Fall 2025 Data Science Bootcamp

Week 4: SQL Fundamentals

Take-home assignment

Section 1 (Mondays): 2:00PM-4:00PM EST

Instructor: Shaahid Ahmed Nadeem

Made by: Mahdi (Matt) Ghadimi

N14369173

mg8786@nyu.edu

1. Actors and Directors Who Cooperated At Least Three Times:

SQL:

```sql
# Write your MySQL query statement below
SELECT actor_id, director_id
FROM ActorDirector
GROUP BY actor_id, director_id
HAVING COUNT(*) >= 3;
```

Pandas:

```python
import pandas as pd

def actors_and_directors(actor_director: pd.DataFrame) -> pd.DataFrame:
    result = (
        actor_director
        .groupby(['actor_id', 'director_id'])
        .size()
        .reset_index(name='cnt')
        .query('cnt >= 3')[['actor_id', 'director_id']]
    )
    return result
```

2. Fix Names in a Table

SQL:

```sql
SELECT
    user_id,
    CONCAT(UPPER(LEFT(name, 1)), LOWER(SUBSTRING(name, 2))) AS name
FROM Users
ORDER BY user_id;
```

Pandas:

```python
import pandas as pd

def fix_names(users: pd.DataFrame) -> pd.DataFrame:
    users['name'] = users['name'].str.capitalize()
    return users.sort_values('user_id')
```

3. Combine Two Tables

SQL:

```sql
SELECT
    Person.firstName,
    Person.lastName,
    Address.city,
    Address.state
FROM Person
LEFT JOIN Address
    ON Person.personId = Address.personId;
```

Pandas:

```python
import pandas as pd

def combine_two_tables(person: pd.DataFrame, address: pd.DataFrame) -> pd.DataFrame:
    result = person.merge(
        address[['personId', 'city', 'state']],
        on='personId',
        how='left'
    )
    return result[['firstName', 'lastName', 'city', 'state']]
```

4. Second Highest Salary

SQL:

```
SELECT
    (SELECT DISTINCT salary
     FROM Employee
     ORDER BY salary DESC
     LIMIT 1 OFFSET 1) AS SecondHighestSalary;
```

Pandas:

```
import pandas as pd

def second_highest_salary(employee: pd.DataFrame) -> pd.DataFrame:
    distinct_salaries = employee['salary'].drop_duplicates().sort_values(ascending=False)
    second = distinct_salaries.iloc[1] if len(distinct_salaries) > 1 else None
    return pd.DataFrame({'SecondHighestSalary': [second]})
```

5. List the Products Ordered in a Period

SQL:

```
SELECT
    p.product_name,
    SUM(o.unit) AS unit
FROM Orders o
JOIN Products p
    ON o.product_id = p.product_id
WHERE o.order_date >= '2020-02-01'
  AND o.order_date < '2020-03-01'
GROUP BY p.product_name
HAVING SUM(o.unit) >= 100;
```

Pandas:

```
import pandas as pd

def list_products(products: pd.DataFrame, orders: pd.DataFrame) -> pd.DataFrame:
    # Filter orders in Feb 2020
    feb_orders = orders[
        (orders['order_date'] >= '2020-02-01') &
        (orders['order_date'] < '2020-03-01')
    ]

    # Sum units by product_id
    summed = feb_orders.groupby('product_id', as_index=False)['unit'].sum()

    # Filter products with at least 100 units
    filtered = summed[summed['unit'] >= 100]

    # Join with products to get product_name
    result = filtered.merge(products[['product_id', 'product_name']], on='product_id')

    return result[['product_name', 'unit']]
```

6. Replace Employee ID With The Unique Identifier

SQL:

```sql
SELECT
    eui.unique_id,
    e.name
FROM Employees e
LEFT JOIN EmployeeUNI eui
    ON e.id = eui.id;
```

Pandas:

```python
import pandas as pd

def replace_employee_id(employees: pd.DataFrame, employee_uni: pd.DataFrame) -> pd.DataFrame:
    result = employees.merge(
        employee_uni[['id', 'unique_id']],
        on='id',
        how='left'
    )
    return result[['unique_id', 'name']]
```

7. Game Play Analysis IV

SQL:

```sql
WITH first_login AS (
    SELECT player_id, MIN(event_date) AS first_date
    FROM Activity
    GROUP BY player_id
)
SELECT ROUND(
    SUM(
        CASE
            WHEN EXISTS (
                SELECT 1
                FROM Activity a2
                WHERE a2.player_id = f.player_id
                  AND a2.event_date = DATE_ADD(f.first_date, INTERVAL 1 DAY)
            ) THEN 1 ELSE 0
        END
    ) * 1.0 / COUNT(*), 2
) AS fraction
FROM first_login f;
```

Pandas:

```python
import pandas as pd

def gameplay_analysis(activity: pd.DataFrame) -> pd.DataFrame:
    activity['event_date'] = pd.to_datetime(activity['event_date'])

    first_login = activity.groupby('player_id')['event_date'].min().reset_index()

    def logged_next_day(row):
        return ((activity['player_id'] == row['player_id']) &
                (activity['event_date'] == row['event_date'] + pd.Timedelta(days=1))).any()

    count_next_day = first_login.apply(logged_next_day, axis=1).sum()
    total_players = first_login.shape[0]

    fraction = round(count_next_day / total_players, 2)

    return pd.DataFrame({'fraction': [fraction]})
```

8. Project Employees I

SQL:

```sql
SELECT
    p.project_id,
    ROUND(AVG(e.experience_years), 2) AS average_years
FROM Project p
JOIN Employee e
    ON p.employee_id = e.employee_id
GROUP BY p.project_id;
```

Pandas:

```python
import pandas as pd

def project_employees_i(project: pd.DataFrame, employee: pd.DataFrame) -> pd.DataFrame:
    merged = project.merge(employee[['employee_id', 'experience_years']], on='employee_id')

    result = merged.groupby('project_id', as_index=False)['experience_years'].mean()

    result['average_years'] = result['experience_years'].round(2)

    return result[['project_id', 'average_years']]
```

9. Department Top Three Salaries

SQL:

```sql
WITH RankedSalaries AS (
    SELECT
        e.id,
        e.name AS Employee,
        e.salary AS Salary,
        d.name AS Department,
        DENSE_RANK() OVER(PARTITION BY e.departmentId ORDER BY e.salary DESC) AS salary_rank
    FROM Employee e
    JOIN Department d
        ON e.departmentId = d.id
)
SELECT Department, Employee, Salary
FROM RankedSalaries
WHERE salary_rank <= 3;
```

Pandas:

```python
import pandas as pd

def top_three_salaries(employee: pd.DataFrame, department: pd.DataFrame) -> pd.DataFrame:
    merged = employee.merge(department, left_on='departmentId', right_on='id', suffixes=('',
'_dept'))

    merged['salary_rank'] = merged.groupby('departmentId')['salary'] \
                             .rank(method='dense', ascending=False)

    top_earners = merged[merged['salary_rank'] <= 3]

    return top_earners[['name_dept', 'name', 'salary']].rename(
        columns={'name_dept': 'Department', 'name': 'Employee', 'salary': 'Salary'}
    )
```