פרק 22 - SQLite

SQLite היא שפה לשמירת מידע בעזרת טבלאות של מסד נתונים.
ניתן לקרוא מהטבלה, להוסיף, למחוק ולעדכן את השדות שבטבלה והכל נשמר.

מכון התקנים האמריקאי הגדיר את SQL כשפה סטנדרטית לעבודה מול נתונים בטבלאות.

דוגמא 1

tblPerson(personId,fname,lname,salary,birthday)

personId	fname	lname	salary	birthday
1	oren	uziel	2000	12/10/1987
2	david	azulay	3000	15/6/1990
3	oren	kabuli	1500	15/4/1984
4	shula	afula	5000	10//3/1995

לא יכולות להיות שתי שורות בטבלה בעלות אותו ID

מבנה פקודה

select	מגדיר את העמודות,קבועים אותם נשלף.חובה בפקודה
from	שמות הטבלאות מהן יישלף המידע - חובה בפקודה
where	תנאי השליפה שיקבעו את השורות שישלפו
group by	מאפשר קיבוץ רשומות לקבוצות
Having	מגדיר תנאי לוגי על שורות המקובציות
Order By	מציג את הסדר שבו תוצגנה הרשומות הנשלפות

נוכל לקבוע מאת איזה שדות ניקח, מאיזה טבלאות:

שליפת שורות מטבלה

```
select id,last_name,first_name
from employees
where start_date = '14.5.90'
```

הסדר בו תוצגנה העמודות הוא כסדר הגדרתן בשלב הגדרת הטבלה.

בדוגמה לעיל נשלוף את העמודות id, last_name ו first_name מהטבלה employees

רק מהשורות שבהן ערך העמודה start_date הוא 14.5.90.

select*from

כוכבית מציין - שלוף את כל העמודת

שווה	=
לא שווה	!=, <>, =^
גודל מ..	>
גדול או שווה	=<
קטן מ	<
קטן או שווה	=<

Not Between

```
select last_name,dept_id,start_date
from employee
where start_date NOT BETWEEN ‘1.1.91’
AND ‘31.12.91’
```

בדוגמה לעיל נשלוף מהטבלה employee את העמודות last_name, dept_id ו start_date מהשורות בהן ערך העמודה start_date **לא בין** 1.1.91 ל 31.12.91.

דוגמא

```
select fname,salary
from tblPerson
where salary>4000
```

יחזיר :

fname	salary
shula	5000

נוכל לבטל שורות בהן ערך מסוים מופיע כבר בשורה אחרת:

DISTINCT

תפקידו לבטל שורות כפולות

```
select distinct fname
from person
```

output

fname
oren
shula
david

Like:

דוגמא ל - like

```
select personId , fname
from tblPerson
where fname like 'a%'
```

הצגת כל האנשים ששמם מתחיל באות a

```
select personId , fname
from tblPerson
where fname like '%a'
```

הצגת כל האנשים ששמם מסתיים באות a

Null:

דוגמא ל - null

ערך NULL בשדה משמעו כי הערך לא ידוע, חסר.

הצגת כל האנשים שחסר להם שם משפחה בטבלה

```
select personId , fname
from tblPerson
where fname  is null
```

הצגת כל האנשים שלא חסר להם שם משפחה בטבלה

```
select personId , fname
from tblPerson
where fname  is  not null
```

תנאי מורכב:

דוגמא לתנאי מורכב

הצגת כל האנשים שקוראים להם עוזי ומשכורתם גבוה מ - 2000

```
select personId , fname
from tblPerson
where salary > 2000 AND fname = 'uzi'
```

הצגת כל האנשים שקוראים להם עוזי או שמשכורתם גבוה מ - 2000

```
select personId , fname
from tblPerson
where salary>2000  OR fname = 'uzi'
```

הצגת שורות בסדר מסוים:

הצגת שורות בסדר מסוים

יציג את כל העובדים שמשכורתם גבוה מ - 2000 בסדר יורד של המשכורת"

```
select lname  , fname
from tblPerson
where salaty>2000
ORDER BY salary desc
```

נוכל גם לעדכן ערכים בטבלה:

```
UPDATE tblPerson
SET fname='shoosha' lname='gusha'
WHERE fname='uzi';
```

יגש לכל השורות בהן השם הוא uzi וישנה את שם הפרטי ל shoosha ואת שם המשפחה ל gusha -.

הכנסת נתונים לטבלה:

INSERT

באמצעות פקודת INSERT נכניס נתונים לטבלה

```
INSERT INTO
tblPerson('uzi','chafuzi',10000,'12,7,1976')
```

יכניס שורה נוספת לטבלה.שימו לב personId יועלה באופן אוטומטי.

ברגע שניצור את הטבלה נגדיר את ה ID כאוטומטי, כך לא נצטרך לדאוג לו.

מחיקת שורות מהטבלה:

```
DELETE FROM tblPerson
WHER salary>2000
```

ימחק את כל השורות בהם המשכורת גבוה מ - 2000.

בדוגמאות שנראה בפרק נשתמש רק בפקודות הללו, אך יש עוד פקודות רבות שבהן לא ניגע בקורס.

דוגמה 1:

נבנה טבלה של אנשים שניתן להוסיף לה מוצרים, להציג את המוצרים, לעדכן נתונים למוצרים ולמחוק.

שלב 1 - יצירת class Product

ניצור קלאס חדש בשם Product:

שלב 2 - נערוך את ה class Product:

```
public class Product {
    private String name;
    private String description;
    private int price;
    private long productId;

    public Product(long productId,String name, String description, int price) {
        this.name = name;
        this.description = description;
        this.price = price;
        this.productId = productId;
    }
    public Product(String name, String description, int price) {
        this.name = name;
        this.description = description;
        this.price = price;
        this.productId = 0;
    }
```

נצהיר על תכונות שנרצה שיהיו בכל מוצר שלנו

לכל מוצר יהיה id אופייני לו. ה id של המוצר הראשון יהיה 0, לשני 1 וכן הלאה

בנאי שמקבל id

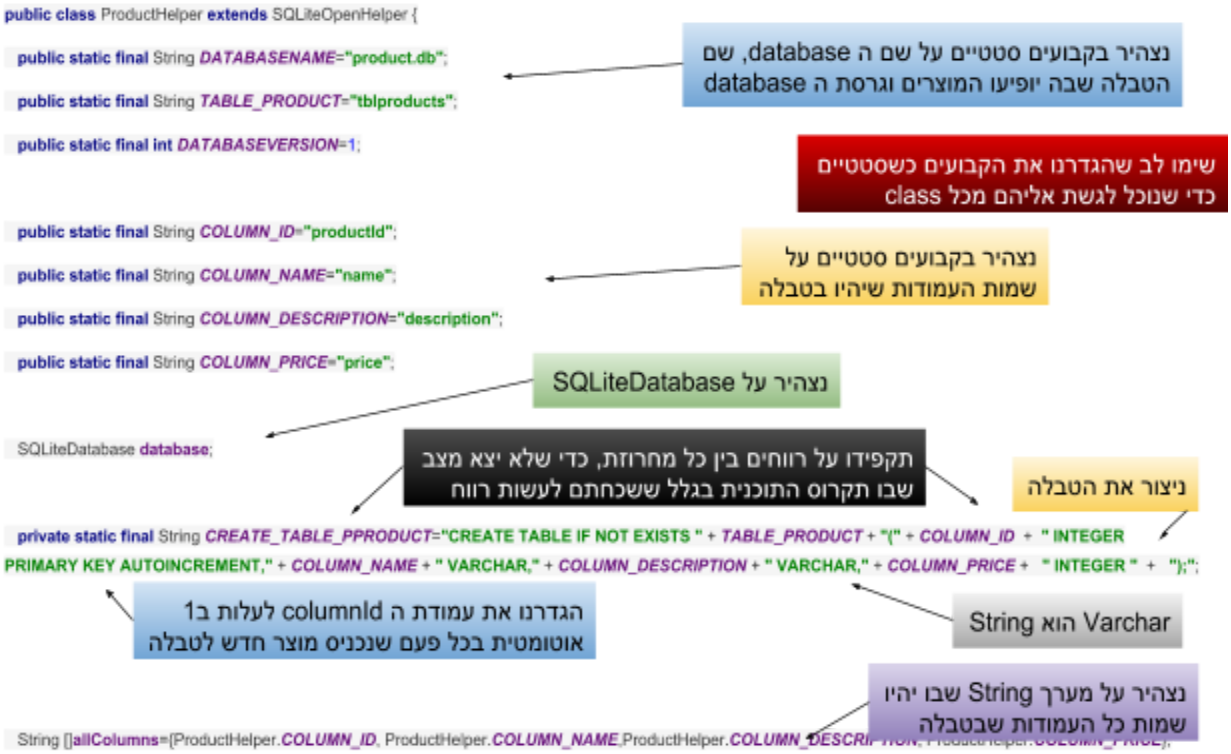
בנאי שלא מקבל את ה id (SQL ייתן לנו אוטומטית את ה id המתאים

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}
public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public int getPrice() {
    return price;
}

public void setPrice(int price) {
    this.price = price;
}
public long getProductId() {
    return productId;
}
public void setProductId(long productId) {
    this.productId = productId;
}

 @Override
 public String toString() {
     return "Product{" +
             "name='" + name + '\'' +
             ", description='" + description + '\'' +
             ", price=" + price +
             ", productId=" + productId +
             '}';
 }
}
```

Setters ו Getters

פונקציית toString

שלב 3 - יצירת class ProductHelper ועריכתו:




```
public class ProductHelper extends SQLiteOpenHelper {

    public static final String DATABASENAME="product.db";

    public static final String TABLE_PRODUCT="tblproducts";

    public static final int DATABASEVERSION=1;


    public static final String COLUMN_ID="productId";

    public static final String COLUMN_NAME="name";

    public static final String COLUMN_DESCRIPTION="description";

    public static final String COLUMN_PRICE="price";


    SQLiteDatabase database;

    private static final String CREATE_TABLE_PPRODUCT="CREATE TABLE IF NOT EXISTS " + TABLE_PRODUCT + "(" + COLUMN_ID + " INTEGER
PRIMARY KEY AUTOINCREMENT," + COLUMN_NAME + " VARCHAR," + COLUMN_DESCRIPTION + " VARCHAR," + COLUMN_PRICE + " INTEGER " + ");";

    String []allColumns={ProductHelper.COLUMN_ID, ProductHelper.COLUMN_NAME,ProductHelper.COLUMN_DESCRIPTION, ProductHelper.COLUMN_PRICE};
```

```
public ProductHelper(Context context) {
    super(context, DATABASENAME, null, DATABASEVERSION);
    // TODO Auto-generated constructor stub
}

@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_TABLE_PPRODUCT);
    Log.i("data1", "Table customer created");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_PRODUCT);
    onCreate(db);
}

public void open(){
    database=this.getWritableDatabase();
    Log.i("data", "Database connection open");
}
```

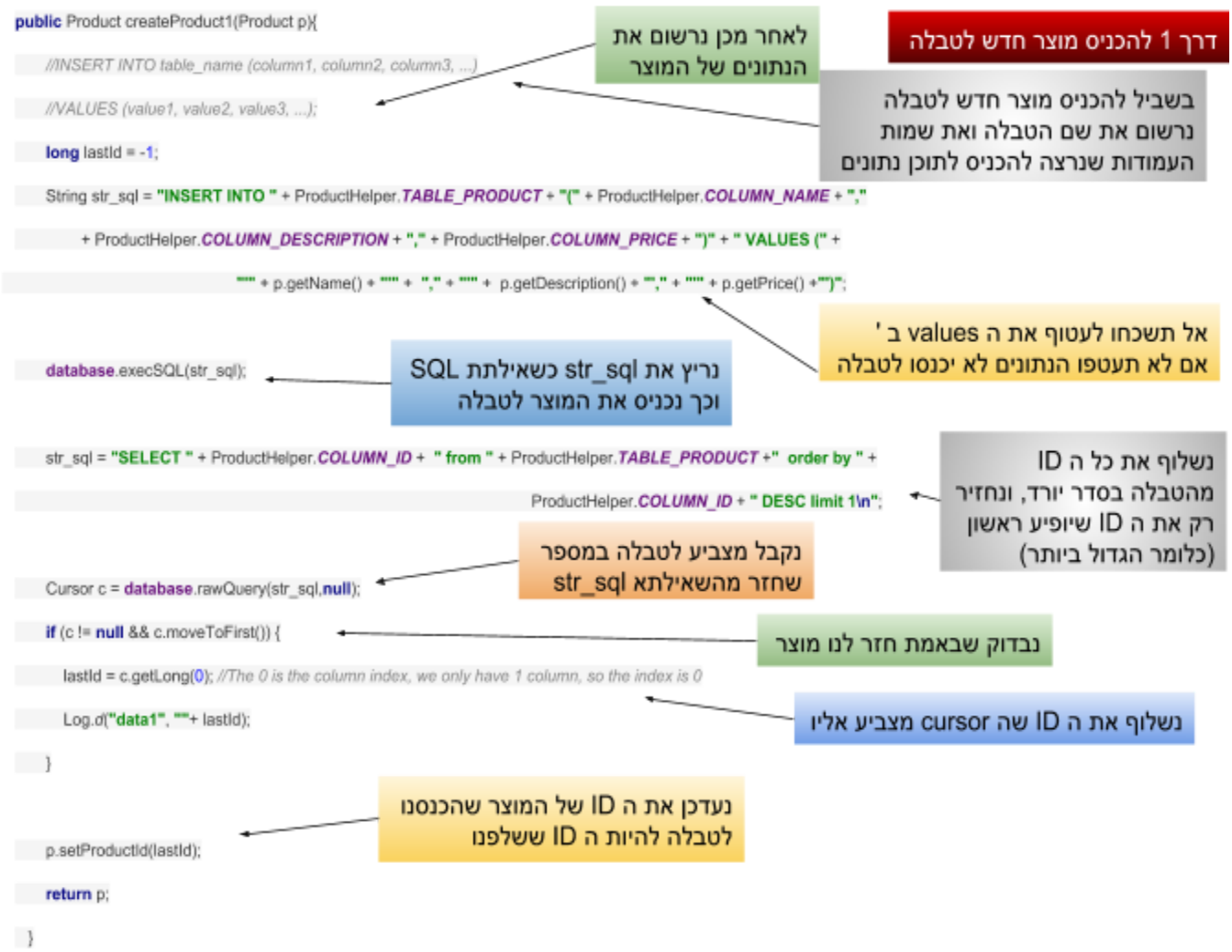
נעביר בבנאי את ה context, שם ה database וגרסת ה database לבנאי של המחלקה ממנה אנחנו יורשים

פונקציות סטטיות שחובה לממש כחלק מהמחלקה שאנחנו יורשים

ב onCreate נממש את הפונקציה שמריצה את המשפט CREATE_TABLE_PRODUCT שהגדרנו לפני כן

בפונקציה onUpgrade נריץ את השאילתא DROP TABLE IF EXISTS שמוחקת את הטבלה שלנו, ונעביר לה את ה TABLE_PRODUCT העדכני כדי שתיצור אותו מחדש עם הנתונים העדכניים

בפונקציה open נהפוך את ה database לאפשרי לעריכה




```
public Product createProduct1(Product p){
    //INSERT INTO table_name (column1, column2, column3, ...)
    //VALUES (value1, value2, value3, ...);
    long lastId = -1;
    String str_sql = "INSERT INTO " + ProductHelper.TABLE_PRODUCT + "(" + ProductHelper.COLUMN_NAME + ","
            + ProductHelper.COLUMN_DESCRIPTION + "," + ProductHelper.COLUMN_PRICE + ")" + " VALUES (" +
                    """ + p.getName() + """ + "," + """ + p.getDescription() + "","" + """ + p.getPrice() +"")";
    database.execSQL(str_sql);
    str_sql = "SELECT " + ProductHelper.COLUMN_ID + " from " + ProductHelper.TABLE_PRODUCT +" order by " +
                                                    ProductHelper.COLUMN_ID + " DESC limit 1\n";
    Cursor c = database.rawQuery(str_sql,null);
    if (c != null && c.moveToFirst()) {
        lastId = c.getLong(0); //The 0 is the column index, we only have 1 column, so the index is 0
        Log.d("data1", ""+ lastId);
    }
    p.setProductId(lastId);
    return p;
}
```

```
public Product createProduct2(Product p){
    ContentValues values=new ContentValues();
    values.put(ProductHelper.COLUMN_NAME, p.getName());
    values.put(ProductHelper.COLUMN_DESCRIPTION, p.getDescription());
    values.put(ProductHelper.COLUMN_PRICE, p.getPrice());

    long insertId=database.insert(ProductHelper.TABLE_PRODUCT, null, values);
    Log.i("data", "Product " + insertId + "insert to database");
    p.setProductId(insertId);
    return p;
  }
}
```

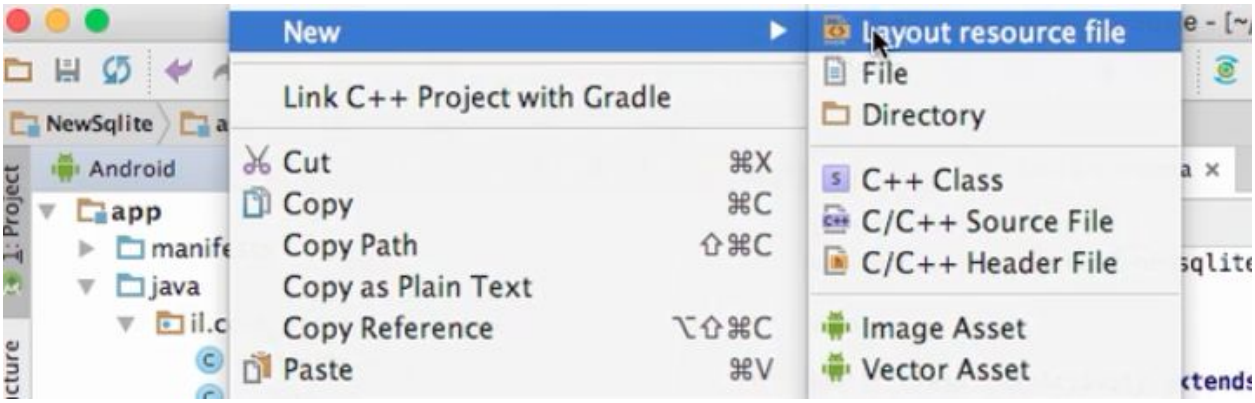
דרך 2 להכניס מוצר חדש לטבלה

בעזרת ContentValues נוכל להכניס נתונים לטבלה בשיטה של key ו value

נכניס את המוצר החדש לטבלה עם ה ContentValues ויוחזר לנו ID

נגדיר את ה ID שחזר אלינו למוצר שהכנסנו לטבלה

שלב 4 - יצירת layout ל custom dialog: (קראנו לו layout_add)



שלב 5 - עיצוב ה custom dialog:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"

  android:orientation="vertical"

  android:layout_width="match_parent"
```

```
  android:layout_height="match_parent">


  <EditText
      android:layout_width="200dp"
      android:layout_height="wrap_content"
      android:hint="product name"
      android:id="@+id/etName"
      android:background="#13ea2f"
      android:textColor="#e90a0a"
      android:textSize="30sp"
      android:layout_margin="10dp"
      />


  <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="product descriptin"
      android:id="@+id/etDescripion"
      android:background="#13ea2f"
```

```
    android:textColor="#e90a0a"
    android:textSize="30sp"
    android:layout_margin="10dp"
    />

  <EditText
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:hint="product price"
    android:id="@+id/etPrice"
    android:background="#13ea2f"
    android:textColor="#e90a0a"
    android:textSize="30sp"
    android:layout_margin="10dp"
    />

  <Button
    android:layout_width="200dp"
    android:layout_height="wrap_content"
```

```
    android:text="save"
    android:id="@+id/btnSave"
    android:background="#13ea2f"
    android:textColor="#e90a0a"
    android:textSize="30sp"
    android:layout_margin="10dp"
    />

  <Button
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:text="Delete"
    android:id="@+id/btnDeleteDialog"
    android:background="#13ea2f"
    android:textColor="#e90a0a"
    android:textSize="30sp"
    android:layout_margin="10dp"
    />
```

```
  <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/tvSol"
      android:text=""
      android:textSize="30sp"
      />


</LinearLayout>
```

שלב 6 - עיצוב ה MainActivity:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Manage product!"
        android:textSize="30sp"
        android:layout_gravity="center"
        />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Add Product"
        android:id="@+id/btnShowAddDialog"
        android:background="#13ea2f"
        android:textColor="#e90a0a"
        android:textSize="30sp"
        android:layout_margin="10dp"
        />


    <Button
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:text="show all"
        android:id="@+id/btnShow"
        android:background="#13ea2f"
        android:textColor="#e90a0a"
        android:textSize="30sp"
```

```
        <Button
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:text="Delete By Id"
            android:id="@+id/btnDelete"
            />


    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_margin="20dp">
        <EditText
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:id="@+id/etUpdateId"
            android:hint="Print ID"
            />
        <Button
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:text="Update By Id"
            android:id="@+id/btnShowUpdate"
            />

    </LinearLayout>


</LinearLayout>
```

```
public class MainActivity extends AppCompatActivity {
    ProductHelper productHelper;
    Dialog addDialog;
    Button btnAdd,btnShowAddDialo;
    EditText etName, etPrice, etDesc;
    TextView tvSol;
    ArrayList<Product> products;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        productHelper = new ProductHelper(this);
        btnShowAddDialog = (Button)findViewById(R.id.btnShowAddDialog);
        btnShowAddDialog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addDialog();
            }
        });
```

הצהרות

ניתן הפניה ל ProductHelper ונעביר לו את ה context

ניתן לכפתור הפניה והאזנה

כשהמשתמש ילחץ על btnShowAddDialog תופעל הפונקציה addDialog, אותה נממש בהמשך

```
btnShow = (Button) findViewById(R.id.btnShow);
btnShow.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        productHelper.open();
        products = productHelper.getAllProducts1();
        if (products.size() > 0) {
            Log.d("data1", products.toString());
        }

        productHelper.close();
    }
});

}//end of create
```

הפניה לכפתור והאזנה

כשהמשתמש ילחץ על btnShow ניתן לטבלה שלנו אפשרות לעריכה

נשלוף את כל המוצרים מהטבלה לתוך רשימה

נדפיס את כל המוצרים

נסגור את האפשרות לעריכת הטבלה

```
public void addDialog(){
    addDialog = new Dialog(this);
    addDialog.setTitle("Add Product");
    addDialog.setContentView(R.layout.layout_add);
    tvSol = (TextView)addDialog.findViewById(R.id.tvSol);
    etName=(EditText)addDialog.findViewById(R.id.etName);
    etPrice=(EditText)addDialog.findViewById(R.id.etPrice);
    etDesc=(EditText)addDialog.findViewById(R.id.etDescripion);
    btnAdd = (Button)addDialog.findViewById(R.id.btnSave);

    btnAdd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String name = etName.getText().toString();
            String desc = etDesc.getText().toString();
            int price  = Integer.valueOf(etPrice.getText().toString());
```

ניתן הפניה לדיאלוג

נגדיר את הכותרת של הדיאלוג

נגדיר את ה layout של הדיאלוג להיות ה layout_addשעיצבנו

ניתן הפניה לכל האובייקטים שבדיאלוג (לא לשכוח addDialog.findViewById)

כשהמשתמש ילחץ על btnAdd נשמור את כל הנתונים שהוא הזין ב EditTextים

```
        Product p= new Product(name,desc,price);
        productHelper.open();
        p = productHelper.createProduct1(p);
        productHelper.close();
        if(p.getProductId() > 0 ) {
            tvSol.setText("product : " + p.getName() + "insert to database " + " productId = " + p.getProductId());
            etName.setText("");
            etPrice.setText("");
            etDesc.setText("");
        }
        else   tvSol.setText("error insert : " + p.getName() + " to database");
      }


    });// end of onClickListener
    addDialog.show();
  }// end of addDialog
}
```

ניצור מוצר עם הנתונים שהמשתמש הזין

ניתן לטבלה אפשרות לעריכה

נוסיף את המוצר לטבלה

נסגור את האפשרות לעריכת הטבלה

נחזיר את הEditTextים להיות ריקים

נציג את הדיאלוג

שלב 5 - מימוש הפונקציה להצגת כל האנשים ב MainActivity:

```
public ArrayList<Product> getAllProducts1() {
  ArrayList<Product> l = new ArrayList<Product>();
  String query = "select * from " + TABLE_PRODUCT;
  Cursor cursor=database.rawQuery(query,null);

  if(cursor.getCount()>0){
    while(cursor.moveToNext()){
      long id=cursor.getLong(cursor.getColumnIndex(ProductHelper.COLUMN_ID));
      String name=cursor.getString(cursor.getColumnIndex(ProductHelper.COLUMN_NAME));
      String desc=cursor.getString(cursor.getColumnIndex(ProductHelper.COLUMN_DESCRIPTION));
      int  price=cursor.getInt(cursor.getColumnIndex(ProductHelper.COLUMN_PRICE));
      Product c=new Product(id,name,desc,price);
      l.add(c);
    }
  }
  return l;
}
```

נצהיר על ArrayList שבהמשך נכניס לתוכו את המוצרים

נגדיר את השאילתא שבה נשתמש כדי לשלוף את המוצרים מהטבלה

נגדיר מצביע שיצביע למוצר בטבלה

בלולאת while נגדיר שהמצביע יעבור בכל פעם למוצר הבא, עד שיגמרו המוצרים

נשלוף את הנתונים של המוצר שהמצביע מצביע אליו

ניצור מוצר חדש עם הנתונים ששלפנו

נוסיף את המוצר ל ArrayList שהגדרנו

נחזיר את הArrayList עם כל המוצרים

שלב 6 - הוספת אפשרות למחיקת מוצר מהטבלה ב MainActivity:

נוסיף EditText וכפתור למחיקה ונצהיר עליהם:

```
Button btnDelete;

EditText etId;
```

ניתן הפניה לכפתור ול editText ונאזין לכפתור:

```
etId = (EditText) findViewById(R.id.etId);
btnDelete = (Button) findViewById(R.id.btnDelete);
btnDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        productHelper.open();
        long id = Long.valueOf(etId.getText().toString());
        productHelper.deleteCustomerByRow1(id);
        products = productHelper.getAllProducts1();
        if (products.size() > 0) {
            Log.d("data1", products.toString());
        }
        productHelper.close();
    }
});
```

כשהמשתמש ילחץ על btnDelete
ניתן לטבלה אפשרות לעריכה

נשלוף את ה ID שהמשתמש הזין ב EditText

נקרא לפונקציה שנממש בהמשך

נשלוף את כל המוצרים מהטבלה לתוך ArrayList

נדפיס את כל המוצרים שנשארו, כך נוכל
לראות שבאמת נמחק המוצר שרצינו

נסגור את האפשרות
לעריכה של הטבלה

נממש את הפונקציה למחיקת מוצר מהטבלה ב ProductHelper:

```
public void deleteCustomerByRow1(long rowId){
    String str_sql = "delete from " + TABLE_PRODUCT + " where " + ProductHelper.COLUMN_ID + " = " + rowId ;
    database.execSQL(str_sql);
}
```

דרך 1 למחיקת מוצר מהטבלה

ניצור שאילתא למחיקת איבר מטבלה ב SQL לפי ID נריץ אותה

```
public long deleteCustomerByRow2(long rowId){
    return database.delete(ProductHelper.TABLE_PRODUCT, ProductHelper.COLUMN_ID + "=" + rowId, null);
}
```

דרך 2 למחיקה מוצר מהטבלה

נריץ פונקציה למחיקת איבר מטבלה ב SQL לפי ID ונחזיר את האיבר

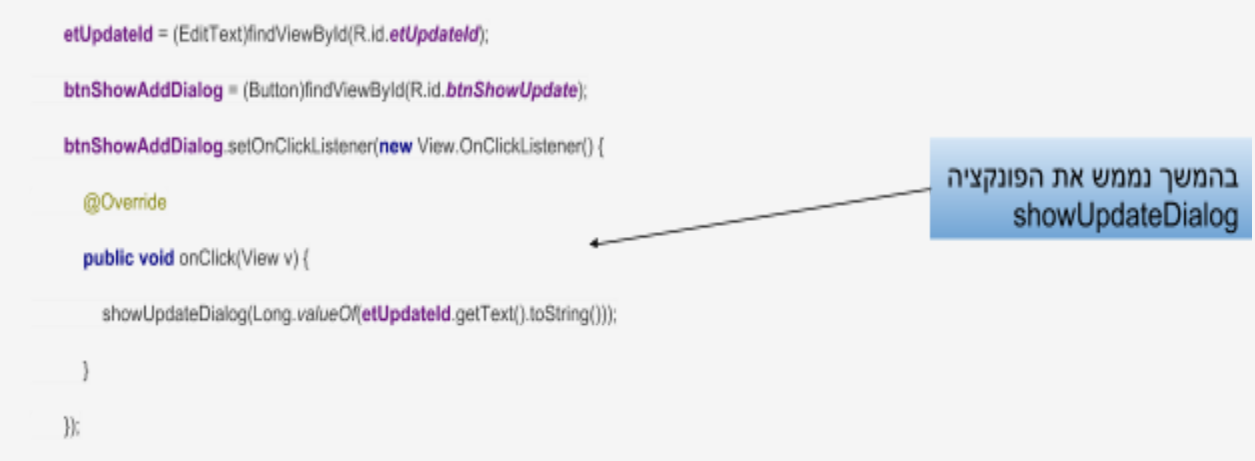
שלב 7 - עדכון הנתונים למוצר בטבלה:

נוסיף כפתור שיפתח לנו דיאלוג לעדכון נתונים למוצר לפי ה ID. (ה layout של הדיאלוג יהיה אותו layout_add אבל נשנה את הכותרת)

נצהיר על הכפתור ועל ה EditText לעדכון נתונים (כבר הוספנו ל XML):

```
Button btnShowUpdateDialog;

EditText etUpdateId;
```

ניתן הפניה לכפתור ול EditText ונאזין לכפתור:



נממש את הפונקציה showUpdateDialog:

```
    btnAdd = (Button)addDialog.findViewById(R.id.btnSave);
    btnAdd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String name = etName.getText().toString();
            String desc = etDesc.getText().toString();
            int price  = Integer.valueOf(etPrice.getText().toString());

            Product p= new Product(id,name,desc,price);
            productHelper.open();
            long sol = productHelper.updateByRow(p);
            productHelper.close();
            if(sol > 0 ) {
                tvSol.setText("product : " + id + " saved");
            }
            else     tvSol.setText("error update : productId " + id);
        }
    });
addDialog.show();
 }
}
```

במידה והמשתמש לחץ על כפתור שמירת הנתונים

נשמור את הנתונים שהמשתמש הזין

ניצור מוצר חדש עם הנתונים שהמשתמש הזין

ניתן לטבלה אפשרות לעריכה

בהמשך נממש את הפונקציה updateByRow בקלאס ProductHelper

נסגור את האפשרות לעריכת הטבלה

נממש את הפונקציה getCustomerById בקלאס ProductHelper:

```
public Product getCustomerById(long rowId){
    Cursor cursor=database.query(ProductHelper.TABLE_PRODUCT, allColumns, ProductHelper.COLUMN_ID + "=" +rowId, null, null, null, null);
    cursor.moveToFirst();
    if(cursor.getCount()>0){
        long id=cursor.getLong(cursor.getColumnIndex(ProductHelper.COLUMN_ID));
        String name=cursor.getString(cursor.getColumnIndex(ProductHelper.COLUMN_NAME));
        String desc=cursor.getString(cursor.getColumnIndex(ProductHelper.COLUMN_DESCRIPTION));
        int  price=cursor.getInt(cursor.getColumnIndex(ProductHelper.COLUMN_PRICE));
        Product c=new Product(id,name,desc,price);
        this.close();
        return c;
    }
    return null;
}
```

נצביע לטבלה, ונגדיר למצביע שלנו לעבור על כל העמודות של המוצר שה id שלו שווה ל id שהעברנו כפרמטר של הפונקציה

נצביע למוצר שעונה על קריטריון ה id שלנו

במידה ובאמת קיים מוצר שעונה על הקריטריון שלנו והמצביע מצביע אליו

נשמור את הנתונים של המוצר

ניצור מוצר חדש עם הנתונים של המוצר ששמרנו

נסגור את האפשרות לעריכת הטבלה

נחזיר את המוצר

נממש את הפונקציה שמעדכנת את הנתונים של המוצר בטבלה:

```
public long updateByRow(Product p){
    ContentValues values=new ContentValues();
    values.put(ProductHelper.COLUMN_ID, p.getProductId());
    values.put(ProductHelper.COLUMN_NAME, p.getName());
    values.put(ProductHelper.COLUMN_DESCRIPTION, p.getDescription());
    values.put(ProductHelper.COLUMN_PRICE, p.getPrice());

    return database.update(ProductHelper.TABLE_PRODUCT, values, ProductHelper.COLUMN_ID +"=" + p.getProductId(), null);
}
```

נערוך את המוצר בעזרת ContentValues

נשמור את הנתונים של המוצר ב ContentValues בעזרת key ו value

נעדכן את המוצר בטבלה ונחזיר את ה ID שלו

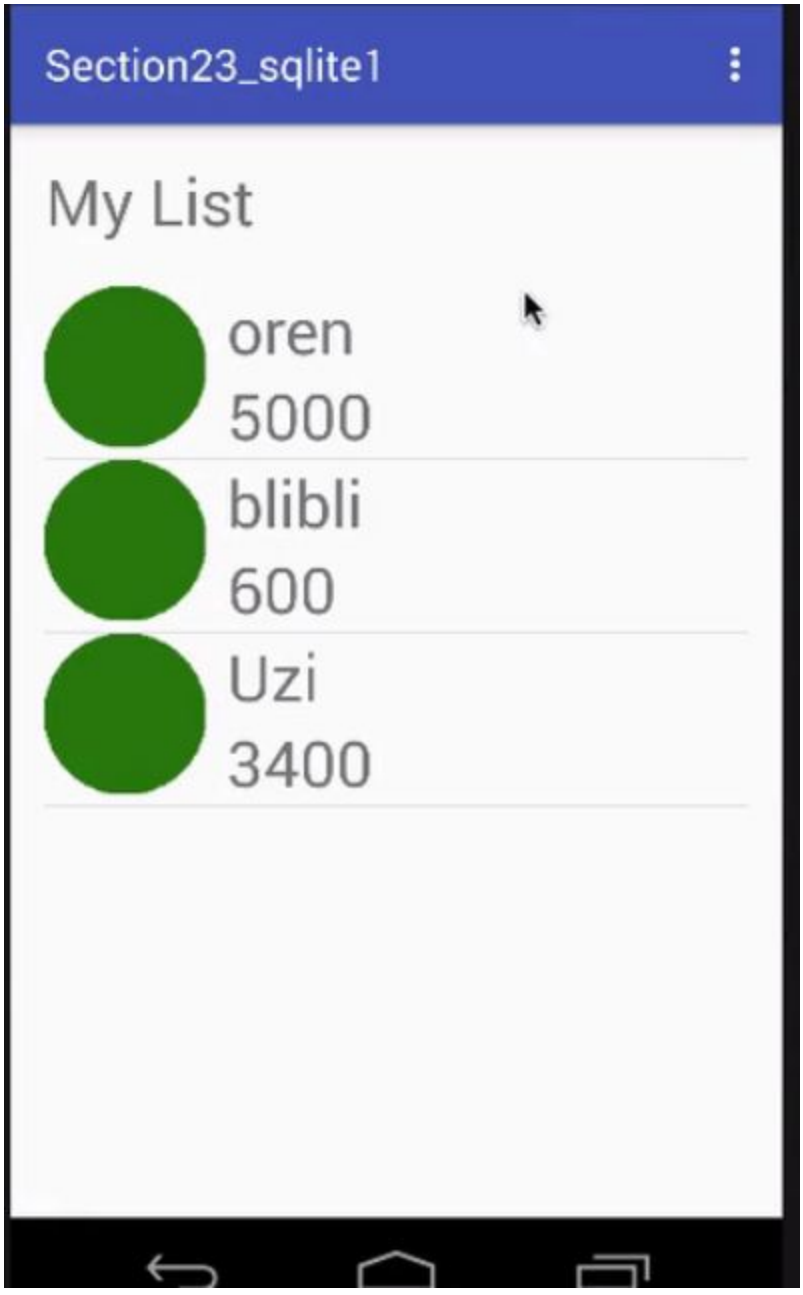
בנינו עד כה אפליקציה בסיסית, רק בשביל לראות איך משתמשים בכל אחת מהפקודות הבסיסיות של SQLite.

בהמשך נבנה אפליקציה מורכבת יותר.

דוגמה 2 לשימוש ב SQLite:

באפליקציה השנייה שנבנה המשתמש יוכל להוסיף, לעדכן ולמחוק לקוחות שישמרו בטבלה SQLite.

נוכל לשלוף את הלקוחות ולהציג אותם ב ListView.



שלב 1 - יצירת class Customer:

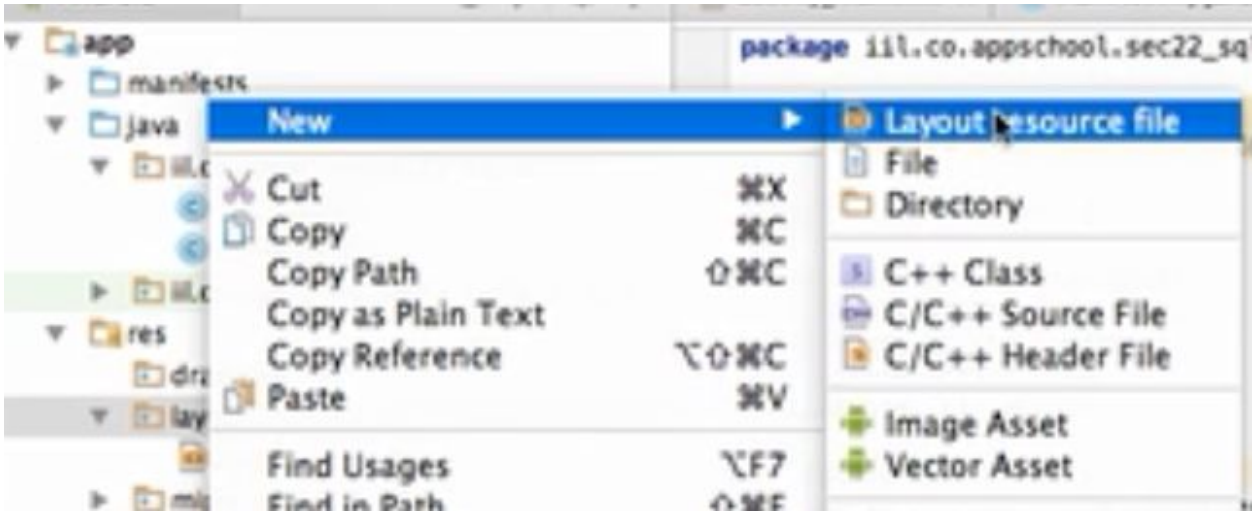
```
public class Customer {
 private long custormerId;
 private String firstName;
 private String lastName;
 private String city;
 private int waste;


 public Customer(){}
 public Customer(long custormerId, String firstName, String lastName, String city, int waste) {
    super();
    this.custormerId = custormerId;
    this.firstName = firstName;
    this.lastName = lastName;
    this.city = city;
    this.waste = waste;
 }
```

Setters ו Getters

```
public long getCustormerId() {
    return custormerId;
}
public void setCustormerId(long custormerId) {
    this.custormerId = custormerId;
}
public String getFirstName() {
    return firstName;
}
public void setFirstName(String firstName) {
    this.firstName = firstName;
}
public String getLastName() {
    return lastName;
}
public void setLastName(String lastName) {
    this.lastName = lastName;
}
```

```
public String getCity() {
    return city;
}
public void setCity(String city) {
    this.city = city;
}
public int getWaste() {
    return waste;
}
public void setWaste(int waste) {
    this.waste = waste;
}

}
```

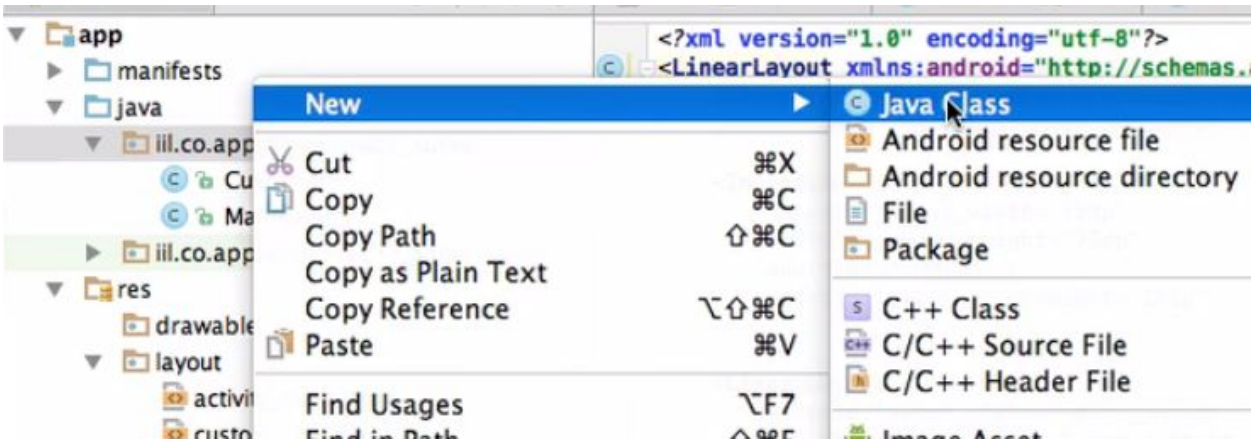
שלב 2 - יצירת custom_row ל ListView:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <ImageView
     android:layout_width="75dp"
     android:layout_height="75dp"
     android:id="@+id/iv"
     android:layout_marginRight="10dp"
     />

  <LinearLayout
     android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:orientation="vertical"
    >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/tvFirstName"
        android:text="title"
        android:textSize="30sp"
        />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/tvWaste"
        android:text="price"
        android:textSize="30sp"
        />

  </LinearLayout>
</LinearLayout>
```

שלב 3 - יצירת Adapter ל ListView:




```
public class CustomerAdapter extends ArrayAdapter<Customer> {

    Context context;

    List<Customer>objects;


    public CustomerAdapter(Context context, int resource, List<Customer> objects) {

        super(context, resource, objects);

        this.objects=objects;

        this.context=context;

    }
```

נירש מ arrayAdapter מסוג Customer

המסך שבו נמצא ה ListView

רשימה של הלקוחות
שבהמשך נוסיף ל ListView

בנאי

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    LayoutInflater layoutInflater = ((Activity)context).getLayoutInflater();
    View view = layoutInflater.inflate(R.layout.custom_row, parent, false);

    TextView tvFirstName =(TextView) view.findViewById(R.id.tvFirstName);
    TextView tvWaste =(TextView) view.findViewById(R.id.tvWaste);
    ImageView iv = (ImageView)view.findViewById(R.id.iv);
    Customer temp = objects.get(position);
    tvWaste.setText(String.valueOf(temp.getWaste()));
    tvFirstName.setText(temp.getFirstName());

    if(temp.getWaste()>500)
        iv.setImageResource(R.drawable.green_c);
    else if(temp.getWaste()<500)
        iv.setImageResource(R.drawable.red_c);

    return view;
  }
}
```

הפונקציה getView מספר פעמים כמספר הלקוחות שברשימה

נתחבר ל inflater של האקטיביטי

"ננפח" את ה custom_row ל View

ניתן הפניה לאובייקטים שבתא ב ListView

נקבל את הלקוח שבמקום ה position ברשימה

נשנה את הנתונים הרשומים באובייקטים שבתא להיות הנתונים של הלקוח שקיבלנו

במידה וכמות הבזבוזים של הלקוח גדולה מ 500 - התמונה שלו תהיה תמונה ירוקה שהוספנו ידנית מהמחשב

במידה וכמות הבזבוזים של הלקוח קטנה מ 500 - התמונה שלו תהיה תמונה אדומה שהוספנו ידנית מהמחשב

נחזיר את ה view שבתא

שלב 4 - יצירת class Helper:

נבנה קלאס שבו פונקציות של קוד שבו נצטרך להשתמש הרבה.
תפקיד הקלאס לעזור לנו להפחית בשורות קוד שחוזרות על עצמן.
בנוסף, כך נוכל לקבל מכל מקום בפרויקט את הנתיב.

```
public class CustomerOpenHelper extends SQLiteOpenHelper {

    public static final String DATABASENAME="customer.db";
    public static final String TABLE_CUSTOMER="tblcustomer";
    public static final int DATABASEVERSION=1;


    public static final String COLUMN_ID="customerId";
    public static final String COLUMN_FIRSTNAME="firstName";
    public static final String COLUMN_LASTNAME="lastName";
    public static final String COLUMN_CITY="city";
    public static final String COLUMN_WASTE="waste";

    private static final String CREATE_TABLE_CUSTOMER="CREATE TABLE IF NOT EXISTS " + TABLE_CUSTOMER + "(" +
COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + COLUMN_FIRSTNAME + " VARCHAR," +
COLUMN_LASTNAME + " VARCHAR," + COLUMN_CITY +" VARCHAR," + COLUMN_WASTE + " INTEGER " + ");";

String []allColumns={CustomerOpenHelper.COLUMN_ID,
CustomerOpenHelper.COLUMN_FIRSTNAME,CustomerOpenHelper.COLUMN_LASTNAME,
    CustomerOpenHelper.COLUMN_CITY,CustomerOpenHelper.COLUMN_WASTE};

SQLiteDatabase database;
```

נצהיר על קבועים סטטיים של שם ה Database, שם הטבלה וגרסת ה database

נצהיר על קבועים סטטיים של שמות כל העמודות בטבלה

נרשום משפט SQL ליצירת טבלה עם כל העמודות שהצהרנו עליהן. בהמשך נריץ את המשפט כשאילתת SQL

נשמור את השמות של כל העמודות במערך String

נצהיר על database מסוג SQLiteDatabase

```
public CustomerOpenHelper(Context context) {
    super(context, DATABASENAME, null, DATABASEVERSION);
    // TODO Auto-generated constructor stub
}
```

נעביר בבנאי את ה context, שם ה database ואת גרסת ה database

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_TABLE_CUSTOMER);
    Log.i("data", "Table customer created");
}
```

ב onCreate נריץ את המשפט ליצירת הטבלה כשאילתת SQL

במידה והמשתמש ירצה לעדכן את הטבלה "נזרוק" את הטבלה הקודמת ונריץ את onCreate עם הנתונים המעודכנים

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_CUSTOMER);
    onCreate(db);
}
```

```
public void open(){
    database=this.getWritableDatabase();
    Log.i("data", "Database connection open");
}

    public Customer createCustomer(Customer c){
    ContentValues values=new ContentValues();
    values.put(CustomerOpenHelper.COLUMN_FIRSTNAME, c.getFirstName());
    values.put(CustomerOpenHelper.COLUMN_LASTNAME, c.getLastName());
    values.put(CustomerOpenHelper.COLUMN_CITY, c.getCity());
    values.put(CustomerOpenHelper.COLUMN_WASTE, c.getWaste());

    long insertId=database.insert(CustomerOpenHelper.TABLE_CUSTOMER, null, values);
    Log.i("data", "Customer " + insertId + "insert to database");
    c.setCustormerId(insertId);
    return c;
}
```

בעזרת הפונקציה open נפתח אפשרות לעריכת הטבלה

כשהמשתמש ירצה להוסיף לקוח לטבלה

הפונקציה מקבלת כפרמטר לקוח

נשמור את הנתונים של הלקוח בשיטת key ו value בעזרת ContentValues

נכניס את הלקוח לטבלה בעזרת ה ContentValues ונקבל חזרה את ה id שלו

נגדיר שה id של הלקוח יהיה ה id שהוחזר אלינו

נחזיר את הלקוח עם ה id

שלב 5 - עיצוב MainActivity:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView android:text="My List"
        android:textSize="30sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <ListView
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:id="@+id/lv"
        android:layout_marginTop="20dp"
        >

    </ListView>
</LinearLayout>
```

שלב 6 - עריכת MainActivity:

```
public class MainActivity extends AppCompatActivity {
    CustomerOpenHelper coh;
    ArrayList<Customer>listOfCustomer;
    ListView lv;
    CustomerAdapter customerAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        lv=(ListView)findViewById(R.id.lv);
        coh=new CustomerOpenHelper(this);
        listOfCustomer=new ArrayList<Customer>();

        Log.i("data", "list size is " + listOfCustomer.size());
        createCustomers();
    }
```

הצהרות

הפניות

בהמשך נממש את הפונקציה createCustomers

```
@Override
protected void onResume() {
    // TODO Auto-generated method stub
    super.onResume();
}

@Override
protected void onPause() {
    // TODO Auto-generated method stub
    super.onPause();
}

public void createCustomers(){
    coh.open();
    Customer c1=new Customer(0,"oren","davidi","tel-aviv",50);
    c1=coh.createCustomer(c1);
    listOfCustomer.add(c1);

    Customer c2=new Customer(0,"blibli","oli","ranat-aviv",600);
    c2=coh.createCustomer(c2);
    listOfCustomer.add(c2);
    coh.close();
}
```

ניתן לטבלה אפשרות לעריכה

ניצור לקוח חדש

נכניס את הלקוח לטבלה

נוסיף את הלקוח לרשימת הלקוחות

ניצור לקוח נוסף

נכניס את הלקוח לטבלה

נוסיף את הלקוח לרשימת הלקוחות

נסגור את האפשרות לעריכת הטבלה

שלב 7 - נציג את כל הלקוחות ב ListView:

נממש ב CustomerOpenHelper פונקציה שבעזרה נוכל לקבל את רשימת הלקוחות מהטבלה:

```
public ArrayList<Customer>getAllCustomers() {
    ArrayList<Customer> l = new ArrayList<Customer>();
   Cursor cursor=database.query(CustomerOpenHelper.TABLE_CUSTOMER, allColumns, null, null, null, null, null);

   if(cursor.getCount()>0){
     while(cursor.moveToNext()){
        long id=cursor.getLong(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_ID));
        String fname=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_FIRSTNAME));
        String lname=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_LASTNAME));
        String city=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_CITY));
        int  waste=cursor.getInt(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_WASTE));
        Customer c=new Customer(id,fname,lname,city,waste);
        l.add(c);
     }
   }
   return l;
}
```

נצהיר על ArrayList

נריץ שאילתת SQL שדרכה נקבל את כל הלקוחות מהטבלה, עם כל העמודות שלהן ללא יוצא מהכלל. נצביע ללקוח הראשון

בעזרת לולאת while נרוץ על כל הלקוחות שבטבלה

נשמור את כל הנתונים של הלקוח שהמצביע מצביע אליו

ניצור לקוח חדש עם הנתונים ששמרנו

נוסיף את הלקוח לרשימה

נחזיר את רשימת הלקוחות

```
public ArrayList<Customer>getAllCustomersByFIlter(String selection,String OrderBy){
   Cursor cursor=database.query(CustomerOpenHelper.TABLE_CUSTOMER,
                                allColumns, selection, null, null, null, OrderBy);
   ArrayList<Customer>l=convertCurserToList(cursor);
   return  l;
}
```

פונקציה שממחזיר רשימה של כל הלקוחות לפי פילטר שנבחר

בהמשך נממש את הפונקציה convertCursorToList

נחזיר את רשימת הלקוחות

```
private ArrayList<Customer> convertCurserToList(Cursor cursor) {
 ArrayList<Customer>l=new ArrayList<Customer>();

 if(cursor.getCount()>0){
   while(cursor.moveToNext()){
     long id=cursor.getLong(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_ID));
     String fname=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_FIRSTNAME));
     String lname=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_LASTNAME));
     String city=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_CITY));
     int  waste=cursor.getInt(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_WASTE));
     Customer c=new Customer(id,fname,lname,city,waste);
     l.add(c);
   }
 }
 return l;
}
```

ניצור ArrayList חדש

נעבור כל הלקוחות בעזרת לולאת while

נשמור את הנתונים של הלקוח

ניצור לקוח חדש עם הנתונים ששמרנו

נוסיף את הלקוח לרשימה

נחזיר את רשימת הלקוחות

ב MainActivity נקבל את רשימת הלקוחות ונכניס אותה ל ListView:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    lv=(ListView)findViewById(R.id.lv);
    coh=new CustomerOpenHelper(this);
    listOfCustomer=new ArrayList<Customer>();
    coh.open();
    listOfCustomer=coh.getAllCustomers();
    coh.close();

    Log.i("data", "list size is " + listOfCustomer.size());
    if(listOfCustomer.size()==0)
        createCustomers();

    customerAdapter=new CustomerAdapter(this,0,listOfCustomer);
    lv.setAdapter(customerAdapter);
}
```

הפעם נפתח גישה ל database כדי לקרוא לפונקציה getAllCustomers

נכניס את רשימת הלקוחות ששלפנו מה database לאדפטר

ניתן ל ListView את האדפטר שיצרנו

שלב 8 - נוסיף פונקציה ל CustomerOpenHelper למציאת לקוח לפי id:

```
public Customer getCustomerById(long rowId){

  Cursor cursor=database.query(CustomerOpenHelper.TABLE_CUSTOMER, allColumns,
CustomerOpenHelper.COLUMN_ID + "=" +rowId, null, null, null, null);

  cursor.moveToFirst();


  if(cursor.getCount()>0){

     long id=cursor.getLong(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_ID));

     String fname=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_FIRSTNAME));

     String lname=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_LASTNAME));

     String city=cursor.getString(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_CITY));

     int  waste=cursor.getInt(cursor.getColumnIndex(CustomerOpenHelper.COLUMN_WASTE));

     Customer c=new Customer(id,fname,lname,city,waste);

     this.close();

     return c;

  }
```

נריץ שאילתא להצגת כל העמודות של הלקוח עם ה ID המתאים

נצביע ללקוח הראשון

נבדוק שבאמת קיים לקוח עם ID כזה

נשמור את הנתונים של הלקוח שאנחנו מצביעים אליו

ניצור לקוח חדש עם הנתונים ששמרנו

נסגור את האפשרות לעריכת הטבלה

נחזיר את הלקוח עם ה ID המתאים

שלב 9 - מחיקת נתונים מהטבלה:

נוסיף ל CustomerOpenHelper פונקציה שמוחקת את כל הלקוחות:

```
public long deleteAll(){
return database.delete(CustomerOpenHelper.TABLE_CUSTOMER, null, null);
}
```

נחזיר את הטבלה ללא לקוחות

נוסיף גם פונקציה שמוחקת לקוח מסוים לפי ID:

```
public long deleteCustomerByRow(long rowId){
return database.delete(CustomerOpenHelper.TABLE_CUSTOMER, CustomerOpenHelper.COLUMN_ID + "=" + rowId, null);
}
```

הפונקציה delete מוחקת לקוחות לפי התנאי שהיא מקבלת כפרמטר השני

שלב 10 - נוסיף ל CustomerOpenHelper פונקציה לעדכון נתונים של לקוח בטבלה

```
public long updateByRow(Customer c){
  ContentValues values=new ContentValues();
  values.put(CustomerOpenHelper.COLUMN_ID, c.getCustormerId());
  values.put(CustomerOpenHelper.COLUMN_FIRSTNAME, c.getFirstName());
  values.put(CustomerOpenHelper.COLUMN_LASTNAME, c.getLastName());
  values.put(CustomerOpenHelper.COLUMN_CITY, c.getCity());
  values.put(CustomerOpenHelper.COLUMN_WASTE, c.getWaste());

  return database.update(CustomerOpenHelper.TABLE_CUSTOMER, values,
CustomerOpenHelper.COLUMN_ID +"=" + c.getCustormerId(), null);
}
```

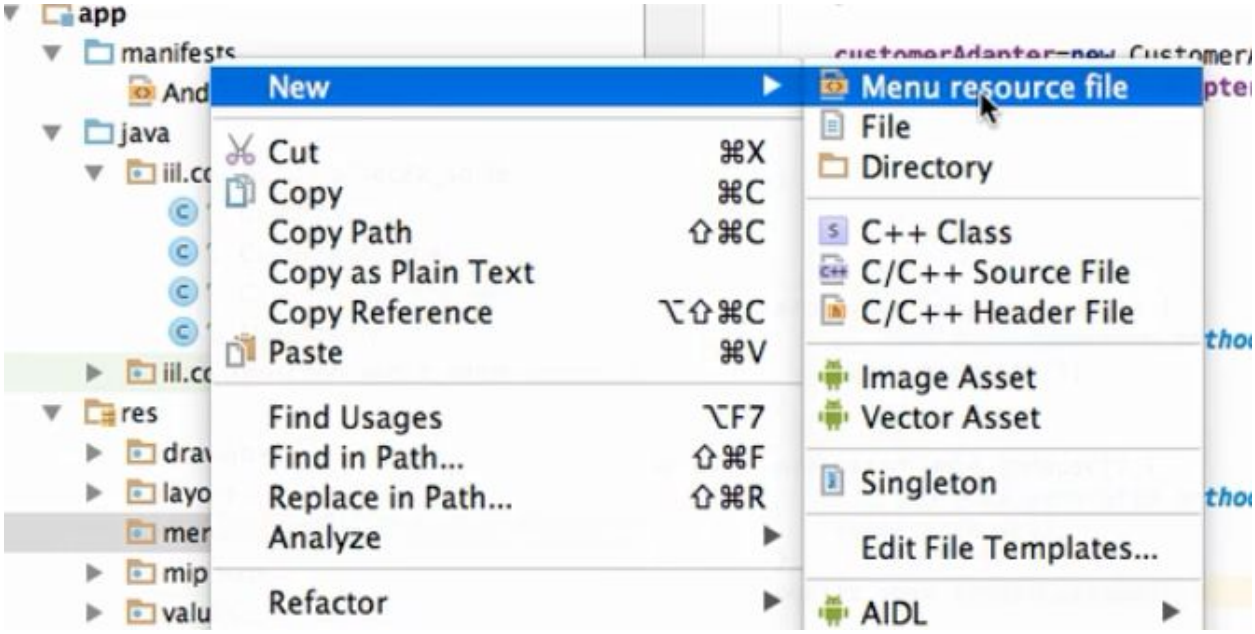
נשמור את הנתונים של הלקוח בעזרת key ו value בשיטת ContentValues

נעדכן את הנתונים של הלקוח בטבלה לפי ה ID שלו.
נחזיר את הטבלה המעודכנת

שלב 11 - הוספת תפריט לטבלה

נוסיף תפריט לטבלה, בעזרתה המשתמש יוכל לבחור את איזה לקוחות הוא רוצה לראות.
בנוסף המשתמש יוכל לעבור דרך התפריט למסך InsertActivity שאותו נבנה בהמשך,
ולהוסיף לקוח לטבלה.

ניצור תיקיה menu תחת resources ותחתיה קובץ בשם main_menu מסוג
Menu resource file.



נעצב את התפריט:

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:tools="http://schemas.android.com/tools"

  xmlns:android="http://schemas.android.com/apk/res/android">


  <item

    android:id="@+id/menu_settings"

    android:orderInCategory="100"
```

```
    android:showAsAction="never"
    android:title="setting"
    tools:ignore="AppCompatResource" />
  <item
    android:id="@+id/menu_allcustomers"
    android:orderInCategory="100"
    android:showAsAction="always"
    tools:ignore="AppCompatResource"
    android:title="all customers"/>

  <item
    android:id="@+id/menu_cheapcustomers"
    android:orderInCategory="100"
    android:showAsAction="never"
    tools:ignore="AppCompatResource"
    android:title="cheapers"/>

  <item
    android:id="@+id/menu_richcustomers"
    android:orderInCategory="100"
    android:showAsAction="never"
    tools:ignore="AppCompatResource"
```

```
    android:title="reachcustomers"/>

  <item
    android:id="@+id/menu_new"
    android:orderInCategory="100"
    android:showAsAction="never"
    tools:ignore="AppCompatResource"
    android:title="New"/>

</menu>
```

המשתמש יוכל לראות את הלקוחות העשירים ביותר, את העניים ביותר או את כל הלקוחות.

שלב 12 - מעבר בבחירה מהתפריט, מהמסך MainActivity למסך InsertActivity:

נוסיף את שתי הפונקציות הבאות לשליטה בתפריט ל MainActivity:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main_menu, menu);
    return true;
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    coh.open();
    switch (item.getItemId()) {
        case R.id.menu_allcustomers:
            listOfCustomer=coh.getAllCustomers();
            Log.i("filter", "list count is " + listOfCustomer.size());
            refreshMyAdapter();
            break;
        case R.id.menu_new:
            Intent i=new Intent(MainActivity.this,InsertActivity.class);
            startActivityForResult(i, 1);//Request code 1 is for ------>insert screen
        default:
            break;
    }
    coh.close();

    return super.onOptionsItemSelected(item);
}
```

במקום להשתמש בהרבה משפטי if נשתמש ב switch (לא חובה)

נפתח אפשרות לעריכת ה database

נקבל את ה id של האפשרות אותה בחר המשתמש

במידה והמשתמש בחר בהצגת כל הלקוחות

נקבל את כל הלקוחות

בהמשך נממש את הפונקציה refreshMyAdapter

במידה והמשתמש בחר בהוספת לקוח

נגדיר intent למעביר מ MainActivity ל InsertActivity

נפעיל את ה intent

נסגור את האפשרות לעריכת ה database

נממש את הפונקציה refreshMyAdapter:

```
public void refreshMyAdapter(){
    customerAdapter=new CustomerAdapter(this,0,listOfCustomer);
    lv.setAdapter(customerAdapter);
}
```

נגדיר את האדפטר

ניתן ל ListView את האדפטר שהגדרנו

שלב 13 - עיצוב המסך insertActivity:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".InsertMainActivity"
  android:orientation="vertical"
  android:background="@drawable/list"
  >

  <TextView
     android:id="@+id/inserttvId"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Insert new Customer"
     android:layout_gravity="center"
     android:textSize="15sp"
     android:layout_marginTop="90dp"
     />

  <EditText
     android:id="@+id/insertEtFname"
     android:layout_marginTop="20dp"
```

```
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:ems="10"
    android:hint="First name"/>

  <EditText
    android:id="@+id/insertEtLname"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:ems="10"
    android:hint="Last name"
    />

  <EditText
    android:id="@+id/insertEtCity"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:ems="10"
    android:hint="City"
```

```
    />

  <EditText
    android:id="@+id/insertEtWaste"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:ems="10"
    android:hint="Waste"
    />

  <Button
    android:id="@+id/insertBtnSave"
    android:layout_width="200dp"
    android:layout_height="50dp"
    android:layout_gravity="center"
    android:layout_marginTop="20dp"
    android:text="Save And Insert"
    android:background="@drawable/button_green" />
</LinearLayout>
```

שלב 14 - עריכת המסך InsertActivity:

```
public class InsertActivity extends AppCompatActivity {
    EditText etfanme, etlname, etcity, etwaste, tvId;
    Button btnsave;
    CustomerOpenHelper cds;
    long id;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_insert);

        cds=new CustomerOpenHelper(this);
        init();
    }

    public void init(){
        etfanme=(EditText) findViewById(R.id.insertEtFname);
        etlname=(EditText) findViewById(R.id.insertEtLname);
        etcity=(EditText) findViewById(R.id.insertEtCity);
        etwaste=(EditText) findViewById(R.id.insertEtWaste);
        btnsave=(Button) findViewById(R.id.insertBtnSave);
        tvId=(TextView) findViewById(R.id.inserttvId);
```

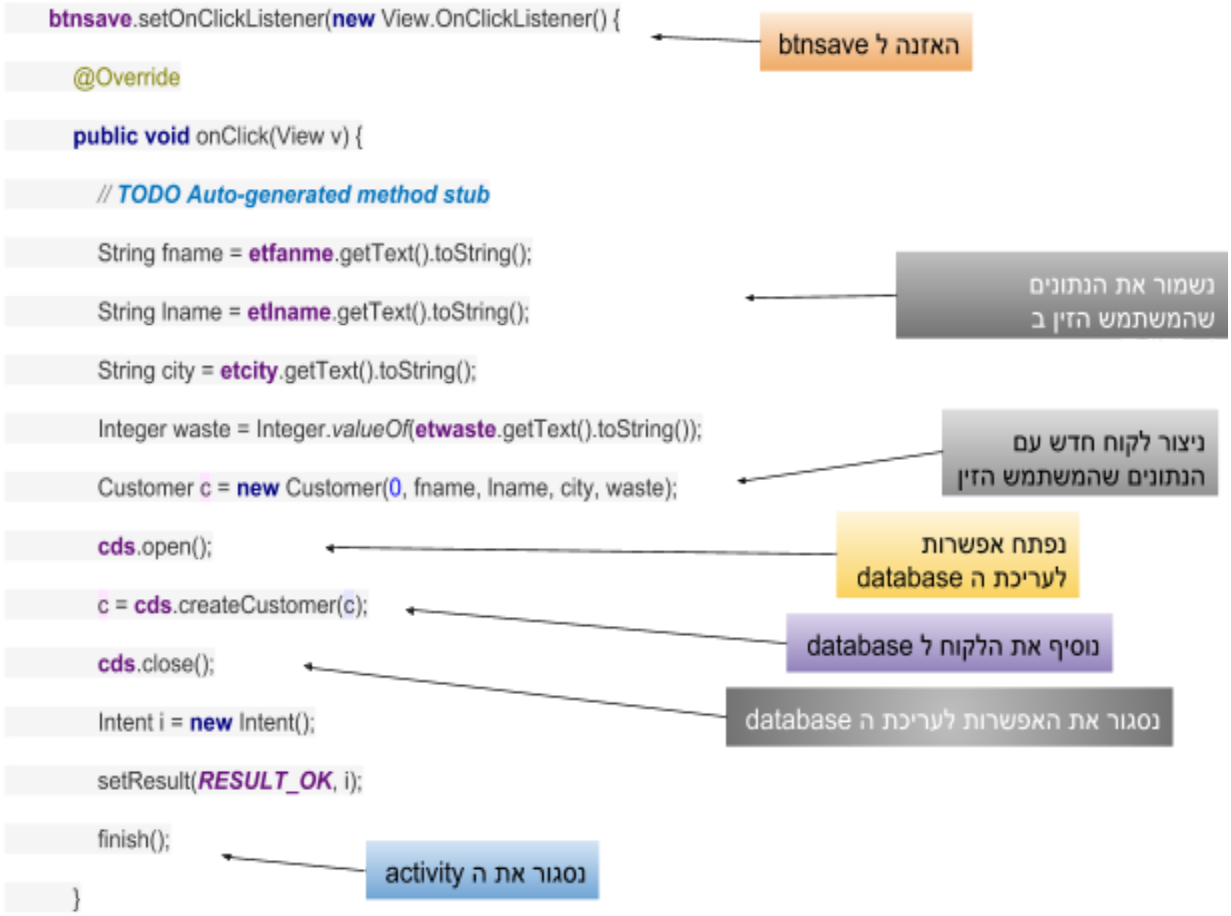
הצהרות

יצירת CustomerOpenHelper חדש

בהמשך נממש את הפונקציה init

בפונקציה init ניתן הפניות לאובייקטים שלנו (נוכל לעשות זאת גם ב onCreate, אבל בשביל הנוחות יצרנו פונקציה חדשה)

```
    btnsave.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            String fname = etfanme.getText().toString();
            String lname = etlname.getText().toString();
            String city = etcity.getText().toString();
            Integer waste = Integer.valueOf(etwaste.getText().toString());
            Customer c = new Customer(0, fname, lname, city, waste);
            cds.open();
            c = cds.createCustomer(c);
            cds.close();
            Intent i = new Intent();
            setResult(RESULT_OK, i);
            finish();
        }
    });
  }
}
```



נוסיף ל MainActivity את הפונקציה onActivityResult:

כשנחזור ממסך להוספת לקוח או לעדכון נתוני לקוח (בהמשך נוסיף) נגיע לפונקציה onActivityResult, כיוון שהגענו אליהם עם startActivityForResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // TODO Auto-generated method stub
    super.onActivityResult(requestCode, resultCode, data);

    if (resultCode==RESULT_OK){
        coh.open();
        listOfCustomer=coh.getAllCustomers();
        refreshMyAdapter();
        coh.close();
        if(requestCode==0)
            Toast.makeText(getBaseContext(), "Database updated", Toast.LENGTH_SHORT).show();
        else if (requestCode==1)
            Toast.makeText(getBaseContext(), "New Customer add to database", Toast.LENGTH_SHORT).show();
    }
  }
}
```

נפתח אפשרות לעריכת ה database

נעדכן את רשימת הלקוחות

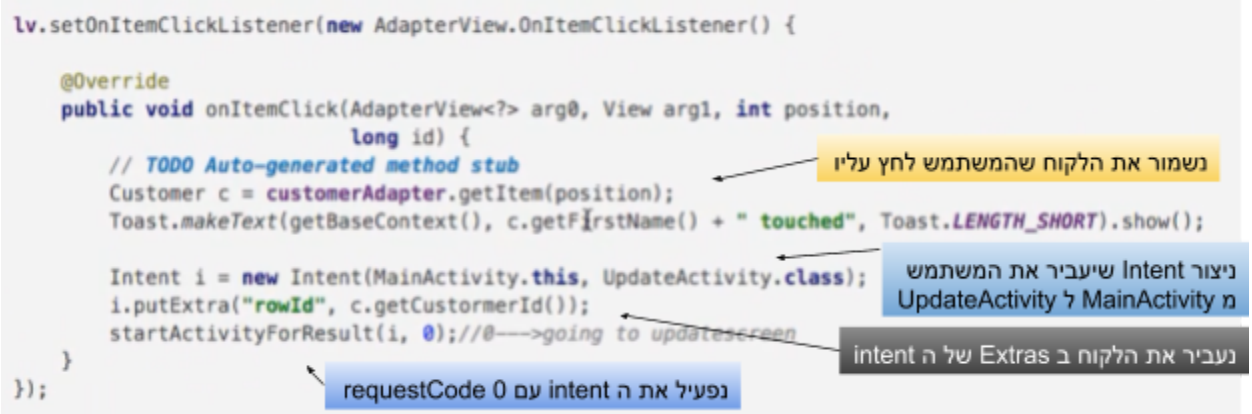
נפעיל את הפונקציה refreshMyAdapter שמימשנו

נסגור את האפשרות לעריכת ה database

בהמשך נוסיף אפשרות לעריכת נתוני לקוח. למסך זה ה requestCode יהיה 0

שלב 14 - לחיצה על לקוח ברשימה תעביר אותנו למסך UpdateActivity:

נממש ב MainActivity האזנה ללחיצה על איבר ב ListView:



שלב 15 - עיצוב UpdateActivity:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/an
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UpdateMainActivity"
    android:orientation="vertical"
    android:background="@drawable/list"
    >
    <TextView
        android:id="@+id/updatetvId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Update details of customer"
        android:layout_gravity="center"

        android:layout_marginTop="90dp"
        />
    <EditText
        android:id="@+id/updateEtFname"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:ems="10"
        android:hint="First name"
        />
    <EditText
        android:id="@+id/updateEtLname"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:ems="10"
        android:hint="Last name"
        />
    <EditText
        android:id="@+id/updateEtCity"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
```

```
        android:hint="City"

        />
    <EditText
        android:id="@+id/updateEtWaste"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:ems="10"
        android:hint="Waste"

        />
    <Button
        android:id="@+id/updateBtnSave"
        android:layout_width="200dp"
        android:layout_height="30dp"
        android:layout_gravity="center"
        android:layout_marginTop="10dp"
        android:text="Save"
        android:background="@drawable/button_green" />

    <Button
        android:id="@+id/updateBtnDelete"
        android:layout_width="200dp"
        android:layout_height="30dp"
        android:layout_gravity="center"
        android:layout_marginTop="25dp"
        android:text="Delete"
        android:background="@drawable/button_red" />



</LinearLayout>
```

שלב 16 - עריכת ה UpdateActivity:

```
public class UpdateActivity extends AppCompatActivity {

    EditText etfanme, etlname, etcity, etwaste;
    Button btnsave, btndelete;
    TextView tvId;
    CustomerOpenHelper coh;
    long id;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update);


        coh=new CustomerOpenHelper(this);
        init();
    }
```

הצהרות

ניצור CustomerOpenHelper חדש

בהמשך נממש את הפונקציה init

```
public void init(){
    etfanme=(EditText) findViewById(R.id.updateEtFname);
    etlname=(EditText) findViewById(R.id.updateEtLname);
    etcity=(EditText) findViewById(R.id.updateEtCity);
    etwaste=(EditText) findViewById(R.id.updateEtWaste);
    btnsave=(Button) findViewById(R.id.updateBtnSave);
    btndelete=(Button) findViewById(R.id.updateBtnDelete);
    tvId=(TextView) findViewById(R.id.updatetvId);

    id=getIntent().getLongExtra("rowId", 0);
    if(id!=0){
        coh.open();
        Customer c=coh.getCustomerById(id);
        etfanme.setText(c.getFirstName());
        etlname.setText(c.getLastName());
        etcity.setText(c.getCity());
        etwaste.setText(String.valueOf(c.getWaste()));
        tvId.setText("Details of customer" + c.getCustormerId());
        coh.close();
    }
```

הפניות לאובייקטים שלנו

נשמור את ה id שהעברנו ב extras של ה intent

נפתח אפשרות לעריכת ה database

נשלוף את הלקוח מה database בעזרת ה id

נגדיר את הטקסט שיופיע ב editText כשהמשתמש יגיע למסך להיות הנתונים של הלקוח

נסגור את האפשרות לעריכת ה database

```
btnsave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub
        String fname=etfanme.getText().toString();
        String lname=etlname.getText().toString();
        String city=etcity.getText().toString();
        Integer waste=Integer.valueOf(etwaste.getText().toString());

        Customer c=new Customer(id,fname,lname,city,waste);
        coh.open();
        coh.updateByRow(c);
        coh.close();
        Intent i=new Intent();
        setResult(RESULT_OK,i);
        finish();
    }
});
```

האזנה ללחיצה על btnsave

נשלוף את הנתונים שהמשתמש הזין ב EditTextים

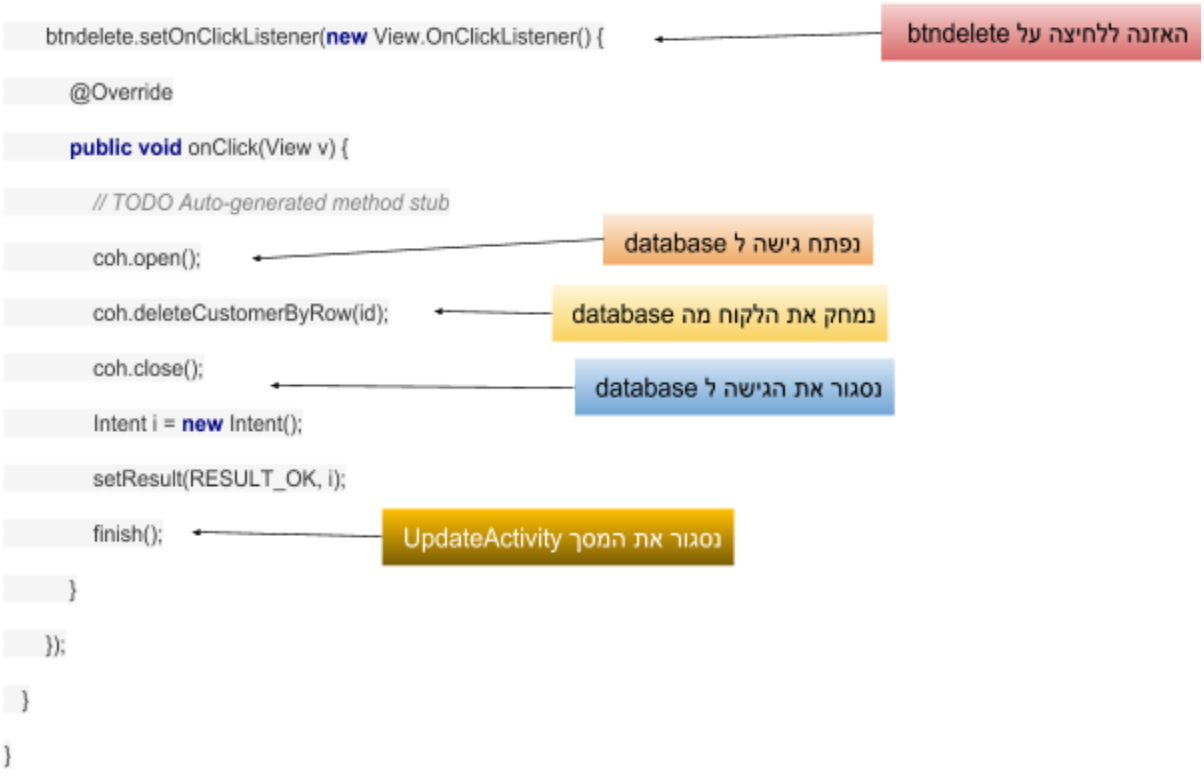
ניצור לקוח חדש עם הנתונים שהמשתמש הזין וה id ששמרנו

נפתח אפשרות לעריכת ה database

נעדכן את הנתונים של הלקוח ב database בעזרת הפונקציה updateByRow שיצרנו

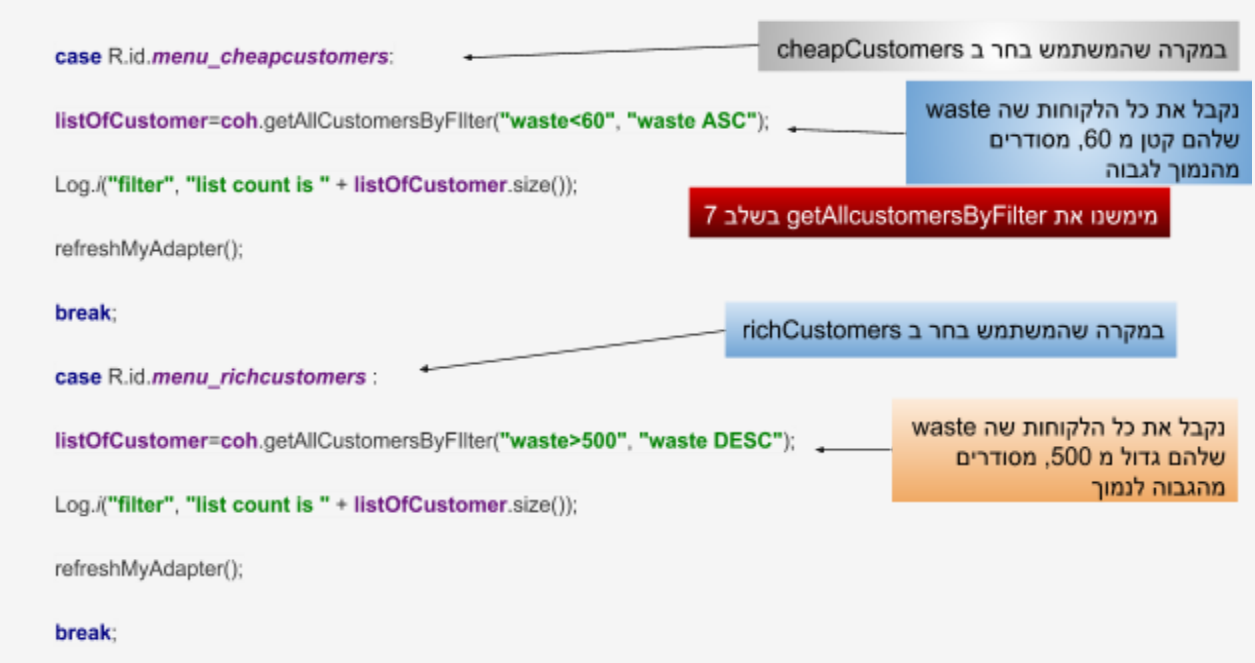
נסגור את הגישה לdatabase

נסגור את המסך UpdateActivity



שלב 7 - הוספת פילטרים לסינון הלקוחות לפי הבחירה של המשתמש בתפריט:

נוסיף ל onOptionsItemSelected ב MainActivity פילטר לסינון הלקוחות שנציג לפי הבחירה של המשתמש בתפריט:



זהו זה! סיימנו לבנות את האפליקציה שלנו ב SQLite.

הריצו את האפליקציה :)