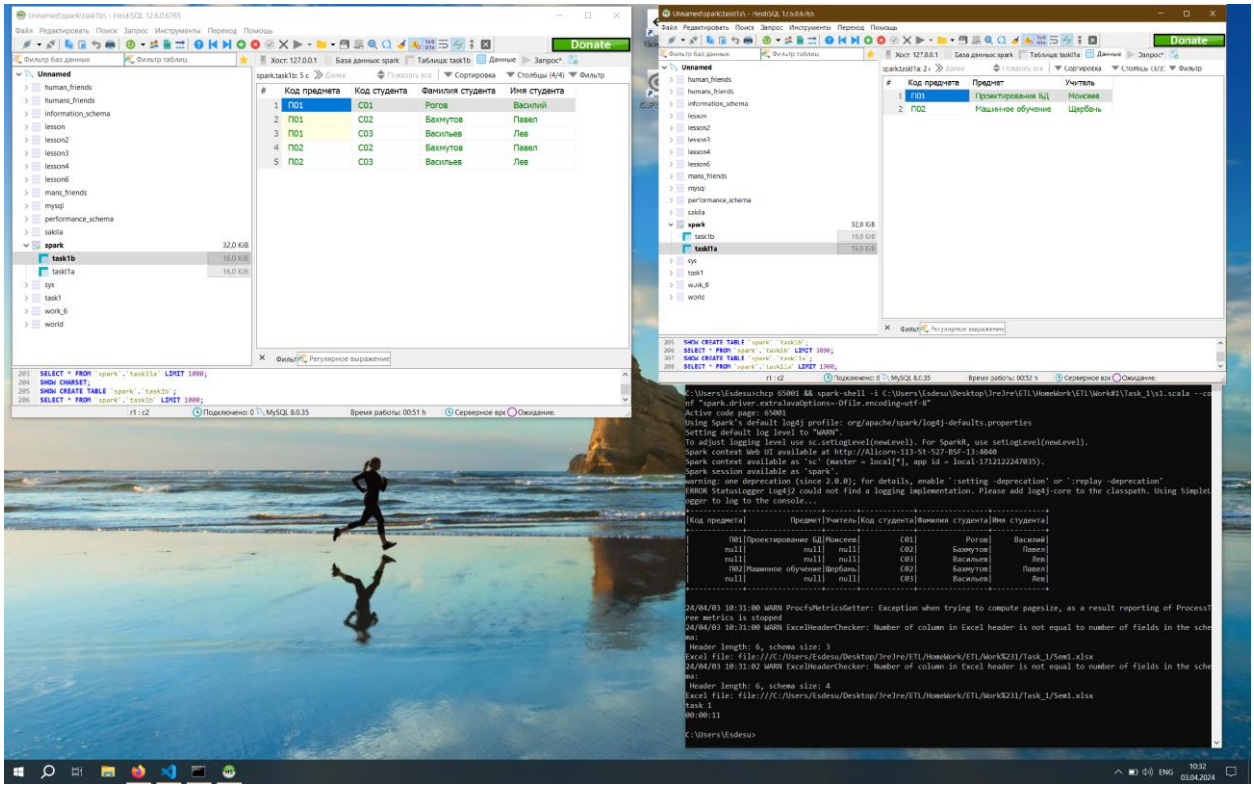
Урок 1. Модели данных и нормализация таблиц. Схема «звезда»

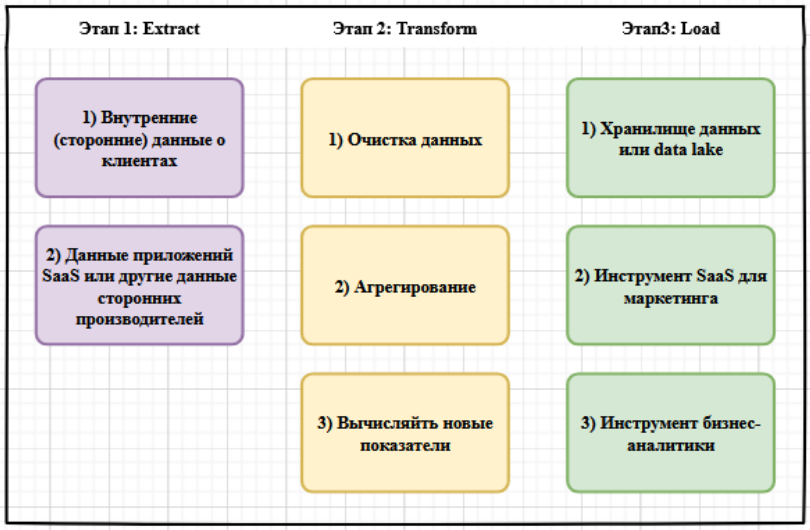
1. Условия ДЗ написаны на слайде 44 s1.pdf. Если не получиться, то просто запустите скрипт s1.scala, скоректируйте пути. В рамках первого семинара вы установили инструментарий ETL.

```
/*
chcp 65001 && spark-shell -i
C:\Users\Esdesu\Desktop\JreJre\ETL\HomeWork\ETL\Work#1\Task_1\s1.scala --conf
"spark.driver.extraJavaOptions=-Dfile.encoding=utf-8"
*/

import org.apache.spark.internal.Logging
import org.apache.spark.sql.functions.{col, collect_list, concat_ws}
import org.apache.spark.sql.{DataFrame, SparkSession}
import org.apache.spark.sql.expressions.Window
import java.io.{File, FileInputStream}
import java.util.Properties

val filePath = "C:/Users/Esdesu/Desktop/JreJre/ETL/config.properties"
val prop = new Properties()
val file = new File(filePath)
val fis = new FileInputStream(file)
prop.load(fis)

val login = prop.getProperty("login")
val password = prop.getProperty("password")

var sqlCoun = s"jdbc:mysql://localhost:3306/spark?user=$login&password=$password"
var driver = "com.mysql.cj.jdbc.Driver"

val t1 = System.currentTimeMillis()

if(1==1){
    var df1 = spark.read.format("com.crealytics.spark.excel")
        .option("sheetName", "Sheet1")
        .option("useHeader", "false")
        .option("treatEmptyValuesAsNulls", "false")
        .option("inferSchema", "true").option("addColorColumns", "true")
        .option("usePlainNumberFormat","true")
        .option("startColumn", 0)
        .option("endColumn", 99)
        .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")
        .option("maxRowsInMemory", 20)
        .option("excerptSize", 10)
        .option("header", "true")
        .format("excel")
        .load("C:/Users/Esdesu/Desktop/JreJre/ETL/HomeWork/ETL/Work#1/Task_1/Sem1
.xlsx")
    df1.show()
```

```
        df1.filter(col("Код предмета").isNotNull).select("Код
предмета","Предмет","Учитель")
            .write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "w1task1a")
            .mode("overwrite").save()

    val window1 =
Window.partitionBy(lit(1)).orderBy(("id")).rowsBetween(Window.unboundedPreceding,
Window.currentRow)
        df1.withColumn("id", monotonicallyIncreasingId())
            .withColumn("Код предмета", when(col("Код предмета").isNull,
last("Код предмета", ignoreNulls = true).over(window1)).otherwise(col("Код
предмета")))
            .orderBy("id").drop("id","Предмет","Учитель")
            .write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "w1task1b")
            .mode("overwrite").save()

    println("Work 1, Task 1, Successful Load and Save")
}

val s0 = (System.currentTimeMillis() - t1)/1000
val s = s0 % 60
val m = (s0/60) % 60
val h = (s0/60/60) % 24
println("%02d:%02d:%02d".format(h, m, s))
System.exit(0)
```



2. Нарисуйте архитектуру ETL процесса для сбора и анализа данных компанией которая хочет провести маркетинговую кампанию, используя app.diagrams.net. Сделайте описание почему вы считаете что архитектура должна выглядеть именно так.



Этап 1: Извлечение данных

Каждая маркетинговая команда использует данные из разных источников:
1) Внутренние (сторонние) данные о клиентах: к ним относятся файлы с информацией о клиентах (CSV, Microsoft Excel, Google Sheets и т.д.), наборы данных с покупками клиентов и информацией о доставке (например, в локальной реляционной базе данных SQL или облачном хранилище, таком как Amazon S3, для транзакций электронной коммерции).
2) Данные приложений SaaS или другие данные сторонних производителей: В эту категорию входят электронные письма клиентов для CRM-платформ, таких как Hubspot и Salesforce, рекламные данные с Facebook ads и Google ads, взаимодействия с веб-сайтами из Google Analytics и информация о продуктах поставщика услуг электронной коммерции.

Этап 2: Преобразование данных

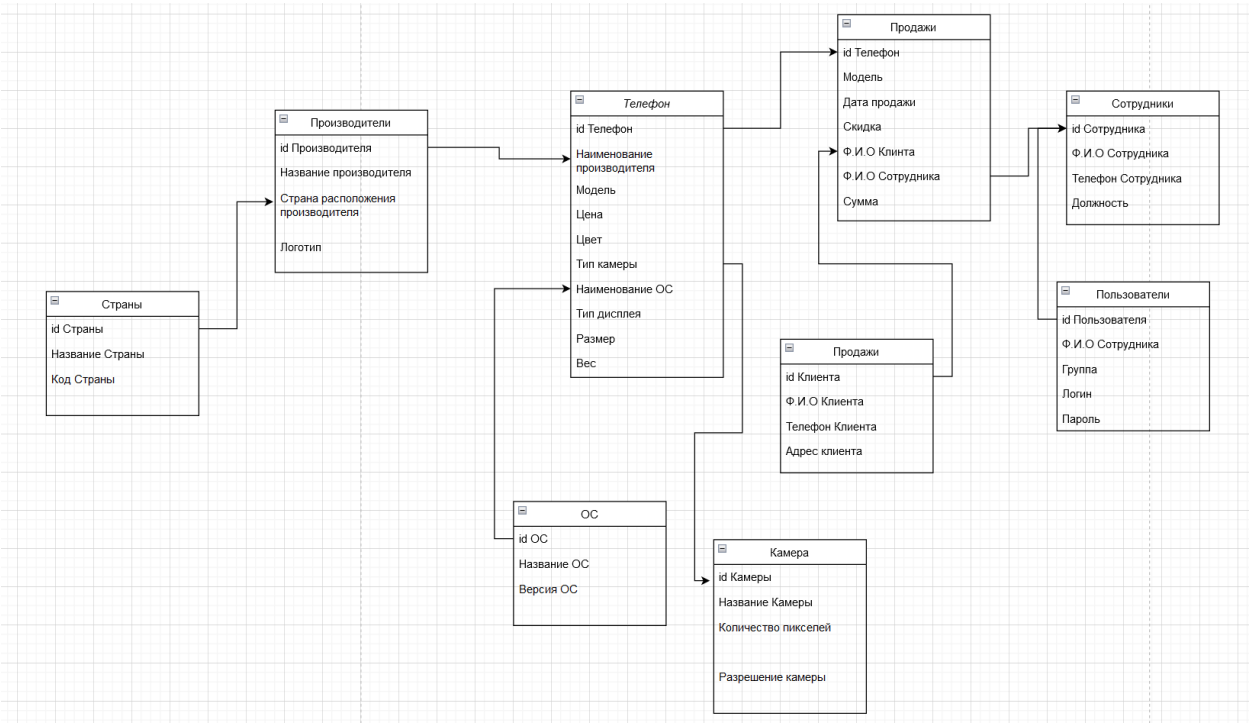
Преобразование данных включает в себя следующие этапы:
1) Очистка данных: Вы удаляете дублированные или поврежденные записи, отфильтровываете информацию, которая не нужна для маркетинговой деятельности, а также ошибки и несоответствия из ваших наборов данных.
2) Агрегирование данных: данные, которые собираются на этапе 1, часто слишком детализированы. Например не нужно, чтобы каждое событие, связанное с кликом клиента, демонстрировалось отдельно. Вместо этого нам нужно агрегированное общее количество кликов, чтобы определить, какая реклама показала наилучшие результаты.
3) Вычисляйть новые показатели: данные, которые извлекаются из различных источников, не всегда оптимизированы для анализа. Чтобы решить эту проблему, необходимо рассчитать свои маркетинговые показатели, такие как ARPU или пожизненная ценность клиента.

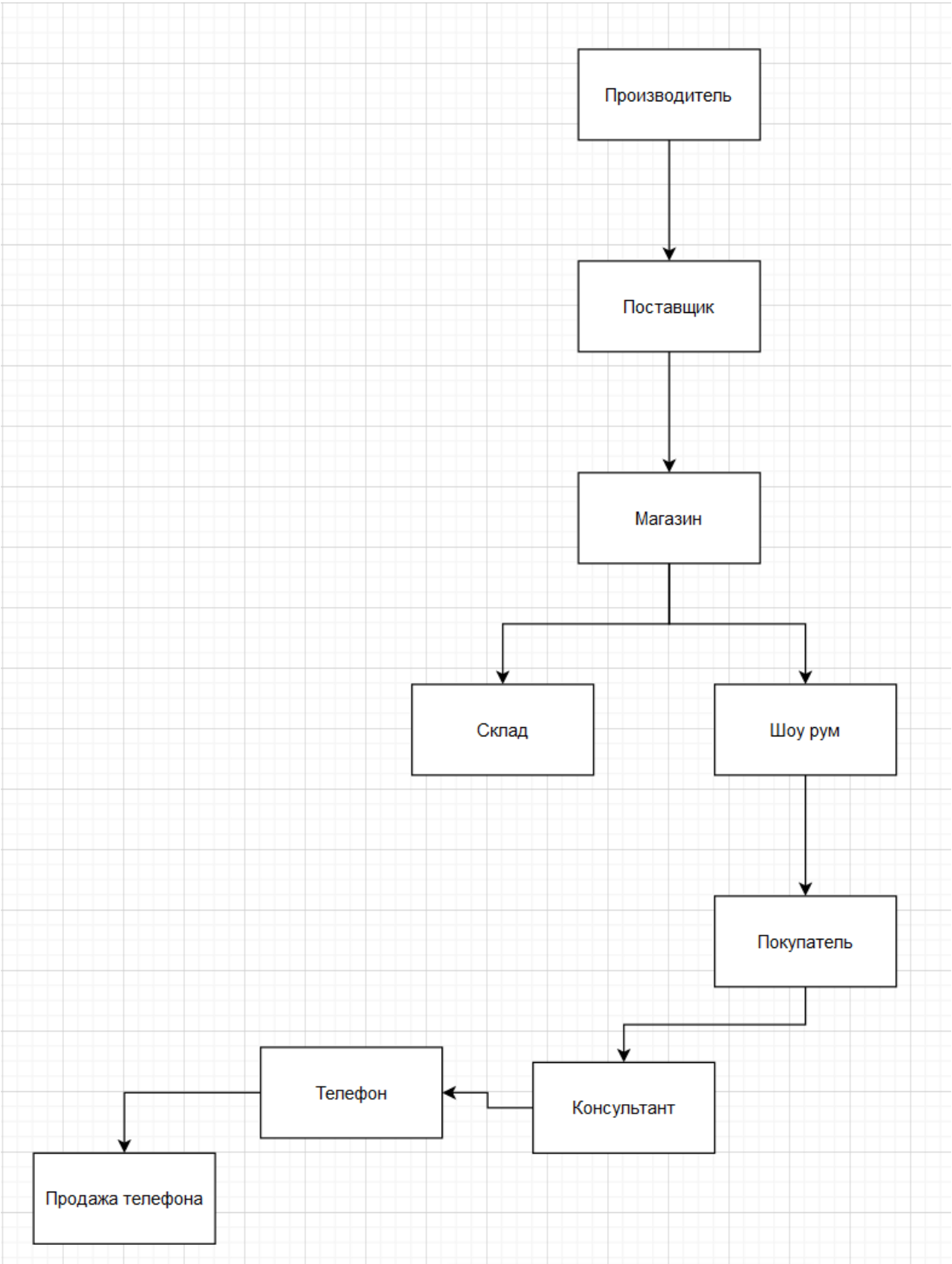
Этап 3: Загрузка данных

Загрузка данных, проще говоря, означает сохранение данных. Ваши варианты включают:

1) Хранилище данных или data lake: Наилучший способ переместить очищенные данные в хранилище данных. Наличие данных в Snowflake или BigQuery поможет создать "единый источник достоверности". Единое хранилище для всех ваших данных о клиентах, которое обеспечивает согласованность ваших данных. Хранилище данных также позволяет членам команды получать доступ к одним и тем же данным без ущерба для согласованности данных или дублирования работы с ETL.
2) Инструмент SaaS для маркетинга: используйте свои данные в инструменте SaaS, чтобы подготовить их к маркетинговым кампаниям. Например, когда создается набор данных, включающий всех, кто приобрел ваши последние продукты, вы можете направить этот набор данных в Hubspot (список адресов электронной почты) или Facebook Ads (аудитории) для проведения гипертаргетированных кампаний
3) Инструмент бизнес-аналитики: загружая данные в такие инструменты, как PowerBI, Tableau или Google Data Studio, данные сразу же готовятся к анализу. Создавайте информационные панели с маркетинговыми показателями и ключевыми показателями эффективности, чтобы выявлять тенденции.

3. Постройте реляционную и иерархическую модели данных для магазина который продает телефоны.


Продажи
id Телефон
Модель
Дата продажи
Скидка
Ф.И.О Клинта
Ф.И.О Сотрудника
Сумма
Сотрудники
id Сотрудника
Ф.И.О Сотрудника
Телефон Сотрудника
Должность
Производители
id Производителя
Название производителя
Страна расположения производителя
Логотип
Телефон
id Телефон
Наименование производителя
Модель
Цена
Цвет
Тип камеры
Наименование ОС
Тип дисплея
Размер
Вес
Страны
id Страны
Название Страны
Код Страны
Пользователи
id Пользователя
Ф.И.О Сотрудника
Группа
Логин
Пароль
Продажи
id Клиента
Ф.И.О Клиента
Телефон Клиента
Адрес клиента
ОС
id ОС
Название ОС
Версия ОС
Камера
id Камеры
Название Камеры
Количество пикселей
Разрешение камеры



4. Определите в какой нормальной форме данная таблица, приведите ее ко 2 и 3 нормальным формам последовательно.

Нужно в apache spark создать таблицу сданными ниже. Можно импортировать из Excel либо с генерировать ее кодом.

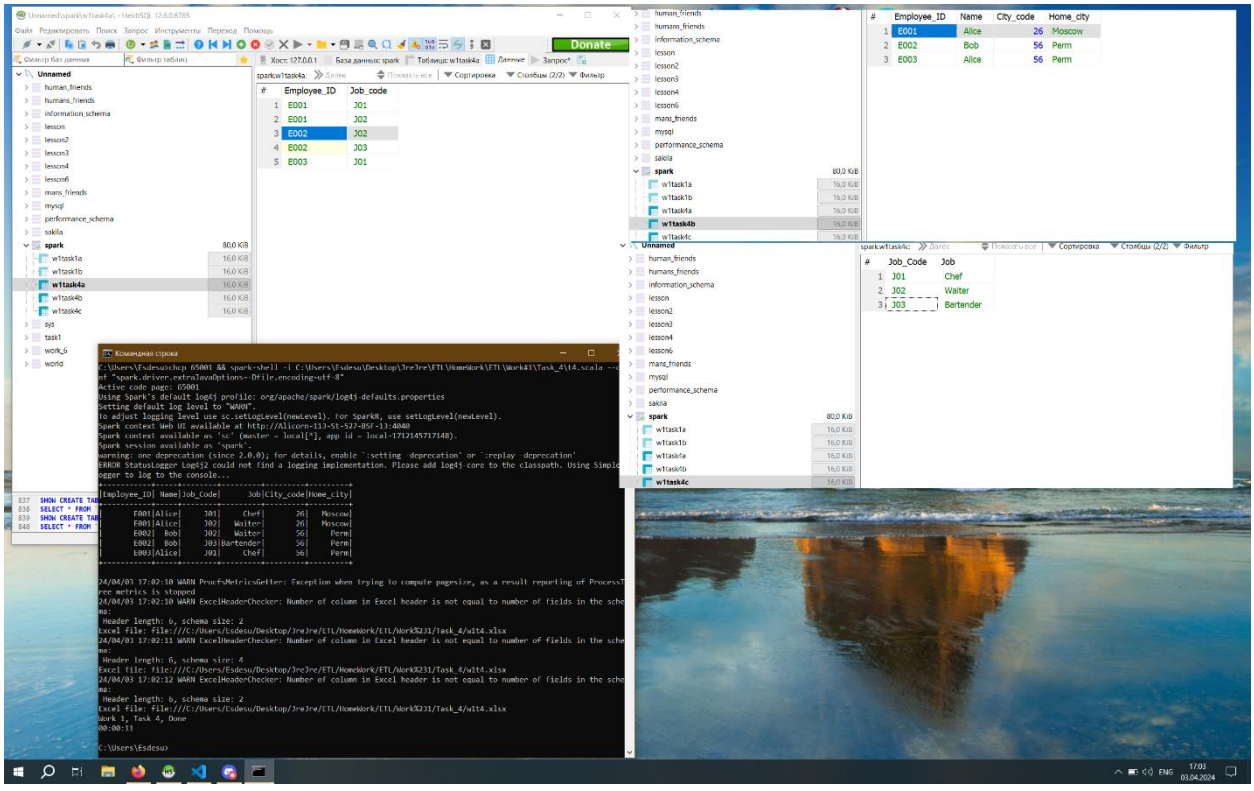
Код:

```scala
/*
chcp 65001 && spark-shell -i
C:\Users\Esdesu\Desktop\JreJre\ETL\HomeWork\ETL\Work#1\Task_4\t4.scala --conf
"spark.driver.extraJavaOptions=-Dfile.encoding=utf-8"
*/

import org.apache.spark.internal.Logging
import org.apache.spark.sql.functions.{col, collect_list, concat_ws}
import org.apache.spark.sql.{DataFrame, SparkSession}
import org.apache.spark.sql.expressions.Window
import java.io.{File, FileInputStream}
import java.util.Properties

val filePath = "C:/Users/Esdesu/Desktop/JreJre/ETL/config.properties"
val prop = new Properties()
val file = new File(filePath)
val fis = new FileInputStream(file)
prop.load(fis)

val login = prop.getProperty("login")
val password = prop.getProperty("password")

var sqlCoun = s"jdbc:mysql://localhost:3306/spark?user=$login&password=$password"
var driver = "com.mysql.cj.jdbc.Driver"

val t1 = System.currentTimeMillis()

if(1==1){
    var df1 = spark.read.format("com.crealytics.spark.excel")
        .option("sheetName", "Sheet1")
        .option("useHeader", "false")
        .option("treatEmptyValuesAsNulls", "false")
        .option("inferSchema", "true").option("addColorColumns", "true")
        .option("usePlainNumberFormat","true")
        .option("startColumn", 0)
        .option("endColumn", 99)
        .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")
        .option("maxRowsInMemory", 20)
        .option("excerptSize", 10)
        .option("header", "true")
        .format("excel")
        .load("C:/Users/Esdesu/Desktop/JreJre/ETL/HomeWork/ETL/Work#1/Task_4/w1t4
.xlsx")
    df1.show()

        df1.filter(col("Employee_ID").isNotNull).select("Employee_ID",
"Job_code")
            .write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "w1task4a")
            .mode("overwrite").save()
```

```
    val nf2 =
Window.partitionBy(lit(1)).orderBy(("id")).rowsBetween(Window.unboundedPreceding,
Window.currentRow)

    val df2 = df1.withColumn("id", monotonicallyIncreasingId())

        df2.withColumn("Employee_ID", when(col("Employee_ID").isNull,
last("Employee_ID", ignoreNulls = true).over(nf2)).otherwise(col("Employee_ID")))
            .withColumn("table", lit("w1task4b"))

            .orderBy("id").drop("id", "Job_Code", "Job")
            .filter(col("table") === "w1task4b")
            .dropDuplicates()
            .drop("table")
            .write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "w1task4b")
            .mode("overwrite").save()

        df2.withColumn("table", lit("w1task4c"))
            .orderBy("id").drop("id", "Employee_ID", "Name", "City_code",
"Home_city")
            .filter(col("table") === "w1task4c")
            .dropDuplicates()
            .drop("table")
            .write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "w1task4c")
            .mode("overwrite").save()

    val nf3 =
Window.partitionBy(lit(1)).orderBy(("id")).rowsBetween(Window.unboundedPreceding,
Window.currentRow)

    val df3 = df1.withColumn("id", monotonicallyIncreasingId())

        df3.withColumn("Employee_ID", when(col("Employee_ID").isNull,
last("Employee_ID", ignoreNulls = true).over(nf2)).otherwise(col("Employee_ID")))
            .withColumn("table", lit("w1task4b"))

            .orderBy("id").drop("id", "Job_Code", "Job", "Home_city")
            .filter(col("table") === "w1task4b")
            .dropDuplicates()
            .drop("table")
            .write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "w1task4b")
            .mode("overwrite").save()

        df3.withColumn("table", lit("w1task4c"))
            .orderBy("id").drop("id", "Employee_ID", "Name", "City_code",
"Home_city")
            .filter(col("table") === "w1task4c")
            .dropDuplicates()
```

```
            .drop("table")
            .write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "w1task4c")
            .mode("overwrite").save()

        df3.withColumn("table", lit("w1task4d"))
            .orderBy("id").drop("id", "Employee_ID", "Name", "Job_Code", "Job")
            .filter(col("table") === "w1task4d")
            .dropDuplicates()
            .drop("table")
            .write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "w1task4d")
            .mode("overwrite").save()

    println("Work 1, Task 4, Successful Load and Save")
}

val s0 = (System.currentTimeMillis() - t1)/1000
val s = s0 % 60
val m = (s0/60) % 60
val h = (s0/60/60) % 24
println("%02d:%02d:%02d".format(h, m, s))
System.exit(0)
```

NF2:



NF3: