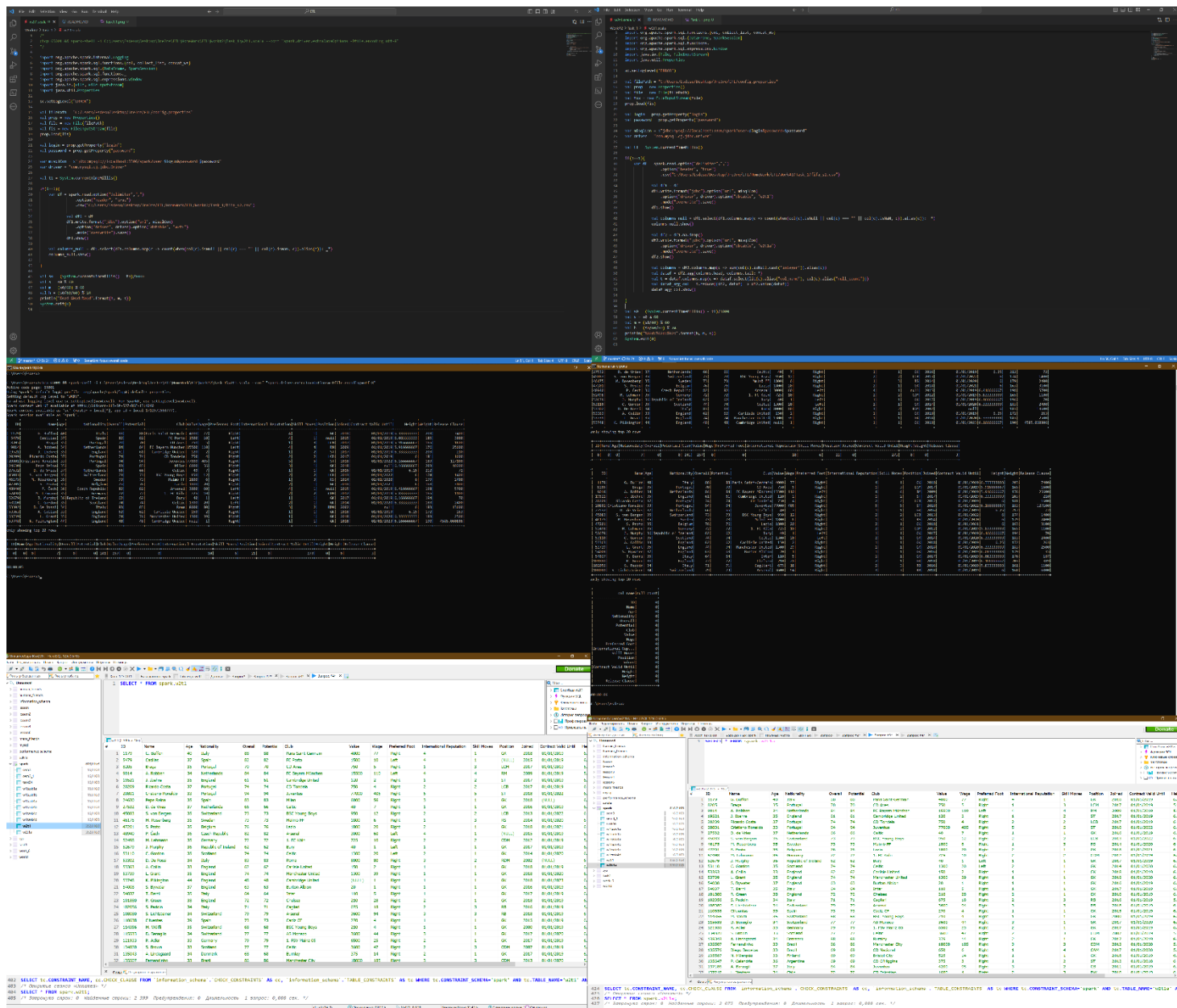


Урок 2. Введение в подготовку данных для аналитиков. Таблицы фактов и таблицы измерений

1. Скачайте датасет `fifas2.csv`. Проанализируйте его и определите, какие данные являются неполными. Удалите ненужные колонки и недостающие значения.
 2. Найдите в датафрейме полные дубликаты и удалите их. Значения могут быть одинаковыми, но написаны по-разному. Например, может отличаться размер регистра (заглавные и строчные буквы). Особое внимание уделить колонке с названиями команд.
 3. Напишите функцию, которая добавит колонку с разбиением возраста по группам: до 20, от 20 до 30, от 30 до 36 и старше 36. Посчитайте количество футболистов в каждой категории.
- ЗАДАНИЕ НУЖНО СДЕЛАТЬ С ПОМОЧЬЮ APACHE SPARK КАК ПОКАЗАНО НА СЕМИНАРЕ.
- Соберите все в один pdf, код и скриншоты записанных таблиц с данными, скриншоты отработки лога спарка.



File Edit View Database Tools Help

Python 3.7.4 Shell

Python 3.7.4 Shell

```
def main():
    # Create a new SQLite database
    conn = sqlite3.connect('data.db')
    cur = conn.cursor()

    # Create a table for storing data
    cur.execute('CREATE TABLE IF NOT EXISTS data (id INTEGER PRIMARY KEY, name TEXT, age INTEGER, gender TEXT, blood TEXT, height INTEGER, weight INTEGER, date TEXT)')

    # Insert data into the table
    cur.execute('INSERT INTO data (name, age, gender, blood, height, weight, date) VALUES (?, ?, ?, ?, ?, ?, ?)')

    # Commit the changes
    conn.commit()

    # Close the connection
    conn.close()

if __name__ == '__main__':
    main()
```

Python 3.7.4 Shell

Python 3.7.4 Shell

| id | name | age | gender | blood | height | weight | date |
|----|----------------|-----|--------|-------|--------|--------|------------|
| 1 | John Doe | 30 | Male | A | 180 | 75 | 2019-01-01 |
| 2 | Jane Smith | 25 | Female | B | 165 | 60 | 2019-02-01 |
| 3 | Mike Johnson | 35 | Male | O | 190 | 85 | 2019-03-01 |
| 4 | Sarah Brown | 28 | Female | A | 170 | 65 | 2019-04-01 |
| 5 | David Wilson | 32 | Male | B | 185 | 70 | 2019-05-01 |
| 6 | Emily Davis | 27 | Female | O | 160 | 55 | 2019-06-01 |
| 7 | Chris Miller | 31 | Male | A | 175 | 68 | 2019-07-01 |
| 8 | Alice Taylor | 29 | Female | B | 172 | 62 | 2019-08-01 |
| 9 | Bob Anderson | 33 | Male | O | 188 | 78 | 2019-09-01 |
| 10 | Grace White | 26 | Female | A | 168 | 58 | 2019-10-01 |
| 11 | Frank Green | 34 | Male | B | 192 | 82 | 2019-11-01 |
| 12 | Heidi Black | 24 | Female | O | 162 | 52 | 2019-12-01 |
| 13 | Timothy Gray | 36 | Male | A | 195 | 88 | 2020-01-01 |
| 14 | Laura King | 23 | Female | B | 158 | 50 | 2020-02-01 |
| 15 | Kevin Lee | 37 | Male | O | 198 | 90 | 2020-03-01 |
| 16 | Nicole Hall | 22 | Female | A | 155 | 48 | 2020-04-01 |
| 17 | Brandon Young | 38 | Male | B | 200 | 92 | 2020-05-01 |
| 18 | Sophia Scott | 21 | Female | O | 152 | 45 | 2020-06-01 |
| 19 | Benjamin Adams | 39 | Male | A | 202 | 95 | 2020-07-01 |
| 20 | Olivia Baker | 20 | Female | B | 150 | 42 | 2020-08-01 |

Python 3.7.4 Shell

Python 3.7.4 Shell

```
SELECT * FROM data;
```

| id | name | age | gender | blood | height | weight | date |
|----|----------------|-----|--------|-------|--------|--------|------------|
| 1 | John Doe | 30 | Male | A | 180 | 75 | 2019-01-01 |
| 2 | Jane Smith | 25 | Female | B | 165 | 60 | 2019-02-01 |
| 3 | Mike Johnson | 35 | Male | O | 190 | 85 | 2019-03-01 |
| 4 | Sarah Brown | 28 | Female | A | 170 | 65 | 2019-04-01 |
| 5 | David Wilson | 32 | Male | B | 185 | 70 | 2019-05-01 |
| 6 | Emily Davis | 27 | Female | O | 160 | 55 | 2019-06-01 |
| 7 | Chris Miller | 31 | Male | A | 175 | 68 | 2019-07-01 |
| 8 | Alice Taylor | 29 | Female | B | 172 | 62 | 2019-08-01 |
| 9 | Bob Anderson | 33 | Male | O | 188 | 78 | 2019-09-01 |
| 10 | Grace White | 26 | Female | A | 168 | 58 | 2019-10-01 |
| 11 | Frank Green | 34 | Male | B | 192 | 82 | 2019-11-01 |
| 12 | Heidi Black | 24 | Female | O | 162 | 52 | 2019-12-01 |
| 13 | Timothy Gray | 36 | Male | A | 195 | 88 | 2020-01-01 |
| 14 | Laura King | 23 | Female | B | 158 | 50 | 2020-02-01 |
| 15 | Kevin Lee | 37 | Male | O | 198 | 90 | 2020-03-01 |
| 16 | Nicole Hall | 22 | Female | A | 155 | 48 | 2020-04-01 |
| 17 | Brandon Young | 38 | Male | B | 200 | 92 | 2020-05-01 |
| 18 | Sophia Scott | 21 | Female | O | 152 | 45 | 2020-06-01 |
| 19 | Benjamin Adams | 39 | Male | A | 202 | 95 | 2020-07-01 |
| 20 | Olivia Baker | 20 | Female | B | 150 | 42 | 2020-08-01 |

SQL Editor

```
SELECT * FROM ...
```

Query Results

| id | name | value | ... |
|----|------|-------|-----|
| 1 | ... | ... | ... |
| 2 | ... | ... | ... |

SQL Editor

```
SELECT * FROM ...
```

Query Results

| id | name | value | ... |
|----|------|-------|-----|
| 1 | ... | ... | ... |
| 2 | ... | ... | ... |

```

/*
chcp 65001 && spark-shell -i C:\Users\Esdesu\Desktop\JreJre\ETL\Homework\ETL\Work#2\Task_1\w2t1.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.functions._
import org.apache.spark.sql.expressions.Window
import java.io.{File, FileInputStream}
import java.util.Properties

sc.setLogLevel("ERROR")

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 misqlCon = 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 df = spark.read.option("delimiter",",")
                        .option("header", "true")

```

```

.csv("C:/Users/Esdesu/Desktop/JreJre/ETL/HomeWork/ETL/Work#2/Task_1/fifa_s2.csv")

val df1 = df
df1.write.format("jdbc").option("url", misqlCon)
    .option("driver", driver).option("dbtable", "w2t1")
    .mode("overwrite").save()
df1.show()

val columns_null = df1.select(df1.columns.map(c => count(when(col(c).isNull || col(c) === "" || col(c).isNaN,
c)).alias(c)):_*)
columns_null.show()

val df2 = df1.na.drop()
df2.write.format("jdbc").option("url", misqlCon)
    .option("driver", driver).option("dbtable", "w2t1a")
    .mode("overwrite").save()
df2.show()

val columns = df2.columns.map(c => sum(col(c).isNull.cast("integer")).alias(c))
val dataf = df2.agg(columns.head, columns.tail:_)
val t = dataf.columns.map(c => dataf.select(lit(c).alias("col_name"), col(c).alias("null_count")))
val dataf_agg_col = t.reduce((df2, dataf) => df2.union(dataf))
dataf_agg_col.show()

val df3 = df2
    .withColumn("Name", lower(col("Name")))
    .withColumn("Nationality", lower(col("Nationality")))
    .withColumn("Club", lower(col("Club")))
    .withColumn("Preferred Foot", lower(col("Preferred Foot")))
    .withColumn("Position", lower(col("Position")))
    .dropDuplicates()
df3.write.format("jdbc").option("url", misqlCon)

```

```

        .option("driver", driver).option("dbtable", "w2t1b")
        .mode("overwrite").save()
df3.show()

val df4 = df3
    .withColumn("Age Group", when(col("Age") < 20, "0-19")
    .when(col("Age") >= 20 && col("Age") < 30, "20-29")
    .when(col("Age") >= 30 && col("Age") < 36, "30-35")
    .otherwise("36+"))

// val agePlayerSum = df4.groupBy("Age Group").count() // Колонки без изменения названия
// val agePlayerSum = df4.groupBy("Age Group").agg(count("*").alias("Sum Players for Age")) // Меняет наименование
последней колонки
val agePlayerSum = df4.groupBy("Age Group").count().toDF("Age Group Player", "Sum Players for Age") // Менет
наименование обеих колонок
agePlayerSum.show()

df4.write.format("jdbc").option("url", misqlCon)
    .option("driver", driver).option("dbtable", "w2t1c")
    .mode("overwrite").save()
df4.show()
}

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)

```