**SQLite**

Create another class ad extend SQLiteOpenHelper

public class Brains extends SQLiteOpenHelper {  
 private Brains(Context context){ - context is getActivity() in any class  
 super(context, "name", null, 1); - creating database (context, name of db, factory, version)   
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 db.execSQL("create table test (id number, name text, email text)"); - creating table  
  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { - updating table  
 db.execSQL("drop table if exists test");  
 onCreate(db);  
 }  
}

**insert**

in main class with db create function add, which requires needed parameters and database, then:

ContentValues contentValues = new ContentValues(); - create this variable  
contentValues.put("row", value); - add parameters  
database.insert("table", null, contentValues); - using got db, you can insert info in it

In another class set, where you want to use method of creation in listeners:

Brains brains = new Brains(getActivity()); - brain is example of name for class with db  
SQLiteDatabase database = brains.getWritableDatabase(); - get db

brains.addContact(Integer.parseInt(Id),Name,Email,database); - use method from that class (last value must be db)  
brains.close(); - close

**View**

public Cursor view(SQLiteDatabase database){  
 String[] required = {"id", "name", "email"};  
 Cursor cursor = database.query("table",required,null,null,null,null,order by);  
 return cursor;  
} – returns info in Cursor example

in another class

Brains brains = new Brains(getActivity()); - class with database communications   
SQLiteDatabase database = brains.getReadableDatabase();

Cursor cursor = brains.getContacts(database);

while (cursor.moveToNext()){  
 cursor.getInt(cursor.getColumnIndex("id")); - getting integer  
 cursor.getString(cursor.getColumnIndex("name")); - getting string  
}

**Update**

public void update(int id, String name, SQLiteDatabase database){   
ContentValues contentValues = new ContentValues();  
contentValues.put("name", name);  
database.update("table", contentValues, "id = " + id, null); - update (table, variables, where id = …, else where(set null))  
}

**Delete**

public void delete(int id, SQLiteDatabase database){  
 database.delete("test", "id = " + id, null);  
}

Room database

**Adding to the project**

In file Gradle Scripts in file bild.grade(module.app) add to the dependencies:

implementation "android.arch.persistence.room:runtime:1.1.1"  
annotationProcessor "android.arch.persistence.room:compiler:1.1.1"

also check bild.grade(Project:first) it has to have function google()

then create class of the table (its name will be a table name) for example user

@Entity(tableName = "users") – rename table  
public class User {  
  
 @PrimaryKey  
 private int id; - create id and annotate it as a primary key  
  
 @ColumnInfo(name = "name")  
 private String name; - create another field and change its name if you want to with annotate  
  
}

Then create interface with methods of your database and annotate each of them:

@DAO

public interface MyDao {  
 @Insert - tells that it is insert method   
 public void addUser(User user);

@Query("select \* from users") - query method   
 public List<User> getUsers();

@Delete - delete  
 public void deleteUser(User user);

}

and finally create database class:

@Database(entities = {User.class}, version = 1) – set tables (created classes) and version  
public abstract class MyAppDatabase extends RoomDatabase {   
 public abstract MyDao myDao(); - create method (dao – data access object)  
}

Add field

In main activitie create example of database(MyAppDatabase in example) and initialize it:

database = Room.databaseBuilder(getApplicationContext(), MyAppDatabase.class, "name of db").allowMainThreadQueries().build(); - create database and save it, allow function is to allow using in main thread

then in fragment, where you add field create example of class, which is a table and initialize its variables, than record

User user = new User();  
user.setId(Id);  
MainActivity.database.myDao().addUser(user); - function addUser will must be created in interface

View fields

In interface add public method, which returns list of your table class and create annotation on it @Query("select \* from users"), then in fragment:

List<User> list = MainActivity.database.myDao().getUsers();  
String res = "";  
for (User user : list){  
 res += user.getId() + ": " + user.getName() + " - " + user.getEmail() + "\n";  
}  
textView.setText(res);

Delete

In interface add public void method, which requires example of your class table and create annotation on it @Delete , then in fragment:

user.setId(Integer.parseInt(id.getText().toString()));  
MainActivity.database.myDao().deleteUser(user); - delete user by primary key

Update

As well as other, create, but on the method in interface create annotation @Update and use it by primary key