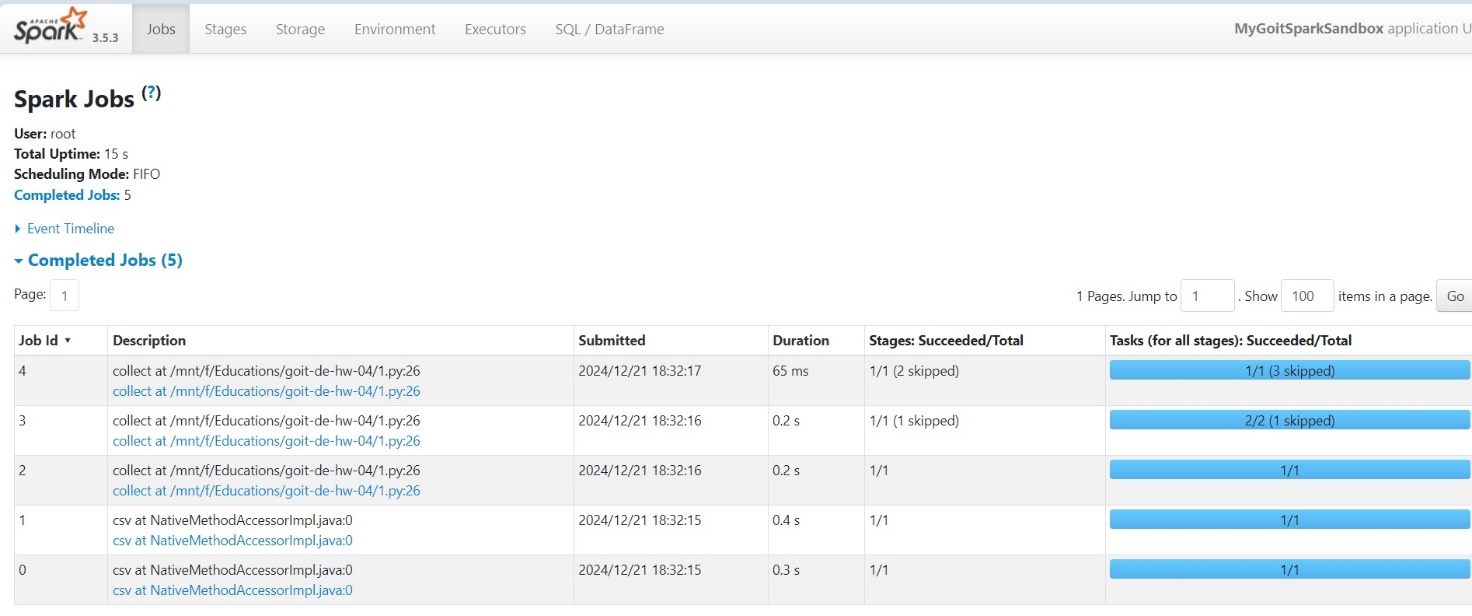
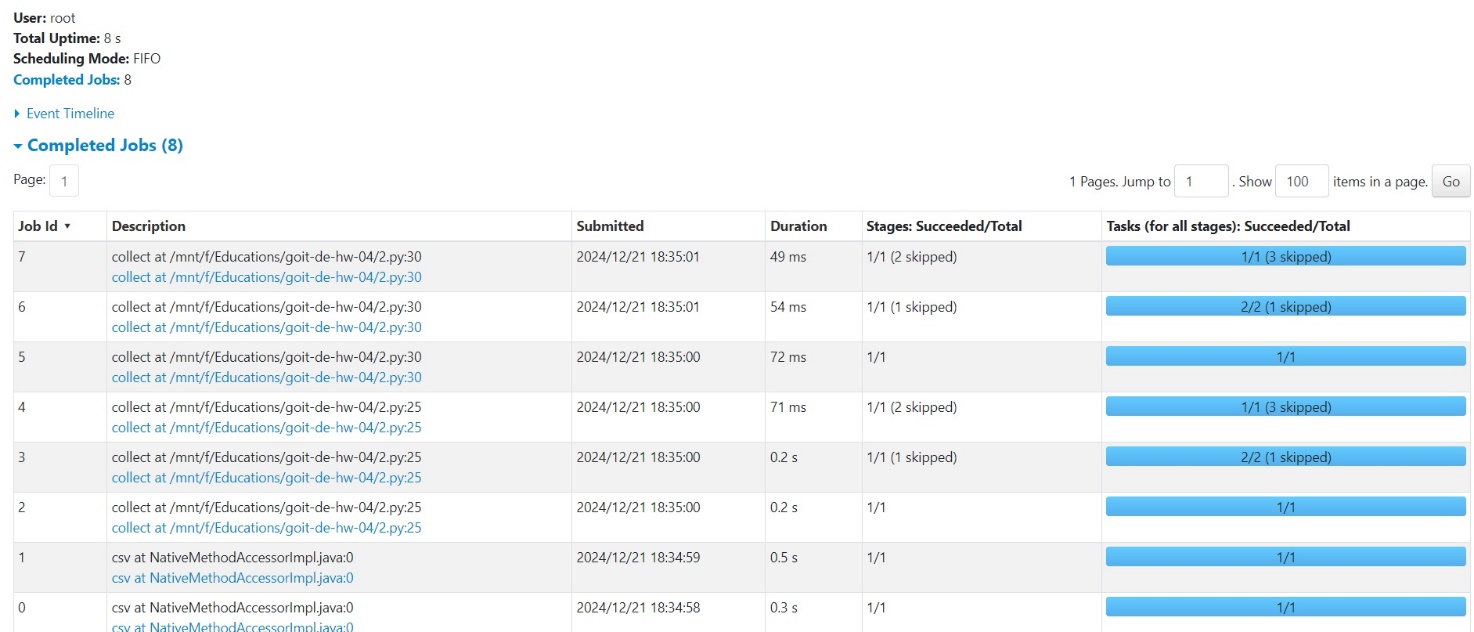
**Apache Spark. Оптимізація та SparkUI**

**Завдання 1**

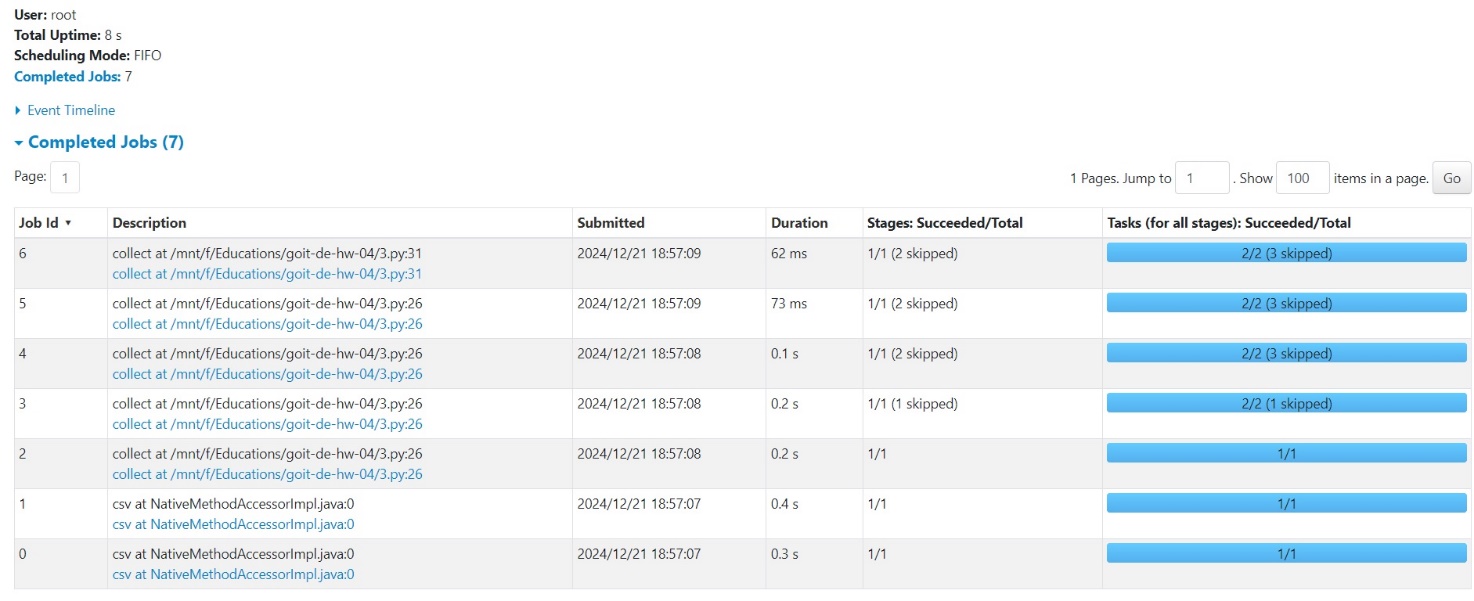


**Завдання 2**



Додавання другого collect() і справді збільшило кількість Jobs з 5 до 8. Це відбувається через принцип "лінивих обчислень" (lazy evaluation) у Spark. Spark відкладає виконання трансформацій до моменту, коли зустріне команду дії. Ось Коли зустрічається команда дії, як-от collect(), Spark запускає весь ланцюжок обчислень. Кожна нова команда дії змушує Spark виконувати всі обчислення з самого початку, навіть якщо деякі результати вже були пораховані раніше. В нашому випадку перший collect() запустив початковий набір Jobs для обробки даних, другий collect() повторно виконав всі попередні обчислення, додавши новий фільтр where("count > 2"). Ось чому загальна кількість Jobs зросла – Spark не зберігає результати попередніх обчислень, а виконує кожен ланцюжок трансформацій заново для кожної команди дії.

**Завдання 3**



Загалом функція cache() значно зменшує кількість Jobs у Spark завдяки збереженню проміжних результатів. cache() зберігає результати обчислень в оперативній пам'яті або на диску. Коли дані закешовані, Spark може використовувати їх повторно замість того, щоб обчислювати все спочатку. В даному коді перший collect() виконує обчислення та зберігає результат у кеші. Наступні collect() використовують збережені дані і кількість Jobs зменшується, оскільки немає потреби повторювати обчислення.