# Practice2

December 5, 2025

[1]:
```
%reload_ext sql
%config SqlMagic.autopandas = True
%config SqlMagic.feedback = False
%config SqlMagic.style = 'default'  # This is the key fix
```

[2]:
```
# # First, close all SQL connections
# %sql --close sqlite:///practice.db
```

[3]:
```
# Delete the database file
!rm -f practice1.db
```

[4]:
```
# Connect to an SQLite database (creates if doesn't exist)
%sql sqlite:///practice1.db
```

[5]:
```
%%sql
create table Customers (CustomerID integer primary key, CustomerName text, City␣
 ↪text);
```

```
 * sqlite:///practice1.db
```

[5]:
```
Empty DataFrame
Columns: []
Index: []
```

[6]:
```
%%sql
insert into Customers(CustomerID,CustomerName,City) values␣
 ↪(1,'ankit','kanpur'),(2,'kiio','pathankoat')
```

```
 * sqlite:///practice1.db
```

[6]:
```
Empty DataFrame
Columns: []
Index: []
```

[7]:
```
%sql select * from Customers;
```

```
 * sqlite:///practice1.db
```

[7]:
```
   CustomerID CustomerName        City
0           1        ankit      kanpur
1           2         kiio  pathankoat
```

[8]:
```
%sql select [CustomerID],[City] from Customers
```

```
 * sqlite:///practice1.db
```

[8]:
```
   CustomerID        City
0           1      kanpur
1           2  pathankoat
```

[9]:
```
%%sql
insert into Customers(CustomerID,CustomerName,City) values␣
 ↪(3,'arpit','kanpur'),(4,'kiio','delhi')
```

```
 * sqlite:///practice1.db
```

[9]:
```
Empty DataFrame
Columns: []
Index: []
```

[10]:
```
%sql select [CustomerID],CustomerName,[City] from Customers
```

```
 * sqlite:///practice1.db
```

[10]:
```
   CustomerID CustomerName        City
0           1        ankit      kanpur
1           2         kiio  pathankoat
2           3        arpit      kanpur
3           4         kiio       delhi
```

[11]:
```
%sql select distinct CustomerName,City from Customers;
```

```
 * sqlite:///practice1.db
```

[11]:
```
  CustomerName        City
0        ankit      kanpur
1         kiio  pathankoat
2        arpit      kanpur
3         kiio       delhi
```

[12]:
```
%sql select distinct CustomerName from Customers;
```

```
 * sqlite:///practice1.db
```

[12]:
```
  CustomerName
0        ankit
```

```
1         kiio
2        arpit
```

[13]:
```
%sql select CustomerName,sum(CustomerID) sum_ids from Customers where␣
↪CustomerName='kiio' group by CustomerName
```

```
 * sqlite:///practice1.db
```

[13]:

| | CustomerName | sum_ids |
|---|---|---|
| 0 | kiio | 6 |

[14]:
```
%sql select * from Customers order by CustomerName,City
```

```
 * sqlite:///practice1.db
```

[14]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 3 | arpit | kanpur |
| 2 | 4 | kiio | delhi |
| 3 | 2 | kiio | pathankoat |

[15]:
```
%sql select * from Customers order by City ASC,CustomerName Desc
```

```
 * sqlite:///practice1.db
```

[15]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 4 | kiio | delhi |
| 1 | 3 | arpit | kanpur |
| 2 | 1 | ankit | kanpur |
| 3 | 2 | kiio | pathankoat |

[16]:
```
%sql select * from Customers where CustomerName='kiio' and  City like '%h%'
```

```
 * sqlite:///practice1.db
```

[16]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 2 | kiio | pathankoat |
| 1 | 4 | kiio | delhi |

[17]:
```
%sql select * from Customers where not CustomerName='ankit' or  City like '_e%i'
```

```
 * sqlite:///practice1.db
```

[17]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 2 | kiio | pathankoat |
| 1 | 3 | arpit | kanpur |
| 2 | 4 | kiio | delhi |

[18]:
```
%sql select * from Customers where CustomerName='kiio' and  City not like '%e%'
```

```
 * sqlite:///practice1.db
```

[18]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 2 | kiio | pathankoat |

[19]:
```
%sql select * from Customers where CustomerID not between 2 and 3
```

```
 * sqlite:///practice1.db
```

[19]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 4 | kiio | delhi |

[20]:
```
%sql select * from Customers where CustomerID between 2 and 3
```

```
 * sqlite:///practice1.db
```

[20]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 2 | kiio | pathankoat |
| 1 | 3 | arpit | kanpur |

[21]:
```
%sql select * from Customers where CustomerID not in (2,3)
```

```
 * sqlite:///practice1.db
```

[21]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 4 | kiio | delhi |

[22]:
```
%sql select * from Customers where CustomerID in (2,3)
```

```
 * sqlite:///practice1.db
```

[22]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 2 | kiio | pathankoat |
| 1 | 3 | arpit | kanpur |

[23]:
```
%sql select CustomerName from Customers where City is not null
```

```
 * sqlite:///practice1.db
```

[23]:

| | CustomerName |
|---|---|
| 0 | ankit |
| 1 | kiio |
| 2 | arpit |
| 3 | kiio |

[24]:
```
%sql select distinct CustomerName from Customers where City is not null
```

```
 * sqlite:///practice1.db
```

[24]:

| | CustomerName |
|---|---|
| 0 | ankit |
| 1 | kiio |
| 2 | arpit |

[25]:
```
%%sql
update Customers set CustomerName='Summi' where CustomerName='arpit';
update Customers set CustomerName='Summi', City='Hathras' where CustomerID=3;
select * from Customers
```

```
 * sqlite:///practice1.db
```

[25]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | Summi | Hathras |
| 3 | 4 | kiio | delhi |

[26]:
```
%%sql
delete from Customers where CustomerID=3;
select * from Customers
```

```
 * sqlite:///practice1.db
```

[26]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 4 | kiio | delhi |

[27]:
```
%%sql
delete from Customers ;
select * from Customers
```

```
 * sqlite:///practice1.db
```

[27]:
```
Empty DataFrame
Columns: []
Index: []
```

[28]:
```
%sql truncate table Customers -- sqlite does not support truncate
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "truncate": syntax error
```

```
[SQL: truncate table Customers]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

[29]:
```
%sql drop table Customers
```

```
 * sqlite:///practice1.db
```

[29]:
```
Empty DataFrame
Columns: []
Index: []
```

[30]:
```
%sql select * from Customers
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) no such table: Customers
[SQL: select * from Customers]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

[31]:
```
%%sql
create table Customers (CustomerID integer primary key, CustomerName text, City␣
 ↪text);
insert into Customers(CustomerID,CustomerName,City) values␣
 ↪(1,'ankit','kanpur'),(2,'kiio','pathankoat'),(3,'arpit','kanpur'),(4,'kiio','delhi');
 ↪
select * from Customers
```

```
 * sqlite:///practice1.db
```

[31]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[32]:
```
%%sql
select * from Customers;
```

```
 * sqlite:///practice1.db
```

[32]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[33]:
```
%sql select * from Customers order by City limit 2;
```

```
 * sqlite:///practice1.db
```

[33]:
```
   CustomerID CustomerName    City
0           4         kiio   delhi
1           1        ankit  kanpur
```

[34]:
```
%%sql
select CustomerName, sum(CustomerID) from Customers group by CustomerName␣
 ↪having CustomerName='kiio'
```

```
 * sqlite:///practice1.db
```

[34]:
```
  CustomerName  sum(CustomerID)
0         kiio               6
```

[35]:
```
%%sql
select CustomerName, count(CustomerName) from Customers group by CustomerName␣
 ↪having CustomerName='kiio'
```

```
 * sqlite:///practice1.db
```

[35]:
```
  CustomerName  count(CustomerName)
0         kiio                   2
```

[36]:
```
%%sql
select CustomerName, avg(CustomerID) from Customers group by CustomerName␣
 ↪having CustomerName='kiio'
```

```
 * sqlite:///practice1.db
```

[36]:
```
  CustomerName  avg(CustomerID)
0         kiio             3.0
```

[37]:
```
%%sql
select CustomerName, max(CustomerID) [Maximum Customer ID Value]
from Customers group by CustomerName having CustomerName='kiio'
```

```
 * sqlite:///practice1.db
```

[37]:
```
  CustomerName  Maximum Customer ID Value
0         kiio                         4
```

[38]:
```
%%sql
select * from Customers where CustomerName like '%' -- % means zero or more␣
 ↪than zero characters searching
```

```
 * sqlite:///practice1.db
```

[38]:
```
   CustomerID CustomerName        City
0           1        ankit      kanpur
```

```
1           2         kiio  pathankoat
2           3        arpit      kanpur
3           4         kiio       delhi
```

[39]:
```
%%sql
select * from Customers where CustomerName like 'k%'
```

```
 * sqlite:///practice1.db
```

[39]:
```
   CustomerID CustomerName        City
0           2         kiio  pathankoat
1           4         kiio       delhi
```

[40]:
```
%%sql
-- [regex not supported in sqlite] select * from Customers where CustomerName␣
 ↪like '%[o]%'
select * from Customers where CustomerName like '%o%'
```

```
 * sqlite:///practice1.db
```

[40]:
```
   CustomerID CustomerName        City
0           2         kiio  pathankoat
1           4         kiio       delhi
```

[41]:
```
%%sql
-- [regex not supported in sqlite] select * from Customers where CustomerName␣
 ↪like '[k-o]%'
select * from Customers
where CustomerName like 'k%'
   or CustomerName like 'l%'
   or CustomerName like 'm%'
   or CustomerName like 'n%'
   or CustomerName like 'o%'
```

```
 * sqlite:///practice1.db
```

[41]:
```
   CustomerID CustomerName        City
0           2         kiio  pathankoat
1           4         kiio       delhi
```

[42]:
```
%%sql
select * from Customers
where CustomerName glob '[k-o]*'
```

```
 * sqlite:///practice1.db
```

[42]:
```
   CustomerID CustomerName        City
0           2         kiio  pathankoat
```

```
1           4         kiio       delhi
```

[43]:
```
%%sql
select * from Customers where CustomerName glob '[^k-o]*'
```

```
 * sqlite:///practice1.db
```

[43]:
```
   CustomerID CustomerName    City
0           1        ankit  kanpur
1           3        arpit  kanpur
```

[44]:
```
%%sql

-- second max customer id

select CustomerName, CustomerID from Customers
where CustomerID not in (select max(CustomerID) from Customers) order by␣
 ↪CustomerID desc limit 1
```

```
 * sqlite:///practice1.db
```

[44]:
```
  CustomerName  CustomerID
0        arpit           3
```

[45]:
```
%%sql
select CustomerID+5 as "New Customer ID" from Customers
```

```
 * sqlite:///practice1.db
```

[45]:
```
   New Customer ID
0                6
1                7
2                8
3                9
```

[46]:
```
%%sql
select CustomerID+5 as "New Customer ID", CustomerName+' and '+City as␣
 ↪"CustomerName and City Name"
from Customers
```

```
 * sqlite:///practice1.db
```

[46]:
```
   New Customer ID  CustomerName and City Name
0                6                           0
1                7                           0
2                8                           0
3                9                           0
```

SQLite uses the || operator for string concatenation instead of the + operator.

[47]:
```
%%sql
SELECT CustomerID+5 as "New Customer ID",
       CustomerName || ' and ' || City as "CustomerName and City Name"
FROM Customers
```

```
 * sqlite:///practice1.db
```

[47]:

| | New Customer ID | CustomerName and City Name |
|---|---|---|
| 0 | 6 | ankit and kanpur |
| 1 | 7 | kiio and pathankoat |
| 2 | 8 | arpit and kanpur |
| 3 | 9 | kiio and delhi |

[48]:
```
%%sql
create table Orders (CustomerID int primary key, ProductName varchar(20),␣
 ↪ProductID varchar(20));
insert into Orders values (1,'Washing Machine', 567),(2,'TV', 768),(5,'Radio',␣
 ↪786),(6,'Computer', 987);
select * from Customers
```

```
 * sqlite:///practice1.db
```

[48]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[49]:
```
%sql select * from Orders;
```

```
 * sqlite:///practice1.db
```

[49]:

| | CustomerID | ProductName | ProductID |
|---|---|---|---|
| 0 | 1 | Washing Machine | 567 |
| 1 | 2 | TV | 768 |
| 2 | 5 | Radio | 786 |
| 3 | 6 | Computer | 987 |

### 0.0.1 joins

[50]:
```
%%sql
-- Check Customers table structure
PRAGMA table_info(Customers);

-- Check Orders table structure
PRAGMA table_info(Orders);
```

```
 * sqlite:///practice1.db
```

[50]:

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 0 | 0 | CustomerID | INT | 0 | None | 1 |
| 1 | 1 | ProductName | varchar(20) | 0 | None | 0 |
| 2 | 2 | ProductID | varchar(20) | 0 | None | 0 |

[51]:
```
%%sql
select C.CustomerID,C.CustomerName,O.CustomerID,O.ProductName
from Customers C inner join Orders O on C.CustomerID=O.CustomerID
```

```
 * sqlite:///practice1.db
```

[51]:

| | CustomerID | CustomerName | CustomerID | ProductName |
|---|---|---|---|---|
| 0 | 1 | ankit | 1 | Washing Machine |
| 1 | 2 | kiio | 2 | TV |

[52]:
```
%%sql
select C.CustomerID,C.CustomerName,O.CustomerID,O.ProductName
from Customers C right join Orders O on C.CustomerID=O.CustomerID
```

```
 * sqlite:///practice1.db
```

[52]:

| | CustomerID | CustomerName | CustomerID | ProductName |
|---|---|---|---|---|
| 0 | 1.0 | ankit | 1 | Washing Machine |
| 1 | 2.0 | kiio | 2 | TV |
| 2 | NaN | None | 5 | Radio |
| 3 | NaN | None | 6 | Computer |

[53]:
```
%%sql
select C.CustomerID,C.CustomerName,O.CustomerID,O.ProductName
from Customers C left join Orders O on C.CustomerID=O.CustomerID
```

```
 * sqlite:///practice1.db
```

[53]:

| | CustomerID | CustomerName | CustomerID | ProductName |
|---|---|---|---|---|
| 0 | 1 | ankit | 1.0 | Washing Machine |
| 1 | 2 | kiio | 2.0 | TV |
| 2 | 3 | arpit | NaN | None |
| 3 | 4 | kiio | NaN | None |

[54]:
```
%%sql
select C.CustomerID,C.CustomerName,O.CustomerID,O.ProductName
from Customers C full outer join Orders O on C.CustomerID=O.CustomerID
```

```
 * sqlite:///practice1.db
```

[54]:

| | CustomerID | CustomerName | CustomerID | ProductName |
|---|---|---|---|---|
| 0 | 1.0 | ankit | 1.0 | Washing Machine |
| 1 | 2.0 | kiio | 2.0 | TV |
| 2 | 3.0 | arpit | NaN | None |
| 3 | 4.0 | kiio | NaN | None |
| 4 | NaN | None | 5.0 | Radio |
| 5 | NaN | None | 6.0 | Computer |

[55]:
```
%%sql
select C1.CustomerID,C1.CustomerName,C2.CustomerID,C2.City
from Customers C1 inner join Customers C2 on C1.CustomerID=C2.CustomerID
```

```
 * sqlite:///practice1.db
```

[55]:

| | CustomerID | CustomerName | CustomerID | City |
|---|---|---|---|---|
| 0 | 1 | ankit | 1 | kanpur |
| 1 | 2 | kiio | 2 | pathankoat |
| 2 | 3 | arpit | 3 | kanpur |
| 3 | 4 | kiio | 4 | delhi |

[56]:
```
%%sql
select C.CustomerID,C.CustomerName,O.CustomerID,O.ProductName from Customers C␣
 ↪cross join Orders O
```

```
 * sqlite:///practice1.db
```

[56]:

| | CustomerID | CustomerName | CustomerID | ProductName |
|---|---|---|---|---|
| 0 | 1 | ankit | 1 | Washing Machine |
| 1 | 1 | ankit | 2 | TV |
| 2 | 1 | ankit | 5 | Radio |
| 3 | 1 | ankit | 6 | Computer |
| 4 | 2 | kiio | 1 | Washing Machine |
| 5 | 2 | kiio | 2 | TV |
| 6 | 2 | kiio | 5 | Radio |
| 7 | 2 | kiio | 6 | Computer |
| 8 | 3 | arpit | 1 | Washing Machine |
| 9 | 3 | arpit | 2 | TV |
| 10 | 3 | arpit | 5 | Radio |
| 11 | 3 | arpit | 6 | Computer |
| 12 | 4 | kiio | 1 | Washing Machine |
| 13 | 4 | kiio | 2 | TV |
| 14 | 4 | kiio | 5 | Radio |
| 15 | 4 | kiio | 6 | Computer |

### 0.0.2 union,intersection

[57]:
```
%%sql
select CustomerID from Customers where CustomerID not IN (2,4)
union all
select CustomerID from Customers where CustomerID not IN (1,3)
```

```
 * sqlite:///practice1.db
```

[57]:

| | CustomerID |
|---|---|
| 0 | 1 |
| 1 | 3 |
| 2 | 2 |
| 3 | 4 |

[58]:
```
%%sql
select CustomerID from Customers where CustomerID not IN (2,4)
union
select CustomerID from Customers where CustomerID IN (1,3)
```

```
 * sqlite:///practice1.db
```

[58]:

| | CustomerID |
|---|---|
| 0 | 1 |
| 1 | 3 |

[59]:
```
%%sql
select CustomerID from Customers where CustomerID not IN (3,4)
intersect
select CustomerID from Customers where CustomerID IN (1,3)
```

```
 * sqlite:///practice1.db
```

[59]:

| | CustomerID |
|---|---|
| 0 | 1 |

[60]:
```
%sql select * from Customers
```

```
 * sqlite:///practice1.db
```

[60]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[61]:
```
%%sql
```

```
select City,sum(CustomerID) Sum_ID from Customers group by City having City <>␣
↪'kanpur' order by Sum_ID desc
```

```
 * sqlite:///practice1.db
```

[61]:

| | City | Sum_ID |
|---|---|---|
| 0 | delhi | 4 |
| 1 | pathankoat | 2 |

[62]:
```
%%sql
-- The EXISTS operator is used to test for the existence of any record in a␣
↪subquery.

-- The EXISTS operator returns TRUE if the subquery returns one or more records.

select * from Customers where exists (select * from Customers where␣
↪City='kanpur')
```

```
 * sqlite:///practice1.db
```

[62]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[63]:
```
%%sql
select * from Customers where not exists (select * from Customers where␣
↪City='lucknow')
```

```
 * sqlite:///practice1.db
```

[63]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[64]:
```
%%sql
select * from Customers where not exists (select * from Customers where␣
↪City='kanpur')
```

```
 * sqlite:///practice1.db
```

[64]:
```
Empty DataFrame
Columns: []
Index: []
```

[65]:
```
%%sql
select * from Customers where CustomerID= ANY(select CustomerID from Customers␣
 ↪where CustomerName='kiio')
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "select": syntax error
[SQL: select * from Customers where CustomerID= ANY(select CustomerID from
Customers where CustomerName='kiio')]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

SQLite does not support the ANY operator. SQLite has limited support for subquery operators compared to other SQL databases.

[66]:
```
%%sql
SELECT * FROM Customers
WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE CustomerName =␣
 ↪'kiio');
```

```
 * sqlite:///practice1.db
```

[66]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 2 | kiio | pathankoat |
| 1 | 4 | kiio | delhi |

[67]:
```
%%sql
select *  from Customers where CustomerID= ALL(select CustomerID from Customers␣
 ↪where CustomerName='kiio')
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "ALL": syntax error
[SQL: select *  from Customers where CustomerID= ALL(select CustomerID from
Customers where CustomerName='kiio')]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

SQLite does not support the ALL operator. SQLite has limited subquery operator support compared to other SQL databases.

[68]:
```
%%sql
SELECT * FROM Customers
WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE CustomerName =␣
 ↪'kiio')
AND (SELECT COUNT(DISTINCT CustomerID) FROM Customers WHERE CustomerName =␣
 ↪'kiio') = 1;
```

```
 * sqlite:///practice1.db
```

[68]:
```
Empty DataFrame
Columns: []
Index: []
```

[69]:
```
%%sql
SELECT * FROM Customers
WHERE CustomerID = (SELECT CustomerID FROM Customers WHERE CustomerName =␣
 ↪'kiio' LIMIT 1);
```

```
 * sqlite:///practice1.db
```

[69]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 2 | kiio | pathankoat |

insert recrds from one table to another table even if it is not existed or existed but duplicate records are ignored.

[70]:
```
%%sql
select * into [New Customers] from Customers
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "into": syntax error
[SQL: select * into [New Customers] from Customers]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

The error occurs because SQLite doesn't support the SELECT INTO syntax. SQLite uses a different approach for creating a new table from an existing table.

[71]:
```
%%sql
CREATE TABLE "New Customers" AS
SELECT * FROM Customers
```

```
 * sqlite:///practice1.db
```

[71]:

| |
|---|

Empty DataFrame
Columns: []
Index: []

[72]:
```
%%sql
select * from [New Customers]
```

```
 * sqlite:///practice1.db
```

[72]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[73]:
```
%%sql
select * from "New Customers"
```

```
 * sqlite:///practice1.db
```

[73]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[74]:
```
%%sql
create table New_Customers in 'backup.mdb' as select * from Customers
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "in": syntax error
[SQL: create table New_Customers in 'backup.mdb' as select * from Customers]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

The error occurs because the IN 'backup.mdb' clause is not valid SQLite syntax. You're trying to create a table in a different database file, but SQLite doesn't support that syntax.

[75]:
```
%%sql
-- Attach the backup database
ATTACH DATABASE 'backup.mdb' AS backup_db;

-- Create the table in the attached database
CREATE TABLE backup_db.New_Customers AS
SELECT * FROM Customers;

-- Detach when done (optional)
DETACH DATABASE backup_db;
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) table New_Customers already exists
[SQL: -- Create the table in the attached database
CREATE TABLE backup_db.New_Customers AS
SELECT * FROM Customers;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

[76]:
```
%%sql
-- Attach the backup database
ATTACH DATABASE 'backup.mdb' AS backup;

-- Query from the backup database
SELECT * FROM backup.New_Customers;

-- With LIMIT if you have many rows
SELECT * FROM backup.New_Customers LIMIT 10;

-- Count rows
SELECT COUNT(*) FROM backup.New_Customers;
```

```
-- Detach when done
DETACH DATABASE backup;
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) database backup is locked
[SQL: -- Detach when done
DETACH DATABASE backup;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

[77]:
```
%%sql
-- First, check if database is already attached
SELECT name FROM pragma_database_list;
```

```
 * sqlite:///practice1.db
```

[77]:

| | name |
|---|---|
| 0 | main |
| 1 | backup_db |
| 2 | backup |

[78]:
```
%%sql
-- Backup is already attached (as shown in your query), so just query it
SELECT * FROM backup.New_Customers LIMIT 10;
```

```
 * sqlite:///practice1.db
```

[78]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |
| 4 | 1 | ankit | kanpur |
| 5 | 2 | kiio | pathankoat |
| 6 | 3 | arpit | kanpur |
| 7 | 4 | kiio | delhi |
| 8 | 1 | ankit | kanpur |
| 9 | 2 | kiio | pathankoat |

copy the data from one table to another table which should be existed already

[79]:
```
%%sql
insert into new_customers select * from Customers
```

```
 * sqlite:///practice1.db
```

[79]:
```
Empty DataFrame
Columns: []
Index: []
```

[80]:
```
%%sql
select * from new_customers
```

```
 * sqlite:///practice1.db
```

[80]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |
| 4 | 1 | ankit | kanpur |
| 5 | 2 | kiio | pathankoat |
| 6 | 3 | arpit | kanpur |
| 7 | 4 | kiio | delhi |
| 8 | 1 | ankit | kanpur |
| 9 | 2 | kiio | pathankoat |
| 10 | 3 | arpit | kanpur |
| 11 | 4 | kiio | delhi |
| 12 | 1 | ankit | kanpur |
| 13 | 2 | kiio | pathankoat |
| 14 | 3 | arpit | kanpur |
| 15 | 4 | kiio | delhi |
| 16 | 1 | ankit | kanpur |
| 17 | 2 | kiio | pathankoat |
| 18 | 3 | arpit | kanpur |
| 19 | 4 | kiio | delhi |

**CASE Expression** – when both cases are true then both statements executed and else statement executed where when condition is false

– only one column name can be put in all the when conditions

[81]:
```
%%sql
select *,
case
when CustomerName='kiio' then 'this is kiio'
when CustomerName='ankit' then 'this is me'
else 'this is arpit'
end as Msg
from Customers
```

```
 * sqlite:///practice1.db
```

[81]:

| | CustomerID | CustomerName | City | Msg |
|---|---|---|---|---|
| 0 | 1 | ankit | kanpur | this is me |
| 1 | 2 | kiio | pathankoat | this is kiio |
| 2 | 3 | arpit | kanpur | this is arpit |
| 3 | 4 | kiio | delhi | this is kiio |

[82]:
```
%%sql
select *,case when CustomerID in (1,2) then 'this is me' end as msg
into experiment  from Customers
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "into": syntax error
[SQL: select *,case when CustomerID in (1,2) then 'this is me' end as msg
into experiment  from Customers]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

The error occurs because SQLite doesn't support the SELECT INTO syntax. This is a SQL Server/Microsoft SQL syntax, not SQLite.

[83]:
```
%%sql
CREATE TABLE experiment AS
SELECT *,
       CASE
         WHEN CustomerID IN (1, 2) THEN 'this is me'
         ELSE NULL
       END as msg
FROM Customers;
```

```
 * sqlite:///practice1.db
```

[83]:
```
Empty DataFrame
Columns: []
Index: []
```

[84]:
```
%%sql
select * from experiment
```

```
 * sqlite:///practice1.db
```

[84]:

| | CustomerID | CustomerName | City | msg |
|---|---|---|---|---|
| 0 | 1 | ankit | kanpur | this is me |
| 1 | 2 | kiio | pathankoat | this is me |
| 2 | 3 | arpit | kanpur | None |
| 3 | 4 | kiio | delhi | None |

[85]:
```
%%sql
select * from experiment order by (case when msg is null then CustomerName else␣
↪City end) Desc
```

```
 * sqlite:///practice1.db
```

[85]:

| | CustomerID | CustomerName | City | msg |
|---|---|---|---|---|
| 0 | 2 | kiio | pathankoat | this is me |
| 1 | 4 | kiio | delhi | None |

```
2           1        ankit      kanpur  this is me
3           3        arpit      kanpur        None
```

[86]:
```
%%sql
select ISNULL(msg,'this is not null') as Message from experiment
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "ISNULL": syntax error
[SQL: select ISNULL(msg,'this is not null') as Message from experiment]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

The error occurs because ISNULL() is a SQL Server function, not SQLite. SQLite uses different functions for handling NULL values.

[87]:
```
%%sql
SELECT IFNULL(msg, 'this is null') as Message FROM experiment;
```

```
 * sqlite:///practice1.db
```

[87]:
```
        Message
0    this is me
1    this is me
2  this is null
3  this is null
```

[88]:
```
%%sql
select COALESCE(msg,'this is not null') as Message from experiment
```

```
 * sqlite:///practice1.db
```

[88]:
```
            Message
0        this is me
1        this is me
2  this is not null
3  this is not null
```

### 0.0.3 Stored Procedure

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

You can also pass parameters to a stored procedure

[89]:
```
%%sql
create procedure second_max_id
as select CustomerName, CustomerID from Customers where CustomerID not in
(select max(CustomerID) from Customers) order by CustomerID desc;
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "procedure": syntax error
[SQL: create procedure second_max_id
as select CustomerName, CustomerID from Customers where CustomerID not in
(select max(CustomerID) from Customers) order by CustomerID desc;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

[90]:
```
# %%sql
# exec second_max_id
```

– with parameters

[91]:
```
# %%sql
# create procedure kiio_fetch @name nvarchar(30), @city nvarchar(30)
# as
# select * from Customers where CustomerName=@name and City=@city;

# exec kiio_fetch @name='kiio', @city='pathankoat'
```

The error occurs because SQLite doesn't support stored procedures like other database systems (MySQL, SQL Server, etc.).

### 0.0.4 DDL

[92]:
```
%%sql
create database TestDB;
```

```
 * sqlite:///practice1.db
(sqlite3.OperationalError) near "database": syntax error
[SQL: create database TestDB;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

This error occurs because SQLite doesn't support the CREATE DATABASE command that other databases like MySQL or PostgreSQL use. In SQLite, a database is simply a file.

Soluton 1:

[93]:
```
from sqlalchemy import create_engine, text

# This creates a new SQLite database file called TestDB.db
engine = create_engine('sqlite:///TestDB.db')

# Now you can use this connection to create tables
with engine.connect() as conn:
    # Use text() wrapper for raw SQL
    conn.execute(text("CREATE TABLE IF NOT EXISTS my_table (id INTEGER PRIMARY␣
 ↪KEY, name TEXT)"))
    conn.commit()  # Don't forget to commit!
```

Solution 2:

[94]:
```
# First, load the SQL extension
%load_ext sql

# Connect to a new database file
%sql sqlite:///TestDB.db
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

[95]:
```
%%sql

CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY,
    name TEXT,
    email TEXT
);

-- Insert some data
INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com');
```

```
 * sqlite:///TestDB.db
   sqlite:///practice1.db
```

[95]:
```
Empty DataFrame
Columns: []
Index: []
```

[96]:
```
%%sql
select * from users
```

```
 * sqlite:///TestDB.db
   sqlite:///practice1.db
```

[96]:

| | id | name | email |
|---|---|---|---|
| 0 | 1 | John Doe | john@example.com |

[97]:
```
%%sql
DROP DATABASE TestDB;
```

```
 * sqlite:///TestDB.db
   sqlite:///practice1.db
(sqlite3.OperationalError) near "DATABASE": syntax error
[SQL: DROP DATABASE TestDB;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

As with CREATE DATABASE, SQLite also doesn't support DROP DATABASE. In SQLite, databases are files, so you need to delete the file instead.

[98]:
```
import os

# Delete the SQLite database file
if os.path.exists('TestDB.db'):
    os.remove('TestDB.db')
    print("Database deleted successfully")
else:
    print("Database file doesn't exist")
```

```
Database deleted successfully
```

[99]:
```
%%sql
SHOW DATABASES
```

```
 * sqlite:///TestDB.db
   sqlite:///practice1.db
(sqlite3.OperationalError) near "SHOW": syntax error
[SQL: SHOW DATABASES]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

Again, SQLite doesn't support SHOW DATABASES - that's a MySQL command. In SQLite, you work with database files, so you need to check for files or use SQLite-specific commands.

[100]:
```
%load_ext sql

# Connect to a database
%sql sqlite:///TestDB.db

# Check what tables exist in current database
result = %sql SELECT name FROM sqlite_master WHERE type='table'
result
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
 * sqlite:///TestDB.db
   sqlite:///practice1.db
```

[100]:

| | name |
|---|---|
| 0 | my_table |
| 1 | users |

SQLite doesn't support BACKUP DATABASE like SQL Server does.

[101]:
```
import shutil
import datetime

# Backup by copying the .db file
backup_name = f"TestDB_backup_{datetime.datetime.now().
 ↪strftime('%Y%m%d_%H%M%S')}.db"
```

```
shutil.copy2('TestDB.db', backup_name)
print(f"Backup created: {backup_name}")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[101], line 6
      4 # Backup by copying the .db file
      5 backup_name = f"TestDB_backup_{datetime.datetime.now().
 ↪strftime('%Y%m%d_%H%M%S')}.db"
----> 6 shutil.copy2('TestDB.db', backup_name)
      7 print(f"Backup created: {backup_name}")

File ~/Desktop/ml/InterviewPreparation/AI-ML-DS/practice_env/lib/python3.10/
 ↪shutil.py:434, in copy2(src, dst, follow_symlinks)
    432 if os.path.isdir(dst):
    433     dst = os.path.join(dst, os.path.basename(src))
--> 434 copyfile(src, dst, follow_symlinks=follow_symlinks)
    435 copystat(src, dst, follow_symlinks=follow_symlinks)
    436 return dst

File ~/Desktop/ml/InterviewPreparation/AI-ML-DS/practice_env/lib/python3.10/
 ↪shutil.py:254, in copyfile(src, dst, follow_symlinks)
    252     os.symlink(os.readlink(src), dst)
    253 else:
--> 254     with open(src, 'rb') as fsrc:
    255         try:
    256             with open(dst, 'wb') as fdst:
    257                 # macOS

FileNotFoundError: [Errno 2] No such file or directory: 'TestDB.db'
```

[ ]:
```
import sqlite3
import datetime

# Create connection to source and backup databases
source_conn = sqlite3.connect('TestDB.db')

# Create backup filename with timestamp
backup_filename = f"TestDB_backup_{datetime.datetime.now().
 ↪strftime('%Y%m%d_%H%M%S')}.db"
backup_conn = sqlite3.connect(backup_filename)

# Backup using SQLite's backup API
source_conn.backup(backup_conn)

# Close connections
```

```
backup_conn.close()
source_conn.close()

print(f"Backup created successfully: {backup_filename}")
```

```
Backup created successfully: TestDB_backup_20251205_183045.db
```

[ ]:
```
import sqlite3
import datetime

def backup_database(source_db, backup_dir='backups'):
    """Create a backup of SQLite database"""

    # Create backups directory if it doesn't exist
    import os
    if not os.path.exists(backup_dir):
        os.makedirs(backup_dir)

    # Generate backup filename with timestamp
    timestamp = datetime.datetime.now().strftime('%Y%m%d_%H%M%S')
    backup_db = os.path.join(backup_dir, f"{os.path.
 ↪basename(source_db)}_backup_{timestamp}.db")

    # Connect to source database
    source_conn = sqlite3.connect(source_db)

    # Create backup database
    backup_conn = sqlite3.connect(backup_db)

    # Backup with progress reporting
    def progress(status, remaining, total):
        print(f'Copied {total-remaining} of {total} pages...')

    # Perform the backup
    with backup_conn:
        source_conn.backup(backup_conn, pages=1, progress=progress)

    # Close connections
    backup_conn.close()
    source_conn.close()

    print(f" Backup completed: {backup_db}")
    return backup_db

# Usage
backup_database('TestDB.db')
```

```
Copied 0 of 0 pages…
  Backup completed: backups/TestDB.db_backup_20251205_184049.db
```

[ ]:
```
'backups/TestDB.db_backup_20251205_184049.db'
```

[ ]:
```
# First, connect to your database
%load_ext sql
%sql sqlite:///TestDB.db

# Create a SQL dump (backup as SQL commands)
import subprocess
import datetime

timestamp = datetime.datetime.now().strftime('%Y%m%d_%H%M%S')
dump_file = f"TestDB_backup_{timestamp}.sql"

# Use sqlite3 command line tool to create dump
subprocess.run(['sqlite3', 'TestDB.db', '.dump'], stdout=open(dump_file, 'w'))
print(f"SQL dump created: {dump_file}")
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
SQL dump created: TestDB_backup_20251205_184050.sql
```

SQLite doesn't have built-in differential backup functionality like SQL Server.A differential back up only backs up the parts of the database that have changed since the last full database backup.

Solution 1: Using Write-Ahead Logging (WAL) for incremental changes

[ ]:
```
import sqlite3
import os
import shutil
from datetime import datetime

def enable_wal_mode(db_path):
    """Enable Write-Ahead Logging for incremental backups"""
    conn = sqlite3.connect(db_path)
    conn.execute("PRAGMA journal_mode=WAL;")
    conn.close()
    print(f"WAL mode enabled for {db_path}")

def create_differential_backup(source_db, last_full_backup_time,␣
 ↪backup_dir='differential_backups'):
    """Create differential backup using WAL files"""

    if not os.path.exists(backup_dir):
        os.makedirs(backup_dir)

    # Check if WAL files exist
```

```
    wal_file = source_db + '-wal'
    shm_file = source_db + '-shm'

    if os.path.exists(wal_file):
        timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
        diff_backup = os.path.join(backup_dir, f"diff_backup_{timestamp}")

        # Save WAL and SHM files
        os.makedirs(diff_backup, exist_ok=True)

        if os.path.exists(wal_file):
            shutil.copy2(wal_file, os.path.join(diff_backup, 'wal'))
        if os.path.exists(shm_file):
            shutil.copy2(shm_file, os.path.join(diff_backup, 'shm'))

        print(f"Differential backup created: {diff_backup}")
        return diff_backup
    else:
        print("No WAL changes since last checkpoint")
        return None

# First, enable WAL mode
enable_wal_mode('TestDB.db')

# Create differential backup
last_full_backup_time = datetime.now()  # In real scenario, store this timestamp
create_differential_backup('TestDB.db', last_full_backup_time)
```

```
WAL mode enabled for TestDB.db
No WAL changes since last checkpoint
```

[ ]:
```
# Connect to an SQLite database (creates if doesn't exist)
%sql sqlite:///practice1.db
```

[ ]:
```
%sql select * from Customers --- IGNORE ---
```

```
   sqlite:///TestDB.db
 * sqlite:///practice1.db
```

[ ]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[102]:
```
%%sql sqlite:///practice1.db?mode=rwc
create table customers_copy as select * from Customers;
```

[102]:
```
Empty DataFrame
Columns: []
Index: []
```

[104]:
```
%sql select * from customers_copy
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
  * sqlite:///practice1.db?mode=rwc
```

[104]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

[105]:
```
%%sql
alter table Customers
add Email nvarchar(30);

select * from Customers
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
  * sqlite:///practice1.db?mode=rwc
```

[105]:

| | CustomerID | CustomerName | City | Email |
|---|---|---|---|---|
| 0 | 1 | ankit | kanpur | None |
| 1 | 2 | kiio | pathankoat | None |
| 2 | 3 | arpit | kanpur | None |
| 3 | 4 | kiio | delhi | None |

–alter table Customers rename column Email to New_Email

[106]:
```
%%sql
EXEC sp_rename 'Customers.Email', 'New_Email', 'column'
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
  * sqlite:///practice1.db?mode=rwc
(sqlite3.OperationalError) near "EXEC": syntax error
[SQL: EXEC sp_rename 'Customers.Email', 'New_Email', 'column']
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

The error occurs because EXEC sp_rename is SQL Server/Microsoft SQL Server syntax, but you're using SQLite. SQLite has different commands for renaming columns.

[107]:
```
%%sql
ALTER TABLE Customers RENAME COLUMN Email TO New_Email;
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[107]:
```
Empty DataFrame
Columns: []
Index: []
```

[108]:
```
%sql select * from Customers --- IGNORE ---
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[108]:

| | CustomerID | CustomerName | City | New_Email |
|---|---|---|---|---|
| 0 | 1 | ankit | kanpur | None |
| 1 | 2 | kiio | pathankoat | None |
| 2 | 3 | arpit | kanpur | None |
| 3 | 4 | kiio | delhi | None |

– drop column

[110]:
```
%%sql
-- First cell: Drop column
alter table Customers drop column New_Email;
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[110]:
```
Empty DataFrame
Columns: []
Index: []
```

[111]:
```
%%sql
-- Second cell: Select data
select * from Customers;
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[111]:

| | CustomerID | CustomerName | City |
|---|---|---|---|
| 0 | 1 | ankit | kanpur |
| 1 | 2 | kiio | pathankoat |
| 2 | 3 | arpit | kanpur |
| 3 | 4 | kiio | delhi |

– add column

[112]:
```
%%sql
alter table Customers
add DoB date
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[112]:
```
Empty DataFrame
Columns: []
Index: []
```

[113]:
```
%%sql
-- Second cell: Select data
select * from Customers;
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[113]:

| | CustomerID | CustomerName | City | DoB |
|---|---|---|---|---|
| 0 | 1 | ankit | kanpur | None |
| 1 | 2 | kiio | pathankoat | None |
| 2 | 3 | arpit | kanpur | None |
| 3 | 4 | kiio | delhi | None |

[114]:
```
%%sql
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[114]:
```
Empty DataFrame
Columns: []
Index: []
```

[116]:
```
%sql select * from Persons --- IGNORE ---
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[116]:
```
Empty DataFrame
Columns: []
Index: []
```

[119]:
```
%%sql
SELECT name, type, sql
FROM sqlite_master
WHERE type IN ('table', 'view')
ORDER BY type, name;
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[119]:

| | name | type | sql |
|---|---|---|---|
| 0 | Customers | table | CREATE TABLE Customers (CustomerID integer pri… |
| 1 | New Customers | table | CREATE TABLE "New Customers"(\n  CustomerID IN… |
| 2 | Orders | table | CREATE TABLE Orders (CustomerID int primary ke… |
| 3 | Persons | table | CREATE TABLE Persons (\n    ID int NOT NULL,\n… |
| 4 | customers_copy | table | CREATE TABLE customers_copy(\n  CustomerID INT… |
| 5 | experiment | table | CREATE TABLE experiment(\n  CustomerID INT,\n … |

[120]:
```
%%sql
-- For a single table
SELECT sql
FROM sqlite_master
WHERE type = 'table' AND name = 'Customers';
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[120]:

| | sql |
|---|---|
| 0 | CREATE TABLE Customers (CustomerID integer pri… |

[121]:
```
%%sql
PRAGMA table_info('Customers');
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[121]:

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 0 | 0 | CustomerID | INTEGER | 0 | None | 1 |
| 1 | 1 | CustomerName | TEXT | 0 | None | 0 |
| 2 | 2 | City | TEXT | 0 | None | 0 |
| 3 | 3 | DoB | date | 0 | None | 0 |

[122]:
```
%%sql
PRAGMA table_info('Persons');
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[122]:

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 0 | 0 | ID | INT | 1 | None | 1 |
| 1 | 1 | LastName | varchar(255) | 1 | None | 2 |
| 2 | 2 | FirstName | varchar(255) | 0 | None | 0 |
| 3 | 3 | Age | INT | 0 | None | 0 |

[123]:
```
%%sql
-- Tables with their SQL definition
SELECT
    name as 'Table',
    type,
    sql as 'Definition'
FROM sqlite_master
WHERE type = 'table'
ORDER BY name;
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[123]:

| | Table | type | Definition |
|---|---|---|---|
| 0 | Customers | table | CREATE TABLE Customers (CustomerID integer pri… |
| 1 | New Customers | table | CREATE TABLE "New Customers"(\n  CustomerID IN… |
| 2 | Orders | table | CREATE TABLE Orders (CustomerID int primary ke… |
| 3 | Persons | table | CREATE TABLE Persons (\n    ID int NOT NULL,\n… |
| 4 | customers_copy | table | CREATE TABLE customers_copy(\n  CustomerID INT… |
| 5 | experiment | table | CREATE TABLE experiment(\n  CustomerID INT,\n … |

[124]:
```
%%sql
-- Get all tables with their column details
SELECT
    m.name as table_name,
    p.cid as column_id,
    p.name as column_name,
    p.type as data_type,
    p."notnull" as is_not_null,
    p.dflt_value as default_value,
    p.pk as is_primary_key
FROM sqlite_master m
JOIN pragma_table_info(m.name) p
```

```
WHERE m.type = 'table'
ORDER BY m.name, p.cid;
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[124]:

| | table_name | column_id | column_name | data_type | is_not_null | default_value | is_primary_key |
|---|---|---|---|---|---|---|---|
| 0 | Customers | 0 | CustomerID | INTEGER | 0 | None | 1 |
| 1 | Customers | 1 | CustomerName | TEXT | 0 | None | 0 |
| 2 | Customers | 2 | City | TEXT | 0 | None | 0 |
| 3 | Customers | 3 | DoB | date | 0 | None | 0 |
| 4 | New Customers | 0 | CustomerID | INT | 0 | None | 0 |
| 5 | New Customers | 1 | CustomerName | TEXT | 0 | None | 0 |
| 6 | New Customers | 2 | City | TEXT | 0 | None | 0 |
| 7 | Orders | 0 | CustomerID | INT | 0 | None | 1 |
| 8 | Orders | 1 | ProductName | varchar(20) | 0 | None | 0 |
| 9 | Orders | 2 | ProductID | varchar(20) | 0 | None | 0 |
| 10 | Persons | 0 | ID | INT | 1 | None | 1 |
| 11 | Persons | 1 | LastName | varchar(255) | 1 | None | 2 |
| 12 | Persons | 2 | FirstName | varchar(255) | 0 | None | 0 |
| 13 | Persons | 3 | Age | INT | 0 | None | 0 |
| 14 | customers_copy | 0 | CustomerID | INT | 0 | None | 0 |
| 15 | customers_copy | 1 | CustomerName | TEXT | 0 | None | 0 |
| 16 | customers_copy | 2 | City | TEXT | 0 | None | 0 |
| 17 | experiment | 0 | CustomerID | INT | 0 | None | 0 |
| 18 | experiment | 1 | CustomerName | TEXT | 0 | None | 0 |
| 19 | experiment | 2 | City | TEXT | 0 | None | 0 |
| 20 | experiment | 3 | msg | | 0 | None | 0 |

```
16          None                0
17          None                0
18          None                0
19          None                0
20          None                0
```

[125]:
```
%%sql
-- List all indexes
SELECT
    name as index_name,
    tbl_name as table_name,
    sql as index_sql
FROM sqlite_master
WHERE type = 'index'
ORDER BY tbl_name, name;
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[125]:

| | index_name | table_name | index_sql |
|---|---|---|---|
| 0 | sqlite_autoindex_Orders_1 | Orders | None |
| 1 | sqlite_autoindex_Persons_1 | Persons | None |

[126]:
```
%%sql
-- List foreign key constraints
SELECT
    m.name as table_name,
    p."table" as referenced_table,
    p."from" as from_column,
    p."to" as to_column
FROM sqlite_master m
JOIN pragma_foreign_key_list(m.name) p
WHERE m.type = 'table'
ORDER BY m.name;
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[126]:
```
Empty DataFrame
Columns: []
Index: []
```

[130]:
```
%%sql
ALTER TABLE Persons DROP CONSTRAINT PK_Person;
```

```
  sqlite:///TestDB.db
```

```
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
(sqlite3.OperationalError) near "CONSTRAINT": syntax error
[SQL: ALTER TABLE Persons DROP CONSTRAINT PK_Person;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

The error occurs because SQLite doesn't support dropping constraints directly. You need to recreate the table without the constraint.

[133]:
```
%%sql
-- First drop the table if it exists
DROP TABLE IF EXISTS Persons;

-- Then create it
CREATE TABLE Persons (
    ID INTEGER NOT NULL,
    LastName TEXT NOT NULL,
    FirstName TEXT,
    Age INTEGER,
    City TEXT,
    CHECK (Age >= 18 AND City = 'Sandnes')
);
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[133]:
```
Empty DataFrame
Columns: []
Index: []
```

[134]:
```
%%sql
-- First drop the table if it exists
DROP TABLE IF EXISTS Orders;
-- Cell 2: Create Orders table with foreign key
CREATE TABLE Orders (
    OrderID INTEGER NOT NULL PRIMARY KEY,
    OrderNumber INTEGER NOT NULL,
    PersonID INTEGER,
    FOREIGN KEY (PersonID) REFERENCES Persons(ID)
);
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[134]:
```
Empty DataFrame
Columns: []
Index: []
```

[135]:
```
%%sql
-- Create another table (with different name since Orders already exists)
CREATE TABLE Orders2 (
    ID INTEGER NOT NULL PRIMARY KEY,
    OrderNumber INTEGER NOT NULL,
    OrderDate TEXT DEFAULT CURRENT_DATE
);
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[135]:
```
Empty DataFrame
Columns: []
Index: []
```

[138]:
```
%%sql
CREATE INDEX idx_lastname
ON Persons (LastName);
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[138]:
```
Empty DataFrame
Columns: []
Index: []
```

[ ]:
```
%%sql
CREATE TABLE Persons1 (
    Personid INTEGER PRIMARY KEY AUTOINCREMENT,
    LastName TEXT NOT NULL,
    FirstName TEXT,
    Age INTEGER
);
```

```
  sqlite:///TestDB.db
  sqlite:///practice1.db
* sqlite:///practice1.db?mode=rwc
```

[ ]:
```
Empty DataFrame
Columns: []
Index: []
```

[143]:
```
%%sql
ALTER TABLE Persons AUTOINCREMENT=100;
```

```
  sqlite:///TestDB.db
```

```
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
(sqlite3.OperationalError) near "AUTOINCREMENT": syntax error
[SQL: ALTER TABLE Persons AUTOINCREMENT=100;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

The error occurs because SQLite doesn't support directly setting the AUTOINCREMENT starting value with ALTER TABLE.

[144]:
```
%%sql
CREATE VIEW myview AS
SELECT CustomerName, City
FROM Customers
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[144]:
```
Empty DataFrame
Columns: []
Index: []
```

[145]:
```
%sql select * from myview
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[145]:

| | CustomerName | City |
|---|---|---|
| 0 | ankit | kanpur |
| 1 | kiio | pathankoat |
| 2 | arpit | kanpur |
| 3 | kiio | delhi |

[146]:
```
%sql drop view myview
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

[146]:
```
Empty DataFrame
Columns: []
Index: []
```

[147]:
```
%sql select * from myview
```

```
   sqlite:///TestDB.db
   sqlite:///practice1.db
 * sqlite:///practice1.db?mode=rwc
```

```
(sqlite3.OperationalError) no such table: myview
[SQL: select * from myview]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

[ ]: