Целью задачи является создать файл csv файл, в котором сохранены некоторые параметры. Сначала избавляемся от мусорных логов, выставляем логгер на Level.ERROR. Теперь будут выводится только сообщения об ошибках, а не весь процесс.

Далее создается вспомогательный класс Repository, в конструктор передается: путь к winutils, а точнее к папке /bin, в которой он лежит. Я положил его в проект, однако почему то у меня не работает путь к нему в папке data. Так что вам стоит указать его самостоятельно. PATH\_TO\_CLIENTS\_CSV, PATH\_TO\_TRANSACTIONS\_CSV, PATH\_TO\_TERMINAL\_CSV – соответственно, пути к файлам csv, мои файлы лежат в папке data. Что происходит в конструкторе Repository? Во первых, инициализируется сессия по методу Connection.getSession, где в методе getSession я забыл поменять хардкод “C:\\Spark\\” на pathToWinutils.

Далее делаем датасеты с помощью наших csv файлов методом createDataset(pathToClientsCsv), параметром идет путь к csv файлу, сам метод вызывает loadFromCsv. Но перед этим, так как в условии задачи у нас в качестве разделителя используется точка с запятой “;” а не запятая, появляются проблемы. У датасета есть метод .csv(String path) принимающий на вход путь к файлу, и парсящий его по запятым. Логично было бы передавать вторым параметром, допустим, разделитель, но к сожалению это так не работает. А так же я копался в других методах датасета, пытаясь заменить его. В итоге пришлось написать свой собственный метод, чтобы парсить csv файлы, csvParser(String Path). Он выполняет две функции: первая, собственно, меняет точку с запятой на запятую, теперь с таким файлом уже может корректно работать датасет. Проблемы были в том, что почему то после такого преобразования автоматически добавляются ковычки с обеих сторон, так что пришлось их заменять на пустую строку (line[i] + ",").replace("\"", ""). Далее в else прописано, что если значение последнее, после него следует перейти на другую строку.

Вторая проблема: если не хватает каких – то данных, пропускаем их. Поэтому мы сплитим хидер в массив String[] columns. Сравниваем длину массива в каждой итерации с длиной поля, и если они не равны друг другу, пропускаем такие данные. Допустим, у нас есть “client\_id, name, lastName” – длина хидера будет 3. И если в строке “2, Oleg” всего два значения, то такая строка пропустится.