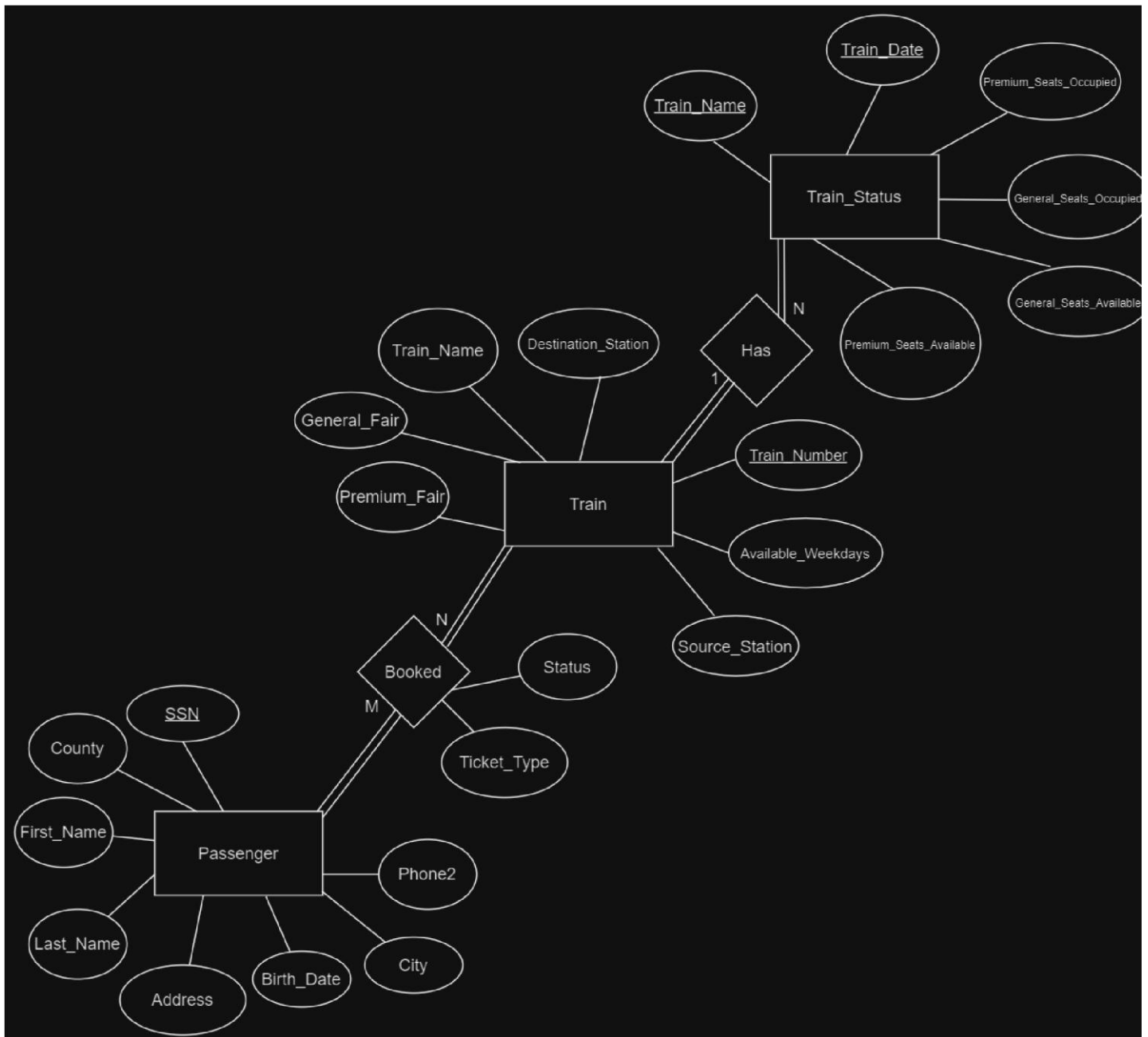


Project 1 - CSE 3330-004 – Project #1 – Railway Reservation System

Dev Patel

ER DIAGRAM



LOAD DATA METHOD

To load the data into the tables, we used the INSERT INTO SQL command. This method explicitly adds rows of data into the specified tables by defining the columns and their corresponding values. Each INSERT INTO statement consists of the table name followed by a list of column names and a VALUES clause that includes the data for each column in the order they are defined.

For example, for our project we have 4 INSERT INTO SQL commands, one for each table, which are

- 1) INSERT INTO Passenger (First_Name, Last_Name, Address, City, County, Phone2, SSN, Birth_Date) VALUES
- 2) INSERT INTO Booked (Passenger_SSN, Train_Number, Ticket_Type, Status) VALUES
- 3) INSERT INTO Train (Train_Number, Train_Name, Premium_Fair, General_Fair, Source_Station, Destination_Station, Available_Weekdays) VALUES
- 4) INSERT INTO Train_status (Train_Date, Train_Name, Premium_Seats_Available, General_Seats_Available, Premium_Seats_Occupied, General_Seats_Occupied) VALUES

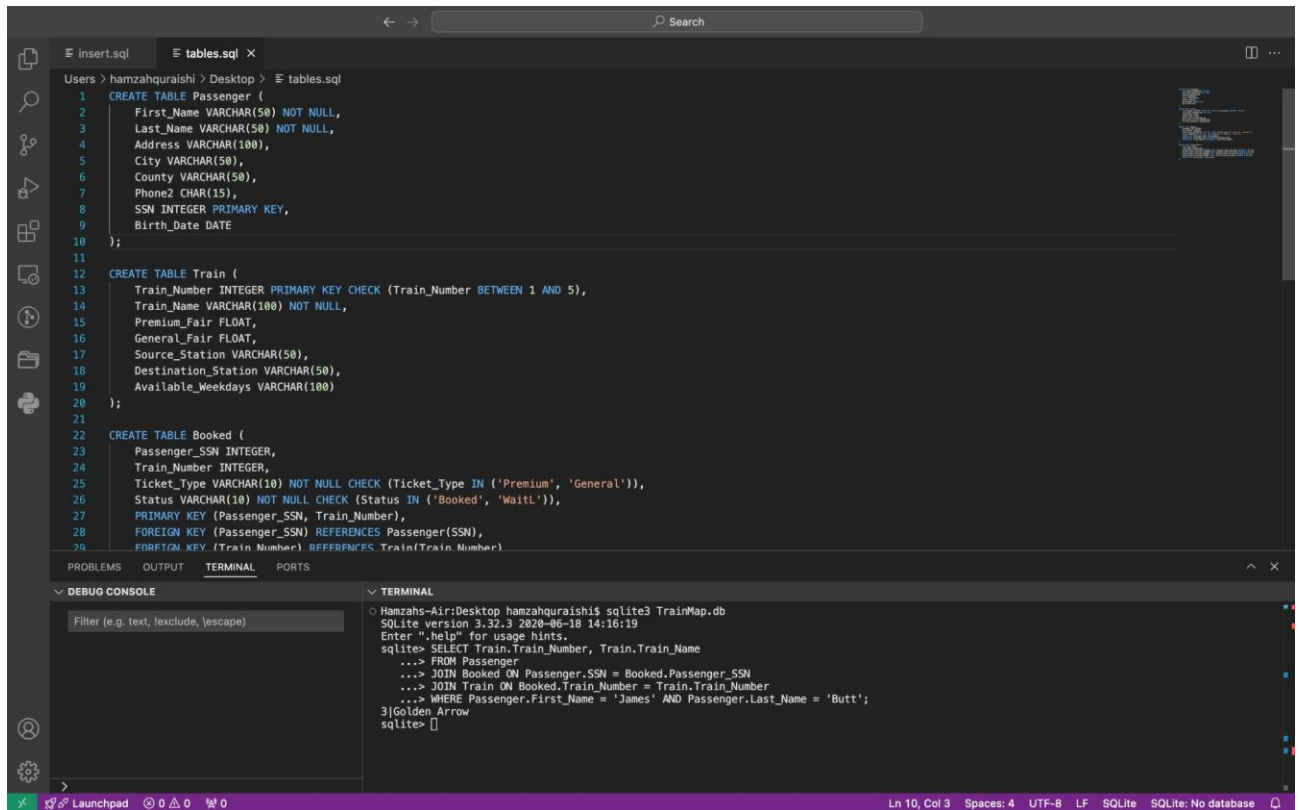
For each table, separate INSERT INTO statements were constructed to load the respective data, ensuring that each row of data matches the schema of the table (taken from the provided zip file containing 4 excel files on Canvas submission module for Project 1). This approach allows for precise insertion of multiple records all at once into the database tables.

ReadMe (TOOLS USED FOR PROJECT)

- Visual Studio Code version 1.93.1 (IDE to run program)
- SQLiteStudio version 3.4.4 (to access database)
- SQLite version 0.14.1 (SQL extension from VSCode)
- draw.io (to create ER Diagram)

SQL SELECT STATEMENTS - QUERY RESULT SCREENSHOTS

- 1) Given a passenger's last name and first name and retrieve all trains they are booked on.
- used James Butt as an example to run query (specified custom input for name is correct/allowed)



The screenshot displays a code editor with three tabs: 'insert.sql', 'tables.sql', and 'tables.sql'. The 'tables.sql' tab is active, showing the following SQL code:

```
1 CREATE TABLE Passenger (  
2     First_Name VARCHAR(50) NOT NULL,  
3     Last_Name VARCHAR(50) NOT NULL,  
4     Address VARCHAR(100),  
5     City VARCHAR(50),  
6     County VARCHAR(50),  
7     Phone2 CHAR(15),  
8     SSN INTEGER PRIMARY KEY,  
9     Birth_Date DATE  
10 );  
11  
12 CREATE TABLE Train (  
13     Train_Number INTEGER PRIMARY KEY CHECK (Train_Number BETWEEN 1 AND 5),  
14     Train_Name VARCHAR(100) NOT NULL,  
15     Premium_Fair FLOAT,  
16     General_Fair FLOAT,  
17     Source_Station VARCHAR(50),  
18     Destination_Station VARCHAR(50),  
19     Available_Weekdays VARCHAR(100)  
20 );  
21  
22 CREATE TABLE Booked (  
23     Passenger_SSN INTEGER,  
24     Train_Number INTEGER,  
25     Ticket_Type VARCHAR(10) NOT NULL CHECK (Ticket_Type IN ('Premium', 'General')),  
26     Status VARCHAR(10) NOT NULL CHECK (Status IN ('Booked', 'WaitL')),  
27     PRIMARY KEY (Passenger_SSN, Train_Number),  
28     FOREIGN KEY (Passenger_SSN) REFERENCES Passenger(SSN),  
29     FOREIGN KEY (Train_Number) REFERENCES Train(Train_Number)  
30 );
```

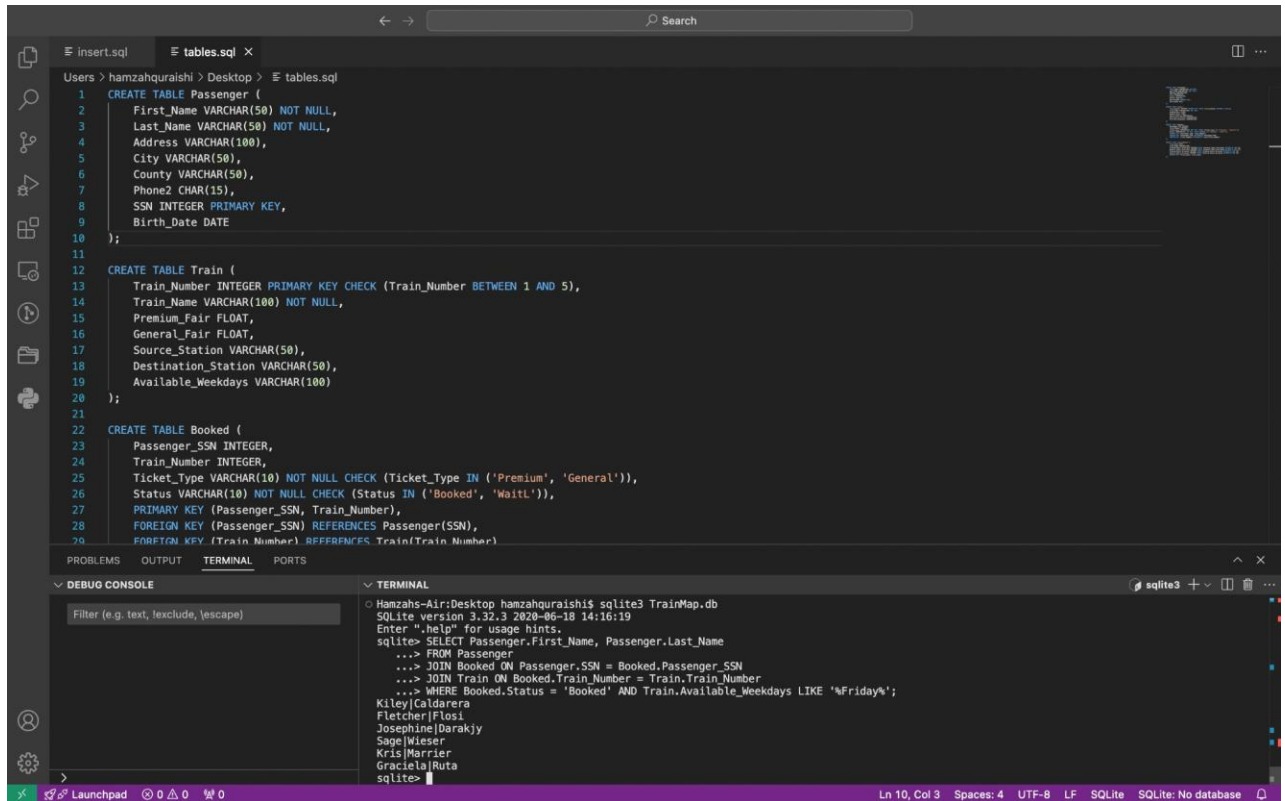
The terminal window at the bottom shows the execution of the query:

```
Hamzahs-Air:Desktop hamzahquraishi$ sqlite3 TrainMap.db  
SQLite version 3.32.3 2020-06-18 14:16:19  
Enter ".help" for usage hints.  
sqlite> SELECT Train.Train_Number, Train.Train_Name  
...> FROM Passenger  
...> JOIN Booked ON Passenger.SSN = Booked.Passenger_SSN  
...> JOIN Train ON Booked.Train_Number = Train.Train_Number  
...> WHERE Passenger.First_Name = 'James' AND Passenger.Last_Name = 'Butt';  
3|Golden Arrow  
sqlite> []
```

The status bar at the bottom indicates the current position is Ln 10, Col 3, with 4 spaces, UTF-8 encoding, LF line endings, and the SQLite database is currently empty.

2)

Given a day list the passengers traveling on that day with confirmed tickets. - used Friday as example to run query (specified custom input for name is correct/allowed - TA Priyanka)



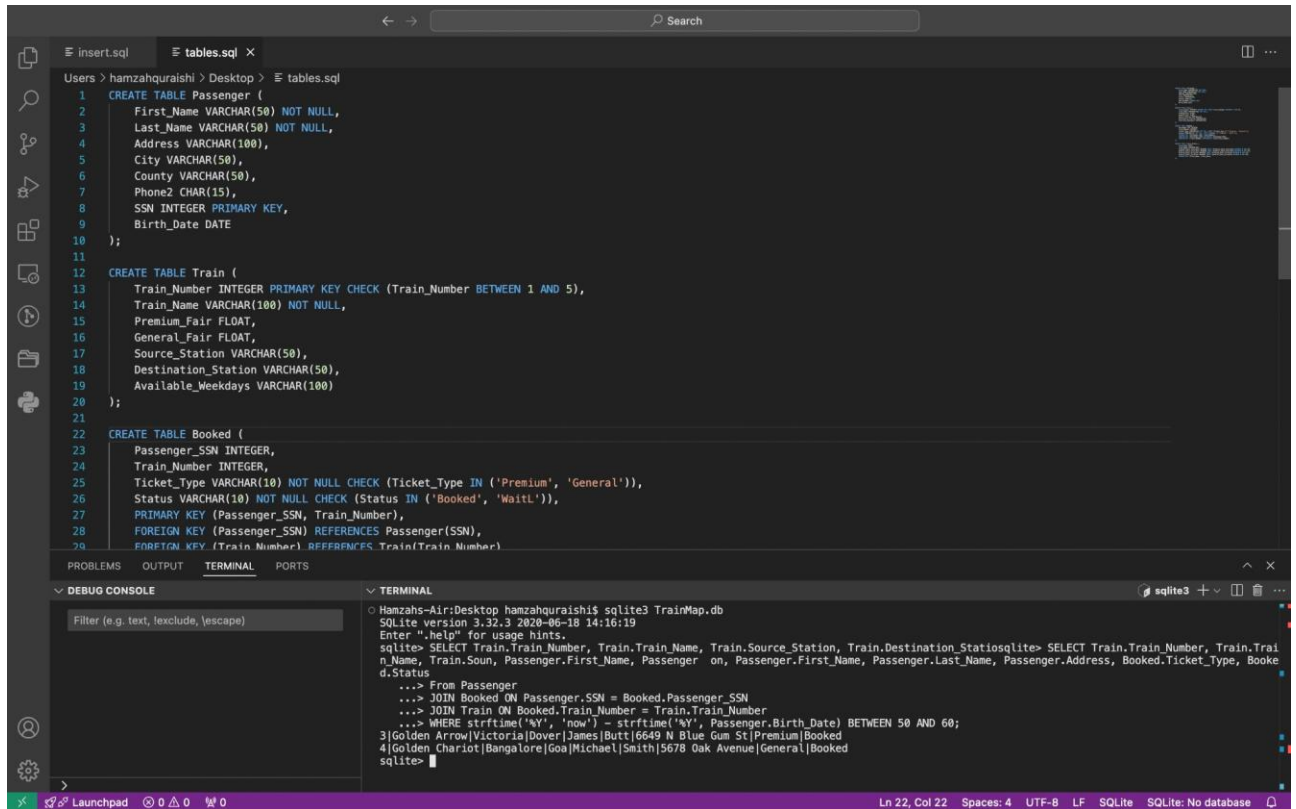
The screenshot shows a SQLite IDE interface with a dark theme. The main editor displays the SQL schema for three tables: Passenger, Train, and Booked. The Passenger table has columns: First_Name (VARCHAR(50) NOT NULL), Last_Name (VARCHAR(50) NOT NULL), Address (VARCHAR(100)), City (VARCHAR(50)), County (VARCHAR(50)), Phone2 (CHAR(15)), SSN (INTEGER PRIMARY KEY), and Birth_Date (DATE). The Train table has columns: Train_Number (INTEGER PRIMARY KEY CHECK (Train_Number BETWEEN 1 AND 5)), Train_Name (VARCHAR(100) NOT NULL), Premium_Fair (FLOAT), General_Fair (FLOAT), Source_Station (VARCHAR(50)), Destination_Station (VARCHAR(50)), and Available_Weekdays (VARCHAR(100)). The Booked table has columns: Passenger_SSN (INTEGER), Train_Number (INTEGER), Ticket_Type (VARCHAR(10) NOT NULL CHECK (Ticket_Type IN ('Premium', 'General'))), and Status (VARCHAR(10) NOT NULL CHECK (Status IN ('Booked', 'WaitL'))). It also includes foreign key constraints for Passenger_SSN and Train_Number. Below the editor, the TERMINAL pane shows the execution of a query: `sqlite> SELECT Passenger.First_Name, Passenger.Last_Name FROM Passenger ...> JOIN Booked ON Passenger.SSN = Booked.Passenger_SSN ...> JOIN Train ON Booked.Train_Number = Train.Train_Number ...> WHERE Booked.Status = 'Booked' AND Train.Available_Weekdays LIKE '%Friday%';` The result of the query is displayed in the terminal, listing the first and last names of passengers with confirmed tickets for Friday: Kiley|Calderera, Fletcher|Flosi, Josephine|Darakjy, Sage|Wieser, Kris|Marrier, and Graciela|Ruta.

```
Users > hamzahquraishi > Desktop > tables.sql
1 CREATE TABLE Passenger (
2   First_Name VARCHAR(50) NOT NULL,
3   Last_Name VARCHAR(50) NOT NULL,
4   Address VARCHAR(100),
5   City VARCHAR(50),
6   County VARCHAR(50),
7   Phone2 CHAR(15),
8   SSN INTEGER PRIMARY KEY,
9   Birth_Date DATE
10 );
11
12 CREATE TABLE Train (
13   Train_Number INTEGER PRIMARY KEY CHECK (Train_Number BETWEEN 1 AND 5),
14   Train_Name VARCHAR(100) NOT NULL,
15   Premium_Fair FLOAT,
16   General_Fair FLOAT,
17   Source_Station VARCHAR(50),
18   Destination_Station VARCHAR(50),
19   Available_Weekdays VARCHAR(100)
20 );
21
22 CREATE TABLE Booked (
23   Passenger_SSN INTEGER,
24   Train_Number INTEGER,
25   Ticket_Type VARCHAR(10) NOT NULL CHECK (Ticket_Type IN ('Premium', 'General')),
26   Status VARCHAR(10) NOT NULL CHECK (Status IN ('Booked', 'WaitL')),
27   PRIMARY KEY (Passenger_SSN, Train_Number),
28   FOREIGN KEY (Passenger_SSN) REFERENCES Passenger(SSN),
29   FOREIGN KEY (Train_Number) REFERENCES Train(Train_Number)
30 );

PROBLEMS OUTPUT TERMINAL PORTS
DEBUO CONSOLE
Filter (e.g. text, exclude, escape)
TERMINAL
Hamzahs-Air:Desktop hamzahquraishi$ sqlite3 TrainMap.db
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> SELECT Passenger.First_Name, Passenger.Last_Name
...> FROM Passenger
...> JOIN Booked ON Passenger.SSN = Booked.Passenger_SSN
...> JOIN Train ON Booked.Train_Number = Train.Train_Number
...> WHERE Booked.Status = 'Booked' AND Train.Available_Weekdays LIKE '%Friday%';
Kiley|Calderera
Fletcher|Flosi
Josephine|Darakjy
Sage|Wieser
Kris|Marrier
Graciela|Ruta
sqlite>
```

3)

Display the train information (Train Number, Train Name, Source and Destination) and passenger information (Name, Address, Category, ticket status) of passengers who are between the ages of 50 to 60.



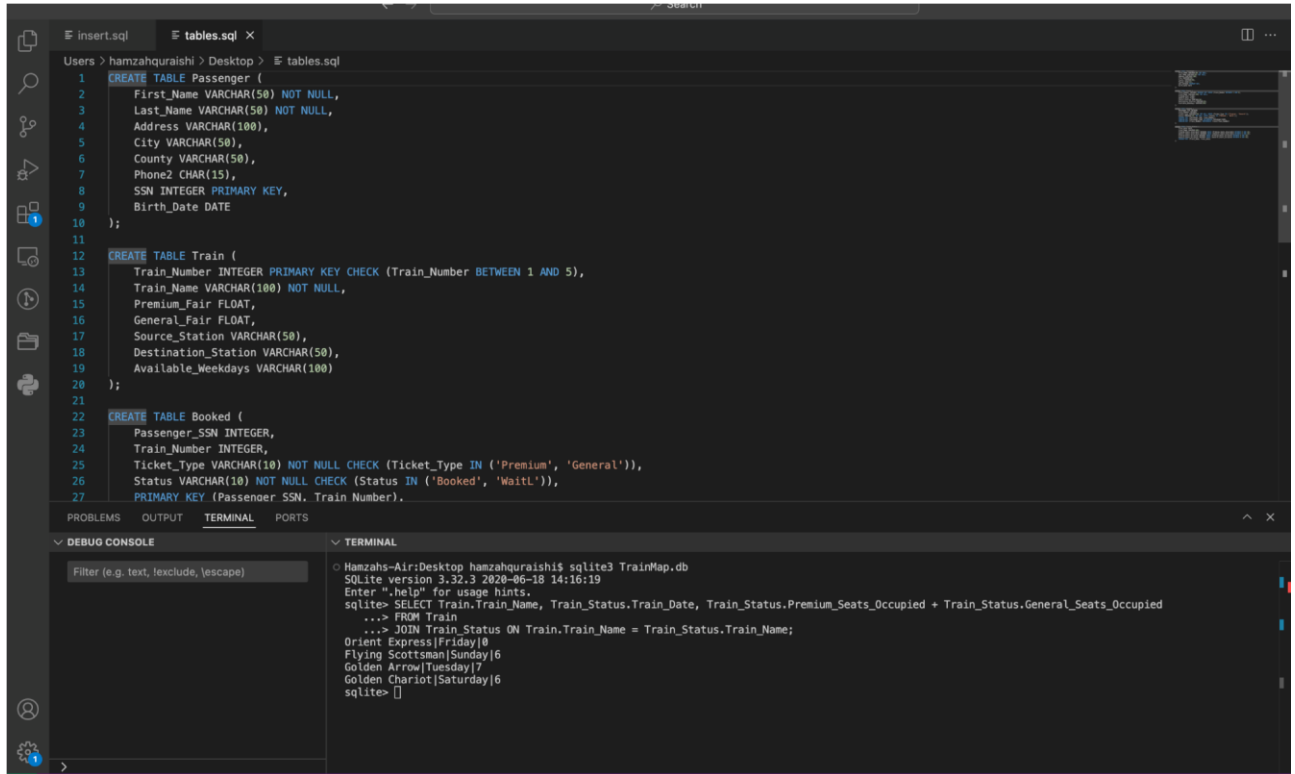
The screenshot shows a SQLite IDE interface with a dark theme. The main editor displays the SQL schema for three tables: Passenger, Train, and Booked. The Passenger table has columns for First_Name, Last_Name, Address, City, County, Phone2, SSN (Primary Key), and Birth_Date. The Train table has columns for Train_Number (Primary Key with a check constraint), Train_Name, Premium_Fair, General_Fair, Source_Station, Destination_Station, and Available_Weekdays. The Booked table has columns for Passenger_SSN, Train_Number, Ticket_Type, and Status, with a composite primary key on Passenger_SSN and Train_Number, and foreign key references to the Passenger and Train tables. Below the editor, the TERMINAL pane shows the execution of a query. The query selects train and passenger information for passengers aged between 50 and 60, joining the Train, Booked, and Passenger tables. The results show two rows: a Golden Arrow train from Victoria/Dover to James/Butt, and a Golden Chariot train from Bangalore/Goa to Michael/Smith.

```
1 CREATE TABLE Passenger (  
2     First_Name VARCHAR(50) NOT NULL,  
3     Last_Name VARCHAR(50) NOT NULL,  
4     Address VARCHAR(100),  
5     City VARCHAR(50),  
6     County VARCHAR(50),  
7     Phone2 CHAR(15),  
8     SSN INTEGER PRIMARY KEY,  
9     Birth_Date DATE  
10 );  
11  
12 CREATE TABLE Train (  
13     Train_Number INTEGER PRIMARY KEY CHECK (Train_Number BETWEEN 1 AND 5),  
14     Train_Name VARCHAR(100) NOT NULL,  
15     Premium_Fair FLOAT,  
16     General_Fair FLOAT,  
17     Source_Station VARCHAR(50),  
18     Destination_Station VARCHAR(50),  
19     Available_Weekdays VARCHAR(100)  
20 );  
21  
22 CREATE TABLE Booked (  
23     Passenger_SSN INTEGER,  
24     Train_Number INTEGER,  
25     Ticket_Type VARCHAR(10) NOT NULL CHECK (Ticket_Type IN ('Premium', 'General')),  
26     Status VARCHAR(10) NOT NULL CHECK (Status IN ('Booked', 'Waitl')),  
27     PRIMARY KEY (Passenger_SSN, Train_Number),  
28     FOREIGN KEY (Passenger_SSN) REFERENCES Passenger(SSN),  
29     FOREIGN KEY (Train_Number) REFERENCES Train(Train_Number)  
30 );
```

```
sqlite> SELECT Train.Train_Number, Train.Train_Name, Train.Source_Station, Train.Destination_Station,  
Train.Soun, Passenger.First_Name, Passenger on, Passenger.First_Name, Passenger.Last_Name, Passenger.Address, Booked.Ticket_Type, Booke  
d.Status  
...> From Passenger  
...> JOIN Booked ON Passenger.SSN = Booked.Passenger_SSN  
...> JOIN Train ON Booked.Train_Number = Train.Train_Number  
...> WHERE strftime('%Y', 'now') - strftime('%Y', Passenger.Birth_Date) BETWEEN 50 AND 60;  
3|Golden Arrow|Victoria|Dover|James|Butt|6649 N Blue Gum St|Premium|Booked  
4|Golden Chariot|Bangalore|Goa|Michael|Smith|5678 Oak Avenue|General|Booked  
sqlite>
```

4)

List train name, day and number of passengers on that train. - listed all 4 train names with day and number of passengers



The screenshot shows a SQLite IDE interface with a dark theme. The main editor window displays SQL code for creating three tables: Passenger, Train, and Booked. The Passenger table has columns for First_Name, Last_Name, Address, City, County, Phone2, SSN (Primary Key), and Birth_Date. The Train table has columns for Train_Number (Primary Key with a CHECK constraint), Train_Name, Premium_Fair, General_Fair, Source_Station, Destination_Station, and Available_Weekdays. The Booked table has columns for Passenger_SSN, Train_Number, Ticket_Type, and Status, with foreign key constraints to the Passenger and Train tables. Below the editor, the TERMINAL pane shows the execution of a query that joins the Train and Booked tables to list train names, days, and the number of passengers. The query result is displayed in the terminal, showing four trains: Orient Express, Flying Scotsman, Golden Arrow, and Golden Chariot, each with its respective day and passenger count.

```
1 CREATE TABLE Passenger (
2   First_Name VARCHAR(50) NOT NULL,
3   Last_Name VARCHAR(50) NOT NULL,
4   Address VARCHAR(100),
5   City VARCHAR(50),
6   County VARCHAR(50),
7   Phone2 CHAR(15),
8   SSN INTEGER PRIMARY KEY,
9   Birth_Date DATE
10 );
11
12 CREATE TABLE Train (
13   Train_Number INTEGER PRIMARY KEY CHECK (Train_Number BETWEEN 1 AND 5),
14   Train_Name VARCHAR(100) NOT NULL,
15   Premium_Fair FLOAT,
16   General_Fair FLOAT,
17   Source_Station VARCHAR(50),
18   Destination_Station VARCHAR(50),
19   Available_Weekdays VARCHAR(100)
20 );
21
22 CREATE TABLE Booked (
23   Passenger_SSN INTEGER,
24   Train_Number INTEGER,
25   Ticket_Type VARCHAR(10) NOT NULL CHECK (Ticket_Type IN ('Premium', 'General')),
26   Status VARCHAR(10) NOT NULL CHECK (Status IN ('Booked', 'WaitL')),
27   PRIMARY KEY (Passenger_SSN, Train_Number);
```

DEBUG CONSOLE

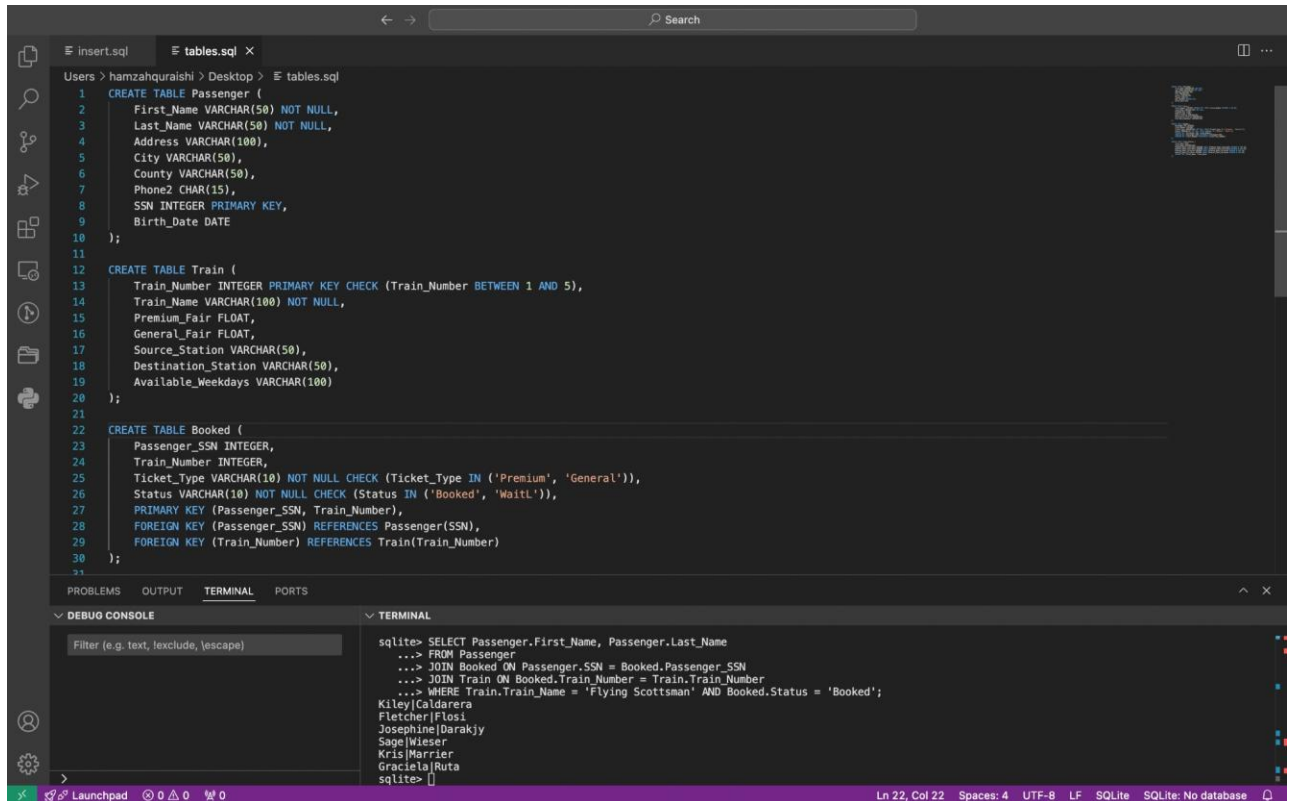
Filter (e.g. text, exclude, _escape)

TERMINAL

```
Hamzahs-Air:Desktop hamzahquraishi$ sqlite3 TrainMap.db
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> SELECT Train.Train_Name, Train_Status.Train_Date, Train_Status.Premium_Seats_Occupied + Train_Status.General_Seats_Occupied
...> FROM Train
...> JOIN Train_Status ON Train.Train_Name = Train_Status.Train_Name;
Orient Express|Friday|0
Flying Scotsman|Sunday|6
Golden Arrow|Tuesday|7
Golden Chariot|Saturday|6
sqlite> []
```

5)

Enter a train name and retrieve all the passengers with confirmed status traveling on that train. - used *Flying Scotsman* as an example for the train (specified custom input for name is correct/allowed)



The screenshot shows a code editor with three tabs: 'insert.sql', 'tables.sql', and 'tables.sql'. The 'tables.sql' tab is active, displaying the following SQL code:

```
1 CREATE TABLE Passenger (  
2   First_Name VARCHAR(50) NOT NULL,  
3   Last_Name VARCHAR(50) NOT NULL,  
4   Address VARCHAR(100),  
5   City VARCHAR(50),  
6   County VARCHAR(50),  
7   Phone2 CHAR(15),  
8   SSN INTEGER PRIMARY KEY,  
9   Birth_Date DATE  
10 );  
11  
12 CREATE TABLE Train (  
13   Train_Number INTEGER PRIMARY KEY CHECK (Train_Number BETWEEN 1 AND 5),  
14   Train_Name VARCHAR(100) NOT NULL,  
15   Premium_Fair FLOAT,  
16   General_Fair FLOAT,  
17   Source_Station VARCHAR(50),  
18   Destination_Station VARCHAR(50),  
19   Available_Weekdays VARCHAR(100)  
20 );  
21  
22 CREATE TABLE Booked (  
23   Passenger_SSN INTEGER,  
24   Train_Number INTEGER,  
25   Ticket_Type VARCHAR(10) NOT NULL CHECK (Ticket_Type IN ('Premium', 'General')),  
26   Status VARCHAR(10) NOT NULL CHECK (Status IN ('Booked', 'Waitl')),  
27   PRIMARY KEY (Passenger_SSN, Train_Number),  
28   FOREIGN KEY (Passenger_SSN) REFERENCES Passenger(SSN),  
29   FOREIGN KEY (Train_Number) REFERENCES Train(Train_Number)  
30 );  
31
```

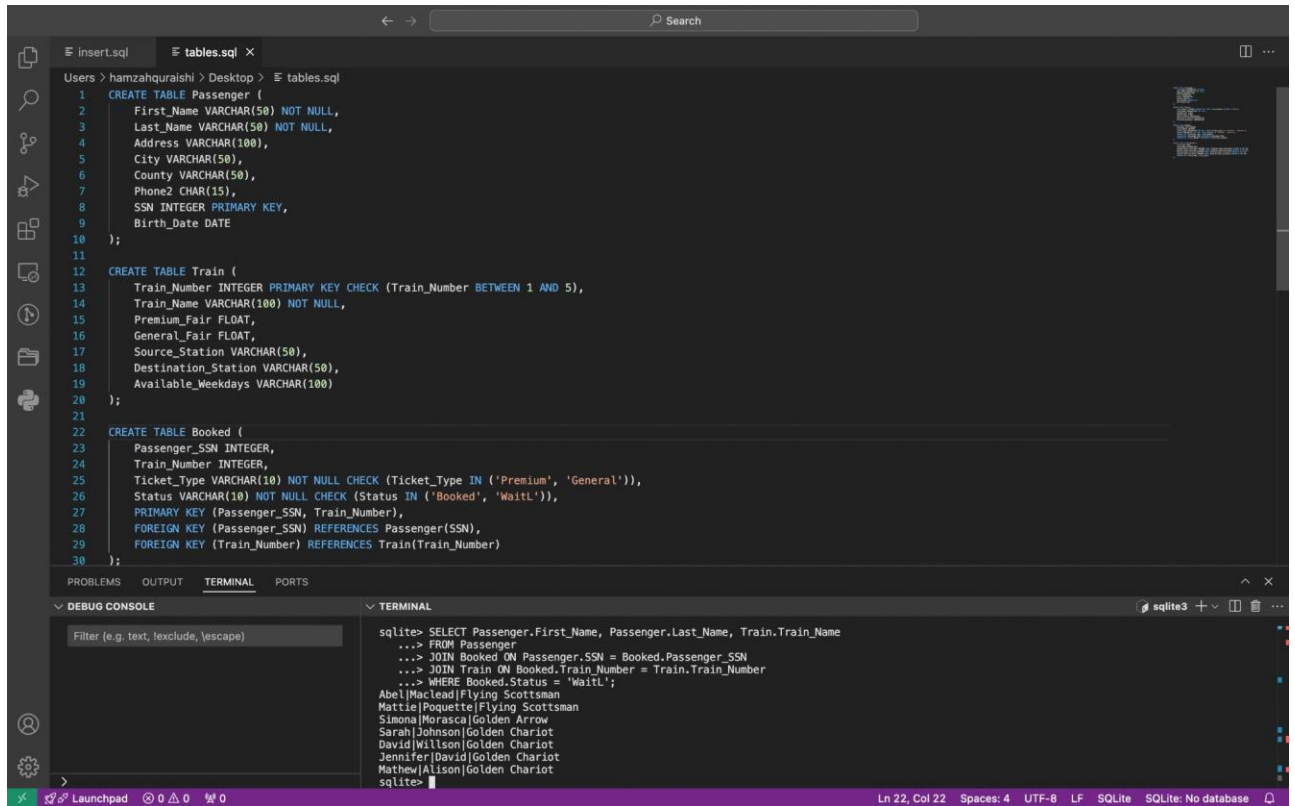
The terminal window at the bottom shows the following query and result:

```
sqlite> SELECT Passenger.First_Name, Passenger.Last_Name  
...> FROM Passenger  
...> JOIN Booked ON Passenger.SSN = Booked.Passenger_SSN  
...> JOIN Train ON Booked.Train_Number = Train.Train_Number  
...> WHERE Train.Train_Name = 'Flying Scotsman' AND Booked.Status = 'Booked';  
Kiley|Caldarera  
Fletcher|Flosi  
Josephine|Darakjy  
Sage|Wieser  
Kris|Marrier  
Graciela|Ruta  
sqlite>
```

The status bar at the bottom indicates: Ln 22, Col 22 Spaces: 4 UTF-8 LF SQLite SQLite: No database

6)

List passengers that are waitlisted including the name of the train.



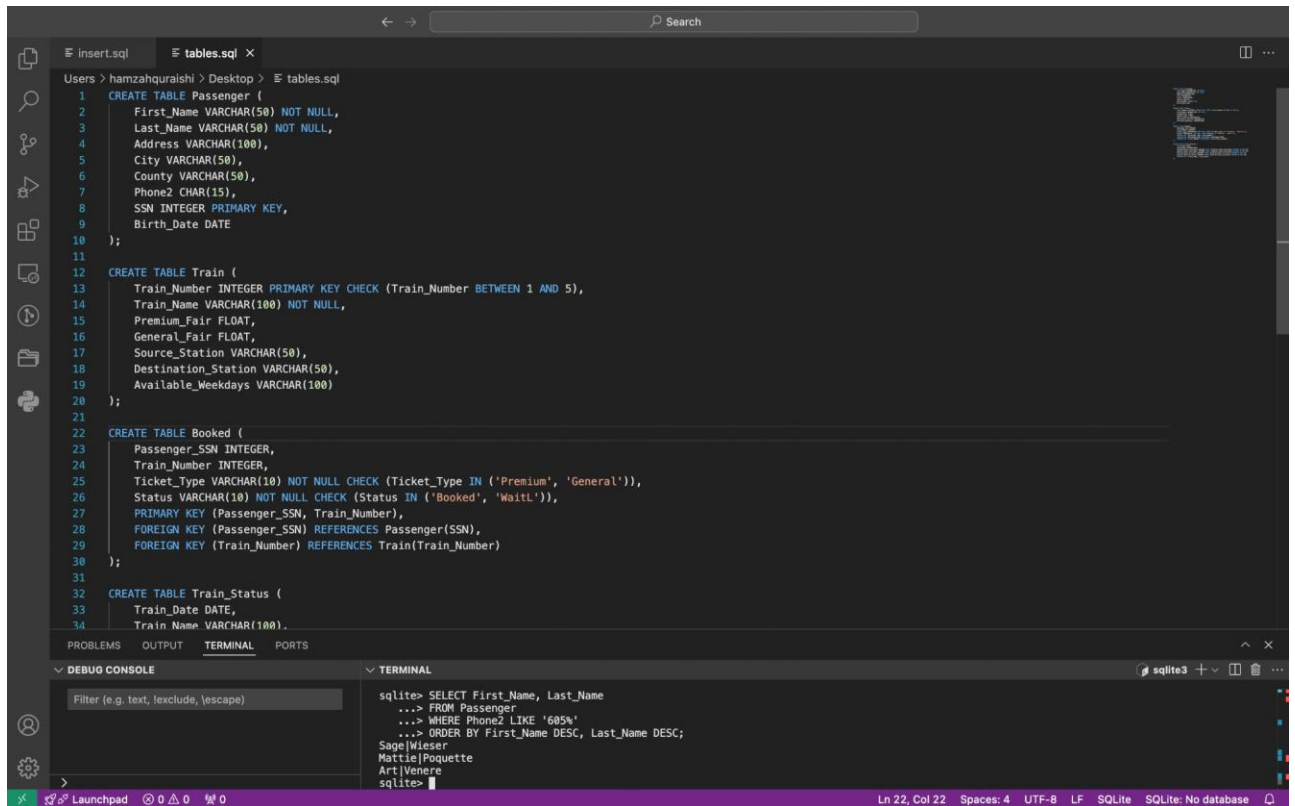
The screenshot shows a code editor with a file named 'tables.sql'. The code defines three tables: 'Passenger', 'Train', and 'Booked'. The 'Booked' table has a foreign key to 'Passenger' and a foreign key to 'Train'. Below the code, the 'TERMINAL' panel shows the execution of a query that selects the first and last names of passengers and the train name for those who are waitlisted. The query result is displayed in the terminal.

```
1 CREATE TABLE Passenger (  
2   First_Name VARCHAR(50) NOT NULL,  
3   Last_Name VARCHAR(50) NOT NULL,  
4   Address VARCHAR(100),  
5   City VARCHAR(50),  
6   County VARCHAR(50),  
7   Phone2 CHAR(15),  
8   SSN INTEGER PRIMARY KEY,  
9   Birth_Date DATE  
10 );  
11  
12 CREATE TABLE Train (  
13   Train_Number INTEGER PRIMARY KEY CHECK (Train_Number BETWEEN 1 AND 5),  
14   Train_Name VARCHAR(100) NOT NULL,  
15   Premium_Fair FLOAT,  
16   General_Fair FLOAT,  
17   Source_Station VARCHAR(50),  
18   Destination_Station VARCHAR(50),  
19   Available_Weekdays VARCHAR(100)  
20 );  
21  
22 CREATE TABLE Booked (  
23   Passenger_SSN INTEGER,  
24   Train_Number INTEGER,  
25   Ticket_Type VARCHAR(10) NOT NULL CHECK (Ticket_Type IN ('Premium', 'General')),  
26   Status VARCHAR(10) NOT NULL CHECK (Status IN ('Booked', 'Waitl')),  
27   PRIMARY KEY (Passenger_SSN, Train_Number),  
28   FOREIGN KEY (Passenger_SSN) REFERENCES Passenger(SSN),  
29   FOREIGN KEY (Train_Number) REFERENCES Train(Train_Number)  
30 );
```

```
sqlite> SELECT Passenger.First_Name, Passenger.Last_Name, Train.Train_Name  
...> FROM Passenger  
...> JOIN Booked ON Passenger.SSN = Booked.Passenger_SSN  
...> JOIN Train ON Booked.Train_Number = Train.Train_Number  
...> WHERE Booked.Status = 'Waitl';  
Abel|Maclead|Flying Scotsman  
Mattie|Poquette|Flying Scotsman  
Simona|Morasca|Golden Arrow  
Sarah|Johnson|Golden Chariot  
David|Wiltson|Golden Chariot  
Jennifer|David|Golden Chariot  
Mathew|Alison|Golden Chariot  
sqlite>
```


7)

List passenger names in descending order that have '605' phone area code. *assorted passengers by first name (first name is allowed specified) in descending order*



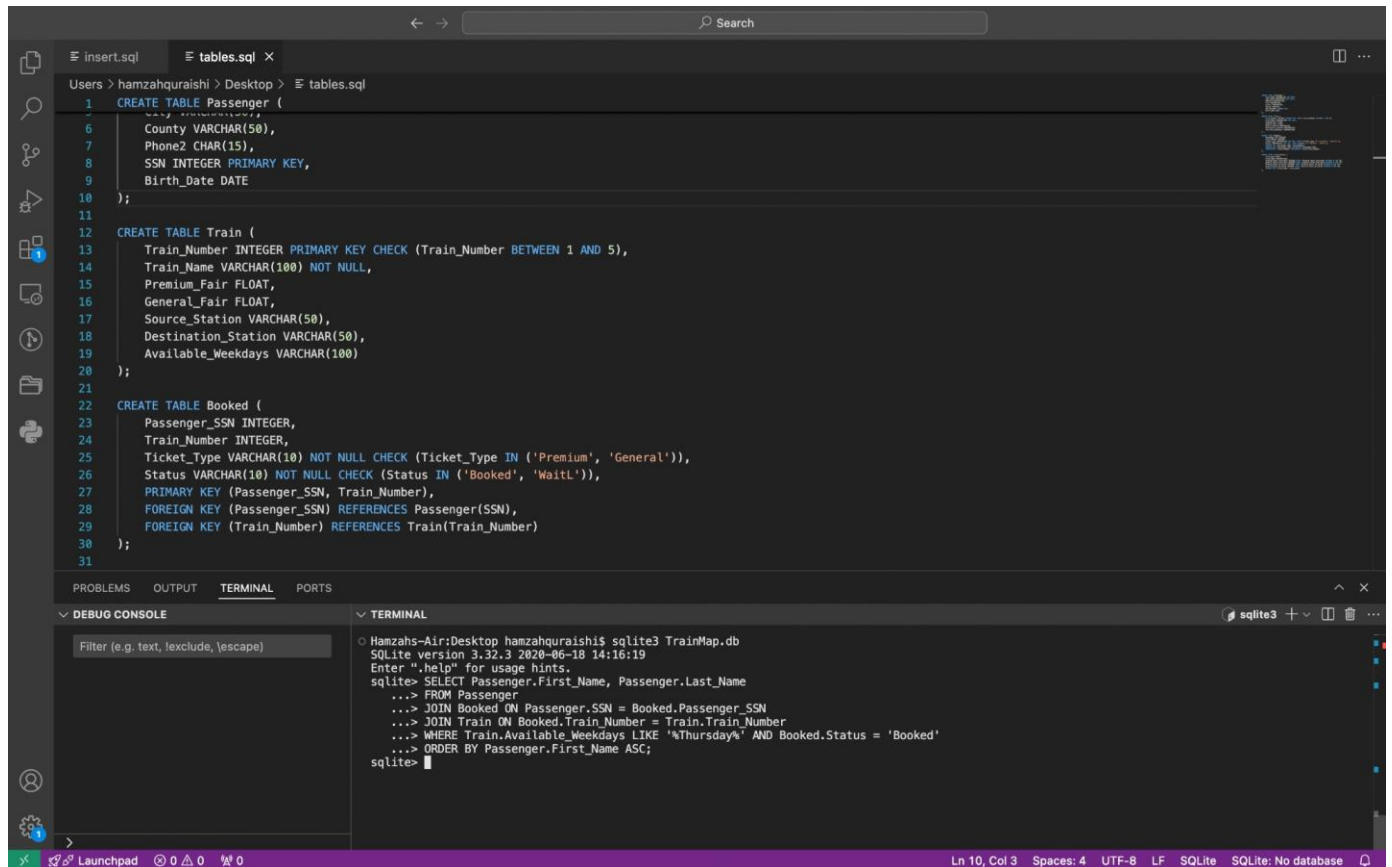
The screenshot shows a SQLite IDE interface with a dark theme. The main editor displays SQL schema for a database. The schema includes tables for Passengers, Trains, Bookings, and Train Statuses. The Passengers table has columns for First Name, Last Name, Address, City, County, Phone2, SSN (Primary Key), and Birth Date. The Trains table has columns for Train Number (Primary Key), Train Name, Premium Fair, General Fair, Source Station, Destination Station, and Available Weekdays. The Booked table has columns for Passenger SSN, Train Number, Ticket Type, Status, and foreign keys to the Passengers and Trains tables. The Train Status table has columns for Train Date and Train Name.

The terminal window at the bottom shows the execution of a query to select the first and last names of passengers with a phone area code of 605, ordered by first name in descending order. The results show three passengers: Sage Wieser, Mattie Poquette, and Art Venere.

```
1 CREATE TABLE Passenger (  
2     First_Name VARCHAR(50) NOT NULL,  
3     Last_Name VARCHAR(50) NOT NULL,  
4     Address VARCHAR(100),  
5     City VARCHAR(50),  
6     County VARCHAR(50),  
7     Phone2 CHAR(15),  
8     SSN INTEGER PRIMARY KEY,  
9     Birth_Date DATE  
10 );  
11  
12 CREATE TABLE Train (  
13     Train_Number INTEGER PRIMARY KEY CHECK (Train_Number BETWEEN 1 AND 5),  
14     Train_Name VARCHAR(100) NOT NULL,  
15     Premium_Fair FLOAT,  
16     General_Fair FLOAT,  
17     Source_Station VARCHAR(50),  
18     Destination_Station VARCHAR(50),  
19     Available_Weekdays VARCHAR(100)  
20 );  
21  
22 CREATE TABLE Booked (  
23     Passenger_SSN INTEGER,  
24     Train_Number INTEGER,  
25     Ticket_Type VARCHAR(10) NOT NULL CHECK (Ticket_Type IN ('Premium', 'General')),  
26     Status VARCHAR(10) NOT NULL CHECK (Status IN ('Booked', 'Waitl')),  
27     PRIMARY KEY (Passenger_SSN, Train_Number),  
28     FOREIGN KEY (Passenger_SSN) REFERENCES Passenger(SSN),  
29     FOREIGN KEY (Train_Number) REFERENCES Train(Train_Number)  
30 );  
31  
32 CREATE TABLE Train_Status (  
33     Train_Date DATE,  
34     Train_Name VARCHAR(100).  
35 );
```

```
sqlite> SELECT First_Name, Last_Name  
...> FROM Passenger  
...> WHERE Phone2 LIKE '605%'  
...> ORDER BY First_Name DESC, Last_Name DESC;  
Sage|Wieser  
Mattie|Poquette  
Art|Venere  
sqlite>
```

8) List name of passengers that are traveling on Thursdays in ascending order. *assorted passengers by first name in ascending order. Used Thursday as input (specified by question) and Booked as input - since those are the only passengers that confirmed traveling, not waitlisted. Results are empty which is correct due to in table Booked, none of the passengers have a Train Number of 1 (Train Number 1 is only the train that has Thursday as its available weekday/s)). Assorting passengers by first name, and adding Booked as input is correct/allowed)*



The screenshot shows a code editor with a file named `tables.sql` containing SQL code to create three tables: `Passenger`, `Train`, and `Booked`. The `Booked` table has foreign keys to `Passenger` and `Train`. Below the code editor is a terminal window with the following output:

```
Hamzahs-Air:Desktop hamzahquraishi$ sqlite3 TrainMap.db
SQLite version 3.32.3 2020-06-18 14:16:19
Enter ".help" for usage hints.
sqlite> SELECT Passenger.First_Name, Passenger.Last_Name
...> FROM Passenger
...> JOIN Booked ON Passenger.SSN = Booked.Passenger_SSN
...> JOIN Train ON Booked.Train_Number = Train.Train_Number
...> WHERE Train.Available_Weekdays LIKE '%Thursday%' AND Booked.Status = 'Booked'
...> ORDER BY Passenger.First_Name ASC;
sqlite>
```

The status bar at the bottom indicates the file is `Ln 10, Col 3`, with `Spaces: 4`, `UTF-8` encoding, `LF` line endings, and the database is `SQLite: No database`.