Part – B (MongoDB)

Lab-9 **createcollection, dropcollection and insert method**

Perform following queries using use, drop, createcollection, dropcollection, insertOne, insertMany and find method.

Part - A

1. Create a new database named “Darshan”.
2. Create another new database named “DIET”.
3. List all databases.
4. Check the current database.
5. Drop “DIET” database.
6. Create a collection named “Student” in the “Darshan” database.
7. Create a collection named “Department” in the “Darshan” database.
8. List all collections in the “Darshan” database.
9. Insert a single document using insertOne into “Department” collection. (Dname:’CE’, HOD:’Patel’)
10. Insert two document using insertMany into “Department” collection. (Dname:’IT’ and Dname:’ICT’)
11. Drop a collection named “Department” from the “Darshan” database.
12. Insert a single document using insertOne into “Student” collection.
    (Fields are Name, City, Branch, Semester, Age) Insert your own data.
13. Insert three documents using insertMany into “Student” collection.
    (Fields are Name, City, Branch, Semester, Age) Insert your three friend’s data.
14. Check whether “Student” collection exists or not.
15. Check the stats of “Student” collection.
16. Drop the “Student” collection.
17. Create a collection named “Deposit”.
18. Insert following data in to “Deposit” collection.

Deposit

ACTNO	CNAME	BNAME	AMOUNT	CITY
101	ANIL	VRCE	1000.00	RAJKOT
102	SUNIL	AJNI	5000.00	SURAT
103	MEHUL	KAROLBAGH	3500.00	BARODA
104	MADHURI	CHANDI	1200.00	AHMEDABAD
105	PRMOD	M.G. ROAD	3000.00	SURAT
106	SANDIP	ANDHERI	2000.00	RAJKOT
107	SHIVANI	VIRAR	1000.00	SURAT
108	KRANTI	NEHRU PLACE	5000.00	RAJKOT

19. Display all the documents of “Deposit” collection.
20. Drop the “Deposit” collection.

Part – B

1. Create a new database named “Computer”.
2. Create a collection named “Faculty” in the “Computer” database.
3. Insert a below document using insertOne into “Faculty” collection.

Faculty

FID	FNAME	BNAME	SALARY	JDATE
1	ANIL	CE	10000	1-3-95

4. Insert below documents using insertMany into “Faculty” collection.

Faculty				
FID	FNAME	BNAME	SALARY	JDATE
2	SUNIL	CE	50000	4-1-96
3	MEHUL	IT	35000	17-11-95
4	MADHURI	IT	12000	17-12-95
5	PRMOD	CE	30000	27-3-96
6	SANDIP	CE	20000	31-3-96
7	SHIVANI	CE	10000	5-9-95
8	KRANTI	IT	50000	2-7-95

5. Display all the documents of "Faculty" collection.
6. Drop the "Faculty" collection.
7. Drop the "Computer" database.

Part – C (Perform following operation using UI)

1. Create a new database named "Computer".
2. Create a collection named "Faculty" in the "Computer" database.
3. Insert a below documents into "Faculty" collection.

Faculty				
FID	FNAME	BNAME	SALARY	JDATE
1	ANIL	CE	10000	1-3-95
2	SUNIL	CE	50000	4-1-96
3	MEHUL	IT	35000	17-11-95
4	MADHURI	IT	12000	17-12-95
5	PRMOD	CE	30000	27-3-96
6	SANDIP	CE	20000	31-3-96
7	SHIVANI	CE	10000	5-9-95
8	KRANTI	IT	50000	2-7-95

4. Display all the documents of "Faculty" collection.
5. Drop the "Faculty" collection.
6. Drop the "Computer" database.

Lab-10 — **Find, limit, skip and sort method**

Mongo DB

Perform following queries using find, limit, skip and sort method.

Create and Select Database Named: "BANK_INFO" and create below Deposit collection with 8 Documents.

Deposit (Collection name)				
ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000	1-3-95
102	SUNIL	AJNI	5000	4-1-96
103	MEHUL	KAROLBAGH	3500	17-11-95
104	MADHURI	CHANDI	1200	17-12-95
105	PRMOD	M.G. ROAD	3000	27-3-96
106	SANDIP	ANDHERI	2000	31-3-96
107	SHIVANI	VIRAR	1000	5-9-95
108	KRANTI	NEHRU PLACE	5000	2-7-95

From the above given collection perform the following queries using find, limit, skip and sort method:

Part - A

1. Retrieve/Display every document of Deposit collection.

2. Display only one document of Deposit collection. (Use: findOne())
3. Insert following document into Deposit collection. (Use: insertOne())

109	KIRTI	VIRAR	3000	3-5-97

4. Insert following documents into Deposit collection. (Use: insertMany())

110	MITALI	ANDHERI	4500	4-9-95
111	RAJIV	NEHRU PLACE	7000	2-10-98

5. Display all the documents of 'VIRAR' branch from Deposit collection.
6. Display all the documents of Deposit collection whose amount is between 3000 and 5000.
7. Display all the documents of Deposit collection whose amount is greater than 2000 and branch is VIRAR.
8. Display all the documents with CNAME, BNAME and AMOUNT fields from Deposit collection.
9. Display all the documents of Deposit collection in ascending order by CNAME.
10. Display all the documents of Deposit collection in descending order by BNAME.
11. Display all the documents of Deposit collection in ascending order by ACTNO and descending order by AMOUNT.
12. Display only first two documents of Deposit collection.
13. Display 3rd document of Deposit collection.
14. Display 6th and 7th documents of Deposit collection.
15. Display the count of documents in Deposit collection.

Part- B

1. Insert following documents into "Student" collection. (Use: insertMany())

```
{ "_id": 1, "name": "John", "age": 30, "city": "New York", "isActive": true }
{ "_id": 2, "name": "Jane", "age": 25, "city": "Los Angeles", "isActive": false }
{ "_id": 3, "name": "Tom", "age": 35, "city": "Chicago", "isActive": true }
{ "_id": 4, "name": "Lucy", "age": 28, "city": "San Francisco", "isActive": true }
{ "_id": 5, "name": "David", "age": 40, "city": "Miami", "isActive": false }
{ "_id": 6, "name": "Eva", "age": 23, "city": "Boston", "isActive": true }
{ "_id": 7, "name": "Nick", "age": 38, "city": "Seattle", "isActive": false }
{ "_id": 8, "name": "Sophia", "age": 27, "city": "New York", "isActive": true }
{ "_id": 9, "name": "Liam", "age": 32, "city": "Los Angeles", "isActive": false }
{ "_id": 10, "name": "Olivia", "age": 29, "city": "San Diego", "isActive": true }
```

2. Display all documents of "Student" collection.
3. Display all documents of "Student" collection whose age is 30.
4. Display all documents of "Student" collection whose age is greater than 25.
5. Display all documents of "Student" collection whose name is "John" and age is 30.
6. Display all documents of "Student" collection whose age is not equal to 25.
7. Display all documents of "Student" collection whose age is equal to 25 or 30 or 35. (using $or as well as using $in).
8. Display all documents of "Student" collection whose name is "John" or age is 30.
9. Display all documents of "Student" collection whose name is "John" and city is New York.
10. Display name and age of students from "Student" collection whose name is "John" and city is New York.

Part – C

1. Display name of students from "Student" collection whose age is between to 25 and 35 and sort output by age in ascending order.
2. Display all documents of "Student" collection and sort all the documents by name in ascending order and then by age in descending order.

	3. Display first five documents of "Student" collection. 4. Display fourth and fifth documents of "Student" collection. 5. Display the name of oldest student from "Student" collection. 6. Display all documents of "Student" collection in such a way that skip the first 2 documents and return the rest documents.

Lab-11	**update, delete and rename method**
Mongo DB	**Perform the following queries using update, delete, rename and createcollection method:** **Part – A (Use collection "Student" created in Lab-9)** 1. Update the age of John's to 31. 2. Update the city of all students from 'New York' to 'New Jersey'. 3. Set isActive to false for every student older than 35. 4. Increment the age of all students by 1 year. 5. Set the city of 'Eva' to 'Cambridge'. 6. Update 'Sophia's isActive status to false. 7. Update the city field of student aged below 30 to 'San Diego'. 8. Rename the age field to years for all documents. 9. Update 'Nick' to make him active (isActive = true). 10. Update all documents to add a new field country with the value 'USA'. 11. Update 'David's city to 'Orlando'. 12. Multiply the age of all students by 2. 13. Unset (remove) the city field for 'Tom'. 14. Add a new field premiumUser and make it to true for users older than 30. 15. Set isActive to true for 'Jane'. 16. Update isActive field of 'Lucy' to false. 17. Delete a document of 'Nick' from the collection. 18. Delete all students who are inactive (isActive = false). 19. Delete all students who live in 'New York'. 20. Delete all the students aged above 35. 21. Delete a student named ''Olivia'' from the student collection. 22. Delete all the students whose age is below 25. 23. Delete the first student whose isActive field is true. 24. Delete all students from 'Los Angeles'. 25. Delete all students who have city field missing. 26. Rename 'city' field to 'location' for all documents. 27. Rename the name field to FullName for 'John'. 28. Rename the isActive field to status for all documents. 29. Rename age to yearsOld for everyone from 'San Francisco' student only. 30. Create a Capped Collection named "Employee" which allows maximum 3 documents: Insert following documents into above "Employee" collection and check how many documents are there in above "Employee" collection. {"Ecode": 1, "Ename": "John"} {"Ecode ": 2, "Ename": "Jane", "age": 25, "city": "Los Angeles"} {"Ecode ": 3, "Ename": "Tom", "age": 35} {"Ecode ": 4, "Ename": "Lucy", "age": 28, "city": "San Francisco", "isActive": true}

{"Ename": "Dino"}

Part – B Create collection named "Student_data" and insert following 10 documents into it.

Student_data						
ROLLNO	**SNAME**	**DEPARTMENT**	**FEES**	**SEM**	**GENDER**	**CITY**
101	Vina	CE	15000	3	Female	Rajkot
102	Krisha	EC	8000	5	Female	Ahmedabad
103	Priti	Civil	12000	7	Female	Baroda
104	Mitul	CE	15000	3	Male	Rajkot
105	Keshav	CE	15000	3	Male	Jamnagar
106	Zarna	Civil	12000	5	Female	Ahmedabad
107	Nima	EE	9000	5	Female	Rajkot
108	Dhruv	Mechanical	10000	5	Male	Rajkot
109	Krish	Mechanical	10000	7	Male	Baroda
110	Zeel	EE	9000	3	Female	Jamnagar

From the above given "Student_data" collection perform the following queries:

1. Display Female students and belong to Rajkot city.
2. Display students not studying in 3rd sem.
3. Display students whose city is Jamnagar or Baroda. (use: IN)
4. Display first 2 students names who lives in Baroda.
5. Display Male students who studying in 3rd sem.
6. Display sname and city and fees of those students whose roll no is less than 105.
7. Update City of all students from 'Jamnagar' City and Department as 'CE' to 'Surat'.
8. Increase Fees by 500 where the Gender is not 'Female'. (Use: Not)
9. Set the Department of all students from 'EE' and in Sem 3 to 'Electrical'.
10. Update the Fees of students from 'Rajkot' city who are male to 15000.
11. Change City to 'Vadodara' for students in Sem 5 and with fees less than 10000.
12. Delete all students where the City is 'Ahmedabad' or GENDER is 'Male'.
13. Delete students whose Rollno is not in the list [101, 105, 110].
14. Delete students from the 'Civil' department who are in Sem 5 or Sem 7.
15. Delete all students who are not from cities 'Rajkot', 'Baroda', or 'Jamnagar'.
16. Delete students whose Rollno is between 105 and 108.
17. Rename the City field to LOCATION for all students.
18. Rename the Department field to Branch where the Fees is less than 10000.
19. Rename Sname to Fullname for students with Rollno in [106, 107, 108].
20. Rename Fees to Tuition_Fees for all students with Fees greater than 9000.
21. Rename Department to Major where the Fees is less than 15000 and Gender is 'Female'.
22. Rename City to Hometown for all students whose SEM is 3 and Department is not 'Mechanical'.

Part – C

1. Create a capped collection named" logs" with a maximum size of 100 KB and a maximum of 10 documents.
2. Insert below 12 log entries into the "logs" collection. Each entry should contain a message, level (e.g., "info", "warning", "error"), and a timestamp field. Use the insertMany() method.
   { message: "System started", level: "info", timestamp: new Date() }
   { message: "Disk space low", level: "warning", timestamp: new Date() }
   { message: "User login", level: "info", timestamp: new Date() }

{ message: "System reboot", level: "info", timestamp: new Date() }
{ message: "Error in module", level: "error", timestamp: new Date() }
{ message: "Memory usage high", level: "warning", timestamp: new Date() }
{ message: "User logout", level: "info", timestamp: new Date() }
{ message: "File uploaded", level: "info", timestamp: new Date() }
{ message: "Network error", level: "error", timestamp: new Date() }
{ message: "Backup completed", level: "info", timestamp: new Date() }
{ message: "Database error", level: "error", timestamp: new Date() }
{ message: "Service started", level: "info", timestamp: new Date() }

3. Perform find method on "logs" collection to ensure only the **last 10 documents** are retained (even though you inserted 12).
4. Insert below 5 more documents and check if the oldest ones are automatically removed.

{ message: "New log entry 1", level: "info", timestamp: new Date() }
{ message: "New log entry 2", level: "info", timestamp: new Date() }
{ message: "New log entry 3", level: "info", timestamp: new Date() }
{ message: "New log entry 4", level: "warning", timestamp: new Date() }
{ message: "New log entry 5", level: "error", timestamp: new Date() }

Lab-12 — **regex method**

Mongo DB

Perform the following queries using Regex:
Part – A Create collection named "Employee" and insert following 10 documents into it.

employee

EID	ENAME	GENDER	JOININGDATE	SALARY	CITY
1	Nick	Male	01-JAN-13	4000	London
2	Julian	Female	01-OCT-14	3000	New York
3	Roy	Male	01-JUN-16	3500	London
4	Tom	Male	NULL	4500	London
5	Jerry	Male	01-FEB-13	2800	Sydney
6	Philip	Male	01-JAN-15	7000	New York
7	Sara	Female	01-AUG-17	4800	Sydney
8	Emily	Female	01-JAN-15	5500	New York
9	Michael	Male	NULL	6500	London
10	John	Male	01-JAN-15	8800	London

1. Find employees whose name start with E.
2. Find employees whose name ends with n.
3. Find employees whose name starts with S or M in your collection.
4. Find employees where city starts with A to M in your collection.
5. Find employees where city name ends in 'ney'.
6. Display employee info whose name contains n. (Both uppercase(N) and lowercase(n))
7. Display employee info whose name starts with E and having 5 characters.
8. Display employee whose name start with S and ends in a.
9. Display EID, ENAME, CITY and SALARY whose name starts with 'Phi'.
10. Display ENAME, JOININGDATE and CITY whose city contains 'dne' as three letters in city name.
11. Display ENAME, JOININGDATE and CITY who does not belongs to city London or Sydney.
12. Find employees whose names start with 'J'.

13. Find employees whose names end with 'y'.
14. Find employees whose names contain the letter 'a'.
15. Find employees whose names contain either 'a' or 'e'.
16. Find employees whose names start with 'J' and end with 'n'.
17. Find employees whose CITY starts with 'New'.
18. Find employees whose CITY does not start with 'L'
19. Find employees whose CITY contains the word 'York'.
20. Find employees whose names have two consecutive vowels (a, e, i, o, u).
21. Find employees whose names have three or more letters.
22. Find employees whose names have exactly 4 letters.
23. Find employees whose names start with either 'S' or 'M'.
24. Find employees whose names contain 'il' anywhere.
25. Find employees whose names do not contain 'a'.
26. Find employees whose names contain any digit.
27. Find employees whose names contain exactly one vowel.
28. Find employees whose names start with any uppercase letter followed by any lowercase letter.

Part – B Create collection named "Student" and insert following 10 documents into it.

Student						
ROLLNO	**SNAME**	**DEPARTMENT**	**FEES**	**SEM**	**GENDER**	**CITY**
101	Vina	CE	15000	3	Female	Rajkot
102	Krisha	EC	8000	5	Female	Ahmedabad
103	Priti	Civil	12000	7	Female	Baroda
104	Mitul	CE	15000	3	Male	Rajkot
105	Keshav	CE	15000	3	Male	Jamnagar
106	Zarna	Civil	12000	5	Female	Ahmedabad
107	Nima	EE	9000	5	Female	Rajkot
108	Dhruv	Mechanical	10000	5	Male	Rajkot
109	Krish	Mechanical	10000	7	Male	Baroda
110	Zeel	EE	9000	3	Female	Jamnagar

1. Display documents where sname start with K.
2. Display documents where sname starts with Z or D.
3. Display documents where city starts with A to R.
4. Display students' info whose name start with P and ends with i.
5. Display students' info whose department name starts with 'C'.
6. Display name, sem, fees, and department whose city contains 'med' as three letters somewhere in city name.
7. Display name, sem, fees, and department who does not belongs to city Rajkot or Baroda.
8. Find students whose names start with 'K' and are followed by any character.
9. Find students whose names end with 'a'.
10. Find students whose names contain 'ri'. (case-insensitive)
11. Find students whose names start with a vowel (A, E, I, O, U).
12. Find students whose CITY ends with 'pur' or 'bad'.
13. Find students whose FEES starts with '1'.
14. Find students whose SNAME starts with 'K' or 'V'.
15. Find students whose CITY contains exactly five characters.

16. Find students whose names do not contain the letter 'e'.
17. Find students whose CITY starts with 'Ra' and ends with 'ot'.
18. Find students whose names contain exactly one vowel.
19. Find students whose names start and end with the same letter.
20. Find students whose DEPARTMENT starts with either 'C' or 'E'.
21. Find students whose SNAME has exactly 5 characters.
22. Find students whose GENDER is Female and CITY starts with 'A'.

Part – C Create collection named "Company" and insert following 10 documents into it.

CID	CNAME	EMAIL	INDUSTRY	CITY	EMPLOYEES
1	TechNova	info@technova.com	IT	Bangalore	250
2	GreenWorld	support@greenw.com	Agriculture	Ahmedabad	120
3	SkyHigh Ltd	contact@skyhigh.in	Aviation	Mmumbai	300
4	UrbanBuild	info@urbanbuild.org	Construction	Surat	180
5	MediCore	hello@medicore.net	Healthcare	Pune	90
6	FinEdge	info@finedge.co	Finance	Kolkata	200
7	AutoSphere	sales@autos.com	Automotive	Chennai	400
8	EduQuest	info@eduquest.edu	Education	Rajkot	75
9	FoodiesHub	contact@foodies.org	Food	Baroda	60
10	BioPure	info@biopure.bio	Pharma	Hyderabad	150

1. Find companies whose name starts with 'B' or 'F'
2. Find companies located in cities ending with 'pur'
3. Find companies whose name contains the word "Core" Find companies with email addresses starting with "info"
4. Find companies whose INDUSTRY starts with a capital letter followed by 4 lowercase letters
5. Find companies whose CNAME ends with a capital letter
6. Find companies whose CITY starts with any letter from A to K
7. Find companies whose INDUSTRY name has more than 8 letters
8. Find companies whose EMAIL has a number in it
9. Find companies whose name starts and ends with vowels
10. Find companies with CITY names that contain the same letter twice in a row
11. Find companies whose email starts with any two letters followed by digits
12. Find companies whose EMAIL includes an underscore _
13. Find companies whose EMAIL domain (after @) starts with 'g' or 'h'
14. Find companies whose CNAME contains a repeating pattern like "ana", "ele", etc.
15. Find companies whose CNAME contains at least 3 vowels
16. Find companies whose EMAIL domain ends with '.com' and starts with 'out'
17. Find companies whose INDUSTRY does not contain any vowels
18. Find companies whose CNAME contains two or more consecutive consonants
19. Find companies whose CNAME has alternating vowels and consonants (at least 4 characters)
23. Find companies where CITY starts with two same letters (e.g., "Mehsana" doesn't match, but "AAhmedabad" would)

Lab-13 **Aggregate method**

Mongo DB

Perform the following queries using Aggregate:

Part – A Create collection named "Student" and insert following 10 documents into it.

Student						
ROLLNO	**SNAME**	**DEPARTMENT**	**FEES**	**SEM**	**GENDER**	**CITY**
101	Vina	CE	15000	3	Female	Rajkot
102	Krisha	EC	8000	5	Female	Ahmedabad
103	Priti	Civil	12000	7	Female	Baroda
104	Mitul	CE	15000	3	Male	Rajkot
105	Keshav	CE	15000	3	Male	Jamnagar
106	Zarna	Civil	12000	5	Female	Ahmedabad
107	Nima	EE	9000	5	Female	Rajkot
108	Dhruv	Mechanical	10000	5	Male	Rajkot
109	Krish	Mechanical	10000	7	Male	Baroda
110	Zeel	EE	9000	3	Female	Jamnagar

1. Display distinct city.
2. Display city wise count of number of students.
3. Display sum of fees in your collection.
4. Display average of fees in your collection.
5. Display maximum and minimum fees of your collection.
6. Display city wise total fees in your collection.
7. Display gender wise maximum fees in your collection.
8. Display city wise maximum and minimum fees.
9. Display count of persons lives in Baroda city in your collection.
10. Display average fees of Rajkot city.
11. Count the number of male and female students in each Department
12. Find the total Fees collected from each Department.
13. Find the minimum Fees paid by male and female students in each City.
14. Sort students by Fees in descending order and return the top 5.
15. Group students by City and calculate the average Fees for each city, only including cities with avg fees more than 12000.
16. Filter students from CE or Mechanical department, then calculate the total Fees.
17. Group students by City, Department and Gender and calculate the max, min and average Fees.
18. Filter students from Rajkot, then group by Department and find the average Fees for each department.
19. Group by Sem and calculate both the total and average Fees, then sort by total fees in descending order.
20. Find the top 3 cities with the highest total Fees collected by summing up all students' fees in those cities.

Part – B

1. Create a collection named" Stock."
2. Insert below 9 documents into the "Stock" collection.

```
{  "_id": 1,
   "company": "Company-A",
   "sector": "Technology",
   "eps": 5.2,
   "pe": 15.3,
   "roe": 12.8,
```

```
    "sales": 300000,
    "profit": 25000
  }
  {  "_id": 2,
    "company": "Company-B",
    "sector": "Finance",
    "eps": 7.1,
    "pe": 12.4,
    "roe": 10.9,
    "sales": 500000,
    "profit": 55000
  }
  {  "_id": 3,
    "company": "Company-C",
    "sector": "Retail",
    "eps": 3.8,
    "pe": 22.1,
    "roe": 9.5,
    "sales": 200000,
    "profit": 15000
  }
  {  "_id": 4,
    "company": "Company-D",
    "sector": "Technology",
    "eps": 5.2,
    "pe": 15.3,
    "roe": 12.8,
    "sales": 300000,
    "profit": 25000
  }
  {  "_id": 5,
    "company": "Company-E",
    "sector": "Finance",
    "eps": 7.1,
    "pe": 12.4,
    "roe": 10.9,
    "sales": 450000,
    "profit": 40000
  }
  {  "_id": 6,
    "company": "Company-F",
    "sector": "Healthcare",
    "eps": 3.8,
    "pe": 18.9,
    "roe": 9.5,
    "sales": 500000,
    "profit": 35000
```

```
    }
    {   "_id": 7,
        "company": "Company-G",
        "sector": "Retail",
        "eps": 4.3,
        "pe": 22.1,
        "roe": 14.2,
        "sales": 600000,
        "profit": 45000
    }
    {
        "_id": 8,
        "company": "Company-H",
        "sector": "Energy",
        "eps": 6.5,
        "pe": 10.5,
        "roe": 16.4,
        "sales": 550000,
        "profit": 50000
    }
    {
        "_id": 9,
        "company": "Company-I",
        "sector": "Consumer Goods",
        "eps": 2.9,
        "pe": 25.3,
        "roe": 7.8,
        "sales": 350000,
        "profit": 20000
    }
```


3. Calculate the total sales of all companies.
4. Find the average profit for companies in each sector.
5. Get the count of companies in each sector.
6. Find the company with the highest PE ratio.
7. Filter companies with PE ratio greater than 20.(Use: Aggregate)
8. Calculate the total profit of companies with sales greater than 250,000.
9. Project only the company name and profit fields.(Use: Aggregate)
10. Find companies where EPS is greater than the average EPS.
11. Group companies by sector and get the maximum sales in each sector.
12. Calculate the total sales and total profit of companies in each sector.
13. Sort companies by profit in descending order.(Use: Aggregate)
14. Find the average ROE across all companies.
15. Group companies by sector and calculate both the minimum and maximum EPS.

Part – C

1. Count the number of companies with profit greater than 30,000.
2. Get the total profit by sector and sort by descending order by total profit.
3. Find the top 3 companies with the highest sales.

	4. Calculate the average PE ratio of companies grouped by sector. 5. Get the sum of sales and profit for each company. 6. Find companies with sales less than 400,000 and sort them by sales in descending order. 7. Group companies by sector and find the total number of companies in each sector. 8. Get the average ROE for companies with sales greater than 200,000. 9. Find the maximum profit in each sector. 10. Get the total sales and count of companies in each sector. 11. Project fields where profit is more than 20,000 and only show company and profit. 12. Find companies with the lowest ROE and sort them in ascending order.(Use: Aggregate)

Lab-14	**Aggregate method**
Mongo DB	**Perform the following queries using Aggregate:** **Part – A Create following collection and perform query on Aggregate.** **Student:** [{ "_id": 1, "name": "Amit", "dept_id": 101, "marks": 82 }, { "_id": 2, "name": "Rina", "dept_id": 102, "marks": 91 }, { "_id": 3, "name": "Karan", "dept_id": 101, "marks": 76 }, { "_id": 4, "name": "Meera", "dept_id": 103, "marks": 88 }, { "_id": 5, "name": "Iqbal", "dept_id": 102, "marks": 67 }] **Department:** [{ "_id": 101, "dept_name": "Computer Science" }, { "_id": 102, "dept_name": "Electronics" }, { "_id": 103, "dept_name": "Mechanical" }] **Fees:** [{ "_id": 1, "student_id": 1, "amount": 25000, "status": "Paid" }, { "_id": 2, "student_id": 2, "amount": 30000, "status": "Unpaid" }, { "_id": 3, "student_id": 3, "amount": 25000, "status": "Paid" }, { "_id": 4, "student_id": 4, "amount": 35000, "status": "Paid" }] 1. Fetches each student and attaches matching department details using $lookup. 2. Joins students with departments and displays only the student's name and their department name. 3. Retrieves every student along with their corresponding fee information. 4. Finds students whose fee status is marked as "Unpaid". 5. Fetches students with both their department details and fee records in a single pipeline. 6. Returns only those students for whom no matching department exists. 7. Selects students whose department name is "Computer Science" using join + match. 8. Performs a join with departments and sorts all students by their marks in descending order. 9. Counts how many students belong to each department after joining. 10. Joins with departments and filters students whose marks are greater than 80. 11. Finds the maximum marks obtained within each department using join + group. 12. Returns students whose fee status is "Paid" using a join on the fees collection. 13. Joins students with fee records and calculates total fee amount per student. 14. Fetches each department along with all students enrolled in that department.

	15. Returns only those departments which do not have any students mapped to them. **Part – B** 1. Returns the maximum marks scored by any student. 2. Calculates the average marks of students separately for each department. 3. Computes the sum of all fee amounts paid by students. 4. Counts how many students have scored marks ≥ 80. 5. Finds the lowest marks within each department. **Part – C** 1. Arranges all student records in descending order of marks. 2. Divides students into buckets based on marks: 0–60, 60–80, and 80–100. 3. Calculates the total (sum) of marks obtained by students in each department. 4. Finds the average amount of fee recorded in the fees collection. 5. Compares each student's marks with the average marks of their department.

Lab-15	**Index, Cursor and Schema Validation**
Mongo DB	**Perform the following queries on Index, Cursor, Schema Validation, Embedded and Multivalued Documents:** **Part – A (Use collection "Stock" created in Lab-12)** 1. Create an index on the company field in the stocks collection. 2. Create a compound index on the sector and sales fields in the stocks collection. 3. List all the indexes created on the stocks collection. 4. Drop an existing index on the company field from the stocks collection. 5. Use a cursor to retrieve and iterate over documents in the stocks collection, displaying each document. 6. Limit the number of documents returned by a cursor to the first 3 documents in the stocks collection. 7. Sort the documents returned by a cursor in descending order based on the sales field. 8. Skip the first 2 documents in the result set and return the next documents using the cursor. 9. Convert the cursor to an array and return all documents from the stocks collection. 10. Create a collection named "Companies" with schema validation to ensure that each document must contains a company field (string) and a sector field (string). **Part – B** 1. Create a collection named "Scripts" with validation for fields like eps, pe, and roe to ensure that they are numbers and required/compulsory fields. 2. Create a collection named "Products" where each product has an embedded document for manufacturer details and a multivalued field for categories that stores an array of category names the product belongs to. • manufacturer details: The manufacturer will be an embedded document with fields like name, country, and establishedYear. • categories: The categories will be an array field that holds multiple values. (i.e. Electronics, Mobile, Smart Devices). **Part – C** 1. Create a collection named "financial_Reports" that requires revenue (a positive number) but allows optional fields like expenses and netIncome (if provided, they should also be numbers). 2. Create a collection named "Student" where each student has name and address are embedded document and mobilenumber and emailaddress are multivalued field that stores an array of values.