# Database Helper class

```
class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME,
null, 1) {
  companion object {
    const val DATABASE_NAME = "RestaurantDB.db"
  }

  override fun onCreate(db: SQLiteDatabase?) {
    db!!.execSQL("CREATE TABLE tblUser(UserID INTEGER PRIMARY KEY
AUTOINCREMENT,UNAME TEXT, PASSWORD TEXT)")
    with(db) {
      execSQL("CREATE TABLE tblAdmin(AdminID INTEGER PRIMARY KEY
AUTOINCREMENT,UNAME TEXT, PASSWORD TEXT)")
      execSQL("CREATE TABLE tblCategory(CategoryID INTEGER PRIMARY KEY
AUTOINCREMENT,CATEGORY TEXT)")
      execSQL(
        "CREATE TABLE tblProduct(ProductID INTEGER PRIMARY KEY
AUTOINCREMENT,ProductName TEXT,PRICE INTEGER,CategoryName TEXT,DESCRIPTION
TEXT,image BLOB," +
            "FOREIGN KEY(CategoryName) REFERENCES tblCategory(CATEGORY))"
      )
      execSQL("CREATE TABLE tblOrder(OrderID INTEGER PRIMARY KEY
AUTOINCREMENT,ProductName TEXT,QUANTITY INTEGER,TableNumber TEXT,TotalPrice
TEXT)")
    }

    val contentValues = ContentValues()
    contentValues.put("UNAME", "dev")
    contentValues.put("PASSWORD", "dev123")
    db.insert("tblUser", null, contentValues)

    val contentValue = ContentValues()
    contentValue.put("UNAME", "admin")
    contentValue.put("PASSWORD", "admin123")
    db.insert("tblAdmin", null, contentValue)
  }

  val seeOrder: Cursor
    get() {
      val db = this.writableDatabase
      return db.rawQuery("SELECT * FROM tblOrder", null)
    }
```

```
fun order(pname: String, qty: String, tblNo: String, totPrice: String) {
    val db = this.writableDatabase
    val contentValues = ContentValues()
    contentValues.put("ProductName", pname)
    contentValues.put("QUANTITY", qty)
    contentValues.put("TableNumber", tblNo)
    contentValues.put("TotalPrice", totPrice)
    db.insert("tblOrder", null, contentValues)
}

fun updItem(name: String, id: String, price: String, desc: String) {
    val db = this.writableDatabase
    val contentValues = ContentValues()
    contentValues.put("ProductName", name)
    contentValues.put("PRICE", price)
    contentValues.put("DESCRIPTION", desc)
    db.update("tblProduct", contentValues, " ProductID = ?", arrayOf(id))
}

fun deleteItem(id: String) {
    val db = this.writableDatabase
    db.delete("tblProduct", "ProductID = ?", arrayOf(id))
}

fun showMenu(): List<menuModel> {
    val db = this.readableDatabase
    val placeList = mutableListOf<menuModel>()
    val cursor = db.rawQuery("select * from tblProduct", null)
    while (cursor.moveToNext()) {
        val id = cursor.getInt(cursor.getColumnIndexOrThrow("ProductID"))
        val photo = cursor.getBlob(cursor.getColumnIndexOrThrow("image"))
        val pname = cursor.getString(cursor.getColumnIndexOrThrow("ProductName"))
        val price = cursor.getString(cursor.getColumnIndexOrThrow("PRICE"))
        val desc = cursor.getString(cursor.getColumnIndexOrThrow("DESCRIPTION"))
        val data = menuModel(id, pname, photo, price, desc)
        placeList.add(data)
    }
    return placeList
}

fun buyNow(id: String): Cursor? {
    val db = this.readableDatabase
    val cursor = db.rawQuery("select * from tblProduct where ProductId = $id", null)
    return cursor
```

```
    }

    fun deleteOrder(id: String) {
        val db = this.writableDatabase
        db.delete("tblOrder", "OrderID = ?", arrayOf(id))
    }

    fun addMenu(pname: String, price: Int, cname: String, desc: String, image:
ByteArray?): Long {
        val db = this.writableDatabase
        val contentValues = ContentValues()
        contentValues.put("ProductName", pname)
        contentValues.put("PRICE", price)
        contentValues.put("CategoryName", cname)
        contentValues.put("DESCRIPTION", desc)
        contentValues.put("image", image)
        val res = db.insert("tblProduct", null, contentValues)
        return res
    }

    fun checkUser(user: String): Int {
        val db = this.writableDatabase
        val res = db.rawQuery("SELECT * FROM  tblUser where  UNAME='$user'", null)
        return res.count
    }

    fun addUser(uname: String, password: String): Long {
        val db = this.writableDatabase
        val contentValues = ContentValues()
        contentValues.put("UNAME", uname)
        contentValues.put("PASSWORD", password)
        return db.insert("tblUser", null, contentValues)
    }

    fun addCategory(cname: String): Long {
        val db = this.writableDatabase
        val contentValues = ContentValues()
        contentValues.put("CATEGORY", cname)
        return db.insert("tblCategory", null, contentValues)
    }

    fun deleteData(id: String) {
        val db = this.writableDatabase
        db.delete("tblUser", "UserID = ?", arrayOf(id))
```

```
    }

    fun deleteCat(id: String) {
        val db = this.writableDatabase
        db.delete("tblCategory", "CategoryID = ?", arrayOf(id))
    }

    fun login(uname: String, password: String): Int {
        val db = this.readableDatabase
        val res = db.rawQuery(
            "SELECT * FROM tblUser WHERE UNAME='" + uname + "' and PASSWORD = '" +
password + "'",
            null
        )
        return res.count
    }

    fun admin(uname: String, password: String): Int {
        val db = this.readableDatabase
        val res = db.rawQuery(
            "SELECT * FROM tblAdmin WHERE UNAME='" + uname + "' and PASSWORD = '" +
password + "'",
            null
        )
        return res.count
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
        db!!.execSQL("DROP TABLE IF EXISTS tblUser")
        onCreate(db)
    }

    val allData: Cursor
        get() {
            val db = this.writableDatabase
            val res = db.rawQuery("SELECT * FROM tblUser", null)
            return res
        }

    fun showCategory(): Cursor? {
        val db = this.writableDatabase
        return db.rawQuery("SELECT * FROM tblCategory", null)
    }
}
```