COMPARISON OF BASIC OPERATIONS IN PANDAS AND SQL

GRA 4142 Data Management and Python Programming

Jan Kudlicka (jan.kudlicka@bi.no)

Department of Data Science and Analytics

November 11, 2022

We will use the same database as during the SQL lectures. The Pandas data frames contain the same data, using the values in the *id* column as labels.

In [4]:
```
employee_df
```

Out[4]:

	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id	note
id							
1	Alnes	Bernt	1967	1	200	2.0	None
2	Fjelldal	Mads	1953	1	250	NaN	None
3	Lekve	Karoline	1980	1	195	2.0	At maternity leave
4	Longva	Victor	1978	1	190	2.0	None
5	Nymo	Ingvar	1976	2	240	6.0	HSE manager
6	Bodin	Runar	1969	2	240	NaN	None
7	Bakke	Alfred	1960	2	180	6.0	None
8	Vie	Tor	1974	2	190	6.0	None
9	Westgaard	Sten	1975	2	190	6.0	None
10	Liseth	Rakel	1969	3	190	13.0	None
11	Norman	Emil	1982	3	170	13.0	None
12	Dyrhaug	Atle	1971	3	200	13.0	None
13	Kvistad	Jens	1952	3	230	NaN	None
14	Ulset	Lucas	1983	3	170	13.0	None
15	Kvien	Amalie	1977	4	205	16.0	None
16	Tveten	Thomas	1968	4	260	NaN	None
17	Lende	Marita	1972	4	210	16.0	None

SUBSET OF COLUMNS

List the last name and the first name of all employees.

In [5]:

```
%%sql
SELECT id, last_name, first_name
FROM employee
```

Out[5]:

id	last_name	first_name
1	Alnes	Bernt
2	Fjelldal	Mads
3	Lekve	Karoline
4	Longva	Victor
5	Nymo	Ingvar
6	Bodin	Runar
7	Bakke	Alfred
8	Vie	Tor
9	Westgaard	Sten
10	Liseth	Rakel
11	Norman	Emil
12	Dyrhaug	Atle
13	Kvistad	Jens
14	Ulset	Lucas
15	Kvien	Amalie
16	Tveten	Thomas
17	Lende	Marita

In [6]:

```
employee_df[['last_name', 'first_name']]
```

Out[6]:

	last_name	first_name
id		
1	Alnes	Bernt
2	Fjelldal	Mads
3	Lekve	Karoline
4	Longva	Victor
5	Nymo	Ingvar
6	Bodin	Runar
7	Bakke	Alfred
8	Vie	Tor
9	Westgaard	Sten
10	Liseth	Rakel
11	Norman	Emil
12	Dyrhaug	Atle
13	Kvistad	Jens
14	Ulset	Lucas
15	Kvien	Amalie
16	Tveten	Thomas
17	Lende	Marita

FILTERING ROWS

Which employees work at the department with ID 1?

In [7]:

```
%%sql
SELECT *
FROM employee
WHERE department_id = 1
```

Out[7]:

id	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id
1	Alnes	Bernt	1967	1	200	2
2	Fjelldal	Mads	1953	1	250	
3	Lekve	Karoline	1980	1	195	2
4	Longva	Victor	1978	1	190	2

In [8]:

```
employee_df.loc[employee_df['department_id'] == 1]
# or:
employee_df.query("department_id == 1")
```

Out[8]:

	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id
id						
1	Alnes	Bernt	1967	1	200	2.0
2	Fjelldal	Mads	1953	1	250	NaN
3	Lekve	Karoline	1980	1	195	2.0
4	Longva	Victor	1978	1	190	2.0

FILTERING ROWS, CONT.

Which employees from the department with ID 1 were born after 1970?

In [9]:
```
%%sql
SELECT *
FROM employee
WHERE department_id = 1
  AND year_of_birth > 1970
```

Out[9]:

id	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id
3	Lekve	Karoline	1980	1	195	2
4	Longva	Victor	1978	1	190	2

In [10]:
```
employee_df.loc[
    (employee_df.department_id == 1) &
    (employee_df.year_of_birth > 1970)
]
# or:
employee_df.query("department_id == 1 and year_of_birth > 19
```

Out[10]:

	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id
id						
3	Lekve	Karoline	1980	1	195	2.0
4	Longva	Victor	1978	1	190	2.0

In SQL and Pandas' query: `AND`, `OR` and `NOT`

In Pandas: `&`, `|` and `~` (remember parentheses!)

LIMITING THE NUMBER OF ROWS

List the last name and the first name of the employees, limit the output to 5 rows.

In [11]:

```
%%sql
SELECT id, last_name, first_name
FROM employee
LIMIT 5
```

Out[11]:

id	last_name	first_name
1	Alnes	Bernt
2	Fjelldal	Mads
3	Lekve	Karoline
4	Longva	Victor
5	Nymo	Ingvar

In [12]:

```
employee_df[['last_name', 'first_name']].head(5)
```

Out[12]:

	last_name	first_name
id		
1	Alnes	Bernt
2	Fjelldal	Mads
3	Lekve	Karoline
4	Longva	Victor
5	Nymo	Ingvar

UNIQUE ROWS

Which departments have employees that earn more than 230 NOK per hour?

In [13]:
```
%%sql
SELECT DISTINCT department_id
FROM employee
WHERE hour_salary > 230
```

Out[13]:

department_id
1
2
4

In [14]:
```
employee_df.query("hour_salary > 230") \
            ['department_id'] \
            .drop_duplicates()
# or:
# employee_df.loc[employee_df.hour_salary > 230, 'department
#               .drop_duplicates()
```

Out[14]:
```
id
2      1
5      2
16     4
Name: department_id, dtype: int64
```

EXPRESSIONS WITH ATTRIBUTES

Find the first name, the last name and the monthly salary of all employees that make less than 40000 per month. (Assume that each month has 176 working hours.)

In [15]:
```
%%sql
SELECT first_name, last_name, hour_salary*176 AS salary
FROM employee
WHERE salary < 40000
```

Out[15]:

first_name	last_name	salary
Bernt	Alnes	35200
Karoline	Lekve	34320
Victor	Longva	33440
Alfred	Bakke	31680
Tor	Vie	33440
Sten	Westgaard	33440
Rakel	Liseth	33440
Emil	Norman	29920
Atle	Dyrhaug	35200
Lucas	Ulset	29920
Amalie	Kvien	36080
Marita	Lende	36960

In [16]:
```
employee_df.assign(salary=employee_df['hour_salary']*176) \
           .query('salary < 40000') \
           [['first_name', 'last_name', 'salary']]
```

Out[16]:

	first_name	last_name	salary
id			
1	Bernt	Alnes	35200
3	Karoline	Lekve	34320
4	Victor	Longva	33440
7	Alfred	Bakke	31680
8	Tor	Vie	33440
9	Sten	Westgaard	33440
10	Rakel	Liseth	33440
11	Emil	Norman	29920
12	Atle	Dyrhaug	35200
14	Lucas	Ulset	29920
15	Amalie	Kvien	36080
17	Marita	Lende	36960

IN OPERATOR

Which employees work in the departments with IDs 1 and 3?

In [17]:
```
%%sql
SELECT *
FROM employee
WHERE department_id IN (1, 3)
```

Out[17]:

id	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id
1	Alnes	Bernt	1967	1	200	2
2	Fjelldal	Mads	1953	1	250	
3	Lekve	Karoline	1980	1	195	2
4	Longva	Victor	1978	1	190	2
10	Liseth	Rakel	1969	3	190	13
11	Norman	Emil	1982	3	170	13
12	Dyrhaug	Atle	1971	3	200	13
13	Kvistad	Jens	1952	3	230	
14	Ulset	Lucas	1983	3	170	13

In [18]:
```
employee_df.loc[employee_df['department_id'].isin([1, 3])]
# or:
employee_df.query("department_id in [1, 3]")
```

Out[18]:

	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id
id						
1	Alnes	Bernt	1967	1	200	2.0
2	Fjelldal	Mads	1953	1	250	NaN
3	Lekve	Karoline	1980	1	195	2.0
4	Longva	Victor	1978	1	190	2.0
10	Liseth	Rakel	1969	3	190	13.0
11	Norman	Emil	1982	3	170	13.0
12	Dyrhaug	Atle	1971	3	200	13.0
13	Kvistad	Jens	1952	3	230	NaN
14	Ulset	Lucas	1983	3	170	13.0

TESTING FOR MISSING DATA

List the last name and the first name of all supervisors.

In [19]:

```
%%sql
SELECT id, last_name, first_name
FROM employee
WHERE supervisor_id IS NULL
```

Out[19]:

id	last_name	first_name
2	Fjelldal	Mads
6	Bodin	Runar
13	Kvistad	Jens
16	Tveten	Thomas

In [20]:

```
employee_df.loc[
    employee_df['supervisor_id'].isnull(),
    ['last_name', 'first_name']
]
# or:
#employee_df.query("supervisor_id.isnull()", engine='python
#             [['last_name', 'first_name']]
# Note: We can also use .isna() instead of .isnull()
```

Out[20]:

	last_name	first_name
id		
2	Fjelldal	Mads
6	Bodin	Runar
13	Kvistad	Jens
16	Tveten	Thomas

SORTING ROWS

List the top 5 best paid employees.

Remember that order of rows in relations is not relevant, but we can sort them in result sets.

In [21]:
```
%%sql
SELECT *
FROM employee
ORDER BY hour_salary DESC
LIMIT 5
```

Out[21]:

id	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id
16	Tveten	Thomas	1968	4	260	
2	Fjelldal	Mads	1953	1	250	
5	Nymo	Ingvar	1976	2	240	6
6	Bodin	Runar	1969	2	240	
13	Kvistad	Jens	1952	3	230	

In [22]:
```
employee_df.sort_values('hour_salary', ascending=False) \
           .head(5)
# Note: use .sort_index for ordering by the labels (the inde
```

Out[22]:

	last_name	first_name	year_of_birth	department_id	hour_salary	supervisor_id
id						
16	Tveten	Thomas	1968	4	260	NaN
2	Fjelldal	Mads	1953	1	250	NaN
5	Nymo	Ingvar	1976	2	240	6.0
6	Bodin	Runar	1969	2	240	NaN
13	Kvistad	Jens	1952	3	230	NaN

SUMMARY STATISTICS

What is the mean of the (hour) salary of all employees?

In [23]:
```
%%sql
SELECT avg(hour_salary)
FROM employee
```

Out[23]:

avg(hour_salary)
206.47058823529412

In [24]:
```
employee_df['hour_salary'].mean()
# or:
employee_df['hour_salary'].agg('mean')
```

Out[24]:
```
206.47058823529412
```

GROUPING AND SUMMARY STATISTICS

For each department list the number of its employees and their average salary.

In [25]:

```
%%sql
SELECT department_id, count(*) AS employee_count,
       avg(hour_salary) AS avg_hour_salary
FROM employee
GROUP BY department_id
```

Out[25]:

department_id	employee_count	avg_hour_salary
1	4	208.75
2	5	208.0
3	5	192.0
4	3	225.0

In [26]:

```
employee_df.groupby('department_id') \
           .agg(employee_count=('department_id', 'size'),
                avg_hour_salary=('hour_salary', 'mean'))
```

Out[26]:

	employee_count	avg_hour_salary
department_id		
1	4	208.75
2	5	208.00
3	5	192.00
4	3	225.00

FILTERING GROUPS

What is the maximum salary in each department that has at least 5 employees?

In [27]:
```
%%sql
SELECT department_id, max(hour_salary) AS max_salary
FROM employee
GROUP BY department_id
HAVING count(*)>=5
```

Out[27]:

department_id	max_salary
2	240
3	230

In [28]:
```
employee_df.groupby('department_id') \
           .agg(emp_count=('department_id', 'size'),
                max_salary=('hour_salary', 'max')) \
           .query("emp_count >= 5") \
           ['max_salary']
```

Out[28]:
```
department_id
2    240
3    230
Name: max_salary, dtype: int64
```

UNIONS

List the titles of all departments and all projects in one table / frame.

```
In [29]: %%sql
         SELECT title FROM department
         UNION ALL
         SELECT title FROM project
```

Out[29]:

title
Planning
Production A
Production B
Sales and administration
New department
Project Alfa, Uppsala
Project Bravo, Knivsta
Project Charlie, Stockholm

```
In [30]: pd.concat([
             department_df['title'],
             project_df['title']
         ])
         # Add .reset_index(drop=True) to reset the index
         # For UNION, add .drop_duplicates()
```

```
Out[30]: id
         1                       Planning
         2                   Production A
         3                   Production B
         4       Sales and administration
         5                 New department
         1          Project Alfa, Uppsala
         2         Project Bravo, Knivsta
         3     Project Charlie, Stockholm
         Name: title, dtype: object
```

INNER JOINS

List the first name and the last name of all employees, together with the title of their department.

In [31]:

```
%%sql
SELECT e.id, e.first_name, e.last_name, d.title
FROM employee e
JOIN department d ON e.department_id=d.id
```

Out[31]:

id	first_name	last_name	title
1	Bernt	Alnes	Planning
2	Mads	Fjelldal	Planning
3	Karoline	Lekve	Planning
4	Victor	Longva	Planning
5	Ingvar	Nymo	Production A
6	Runar	Bodin	Production A
7	Alfred	Bakke	Production A
8	Tor	Vie	Production A
9	Sten	Westgaard	Production A
10	Rakel	Liseth	Production B
11	Emil	Norman	Production B
12	Atle	Dyrhaug	Production B
13	Jens	Kvistad	Production B
14	Lucas	Ulset	Production B
15	Amalie	Kvien	Sales and administration
16	Thomas	Tveten	Sales and administration
17	Marita	Lende	Sales and administration

In [32]:

```
pd.merge(employee_df, department_df,
         left_on='department_id',
         right_index=True) \
    [['first_name', 'last_name', 'title']]
```

Out[32]:

	first_name	last_name	title
id			
1	Bernt	Alnes	Planning
2	Mads	Fjelldal	Planning
3	Karoline	Lekve	Planning
4	Victor	Longva	Planning
5	Ingvar	Nymo	Production A
6	Runar	Bodin	Production A
7	Alfred	Bakke	Production A
8	Tor	Vie	Production A
9	Sten	Westgaard	Production A
10	Rakel	Liseth	Production B
11	Emil	Norman	Production B
12	Atle	Dyrhaug	Production B
13	Jens	Kvistad	Production B
14	Lucas	Ulset	Production B
15	Amalie	Kvien	Sales and administration
16	Thomas	Tveten	Sales and administration
17	Marita	Lende	Sales and administration