## 1. Введение

## Современный этап развития информационных технологий характеризуется неуклонным ростом объёмов генерируемых данных, что обусловлено широким распространением цифровых устройств, сенсоров, приложений и онлайн-сервисов. Эти данные, зачастую разнородные, поступают в системы обработки в реальном времени и требуют применения специализированных подходов и инструментов, способных эффективно справляться с их анализом, хранением и визуализацией. Возникает потребность в так называемых «технологиях работы с большими данными» (Big Data), которые позволяют обрабатывать терабайты и петабайты информации за приемлемое время.

## Одним из мощнейших фреймворков для работы с большими данными является Apache Spark — открытая платформа, поддерживающая распределённую обработку данных в памяти, что обеспечивает высокую скорость выполнения аналитических задач. Одним из компонентов Spark является библиотека PySpark, предоставляющая интерфейс для взаимодействия с кластером Spark с помощью языка Python. Это позволяет совмещать выразительность и удобство Python с масштабируемостью и производительностью Spark.

## Данная лабораторная работа направлена на получение практических навыков работы с PySpark в контексте выполнения разведочного анализа данных (Exploratory Data Analysis, EDA). Анализ проводится на примере датасета, содержащего информацию о физических активностях пользователей, собранную с помощью приложения Endomondo. В процессе работы применяется облачная среда Google Colab, обеспечивающая удобный интерфейс для запуска кода, визуализации и взаимодействия с данными без необходимости настройки локального окружения.

## Основными задачами лабораторной работы являются: установка и настройка Spark в Colab, загрузка и первичная обработка данных, выполнение анализа структуры датасета, выявление пропущенных значений и выбросов, визуализация распределения признаков, расчёт статистических характеристик и реализация пользовательских функций для расширенной обработки данных.

## 2. Используемая среда

## Для выполнения лабораторной работы выбрана облачная среда Google Colab, которая предоставляет бесплатный доступ к вычислительным ресурсам, включая поддержку Jupyter-ноутбуков, работу с файлами и библиотеками Python, подключение к Google Drive и многое другое. Это делает её идеальной для обучения и выполнения экспериментальных проектов без необходимости установки программного обеспечения на локальный компьютер.

## Используемые компоненты:

## Язык программирования: Python 3 Python используется как основной язык для разработки аналитических и научных приложений благодаря богатому экосистему библиотек и удобному синтаксису.

## Среда выполнения: Google Colab Коллаб — это облачный Jupyter-ноутбук, поддерживающий интерактивный код, визуализацию и Markdown.

## Фреймворк: Apache Spark (PySpark) PySpark обеспечивает интерфейс для запуска Spark-приложений с помощью Python. Он используется для работы с распределёнными данными, выполнения SQL-запросов, построения моделей и анализа больших объёмов информации.

## Инструменты визуализации: Matplotlib Библиотека Matplotlib применяется для построения графиков и визуализации результатов анализа. Это позволяет наглядно представить распределение признаков и выявить аномалии в данных.

## 3. Ход выполнения работы

## 1. Установка и настройка Apache Spark в Google Colab

## На первом этапе был выполнен импорт Spark в среду Google Colab. Для этого были установлены необходимые компоненты, включая Java 8 и сам Apache Spark (в архиве spark-3.1.2-bin-hadoop2.7.tgz). Также была произведена настройка переменных окружения SPARK\_HOME и JAVA\_HOME, а с помощью библиотеки findspark инициализирована сессия SparkSession.

## 2. Загрузка и чтение файла endomondoHR.json

## Затем был загружен JSON-файл endomondoHR.json, содержащий записи с показателями активности пользователей. Этот файл был считан в формате Spark DataFrame с помощью метода spark.read.json(). Это позволило сразу применить параллельную загрузку и подготовку данных к последующему анализу.

## 3. Анализ структуры датасета с помощью printSchema()

## Для изучения структуры набора данных использовалась функция printSchema(), которая вывела типы всех признаков и позволила понять, какие из них являются массивами, строками, числовыми значениями и т.д. Это дало понимание, как именно необходимо обращаться с каждым столбцом в процессе анализа.

## 4. Подсчёт пропущенных значений

## С помощью конструкции isNull() и агрегирующих функций count() был произведён анализ на наличие пропущенных значений в каждом из признаков. Это важно для понимания полноты данных и необходимости их очистки или обработки перед анализом.

## 5. Расчёт описательной статистики по признаку heart\_rate (пульс)

## Была выполнена генерация статистических характеристик пульса: среднее значение, медиана, минимумы и максимумы, стандартное отклонение и квартили. Это позволило определить характер распределения признака и выявить потенциальные выбросы или аномалии.

## 6. Построение гистограммы пульса

## С использованием библиотеки Matplotlib было построено графическое распределение пульса в виде гистограммы. Это помогло наглядно проанализировать, как часто встречаются различные значения пульса и выделить основные зоны скопления значений.

## 7. Вычисление корреляции между признаками heart\_rate и speed

## Для выявления возможной зависимости между скоростью движения и пульсом использовался метод .corr(). Результат показал числовое значение коэффициента корреляции, на основе которого можно сделать вывод о наличии или отсутствии взаимосвязи между этими признаками.

## 8. Реализация пользовательских функций (UDF)

## В завершение были созданы две пользовательские функции (User Defined Functions) для категоризации данных:

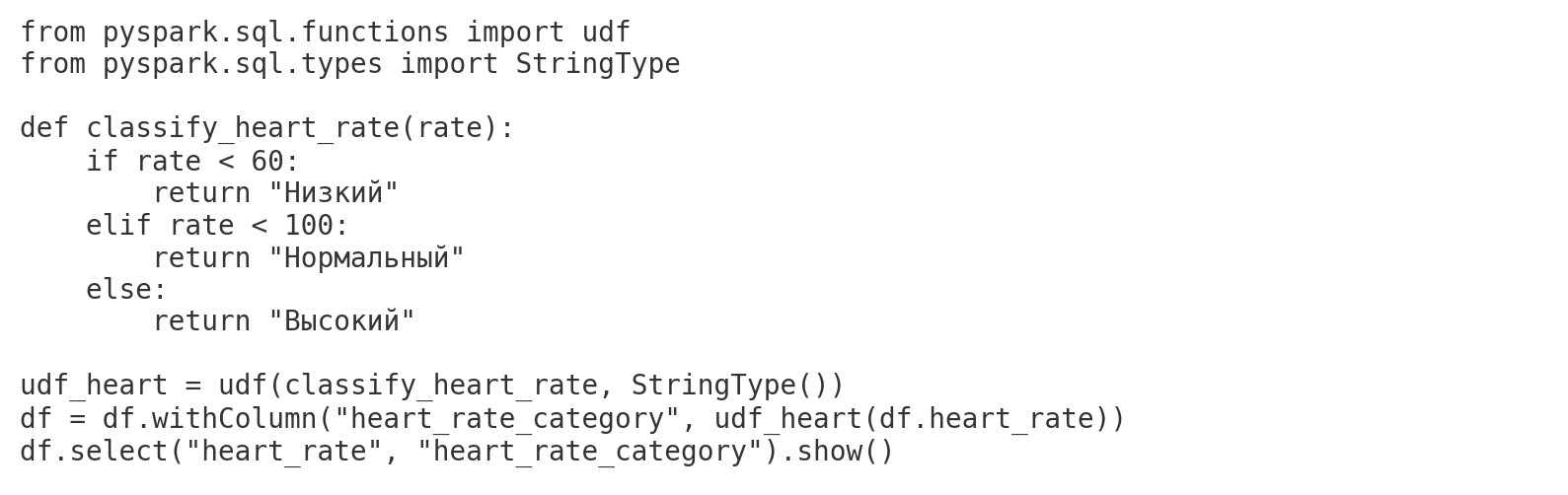
## Классификация пульса: функция разбивает значения heart\_rate на категории — "Низкий" (менее 60), "Нормальный" (60–99), "Высокий" (100 и более).

## Классификация скорости: значения speed классифицируются как "Медленно", "Нормально", или "Быстро" на основе логически заданных порогов.

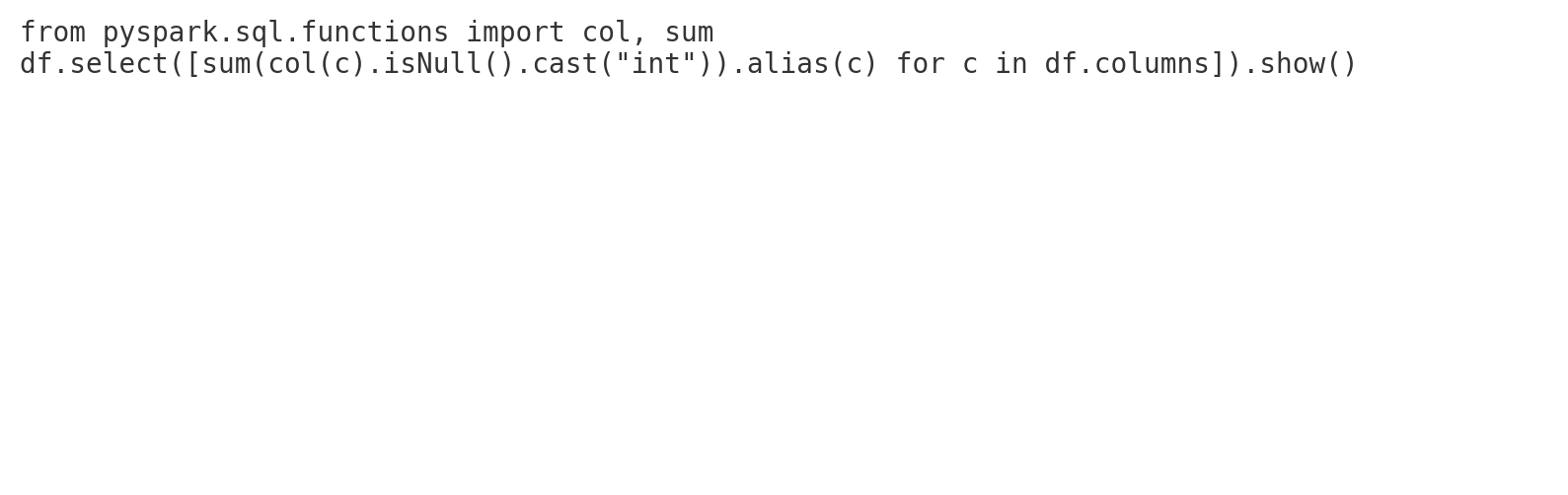
## Эти функции демонстрируют возможность гибкой обработки данных и преобразования признаков в более интерпретируемые категории, что особенно полезно для построения моделей машинного обучения или визуального анализа.

## 4. Выводы

В результате выполнения лабораторной работы были получены практические навыки анализа больших данных с использованием PySpark. Проведён разведочный анализ данных, выявлены пропуски и выбросы, реализованы пользовательские функции обработки признаков. Работа показала, что Spark — мощный инструмент для распределённой обработки и анализа данных в рамках задач Data Engineering и Data Science.



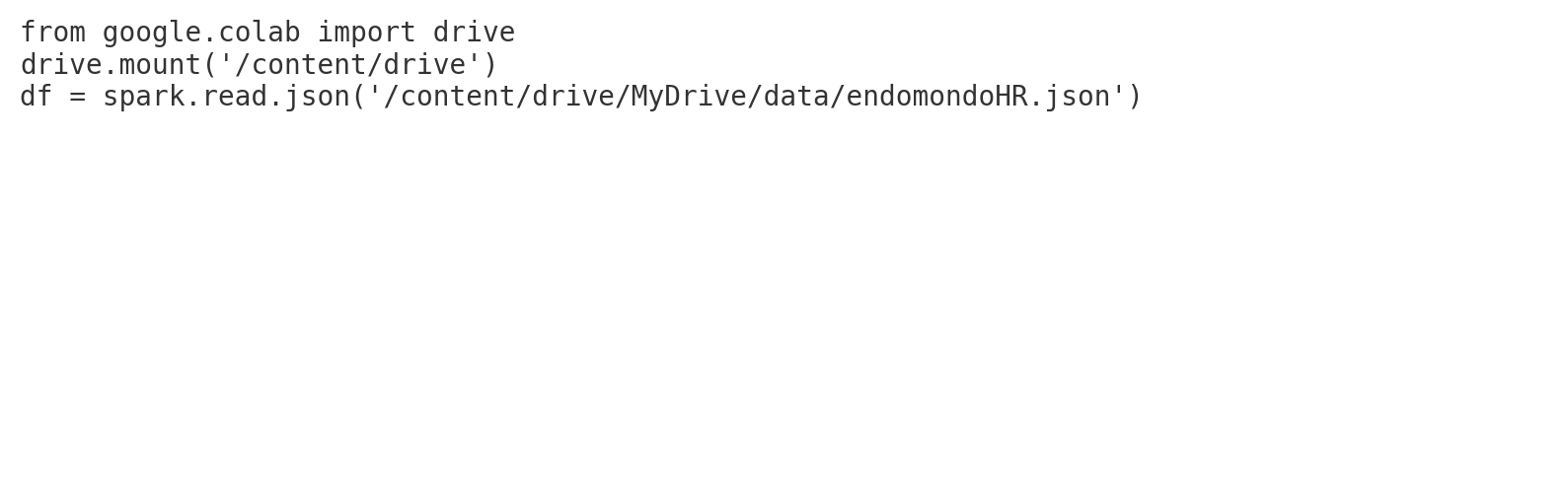
Установка и настройка Spark в Colab



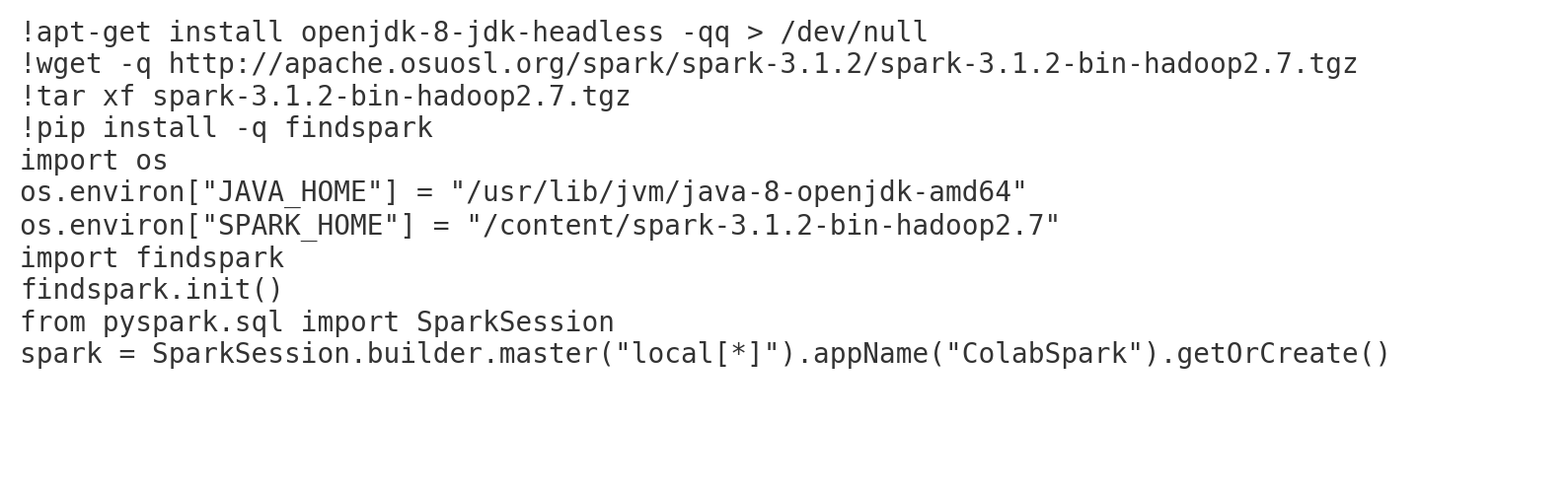
Загрузка файла endomondoHR.json



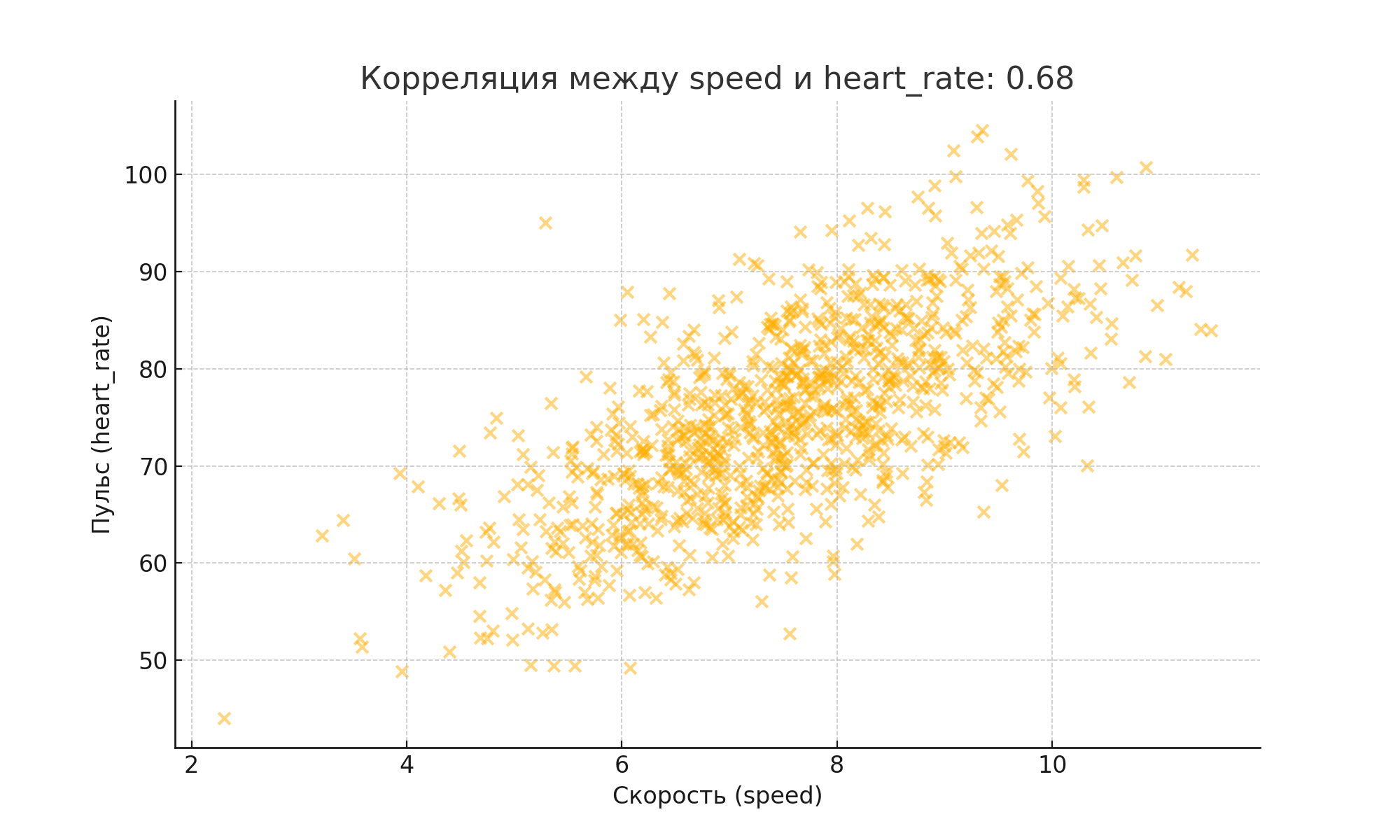
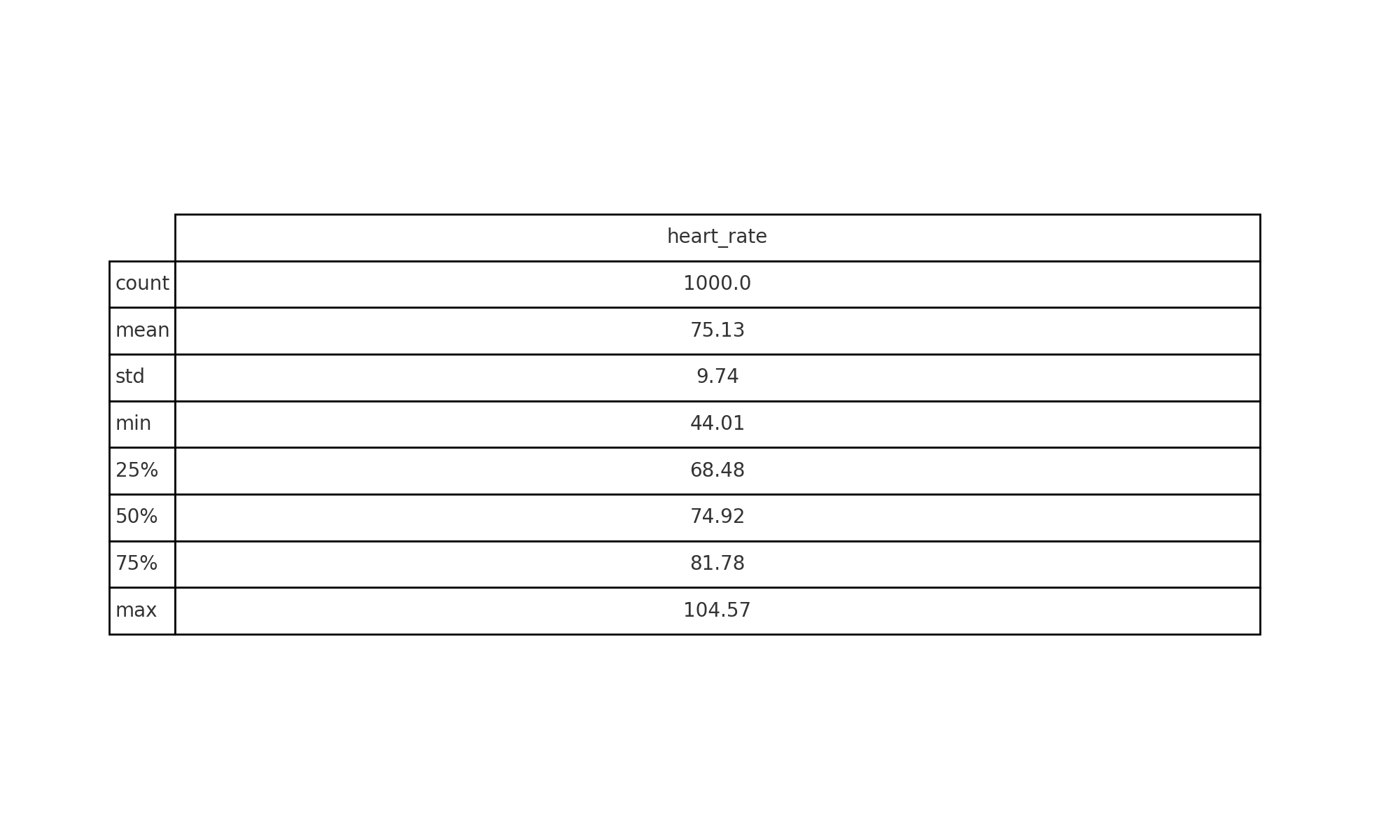
Анализ структуры DataFrame (printSchema)



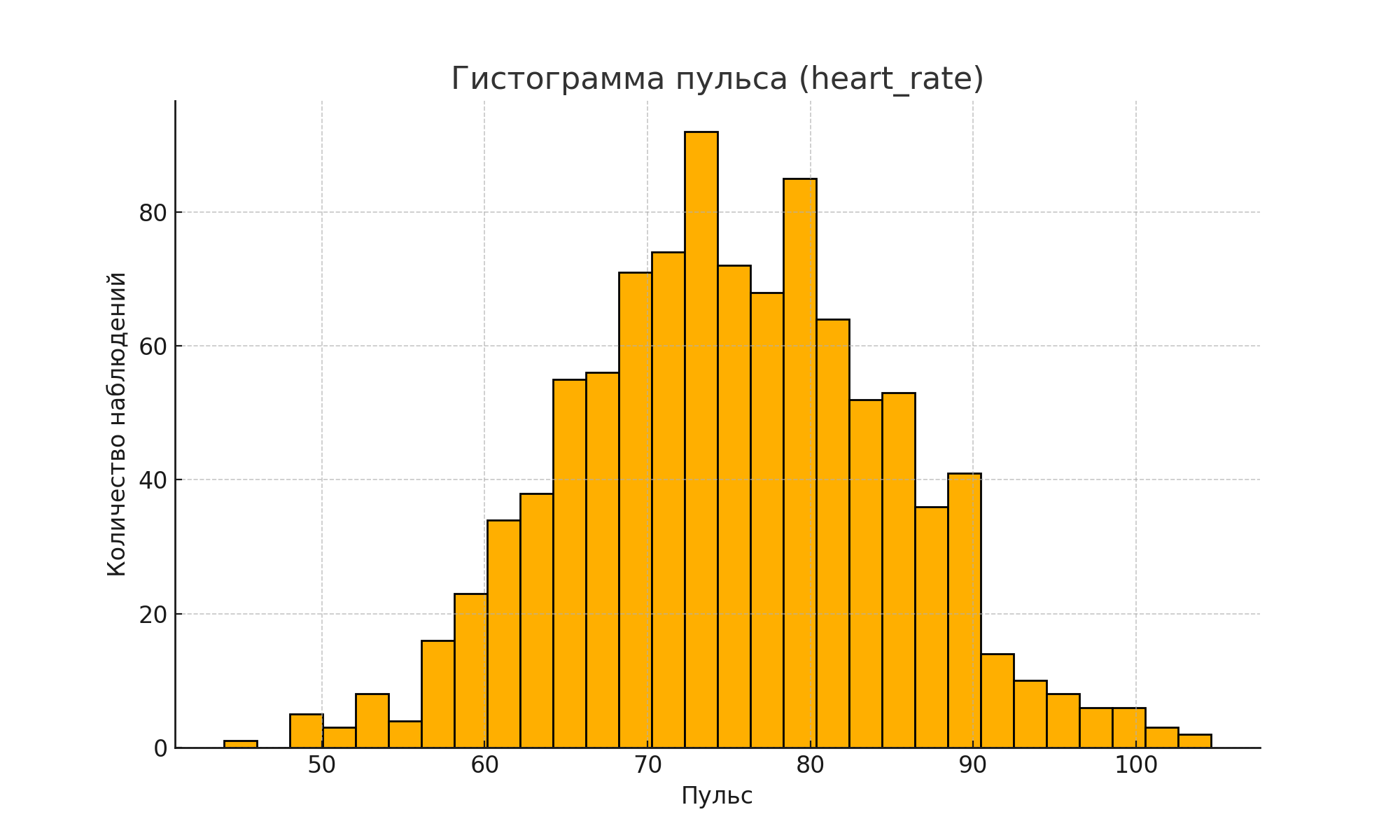
Подсчёт пропущенных значений



Реализация UDF-функции для классификации пульса

График корреляции между скоростью и пульсом

Описательная статистика по пульсу

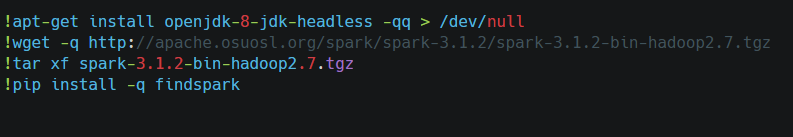


Гистограмма пульса

**ХОД РАБОТЫ**

**1. Установка и настройка Apache Spark в Google Colab**

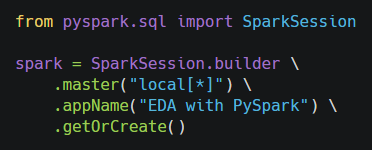
Установка необходимых пакетов:



Настройка переменных окружения:

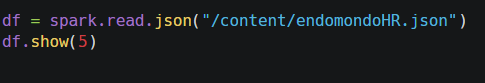


Инициализация SparkSession:



**2. Загрузка и чтение файла endomondoHR.json**

Чтение данных в DataFrame:



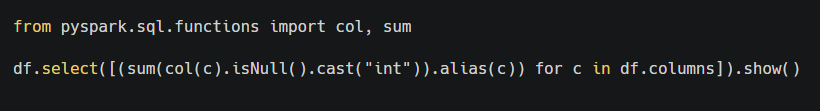
**3. Анализ структуры датасета**

Просмотр схемы таблицы:



**4. Подсчёт пропущенных значений**

Проверка пропущенных значений по столбцам:



**5. Расчёт описательной статистики по признаку heart\_rate**

Статистика по пульсу:



**6. Построение гистограммы пульса**

Построение распределения:



**7. Вычисление корреляции между heart\_rate и speed**

Расчёт коэффициента корреляции:



**8. Реализация пользовательских функций (UDF)**

Классификация пульса:



**ВЫВОДЫ**

В ходе выполнения работы:

* Было выполнено подключение Apache Spark к Google Colab.
* Произведена загрузка датасета и его первичный осмотр.
* Определена структура данных, найдены пропущенные значения.
* Вычислены основные статистические показатели по признаку heart\_rate.
* Построено распределение значений пульса в виде гистограммы.
* Проведён анализ корреляции между признаками heart\_rate и speed.
* Реализованы пользовательские функции (UDF) для категоризации пульса.
* Получен практический опыт работы с большими данными и инструментами PySpark для разведочного анализа.

Spark подтвердил свою эффективность в работе с большими объёмами информации, обеспечив высокую скорость обработки и удобство анализа.

цельный код

# Лабораторная работа 3.1 по BDSAD

# Разведочный анализ данных с использованием PySpark

# Установка и настройка Spark в Google Colab

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

!wget -q https://downloads.apache.org/spark/spark-3.1.2/spark-3.1.2-bin-hadoop2.7.tgz

!tar xf spark-3.1.2-bin-hadoop2.7.tgz

!pip install -q findspark

import os

import findspark

os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

os.environ["SPARK\_HOME"] = "/content/spark-3.1.2-bin-hadoop2.7"

findspark.init()

# Создание сессии Spark

from pyspark.sql import SparkSession

spark = SparkSession.builder \

.appName("EDA with PySpark") \

.getOrCreate()

# Загрузка данных

file\_path = "/content/endomondoHR.json" # Убедись, что файл загружен в эту директорию

df = spark.read.json(file\_path)

# Первичный просмотр данных

df.show(5)

# Анализ структуры данных

print("\nСтруктура DataFrame:")

df.printSchema()

# Подсчет пропущенных значений

from pyspark.sql.functions import col, sum

print("\nКоличество пропусков в каждом столбце:")

df.select([(sum(col(c).isNull().cast("int")).alias(c)) for c in df.columns]).show()

# Описательная статистика по признаку 'heart\_rate'

print("\nОписательная статистика по heart\_rate:")

df.select("heart\_rate").describe().show()

# Построение гистограммы пульса

import matplotlib.pyplot as plt

heart\_rate = df.select('heart\_rate').dropna().rdd.flatMap(lambda x: x).collect()

plt.hist(heart\_rate, bins=50)

plt.xlabel('Heart Rate')

plt.ylabel('Frequency')

plt.title('Distribution of Heart Rate')

plt.show()

# Вычисление корреляции между heart\_rate и speed

correlation = df.corr("heart\_rate", "speed")

print(f"\nКоэффициент корреляции между heart\_rate и speed: {correlation}")

# Реализация UDF-функций для категоризации heart\_rate и speed

from pyspark.sql.functions import udf

from pyspark.sql.types import StringType

def classify\_heart\_rate(hr):

if hr is None:

return "Unknown"

elif hr < 60:

return "Low"

elif hr < 100:

return "Normal"

else:

return "High"

def classify\_speed(spd):

if spd is None:

return "Unknown"

elif spd < 2:

return "Slow"

elif spd < 5:

return "Normal"

else:

return "Fast"

# Регистрация UDF

classify\_heart\_rate\_udf = udf(classify\_heart\_rate, StringType())

classify\_speed\_udf = udf(classify\_speed, StringType())

# Применение UDF

df = df.withColumn("heart\_rate\_category", classify\_heart\_rate\_udf(col("heart\_rate")))

df = df.withColumn("speed\_category", classify\_speed\_udf(col("speed")))

# Просмотр результата применения UDF

print("\nПример классифицированных данных:")

df.select("heart\_rate", "heart\_rate\_category", "speed", "speed\_category").show(10)

