Fourth Assignment

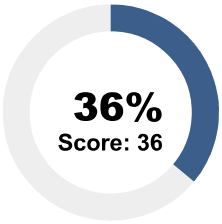
Performance summary	
Maximum number of attempts	Unlimited
Number of attempts	1
Your score	36
Status	**Not available**

Results

Course	**Big Data**
Test	**4_Assignment_BD**

This are your test results

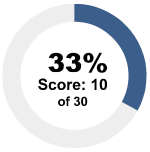
Duration	8h 44m 11s
Answered	8 of 8 questions (100%)
Your score	36 of 100 points (36%)



Knowledge questions 3

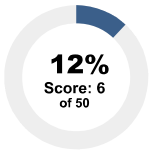
go to section

Practice 2



go to section

Programming questions 3



go to section

Knowledge questions 20 of 20 points (100%)

☑ Volcano Iterator Model

Status	Answered		
Your score	5 / 5	[redacted]	100%

Response

Which of the following statements are correct about Volcano Iterator Model?

Note:
Maximum score -> 5 points
Minimum overall score -> 0 points
Wrong answer -> -1 points
Unanswered -> 0 points

- ☐ All function calls are fused into one function call
- ☐ Data in CPU
- ☑ Data is read and written to memory
- ☐ No function calls
- ☑ Lots of virtual function calls

▸ Solution

☑ Whole-stage code generation

Status	Answered		
Your score	5 / 5	[redacted]	100%

Response

Which of the following statements are correct about whole-stage code generation?

Note:
Maximum score -> 5 points
Minimum overall score -> 0 points
Wrong answer -> -1 points
Unanswered -> 0 points

- ☑ works well with simple operations
- ☐ Data in CPU
- ☐ Data is read and written to memory
- ☐ Lots of virtual function calls
- ☑ All function calls are fused into one function call

▸ Solution

Spark Joins

Status	Answered	
Your score	10 / 10	100%

Response

Please match the following concepts from the left side to the correct join type on the right side.

- Returns only the data from left table that doesn't have a match on the right table based on some conditions.
- Returns only the data from left table that has a match on the right table based on some conditions.
- Returns all the data from both tables that has a match on the join conditions.
- Returns all the data from both tables even if there is condition match or not.
- Cartesion product of rows from all tables that involved in join.

Left Semi Join
- Returns only the data from left table that has a match on the right table based on some conditions.

Left Anti Join
- Returns only the data from left table that doesn't have a match on the right table based on some conditions.

Inner Join
- Returns all the data from both tables that has a match on the join conditions.

Full Outer Join
- Returns all the data from both tables even if there is condition match or not.

Cross Join
- Cartesion product of rows from all tables that involved in join.

▸ Solution

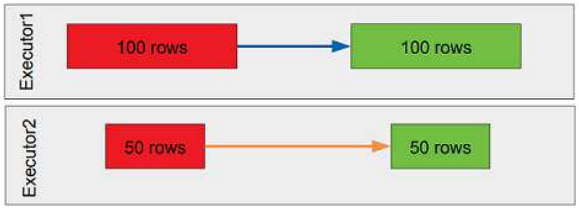
go back to overview

Practice 10 of 30 points (33%)

Cost for Narrow Transformation

Status	Answered	
Your score	10 / 10	100%

Response



What would be the communication cost and elapsed communication cost for Narrow transformation?

Communication Cost 300.0

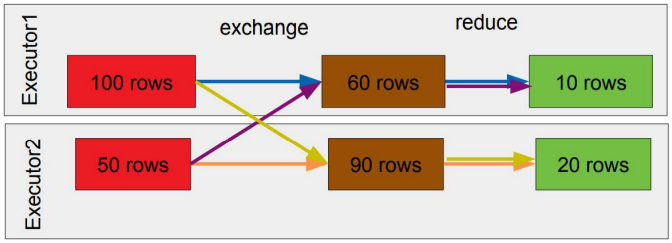
Elapsed Communication Cost 200.0

▸ Solution

Cost for Wide Transformation

Status	Answered	
Your score	0 / 20	0%

Response



What would be the communication cost and elapsed communication cost for Wide transformation?

Communication Cost 330.0

Elapsed Communication Cost 210.0

Solution

go back to overview

Programming questions 6 of 50 points (12%)

Pivot

Status	Answered		
Your score	2 / 10		20%

Response

For this section, you are provided with the following retails dataset. Please use the dataset to answer the following question.

Download the dataset from here: https://drive.google.com/file/d/13y81xA5ilsse4jE9HWtLkwf-3-_Lw3-1/view?usp=sharing

If the link does not work, the dataset is available in the materials folder.

Q: How many instances of each product were sold in each country?

Please select the correct options.

Note:

Maximum score -> 10 points

Minimum overall score -> 0 points

Wrong answer -> -2.0 points

Unanswered -> 0 points

Unanswered	Right	Wrong	
○	○	◉	retail_data.groupBy("Country").pivot("StockCode").sum("Quantity").show()
○	○	◉	retail_data.groupBy("StockCode").pivot("Country").sum("Quantity").show()
○	◉	○	retail_data.groupBy("StockCode").pivot("Country").count().show()
○	○	◉	retail_data.groupBy("InvoiceNo").pivot("Country").sum("Quantity").show()
○	○	◉	retail_data.groupBy("InvoiceNo").pivot("Country").count().show()

Solution

For this section, you are provided with the following retails dataset. Please use the dataset to answer the following question.

Download the dataset from here: https://drive.google.com/file/d/13y81xA5ilsse4jE9HWtLkwf-3-_Lw3-1/view?usp=sharing

If the link does not work, the dataset is available in the materials folder.

Q: How many instances of each product were sold in each country?

Please select the correct options.

Note:

Maximum score -> 10 points

Minimum overall score -> 0 points

Wrong answer -> -2.0 points

Unanswered -> 0 points

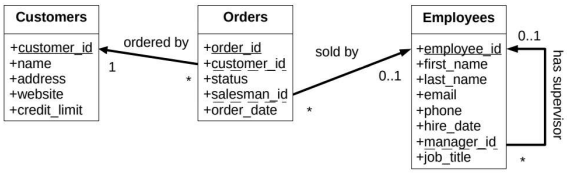
Unanswered	Right	Wrong	
○	○	◉	retail_data.groupBy("Country").pivot("StockCode").sum("Quantity").show()
○	◉	○	retail_data.groupBy("StockCode").pivot("Country").sum("Quantity").show()
○	○	◉	retail_data.groupBy("StockCode").pivot("Country").count().show()
○	○	◉	retail_data.groupBy("InvoiceNo").pivot("Country").sum("Quantity").show()
○	○	◉	retail_data.groupBy("InvoiceNo").pivot("Country").count().show()

Joins 1

Status	Answered	
Your score	0 / 20	0%

Response

The following schema describes customers who can place orders. Furthermore, there are employees who can convince customers to place an order. Not every order has to have a corresponding salesman. In this case, NULL is stored in the salesman id column. Employees can have supervisors. The orders table is the fact table and the other tables are dimensional tables. Use this schema for the following tasks. The DataFrames Customers, Orders, and Employees are available as variable and view with the names customers, orders, and employees respectively.



Q: Which employees could convince customers to order products?

Please select the correct options.

Note:

Maximum score -> 20 points

Minimum overall score -> 0 points

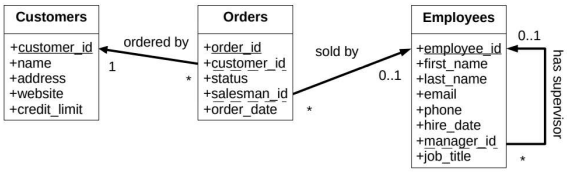
Wrong answer -> -2.0 points

Unanswered -> 0 points

Unanswered	Right	Wrong	
○	⊙	○	SELECT * FROM employees e LEFT OUTER JOIN orders o ON e.employee_id = o.salesman_id
○	○	⊙	SELECT * FROM employees e LEFT ANTI JOIN orders o ON e.employee_id = o.salesman_id
○	○	⊙	SELECT * FROM employees e LEFT SEMI JOIN orders o ON e.employee_id = o.salesman_id
○	⊙	○	employees.join(orders, f.expr("employee_id = salesman_id"), "left_outer")
○	○	⊙	employees.join(orders, f.expr("employee_id = salesman_id"), "left_semi")
○	○	⊙	employees.join(orders, f.expr("employee_id = salesman_id"), "left_anti")
○	⊙	○	SELECT * FROM orders o LEFT SEMI JOIN employees e ON e.employee_id = o.salesman_id
○	⊙	○	orders.join(employees, f.expr("employee_id = salesman_id"), "left_semi")
○	○	⊙	SELECT * FROM orders o LEFT ANTI JOIN employees e ON o.salesman_id = e.employee_id
○	○	⊙	orders.join(employees, f.expr("salesman_id = employee_id"), "left_anti")

Solution

The following schema describes customers who can place orders. Furthermore, there are employees who can convince customers to place an order. Not every order has to have a corresponding salesman. In this case, NULL is stored in the salesman id column. Employees can have supervisors. The orders table is the fact table and the other tables are dimensional tables. Use this schema for the following tasks. The DataFrames Customers, Orders, and Employees are available as variable and view with the names customers, orders, and employees respectively.



Q: Which employees could convince customers to order products?

Please select the correct options.

Note:

Maximum score -> 20 points

Minimum overall score -> 0 points

Wrong answer -> -2.0 points

Unanswered -> 0 points

Unanswered	Right	Wrong	
○	○	◉	SELECT * FROM employees e LEFT OUTER JOIN orders o ON e.employee_id = o.salesman_id
○	○	◉	SELECT * FROM employees e LEFT ANTI JOIN orders o ON e.employee_id = o.salesman_id
○	◉	○	SELECT * FROM employees e LEFT SEMI JOIN orders o ON e.employee_id = o.salesman_id
○	○	◉	employees.join(orders, f.expr("employee_id = salesman_id"), "left_outer")
○	◉	○	employees.join(orders, f.expr("employee_id = salesman_id"), "left_semi")
○	○	◉	employees.join(orders, f.expr("employee_id = salesman_id"), "left_anti")
○	○	◉	SELECT * FROM orders o LEFT SEMI JOIN employees e ON e.employee_id = o.salesman_id
○	○	◉	orders.join(employees, f.expr("employee_id = salesman_id"), "left_semi")
○	○	◉	SELECT * FROM orders o LEFT ANTI JOIN employees e ON o.salesman_id = e.employee_id
○	○	◉	orders.join(employees, f.expr("salesman_id = employee_id"), "left_anti")

Joins 2

Status	Answered	
Your score	4 / 20	20%

Response

Please follow the schema from the Joins part1 and answer the following question.

Q. Which employees could convince most customers to order products?

Please select the correct options.

Note:

Maximum score -> 20 points

Minimum overall score -> 0 points

Wrong answer -> -2.0 points

Unanswered -> 0 points

Unanswered	Right	Wrong	
○	⊙	○	SELECT e.employee_id, count(*) as sales FROM employees e FULL JOIN orders o ON e.employee_id = o.salesman_id GROUP BY e.employee_id ORDER BY sales DESC
○	⊙	○	employees.join(orders, f.expr("employee_id = salesman_id"),"inner").groupBy("employee_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=False)
○	⊙	○	SELECT e.employee_id, count(*) as sales FROM employees e SEMI JOIN orders o ON e.employee_id = o.salesman_id GROUP BY e.employee_id ORDER BY sales DESC
○	⊙	○	employees.join(orders, f.expr("employee_id = salesman_id"),"full").groupBy("employee_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=False)
○	○	⊙	SELECT o.salesman_id, count(*) as sales FROM orders o LEFT ANTI JOIN employees e ON o.salesman_id = e.employee_id GROUP BY o.salesman_id ORDER BY sales
○	⊙	○	employees.join(orders, f.expr("employee_id = salesman_id"),"left_semi").groupBy("employee_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=False)
○	○	⊙	orders.join(employees, f.expr("salesman_id = employee_id"),"left_anti").groupBy("salesman_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=True).show()
○	⊙	○	SELECT e.employee_id, count(*) as sales FROM employees e INNER JOIN orders o ON e.employee_id = o.salesman_id GROUP BY e.employee_id ORDER BY sales DESC
○	○	⊙	orders.join(employees, f.expr("salesman_id = employee_id"),"left_semi").groupBy("employee_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=False)
○	○	⊙	SELECT e.employee_id, count(*) as sales FROM employees e INNER JOIN orders o ON e.employee_id = o.salesman_id GROUP BY o.salesman_id ORDER BY sales DESC

Solution

Please follow the schema from the Joins part1 and answer the following question.

Q. Which employees could convince most customers to order products?

Please select the correct options.

Note:

Maximum score -> 20 points

Minimum overall score -> 0 points

Wrong answer -> -2.0 points

Unanswered -> 0 points

Unanswered	Right	Wrong	
○	○	⊙	SELECT e.employee_id, count(*) as sales FROM employees e FULL JOIN orders o ON e.employee_id = o.salesman_id GROUP BY e.employee_id ORDER BY sales DESC
○	⊙	○	employees.join(orders, f.expr("employee_id = salesman_id"),"inner").groupBy("employee_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=False)
○	○	⊙	SELECT e.employee_id, count(*) as sales FROM employees e SEMI JOIN orders o ON e.employee_id = o.salesman_id GROUP BY e.employee_id ORDER BY sales DESC
○	○	⊙	employees.join(orders, f.expr("employee_id = salesman_id"),"full").groupBy("employee_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=False)
○	○	⊙	SELECT o.salesman_id, count(*) as sales FROM orders o LEFT ANTI JOIN employees e ON o.salesman_id = e.employee_id GROUP BY o.salesman_id ORDER BY sales
○	○	⊙	employees.join(orders, f.expr("employee_id = salesman_id"),"left_semi").groupBy("employee_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=False)
○	○	⊙	orders.join(employees, f.expr("salesman_id = employee_id"),"left_anti").groupBy("salesman_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=True).show()
○	⊙	○	SELECT e.employee_id, count(*) as sales FROM employees e INNER JOIN orders o ON e.employee_id = o.salesman_id GROUP BY e.employee_id ORDER BY sales DESC
○	○	⊙	orders.join(employees, f.expr("salesman_id = employee_id"),"left_semi").groupBy("employee_id").agg(f.expr("count(*) as sales")).orderBy("sales",ascending=False)
○	○	⊙	SELECT e.employee_id, count(*) as sales FROM employees e INNER JOIN orders o ON e.employee_id = o.salesman_id GROUP BY o.salesman_id ORDER BY sales DESC

Test period was over at **5/12/2021, 8:00 PM**.