ASSIGNMENT – 6
Advanced Database Systems LAB
Name: Utsav Balar
Roll no: T24CS003

1. Write a MongoDB query to find all students excluding _id and LastName Field
2. Update the student's age with Rollnum 67 to 26 years.
3. Increase the Age of all students by 1
4. Increment 5 Mark for all students in the "Computer Science" department.
5. Reduce 10 Marks for all students in the "Mathematics" department.
6. Update the Department name of all students in the "Physics" department to “Physical Science”.
7. Calculate the average age of all students.
8. Calculate the average Mark of all students in the "Physical Science" department.
9. Remove all the students of the "Mathematics" department.
10. Remove all the students whose mark is less than 60.

Code:

```
extern crate mongodb;
use futures_util::TryStreamExt;
use mongodb::bson::{doc, Document};
use mongodb::{options::ClientOptions, Client};
use serde::{Deserialize, Serialize};
use termion::terminal_size;

#[derive(Serialize, Deserialize)]
struct Student {
    roll_num: usize,
    first_name: String,
    last_name: String,
    age: usize,
    department: String,
    marks: usize,
}

impl Student {
```

```
    fn new(
        roll_num: usize,
        first_name: String,
        last_name: String,
        age: usize,
        department: String,
        marks: usize,
    ) -> Self {
        Self {
            roll_num,
            first_name,
            last_name,
            age,
            department,
            marks,
        }
    }
}

async fn insert_student(client: &Client, db_name: &str, doc:
Student, coll_name: &str) {
    let db = client.database(db_name);
    let coll = db.collection(coll_name);

    coll.insert_one(doc).await.unwrap();
}

async fn get_filtered_documents(
    client: &Client,
    db_name: &str,
    coll_name: &str,
    filter: Document,
    projection: Option<Document>,
) {
    let db = client.database(db_name);
    let coll: mongodb::Collection<Document> =
db.collection(coll_name);

    let mut cursor = coll
        .find(filter)
        .projection({
            if let Some(projection) = projection {
                projection
            } else {
                doc! {"_id": 0}
            }
```

```
        })
        .await
        .unwrap();

    if let Ok((width, _height)) = terminal_size() {
        println!("{}", "-".repeat(width as usize));
    }

    while let Some(doc) = cursor.try_next().await.unwrap() {
        println!("{}", doc);
    }

    if let Ok((width, _height)) = terminal_size() {
        println!("{}", "-".repeat(width as usize));
    }
    println!();
}

async fn update_document(
    client: &Client,
    db_name: &str,
    coll_name: &str,
    filter: Document,
    update: Document,
    multiple: bool,
) {
    let db = client.database(db_name);
    let coll: mongodb::Collection<Document> =
db.collection(coll_name);

    if !multiple {
        coll.update_one(filter, update)
            .await
            .expect("Failed to update document");
    } else {
        coll.update_many(filter, update)
            .await
            .expect("Failed to update document");
    }
}

async fn calculate_average(
    client: &Client,
    db_name: &str,
    coll_name: &str,
    avg_name: &str,
```

```
    pipeline: Vec<Document>,
) → Option<f64> {
    let db = client.database(db_name);
    let coll: mongodb::Collection<Document> =
db.collection(coll_name);

    let mut cursor = coll
        .aggregate(pipeline)
        .await
        .expect("Failed to aggregate documents");

    cursor
        .try_next()
        .await
        .unwrap()
        .map(|result| result.get_f64(avg_name).unwrap())
}

async fn delete_document(
    client: &Client,
    db_name: &str,
    coll_name: &str,
    filter: Document,
    multiple: bool,
) {
    let db = client.database(db_name);
    let coll: mongodb::Collection<Document> =
db.collection(coll_name);

    if !multiple {
        coll.delete_one(filter).await.unwrap();
    } else {
        coll.delete_many(filter).await.unwrap();
    }
}

fn generate_students_data() → Vec<Student> {
    vec![
        Student::new(
            43,
            "John".to_string(),
            "Doe".to_string(),
            20,
            "Computer Science".to_string(),
            78,
        ),
```

```
        Student::new(
            67,
            "Alice".to_string(),
            "Smith".to_string(),
            22,
            "Physics".to_string(),
            59,
        ),
        Student::new(
            23,
            "Bob".to_string(),
            "Johnson".to_string(),
            21,
            "Computer Science".to_string(),
            81,
        ),
        Student::new(
            18,
            "Eve".to_string(),
            "Adams".to_string(),
            19,
            "Mathematics".to_string(),
            56,
        ),
        Student::new(
            84,
            "Mike".to_string(),
            "Brown".to_string(),
            23,
            "Physics".to_string(),
            92,
        ),
    ]
}

#[tokio::main]
async fn main() {
    let client_options =
ClientOptions::parse("mongodb://localhost:27017")
        .await
        .expect("ClientOptions failed to parse");
    let client =
Client::with_options(client_options).expect("Failed to create
client");
    let db_name: &str = "t24cs004_lab5";
    let db = client.database(db_name);
```

```
    let collection: &str = "cs553";

    db.create_collection(collection)
        .await
        .expect("Failed to create collection");

    let students: Vec<Student> = generate_students_data();

    for student in students {
        insert_student(&client, db_name, student,
collection).await;
    }

    println!("Write a MongoDB query to find all students excluding
_id and LastName Field");
    let filter = doc! {};
    let projection = doc! {"_id": 0, "last_name": 0};
    get_filtered_documents(&client, db_name, collection, filter,
Some(projection)).await;

    println!("2. Update the student's age with Rollnum 67 to 26
years.");
    let filter = doc! {"roll_num": 67};
    let update = doc! {"$set": {"age": 26}};
    update_document(&client, db_name, collection, filter.clone(),
update, false).await;
    get_filtered_documents(&client, db_name, collection,
filter.clone(), None).await;

    println!("3. Increase the Age of all students by 1");
    let filter = doc! {};
    let update = doc! {"$inc": {"age": 1}};
    update_document(&client, db_name, collection, filter.clone(),
update, true).await;
    get_filtered_documents(&client, db_name, collection,
filter.clone(), None).await;

    println!("4. Increment 5 Mark for all students in the
\"Computer Science\" department.");
    let filter = doc! {"department": "Computer Science"};
    let update = doc! {"$inc": {"marks": 5}};
    update_document(&client, db_name, collection, filter.clone(),
update, true).await;
    get_filtered_documents(&client, db_name, collection,
filter.clone(), None).await;
```

```
    println!("5. Reduce 10 Marks for all students in the
\"Mathematics\" department.");
    let filter = doc! {"department": "Mathematics"};
    let update = doc! {"$inc": {"marks": -10}};
    update_document(&client, db_name, collection, filter.clone(),
update, true).await;
    get_filtered_documents(&client, db_name, collection,
filter.clone(), None).await;

    println!("6. Update the Department name of all students in the
\"Physics\" department to \"Physical Science\"");
    let filter = doc! {"department": "Physics"};
    let update = doc! {"$set": {"department": "Physical Science"}};
    update_document(&client, db_name, collection, filter, update,
true).await;
    let filter = doc! {"department": "Physical Science"};
    get_filtered_documents(&client, db_name, collection, filter,
None).await;

    println!("7. Calculate the average age of all students.");
    let pipeline = vec![doc! { "$group": { "_id": null,
"average_age": { "$avg": "$age" }}}];
    let average_age =
        calculate_average(&client, db_name, collection,
"average_age", pipeline).await;

    if let Some(average_age) = average_age {
        if let Ok((width, _height)) = terminal_size() {
            println!("{}", "-".repeat(width as usize));
        }
        println!("Average age of all students: {}", average_age);
        if let Ok((width, _height)) = terminal_size() {
            println!("{}", "-".repeat(width as usize));
        }
        println!();
    }

    println!(
        "8. Calculate the average Mark of all students in the
\"Physical Science\" department."
    );
    let pipeline = vec![
        doc! { "$match": { "department": "Physical Science" }},
        doc! { "$group": { "_id": null, "average_marks": { "$avg":
"$marks" }}},
    ];
```

```
    let average_marks =
        calculate_average(&client, db_name, collection,
"average_marks", pipeline).await;

    if let Some(average_marks) = average_marks {
        if let Ok((width, _height)) = terminal_size() {
            println!("{}", "-".repeat(width as usize));
        }
        println!(
            "Average marks of all students in the \"Physical
Science\" department: {}",
            average_marks
        );
        if let Ok((width, _height)) = terminal_size() {
            println!("{}", "-".repeat(width as usize));
        }
        println!();
    }

    println!("9. Remove all the students of the \"Mathematics\"
department.");
    let filter = doc! {"department": "Mathematics"};
    delete_document(&client, db_name, collection, filter,
true).await;

    let filter = doc! {};
    get_filtered_documents(&client, db_name, collection, filter,
None).await;

    println!("10. Remove all the students whose mark is less than
60.");
    let filter = doc! {"marks": {"$lt": 60}};
    delete_document(&client, db_name, collection, filter,
true).await;

    let filter = doc! {};
    get_filtered_documents(&client, db_name, collection, filter,
None).await;

    db.drop().await.expect("Failed to drop database");
}
```

Results:

```
Write a MongoDB query to find all students excluding _id and LastName Field
{ "roll_num": 43, "first_name": "John", "age": 20, "department": "Computer Science", "marks": 78 }
{ "roll_num": 67, "first_name": "Alice", "age": 22, "department": "Physics", "marks": 59 }
{ "roll_num": 23, "first_name": "Bob", "age": 21, "department": "Computer Science", "marks": 81 }
{ "roll_num": 18, "first_name": "Eve", "age": 19, "department": "Mathematics", "marks": 56 }
{ "roll_num": 84, "first_name": "Mike", "age": 23, "department": "Physics", "marks": 92 }

2. Update the student's age with Rollnum 67 to 26 years.
{ "roll_num": 67, "first_name": "Alice", "last_name": "Smith", "age": 26, "department": "Physics", "marks": 59 }

3. Increase the Age of all students by 1
{ "roll_num": 43, "first_name": "John", "last_name": "Doe", "age": 21, "department": "Computer Science", "marks": 78 }
{ "roll_num": 67, "first_name": "Alice", "last_name": "Smith", "age": 27, "department": "Physics", "marks": 59 }
{ "roll_num": 23, "first_name": "Bob", "last_name": "Johnson", "age": 22, "department": "Computer Science", "marks": 81 }
{ "roll_num": 18, "first_name": "Eve", "last_name": "Adams", "age": 20, "department": "Mathematics", "marks": 56 }
{ "roll_num": 84, "first_name": "Mike", "last_name": "Brown", "age": 24, "department": "Physics", "marks": 92 }

4. Increment 5 Mark for all students in the "Computer Science" department.
{ "roll_num": 43, "first_name": "John", "last_name": "Doe", "age": 21, "department": "Computer Science", "marks": 83 }
{ "roll_num": 23, "first_name": "Bob", "last_name": "Johnson", "age": 22, "department": "Computer Science", "marks": 86 }

5. Reduce 10 Marks for all students in the "Mathematics" department.
{ "roll_num": 18, "first_name": "Eve", "last_name": "Adams", "age": 20, "department": "Mathematics", "marks": 46 }

6. Update the Department name of all students in the "Physics" department to "Physical Science"
{ "roll_num": 67, "first_name": "Alice", "last_name": "Smith", "age": 27, "department": "Physical Science", "marks": 59 }
{ "roll_num": 84, "first_name": "Mike", "last_name": "Brown", "age": 24, "department": "Physical Science", "marks": 92 }

7. Calculate the average age of all students.
Average age of all students: 22.8

8. Calculate the average Mark of all students in the "Physical Science" department.
Average marks of all students in the "Physical Science" department: 75.5

9. Remove all the students of the "Mathematics" department.
{ "roll_num": 43, "first_name": "John", "last_name": "Doe", "age": 21, "department": "Computer Science", "marks": 83 }
{ "roll_num": 67, "first_name": "Alice", "last_name": "Smith", "age": 27, "department": "Physical Science", "marks": 59 }
{ "roll_num": 23, "first_name": "Bob", "last_name": "Johnson", "age": 22, "department": "Computer Science", "marks": 86 }
{ "roll_num": 84, "first_name": "Mike", "last_name": "Brown", "age": 24, "department": "Physical Science", "marks": 92 }

10. Remove all the students whose mark is less than 60.
{ "roll_num": 43, "first_name": "John", "last_name": "Doe", "age": 21, "department": "Computer Science", "marks": 83 }
{ "roll_num": 23, "first_name": "Bob", "last_name": "Johnson", "age": 22, "department": "Computer Science", "marks": 86 }
{ "roll_num": 84, "first_name": "Mike", "last_name": "Brown", "age": 24, "department": "Physical Science", "marks": 92 }
```