**Завдання 1**

A screenshot of a computer

Description automatically generated

**Job 1:** Зчитування CSV-файлу.

* Операція spark.read.csv створює Job для завантаження даних.

**Job 2:** Репартиція (repartition(2)).

* Ця трансформація викликає shuffle, створюючи окремий Job.

**Job 3:** Групування та підрахунок (groupBy і count).

* Коли викликається перший .collect(), виконується весь DAG до цього моменту (включаючи where і groupBy).

**Job 4:** Фільтрація результатів (where("count>2")).

* Друге .collect() змушує Spark виконати новий DAG, який включає цю фільтрацію.

**Job 5:** Заключне збирання результатів.

* Другий виклик .collect() завершується як окремий Job.

**Завдання 2**

**A screenshot of a computer

Description automatically generated**

1. Перше .collect() запускає Jobs для обчислення всього попереднього DAG (Directed Acyclic Graph).
2. Додане фільтрування створює новий DAG.
3. Друге .collect() запускає Jobs для нового DAG.
4. Spark не використовує кешування між викликами .collect() (якщо не вказати це явно), тому всі операції виконуються повторно.

**Завдання 3**

**A screenshot of a computer

Description automatically generated**

* При першому виконанні Spark обчислює весь DAG і кешує результат.
* Усі подальші виклики action використовують кешовані дані, зменшуючи обсяг роботи та кількість Jobs.