

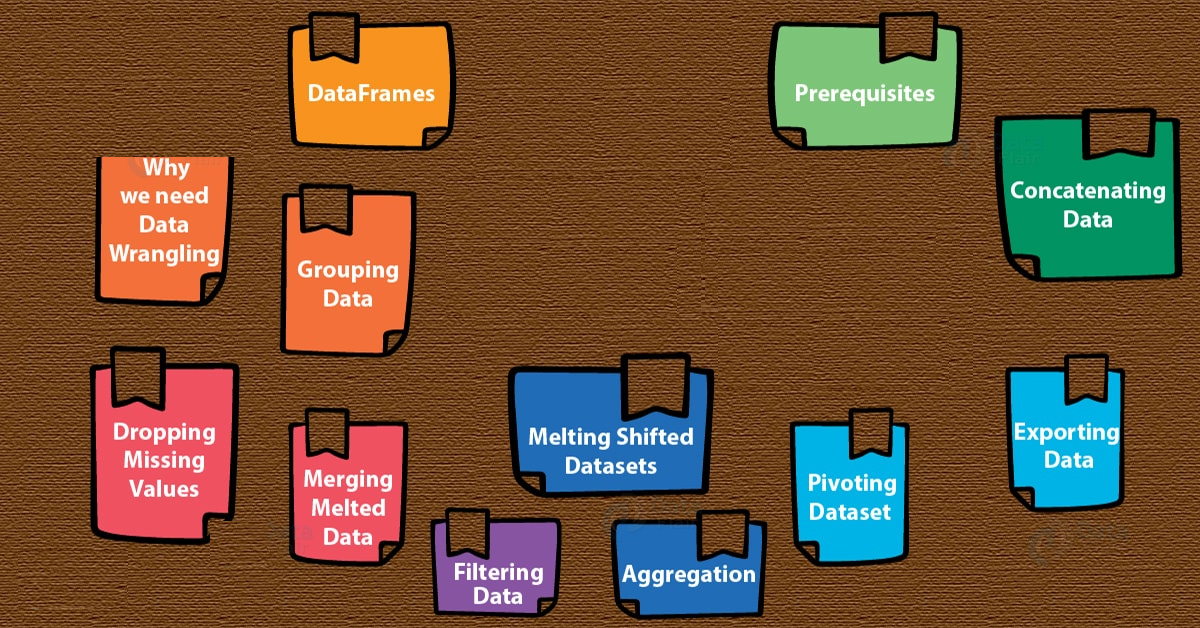
**Spark Hackathon on Interview scenarios**

**Time: 4 Hours**

**This Hackathon helps you to manage the cleansing, scrubbing, curation, cleanup, sanitization, preprocessing, transformation, ETL and schema migration of unpredicted data sets using Spark core and Dataframe functions which can help you solving interview questions by splitting, merging, applying functions in data sets.**

**Please read the below points before proceeding to the Hackathon:**

* **Download the data into HDFS location (/user/hduser/sparkhack2) and start progress.**
* **Import required classes including sparkcontext, sqlcontext and other required objects, classes. Lets import other required libraries later as and when it is needed.**
* **Create Sparkcontext, sqlcontext or spark session.**
* **Provide the final code developed in Eclipsce, you can use REPL for development, but finally code should be in Eclipsce.**
* **Check where ever performance can be improved and add accordingly.**
* **Follow the hints if you struck anywhere for syntax or similar examples, refer pdfs, programs, online, worst case seek for help from others. If you are struck fully in some steps ignore those steps or work on other usecases then comeback and try.**
* **All the use cases given below are categorized and stated with completion %, try to achieve maximum percentage of 100 by completing all scenarios.** 
  + **Note: You can either start 1 and 2 first or 3 and 4 first as per your convenience, but complete all.**



1. **Data cleaning, cleansing, scrubbing (30% Completion)**

**Loading RDDs, remove header, blank lines and impose caseclass schema**

1. **Load** the file1 (insuranceinfo1.csv) from HDFS using textFile API into an RDD insuredata
2. **Remove the header** line from the RDD contains column names.

**Hint:** Use first/take -> convert to rdd -> subtract from rdd created in step 2 or filter etc to achieve the above scenario.

1. Display the count and **show** few rows and check whether header is removed.
2. **Remove the blank lines** in the rdd.

**Hint:** Before splitting, trim it, calculate the length and filter only row size is not = 0.

1. **Map and split** using ‘,’ delimiter.

**Hint:** To avoid getting incomplete rows for eg. (1,inceptez,,,) truncated, use **split(“,”,-1)** instead of split(“,”)

1. **Filter number of fields** are equal to 10.
2. **Add case class** namely insureclass with the field names used as per the header record in the file and apply to the above data to create schemaed RDD.
3. **Take the count of the RDD** created in step 7 and step 1 and print how many rows are removed in the cleanup process of removing fields does not equals 10.
4. **Create another RDD namely rejectdata** and store the row that does not equals 10 fields, and analyze why and provide your view here.

For some of the data values in the last two columns are not there. The data needs to be cleaned as for that particular data, it is not clear that what is the market coverage and whether the insurance is a dental only plan.

1. **Load** the file2 (insuranceinfo2.csv) from HDFS using textFile API into an RDD insuredata2
2. Repeat from step 2 to 9 for this file also and create the final rdd.
3. **Data merging, Deduplication, Performance Tuning & Persistance (20% Completion)**
4. **Merge** the both header removed RDDs derived in steps 7 and 11 into an RDD namely insuredatamerged
5. **Cache** it either to memory or any other **persistence levels** you want, display only **first** few rows
6. **Calculate the count of rdds** created in step 7+11 and rdd in step 12, check whether they are matching.
7. **Remove duplicates** from this merged RDD created in step 12 and print how many duplicate rows are there.
8. **Increase the number of partitions to 8** and name it as insuredatarepart.
9. **Split the above RDD using the businessdate field** into rdd\_20191001 and rdd\_ 20191002 based on the BusinessDate of 2019-10-01 and 2019-10-02 respectively
10. **Store the RDDs** created in step 9, 12, 17 into HDFS locations.
11. **Convert the RDD created in step 16 above into Dataframe** namely insuredaterepartdf applying the structtype created in the step 20 given in the next usecase.

**Hint:** Think of converting df to rdd then use createdataframe to apply schema.

1. **DataFrames operations (20% Completion)**

**Apply Structure, DSL column management functions, transformation, custom udf & schema migration.**

1. Create structuretype for all the columns as per the insuranceinfo1.csv.

**Hint:** Do it carefully without making typo mistakes. Fields issuerid, issuerid2 should be of IntegerType, businessDate should be DateType and all other fields are StringType, ensure to import sql.types library.

1. Create dataframes/datasets using the csv module with option to escape ‘,’ accessing the insuranceinfo1.csv and insuranceinfo2.csv files, apply the schema of the structure type created in the step 20.
2. Apply the below **DSL** functions in the DFs created in step 21.
   1. **Rename** the fields StateCode and SourceName as stcd and srcnm respectively.
   2. **Concat** IssuerId,IssuerId2 as issueridcomposite and make it as a new field

**Hint :** Cast to string and concat.

* 1. **Remove** DentalOnlyPlan column
  2. **Add columns** that should show the current system date and timestamp with the fields name of sysdt and systs respectively.

1. **Remove the rows** contains null in any one of the field and count the number of rows which contains all columns with some value.

**Hint:** Use drop.na options

1. **Custom Method creation:** Create a package (org.inceptez.hack), class (allmethods), method (remspecialchar) **Hint:** First create the function directly and then later add inside pkg, class etc..
   1. Method should take 1 string argument and 1 return of type string
   2. Method should remove all special characters and numbers 0 to 9 - ? , / \_ ( ) [ ] **Hint:** Use replaceAll function, usage of [] symbol should use \\ escape sequence.
   3. For eg. If I pass to the method value as **Pathway - 2X (with dental)** it has to return **Pathway X with dental** as output.
2. Import the package, instantiate the class and register the method generated in step 24 as a udf for invoking in the DSL function.
3. Call the above udf in the DSL by passing NetworkName column as an argument to get the special characters removed.
4. Save the DF generated in step 26 in JSON into HDFS with overwrite option.
5. Save the DF generated in step 26 into CSV format with header name as per the DF and delimited by ~ into HDFS with overwrite option.
6. Save the DF generated in step 26 into hive table with append option.
7. **Tale of handling RDDs, DFs and TempViews (20% Completion)**

**Loading RDDs, split RDDs, Load DFs, Split DFs, Load Views, Split Views, write UDF, register to use in Spark SQL, Transform, Aggregate, store in disk/DB**

**Use RDD functions:**

1. **Load** the file3 (custs\_states.csv) from the HDFS location, using textfile API in an RDD custstates, this file contains 2 type of data one with 5 columns contains customer master info and other data with statecode and description of 2 columns.
2. **Split** the above data into 2 RDDs, first RDD namely custfilter should be loaded only with 5 columns data and second RDD namely statesfilter should be only loaded with 2 columns data.

**Use DSL functions:**

1. **Load** the file3 (custs\_states.csv) from the HDFS location, using CSV Module in a DF custstatesdf, this file contains 2 type of data one with 5 columns contains customer master info and other data with statecode and description of 2 columns.
2. **Split** the above data into 2 DFs, first DF namely custfilterdf should be loaded only with 5 columns data and second DF namely statesfilterdf should be only loaded with 2 columns data.

**Hint:** Use filter DSL function to check isnull or isnotnull to achieve the above functionality then rename and drop columns in the above 2 DFs accordingly.

**Use SQL Queries:**

1. **Register the above DFs as temporary views** as custview and statesview.
2. **Register the DF generated in step 22 as a tempview** namely insureview
3. **Import the package, instantiate the class and Register the method** created in step 24 in the name of remspecialcharudf using **spark udf registration**.
4. Write an **SQL query** with the below processing

**Hint:** Try doing the below , step by step, skip if you can’t do it, later try.

* 1. Pass NetworkName to remspecialcharudf and get the new column called cleannetworkname
  2. Add current date, current timestamp fields as curdt and curts.
  3. Extract the year and month from the businessdate field and get it as 2 new fields called yr,mth respectively.
  4. Extract from the protocol either http/https from the NetworkURL column, if no protocol found then display **noprotocol.** For Eg: if <http://www2.dentemax.com/> then show http if [www.bridgespanhealth.com](http://www.bridgespanhealth.com) then show as **noprotocol** store in a column called protocol.
  5. Display all the columns from insureview including the columns derived from above a, b, c, d steps with statedesc column from statesview with age,profession column from custview . Do an Inner Join of insureview with statesview using stcd=stated and join insureview with custview using custnum=custid.

1. Store the above selected Dataframe in **Parquet** formats in a HDFS location as a **single file**.
2. Write an SQL query to identify average age, count group by statedesc, protocol, profession.
3. Store the DF generated in step 39 into MYSQL table insureinfo *if time permits.*
4. **Visualization (10% Completion)**

Login to spark ui and take the snapshot of the below items.

1. Jobs
2. Stages

**Display the below items**

* 1. Task locality level in the Tasks
  2. Scheduler Delay
  3. Task Deserialization Time
  4. Shuffle Read Blocked Time
  5. Shuffle Remote Reads
  6. Result Serialization Time
  7. Getting Result Time
  8. Peak Execution Memory

1. Storage
   1. Storage level
   2. Partitions info
2. Executors
3. SQL
   1. DAG
   2. AST (Abstract Syntax Tree plan)

