Hands-On Lab: Querying & Aggregations (DBS Tech Bank)

Lab Objectives

By the end of this session, learners will:

- Understand the **Aggregation Pipeline Basics**
- Use **$match, $group, $project, $sort, $limit**
- Filter & transform data using the pipeline
- Build a **Customer Transaction Summary** (Case Study)
- Understand all key pipeline operators with **real DBS Bank examples**

Step 0 — Setup

```
use dbs_tech_bank;
```

Verify:

```
show collections;
```

You should see:

```
customers
accounts
transactions
products
```

Make sure your transactions collection already has sample data from previous labs.

1️⃣ Aggregation Pipeline Basics (Hands-On)

The Aggregation Pipeline works like **a conveyor belt**:

```
db.collection.aggregate([
  { Stage 1 },
  { Stage 2 },
  { Stage 3 }
])
```

Each stage transforms data.

We will run the following simple pipeline first:

Step 1: Basic Pipeline Example

```
db.transactions.aggregate([
  { $match: { type: "DEBIT" } },
  { $project: { _id: 0, custId: 1, accNo: 1, amount: 1 } },
  { $sort: { amount: -1 } },
  { $limit: 5 }
]).pretty();
```

👉 **What this does:**

Stage	Meaning	Purpose
$match	Filter	Get only DEBIT transactions
$project	Select/shape fields	Choose which fields to show
$sort	Order	Sort by amount (DESC)
$limit	Restrict	Keep top 5

Run it → check output.

2️⃣ **Understanding Each Aggregation Stage (Hands-On)**

Now we test each stage separately to understand them deeply.

2.1 $match – Filtering Documents (SQL WHERE)

Example 1: High-value DEBIT transactions (> ₹50,000)

```
db.transactions.aggregate([
  { $match: { type: "DEBIT", amount: { $gt: 50000 } } }
]).pretty();
```

Example 2: Transactions in 2025

```
db.transactions.aggregate([
  {
    $match: {
      timestamp: {
        $gte: ISODate("2025-01-01T00:00:00Z"),
        $lt: ISODate("2026-01-01T00:00:00Z")
      }
```

```
    }
  }
]).pretty();
```

💡 **Why $match first?**

It reduces the dataset → faster pipeline.

🔷 **2.2 $project – Selecting & Transforming Fields**

Example 1: Show only key output fields

```
db.transactions.aggregate([
  {
    $project: {
      _id: 0,
      txnId: 1,
      custId: 1,
      amount: 1,
      txnYear: { $year: "$timestamp" }
    }
  }
]).pretty();
```

Example 2: Add computed fields

```
db.transactions.aggregate([
  {
    $project: {
      custId: 1,
      amount: 1,
      isHighValue: { $gt: ["$amount", 100000] }
    }
  }
]).pretty();
```

2.3 $group – Grouping & Summaries (SQL GROUP BY)

Example: Total spent per customer

```
db.transactions.aggregate([
  {
    $group: {
      _id: "$custId",
      totalAmount: { $sum: "$amount" },
```

```
      totalTxns: { $sum: 1 }
    }
  }
]).pretty();
```

2.4 $sort – Ordering Results

Sort by transaction amount (DESC)

```
db.transactions.aggregate([
  { $sort: { amount: -1 } }
]).pretty();
```

2.5 $limit – Restrict Number of Records

Top 3 biggest transactions

```
db.transactions.aggregate([
  { $sort: { amount: -1 } },
  { $limit: 3 }
]).pretty();
```

Filtering & Transforming Data (Hands-On)

We combine $match, $project, and $addFields.

🔷 Step 3.1 – Filter + Transform

```
db.transactions.aggregate([
  {
    $match: {
      type: "DEBIT",
      amount: { $gt: 20000 }
    }
  },
  {
    $project: {
      _id: 0,
      custId: 1,
      accNo: 1,
      amount: 1,
      day: { $dayOfMonth: "$timestamp" },
      month: { $month: "$timestamp" },
      year: { $year: "$timestamp" }
    }
```

```
  }
]).pretty();
```

👉 **What you see:**

- Filter → only high-value DEBIT transactions
- Transform → extract date parts

🔷 **Step 3.2 – Add Flags**

```
db.transactions.aggregate([
  {
    $addFields: {
      highAmtFlag: { $gt: ["$amount", 100000] },
      digitalChannelFlag: { $in: ["$channel", ["UPI",
"NETBANKING"]] }
    }
  }
]).pretty();
```

👉 **This is how analytics pipelines flag risky transactions.**

Case Study (Main Lab) – Summarize Total Transactions per Customer

This is the **core Customer360 analytics** use case.

✔ **Goal**

For every customer:

- Count number of transactions
- Compute total amount
- Compute average transaction value
- Sort by total transaction value (high → low)

🧩 **Step-by-Step Aggregation Pipeline**

Step 1 — Filter POSTED transactions

```
{ $match: { status: "POSTED" } }
```

Step 2 — Group by customer

```
{
  $group: {
    _id: "$custId",
    totalAmount: { $sum: "$amount" },
    txnCount: { $sum: 1 }
  }
}
```

Step 3 — Format Output using

$project

```
{
  $project: {
    _id: 0,
    custId: "$_id",
    totalAmount: 1,
    txnCount: 1,
    avgAmount: { $divide: ["$totalAmount", "$txnCount"] }
  }
}
```

Step 4 — Sort by total amount (DESC)

```
{ $sort: { totalAmount: -1 } }
```

Step 5 — Limit top 5

```
{ $limit: 5 }
```

Full Case Study Pipeline

```
db.transactions.aggregate([
  // 1. Filter
  {
    $match: {
      status: "POSTED"
    }
  },

  // 2. Group by customer
  {
    $group: {
      _id: "$custId",
      totalAmount: { $sum: "$amount" },
      txnCount: { $sum: 1 }
```

```
    }
  },

  // 3. Shape output
  {
    $project: {
      _id: 0,
      custId: "$_id",
      totalAmount: 1,
      txnCount: 1,
      avgAmount: {
        $cond: {
          if: { $gt: ["$txnCount", 0] },
          then: { $divide: ["$totalAmount", "$txnCount"] },
          else: 0
        }
      }
    }
  },

  // 4. Order by largest spenders
  { $sort: { totalAmount: -1 } },

  // 5. Limit to top 5 customers
  { $limit: 5 }
]).pretty();
```

Interpretation (Explaining Output)

For each customer (e.g., C1001, C1002…):

- totalAmount: how much they spent (or total transaction volume)
- txnCount: number of transactions performed
- avgAmount: customer's average transaction value
- Sorted by highest totalAmount → identify **premium / high-value / risky** customers.

This is the **foundation of Customer360 dashboards** used by modern banks like DBS.

🧠 Bonus: Explain Aggregation Stages (Cheat Sheet)

Operator	Meaning	Example Purpose
$match	Filter docs	High-value txns
$group	Summarize	Total per customer
$project	Select + compute fields	Add avgAmount
$sort	Order	Top spenders
$limit	Restrict N	Show top 5
$addFields	Add flags	Risk features
$lookup	Join collections	Customer + Account
$unwind	Flatten arrays	Expand transactions
$count	Count docs	Simple counts
$sum	Sum values	Total debit
$avg	Average	Mean txn size
$dateToString	Format dates	YY-MM-DD

Lab Completion Checklist

✔ Understand $match, $project, $group, $sort, $limit

✔ Create simple & complex pipelines

✔ Filter & transform data

✔ Build Customer360 transaction summary

✔ Interpret aggregation output

✔ Apply best practices for performance
