.ipynb

April 27, 2022

[2]:
```
%load_ext sql
```

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

[3]:
```
%sql sqlite://
```

1 PROBLEM 1 : FIND THE % SPEND ON EACH ORDER FOR EACH CUSTOERS

[11]:
```
%%sql
CREATE TABLE ORDER_TABLE(Order_Id int, Order_Cost int, Customer_Id int);
```

```
 * sqlite://
(sqlite3.OperationalError) table ORDER_TABLE already exists
[SQL: CREATE TABLE ORDER_TABLE(Order_Id int, Order_Cost int, Customer_Id int);]
(Background on this error at: http://sqlalche.me/e/14/e3q8)
```

[12]:
```
%%sql

INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70001, 150,␣
 ↪3005);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70009, 270,␣
 ↪3001);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70002, 65,␣
 ↪3002);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70004, 110,␣
 ↪3009);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70007, 948,␣
 ↪3005);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70005, 2400,␣
 ↪3007);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70008, 5760,␣
 ↪3002);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70010, 1983,␣
 ↪3004);
```

```
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70003, 2480,␣
 ↪3009);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70012, 250,␣
 ↪3008);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70011, 75,␣
 ↪3003);
INSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70013, 3045,␣
 ↪3002);
```

```
 * sqlite://
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

```
---------------------------------------------------------------------------
ResourceClosedError                       Traceback (most recent call last)
<ipython-input-12-1d49e178325d> in <module>
----> 1 get_ipython().run_cell_magic('sql', '', '\nINSERT INTO␣
 ↪ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70001, 150, 3005);
 ↪\nINSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70009,␣
 ↪270, 3001);\nINSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUE␣
 ↪(70002, 65, 3002);\nINSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id␣
 ↪VALUES (70004, 110, 3009);\nINSERT INTO ORDER_TABLE(Order_Id, Order_Cost,␣
 ↪Customer_Id) VALUES (70007, 948, 3005);\nINSERT INTO ORDER_TABLE(Order_Id,␣
 ↪Order_Cost, Customer_Id) VALUES (70005, 2400, 3007);\nINSERT INTO␣
 ↪ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70008, 5760, 3002);
 ↪\nINSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70010,␣
 ↪1983, 3004);\nINSERT INTO ORDER_TABLE(Order_Id, Order_Cost, Customer_Id)␣
 ↪VALUES (70003, 2480, 3009);\nINSERT INTO ORDER_TABLE(Order_Id, Order_Cost,␣
 ↪Customer_Id) VALUES (70012, 250, 3008);\nINSERT INTO ORDER_TABLE(Order_Id,␣
 ↪Order_Cost, Customer_Id) VALUES (70011, 75, 3003);\nINSERT INTO␣
 ↪ORDER_TABLE(Order_Id, Order_Cost, Customer_Id) VALUES (70013, 3045, 3002);\n'

~\Downloads\conpak\lib\site-packages\IPython\core\interactiveshell.py in␣
 ↪run_cell_magic(self, magic_name, line, cell)
   2397             with self.builtin_trap:
   2398                 args = (magic_arg_s, cell)
-> 2399                 result = fn(*args, **kwargs)
   2400             return result
   2401

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
```

```
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\sql\magic.py in execute(self, line, cell,␣
↪local_ns)
    215
    216         try:
--> 217             result = sql.run.run(conn, parsed["sql"], self, user_ns)
    218
    219             if (

~\Downloads\conpak\lib\site-packages\sql\run.py in run(conn, sql, config,␣
↪user_namespace)
    369             if result and config.feedback:
    370                 print(interpret_rowcount(result.rowcount))
--> 371         resultset = ResultSet(result, statement, config)
    372         if config.autopandas:
    373             return resultset.DataFrame()

~\Downloads\conpak\lib\site-packages\sql\run.py in __init__(self, sqlaproxy,␣
↪sql, config)
    105
```

```
    106     def __init__(self, sqlaproxy, sql, config):
--> 107         self.keys = sqlaproxy.keys()
    108         self.sql = sql
    109         self.config = config

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\result.py in keys(self)
    705
    706         """
--> 707         return self._metadata.keys
    708
    709

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in keys(self)
   1199     @property
   1200     def keys(self):
-> 1201         self._we_dont_return_rows()
   1202
   1203

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in␣
↪_we_dont_return_rows(self, err)
   1176
   1177     def _we_dont_return_rows(self, err=None):
-> 1178         util.raise_(
   1179             exc.ResourceClosedError(
   1180                 "This result object does not return rows. "

~\Downloads\conpak\lib\site-packages\sqlalchemy\util\compat.py in␣
↪raise_(***failed resolving arguments***)
    209
    210         try:
--> 211             raise exception
    212         finally:
    213             # credit to

ResourceClosedError: This result object does not return rows. It has been close␣
↪automatically.
```

[13]:
```
%sql SELECT * from ORDER_TABLE
```

```
 * sqlite://
Done.
```

[13]:
```
[(70001, 150, 3005, None),
 (70009, 270, 3001, None),
 (70002, 65, 3002, None),
```

```
 (70004, 110, 3009, None),
 (70007, 948, 3005, None),
 (70005, 2400, 3007, None),
 (70008, 5760, 3002, None),
 (70010, 1983, 3004, None),
 (70003, 2480, 3009, None),
 (70012, 250, 3008, None),
 (70011, 75, 3003, None),
 (70013, 3045, 3002, None),
 (70001, 150, 3005, None),
 (70009, 270, 3001, None),
 (70002, 65, 3002, None),
 (70004, 110, 3009, None),
 (70007, 948, 3005, None),
 (70005, 2400, 3007, None),
 (70008, 5760, 3002, None),
 (70010, 1983, 3004, None),
 (70003, 2480, 3009, None),
 (70012, 250, 3008, None),
 (70011, 75, 3003, None),
 (70013, 3045, 3002, None)]
```

[17]:
```
%%sql
CREATE TABLE CUSTOMER_TABLE(Cust_Id int, Customer_Name varchar(50), City␣
 ↪char(50));
```

```
 * sqlite://
(sqlite3.OperationalError) table CUSTOMER_TABLE already exists
[SQL: CREATE TABLE CUSTOMER_TABLE(Cust_Id int, Customer_Name varchar(50), City
char(50));]
(Background on this error at: http://sqlalche.me/e/14/e3q8)
```

[21]:
```
%%sql

INSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3002, "Nick␣
 ↪Rimando", "New York");
INSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3007, "Brad␣
 ↪Davis", "New York");
INSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3005, "Graham␣
 ↪Zusi", "California");
INSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3008, "Julian␣
 ↪Green", "London");
INSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3004, "Fabian␣
 ↪Johnson", "Paris");
INSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3009, "Geoff␣
 ↪Cameron", "Berlin");
```

```
INSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3003, "Jozy␣
 ↪Altidor", "Moscow");
INSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3001, "Brad␣
 ↪Guzan", "London");
```

```
 * sqlite://
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

```
---------------------------------------------------------------------------
ResourceClosedError                       Traceback (most recent call last)
<ipython-input-21-48630d67ebbf> in <module>
----> 1 get_ipython().run_cell_magic('sql', '', '\nINSERT INTO␣
 ↪CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3002, "Nick Rimando",␣
 ↪"New York");\nINSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES
 ↪(3007, "Brad Davis", "New York");\nINSERT INTO CUSTOMER_TABLE(Cust_Id,␣
 ↪Customer_Name, City) VALUES (3005, "Graham Zusi", "California");\nINSERT INTO
 ↪CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3008, "Julian Green",␣
 ↪"London");\nINSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES␣
 ↪(3004, "Fabian Johnson", "Paris");\nINSERT INTO CUSTOMER_TABLE(Cust_Id,␣
 ↪Customer_Name, City) VALUES (3009, "Geoff Cameron", "Berlin");\nINSERT INTO␣
 ↪CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES (3003, "Jozy Altidor",␣
 ↪"Moscow");\nINSERT INTO CUSTOMER_TABLE(Cust_Id, Customer_Name, City) VALUES␣
 ↪(3001, "Brad Guzan", "London");\n')

~\Downloads\conpak\lib\site-packages\IPython\core\interactiveshell.py in␣
 ↪run_cell_magic(self, magic_name, line, cell)
   2397             with self.builtin_trap:
   2398                 args = (magic_arg_s, cell)
-> 2399                 result = fn(*args, **kwargs)
   2400             return result
   2401

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
 ↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
```

```
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\sql\magic.py in execute(self, line, cell,␣
↪local_ns)
    215
    216         try:
--> 217             result = sql.run.run(conn, parsed["sql"], self, user_ns)
    218
    219             if (

~\Downloads\conpak\lib\site-packages\sql\run.py in run(conn, sql, config,␣
↪user_namespace)
    369             if result and config.feedback:
    370                 print(interpret_rowcount(result.rowcount))
--> 371         resultset = ResultSet(result, statement, config)
    372         if config.autopandas:
    373             return resultset.DataFrame()

~\Downloads\conpak\lib\site-packages\sql\run.py in __init__(self, sqlaproxy,␣
↪sql, config)
    105
    106     def __init__(self, sqlaproxy, sql, config):
--> 107         self.keys = sqlaproxy.keys()
    108         self.sql = sql
    109         self.config = config

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\result.py in keys(self)
    705
    706         """
--> 707         return self._metadata.keys
    708
```

```
    709

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in keys(self)
   1199     @property
   1200     def keys(self):
-> 1201         self._we_dont_return_rows()
   1202
   1203

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in␣
 ↪_we_dont_return_rows(self, err)
   1176
   1177     def _we_dont_return_rows(self, err=None):
-> 1178         util.raise_(
   1179             exc.ResourceClosedError(
   1180                 "This result object does not return rows. "

~\Downloads\conpak\lib\site-packages\sqlalchemy\util\compat.py in␣
 ↪raise_(***failed resolving arguments***)
    209
    210         try:
--> 211             raise exception
    212         finally:
    213             # credit to

ResourceClosedError: This result object does not return rows. It has been close␣
 ↪automatically.
```

[22]:
```
%sql SELECT * from CUSTOMER_TABLE;
```

```
 * sqlite://
Done.
```

[22]:
```
[(3002, 'Nick Rimando', 'New York'),
 (3007, 'Brad Davis', 'New York'),
 (3005, 'Graham Zusi', 'California'),
 (3008, 'Julian Green', 'London'),
 (3004, 'Fabian Johnson', 'Paris'),
 (3009, 'Geoff Cameron', 'Berlin'),
 (3003, 'Jozy Altidor', 'Moscow'),
 (3001, 'Brad Guzan', 'London')]
```

[ ]:
```
# From the order table and Customer Table find the percentage of total spend a␣
 ↪customer spent on each item / order. Output
# Customer_Name, the % of total spent on each order for each customer
```

```
# APPROACH: AMAZON INTERVIEW QUESTION: First we need to join the table and␣
 ↪select Customer Name. Then when ou order by

# Customer Name you will see customer name repeated on the Customer_Name Column␣
 ↪because of multiple purchase order - buying

# different iteam or order at same or different dates each registered as unique␣
 ↪transaction id. In this situation since we

# cannot sum the total spent of each costomer we appl SUM OVER PARTITION to sum␣
 ↪the multiple transaction amount of a

# customer for each customer. Once done, cast the int into float and * by 100␣
 ↪to get the partition and later rounding it off.

# You can see Graham Zus's spent percentage on his total spent for each␣
 ↪transaction or order / item; that is his spent

# percentage on each orders and the are 7, 43, 7, 43 on 4 orders placed /␣
 ↪purchased by him.
```

[40]:
```
%%sql

SELECT Customer_Name, round(cast(Order_Cost as FLOAT)/ sum(Order_Cost) over␣
 ↪(PARTITION BY Customer_Name) * 100) as Percentage_of_total_spent
from ORDER_TABLE o JOIN CUSTOMER_TABLE c
ON c.Cust_Id = o.Customer_Id
ORDER BY Customer_Name;
```

```
 * sqlite://
Done.
```

[40]:
```
[('Brad Davis', 50.0),
 ('Brad Davis', 50.0),
 ('Brad Guzan', 50.0),
 ('Brad Guzan', 50.0),
 ('Fabian Johnson', 50.0),
 ('Fabian Johnson', 50.0),
 ('Geoff Cameron', 2.0),
 ('Geoff Cameron', 48.0),
 ('Geoff Cameron', 2.0),
 ('Geoff Cameron', 48.0),
 ('Graham Zusi', 7.0),
 ('Graham Zusi', 43.0),
 ('Graham Zusi', 7.0),
 ('Graham Zusi', 43.0),
 ('Jozy Altidor', 50.0),
```

```
 ('Jozy Altidor', 50.0),
 ('Julian Green', 50.0),
 ('Julian Green', 50.0),
 ('Nick Rimando', 0.0),
 ('Nick Rimando', 32.0),
 ('Nick Rimando', 17.0),
 ('Nick Rimando', 0.0),
 ('Nick Rimando', 32.0),
 ('Nick Rimando', 17.0)]
```

2 BASIC SUBQUERY 6 QUESTION

[45]:
```
%%sql
CREATE TABLE EMPLOYEE_TAB(Emp_Id int, First_Name varchar(51), Last_Name␣
 ↪varchar(51), Gender varchar(5), Position varchar(55), Dept_Id int, Salary␣
 ↪int);

INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (2, "Super", "Man", "M", "Tester", 1, 75555);
INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (3, "Ram", "Das", "F", "Architect", 1, 68555);
INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (4, "Shyam", "Det", "F", "Project", 1, 89555);
INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (5, "Dharam", "Kar", "M", "Software", 1, 51555);
INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (6, "Arijit", "Giri", "M", "Sales Assistant", 2,␣
 ↪89955);
INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (7, "Mou", "Sen", "F", "Sales Engineer", 2, 75555);
INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (8, "Sumit", "Saha", "M", "Sales Representative", 2,␣
 ↪76555);
INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (9, "Papa", "Roy", "F", "Sales Manager", 2, 59555);
INSERT INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position,␣
 ↪Dept_Id, Salary) VALUES (10, "Jodu", "Sinha", "M", "Sales Director", 2,␣
 ↪95555);
```

```
 * sqlite://
Done.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

```
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

```
---------------------------------------------------------------------------
ResourceClosedError                       Traceback (most recent call last)
<ipython-input-45-db696dbcc6b5> in <module>
----> 1 get_ipython().run_cell_magic('sql', '', 'CREATE TABLE␣
 ↪EMPLOYEE_TAB(Emp_Id int, First_Name varchar(51), Last_Name varchar(51), Gende␣
 ↪varchar(5), Position varchar(55), Dept_Id int, Salary int);\n\nINSERT INTO␣
 ↪EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id, Salary␣
 ↪VALUES (2, "Super", "Man", "M", "Tester", 1, 75555);\nINSERT INTO␣
 ↪EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id, Salary␣
 ↪VALUES (3, "Ram", "Das", "F", "Architect", 1, 68555);\nINSERT INTO␣
 ↪EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id, Salary␣
 ↪VALUES (4, "Shyam", "Det", "F", "Project", 1, 89555);\nINSERT INTO␣
 ↪EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id, Salary␣
 ↪VALUES (5, "Dharam", "Kar", "M", "Software", 1, 51555);\nINSERT INTO␣
 ↪EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id, Salary␣
 ↪VALUES (6, "Arijit", "Giri", "M", "Sales Assistant", 2, 89955);\nINSERT INTO␣
 ↪EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id, Salary␣
 ↪VALUES (7, "Mou", "Sen", "F", "Sales Engineer", 2, 75555);\nINSERT INTO␣
 ↪EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id, Salary␣
 ↪VALUES (8, "Sumit", "Saha", "M", "Sales Representative", 2, 76555);\nINSERT␣
 ↪INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id,␣
 ↪Salary) VALUES (9, "Papa", "Roy", "F", "Sales Manager", 2, 59555);\nINSERT␣
 ↪INTO EMPLOYEE_TAB(Emp_Id, First_Name, Last_Name, Gender, Position, Dept_Id,␣
 ↪Salary) VALUES (10, "Jodu", "Sinha", "M", "Sales Director", 2, 95555);\n')

~\Downloads\conpak\lib\site-packages\IPython\core\interactiveshell.py in␣
 ↪run_cell_magic(self, magic_name, line, cell)
   2397             with self.builtin_trap:
   2398                 args = (magic_arg_s, cell)
-> 2399                 result = fn(*args, **kwargs)
   2400             return result
   2401

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
 ↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):
```

```
~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\sql\magic.py in execute(self, line, cell,␣
↪local_ns)
    215
    216         try:
--> 217             result = sql.run.run(conn, parsed["sql"], self, user_ns)
    218
    219             if (

~\Downloads\conpak\lib\site-packages\sql\run.py in run(conn, sql, config,␣
↪user_namespace)
    369             if result and config.feedback:
    370                 print(interpret_rowcount(result.rowcount))
--> 371         resultset = ResultSet(result, statement, config)
    372         if config.autopandas:
    373             return resultset.DataFrame()

~\Downloads\conpak\lib\site-packages\sql\run.py in __init__(self, sqlaproxy,␣
↪sql, config)
    105
    106     def __init__(self, sqlaproxy, sql, config):
--> 107         self.keys = sqlaproxy.keys()
    108         self.sql = sql
    109         self.config = config

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\result.py in keys(self)
    705
    706         """
--> 707         return self._metadata.keys
    708
    709

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in keys(self)
   1199     @property
```

```
   1200     def keys(self):
-> 1201         self._we_dont_return_rows()
   1202
   1203

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in␣
↪_we_dont_return_rows(self, err)
   1176
   1177     def _we_dont_return_rows(self, err=None):
-> 1178         util.raise_(

   1179             exc.ResourceClosedError(
   1180                 "This result object does not return rows. "

~\Downloads\conpak\lib\site-packages\sqlalchemy\util\compat.py in␣
↪raise_(***failed resolving arguments***)
    209
    210         try:
--> 211             raise exception
    212         finally:
    213             # credit to

ResourceClosedError: This result object does not return rows. It has been close␣
↪automatically.
```

[46]:
```
%sql SELECT * from EMPLOYEE_TAB;
```

```
 * sqlite://
Done.
```

[46]:
```
[(2, 'Super', 'Man', 'M', 'Tester', 1, 75555),
 (3, 'Ram', 'Das', 'F', 'Architect', 1, 68555),
 (4, 'Shyam', 'Det', 'F', 'Project', 1, 89555),
 (5, 'Dharam', 'Kar', 'M', 'Software', 1, 51555),
 (6, 'Arijit', 'Giri', 'M', 'Sales Assistant', 2, 89955),
 (7, 'Mou', 'Sen', 'F', 'Sales Engineer', 2, 75555),
 (8, 'Sumit', 'Saha', 'M', 'Sales Representative', 2, 76555),
 (9, 'Papa', 'Roy', 'F', 'Sales Manager', 2, 59555),
 (10, 'Jodu', 'Sinha', 'M', 'Sales Director', 2, 95555)]
```

[71]:
```
%%sql

CREATE TABLE DEPARTMENT_TAB1(Dept_Id int, Dept_Name varchar (55));
INSERT INTO DEPARTMENT_TAB1(Dept_Id, Dept_Name) VALUES (1, "SOFTWARE");
INSERT INTO DEPARTMENT_TAB1(Dept_Id, Dept_Name) VALUES (2, "SALES");
```

```
 * sqlite://
Done.
```

```
1 rows affected.
1 rows affected.
```

```
---------------------------------------------------------------------------
ResourceClosedError                       Traceback (most recent call last)
<ipython-input-71-9200bbc41c19> in <module>
----> 1 get_ipython().run_cell_magic('sql', '', '\nCREATE TABLE␣
 ↪DEPARTMENT_TAB1(Dept_Id int, Dept_Name varchar (55));\nINSERT INTO␣
 ↪DEPARTMENT_TAB1(Dept_Id, Dept_Name) VALUES (1, "SOFTWARE");\nINSERT INTO␣
 ↪DEPARTMENT_TAB1(Dept_Id, Dept_Name) VALUES (2, "SALES");\n')

~\Downloads\conpak\lib\site-packages\IPython\core\interactiveshell.py in␣
 ↪run_cell_magic(self, magic_name, line, cell)
   2397             with self.builtin_trap:
   2398                 args = (magic_arg_s, cell)
-> 2399                 result = fn(*args, **kwargs)
   2400             return result
   2401

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
 ↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
 ↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):
```

```
~\Downloads\conpak\lib\site-packages\sql\magic.py in execute(self, line, cell,␣
↪local_ns)
    215
    216         try:
--> 217             result = sql.run.run(conn, parsed["sql"], self, user_ns)
    218
    219             if (

~\Downloads\conpak\lib\site-packages\sql\run.py in run(conn, sql, config,␣
↪user_namespace)
    369             if result and config.feedback:
    370                 print(interpret_rowcount(result.rowcount))
--> 371         resultset = ResultSet(result, statement, config)
    372         if config.autopandas:
    373             return resultset.DataFrame()

~\Downloads\conpak\lib\site-packages\sql\run.py in __init__(self, sqlaproxy,␣
↪sql, config)
    105
    106     def __init__(self, sqlaproxy, sql, config):
--> 107         self.keys = sqlaproxy.keys()
    108         self.sql = sql
    109         self.config = config

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\result.py in keys(self)
    705
    706         """
--> 707         return self._metadata.keys
    708
    709

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in keys(self)
   1199     @property
   1200     def keys(self):
-> 1201         self._we_dont_return_rows()
   1202
   1203

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in␣
↪_we_dont_return_rows(self, err)
   1176
   1177     def _we_dont_return_rows(self, err=None):
-> 1178         util.raise_(
   1179             exc.ResourceClosedError(
   1180                 "This result object does not return rows. "
```

```
~\Downloads\conpak\lib\site-packages\sqlalchemy\util\compat.py in␣
 ↪raise_(***failed resolving arguments***)
    209
    210         try:
--> 211             raise exception
    212         finally:
    213             # credit to

ResourceClosedError: This result object does not return rows. It has been close␣
 ↪automatically.
```

[72]:
```
%sql SELECT * from DEPARTMENT_TAB1;
```

```
 * sqlite://
Done.
```

[72]:
```
[(1, 'SOFTWARE'), (2, 'SALES')]
```

3 Q1: RETURN THE RECORD WITH MAXIMUM SALARY

[52]:
```
%%sql

SELECT * from EMPLOYEE_TAB
WHERE Salary = (SELECT Max(Salary) from EMPLOYEE_TAB);
```

```
 * sqlite://
Done.
```

[52]:
```
[(10, 'Jodu', 'Sinha', 'M', 'Sales Director', 2, 95555)]
```

4 Q2: SELECT HIGHEST SALARY IN EMPLOYEE TABLE

[53]:
```
%%sql

SELECT Max(Salary) from EMPLOYEE_TAB;
```

```
 * sqlite://
Done.
```

[53]:
```
[(95555,)]
```

5 Q3: SELECT SECOND HIGHEST SALARY IN EMPLOYEE TABLE

[56]:
```
%%sql

SELECT Max(Salary), First_Name, Position from EMPLOYEE_TAB
WHERE Salary Not In (SELECT Max(Salary) from EMPLOYEE_TAB);
```

```
 * sqlite://
Done.
```

[56]:
```
[(89955, 'Arijit', 'Sales Assistant')]
```

6 Q4: SELECT RANGE OF EMPLOEE BASED ON ID

[63]:
```
%%sql

SELECT * from EMPLOYEE_TAB
WHERE Emp_Id BETWEEN 3 and 8;
```

```
 * sqlite://
Done.
```

[63]:
```
[(3, 'Ram', 'Das', 'F', 'Architect', 1, 68555),
 (4, 'Shyam', 'Det', 'F', 'Project', 1, 89555),
 (5, 'Dharam', 'Kar', 'M', 'Software', 1, 51555),
 (6, 'Arijit', 'Giri', 'M', 'Sales Assistant', 2, 89955),
 (7, 'Mou', 'Sen', 'F', 'Sales Engineer', 2, 75555),
 (8, 'Sumit', 'Saha', 'M', 'Sales Representative', 2, 76555)]
```

7 Q5: RETURN EMPLOEE NAME HIGHEST SALARY AND DEPARTMENT

[79]:
```
%%sql

SELECT First_Name, Last_Name, Salary, Dept_Name
from
EMPLOYEE_TAB INNER JOIN
DEPARTMENT_TAB1
ON EMPLOYEE_TAB.Dept_Id = DEPARTMENT_TAB1.Dept_Id
WHERE Salary IN (SELECT Max(Salary) from EMPLOYEE_TAB);
```

```
 * sqlite://
Done.
```

[79]: [('Jodu', 'Sinha', 95555, 'SALES')]

8 Q6: EMP_NAME, DEPT_NAME, HIGHEST SALARY FOR EACH DEPARTMENT

[90]:
```
%%sql

SELECT First_Name, Dept_Name, Max(Salary) over (PARTITION BY Dept_Name)
from
EMPLOYEE_TAB e JOIN
DEPARTMENT_TAB1 d
ON e.Dept_Id = d.Dept_Id;
```

```
 * sqlite://
Done.
```

[90]:
```
[('Arijit', 'SALES', 95555),
 ('Mou', 'SALES', 95555),
 ('Sumit', 'SALES', 95555),
 ('Papa', 'SALES', 95555),
 ('Jodu', 'SALES', 95555),
 ('Super', 'SOFTWARE', 89555),
 ('Ram', 'SOFTWARE', 89555),
 ('Shyam', 'SOFTWARE', 89555),
 ('Dharam', 'SOFTWARE', 89555)]
```

[91]:
```
%%sql

SELECT Dept_Name, Max(Salary) over (PARTITION BY Dept_Name)
from
EMPLOYEE_TAB e JOIN
DEPARTMENT_TAB1 d
ON e.Dept_Id = d.Dept_Id;
```

```
 * sqlite://
Done.
```

[91]:
```
[('SALES', 95555),
 ('SALES', 95555),
 ('SALES', 95555),
 ('SALES', 95555),
 ('SALES', 95555),
 ('SOFTWARE', 89555),
 ('SOFTWARE', 89555),
 ('SOFTWARE', 89555),
 ('SOFTWARE', 89555)]
```

[96]:
```
%%sql

# This is the proper output

SELECT First_Name, Dept_Name, Max(Salary)
from
EMPLOYEE_TAB e JOIN
DEPARTMENT_TAB1 d
ON e.Dept_Id = d.Dept_Id
GROUP BY Dept_Name;
```

```
 * sqlite://
Done.
```

[96]:
```
[('Jodu', 'SALES', 95555), ('Shyam', 'SOFTWARE', 89555)]
```

9 ADVANCE QUERY SET OF QUESTIONS

[111]:
```
%%sql

CREATE TABLE EMPLOYEE_Z3(ENO int, ENAME varchar(51), DESIGNATION varchar(51),␣
 ↪Salary int, MGR int, DEPTNO int);
INSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES␣
 ↪(1, "aaa", "salesman", 7555, 2, 10);
INSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES␣
 ↪(1, "bbb", "manager", 17555, 2, 10);
INSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES␣
 ↪(1, "ccc", "president", 41555,3, 30);
INSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES␣
 ↪(1, "ddd", "clerk", 5555, 5, 20);
INSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES␣
 ↪(1, "eee", "manager", 21555, 3, 20);
INSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES␣
 ↪(1, "fff", "clerk", 8555, 5, 30);
```

```
 * sqlite://
Done.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

```
---------------------------------------------------------------------------
ResourceClosedError                       Traceback (most recent call last)
```

```
<ipython-input-111-cae80bfa810d> in <module>
----> 1 get_ipython().run_cell_magic('sql', '', '\nCREATE TABLE EMPLOYEE_Z3(ENO
 ↪int, ENAME varchar(51), DESIGNATION varchar(51), Salary int, MGR int, DEPTNO
 ↪int);\nINSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO)
 ↪VALUES (1, "aaa", "salesman", 7555, 2, 10);\nINSERT INTO EMPLOYEE_Z3(ENO,
 ↪ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES (1, "bbb", "manager", 17555,
 ↪2, 10);\nINSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO
 ↪VALUES (1, "ccc", "president", 41555,3, 30);\nINSERT INTO EMPLOYEE_Z3(ENO,
 ↪ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES (1, "ddd", "clerk", 5555, 5,
 ↪20);\nINSERT INTO EMPLOYEE_Z3(ENO, ENAME, DESIGNATION, Salary, MGR, DEPTNO)
 ↪VALUES (1, "eee", "manager", 21555, 3, 20);\nINSERT INTO EMPLOYEE_Z3(ENO,
 ↪ENAME, DESIGNATION, Salary, MGR, DEPTNO) VALUES (1, "fff", "clerk", 8555, 5,
 ↪30);\n')

~\Downloads\conpak\lib\site-packages\IPython\core\interactiveshell.py in
 ↪run_cell_magic(self, magic_name, line, cell)
   2397             with self.builtin_trap:
   2398                 args = (magic_arg_s, cell)
-> 2399                 result = fn(*args, **kwargs)
   2400             return result
   2401

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,
 ↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,
 ↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
```

```
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\sql\magic.py in execute(self, line, cell,␣
↪local_ns)
    215
    216         try:
--> 217             result = sql.run.run(conn, parsed["sql"], self, user_ns)
    218
    219             if (

~\Downloads\conpak\lib\site-packages\sql\run.py in run(conn, sql, config,␣
↪user_namespace)
    369             if result and config.feedback:
    370                 print(interpret_rowcount(result.rowcount))
--> 371         resultset = ResultSet(result, statement, config)
    372         if config.autopandas:
    373             return resultset.DataFrame()

~\Downloads\conpak\lib\site-packages\sql\run.py in __init__(self, sqlaproxy,␣
↪sql, config)
    105
    106     def __init__(self, sqlaproxy, sql, config):
--> 107         self.keys = sqlaproxy.keys()
    108         self.sql = sql
    109         self.config = config

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\result.py in keys(self)
    705
    706         """
--> 707         return self._metadata.keys
    708
    709

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in keys(self)
   1199     @property
   1200     def keys(self):
-> 1201         self._we_dont_return_rows()
   1202
   1203

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in␣
↪_we_dont_return_rows(self, err)
   1176
   1177     def _we_dont_return_rows(self, err=None):
-> 1178         util.raise_(
   1179             exc.ResourceClosedError(
   1180                 "This result object does not return rows. "
```

```
~\Downloads\conpak\lib\site-packages\sqlalchemy\util\compat.py in␣
 ↪raise_(***failed resolving arguments***)
    209
    210         try:
--> 211             raise exception
    212         finally:
    213             # credit to

ResourceClosedError: This result object does not return rows. It has been close␣
 ↪automatically.
```

[112]:
```
%sql SELECT * from EMPLOYEE_Z3;
```

```
 * sqlite://
Done.
```

[112]:
```
[(1, 'aaa', 'salesman', 7555, 2, 10),
 (1, 'bbb', 'manager', 17555, 2, 10),
 (1, 'ccc', 'president', 41555, 3, 30),
 (1, 'ddd', 'clerk', 5555, 5, 20),
 (1, 'eee', 'manager', 21555, 3, 20),
 (1, 'fff', 'clerk', 8555, 5, 30)]
```

[113]:
```
%%sql

CREATE TABLE DEPT_D(DEPTNO int, DEPTNAME varchar(55), LOC varchar(55));
```

```
 * sqlite://
Done.
```

```
---------------------------------------------------------------------------
ResourceClosedError                         Traceback (most recent call last)
<ipython-input-113-f5f70da67cff> in <module>
----> 1 get_ipython().run_cell_magic('sql', '', '\nCREATE TABLE DEPT_D(DEPTNO␣
 ↪int, DEPTNAME varchar(55), LOC varchar(55));\n')

~\Downloads\conpak\lib\site-packages\IPython\core\interactiveshell.py in␣
 ↪run_cell_magic(self, magic_name, line, cell)
   2397             with self.builtin_trap:
   2398                 args = (magic_arg_s, cell)
-> 2399                 result = fn(*args, **kwargs)
   2400             return result
   2401

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
```

```
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\sql\magic.py in execute(self, line, cell,␣
↪local_ns)
    215
    216         try:
--> 217             result = sql.run.run(conn, parsed["sql"], self, user_ns)
    218
    219             if (

~\Downloads\conpak\lib\site-packages\sql\run.py in run(conn, sql, config,␣
↪user_namespace)
    369             if result and config.feedback:
    370                 print(interpret_rowcount(result.rowcount))
--> 371         resultset = ResultSet(result, statement, config)
    372         if config.autopandas:
    373             return resultset.DataFrame()

~\Downloads\conpak\lib\site-packages\sql\run.py in __init__(self, sqlaproxy,␣
↪sql, config)
    105
    106     def __init__(self, sqlaproxy, sql, config):
```

```
--> 107         self.keys = sqlaproxy.keys()
    108         self.sql = sql
    109         self.config = config

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\result.py in keys(self)
    705
    706         """
--> 707         return self._metadata.keys
    708
    709

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in keys(self)
   1199     @property
   1200     def keys(self):
-> 1201         self._we_dont_return_rows()
   1202
   1203

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in␣
↪_we_dont_return_rows(self, err)
   1176
   1177     def _we_dont_return_rows(self, err=None):
-> 1178         util.raise_(

   1179             exc.ResourceClosedError(
   1180                 "This result object does not return rows. "

~\Downloads\conpak\lib\site-packages\sqlalchemy\util\compat.py in␣
↪raise_(***failed resolving arguments***)
    209
    210         try:
--> 211             raise exception
    212         finally:
    213             # credit to

ResourceClosedError: This result object does not return rows. It has been close␣
↪automatically.
```

[114]:
```
%%sql
INSERT INTO DEPT_D (DEPTNO, DEPTNAME, LOC) VALUES (10, "SALES", "MUMBAI");
INSERT INTO DEPT_D (DEPTNO, DEPTNAME, LOC) VALUES (20, "HR", "DELHI");
INSERT INTO DEPT_D (DEPTNO, DEPTNAME, LOC) VALUES (30, "ACCOUNTS", "CHENNAI");
INSERT INTO DEPT_D (DEPTNO, DEPTNAME, LOC) VALUES (40, "PRODUCTION",␣
↪"BANGALORE");
```

```
 * sqlite://
1 rows affected.
1 rows affected.
```

```
1 rows affected.
1 rows affected.
```

```
---------------------------------------------------------------------------
ResourceClosedError                       Traceback (most recent call last)
<ipython-input-114-ef26fdf65326> in <module>
----> 1 get_ipython().run_cell_magic('sql', '', 'INSERT INTO DEPT_D (DEPTNO,␣
 ↪DEPTNAME, LOC) VALUES (10, "SALES", "MUMBAI");\nINSERT INTO DEPT_D (DEPTNO,␣
 ↪DEPTNAME, LOC) VALUES (20, "HR", "DELHI");\nINSERT INTO DEPT_D (DEPTNO,␣
 ↪DEPTNAME, LOC) VALUES (30, "ACCOUNTS", "CHENNAI");\nINSERT INTO DEPT_D␣
 ↪(DEPTNO, DEPTNAME, LOC) VALUES (40, "PRODUCTION", "BANGALORE");\n')

~\Downloads\conpak\lib\site-packages\IPython\core\interactiveshell.py in␣
 ↪run_cell_magic(self, magic_name, line, cell)
   2397             with self.builtin_trap:
   2398                 args = (magic_arg_s, cell)
-> 2399                 result = fn(*args, **kwargs)
   2400             return result
   2401

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
 ↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\decorator.py in fun(*args, **kw)
    229             if not kwsyntax:
    230                 args, kw = fix(args, kw, sig)
--> 231             return caller(func, *(extras + args), **kw)
    232     fun.__name__ = func.__name__
    233     fun.__doc__ = func.__doc__

~\Downloads\conpak\lib\site-packages\IPython\core\magic.py in <lambda>(f, *a,␣
 ↪**k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
```

```
    189         if callable(arg):

~\Downloads\conpak\lib\site-packages\sql\magic.py in execute(self, line, cell,␣
↪local_ns)
    215
    216         try:
--> 217             result = sql.run.run(conn, parsed["sql"], self, user_ns)
    218
    219             if (

~\Downloads\conpak\lib\site-packages\sql\run.py in run(conn, sql, config,␣
↪user_namespace)
    369             if result and config.feedback:
    370                 print(interpret_rowcount(result.rowcount))
--> 371         resultset = ResultSet(result, statement, config)
    372         if config.autopandas:
    373             return resultset.DataFrame()

~\Downloads\conpak\lib\site-packages\sql\run.py in __init__(self, sqlaproxy,␣
↪sql, config)
    105
    106     def __init__(self, sqlaproxy, sql, config):
--> 107         self.keys = sqlaproxy.keys()
    108         self.sql = sql
    109         self.config = config

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\result.py in keys(self)
    705
    706         """
--> 707         return self._metadata.keys
    708
    709

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in keys(self)
   1199     @property
   1200     def keys(self):
-> 1201         self._we_dont_return_rows()
   1202
   1203

~\Downloads\conpak\lib\site-packages\sqlalchemy\engine\cursor.py in␣
↪_we_dont_return_rows(self, err)
   1176
   1177     def _we_dont_return_rows(self, err=None):
-> 1178         util.raise_(
   1179             exc.ResourceClosedError(
   1180                 "This result object does not return rows. "
```

```
~\Downloads\conpak\lib\site-packages\sqlalchemy\util\compat.py in␣
 ↪raise_(***failed resolving arguments***)
    209
    210         try:
--> 211             raise exception
    212         finally:
    213             # credit to

ResourceClosedError: This result object does not return rows. It has been close␣
 ↪automatically.
```

[115]:
```
%sql SELECT * from DEPT_D;
```

```
 * sqlite://
Done.
```

[115]:
```
[(10, 'SALES', 'MUMBAI'),
 (20, 'HR', 'DELHI'),
 (30, 'ACCOUNTS', 'CHENNAI'),
 (40, 'PRODUCTION', 'BANGALORE')]
```

[116]:
```
%sql SELECT * from EMPLOYEE_Z3;
```

```
 * sqlite://
Done.
```

[116]:
```
[(1, 'aaa', 'salesman', 7555, 2, 10),
 (1, 'bbb', 'manager', 17555, 2, 10),
 (1, 'ccc', 'president', 41555, 3, 30),
 (1, 'ddd', 'clerk', 5555, 5, 20),
 (1, 'eee', 'manager', 21555, 3, 20),
 (1, 'fff', 'clerk', 8555, 5, 30)]
```

Q1 : SALARY GREATER THAN AVG SALARY OF DEPTNO 10

[118]:
```
%%sql
SELECT ENAME, Salary from EMPLOYEE_Z3
WHERE Salary > (SELECT Avg(Salary) from EMPLOYEE_Z3 WHERE DEPTNO = 10);
```

```
 * sqlite://
Done.
```

[118]:
```
[('bbb', 17555), ('ccc', 41555), ('eee', 21555)]
```

Q2: DISPLAY EMPLOYEES OF DEPT NO 10 WHO GETS MORE THAN THE AVG SALARY OF DEPT NO 10

[122]:
```
%%sql

SELECT ENAME, ENO, DEPTNO, Salary from EMPLOYEE_Z3
WHERE DEPTNO = 10 and Salary > (SELECT Avg(Salary) from EMPLOYEE_Z3 WHERE␣
 ↪DEPTNO = 10);
```

```
 * sqlite://
Done.
```

[122]:
```
[('bbb', 1, 10, 17555)]
```

Q3: DISPLAY EMPLOYEES GETTING MORE SALARY THAN THEIR DEPARTMENTS

[124]:
```
%%sql

SELECT ENAME, ENO, DEPTNO, Salary from EMPLOYEE_Z3 S
WHERE Salary > (SELECT Avg(Salary) from EMPLOYEE_Z3 WHERE DEPTNO = S.DEPTNO);
```

```
 * sqlite://
Done.
```

[124]:
```
[('bbb', 1, 10, 17555), ('ccc', 1, 30, 41555), ('eee', 1, 20, 21555)]
```

10 The above example is called corelated queries where the inner query takes the reference S and each time the inner

11 query is processed for each departments it takes the reference of outer query which is also processed

12 each time the inner quer gets processed

Q4: AMONG MANAGERS WHO GETS THE HIGHEST SALARY ?

First find out highest salary from Managers in inner query then map with the manager's name in the outer query as in select statement with employee name with a where clase as in designation = Manager

[135]:
```
%%sql

SELECT ENO, ENAME, Salary from EMPLOYEE_Z3
WHERE DESIGNATION = "manager" and Salary = (SELECT Max(Salary) from EMPLOYEE_Z3␣
 ↪WHERE DESIGNATION = "manager"); The
```

```
 * sqlite://
Done.
```

[135]:
```
[(1, 'eee', 21555)]
```

[138]:
```
%%sql

SELECT ENO, ENAME, Salary from EMPLOYEE_Z3
WHERE (DESIGNATION, Salary) IN (SELECT DESIGNATION, Max(Salary) from␣
 ↪EMPLOYEE_Z3 WHERE DESIGNATION = "manager");
# The above code is same and gives same output
```

```
 * sqlite://
Done.
```

[138]:
```
[(1, 'eee', 21555)]
```

Q5: WHICH DESIGNATION HAS EXACTLY 2 EMPLOYEES

[147]:
```
%%sql

SELECT ENO, ENAME, DESIGNATION, count(*) from EMPLOYEE_Z3
GROUP BY DESIGNATION
HAVING count(*) =  2;
```

```
 * sqlite://
Done.
```

[147]:
```
[(1, 'ddd', 'clerk', 2), (1, 'bbb', 'manager', 2)]
```

Q6: WHICH DESIGNATION HAS MOST NO OF EMPLOYEES

[182]:
```
%%sql

SELECT DESIGNATION, count(ENO) as empcount
from EMPLOYEE_Z3
GROUP BY DESIGNATION
ORDER BY empcount DESC;
```

```
 * sqlite://
Done.
```

[182]:
```
[('manager', 2), ('clerk', 2), ('salesman', 1), ('president', 1)]
```

13 SIMPLE QUERY AND JOINS

[127]:
```
%sql SELECT * from ORDER_TABLE;
```

```
 * sqlite://
Done.
```

[127]:
```
[(70001, 150, 3005, None),
 (70009, 270, 3001, None),
 (70002, 65, 3002, None),
 (70004, 110, 3009, None),
 (70007, 948, 3005, None),
 (70005, 2400, 3007, None),
 (70008, 5760, 3002, None),
 (70010, 1983, 3004, None),
 (70003, 2480, 3009, None),
 (70012, 250, 3008, None),
 (70011, 75, 3003, None),
 (70013, 3045, 3002, None),
 (70001, 150, 3005, None),
 (70009, 270, 3001, None),
 (70002, 65, 3002, None),
 (70004, 110, 3009, None),
 (70007, 948, 3005, None),
 (70005, 2400, 3007, None),
 (70008, 5760, 3002, None),
 (70010, 1983, 3004, None),
 (70003, 2480, 3009, None),
 (70012, 250, 3008, None),
 (70011, 75, 3003, None),
 (70013, 3045, 3002, None)]
```

[128]:
```
%sql SELECT * from CUSTOMER_TABLE;
```

```
 * sqlite://
Done.
```

[128]:
```
[(3002, 'Nick Rimando', 'New York'),
 (3007, 'Brad Davis', 'New York'),
 (3005, 'Graham Zusi', 'California'),
 (3008, 'Julian Green', 'London'),
 (3004, 'Fabian Johnson', 'Paris'),
 (3009, 'Geoff Cameron', 'Berlin'),
 (3003, 'Jozy Altidor', 'Moscow'),
 (3001, 'Brad Guzan', 'London')]
```

[131]:
```
%%sql
SELECT Customer_Name, Order_Cost, Avg(Order_Cost)
from ORDER_TABLE o JOIN CUSTOMER_TABLE c
ON c.Cust_Id = o.Customer_Id
GROUP BY Customer_Name;
```

```
 * sqlite://
Done.
```

[131]:
```
[('Brad Davis', 2400, 2400.0),
 ('Brad Guzan', 270, 270.0),
 ('Fabian Johnson', 1983, 1983.0),
 ('Geoff Cameron', 110, 1295.0),
 ('Graham Zusi', 150, 549.0),
 ('Jozy Altidor', 75, 75.0),
 ('Julian Green', 250, 250.0),
 ('Nick Rimando', 65, 2956.6666666666665)]
```

[133]:
```
%%sql
SELECT Customer_Id, Order_Cost, (SELECT Avg(Order_Cost) from ORDER_TABLE)
from
ORDER_TABLE;
# This is an example of subquer used in the SELECT STATEMENT
```

```
 * sqlite://
Done.
```

[133]:
```
[(3005, 150, 1461.3333333333333),
 (3001, 270, 1461.3333333333333),
 (3002, 65, 1461.3333333333333),
 (3009, 110, 1461.3333333333333),
 (3005, 948, 1461.3333333333333),
 (3007, 2400, 1461.3333333333333),
 (3002, 5760, 1461.3333333333333),
 (3004, 1983, 1461.3333333333333),
 (3009, 2480, 1461.3333333333333),
 (3008, 250, 1461.3333333333333),
 (3003, 75, 1461.3333333333333),
 (3002, 3045, 1461.3333333333333),
 (3005, 150, 1461.3333333333333),
 (3001, 270, 1461.3333333333333),
 (3002, 65, 1461.3333333333333),
 (3009, 110, 1461.3333333333333),
 (3005, 948, 1461.3333333333333),
 (3007, 2400, 1461.3333333333333),
 (3002, 5760, 1461.3333333333333),
 (3004, 1983, 1461.3333333333333),
 (3009, 2480, 1461.3333333333333),
 (3008, 250, 1461.3333333333333),
 (3003, 75, 1461.3333333333333),
 (3002, 3045, 1461.3333333333333)]
```

[134]:
```
%%sql

SELECT Customer_Name, City, Order_Cost, Avg(Order_Cost)
from ORDER_TABLE o JOIN CUSTOMER_TABLE c
```

```
ON c.Cust_Id = o.Customer_Id
GROUP BY Customer_Name;
# Basically finding the Avg Order Cost afetr joining two tables
```

```
 * sqlite://
Done.
```

[134]:
```
[('Brad Davis', 'New York', 2400, 2400.0),
 ('Brad Guzan', 'London', 270, 270.0),
 ('Fabian Johnson', 'Paris', 1983, 1983.0),
 ('Geoff Cameron', 'Berlin', 110, 1295.0),
 ('Graham Zusi', 'California', 150, 549.0),
 ('Jozy Altidor', 'Moscow', 75, 75.0),
 ('Julian Green', 'London', 250, 250.0),
 ('Nick Rimando', 'New York', 65, 2956.6666666666665)]
```

[ ]: