Great — now that you have **three Hive tables** (`user_tag_value`, `moc_course`, and `moc_score`), here's a comprehensive list of **useful aggregations** you can perform by querying **one table**, or **joining multiple tables**.

🔹 1. Count users per district

From: `user_tag_value`

```sql
SELECT district, COUNT(*) AS user_count
FROM user_tag_value
GROUP BY district;
```

🔹 2. Top-scoring user per course

From: `moc_score`

```sql
SELECT course_id, MAX(CAST(score AS INT)) AS top_score
FROM moc_score
GROUP BY course_id;
```

> ℹ Cast `score` to `INT` or `DOUBLE` since it's stored as a `STRING`.

🔹 3. Average score per course

```sql
SELECT course_id, AVG(CAST(score AS DOUBLE)) AS avg_score
FROM moc_score
GROUP BY course_id;
```

🔹 4. Total number of students per course

```sql
SELECT course_id, COUNT(DISTINCT userid) AS num_students
FROM moc_score
GROUP BY course_id;
```

🔹 5. Average score per district (using `JOIN`)

Join `user_tag_value` and `moc_score` on `userid`:

```sql
SELECT u.district, AVG(CAST(s.score AS DOUBLE)) AS avg_score
FROM user_tag_value u
JOIN moc_score s ON u.userid = s.userid
GROUP BY u.district;
```

6. List of courses each user enrolled in

```sql
SELECT userid, COLLECT_SET(course_id) AS courses
FROM moc_score
GROUP BY userid;
```

7. Total course load per user

Join **moc_score** with **moc_course** using **course_id**:

```sql
SELECT s.userid, SUM(CAST(c.course_load AS INT)) AS total_course_load
FROM moc_score s
JOIN moc_course c ON s.course_id = c.id
GROUP BY s.userid;
```

8. Number of users who took each course and their average score

```sql
SELECT course_id, COUNT(DISTINCT userid) AS student_count, AVG(CAST(score AS DOUBLE)) AS
avg_score
FROM moc_score
GROUP BY course_id;
```

9. Most recent login per district

```sql
SELECT district, MAX(last_login_time) AS last_seen
FROM user_tag_value
GROUP BY district;
```

10. Previous course popularity

From **moc_course**, count how many times each **previous_course** appears: