Prompt

Database

```
-- Database: `rentman`

CREATE TABLE `equipment` (
  `id` int(11) NOT NULL,
  `modified` datetime DEFAULT NULL,
  `modified_db` timestamp NOT NULL DEFAULT current_timestamp(),
  `displayname` varchar(128) DEFAULT NULL,
  `folder` int(11) DEFAULT NULL,
  `code` varchar(8) DEFAULT NULL,
  `in_shop` int(11) DEFAULT NULL,
  `surface_article` int(11) DEFAULT NULL,
  `shop_description_short` varchar(128) DEFAULT NULL,
  `shop_description_long` text DEFAULT NULL,
  `shop_seo_title` varchar(80) DEFAULT NULL,
  `shop_seo_keyword` varchar(80) DEFAULT NULL,
  `shop_seo_description` varchar(80) DEFAULT NULL,
  `shop_featured` int(11) DEFAULT NULL,
  `price` float DEFAULT NULL,
  `subrental_costs` int(11) DEFAULT NULL,
  `critical_stock_level` int(11) DEFAULT NULL,
  `type` varchar(80) DEFAULT NULL,
  `rental_sales` varchar(80) DEFAULT NULL,
  `volume` int(11) DEFAULT NULL,
  `packed_per` int(11) DEFAULT NULL,
  `height` int(11) DEFAULT NULL,
  `width` int(11) DEFAULT NULL,
  `length` int(11) DEFAULT NULL,
  `weight` int(11) DEFAULT NULL,
  `empty_weight` int(11) DEFAULT NULL,
  `power` int(11) DEFAULT NULL,
  `current` int(11) DEFAULT NULL,
  `image` varchar(40) DEFAULT NULL,
  `image_path` varchar(255) DEFAULT NULL,
```

```
  `is_combination` int(11) DEFAULT NULL,
  `is_physical` varchar(80) DEFAULT NULL,
  `tags` varchar(128) DEFAULT NULL,
  `tag_hoofditem` tinyint(1) NOT NULL DEFAULT 0,
  `tag_set` tinyint(1) NOT NULL DEFAULT 0,
  `location_in_warehouse` varchar(80) DEFAULT NULL,
  `current_quantity` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `equipmentsetscontent` (
  `id` int(11) NOT NULL,
  `modified` text DEFAULT NULL,
  `displayname` text DEFAULT NULL,
  `quantity` int(11) DEFAULT NULL,
  `parent_equipment` text DEFAULT NULL,
  `order` int(11) DEFAULT NULL,
  `equipment` text DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `plannedequipment` (
  `id` int(11) NOT NULL,
  `modified` datetime DEFAULT NULL,
  `displayname` varchar(128) DEFAULT NULL,
  `equipment` int(11) DEFAULT NULL,
  `quantity` int(11) DEFAULT NULL,
  `quantity_total` int(11) DEFAULT NULL,
  `equipment_group` int(11) DEFAULT NULL,
  `planperiod_start` varchar(26) DEFAULT NULL,
  `planperiod_end` varchar(26) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `reservations` (
  `equipment_id` int(11) NOT NULL,
  `current_quantity` int(11) DEFAULT NULL,
  `year` int(11) NOT NULL,
  `month` int(11) NOT NULL,
  `availability` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL CHECK
(json_valid(`availability`))
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```

Class

```
class AvailabilityCalculator(DatabaseBaseClass):

  def __init__(self, config_path="../config/database.txt"):
    super().__init__()

  def get_ultimate_parents(self):
    self.cursor.execute("SELECT DISTINCT parent_equipment FROM equipmentsetscontent")
    parents = set(row[0] for row in self.cursor.fetchall())
    self.cursor.execute("SELECT DISTINCT equipment FROM equipmentsetscontent")
    children = set(row[0] for row in self.cursor.fetchall())

    # Parents that are not children themselves
    ultimate_parents = parents - children

    return list(ultimate_parents)
```

Reservations look like this:

```
3047, 3, 2025, 5, [3, 1, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3..., 2]
3048, 3, 2025, 5, [3, 3, 3, 3, 3, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3..., 3]
3043, 4, 2025, 5, [4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4..., 4]
3031, 1, 2025, 5, [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1..., 1]
....
```

The reservations are a list of numbers. Each number represents a day for that month (in the example May) and the number tells how many of this equipment is available.

Question

I want add a method: get_set_parts(equipment_id)

This method will select all children of the parent with equipment_id. Use table equipmentsetscontent to get this. Per child we need to store the quantity.quantity

Then if there's a reservation record for this child-equipment_id, add all these reservations to this child.

The output of the function needs to be something like this (please create a proper data structure).

```
parent: 3029
  child: 1086
    quantity: 12
    2025-06: 1,1,1,2,3,4,4,4,0,......
    2025-07: 1,1,1,2,3,4,4,4,0,......
    etc, etc.
parent: 2029
  child: 1087
    quantity: 10
    2025-06: 1,1,1,2,3,4,4,4,0,......
    2025-07: 1,1,1,2,3,4,4,4,0,......
    etc, etc.
  child: 1091
    quantity: 10
    2025-06: 1,1,1,2,1,1,1,1,0,......
    2025-08: 1,1,1,2,3,4,4,0,0,......
    etc, etc.
etc, etc
```

Do you need additional information?

```
import mysql.connector
import json

class DatabaseBaseClass:
    """A base class to handle database connections."""
    def __init__(self, host="localhost", user="root", password="", database="rentman"):
        try:
            self.db = mysql.connector.connect(
                host=host,
                user=user,
                password=password,
                database=database
            )
            self.cursor = self.db.cursor()
        except mysql.connector.Error as err:
```

```
            print(f"Error connecting to database: {err}")
            # In a real application, you might want to handle this more gracefully
            self.db = None
            self.cursor = None

    def __del__(self):
        if self.db and self.db.is_connected():
            self.cursor.close()
            self.db.close()

class AvailabilityCalculator(DatabaseBaseClass):

    def __init__(self, host="localhost", user="root", password="", database="rentman"):
        super().__init__(host, user, password, database)

    def get_ultimate_parents(self):
        """
        Gets a list of all parent equipment that are not themselves children of another set.
        """
        if not self.cursor:
            return []
        self.cursor.execute("SELECT DISTINCT parent_equipment FROM equipmentsetscontent")
        parents = set(row[0] for row in self.cursor.fetchall())
        self.cursor.execute("SELECT DISTINCT equipment FROM equipmentsetscontent")
        children = set(row[0] for row in self.cursor.fetchall())

        # Parents that are not children themselves
        ultimate_parents = parents - children

        return list(ultimate_parents)

    def get_set_parts(self, equipment_id: int):
        """
        Retrieves all child equipment for a given parent equipment_id,
        including their quantities and monthly reservation data.

        Args:
            equipment_id: The ID of the parent equipment.

        Returns:
```

```
            A dictionary containing the parent's children and their details,
            structured for clarity and easy access. Returns an empty dictionary
            if the parent ID is not found or has no children.
        """
        if not self.cursor:
            return {}

        # The main dictionary to hold the final structured data.
        result = {}

        # Step 1: Find all children of the given parent_equipment and their quantities.
        # The 'equipment' and 'parent_equipment' columns are TEXT, so we search for the ID as a string.
        query_children = """
            SELECT equipment, quantity
            FROM equipmentsetscontent
            WHERE parent_equipment = %s
        """
        self.cursor.execute(query_children, (str(equipment_id),))
        children = self.cursor.fetchall()

        if not children:
            return {} # Return empty if no children are found

        # Initialize the parent key in the result dictionary.
        result[equipment_id] = {'children': {}}

        # Step 2: For each child, fetch its reservation data.
        for child_id_str, quantity in children:
            try:
                # The child_id from `equipmentsetscontent` is TEXT, needs to be INT for the next query.
                child_id = int(child_id_str)
            except (ValueError, TypeError):
                # Skip if the child ID is not a valid integer.
                continue

            child_data = {
                'quantity': quantity,
                'reservations': {}
```

```
            }

            # Query to get all reservations for the current child.
            query_reservations = """
                SELECT year, month, availability
                FROM reservations
                WHERE equipment_id = %s
                ORDER BY year, month
            """
            self.cursor.execute(query_reservations, (child_id,))
            reservations = self.cursor.fetchall()

            for year, month, availability_json in reservations:
                # Format the year and month into a 'YYYY-MM' key.
                date_key = f"{year}-{str(month).zfill(2)}"

                # The 'availability' is stored as a JSON string (e.g., '[3, 1, 0, ...]').
                # We load it, convert each number to a string, and join with commas.
                try:
                    availability_list = json.loads(availability_json)
                    availability_str = ",".join(map(str, availability_list))
                    child_data['reservations'][date_key] = availability_str
                except json.JSONDecodeError:
                    # Handle cases where the JSON is invalid.
                    child_data['reservations'][date_key] = "Error: Invalid availability
format"

            # Add the complete child data to the result.
            result[equipment_id]['children'][child_id] = child_data

        return result
```

On the top level, the level of the parent I want to add an extra field.`
perform
select current_quantity from equipment where id = equipment_id (the function parameter).
and store the current_quantity on the level of the parent.

Change code to add this to the data structure.

The output of this method is

```
{3024: {'children': {
    3025: {'quantity': 1,
        'available': { '2025-05':
'1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0',
                       '2025-06':
'0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0',
                       '2025-07':
'1,1,1,1,0,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,1,1,1,1,1,1,1,0,0,0',
                       '2025-08':
'0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0',
                       '2025-09':
'0,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,0,1,1,1,1,0,0,0,0,0,0,1',
                       '2025-10':
'1,1,0,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1'}},
     3026: {'quantity': 4,
        'available': { '2025-05':
'4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0',
                       '2025-06':
'0,0,0,0,0,0,0,0,0,0,4,0,0,0,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0',
                       '2025-07':
'4,4,4,4,0,0,0,4,4,0,0,0,0,0,4,4,4,0,0,0,0,4,4,4,4,4,4,4,0,0,0',
                       '2025-08':
'0,0,0,0,4,0,0,0,0,0,0,0,0,4,0,0,0,0,4,4,0,0,0,0,0,0,0,0,0,0,0',
                       '2025-09':
'0,4,4,0,0,0,0,0,4,0,0,0,0,0,0,4,4,4,0,4,4,4,4,0,0,0,0,0,0,4',
                       '2025-10':
'4,4,0,4,4,4,4,4,4,0,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4'}},
   3027: {'quantity': 1,
        'available': {'2025-05':
'2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,0,0,0,0,0',
                        '2025-06':
'0,0,1,0,0,0,1,1,1,1,1,0,0,0,1,2,2,1,0,0,0,0,1,0,0,0,0,0,0,0',
                       '2025-07':
'2,2,0,0,0,0,0,2,2,1,1,1,1,1,2,2,2,1,1,1,1,2,2,2,2,2,2,2,1,1,1',
                       '2025-08':
'1,1,1,1,2,1,1,1,1,1,1,1,1,2,1,1,1,1,2,2,1,0,0,0,0,1,1,0,0,0,0',
                       '2025-09':
'0,2,2,1,1,1,1,1,2,1,1,0,0,0,0,2,2,2,1,2,2,2,2,0,0,0,0,0,0,2',
                       '2025-10':
'2,2,1,2,2,2,2,2,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2'}},
```

```
    5641: {'quantity': 3,
        'available': { '2025-05':
'50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,47,47,47,46,46'
,
                       '2025-06':
'46,46,47,47,47,47,47,47,47,47,50,47,47,47,50,50,50,47,47,47,47,47,47,47,47,47,47,47,47',
                       '2025-07':
'50,50,50,50,47,47,47,50,50,47,47,47,47,47,50,50,50,47,47,47,47,50,50,50,50,50,50,47,47,47'
,
                       '2025-08':
'47,47,47,47,50,47,47,47,47,47,47,47,47,50,47,47,47,47,50,50,47,47,47,47,47,47,47,47,47,47'
,
                       '2025-09':
'47,50,50,47,47,47,47,47,50,47,47,47,47,47,47,50,50,50,47,50,50,50,50,47,47,47,47,47,47,50',
                       '2025-10':
'50,50,47,50,50,50,50,50,50,47,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50,50'
}},
 10857: {'quantity': 5,
       'available': {' 2025-06':
'10,5,5,10,7,7,7,5,5,8,4,4,4,4,4,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10',
                      '2025-07':
'10,10,7,7,2,2,2,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10',
                      '2025-08':
'10,10,10,10,10,10,10,10,10,10,10,10,10,10,5,5,5,5,10,10,10,10,10,10,10,10,10,10,10,10',
                      '2025-09':
'10,10,10,5,5,5,5,5,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10'}}},
'current_quantity': 2}}
```

We changed 'reservations' into 'available', because this reflects the values better.

The interpretation of the data is as follows.

We have equipment with id 3024, the current_quantity is 2.

We see that we need 1 times child 3025 in order to complete the set 3024. In may we have 1 available and the last 5 days of the month we have 0 available. This means that we can create only 1 set 3024 in may and the last 5 days of May we have 0 availability of 3024.

We also need 4 times 3026 and we have 4 available in the first part of May. Meaning we can create one set 3024.

Do you understand the interpretation?

Result


```
{3024: {'available': {'2025-05':
'1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0',
                      '2025-06':
'0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0',
                      '2025-07':
'1,1,0,0,0,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,1,1,1,1,1,1,1,0,0,0',
                      '2025-08':
'0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0',
                      '2025-09':
'0,1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,0,1,1,1,1,0,0,0,0,0,0,1',
                      '2025-10':
'1,1,0,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1'},
```

Revision #7
Created 2025-06-10 19:24:29 UTC by Max
Updated 2025-06-10 21:43:48 UTC by Max