Use Python and ascyio to implement a client that connects to a server and completes a task.

1. The client will open a primary connection (on port 51300) at the beginning of the program.
2. The client will send an id message (`id yz605`) to the server on the primary connection.
3. The server will send a query message (`query {port}`) to the client on the primary connection. (port range: 51301-51350)
4. The client will establish a new connection (secondary connection) to the server at this specified TCP port.
5. The server will send a new query message (`query {port}`) to this secondary connection, which directs the client to the next port. (port range: 51301-51350)
6. The client either (1) establishes a new connection to the server at this new port if such a connection does not exist, or (2) send a new identification message to an existing connection to the specified port.
7. The server may send a listen message (`listen {port}`) to the client on the current secondary connection. (port range: 51401-51450)
8. For each listen message, the client should open a listening socket and wait for a connection.
9. After the server establishes a connection, it may send multiple query messages to the client, which is required to identify itself with an id message at the port specified in each query message.
10. Keep the listening socket always open and listen to multiple query messages. Repeat steps 5-9 until the server sends a status message (`status {message}`) to the client on the primary connection.
11. Close all connections and exit the program.

Note that:

- Every message ends with a newline character; and the next message starts right after the newline character. All text should be converted to bytes using plain ASCII; and bytes received over the network should be converted to ASCII text.

Can you complete this task using Python and the asyncio library?

I can provide a program that you can start with (this program does not proceed the listen message). You can use this program as a starting point and modify it to complete the task.

```
import asyncio
import socket

HOST = 'vcm-32603.vm.duke.edu'
PORT = 51200
ID = 'yz605'

connected_ports = []
secondary_connections = {}

async def handle_primary_connection(reader, writer):
    print("Connected to server on primary connection")
    writer.write(f"id {ID}\n".encode('ascii'))
    await writer.drain()
    data = await reader.readline()
    port = int(data.decode('ascii').strip().split()[1])

    try:
        secondary_task =
asyncio.create_task(handle_secondary_connection(port, connected_ports))
        status_task =
asyncio.create_task(handle_status_messages(reader))

        await asyncio.wait([secondary_task, status_task],
return_when=asyncio.FIRST_COMPLETED)

        for task in [secondary_task, status_task]:
            if not task.done():
                task.cancel()
    except asyncio.CancelledError:
        writer.close()
        await writer.wait_closed()

    print(f"The connected ports are: {connected_ports}")

async def handle_secondary_connection(port, connected_ports):
    while True:
        if port in connected_ports:
            reader, writer = secondary_connections[port]
        else:
            reader, writer = await asyncio.open_connection(HOST, port)
            connected_ports.append(port)
            secondary_connections[port] = reader, writer

        print(f"Connected to server on secondary connection to port
{port}")
        writer.write(f"id {ID}\n".encode('ascii'))
        await writer.drain()

        data = await reader.readline()
        if not data:
            writer.close()
            await writer.wait_closed()
            break

        new_port = int(data.decode('ascii').strip().split()[1])
        port = new_port

async def handle_status_messages(reader):
    while True:
        data = await reader.readline()
        if not data:
            break
        message = data.decode('ascii').strip()
        print(f"Received status message: {message}")
        for task in asyncio.all_tasks():
            task.cancel()

async def main():
    try:
        reader, writer = await asyncio.open_connection(HOST, PORT)
        await handle_primary_connection(reader, writer)
    except (ConnectionRefusedError, socket.gaierror):
        print("Failed to connect to server")

if __name__ == '__main__':
    asyncio.run(main())
```