***1) From home\_data.csv, how many houses sold were built prior to 1979? (We promise that is the last time we’ll ask this question).***

case class HomeData(id:BigInt,date:String,price:Integer,bedrooms:Integer,bathrooms:Double,sqft\_living:Integer,sqft\_lot:Integer,floors:Double,waterfront:Integer,view:Integer,condition:Integer,grade:Integer,sqft\_above:Integer,sqft\_basement:Integer,yr\_built:Integer,yr\_renovated:Integer,zipcode:Integer,latitude:Double,longitude:Double,sqft\_living15:Integer,sqft\_lot15:Integer)

val home\_data = spark.read.option("inferSchema","true").option("header","true").csv("file:///root/home\_data1.csv").as[HomeData]

val build\_prior\_1979 = home\_data.filter(\_.yr\_built < 1979)

build\_prior\_1979.count

Long = 11991

***2) From home\_data.csv, what is the most expensive zipcode in the data set, defined as the zipcode having the highest average sales price?***

case class HomeData(id:BigInt,date:String,price:Integer,bedrooms:Integer,bathrooms:Double,sqft\_living:Integer,sqft\_lot:Integer,floors:Double,waterfront:Integer,view:Integer,condition:Integer,grade:Integer,sqft\_above:Integer,sqft\_basement:Integer,yr\_built:Integer,yr\_renovated:Integer,zipcode:Integer,latitude:Double,longitude:Double,sqft\_living15:Integer,sqft\_lot15:Integer)

val home\_data = spark.read.option("inferSchema","true").option("header","true").csv("file:///root/home\_data1.csv").as[HomeData]

val ExpensiveZipcode = home\_data.groupBy("zipcode").avg("price")

ExpensiveZipcode.sort($"avg(price)".desc).show(1)

|zipcode |avg(price)|

| 98039| 2160606.6|

***3) How many unique zipcodes have sales data in the home\_data.csv data set?***

case class HomeData(id:BigInt,date:String,price:Integer,bedrooms:Integer,bathrooms:Double,sqft\_living:Integer,sqft\_lot:Integer,floors:Double,waterfront:Integer,view:Integer,condition:Integer,grade:Integer,sqft\_above:Integer,sqft\_basement:Integer,yr\_built:Integer,yr\_renovated:Integer,zipcode:Integer,latitude:Double,longitude:Double,sqft\_living15:Integer,sqft\_lot15:Integer)

val home\_data = spark.read.option("inferSchema","true").option("header","true").csv("file:///root/home\_data1.csv").as[HomeData]

val uniqueZipcode = home\_data.agg(countDistinct("zipcode"))

uniqueZipcode.show(1)

|count(DISTINCT zipcode)|

| 70|

***4) Demonstrate how to drop the “sqft\_living15” and “sqft\_lot15” columns from your dataset.***

case class HomeData(id:BigInt,date:String,price:Integer,bedrooms:Integer,bathrooms:Double,sqft\_living:Integer,sqft\_lot:Integer,floors:Double,waterfront:Integer,view:Integer,condition:Integer,grade:Integer,sqft\_above:Integer,sqft\_basement:Integer,yr\_built:Integer,yr\_renovated:Integer,zipcode:Integer,latitude:Double,longitude:Double,sqft\_living15:Integer,sqft\_lot15:Integer)

val home\_data = spark.read.option("inferSchema","true").option("header","true").csv("file:///root/home\_data1.csv").as[HomeData]

val newDs = home\_data1.drop("sqft\_living15","sqft\_lot15")

***5) Access the zipcode table stored in Hive (it’s ok to use a SQL string query here) and demonstrate joining that data with a DataFrame created from home\_data.csv.***

case class HomeData(id:BigInt,date:String,price:Integer,bedrooms:Integer,bathrooms:Double,sqft\_living:Integer,sqft\_lot:Integer,floors:Double,waterfront:Integer,view:Integer,condition:Integer,grade:Integer,sqft\_above:Integer,sqft\_basement:Integer,yr\_built:Integer,yr\_renovated:Integer,zipcode:Integer,latitude:Double,longitude:Double,sqft\_living15:Integer,sqft\_lot15:Integer)

val home\_data = spark.read.option("inferSchema","true").option("header","true").csv("file:///root/home\_data1.csv").as[HomeData]

val hiveContext = new org.apache.spark.sql.hive.HiveContext(sc)

val wa\_zipcodes = hiveContext.table("wa\_zipcodes")

case class WaZipcode(zipcode:Int,city:String,state:String)

val WaZipcodeDs = wa\_zipcodes.as[WaZipcode]

val joined = home\_data.join(WaZipcodeDs,home\_data("zipcode")=== WaZipcodeDs("zipcode"))

joined.show(10)