UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Experiment No. 10th

Student Name: Rishav Kumar **UID: 22MCC20039**
Branch: MCA - CCD **Section/Group: 22MCD-1/ Grp A**
Semester: IV **Date of Performance: 10th April 24**
Subject Name: Big Data & Analytics Lab **Subject Code: 22CAH-782**

1. Aim/Overview of the practical:

Implementation of built in and user-defined function.

2. Code/Steps & Output for practical:

Built-In Functions

Hive supports a lot of built-in functions. Here are a few of the built-in functions:

Return Type	Signature	Description
BIGINT	round(double a)	It returns the rounded BIGINT value of the double.
double	rand(), rand(int seed)	It returns a random number that changes from row to row.
string	concat(string A, string B,...)	It returns the string resulting from concatenating B after A.
string	substr(string A, int start)	It returns the substring of A starting from start position till the end of string A.
string	substr(string A, int start, int length)	It returns the substring of A starting from start position with the given length.
string	upper(string A)	It returns the string resulting from converting all characters of A to upper case.
int	size(Map<K.V>)	It returns the number of elements in the map type.
int	size(Array<T>)	It returns the number of elements in the array type.

Example

The following queries demonstrate some built-in functions:

round() function

```
hive> SELECT round(2.6) from temp;
```

On successful execution of query, you get to see the following response:

```
3.0
```

floor() function

```
hive> SELECT floor(2.6) from temp;
```

On successful execution of the query, you get to see the following response:

```
2.0
```

ceil() function

```
hive> SELECT ceil(2.6) from temp;
```

On successful execution of the query, you get to see the following response:

```
3.0
```

User-Defined Functions

The following Hive built-in functions are supported in Db2 Big SQL:

Function Name	Arguments
datediff	`end_date varchar(50), start_date varchar(50)`
date_add	`start_date varchar(50), daysToAdd int`
date_sub	`start_date varchar(50), daysToSubtract int`
decode	`binToDecode varbinary(50), charSet varchar(50)`

encode	`sourceString varchar(50), charSet varchar(50)`
format_number	`numberToConvert double, decimalPlaces int`
from_unixtime	`unixtime bigint`
from_utc_timestamp	This function is available as a built-in function in the SYSIBM schema. For more information about this function, see FROM_UTC_TIMESTAMP scalar function.
get_json_object	`JSON_TEXT varchar(32672), JSON_PATH varchar(4096)`
log	`base double,number double`
log2	`number double`
parse_url	`urlString varchar(32672), partToExtract varchar(4096) [, key varchar(4096)]`

Examples

datediff

The result of the function is the number of days from *start_date* to *end_date* as an integer. In this example, the column introduction_date is a VARCHAR(15). The query returns those rows where the number of days is greater than 30.

```
SELECT * FROM PRODUCT WHERE HIVE.DATEDIFF('2014-06-1',introduction_date) > 30;
```

date_add

The result of the function adds a number of days to *start_date*. In this example, the column introduction_date is a VARCHAR(15). The function returns the adjusted date.

```
SELECT HIVE.DATE_ADD(introduction_date,1),introduction_date FROM PRODUCT;
```

date_sub

The result of the function subtracts a number of days from *start_date*. In this example, the column introduction_date is a VARCHAR(15). The function returns the adjusted date.

```
SELECT HIVE.DATE_SUB(introduction_date,1),introduction_date FROM PRODUCT;
```

format_number

This function formats the number in the first expression to a format like '#,###,###.##', rounded to the number of decimal places in the second expression. It returns the result as a STRING. If the second expression is 0, the result has no decimal point or fractional part.

```
SELECT hive.format_number(12332.123456, 4),
      hive.format_number(12332.1,4),
      hive.format_number(12332.2,0) FROM sysibm.sysdummy1;
```

The output:

```
      '12,332.1235'  |    '12,332.1000'   |   '12,332'
```

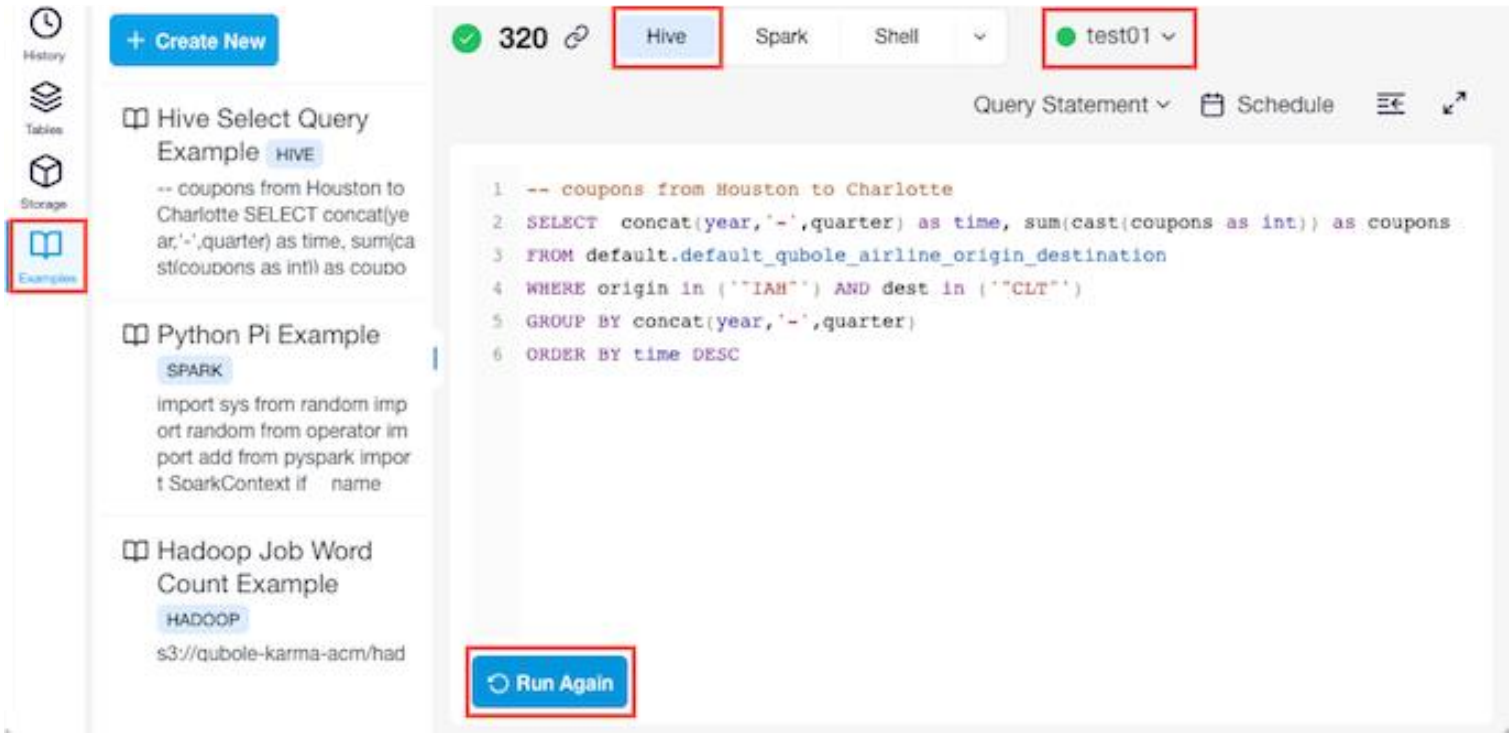
log

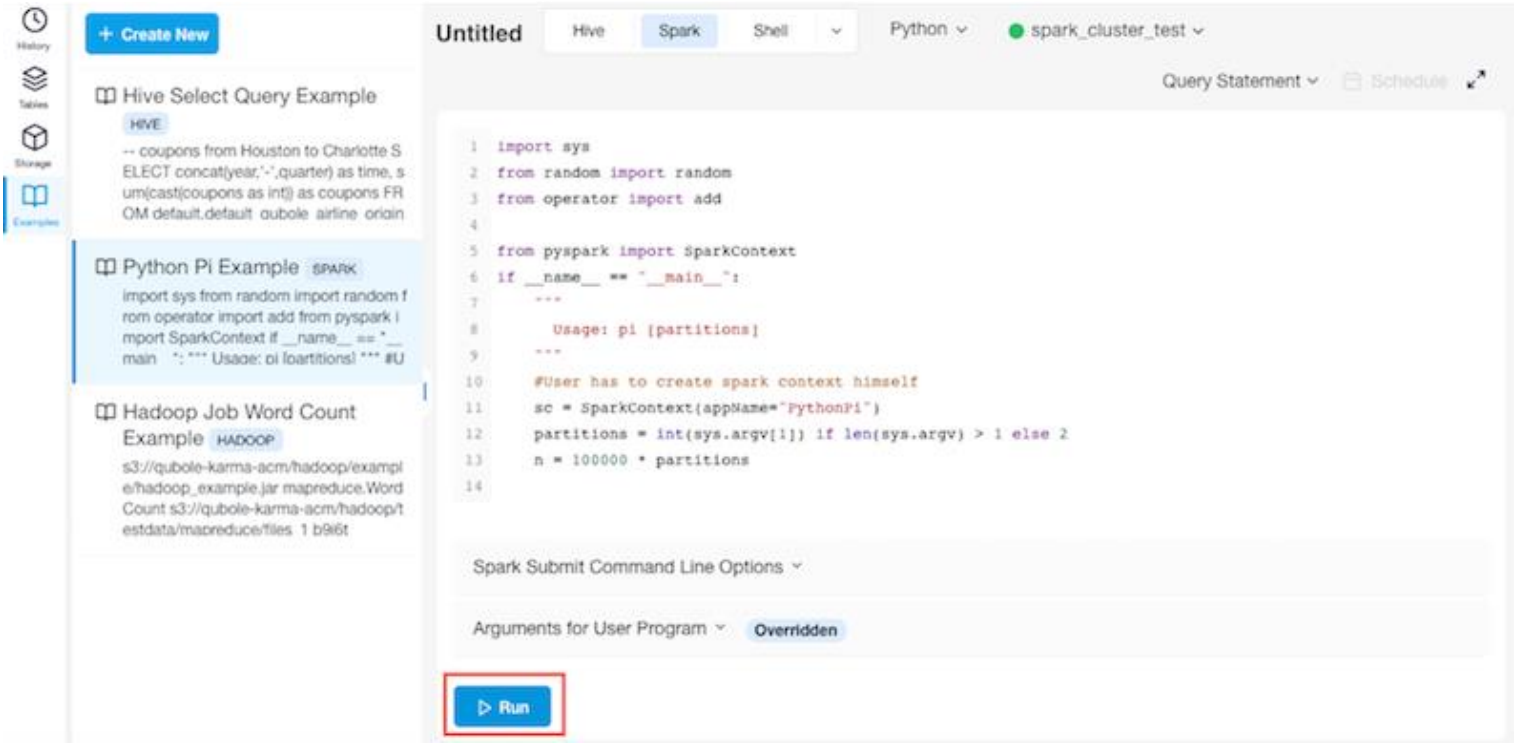
This function returns the base logarithm of the argument in the second expression.

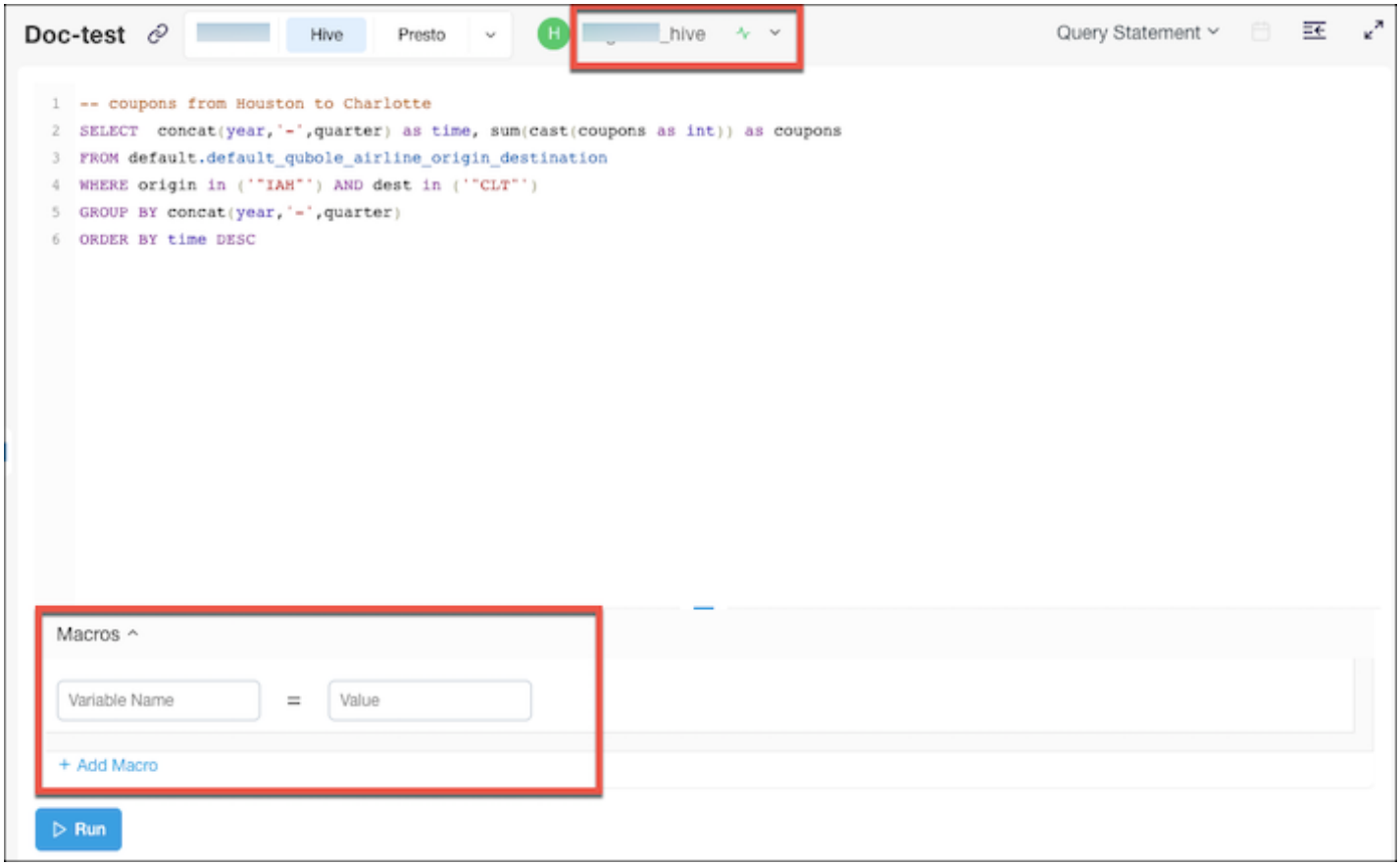
```
SELECT HIVE.LOG(3, 66) FROM sysibm.sysdummy1 ;
```

The output:

```
+3.81358809221559E+000
```


History
Tables
Storage
Examples
+ Create New
Hive Select Query Example
HIVE
-- coupons from Houston to Charlotte SELECT concat(year,'-',quarter) as time, sum(cast(coupons as int)) as coupo
Python Pi Example
SPARK
import sys from random import random from operator import add from pyspark import SparkContext if __name
Hadoop Job Word Count Example
HADOOP
s3://qubole-karma-acm/had
320
Hive
Spark
Shell
test01
Query Statement
Schedule
1 -- coupons from Houston to Charlotte
2 SELECT concat(year,'-',quarter) as time, sum(cast(coupons as int)) as coupons
3 FROM default.default_qubole_airline_origin_destination
4 WHERE origin in ('"IAH"') AND dest in ('"CLT"')
5 GROUP BY concat(year,'-',quarter)
6 ORDER BY time DESC
Run Again


History
Tables
Storage
Examples
+ Create New
Hive Select Query Example
HIVE
-- coupons from Houston to Charlotte SELECT concat(year,'-',quarter) as time, sum(cast(coupons as int)) as coupons FROM default.default_qubole_airline_origin
Python Pi Example
SPARK
import sys from random import random from operator import add from pyspark import SparkContext if __name__ == "__main__": *** Usage: pi [partitions] *** #U
Hadoop Job Word Count Example
HADOOP
s3://qubole-karma-acm/hadoop/example/hadoop_example.jar mapreduce.WordCount s3://qubole-karma-acm/hadoop/testdata/mapreduce/files 1 b9i6t
Untitled
Hive
Spark
Shell
Python
spark_cluster_test
Query Statement
Schedule
1 import sys
2 from random import random
3 from operator import add
4
5 from pyspark import SparkContext
6 if __name__ == "__main__":
7 """
8 Usage: pi [partitions]
9 """
10 #User has to create spark context himself
11 sc = SparkContext(appName="PythonPi")
12 partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
13 n = 100000 * partitions
14
Spark Submit Command Line Options
Arguments for User Program
Overridden
Run



3. Learning outcomes (What I have learned):

a) Learned about the built in functions in Hive & to use a user-defined function.