and Python Pandas.

DataFrames support all common relational operators, including projection (select), filter (where), join, and aggregations (groupBy).

These operators all take expression objects in a limited DSL that lets Spark capture the structure of the expression. For example, the following code computes the number of female employees in each department.

employees

.join(dept , employees (" deptId ") === dept ("id "))

.where(employees (" gender ") === "female ")

.groupBy(dept ("id"), dept (" name "))

.agg(count (" name "))

- [x] True
- [ ] False

View question 1 feedback

Question 2 **20 / 20 points**

Like Shark before it, Spark SQL can materialize (often referred to as “cache") hot data in memory using columnar storage. Compared with Spark’s native cache, which simply stores data as JVM objects, the columnar cache can reduce memory footprint by an order of magnitude because it applies columnar compression schemes such as dictionary encoding and run-length encoding.

Caching is particularly useful for interactive queries and for the iterative algorithms common in machine learning. It can be invoked by calling cache() on a DataFrame.

be invoked by calling cache() on a DataFrame.

- [x] True
- [ ] False

View question 2 feedback

Question 3 **20 / 20 points**

Spark SQL runs as a library on top of Spark, as shown below. It exposes SQL interfaces, which can be accessed through