```
EXAM EDITION

Touch somequestion.scala

hdfs dfs -put clusterdata.csv /user/hive/warehouse/

:load mycode.scala
spark-shell<enter>

import spark.implicits._

spark.sql("show tables").show()


spark.sql("select * from purchases").show()

val purch = spark.sql("select * from purchases")

purch.show()
```

Filter

```
val purchOverFifteen = purch.filter("purchase_id > 15")
bdf.filter("buyer_name == 'Raj'").show()
dfTags.filter("tag like 's%'").show(10)
bdf.filter(" buyer_name == 'Raj' or buyer_name == 'Anu' ").show()
dfTags.filter("id in (25, 108)").show(10)
```

Columns manipulation

```
val purchWithMonth =
purchOverFifteen.select("*").withColumn("Month",lit("Nov"))

df.select($"name", $"age" + 1).show()
```

Joins in SQL , Not liked by me

```
purchWithMonth.createOrReplaceTempView("purchWithMonth")


val purchWithBuyerNum = spark.sql("select purchWithMonth.*, buyers.buyer_num

                                    from purchWithMonth
                                    join buyers
                                    on purchWithMonth.buyer =
buyers.buyer_name")
```

Grouping

```
val buyerSummary = purchWithBuyerNum.groupBy("buyer").agg(count("*"))
```

JOINS IN DATA FRAMES spark

```
val buyers = spark.sql("select * from buyers")
val purchWithBuyerAgain = purchWithMonth.join(buyers,
purchWithMonth.col("buyer") === buyers.col("buyer_name"), "inner")
```

```
bdf.join(bdf3,bdf("buyer_num") === bdf3("num"),"left_outer").show()
```

I'd rename before join: Or else there is too much trouble

```
df1.alias("tab1").join(
  df2.withColumnRenamed("descr", "dept_full_description").alias("tab2"),
  Seq("id"), "left_outer")
```

Classes in scala

Class   myclass {

Val myval1 = “any thing”

Var myvar2 = “ any thing 2”

Def func1(x: Int,y:Int) : Unit = {   }

Def func2(x: Int,y: String) : Int = {   }

Def func3(x: String,y: String) : String = {   }

}

// instant of the class or an object of the class

Val myClassObject = new myclass()

Function

Def myfunc1(x: Int, y:int): Int={   // function starts

Println(s”x=$x”)   // string interpretation , I will put $ and a var

Println(s”y=$y”)   // print y= value of y

X*y }   //instead of Int you can also write unit or string

Println(myfunc1(3,2)

For loop

for(i <- 1 to10) { println(myfunc(i,i) }

conditions

if(x>y){   println(“X is larger than y”)}

```
else{ println("x is not larger than y")}
```

use string s with $ sign

```
for( I <- 10 to 90) { println(s" the value of I is $i ")
```

yield statement

```
for (k <- 20 to 30) yield k
```