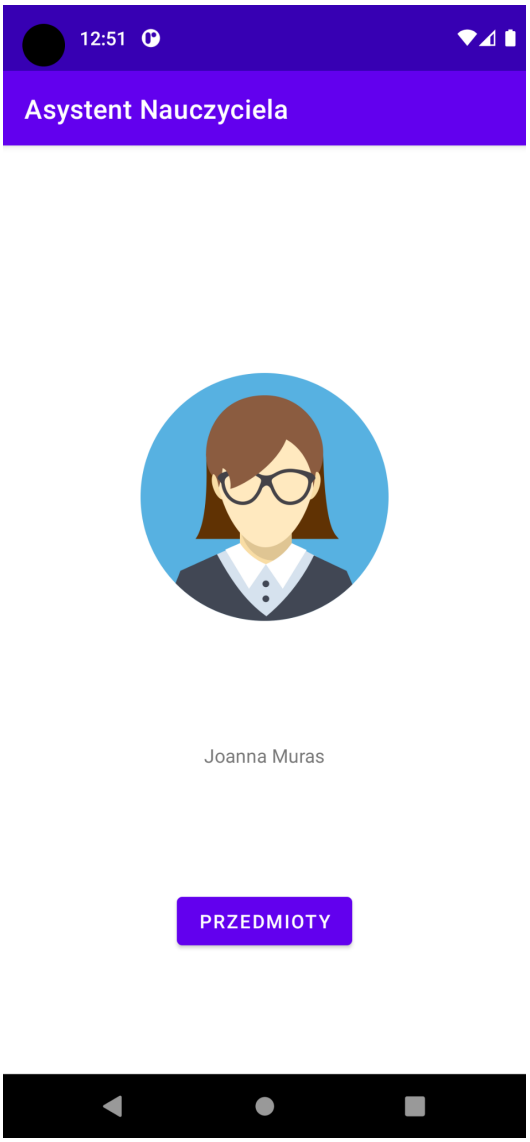
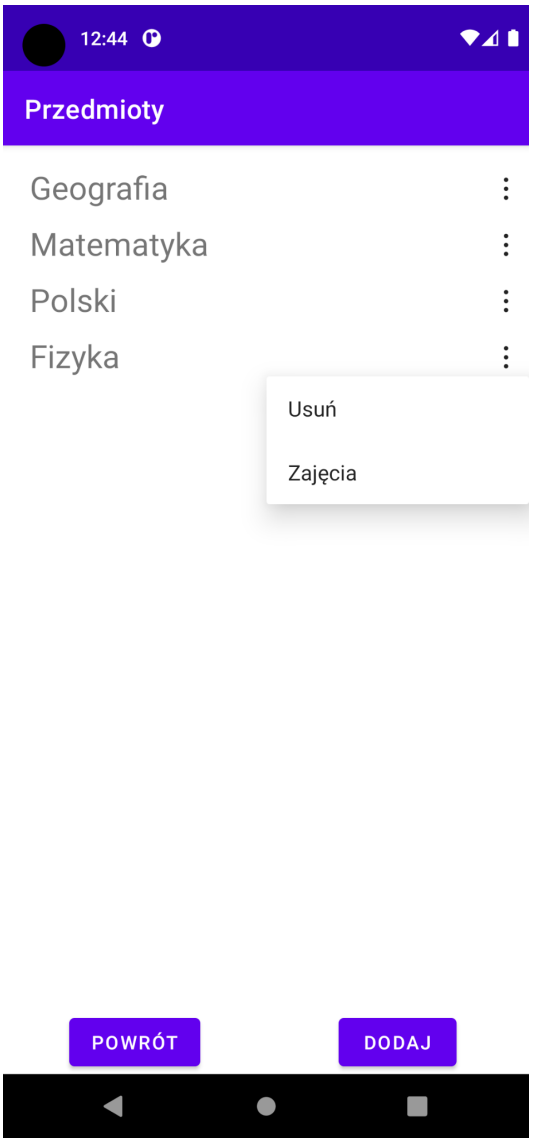
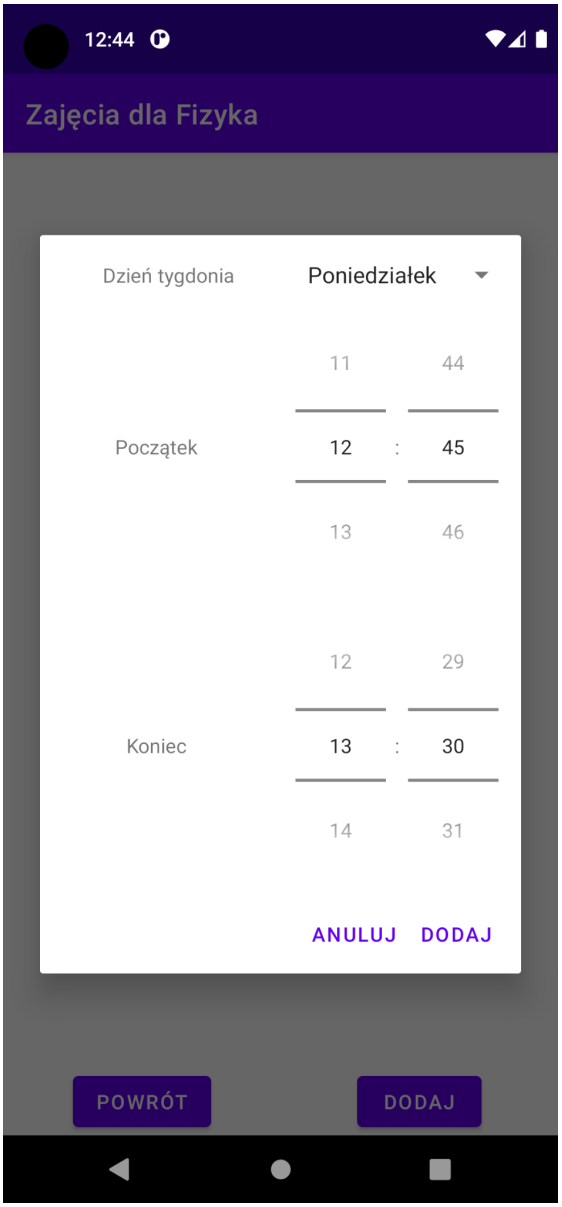
Asystent Nauczyciela - Dokumentacja

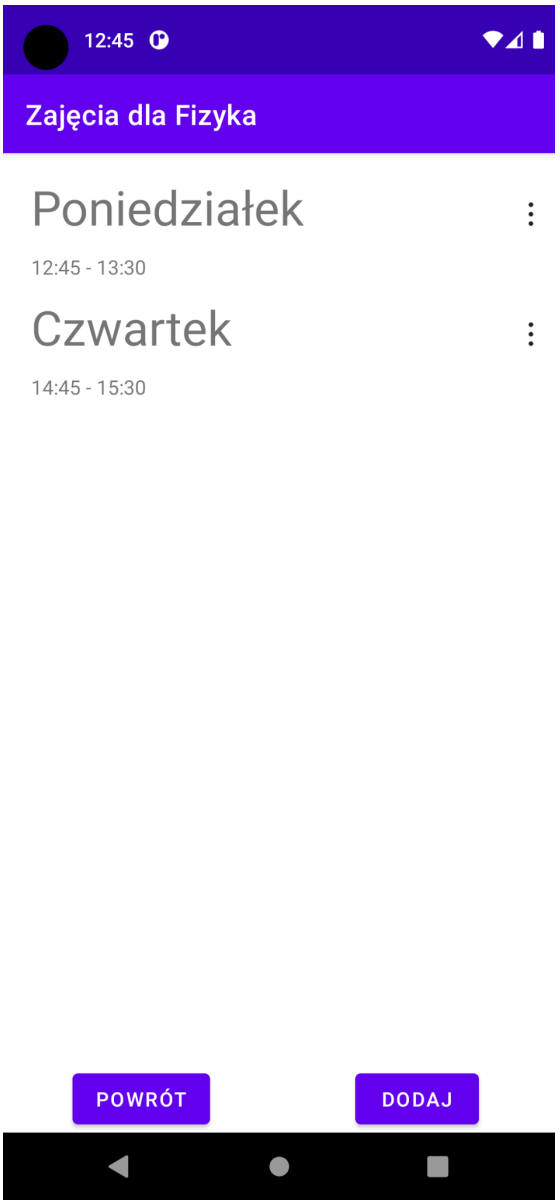
Julian Kominiak

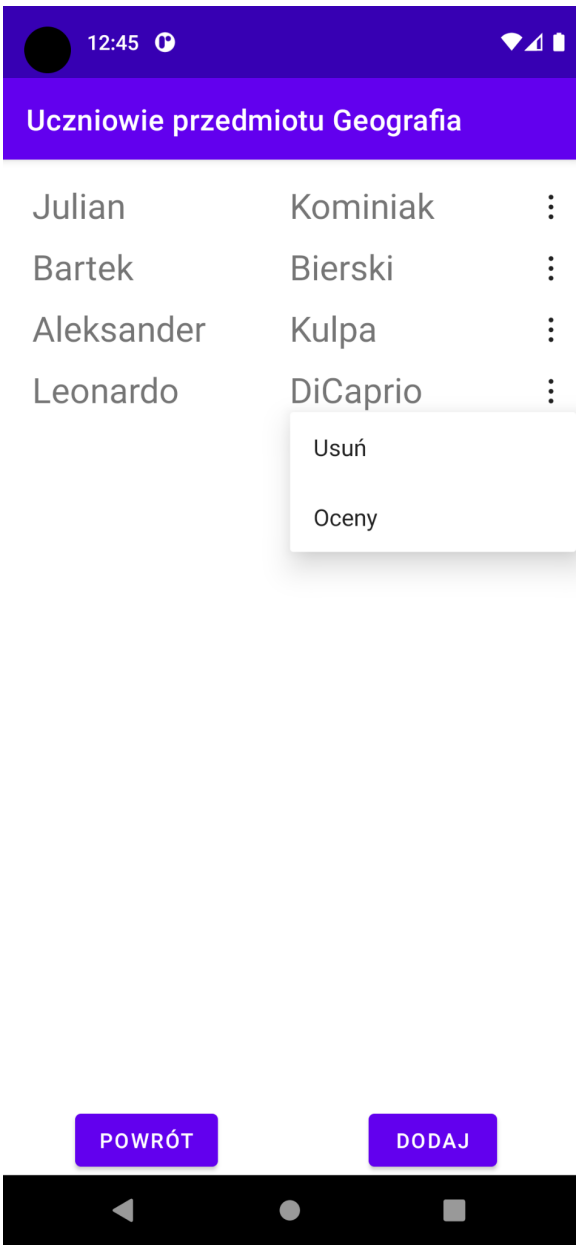
1. Projekt interfejsu graficznego

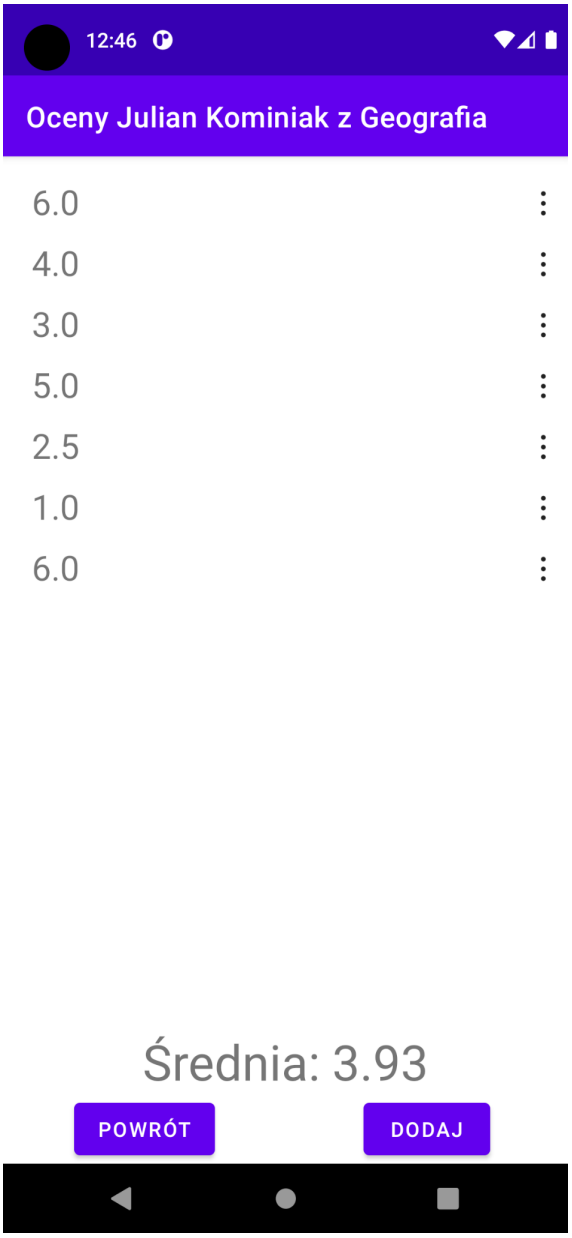


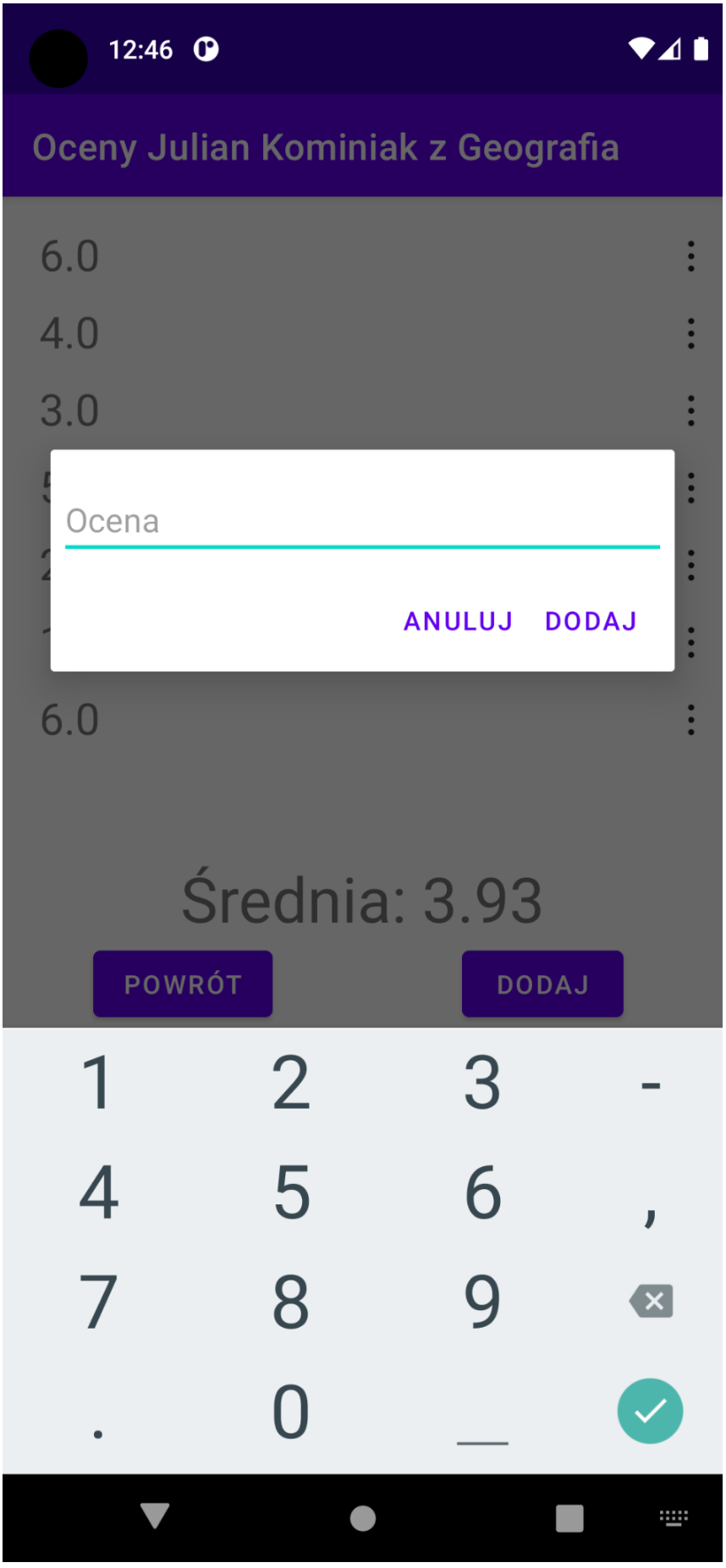



12:44
Zajęcia dla Fizyka
Dzień tygdonia
Poniedziałek
11
44
Początek
12
:
45
13
46
12
29
Koniec
13
:
30
14
31
ANULUJ
DODAJ
POWRÓT
DODAJ


12:45
Zajęcia dla Fizyka
Poniedziałek
12:45 - 13:30
Czwartek
14:45 - 15:30
POWRÓT
DODAJ


12:45
Uczniowie przedmiotu Geografia
Julian Kominiak
Bartek Bierski
Aleksander Kulpa
Leonardo DiCaprio
Usuń
Oceny
POWRÓT
DODAJ


12:46
Oceny Julian Kominiak z Geografia
6.0
4.0
3.0
5.0
2.5
1.0
6.0
Średnia: 3.93
POWRÓT
DODAJ


12:46
Oceny Julian Kominiak z Geografia
6.0
4.0
3.0
Ocena
ANULUJ
DODAJ
6.0
Średnia: 3.93
POWRÓT
DODAJ
1
2
3
-
4
5
6
,
7
8
9
.
0

2. Tworzenie bazy danych

Do stworzenia bazy danych wykorzystałem lokalną bazę danych room, która zainicjalizowana jest w następujący sposób:

```
@Database(entities = [Subject::class, Student::class, Class::class, Mark::class], version = 13,
    exportSchema = false)
abstract class TeacherAssistantDatabase : RoomDatabase() {

    abstract val dao: TeacherAssistantDAO

    companion object {

        @Volatile
        private var INSTANCE: TeacherAssistantDatabase? = null

        fun getInstance(context: Context): TeacherAssistantDatabase {
            synchronized(this) {
                var instance = INSTANCE

                if (instance == null) {
                    instance = Room.databaseBuilder(
                        context.applicationContext,
                        TeacherAssistantDatabase::class.java,
                        "teacher_assistant_database")
                        .fallbackToDestructiveMigration()
                        .allowMainThreadQueries()
                        .build()

                    INSTANCE = instance
                }
                return instance
            }
        }
    }
}
```

Następnie utworzyłem potrzebnę encje wraz z ich atrybutami:

```
@Entity(tableName = "subjects_table")
data class Subject(
    @PrimaryKey(autoGenerate = false)
    @ColumnInfo(name = "subject_name") var subjectName: String,
)
```

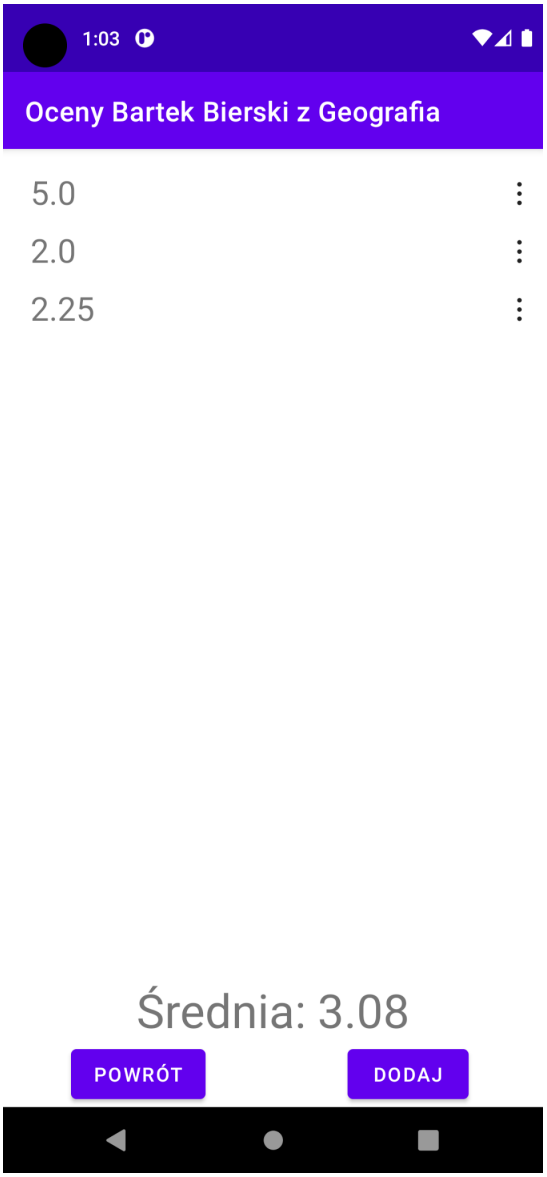
```
@Entity(tableName = "marks_table")
data class Mark(
    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "id") var id: Long,
    @ColumnInfo(name = "value") var value: Float,
    @ColumnInfo(name = "parent_student_id") var parentStudentId: Long
)
```

```
@Entity(tableName = "students_table")
data class Student(
    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "id") var id: Long,
    @ColumnInfo(name = "parent_subject_name") var parentSubjectName: String,
    @ColumnInfo(name = "name") var name: String,
    @ColumnInfo(name = "surname") var surname: String
)
```

```
@Entity(tableName = "classes_table")
data class Class(
    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "id") var id: Long,
    @ColumnInfo(name = "parent_subject_name") var parentSubjectName: String,
    @ColumnInfo(name = "day_of_week") var dayOfWeek: String,
    @ColumnInfo(name = "start_time") var startTime: String,
    @ColumnInfo(name = "end_time") var endTime: String
)
```

Operacja na bazie wykonywane są za pomocą interfejsu DAO.

3. Dodatkowa funkcjonalność - obliczanie średniej



Dodatkową funkcjonalnością jest automatyczne obliczanie średniej dla danej osoby z wybranego przedmiotu. Kod odpowiedzialny za tą funkcjonalność wygląda następująco:

View:

```
<TextView
    android:id="@+id/average_mark"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:textAppearance="@style/TextAppearance.AppCompat.Display1"
    app:layout_constraintBottom_toTopOf="@+id/button_marks_to_students"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
```

```
viewModel.marks.observe(viewLifecycleOwner) {
    marksListAdapter.notifyDataSetChanged()
    view.findViewById<TextView>(R.id.average_mark).text = buildString {
        append("Średnia: ")
        append(viewModel.getAverage())
    }
}
```

ViewModel:

```
fun getAverage(): String {
    val count = dao.countAllMarks(arguments?.get("studentId") as Long)
    val sum = dao.sumAllMarks(arguments.get("studentId") as Long)
    return "%.2f".format(sum / count)
}
```

Model:

```
@Query("SELECT COUNT(*) FROM marks_table " +
        "WHERE parent_student_id = :parentStudent")
fun countAllMarks(parentStudent: Long): Int

@Query("SELECT SUM(value) FROM marks_table " +
        "WHERE parent_student_id = :parentStudent")
fun sumAllMarks(parentStudent: Long): Float
```