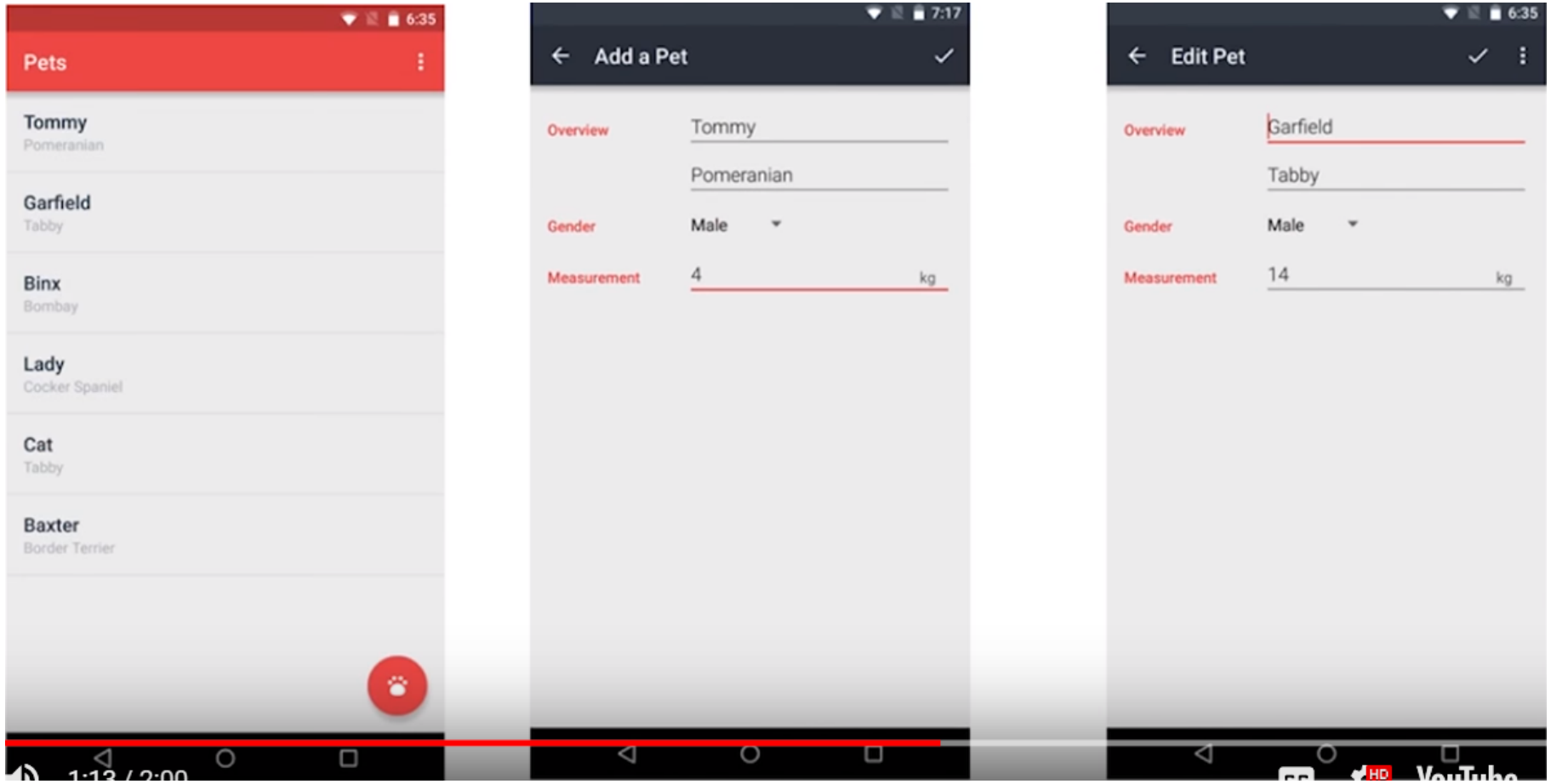
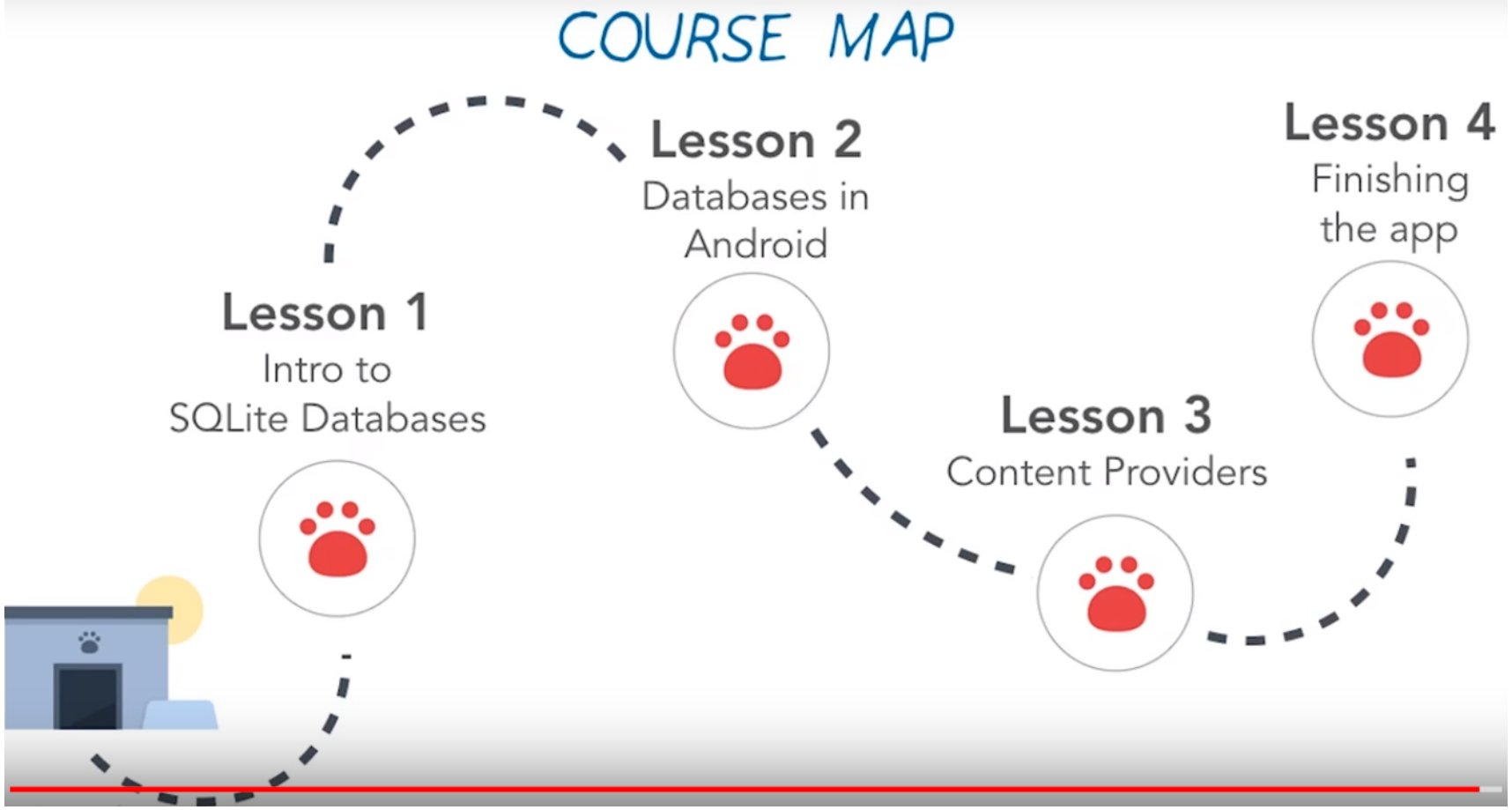
Pets App

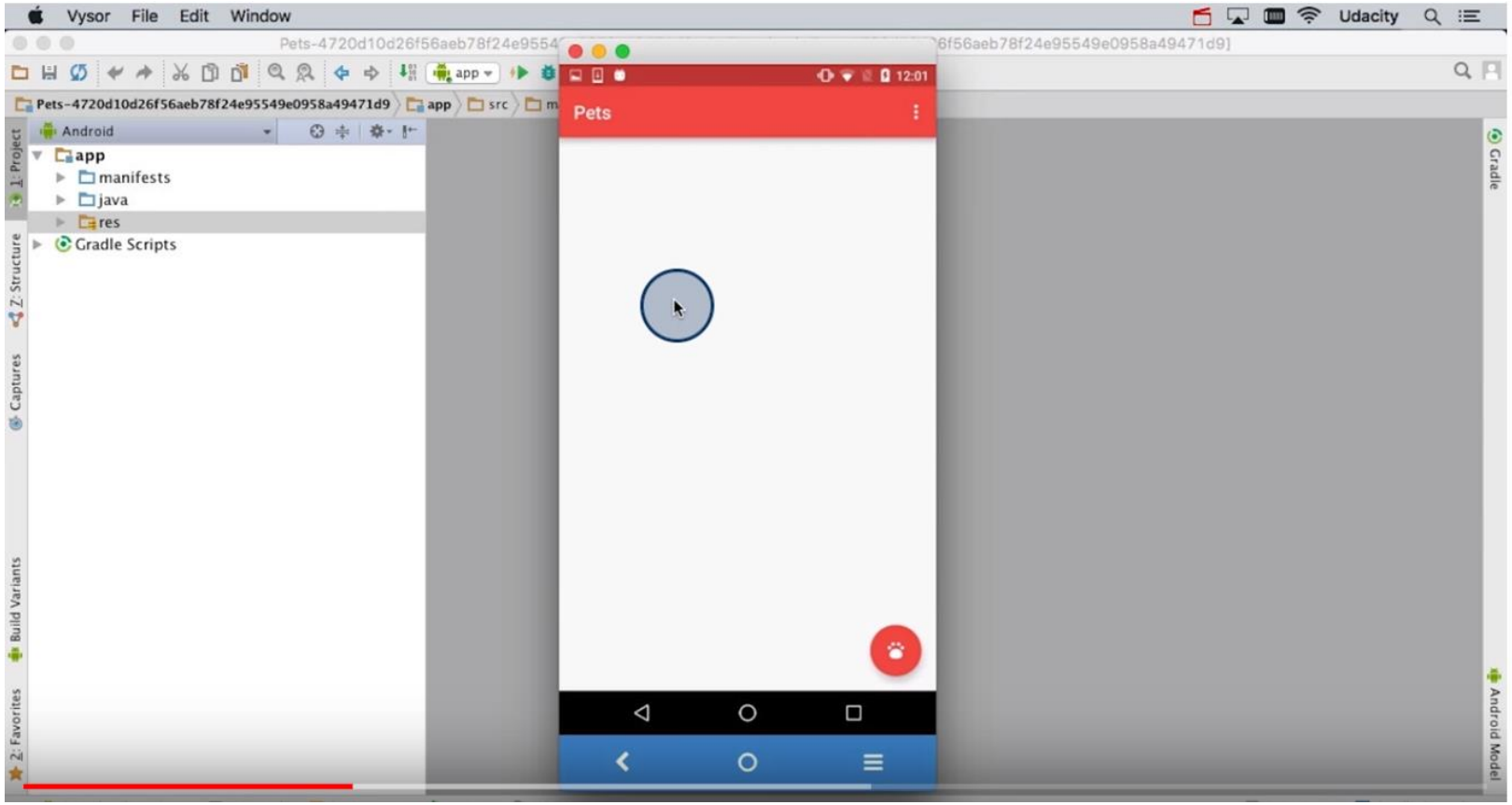
Pets App


6:35
Pets
Tommy
Pomeranian
Garfield
Tabby
Binx
Bombay
Lady
Cocker Spaniel
Cat
Tabby
Baxter
Border Terrier
7:17
Add a Pet
Overview
Tommy
Pomeranian
Gender
Male
Measurement
4
kg
6:35
Edit Pet
Overview
Garfield
Tabby
Gender
Male
Measurement
14
kg


COURSE MAP
Lesson 1
Intro to
SQLite Databases
Lesson 2
Databases in
Android
Lesson 3
Content Providers
Lesson 4
Finishing
the app

Download Starter Code

- https://github.com/udacity/ud845-Pets/tree/starting-point



Pets-4720d10d26f56aeb78f24e95549e0958a49471d9 > app > src > main > AndroidManifest.xml

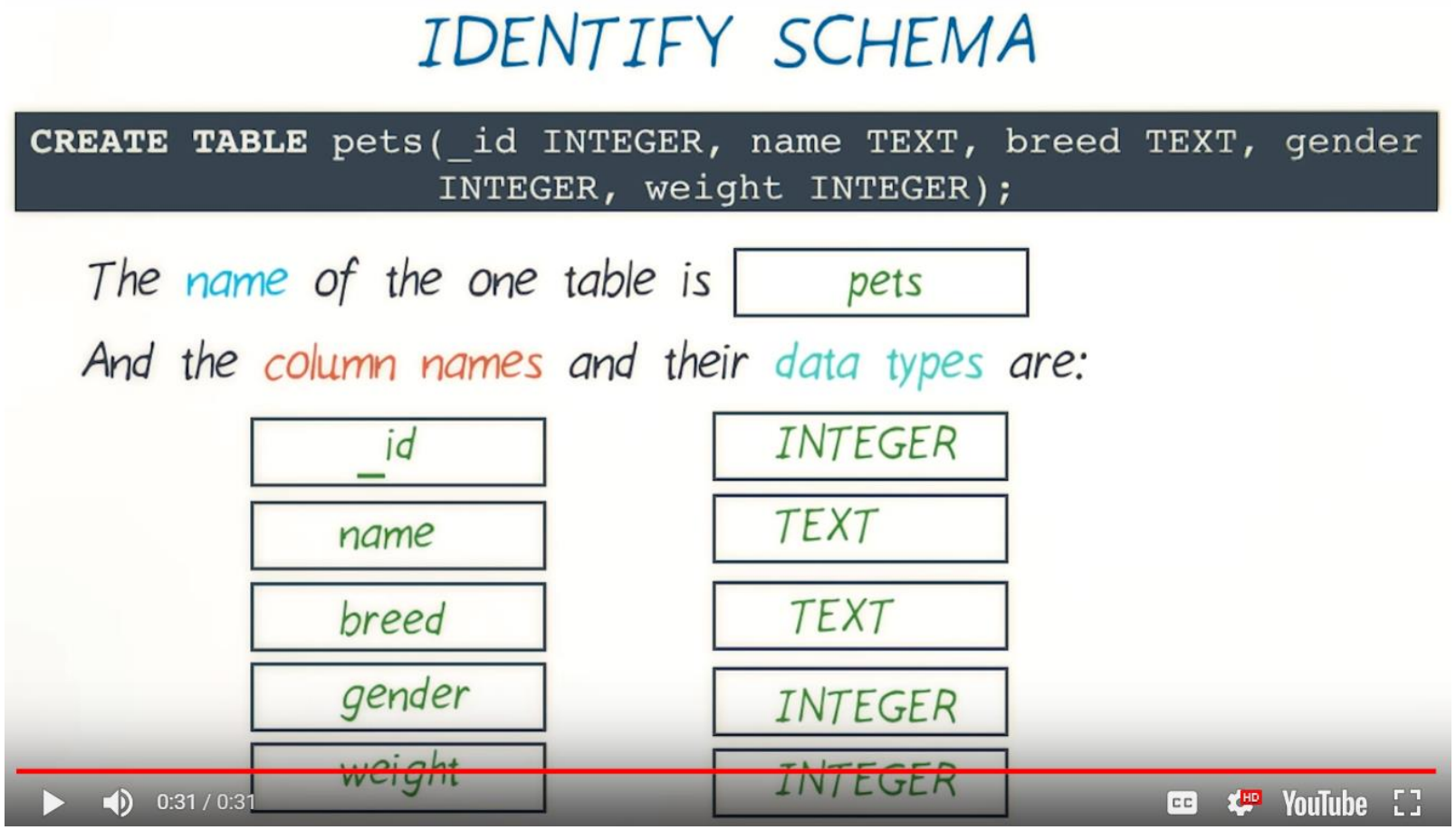
Android

- app
  - manifests
  - java
    - com.example.android.pets
      - CatalogActivity
      - EditorActivity
  - res
    - drawable
    - layout
      - activity_catalog.xml
      - activity_editor.xml
    - menu
    - mipmap
    - values
- Gradle Scripts

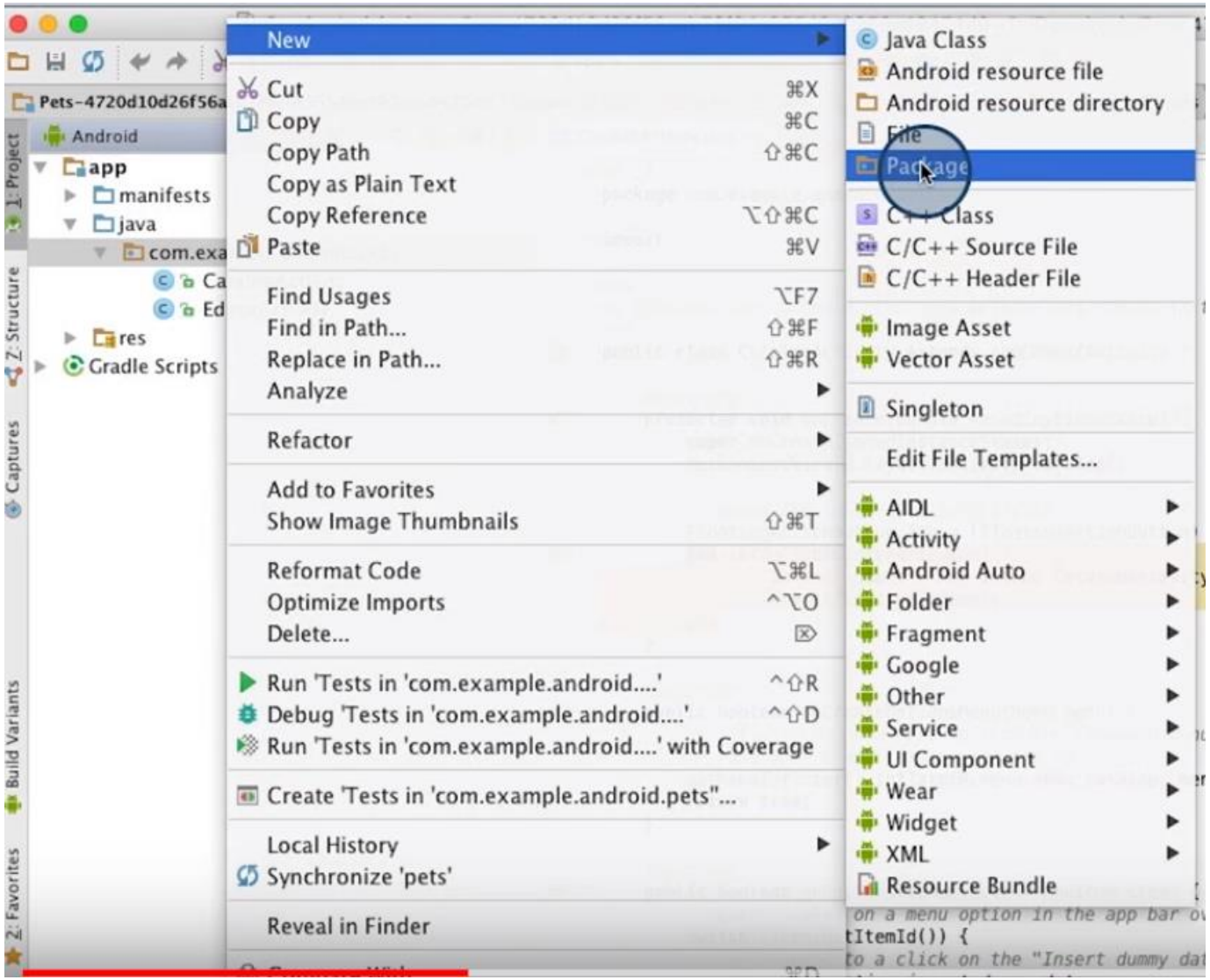
AndroidManifest.xml

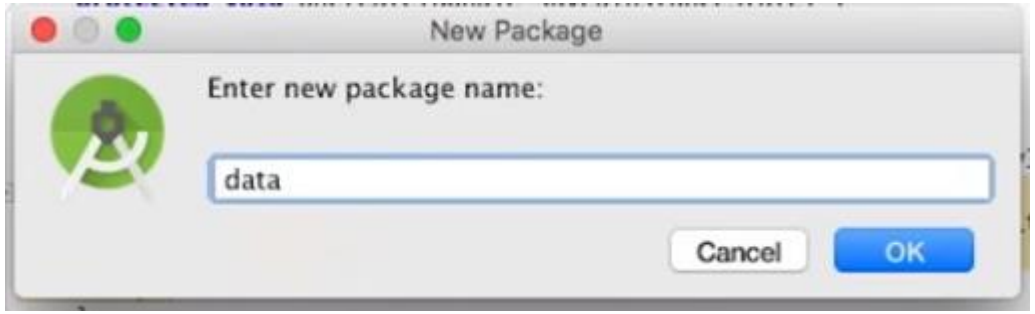
```
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.
-->
<manifest package="com.example.android.pets"
          xmlns:android="http://schemas.android.com/apk/res/android">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Pets"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".CatalogActivity"
            android:label="Pets">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity
            android:name=".EditorActivity"
            android:label="Add a Pet"
            android:theme="@style/EditorTheme"
            android:parentActivityName=".CatalogActivity" >
            <!-- Parent activity meta-data to support 4.0 and lower -->
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".CatalogActivity" />
        </activity>
    </application>
```

Schema and Contract

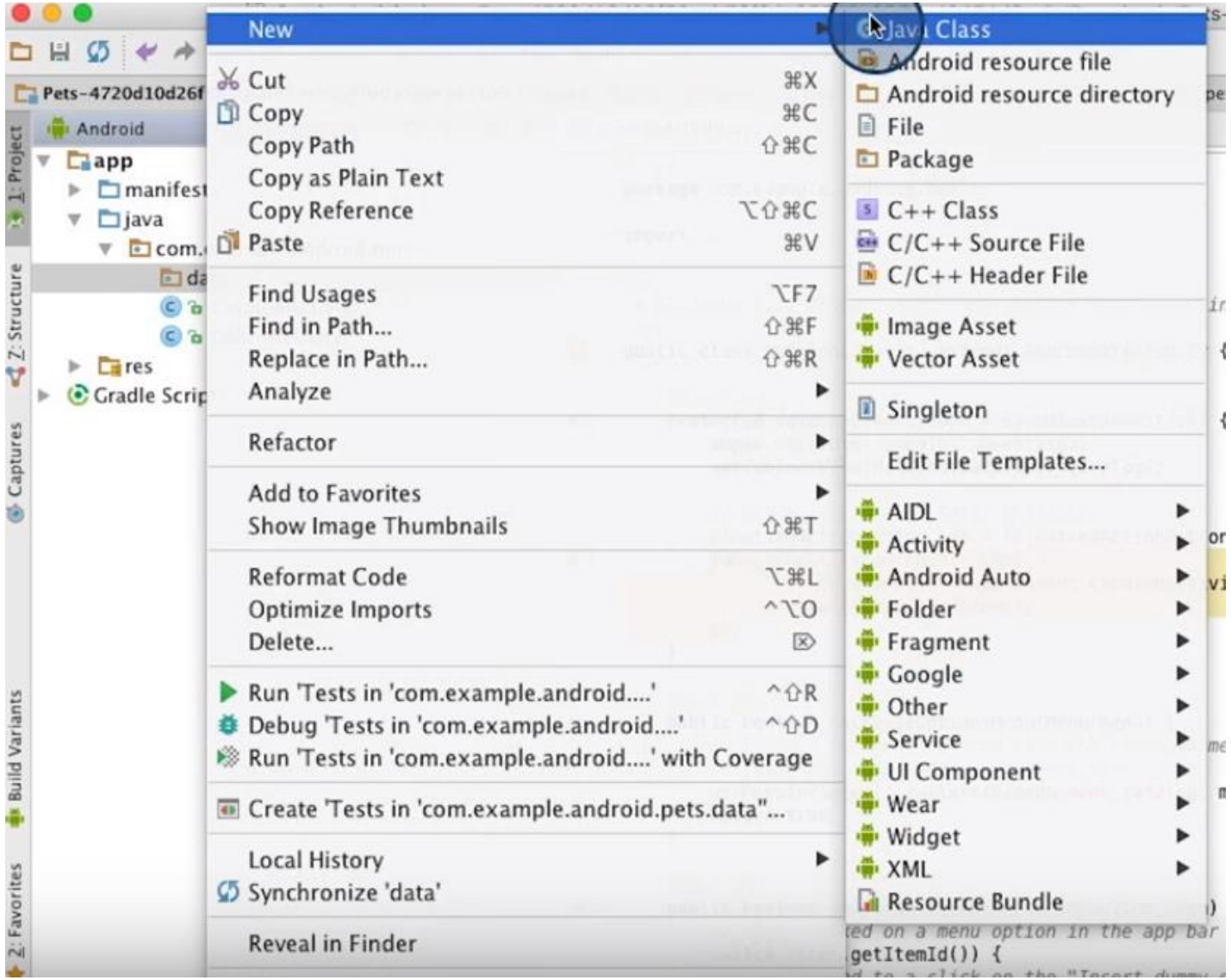


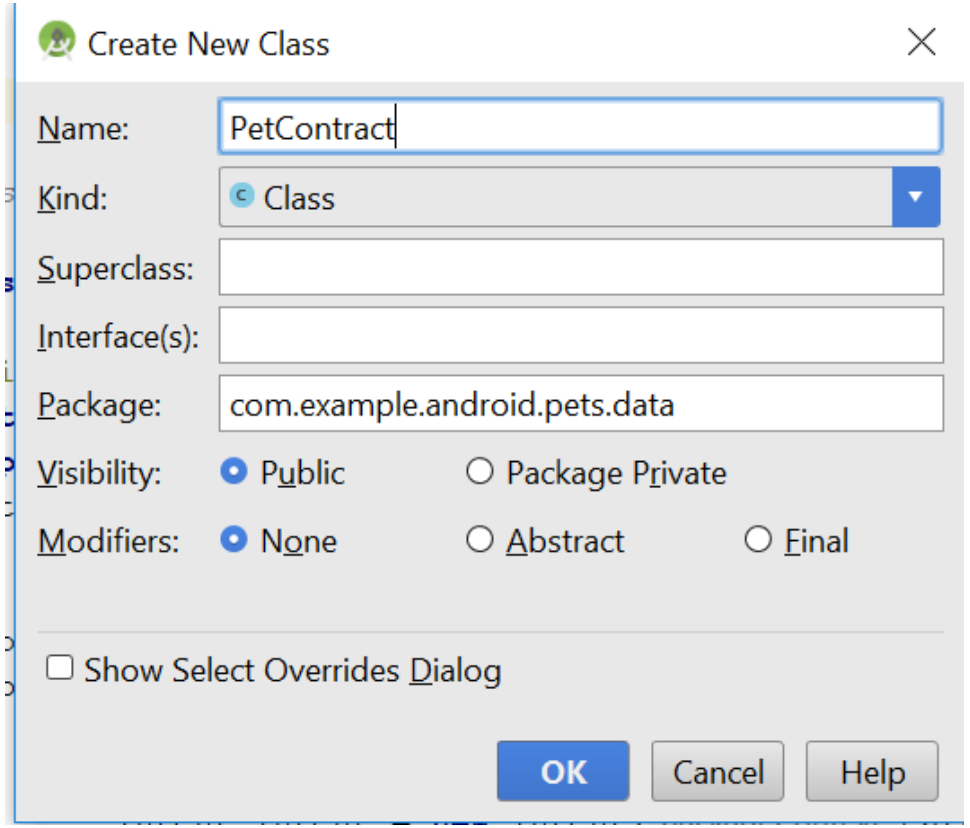
01 - Add PetContract class (create package data)





Create PetContract.java





Contract Class

```java
package com.example.android.pets.data;

import android.provider.BaseColumns;

/**
 * Created by KnowIt on 8/3/16.
 */
public final class PetContract {

    private PetContract() {}


    public static final class PetEntry implements BaseColumns{

        public final static String TABLE_NAME = "pets";

        public final static String _ID = BaseColumns._ID;
        public final static String COLUMN_PET_NAME ="name";
        public final static String COLUMN_PET_BREED = "breed";
        public final static String COLUMN_PET_GENDER = "gender";
        public final static String COLUMN_PET_WEIGHT = "weight";

        public static final int GENDER_UNKNOWN = 0;
        public static final int GENDER_MALE = 1;
        public static final int GENDER_FEMALE = 2;


    }
}
```

Modify EditorActivity.java

```
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    String selection = (String) parent.getItemAtPosition(position);
    if (!TextUtils.isEmpty(selection)) {
        if (selection.equals("Male")) {
            mGender = PetContract.PetEntry.GENDER_MALE; // Male
        } else if (selection.equals("Female")) {
            mGender = PetContract.PetEntry.GENDER_FEMALE; // Female
        } else {
            mGender = PetContract.PetEntry.GENDER_UNKNOWN; // Unknown
        }
    }
}
```

```
import com.example.android.pets.data.PetContract.PetEntry;

public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    String selection = (String) parent.getItemAtPosition(position);
    if (!TextUtils.isEmpty(selection)) {
        if (selection.equals("Male")) {
            mGender = PetEntry.GENDER_MALE; // Male
        } else if (selection.equals("Female")) {
            mGender = PetEntry.GENDER_FEMALE; // Female
        } else {
            mGender = PetEntry.GENDER_UNKNOWN; // Unknown
        }
    }
}
```

02 - Add PetDbHelper class

```
CREATE TABLE pets(_id INTEGER PRIMARY KEY AUTOINCREMENT,
                  name TEXT NOT NULL,
                  breed TEXT,
                  gender INTEGER NOT NULL,
                  weight INTEGER NOT NULL DEFAULT 0
                  );
```

Modify CatalogActivity.java (after onCreate (displayDatabaseInfo())

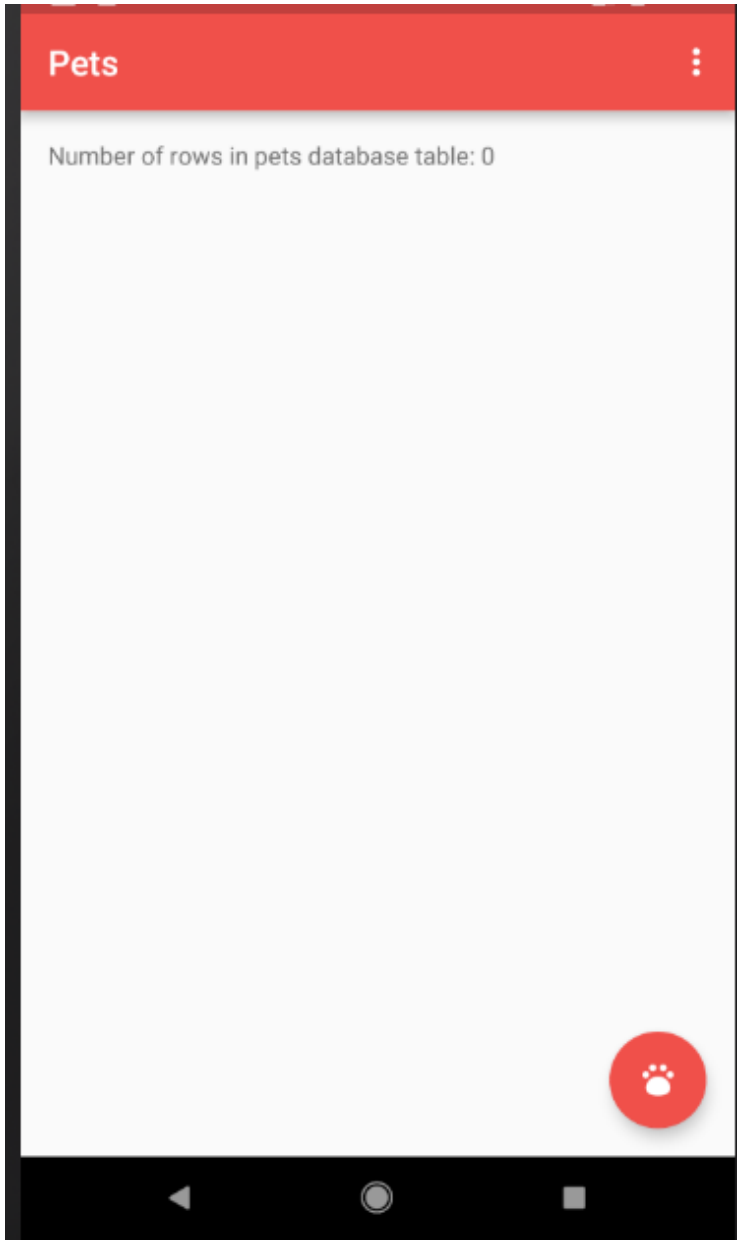
```
35          });
36
37          displayDatabaseInfo();
38      }
39
40      /**
41       * Temporary helper method to display information in the onscreen TextView about the state of
42       * the pets database.
43       */
44      private void displayDatabaseInfo() {
45          // To access our database, we instantiate our subclass of SQLiteOpenHelper
46          // and pass the context, which is the current activity.
47          PetDbHelper mDbHelper = new PetDbHelper(this);
48
49          // Create and/or open a database to read from it
50          SQLiteDatabase db = mDbHelper.getReadableDatabase();
51
52          // Perform this raw SQL query "SELECT * FROM pets"
53          // to get a Cursor that contains all rows from the pets table.
54          Cursor cursor = db.rawQuery("SELECT * FROM " + PetEntry.TABLE_NAME, null);
```

```
55          try {
56              // Display the number of rows in the Cursor (which reflects the number of rows in the
57              // pets table in the database).
58              TextView displayView = (TextView) findViewById(R.id.text_view_pet);
59              displayView.setText("Number of rows in pets database table: " + cursor.getCount());
60          } finally {
61              // Always close the cursor when you're done reading from it. This releases all its
62              // resources and makes it invalid.
63              cursor.close();
64          }
65      }
```

PetDbHelper.java

```java
13  public class PetDbHelper extends SQLiteOpenHelper {
14      private static final String DATABASE_NAME = "shelter.db";
15      private static final int DATABASE_VERSION = 1;
16
17      public PetDbHelper(Context context) {
18          super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
19      }
20
21      @Override
22      public void onCreate(SQLiteDatabase db) {
23          // Create a String that contains the SQL statement to create the pets table
24
25          String SQL_CREATE_PETS_TABLE =  "CREATE TABLE " + PetEntry.TABLE_NAME + " ("
26                  + PetEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
27                  + PetEntry.COLUMN_PET_NAME + " TEXT NOT NULL, "
28                  + PetEntry.COLUMN_PET_BREED + " TEXT, "
29                  + PetEntry.COLUMN_PET_GENDER + " INTEGER NOT NULL, "
30                  + PetEntry.COLUMN_PET_WEIGHT + " INTEGER NOT NULL DEFAULT 0);";
31          // Execute the SQL statement
32          db.execSQL(SQL_CREATE_PETS_TABLE);
33      }
34
35      @Override
36      public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
37
38      }
39  }
```

Run it



03 - Insert specific pet from CatalogActivity menu option

Modify CatalogActivity.java

```
private PetDbHelper mDbHelper;
```

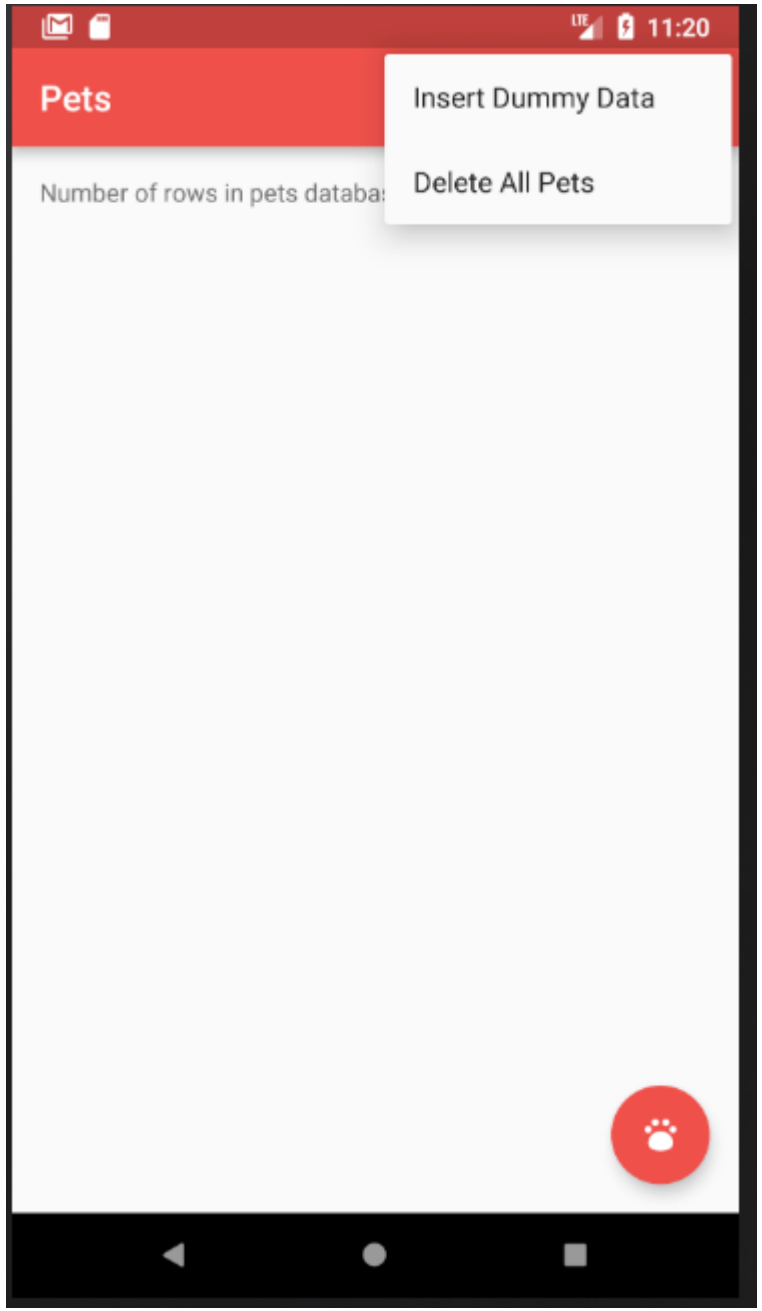
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_catalog);

    // Setup FAB to open EditorActivity
    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener((view) → {
            Intent intent = new Intent( packageContext: CatalogActivity.this, EditorActivity.class
            startActivity(intent);
    });

    mDbHelper = new PetDbHelper( context: this);
    displayDatabaseInfo();
}
```

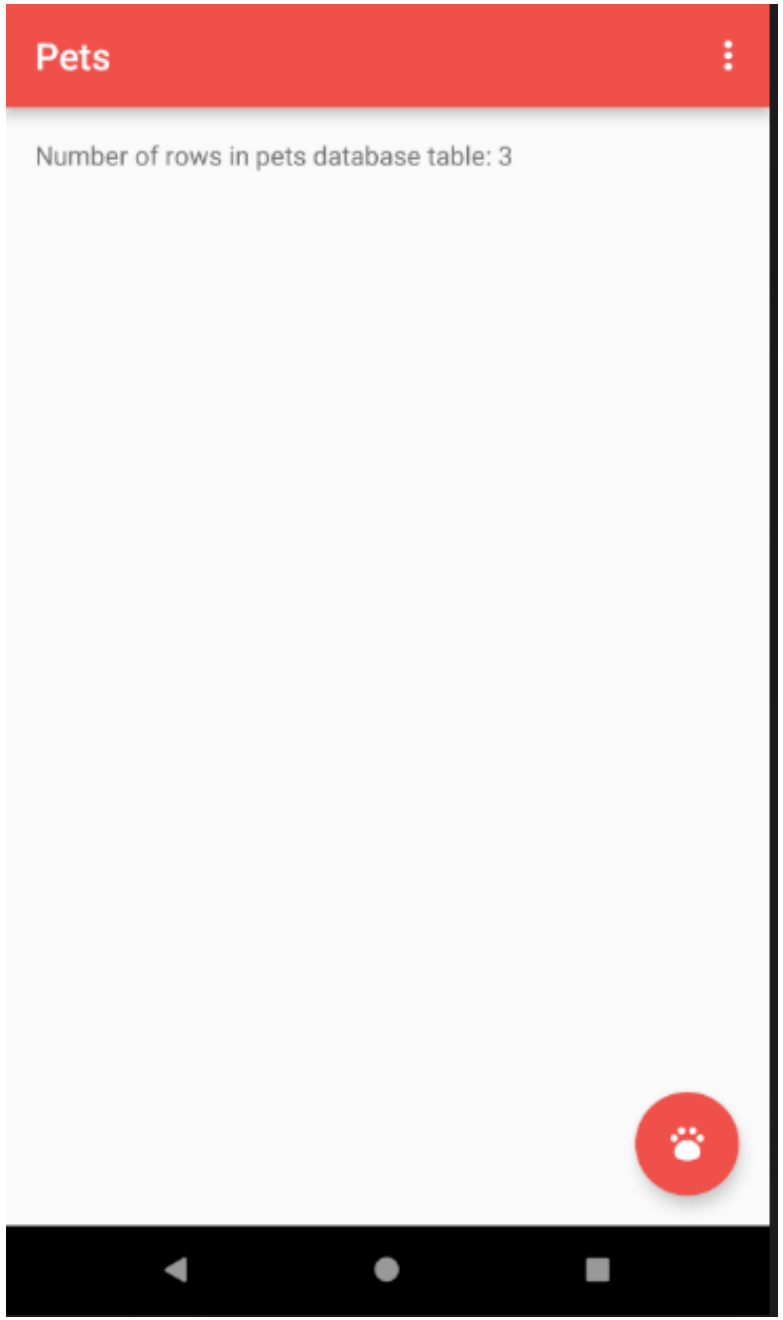
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // User clicked on a menu option in the app bar overflow menu
    switch (item.getItemId()) {
        // Respond to a click on the "Insert dummy data" menu option
        case R.id.action_insert_dummy_data:
            insertPet();
            displayDatabaseInfo();
```

Put on top of onCreateOptionMenu

```
private void insertPet(){
    //Gets the data respository in write mode
    SQLiteDatabase db = mDbHelper.getWritableDatabase();

    ContentValues values = new ContentValues();

    values.put(PetEntry.COLUMN_PET_NAME, "Toto");
    values.put(PetEntry.COLUMN_PET_BREED, "Terrier");
    values.put(PetEntry.COLUMN_PET_GENDER, PetEntry.GENDER_MALE);
    values.put(PetEntry.COLUMN_PET_WEIGHT, 7);

    long newRowId = db.insert(PetEntry.TABLE_NAME, null, values);

    Log.v("CatalogActivity", "New row ID " + newRowId);

}
```

RUN APP


11:20
Pets
Insert Dummy Data
Delete All Pets
Number of rows in pets databa


Pets
Number of rows in pets database table: 3

04 - Insert pet from EditorActivity

EditorActivity.java

On top of onCreateOptionsMenu

```
116     private void insertPet() {
117         // Read from input fields
118         // Use trim to eliminate leading or trailing white space
119         String nameString = mNameEditText.getText().toString().trim();
120         String breedString = mBreedEditText.getText().toString().trim();
121         String weightString = mWeightEditText.getText().toString().trim();
122         int weight = Integer.parseInt(weightString);
123
124         // Create database helper
125         PetDbHelper mDbHelper = new PetDbHelper(this);
126
127         // Gets the database in write mode
128         SQLiteDatabase db = mDbHelper.getWritableDatabase();
129
```

EditorActivity.java

```
130         // Create a ContentValues object where column names are the keys,
131         // and pet attributes from the editor are the values.
132         ContentValues values = new ContentValues();
133         values.put(PetEntry.COLUMN_PET_NAME, nameString);
134         values.put(PetEntry.COLUMN_PET_BREED, breedString);
135         values.put(PetEntry.COLUMN_PET_GENDER, mGender);
136         values.put(PetEntry.COLUMN_PET_WEIGHT, weight);
137
138         // Insert a new row for pet in the database, returning the ID of that new row.
139         long newRowId = db.insert(PetEntry.TABLE_NAME, null, values);
140
141         // Show a toast message depending on whether or not the insertion was successful
142         if (newRowId == -1) {
143             // If the row ID is -1, then there was an error with insertion.
144             Toast.makeText(this, "Error with saving pet", Toast.LENGTH_SHORT).show();
145         } else {
146             // Otherwise, the insertion was successful and we can display a toast with the row ID.
147             Toast.makeText(this, "Pet saved with row id: " + newRowId, Toast.LENGTH_SHORT).show();
148         }
149     }
```

EditorActivity.java

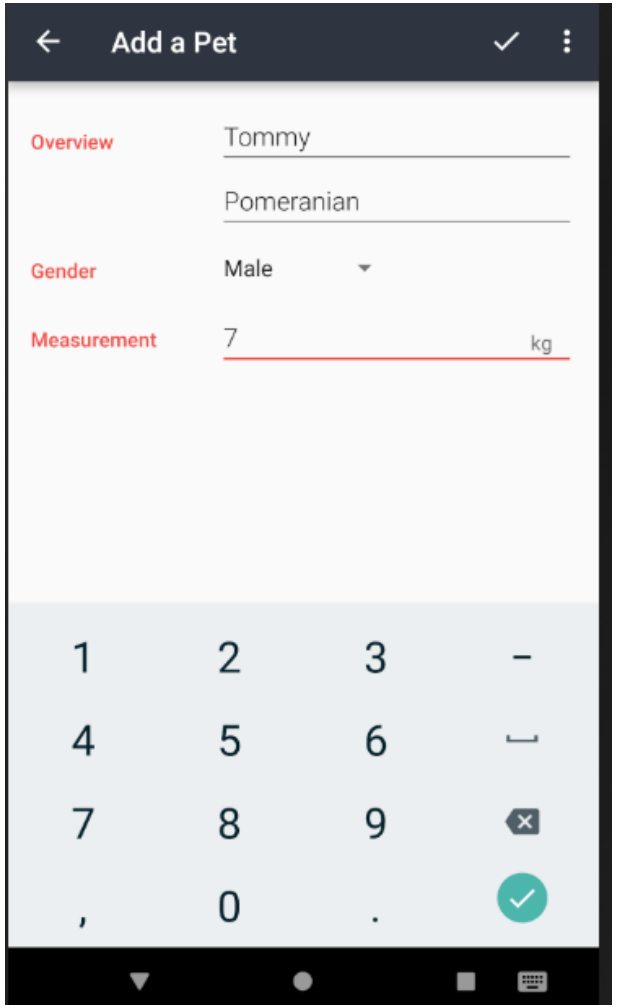
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // User clicked on a menu option in the app bar overflow menu
    switch (item.getItemId()) {
        // Respond to a click on the "Save" menu option
        case R.id.action_save:
            // Save pet to database
            insertPet();
            // Exit acitivity
            finish();
            return true;
```

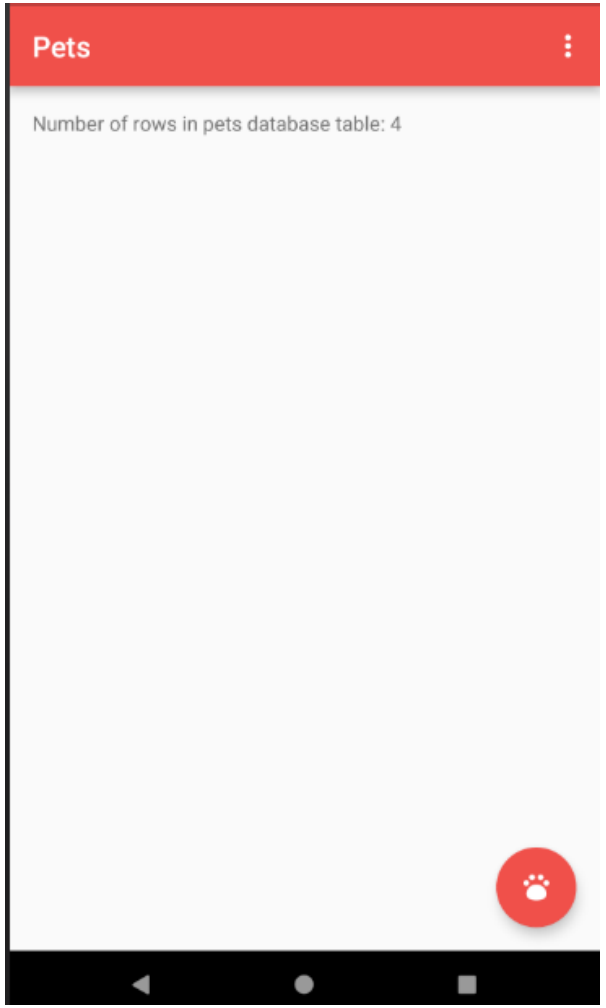
CatalogActivity.java

On top of displayDatabaseInfo()

```
@Override
protected void onStart() {
    super.onStart();
    displayDatabaseInfo();
}
```

Run APP and add the pet





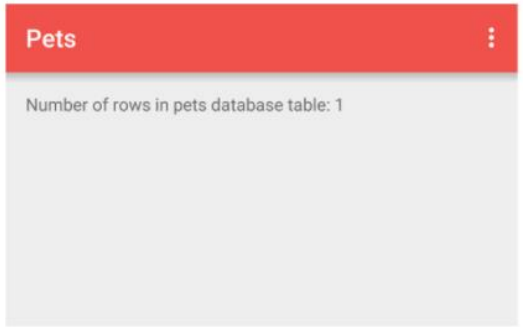
05 - Use SQLiteDatabase query method in CatalogActivity

- CatalogActivity.java
- In displayDatabaseInfo()

USE DATABASE QUERY METHOD

☐ Modify CatalogActivity.java:

Replace the SQLiteDatabase rawQuery() line with a call to the query() method instead. If needed, specify a projection, selection, selection arguments, etc...



The UI should stay the same as before. Number of rows in your database table may differ, and that's okay.

CatalogActivity.java

```
private void displayDatabaseInfo() {
    // Create and/or open a database to read from it
    SQLiteDatabase db = mDbHelper.getReadableDatabase();


    String[] projection = {
            PetEntry._ID,
            PetEntry.COLUMN_PET_NAME,
            PetEntry.COLUMN_PET_BREED,
            PetEntry.COLUMN_PET_GENDER,
            PetEntry.COLUMN_PET_WEIGHT
    };

    Cursor cursor = db.query(
            PetEntry.TABLE_NAME,
            projection,
            null,
            null,
            null,
            null,
            null);


    try {
        // Display the number of rows in the Cursor (which reflects the number of rows in the
        // pets table in the database).
        TextView displayView = (TextView) findViewById(R.id.text_view_pet);
        displayView.setText("Number of rows in pets database table: " + cursor.getCount());
    } finally {
```

06 - Read pet attributes from Cursor

Modify Catalog Activity.java (displayDatabaseInfo()

```
71      private void displayDatabaseInfo() {
72          // Create and/or open a database to read from it
73          SQLiteDatabase db = mDbHelper.getReadableDatabase();
74
75          // Define a projection that specifies which columns from the database
76          // you will actually use after this query.
77          String[] projection = {
78                  PetEntry._ID,
79                  PetEntry.COLUMN_PET_NAME,
80                  PetEntry.COLUMN_PET_BREED,
81                  PetEntry.COLUMN_PET_GENDER,
82                  PetEntry.COLUMN_PET_WEIGHT };
83
84          // Perform a query on the pets table
85          Cursor cursor = db.query(
86                  PetEntry.TABLE_NAME,   // The table to query
87                  projection,            // The columns to return
88                  null,                  // The columns for the WHERE clause
89                  null,                  // The values for the WHERE clause
90                  null,                  // Don't group the rows
91                  null,                  // Don't filter by row groups
92                  null);                   // The sort order
```

```
94          TextView displayView = (TextView) findViewById(R.id.text_view_pet);
95
96          try {
97              // Create a header in the Text View that looks like this:
98              //
99              // The pets table contains <number of rows in Cursor> pets.
100             // _id - name - breed - gender - weight
101             //
102             // In the while loop below, iterate through the rows of the cursor and display
103             // the information from each column in this order.
104             displayView.setText("The pets table contains " + cursor.getCount() + " pets.\n\n");
105             displayView.append(PetEntry._ID + " - " +
106                     PetEntry.COLUMN_PET_NAME + " - " +
107                     PetEntry.COLUMN_PET_BREED + " - " +
108                     PetEntry.COLUMN_PET_GENDER + " - " +
109                     PetEntry.COLUMN_PET_WEIGHT + "\n");
110
111             // Figure out the index of each column
112             int idColumnIndex = cursor.getColumnIndex(PetEntry._ID);
113             int nameColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_NAME);
114             int breedColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_BREED);
115             int genderColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_GENDER);
116             int weightColumnIndex = cursor.getColumnIndex(PetEntry.COLUMN_PET_WEIGHT);
```

```
118             // Iterate through all the returned rows in the cursor
119             while (cursor.moveToNext()) {
120                 // Use that index to extract the String or Int value of the word
121                 // at the current row the cursor is on.
122                 int currentID = cursor.getInt(idColumnIndex);
123                 String currentName = cursor.getString(nameColumnIndex);
124                 String currentBreed = cursor.getString(breedColumnIndex);
125                 int currentGender = cursor.getInt(genderColumnIndex);
126                 int currentWeight = cursor.getInt(weightColumnIndex);
127                 // Display the values from each column of the current row in the cursor in the TextView
128                 displayView.append(("\n" + currentID + " - " +
129                         currentName + " - " +
130                         currentBreed + " - " +
131                         currentGender + " - " +
132                         currentWeight));
133             }
134         } finally {
135             // Always close the cursor when you're done reading from it. This releases all its
136             // resources and makes it invalid.
137             cursor.close();
138         }
139     }
140
```

Run App


Pets
The pets table contains 5 pets.
_id - name - breed - gender - weight
1 - Toto - Terrier - 1 - 7
2 - Toto - Terrier - 1 - 7
3 - Toto - Terrier - 1 - 7
4 - Tommy - Pomeronian - 1 - 9
5 - Lenny - Bombay - 2 - 6