**Updating Column Size in Room (Android) Using Kotlin:**

1. **Modify the Entity Class:**

Update the entity class to reflect the new column size. For instance, if the `User` table has a `name` column, update its definition in the `User` class.

**Kotlin:**

@Entity(tableName = "users")

data class User(

@PrimaryKey(autoGenerate = true) val id: Int,

@ColumnInfo(name = "name") val name: String

// Other fields...

)

1. **Create a Migration:**

- \*\*Create a New Table\*\*: Define a new table with the desired column size.

- \*\*Copy Data\*\*: Move data from the old table to the new one.

- \*\*Drop the Old Table\*\*: Remove the old table from the database.

- \*\*Rename the New Table\*\*: Rename the new table to match the original table's name.

**Example:**

Kotlin:

val MIGRATION\_1\_2 = object : Migration(1, 2) {

override fun migrate(database: SupportSQLiteDatabase) {

// Create a new table with the updated schema

database.execSQL("""

CREATE TABLE users\_new (

id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

name TEXT

)

""".trimIndent())

// Copy data from the old table to the new table

database.execSQL("INSERT INTO users\_new (id, name) SELECT id, name FROM users")

// Drop the old table

database.execSQL("DROP TABLE users")

// Rename the new table to the original table name

database.execSQL("ALTER TABLE users\_new RENAME TO users")

}

}

```

1. **\*\*Update Database Version\*\*:**

Increment the database version in your Room database class and register the migration.

Example:

```kotlin

@Database(entities = [User::class], version = 2)

abstract class AppDatabase : RoomDatabase() {

abstract fun userDao(): UserDao

companion object {

@Volatile private var INSTANCE: AppDatabase? = null

fun getDatabase(context: Context): AppDatabase {

return INSTANCE ?: synchronized(this) {

val instance = Room.databaseBuilder(

context.applicationContext,

AppDatabase::class.java,

"user\_database"

).addMigrations(MIGRATION\_1\_2)

.build()

INSTANCE = instance

instance

}

}

}

}

```