

DR. ALVIN'S PUBLICATIONS

LEARNING SQLITE

DR. ALVIN ANG



COPYRIGHTED BY DR ALVIN ANG
WWW.ALVINANG.SG

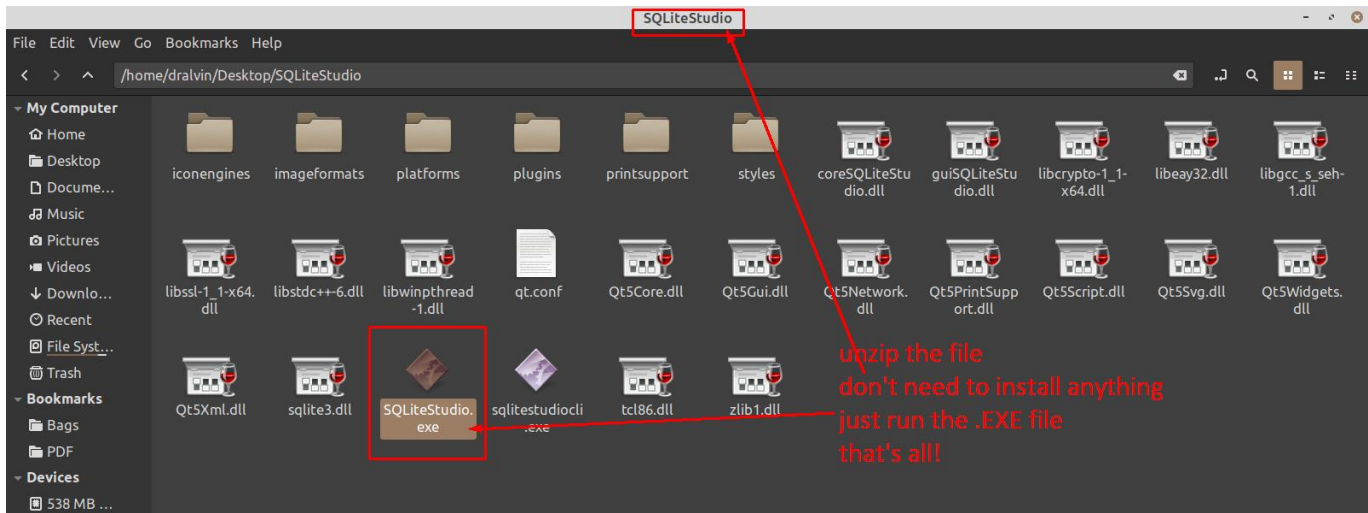
CONTENTS

Contents	2
I. Using SQLiteStudio	4
A. Select	11
B. Create (Table)	13
C. Drop (Table).....	14
D. Not Null	15
E. Alter	18
F. Default	19
G. Unique.....	21
H. Check.....	22
I. Relational vs Non Relational Database.....	24
J. Primary Key and Foreign Key	26
1. Jobs → Employees (One to Many Relationship)	27
2. Employees → Dependants (One to Many Relationship)	27
3. Employees → Employees (Zero or One to Many Relationship)	28
4. Creating Primary Key.....	29
5. Creating Foreign Key	30
K. Auto Increment.....	32
L. Create Index	35
M. Distinct	38
N. Where	40
1. Like	42
2. IS NULL	43
3. Exists	45
4. And.....	46
5. Or	47
6. Not	48
7. In	49
O. Order By	51
1. ACTIVITY QUESTION: ORDER BY / OPERATORS.....	54
2. ACTIVITY ANSWER: ORDER BY / OPERATORS	55
3. ACTIVITY QUESTION: ORDER BY / OPERATORS.....	56
4. ACTIVITY ANSWER: ORDER BY / OPERATORS	57
5. ACTIVITY QUESTION: ORDER BY / OPERATORS.....	58

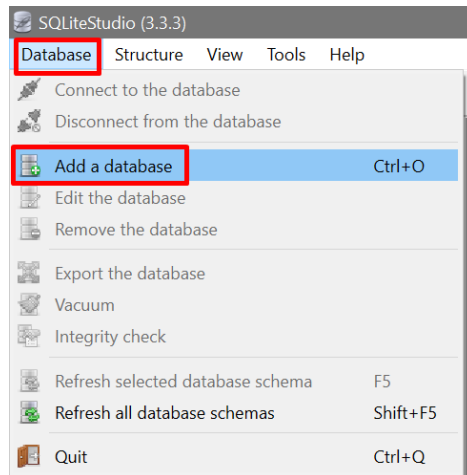
6.	ACTIVITY ANSWER: ORDER BY / OPERATORS	59
P.	Between	60
Q.	Limit	61
1.	ACTIVITY QUESTION: SELECT DATA / LIMIT	63
2.	ACTIVITY ANSWER: SELECT DATA / LIMIT	63
3.	ACTIVITY QUESTION: SELECT DATA / LIMIT	64
4.	ACTIVITY ANSWER: SELECT DATA / LIMIT	65
5.	Offset	66
R.	As	67
S.	Insert Into	68
T.	Update	70
U.	Delete	71
V.	Aggregate Functions	72
1.	Count	73
a)	ACTIVITY QUESTION: AGGREGATE FUNCTIONS	75
b)	ACTIVITY ANSWER: AGGREGATE FUNCTIONS	76
c)	ACTIVITY QUESTION: AGGREGATE FUNCTIONS	77
d)	ACTIVITY ANSWER: AGGREGATE FUNCTIONS	77
2.	SUM	78
3.	Average	79
4.	Max	80
5.	Min	81
6.	Combinations	82
W.	Group By	84
1.	Having	86
a)	ACTIVITY QUESTION: GROUPBY AND HAVING	88
b)	ACTIVITY ANSWER: GROUPBY AND HAVING	89
2.	Where	90
X.	Joins	93
1.	Inner Join	94
a)	Test.DB	94
b)	World.DB	96
(1)	ACTIVITY QUESTION: INNER JOIN	96
(2)	ACTIVITY ANSWER: INNER JOIN	97
(3)	ACTIVITY QUESTION: INNER JOIN	98
(4)	ACTIVITY ANSWER: INNER JOIN	99
About Dr. Alvin Ang		100

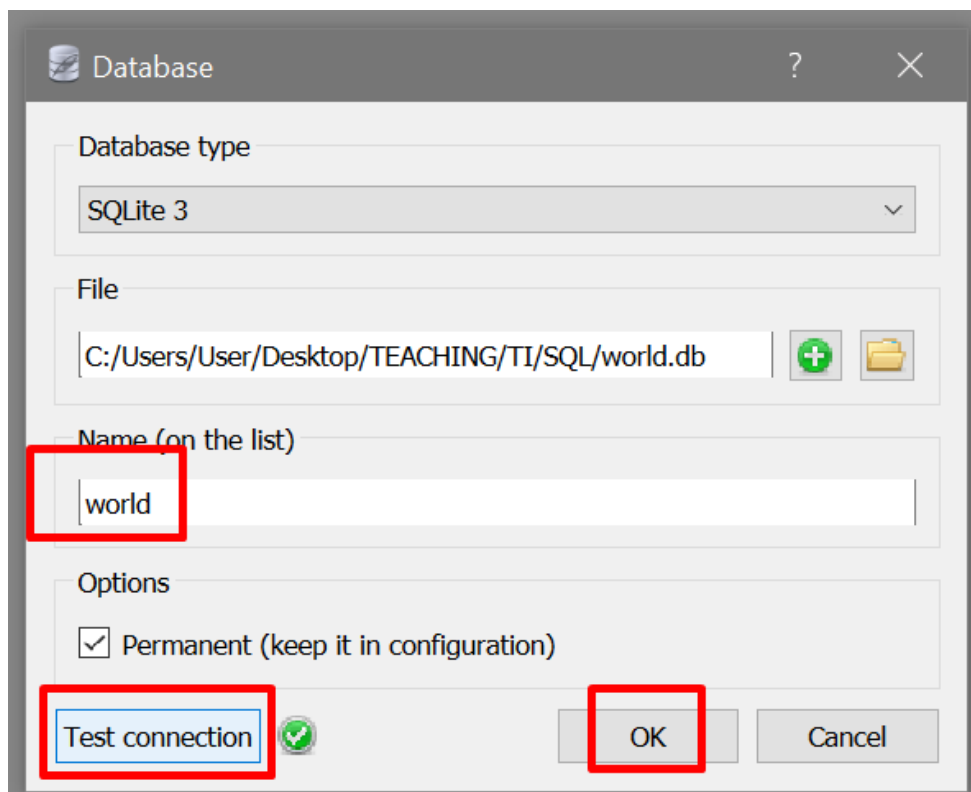
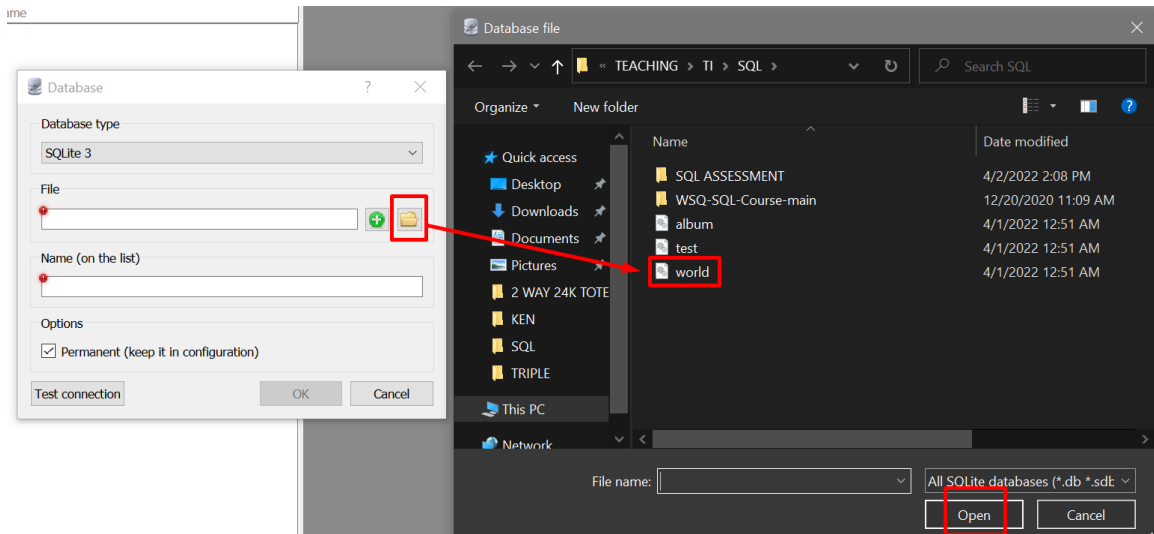
I. USING SQLITESTUDIO

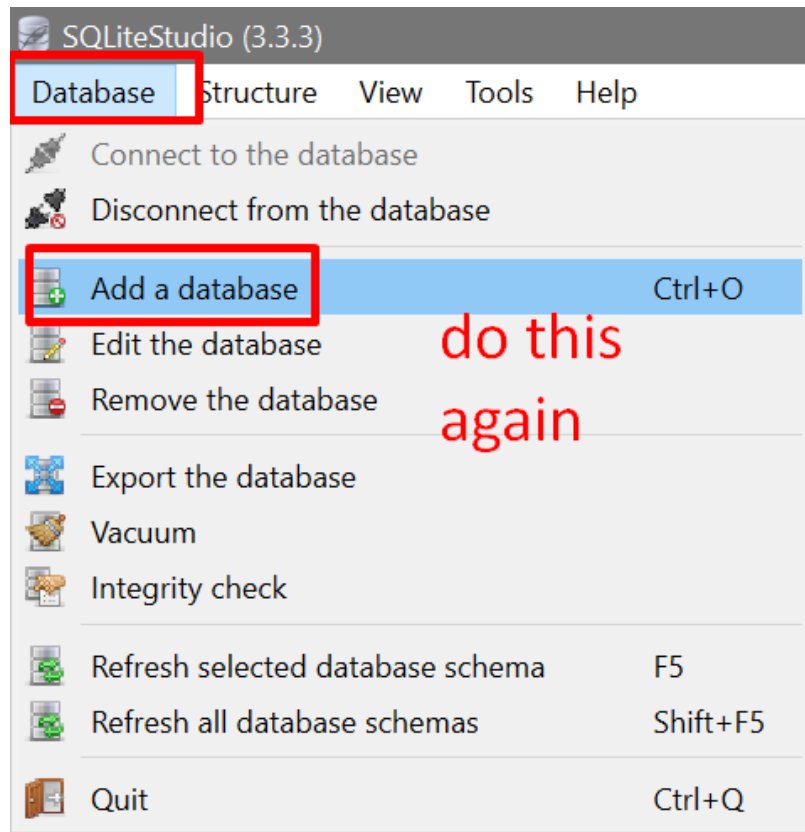
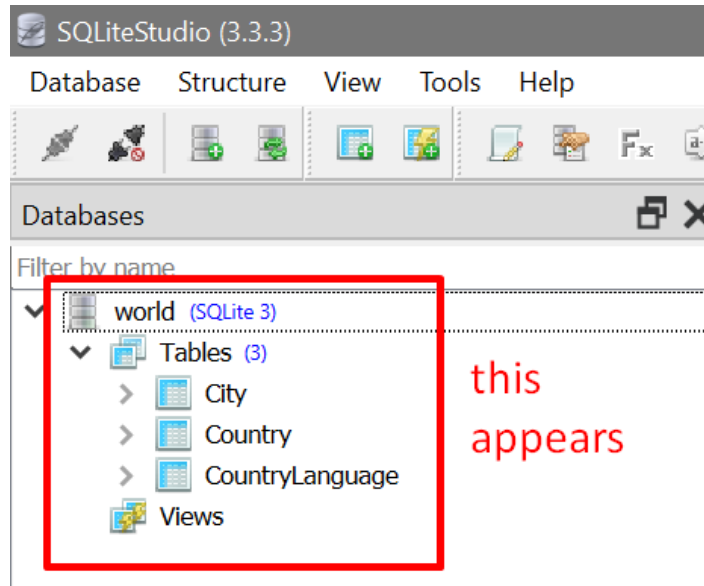
- Go here <https://sqlitestudio.pl/>

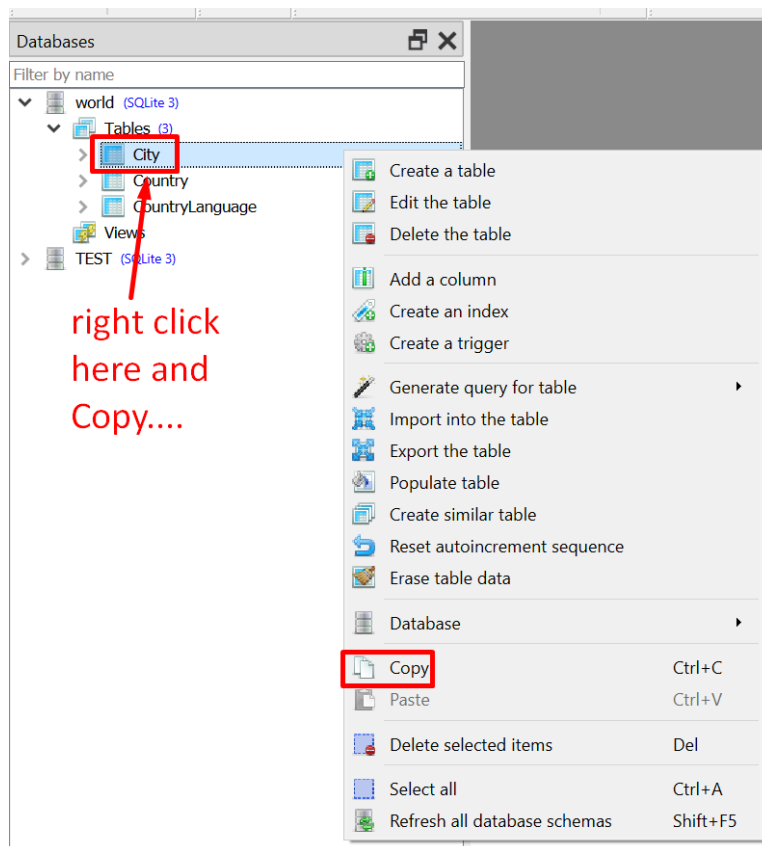
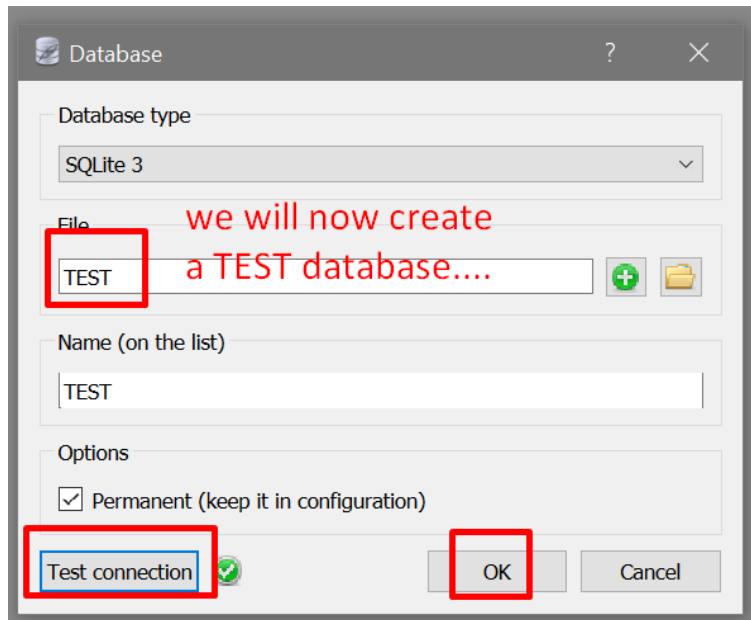


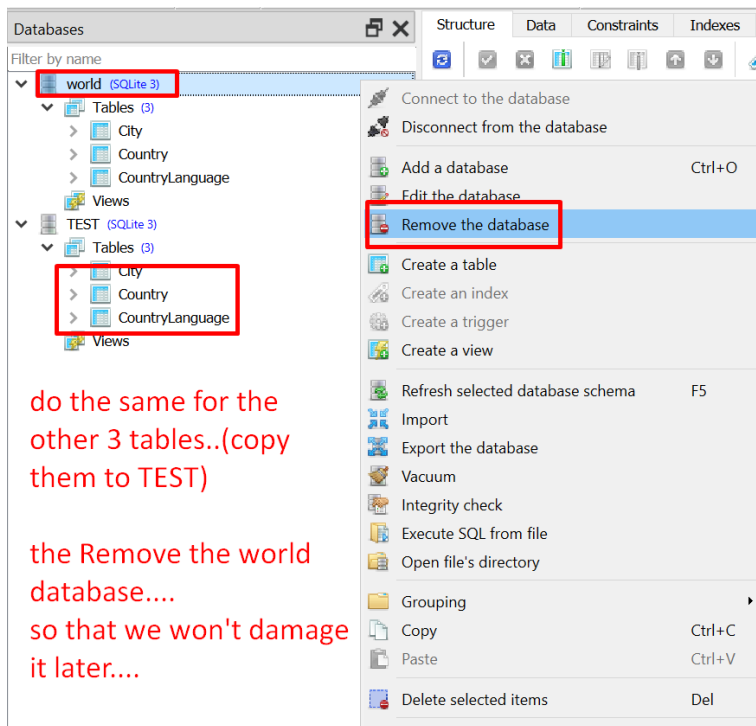
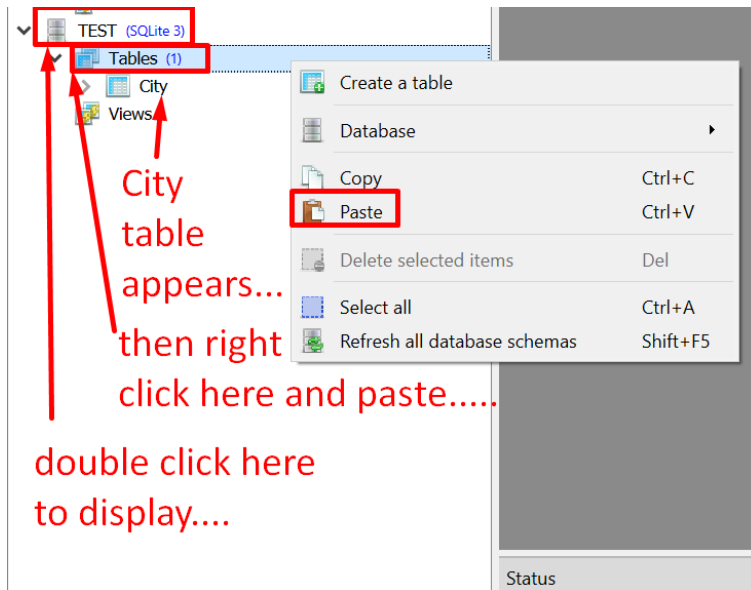
- Run the SQLite studio software.
- Go here <https://www.alvinang.sg/s/world.db> download the file.

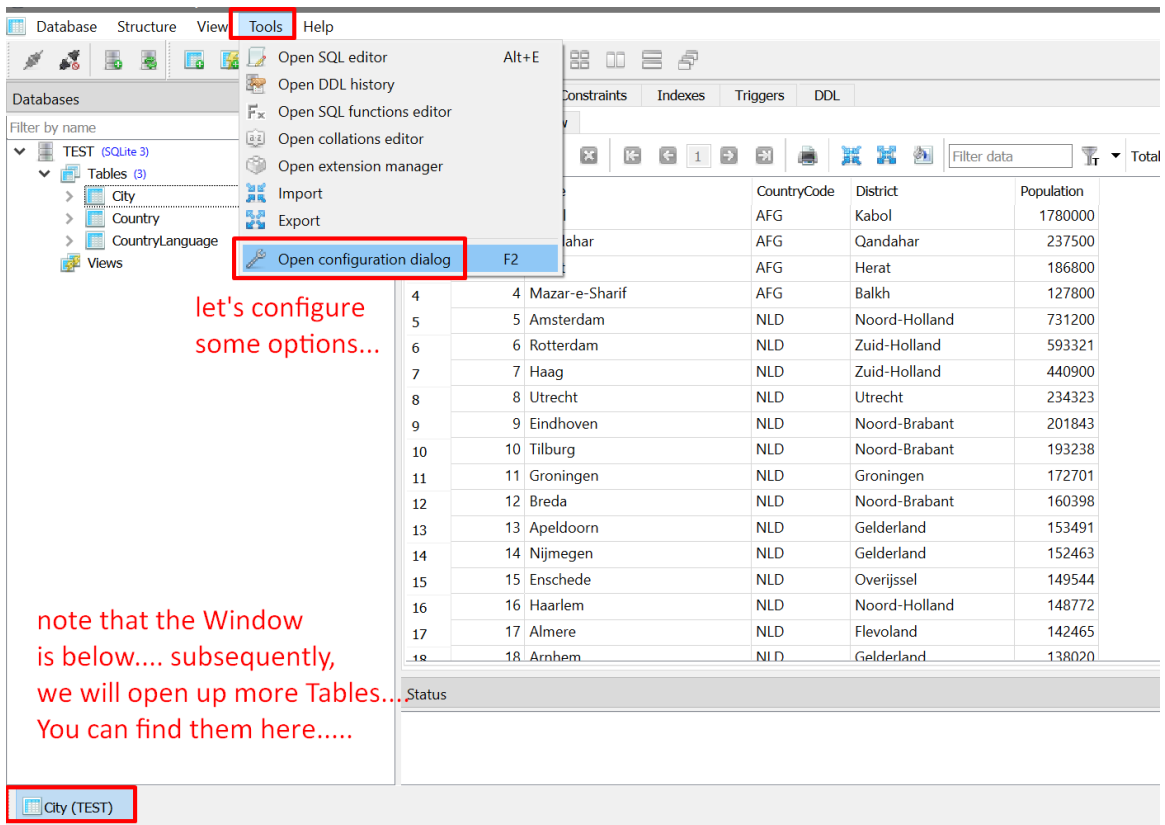
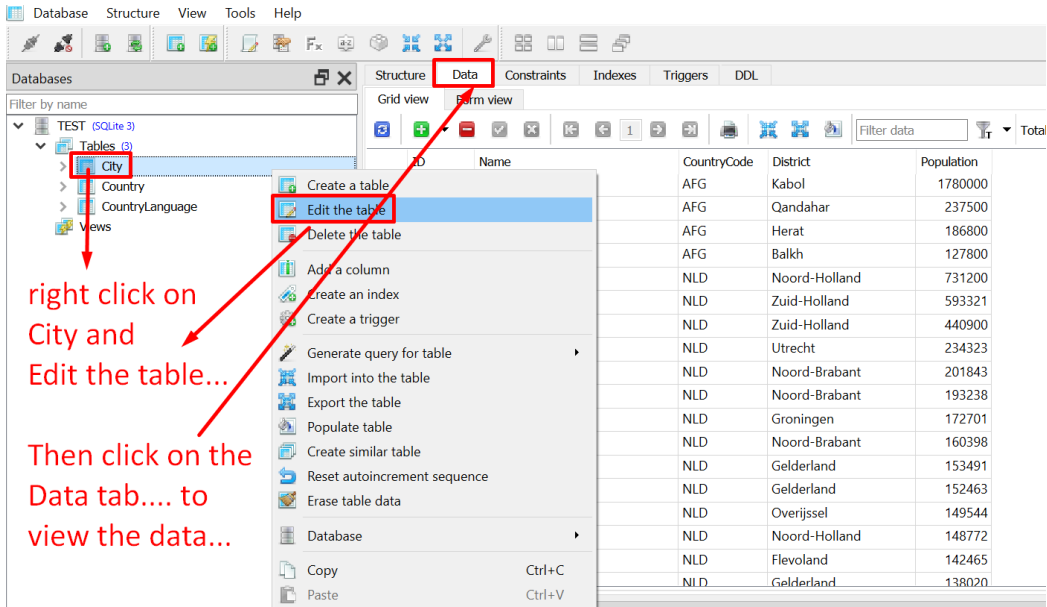


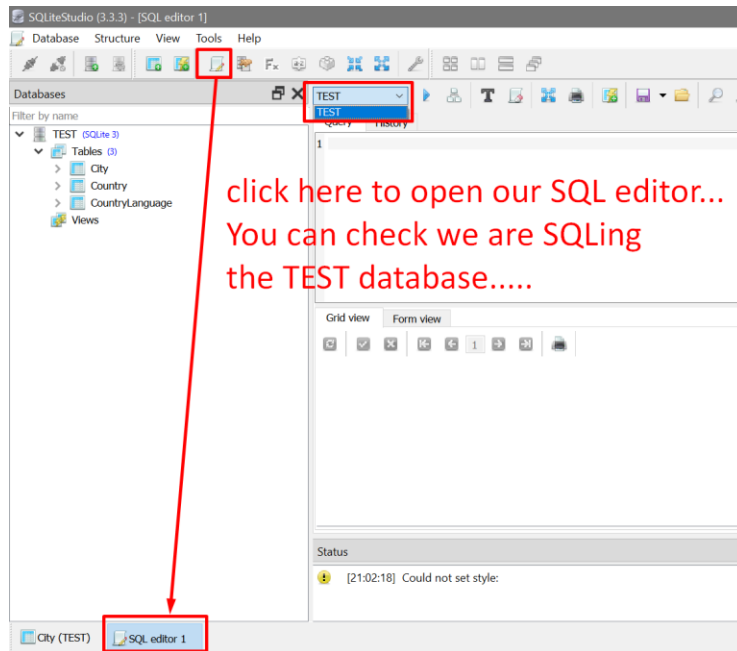
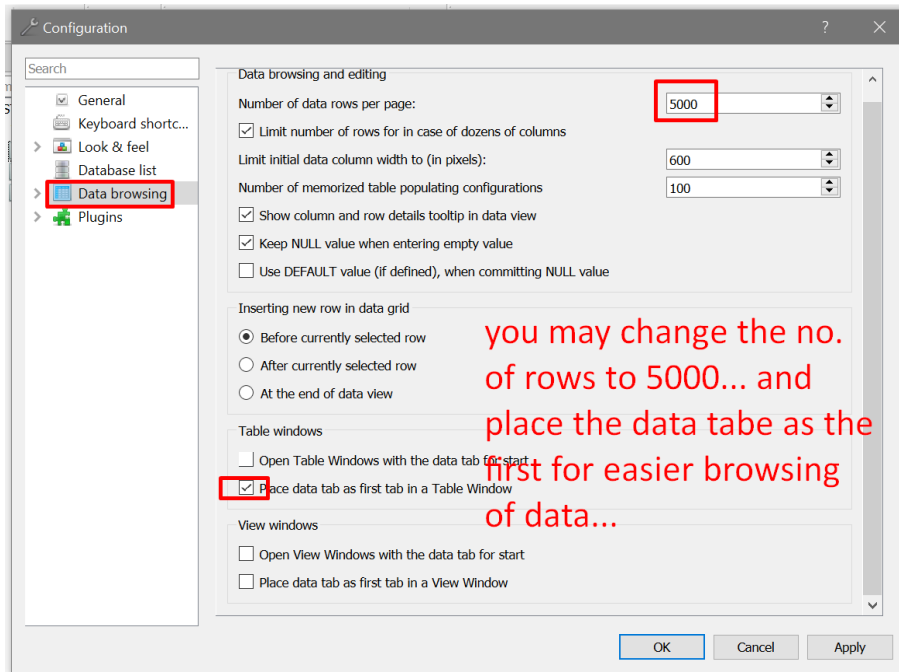












A. SELECT

our very first code displays everything in the City table....

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabul	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398

```
SELECT *  
FROM City
```

The screenshot shows the Microsoft Access interface. On the left, the 'Databases' pane shows the 'world' database with the 'Country' table selected. The 'Columns' list for 'Country' includes Name, Region, and GNP, which are highlighted with red boxes. The SQL editor on the right contains the following query:

```

1 SELECT Name,
2     Region,
3     GNP
4 FROM Country;
5
6

```

Below the query editor, the 'Grid view' shows the results of the query. The results are displayed in a table with 11 rows and 3 columns: Name, Region, and GNP. The data is as follows:

	Name	Region	GNP
1	Afghanistan	Southern and Central Asia	5976
2	Netherlands	Western Europe	371362
3	Netherlands Antilles	Caribbean	1941
4	Albania	Southern Europe	3205
5	Algeria	Northern Africa	49982
6	American Samoa	Polynesia	334
7	Andorra	Southern Europe	1630
8	Angola	Central Africa	6648
9	Anguilla	Caribbean	63.2
10	Antigua and Barbuda	Caribbean	612
11	United Arab Emirates	Middle East	37966

At the bottom of the SQL editor, the status bar shows the following messages:

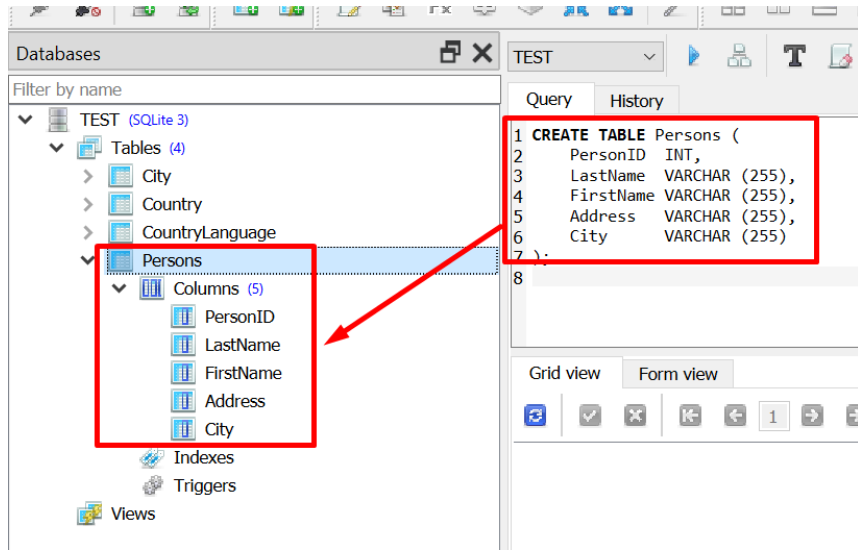
```

[22:12:22] Query finished in 0.000 second(s).
[22:13:09] Query finished in 0.000 second(s).
[22:17:20] Query finished in 0.006 second(s).

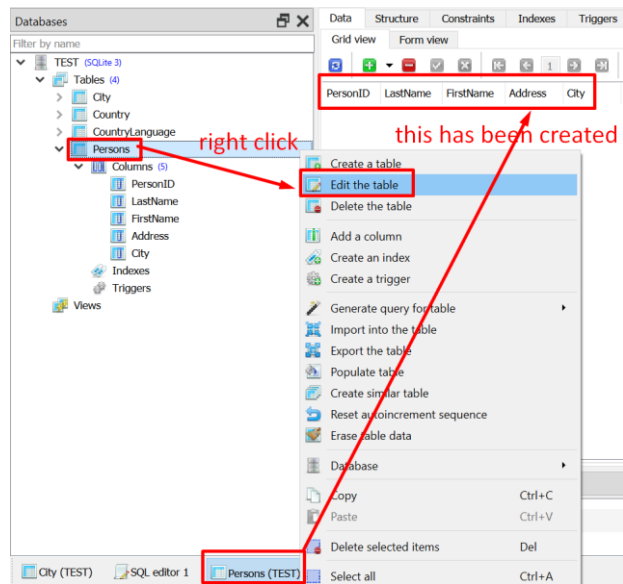
```

SELECT Name,
Region,
GNP
FROM Country;

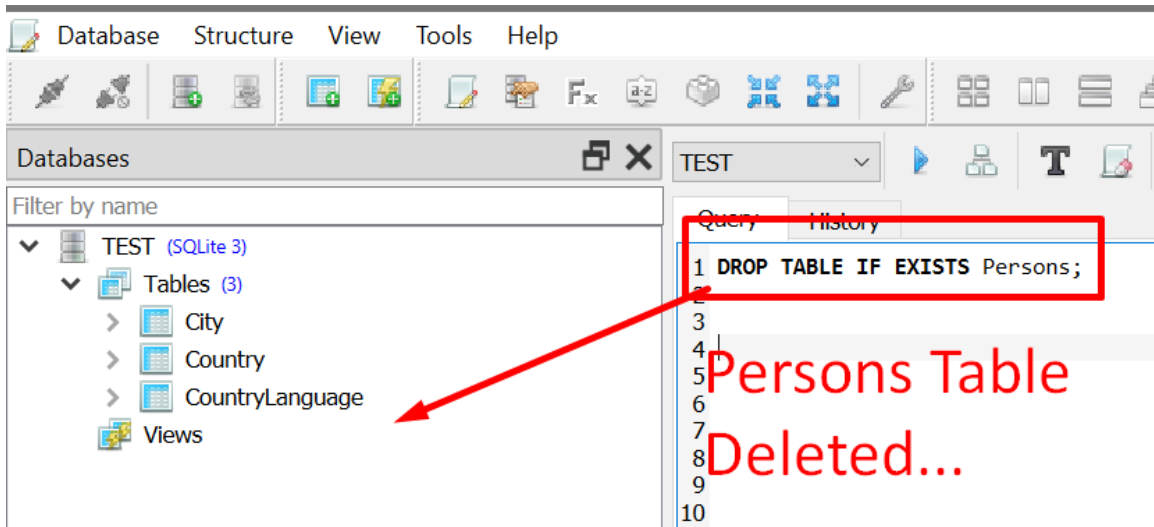
B. CREATE (TABLE)



```
CREATE TABLE Persons (  
  PersonID INT,  
  LastName VARCHAR (255),  
  FirstName VARCHAR (255),  
  Address VARCHAR (255),  
  City VARCHAR (255)  
);
```



C. DROP (TABLE)



DROP TABLE

IF EXISTS Persons;

D. NOT NULL

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'Databases' tree is expanded to show the 'TEST' database, then 'Tables (4)', and finally 'Persons'. The 'Columns (4)' folder under 'Persons' is highlighted with a red box. A red arrow points from this box to the 'Query' window on the right. The 'Query' window contains the following SQL code:

```
CREATE TABLE Persons (  
  ID INT NOT NULL,  
  LastName VARCHAR (255) NOT NULL,  
  FirstName VARCHAR (255),  
  Age INT  
);
```

Below the code, the 'Grid view' and 'Form view' tabs are visible, along with navigation buttons and a 'Total row' indicator.

we create
back a
Persons Table again...

```
CREATE TABLE Persons (  
  ID INT NOT NULL,  
  LastName VARCHAR (255) NOT NULL,  
  FirstName VARCHAR (255),  
  Age INT  
);
```

The screenshot shows the 'Structure' tab of the SQL Enterprise Manager interface. The 'Table name: Persons' is selected. The table structure is displayed in a table with the following columns:

Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated
1 ID	INT					☹		NULL
2 LastName	VARCHAR (255)					☹		NULL
3 FirstName	VARCHAR (255)							NULL
4 Age	INT							NULL

Red boxes highlight the 'Data type' column and the 'Not NULL' column. A red arrow points from the 'Data type' column to the 'Not NULL' column. Below the table, the 'Details' tab is visible, showing the 'Type' and 'Name' columns. The 'Status' section at the bottom shows the following messages:

```
[22:44:09] Query finished in 0.018 second(s).  
[22:45:48] Query finished in 0.010 second(s).
```

we see that the Persons
Table has been created...

Data Type and
Not NULL has been defined

when we go to the Data pane and create a new row

if we leave the row blank we see that error pops up

because of the NOT NULL constraint for ID and LastName

ID	LastName	FirstName	Age
1	NULL	NULL	NULL

Status

- [22:58:57] Error while committing new row: NOT NULL constraint failed: Persons.ID
- [22:59:00] Error while committing new row: NOT NULL constraint failed: Persons.ID
- [22:59:01] Error while committing new row: NOT NULL constraint failed: Persons.ID

if we input AA into ID we see that error still occurs

because LastName is still NULL which goes against the NOT NULL constraint...

ID	LastName	FirstName	Age
1	AA	NULL	NULL

Status

- [22:59:00] Error while committing new row: NOT NULL constraint failed: Persons.ID
- [22:59:01] Error while committing new row: NOT NULL constraint failed: Persons.ID
- [23:00:48] Error while committing new row: NOT NULL constraint failed: Persons.LastName

now when we input AA to ID
and 1`23 to LastName and click Commit..

it goes thru!
this is not supposed to be
because ID is supposed to be INT
and LastName supposed to be VARCHAR
--> SQLite has bugs to....

Status

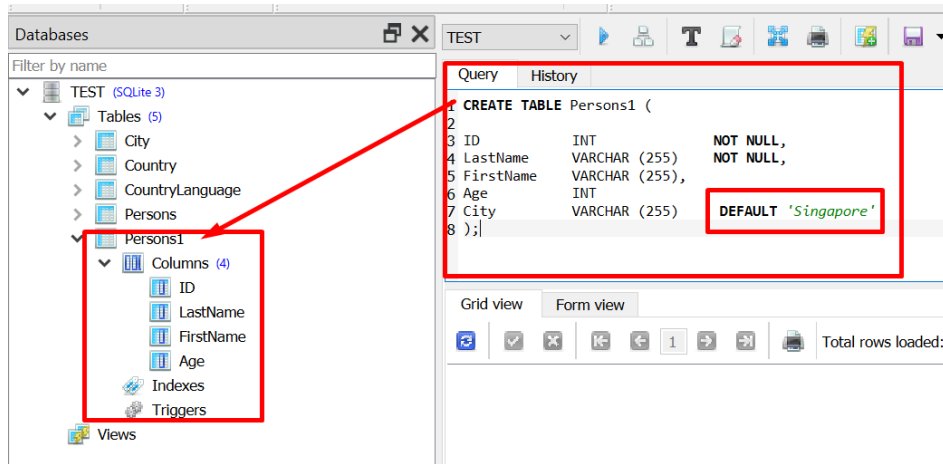
- [22:59:00] Error while committing new row: NOT NULL constraint failed: Persons.ID
- [22:59:01] Error while committing new row: NOT NULL constraint failed: Persons.ID
- [23:00:48] Error while committing new row: NOT NULL constraint failed: Persons.LastName

E. ALTER

The screenshot shows a database management interface. On the left, a tree view displays the database structure: TEST (SQLite 3) > Tables (4) > City, Country, CountryLanguage, Persons > Columns (5) > ID, LastName, FirstName, Age, Email. The 'Email' column is highlighted with a red box. A red arrow points from this box to the 'Email VARCHAR(255);' part of the SQL query in the main window. The query is: 1 ALTER TABLE Persons, 2 ADD COLUMN Email VARCHAR(255);. Below the query, red text explains: VARCHAR(255) means the column is of Variable Character Datatype that can accept 255 characters. The interface also shows 'Query' and 'History' tabs, and 'Grid view' and 'Form view' options.

```
ALTER TABLE Persons  
ADD COLUMN Email VARCHAR(255);
```

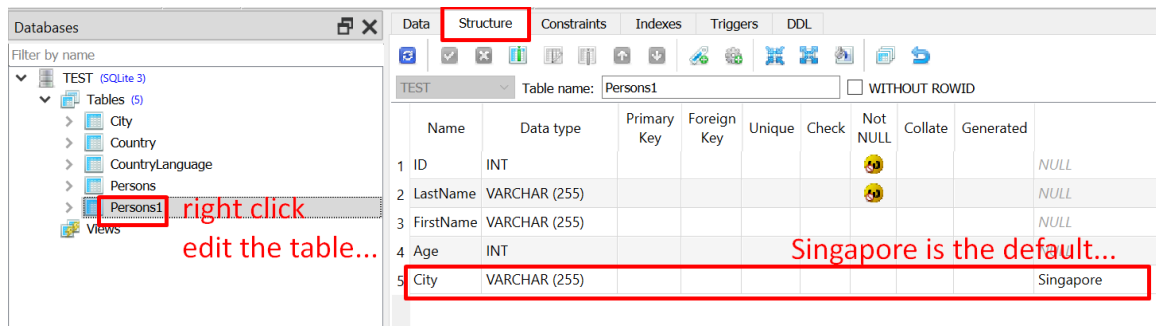
F. DEFAULT



CREATE TABLE Persons1 (

```

  ID INT NOT NULL,
  LastName VARCHAR (255) NOT NULL,
  FirstName VARCHAR (255),
  Age INT,
  City VARCHAR (255) DEFAULT 'Singapore'
);
  
```



Databases

TEST (SQLite 3)

- Tables (5)
 - City
 - Country
 - CountryLanguage
 - Persons
 - Persons1
 - Columns (5)
 - Indexes
 - Triggers
 - Views

Data | Structure | Constraints | Indexes | Triggers | DDL

Grid view | Form view

ID	LastName	FirstName	Age	City
1	ALVIN	NULL	NULL	NULL

if you created a new row and click on committ.....

the City should display Singapore by default....

Data | Structure | Constraints | Indexes | Triggers | DDL

Grid view | Form view

Filter data | Total rows loaded: 3

ID	LastName	FirstName	Age	City
1	afdad	NULL	NULL	Singapore
2	tyet	NULL	NULL	Singapore
3	afdasf	NULL	NULL	Msia

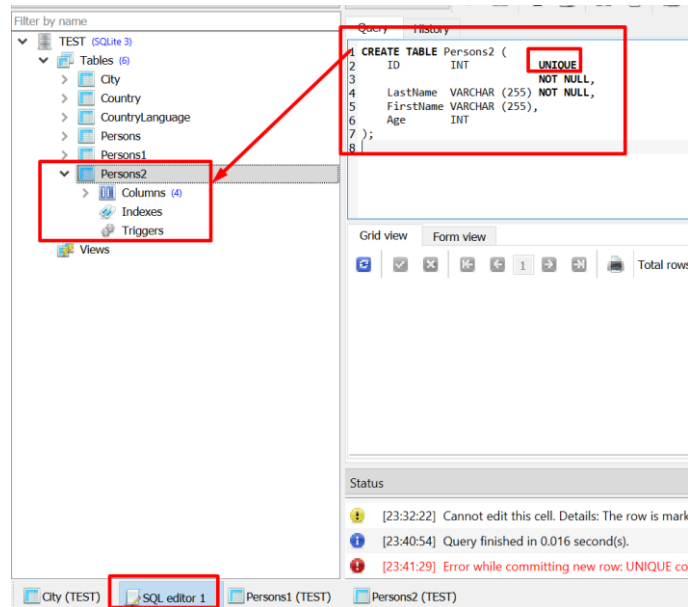
Column: City
Data type: VARCHAR
Table: Persons1
ROWID: 3

Constraints: DEFAULT (Singapore)

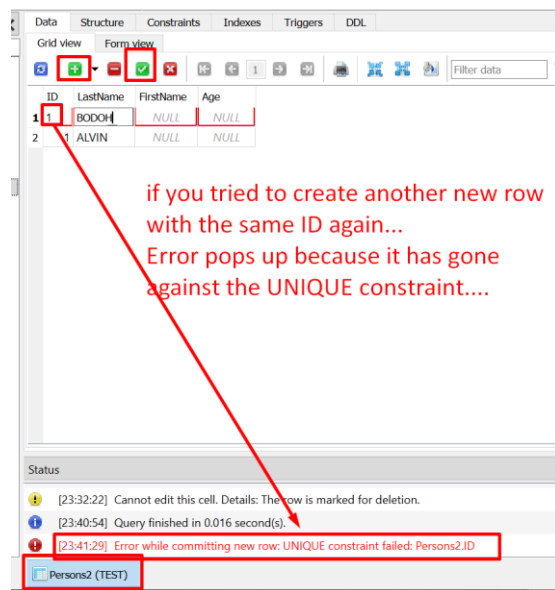
if you typed in Malaysia....
the default value will be overwritten and Msia will be displayed instead

if u kept creating new rows but leaving the City blank... by right, SQLite studio should label Singapore into City by default (but note that sometimes it doesn't display due to bugs)

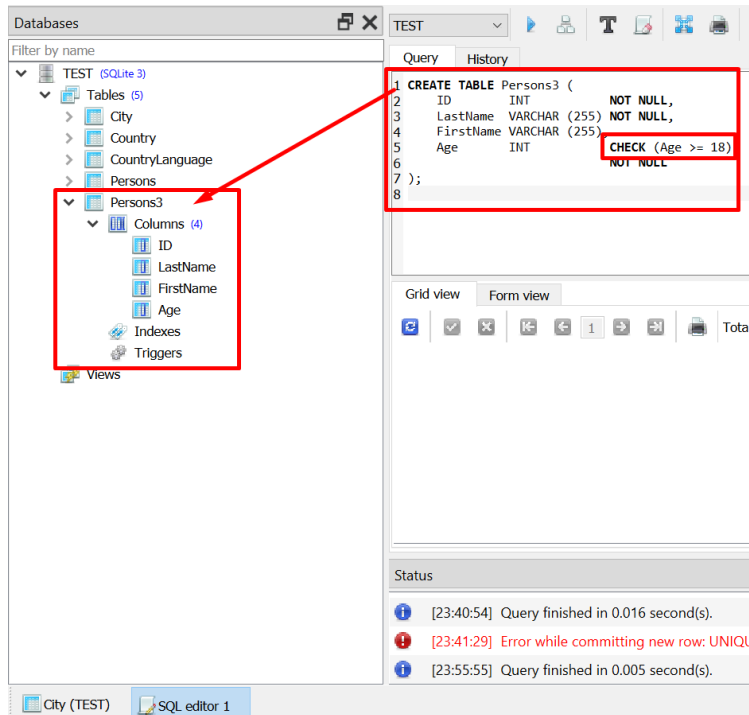
G. UNIQUE



```
CREATE TABLE Persons2 (  
    ID INT UNIQUE NOT NULL,  
    LastName VARCHAR (255) NOT NULL,  
    FirstName VARCHAR (255),  
    Age INT  
);
```



H. CHECK



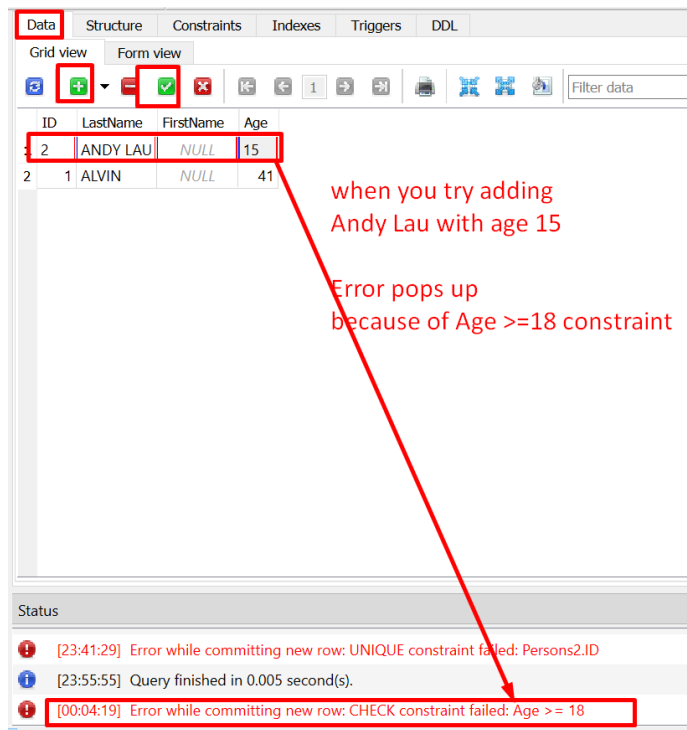
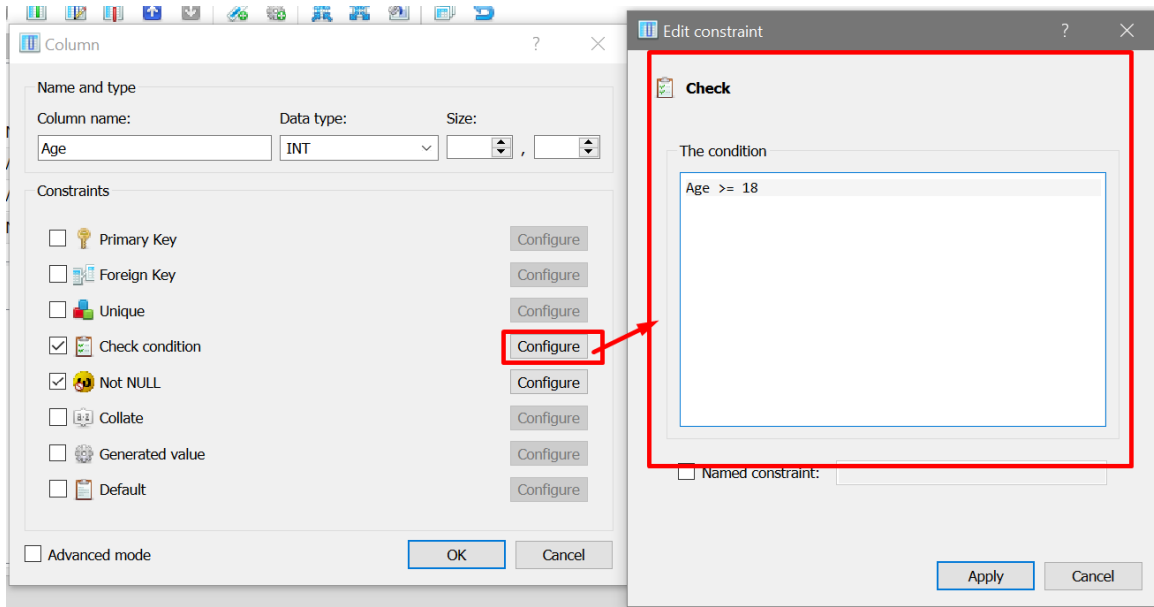
```

CREATE TABLE Persons3 (
  ID INT NOT NULL,
  LastName VARCHAR (255) NOT NULL,
  FirstName VARCHAR (255),
  Age INT CHECK (Age >= 18) NOT NULL
);

```

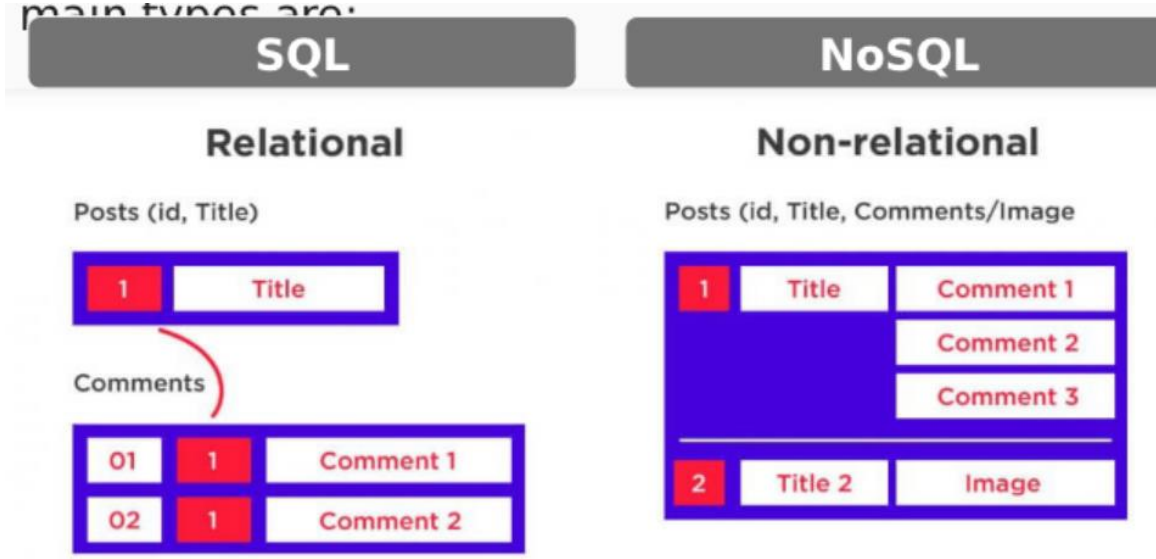
	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated
1	ID	INT					🚫		NULL
2	LastName	VARCHAR (255)					🚫		NULL
3	FirstName	VARCHAR (255)							NULL
4	Age	INT				📄	🚫		NULL

double click here



I. RELATIONAL VS NON RELATIONAL DATABASE

main types are:

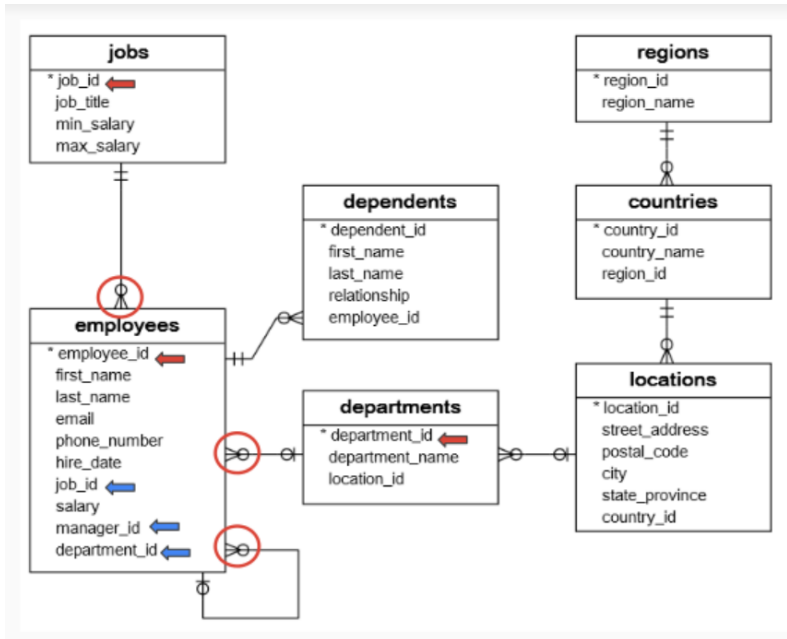


- SQL is a Relational Database because it contains many tables with columns that relate to one another.
- Excel is a Non-Relational Database because it might contain many tables but their columns don't relate to one another.

"Order ID"	"Order Date"	"Order Amount"	Customer	"Customer Address"	"Customer Phone"	"Tax Identifier"
1	jun-23	\$10 248,15	International Services Ltd	1247 St River Blvd, Charlotte, NC	(555) 478-8741	IS789456
2	jun-27	\$14 785,45	World Master Importing Inc.	354 Mountain Hill Rd, Los Angeles, CA	(555) 774-8888	WM321456

- For example, if we created a table to store all our orders like above.
- Everything dumped into 1 table.
- One same customer can make multiple orders.
- But what if he changes his handphone number?
- You need to edit his handphone number many many times!
- Storing all the information in a single table just doesn't work.
- A relational model requires us to define each entity as a separate table and establish relationships between them.
- Relational Model means Tables are related via Primary Key (PK) and Foreign Key (FK).

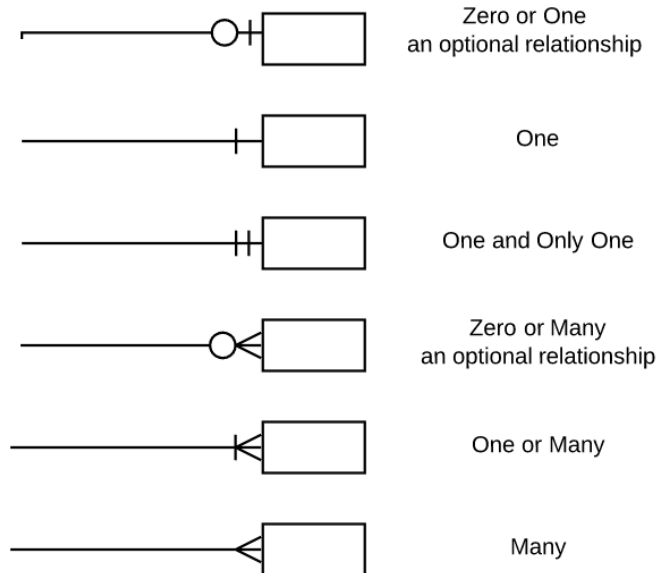
J. PRIMARY KEY AND FOREIGN KEY



- The "*" is the **Primary Key (PK)** for each table.

For the "employees" table:

- "job_id" is the **Foreign Key (FK)** for the "jobs" table.
- "manager_id" is the FK for its own table (manager is also an employee).
- "department_id" is the FK for the "departments" table.



1. JOBS → EMPLOYEES (ONE TO MANY RELATIONSHIP)

- Jobs Table:
 - PK = Job ID, which means that the Job ID is the index column.
 - There's one and only one job id per job.
- Employee Table:
 - Job ID = FK. Can have multiple Job IDs.
 - Each employee can do multiple jobs.
 - Thus, though there is only one Employee ID for each employee, he/she can have multiple Job IDs.
 - Thus, Jobs Table → Employee Table is a One-to-Many Relationship.

2. EMPLOYEES → DEPENDANTS (ONE TO MANY RELATIONSHIP)

- Employees Table:
 - PK = Employee ID, which means that the Employee ID is the Index Column.
 - There's one and only one Employee ID per employee.
- Dependants Table"
 - Employee ID = FK. Can have multiple Employee IDs.
 - Each dependant can have 2 parents working in the same company.
 - E.g. 1 child can have both father and mother working in the same company.
 - Thus, that 1 Dependand ID might be tied to 2 Employee IDs.

3. EMPLOYEES → EMPLOYEES (ZERO OR ONE TO MANY RELATIONSHIP)

- Employee Table:
 - PK = Employee ID, which means that the Employee ID is the Index Column.
 - There's one and only one Employee ID per employee.
 - Each and every Employee ID is tied to one Manager ID.
 - If one manager is managing a team of employees, his Manager ID will show up many times because its tied to multiple Employee ID.
 - Also, each manager is also an employee of the company.
 - He could have both a Manager ID+ Employee ID too; which means it might show up multiple times.

4. CREATING PRIMARY KEY

note that INT(3) means the column can only accept 3 integers....however, if you try keying in more than 3 integers and it allows.....

is because SQLite Studio has bugs and sometimes don't always work.....

CREATE TABLE Persons5 (

ID INT,
 LastName VARCHAR (255) NOT NULL,
 FirstName VARCHAR (255),
 Age INT (3),

PRIMARY KEY (ID)

);

PK has been identified by SQLite

Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated
1 ID	INT	PK						NULL
2 LastName	VARCHAR (255)					☹		NULL
3 FirstName	VARCHAR (255)							NULL
4 Age	INT (3)							NULL

Type	Name	Details
1 PRIMARY KEY	(ID)	

Status
 [15:20:00] Query finished in 0.008 second(s).

5. CREATING FOREIGN KEY

purpose of this is just to delete any previous Orders table

```
10  
11  
12 DROP TABLE IF EXISTS Orders;  
13  
14 CREATE TABLE Orders (  
15   OrderID INT NOT NULL,  
16   OrderNum INT NOT NULL,  
17   PersonID INT,  
18   PRIMARY KEY (OrderID),  
19   FOREIGN KEY (PersonID) REFERENCES Persons(ID)  
20 );  
21  
22
```

```
DROP TABLE IF EXISTS Orders;
```

```
CREATE TABLE Orders (  
  
    OrderID INT NOT NULL,  
    OrderNum INT NOT NULL,  
    PersonID INT,  
  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons5(ID)  
);
```

```
OrderID INT NOT NULL,  
OrderNum INT NOT NULL,  
PersonID INT,
```

```
PRIMARY KEY (OrderID),  
FOREIGN KEY (PersonID) REFERENCES Persons5(ID)
```

```
);
```

Orders

Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated
1 OrderID	INT	PK identified				☹		NULL
2 OrderNum	INT					☹		NULL
3 PersonID	INT		FK identified					NULL

Type	Name	Details
1 PRIMARY KEY	(OrderID)	
2 FOREIGN KEY	(PersonID) REFERENCES Persons5 (ID)	

Orders.PersonID(FK) is now linked to Persons5.ID (PK)

Visio

New

Crow's Foot Database Notation

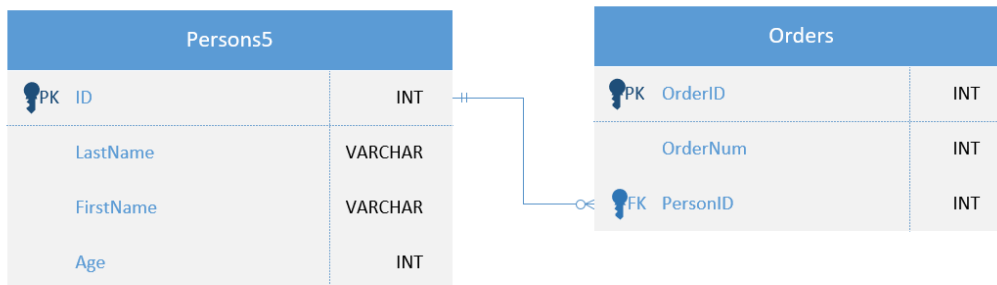
Provided by: Microsoft Corporation

Model a database using one of the most common notations, Crow's Foot, to create an Entity Relationship Diagram.

Metric Units
 US Units

we can create a PK FK relationship diagram in Visio

Create



Entity Relationship Diagram

K. AUTO INCREMENT

The screenshot shows a database management interface. On the left, a tree view displays the database structure under 'TEST (SQLite 3)'. The 'PERSONS6' table is selected, and its columns are listed: PersonID, LastName, FirstName, and Age. On the right, a query editor shows the following SQL code:

```
1 --AUTOINCREMENT allows a unique number to be generated automatically
2 --AUTOINCREMENT is only allowed in an INTEGER PRIMARY KEY
3
4 CREATE TABLE PERSONS6 (
5     PersonID INTEGER PRIMARY KEY AUTOINCREMENT,
6     LastName VARCHAR (255) NOT NULL,
7     FirstName VARCHAR (255),
8     Age INT
9 );
10
```

--AUTOINCREMENT allows a unique number to be generated automatically

--AUTOINCREMENT is only allowed in an INTEGER PRIMARY KEY

```
CREATE TABLE PERSONS6 (
```

```
    PersonID    INTEGER    PRIMARY KEY AUTOINCREMENT,
```

```
    LastName    VARCHAR (255) NOT NULL,
```

```
    FirstName    VARCHAR (255),
```

```
    Age INT
```

```
);
```


if we only keyed in ALVIN
but left PersonID blank and Commit...

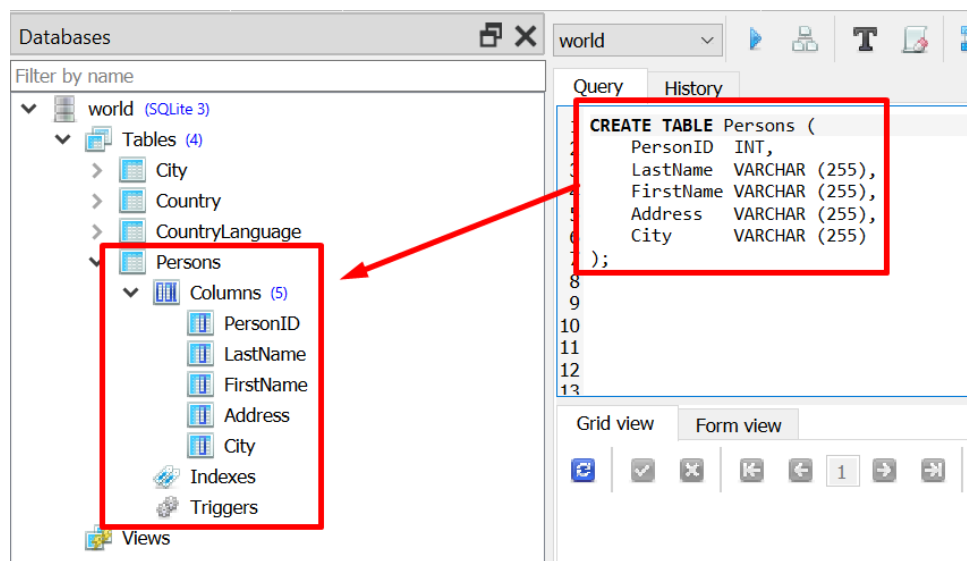
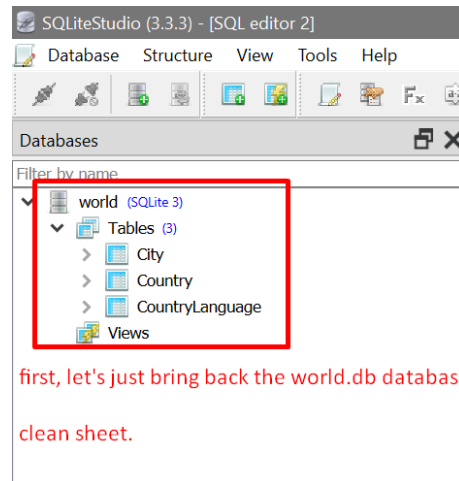
u see that it auto increments and
generates a number

if we recreate another new row, we
get back the same thing....

however if we create a new row and key in 8 for PersonID..

it auto picks up from the previous number and continues from there...

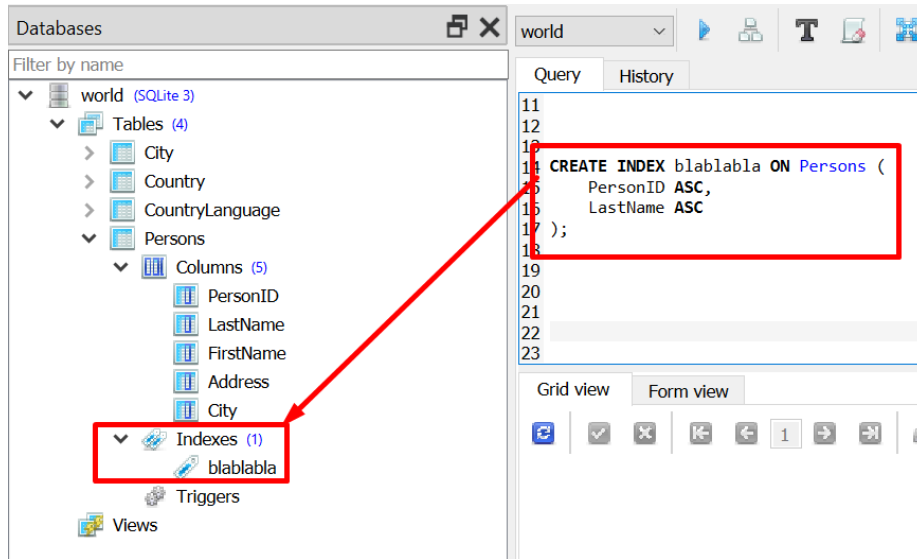
L. CREATE INDEX



CREATE TABLE Persons (

```
PersonID    INT,  
LastName    VARCHAR (255),  
FirstName    VARCHAR (255),  
Address     VARCHAR (255),  
City        VARCHAR (255)
```

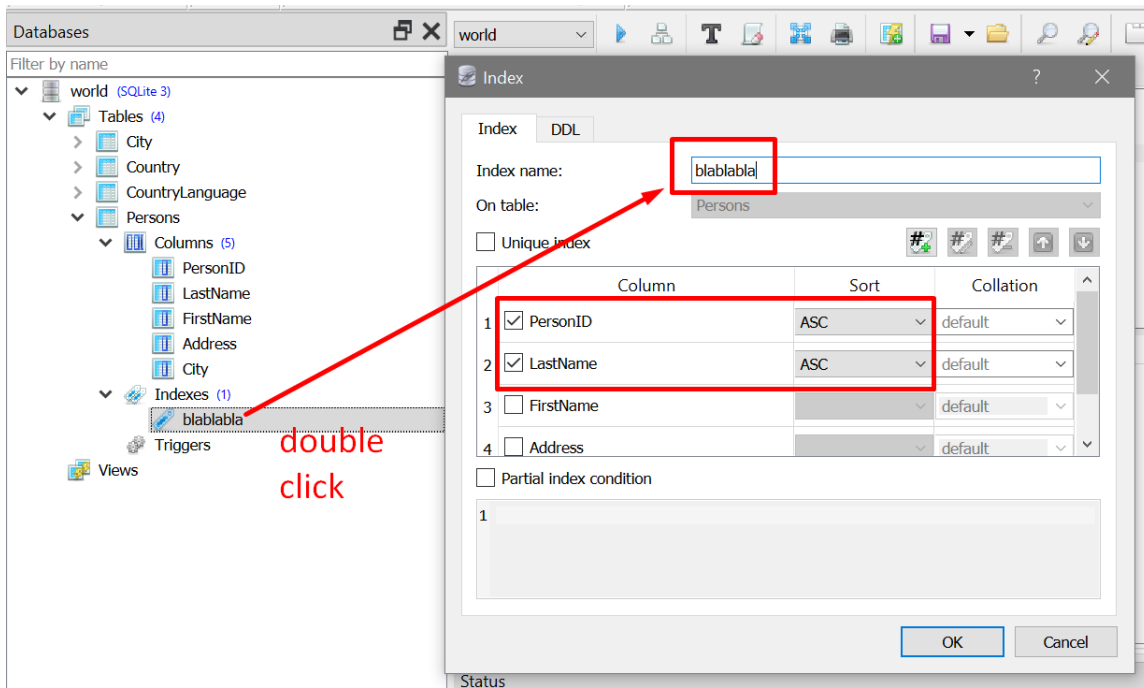
);



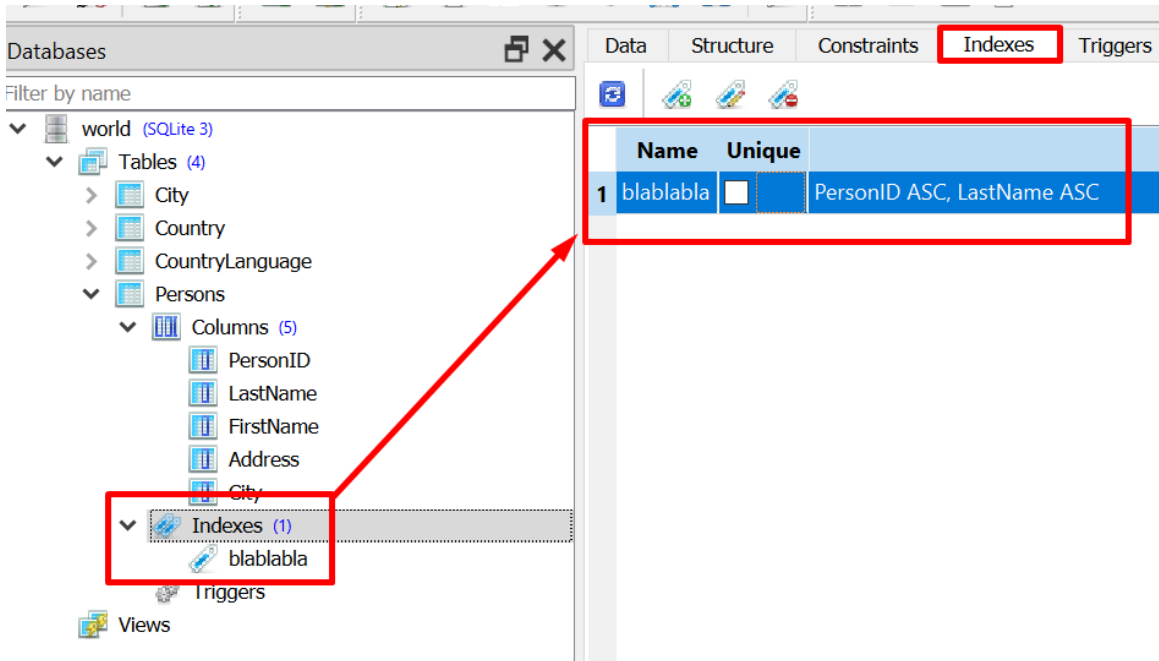
CREATE INDEX blablabla ON Persons (

PersonID ASC,
 LastName ASC

);



- The CREATE INDEX statement is used to create indexes in tables.
- Indexes are used to retrieve data from the database more quickly than otherwise.
- The users cannot see the indexes, they are just used to speed up searches/queries.
- In other words, you can't feel nor see the effect of INDEXES.
- They occur backend.



M. DISTINCT

let's first take a glance at the Region column in the Country Table.....

```
SELECT Region
FROM Country;
```

we display all 239 rows of Region....

Region
1 Southern and Central Asia
2 Western Europe
3 Caribbean
4 Southern Europe
5 Northern Africa
6 Polynesia
7 Southern Europe

SELECT Region
FROM Country;

world

Query History

```
5  
6  
7 SELECT DISTINCT Region  
8 FROM Country;  
9
```

Grid view Form view

Total rows loaded: 25

	Region
1	Southern and Central Asia
2	Western Europe
3	Caribbean
4	Southern Europe
5	Northern Africa
6	Polynesia
7	Central Africa
8	Middle East
9	South America
10	Australia and New Zealand
11	Central America

but there are only 25 distinct regions in the country table....

```
SELECT DISTINCT Region  
FROM Country;
```

N. WHERE

The screenshot shows a database application interface. On the left, a tree view displays the database structure for 'world (SQLite 3)', with 'City' and its 'District' column highlighted. The main query editor shows the following SQL code:

```
1 SELECT *  
2 FROM City  
3 WHERE District = 'Kabool';
```

Below the query editor, a grid view displays the results of the query. The grid has 5 columns: ID, Name, CountryCode, District, and Population. One row is visible, representing the city of Kabul.

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabool	1780000

```
SELECT *  
FROM City  
WHERE District = 'Kabool';
```

The screenshot shows a database application interface. On the left, a tree view displays the database structure for 'world (SQLite 3)', with 'Country' and its 'Continent' column highlighted. The main query editor shows the following SQL code:

```
6 SELECT *  
7 FROM Country  
8 WHERE Continent = 'Asia';
```

Below the query editor, a grid view displays the results of the query. The grid has 11 columns: Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpect, GNP, GNPOld, and LocalName. 11 rows are visible, representing various countries in Asia.

Code	Name	Continent	Region	SurfaceAr	IndepYear	Population	LifeExpect	GNP	GNPOld	LocalName
1	AFG	Asia	Southern and Central Asia	652090	1919	22720000	45.9	5976	NULL	Afghanistan/Afghanistan
2	ARE	Asia	Middle East	83600	1971	2441000	74.1	37966	36846	Al-Imarat al-'Arabiya al-Mut
3	ARM	Asia	Middle East	29800	1991	3520000	66.4	1813	1627	Hajastan
4	AZE	Asia	Middle East	86600	1991	7734000	62.9	4127	4100	Azərbaycan
5	BHR	Asia	Middle East	694	1971	617000	73	6366	6097	Al-Bahrayn
6	BGD	Asia	Southern and Central Asia	143998	1971	129155000	60.2	32852	31966	Bangladesh
7	BTN	Asia	Southern and Central Asia	47000	1910	2124000	52.4	372	383	Druk-Yul
8	BRN	Asia	Southeast Asia	5765	1984	328000	73.6	11705	12460	Brunei Darussalam
9	PHL	Asia	Southeast Asia	300000	1946	75967000	67.5	65107	82239	Pilipinas
10	GEO	Asia	Middle East	69700	1991	4968000	64.5	6064	5924	Sakartvelo
11	HKG	Asia	Eastern Asia	1075	NULL	6782000	79.5	166448	173610	Xianqiang/Hong Kong

```
SELECT *  
FROM Country  
WHERE Continent = 'Asia';
```


The screenshot shows a database query tool interface. At the top, there is a dropdown menu set to 'world' and a toolbar with various icons. Below that, there are tabs for 'Query' and 'History'. The query editor contains the following SQL code:

```
13  
14 SELECT Name, Region  
15 FROM Country  
16 WHERE Region = 'Southeast Asia';  
17
```

Below the query editor, there are tabs for 'Grid view' and 'Form view'. The 'Grid view' is selected, and the results are displayed in a table. The table has two columns: 'Name' and 'Region'. The results are as follows:

	Name	Region
1	Brunei	Southeast Asia
2	Philippines	Southeast Asia
3	Indonesia	Southeast Asia
4	East Timor	Southeast Asia
5	Cambodia	Southeast Asia
6	Laos	Southeast Asia
7	Malaysia	Southeast Asia
8	Myanmar	Southeast Asia
9	Singapore	Southeast Asia
10	Thailand	Southeast Asia
11	Vietnam	Southeast Asia

```
SELECT Name, Region  
FROM Country  
WHERE Region = 'Southeast Asia';
```

1. LIKE

```
1 SELECT *
2 FROM Country
3 WHERE Name LIKE 'K%';
4
5 -- $zero, one, or multiple characters match
6
```

Code	Name	Continent	Region	SurfaceAr	IndepYear	Population	LifeExpect	GNP	GNPOld	LocalName	GovernmentForm
1 KAZ	Kazakistan	Asia	Southern and Central Asia	2724900	1991	16223000	63.2	24375	23383	Qazaqstan	Republic
2 KEN	Kenya	Africa	Eastern Africa	580367	1963	30080000	48	9217	10241	Kenya	Republic
3 KGZ	Kyrgyzstan	Asia	Southern and Central Asia	199900	1991	4699000	63.4	1626	1767	Kyrgyzstan	Republic
4 KIR	Kiribati	Oceania	Micronesia	726	1979	83000	59.8	40.7	NULL	Kiribati	Republic
5 KWT	Kuwait	Asia	Middle East	17818	1961	1972000	76.1	27037	30373	Al-Kuwayt	Constitutional Monarchy (Emirate)

```
SELECT *
FROM Country
WHERE Name
```

```
LIKE 'K%';
```

```
7 SELECT *
8 FROM Country
9 WHERE Name LIKE 'S_____';
10
11 -- _ a single character match %;
```

Code	Name	Continent	Region	SurfaceAr	IndepYear	Population	LifeExpect	GNP	GNPOld	LocalName	GovernmentForm	HeadOfState
1 SWE	Sweden	Europe	Nordic Countries	449964	836	8861400	79.6	226492	227757	Sverige	Constitutional Monarchy	Carl XVI Gustaf

```
SELECT *
FROM Country
WHERE Name
LIKE 'S_____';
```

2. IS NULL

world

Query History

```
1 SELECT Name,  
2     HeadOfState,  
3     Population,  
4     IndepYear  
5 FROM Country  
6 WHERE IndepYear ISNULL;  
7
```

Grid view Form view

Total rows loaded: 47

	Name	HeadOfState	Population	IndepYear
1	Netherlands Antilles	Beatrix	217000	NULL
2	American Samoa	George W. Bush	68000	NULL
3	Anguilla	Elisabeth II	8000	NULL
4	Aruba	Beatrix	103000	NULL
5	Bermuda	Elisabeth II	65000	NULL
6	Virgin Islands, British	Elisabeth II	21000	NULL
7	Cayman Islands	Elisabeth II	38000	NULL
8	Cook Islands	Elisabeth II	20000	NULL
9	Falkland Islands	Elisabeth II	2000	NULL
10	Faroe Islands	Margrethe II	43000	NULL

```
SELECT Name,  
       HeadOfState,  
       Population,  
       IndepYear
```

```
FROM Country  
WHERE IndepYear ISNULL;
```

from the full table, we see that there are many NULLS in IndepYear...

	Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpect	GNP	GNPOld
1	AFG	Afghanistan	Asia	Southern and Central Asia	652090	1919	22720000	45.9	5976	NULL
2	NLD	Netherlands	Europe	Western Europe	41526	1581	15864000	78.3	371362	3604
3	ANT	Netherlands Antilles	North America	Caribbean	800	NULL	217000	74.7	1941	NULL
4	ALB	Albania	Europe	Southern Europe	28748	1912	3401200	71.6	3205	25
5	DZA	Algeria	Africa	Northern Africa	2381741	1962	31471000	69.7	49982	469
6	ASM	American Samoa	Oceania	Polynesia	199	NULL	68000	75.1	334	NULL
7	AND	Andorra	Europe	Southern Europe	468	1278	78000	83.5	1630	NULL
8	AGO	Angola	Africa	Central Africa	1246700	1975	12878000	38.3	6648	79
9	AIA	Anguilla	North America	Caribbean	96	NULL	8000	76.1	63.2	NULL
10	ATG	Antigua and Barbuda	North America	Caribbean	442	1981	68000	70.5	612	5
11	ARE	United Arab Emirates	Asia	Middle East	83600	1971	2441000	74.1	37966	366
12	ARG	Argentina	South America	South America	2780400	1816	37032000	75.1	340238	3233
13	ARM	Armenia	Asia	Middle East	29800	1991	3520000	66.4	1813	16
14	ABW	Aruba	North America	Caribbean	193	NULL	103000	78.4	828	7
15	AUS	Australia	Oceania	Australia and New Zealand	7741220	1901	18886000	79.8	351182	3925
16	AZE	Azerbaijan	Asia	Middle East	86600	1991	7734000	62.9	4127	41

one way to quickly get all the NULLS is to just click here....

	Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpect	GNP	GNPOld
1	ANT	Netherlands Antilles	North America	Caribbean	800	NULL	217000	74.7	1941	NULL
2	ASM	American Samoa	Oceania	Polynesia	199	NULL	68000	75.1	334	NULL
3	AIA	Anguilla	North America	Caribbean	96	NULL	8000	76.1	63.2	NULL
4	ABW	Aruba	North America	Caribbean	193	NULL	103000	78.4	828	7
5	BMU	Bermuda	North America	North America	53	NULL	65000	76.9	2328	21
6	VGB	Virgin Islands, British	North America	Caribbean	151	NULL	21000	75.4	612	5
7	CYM	Cayman Islands	North America	Caribbean	264	NULL	38000	78.9	1263	11
8	COK	Cook Islands	Oceania	Polynesia	236	NULL	20000	71.1	100	NULL
9	FLK	Falkland Islands	South America	South America	12173	NULL	2000	NULL	0	NULL
10	FRO	Faroe Islands	Europe	Nordic Countries	1399	NULL	43000	78.4	0	NULL
11	GIB	Gibraltar	Europe	Southern Europe	6	NULL	25000	79	258	NULL
12	GRL	Greenland	North America	North America	2166090	NULL	56000	68.1	0	NULL
13	GLP	Guadeloupe	North America	Caribbean	1705	NULL	456000	77	3501	NULL
14	GUM	Guam	Oceania	Micronesia	549	NULL	168000	77.8	1197	11
15	HKG	Hong Kong	Asia	Eastern Asia	1075	NULL	6782000	79.5	166448	1736
16	SJM	Svalbard and Jan Mayen	Europe	Nordic Countries	62422	NULL	3200	NULL	0	NULL

3. EXISTS

The screenshot displays a database management tool interface. On the left, a tree view shows the database structure for 'world (SQLite 3)', including tables 'City' and 'Country'. The 'City' table has columns 'ID', 'Name', 'CountryCode', 'District', and 'Population'. The 'Country' table has columns 'Code', 'Name', 'Continent', 'Region', 'SurfaceArea', 'IndepYear', 'Population', 'LifeExpectancy', 'GNP', 'GNPOld', 'LocalName', 'GovernmentForm', 'HeadOfState', 'Capital', and 'Code2'. The 'CountryCode' column in the 'City' table and the 'Code' column in the 'Country' table are highlighted with red boxes. The query editor shows the following SQL query:

```
SELECT *
FROM City
WHERE EXISTS (
    SELECT Country.Code
    FROM Country
    WHERE Country.Code = City.CountryCode AND
    LifeExpectancy > 80
);
```

The data grid shows the results of the query, with 9 rows. The first row is highlighted in red and has the following values: ID: 55, Name: Andorra la Vella, CountryCode: AND, District: Andorra la Vella, Population: 21189. The status bar shows an error message: '[23:55:02] Error while executing SQL query on database 'world': no such table: Country'.

```
SELECT *
FROM City
WHERE EXISTS
(
    SELECT Country.Code
    FROM Country
    WHERE Country.Code = City.CountryCode
    AND LifeExpectancy > 80
);
```

4. AND

world

Query History

```
1 SELECT *
2 FROM Country
3 WHERE Continent = 'Asia' AND
4 LifeExpectancy < 70;
5
6
7
8
9
10
```

Grid view Form view

Total rows loaded: 29

	Code	Name	Continent	Region	SurfaceAr	IndepYear	Population	LifeExpect	GNP	GNPOld	LocalName	Goverr
1	AFG	Afghanistan	Asia	Southern and Central Asia	652090	1919	22720000	45.9	5976	NULL	Afghanistan/Afqanestan	Islamic
2	ARM	Armenia	Asia	Middle East	29800	1991	3520000	66.4	1813	1627	Hajastan	Republ
3	AZE	Azerbaijan	Asia	Middle East	86600	1991	7734000	62.9	4127	4100	Azərbaycan	Federa
4	BGD	Bangladesh	Asia	Southern and Central Asia	143998	1971	129155000	60.2	32852	31966	Bangladesh	Republ
5	BTN	Bhutan	Asia	Southern and Central Asia	47000	1910	2124000	52.4	372	383	Druk-Yul	Monar
6	PHL	Philippines	Asia	Southeast Asia	300000	1946	75967000	67.5	65107	82239	Pilipinas	Republ
7	GEO	Georgia	Asia	Middle East	69700	1991	4968000	64.5	6064	5924	Sakartvelo	Republ
8	IDN	Indonesia	Asia	Southeast Asia	1904569	1945	212107000	68	84982	215002	Indonesia	Republ
<	IND	India	Asia	Southern and Central Asia	2287262	1947	1012662000	62.5	447114	420573	Bharat/India	Federa

Status

[2:55:14] Query finished in 0.002 seconds(s).

```
SELECT *
FROM Country
WHERE Continent = 'Asia' AND
LifeExpectancy < 70;
```

5. OR

world

Query History

```
10  
11 SELECT *  
12 FROM Country  
13 WHERE GovernmentForm = 'Monarchy' OR  
14 LifeExpectancy < 40;  
15
```

Grid view Form view

Total rows loaded: 12

	Region	SurfaceArea	IndepYear	Population	LifeExpectancy	GNP	GNPOld	LocalName	GovernmentForm	HeadOfState	Capital
1	Central Africa	1246700	1975	12878000	38.3	6648	7984	Angola	Republic	José Eduardo dos Santos	
2	Southern and Central Asia	47000	1910	2124000	52.4	372	383	Druk-Yul	Monarchy	Jigme Singye Wangchuk	
3	Southern Africa	581730	1966	1622000	39.3	4834	4935	Botswana	Republic	Festus G. Mogae	
4	Eastern Africa	118484	1964	10925000	37.6	1687	2527	Malawi	Republic	Bakili Muluzi	
5	Eastern Africa	801590	1975	19680000	37.5	2891	2711	Moçambique	Republic	Joaquim A. Chissano	
6	Middle East	11000	1971	599000	72.4	9472	8920	Qatar	Monarchy	Hamad ibn Khalifa al-Thani	
7	Eastern Africa	26338	1962	7733000	39.3	2036	1863	Rwanda/Urwanda	Republic	Paul Kagame	
8	Eastern Africa	752618	1964	9169000	37.2	3377	3922	Zambia	Republic	Frederick Chiluba	
9	Middle East	2149690	1932	21607000	67.8	137635	146171	al-'Arabiyyah as-Su'ūdiyyah	Monarchy	Fahd ibn Abdul-Aziz al-Sa'ud	
10	Southern Africa	17364	1968	1008000	40.4	1206	1312	kaNgwane	Monarchy	Mswati III	
11	Polynesia	650	1970	99000	67.9	146	170	Tonga	Monarchy	Taufa'ahau Tupou IV	

Status

```
SELECT *  
FROM Country  
WHERE GovernmentForm = 'Monarchy' OR  
LifeExpectancy < 40;
```

6. NOT

world

Query History

```
21  
22 SELECT *  
23 FROM Country  
24 WHERE NOT (GovernmentForm = 'Monarchy' OR  
25 LifeExpectancy > 40);
```

Grid view Form view

Total rows loaded: 7

	Continent	Region	SurfaceAr	IndepYear	Population	LifeExpect	GNP	GNPOld	LocalName	GovernmentForm	HeadOfState	Capital
1	Africa	Central Africa	1246700	1975	12878000	38.3	6648	7984	Angola	Republic	José Eduardo dos Santos	56
2	Africa	Southern Africa	581730	1966	1622000	39.3	4834	4935	Botswana	Republic	Festus G. Mogae	204
3	Africa	Eastern Africa	118484	1964	10925000	37.6	1687	2527	Malawi	Republic	Bakili Muluzi	2462
4	Africa	Eastern Africa	801590	1975	19680000	37.5	2891	2711	Moçambique	Republic	Joaquim A. Chissano	2698
5	Africa	Eastern Africa	26338	1962	7733000	39.3	2036	1863	Rwanda/Urwanda	Republic	Paul Kagame	3047
6	Africa	Eastern Africa	752618	1964	9169000	37.2	3377	3922	Zambia	Republic	Frederick Chiluba	3162
7	Africa	Eastern Africa	390757	1980	11669000	37.8	5951	8670	Zimbabwe	Republic	Robert G. Mugabe	4068

```
SELECT *  
FROM Country  
WHERE NOT  
(  
    GovernmentForm = 'Monarchy' OR  
    LifeExpectancy > 40  
);
```


7. IN

The screenshot shows a database query tool interface. The query editor displays the following SQL query:

```
1 SELECT *
2 FROM City
3 WHERE CountryCode IN ('AFG', 'NLD');
4
5
6
```

The results are shown in a grid view with the following columns: ID, Name, CountryCode, District, and Population. The 'CountryCode' column is highlighted in red. The total rows loaded is 32.

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabul	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398

```
SELECT *
FROM City
WHERE CountryCode IN ('AFG', 'NLD');
```

Query History

```
1 SELECT *
2 FROM City
3 WHERE CountryCode IN ('AFG', 'NLD');
```

```
7 SELECT *
8 FROM City
9 WHERE (CountryCode = 'AFG' OR
10 CountryCode = 'NLD');
```

this is the same as
this...
both gives the same result

Grid view Form view

Total rows loaded: 32

	ID	Name	CountryCode	District	Population
1	1	Kabul	AFG	Kabul	1780000
2	2	Qandahar	AFG	Qandahar	237500
3	3	Herat	AFG	Herat	186800
4	4	Mazar-e-Sharif	AFG	Balkh	127800
5	5	Amsterdam	NLD	Noord-Holland	731200
6	6	Rotterdam	NLD	Zuid-Holland	593321
7	7	Haag	NLD	Zuid-Holland	440900
8	8	Utrecht	NLD	Utrecht	234323

```
SELECT *
FROM City
WHERE (CountryCode = 'AFG' OR
CountryCode = 'NLD');
```

O. ORDER BY

Query

```
1 SELECT *  
2 FROM City  
3 ORDER BY district;  
4  
5
```

Grid view Form view

Total rows loaded: 4079

	ID	Name	CountryCode	District	Population
1	3285	Taiping	TWN		165524
2	3293	Taliao	TWN		115897
3	3294	Kueishan	TWN		112195
4	3563	Ciudad Losada	VEN		134501
5	893	Sultan Kudarat	PHL	ARMM	94861
6	909	Sohumi	GEO	Abhasia [Aphazeti]	111700
7	2812	Abidjan	CIV	Abidjan	2500000
8	1498	Pescara	ITA	Abruzzit	115698
9	65	Abu Dhabi	ARE	Abu Dhabi	398695
10	67	al-Ayn	ARE	Abu Dhabi	225970
11	982	Banda Aceh	IDN	Aceh	143409
12	999	Lhokseumawe	IDN	Aceh	109600

ASC is default if not mentioned...

```
SELECT *  
FROM City  
ORDER BY district;
```

Query History

```
SELECT *  
FROM City  
ORDER BY CountryCode;
```

Grid view Form view

Total rows loaded: 3333

	ID	Name	CountryCode	District	Population
1	129	Oranjestad	ABW		29034
2	1	Kabul	AFG	Kabul	1780000
3	2	Qandahar	AFG	Qandahar	237500
4	3	Herat	AFG	Herat	186800
5	4	Mazar-e-Sharif	AFG	Balkh	127800
6	56	Luanda	AGO	Luanda	2022000
7	57	Huambo	AGO	Huambo	163100
8	58	Lobito	AGO	Benguela	130000
9	59	Benguela	AGO	Benguela	128300
10	60	Namibe	AGO	Namibe	118200
11	61	South Hill	AIA		961
12	62	The Valley	AIA		595

```
SELECT *  
FROM City  
ORDER BY CountryCode;
```

world

Query History

```

5 SELECT *
6 FROM Country
7 WHERE LifeExpectancy > 40
8 ORDER BY NAME DESC;
9
10
11
12
13

```

Grid view Form view

Total rows loaded: 215

	Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpecta	GNP	GNPOld	Loc
1	YUG	Yugoslavia	Europe	Southern Europe	102173	1918	10640000	72.4	17000	NULL	Jug
2	YEM	Yemen	Asia	Middle East	527968	1918	18112000	59.8	6041	5729	Al-
3	ESH	Western Sahara	Africa	Northern Africa	266000	NULL	293000	49.8	60	NULL	As-
4	VIR	Virgin Islands, U.S.	North America	Caribbean	347	NULL	93000	78.1	0	NULL	Vir
5	VGB	Virgin Islands, British	North America	Caribbean	151	NULL	21000	75.4	612	573	Brit
6	VNM	Vietnam	Asia	Southeast Asia	331689	1945	79832000	69.3	21929	22834	Viê
7	VEN	Venezuela	South America	South America	912050	1811	24170000	73.1	95023	88434	Ver
8	VUT	Vanuatu	Oceania	Melanesia	12189	1980	190000	60.6	261	246	Var
9	UZB	Uzbekistan	Asia	Southern and Central Asia	447400	1991	24318000	63.7	14194	21300	Uzt

```

SELECT      *
FROM        Country
WHERE       LifeExpectancy > 40
ORDER BY   NAME DESC;

```

1. ACTIVITY QUESTION: ORDER BY / OPERATORS

<https://www.alvinang.sg/s/world.db>

- Select Name, Continent, and Population from Country
- Where the Name is like ISLAND
- Order by Name

The screenshot shows a database query tool interface. At the top, there are tabs for 'Query' and 'History'. The 'Query' tab is active, displaying a SQL query:

```
1 SELECT Name,  
2     Continent,  
3     Population  
4 FROM Country  
5  
6 WHERE Name LIKE '%island%'  
7 ORDER BY Name;  
8
```

Below the query editor, there are tabs for 'Grid view' and 'Form view'. The 'Grid view' tab is active, showing a table of results. The table has three columns: 'Name', 'Continent', and 'Population'. The results are sorted by Name in ascending order. The first row is 'Bouvet Island' with a population of NULL. The last row is 'Marshall Islands' with a population of 54000. The total number of rows loaded is 18.

	Name	Continent	Population
1	Bouvet Island	Antarctica	NULL
2	Cayman Islands	North America	38000
3	Christmas Island	Oceania	2500
4	Cocos (Keeling) Islands	Oceania	600
5	Cook Islands	Oceania	20000
6	Falkland Islands	South America	2000
7	Faroe Islands	Europe	43000
8	Fiji Islands	Oceania	817000
9	Heard Island and McDonald Islands	Antarctica	NULL
10	Marshall Islands	Oceania	54000

2. ACTIVITY ANSWER: ORDER BY / OPERATORS

```
SELECT Name,  
        Continent,  
        Population  
FROM Country  
  
WHERE Name LIKE '%island%'  
ORDER BY Name;
```

3. ACTIVITY QUESTION: ORDER BY / OPERATORS

<https://www.alvinang.sg/s/world.db>

- Select Name, Continent, and Population
- From Country
- Where Continent is in Europe and Asia
- Order by Name

```
1 SELECT Name,  
2     Continent,  
3     Population  
4 FROM Country  
5  
6 WHERE Continent IN ('Europe', 'Asia')  
7 ORDER BY Name;  
8
```

Grid view Form view

Total rows loaded: 97

	Name	Continent	Population
1	Afghanistan	Asia	22720000
2	Albania	Europe	3401200
3	Andorra	Europe	78000
4	Armenia	Asia	3520000
5	Austria	Europe	8091800
6	Azerbaijan	Asia	7734000
7	Bahrain	Asia	617000
8	Bangladesh	Asia	129155000
9	Belarus	Europe	10236000
10	Belgium	Europe	10230000

4. ACTIVITY ANSWER: ORDER BY / OPERATORS

```
SELECT Name,  
        Continent,  
        Population  
FROM Country  
  
WHERE Continent IN ('Europe', 'Asia')  
ORDER BY Name;
```

5. ACTIVITY QUESTION: ORDER BY / OPERATORS

<https://www.alvinang.sg/s/world.db>

- Retrieve all the City Names
- With Population more than 1 million and
- Order By Ascending Order

The screenshot shows a database query tool interface. The top part displays a SQL query in a text editor with line numbers 1 through 7. The query is: `1 SELECT Name,
2 Population
3 FROM City
4
5 WHERE Population > 1000000
6 ORDER BY Population DESC;
7`. The 'WHERE' and 'ORDER BY' clauses are highlighted with a red box. Below the query editor, there are tabs for 'Grid view' and 'Form view', and a toolbar with various icons. The 'Grid view' is selected, and the results are displayed in a table. The table has two columns: 'Name' and 'Population'. The results are sorted in descending order of population. The 'Population' column is highlighted with a red box. The total rows loaded is 237.

	Name	Population
1	Mumbai (Bombay)	10500000
2	Seoul	9981619
3	São Paulo	9968485
4	Shanghai	9696300
5	Jakarta	9604900
6	Karachi	9269265
7	Istanbul	8787958
8	Ciudad de México	8591309
9	Moscow	8389200
10	New York	8008278
11	Tokyo	7980230

6. ACTIVITY ANSWER: ORDER BY / OPERATORS

```
SELECT    Name,  
          Population  
FROM      City  
  
WHERE     Population > 1000000  
ORDER BY  Population DESC;
```

P. BETWEEN

The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons. Below the toolbar, there are tabs for 'Query' and 'History'. The 'Query' tab is active, and it contains the following SQL query:

```
SELECT *  
FROM Country  
WHERE Population BETWEEN 100000 AND 200000;  
;
```

Below the query editor, there are tabs for 'Grid view' and 'Form view'. The 'Grid view' tab is active, and it displays a table of results. The table has 11 columns: Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpecta, GNP, GNPOld, and LocalName. The 'Population' column is highlighted in red. The table contains 10 rows of data, with the first row being Aruba and the last row being Vanuatu.

	Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpecta	GNP	GNPOld	LocalName
1	ABW	Aruba	North America	Caribbean	193	NULL	103000	78.4	828	793	Aruba
2	GUM	Guam	Oceania	Micronesia	549	NULL	168000	77.8	1197	1136	Guam
3	MYT	Mayotte	Africa	Eastern Africa	373	NULL	149000	59.5	0	NULL	Mayotte
4	FSM	Micronesia, Federated States of	Oceania	Micronesia	702	1990	119000	68.6	212	NULL	Micronesia
5	GUF	French Guiana	South America	South America	90000	NULL	181000	76.1	681	NULL	Guyane française
6	LCA	Saint Lucia	North America	Caribbean	622	1979	154000	72.3	571	NULL	Saint Lucia
7	VCT	Saint Vincent and the Grenadines	North America	Caribbean	388	1979	114000	72.3	285	NULL	Saint Vincent and the Grenadines
8	WSM	Samoa	Oceania	Polynesia	2831	1962	180000	69.2	141	157	Samoa
9	STP	Sao Tome and Principe	Africa	Central Africa	964	1975	147000	65.3	6	NULL	São Tomé e Príncipe
10	VUT	Vanuatu	Oceania	Melanesia	12189	1980	190000	60.6	261	246	Vanuatu

```
SELECT *  
FROM Country  
WHERE Population BETWEEN 100000 AND 200000;
```

Q. LIMIT

<https://www.alvinang.sg/s/world.db>

The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons. Below the toolbar, there are tabs for 'Query' and 'History'. The 'Query' tab is active, and it contains the following SQL query:

```
1 SELECT *  
2 FROM City  
3 LIMIT 10;  
4  
5
```

The query is highlighted with a red box. Below the query editor, there are tabs for 'Grid view' and 'Form view'. The 'Grid view' tab is active, and it shows a table of results. The table has 10 rows and 5 columns: ID, Name, CountryCode, District, and Population. The first 10 rows are highlighted with a red box.

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabul	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238

```
SELECT *  
FROM City  
LIMIT 10;
```

world

Query History

```

1 SELECT *
2 FROM Country
3 ORDER BY SurfaceArea
4 LIMIT 10;

```

Grid view Form view

Total rows loaded: 10

	Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpect	GNP	GNPOld	LocalName
1	VAT	Holy See (Vatican City State)	Europe	Southern Europe	0.4	1929	1000	NULL	9	NULL	Santa Sede/Città
2	MCO	Monaco	Europe	Western Europe	1.5	1861	34000	78.8	776	NULL	Monaco
3	GIB	Gibraltar	Europe	Southern Europe	6	NULL	25000	79	258	NULL	Gibraltar
4	TKL	Tokelau	Oceania	Polynesia	12	NULL	2000	NULL	0	NULL	Tokelau
5	CCK	Cocos (Keeling) Islands	Oceania	Australia and New Zealand	14	NULL	600	NULL	0	NULL	Cocos (Keeling) Is
6	UMI	United States Minor Outlying Islands	Oceania	Micronesia/Caribbean	16	NULL	NULL	NULL	0	NULL	United States Mir
7	MAC	Macao	Asia	Eastern Asia	18	NULL	473000	81.6	5749	5940	Macao/Aomen
8	NRU	Nauru	Oceania	Micronesia	21	1968	12000	60.8	197	NULL	Naoero/Nauru
9	TUV	Tuvalu	Oceania	Polynesia	26	1978	12000	66.3	6	NULL	Tuvalu
10	NFK	Norfolk Island	Oceania	Australia and New Zealand	36	NULL	2000	NULL	0	NULL	Norfolk Island

```

SELECT      *
FROM        Country
ORDER BY   SurfaceArea
LIMIT      10;

```

<https://www.alvinang.sg/s/world.db>

1. ACTIVITY QUESTION: SELECT DATA / LIMIT

- Select the Top 10 Cities Order by Population in Descending Order

```
SELECT *
FROM City
ORDER BY Population DESC
LIMIT 10;
```

Grid view Form view

Total rows loaded: 10

	ID	Name	CountryCode	District	Population
1	1024	Mumbai (Bombay)	IND	Maharashtra	10500000
2	2331	Seoul	KOR	Seoul	9981619
3	206	São Paulo	BRA	São Paulo	9968485
4	1890	Shanghai	CHN	Shanghai	9696300
5	939	Jakarta	IDN	Jakarta Raya	9604900
6	2822	Karachi	PAK	Sindh	9269265
7	3357	Istanbul	TUR	Istanbul	8787958
8	2515	Ciudad de México	MEX	Distrito Federal	8591309
9	3580	Moscow	RUS	Moscow (City)	8389200
10	3793	New York	USA	New York	8008278

2. ACTIVITY ANSWER: SELECT DATA / LIMIT

```
SELECT *
FROM City
ORDER BY Population DESC
LIMIT 10;
```

3. ACTIVITY QUESTION: SELECT DATA / LIMIT

<https://www.alvinang.sg/s/world.db>

- Select 5 Name, Continent and Region
- Where Continent is Europe
- Order by Name

The screenshot shows a database query tool interface. The top section is labeled 'Query' and contains the following SQL query:

```
1 SELECT Name,  
2     Continent,  
3     Region  
4 FROM Country  
5 WHERE Continent = 'Europe'  
6 ORDER BY Name  
7 LIMIT 5;
```

The bottom section is labeled 'Grid view' and shows the results of the query in a table. The table has three columns: Name, Continent, and Region. The results are:

	Name	Continent	Region
1	Albania	Europe	Southern Europe
2	Andorra	Europe	Southern Europe
3	Austria	Europe	Western Europe
4	Belarus	Europe	Eastern Europe
5	Belgium	Europe	Western Europe

The interface also includes a toolbar with various icons and a status bar that reads 'Total rows loaded: 5'.

4. ACTIVITY ANSWER: SELECT DATA / LIMIT

```
SELECT Name,  
        Continent,  
        Region  
FROM    Country  
  
WHERE   Continent = 'Europe'  
ORDER BY Name  
LIMIT 5;
```

5. OFFSET

```
SELECT *
FROM Country
ORDER BY SurfaceArea
LIMIT 10;
```

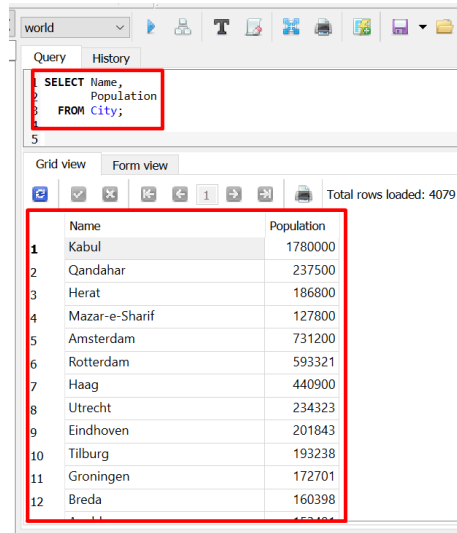
Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectz	GNP	GNPOLD	LocalName
1	VAT	Europe	Southern Europe	0.4	1929	1000	NULL	9	NULL	Santa Sede/Città
2	MCO	Europe	Western Europe	1.5	1861	34000	78.8	776	NULL	Monaco
3	GIB	Europe	Southern Europe	6	NULL	25000	79	258	NULL	Gibraltar
4	TKL	Oceania	Polynesia	12	NULL	2000	NULL	0	NULL	Tokelau
5	CCK	Oceania	Australia and New Zealand	14	NULL	600	NULL	0	NULL	Cocos (Keeling) Is
6	UMI	Oceania	Micronesia/Caribbean	16	NULL	NULL	NULL	0	NULL	United States Min
7	MAC	Asia	Eastern Asia	18	NULL	473000	81.6	5749	5940	Macau/Aome
8	NRU	Oceania	Micronesia	21	1968	12000	60.8	197	NULL	Naoero/Nauru
9	TUV	Oceania	Polynesia	26	1978	12000	66.3	6	NULL	Tuvalu
10	NFK	Oceania	Australia and New Zealand	36	NULL	2000	NULL	0	NULL	Norfolk Island

```
1 SELECT *
2 FROM Country
3 ORDER BY SurfaceArea
4 LIMIT 10 OFFSET 3;
```

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectz	GNP	GNPOLD	LocalName
1	TKL	Oceania	Polynesia	12	NULL	2000	NULL	0	NULL	Tokelau
2	CCK	Oceania	Australia and New Zealand	14	NULL	600	NULL	0	NULL	Cocos (Keelir
3	UMI	Oceania	Micronesia/Caribbean	16	NULL	NULL	NULL	0	NULL	United State
4	MAC	Asia	Eastern Asia	18	NULL	473000	81.6	5749	5940	Macau/Aome
5	NRU	Oceania	Micronesia	21	1968	12000	60.8	197	NULL	Naoero/Nauru
6	TUV	Oceania	Polynesia	26	1978	12000	66.3	6	NULL	Tuvalu
7	NFK	Oceania	Australia and New Zealand	36	NULL	2000	NULL	0	NULL	Norfolk Islan
8	PCN	Oceania	Polynesia	49	NULL	50	NULL	0	NULL	Pitcairn
9	BMU	North America	North America	53	NULL	65000	76.9	2328	2190	Bermuda
10	BVT	Antarctica	Antarctica	59	NULL	NULL	NULL	0	NULL	Bouvetøya

```
SELECT *
FROM Country
ORDER BY SurfaceArea
LIMIT 10 OFFSET 3;
```

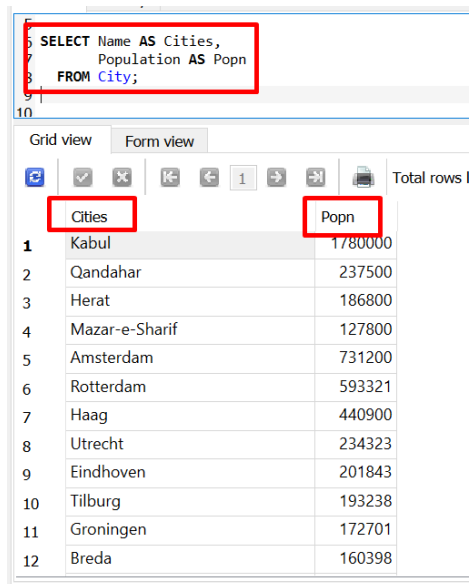
R. AS



```
1 SELECT Name,  
2     Population  
3 FROM City;  
4  
5
```

	Name	Population
1	Kabul	1780000
2	Qandahar	237500
3	Herat	186800
4	Mazar-e-Sharif	127800
5	Amsterdam	731200
6	Rotterdam	593321
7	Haag	440900
8	Utrecht	234323
9	Eindhoven	201843
10	Tilburg	193238
11	Groningen	172701
12	Breda	160398

```
SELECT Name,  
        Population  
FROM City;
```



```
1 SELECT Name AS Cities,  
2     Population AS Popn  
3 FROM City;  
4  
5
```

	Cities	Popn
1	Kabul	1780000
2	Qandahar	237500
3	Herat	186800
4	Mazar-e-Sharif	127800
5	Amsterdam	731200
6	Rotterdam	593321
7	Haag	440900
8	Utrecht	234323
9	Eindhoven	201843
10	Tilburg	193238
11	Groningen	172701
12	Breda	160398

```
SELECT Name AS Cities,  
        Population AS Popn  
FROM City;
```

S. INSERT INTO

ID	Name	CountryCode	District	Population
4063	Elgin	USA	Illinois	89408
4064	Odessa	USA	Texas	89293
4065	Carson	USA	California	89089
4066	Charleston	USA	South Carolina	89063
4067	Charlotte Amalie	VIR	St Thomas	13000
4068	Harare	ZWE	Harare	1410000
4069	Bulawayo	ZWE	Bulawayo	621742
4070	Chitungwiza	ZWE	Harare	274912
4071	Mount Darwin	ZWE	Harare	164362
4072	Mutare	ZWE	Manicaland	131367
4073	Gweru	ZWE	Midlands	128037
4074	Gaza	PSE	Gaza	353632
4075	Khan Yunis	PSE	Khan Yunis	123175
4076	Hebron	PSE	Hebron	119401
4077	Jabaliya	PSE	North Gaza	113901
4078	Nabulus	PSE	Nabulus	100231
4079	Rafah	PSE	Rafah	92020

previously we only had 4079 rows in the City Table

(there was no Singapore)

```
INSERT INTO City (
    Name,
    CountryCode
)
VALUES (
    'Singapore',
    'SGP2'
);
```

Total rows loaded: 0

when we ran this code, nothing shows up here...

but it says that the Query has been executed....

Status

[12:50:28] Query finished in 0.006 second(s). Rows affected: 1

[12:50:30] Query finished in 0.004 second(s). Rows affected: 1

The screenshot shows a SQL Server Enterprise Manager interface. On the left, the 'Databases' pane shows a tree view for a database named 'world (SQLite 3)', with a table named 'City' expanded to show its columns: ID, Name, CountryCode, District, and Population. The main pane displays the 'City' table in 'Grid view'. The first row (ID 4080) is highlighted in red and contains the following data: Name: Singapore, CountryCode: SGP2, District: (empty), Population: 0. A red box highlights the 'Total rows loaded: 4080' status bar in the top right. Red text annotations state: 'but now when you check the table...' and 'Row 4080 has been created with Singapore...'. The bottom status bar shows two queries: '[12:51:53] Query finished in 0.012 second(s). Rows affected: 1' and '[12:53:17] Query finished in 0.009 second(s). Rows affected: 1'. The bottom taskbar shows 'SQL editor 1' and 'City (world)'.

ID	Name	CountryCode	District	Population
1	Singapore	SGP2		0
2	Rafah	PSE	Rafah	92020
3	Nablus	PSE	Nablus	100231
4	Jabaliya	PSE	North Gaza	113901
5	Hebron	PSE	Hebron	119401
6	Khan Yunis	PSE	Khan Yunis	123175
7	Gaza	PSE	Gaza	353632
8	Gweru	ZWE	Midlands	128037
9	Mutare	ZWE	Manicaland	131367
10	Mount Darwin	ZWE	Harare	164362
11	Chitungwiza	ZWE	Harare	274912
12	Bulawayo	ZWE	Bulawayo	621742
13	Harare	ZWE	Harare	1410000
14	Charlotte Amalie	VIR	St Thomas	13000
15	Charleston	USA	South Carolina	89063
16	Carson	USA	California	89089
17	Odessa	USA	Texas	89293
18	Elgin	USA	Illinois	80408

```

INSERT INTO City (
    Name,
    CountryCode
)
VALUES (
    'Singapore',
    'SGP2'
);

```

T. UPDATE

```
14
15
16 UPDATE City
17 SET Population = 5700000
18 WHERE CountryCode = 'SGP2';
19
20
21
22
```

Grid view Form view
Total rows loaded: 0

Status
[12:53:17] Query finished in 0.009 second(s). Rows affected: 1
[12:56:24] Query finished in 0.009 second(s). Rows affected: 1

likewise when we ran this code,....nothing happens here....

but the Query shows its finished....

ID	Name	CountryCode	District	Population
1	4080	Singapore	SGP2	5700000
2	4079	Rafah	PSE	92020
3	4078	Nablu	PSE	100231
4	4077	Jabaliya	PSE	113901
5	4076	Hebron	PSE	119401
6	4075	Khan Yunis	PSE	123175
7	4074	Gaza	PSE	353632
8	4073	Gweru	ZWE	128037
9	4072	Mutare	ZWE	131367
10	4071	Mount Darwin	ZWE	164362
11	4070	Chitungwiza	ZWE	274912
12	4069	Bulawayo	ZWE	621742
13	4068	Harare	ZWE	1410000
14	4067	Charlotte Amalie	VIR	13000
15	4066	Charleston	USA	89063
16	4065	Carson	USA	89089
17	4064	Odessa	USA	89293
18	4063	Elgin	USA	89408

Grid view Form view
Filter data Total rows loaded: 4080

previously this was showing empty

but now is showing a population of 5,700,000

```
UPDATE City
SET Population = 5700000
WHERE CountryCode = 'SGP2';
```

U. DELETE

The screenshot shows a query editor with the following SQL statement highlighted in a red box:

```
DELETE FROM City  
WHERE ID = '4080';
```

Below the query editor, the status bar shows the following messages:

- [12:56:24] Query finished in 0.009 second(s). Rows affected: 1
- [13:00:24] Query finished in 0.005 second(s). Rows affected: 1

Red arrows point from the SQL statement to the status bar messages. Red text annotations are present:

- now when we ran this code, nothing appears here again...
- but the Query shows completed....

The screenshot shows a grid view of a table with the following columns: ID, Name, CountryCode, District, and Population. The total rows loaded is 4079. Row 4079 is highlighted in a red box.

ID	Name	CountryCode	District	Population
1	4079 Rafah	PSE	Rafah	92020
2	4078 Nablus	PSE	Nablus	100231
3	4077 Jabaliya	PSE	North Gaza	113901
4	4076 Hebron	PSE	Hebron	119401
5	4075 Khan Yunis	PSE	Khan Yunis	123175
6	4074 Gaza	PSE	Gaza	353632
7	4073 Gweru	ZWE	Midlands	128037
8	4072 Mutare	ZWE	Manicaland	131367
9	4071 Mount Darwin	ZWE	Harare	164362
10	4070 Chitungwiza	ZWE	Harare	274912
11	4069 Bulawayo	ZWE	Bulawayo	621742
12	4068 Harare	ZWE	Harare	1410000
13	4067 Charlotte Amalie	VIR	St Thomas	13000
14	4066 Charleston	USA	South Carolina	89063
15	4065 Carson	USA	California	89089
16	4064 Odessa	USA	Texas	89293
17	4063 Elgin	USA	Illinois	89408
18	4062 Kenoche	USA	Wisconsin	89447

Red text annotations are present:

- however we see that row 4080 has been deleted...

```
DELETE FROM City  
WHERE ID = '4080';
```

V. AGGREGATE FUNCTIONS

The screenshot shows a database query tool interface. At the top, there are tabs for 'Query' and 'History'. The 'Query' tab is active, displaying a SQL query in a text editor:

```
1 SELECT Region  
2 FROM Country;
```

The query is highlighted with a red box. To the right of the query editor, red text reads: "this gives us 239 rows....". Below the query editor, there are tabs for 'Grid view' and 'Form view'. The 'Grid view' tab is active, showing a table of results. The table has a single column labeled 'Region'. The first row is highlighted with a blue background and contains the text 'Southern and Central Asia'. The table is also highlighted with a red box. To the right of the table, a status bar displays 'Total rows loaded: 239', which is also highlighted with a red box.

Region
1 Southern and Central Asia
2 Western Europe
3 Caribbean
4 Southern Europe
5 Northern Africa
6 Polynesia
7 Southern Europe
8 Central Africa
9 Caribbean
10 Caribbean

1. COUNT

The screenshot shows a SQL query editor with two tabs: 'Query' and 'History'. The 'Query' tab is active, displaying the following SQL query:

```
1 SELECT COUNT(Region)
2 FROM Country;
```

The query is highlighted with a red box. Below the query editor, there are two view options: 'Grid view' and 'Form view'. The 'Form view' is selected. Below the view options, there is a toolbar with various icons (refresh, check, close, back, forward, search, print) and a status bar that reads 'Total rows loaded: 1'. Below the toolbar, there is a table with one row and two columns:

1	COUNT(Region)	239
---	---------------	-----

The table is also highlighted with a red box.

```
SELECT COUNT(Region)
FROM Country;
```

world

Query History

```

1 SELECT COUNT(Population)
2 FROM Country;
3
4
5

```

Grid view Form view

COUNT(Population)

232

why???

239 - 7 nulls = 232... that's why....

Total rows loaded: 239

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpect	GNP
1	ATA	Antarctica	Antarctica	13120000	NULL	NULL	NULL	
2	BVT	Bouvet Island	Antarctica	59	NULL	NULL	NULL	
3	IOT	British Indian Ocean Territory	Africa	78	NULL	NULL	NULL	
4	SGS	South Georgia and the South Sandwich Islands	Antarctica	3903	NULL	NULL	NULL	
5	HMD	Heard Island and McDonald Islands	Antarctica	359	NULL	NULL	NULL	
6	ATF	French Southern territories	Antarctica	7780	NULL	NULL	NULL	
7	UMI	United States Minor Outlying Islands	Oceania	16	NULL	NULL	NULL	
8	PCN	Pitcairn	Oceania	49	NULL	50	NULL	
9	CCK	Cocos (Keeling) Islands	Oceania	14	NULL	600	NULL	
10	VAT	Holy See (Vatican City State)	Europe	0.4	1929	1000	NULL	
11	FLK	Falkland Islands	South America	12173	NULL	2000	NULL	
12	NIU	Niue	Oceania	260	NULL	2000	NULL	
13	NFK	Norfolk Island	Oceania	36	NULL	2000	NULL	
14	TKL	Tokelau	Oceania	12	NULL	2000	NULL	
15	CXR	Christmas Island	Australia and New Zealand	135	NULL	2500	NULL	
16	SJM	Svalbard and Jan Mayen	Nordic Countries	62422	NULL	3200	NULL	
17	SHN	Saint Helena	Western Africa	314	NULL	6000	76.8	

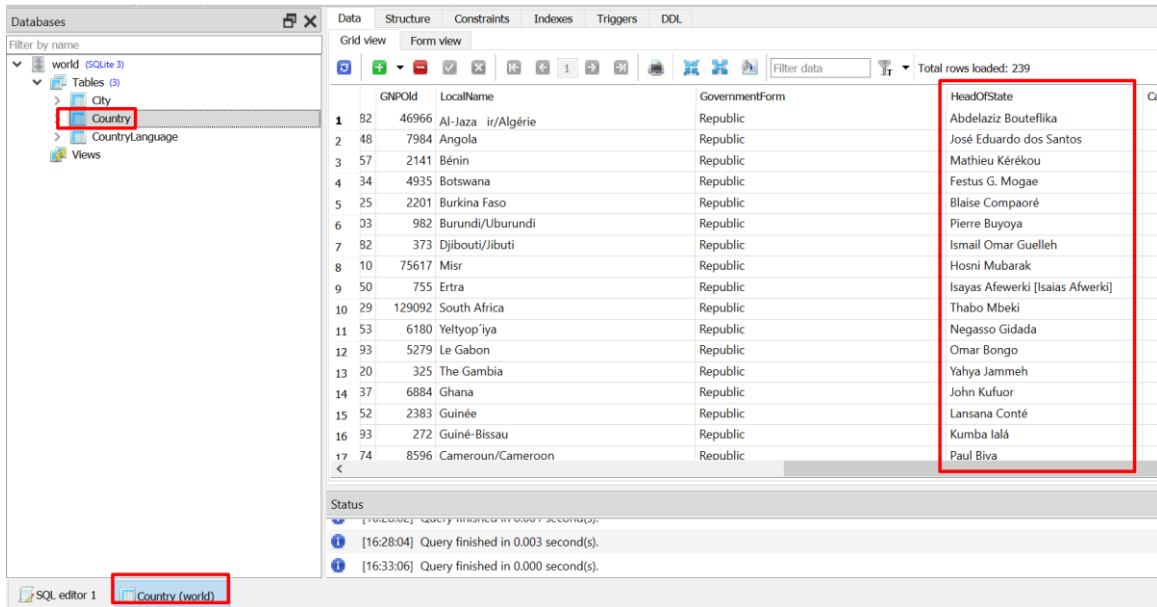
Column: Name
Data type: TEXT
Table: Country
ROWID: 204

a) *ACTIVITY QUESTION: AGGREGATE FUNCTIONS*

- Count the Number of Distinct Head Of State

The screenshot shows a database query tool interface. At the top, there are tabs for 'Query' and 'History'. The 'Query' tab is active, displaying a SQL query: `1 SELECT COUNT(DISTINCT HeadOfState)`, `2 FROM Country;`, and `3`. Below the query editor, there are tabs for 'Grid view' and 'Form view', with 'Grid view' selected. A toolbar contains various icons for navigation and execution, including a refresh icon, a checkmark, a close icon, left and right arrow icons, a page number '1', and a print icon. To the right of the toolbar, it says 'Total rows loaded: 1'. Below the toolbar, a grid view shows the result of the query: a single row with the value '179' under the column header 'COUNT(DISTINCT HeadOfState)'. The row number '1' is visible in the first column of the grid.

	COUNT(DISTINCT HeadOfState)
1	179



b) ACTIVITY ANSWER: AGGREGATE FUNCTIONS

```
SELECT COUNT(DISTINCT HeadOfState)
FROM Country;
```

c) *ACTIVITY QUESTION: AGGREGATE FUNCTIONS*

- Count the Number of Country where Population > 10,000,000 in Europe

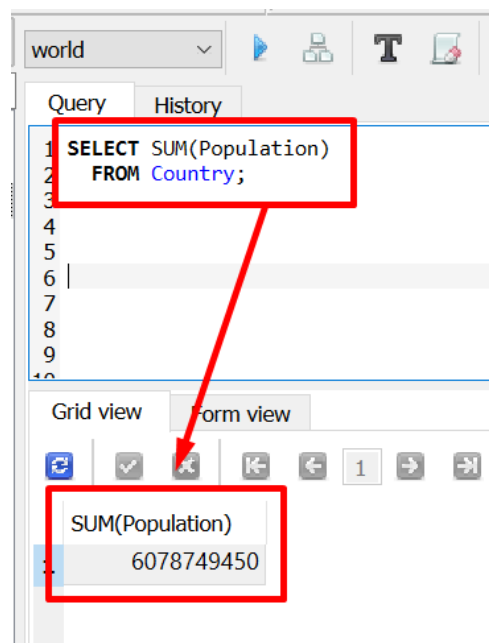
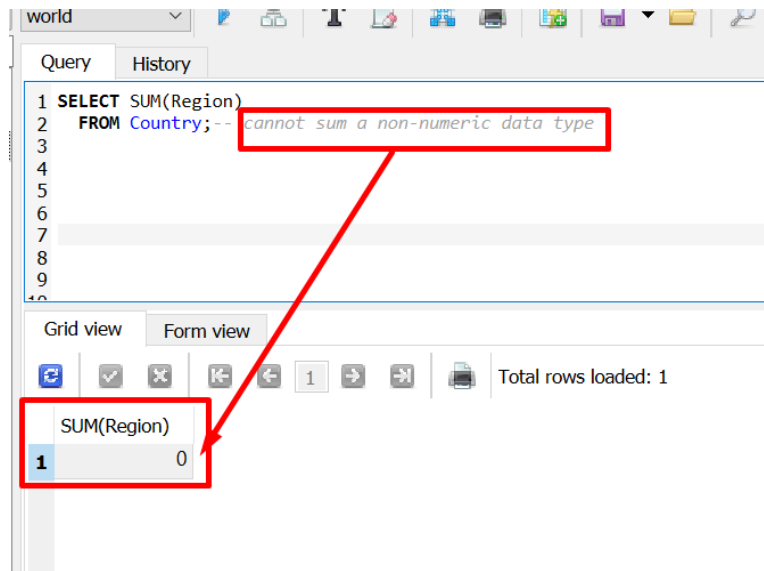
The screenshot shows a database query tool interface. At the top, there are tabs for 'Query' and 'History'. Below the tabs, a SQL query is entered in a text area, highlighted with a red box. The query is: `SELECT COUNT(Name) AS [Country Count:] FROM Country WHERE (Population > 10000000 AND Continent = 'Europe');`. Below the query area, there are tabs for 'Grid view' and 'Form view'. Under 'Grid view', there is a toolbar with various icons. Below the toolbar, a table is displayed, also highlighted with a red box. The table has one column labeled 'Country Count:' and one row with the value '16'.

	Country Count:
1	16

d) *ACTIVITY ANSWER: AGGREGATE FUNCTIONS*

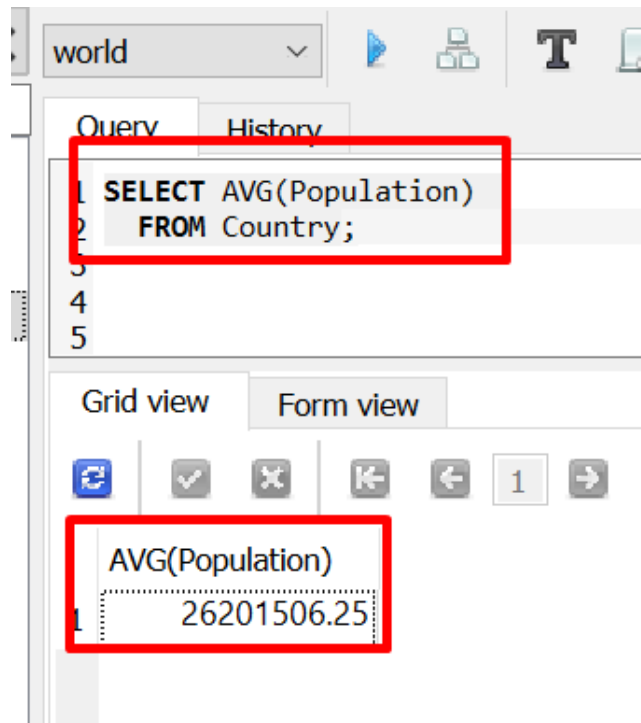
```
SELECT COUNT(Name) AS [Country Count:]  
FROM Country  
WHERE (Population > 10000000 AND Continent = 'Europe');
```

2. SUM



```
SELECT SUM(Population)
FROM Country;
```

3. AVERAGE

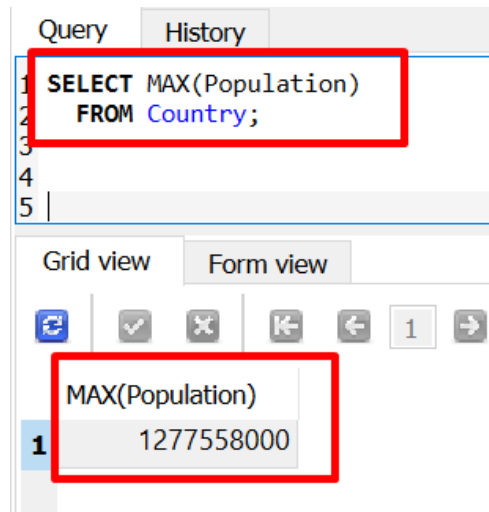


6078749450 ÷ 232 =

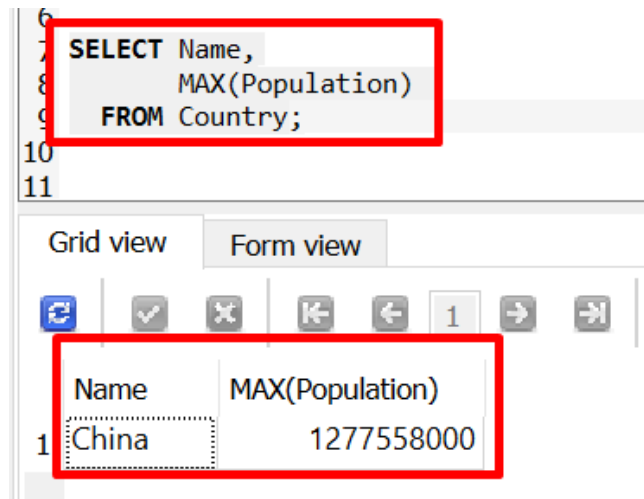
26,201,506.25

```
SELECT AVG(Population)
FROM Country;
```

4. MAX

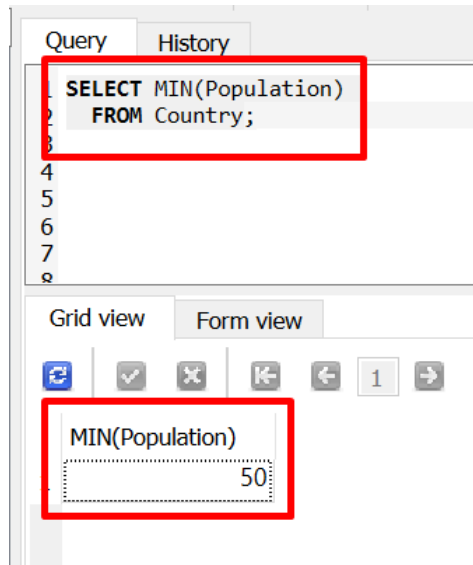


```
SELECT MAX(Population)  
FROM Country;
```

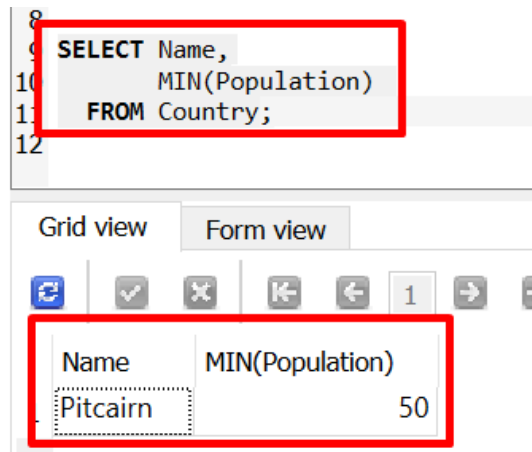


```
SELECT Name,  
MAX(Population)  
FROM Country;
```


5. MIN



```
SELECT MIN(Population)
FROM Country;
```



```
SELECT Name,
MIN(Population)
FROM Country;
```

6. COMBINATIONS

The screenshot shows a database query editor with two tabs: 'Query' and 'History'. The 'Query' tab is active, displaying the following SQL query:

```
SELECT SUM(Population),  
COUNT(Population),  
AVG(Population)  
FROM Country;
```

Below the query editor, there are two view options: 'Grid view' and 'Form view'. The 'Grid view' is selected, showing a table with the following data:

SUM(Population)	COUNT(Population)	AVG(Population)
6078749450	232	26201506.25









The interface also includes a toolbar with various icons (refresh, check, close, left arrow, right arrow, page number 1, double right arrow, print) and a status bar indicating 'Total rows loaded: 1'.

```
SELECT SUM(Population),  
COUNT(Population),  
AVG(Population)  
FROM Country;
```

Query History

```
1 SELECT COUNT(DISTINCT Region),  
2     COUNT(Region)  
3 FROM Country;  
4
```

Grid view Form view

     1    Total rows loaded: 1

	COUNT(DISTINCT Region)	COUNT(Region)
1	25	239

```
SELECT COUNT(DISTINCT Region),  
       COUNT(Region)  
FROM   Country;
```

W. GROUP BY

<https://www.alvinang.sg/s/world.db>

```
Query History
1 SELECT Region,
2     COUNT(Name),
3     SUM(Population)
4 FROM Country
5 GROUP BY Region;
6
7 -- Notice how 'SELECT Region' must be included in order to 'GROUP BY Region'
8
```

Grid view Form view

Total rows loaded: 25

	Region	COUNT(Name)	SUM(Population)
1	Antarctica	5	NULL
2	Australia and New Zealand	5	22753100
3	Baltic Countries	3	7561900
4	British Islands	2	63398500
5	Caribbean	24	38140000
6	Central Africa	9	95652000
7	Central America	8	135221000
8	Eastern Africa	20	246999000
9	Eastern Asia	8	1507328000
10	Eastern Europe	10	307026000
11	Melanesia	5	6472000

```
SELECT    Region,
          COUNT (Name),
          SUM   (Population)
FROM      Country
GROUP BY  Region;
```

-- Notice how 'SELECT Region' must be included in order to 'GROUP BY Region'

Query History

```

1 SELECT 
2     COUNT(Name)
3     SUM(Population)
4 FROM Country
5 GROUP BY Region;
6
7 -- Notice how 'SELECT Region' must be included in order to 'G
8

```

Grid view Form view

Total rows loaded: 25

	COUNT(Name)	SUM(Population)
1	5	NULL
2	5	22753100
3	3	7561900
4	2	63398500
5	24	38140000
6	9	95652000
7	8	135221000
8	20	246999000
9	8	1507328000
10	10	307026000
11	5	6472000

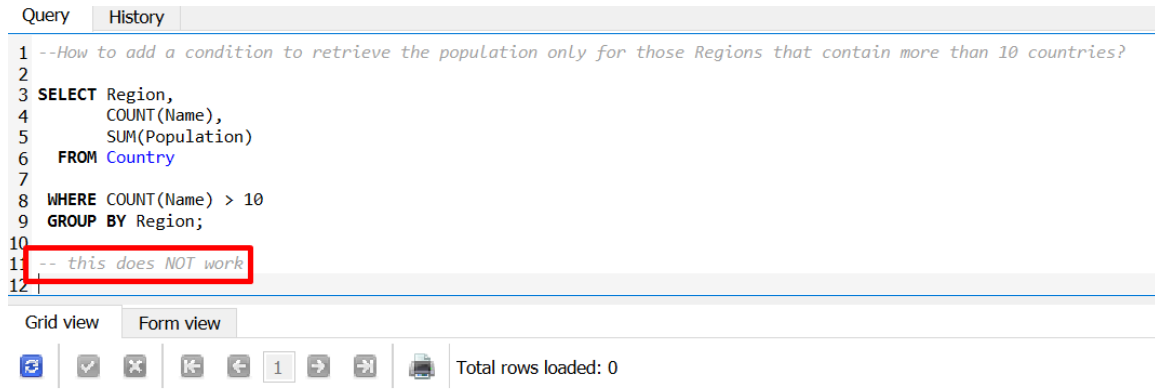
if we remove the Region word

notice the Region column disappears

and we won't know what we are grouping by...

1. HAVING

--How to add a condition to retrieve the population only for those Regions that contain more than 10 countries?



```
Query History
1 --How to add a condition to retrieve the population only for those Regions that contain more than 10 countries?
2
3 SELECT Region,
4     COUNT(Name),
5     SUM(Population)
6 FROM Country
7
8 WHERE COUNT(Name) > 10
9 GROUP BY Region;
10
11 -- this does NOT work
12
```

Grid view Form view

Total rows loaded: 0

you can see that it gives a blank result...

```

1 SELECT Region,
2     COUNT(Name),
3     SUM(Population)
4 FROM Country
5
6 GROUP BY Region
7 HAVING COUNT(Name) > 10;
8
9 -- this does works now!
10

```

Grid view Form view

Total rows loaded: 8

	Region	COUNT(Name)	SUM(Population)
1	Caribbean	24	38140000
2	Eastern Africa	20	246999000
3	Middle East	18	188380700
4	South America	14	345780000
5	Southeast Asia	11	518541000
6	Southern Europe	15	144674200
7	Southern and Central Asia	14	1490776000
8	Western Africa	17	221672000

```

SELECT      Region,
            COUNT(Name),
            SUM(Population)
FROM        Country

GROUP BY   Region
HAVING     COUNT(Name) > 10;

-- this does work now!

```

a) *ACTIVITY QUESTION: GROUPBY AND HAVING*

- Retrieve the Average Population (from the City Table),
- Group by the Country code having more than 270,000 average population

The screenshot displays a database query interface. At the top, there are tabs for 'Query' and 'History'. The SQL query is as follows:

```
1 SELECT Countrycode,  
2     Population,  
3     ROUND(AVG(Population), 0),  
4     COUNT(Name)  
5 FROM City  
6  
7 GROUP BY CountryCode  
8 HAVING AVG(Population) > 270000  
9 ORDER BY CountryCode;
```

Below the query, there are tabs for 'Grid view' and 'Form view'. A toolbar contains various icons for refreshing, saving, deleting, navigating, and printing. The text 'Total rows loaded: 106' is visible. The results are shown in a grid table:

	Countrycode	Population	ROUND(AVG(Population), 0)	COUNT(Name)
1	AFG	1780000	583025	4
2	AGO	2022000	512320	5
3	ARE	669181	345667	5
4	ARG	2982146	350817	57
5	ARM	1248700	544367	3
6	AUS	3276207	808119	14
7	AUT	1608144	397379	6
8	AZE	1787800	616000	4
9	BDI	300000	300000	1
10	BEA	821000	109667	3

b) ACTIVITY ANSWER: GROUPBY AND HAVING

```
SELECT      Countrycode,  
            Population,  
ROUND      (AVG(Population), 0),  
COUNT     (Name)  
  
FROM City  
GROUP BY   CountryCode  
HAVING AVG (Population) > 270000  
ORDER BY   CountryCode;
```

2. WHERE

The screenshot shows a SQL query editor with the following query:

```
1 SELECT Region,  
2     COUNT(Name),  
3     SUM(Population)  
4 FROM Country  
5  
6 --WHERE Region > 'F*'  
7  
8 GROUP BY Region  
9 HAVING COUNT(Name) > 10;  
10
```

Red annotations explain the effect of the WHERE clause:

- if this was run....
it will display F onwards..
- meaning, it won't display
Caribbean and
Eastern Africa...

The results are displayed in a grid view with 8 rows:

Region	COUNT(Name)	SUM(Population)
1 Caribbean	24	38140000
2 Eastern Africa	20	246999000
3 Middle East	18	188380700
4 South America	14	345780000
5 Southeast Asia	11	518541000
6 Southern Europe	15	144674200
7 Southern and Central Asia	14	1490776000
8 Western Africa	17	221672000

Query History

```

1 SELECT Region,
2     COUNT(Name),
3     SUM(Population)
4 FROM Country
5
6 WHERE Region > 'F*'
7
8 GROUP BY Region
9 HAVING COUNT(Name) > 10;
10

```

Grid view Form view

Total rows loaded: 6

	Region	COUNT(Name)	SUM(Population)
1	Middle East	18	188380700
2	South America	14	345780000
3	Southeast Asia	11	518541000
4	Southern Europe	15	144674200
5	Southern and Central Asia	14	1490776000
6	Western Africa	17	221672000

```

SELECT    Region,
          COUNT(Name),
          SUM(Population)
FROM      Country

WHERE     Region > 'F*'

GROUP BY Region
HAVING   COUNT(Name) > 10;

```









Query History

```

1 SELECT Region,
2     COUNT(Name),
3     SUM(Population)
4 FROM Country
5
6 WHERE Region > 'S*'
7
8 GROUP BY Region
9 HAVING COUNT(Name) > 10;
10

```

Grid view Form view






1



 Total rows loaded: 5

	Region	COUNT(Name)	SUM(Population)
1	South America	14	345780000
2	Southeast Asia	11	518541000
3	Southern Europe	15	144674200
4	Southern and Central Asia	14	1490776000
5	Western Africa	17	221672000

```

SELECT      Region,
            COUNT(Name),
            SUM(Population)
FROM        Country
WHERE       Region > 'S*'

GROUP BY   Region
HAVING     COUNT(Name) > 10;

```

X. JOINS

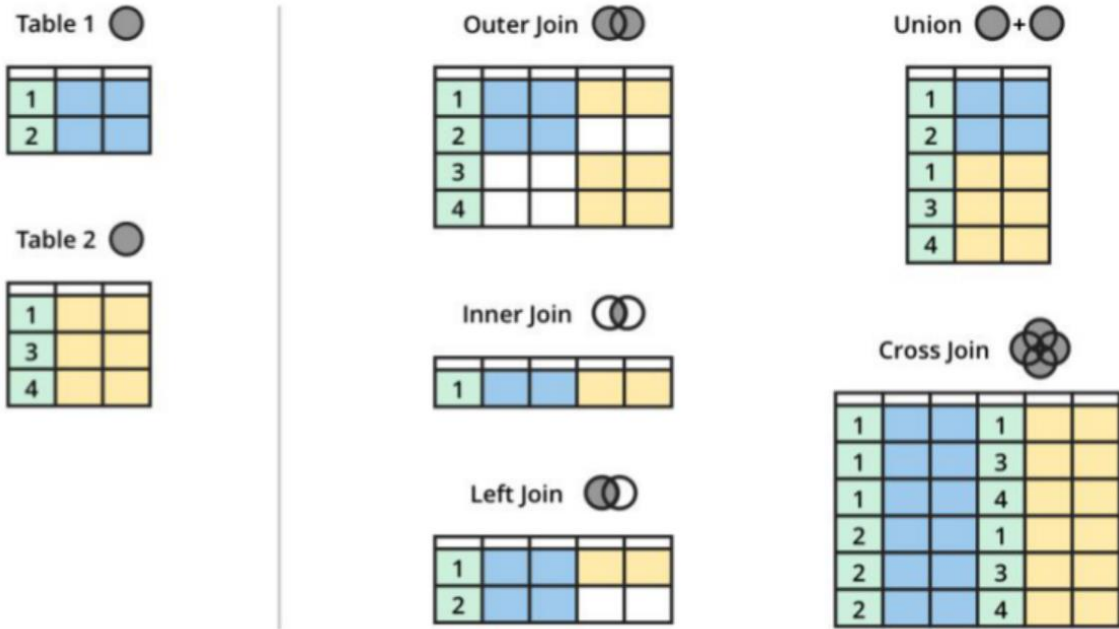


TABLE 1	INNER JOIN					OUTER JOIN						
ID	Colour	Gender	ID	Colour	Gender	Colour2	Item	ID	Colour	Gender	Colour2	Item
1	Red	F	1	Red	F	Red	Dress	1	Red	F	Red	Dress
2	Blue	M						2	Blue	M		
						Blue	Jeans	3			Blue	Jeans
						Green	Shoes	4			Green	Shoes

TABLE 2	LEFT JOIN					RIGHT JOIN						
ID	Colour	Item	ID	Colour	Gender	Colour2	Item	ID	Colour	Gender	Colour2	Item
1	Red	Dress	1	Red	F	Red	Dress	1	Red	F	Red	Dress
3	Blue	Jeans						3			Blue	Jeans
4	Green	Shoes						4			Green	Shoes

CROSS JOIN							UNION		
ID	Colour	Gender	ID2	Colour2	Item	ID	Colour	Column2	
1	Red	F	1	Red	Dress	1	Red	F	
1	Red	F	3	Blue	Jeans	2	Blue	M	
1	Red	F	4	Green	Shoes	1	Red	Dress	
2	Blue	M	1	Red	Dress	3	Blue	Jeans	
2	Blue	M	3	Blue	Jeans	4	Green	Shoes	
2	Blue	M	4	Green	Shoes				

1. INNER JOIN

a) Test.DB

File can be found here: <https://www.alvinang.sg/s/test.db>

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure for 'test (SQLite 3)', including tables 'customer' and 'item', and table 'sale'. The 'sale' table is selected, and its columns are listed: id, item_id, customer_id, date, quantity, and price. A red box highlights the 'sale' table in the tree view, with an arrow pointing to the data grid. The data grid shows the following data:

id	item_id	customer_id	date	quantity	price
1	1	1	2009-02-27	3	2995
2	2	2	2009-02-27	1	1995
3	3	1	2009-02-28	1	2995
4	4	4	2009-02-28	2	999
5	5	1	2009-02-28	1	2995

initial view of the Sale table...

The screenshot shows the same database management tool interface. The 'item' table is selected in the tree view, and its columns are listed: id, name, and description. A red box highlights the 'item' table in the tree view, with an arrow pointing to the data grid. The data grid shows the following data:

id	name	description
1	Box of 64 Pixels	64 RGB pixels in a decorative box
2	Sense of Humor	Especially dry. Imported from England.
3	Beauty	Inner beauty. No cosmetic surgery required!
4	Bar Code	Unused. In original packaging.

initial view of the item table...

Query History

```

1 SELECT *
2 FROM Sale
3 INNER JOIN
4 Item ON Sale.Item_id = item.id;
5

```

Grid view Form view

Total rows loaded: 5

id	item_id	customer_id	date	quantity	price	id:1	name	description
1	1	1	2009-02-27	3	2995	1	Box of 64 Pixels	64 RGB pixels in a decorative box
2	2	2	2009-02-27	1	1995	2	Sense of Humor	Especially dry. Imported from England.
3	3	1	2009-02-28	1	2995	1	Box of 64 Pixels	64 RGB pixels in a decorative box
4	4	4	2009-02-28	2	999	4	Bar Code	Unused. In original packaging.
5	5	1	2009-02-28	1	2995	1	Box of 64 Pixels	64 RGB pixels in a decorative box

from the Sale table...

from the item table...

we are matching this column to this column

We see that the column that is inner joined is 1-2-1-4-1

```

SELECT *
FROM Sale
INNER JOIN Item
ON Sale.Item_id = item.id;

```

b) World.DB

<https://www.alvinang.sg/s/world.db>

(1) ACTIVITY QUESTION: INNER JOIN

- Inner Join Country and City Tables.
- Retrieve the Country Name (as Country) and City Name (as City)

The screenshot displays a database management interface with the following components:

- Left Panel (Database Explorer):** Shows the 'world (SQLite 3)' database structure. The 'City' table has columns: ID, Name, CountryCode, District, Population. The 'Country' table has columns: Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, Code2. Red boxes highlight 'City', 'CountryCode', 'Country', and 'Code'.
- Query Editor:** Contains the SQL query:

```
1 SELECT Country.Name AS Country,  
2 City.Name AS City  
3 FROM City  
4 INNER JOIN  
5 Country ON City.CountryCode = Country.Code;  
6
```
- Grid view:** Shows the results of the query. A red box highlights the first 10 rows:

	Country	City
1	Afghanistan	Kabul
2	Afghanistan	Qandahar
3	Afghanistan	Herat
4	Afghanistan	Mazar-e-Sharif
5	Netherlands	Amsterdam
6	Netherlands	Rotterdam
7	Netherlands	Haag
8	Netherlands	Utrecht
9	Netherlands	Eindhoven
10	Netherlands	Tilburg
- Status Bar:** Shows query execution logs, including: "[19:08:31] Committed changes for table 'sale' successfully." and "[20:13:16] Query finished in 0.004 second(s)."

(2) ACTIVITY ANSWER: INNER JOIN

```
SELECT Country.Name AS Country,  
       City.Name AS City  
FROM   City
```

```
INNER JOIN  
Country ON City.CountryCode = Country.Code;
```

(3) ACTIVITY QUESTION: INNER JOIN

- Retrieve the Country Name and Number of Cities for each country, group by Country Code.
- Order by the Number of Cities in Descending Order, followed by Country Name.

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure for 'world (SQLite 3)', including tables 'City' and 'Country'. The 'City' table has columns: ID, Name, CountryCode, District, and Population. The 'Country' table has columns: Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, and Code2. On the right, the 'Query' tab shows the following SQL query:

```
SELECT Country.Name,  
COUNT(City.Name)  
FROM Country  
INNER JOIN  
City ON Country.Code = City.CountryCode  
GROUP BY Country.Code  
ORDER BY COUNT(City.Name) DESC,  
Country.Name ASC;
```

Below the query, the 'Grid view' shows the results of the query. The results are displayed in a table with two columns: 'Name' and 'COUNT(City.Name)'. The results are ordered by the number of cities in descending order, followed by the country name in ascending order. The first 10 rows are highlighted with a red box:

	Name	COUNT(City.Name)
1	China	363
2	India	341
3	United States	274
4	Brazil	250
5	Japan	248
6	Russian Federation	189
7	Mexico	173
8	Philippines	136
9	Germany	93
10	Indonesia	85

At the bottom, the 'Status' tab shows the execution history of the query, indicating that the query finished in 0.012 second(s).

(4) ACTIVITY ANSWER: INNER JOIN

```
SELECT    Country.Name,    COUNT(City.Name)
FROM      Country
INNER JOIN
          City ON Country.Code = City.CountryCode

GROUP BY  Country.Code
ORDER BY  COUNT(City.Name) DESC, Country.Name ASC;
```

ABOUT DR. ALVIN ANG



Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He was previously a Professor, Scientist and Financial Consultant. Currently, he owns multiple startups and is a Personal/Business Advisor.

More about him at www.AlvinAng.sg