

**SLIP 4 – PyMongo Operations (FULL STEPS)**

**A. Create Python File**

```
cd ~  
mkdir slip4  
cd slip4  
nano slip4.py
```

**B. Paste Code**

```
from pymongo import MongoClient
```

```
# 1. Connect to MongoDB
```

```
client = MongoClient('mongodb://localhost:27017/')  
db = client['company_db']  
employees = db['employees']
```

```
# 2. Insert 3 employee documents
```

```
employee_data = [  
    {"name": "John", "department": "IT", "salary": 60000},  
    {"name": "Alice", "department": "HR", "salary": 45000},  
    {"name": "Bob", "department": "Finance", "salary": 75000}  
]  
employees.insert_many(employee_data)
```

```
# 3. Retrieve records where salary > 50,000
```

```
print("Employees with salary > 50,000:")  
for emp in employees.find({"salary": {"$gt": 50000}}):  
    print(emp)
```

```
# 4. Update one record and display all
```

```
employees.update_one({"name": "Alice"}, {"$set": {"salary": 52000}})
```

```
print("\nAll Employees:")
```

```
for emp in employees.find():  
    print(emp)
```

Save → **CTRL+O** → **ENTER** → **CTRL+X**

**C. Run the Program**

```
python3 slip4.py
```

---

**SLIP 5 – Hive Basic Querying (FULL STEPS)**

**A. Start Hadoop & Hive**

```
start-all.sh  # If single node Hadoop  
hive
```

**B. Create Table**

```

CREATE TABLE movies (
    title STRING,
    type STRING,
    release_year INT,
    country STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

```

#### **C. Keep CSV File and Load Data**

1. Create a CSV file:

```
cd ~
```

```
mkdir hive_slip5
```

```
cd hive_slip5
```

```
nano movies.csv
```

Paste example:

```
Inception,Movie,2010,USA
```

```
Dangal,Movie,2016,India
```

```
Friends,TV Show,1994,USA
```

```
3 Idiots,Movie,2009,India
```

```
Interstellar,Movie,2014,USA
```

Save.

2. Load into Hive:

```

LOAD DATA LOCAL INPATH '/home/yourusername/hive_slip5/movies.csv'
INTO TABLE movies;

```

#### **D. Queries**

##### **1. No. of Movies per Country**

```

SELECT country, COUNT(*) AS movie_count
FROM movies
GROUP BY country
ORDER BY movie_count DESC;

```

##### **2. Top 5 Recent Release Years**

```

SELECT release_year, COUNT(*) AS movie_count
FROM movies
GROUP BY release_year
ORDER BY release_year DESC
LIMIT 5;

```

---

#### **SLIP 6 – Hive Sorting & Aggregation (FULL STEPS)**

##### **A. Create Table**

```

CREATE TABLE sales_data (
    region STRING,

```

```
product STRING,  
amount DOUBLE  
)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '\t';
```

#### B. Create & Load Sample Data

```
cd ~  
mkdir hive_slip6  
cd hive_slip6  
nano sales.txt
```

Paste sample:

Region	Product	Amount
North	Laptop	50000
South	TV	40000
East	Fridge	30000
West	Mobile	45000
North	Mobile	20000

Save.

Load to Hive:

```
LOAD DATA LOCAL INPATH '/home/yourusername/hive_slip6/sales.txt'  
INTO TABLE sales_data;
```

#### C. Queries

##### 1. Total Sales per Region

```
SELECT region, SUM(amount) AS total_sales  
FROM sales_data  
GROUP BY region;
```

##### 2. Sort by Total Sales Descending

```
SELECT region, SUM(amount) AS total_sales  
FROM sales_data  
GROUP BY region  
ORDER BY total_sales DESC;
```

#### SLIP 7 – Hive Joins & Filtering (FULL STEPS)

#### A. Start Hive

```
hive
```

#### B. Create Tables

```
CREATE TABLE customers (  
    cust_id INT,  
    name STRING,  
    city STRING  
)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ',';
```

```
CREATE TABLE orders (
    order_id INT,
    cust_id INT,
    amount DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

#### C. Prepare & Load Data

Create data files:

```
cd ~
mkdir hive_slip7
cd hive_slip7
```

```
nano customers.csv
```

Paste:

```
1,John,Mumbai
2,Alice,Delhi
3,Bob,Pune
```

Save.

```
nano orders.csv
```

Paste:

```
101,1,5000
102,2,7000
103,1,3000
104,3,6500
```

Save.

Load into Hive:

```
LOAD DATA LOCAL INPATH '/home/yourusername/hive_slip7/customers.csv' INTO TABLE customers;
LOAD DATA LOCAL INPATH '/home/yourusername/hive_slip7/orders.csv' INTO TABLE orders;
```

#### D. Query – Total Order Amount per Customer

```
SELECT c.name, SUM(o.amount) AS total_amount
FROM customers c
JOIN orders o ON c.cust_id = o.cust_id
GROUP BY c.name
ORDER BY total_amount DESC;
```

---

#### SLIP 8 – Hive UDF (Convert Title to Uppercase) – FULL STEPS

**NOTE:** UDF requires a Java class file. Most exams don't ask for actual Java code execution.  
I'll give the easiest working version that passes in practicals.

#### A. Create Java File

```
cd ~
mkdir hive_slip8
```

```
cd hive_slip8  
nanoUpperCaseUDF.java  
Paste:  
  
import org.apache.hadoop.hive.ql.exec.UDF;  
import org.apache.hadoop.io.Text;  
  
public classUpperCaseUDF extends UDF {  
    public Text evaluate(Text input) {  
        if (input == null) return null;  
        return new Text(input.toString().toUpperCase());  
    }  
}
```

Save.

#### B. Compile & Create JAR

```
javac -cp $(hadoop classpath):$(hive --auxpath)UpperCaseUDF.java  
jar -cf uppercase-udf.jarUpperCaseUDF.class
```

#### C. Open Hive & Add JAR

```
ADD JAR /home/yourusername/hive_slip8/uppercase-udf.jar;
```

```
CREATE TEMPORARY FUNCTION uppercase AS 'UpperCaseUDF';
```

#### D. Use UDF on Movies Table

```
SELECT uppercase(title), type, release_year  
FROM movies;
```

---

### SLIP 9 – Pig Basic Operations (FULL STEPS)

#### A. Start Pig

Most labs: **Pig Local Mode**

```
pig -x local
```

#### B. Create Input File

```
cd ~  
mkdir pig_slip9  
cd pig_slip9  
nano students.txt  
Paste:  
John,67  
Alice,85  
Bob,72  
Meera,90  
Raj,55  
Save.
```

#### C. Pig Script

```
nano slip9.pig
```

Paste:

```
students = LOAD 'students.txt' USING PigStorage(',')
AS (name:chararray, marks:int);

good_students = FILTER students BY marks > 70;

result = FOREACH good_students GENERATE name, marks;

DUMP result;
```

Save.

**D. Run Script**

```
pig -x local slip9.pig
```

---

**SLIP 10 – Pig Grouping & Aggregation**

**A. Start Pig**

```
pig -x local
```

**B. Create Input File**

```
cd ~
```

```
mkdir pig_slip10
```

```
cd pig_slip10
```

```
nano sales_data.txt
```

Paste:

```
Electronics,Mobile,50000
```

```
Electronics,Laptop,70000
```

```
Clothing,Shirt,2000
```

```
Clothing,Jeans,3000
```

```
Grocery,Rice,1500
```

```
Grocery,Oil,2500
```

```
Electronics,TV,40000
```

Save.

**C. Pig Script**

```
nano slip10.pig
```

Paste:

```
sales = LOAD 'sales_data.txt' USING PigStorage(',')
AS (category:chararray, product:chararray, amount:double);
```

```
grouped_sales = GROUP sales BY category;
```

```
avg_sales = FOREACH grouped_sales
GENERATE group AS category, AVG(sales.amount) AS avg_amount;
```

```
DUMP avg_sales;
```

Save.

#### D. Run Script

```
pig -x local slip10.pig
```

---

#### SLIP 11 – Pig Join Operation

##### A. Create Files

```
cd ~
```

```
mkdir pig_slip11
```

```
cd pig_slip11
```

```
nano employee_details.txt
```

Paste:

```
1,John,101
```

```
2,Alice,103
```

```
3,Bob,102
```

```
4,David,101
```

Save.

```
nano department.txt
```

Paste:

```
101,IT
```

```
102,HR
```

```
103,Finance
```

Save.

##### B. Pig Script

```
nano slip11.pig
```

Paste:

```
employees = LOAD 'employee_details.txt' USING PigStorage(',')
```

```
AS (emp_id:int, name:chararray, dept_id:int);
```

```
departments = LOAD 'department.txt' USING PigStorage(',')
```

```
AS (dept_id:int, dept_name:chararray);
```

```
joined_data = JOIN employees BY dept_id, departments BY dept_id;
```

```
result = FOREACH joined_data GENERATE employees::name, departments::dept_name;
```

```
DUMP result;
```

Save.

#### C. Run Script

```
pig -x local slip11.pig
```

---

#### SLIP 12 – Pig Sorting & Filtering

##### A. Create File

```
cd ~  
mkdir pig_slip12  
cd pig_slip12  
nano movies.txt  
  
Paste:  
  
Inception,Movie,2010,8.8  
Friends,TV Show,1994,9.3  
Mirzapur,TV Show,2018,8.5  
Sacred Games,TV Show,2018,8.6  
Dangal,Movie,2016,8.4  
Stranger Things,TV Show,2016,8.7  
Breaking Bad,TV Show,2008,9.5  
Money Heist,TV Show,2017,8.3  
Dark,TV Show,2017,8.8  
The Office,TV Show,2005,8.9  
GOT,TV Show,2011,9.2  
  
Save.
```

#### B. Pig Script

```
nano slip12.pig  
  
Paste:  
  
movies = LOAD 'movies.txt' USING PigStorage(',')  
AS (title:chararray, type:chararray, release_year:int, rating:double);  
  
tv_shows = FILTER movies BY type == 'TV Show';  
  
sorted_shows = ORDER tv_shows BY release_year DESC;  
  
top_10 = LIMIT sorted_shows 10;
```

```
DUMP top_10;
```

```
Save.
```

#### C. Run Script

```
pig -x local slip12.pig
```

#### SLIP 1 – MongoDB CRUD Operations (FULL STEPS)

##### A. Start MongoDB & Open Mongo Shell

```
sudo service mongod start
```

```
mongo
```

##### B. Create DB, Collection & Insert 5 Documents

```
use student_db
```

```
db.Students.insertMany([  
  {name: "John", dept: "CS", marks: 85},  
  {name: "Mike", dept: "Math", marks: 90},  
  {name: "Sarah", dept: "Physics", marks: 88},  
  {name: "David", dept: "Chemistry", marks: 92},  
  {name: "Emily", dept: "Biology", marks: 89}])
```

```
{name: "Alice", dept: "IT", marks: 78},  
{name: "Bob", dept: "CS", marks: 92},  
{name: "Carol", dept: "ECE", marks: 65},  
{name: "David", dept: "IT", marks: 88}  
])
```

#### C. Update Marks

```
db.Students.updateOne(  
  {name: "Carol"},  
  {$set: {marks: 75}}  
)
```

#### D. Delete One Record

```
db.Students.deleteOne({name: "Bob"})
```

#### E. Display All Records

```
db.Students.find().pretty()
```

---

### SLIP 2 – Querying JSON Data in MongoDB (FULL STEPS)

#### A. Create JSON File

```
cd ~  
mkdir mongo_slip2  
cd mongo_slip2  
nano products.json
```

Paste the following sample JSON array:

```
[  
  {"name": "Laptop", "category": "Electronics", "price": 55000},  
  {"name": "Headphones", "category": "Electronics", "price": 2500},  
  {"name": "Shirt", "category": "Clothing", "price": 1200},  
  {"name": "Mobile", "category": "Electronics", "price": 18000},  
  {"name": "Shoes", "category": "Footwear", "price": 3000}
```

```
]
```

Save: **CTRL+O → ENTER → CTRL+X**

#### B. Import JSON into MongoDB

```
mongoimport --db product_db --collection products --file products.json --jsonArray
```

#### C. Open Mongo Shell and Run Queries

```
mongo  
use product_db
```

##### 1. Display Electronics Products

```
db.products.find({category: "Electronics"})
```

##### 2. Count Products Above ₹10,000

```
db.products.countDocuments({price: {$gt: 10000}})
```

---

### SLIP 3 – MongoDB Aggregation Pipeline (FULL STEPS)

#### A. Open Mongo Shell

```
mongo
use company_db

B. Insert Sample Data

db.employees.insertMany([
    {name: "John", department: "IT", salary: 60000},
    {name: "Alice", department: "HR", salary: 45000},
    {name: "Bob", department: "Finance", salary: 75000},
    {name: "Carol", department: "IT", salary: 72000},
    {name: "David", department: "Finance", salary: 65000}
])
```

**C. Aggregation: Avg Salary by Department (Descending)**

```
db.employees.aggregate([
    { $group: { _id: "$department", averageSalary: { $avg: "$salary" } } },
    { $sort: { averageSalary: -1 } }
])
```

---

**Part 1 Completed**

Would you like **Part 2 now** (Slips 4, 5, 6)?

Reply:  
NEXT to continue.

You said:

2

ChatGPT said:

Great — **Part 2 = Slips 4, 5, 6**  
As promised: all steps included (how to start, create files, run commands, etc.).

---

**SLIP 4 – PyMongo Operations (FULL STEPS)**

**A. Create Python File**

```
cd ~
mkdir slip4
cd slip4
nano slip4.py
```

**B. Paste Code**

```
from pymongo import MongoClient
```

```
# 1. Connect to MongoDB
client = MongoClient('mongodb://localhost:27017/')
db = client['company_db']
employees = db['employees']
```

```
# 2. Insert 3 employee documents
```

```
employee_data = [
    {"name": "John", "department": "IT", "salary": 60000},
```

```

        {"name": "Alice", "department": "HR", "salary": 45000},
        {"name": "Bob", "department": "Finance", "salary": 75000}
    ]
employees.insert_many(employee_data)

# 3. Retrieve records where salary > 50,000
print("Employees with salary > 50,000:")
for emp in employees.find({"salary": {"$gt": 50000}}):
    print(emp)

# 4. Update one record and display all
employees.update_one({"name": "Alice"}, {"$set": {"salary": 52000}})

print("\nAll Employees:")
for emp in employees.find():
    print(emp)
Save → CTRL+O → ENTER → CTRL+X

```

### C. Run the Program

python3 slip4.py

---

## SLIP 5 – Hive Basic Querying (FULL STEPS)

### A. Start Hadoop & Hive

```
start-all.sh # If single node Hadoop
hive
```

### B. Create Table

```
CREATE TABLE movies (
    title STRING,
    type STRING,
    release_year INT,
    country STRING
)
```

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

### C. Keep CSV File and Load Data

1. Create a CSV file:

```
cd ~
mkdir hive_slip5
cd hive_slip5
nano movies.csv
```

Paste example:

Inception,Movie,2010,USA

Dangal,Movie,2016,India

Friends,TV Show,1994,USA

3 Idiots,Movic,2009,India

Interstellar,Movie,2014,USA

Save.

2. Load into Hive:

```
LOAD DATA LOCAL INPATH '/home/yourusername/hive_slip5/movies.csv'  
INTO TABLE movies;
```

#### D. Queries

##### 1. No. of Movies per Country

```
SELECT country, COUNT(*) AS movie_count  
FROM movies  
GROUP BY country  
ORDER BY movie_count DESC;
```

##### 2. Top 5 Recent Release Years

```
SELECT release_year, COUNT(*) AS movie_count  
FROM movies  
GROUP BY release_year  
ORDER BY release_year DESC  
LIMIT 5;
```

---

#### SLIP 6 – Hive Sorting & Aggregation (FULL STEPS)

##### A. Create Table

```
CREATE TABLE sales_data (  
region STRING,  
product STRING,  
amount DOUBLE  
)
```

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t';

##### B. Create & Load Sample Data

cd ~

mkdir hive\_slip6

cd hive\_slip6

nano sales.txt

Paste sample:

North	Laptop	50000
South	TV	40000
East	Fridge	30000
West	Mobile	45000
North	Mobile	20000

Save.

Load to Hive:

```
LOAD DATA LOCAL INPATH '/home/yourusername/hive_slip6/sales.txt'  
INTO TABLE sales_data;
```

### C. Queries

#### 1. Total Sales per Region

```
SELECT region, SUM(amount) AS total_sales  
FROM sales_data  
GROUP BY region;
```

#### 2. Sort by Total Sales Descending

```
SELECT region, SUM(amount) AS total_sales  
FROM sales_data  
GROUP BY region  
ORDER BY total_sales DESC;
```