Andy Ni

Assignment 6

COMP 3710

Database Design

public static final String *DATABASE\_NAME* = "Trans.db";  
public static final String *TABLE\_NAME* = "transaction\_table";  
public static final String *COL\_1* = "ID";  
public static final String *COL\_2* = "DATE";  
public static final String *COL\_3* = "AMOUNT";  
public static final String *COL\_4* = "CATEGORY";  
  
public DataBaseHelper(Context context) {  
 super(context, *DATABASE\_NAME*, null, 1);  
}  
  
@Override  
public void onCreate(SQLiteDatabase db) {  
 String CREATE\_TABLE = "CREATE TABLE " + *TABLE\_NAME* + " (" + *COL\_1* + " INTEGER PRIMARY KEY AUTOINCREMENT, "  
 + *COL\_2* + " TEXT, "  
 + *COL\_3* + " REAL, "  
 + *COL\_4* + " TEXT)";  
  
 db.execSQL(CREATE\_TABLE);  
  
  
  
}

CREATE TABLE transaction\_table (

"ID" INTEGER NOT NULL PRIMARY KEY,

"DATE" TEXT NOT NULL,

"AMOUNT" TEXT NOT NULL,

"CATEGORY" REAL NOT NULL

);

**Used this function to insert into table**

public boolean insertData(String date,double amount,String category) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues contentValues = new ContentValues();  
 contentValues.put(*COL\_2*,date);  
 contentValues.put(*COL\_3*,amount);  
 contentValues.put(*COL\_4*,category);  
 long result = db.insert(*TABLE\_NAME*,null ,contentValues);  
 if(result == -1)  
 return false;  
 else  
 return true;  
}

**Used bottom two functions to load table at start up**

public Cursor getAllData() {  
 SQLiteDatabase db = this.getWritableDatabase();  
 Cursor res = db.rawQuery("select \* from "+*TABLE\_NAME*,null);  
 return res;  
}

public void loadTable() {  
 Cursor res = myDB.getAllData();  
 if (res.getCount() == 0) {  
 Log.*i*("Database Empty", "Database has no records");  
 return;  
 }  
  
 while (res.moveToNext()) {  
 View myView = layoutInflater.inflate(R.layout.*table\_row*, null, false);  
  
 TextView textDate = myView.findViewById(R.id.*txtTableDate*);  
 TextView textAmount = myView.findViewById(R.id.*txtTableAmount*);  
 TextView textCategory = myView.findViewById(R.id.*txtTableCategory*);  
  
 double cost = res.getDouble(2);  
 balance += cost;  
 String costS = *df2*.format(cost);  
  
 textDate.setText(res.getString(1));  
 textAmount.setText(costS);  
 textCategory.setText(res.getString(3));  
 transTable.addView(myView);  
  
 }  
  
 tBalance.setText(*df2*.format(balance));  
  
}

**For Searching Specific Transactions Used These 5 Methods which handle database querires.**

public Cursor getAllData() {  
 SQLiteDatabase db = this.getWritableDatabase();  
 Cursor res = db.rawQuery("select \* from "+*TABLE\_NAME*,null);  
 return res;  
}  
  
public Cursor getSelectedData(String startDate, String endDate, Double minPrice, Double maxPrice) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 Cursor res = db.rawQuery("select \* from "+*TABLE\_NAME* + " WHERE " + *COL\_2* +  
 " BETWEEN '" + startDate + "' AND '" + endDate + "'" + " AND " +  
 *COL\_3* + " >= " + minPrice + " AND " + *COL\_3* + " <= " + maxPrice,null);  
 return res;  
}  
  
public Cursor getSelectedData(String startDate, String endDate, Double minPrice, Double maxPrice, Double negMin, Double negMax) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 Cursor res = db.rawQuery("select \* from "+*TABLE\_NAME* + " WHERE " + *COL\_2* +  
 " BETWEEN '" + startDate + "' AND '" + endDate + "'" + " AND ((" +  
 *COL\_3* + " >= " + minPrice + " AND " + *COL\_3* + " <= " + maxPrice + ") OR (" + *COL\_3* + " >= " + negMax + " AND " + *COL\_3* + " <=" + negMin + "))",null);  
 return res;  
}  
  
public Cursor getSelectedData(Double minPrice, Double maxPrice) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 Cursor res = db.rawQuery("select \* from "+*TABLE\_NAME* + " WHERE " +  
 *COL\_3* + " >= " + minPrice + " AND " + *COL\_3* + " <= " + maxPrice,null);  
 return res;  
}  
  
public Cursor getSelectedData(Double minPrice, Double maxPrice, Double negMin, Double negMax) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 Cursor res = db.rawQuery("select \* from "+*TABLE\_NAME* + " WHERE " + *COL\_3* + " >= " + minPrice + " AND " +  
 *COL\_3* + " <= " + maxPrice + " OR " + *COL\_3* + " >= " + negMax + " AND " + *COL\_3* + " <=" + negMin,null);  
 return res;  
}

**Searching for only income with a set of date ranges. I used**

SELECT \* FROM transaction\_table WHERE DATE BETWEEN ‘startDate’ AND ‘endDate’ AND AMOUNT >= minPrice AND AMOUNT <= maxPrice

**FOR expenses only searches I used the above function and I call it with variables negMax and negMin. I use negMax for the above minPrice and negMin for the above maxPrice.**

**Searching for income and expenses with a set of date ranges. I used**

SELECT \* FROM transaction\_table WHERE DATE BETWEEN ‘startDate’ AND ‘endDate’ AND ((AMOUNT >= minPrice AND AMOUNT <= maxPrice) OR (AMOUNT >= negMax AND AMOUNT <= negMin))

**Searching for all income transactions**

SELECT \* FROM transaction\_table WHERE AMOUNT >= minPrice AND AMOUNT <= maxPrice

**Search for all expense transactions. Use the same function but called it with the negMin and negMax variables and use negMax for the minPrice in the sql code and negMin for the maxPrice in the sql code**

**Searching for all income and expense transactions**

SELECT \* FROM transaction\_table WHERE ((AMOUNT >= minPrice AND AMOUNT <= maxPrice) OR (AMOUNT >= negMax AND AMOUNT <= negMin))