CRUD Databases
(Create, Read, Update and Delete)
Part 6 - Traveler's Database using PostgreSQL

Introduction

A demonstration of **CRUD** database programming, based on using **PostgreSQL**. CRUD stands for **create, read, update** and **delete**. These are the four basic operations every database must perform.

PostgreSQL is a relational database. It uses **SQL** (**Structured Query Language**).

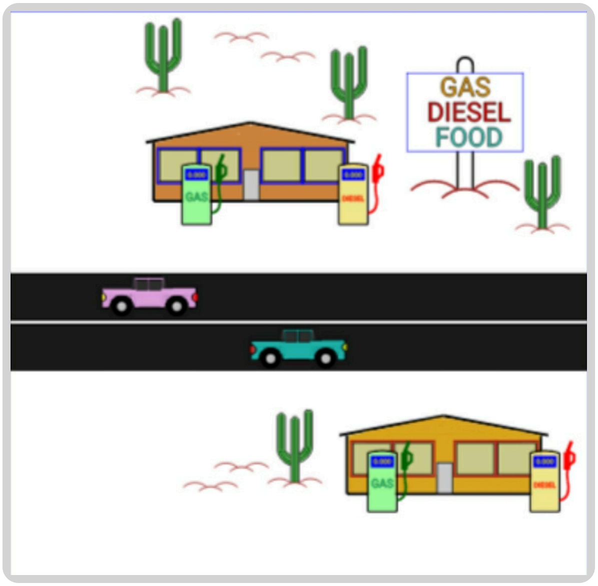

Figure 1: Travel on a long road trip.

Scenario - You are driving through the desert on a long road trip. Gas stations are few and far between, so it's important to know where you can get fuel so you don't run out. You also need to know where you can find food and a refreshing beverage. Calculate the distance to these things relative to your location. The data is **dynamically displayed** by querying the database using SQL.

When a query occurs, the app searches the database and returns the database entries which are the closest based on the location indicated in the query. Web SQL allows us to embed the database in either a Web or mobile device app.

The database will store the following information:

- **name**
- **longtitude**
- **lattitude**
- **GAS** or **NO GAS**
- **DIESEL** or **NO DIESEL**
- **FOOD** or **NO FOOD**

At the beginning of the demo, the closest items in the database will be displayed.

Database Access

Create The database is ***read-only***, so we aren't going to create any records after we initialize the database.

Read We read the records any time there is a GPS update.

Update The database is ***read-only***, so we aren't going to update any records after we initialize the database.

Delete The database is ***read-only***, so we aren't going to delete any records after we initialize the database.

Various SQL Databases Compared

SQLite	MySQL	SQL Server	PostgreSQL
no server	server	server	server
embedded database	client / server	client / server	client / server
Fast.	Fast.	Fast.	Fast.
Library: 250 - 500 KB single file. cross-platform.	Server: around 600 MB.	Server: 512 MB minimum.	Server: more than 200 MB.
Dynamic typing Supports: Blob, Integer, Null, Text, Real.	Static typing Supports: Tinyint, Smallint, Mediumint, Int, Bigint, Double, Float, Real, Decimal, Double precision, Numeric, Timestamp, Date, Datetime, Char, Varchar, Year, Tinytext, Tinyblob, Blob, Text, MediumBlob, MediumText, Enum, Set, Longblob, Longtext	Static typing Supports: bigint, numeric, bit, smallint, decimal, smallmoney, int, tinyint, money, float, real, date, datetimeoffset, datetime2, smalldatetime, datetime, time, char, varchar, text, nchar, nvarchar, ntext, binary, varbinary, image, cursor, rowversion, hiearchyid, uniqueidentifier, sql_variant, xml, Spatial Geometry Types, Spatial Geography Types, table	Static typing Supports: bigint, bigserial, double precision, integer, real, smallint, smallserial, serial, character, varchar, text, date, interval, time, time without time zone, time with time zone, timestamp, timestamp without time zone, timestamp with time zone, box, circle, line, lseg, path, point, polygon, cidr, inet, macaddr, bit, bit varying, tsquery, tsvector, json, jsonb, boolean, bytea, money, pg_lsn, txid_snapshot, uuid, xml
Supports "if exists" in drop statements.	Supports "if exists" in drop statements.	Supports "if exists" in drop statements. **(SQL Server 2016 and higher)**	Supports "if exists" in drop statements.
Does not support TOP.	**Does not support TOP.**	Supports TOP.	**Does not support TOP.**
Supports LIMIT.	Supports LIMIT.	Supports LIMIT.	Supports LIMIT.
Does not support right or full outer joins.	**Does not support full outer joins.** Supports right join.	Supports right or full outer joins.	Supports right or full outer joins.
Supports cross joins, inner joins, and left outer joins.	Supports cross joins, inner joins, and left outer joins.	Supports cross joins, inner joins, and left outer joins.	Supports cross joins, inner joins, and left outer joins.
No configurations.	Requires configuration.	Requires configuration.	Requires configuration.
Does not support simultaneous multiple users.	Supports simultaneous multiple users.	Supports simultaneous multiple users.	Supports simultaneous multiple users.
Authentication not included, may be added.	Includes authentication (username, password, and SSH).	Includes authentication.	Includes authentication, many security features.

Figure 2: Comparison of SQLite with other SQL databases.
Derived from [2] [55] [56] [57] [58] [61] [62] [63] [64] [65] [66] [69] [70] [71] [72] [73] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90].

Database Using Foreign Keys

This demo illustrates the indexing of **foreign keys**

Demo Platform

We will use **SQL playground (https://sqlplayground.app/)** to enter the SQL commands. SQL playground allows us to create a free online SQL sandbox for learning SQL and prototyping databases. SQL playground supports both MySQL 8.0 and PostgreSQL 16.0.

Verify the version of PostgreSQL that we are running:

```
> SELECT version();
version
PostgreSQL 16.3 (Debian 16.3-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0,
64-bit
>
```

```
> SHOW server_version;
server_version 16.3 (Debian 16.3-1.pgdg120+1)
>
```

Create the Travelers Database

Let's make sure we start the demo with a fresh database.

```
> DROP DATABASE Travelers;
>
```

Show the existing databases.

```
> SELECT datname FROM pg_database;
datname
postgres
template1
template0
>
```

```
> CREATE DATABASE Travelers;
Running SQL error:
CREATE DATABASE cannot run inside a transaction block
>
```

SQL playground doesn't allow this. For testing purposes, we create the tables without enclosing them within their own database.

Verify that nothing has changed:

```
> SELECT datname FROM pg_database;
datname
postgres
template1
template0
>
```

Create the Rest Stop Names table

```
> CREATE TABLE STOP_NAME (
      stopName_id INTEGER NOT NULL,
      name VARCHAR(28),
      PRIMARY KEY(stopName_id)
);
>
```

Verify that the table was created correctly

```
> SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE table_name='stop_name';
table_catalog    table_schema    table_name    table_type    self_referencing_column_name
reference_generation    user_defined_type_catalog    user_defined_type_schema
user_defined_type_name    is_insertable_into    is_typed    commit_action

postgres    eac75bd05ee911f0aea5a1dd2cff911c    stop_name    BASE TABLE    <NULL>    <NULL>
<NULL>    <NULL>    <NULL>    YES    NO    <NULL>
>
```

```
> SELECT table_catalog,
      table_schema,
      table_name,
      column_name,
      ordinal_position,
      data_type,
      character_maximum_length
FROM information_schema.columns
WHERE table_name='stop_name';
table_catalog    table_schema    table_name    column_name    ordinal_position    data_type
character_maximum_length

postgres   a9044ed05eec11f0aea5a1dd2cff911c   stop_name   stopname_id   1   integer     <NULL>
postgres   a9044ed05eec11f0aea5a1dd2cff911c   stop_name   name          2   character   varying 28
>
```

Insert the Rest Stop Names into the table

```
> INSERT INTO STOP_NAME (stopName_id, name) VALUES (1, 'Petrol King 1');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (2, 'Oil City 2');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (3, 'Burger Stop 3');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (4, 'Fill ''Er Up 4');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (5, 'Taco Town 5');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (6, 'Oil City 6');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (7, 'Petrol King 7');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (8, 'Fill ''Er Up 8');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (9, 'Burger Stop 9');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (10, 'Petrol King 10');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (11, 'Oil City 11');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (12, 'Burger Stop 12');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (13, 'Fill ''Er Up 13');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (14, 'Taco Town 14');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (15, 'Oil City 15');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (16, 'Petrol King 16');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (17, 'Burger Stop 17');
INSERT INTO STOP_NAME (stopName_id, name) VALUES (18, 'Fill ''Er Up 18');
>
```

Verify that the table is correct

```
> SELECT * FROM stop_name;
stopname_id         name
    1              Petrol King 1
    2              Oil City 2
    3              Burger Stop 3
    4              Fill 'Er Up 4
    5              Taco Town 5
    6              Oil City 6
    7              Petrol King 7
    8              Fill 'Er Up 8
    9              Burger Stop 9
    10             Petrol King 10
    11             Oil City 11
    12             Burger Stop 12
    13             Fill 'Er Up 13
    14             Taco Town 14
    15             Oil City 15
    16             Petrol King 16
    17             Burger Stop 17
    18             Fill 'Er Up 18
>
```

Create the GAS AVAILABILITY table

```
> CREATE TABLE gas (
    gas_id INTEGER NOT NULL,
    gstatus VARCHAR(6),
    PRIMARY KEY(gas_id)
);
>
```

Verify that the table was created correctly

```
> SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE table_name='gas';
table_catalog    table_schema    table_name    table_type    self_referencing_column_name
reference_generation    user_defined_type_catalog    user_defined_type_schema
user_defined_type_name    is_insertable_into    is_typed    commit_action

postgres    594b0e005ee811f0aea5a1dd2cff911c    gas    BASE TABLE    <NULL>    <NULL>    <NULL>
    <NULL>    <NULL>    YES    NO    <NULL>
>
```

```
> SELECT table_catalog,
    table_schema,
    table_name,
    column_name,
    ordinal_position,
    data_type,
    character_maximum_length
FROM information_schema.columns
WHERE table_name='gas';
table_catalog    table_schema    table_name    column_name    ordinal_position    data_type
character_maximum_length

postgres    11d415905eec11f0aea5a1dd2cff911c    gas    gas_id     1    integer      <NULL>
postgres    11d415905eec11f0aea5a1dd2cff911c    gas    gstatus    2    character    varying 6
>
```

Insert the entries into the GAS table

```
> INSERT INTO gas (gas_id, gstatus) VALUES (1, 'GAS');
INSERT INTO gas (gas_id, gstatus) VALUES (2, 'NO GAS');
>
```

Verify that the table is correct

```
> SELECT * FROM gas;
gas_id    gstatus
1          GAS
2          NO GAS
>
```

Create the DIESEL AVAILABILITY table

```
> CREATE TABLE diesel ( diesel_id INTEGER NOT NULL,
      dstatus VARCHAR(9),
      PRIMARY KEY(diesel_id)
);
>
```

Verify that the table was created correctly

```
> SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE table_name='diesel';
table_catalog    table_schema    table_name    table_type    self_referencing_column_name
reference_generation    user_defined_type_catalog    user_defined_type_schema
user_defined_type_name    is_insertable_into    is_typed    commit_action

postgres    d5e343b05ee811f0aea5a1dd2cff911c    diesel    BASE TABLE    <NULL>    <NULL>
<NULL>    <NULL>    <NULL>    YES    NO    <NULL>
>
```

```
> SELECT table_catalog,
      table_schema,
      table_name,
      column_name,
      ordinal_position,
      data_type,
      character_maximum_length
FROM information_schema.columns
WHERE table_name='diesel';
table_catalog    table_schema    table_name    column_name    ordinal_position    data_type
character_maximum_length

postgres    54b869b05eec11f0aea5a1dd2cff911c    diesel    diesel_id    1    integer    <NULL>
postgres    54b869b05eec11f0aea5a1dd2cff911c    diesel    dstatus      2    character    varying 9
>
```

Insert the entries into the DIESEL table

```
> INSERT INTO diesel (diesel_id, dstatus) VALUES (1, 'DIESEL');
INSERT INTO diesel (diesel_id, dstatus) VALUES (2, 'NO DIESEL');
>
```

Verify that the table is correct

```
> SELECT * FROM diesel;
diesel_id    dstatus
1            DIESEL
2            NO DIESEL
>
```

Create the FOOD AVAILABILITY table

```
> CREATE TABLE food ( food_id INTEGER NOT NULL,
      fstatus VARCHAR(7),
      PRIMARY KEY(food_id)
);
>
```

Verify that the table was created correctly

```
> SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE table_name='food';
postgres      7e62f2105ee911f0aea5a1dd2cff911c      food      BASE TABLE      <NULL>      <NULL>      <NULL>
      <NULL>      <NULL>      YES      NO      <NULL>
>


mysql> SELECT table_catalog,
      table_schema,
      table_name,
      column_name,
      ordinal_position,
      data_type,
      character_maximum_length
FROM information_schema.columns
WHERE table_name='food';
table_catalog      table_schema      table_name      column_name      ordinal_position      data_type
character_maximum_length

postgres      7980c3a05eec11f0aea5a1dd2cff911c      food      food_id      1      integer         <NULL>
postgres      7980c3a05eec11f0aea5a1dd2cff911c      food      fstatus      2      character      varying 7
mysql>
```

Insert the entries into the FOOD table

```
> INSERT INTO food (food_id, fstatus) VALUES (1, 'FOOD');
INSERT INTO food (food_id, fstatus) VALUES (2, 'NO FOOD');
>
```

Verify that the table is correct

```
> SELECT * FROM food;
food_id      fstatus
      1          FOOD
      2          NO FOOD
>
```

Create the REST STOP table

```
> CREATE TABLE REST_STOP (
    restStop_id INTEGER NOT NULL,
    restStop_name INTEGER,
    longtitude INTEGER,
    lattitude INTEGER,
    gas_available INTEGER,
    diesel_available INTEGER,
    food_available INTEGER,
FOREIGN KEY (restStop_name) REFERENCES STOP_NAME (stopName_id)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (gas_available) REFERENCES GAS (gas_id)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (diesel_available) REFERENCES DIESEL (diesel_id)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (food_available) REFERENCES FOOD (food_id)
ON DELETE CASCADE ON UPDATE CASCADE,
PRIMARY KEY(restStop_id)
);
>
```

Verify that the table was created correctly

```
> SELECT * FROM INFORMATION_SCHEMA.TABLES
WHERE table_name='rest_stop';
table_catalog    table_schema    table_name    table_type    self_referencing_column_name
reference_generation    user_defined_type_catalog    user_defined_type_schema
user_defined_type_name    is_insertable_into    is_typed    commit_action

postgres    5400b9c05fad11f0aea5a1dd2cff911c    rest_stop    BASE TABLE    <NULL>    <NULL>
<NULL>    <NULL>    <NULL>    YES    NO    <NULL>
>
```

```
> SELECT table_catalog,
    table_schema,
    table_name,
    column_name,
    ordinal_position,
    data_type,
    character_maximum_length
FROM information_schema.columns
WHERE table_name='rest_stop';
table_catalog    table_schema    table_name    column_name    ordinal_position    data_type
character_maximum_length

postgres    98705d505ef311f0aea5a1dd2cff911c    rest_stop    reststop_id        1    integer    <NULL>
postgres    98705d505ef311f0aea5a1dd2cff911c    rest_stop    reststop_name      2    integer    <NULL>
postgres    98705d505ef311f0aea5a1dd2cff911c    rest_stop    longtitude         3    integer    <NULL>
postgres    98705d505ef311f0aea5a1dd2cff911c    rest_stop    lattitude          4    integer    <NULL>
postgres    98705d505ef311f0aea5a1dd2cff911c    rest_stop    gas_available      5    integer    <NULL>
postgres    98705d505ef311f0aea5a1dd2cff911c    rest_stop    diesel_available   6    integer    <NULL>
postgres    98705d505ef311f0aea5a1dd2cff911c    rest_stop    food_available     7    integer    <NULL>
>
```

Insert the entries into the REST STOP table

```
mysql> INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (1, 1, 450, 900, 1, 1, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (2, 2, 457, 900, 2, 2, 2);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (3, 3, 462, 900, 2, 2, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (4, 4, 462, 900, 1, 1, 2);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (5, 5, 467, 900, 2, 2, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (6, 6, 467, 900, 1, 1, 2);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (7, 7, 475, 900, 1, 1, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (8, 8, 482, 900, 1, 1, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (9, 9, 487, 900, 2, 2, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (10, 10, 487, 900, 1, 1, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (11, 11, 492, 900, 2, 2, 2);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (12, 12, 492, 900, 2, 2, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (13, 13, 492, 900, 1, 1, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (14, 14, 502, 900, 2, 2, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (15, 15, 502, 900, 1, 1, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (16, 16, 510, 900, 1, 1, 2);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (17, 17, 518, 900, 2, 2, 1);
INSERT INTO REST_STOP (restStop_id, restStop_name, longtitude, lattitude, gas_available,
diesel_available, food_available) VALUES (18, 18, 518, 900, 1, 1, 2);
>
```

Verify that the table is correct (no FOREIGN KEY values)

```
> SELECT * FROM rest_stop;
reststop_id   reststop_name   longtitude   lattitude   gas_available   diesel_available   food_available
     1              1            450          900            1                1                 1
     2              2            457          900            2                2                 2
     3              3            462          900            2                2                 1
     4              4            462          900            1                1                 2
     5              5            467          900            2                2                 1
     6              6            467          900            1                1                 2
     7              7            475          900            1                1                 1
     8              8            482          900            1                1                 1
     9              9            487          900            2                2                 1
    10             10            487          900            1                1                 1
    11             11            492          900            2                2                 2
    12             12            492          900            2                2                 1
    13             13            492          900            1                1                 1
    14             14            502          900            2                2                 1
    15             15            502          900            1                1                 1
    16             16            510          900            1                1                 2
    17             17            518          900            2                2                 1
    18             18            518          900            1                1                 2
>
```

Limit the entries by LONGTITUDE

```
> SELECT * FROM rest_stop WHERE longtitude = 467;
reststop_id   reststop_name   longtitude   lattitude   gas_available   diesel_available   food_available
     5             5             467          900            2                 2                  1
     6             6             467          900            1                 1                  2
>
```

Limit the entries to a RANGE of LONGTITUDES

```
> SELECT * FROM rest_stop WHERE longtitude BETWEEN 467 AND 492;
reststop_id   reststop_name   longtitude   lattitude   gas_available   diesel_available   food_available
     5             5             467          900            2                 2                  1
     6             6             467          900            1                 1                  2
     7             7             475          900            1                 1                  1
     8             8             482          900            1                 1                  1
     9             9             487          900            2                 2                  1
    10            10             487          900            1                 1                  1
    11            11             492          900            2                 2                  2
    12            12             492          900            2                 2                  1
    13            13             492          900            1                 1                  1
>
```

Verify that the table is correct (FOREIGN KEY values)

```
> SELECT distinct b.name, a.restStop_id, a.longtitude, a.lattitude, c.gstatus, d.dstatus, e.fstatus
FROM rest_stop a
JOIN stop_name b
ON(b.stopName_id=a.restStop_id)
JOIN GAS c
ON(a.gas_available=c.gas_id)
JOIN DIESEL d
ON(a.diesel_available=d.diesel_id)
JOIN FOOD e
ON(a.food_available=e.food_id)
ORDER BY longtitude ASC
LIMIT 18;
name              reststop_id     longtitude     lattitude     gstatus      dstatus       fstatus
Petrol King 1         1               450           900         GAS         DIESEL        FOOD
Oil City 2            2               457           900         NO GAS      NO DIESEL     NO FOOD
Burger Stop 3         3               462           900         NO GAS      NO DIESEL     FOOD
Fill 'Er Up 4         4               462           900         GAS         DIESEL        NO FOOD
Oil City 6            6               467           900         GAS         DIESEL        NO FOOD
Taco Town 5           5               467           900         NO GAS      NO DIESEL     FOOD
Petrol King 7         7               475           900         GAS         DIESEL        FOOD
Fill 'Er Up 8         8               482           900         GAS         DIESEL        FOOD
Burger Stop 9         9               487           900         NO GAS      NO DIESEL     FOOD
Petrol King 10        10              487           900         GAS         DIESEL        FOOD
Burger Stop 12        12              492           900         NO GAS      NO DIESEL     FOOD
Fill 'Er Up 13        13              492           900         GAS         DIESEL        FOOD
Oil City 11           11              492           900         NO GAS      NO DIESEL     NO FOOD
Oil City 15           15              502           900         GAS         DIESEL        FOOD
Taco Town 14          14              502           900         NO GAS      NO DIESEL     FOOD
Petrol King 16        16              510           900         GAS         DIESEL        NO FOOD
Burger Stop 17        17              518           900         NO GAS      NO DIESEL     FOOD
Fill 'Er Up 18        18              518           900         GAS         DIESEL        NO FOOD
>
```

Limit the entries to a RANGE of LONGTITUDES

```
> SELECT DISTINCT b.name, a.longtitude, a.lattitude, c.gstatus, d.dstatus, e.fstatus
FROM rest_stop a
JOIN stop_name b
ON(b.stopName_id=a.restStop_id)
JOIN GAS c
ON(a.gas_available=c.gas_id)
JOIN DIESEL d
ON(a.diesel_available=d.diesel_id)
JOIN FOOD e
ON(a.food_available=e.food_id)
WHERE longtitude BETWEEN 467 AND 492
ORDER BY longtitude ASC
LIMIT 18;
name             longtitude      lattitude      gstatus      dstatus        fstatus
Oil City 6          467              900        GAS          DIESEL         NO FOOD
Taco Town 5         467              900        NO GAS       NO DIESEL      FOOD
Petrol King 7       475              900        GAS          DIESEL         FOOD
Fill 'Er Up 8       482              900        GAS          DIESEL         FOOD
Burger Stop 9       487              900        NO GAS       NO DIESEL      FOOD
Petrol King 10      487              900        GAS          DIESEL         FOOD
Burger Stop 12      492              900        NO GAS       NO DIESEL      FOOD
Fill 'Er Up 13      492              900        GAS          DIESEL         FOOD
Oil City 11         492              900        NO GAS       NO DIESEL      NO FOOD
>
```

Now that the SQL queries return the correct range of entries based on the current longtitude, the app can display the closest rest stops within our search window. The results are displayed whenever the GPS location info is updated.

End of Demo

This page intentionally left blank.

References

[1] "SQLite Home Page." *SQLite.org*, 27-Dec-2022. [Online]. Available: https://www.sqlite.org/index.html. [Accessed: 03-Jan-2023].

[2] "SQLite Frequently Asked Questions." *SQLite.org*, 27-Jul-2021. [Online]. Available: https://www.sqlite.org/faq.html. [Accessed: 03-Jan-2023].

[3] "Most Widely Deployed SQL Database Engine." *SQLite.org*, 03-Jun-2021. [Online]. Available: https://www.sqlite.org/mostdeployed.html. [Accessed: 03-Jan-2023].

[4] "Long Term Support." *SQLite.org*, 01-Dec-2020. [Online]. Available: https://www.sqlite.org/lts.html. [Accessed: 03-Jan-2023].

[5] "Home | Library of Congress," *Library of Congress*. [Online]. Available: https://www.loc.gov/. [Accessed: 03-Jan-2023].

[6] "Alphabetical List Of SQLite Documents." *SQLite.org*. [Online]. Available: https://www.sqlite.org/doclist.html. [Accessed: 03-Jan-2023].

[7] "SQLite Copyright." *SQLite.org*, 10-Nov-2021. [Online]. Available: https://www.sqlite.org/copyright.html. [Accessed: 03-Jan-2023].

[8] "Well-Known Users Of SQLite." *SQLite.org*, 18-Jun-2022. [Online]. Available: https://www.sqlite.org/famous.html. [Accessed: 03-Jan-2023].

[9] "SQLite In 5 Minutes Or Less." *SQLite.org*, 04-Apr-2016. [Online]. Available: https://www.sqlite.org/quickstart.html. [Accessed: 03-Jan-2023].

[10] "SQLite Download Page." *SQLite.org*. [Online]. Available: https://www.sqlite.org/download.html. [Accessed: 03-Jan-2023].

[11] "Command Line Shell For SQLite." *SQLite.org*, 27-Dec-2022. [Online]. Available: https://www.sqlite.org/cli.html. [Accessed: 03-Jan-2023].

[12] "The Schema Table." *SQLite.org*, 15-Feb-2022. [Online]. Available: https://www.sqlite.org/schematab.html. [Accessed: 03-Jan-2023].

[13] "SQLite Show Tables | Basic Syntax and the Different Examples," *EDUCBA*, 05-May-2021. [Online]. Available: https://www.educba.com/sqlite-show-tables/. [Accessed: 03-Jan-2023].

[14] "4 Ways to Get Information about a Table's Structure in SQLite." *database.guide*, 30-May-2020. [Online]. Available: https://database.guide/4-ways-to-get-information-about-a-tables-structure-in-sqlite/. [Accessed: 03-Jan-2023].

[15] R. Peterson, "SQLite Database Tutorial for Beginners: Learn with Examples," *Guru99*, 24-Dec-2022. [Online]. Available: https://www.guru99.com/sqlite-tutorial.html. [Accessed: 03-Jan-2023].

[16] J. Zhao, "Html5 Web SQLite Database Example," *dev2qa.com*, 04-Apr-2022. [Online]. Available: https://www.dev2qa.com/html5-web-sqlite-database-example/. [Accessed: 03-Jan-2023].

[17] R. Sharp, "Introducing Web SQL Databases | HTML5 Doctor." *HTML5 Doctor*, 24-Feb-2010. [Online]. Available: http://html5doctor.com/introducing-web-sql-databases/. [Accessed: 03-Jan-2023].

[18] S. Bose, "Working SqLite Javascript," *Gist*. [Online]. Available: https://gist.github.com/siddharthabose03/8450242. [Accessed: 03-Jan-2023].

[19] "Use Query Parameters | Reporting | DevExpress Documentation." *DevExpress*, 28-Aug-2022. [Online]. Available: https://docs.devexpress.com/XtraReports/17387/detailed-guide-to-devexpress-reporting/bind-reports-to-data/sql-database/specify-query-parameters. [Accessed: 03-Jan-2023].

References (continued)

[20] "Web SQL API Tutorial - Zebra Technologies TechDocs." *Zebra TechDocs*, 2020. [Online]. Available: https://techdocs.zebra.com/enterprise-browser/3-0/tutorial/websql/. [Accessed: 03-Jan-2023].

[21] T. Steiner, "Deprecating and removing Web SQL," *Chrome Developers*, 13-Dec-2022. [Online]. Available: https://developer.chrome.com/blog/deprecating-web-sql/. [Accessed: 04-Jan-2023].

[22] K. Puls, "Pass Parameters to SQL Queries," *Excelguru*, 28-Apr-2016. [Online]. Available: https://excelguru.ca/pass-parameters-to-sql-queries/. [Accessed: 04-Jan-2023].

[23] A. Ravikiran, "SQL Insert: The Best Way to Populate Database Tables [Updated]," *Simplilearn*, 27-Oct-2022. [Online]. Available: https://www.simplilearn.com/tutorials/sql-tutorial/sql-insert. [Accessed: 04-Jan-2023].

[24] "What is WEB SQL?," *GeeksforGeeks*, 11-Oct-2022. [Online]. Available: https://www.geeksforgeeks.org/what-is-web-sql/. [Accessed: 04-Jan-2023].

[25] A. Choudhary, "Handling multiple records in Web Sql by recursive method," *Oodles Technologies*, 22-Jun-2015. [Online]. Available: https://www.oodlestechnologies.com/blogs/handling-multiple-records-in-web-sql-by-recursive-method/. [Accessed: 04-Jan-2023].

[26] A. Choudhary, "Use of Web SQL in Phonegap," *Oodles Technologies*, 01-Sep-2014. [Online]. Available: https://www.oodlestechnologies.com/blogs/Use%20of%20Web%20SQL%20in%20Phonegap/. [Accessed: 04-Jan-2023].

[27] S. Dixit, "Dev.Opera — Taking Your Web Apps Offline: A Tale of Web Storage, Application Cache and WebSQL." *Opera Software AS*, 22-Mar-2011. [Online]. Available: https://dev.opera.com/articles/offline-web-apps/. [Accessed: 04-Jan-2023].

[28] M. Mansuriya, "WEBSQL, SQL at the client's end," *Medium*, 27-Jul-2020. [Online]. Available: https://madhavmansuriya40.medium.com/websql-sql-at-the-clients-end-e70579a0401f. [Accessed: 04-Jan-2023].

[29] "SQLite DROP TABLE Statement with Examples," *SQLite Tutorial*. [Online]. Available: https://www.sqlitetutorial.net/sqlite-drop-table/. [Accessed: 04-Jan-2023].

[30] "Connecting To SQLite Database Using Node.js," *SQLite Tutorial*. [Online]. Available: https://www.sqlitetutorial.net/sqlite-nodejs/connect/. [Accessed: 04-Jan-2023].

[31] "HTML5 - Web SQL Database." *Tutorials Point*. [Online]. Available: https://www.tutorialspoint.com/html5/html5_web_sql.htm. [Accessed: 04-Jan-2023].

[32] "Web SQL by james-priest." *github.io*. [Online]. Available: https://james-priest.github.io/100-days-of-code-log-r2/CH16-Offline1-WebSQL.html. [Accessed: 04-Jan-2023].

[33] "Web SQL Database | Tizen Docs." *Tizen Project*, 2023. [Online]. Available: https://docs.tizen.org/application/web/guides/w3c/storage/websql/. [Accessed: 04-Jan-2023].

[34] S. Somani, "HTML 5 Web SQL Database." *C# Corner*, 2023. [Online]. Available: https://www.c-sharpcorner.com/uploadfile/75a48f/html-5-web-sql-database/. [Accessed: 04-Jan-2023].

[35] "HTML5 - Web SQL Database." *Tutorials Point*. [Online]. Available: https://www.tutorialspoint.com/html5/html5_web_sql.htm. [Accessed: 04-Jan-2023].

[36] S. Jaiswal, "What is Web SQL - javatpoint," *Javatpoint*. [Online]. Available: https://www.javatpoint.com/what-is-web-sql. [Accessed: 04-Jan-2023].

[37] "SQLite Tutorial - An Easy Way to Master SQLite Fast," *SQLite Tutorial*. [Online]. Available: https://www.sqlitetutorial.net/. [Accessed: 04-Jan-2023].

[38] "How To Download & Install SQLite Tools," *SQLite Tutorial*. [Online]. Available: https://www.sqlitetutorial.net/download-install-sqlite/. [Accessed: 04-Jan-2023].

References (continued)

[39] "SQLite Sample Database And Its Diagram (in PDF format)," *SQLite Tutorial*. [Online]. Available: https://www.sqlitetutorial.net/sqlite-sample-database/. [Accessed: 04-Jan-2023].

[40] "SQLite Cheat Sheet," *SQLite Tutorial*. [Online]. Available: https://www.sqlitetutorial.net/sqlite-cheat-sheet/. [Accessed: 04-Jan-2023].

[41] "SQLite Show Tables: Listing All Tables in a Database," *SQLite Tutorial*, 2022. [Online]. Available: https://www.sqlitetutorial.net/sqlite-show-tables/. [Accessed: 05-Jan-2023].

[42] "Appropriate Uses For SQLite." *SQLite.org*, 14-Dec-2022. [Online]. Available: https://www.sqlite.org/whentouse.html. [Accessed: Dec. 05-Jan-2023].

[43] 262588213843476, "Reset WebSQL database dropping every tables," *Gist*. [Online]. Available: https://gist.github.com/partageit/d091039f1942baed910a3f54f491056c. [Accessed: Dec. 05-Jan-2023].

[44] N. Lawson, "Web SQL Database: In Memoriam," *Read the Tea Leaves*, Apr. 26, 2014. [Online]. Available: https://nolanlawson.com/2014/04/26/web-sql-database-in-memoriam/. [Accessed: 05-Jan-2023].

[45] user2250152, "How to read metadata from Sqlite database," *Database Administrators Stack Exchange*, 05-Apr-2017. [Online]. Available: https://dba.stackexchange.com/q/169246. [Accessed: 05-Jan-2023].

[46] "Blind WebSQL and Storage extraction for HTML5 Apps." *Blueinfy's blog*, 2012. [Online]. Available: http://blog.blueinfy.com/2012/01/blind-websql-and-storage-extraction-for.html. [Accessed: 05-Jan-2023].

[47] "HTML5 - Web SQL Database." *Tutorials Point*, 2022. [Online]. Available: https://www.tutorialspoint.com/html5/html5_web_sql.htm. [Accessed: 05-Jan-2023].

[48] "Hosting SQLite databases on Github Pages - (or any static file hoster) - phiresky's blog." *phiresky's blog*, 03-May-2021. [Online]. Available: https://phiresky.github.io/blog/2021/hosting-sqlite-databases-on-github-pages/. [Accessed: 05-Jan-2023].

[49] phpMyAdmin contributors, "phpMyAdmin," *phpMyAdmin*, 2023. [Online]. Available: https://www.phpmyadmin.net/. [Accessed: 05-Jan-2023].

[50] "SQL Introduction." *W3Schools*, 2023. [Online]. Available: https://www.w3schools.com/sql/sql_intro.asp. [Accessed: 05-Jan-2023].

[51] P. Loshin and J. Sirkin, "What is Structured Query Language (SQL)?," *TechTarget*, 2023. [Online]. Available: https://www.techtarget.com/searchdatamanagement/definition/SQL. [Accessed: 05-Jan-2023].

[52] C. Brooks, "What Is SQL, and How Is It Used? - businessnewsdaily.com," *Business News Daily*, 16-Nov-2022. [Online]. Available: https://www.businessnewsdaily.com/5804-what-is-sql.html. [Accessed: 05-Jan-2023].

[53] A. Kozubek-Krycuń, "The History of SQL – How It All Began," *LearnSQL.com*, 17-Nov-2020. [Online]. Available: https://learnsql.com/blog/history-of-sql/. [Accessed: 05-Jan-2023].

[54] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377â€“387, Jun. 1970, doi: 10.1145/362384.362685. [Online]. Available: https://doi.org/10.1145/362384.362685. [Accessed: 05-Jan-2023].

[55] E. S, "SQLite vs MySQL – What's the Difference," *Hostinger Tutorials*, 27-Dec-2022. [Online]. Available: https://www.hostinger.com/tutorials/sqlite-vs-mysql-whats-the-difference/. [Accessed: 05-Jan-2023].

[56] T. Wiseman, "Comparing some differences of SQL Server to SQLite." *MSSQLTips.com*, 23-Mar-2017. [Online]. Available: https://www.mssqltips.com/sqlservertip/4777/comparing-some-differences-of-sql-server-to-sqlite/. [Accessed: 05-Jan-2023].

References (continued)

[57] N. Samuel, "SQLite vs PostgreSQL: 8 Critical Differences - Learn | Hevo," *Hevo*, 18-May-2021. [Online]. Available: https://hevodata.com/learn/sqlite-vs-postgresql/. [Accessed: 05-Jan-2023].

[58] "SQLite: Documentation." *SQLite.org*, 02-Jan-2023. [Online]. Available: https://www.sqlite.org/src/doc/trunk/ext/userauth/user-auth.txt. [Accessed: 05-Jan-2023].

[59] "Defense Against The Dark Arts." *SQLite.org*, 07-Nov-2022. [Online]. Available: https://www.sqlite.org/security.html. [Accessed: 05-Jan-2023].

[60] "SQLite Database Speed Comparison." *SQLite.org*, 01-Apr-2014. [Online]. Available: https://www.sqlite.org/speed.html. [Accessed: 05-Jan-2023].

[61] D. Team, "MySQL vs PostgreSQL vs SQLite: A comparison of 3 popular RDBMS," *Devathon*, 15-Jan-2021. [Online]. Available: https://devathon.com/blog/mysql-vs-postgresql-vs-sqlite/. [Accessed::06-Jan-2022].

[62] H. Zahid, "MySQL vs SQLite – Compared." *Linux Hint*. [Online]. Available: https://linuxhint.com/mysql-vs-sqlite/. [Accessed: 06-Jan-2023].

[63] ostezer and M. Drake, "SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean." *DigitalOcean*, 10-Mar-2022. [Online]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Accessed: 06-Jan-2023].

[64] M. Ray *et al.*, "Data types (Transact-SQL) - SQL Server." *Microsoft Learn*, 19-Nov-2022. [Online]. Available: https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql. [Accessed: 06-Jan-2023].

[65] V. To *et al.*, "TOP (Transact-SQL) - SQL Server." *Microsoft Learn*, 31-Dec-2022. [Online]. Available: https://learn.microsoft.com/en-us/sql/t-sql/queries/top-transact-sql. [Accessed: 06-Jan-2023].

[66] "SQL SELECT TOP, LIMIT, FETCH FIRST ROWS ONLY, ROWNUM." *W3Schools*, 2023. [Online]. Available: https://www.w3schools.com/sql/sql_top.asp. [Accessed: 06-Jan-2023].

[67] D. Jalli, "How To Use The SQL NOT EXISTS and EXISTS Operator?," *Janbasktraining*, 17-Apr-2022. [Online]. Available: https://www.janbasktraining.com/blog/sql-exists-operator. [Accessed: 06-Jan-2023].

[68] H. Zahid, "How to create table in MySQL using 'if not exists' technique." *Linux Hint*. [Online]. Available: https://linuxhint.com/create-table-if-not-exist-mysql/. [Accessed: 06-Jan-2023].

[69] "Drop Tables in PostgreSQL Database." *TutorialsTeacher*, 2023. [Online]. Available: https://www.tutorialsteacher.com/postgresql/drop-tables. [Accessed: 06-Jan-2023].

[70] V. Kaplarevic, "MySQL DROP TABLE: With Examples & Options," *phoenixNAP*, 30-Jun-2020. [Online]. Available: https://phoenixnap.com/kb/mysql-drop-table. [Accessed: 06-Jan-2023].

[71] J. Gavin, "SQL Server DROP TABLE IF EXISTS Examples." *MSSQLTips*, 23-Mar-2021. [Online]. Available: https://www.mssqltips.com/sqlservertip/6769/sql-server-drop-table-if-exists/. [Accessed: 06-Jan-2023].

[72] M. Ray, J. Roth, R. Konidena and R. West, "SQL Server 2019: Hardware & software requirements - SQL Server." *Microsoft Learn*, 15-Dec-2022. [Online]. Available: https://learn.microsoft.com/en-us/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server-2019. [Accessed: 06-Jan-2023].

[73] "Difference Between SQL Vs MySQL Vs SQL Server (with Examples)" *SoftwareTestingHelp*, 05-Dec-2022. [Online]. Available: https://www.softwaretestinghelp.com/sql-vs-mysql-vs-sql-server/. [Accessed: 06-Jan-2023].

[74] I. Hickson and Google, Inc., "Web SQL Database W3C Working Group Note" *W3C*, 18-Nov-2010. [Online]. Available: http://www.w3.org/TR/webdatabase/. [Accessed: 06-Jan-2023].

References (continued)

[75] "7.6. LIMIT and OFFSET," *The PostgreSQL Global Development Group*, 10-Nov-2022. [Online]. Available: https://www.postgresql.org/docs/15/queries-limit.html. [Accessed: 06-Jan-2023].

[76] "SQL Server TOP and FETCH and PostgreSQL LIMIT and OFFSET - SQL Server to Aurora PostgreSQL Migration Playbook." *Amazon Web Services, Inc.*, 2023. [Online]. Available: https://docs.aws.amazon.com/dms/latest/sql-server-to-aurora-postgresql-migration-playbook/chap-sql-server-aurora-pg.tsql.topfetch.html. [Accessed: 06-Jan-2023].

[77] "SQL Features That SQLite Does Not Implement." *SQLite.org*, 13-Apr-2022. [Online]. Available: https://www.sqlite.org/omitted.html. [Accessed: 06-Jan-2023].

[78] "SQL: SELECT LIMIT Statement." *TechOnTheNet.com*, 2023. [Online]. Available: https://www.techonthenet.com/sql/select_limit.php. [Accessed: 06-Jan-2023].

[79] RajuKumar19, "PostgreSQL - LIMIT with OFFSET clause," *GeeksforGeeks*, 28-Aug-2020. [Online]. Available: https://www.geeksforgeeks.org/postgresql-limit-with-offset-clause/. [Accessed: 06-Jan-2023].

[80] "How To Do a Full Outer Join in MySQL," *Ubiq*, 03-Feb-2021. [Online]. Available: https://ubiq.co/database-blog/how-to-do-a-full-outer-join-in-mysql/. [Accessed: 06-Jan-2023].

[81] "MySQL RIGHT JOIN Explained By Practical Examples," *MySQLTutorial.org*, 2022. [Online]. Available: https://www.mysqltutorial.org/mysql-right-join/. [Accessed: 06-Jan-2023].

[82] "SQL Server Full Outer Join Explained By Practical Examples," *sqlservertutorial.net*, 2022. [Online]. Available: https://www.sqlservertutorial.net/sql-server-basics/sql-server-full-outer-join/. [Accessed: 06-Jan-2023].

[83] "Forge SQL Complete - Powerful T-SQL Formatting Tool," *Devart*, 2023. [Online]. Available: https://www.devart.com/dbforge/sql/sqlcomplete/. [Accessed: 06-Jan-2023].

[84] "PostgreSQL - JOINS." *Tutorials Point*. [Online]. Available: https://www.tutorialspoint.com/postgresql/postgresql_using_joins.htm. [Accessed: 06-Jan-2023].

[85] R. Peterson, "MySQL JOINS Tutorial: INNER, OUTER, LEFT, RIGHT, CROSS," *Guru99*, 02-Nov-2022. [Online]. Available: https://www.guru99.com/joins.html. [Accessed: 06-Jan-2023].

[86] "PostgreSQL Joins: A Visual Explanation of PostgreSQL Joins." *PostgreSQL Tutorial*, 2022. [Online]. Available: https://www.postgresqltutorial.com/postgresql-joins/. [Accessed: 06-Jan-2023].

[87] R. West *et al.*, "Server Configuration Options (SQL Server) - SQL Server." *Microsoft Learn*, 27-Dec-2022. [Online]. Available: https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/server-configuration-options-sql-server. [Accessed: 06-Jan-2023].

[88] "Chapter 20. Server Configuration," *The PostgreSQL Global Development Group*, 2023. [Online]. Available: https://www.postgresql.org/docs/15/runtime-config.html. [Accessed: 06-Jan-2023].

[89] I. N. SA, "MySQL: Maximum number of simultaneous connections - Infomaniak." *Infomaniak*, 2022. [Online]. Available: https://www.infomaniak.com/en/support/faq/471/mysql-maximum-number-of-simultaneous-connections. [Accessed: 06-Jan-2023].

[90] R. West *et al.*, "Configure the user connections Server Configuration Option - SQL Server." *Microsoft Learn*, 19-Nov-2022. [Online]. Available: https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/configure-the-user-connections-server-configuration-option. [Accessed: 06-Jan-2023].

[91] "SQLite Foreign Key Support." *SQLite.org*, 20-Jan-2022. [Online]. Available: https://sqlite.org/foreignkeys.html. [Accessed: 06-Jan-2023].

[92] "Pragma statements supported by SQLite." *SQLite.org*, 25-Dec-2022. [Online]. Available: https://www.sqlite.org/pragma.html. [Accessed: 06-Jan-2023].

References (continued)

[93] "SELECT." *SQLite.org*, 26-Oct-2022. [Online]. Available: https://www.sqlite.org/lang_select.html. [Accessed: 06-Jan-2023].

[94] "SQLite WHERE - Filter Rows in a Result Set," *SQLite Tutorial*, 2022. [Online]. Available: https://www.sqlitetutorial.net/sqlite-where/. [Accessed: 06-Jan-2023].

[95] "SQLite - WHERE Clause." *Tutorials Point*, 2022. [Online]. Available: https://www.tutorialspoint.com/sqlite/sqlite_where_clause.htm. [Accessed: 06-Jan-2023].

[96] "SQLite SELECT DISTINCT - Removing Duplicate in Result Set," *SQLite Tutorial*, 2022. [Online]. Available: https://www.sqlitetutorial.net/sqlite-distinct/. [Accessed: 06-Jan-2023].

[97] "SQLite - DISTINCT Keyword." *Tutorials Point*, 2022. [Online]. Available: https://www.tutorialspoint.com/sqlite/sqlite_distinct_keyword.htm. [Accessed: 06-Jan-2023].

[98] "Features Of SQLite." *SQLite.org*, 21-Feb-2022. [Online]. Available: https://www.sqlite.org/features.html. [Accessed: 08-Jan-2023].

[99] A. Deveria, L. Schoors and individual contributors, ˜SQLite' | Can I use... Support tables for HTML5, CSS3, etc." *Can I use*. [Online]. Available: https://caniuse.com/?search=SQLite. [Accessed: 08-Jan-2023].

[100] "Persistent Storage Options." *SQLite.org*. [Online]. Available: https://sqlite.org/wasm/doc/trunk/persistence.md. [Accessed: 08-Jan-2023].

[101] "sql.js." *sql.js.org*. [Online]. Available: https://sql.js.org/. [Accessed: 08-Jan-2023].

[102] "Home." *sql.js.org*. [Online]. Available: https://sql.js.org/documentation/. [Accessed: 08-Jan-2023].

[103] "Quirks, Caveats, and Gotchas In SQLite." *SQLite.org*, 16-Nov-2022. [Online]. Available: https://www.sqlite.org/quirks.html#aggregate_queries_can_contain_non_aggregate_result_columns_that_are_not_in_the_group_by_clause. [Accessed: 10-Jan-2023].

[104] "MySQL." *Oracle Corporation*, 2021. [Online]. Available: https://www.mysql.com/. [Accessed: 13-Jul-2021].

[105] "MySQL :: Download MySQL Community Server." *Oracle Corporation*, 2021. [Online]. Available: https://dev.mysql.com/downloads/mysql/. [Accessed: 13-Jul-2021].

[106] "PHP: Hypertext Preprocessor." *the PHP Group*, 2021. [Online]. Available: https://www.php.net/. [Accessed: 13-Jul-2021].

[107] "PHP: Downloads." *the PHP Group*, 2021. [Online]. Available: https://www.php.net/downloads.php. [Accessed: 13-Jul-2021].

[108] "Explore Windows 10 OS, Computers, Apps, & More | Microsoft," *Microsoft*, 2021. [Online]. Available: https://www.microsoft.com/en-us/windows. [Accessed: 13-Jul-2021].

[109] "Welcome! - The Apache HTTP Server Project." *The Apache Software Foundation*, 2021. [Online]. Available: https://httpd.apache.org/. [Accessed: 13-Jul-2021].

[110] "Welcome to The Apache Software Foundation!" *The Apache Software Foundation*, 2021. [Online]. Available: https://www.apache.org/. [Accessed: 13-Jul-2021].

[111] "Difference between LAMP, MAMP and WAMP stack," *GeeksforGeeks*, 18-Mar-2021. [Online]. Available: https://www.geeksforgeeks.org/what-is-the-difference-between-lamp-stack-mamp-stack-and-wamp-stack/. [Accessed: 14-Jul-2021].

[112] "Wamp.NET." *Wamp.NET*, 2021. [Online]. Available: https://www.wamp.net/. [Accessed: 09-Jul-2021].

References (continued)

[113] S. Yegulalp, "Review: WAMP stacks for Web developers," *Computerworld*, 30-May-2012. [Online]. Available: https://www.computerworld.com/article/2727273/review--wamp-stacks-for-web-developers.html. [Accessed: 09-Jul-2021].

[114] P. G. D. Group, “PostgreSQL,” *PostgreSQL*, 09-Jul-2025. [Online]. Available: https://www.postgresql.org/. [Accessed: 09-Jul-2025].

[115] P. G. D. Group, “PostgreSQL: License.” *PostgreSQL*. [Online]. Available: https://www.postgresql.org/about/licence/. [Accessed: 09-Jul-2025].

[116] “PHP: PostgreSQL - Manual.” [Online]. Available: https://www.php.net/manual/en/book.pgsql.php. [Accessed: 09-Jul-2025].

[117] “phpPgAdmin,” *SourceForge*, 09-Nov-2020. [Online]. Available: https://sourceforge.net/projects/phppgadmin/. [Accessed: 09-Jul-2025].

[118] “SQL Playground - Online SQL Sandbox.” *SQL Playground*. [Online]. Available: https://sqlplayground.app/. [Accessed: 09-Jul-2025].

[119] “PostgreSQL Tutorial,” *GeeksforGeeks*, 23-Feb-2021. [Online]. Available: https://www.geeksforgeeks.org/postgresql/postgresql-tutorial/. [Accessed: 09-Jul-2025].

[120] A. Makauskas, “What is PostgreSQL? Key features, use cases, and benefits explained,” *Hostinger Tutorials*, 04-Jul-2025. [Online]. Available: https://www.hostinger.com/tutorials/what-is-postgresql. [Accessed: 09-Jul-2025].

[121] S. Ravoof, “PostgreSQL vs SQL Server: 16 Critical Differences,” *Kinsta®*, 30-May-2022. [Online]. Available: https://kinsta.com/blog/postgresql-vs-sql-server/. [Accessed: 09-Jul-2025].

[122] “Azure Database for PostgreSQL | Microsoft Azure.” *Microsoft Azure*. [Online]. Available: https://azure.microsoft.com/en-us/products/postgresql. [Accessed: 09-Jul-2025].

[123] “Cloud SQL for PostgreSQL documentation,” *Google Cloud*. [Online]. Available: https://cloud.google.com/sql/docs/postgres. [Accessed: 09-Jul-2025].

End of document.