```
import pandas as pd
import pandasql as ps
from pandasql import sqldf
mysql = lambda q: sqldf(q, globals())
```

```
import os
os.chdir("")
```

Q1 Write a Sql Query to get the second highest salary from the Exployee Table?

Please use csv file named employee to read the raw data.

```
q1_pd=pd.read_csv('q1.csv')
```

```
q1_pd.head()
```

	Id	Salary
0	1	100
1	2	200
2	3	300

Next steps: Generate code with q1_pd View recommended plots

```
from sqlalchemy import create_engine

# Create an in-memory SQLite database engine
engine = create_engine('sqlite:///:memory:')
```

```
# Store DataFrame in the SQLite database
q1_pd.to_sql('Employee',engine,index=False, if_exists='replace')

query = """
        SELECT MAX(Salary) AS SecondHighestSalary
        FROM Employee
        WHERE Salary < (SELECT MAX(Salary) FROM Employee)
        """
second_highest_salary=pd.read_sql_query(query, engine)
print(second_highest_salary)
```

```
        SecondHighestSalary
     0                  200
```

Q2 Write a sql query to rank scores.if there is a tie between scores, both should have the same ranking. Note that after a tie, the next ranking should be the next consecutive integer value.

Please use csv file named rank_scores for pandas data frame

```
q2_pd=pd.read_csv('q2.csv')
```

```
cn2=q2_pd.columns
print(cn2)
```

```
     Index(['Id', 'Score'], dtype='object')
```

```
q2_pd.to_sql('Employee', engine, index=False, if_exists='replace')

query_q2= """
            SELECT Score, DENSE_RANK() OVER (ORDER BY Score DESC) AS Rank
            FROM Employee
            """

ranking_q2=pd.read_sql_query(query_q2, engine)

print(ranking_q2)
```

```
   Score  Rank
0   4.00     1
1   4.00     1
2   3.85     2
3   3.65     3
4   3.65     3
5   3.50     4
```

Q3 Write a SQL query to finds out employees who earn more than their managers?

Please use file named employee_earning for creation of Pandas Data Frame

```
q3_pd=pd.read_csv("q3.csv")

q3_pd.head()
```

	Id	Name	Salary	ManagerId
0	1	Joe	70000	3.0
1	2	Henry	80000	4.0
2	3	Sam	60000	NaN
3	4	Max	90000	NaN

Next steps: Generate code with q3_pd View recommended plots

```
cn=q3_pd.columns
print(cn)
```

```
Index(['Id', 'Name', 'Salary', 'ManagerId'], dtype='object')
```

```
q3_pd.to_sql('Employee', engine, index=False, if_exists='replace')
query_q3 = """
            SELECT e.Name AS Employee
            FROM Employee e
            INNER JOIN Employee m ON e.ManagerId = m.Id
            WHERE e.Salary > m.Salary
            """
employees_higher_salary_than_manager_q3 = pd.read_sql_query(query_q3, engine)
print(employees_higher_salary_than_manager_q3)
```

```
  Employee
0      Joe
```

Q4 Write a SQL query to find employees who earn the top three salaries in each of the department.

Please use department_emploee_salary for creation of Pandas data frame

```
q4_1_pd=pd.read_csv("q4_1.csv")
q4_2_pd=pd.read_csv("q4_2.csv")
```

```
q4_2_pd.head()
```

	Id	Name	Salary	DepartmentId
0	1	Joe	85000	1
1	2	Henry	80000	2
2	3	Sam	60000	2
3	4	Max	90000	1
4	5	Janet	69000	1

Next steps: Generate code with q4_2_pd View recommended plots

```
q4_1_pd.head()
```

	Id	Name
0	1	IT
1	2	Sales

Next steps: Generate code with q4_1_pd View recommended plots

```
q4_1_pd.to_sql('Departments', engine, index=False, if_exists='replace')
q4_2_pd.to_sql('Employees', engine, index=False, if_exists='replace')
query_q4 = """
            SELECT DepartmentName, EmployeeName, Salary
            FROM (
                SELECT d.Name AS DepartmentName, e.Name AS EmployeeName, e.Salary,
                       DENSE_RANK() OVER (PARTITION BY e.DepartmentId ORDER BY e.Salary DESC) AS SalaryRank
                FROM Employees e
                INNER JOIN Departments d ON e.DepartmentId = d.Id
            ) AS ranked_employees
            WHERE SalaryRank <= 3
            """
top_three_salaries_per_department_q4 = pd.read_sql_query(query_q4, engine)
print(top_three_salaries_per_department_q4)
```

```
  DepartmentName EmployeeName  Salary
0             IT          Max   90000
1             IT          Joe   85000
2             IT        Randy   85000
3             IT         Will   70000
4          Sales        Henry   80000
5          Sales          Sam   60000
```

Q5 Write a query to find managers those have at least 5 direct reports?

Please use q5.csv for creation of Pandas data frame

```
q5_pd=pd.read_csv('q5.csv')
q5_pd.head()
```

	Id	Name	Department	ManagerId
0	101	John	A	NaN
1	102	Dan	A	101.0
2	103	James	A	101.0
3	104	Amy	A	101.0
4	105	Anne	A	101.0

Next steps: Generate code with q5_pd View recommended plots

```
query_q5 = """
            SELECT Name
            FROM Employees
            WHERE ManagerId IS NOT NULL
            AND ManagerId != ''
            GROUP BY ManagerId
            HAVING COUNT(*) >= 5
            """
managers_with_at_least_5_direct_reports_q5 = pd.read_sql_query(query_q5, engine)
print(managers_with_at_least_5_direct_reports_q5)
```

```
       Name
    0  Dan
```

Q6 Write a sql query to rank salaries with in department?

Please use q6.csv for creation of Pandas data frame

```
q6_pd=pd.read_csv('q6.csv')
q6_pd.head()
```

	employee_id	full_name	department	salary
0	100	Mary Johns	SALES	1000
1	101	Sean Moldy	IT	1500
2	102	Peter Dugan	SALES	2000
3	103	Lilian Penn	SALES	1700
4	104	Milton Kowarsky	IT	1800

Next steps: Generate code with q6_pd View recommended plots

```
q6_pd.to_sql('Employees', engine, index=False, if_exists='replace')
query_q6 = """
            SELECT DENSE_RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS dept_ranking,department,employee_id,full_name,s
            FROM Employees
            """
salaries_rank_within_department_q6 = pd.read_sql_query(query_q6, engine)
print(salaries_rank_within_department_q6)
```

```
   dept_ranking department  employee_id        full_name  salary
0             1   ACCOUNTS          105    Mareen Bisset    1200
1             2   ACCOUNTS          106     Airton Graue    1100
2             1         IT          104  Milton Kowarsky    1800
3             2         IT          101       Sean Moldy    1500
4             1      SALES          102      Peter Dugan    2000
5             2      SALES          103      Lilian Penn    1700
6             3      SALES          100       Mary Johns    1000
```

Q7 Assume you have train schedule data set. The data set has `Train_id` , `Station Name` and `start_time`. Write a sql query which adds a new column called "time to next station". Please use `Lead` window function.

Please use q7.csv for creation of pandas data frame

```
q7_pd=pd.read_csv('q7.csv')
q7_pd.head()
```

	Train_id	Station	Time
0	110	San Francisco	10:00:00
1	110	Redwood City	10:54:00
2	110	Palo Alto	11:02:00
3	110	San Jose	12:35:00
4	120	San Francisco	11:00:00

Next steps: Generate code with q7_pd View recommended plots

```
q7_pd.to_sql('Train_schedule', engine, index=False, if_exists='replace')
#Using Lead window function
query_q7 = """
            SELECT Train_id, Station, Time as station_time,
                   LEAD(Time) OVER (PARTITION BY Train_id ORDER BY Time) AS time_to_next_station
            FROM Train_schedule
            """
time_to_next_station_q7 = pd.read_sql_query(query_q7, engine)
print(time_to_next_station_q7)
```

```
   Train_id        Station station_time time_to_next_station
0       110  San Francisco     10:00:00             10:54:00
1       110   Redwood City     10:54:00             11:02:00
2       110      Palo Alto     11:02:00             12:35:00
3       110       San Jose     12:35:00                 None
4       120  San Francisco     11:00:00             12:49:00
5       120      Palo Alto     12:49:00             13:30:00
6       120       San Jose     13:30:00                 None
```

Q8 Write an Sql query to find all numbers that apper at least thress times consecutively?

Please use q8.csv for creation of pandas data frame.

```
q8_pd=pd.read_csv('q8.csv')
q8_pd.head()
```

	Id	Num
0	1	1
1	2	1
2	3	1
3	4	2
4	5	1

Next steps: Generate code with q8_pd View recommended plots

```
q8_pd.to_sql('Numbers', engine, index=False, if_exists='replace')
query_q8 = """
            SELECT Num as ConsecutiveNums
            FROM (
                SELECT Num,
                       ROW_NUMBER() OVER (ORDER BY Id) - ROW_NUMBER() OVER (PARTITION BY Num ORDER BY Id) AS grp
                FROM Numbers
            ) AS temp
            GROUP BY Num, grp
            HAVING COUNT(*) >= 3
            """

consecutive_numbers_q8 = pd.read_sql_query(query_q8, engine)
print(consecutive_numbers_q8)
```

```
   ConsecutiveNums
0                1
```

Q9 A university uses 2 data tables, student and department to store data about its students and departments associated with each major. Write a query to print the respective department name and number of students majoring in each department for all the departments in department table (even ones with no current students)?

Please use q9_1.csv and q9_2.csv for creation of pandas data frame.

```
q9_1_pd=pd.read_csv('q9_1.csv')
q9_2_pd=pd.read_csv('q9_2.csv')
```

```
q9_1_pd.head()
```

	student_id	student_name	gender	dept_id
0	1	Jack	M	1
1	2	jane	F	1
2	3	Mark	M	2

Next steps: Generate code with q9_1_pd View recommended plots

```
q9_2_pd.head()
```

	dept_id	dept_name
0	1	Engineering
1	2	Science
2	3	Law

Next steps: Generate code with q9_2_pd View recommended plots

```
q9_1_pd.to_sql('student', engine, index=False, if_exists='replace')
q9_2_pd.to_sql('department', engine, index=False, if_exists='replace')
query_q9 = """
            SELECT d.dept_name, COUNT(s.student_id) AS student_number
            FROM department d
            LEFT JOIN student s ON d.dept_id = s.dept_id
            GROUP BY d.dept_name ORDER BY student_number desc
            """
department_student_count_q9 = pd.read_sql_query(query_q9, engine)
print(department_student_count_q9)
```

```
     dept_name  student_number
0  Engineering               2
1      Science               1
```

```
2          Law              0
```

Q10 Several friends at acinema ticket office would like to reseve consecutive available seats. Can you help to query all the consecutive seats order by seat_id using the cinema table(q10.csv)

Here seat_id us an auto increment int and free is bool (1 means free and 0 means occupied).

Please use q10.csv for creation of pandas data frame

```
q10_pd=pd.read_csv('q10.csv')
```

```
q10_pd.head()
```

	seat_id	free
0	1	1
1	2	0
2	3	1
3	4	1
4	5	1

Next steps: Generate code with q10_pd View recommended plots

```
q10_pd.to_sql('cinema', engine, index=False, if_exists='replace')

query_q10 = """
WITH
    T AS (
        SELECT
            *,
            SUM(free = 1) OVER (
                ORDER BY seat_id
                ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
            ) AS cnt
        FROM Cinema
    )
SELECT seat_id
FROM T
WHERE free = 1 AND cnt > 1
ORDER BY 1;
"""
ans_q10 = pd.read_sql_query(query_q10, engine)
print(ans_q10)
```