne','New Jersey'
'17-08-2019','tie','140','Card','Adam','New York'
```

## 16. Load the records and group them based on the city

```
grunt> sales = load 'sales.csv' using PigStorage(',') as (tdate:chararray,product:chararray,price:chararray,ptype:chararray,name:chararray,city:chararray);
2023-04-11 17:05:02,274 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> gbcity = group sales by city;
grunt> dump gbcity
Total input paths to process : 1
('Dallas',{('28-02-2022','telephone','500','Cash','Calvin','Dallas')})
('Colorado',{('21-10-2019','bottle','100','Card','Michael','Colorado')})
('New York',{('17-08-2019','tie','140','Card','Adam','New York'),('25-09-2023','suitcase','2500','Card','Neo','New York'),('12-03-2020','shirts','250','Cash','Joe','New York')})
('Scranton',{('31-01-2017','soap','50','Cash','Jim','Scranton')})
('New Jersey',{('09-11-2018','batteries','40','Cash','Dwayne','New Jersey')})
('Los Angeles',{('01-12-2022','trousers','450','Card','John','Los Angeles')})
('Pennsylvania',{('19-07-2021','backpack','1000','Cash','John','Pennsylvania')})
grunt>
```

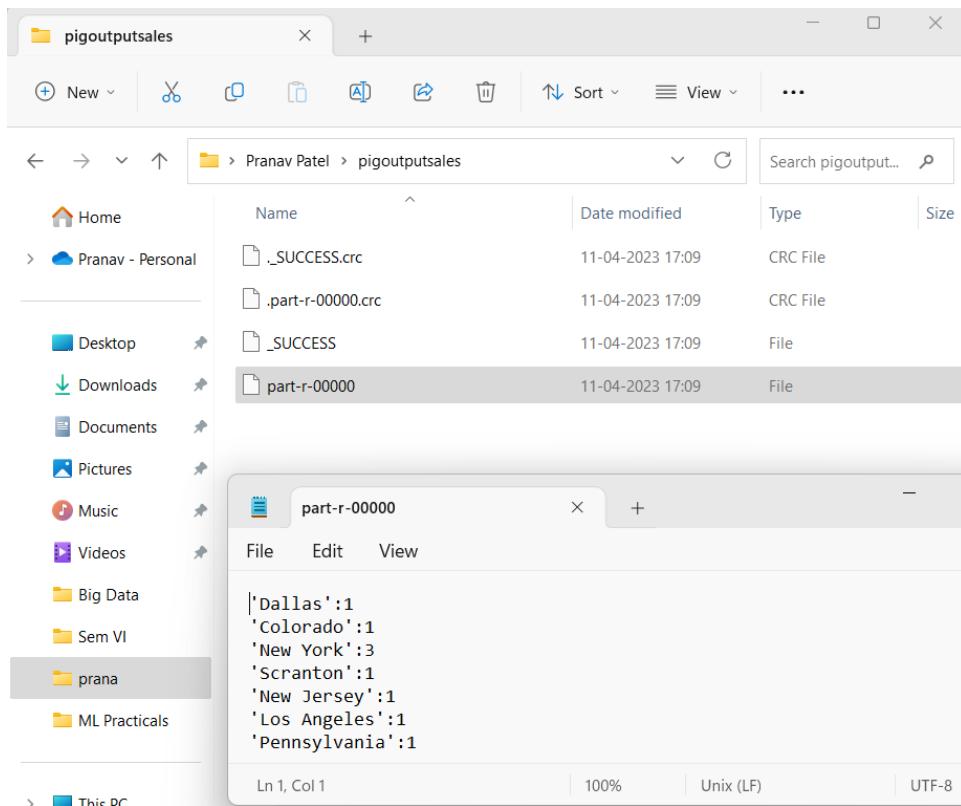
## 17. Count the number of sales in each city.

Note: Concat command will concatenate 2 or more fields of the same type

```
grunt> ccity = foreach gbcity generate CONCAT((chararray)$0,CONCAT(':',(chararray)COUNT($1)));
grunt> dump ccity
Total input paths to process : 1
('Dallas':1)
('Colorado':1)
('New York':3)
('Scranton':1)
('New Jersey':1)
('Los Angeles':1)
('Pennsylvania':1)
```

## 18. Store the sales records obtained earlier into a folder and verify them

```
grunt> store ccity into 'pigoutputsales' using PigStorage('\t');
```



## PRACTICAL 13

### Hbase Installation

Name: Mohd Bilal Ali

Roll No: KCTBCS(005)

Date:

**Pre-requisites:** We are going to make a standalone setup of HBase in our machine which requires us to

- Install Java JDK 1.8 - We can download and install it from <https://www.oracle.com/java/technologies/downloads/> and set JAVA\_HOME in environment variable.
- Download Hbase-bin - Download Apache Hbase-2.2.5 from <https://archive.apache.org/dist/hbase/2.2.5/>

#### Steps:

- Step 1 – Extract all files from the archive
- Step 2 – Create folders named "hbase" and "zookeeper"
- Step 3 – Deleting line in HBase.cmd

Open C:\hbase-2.2.5\bin\hbase.cmd in any text editor.

Search for line %HEAP\_SETTINGS% and remove the highlighted part.

```
set java_arguments=%HEAP_SETTINGS%
%HBASE_OPTS% -classpath "%CLASSPATH%" %CLASS%
%hbase-command-arguments%
```

- Step 4 – Adding lines in hbase-env.cmd

Open C:\hbase-2.2.5\conf\hbase-env.cmd in any text editor.

Add the below lines in the file after the comment session.

```
set JAVA_HOME=%JAVA_HOME%
set HBASE_CLASSPATH=%HBASE_HOME%\lib\client-facing-thirdparty\
set HBASE_HEAPSIZE=8000
set HBASE_OPTS="-XX:+UseConcMarkSweepGC" "-Djava.net.preferIPv4Stack=true"
```

Name
bin
conf
docs
hbase
hbase-webapps
lib
zookeeper
CHANGES
LEGAL
LICENSE
NOTICE

```

set SERVER_GC_OPTS="-verbose:gc" "-XX:+PrintGCDetails" "-XX:+PrintGCDateStamps" %HBASE_GC_OPTS%
set HBASE_USE_GC_LOGFILE=true

set HBASE_JMX_BASE="-Dcom.sun.management.jmxremote.ssl=false" "-Dcom.sun.management.jmxremote.authenticate=false"

set HBASE_MASTER_OPTS=%HBASE_JMX_BASE% "-Dcom.sun.management.jmxremote.port=10101"
set HBASE_REGIONSERVER_OPTS=%HBASE_JMX_BASE% "-Dcom.sun.management.jmxremote.port=10102"
set HBASE_THRIFT_OPTS=%HBASE_JMX_BASE% "-Dcom.sun.management.jmxremote.port=10103"
set HBASE_ZOOKEEPER_OPTS=%HBASE_JMX_BASE% -Dcom.sun.management.jmxremote.port=10104
set HBASE_REGIONSERS=%HBASE_HOME%\conf\regionservers
set HBASE_LOG_DIR=%HBASE_HOME%\logs
set HBASE_IDENT_STRING=%USERNAME%
set HBASE_MANAGES_ZK=true

```

- Step 5 – Adding lines in hbase-site.xml**

Open C:\hbase-2.2.5\conf\hbase-site.xml in any text editor.

Add the below lines inside the <configuration> tag.

```

<property>
  <name>hbase.rootdir</name>
  <value>file:///C:/Documents/hbase-2.2.5/hbase</value>
</property>
<property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/C:/Documents/hbase-2.2.5/zookeeper</value>
</property>
<property>
  <name> hbase.zookeeper.quorum</name>
  <value>localhost</value>
</property>

```

- Step 6 – Setting Environment Variables:**

Input the following:

- Variable name: HBASE\_HOME
- Variable Value: Put the path of the Hbase folder.

## Starting the HBASE shell:

- Step 1 – Use start-hbase.cmd:**

Go to the bin folder of hbase and start a cmd prompt there

Then type: **start-hbase.cmd**

- Step 2 – Start hbase shell:**

Edit User Variable

Variable name:	HBASE_HOME
Variable value:	C:\hbase-2.2.5

```

[HBase Distribution - C:\hbase-2.2.5\bin\hbase.cmd master start]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/hbase-2.2.5/lib/
SLF4J: Found binding in [jar:file:/C:/hadoop-3.3.0/sha
SLF4J: See http://www.slf4j.org/codes.html#multiple_b
2023-02-05T16:20:26.295+0530: [GC (Allocation Failure)
00, real=0.01 secs]
log4j:ERROR Could not find value for key log4j.appende
log4j:ERROR Could not instantiate appender named "DRFA
SLF4J: Actual binding is of type [org.slf4j.impl.Log4j
2023-02-05 16:20:26,361 INFO [main] master.HMaster: S
2023-02-05 16:20:26,362 INFO [main] util.VersionInfo:
2023-02-05 16:20:26,362 INFO [main] util.VersionInfo:
2023-02-05 16:20:26,363 INFO [main] util.VersionInfo:
2023-02-05 16:20:26,363 INFO [main] util.VersionInfo:
2023-02-05 16:20:26,773 INFO [main] master.HMasterCom
2023-02-05 16:20:26,789 INFO [main] server.ZooKeeperS
2023-02-05 16:20:26,790 INFO [main] server.ZooKeeperS
2023-02-05 16:20:26,790 INFO [main] server.ZooKeeperS
doop\mapreduce\hadoop-mapreduce-client-app-3.3.0.jar;C:
ore-3.3.0.jar;C:\hadoop-3.3.0\share\hadoop\mapreduce\h
adoop\mapreduce\hadoop-mapreduce-client-jobclient-3.3.
produce-client-nativetask-3.3.0.jar;C:\hadoop-3.3.0\sh
r;C:\hadoop-3.3.0\share\hadoop\mapreduce\hadoop-mapred
2023-02-05 16:20:26,790 INFO [main] server.ZooKeeperS
2023-02-05 16:20:26,790 INFO [main] server.ZooKeeperS

```

```

C:\hbase-2.2.5\bin>hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book
Version 2.2.5, rf76a601273e834267b55c0cda12474590283fd4c, 202
Took 0.0020 seconds
'stty' is not recognized as an internal or external command,
operable program or batch file.
hbase(main):001:0>

```

In HBase, interactive shell mode is used to interact with HBase for table operations, table management, and data modeling.

## Hbase Commands

### General commands

In Hbase, general commands are categorized into following commands

- **Status**
- **Version**
- **Table\_help ( scan, drop, get, put, disable, etc.)**

- **Whoami**

To get enter into HBase shell command, first of all, we have to execute the code as mentioned below  
**hbase Shell**

Once we get to enter into HBase shell, we can execute all shell commands mentioned below. With the help of these commands, we can perform all type of table operations in the HBase shell mode.

Let us look into all of these commands and their usage one by one with an example.

---

### **Status**

Syntax: status

This command will give details about the system status like a number of servers present in the cluster, active server count, and average load value. You can also pass any particular parameters depending on how detailed status you want to know about the system. The parameters can be ‘summary’, ‘simple’, or ‘detailed’, the default parameter provided is “summary”.

**Example:**

```
hbase(main):023:0> status
1 active master, 0 backup masters, 1 servers, 0 dead,
Took 0.0190 seconds
hbase(main):024:0> status 'simple'
active master: KIRANBHAI:16000 1675594226913
0 backup masters
1 live servers
    KIRANBHAI:16020 1675594229774
        requestsPerSecond=0.0, numberOfOnlineRegions=
ores=6, numberOfStorefiles=6, storefileUncompressedSi
    storefileIndexSizeKB=0, readRequestsCount=88, filter
1, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, total
urrentCompactedKVs=0, compactionProgressPct=NaN, copr
0 dead servers
Aggregate load: 0, regions: 3
Took 0.0230 seconds
```

---

### **whoami**

Syntax:

Syntax: Whoami

This command “whoami” is used to return the current HBase user information from the HBase cluster.

**Example:**

```
hbase(main):027:0> whoami
shahk (auth:SIMPLE)
    groups: docker-users, Administrators, Users
Took 0.0150 seconds
```

---

## **Tables Managements commands**

These commands will allow programmers to create tables and table schemas with rows and column families.

The following are Table Management commands

- Create
- List
- Describe
- Disable
- Disable\_all
- Enable
- Enable\_all
- Drop

- Drop\_all
- Show\_filters
- Alter
- Alter\_status

Let us look into various command usage in HBase with an example.

### Create

Syntax: create <tablename>, <columnfamilyname>

#### Example:

Creating a table named ‘kccs’ with column family ‘students’

```
hbase(main):010:0> create 'kccs','students'
Created table kccs
Took 1.2880 seconds
=> Hbase::Table - kccs
```

The above example explains how to create a table in HBase with the specified name given according to the dictionary or specifications as per column family. In addition to this we can also pass some table-scope attributes as well into it.

In order to check whether the table ‘kccs’ is created or not, we have to use the “list” command as mentioned below.

### List

Syntax:list

```
hbase(main):028:0> list
TABLE
kccs
1 row(s)
Took 0.0100 seconds
=> ["kccs"]
```

- “List” command will display all the tables that are present or created in HBase
- The output showing in above screen shot is currently showing the existing tables in HBase
- We can filter output values from tables by passing optional regular expression parameters

### Describe

Syntax:describe <table name>

#### Example:

```
hbase(main):011:0> describe 'kccs'
Table kccs is ENABLED
kccs
COLUMN FAMILIES DESCRIPTION
{NAME => 'students', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

1 row(s)

QUOTAS
0 row(s)
Took 0.0410 seconds
```

hbase(main):011:0>describe 'kccs'

This command describes the named table.

- It will give more information about column families present in the mentioned table
- In our case, it gives the description about table “kccs”

- It will give information about table name with column families, associated filters, versions and some more details.
- 

### **disable**

Syntax: disable <tablename>

Example: disable 'kccs'

```
hbase(main):029:0> disable 'kccs'  
Took 0.7930 seconds  
hbase(main):030:0> describe 'kccs'  
Table kccs is DISABLED
```

- This command will start disabling the named table
  - If table needs to be deleted or dropped, it has to be disabled first
- 

### **disable\_all**

Syntax: disable\_all<"matching regex">

- This command will disable all the tables matching the given regex.
  - The implementation is same as delete command (Except adding regex for matching)
  - Once the table gets disabled the user can able to delete the table from HBase
  - Before delete or dropping table, it should be disabled first
- 

### **Enable**

Syntax: enable <tablename>

Example: enable 'kccs'

```
hbase(main):031:0> enable 'kccs'  
Took 1.3070 seconds
```

- This command will start enabling the named table
  - Whichever table is disabled, to retrieve back to its previous state we use this command
  - If a table is disabled in the first instance and not deleted or dropped, and if we want to re-use the disabled table then we have to enable it by using this command.
- 

### **show\_filters**

Syntax: show\_filters

```
hbase(main):032:0> show_filters  
DependentColumnFilter  
KeyOnlyFilter  
ColumnCountGetFilter  
SingleColumnValueFilter  
PrefixFilter  
SingleColumnValueExcludeFilter  
FirstKeyOnlyFilter  
ColumnRangeFilter  
ColumnValueFilter  
TimestampsFilter
```

This command displays all the filters present in HBase like ColumnPrefix Filter, TimestampsFilter, PageFilter, FamilyFilter, etc.

---

### **drop**

Syntax: drop <table name>

Example:

```

hbase(main):040:0> disable 'kccs'
Took 1.2760 seconds
hbase(main):041:0> drop 'kccs'
Took 0.2260 seconds
hbase(main):042:0> list
TABLE
0 row(s)
Took 0.0030 seconds
=> []

```

We have to observe below points for drop command

- To delete the table present in HBase, first we have to disable it
  - To drop the table present in HBase, first we have to disable it
  - So either table to drop or delete first the table should be disable using disable command
  - Before execution of this command, it is necessary that we disable table “kccs”
- 

### **drop\_all**

Syntax: drop\_all<"regex">

- This command will drop all the tables matching the given regex
  - Tables have to disable first before executing this command using disable\_all
  - Tables with regex matching expressions are going to drop from HBase
- 

### **alter**

Syntax: alter <tablename>, NAME=><column familyname>, VERSIONS=>5

This command alters the column family schema. To understand what exactly it does, we have explained it here with an example.

**Example:** Altering table kccs, by adding a new column family named ‘teachers’

```

hbase(main):012:0> alter 'kccs', {NAME => 'teachers'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.4870 seconds

```

---

## **Data manipulation commands**

These commands will work on the table related to data manipulations such as putting data into a table, retrieving data from a table and deleting schema, etc.

The commands come under these are

- Count
- Put
- Get
- Delete
- Delete all
- Truncate
- Scan

Let look into these commands usage with an example.

---

### **Count**

Syntax: count <'tablename'>, CACHE =>1000

- The command will retrieve the count of a number of rows in a table. The value returned by this one is the number of rows.
- Current count is shown per every 1000 rows by default.
- Count interval may be optionally specified.
- Default cache size is 10 rows.
- Count command will work fast when it is configured with right Cache.

**Example:**

```
hbase(main):033:0> count 'kccs'
1 row(s)
Took 0.0090 seconds
=> 1
```

---

**Put**

Syntax: put <'tablename'>, <'rowname'>, <'columnvalue'>, <'value'>

This command is used for following things

- It will put a cell ‘value’ at defined or specified table or row or column.
- It will optionally coordinate time stamp.

**Example:**

```
hbase(main):013:0> put 'kccs',1,'students:name','Dhruv'
Took 0.0550 seconds
hbase(main):014:0> put 'kccs',1,'students:roll','59'
Took 0.0030 seconds
hbase(main):015:0> scan 'kccs'
ROW                  COLUMN+CELL
 1                  column=students:name, timestamp=1675597005486, value=Dhruv
 1                  column=students:roll, timestamp=1675597017352, value=59
1 row(s)
Took 0.0220 seconds
hbase(main):016:0> get 'kccs', '1'
COLUMN              CELL
  students:name      timestamp=1675597005486, value=Dhruv
  students:roll      timestamp=1675597017352, value=59
1 row(s)
Took 0.0170 seconds
hbase(main):017:0> put 'kccs',1,'teachers:name','Mr. Naveen Pahuja'
Took 0.0060 seconds
hbase(main):018:0> scan 'kccs'
ROW                  COLUMN+CELL
 1                  column=students:name, timestamp=1675597005486, value=Dhruv
 1                  column=students:roll, timestamp=1675597017352, value=59
 1                  column=teachers:name, timestamp=1675597199115, value=Mr. N
                                              aveen Pahuja
1 row(s)
Took 0.0080 seconds
hbase(main):019:0> scan 'kccs'
ROW                  COLUMN+CELL
 1                  column=students:name, timestamp=1675597005486, value=Dhruv
 1                  column=students:roll, timestamp=1675597017352, value=59
 1                  column=teachers:name, timestamp=1675597199115, value=Mr. N
                                              aveen Pahuja
1 row(s)
Took 0.0080 seconds
```

- hbase> put 'kccs',1,'students:name','Dhruv'

Here we are placing values into table “kccs” under row 1, column family ‘students’ and column ‘name’

- hbase> put 'kccs',1,'students:roll','59'

Here we are placing values into table “kccs” under row 1, column family ‘students’ and column ‘roll’

- hbase> put 'kccs',1,'teachers:name','Mr. Naveen Pahuja'

Here we are placing values into table “kccs” under row 1, column family ‘teachers’ and column ‘name’

---

**Get**

Syntax: get <'tablename'>, <'rowname'>, {< Additional parameters> }

Here <Additional Parameters> include TIMERANGE, TIMESTAMP, VERSIONS and FILTERS.

By using this command, you will get a row or cell contents present in the table. In addition to that you can also add additional parameters to it like TIMESTAMP, TIMERANGE, VERSIONS, FILTERS, etc. to get a particular row or cell content.

**Example:**

```

hbase(main):020:0> get 'kccs', '1', 'teachers'
COLUMN           CELL
  teachers:name   timestamp=1675597199115, value=Mr. Naveen Pahuja
1 row(s)
Took 0.0050 seconds
hbase(main):021:0> get 'kccs', '1', 'students'
COLUMN           CELL
  students:name   timestamp=1675597005486, value=Dhruv
  students:roll    timestamp=1675597017352, value=59
1 row(s)
Took 0.0060 seconds

```

---

### Delete

Syntax: delete <'tablename'>, <'row name'>, <'column name'>

- This command will delete cell value at defined table of row or column.
- Delete must and should match the deleted cells coordinates exactly.
- When scanning, delete cell suppresses older versions of values.

**Example:** hbase> delete 'kccs',1,'students:roll'

```

hbase(main):034:0> delete 'kccs',1,'students:roll'
Took 0.0100 seconds
hbase(main):035:0> get 'kccs', '1', 'students'
COLUMN           CELL
  students:name   timestamp=1675597005486, value=Dhruv
1 row(s)
Took 0.0050 seconds

```

- The above execution will delete column ‘roll’ in the column family ‘students’ from the table ‘kccs’
- 

### deleteall

Syntax: deleteall <'tablename'>, <'rowname'>

- This Command will delete all cells in a given row.
- We can define optionally column names and time stamp to the syntax.
- Optionally we can mention column names in that.

### Truncate

Syntax: truncate <tablename>

After truncate of an hbase table, the schema will present but not the records. This command performs 3 functions; those are listed below

- Disables table if it already presents
- Drops table if it already presents
- Recreates the mentioned table

**Example:**

```

hbase(main):036:0> truncate 'kccs'
Truncating 'kccs' table (it may take a while):
Disabling table...
Truncating table...
Took 2.1190 seconds

```

---

### Scan

Syntax: scan <'tablename'>, {Optional parameters}

This command scans entire table and displays the table contents.

- We can pass several optional specifications to this scan command to get more information about the tables present in the system.
- Scanner specifications may include one or more of the following attributes.
- These are TIMERANGE, FILTER, TIMESTAMP, LIMIT, MAXLENGTH, COLUMNS, CACHE, STARTROW and STOPROW.

**Example:**

```
hbase(main):047:0> scan 'kccs'
ROW                                COLUMN+CELL
 1                               column=students:name, timestamp=1675600898787, value=Dhruv
1 row(s)
Took 0.0050 seconds
```

Name: Krishi Jain

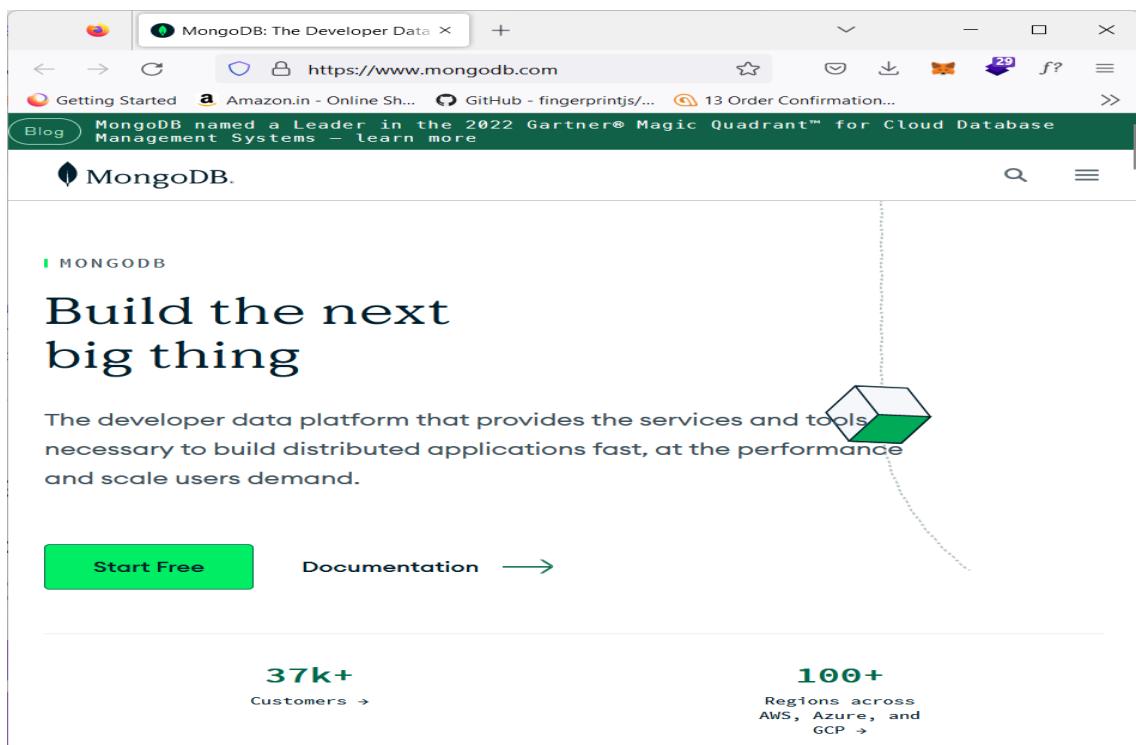
Roll No. KCTBCS026

Date:

16/11/2022

## US-TCS-604 – Big Data Analytics Practical 1 – Installation of MongoDB

Step 1: Navigate to the official MongoDB website i.e <https://www.mongodb.com/>

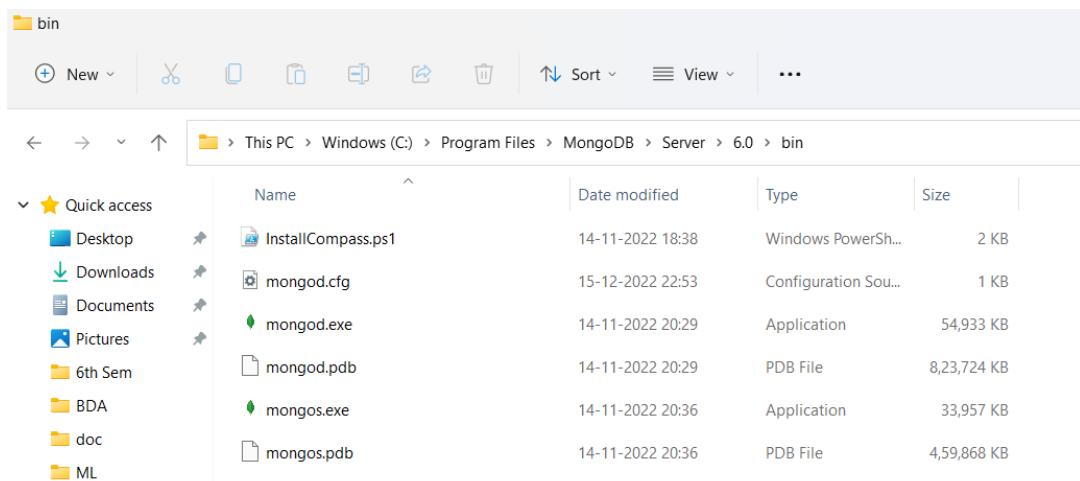


Step 2: Cross-check the Specifications and Download MongoDB

Step 3: Run the setup.exe file and click on next



Step 4: On completing the installation successfully, you will find the software package in your C:\Program Files\MongoDB\Server\6.0\bin



Step 5: Create an Environment Variable

Step 6: For Version 6 Install the mongodb shell zip file

Step 7: Extract it and copy it to C:\Program Files

Step 8: Open the folder

Step 9: Open the mongosh file once and then close it

Step 5: Create an Environment Variable and add a new path C:\Program Files\mongosh-1.6.1-win32-x64\bin

Step 6: Open command prompt and type mongosh

C:\Program Files>cd MongoDB/Server/5.0/bin

C:\Program Files\MongoDB\Server\5.0\bin>mongod.exe --dbpath "C:\Program Files\MongoDB\data"

```
{"t": {"$date": "2022-11-16T11:21:05.067+05:30"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "main", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t": {"$date": "2022-11-16T11:21:05.068+05:30"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "main", "msg": "Initialized wire specification", "attr": {"spec": {"incomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "outgoing": {"minWireVersion": 0, "maxWireVersion": 13}, "isInternalClient": true}}}
{"t": {"$date": "2022-11-16T11:21:05.068+05:30"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer configured during NetworkInterface startup"}
{"t": {"$date": "2022-11-16T11:21:05.069+05:30"}, "s": "I", "c": "NETWORK", "id": 4648602, "ctx": "main", "msg": "Implicit TCP FastOpen in use."}
{"t": {"$date": "2022-11-16T11:21:05.071+05:30"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer configured during NetworkInterface startup"}
{"t": {"$date": "2022-11-16T11:21:05.071+05:30"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "ns": "config.tenantMigrationDonors"}}
{"t": {"$date": "2022-11-16T11:21:05.072+05:30"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "ns": "config.tenantMigrationRecipients"}}
{"t": {"$date": "2022-11-16T11:21:05.072+05:30"}, "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "main", "msg": "Multi threading initialized"}
{"t": {"$date": "2022-11-16T11:21:05.074+05:30"}, "s": "I", "c": "CONTROL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting", "attr": {"pid": 17272, "port": 27017, "dbPath": "C:/Program Files/MongoDB/data", "architecture": "64-bit", "host": "DESKTOP-QAE50JI"}}
{"t": {"$date": "2022-11-16T11:21:05.075+05:30"}, "s": "I", "c": "CONTROL", "id": 23398, "ctx": "initandlisten", "msg": "Target operating system minimum version", "attr": {"targetMinOS": "Windows 7/Windows Server 2008 R2"}}
{"t": {"$date": "2022-11-16T11:21:05.075+05:30"}, "s": "I", "c": "CONTROL", "id": 23403, "ctx": "initandlisten", "msg": "Build"}
```

```

Info","attr":{ "buildInfo":{ "version":"5.0.6","gitVersion":"212a8dbb47f07427dae194a9c75ba
ec1d81d9259","modules":[],"allocator":"tcmalloc","environment":{ "distmod":"windows","di
starch":"x86_64","target_arch":"x86_64"}}}}
{ "t":{ "$date": "2022-11-
16T11:21:05.075+05:30"}, "s": "I", "c": "CONTROL", "id": 51765, "ctx": "initandlisten", "ms
g": "Operating System", "attr": { "os": { "name": "Microsoft Windows 10", "version": "10.0 (build
22000)" }}}
{ "t":{ "$date": "2022-11-
16T11:21:05.076+05:30"}, "s": "I", "c": "CONTROL", "id": 21951, "ctx": "initandlisten", "ms
g": "Options set by command line", "attr": { "options": { "storage": { "dbPath": "C:\\Program
Files\\MongoDB\\data" }}}}
{ "t":{ "$date": "2022-11-
16T11:21:05.077+05:30"}, "s": "E", "c": "CONTROL", "id": 20557, "ctx": "initandlisten", "m
sg": "DBException in initAndListen, terminating", "attr": { "error": "NonExistentPath: Data
directory C:\\Program Files\\MongoDB\\data not found. Create the missing directory or
specify another path using (1) the --dbpath command line option, or (2) by adding the
'storage.dbPath' option in the configuration file." }}
{ "t":{ "$date": "2022-11-16T11:21:05.077+05:30"}, "s": "I", "c": "REPL", "id": 4784900,
"ctx": "initandlisten", "msg": "Stepping down the ReplicationCoordinator for
shutdown", "attr": { "waitForTimeMillis": 15000}}
{ "t":{ "$date": "2022-11-
16T11:21:05.078+05:30"}, "s": "I", "c": "COMMAND", "id": 4784901,
"ctx": "initandlisten", "msg": "Shutting down the MirrorMaestro" }
{ "t":{ "$date": "2022-11-16T11:21:05.078+05:30"}, "s": "I", "c": "SHARDING", "id": 4784902,
"ctx": "initandlisten", "msg": "Shutting down the WaitForMajorityService" }
{ "t":{ "$date": "2022-11-
16T11:21:05.078+05:30"}, "s": "I", "c": "NETWORK", "id": 20562, "ctx": "initandlisten", "m
sg": "Shutdown: going to close listening sockets" }
{ "t":{ "$date": "2022-11-
16T11:21:05.079+05:30"}, "s": "I", "c": "NETWORK", "id": 4784905,
"ctx": "initandlisten", "msg": "Shutting down the global connection pool" }
{ "t":{ "$date": "2022-11-16T11:21:05.079+05:30"}, "s": "I", "c": "CONTROL", "id": 4784906,
"ctx": "initandlisten", "msg": "Shutting down the FlowControlTicketholder" }
{ "t":{ "$date": "2022-11-16T11:21:05.079+05:30"}, "s": "I", "c": "-",
"id": 20520, "ctx": "initandlisten", "msg": "Stopping further Flow Control ticket
acquisitions." }
{ "t":{ "$date": "2022-11-
16T11:21:05.079+05:30"}, "s": "I", "c": "NETWORK", "id": 4784918,
"ctx": "initandlisten", "msg": "Shutting down the ReplicaSetMonitor" }
{ "t":{ "$date": "2022-11-16T11:21:05.080+05:30"}, "s": "I", "c": "SHARDING", "id": 4784921,
"ctx": "initandlisten", "msg": "Shutting down the MigrationUtilExecutor" }
{ "t":{ "$date": "2022-11-
16T11:21:05.081+05:30"}, "s": "I", "c": "ASIO", "id": 22582, "ctx": "MigrationUtil-
TaskExecutor", "msg": "Killing all outstanding egress activity." }
{ "t":{ "$date": "2022-11-
16T11:21:05.081+05:30"}, "s": "I", "c": "COMMAND", "id": 4784923,
"ctx": "initandlisten", "msg": "Shutting down the ServiceEntryPoint" }
{ "t":{ "$date": "2022-11-16T11:21:05.081+05:30"}, "s": "I", "c": "CONTROL", "id": 4784925,
"ctx": "initandlisten", "msg": "Shutting down free monitoring" }

```

```
{"t": {"$date": "2022-11-16T11:21:05.081+05:30"}, "s": "I", "c": "CONTROL", "id": 4784927, "ctx": "initandlisten", "msg": "Shutting down the HealthLog"}  
{"t": {"$date": "2022-11-16T11:21:05.082+05:30"}, "s": "I", "c": "CONTROL", "id": 4784928, "ctx": "initandlisten", "msg": "Shutting down the TTL monitor"}  
{"t": {"$date": "2022-11-16T11:21:05.082+05:30"}, "s": "I", "c": "CONTROL", "id": 4784929, "ctx": "initandlisten", "msg": "Acquiring the global lock for shutdown"}  
{"t": {"$date": "2022-11-16T11:21:05.082+05:30"}, "s": "I", "c": "-", "id": 4784931, "ctx": "initandlisten", "msg": "Dropping the scope cache for shutdown"}  
{"t": {"$date": "2022-11-16T11:21:05.082+05:30"}, "s": "I", "c": "FTDC", "id": 4784926, "ctx": "initandlisten", "msg": "Shutting down full-time data capture"}  
{"t": {"$date": "2022-11-16T11:21:05.083+05:30"}, "s": "I", "c": "CONTROL", "id": 20565, "ctx": "initandlisten", "msg": "Now exiting"}  
{"t": {"$date": "2022-11-16T11:21:05.083+05:30"}, "s": "I", "c": "CONTROL", "id": 23138, "ctx": "initandlisten", "msg": "Shutting down", "attr": {"exitCode": 100}}
```

C:\Program Files\MongoDB\Server\5.0\bin>mongo.exe

MongoDB shell version v5.0.6

connecting to:

mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("293bdf40-c40d-4db3-82d3-c0e572ce653a") }

MongoDB server version: 5.0.6

=====

Warning: the "mongo" shell has been superseded by "mongosh", which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in an upcoming release.

For installation instructions, see

<https://docs.mongodb.com/mongodb-shell/install/>

=====

Welcome to the MongoDB shell.

For interactive help, type "help".

For more comprehensive documentation, see

<https://docs.mongodb.com/>

Questions? Try the MongoDB Developer Community Forums

<https://community.mongodb.com>

---

The server generated these startup warnings when booting:

2022-11-13T10:22:28.319+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

---

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command:

```
db.disableFreeMonitoring()
```

---

```
> show dbs
admin      0.000GB
config     0.000GB
eshajain_08 0.000GB
local      0.000GB
> db
test
```

## US-TCS-604 – Big Data Analytics Practical 2 – CRUD Operations on data in MongoDB

### **1) Display all databases.**

```
test> show dbs
admin    40.00 KiB
cars     176.00 KiB
config   72.00 KiB
letssee  40.00 KiB
local    40.00 KiB
mydb    40.00 KiB
```

### **2) Create a new collection ‘kcc’.**

```
test> use kcc
switched to db kcc
```

### **3) Insert a document with student roll number, name, semester and SGPA.**

```
kcc> db.kcc.insert({rollno:1, sname:'pranav', sem:5, sgpa:9.50})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or
bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63ccbdf04ccbffd526d1b74") }
}
```

### **4) Display all documents in the collection.**

```
kcc> db.kcc.find()
[
  {
    _id: ObjectId("63ccbdf04ccbffd526d1b74"),
    rollno: 1,
    sname: 'pranav',
    sem: 5,
```

```
        sgpa: 9.5
    }
]
```

## 5) Insert more documents into the collection.

### Using insertOne:

```
kcc> db.kcc.insertOne({rollno:2, sname:'krishi', sem:5, sgpa:9.3})
{
  acknowledged: true,
  insertedId: ObjectId("63ccbe3504ccbffd526d1b75")
}
```

### Using insertMany:

```
kcc> db.kcc.insertMany([{rollno:3, sname:'khushi', sem:6, sgpa:8.7}, {rollno:4, sname:'yash', sem:5, sgpa:7.75}, {rollno:5, sname:'shruti', sem:6, sgpa:9.0}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63ccbeb504ccbffd526d1b76"),
    '1': ObjectId("63ccbeb504ccbffd526d1b77"),
    '2': ObjectId("63ccbeb504ccbffd526d1b78")
  }
}
```

### Displaying with beautification:

```
kcc> db.kcc.find().pretty()
[
  {
    _id: ObjectId("63ccbdf04ccbffd526d1b74"),
    rollno: 1,
    sname: 'pranav',
    sem: 5,
    sgpa: 9.5
  },
  {
    _id: ObjectId("63ccbe3504ccbffd526d1b75"),
    rollno: 2,
    sname: 'krishi',
    sem: 5,
    sgpa: 9.3
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b76"),
    rollno: 3,
    sname: 'khushi',
    sem: 6,
    sgpa: 8.7
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b77"),
    rollno: 4,
```

```

rollno: 4,
sname: 'yash',
sem: 5,
sgpa: 7.75
},
{
  _id: ObjectId("63ccbeb504ccbffd526d1b78"),
  rollno: 5,
  sname: 'shruti',
  sem: 6,
  sgpa: 9
}
]

```

**6) Display only the first three documents of the collection.**

```

kcc> db.kcc.find().limit(3)
[
  {
    _id: ObjectId("63ccbd04ccbffd526d1b74"),
    rollno: 1,
    sname: 'pranav',
    sem: 5,
    sgpa: 9.5
  },
  {
    _id: ObjectId("63ccbe3504ccbffd526d1b75"),
    rollno: 2,
    sname: 'krishi',
    sem: 5,
    sgpa: 9.3
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b76"),
    rollno: 3,
    sname: 'khushi',
    sem: 6,
    sgpa: 8.7
  }
]
```

**7) Display all but the first three documents of the collection.**

```

kcc> db.kcc.find().skip(3)
[
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b77"),
    rollno: 4,
    sname: 'yash',
    sem: 5,
    sgpa: 7.75
  },

```

```
{
  _id: ObjectId("63ccbeb504ccbffd526d1b78"),
  rollno: 5,
  sname: 'shruti',
  sem: 6,
  sgpa: 9
}
]
```

**8) Display documents 3-5 of the collection.**

```
kcc> db.kcc.find().skip(2).limit(3)
[
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b76"),
    rollno: 3,
    sname: 'khushi',
    sem: 6,
    sgpa: 8.7
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b77"),
    rollno: 4,
    sname: 'yash',
    sem: 5,
    sgpa: 7.75
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b78"),
    rollno: 5,
    sname: 'shruti',
    sem: 6,
    sgpa: 9
  }
]
```

**9) Display the document having rollno 2.**

```
kcc> db.kcc.find({rollno:2})
[
  {
    _id: ObjectId("63ccbe3504ccbffd526d1b75"),
    rollno: 2,
    sname: 'krishi',
    sem: 5,
    sgpa: 9.3
  }
]
```

**10) Display document having roll no 4 and sem 5.**

```
kcc> db.kcc.find({rollno:4}, {sem:5})
```

```
[ { _id: ObjectId("63ccbeb504ccbffd526d1b77"), sem: 5 } ]
```

**11) Update document where rollno is 2 and change sgpa to 9.5.**

```
kcc> db.kcc.update({ rollno: 2 }, { $set: { sgpa: 9.5 } })  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

**12) Update document/s where sgpa is 9.5 and set their grade to 'A+'.**

```
kcc> db.kcc.updateMany({ sgpa:9.5 }, { $set: { grade:'A+' } })  
{
```

```
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 2,  
  modifiedCount: 2,  
  upsertedCount: 0  
}
```

```
kcc> db.kcc.find()
```

```
[  
  {  
    _id: ObjectId("63ccbdf04ccbffd526d1b74"),  
    rollno: 1,  
    sname: 'pranav',  
    sem: 5,  
    sgpa: 9.5,  
    grade: 'A+'  
  },  
  {  
    _id: ObjectId("63ccbe3504ccbffd526d1b75"),  
    rollno: 2,  
    sname: 'krishi',  
    sem: 5,  
    sgpa: 9.5,  
    grade: 'A+'  
  },  
  {  
    _id: ObjectId("63ccbeb504ccbffd526d1b76"),  
    rollno: 3,  
    sname: 'khushi',  
    sem: 6,  
    sgpa: 8.7  
  },  
  {  
    _id: ObjectId("63ccbeb504ccbffd526d1b77"),  
    rollno: 4,  
    sname: 'yash',  
  }
```

```

    sem: 5,
    sgpa: 7.75
},
{
  _id: ObjectId("63ccbeb504ccbffd526d1b78"),
  rollno: 5,
  sname: 'shruti',
  sem: 6,
  sgpa: 9
}
]

```

**13) Count the number of documents in the collection.**

```

kcc> db.kcc.find().count()
(node:67271) [MONGODB DRIVER] Warning: cursor.count is deprecated and will be
removed in the next major version, please use `collection.estimatedDocumentCount` or
`collection.countDocuments` instead
(Use `node --trace-warnings ...` to show where the warning was created)
5

```

```

kcc> db.kcc.countDocuments()
5

```

**14) Update documents where SGPA is more than 8 and set their semester to 4.**

```

kcc> db.kcc.updateMany({ sgpa:{$gt:8} }, { $set:{sem:4} })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}

```

```

kcc> db.kcc.find()
[
  {
    _id: ObjectId("63ccbdf04ccbffd526d1b74"),
    rollno: 1,
    sname: 'pranav',
    sem: 4,
    sgpa: 9.5,
    grade: 'A+'
  },
  {
    _id: ObjectId("63ccbe3504ccbffd526d1b75"),
    rollno: 2,
    sname: 'krishi',
    sem: 4,
    sgpa: 9.5,
    grade: 'A+'
  },
]

```

```
{
  _id: ObjectId("63ccbeb504ccbffd526d1b76"),
  rollno: 3,
  sname: 'khushi',
  sem: 4,
  sgpa: 8.7
},
{
  _id: ObjectId("63ccbeb504ccbffd526d1b77"),
  rollno: 4,
  sname: 'yash',
  sem: 5,
  sgpa: 7.75
},
{
  _id: ObjectId("63ccbeb504ccbffd526d1b78"),
  rollno: 5,
  sname: 'shruti',
  sem: 4,
  sgpa: 9
}
]
```

### **15) Delete document where rollno is 2.**

```
kcc> db.kcc.deleteOne({rollno:2})
{ acknowledged: true, deletedCount: 1 }
```

```
kcc> db.kcc.find()
[
  {
    _id: ObjectId("63ccbdf04ccbffd526d1b74"),
    rollno: 1,
    sname: 'pranav',
    sem: 4,
    sgpa: 9.5,
    grade: 'A+'
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b76"),
    rollno: 3,
    sname: 'khushi',
    sem: 4,
    sgpa: 8.7
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b77"),
    rollno: 4,
    sname: 'yash',
    sem: 5,
    sgpa: 7.75
  }
]
```

```

},
{
  _id: ObjectId("63ccbeb504ccbffd526d1b78"),
  rollno: 5,
  sname: 'shruti',
  sem: 4,
  sgpa: 9
}
]

```

**16) Display documents sorted in ascending order of student name.**

```

kcc> db.kcc.find().sort({ sname:1 })
[
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b76"),
    rollno: 3,
    sname: 'khushi',
    sem: 4,
    sgpa: 8.7
  },
  {
    _id: ObjectId("63ccbdf04ccbffd526d1b74"),
    rollno: 1,
    sname: 'pranav',
    sem: 4,
    sgpa: 9.5,
    grade: 'A+'
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b78"),
    rollno: 5,
    sname: 'shruti',
    sem: 4,
    sgpa: 9
  },
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b77"),
    rollno: 4,
    sname: 'yash',
    sem: 5,
    sgpa: 7.75
  }
]

```

**17) Display documents sorted in descending order of student name.**

```

kcc> db.kcc.find().sort({ sname:-1 })
[
  {
    _id: ObjectId("63ccbeb504ccbffd526d1b77"),
    rollno: 4,
    sname: 'yash',

```

```

    sem: 5,
    sgpa: 7.75
},
{
  _id: ObjectId("63ccbeb504ccbffd526d1b78"),
  rollno: 5,
  sname: 'shruti',
  sem: 4,
  sgpa: 9
},
{
  _id: ObjectId("63ccbdfd04ccbffd526d1b74"),
  rollno: 1,
  sname: 'pranav',
  sem: 4,
  sgpa: 9.5,
  grade: 'A+'
},
{
  _id: ObjectId("63ccbeb504ccbffd526d1b76"),
  rollno: 3,
  sname: 'khushi',
  sem: 4,
  sgpa: 8.7
}
]

```

## US-TCS-604 – Big Data Analytics Practical 4 – Aggregation Pipeline 1

**Create a new database with the name inventory that contains the documents of the following type:**

**itemname, quantity, size - (height, width, unit of measure), quality (in range A-E), instock - (warehouse, quantity).**

```
> use inventory
switched to db inventory
```

```
> db.inventory.insertMany([
  {itemname: "Pen", quantity: 37, size: {height: 10, width: 2, unit: "cm"}, quality: 'B', instock: [{warehouse: "w2", quantity: 1}, {warehouse: "w3", quantity: 16}]},
  ... {itemname: "Marker", quantity: 58, size: {height: 10, width: 3.5, unit: "cm"}, quality: 'A', instock: [{warehouse: "w1", quantity: 20}, {warehouse: "w2", quantity: 7}, {warehouse: "w3", quantity: 31}]},
  ... {itemname: "Notebook", quantity: 350, size: {height: 12, width: 0.7, unit: "inch"}, quality: 'C', instock: [{warehouse: "w1", quantity: 170}, {warehouse: "w3", quantity: 70}]},
  ... {itemname: "Ruler", quantity: 123, size: {height: 12, width: 0.1, unit: "inch"}, quality: 'E', instock: [{warehouse: "w2", quantity: 46}, {warehouse: "w3", quantity: 54}]}
])
```

```

... {itemname: "Eraser", quantity: 76, size: {height: 5, width:3.5, unit: "cm"}, quality: 'D',
instock: [{warehouse: "w1", quantity: 40}, {warehouse: "w2", quantity: 36}]},
... {itemname: "Cutter", quantity: 68, size: {height: 8, width:2, unit: "cm"}, quality: 'A',
instock: [{warehouse: "w1", quantity: 21}, {warehouse: "w2", quantity: 26}, {warehouse:
"w3", quantity: 21}]},
... {itemname: "Writing Pad", quantity: 107, size: {height: 15, width:0.1, unit: "inch"}, quality: 'B', instock: [{warehouse: "w1", quantity: 35}, {warehouse: "w2", quantity: 27}, {warehouse: "w3", quantity: 45}]},
... {itemname: "Pencil", quantity: 590, size: {height: 12, width:3.5, unit: "cm"}, quality: 'C', instock: [{warehouse: "w2", quantity: 210}, {warehouse: "w3", quantity: 100}]},
... {itemname: "Tape", quantity: 360, size: {height: 6, width:1, unit: "cm"}, quality: 'D', instock: [{warehouse: "w1", quantity: 190}, {warehouse: "w2", quantity: 79}, {warehouse: "w3", quantity: 91}]},
... {itemname: "Boardpins", quantity: 75, size: {height: 2, width:1.5, unit: "cm"}, quality: 'E', instock: [{warehouse: "w1", quantity: 14}, {warehouse: "w3", quantity: 54}]})
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("639add7175bb6a14b0603311"),
        ObjectId("639add7175bb6a14b0603312"),
        ObjectId("639add7175bb6a14b0603313"),
        ObjectId("639add7175bb6a14b0603314"),
        ObjectId("639add7175bb6a14b0603315"),
        ObjectId("639add7175bb6a14b0603316"),
        ObjectId("639add7175bb6a14b0603317"),
        ObjectId("639add7175bb6a14b0603318"),
        ObjectId("639add7175bb6a14b0603319"),
        ObjectId("639add7175bb6a14b060331a")
    ]
}

```

```

> db.inventory.find().pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 1
        },
        {
            "warehouse" : "w3",
            "quantity" : 16
        }
    ]
}

```

```

        }
    ]
{
  "_id" : ObjectId("639add7175bb6a14b0603312"),
  "itemname" : "Marker",
  "quantity" : 58,
  "size" : {
    "height" : 10,
    "width" : 3.5,
    "unit" : "cm"
  },
  "quality" : "A",
  "instock" : [
    {
      "warehouse" : "w1",
      "quantity" : 20
    },
    {
      "warehouse" : "w2",
      "quantity" : 7
    },
    {
      "warehouse" : "w3",
      "quantity" : 31
    }
  ]
}

```

**(Cont.)**

### 1. Find all inventory item names with quantity less than 100.

```

> db.inventory.find({quantity:{$lt:100}}).pretty()
{
  "_id" : ObjectId("639add7175bb6a14b0603311"),
  "itemname" : "Pen",
  "quantity" : 37,
  "size" : {
    "height" : 10,
    "width" : 2,
    "unit" : "cm"
  },
  "quality" : "B",
  "instock" : [
    {
      "warehouse" : "w2",
      "quantity" : 1
    },
    {

```

```

        "warehouse" : "w3",
        "quantity" : 16
    }
]
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "quantity" : 58,
    "size" : {
        "height" : 10,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 20
        },
        {
            "warehouse" : "w2",
            "quantity" : 7
        },
        {
            "warehouse" : "w3",
            "quantity" : 31
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603315"),
    "itemname" : "Eraser",
    "quantity" : 76,
    "size" : {
        "height" : 5,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 40
        },
        {
            "warehouse" : "w2",
            "quantity" : 36
        }
    ]
}

```

```

}
{
    "_id" : ObjectId("639add7175bb6a14b0603316"),
    "itemname" : "Cutter",
    "quantity" : 68,
    "size" : {
        "height" : 8,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 21
        },
        {
            "warehouse" : "w2",
            "quantity" : 26
        },
        {
            "warehouse" : "w3",
            "quantity" : 21
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),
    "itemname" : "Boardpins",
    "quantity" : 75,
    "size" : {
        "height" : 2,
        "width" : 1.5,
        "unit" : "cm"
    },
    "quality" : "E",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 14
        },
        {
            "warehouse" : "w3",
            "quantity" : 54
        }
    ]
}

```

2. Find all inventory item names with quality status as A and display their item name and quality only. (Projection)

```
> db.inventory.find({quality:"A"}, {_id:0, itemname:1, quality:1})
{ "itemname" : "Marker", "quality" : "A" }
{ "itemname" : "Cutter", "quality" : "A" }
```

**3. Find all inventory item names having instock warehouse as w1 and display their instock quantity.**

```
> db.inventory.find({ "instock.warehouse": "w1"}, {_id:0, itemname:1, instock:1}).pretty()
{
  "itemname" : "Marker",
  "instock" : [
    {
      {
        "warehouse" : "w1",
        "quantity" : 20
      },
      {
        "warehouse" : "w2",
        "quantity" : 7
      },
      {
        "warehouse" : "w3",
        "quantity" : 31
      }
    ]
  }
{
  "itemname" : "Notebook",
  "instock" : [
    {
      {
        "warehouse" : "w1",
        "quantity" : 170
      },
      {
        "warehouse" : "w3",
        "quantity" : 70
      }
    ]
  }
{
  "itemname" : "Eraser",
  "instock" : [
    {
      {
        "warehouse" : "w1",
        "quantity" : 40
      },
      {
        "warehouse" : "w2",
        "quantity" : 36
      }
    ]
  }
}
```

```

        ]
    }
{
    "itemname" : "Cutter",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 21
        },
        {
            "warehouse" : "w2",
            "quantity" : 26
        },
        {
            "warehouse" : "w3",
            "quantity" : 21
        }
    ]
}
{
    "itemname" : "Writing Pad",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 35
        },
        {
            "warehouse" : "w2",
            "quantity" : 27
        },
        {
            "warehouse" : "w3",
            "quantity" : 45
        }
    ]
}
{
    "itemname" : "Tape",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 190
        },
        {
            "warehouse" : "w2",
            "quantity" : 79
        },
        {
            "warehouse" : "w3",
            "quantity" : 91
        }
    ]
}

```

```

        }
    ]
}
{
    "itemname" : "Boardpins",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 14
        },
        {
            "warehouse" : "w3",
            "quantity" : 54
        }
    ]
}

```

**4. Find all inventory item names having quality status as B or E and display their item name and quality status.**

```
> db.inventory.find({$or:[{quality:"B"}, {quality:"E"}]}, {_id:0, itemname:1, quality:1,}).pretty()
{
    "itemname" : "Pen", "quality" : "B" 
    "itemname" : "Ruler", "quality" : "E" 
    "itemname" : "Writing Pad", "quality" : "B" 
    "itemname" : "Boardpins", "quality" : "E" }
```

**5. Find all inventory item names having instock quantity between 20 and 40.**

```
> db.inventory.find({$and:[{quantity:{$gt:20}}, {quantity:{$lt:40}}]}).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 1
        },
        {
            "warehouse" : "w3",
            "quantity" : 16
        }
    ]
}
```

**6. Find all inventory item names starting with ‘P’ display item name and quantity except id.**

```
> db.inventory.find({itemname:{$regex:/^P/}}, {_id:0, itemname:1, quantity:1}).pretty()
{ "itemname" : "Pen", "quantity" : 37 }
{ "itemname" : "Pencil", "quantity" : 590 }
```

**7. Display all inventory item names having instock warehouse as w2 and w3 (itemname, instock warehouse except id).**

```
> db.inventory.find({ "instock.warehouse":{$in:["w2", "w3"]}}, {_id:0, itemname:1, "instock.warehouse":1}).pretty()
{
  "itemname" : "Pen",
  "instock" : [
    {
      "warehouse" : "w2"
    },
    {
      "warehouse" : "w3"
    }
  ]
}
{
  "itemname" : "Marker",
  "instock" : [
    {
      "warehouse" : "w1"
    },
    {
      "warehouse" : "w2"
    },
    {
      "warehouse" : "w3"
    }
  ]
}
{
  "itemname" : "Notebook",
  "instock" : [
    {
      "warehouse" : "w1"
    },
    {
      "warehouse" : "w3"
    }
  ]
}
```

```

    "itemname" : "Ruler",
    "instock" : [
        {
            "warehouse" : "w2"
        },
        {
            "warehouse" : "w3"
        }
    ]
}

{
    "itemname" : "Eraser",
    "instock" : [
        {
            "warehouse" : "w1"
        },
        {
            "warehouse" : "w2"
        }
    ]
}

{
    "itemname" : "Cutter",
    "instock" : [
        {
            "warehouse" : "w1"
        },
        {
            "warehouse" : "w2"
        },
        {
            "warehouse" : "w3"
        }
    ]
}

{
    "itemname" : "Writing Pad",
    "instock" : [
        {
            "warehouse" : "w1"
        },
        {
            "warehouse" : "w2"
        },
        {
            "warehouse" : "w3"
        }
    ]
}

```

```

    "itemname" : "Pencil",
    "instock" : [
        {
            "warehouse" : "w2"
        },
        {
            "warehouse" : "w3"
        }
    ]
}
{
    "itemname" : "Tape",
    "instock" : [
        {
            "warehouse" : "w1"
        },
        {
            "warehouse" : "w2"
        },
        {
            "warehouse" : "w3"
        }
    ]
}
{
    "itemname" : "Boardpins",
    "instock" : [
        {
            "warehouse" : "w1"
        },
        {
            "warehouse" : "w3"
        }
    ]
}

```

**8. Display all inventory item names not having instock warehouse as w2 and w3  
(itemname, instock warehouse except id).**

```

> db.inventory.find({ "instock.warehouse":{$nin:["w2", "w3"]}}, {_id:0, itemname:1,
  "instock.warehouse":1}).pretty()
> db.inventory.insertOne({itemname:"Crayons", quantity:40, size:{height:4, width:0.1,
  unit:"inch"}, quality:"C", instock:[{warehouse:"w1", quantity: 40}]})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("639c1d4cf2a551570c47043e")
}
> db.inventory.find({ "instock.warehouse":{$nin:["w2", "w3"]}}, {_id:0, itemname:1,
  "instock.warehouse":1}).pretty()
{ "itemname" : "Crayons", "instock" : [ { "warehouse" : "w1" } ] }

```

**9. Display all inventory item names not having quantity greater than equal to 50.**

```
> db.inventory.find({quantity:{$not:{$gte:50}}}).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 1
        },
        {
            "warehouse" : "w3",
            "quantity" : 16
        }
    ]
}
{
    "_id" : ObjectId("639c1d4cf2a551570c47043e"),
    "itemname" : "Crayons",
    "quantity" : 40,
    "size" : {
        "height" : 4,
        "width" : 0.1,
        "unit" : "inch"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 40
        }
    ]
}
```

**10. Find all inventory item names having quality status as E and quantity greater than 100.**

```
> db.inventory.find({$and:[{quality:"E"}, {quantity:{$gte:100}}]}).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603314"),
    "itemname" : "Ruler",
```

```

"quantity" : 123,
"size" : {
    "height" : 12,
    "width" : 0.1,
    "unit" : "inch"
},
"quality" : "E",
"instock" : [
    {
        "warehouse" : "w2",
        "quantity" : 46
    },
    {
        "warehouse" : "w3",
        "quantity" : 54
    }
]
}

```

**11. Find the count of inventory items names having size in cm.**

```

> db.inventory.find({ "size.unit":"cm" }).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 1
        },
        {
            "warehouse" : "w3",
            "quantity" : 16
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "quantity" : 58,
    "size" : {
        "height" : 10,
        "width" : 3.5,

```

```

        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 20
        },
        {
            "warehouse" : "w2",
            "quantity" : 7
        },
        {
            "warehouse" : "w3",
            "quantity" : 31
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603315"),
    "itemname" : "Eraser",
    "quantity" : 76,
    "size" : {
        "height" : 5,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 40
        },
        {
            "warehouse" : "w2",
            "quantity" : 36
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603316"),
    "itemname" : "Cutter",
    "quantity" : 68,
    "size" : {
        "height" : 8,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [

```

```

        {
            "warehouse" : "w1",
            "quantity" : 21
        },
        {
            "warehouse" : "w2",
            "quantity" : 26
        },
        {
            "warehouse" : "w3",
            "quantity" : 21
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603318"),
    "itemname" : "Pencil",
    "quantity" : 590,
    "size" : {
        "height" : 12,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 210
        },
        {
            "warehouse" : "w3",
            "quantity" : 100
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603319"),
    "itemname" : "Tape",
    "quantity" : 360,
    "size" : {
        "height" : 6,
        "width" : 1,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 190
        },

```

```

        {
            "warehouse" : "w2",
            "quantity" : 79
        },
        {
            "warehouse" : "w3",
            "quantity" : 91
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),
    "itemname" : "Boardpins",
    "quantity" : 75,
    "size" : {
        "height" : 2,
        "width" : 1.5,
        "unit" : "cm"
    },
    "quality" : "E",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 14
        },
        {
            "warehouse" : "w3",
            "quantity" : 54
        }
    ]
}
> db.inventory.count({"size.unit":"cm"})
7

```

## 12. Display first three documents from inventory database.

```

> db.inventory.find().limit(3).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",

```

```

        "quantity" : 1
    },
    {
        "warehouse" : "w3",
        "quantity" : 16
    }
]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "quantity" : 58,
    "size" : {
        "height" : 10,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 20
        },
        {
            "warehouse" : "w2",
            "quantity" : 7
        },
        {
            "warehouse" : "w3",
            "quantity" : 31
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603313"),
    "itemname" : "Notebook",
    "quantity" : 350,
    "size" : {
        "height" : 12,
        "width" : 0.7,
        "unit" : "inch"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 170
        },
        {
            "warehouse" : "w3",

```

```

        "quantity" : 70
    }
]
}

```

**Aggregation functions:**

**13. Display the total quantity of all items.**

```
> db.inventory.aggregate({$group:{_id:null, "Total quantity":{$sum:"$quantity"}}})
{ "_id" : null, "Total quantity:" : 1884 }
```

**14. Display the average quantity of items present in each document within the collection.**

```
> db.inventory.aggregate({$group:{_id:null, "Average quantity":{$avg:"$quantity"}}})
{ "_id" : null, "Average quantity:" : 171.27272727272728 }
```

**15. Display which item has the lowest quantity within the collection.**

```
> db.inventory.aggregate({$group:{_id:null, "Item with lowest quantity":{$min:"$quantity"}}})
{ "_id" : null, "Item with lowest quantity:" : 37 }
```

**16. Display which item has the highest quantity within the collection.**

```
> db.inventory.aggregate({$group:{_id:null, "Item with highest quantity":{$max:"$quantity"}}})
{ "_id" : null, "Item with highest quantity:" : 590 }
```

**17. Find all items having quantity greater than or equal to 50.**

```
> db.inventory.find({quantity:{$gte:50}}).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "quantity" : 58,
    "size" : {
        "height" : 10,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 20
        },
        {

```

```

        "warehouse" : "w2",
        "quantity" : 7
    },
    {
        "warehouse" : "w3",
        "quantity" : 31
    }
]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603313"),
    "itemname" : "Notebook",
    "quantity" : 350,
    "size" : {
        "height" : 12,
        "width" : 0.7,
        "unit" : "inch"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 170
        },
        {
            "warehouse" : "w3",
            "quantity" : 70
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603314"),
    "itemname" : "Ruler",
    "quantity" : 123,
    "size" : {
        "height" : 12,
        "width" : 0.1,
        "unit" : "inch"
    },
    "quality" : "E",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 46
        },
        {
            "warehouse" : "w3",
            "quantity" : 54
        }
    ]
}

```

```

}
{
    "_id" : ObjectId("639add7175bb6a14b0603315"),
    "itemname" : "Eraser",
    "quantity" : 76,
    "size" : {
        "height" : 5,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 40
        },
        {
            "warehouse" : "w2",
            "quantity" : 36
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603316"),
    "itemname" : "Cutter",
    "quantity" : 68,
    "size" : {
        "height" : 8,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 21
        },
        {
            "warehouse" : "w2",
            "quantity" : 26
        },
        {
            "warehouse" : "w3",
            "quantity" : 21
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603317"),
    "itemname" : "Writing Pad",

```

```

"quantity" : 107,
"size" : {
    "height" : 15,
    "width" : 0.1,
    "unit" : "inch"
},
"quality" : "B",
"instock" : [
    {
        "warehouse" : "w1",
        "quantity" : 35
    },
    {
        "warehouse" : "w2",
        "quantity" : 27
    },
    {
        "warehouse" : "w3",
        "quantity" : 45
    }
]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603318"),
    "itemname" : "Pencil",
    "quantity" : 590,
    "size" : {
        "height" : 12,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 210
        },
        {
            "warehouse" : "w3",
            "quantity" : 100
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603319"),
    "itemname" : "Tape",
    "quantity" : 360,
    "size" : {
        "height" : 6,
        "width" : 1,
    }
}

```

```

        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 190
        },
        {
            "warehouse" : "w2",
            "quantity" : 79
        },
        {
            "warehouse" : "w3",
            "quantity" : 91
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),
    "itemname" : "Boardpins",
    "quantity" : 75,
    "size" : {
        "height" : 2,
        "width" : 1.5,
        "unit" : "cm"
    },
    "quality" : "E",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 14
        },
        {
            "warehouse" : "w3",
            "quantity" : 54
        }
    ]
}
> db.inventory.find({quantity:{$gte:50}}, {_id:0, itemname:1, quantity:1}).pretty()
{
    "itemname" : "Marker", "quantity" : 58
}
{
    "itemname" : "Notebook", "quantity" : 350
}
{
    "itemname" : "Ruler", "quantity" : 123
}
{
    "itemname" : "Eraser", "quantity" : 76
}
{
    "itemname" : "Cutter", "quantity" : 68
}
{
    "itemname" : "Writing Pad", "quantity" : 107
}
{
    "itemname" : "Pencil", "quantity" : 590
}
{
    "itemname" : "Tape", "quantity" : 360
}
{
    "itemname" : "Boardpins", "quantity" : 75
}
```

```
> db.inventory.aggregate([{$match:{quantity:{$gte:50}}}]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "quantity" : 58,
    "size" : {
        "height" : 10,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 20
        },
        {
            "warehouse" : "w2",
            "quantity" : 7
        },
        {
            "warehouse" : "w3",
            "quantity" : 31
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603313"),
    "itemname" : "Notebook",
    "quantity" : 350,
    "size" : {
        "height" : 12,
        "width" : 0.7,
        "unit" : "inch"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 170
        },
        {
            "warehouse" : "w3",
            "quantity" : 70
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603314"),
    "itemname" : "Ruler",

```

```

"quantity" : 123,
"size" : {
    "height" : 12,
    "width" : 0.1,
    "unit" : "inch"
},
"quality" : "E",
"instock" : [
    {
        "warehouse" : "w2",
        "quantity" : 46
    },
    {
        "warehouse" : "w3",
        "quantity" : 54
    }
]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603315"),
    "itemname" : "Eraser",
    "quantity" : 76,
    "size" : {
        "height" : 5,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 40
        },
        {
            "warehouse" : "w2",
            "quantity" : 36
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603316"),
    "itemname" : "Cutter",
    "quantity" : 68,
    "size" : {
        "height" : 8,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [

```

```

        {
            "warehouse" : "w1",
            "quantity" : 21
        },
        {
            "warehouse" : "w2",
            "quantity" : 26
        },
        {
            "warehouse" : "w3",
            "quantity" : 21
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603317"),
    "itemname" : "Writing Pad",
    "quantity" : 107,
    "size" : {
        "height" : 15,
        "width" : 0.1,
        "unit" : "inch"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 35
        },
        {
            "warehouse" : "w2",
            "quantity" : 27
        },
        {
            "warehouse" : "w3",
            "quantity" : 45
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603318"),
    "itemname" : "Pencil",
    "quantity" : 590,
    "size" : {
        "height" : 12,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "C",
    "instock" : [

```

```

        {
            "warehouse" : "w2",
            "quantity" : 210
        },
        {
            "warehouse" : "w3",
            "quantity" : 100
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603319"),
    "itemname" : "Tape",
    "quantity" : 360,
    "size" : {
        "height" : 6,
        "width" : 1,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 190
        },
        {
            "warehouse" : "w2",
            "quantity" : 79
        },
        {
            "warehouse" : "w3",
            "quantity" : 91
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),
    "itemname" : "Boardpins",
    "quantity" : 75,
    "size" : {
        "height" : 2,
        "width" : 1.5,
        "unit" : "cm"
    },
    "quality" : "E",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 14
        },

```

```

        {
            "warehouse" : "w3",
            "quantity" : 54
        }
    ]
}

```

**18. Display all item names having unit of measure as cm.**

```

> db.inventory.aggregate([{$match:{ "size.unit":"cm" }}]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 1
        },
        {
            "warehouse" : "w3",
            "quantity" : 16
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "quantity" : 58,
    "size" : {
        "height" : 10,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 20
        },
        {
            "warehouse" : "w2",
            "quantity" : 7
        },
        {
            "warehouse" : "w3",
            "quantity" : 1
        }
    ]
}

```

```

        {
            "warehouse" : "w3",
            "quantity" : 31
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603315"),
    "itemname" : "Eraser",
    "quantity" : 76,
    "size" : {
        "height" : 5,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 40
        },
        {
            "warehouse" : "w2",
            "quantity" : 36
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603316"),
    "itemname" : "Cutter",
    "quantity" : 68,
    "size" : {
        "height" : 8,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 21
        },
        {
            "warehouse" : "w2",
            "quantity" : 26
        },
        {
            "warehouse" : "w3",
            "quantity" : 21
        }
    ]
}

```

```

        ],
    }
{
    "_id" : ObjectId("639add7175bb6a14b0603318"),
    "itemname" : "Pencil",
    "quantity" : 590,
    "size" : {
        "height" : 12,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 210
        },
        {
            "warehouse" : "w3",
            "quantity" : 100
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603319"),
    "itemname" : "Tape",
    "quantity" : 360,
    "size" : {
        "height" : 6,
        "width" : 1,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 190
        },
        {
            "warehouse" : "w2",
            "quantity" : 79
        },
        {
            "warehouse" : "w3",
            "quantity" : 91
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),

```

```

"itemname" : "Boardpins",
"quantity" : 75,
"size" : {
    "height" : 2,
    "width" : 1.5,
    "unit" : "cm"
},
"quality" : "E",
"instock" : [
    {
        "warehouse" : "w1",
        "quantity" : 14
    },
    {
        "warehouse" : "w3",
        "quantity" : 54
    }
]
}

```

**19. Display only itemname and unit of items having unit as cm.**

```

> db.inventory.aggregate([{$match:{"size.unit":"cm"}}, {$project:{itemname:1,
"size.unit":"cm" }}]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603315"),
    "itemname" : "Eraser",
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603316"),
    "itemname" : "Cutter",
    "size" : {
        "unit" : "cm"
    }
}

```

```

}
{
    "_id" : ObjectId("639add7175bb6a14b0603318"),
    "itemname" : "Pencil",
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603319"),
    "itemname" : "Tape",
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),
    "itemname" : "Boardpins",
    "size" : {
        "unit" : "cm"
    }
}

```

## 20. Display entire documents using aggregate.

```

> db.inventory.aggregate([{$match:{}}]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 1
        },
        {
            "warehouse" : "w3",
            "quantity" : 16
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",

```

```

"quantity" : 58,
"size" : {
    "height" : 10,
    "width" : 3.5,
    "unit" : "cm"
},
"quality" : "A",
"instock" : [
    {
        "warehouse" : "w1",
        "quantity" : 20
    },
    {
        "warehouse" : "w2",
        "quantity" : 7
    },
    {
        "warehouse" : "w3",
        "quantity" : 31
    }
]
}
(Cont.)

```

**21. Display all inventory items having item names as pen, pencil, marker and display their item name, quantity and unit of measure.**

```

> db.inventory.aggregate([{$match:{itemname:{$in:["Pen", "Pencil", "Marker"]}}}, {$project:{itemname:1, quantity:1, "size.unit":1}}]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "quantity" : 58,
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603318"),
    "itemname" : "Pencil",
    "quantity" : 590,
    "size" : {

```

```

        "unit" : "cm"
    }
}

```

- 22. Display all inventory items not having item names as pen, pencil, marker and display their item name, quantity and unit of measure.**

```

> db.inventory.aggregate([{$match:{itemname:{$nin:["Pen", "Pencil", "Marker"]}}}, {$project:{itemname:1, quantity:1, "size.unit":1}}]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603313"),
    "itemname" : "Notebook",
    "quantity" : 350,
    "size" : {
        "unit" : "inch"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603314"),
    "itemname" : "Ruler",
    "quantity" : 123,
    "size" : {
        "unit" : "inch"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603315"),
    "itemname" : "Eraser",
    "quantity" : 76,
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603316"),
    "itemname" : "Cutter",
    "quantity" : 68,
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639add7175bb6a14b0603317"),
    "itemname" : "Writing Pad",
    "quantity" : 107,
    "size" : {
        "unit" : "inch"
    }
}

```

```

        "_id" : ObjectId("639add7175bb6a14b0603319"),
        "itemname" : "Tape",
        "quantity" : 360,
        "size" : {
            "unit" : "cm"
        }
    }
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),
    "itemname" : "Boardpins",
    "quantity" : 75,
    "size" : {
        "unit" : "cm"
    }
}
{
    "_id" : ObjectId("639c1d4cf2a551570c47043e"),
    "itemname" : "Crayons",
    "quantity" : 40,
    "size" : {
        "unit" : "inch"
    }
}

```

**23. Display all inventory items having instock quantity between 50 and 100 and display their item name, unit of measure and instock quantity.**

```

> db.inventory.aggregate([
    {
        "$match": {
            "$and": [
                { "instock.quantity": { $gte: 50 } },
                { "instock.quantity": { $lt: 100 } }
            ]
        }
    },
    {
        "$project": {
            "itemname": 1,
            "instock.quantity": 1,
            "size.unit": 1
        }
    }
]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603313"),
    "itemname" : "Notebook",
    "size" : {
        "unit" : "inch"
    },
    "instock" : [
        {
            "quantity" : 170
        },
        {
            "quantity" : 70
        }
    ]
},
{
    "_id" : ObjectId("639add7175bb6a14b0603314"),
    "itemname" : "Ruler",
    "size" : {
        "unit" : "inch"
    }
}

```

```

        },
        "instock" : [
            {
                "quantity" : 46
            },
            {
                "quantity" : 54
            }
        ]
    }
}

{
    "_id" : ObjectId("639add7175bb6a14b0603319"),
    "itemname" : "Tape",
    "size" : {
        "unit" : "cm"
    },
    "instock" : [
        {
            "quantity" : 190
        },
        {
            "quantity" : 79
        },
        {
            "quantity" : 91
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),
    "itemname" : "Boardpins",
    "size" : {
        "unit" : "cm"
    },
    "instock" : [
        {
            "quantity" : 14
        },
        {
            "quantity" : 54
        }
    ]
}

```

#### 24. Display all inventory items sorted on the basis of item names.

```

> db.inventory.aggregate([{$match:{}}, {$sort:{itemname:1}}]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b060331a"),
    "itemname" : "Boardpins",
    "quantity" : 75,

```

```

    "size" : {
        "height" : 2,
        "width" : 1.5,
        "unit" : "cm"
    },
    "quality" : "E",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 14
        },
        {
            "warehouse" : "w3",
            "quantity" : 54
        }
    ]
}
{
    "_id" : ObjectId("639c1d4cf2a551570c47043e"),
    "itemname" : "Crayons",
    "quantity" : 40,
    "size" : {
        "height" : 4,
        "width" : 0.1,
        "unit" : "inch"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 40
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603316"),
    "itemname" : "Cutter",
    "quantity" : 68,
    "size" : {
        "height" : 8,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 21
        },
        {

```

```

        "warehouse" : "w2",
        "quantity" : 26
    },
    {
        "warehouse" : "w3",
        "quantity" : 21
    }
]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603315"),
    "itemname" : "Eraser",
    "quantity" : 76,
    "size" : {
        "height" : 5,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 40
        },
        {
            "warehouse" : "w2",
            "quantity" : 36
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603312"),
    "itemname" : "Marker",
    "quantity" : 58,
    "size" : {
        "height" : 10,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "A",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 20
        },
        {
            "warehouse" : "w2",
            "quantity" : 7
        }
    ]
}

```

```

        "warehouse" : "w3",
        "quantity" : 31
    }
]
{
    "_id" : ObjectId("639add7175bb6a14b0603313"),
    "itemname" : "Notebook",
    "quantity" : 350,
    "size" : {
        "height" : 12,
        "width" : 0.7,
        "unit" : "inch"
    },
    "quality" : "C",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 170
        },
        {
            "warehouse" : "w3",
            "quantity" : 70
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 1
        },
        {
            "warehouse" : "w3",
            "quantity" : 16
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603318"),
    "itemname" : "Pencil",

```

```

"quantity" : 590,
"size" : {
    "height" : 12,
    "width" : 3.5,
    "unit" : "cm"
},
"quality" : "C",
"instock" : [
    {
        "warehouse" : "w2",
        "quantity" : 210
    },
    {
        "warehouse" : "w3",
        "quantity" : 100
    }
]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603314"),
    "itemname" : "Ruler",
    "quantity" : 123,
    "size" : {
        "height" : 12,
        "width" : 0.1,
        "unit" : "inch"
    },
    "quality" : "E",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 46
        },
        {
            "warehouse" : "w3",
            "quantity" : 54
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603319"),
    "itemname" : "Tape",
    "quantity" : 360,
    "size" : {
        "height" : 6,
        "width" : 1,
        "unit" : "cm"
    },
    "quality" : "D",
    "instock" : [

```

```

        {
            "warehouse" : "w1",
            "quantity" : 190
        },
        {
            "warehouse" : "w2",
            "quantity" : 79
        },
        {
            "warehouse" : "w3",
            "quantity" : 91
        }
    ]
}
{
    "_id" : ObjectId("639add7175bb6a14b0603317"),
    "itemname" : "Writing Pad",
    "quantity" : 107,
    "size" : {
        "height" : 15,
        "width" : 0.1,
        "unit" : "inch"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w1",
            "quantity" : 35
        },
        {
            "warehouse" : "w2",
            "quantity" : 27
        },
        {
            "warehouse" : "w3",
            "quantity" : 45
        }
    ]
}

```

**25. Display all inventory items having quantity greater than equal to 70, sorted by their quantity in descending order and display their item names and quantity.**

```

> db.inventory.aggregate([{$match:{quantity:{$gte:70}}}, {$sort:{quantity:-1}},
{$project:{itemname:1, quantity:1}}]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603318"),
    "itemname" : "Pencil",
    "quantity" : 590
}

```

```
{
  "_id" : ObjectId("639add7175bb6a14b0603319"),
  "itemname" : "Tape",
  "quantity" : 360
}
{
  "_id" : ObjectId("639add7175bb6a14b0603313"),
  "itemname" : "Notebook",
  "quantity" : 350
}
{
  "_id" : ObjectId("639add7175bb6a14b0603314"),
  "itemname" : "Ruler",
  "quantity" : 123
}
{
  "_id" : ObjectId("639add7175bb6a14b0603317"),
  "itemname" : "Writing Pad",
  "quantity" : 107
}
{
  "_id" : ObjectId("639add7175bb6a14b0603315"),
  "itemname" : "Eraser",
  "quantity" : 76
}
{
  "_id" : ObjectId("639add7175bb6a14b060331a"),
  "itemname" : "Boardpins",
  "quantity" : 75
}
```

Insert records with same item names.

```
db.inventory.insertMany([
  {itemname: "Pen", quantity: 50, size: {height: 10, width: 2, unit: "cm"}, quality: 'B', instock: [{warehouse: "w3", quantity: 50}]},
  {itemname: "Marker", quantity: 77, size: {height: 10, width: 3.5, unit: "cm"}, quality: 'D', instock: [{warehouse: "w2", quantity: 7}, {warehouse: "w3", quantity: 70}]},
  {itemname: "Pencil", quantity: 90, size: {height: 12, width: 3.5, unit: "cm"}, quality: 'C', instock: [{warehouse: "w1", quantity: 90}]},
  {itemname: "Marker", quantity: 17, size: {height: 10, width: 3.5, unit: "cm"}, quality: 'A', instock: [{warehouse: "w1", quantity: 7}, {warehouse: "w3", quantity: 10}]},
  {itemname: "Pen", quantity: 80, size: {height: 12, width: 3.5, unit: "cm"}, quality: 'B', instock: [{warehouse: "w3", quantity: 80}]}
])
```

```
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("63a01738e43a2a84974a4bd0"),
    ObjectId("63a01738e43a2a84974a4bd1"),
    ObjectId("63a01738e43a2a84974a4bd2"),
  ]
}
```

```

        ObjectId("63a01738e43a2a84974a4bd3"),
        ObjectId("63a01738e43a2a84974a4bd4")
    ]
}

```

## 26. Display all item names having name Pen.

```

> db.inventory.aggregate([{$match:{itemname:"Pen"} }]).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w2",
            "quantity" : 1
        },
        {
            "warehouse" : "w3",
            "quantity" : 16
        }
    ]
}
{
    "_id" : ObjectId("63a01738e43a2a84974a4bd0"),
    "itemname" : "Pen",
    "quantity" : 50,
    "size" : {
        "height" : 10,
        "width" : 2,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w3",
            "quantity" : 50
        }
    ]
}
{
    "_id" : ObjectId("63a01738e43a2a84974a4bd4"),
    "itemname" : "Pen",
    "quantity" : 80,

```

```

    "size" : {
        "height" : 12,
        "width" : 3.5,
        "unit" : "cm"
    },
    "quality" : "B",
    "instock" : [
        {
            "warehouse" : "w3",
            "quantity" : 80
        }
    ]
}

```

**Group:**

```

> db.inventory.aggregate([{$group:{_id:"$itemname"} }])
{
    "_id" : "Notebook"
}
{
    "_id" : "Eraser"
}
{
    "_id" : "Tape"
}
{
    "_id" : "Marker"
}
{
    "_id" : "Ruler"
}
{
    "_id" : "Boardpins"
}
{
    "_id" : "Cutter"
}
{
    "_id" : "Crayons"
}
{
    "_id" : "Pencil"
}
{
    "_id" : "Pen"
}
{
    "_id" : "Writing Pad"
}

```

**27. Display all inventory items grouped by item name.**

```

> db.inventory.aggregate([{$group:{_id:{itemname:"$itemname"} }}])
{
    "_id" : { "itemname" : "Writing Pad" }
}
{
    "_id" : { "itemname" : "Eraser" }
}
{
    "_id" : { "itemname" : "Notebook" }
}
{
    "_id" : { "itemname" : "Boardpins" }
}
{
    "_id" : { "itemname" : "Pencil" }
}
{
    "_id" : { "itemname" : "Pen" }
}
{
    "_id" : { "itemname" : "Cutter" }
}
{
    "_id" : { "itemname" : "Crayons" }
}
{
    "_id" : { "itemname" : "Tape" }
}
{
    "_id" : { "itemname" : "Marker" }
}
{
    "_id" : { "itemname" : "Ruler" }
}

> db.inventory.aggregate([{$group:{_id:{itemname:"$itemname",
    quantity:"$quantity"} }}])
{
    "_id" : { "itemname" : "Eraser", "quantity" : 76 }
}
{
    "_id" : { "itemname" : "Cutter", "quantity" : 68 }
}
{
    "_id" : { "itemname" : "Marker", "quantity" : 58 }
}

```

```
{
  "_id": { "itemname": "Crayons", "quantity": 40 },
  "_id": { "itemname": "Pen", "quantity": 80 },
  "_id": { "itemname": "Marker", "quantity": 17 },
  "_id": { "itemname": "Boardpins", "quantity": 75 },
  "_id": { "itemname": "Marker", "quantity": 77 },
  "_id": { "itemname": "Pen", "quantity": 50 },
  "_id": { "itemname": "Pencil", "quantity": 590 },
  "_id": { "itemname": "Tape", "quantity": 360 },
  "_id": { "itemname": "Notebook", "quantity": 350 },
  "_id": { "itemname": "Writing Pad", "quantity": 107 },
  "_id": { "itemname": "Pencil", "quantity": 90 },
  "_id": { "itemname": "Pen", "quantity": 37 },
  "_id": { "itemname": "Ruler", "quantity": 123 }
}
```

**28. Count the number of inventory items grouped by item name.**

```
> db.inventory.aggregate([{$group:{_id:"$itemname", "Total number of items are:":{$sum:1}}}], {_id: "Cutter", "Total number of items are: " : 1 },
{_id: "Marker", "Total number of items are: " : 3 },
{_id: "Eraser", "Total number of items are: " : 1 },
{_id: "Pen", "Total number of items are: " : 3 },
{_id: "Tape", "Total number of items are: " : 1 },
{_id: "Crayons", "Total number of items are: " : 1 },
{_id: "Boardpins", "Total number of items are: " : 1 },
{_id: "Notebook", "Total number of items are: " : 1 },
{_id: "Pencil", "Total number of items are: " : 2 },
{_id: "Ruler", "Total number of items are: " : 1 },
{_id: "Writing Pad", "Total number of items are: " : 1 }
```

**29. Count the number of inventory items grouped by item name and sort them on item names in ascending order.**

```
> db.inventory.aggregate([{$group:{_id:"$itemname", "Total number of items are:":{$sum:1}}}, {$sort:{_id:1}}], {_id: "Boardpins", "Total number of items are: " : 1 },
{_id: "Crayons", "Total number of items are: " : 1 },
{_id: "Cutter", "Total number of items are: " : 1 },
{_id: "Eraser", "Total number of items are: " : 1 },
{_id: "Marker", "Total number of items are: " : 3 },
{_id: "Notebook", "Total number of items are: " : 1 },
{_id: "Pen", "Total number of items are: " : 3 },
{_id: "Pencil", "Total number of items are: " : 2 },
{_id: "Ruler", "Total number of items are: " : 1 },
{_id: "Tape", "Total number of items are: " : 1 },
{_id: "Writing Pad", "Total number of items are: " : 1 })
```

**30. Count the number of inventory items grouped by item name and sort them on item names in descending order.**

```
> db.inventory.aggregate([{$group:{_id:"$itemname", "Total number of items are:":{$sum:1}}}, {$sort:{_id:-1}}])
{ "_id" : "Writing Pad", "Total number of items are: " : 1 }
{ "_id" : "Tape", "Total number of items are: " : 1 }
{ "_id" : "Ruler", "Total number of items are: " : 1 }
{ "_id" : "Pencil", "Total number of items are: " : 2 }
{ "_id" : "Pen", "Total number of items are: " : 3 }
{ "_id" : "Notebook", "Total number of items are: " : 1 }
{ "_id" : "Marker", "Total number of items are: " : 3 }
{ "_id" : "Eraser", "Total number of items are: " : 1 }
{ "_id" : "Cutter", "Total number of items are: " : 1 }
{ "_id" : "Crayons", "Total number of items are: " : 1 }
{ "_id" : "Boardpins", "Total number of items are: " : 1 }
```

**31. Display the total quantity of all items grouped by itemnames and sort them in ascending order by item name.**

```
> db.inventory.find({itemname:"Pen"}, {itemname:1, quantity:1}).pretty()
{
    "_id" : ObjectId("639add7175bb6a14b0603311"),
    "itemname" : "Pen",
    "quantity" : 37
}
{
    "_id" : ObjectId("63a01738e43a2a84974a4bd0"),
    "itemname" : "Pen",
    "quantity" : 50
}
{
    "_id" : ObjectId("63a01738e43a2a84974a4bd4"),
    "itemname" : "Pen",
    "quantity" : 80
}
> db.inventory.aggregate([{$group:{_id:"$itemname", "Total number of items are:":{$sum:"$quantity"}}, {$sort:{_id:1}}}]
{ "_id" : "Boardpins", "Total number of items are: " : 75 }
{ "_id" : "Crayons", "Total number of items are: " : 40 }
{ "_id" : "Cutter", "Total number of items are: " : 68 }
{ "_id" : "Eraser", "Total number of items are: " : 76 }
{ "_id" : "Marker", "Total number of items are: " : 152 }
{ "_id" : "Notebook", "Total number of items are: " : 350 }
{ "_id" : "Pen", "Total number of items are: " : 167 }
{ "_id" : "Pencil", "Total number of items are: " : 680 }
{ "_id" : "Ruler", "Total number of items are: " : 123 }
{ "_id" : "Tape", "Total number of items are: " : 360 }
{ "_id" : "Writing Pad", "Total number of items are: " : 107 }
```

**32. Display the total quantity of all items grouped by itemnames and sort them in ascending order by item name, having quantity greater than equal to 100.**

```
> db.inventory.aggregate([{$group:{_id:"$itemname", "Total number of items are:":{$sum:"$quantity"}}, {$sort:{_id:1}}, {$match:{ "Total number of items are:":{$gte:100}}}] )
{ "_id" : "Marker", "Total number of items are: " : 152 }
{ "_id" : "Notebook", "Total number of items are: " : 350 }
{ "_id" : "Pen", "Total number of items are: " : 167 }
{ "_id" : "Pencil", "Total number of items are: " : 680 }
{ "_id" : "Ruler", "Total number of items are: " : 123 }
{ "_id" : "Tape", "Total number of items are: " : 360 }
{ "_id" : "Writing Pad", "Total number of items are: " : 107 }
```

**33. Display all inventory item names along with their quantity.**

```
> db.inventory.aggregate([{$group:{_id:"$itemname", "Total number of items are:":{$sum:"$quantity"}}, {$project:{itemname:1, "Total number of items are: ":1}}, {$sort:{_id:1}}] )
{ "_id" : "Boardpins", "Total number of items are: " : 75 }
{ "_id" : "Crayons", "Total number of items are: " : 40 }
{ "_id" : "Cutter", "Total number of items are: " : 68 }
{ "_id" : "Eraser", "Total number of items are: " : 76 }
{ "_id" : "Marker", "Total number of items are: " : 152 }
{ "_id" : "Notebook", "Total number of items are: " : 350 }
{ "_id" : "Pen", "Total number of items are: " : 167 }
{ "_id" : "Pencil", "Total number of items are: " : 680 }
{ "_id" : "Ruler", "Total number of items are: " : 123 }
{ "_id" : "Tape", "Total number of items are: " : 360 }
{ "_id" : "Writing Pad", "Total number of items are: " : 107 }
```

**34. Display all inventory item names along with their quantity, having quantity greater than equal to 50.**

```
> db.inventory.aggregate([{$group:{_id:"$itemname", "Total number of items are:":{$sum:"$quantity"}}, {$match:{ "Total number of items are: ":"{$gte:100}"}}, {$project:{itemname:1, "Total number of items are: ":1}}, {$sort:{_id:1}}] )
{ "_id" : "Marker", "Total number of items are: " : 152 }
{ "_id" : "Notebook", "Total number of items are: " : 350 }
{ "_id" : "Pen", "Total number of items are: " : 167 }
{ "_id" : "Pencil", "Total number of items are: " : 680 }
{ "_id" : "Ruler", "Total number of items are: " : 123 }
{ "_id" : "Tape", "Total number of items are: " : 360 }
{ "_id" : "Writing Pad", "Total number of items are: " : 107 }
```

```

>db.cars.insertMany([
  {"Manufacturer": "Fiat", "Colour": "Black", "Model": "Panda", "Milage": 3292.0, "Price": 2668.32, "Classification": "Motor Cars", "Extras": ["SatNav", "ABS"]},
  ...
  {"Manufacturer": "VW", "Colour": "Blue", "Model": "Polo", "Milage": 79712.0, "Price": 7311.52, "Classification": "Motor Cars", "Extras": ["ABS"]},
  ...
  {"Manufacturer": "VW", "Colour": "Blue", "Model": "Passat", "Milage": 75294.0, "Price": 7488.24, "Classification": "Motor Cars", "Extras": ["Aircon"]},
  ...
  {"Manufacturer": "Fiat", "Colour": "Black", "Model": "Punto", "Milage": 30380.0, "Price": 2784.8, "Classification": "Motor Cars", "Extras": ["PAS", "Power Windows"]},
  ...
  {"Manufacturer": "Fiat", "Colour": "White", "Model": "Panda", "Milage": 32229.0, "Price": 2710.84, "Classification": "Motor Cars", "Extras": ["ABS", "Power Windows"]},
  ...
  {"Manufacturer": "Skoda", "Colour": "Blue", "Model": "Octavia", "Milage": 42901.0, "Price": 6283.96, "Classification": "Motor Cars", "Extras": ["Power Windows", "Auto Wipers", "Aircon"]},
  ...
  {"Manufacturer": "Skoda", "Colour": "Red", "Model": "Fabia", "Milage": 82061.0, "Price": 4717.56, "Classification": "Motor Cars", "Extras": ["Parking Sensors", "SatNav"]},
  ...
  {"Manufacturer": "Fiat", "Colour": "Black", "Model": "Uno", "Milage": 29978.0, "Price": 2800.88, "Classification": "Motor Cars", "Extras": ["Aircon", "Parking Sensors", "PAS"]},
  ...
  {"Manufacturer": "Skoda", "Colour": "Blue", "Model": "Superb", "Milage": 48056.0, "Price": 6077.76, "Classification": "Motor Cars", "Extras": ["ESP", "PAS", "SatNav"]},
  ...
  {"Manufacturer": "Fiat", "Colour": "White", "Model": "Panda", "Milage": 31620.0, "Price": 2735.2, "Classification": "Motor Cars", "Extras": ["SatNav", "Aircon"]},
  ...
  {"Manufacturer": "Fiat", "Colour": "White", "Model": "Uno", "Milage": 28875.0, "Price": 2845.0, "Classification": "Motor Cars", "Extras": ["ABS", "Parking Sensors"]},
  ...
  {"Manufacturer": "VW", "Colour": "Blue", "Model": "Polo", "Milage": 60604.0, "Price": 8075.84, "Classification": "Motor Cars", "Extras": ["PAS", "Auto Wipers", "Power Windows"]},
  ...
  {"Manufacturer": "Fiat", "Colour": "Red", "Model": "Panda", "Milage": 31477.0, "Price": 2740.92, "Classification": "Motor Cars", "Extras": ["Parking Sensors", "SatNav"]},
  ...
  {"Manufacturer": "VW", "Colour": "Red", "Model": "Polo", "Milage": 82795.0, "Price": 7188.2, "Classification": "Motor Cars", "Extras": ["ABS"]},
  ...
  {"Manufacturer": "Fiat", "Colour": "Blue", "Model": "Punto", "Milage": 29553.0, "Price": 2817.88, "Classification": "Motor Cars", "Extras": ["Aircon", "Auto Wipers"]},
  ...
  {"Manufacturer": "Skoda", "Colour": "Green", "Model": "Octavia", "Milage": 55428.0, "Price": 5782.88, "Classification": "Motor Cars", "Extras": ["Parking Sensors"]},
  ...
  {"Manufacturer": "Fiat", "Colour": "White", "Model": "Panda", "Milage": 31274.0, "Price": 2749.04, "Classification": "Motor Cars", "Extras": ["Power Windows", "Parking Sensors"]},

```

```

...
{"Manufacturer":"Fiat","Colour":"Green","Model":"Panda","Milage":31125.0,"Price":2755.0
,"Classification":"Motor Cars","Extras":["Auto Wipers"]},
...
{"Manufacturer":"VW","Colour":"Blue","Model":"Polo","Milage":58557.0,"Price":8157.72,
"Classification":"Motor Cars","Extras":["PAS","ESP"]},
...
{"Manufacturer":"Skoda","Colour":"White","Model":"Fabia","Milage":53574.0,"Price":5857
.04,"Classification":"Motor Cars","Extras":["SatNav"]},
...
{"Manufacturer":"VW","Colour":"Red","Model":"Polo","Milage":61193.0,"Price":8052.28,"
Classification":"Motor Cars","Extras":["Power Windows"]}]
... )
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("63a41361a73fb34b15d891e2"),
        ObjectId("63a41361a73fb34b15d891e3"),
        ObjectId("63a41361a73fb34b15d891e4"),
        ObjectId("63a41361a73fb34b15d891e5"),
        ObjectId("63a41361a73fb34b15d891e6"),
        ObjectId("63a41361a73fb34b15d891e7"),
        ObjectId("63a41361a73fb34b15d891e8"),
        ObjectId("63a41361a73fb34b15d891e9"),
        ObjectId("63a41361a73fb34b15d891ea"),
        ObjectId("63a41361a73fb34b15d891eb"),
        ObjectId("63a41361a73fb34b15d891ec"),
        ObjectId("63a41361a73fb34b15d891ed"),
        ObjectId("63a41361a73fb34b15d891ee"),
        ObjectId("63a41361a73fb34b15d891ef"),
        ObjectId("63a41361a73fb34b15d891f0"),
        ObjectId("63a41361a73fb34b15d891f1"),
        ObjectId("63a41361a73fb34b15d891f2"),
        ObjectId("63a41361a73fb34b15d891f3"),
        ObjectId("63a41361a73fb34b15d891f4"),
        ObjectId("63a41361a73fb34b15d891f5"),
        ObjectId("63a41361a73fb34b15d891f6")
    ]
}

```

**Perform the following mongodb quesries on the above data:-**

**1. Display all documents.**

```

> db.cars.find().pretty()
{
    "_id" : ObjectId("63a41361a73fb34b15d891e2"),
    "Manufacturer" : "Fiat",
    "Colour" : "Black",
    "Model" : "Panda",
    "Milage" : 33292,

```

```

    "Price" : 2668.32,
    "Classification" : "Motor Cars",
    "Extras" : [
        "SatNav",
        "ABS"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891e3"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Polo",
    "Milage" : 79712,
    "Price" : 7311.52,
    "Classification" : "Motor Cars",
    "Extras" : [
        "ABS"
    ]
}

```

**(Cont.)**

## 2. Display cars having "Manufacturer" as "Fiat".

```

> db.cars.find({Manufacturer:"Fiat"}).pretty()
{
    "_id" : ObjectId("63a41361a73fb34b15d891e2"),
    "Manufacturer" : "Fiat",
    "Colour" : "Black",
    "Model" : "Panda",
    "Milage" : 33292,
    "Price" : 2668.32,
    "Classification" : "Motor Cars",
    "Extras" : [
        "SatNav",
        "ABS"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891e5"),
    "Manufacturer" : "Fiat",
    "Colour" : "Black",
    "Model" : "Punto",
    "Milage" : 30380,
    "Price" : 2784.8,
    "Classification" : "Motor Cars",
    "Extras" : [
        "PAS",
        "Power Windows"
    ]
}

```

}

**(Cont.)**

### **3. Display cars having "Extras" as "PAS".**

```
> db.cars.find({Extras:"PAS"}).pretty()
{
    "_id" : ObjectId("63a41361a73fb34b15d891e5"),
    "Manufacturer" : "Fiat",
    "Colour" : "Black",
    "Model" : "Punto",
    "Milage" : 30380,
    "Price" : 2784.8,
    "Classification" : "Motor Cars",
    "Extras" : [
        "PAS",
        "Power Windows"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891e9"),
    "Manufacturer" : "Fiat",
    "Colour" : "Black",
    "Model" : "Uno",
    "Milage" : 29978,
    "Price" : 2800.88,
    "Classification" : "Motor Cars",
    "Extras" : [
        "Aircon",
        "Parking Sensors",
        "PAS"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891ea"),
    "Manufacturer" : "Skoda",
    "Colour" : "Blue",
    "Model" : "Superb",
    "Milage" : 48056,
    "Price" : 6077.76,
    "Classification" : "Motor Cars",
    "Extras" : [
        "ESP",
        "PAS",
        "SatNav"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891ed"),
```

```

        "Manufacturer" : "VW",
        "Colour" : "Blue",
        "Model" : "Polo",
        "Milage" : 60604,
        "Price" : 8075.84,
        "Classification" : "Motor Cars",
        "Extras" : [
            "PAS",
            "Auto Wipers",
            "Power Windows"
        ]
    }
{
    "_id" : ObjectId("63a41361a73fb34b15d891f4"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Polo",
    "Milage" : 58557,
    "Price" : 8157.72,
    "Classification" : "Motor Cars",
    "Extras" : [
        "PAS",
        "ESP"
    ]
}

```

#### **4. Display cars having price greater than 5000.**

```

> db.cars.find({Price:{$gt:5000}}).pretty()
{
    "_id" : ObjectId("63a41361a73fb34b15d891e3"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Polo",
    "Milage" : 79712,
    "Price" : 7311.52,
    "Classification" : "Motor Cars",
    "Extras" : [
        "ABS"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891e4"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Passat",
    "Milage" : 75294,
    "Price" : 7488.24,
    "Classification" : "Motor Cars",
    "Extras" : [

```

```

        "Aircon"
    ]
}

(Cont.)

5. Dipslay cars having "Milage" between 50000 and 60000.

> db.cars.find({$and:[{Milage:{$gte:50000}}, {Milage:{$lte:60000}}]}).pretty()
{
    "_id" : ObjectId("63a41361a73fb34b15d891f1"),
    "Manufacturer" : "Skoda",
    "Colour" : "Green",
    "Model" : "Octavia",
    "Milage" : 55428,
    "Price" : 5782.88,
    "Classification" : "Motor Cars",
    "Extras" : [
        "Parking Sensors"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891f4"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Polo",
    "Milage" : 58557,
    "Price" : 8157.72,
    "Classification" : "Motor Cars",
    "Extras" : [
        "PAS",
        "ESP"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891f5"),
    "Manufacturer" : "Skoda",
    "Colour" : "White",
    "Model" : "Fabia",
    "Milage" : 53574,
    "Price" : 5857.04,
    "Classification" : "Motor Cars",
    "Extras" : [
        "SatNav"
    ]
}

```

## **6. Display cars having "Manufacturer" name starting with "V".**

```
> db.cars.find({Manufacturer:$regex:/^V/}).pretty()
```

```
{
    "_id" : ObjectId("63a41361a73fb34b15d891e3"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Polo",
    "Milage" : 79712,
    "Price" : 7311.52,
    "Classification" : "Motor Cars",
    "Extras" : [
        "ABS"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891e4"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Passat",
    "Milage" : 75294,
    "Price" : 7488.24,
    "Classification" : "Motor Cars",
    "Extras" : [
        "Aircon"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891ed"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Polo",
    "Milage" : 60604,
    "Price" : 8075.84,
    "Classification" : "Motor Cars",
    "Extras" : [
        "PAS",
        "Auto Wipers",
        "Power Windows"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891ef"),
    "Manufacturer" : "VW",
    "Colour" : "Red",
    "Model" : "Polo",
    "Milage" : 82795,
    "Price" : 7188.2,
    "Classification" : "Motor Cars",
    "Extras" : [
        "ABS"
    ]
}
```

```
{
  "_id" : ObjectId("63a41361a73fb34b15d891f4"),
  "Manufacturer" : "VW",
  "Colour" : "Blue",
  "Model" : "Polo",
  "Milage" : 58557,
  "Price" : 8157.72,
  "Classification" : "Motor Cars",
  "Extras" : [
    "PAS",
    "ESP"
  ]
}
{
  "_id" : ObjectId("63a41361a73fb34b15d891f6"),
  "Manufacturer" : "VW",
  "Colour" : "Red",
  "Model" : "Polo",
  "Milage" : 61193,
  "Price" : 8052.28,
  "Classification" : "Motor Cars",
  "Extras" : [
    "Power Windows"
  ]
}
```

## 7. Display cars having "Model" as " Polo", "Uno", "Fabia".

```
> db.cars.find({Model:{$in:["Polo", "Uno", "Fabia"]}}).pretty()
{
  "_id" : ObjectId("63a41361a73fb34b15d891e3"),
  "Manufacturer" : "VW",
  "Colour" : "Blue",
  "Model" : "Polo",
  "Milage" : 79712,
  "Price" : 7311.52,
  "Classification" : "Motor Cars",
  "Extras" : [
    "ABS"
  ]
}
{
  "_id" : ObjectId("63a41361a73fb34b15d891e8"),
  "Manufacturer" : "Skoda",
  "Colour" : "Red",
  "Model" : "Fabia",
  "Milage" : 82061,
  "Price" : 4717.56,
  "Classification" : "Motor Cars",
  "Extras" : [
```

```

        "Parking Sensors",
        "SatNav"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891e9"),
    "Manufacturer" : "Fiat",
    "Colour" : "Black",
    "Model" : "Uno",
    "Milage" : 29978,
    "Price" : 2800.88,
    "Classification" : "Motor Cars",
    "Extras" : [
        "Aircon",
        "Parking Sensors",
        "PAS"
    ]
}

```

(Cont.)

## 8. Display cars having "Model" as " Polo" and "Milage" greater than 60000.

```

> db.cars.find({Model:{$in:["Polo", "Uno", "Fabia"]}}).pretty()
{
    "_id" : ObjectId("63a41361a73fb34b15d891e3"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Polo",
    "Milage" : 79712,
    "Price" : 7311.52,
    "Classification" : "Motor Cars",
    "Extras" : [
        "ABS"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891e8"),
    "Manufacturer" : "Skoda",
    "Colour" : "Red",
    "Model" : "Fabia",
    "Milage" : 82061,
    "Price" : 4717.56,
    "Classification" : "Motor Cars",
    "Extras" : [
        "Parking Sensors",
        "SatNav"
    ]
}

```

```

        "_id" : ObjectId("63a41361a73fb34b15d891e9"),
        "Manufacturer" : "Fiat",
        "Colour" : "Black",
        "Model" : "Uno",
        "Milage" : 29978,
        "Price" : 2800.88,
        "Classification" : "Motor Cars",
        "Extras" : [
            "Aircon",
            "Parking Sensors",
            "PAS"
        ]
    }
{
    "_id" : ObjectId("63a41361a73fb34b15d891ec"),
    "Manufacturer" : "Fiat",
    "Colour" : "White",
    "Model" : "Uno",
    "Milage" : 28875,
    "Price" : 2845,
    "Classification" : "Motor Cars",
    "Extras" : [
        "ABS",
        "Parking Sensors"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891ed"),
    "Manufacturer" : "VW",
    "Colour" : "Blue",
    "Model" : "Polo",
    "Milage" : 60604,
    "Price" : 8075.84,
    "Classification" : "Motor Cars",
    "Extras" : [
        "PAS",
        "Auto Wipers",
        "Power Windows"
    ]
}

```

**(Cont.)**

## **9. Display cars not having price between 5000 and 8000.**

```
> db.cars.find({$or:[{Price:{$not:{$gt:5000}}}, {Price:{$not:{$lt:8000}}}}}).pretty()
{
    "_id" : ObjectId("63a41361a73fb34b15d891e2"),
    "Manufacturer" : "Fiat",
    "Colour" : "Black",
```

```

        "Model" : "Panda",
        "Milage" : 33292,
        "Price" : 2668.32,
        "Classification" : "Motor Cars",
        "Extras" : [
            "SatNav",
            "ABS"
        ]
    }
{
    "_id" : ObjectId("63a41361a73fb34b15d891e5"),
    "Manufacturer" : "Fiat",
    "Colour" : "Black",
    "Model" : "Punto",
    "Milage" : 30380,
    "Price" : 2784.8,
    "Classification" : "Motor Cars",
    "Extras" : [
        "PAS",
        "Power Windows"
    ]
}
{
    "_id" : ObjectId("63a41361a73fb34b15d891e6"),
    "Manufacturer" : "Fiat",
    "Colour" : "White",
    "Model" : "Panda",
    "Milage" : 32229,
    "Price" : 2710.84,
    "Classification" : "Motor Cars",
    "Extras" : [
        "ABS",
        "Power Windows"
    ]
}

```

**(Cont.)**

**10. Count the number of cars having "Colour" as "White".**

```
> db.cars.count({Colour:"White"})
5
```

**11. Calculate the price of cars having "Colour" as "Blue".**

```
> db.cars.aggregate([
    { $match: { Colour: "Blue" } },
    { $group: { _id: null, "Total price of cars having colour Blue": "{$sum: "$Price"}" } }
], {
    "_id": null, "Total price of cars having colour Blue": "46212.92"
})
```

**12. Calculate the average price of cars having "Colour" as "Blue".**

```
> db.cars.aggregate([{$match:{Colour:"Blue"}}, {$group:{_id:null, "Average price of cars having colour Blue: ":{$avg:"$Price"}}}])  
{ "_id" : null, "Average price of cars having colour Blue: " : 6601.845714285714 }
```

### **13. Display cars having maximum price.**

```
> db.cars.aggregate([{$group:{_id:null, "Car having maximum price: ":{$max:"$Price"}}}])  
{ "_id" : null, "Car having maximum price: " : 8157.72 }
```

### **14. Display cars having minimum price.**

```
> db.cars.aggregate([{$group:{_id:null, "Car having minimum price: ":{$min:"$Price"}}}])  
{ "_id" : null, "Car having minimum price: " : 2668.32 }
```

### **15. Count the number of all cars.**

```
> db.cars.count({})  
21
```

### **16. Display cars "Manufacturer" and "Price".**

```
> db.cars.aggregate([{$project:{Manufacturer:1, Price:1}}]).pretty()  
{  
    "_id" : ObjectId("63a41361a73fb34b15d891e2"),  
    "Manufacturer" : "Fiat",  
    "Price" : 2668.32  
}  
{  
    "_id" : ObjectId("63a41361a73fb34b15d891e3"),  
    "Manufacturer" : "VW",  
    "Price" : 7311.52  
}  
{  
    "_id" : ObjectId("63a41361a73fb34b15d891e4"),  
    "Manufacturer" : "VW",  
    "Price" : 7488.24  
}  
{  
    "_id" : ObjectId("63a41361a73fb34b15d891e5"),  
    "Manufacturer" : "Fiat",  
    "Price" : 2784.8  
}  
{  
    "_id" : ObjectId("63a41361a73fb34b15d891e6"),  
    "Manufacturer" : "Fiat",  
    "Price" : 2710.84  
}
```

**(Cont.)**

**17. Display the entire document of the car with maximum price.**

```
cars> db.cars.find().sort({Price:-1}).limit(1).pretty();
[  
  {  
    _id: ObjectId("63b3bb102ee3967c7db04278"),  
    Manufacturer: 'VW',  
    Colour: 'Blue',  
    Model: 'Polo',  
    Milage: 58557,  
    Price: 8157.72,  
    Classification: 'Motor Cars',  
    Extras: [ 'PAS', 'ESP' ]  
  }  
]
```

**19. Display cars where Extras array has exactly 2 elements.**

```
cars> db.cars.find({ "Extras":{$size:2} })
[  
  {  
    _id: ObjectId("63b3bb102ee3967c7db04266"),  
    Manufacturer: 'Fiat',  
    Colour: 'Black',  
    Model: 'Panda',  
    Milage: 33292,  
    Price: 2668.32,  
    Classification: 'Motor Cars',  
    Extras: [ 'SatNav', 'ABS' ]  
  },  
  {  
    _id: ObjectId("63b3bb102ee3967c7db04269"),  
    Manufacturer: 'Fiat',  
    Colour: 'Black',  
    Model: 'Punto',  
    Milage: 30380,  
    Price: 2784.8,  
    Classification: 'Motor Cars',  
    Extras: [ 'PAS', 'Power Windows' ]  
  },  
  {  
    _id: ObjectId("63b3bb102ee3967c7db0426a"),  
    Manufacturer: 'Fiat',  
    Colour: 'White',  
    Model: 'Panda',  
    Milage: 32229,  
    Price: 2710.84,  
    Classification: 'Motor Cars',  
    Extras: [ 'ABS', 'Power Windows' ]  
}
```

```
}
```

**(Cont.)**

```
]
```

## 20. Display cars having colour as White and Model as Panda.

```
cars> db.cars.find({Colour:"White", Model:"Panda"})
[
{
  _id: ObjectId("63b3bb102ee3967c7db0426a"),
  Manufacturer: 'Fiat',
  Colour: 'White',
  Model: 'Panda',
  Milage: 32229,
  Price: 2710.84,
  Classification: 'Motor Cars',
  Extras: [ 'ABS', 'Power Windows' ]
},
{
  _id: ObjectId("63b3bb102ee3967c7db0426f"),
  Manufacturer: 'Fiat',
  Colour: 'White',
  Model: 'Panda',
  Milage: 31620,
  Price: 2735.2,
  Classification: 'Motor Cars',
  Extras: [ 'SatNav', 'Aircon' ]
},
{
  _id: ObjectId("63b3bb102ee3967c7db04276"),
  Manufacturer: 'Fiat',
  Colour: 'White',
  Model: 'Panda',
  Milage: 31274,
  Price: 2749.04,
  Classification: 'Motor Cars',
  Extras: [ 'Power Windows', 'Parking Sensors' ]
}
]
```

## Using aggregate

```
cars> db.cars.aggregate([{$match:{Colour:"White", Model:"Panda"} }])
[
{
  _id: ObjectId("63b3bb102ee3967c7db0426a"),
  Manufacturer: 'Fiat',
  Colour: 'White',
  Model: 'Panda',
  Milage: 32229,
```

```

Price: 2710.84,
Classification: 'Motor Cars',
Extras: [ 'ABS', 'Power Windows' ]
},
{
_id: ObjectId("63b3bb102ee3967c7db0426f"),
Manufacturer: 'Fiat',
Colour: 'White',
Model: 'Panda',
Milage: 31620,
Price: 2735.2,
Classification: 'Motor Cars',
Extras: [ 'SatNav', 'Aircon' ]
},
{
_id: ObjectId("63b3bb102ee3967c7db04276"),
Manufacturer: 'Fiat',
Colour: 'White',
Model: 'Panda',
Milage: 31274,
Price: 2749.04,
Classification: 'Motor Cars',
Extras: [ 'Power Windows', 'Parking Sensors' ]
}
]

```

## **21. Display sum of prices of White coloured Panda cars.**

```

cars> db.cars.aggregate([{$match:{Colour:"White", Model:"Panda"}}, {$group:{_id:null, "Sum of price of cars with colour white and model Panda: ":{$sum:"$Price"}}}])
[
{
  _id: null,
  'Sum of price of cars with colour white and model Panda: ': 8195.08
}
]

```

## **22. Display average price of White coloured Panda cars.**

```

cars> db.cars.aggregate([{$match:{Colour:"White", Model:"Panda"}}, {$group:{_id:null, "Average price of cars with colour white and model Panda: ":{$avg:"$Price"}}}])
[
{
  _id: null,
  'Average price of cars with colour white and model Panda: ': 2731.693333333333
}
]

```

## **23. Display the count of White coloured Panda cars.**

```

cars> db.cars.aggregate([{$match:{Colour:"White", Model:"Panda"}}, {$group:{_id:null, "Number of cars with colour white and model Panda: ":{$sum:1}} }])
[
  {
    _id: null,
    'Number of cars with colour white and model Panda: ': 3
  }
]

```

#### **24. Export a new database with cars where Colour is White.**

```

cars> db.cars.aggregate([{$match:{Colour:"White"}}, {"$out:"whitecars"}])

```

```

cars> db.whitecars.find()
[
  {
    _id: ObjectId("63b3bb102ee3967c7db0426a"),
    Manufacturer: 'Fiat',
    Colour: 'White',
    Model: 'Panda',
    Milage: 32229,
    Price: 2710.84,
    Classification: 'Motor Cars',
    Extras: [ 'ABS', 'Power Windows' ]
  },
  {
    _id: ObjectId("63b3bb102ee3967c7db0426f"),
    Manufacturer: 'Fiat',
    Colour: 'White',
    Model: 'Panda',
    Milage: 31620,
    Price: 2735.2,
    Classification: 'Motor Cars',
    Extras: [ 'SatNav', 'Aircon' ]
  },
  {
    _id: ObjectId("63b3bb102ee3967c7db04270"),
    Manufacturer: 'Fiat',
    Colour: 'White',
    Model: 'Uno',
    Milage: 28875,
    Price: 2845,
    Classification: 'Motor Cars',
    Extras: [ 'ABS', 'Parking Sensors' ]
  },
  {
    _id: ObjectId("63b3bb102ee3967c7db04276"),
    Manufacturer: 'Fiat',
    Colour: 'White',
    Model: 'Panda',
  }
]

```

```

Milage: 31274,
Price: 2749.04,
Classification: 'Motor Cars',
Extras: [ 'Power Windows', 'Parking Sensors' ]
},
{
  _id: ObjectId("63b3bb102ee3967c7db04279"),
  Manufacturer: 'Skoda',
  Colour: 'White',
  Model: 'Fabia',
  Milage: 53574,
  Price: 5857.04,
  Classification: 'Motor Cars',
  Extras: [ 'SatNav' ]
}
]

```

## 25. Export a database where Colour is White and Model is Panda.

```
cars> db.cars.aggregate([{$match:{Colour:"White", Model:"Panda"}}, {$out:"cars1"}])
```

```

cars> db.cars1.find()
[
  {
    _id: ObjectId("63b3bb102ee3967c7db0426a"),
    Manufacturer: 'Fiat',
    Colour: 'White',
    Model: 'Panda',
    Milage: 32229,
    Price: 2710.84,
    Classification: 'Motor Cars',
    Extras: [ 'ABS', 'Power Windows' ]
  },
  {
    _id: ObjectId("63b3bb102ee3967c7db0426f"),
    Manufacturer: 'Fiat',
    Colour: 'White',
    Model: 'Panda',
    Milage: 31620,
    Price: 2735.2,
    Classification: 'Motor Cars',
    Extras: [ 'SatNav', 'Aircon' ]
  },
  {
    _id: ObjectId("63b3bb102ee3967c7db04276"),
    Manufacturer: 'Fiat',
    Colour: 'White',
    Model: 'Panda',
    Milage: 31274,
    Price: 2749.04,
  }
]
```

```

Classification: 'Motor Cars',
Extras: [ 'Power Windows', 'Parking Sensors' ]
}
]

```

**26. Export a database where Colour is White and Model is Panda with only fields Manufacturer and Price only.**

```

cars> db.cars.aggregate([{$match:{Colour:"White", Model:"Panda"}}, {$project:{Manufacturer:1, Price:1}}, {$out:"cars2"}])

```

```

cars> db.cars2.find()
[
{
  _id: ObjectId("63b3bb102ee3967c7db0426a"),
  Manufacturer: 'Fiat',
  Price: 2710.84
},
{
  _id: ObjectId("63b3bb102ee3967c7db0426f"),
  Manufacturer: 'Fiat',
  Price: 2735.2
},
{
  _id: ObjectId("63b3bb102ee3967c7db04276"),
  Manufacturer: 'Fiat',
  Price: 2749.04
}
]

```

**27. Unwind the database on the basis of Extras array.**

```

cars> db.cars.aggregate([{$unwind:"$Extras"}])
[
{
  _id: ObjectId("63b3bb102ee3967c7db04266"),
  Manufacturer: 'Fiat',
  Colour: 'Black',
  Model: 'Panda',
  Milage: 33292,
  Price: 2668.32,
  Classification: 'Motor Cars',
  Extras: 'SatNav'
},
{
  _id: ObjectId("63b3bb102ee3967c7db04266"),
  Manufacturer: 'Fiat',
  Colour: 'Black',
  Model: 'Panda',
  Milage: 33292,

```

```

Price: 2668.32,
Classification: 'Motor Cars',
Extras: 'ABS'
},
{
_id: ObjectId("63b3bb102ee3967c7db04267"),
Manufacturer: 'VW',
Colour: 'Blue',
Model: 'Polo',
Milage: 79712,
Price: 7311.52,
Classification: 'Motor Cars',
Extras: 'ABS'
},
{
_id: ObjectId("63b3bb102ee3967c7db04268"),
Manufacturer: 'VW',
Colour: 'Blue',
Model: 'Passat',
Milage: 75294,
Price: 7488.24,
Classification: 'Motor Cars',
Extras: 'Aircon'
},
{
_id: ObjectId("63b3bb102ee3967c7db04269"),
Manufacturer: 'Fiat',
Colour: 'Black',
Model: 'Punto',
Milage: 30380,
Price: 2784.8,
Classification: 'Motor Cars',
Extras: 'PAS'
},
{
_id: ObjectId("63b3bb102ee3967c7db04269"),
Manufacturer: 'Fiat',
Colour: 'Black',
Model: 'Punto',
Milage: 30380,
Price: 2784.8,
Classification: 'Motor Cars',
Extras: 'Power Windows'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426a"),
Manufacturer: 'Fiat',
Colour: 'White',
Model: 'Panda',
Milage: 32229,

```

```
Price: 2710.84,  
Classification: 'Motor Cars',  
Extras: 'ABS'  
},  
{  
    _id: ObjectId("63b3bb102ee3967c7db0426a"),  
    Manufacturer: 'Fiat',  
    Colour: 'White',  
    Model: 'Panda',  
    Milage: 32229,  
    Price: 2710.84,  
    Classification: 'Motor Cars',  
    Extras: 'Power Windows'  
},  
{  
    _id: ObjectId("63b3bb102ee3967c7db0426b"),  
    Manufacturer: 'Skoda',  
    Colour: 'Blue',  
    Model: 'Octavia',  
    Milage: 42901,  
    Price: 6283.96,  
    Classification: 'Motor Cars',  
    Extras: 'Power Windows'  
},  
{  
    _id: ObjectId("63b3bb102ee3967c7db0426b"),  
    Manufacturer: 'Skoda',  
    Colour: 'Blue',  
    Model: 'Octavia',  
    Milage: 42901,  
    Price: 6283.96,  
    Classification: 'Motor Cars',  
    Extras: 'Auto Wipers'  
},  
{  
    _id: ObjectId("63b3bb102ee3967c7db0426b"),  
    Manufacturer: 'Skoda',  
    Colour: 'Blue',  
    Model: 'Octavia',  
    Milage: 42901,  
    Price: 6283.96,  
    Classification: 'Motor Cars',  
    Extras: 'Aircon'  
},  
{  
    _id: ObjectId("63b3bb102ee3967c7db0426c"),  
    Manufacturer: 'Skoda',  
    Colour: 'Red',  
    Model: 'Fabia',  
    Milage: 82061,
```

```

Price: 4717.56,
Classification: 'Motor Cars',
Extras: 'Parking Sensors'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426c"),
Manufacturer: 'Skoda',
Colour: 'Red',
Model: 'Fabia',
Milage: 82061,
Price: 4717.56,
Classification: 'Motor Cars',
Extras: 'SatNav'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426d"),
Manufacturer: 'Fiat',
Colour: 'Black',
Model: 'Uno',
Milage: 29978,
Price: 2800.88,
Classification: 'Motor Cars',
Extras: 'Aircon'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426d"),
Manufacturer: 'Fiat',
Colour: 'Black',
Model: 'Uno',
Milage: 29978,
Price: 2800.88,
Classification: 'Motor Cars',
Extras: 'Parking Sensors'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426d"),
Manufacturer: 'Fiat',
Colour: 'Black',
Model: 'Uno',
Milage: 29978,
Price: 2800.88,
Classification: 'Motor Cars',
Extras: 'PAS'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426e"),
Manufacturer: 'Skoda',
Colour: 'Blue',
Model: 'Superb',
Milage: 48056,

```

```

Price: 6077.76,
Classification: 'Motor Cars',
Extras: 'ESP'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426e"),
Manufacturer: 'Skoda',
Colour: 'Blue',
Model: 'Superb',
Milage: 48056,
Price: 6077.76,
Classification: 'Motor Cars',
Extras: 'PAS'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426e"),
Manufacturer: 'Skoda',
Colour: 'Blue',
Model: 'Superb',
Milage: 48056,
Price: 6077.76,
Classification: 'Motor Cars',
Extras: 'SatNav'
},
{
_id: ObjectId("63b3bb102ee3967c7db0426f"),
Manufacturer: 'Fiat',
Colour: 'White',
Model: 'Panda',
Milage: 31620,
Price: 2735.2,
Classification: 'Motor Cars',
Extras: 'SatNav'
}
]
Type "it" for more

```

**28. Unwind the database on the basis of Extras array and only display first three documents.**

```

cars> db.cars.aggregate([{$unwind:"$Extras"}, {$limit:3}])
[
{
_id: ObjectId("63b3bb102ee3967c7db04266"),
Manufacturer: 'Fiat',
Colour: 'Black',
Model: 'Panda',
Milage: 33292,
Price: 2668.32,
Classification: 'Motor Cars',

```

```

    Extras: 'SatNav'
},
{
  _id: ObjectId("63b3bb102ee3967c7db04266"),
  Manufacturer: 'Fiat',
  Colour: 'Black',
  Model: 'Panda',
  Milage: 33292,
  Price: 2668.32,
  Classification: 'Motor Cars',
  Extras: 'ABS'
},
{
  _id: ObjectId("63b3bb102ee3967c7db04267"),
  Manufacturer: 'VW',
  Colour: 'Blue',
  Model: 'Polo',
  Milage: 79712,
  Price: 7311.52,
  Classification: 'Motor Cars',
  Extras: 'ABS'
}
]

```

**29. Unwind the database on the basis of Extras array and skip the first three documents.**

```

cars> db.cars.aggregate([{$unwind:"$Extras"}, {$skip:3}])
[
  {
    _id: ObjectId("63b3bb102ee3967c7db04268"),
    Manufacturer: 'VW',
    Colour: 'Blue',
    Model: 'Passat',
    Milage: 75294,
    Price: 7488.24,
    Classification: 'Motor Cars',
    Extras: 'Aircon'
  },
  {
    _id: ObjectId("63b3bb102ee3967c7db04269"),
    Manufacturer: 'Fiat',
    Colour: 'Black',
    Model: 'Punto',
    Milage: 30380,
    Price: 2784.8,
    Classification: 'Motor Cars',
    Extras: 'PAS'
  },
  {

```

```

_id: ObjectId("63b3bb102ee3967c7db04269"),
Manufacturer: 'Fiat',
Colour: 'Black',
Model: 'Punto',
Milage: 30380,
Price: 2784.8,
Classification: 'Motor Cars',
Extras: 'Power Windows'
}
(Cont.)
]

```

**30. Add an array called Dimension:[1000, 500] to all the documents.**

```

cars> db.cars.updateMany( { }, { $set: { Dimension:[1000, 500]} })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 21,
  modifiedCount: 21,
  upsertedCount: 0
}

```

```

cars> db.cars.find().pretty()
[
  {
    _id: ObjectId("63b3bb102ee3967c7db04266"),
    Manufacturer: 'Fiat',
    Colour: 'Black',
    Model: 'Panda',
    Milage: 33292,
    Price: 2668.32,
    Classification: 'Motor Cars',
    Extras: [ 'SatNav', 'ABS' ],
    Dimension: [ 1000, 500 ]
  },
  {
    _id: ObjectId("63b3bb102ee3967c7db04267"),
    Manufacturer: 'VW',
    Colour: 'Blue',
    Model: 'Polo',
    Milage: 79712,
    Price: 7311.52,
    Classification: 'Motor Cars',
    Extras: [ 'ABS' ],
    Dimension: [ 1000, 500 ]
  }
]

```

```

(Cont.)
]

```

US-TCS-604 – Big Data Analytics  
Practical 6 – MapReduce Indexing

**Show all records.**

```
inventory> db.inventory.find()
[ {
  _id: ObjectId("63ccf7df04ccbffd526d1b79"),
  itemname: 'Pen',
  quantity: 37,
  size: { height: 10, width: 2, unit: 'cm' },
  quality: 'B',
  instock: [
    { warehouse: 'w2', quantity: 1 },
    { warehouse: 'w3', quantity: 16 }
  ]
},
{
  _id: ObjectId("63ccf7df04ccbffd526d1b7a"),
  itemname: 'Marker',
  quantity: 58,
  size: { height: 10, width: 3.5, unit: 'cm' },
  quality: 'A',
  instock: [
    { warehouse: 'w1', quantity: 20 },
    { warehouse: 'w2', quantity: 7 },
    { warehouse: 'w3', quantity: 31 }
  ]
},
{
  _id: ObjectId("63ccf7df04ccbffd526d1b7b"),
  itemname: 'Notebook',
  quantity: 350,
  size: { height: 12, width: 0.7, unit: 'inch' },
  quality: 'C',
  instock: [
    { warehouse: 'w1', quantity: 170 },
    { warehouse: 'w3', quantity: 70 }
  ]
},
(Cont.)
]
```

**1. Perform map reduce on the data to display the sum of quantities of each record.**

```
inventory> var map1=function(){emit(this.itemname, this.quantity)}
inventory> var reduce1=function(itemname, quantity){return Array.sum(quantity)}
inventory> db.inventory.mapReduce(map1, reduce1, {out:"newdoc"})
```

DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.  
See <https://docs.mongodb.com/manual/core/map-reduce> for details.

```
{ result: 'newdoc', ok: 1 }
inventory> db.newdoc.find()
[
  { _id: 'Cutter', value: 68 },
  { _id: 'Pencil', value: 680 },
  { _id: 'Ruler', value: 123 },
  { _id: 'Writing Pad', value: 107 },
  { _id: 'Tape', value: 360 },
  { _id: 'Boardpins', value: 75 },
  { _id: 'Eraser', value: 76 },
  { _id: 'Pen', value: 167 },
  { _id: 'Crayons', value: 40 },
  { _id: 'Marker', value: 152 },
  { _id: 'Notebook', value: 350 }
]
```

## 2. Perform map reduce to display the average quantities of all items.

```
inventory> var reduce2=function(itemname, quantity){return Array.avg(quantity)}
inventory> db.inventory.mapReduce(map1, reduce2, {out:"newdoc2"})
{ result: 'newdoc2', ok: 1 }
inventory> db.newdoc2.find()
[
  { _id: 'Notebook', value: 350 },
  { _id: 'Cutter', value: 68 },
  { _id: 'Marker', value: 50.666666666666664 },
  { _id: 'Pencil', value: 340 },
  { _id: 'Crayons', value: 40 },
  { _id: 'Eraser', value: 76 },
  { _id: 'Pen', value: 55.666666666666664 },
  { _id: 'Writing Pad', value: 107 },
  { _id: 'Tape', value: 360 },
  { _id: 'Boardpins', value: 75 },
  { _id: 'Ruler', value: 123 }
]
```

## 3. Sort the resultant records in descending order.

```
inventory> db.newdoc2.find().sort({value:-1})
[
  { _id: 'Tape', value: 360 },
  { _id: 'Notebook', value: 350 },
  { _id: 'Pencil', value: 340 },
  { _id: 'Ruler', value: 123 },
  { _id: 'Writing Pad', value: 107 },
  { _id: 'Eraser', value: 76 },
  { _id: 'Boardpins', value: 75 },
  { _id: 'Cutter', value: 68 },
  { _id: 'Pen', value: 55.666666666666664 },
  { _id: 'Marker', value: 50.666666666666664 }
```

```
[ { _id: 'Crayons', value: 40 } ]
```

**4. Perform Map Reduce to display the sum of quantities of records which have quality as ‘C’.**

```
inventory> db.inventory.mapReduce(map1, reduce1, {query:{quality:"C"}, out:"newdoc3"})
{ result: 'newdoc3', ok: 1 }
inventory> db.newdoc3.find()
[ { _id: 'Crayons', value: 40 },
  { _id: 'Notebook', value: 350 },
  { _id: 'Pencil', value: 680 }
]
```

**5. Perform Map Reduce to display the average of quantities of all records which have quality as ‘C’. Display only the first 2 records.**

```
inventory> db.inventory.mapReduce(map1, reduce2, {query:{quality:"C"}, out:"newdoc4",
limit:2})
{ result: 'newdoc4', ok: 1 }
inventory> db.newdoc4.find()
[ { _id: 'Notebook', value: 350 }, { _id: 'Pencil', value: 340 } ]
```

**6. Create a single index for quantities of items in ascending order.**

```
inventory> db.inventory.createIndex({quantity:1})
quantity_1
```

**7. Create an index to store quantities in descending order and item names in ascending order.**

```
inventory> db.inventory.createIndex({quantity:-1, itemname:1})
quantity_-1_itemname_1
```

**8. Create an index to store the unit of size and item name, both in ascending order and give it a name.**

```
inventory> db.inventory.createIndex({'size.unit':1, itemname:1}, {name:"Size and Name Index"})
Size and Name Index
```

**9. Show all index.**

```
inventory> db.inventory.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { quantity: 1 }, name: 'quantity_1' },
  {
    v: 2,
    key: { quantity: -1, itemname: 1 },
    name: 'quantity_-1_itemname_1'
  },
  {
    v: 2,
```

```

    key: { 'size.unit': 1, itemname: 1 },
    name: 'Size and Name Index'
}
]

```

### **10. Delete an index.**

```

inventory> db.inventory.dropIndex({quantity:1})
{ nIndexesWas: 4, ok: 1 }
inventory> db.inventory.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  {
    v: 2,
    key: { quantity: -1, itemname: 1 },
    name: 'quantity_-1_itemname_1'
  },
  {
    v: 2,
    key: { 'size.unit': 1, itemname: 1 },
    name: 'Size and Name Index'
  }
]

```

US-TCS-604 – Big Data Analytics  
Practical 7 – MongoDB in Python

**Connect MongoDB in Python and perform CRUD operations.**

**Step 1: Install pymongo.**

```
pip install pymongo
```

**Step 2: Run the following:**

```

import pymongo
myclient = pymongo.MongoClient("mongodb://127.0.0.1:27017/")

mydbbase = myclient["mydb"]
print("DB Created")

#List all dbs in mongodb
print(myclient.list_database_names())

dblist=myclient.list_database_names()
if "mydb" in dblist:
    print("Database exists")
else:
    print("Not existing")

```

```

#Add data and check
mydata = mydb["customers"]
print(mydb.list_collection_names())
#Insert data
dict1 = {"name": "Krishi", "address": "Ghatkopar"}
a = mydata.insert_one(dict1)
print(a.inserted_id)

DB Created
['admin', 'cars', 'config', 'inventory', 'kcc', 'letssee', 'local', 'mydb']
Database exists
['customers']
63cd107bdd7f1081071ef3af

b = mydata.find_one()
print(b)
{'_id': ObjectId('63cd107bdd7f1081071ef3af'), 'name': 'Krishi', 'address': 'Ghatkopar'}

list1 = [{"name": "Krishi", "address": "Ghatkopar"}, {"name": "Sabahat", "address": "Mumbra"}, {"name": "Disha", "address": "Andheri"}, {"name": "Yash", "address": "Lower Parel"}, {"name": "Ansh", "address": "Goregaon"}, {"name": "Krishi", "address": "Ghatkopar"}]
c = mydata.insert_many(list1)
print("Records inserted successfully: " + str(c.inserted_ids))
Records inserted successfully: [ObjectId('63cd108ddd7f1081071ef3b0'), ObjectId('63cd108ddd7f1081071ef3b1'), ObjectId('63cd108ddd7f1081071ef3b2'), ObjectId('63cd108ddd7f1081071ef3b3'), ObjectId('63cd108ddd7f1081071ef3b4'), ObjectId('63cd108ddd7f1081071ef3b5')]

print("Records:\n")
for data in mydata.find():
    print(data)

Records:
{'_id': ObjectId('63cd107bdd7f1081071ef3af'), 'name': 'Krishi', 'address': 'Ghatkopar'}
{'_id': ObjectId('63cd108ddd7f1081071ef3b0'), 'name': 'Krishi', 'address': 'Ghatkopar'}
{'_id': ObjectId('63cd108ddd7f1081071ef3b1'), 'name': 'Sabahat', 'address': 'Mumbra'}
{'_id': ObjectId('63cd108ddd7f1081071ef3b2'), 'name': 'Disha', 'address': 'Andheri'}
{'_id': ObjectId('63cd108ddd7f1081071ef3b3'), 'name': 'Yash', 'address': 'Lower Parel'}
{'_id': ObjectId('63cd108ddd7f1081071ef3b4'), 'name': 'Ansh', 'address': 'Goregaon'}
{'_id': ObjectId('63cd108ddd7f1081071ef3b5'), 'name': 'Krishi', 'address': 'Ghatkopar'}

myq = {"address": "Ghatkopar"}
mydoc1 = mydata.find(myq)
print("\nAddress: ")
for data in mydoc1:

```

```
print(data)
```

Address:

```
{'_id': ObjectId('63cd107bdd7f1081071ef3af'), 'name': 'Krishi', 'address': 'Ghatkopar'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b0'), 'name': 'Krishi', 'address': 'Ghatkopar'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b5'), 'name': 'Krishi', 'address': 'Ghatkopar'}
```

```
sorted1 = mydata.find().sort("name")  
print("\n\nSorted:\n\n")  
for data in sorted1:  
    print(data)
```

Sorted:

```
{'_id': ObjectId('63cd108ddd7f1081071ef3b4'), 'name': 'Ansh', 'address': 'Goregaon'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b2'), 'name': 'Disha', 'address': 'Andheri'}  
{'_id': ObjectId('63cd107bdd7f1081071ef3af'), 'name': 'Krishi', 'address': 'Ghatkopar'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b0'), 'name': 'Krishi', 'address': 'Ghatkopar'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b5'), 'name': 'Krishi', 'address': 'Ghatkopar'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b1'), 'name': 'Sabahat', 'address': 'Mumbra'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b3'), 'name': 'Yash', 'address': 'Lower Parel'}
```

```
del1 = {"name": "Krishi"}  
d1 = mydata.delete_one(del1)  
  
for data in mydata.find():  
    print(data)  
  
print("Records deleted: " + str(d1.deleted_count))  
  
del1 = {"name": "Krishi"}  
d1 = mydata.delete_many(del1)  
  
for data in mydata.find():  
    print(data)  
  
print("Records deleted: " + str(d1.deleted_count))  
  
{'_id': ObjectId('63cd108ddd7f1081071ef3b0'), 'name': 'Krishi', 'address': 'Ghatkopar'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b1'), 'name': 'Sabahat', 'address': 'Mumbra'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b2'), 'name': 'Disha', 'address': 'Andheri'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b3'), 'name': 'Yash', 'address': 'Lower Parel'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b4'), 'name': 'Ansh', 'address': 'Goregaon'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b5'), 'name': 'Krishi', 'address': 'Ghatkopar'}
```

Records deleted: 1

```
{'_id': ObjectId('63cd108ddd7f1081071ef3b1'), 'name': 'Sabahat', 'address': 'Mumbra'}
```

```
{'_id': ObjectId('63cd108ddd7f1081071ef3b2'), 'name': 'Disha', 'address': 'Andheri'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b3'), 'name': 'Yash', 'address': 'Lower Parel'}  
{'_id': ObjectId('63cd108ddd7f1081071ef3b4'), 'name': 'Ansh', 'address': 'Goregaon'}
```

Records deleted: 2

US-TCS-604 – Big Data Analytics  
Practical 8 – MongoDB in Java

1. Create a Java project in Eclipse without the module-info.java module.
2. Right click on the project in the Package Explorer and go to Properties.
3. In the Java Build Path section, go to Libraries tab.
4. Select **classpath** and click on Add External Jars.
5. Add both the drivers: **mongodb-driver-core-3.11.2.jar** and **mongo-java-driver-3.11.2.jar**
6. Now, Create a new Class (here: MongoConnection)

**ESTABLISHING CONNECTION TO MONGODB AND CREATING A COLLECTION**

**Code:**

```
import com.mongodb.client.MongoDatabase;  
import com.mongodb.MongoClient;  
  
public class MongoConnection {  
  
    public static void main(String[] args) {  
        MongoClient mongo = new MongoClient("localhost", 27017);  
        System.out.println("Established connection successfully.");  
        MongoDatabase database = mongo.getDatabase("mydb");  
        System.out.println("Connected to database mydb.");  
        database.createCollection("customers");  
        System.out.println("Collection customers created.");  
        mongo.close();  
    }  
}
```

**Output:**

```
Jan 24, 2023 10:03:39 AM com.mongodb.diagnostics.logging.JULLoader log  
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE,  
requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms',  
maxWaitQueueSize=500}
```

**Established connection successfully.**

**Connected to database mydb.**

```
Jan 24, 2023 10:03:39 AM com.mongodb.diagnostics.logging.JULLoader log  
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out  
Jan 24, 2023 10:03:39 AM com.mongodb.diagnostics.logging.JULLoader log
```

```

INFO: Opened connection [connectionId{localValue:1, serverValue:1}] to localhost:27017
Jan 24, 2023 10:03:39 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description
ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED,
ok=true, version=ServerVersion{versionList=[6, 0, 3]}, minWireVersion=0,
maxWireVersion=17, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30,
roundTripTimeNanos=1984900}
Jan 24, 2023 10:03:39 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:2}] to localhost:27017
Collection customers created.

```

## LISTING AND DROPPING COLECTIONS

**Code:**

```

import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import com.mongodb.MongoClient;

public class MongoCollections {

    public static void main(String args[]) {
        MongoClient mongo = new MongoClient( "localhost" , 27017 );
        System.out.println("Created Mongo Connection successfully");
        MongoDatabase database = mongo.getDatabase("mydb");
        System.out.println("Connected to Database.");

        database.createCollection("customers2");
        System.out.println("Collection named 'customers2' created successfully.");

        System.out.println("Collections present in this Database are:");
        for (String name : database.listCollectionNames()) {
            System.out.println(name);
        }

        MongoCollection<Document> collection = database.getCollection("customers2");
        System.out.println("Collection named 'customers2' selected successfully.");

        collection.drop();
        System.out.println("Collection named 'customers2' dropped successfully.");

        System.out.println("Collections present in this Database are:");
        for (String name : database.listCollectionNames()) {
            System.out.println(name);
        }
        mongo.close();

    }
}

```

## **Output:**

```
Jan 24, 2023 10:59:22 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE,
requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms',
maxWaitQueueSize=500}
```

**Created Mongo Connection successfully  
Connected to Database.**

```
Jan 24, 2023 10:59:22 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Jan 24, 2023 10:59:22 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:19}] to localhost:27017
Jan 24, 2023 10:59:22 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description
ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED,
ok=true, version=ServerVersion{versionList=[6, 0, 3]}, minWireVersion=0,
maxWireVersion=17, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30,
roundTripTimeNanos=2100600}
```

```
Jan 24, 2023 10:59:22 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:20}] to localhost:27017
```

**Collection named 'customers2' created successfully.**

**Collections present in this Database are:**

**customers**

**customers2**

**Collection named 'customers2' selected successfully.**

**Collection named 'customers2' dropped successfully.**

**Collections present in this Database are:**

**customers**

```
Jan 24, 2023 10:59:22 AM com.mongodb.diagnostics.logging.JULLogger log
```

```
INFO: Closed connection [connectionId{localValue:2, serverValue:20}] to localhost:27017
because the pool has been closed.
```

## **INSERTING RECORDS**

### **Code:**

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import java.util.ArrayList;
import java.util.List;
import org.bson.Document;
import com.mongodb.MongoClient;

public class MongoInsertRecords {

    public static void main(String[] args) {
        MongoClient mongo = new MongoClient( "localhost" , 27017 );
        System.out.println("Created Mongo Connection successfully");
        MongoDatabase database = mongo.getDatabase("mydb");
```

```

        System.out.println("Connected to Database.");
        MongoCollection<Document> collection =
database.getCollection("customers");
        System.out.println("Collection 'customers' is selected successfully.");
        Document document1 = new Document("name", "Krishi").append("address",
"Ghatkopar");
        collection.insertOne(document1);
        System.out.println("One record inserted successfully.");
        List<Document> list = new ArrayList<Document>();
        Document document2 = new Document("name",
"Sabahat").append("address", "Mumbra");
        Document document3 = new Document("name", "Nidhi").append("address",
"Chembur");
        Document document4 = new Document("name", "Praditi").append("address",
"Kurla");
        Document document5 = new Document("name", "Priti").append("address",
"Thane");
        list.add(document2);
        list.add(document3);
        list.add(document4);
        list.add(document5);
        collection.insertMany(list);
        System.out.println("Multiple records inserted successfully.");
        mongo.close();
    }
}

```

### **Output:**

Jan 24, 2023 10:28:01 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE,  
requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms',  
maxWaitQueueSize=500}

**Created Mongo Connection successfully**

**Connected to Database.**

**Collection 'customers' is selected successfully.**

Jan 24, 2023 10:28:01 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out  
Jan 24, 2023 10:28:01 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Opened connection [connectionId{localValue:1, serverValue:6}] to localhost:27017  
Jan 24, 2023 10:28:01 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Monitor thread successfully connected to server with description  
ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED,  
ok=true, version=ServerVersion{versionList=[6, 0, 3]}, minWireVersion=0,

```

maxWireVersion=17, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30,
roundTripTimeNanos=3152400}
Jan 24, 2023 10:28:01 AM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:7}] to localhost:27017
One record inserted successfully.
Multiple records inserted successfully.

```

## RETRIEVING RECORDS

**Code:**

```

import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCursor;
import java.util.Iterator;
import org.bson.Document;
import com.mongodb.BasicDBObject;
import com.mongodb.MongoClient;

public class MongoRetrieve {

    public static void main(String[] args) {
        MongoClient mongo = new MongoClient( "localhost" , 27017 );
        System.out.println("Created Mongo Connection successfully");
        MongoDatabase database = mongo.getDatabase("mydb");
        System.out.println("Connected to Database.");

        MongoCollection<Document> collection =
database.getCollection("customers");
        System.out.println("Collection 'customers' is selected successfully.");

        FindIterable<Document> iterDoc = collection.find();
        int i = 1;
        System.out.println("Records in Customers collection:");
        Iterator it = iterDoc.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
            i++;
        }

        BasicDBObject searchQuery = new BasicDBObject();
        searchQuery.put("name", "Nidhi");
        System.out.println("Retrieving specific Mongo Document named Nidhi");
        MongoCursor<Document> cursor = collection.find(searchQuery).iterator();
        while (cursor.hasNext()) {
            System.out.println(cursor.next());
        }

        mongo.close();
    }
}

```

```
    }
}
```

### **Output:**

```
Jan 24, 2023 10:39:34 AM com.mongodb.diagnostics.logging.JULLLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE,
requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms',
maxWaitQueueSize=500}
```

**Created Mongo Connection successfully**

**Connected to Database.**

**Collection 'customers' is selected successfully.**

**Records in Customers collection:**

```
Jan 24, 2023 10:39:34 AM com.mongodb.diagnostics.logging.JULLLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Jan 24, 2023 10:39:34 AM com.mongodb.diagnostics.logging.JULLLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:10}] to localhost:27017
Jan 24, 2023 10:39:34 AM com.mongodb.diagnostics.logging.JULLLogger log
INFO: Monitor thread successfully connected to server with description
ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED,
ok=true, version=ServerVersion{versionList=[6, 0, 3]}, minWireVersion=0,
maxWireVersion=17, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30,
roundTripTimeNanos=1987600}
Jan 24, 2023 10:39:34 AM com.mongodb.diagnostics.logging.JULLLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:11}] to localhost:27017
Document{{_id=63cf6559f05b306a9b7edf72, name=Krishi, address=Ghatkopar}}
Document{{_id=63cf6559f05b306a9b7edf73, name=Sabahat, address=Mumbra}}
Document{{_id=63cf6559f05b306a9b7edf74, name=Nidhi, address=Chembur}}
Document{{_id=63cf6559f05b306a9b7edf75, name=Praditi, address=Kurla}}
Document{{_id=63cf6559f05b306a9b7edf76, name=Priti, address=Thane}}
Retrieving specific Mongo Document named Nidhi
Document{{_id=63cf6559f05b306a9b7edf74, name=Nidhi, address=Chembur}}
Jan 24, 2023 10:39:34 AM com.mongodb.diagnostics.logging.JULLLogger log
INFO: Closed connection [connectionId{localValue:2, serverValue:11}] to localhost:27017
because the pool has been closed.
```

## **UPDATING RECORDS**

### **Code:**

```
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Updates;
import com.mongodb.client.MongoCursor;
import java.util.Iterator;
import org.bson.Document;
import com.mongodb.BasicDBObject;
import com.mongodb.MongoClient;
```

```

public class MongoUpdate {

    public static void main(String[] args) {
        MongoClient mongo = new MongoClient( "localhost" , 27017 );
        System.out.println("Created Mongo Connection successfully");
        MongoDB database = mongo.getDatabase("mydb");
        System.out.println("Connected to Database.");

        MongoCollection<Document> collection =
database.getCollection("customers");
        System.out.println("Collection 'customers' is selected successfully.");

        BasicDBObject searchQuery = new BasicDBObject();
        searchQuery.put("address", "Ghatkopar");
        System.out.println("Retrieving specific Mongo Document where address is
Ghatkopar");
        MongoCursor<Document> cursor = collection.find(searchQuery).iterator();
        while (cursor.hasNext()) {
            System.out.println(cursor.next());
        }

        collection.updateOne(Filters.eq("name", "Priti"), Updates.set("address",
"Ghatkopar"));
        System.out.println("Document with name Priti has been updated successfully.");

        searchQuery.put("address", "Ghatkopar");
        System.out.println("Retrieving specific Mongo Document where address is
Ghatkopar");
        cursor = collection.find(searchQuery).iterator();
        while (cursor.hasNext()) {
            System.out.println(cursor.next());
        }

        mongo.close();
    }
}

```

### **Output:**

Jan 24, 2023 10:46:51 AM com.mongodb.diagnostics.logging.JULLoader log  
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE,  
requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms',  
maxWaitQueueSize=500}

**Created Mongo Connection successfully**

**Connected to Database.**

**Collection 'customers' is selected successfully.**

**Retrieving specific Mongo Document where address is Ghatkopar**

```

Jan 24, 2023 10:46:51 AM com.mongodb.diagnostics.logging.JULLoader log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Jan 24, 2023 10:46:51 AM com.mongodb.diagnostics.logging.JULLoader log
INFO: Opened connection [connectionId{localValue:1, serverValue:15}] to localhost:27017
Jan 24, 2023 10:46:51 AM com.mongodb.diagnostics.logging.JULLoader log
INFO: Monitor thread successfully connected to server with description
ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED,
ok=true, version=ServerVersion{versionList=[6, 0, 3]}, minWireVersion=0,
maxWireVersion=17, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30,
roundTripTimeNanos=2863000}
Jan 24, 2023 10:46:51 AM com.mongodb.diagnostics.logging.JULLoader log
INFO: Opened connection [connectionId{localValue:2, serverValue:16}] to localhost:27017
Document{{_id=63cf6559f05b306a9b7edf72, name=Krishi, address=Ghatkopar}}
Document with name Priti has been updated successfully.
Retrieving specific Mongo Document where address is Ghatkopar
Document{{_id=63cf6559f05b306a9b7edf72, name=Krishi, address=Ghatkopar}}
Document{{_id=63cf6559f05b306a9b7edf76, name=Priti, address=Ghatkopar}}
Jan 24, 2023 10:46:51 AM com.mongodb.diagnostics.logging.JULLoader log
INFO: Closed connection [connectionId{localValue:2, serverValue:16}] to localhost:27017
because the pool has been closed.

```

## DELETING RECORDS/DOCUMENTS

**Code:**

```

import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCursor;
import java.util.Iterator;
import org.bson.Document;
import com.mongodb.MongoClient;

public class MongoDelete {

    public static void main(String[] args) {
        MongoClient mongo = new MongoClient( "localhost" , 27017 );
        System.out.println("Created Mongo Connection successfully");
        MongoDatabase database = mongo.getDatabase("mydb");
        System.out.println("Connected to Database.");

        MongoCollection<Document> collection =
database.getCollection("customers");
        System.out.println("Collection 'customers' is selected successfully.");

        FindIterable<Document> iterDoc = collection.find();
        int i = 1;

        System.out.println("Records in Customers collection:");
        Iterator it = iterDoc.iterator();
    }
}

```

```

        while (it.hasNext()) {
            System.out.println(it.next());
            i++;
        }

        collection.deleteOne(Filters.eq("name", "Krishi"));
        System.out.println("Document named 'Krishi' deleted successfully.");
        // Getting the iterable object
        iterDoc = collection.find();
        i = 1;

        System.out.println("Records in Customers collection:");
        it = iterDoc.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
            i++;
        }
    }

    mongo.close();

}

```

### **Output:**

Jan 24, 2023 10:54:32 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}

**Created Mongo Connection successfully**

**Connected to Database.**

**Collection 'customers' is selected successfully.**

**Records in Customers collection:**

Jan 24, 2023 10:54:32 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out  
Jan 24, 2023 10:54:32 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Opened connection [connectionId{localValue:1, serverValue:17}] to localhost:27017  
Jan 24, 2023 10:54:32 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Monitor thread successfully connected to server with description  
ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[6, 0, 3]}, minWireVersion=0, maxWireVersion=17, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=2228400}

Jan 24, 2023 10:54:32 AM com.mongodb.diagnostics.logging.JULLLogger log  
INFO: Opened connection [connectionId{localValue:2, serverValue:18}] to localhost:27017  
**Document{{\_id=63cf6559f05b306a9b7edf72, name=Krishi, address=Ghatkopar}}**  
**Document{{\_id=63cf6559f05b306a9b7edf73, name=Sabahat, address=Mumbra}}**  
**Document{{\_id=63cf6559f05b306a9b7edf74, name=Nidhi, address=Chembur}}**

**Document{{\_id=63cf6559f05b306a9b7edf75, name=Praditi, address=Kurla}}**

**Document{{\_id=63cf6559f05b306a9b7edf76, name=Priti, address=Ghatkopar}}**

**Document named 'Krishi' deleted successfully.**

**Records in Customers collection:**

**Document{{\_id=63cf6559f05b306a9b7edf73, name=Sabahat, address=Mumbra}}**

**Document{{\_id=63cf6559f05b306a9b7edf74, name=Nidhi, address=Chembur}}**

**Document{{\_id=63cf6559f05b306a9b7edf75, name=Praditi, address=Kurla}}**

**Document{{\_id=63cf6559f05b306a9b7edf76, name=Priti, address=Ghatkopar}}**

Jan 24, 2023 10:54:32 AM com.mongodb.diagnostics.logging.JULLoader log

INFO: Closed connection [connectionId{localValue:2, serverValue:18}] to localhost:27017 because the pool has been closed.