UNI FREIBURG

Python Course

WT 24/25

Lecture: Marcus Rockel & Prof. Dr. Eva Lütkebohmert-Holtz, Tutorial: Felix Häschel
https://www.finance.uni-freiburg.de/

Tutorial 7

Exercise 1 (GitHub Copilot Installation)**.**

Sign up for the free student plan of GitHub Copilot, e.g. on GitHub Education. Install the GitHub Copilot extension in Visual Studio Code and the GitHub Copilot plugin in PyCharm. Test the extension/plugin by writing a few lines of code in both editors.

Exercise 2 (Review of the Python Basics - Advent of Code)**.**

To practice and reinforce your Python basics, complete the first exercise on the Advent of Code website. Advent of Code is an annual event featuring daily programming puzzles released every day from December 1st to December 25th. These puzzles are designed to challenge and improve your coding skills, with problems ranging from easy to complex, making it suitable for both beginners and experienced programmers. The event is widely recognized in the programming community and is supported by prominent sponsors. For instance, in 2024, sponsors include **Jane Street**, **JPMorgan Chase**, **Spotify**, **Sony Interactive Entertainment**, and **American Express**.

Tasks:

1. Visit the Advent of Code website at Advent of Code.
2. Navigate to the first exercise of the current year.
3. Solve the problem using Python to apply the basics you have learned so far.
4. If you wish to check your results and participate in the global leaderboard, you can register on the website using your GitHub, Google, Twitter, or Reddit account. Registration is optional but recommended for a complete experience.

Solution.

```
# Part 1
left, right = [], []
with open("data/AoC_day1.txt", "r") as f:
    # Process the data line by line
    for line in f:
        l, r = map(int, line.strip().split())
        left.append(l)
        right.append(r)

part1_result = sum([abs(i - j) for i, j in zip(sorted(left), sorted(right))])
print(f"Part 1: {part1_result}")
```

```python
# Part 2
part2_result = sum([num * right.count(num) for num in set(left)])
print(f"Part 2: {part2_result}")

# Compact solution
# Process all of the data by using slicing to extract the columns
data = [*map(int, open("data/AoC_day1.txt", "r").read().split())]
left, right = sorted(data[::2]), sorted(data[1::2])
part1_compact = sum([abs(i - j) for i, j in zip(left, right)])
part2_compact = sum([num * right.count(num) for num in left])

print(f"Part 1 (direct): {part1_compact}")
print(f"Part 2 (direct): {part2_compact}")

# Alternative solution using map
data = [*map(int, open("data/AoC_day1.txt").read().split())]
A, B = sorted(data[::2]), sorted(data[1::2])
part1_alt = sum(map(lambda a, b: abs(a - b), A, B))
part2_alt = sum([a * B.count(a) for a in A])

print(f"Part 1 (alternative) = {part1_alt}")
print(f"Part 2 (alternative) = {part2_alt}")
```

Exercise 3 (Database Normalization to Third Normal Form)**.**

OrderID	**CustomerID**	**CustomerAddress**	**Item**	**Quantity**
1	1001	123 Main St	Widget A	2
			Widget B	1
2	1002	456 Elm St	Widget C	5
3	1001	123 Main St	Widget A	3
			Widget D	2

(i) Convert the unnormalized table into First Normal Form (1NF).

(ii) Identify the functional dependencies and the primary key in your 1NF table.

(iii) Normalize the 1NF table into Second Normal Form (2NF).

(iv) Normalize the 2NF table into Third Normal Form (3NF).

Solution. **(i) Conversion to First Normal Form (1NF)**
First Normal Form requires that each cell in a table contains only atomic (indivisible) values and that there are no repeating groups.
In the unnormalized table, some orders have multiple items and quantities in the same record (e.g., OrderID 1 and 3). To convert to 1NF, we need to create separate rows for each item in an order:

OrderID	**CustomerID**	**CustomerAddress**	**Item**	**Quantity**
1	1001	123 Main St	Widget A	2
1	1001	123 Main St	Widget B	1
2	1002	456 Elm St	Widget C	5
3	1001	123 Main St	Widget A	3
3	1001	123 Main St	Widget D	2

(ii) Functional Dependencies and Primary Key
From the 1NF table, we can identify the following functional dependencies:

1. **OrderID** $\rightarrow$ **CustomerID**
2. **CustomerID** $\rightarrow$ **CustomerAddress**
3. **OrderID**, **Item** $\rightarrow$ **Quantity**

The primary key is the combination of **OrderID** and **Item**, as each pair uniquely identifies a record.
(iii) Conversion to Second Normal Form (2NF)
Second Normal Form requires the table to be in 1NF and that all non-key attributes are fully functionally dependent on the entire primary key.
In the 1NF table, **CustomerID** depends only on **OrderID**, not on the entire primary key (**OrderID**, **Item**). Similarly, **CustomerAddress** depends on **CustomerID**, not directly on the primary key.

To eliminate partial dependencies, we decompose the table into two relations:
Order Table:

OrderID	**CustomerID**	**CustomerAddress**
1	1001	123 Main St
2	1002	456 Elm St
3	1001	123 Main St

OrderItem Table:

OrderID	**Item**	**Quantity**
1	Widget A	2
1	Widget B	1
2	Widget C	5
3	Widget A	3
3	Widget D	2

Now, all non-key attributes in each table are fully functionally dependent on their respective primary keys.

(iv) Conversion to Third Normal Form (3NF)

Third Normal Form requires the table to be in 2NF and that all attributes are only dependent on the primary key (no transitive dependencies).

In the *Order* table, **CustomerID** determines **CustomerAddress** (assuming each customer has a single address). Therefore, **CustomerAddress** is transitively dependent on **OrderID** via **CustomerID**.

To eliminate this transitive dependency, we create a separate *Customer* table:
Customer Table:

CustomerID	**CustomerAddress**
1001	123 Main St
1002	456 Elm St

Update the *Order* table accordingly:
Order Table (Revised):

OrderID	**CustomerID**
1	1001
2	1002
3	1001

The *OrderItem* table remains the same. Now, all tables are in 3NF with no transitive dependencies.

Exercise 4 (Basic SQL Queries - Music Database)**.**
Answer the following questions by writing appropriate SQL queries:

1. Find all customers from the country "USA".
2. Find all tracks with the word "Love" in their name.
3. Find all tracks with a price between $0.99 and $1.99.
4. List all albums whose title starts with "The".
5. Find all customers whose first name ends with the letter "a".
6. List all genres with names containing the letter "R".
7. Find all tracks with a length greater than 300,000 milliseconds.
8. List all customers who are not from "Canada".
9. Find the top 5 most expensive tracks.
10. List all tracks where the composer is either "Mozart" or "Beethoven".

Solution.

```sql
-- Exercise 4: music.db

-- 1. Find all customers from the country "USA".
SELECT * FROM Customers WHERE Country = 'USA';

-- 2. Find all tracks with the word "Love" in their name.
SELECT * FROM Tracks WHERE Name LIKE '%Love%';

-- 3. Find all tracks with a price between £0.99 and £1.99.
SELECT * FROM Tracks WHERE UnitPrice BETWEEN 0.99 AND 1.99;

-- 4. List all albums whose title starts with "The".
SELECT * FROM Albums WHERE Title LIKE 'The%';

-- 5. Find all customers whose first name ends with the letter "a".
SELECT * FROM Customers WHERE FirstName LIKE '%a';

-- 6. List all genres with names containing the letter "R".
SELECT * FROM Genres WHERE Name LIKE '%R%';

-- 7. Find all tracks with a length greater than 300,000 milliseconds.
SELECT * FROM Tracks WHERE Milliseconds > 300000;

-- 8. List all customers who are not from "Canada".
SELECT * FROM Customers WHERE Country != 'Canada';

-- 9. Find the top 5 most expensive tracks.
SELECT * FROM Tracks ORDER BY UnitPrice DESC LIMIT 5;

-- 10. List all tracks where the composer is either "Mozart" or "Beethoven".
SELECT * FROM Tracks WHERE Composer IN ('Mozart', 'Beethoven');
```

Exercise 5 (Further SQL Queries - Northwind Database)**.**
Add the Northwind database (located in `tutorials/data/`) and answer the following questions by writing appropriate SQL queries:

1. Find all customers whose last names start with "S" and who are from "USA".
2. List all products that cost more than $20 and have a name containing the letter "a".
3. Find all orders placed between January 1, 2016, and December 31, 2017, with freight charges less than $20.
4. List all employees whose titles contain the word "Manager" but are not located in "London".
5. Find the names of customers whose companies have the word "Food" in their name but are not located in "France".
6. List all products where the unit price is greater than $15 and the quantity in stock is less than 10.
7. Find all suppliers whose contact names start with "M" and whose company name ends with "Inc".
8. List all employees whose first names contain exactly 5 letters.
9. Find all orders where the freight charge is not between $20 and $100.
10. List the names of the top 5 most expensive products, sorted in descending order by price.

Solution.

```sql
-- Exercise 5: northwind.db

-- 1. Find all customers whose last names start with "S" and who are from "USA".
SELECT *
FROM Customers
WHERE ContactName LIKE '% S%' AND Country = 'USA';

-- 2. List all products that cost more than £20 and have a name containing the
↪  letter "a".
SELECT *
FROM Products
WHERE UnitPrice > 20 AND ProductName LIKE '%a%';

-- 3. Find all orders placed between January 1, 2016, and December 31, 2017, with
↪  freight charges less than £20.
SELECT *
FROM Orders
WHERE OrderDate BETWEEN '2016-01-01' AND '2017-12-31' AND Freight < 20;

-- 4. List all employees whose titles contain the word "Manager" but are not
↪  located in "London".
SELECT *
FROM Employees
WHERE Title LIKE '%Manager%' AND City != 'London';
```

```sql
-- 5. Find the names of customers whose companies have the word "Food" in their
↪  name but are not located in "France".
SELECT CompanyName
FROM Customers
WHERE CompanyName LIKE '%Food%' AND Country != 'France';

-- 6. List all products where the unit price is greater than £15 and the quantity
↪  in stock is less than 10.
SELECT *
FROM Products
WHERE UnitPrice > 15 AND UnitsInStock < 10;

-- 7. Find all suppliers whose contact names start with "M" and whose company
↪  name ends with "Inc".
SELECT *
FROM Suppliers
WHERE ContactName LIKE 'M%' AND CompanyName LIKE '%Inc';

-- 8. List all employees whose first names contain exactly 5 letters.
SELECT *
FROM Employees
WHERE FirstName LIKE '_____';

-- 9. Find all orders where the freight charge is not between £20 and £100.
SELECT *
FROM Orders
WHERE Freight NOT BETWEEN 20 AND 100;

-- 10. List the names of the top 5 most expensive products, sorted in descending
↪  order by price.
SELECT ProductName, UnitPrice
FROM Products
ORDER BY UnitPrice DESC
LIMIT 5;
```

Exercise 6 (**Optional:** Additional SQL Resources)**.**
To enhance your SQL skills and deepen your understanding of database normalization, complete the following tasks:

1. Explore SQL Learning Platforms:
   - Visit SQLBolt for interactive lessons on SQL basics and queries.
   - Practice diverse SQL exercises on SQLZoo, a comprehensive resource with guided examples.
   - Challenge yourself with advanced SQL exercises on Data Lemur.
   - If you feel confident, explore the LeetCode SQL Problems or the Mode Analytics SQL Tutorial for real-world challenges.
2. Understand Database Normal Forms:
   - Watch the video on database normalization (Understanding Normal Forms) to learn about organizing databases efficiently.