

Experiment 2: Horizontal Fragmentation

- Name: Srikanth Iyengar
- UID: 2020400062
- Branch: IT
- Batch: G
- Subject: Advanced Database Management System

Aim:

Design a distributed database by applying the concept of horizontal fragmentation

Implementation

Flight Table

Flight Id	Flight Name	Source	Destination	Tickets Available
1	Transatlantic Express	New York	London	100
2	City Hopper	London	Paris	90
3	European Voyager	Paris	Berlin	80
4	Continent Cruiser	Berlin	Rome	70
5	Mediterranean Flyer	Rome	Madrid	60
6	Iberian Explorer	Madrid	Barcelona	50
7	Nordic Adventurer	Barcelona	Amsterdam	40
8	Oceanic Odyssey	Amsterdam	New York	30
9	Atlantic Ascension	New York	London	20
10	Skyline Shuttle	London	Paris	10
11	Continental Commuter	Paris	Berlin	20
12	Bella Italia	Berlin	Rome	30
13	España Express	Rome	Madrid	40
14	Dutch Delight	Madrid	Barcelona	50
15	Viking Venture	Barcelona	Amsterdam	60
16	Big Apple Bounce	Amsterdam	New York	70
17	British Airways	New York	London	80
18	French Connection	London	Paris	90
19	German Getaway	Paris	Berlin	100
20	Italian Holiday	Berlin	Rome	110

App user Table

Username	Password	Tickets Booked	Location
johndoe123	5f4dcc3b5aa765d61d8327deb882cf99	2	New York
janedoe456	5e884898da28047151d0e56f8dc62927	1	London
bobsmith789	21232f297a57a5a743894a0e4a801fc3	4	Paris
alicebrown101	ee11cbb19052e40b07aac0ca060c23ee	0	Berlin
tomgreen202	25f9e794323b453885f5181f1b624d0b	3	Madrid
jennypink303	a665a45920422f9d417e4867efdc4fb8	5	Rome
chrisblack404	5d41402abc4b2a76b9719d911017c592	6	Moscow
kimwhite505	bdbafc1b5f5dc8ab2f0196b5cd543a33	7	Toronto
paulred606	1a1dc91c907325c69271ddf0c944bc72	8	Sydney
lucyyellow707	ac0eafeed417f0ffa7befa0d1c01a7a6	9	Seoul
mikeblue808	098f6bcd4621d373cade4e832627b4f6	10	Tokyo
sarahpurple909	0b7c75b0f4b7e585a1c73e72409f3b7c	11	Shanghai
davidorange010	76a2173be6393254e72ffa4d6df1030a	12	Beijing
emmablack112	c1dfd96eea8cc2b62785275bca38ac26	13	Mexico City
danielgreen213	98f13708210194c475687be6106a3bce	14	Rio de Janeiro
elizabethpurple314	f0d2f1e2dc9abceb877557a483e07c3a	15	Sao Paulo
richardblue415	5ab5dbc27120680fa6f0b6ed7b6c0098	16	Buenos Aires
jenniferred516	7c222fb2927d828af22f592134e89324	17	Lima
robertyellow617	7a74f16c3e7d20a03b356f7bcf8c8f7a	18	Bogota

Query for creation of table

```
drop table if exists app_user;

CREATE TABLE IF NOT EXISTS public.app_user
(
    username character varying(100) COLLATE pg_catalog."default" NOT NULL,
    user_password character varying(100) COLLATE pg_catalog."default",
    tickets_booked integer,
    address varchar(100),
    CONSTRAINT app_user_pkey PRIMARY KEY (username)
);

INSERT INTO public.app_user (username, user_password, tickets_booked, address)
```

```

VALUES
('johndoe123', '5f4dcc3b5aa765d61d8327deb882cf99', 2, 'New York'),
('janedoe456', '5e884898da28047151d0e56f8dc62927', 1, 'London'),
('bobsmith789', '21232f297a57a5a743894a0e4a801fc3', 4, 'Paris'),
('alicebrown101', 'ee11cbb19052e40b07aac0ca060c23ee', 0, 'Berlin'),
('tomgreen202', '25f9e794323b453885f5181f1b624d0b', 3, 'Madrid'),
('jennypink303', 'a665a45920422f9d417e4867efdc4fb8', 5, 'Rome'),
('chrisblack404', '5d41402abc4b2a76b9719d911017c592', 6, 'Moscow'),
('kimwhite505', 'bdbafc1b5f5dc8ab2f0196b5cd543a33', 7, 'Toronto'),
('paulred606', '1a1dc91c907325c69271ddf0c944bc72', 8, 'Sydney'),
('lucyyellow707', 'ac0eafeed417f0ffa7befa0d1c01a7a6', 9, 'Seoul'),
('mikeblue808', '098f6bcd4621d373cade4e832627b4f6', 10, 'Tokyo'),
('sarahpurple909', '0b7c75b0f4b7e585a1c73e72409f3b7c', 11, 'Shanghai'),
('davidorange010', '76a2173be6393254e72ffa4d6df1030a', 12, 'Beijing'),
('emmablack112', 'c1dfd96eea8cc2b62785275bca38ac26', 13, 'Mexico City'),
('danielgreen213', '98f13708210194c475687be6106a3bce', 14, 'Rio de Janeiro'),
('elizabethpurple314', 'f0d2f1e2dc9abceb877557a483e07c3a', 15, 'Sao Paulo'),
('richardblue415', '5ab5dbc27120680fa6f0b6ed7b6c0098', 16, 'Buenos Aires'),
('jenniferred516', '7c222fb2927d828af22f592134e89324', 17, 'Lima'),
('robertyellow617', '7a74f16c3e7d20a03b356f7bcf8c8f7a', 18, 'Bogota');

drop table if exists flight;

CREATE TABLE IF NOT EXISTS public.flight
(
    flight_id integer NOT NULL,
    flight_name character varying(100) COLLATE pg_catalog."default" NOT NULL,
    flight_source character varying(100) COLLATE pg_catalog."default" NOT NULL,
    flight_destination character varying(100) COLLATE pg_catalog."default" NOT NULL,
    tickets_available integer NOT NULL,
    CONSTRAINT flight_pkey PRIMARY KEY (flight_id)
);
-----
---

insert into flight values
(1, 'Transatlantic Express', 'New York', 'London', 100),
(2, 'City Hopper', 'London', 'Paris', 90),
(3, 'European Voyager', 'Paris', 'Berlin', 80),
(4, 'Continent Cruiser', 'Berlin', 'Rome', 70),
(5, 'Mediterranean Flyer', 'Rome', 'Madrid', 60),
(6, 'Iberian Explorer', 'Madrid', 'Barcelona', 50),
(7, 'Nordic Adventurer', 'Barcelona', 'Amsterdam', 40),
(8, 'Oceanic Odyssey', 'Amsterdam', 'New York', 30),
(9, 'Atlantic Ascension', 'New York', 'London', 20),
(10, 'Skyline Shuttle', 'London', 'Paris', 10),
(11, 'Continental Commuter', 'Paris', 'Berlin', 20),
(12, 'Bella Italia', 'Berlin', 'Rome', 30),
(13, 'España Express', 'Rome', 'Madrid', 40),
(14, 'Dutch Delight', 'Madrid', 'Barcelona', 50),
(15, 'Viking Venture', 'Barcelona', 'Amsterdam', 60),

```

```

(16, 'Big Apple Bounce', 'Amsterdam', 'New York', 70),
(17, 'British Airways', 'New York', 'London', 80),
(18, 'French Connection', 'London', 'Paris', 90),
(19, 'German Getaway', 'Paris', 'Berlin', 100),
(20, 'Italian Holiday', 'Berlin', 'Rome', 110);
;

-----
--



drop table if exists flight1, flight2, app_user1, app_user2;

CREATE TABLE flight1 (LIKE flight);
CREATE TABLE flight2 (LIKE flight);

CREATE TABLE app_user1 (LIKE app_user);
CREATE TABLE app_user2 (LIKE app_user);

insert into flight1 (SELECT * FROM flight where flight_id % 2 = 0);
insert into flight2 (SELECT * FROM flight WHERE flight_id % 2 = 1);

insert into app_user1 (SELECT * FROM app_user WHERE tickets_booked > 10);
insert into app_user2 (SELECT * FROM app_user WHERE tickets_booked <= 10);

```

PART A

Query 1

```

-- Find all the flight which starts from London
SELECT * FROM flight WHERE flight_source = 'London';
SELECT * FROM flight1 WHERE flight_source = 'London';
SELECT * FROM flight2 WHERE flight_source = 'London';

```

Result From table 1

The screenshot shows the PgAdmin interface with the following details:

- Left Panel (Browser):** Shows the database structure with servers, databases, and tables. The current database is `exp1`.
- Top Bar:** Includes File, Object, Tools, Help, Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and a tab for `answers.sql`.
- Query Editor:** Contains the following SQL code:

```
-- Find all the flight which starts from London
SELECT * FROM flight WHERE flight_source = 'London';
SELECT * FROM flight1 WHERE flight_source = 'London';
SELECT * FROM flight2 WHERE flight_source = 'London';
```
- Data Output:** Displays a table with the following data:

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	2	City Hopper	London	Paris
2	10	Skyline Shuttle	London	Paris
3	18	French Connection	London	Paris
- Message Bar:** Shows "Successfully run. Total query runtime: 6 secs 47 msec. 3 rows affected." and "Ln 3, Col 1".

Result From table 2

The screenshot shows the PgAdmin interface with the following details:

- Left Panel (Browser):** Shows the database structure with servers, databases, and tables. The current database is `exp1`.
- Top Bar:** Includes File, Object, Tools, Help, Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and a tab for `answers.sql`.
- Query Editor:** Contains the following SQL code:

```
-- Find all the flight which starts from London
SELECT * FROM flight WHERE flight_source = 'London';
SELECT * FROM flight1 WHERE flight_source = 'London';
SELECT * FROM flight2 WHERE flight_source = 'London';
```
- Data Output:** Displays a table with the following data:

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	2	City Hopper	London	Paris
2	10	Skyline Shuttle	London	Paris
3	18	French Connection	London	Paris
- Message Bar:** Shows "Successfully run. Total query runtime: 1 secs 811 msec. 0 rows affected." and "Ln 4, Col 1".

Result From original Table

```

-- Result from original table
SELECT * FROM flight WHERE flight_source = 'London';
SELECT * FROM flight1 WHERE flight_source = 'London';
SELECT * FROM flight2 WHERE flight_source = 'London';

```

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	City Hopper	London	Paris	90
2	Skyline Shuttle	London	Paris	10
3	French Connection	London	Paris	90

Total rows: 3 of 3 Query complete 00:00:02.255

Successfully run. Total query runtime: 2 secs 255 msec. 3 rows affected.

Query 2

```

-- Find all the flight where balance tickets > 40
SELECT * FROM flight WHERE tickets_available > 40;
SELECT * FROM flight1 WHERE tickets_available > 40;
SELECT * FROM flight2 WHERE tickets_available > 40;

```

Result From table 1

```

-- Result from table 1
SELECT * FROM flight WHERE tickets_available > 40;
SELECT * FROM flight1 WHERE tickets_available > 40;
SELECT * FROM flight2 WHERE tickets_available > 40;

```

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	City Hopper	London	Paris	90
2	Continent Cruiser	Berlin	Rome	70
3	Iberian Explorer	Madrid	Barcelona	50
4	Dutch Delight	Madrid	Barcelona	50
5	Big Apple Bounce	Amsterdam	New York	70
6	French Connection	London	Paris	90
7	Italian Holiday	Berlin	Rome	110

Total rows: 7 of 7 Query complete 00:00:04.079

Ln 7, Col 2

Result From table 2

The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database structure on the left, including Servers, Databases (Postgres, exp1, exp2), and various system catalogs.
- Query Editor:** The current tab is "answers.sql". The query is:


```
-- Find all the flight where balance tickets > 40
SELECT * FROM flight WHERE tickets_available > 40;
SELECT * FROM flight1 WHERE tickets_available > 40;
SELECT * FROM flight2 WHERE tickets_available > 40;
```
- Data Output:** A table showing the results of the query. The columns are: flight_id, flight_name, flight_source, flight_destination, and tickets_available. The data is:

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	Transatlantic Express	New York	London	100
2	European Voyager	Paris	Berlin	80
3	Mediterranean Flyer	Rome	Madrid	60
4	Viking Venture	Barcelona	Amsterdam	60
5	British Airways	New York	London	80
6	German Getaway	Paris	Berlin	100
- Messages:** Total rows: 6 of 6. Query complete 00:00:01.938. Ln 8, Col 1.

Result From original table

This screenshot is nearly identical to the previous one, showing the same query execution and results. The main difference is the message at the bottom:

Total rows: 13 of 13 Query complete 00:00:02.908 Ln 6, Col 1.

Query 3

```
-- Find all the flight where id % 2 = 0
SELECT * FROM flight WHERE flight_id % 2 = 0;
SELECT * FROM flight1 WHERE flight_id % 2 = 0;
SELECT * FROM flight2 WHERE flight_id % 2 = 0;
```

Result From table 1

The screenshot shows the PgAdmin 4 interface with the following details:

- Servers:** Postgres (7) - Another Supabase, CockroachDB Postgres, NEON DB.
- Databases:** Postgres supabase, exp1, exp2.
- Current Database:** exp2.
- Query Editor:** Contains the following SQL code:

```
-- Find all the flight where id % 2 = 0
SELECT * FROM flight WHERE flight_id % 2 = 0;
SELECT * FROM flight1 WHERE flight_id % 2 = 0;
SELECT * FROM flight2 WHERE flight_id % 2 = 0;
```
- Data Output:** A table showing flight data for rows 1 through 10. The columns are: flight_id, flight_name, flight_source, flight_destination, and tickets_available.

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	City Hopper	London	Paris	90
2	Continent Cruiser	Berlin	Rome	70
3	Iberian Explorer	Madrid	Barcelona	50
4	Oceanic Odyssey	Amsterdam	New York	30
5	Skyline Shuttle	London	Paris	10
6	Bella Italia	Berlin	Rome	30
7	Dutch Delight	Madrid	Barcelona	50
8	Big Apple Bounce	Amsterdam	New York	70
9	French Connection	London	Paris	90
10	Italian Holiday	Berlin	Rome	110

- Message Bar:** Total rows: 10 of 10 | Query complete 00:00:06.975 | Ln 11, Col 1

Result From table 2

The screenshot shows the PgAdmin 4 interface with the following details:

- Servers:** Postgres (7) - Another Supabase, CockroachDB Postgres, NEON DB.
- Databases:** Postgres supabase, exp1, exp2.
- Current Database:** exp2.
- Query Editor:** Contains the following SQL code:

```
-- Find all the flight where id % 2 = 0
SELECT * FROM flight WHERE flight_id % 2 = 0;
SELECT * FROM flight1 WHERE flight_id % 2 = 0;
SELECT * FROM flight2 WHERE flight_id % 2 = 0;
```
- Data Output:** A table showing flight data for rows 1 through 10. The columns are: flight_id, flight_name, flight_source, flight_destination, and tickets_available.

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	City Hopper	London	Paris	90
2	Continent Cruiser	Berlin	Rome	70
3	Iberian Explorer	Madrid	Barcelona	50
4	Oceanic Odyssey	Amsterdam	New York	30
5	Skyline Shuttle	London	Paris	10
6	Bella Italia	Berlin	Rome	30
7	Dutch Delight	Madrid	Barcelona	50
8	Big Apple Bounce	Amsterdam	New York	70
9	French Connection	London	Paris	90
10	Italian Holiday	Berlin	Rome	110

- Message Bar:** Total rows: 0 of 0 | Query complete 00:00:01.807 | Ln 12, Col 1

Result From original table

The screenshot shows the PgAdmin interface with a dark theme. The left sidebar lists several databases, including 'exp1', 'exp2', and 'bit.io'. The main window displays a SQL query and its results. The query is:

```
9 -- Find all the flight where id % 2 = 0
10 SELECT * FROM flight WHERE flight_id % 2 = 0;
11 SELECT * FROM flight1 WHERE flight_id % 2 = 0;
12 SELECT * FROM flight2 WHERE flight_id % 2 = 0;
```

The results table has columns: flight_id [PK] integer, flight_name character varying (100), flight_source character varying (100), flight_destination character varying (100), and tickets_available integer. The data is as follows:

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	2 City Hopper	London	Paris	90
2	4 Continent Cruiser	Berlin	Rome	70
3	6 Iberian Explorer	Madrid	Barcelona	50
4	8 Oceanic Odyssey	Amsterdam	New York	30
5	10 Skyline Shuttle	London	Paris	10
6	12 Bella Italia	Berlin	Rome	30
7	14 Dutch Delight	Madrid	Barcelona	50
8	16 Big Apple Bounce	Amsterdam	New York	70
9	18 French Connection	London	Paris	90
10	20 Italian Holiday	Berlin	Rome	110

At the bottom, a message says "Successfully run. Total query runtime: 2 secs 437 msec. 10 rows affected." and "Ln 10, Col 1".

Query 4

```
-- Find all the flights where flight name contains express
SELECT * FROM flight WHERE strpos(flight_name, 'Delight') > 0;
SELECT * FROM flight1 WHERE strpos(flight_name, 'Delight') > 0;
SELECT * FROM flight2 WHERE strpos(flight_name, 'Delight') > 0;
```

Result From table 1

The screenshot shows the PgAdmin interface with a dark theme. The left sidebar lists several databases, including 'exp1', 'exp2', and 'bit.io'. The main window displays a SQL query and its results. The query is:

```
14 SELECT * FROM flight WHERE strpos(flight_name, 'Delight') > 0;
15 SELECT * FROM flight1 WHERE strpos(flight_name, 'Delight') > 0;
16 SELECT * FROM flight2 WHERE strpos(flight_name, 'Delight') > 0;
17 -- Find all the user who booked atleast 9 tickets
```

The results table has columns: flight_id integer, flight_name character varying (100), flight_source character varying (100), flight_destination character varying (100), and tickets_available integer. The data is as follows:

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	14 Dutch Delight	Madrid	Barcelona	50

At the bottom, a message says "Total rows: 1 of 1 Query complete 00:00:06.001" and "Ln 15, Col 1".

Result From table 2

The screenshot shows the PgAdmin interface with a query editor window. The query is:

```

14 SELECT * FROM flight WHERE strpos(flight_name, 'Delight') > 0;
15 SELECT * FROM flight1 WHERE strpos(flight_name, 'Delight') > 0;
16 SELECT * FROM flight2 WHERE strpos(flight_name, 'Delight') > 0;
17 -- Find all the user who booked atleast 9 tickets

```

The results table has the following columns:

flight_id	flight_name	flight_source	flight_destination	tickets_available

Message bar: Successfully run. Total query runtime: 2 secs 999 msec. 0 rows affected.

Result From original table

The screenshot shows the PgAdmin interface with a query editor window. The query is identical to the one above:

```

14 SELECT * FROM flight WHERE strpos(flight_name, 'Delight') > 0;
15 SELECT * FROM flight1 WHERE strpos(flight_name, 'Delight') > 0;
16 SELECT * FROM flight2 WHERE strpos(flight_name, 'Delight') > 0;
17 -- Find all the user who booked atleast 9 tickets

```

The results table has the following columns:

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	Dutch Delight	Madrid	Barcelona	50

Message bar: Successfully run. Total query runtime: 2 secs 788 msec. 1 rows affected.

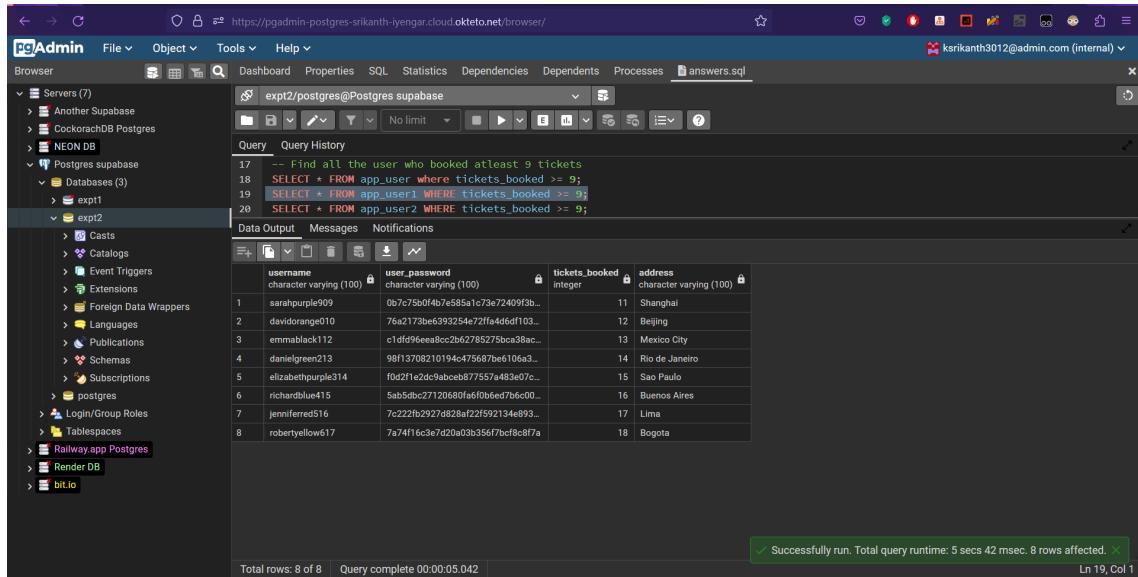
Query 5

```

-- Find all the user who booked atleast 9 tickets
SELECT * FROM app_user where tickets_booked >= 9;
SELECT * FROM app_user1 WHERE tickets_booked >= 9;
SELECT * FROM app_user2 WHERE tickets_booked >= 9;

```

Result From table 1

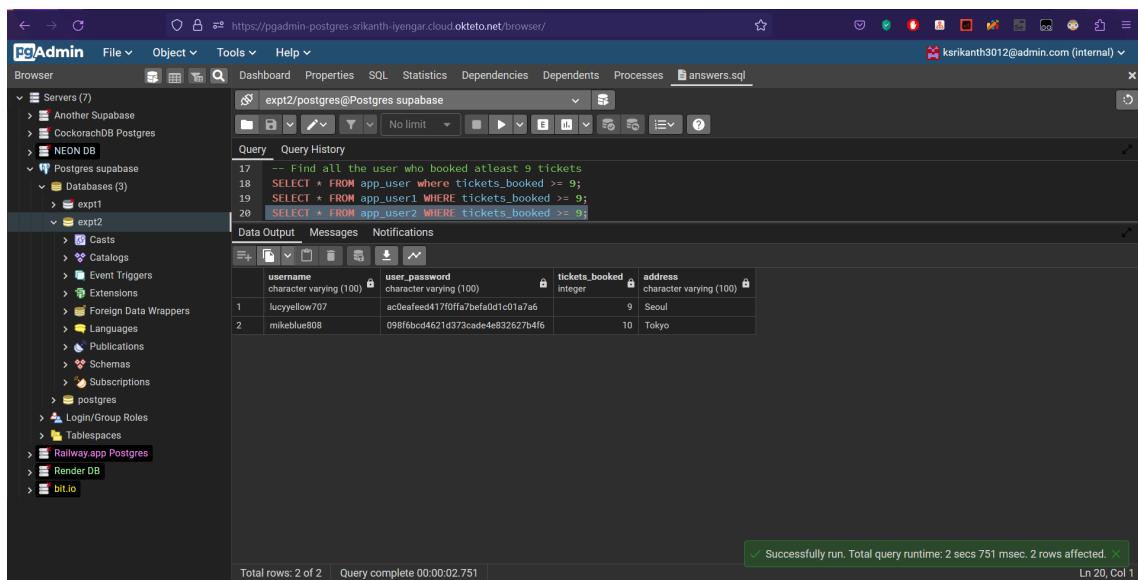


The screenshot shows the pgAdmin interface with the following details:

- Servers:** exp2/postgres@Postgres supabase
- Databases:** exp1, exp2
- Query:** A SQL query to find users who booked at least 9 tickets.
- Results:** 8 rows of data, each containing a username, password hash, number of tickets booked, and address.
- Message Bar:** "Successfully run. Total query runtime: 5 secs 42 msec. 8 rows affected."

username	user_password	tickets_booked	address
saraphurple909	0b7c758d04be585a1c73e72409fb...	11	Shanghai
davidorango010	76a2173be6393254e72ffad4d1f03...	12	Beijing
emmaback112	c1df95ee8ec2662785279ca38ac...	13	Mexico City
danielgreen213	98f1370821094e475687b610ea3...	14	Rio de Janeiro
elizabethpurple314	f0d2f12e2dc9abce8b77557a483e07...	15	Sao Paulo
richardblue415	5a55be2772068fb6f0bde67b6c00...	16	Buenos Aires
jennifered16	7c22fb2927d828af22f992134e893...	17	Lima
robertyellow617	7a74f16c3e7d20a03b356f7bcf8c8f7a	18	Bogota

Result From table 2



The screenshot shows the pgAdmin interface with the following details:

- Servers:** exp2/postgres@Postgres supabase
- Databases:** exp1, exp2
- Query:** A SQL query to find users who booked at least 9 tickets.
- Results:** 2 rows of data, each containing a username, password hash, number of tickets booked, and address.
- Message Bar:** "Successfully run. Total query runtime: 2 secs 751 msec. 2 rows affected."

username	user_password	tickets_booked	address
lucyyellow707	ac0eafe4d4170ffa7befa0d1c01a7a6	9	Seoul
mikeblue808	098f6bcd4621d373cade4e832627b4f6	10	Tokyo

Result From original table

The screenshot shows the PgAdmin interface with the 'exp1' database selected. The query window contains the following SQL code:

```
-- Find all the user who booked atleast 9 tickets
SELECT * FROM app_user WHERE tickets_booked >= 9;
SELECT * FROM app_user1 WHERE tickets_booked >= 9;
SELECT * FROM app_user2 WHERE tickets_booked >= 9;
```

The results table displays 10 rows of data:

username	user_password	tickets_booked	address
lucyyellow707	ac0afeed4170fa7befa01c01a7a6	9	Seoul
mikeblue808	098fbcd4621d373cade4e83267b4...	10	Tokyo
saraphpurple909	0b7c75d94b7e585a1c3e72409f2b...	11	Shanghai
davidorange010	76ea21730e6393254a72ffad4df103...	12	Beijing
emmablack112	c1df969eaa8cc2b6278527bca38ac...	13	Mexico City
daniellegreen213	98f13708210194a756876ed166a3...	14	Rio de Janeiro
elizabetpurple314	f0d2f1e2cd9abcebb77557a483e07c...	15	Sao Paulo
richardblue415	5a55bcb27120680fa6fb0bed7b6c00...	16	Buenos Aires
jennifered516	7c222fb2927d828af22f592134e893...	17	Lima
robertyellow617	7a74f16c3e7d20a03b356f7bcf8c8f7a	18	Bogota

Total rows: 10 of 10 | Query complete 00:00:02.100 | Ln 18, Col 1

Query 6

```
-- Find all the user who live in mumbai
SELECT * FROM app_user WHERE address = 'Paris';
SELECT * FROM app_user1 WHERE address = 'Paris';
SELECT * FROM app_user2 WHERE address = 'Paris';
```

Result From table 1

The screenshot shows the PgAdmin interface with the 'exp1' database selected. The query window contains the following SQL code:

```
-- Find all the user who live in mumbai
SELECT * FROM app_user WHERE address = 'Paris';
SELECT * FROM app_user1 WHERE address = 'Paris';
SELECT * FROM app_user2 WHERE address = 'Paris';
```

The results table displays 0 rows of data:

username	user_password	tickets_booked	address
----------	---------------	----------------	---------

Total rows: 0 of 0 | Query complete 00:00:04.390 | Successfully run. Total query runtime: 4 secs 390 msec. 0 rows affected. | Ln 23, Col 2

Result From table 2

The screenshot shows the PgAdmin 4 interface with the following details:

- Browser:** Postgres supabase
- Query:**

```

21 -- Find all the user who live in mumbai
22 SELECT * FROM app_user WHERE address = 'Paris';
23 SELECT * FROM app_user1 WHERE address = 'Paris';
24 SELECT * FROM app_user2 WHERE address = 'Paris';

```
- Data Output:** A table showing the results of the query. The columns are: username, user_password, tickets_booked, and address.

	username	user_password	tickets_booked	address
1	bobsmith789	21232f297a57a5a743894a0e4a801fc3	4	Paris
2	paulred06	1a1dc91c907325c69271dd0c944bc72	8	Paris

Result From original table

The screenshot shows the PgAdmin 4 interface with the following details:

- Browser:** Postgres supabase
- Query:**

```

21 -- Find all the user who live in mumbai
22 SELECT * FROM app_user WHERE address = 'Paris';
23 SELECT * FROM app_user1 WHERE address = 'Paris';
24 SELECT * FROM app_user2 WHERE address = 'Paris';

```
- Data Output:** A table showing the results of the query. The columns are: username, user_password, tickets_booked, and address.

	username	user_password	tickets_booked	address
1	bobsmith789	21232f297a57a5a743894a0e4a801fc3	4	Paris
2	paulred06	1a1dc91c907325c69271dd0c944bc72	8	Paris

Message bar: Successfully run. Total query runtime: 2 secs 507 msec. 2 rows affected.

Query 7

```

-- Find all flights with capacity between 40 to 70
SELECT * FROM flight where tickets_available BETWEEN 40 AND 70;
SELECT * FROM flight1 where tickets_available BETWEEN 40 AND 70;
SELECT * FROM flight2 where tickets_available BETWEEN 40 AND 70;

```

Result From table 1

The screenshot shows the PgAdmin 4 interface with the following details:

- Left Panel (Browser):** Shows the database structure with servers, databases, and tables.
- Top Bar:** Includes File, Object, Tools, Help, Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and a tab for "answers.sql".
- Query Editor:** Contains the following SQL code:

```
25 -- Find all flights with capacity between 40 to 70
26 SELECT * FROM flight where tickets_available BETWEEN 40 AND 70;
27 SELECT * FROM flight1 where tickets_available BETWEEN 40 AND 70;
28 SELECT * FROM flight2 where tickets_available BETWEEN 40 AND 70;
```
- Data Output:** Displays the results of the query, which are identical to the results shown in the second screenshot.
- Bottom Status:** Shows "Total rows: 4 of 4" and "Query complete 00:00:02.988".

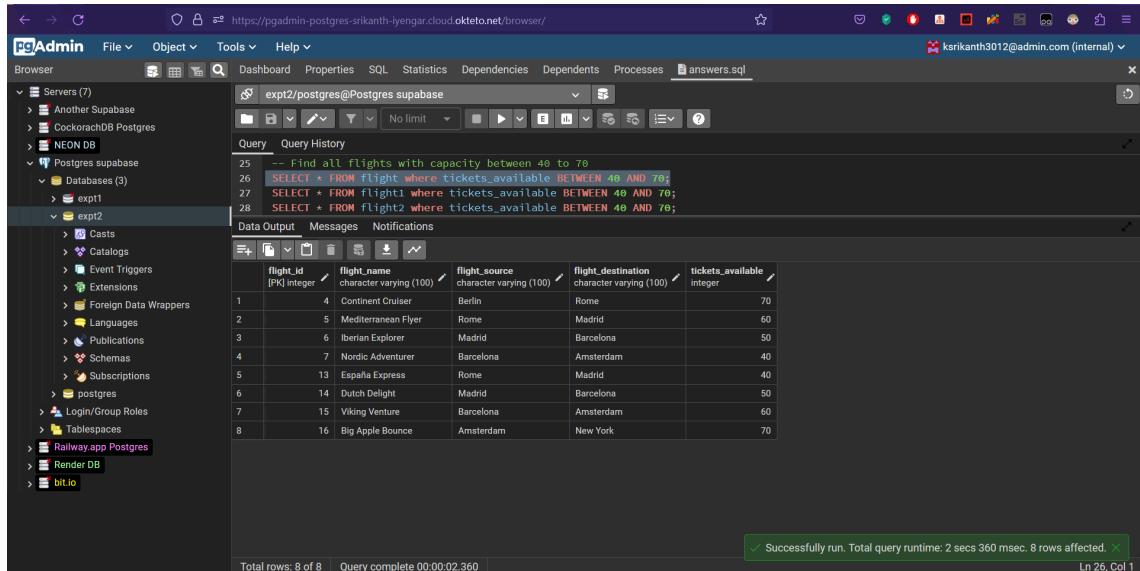
Result From table 2

The screenshot shows the PgAdmin 4 interface with the following details:

- Left Panel (Browser):** Shows the database structure with servers, databases, and tables.
- Top Bar:** Includes File, Object, Tools, Help, Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and a tab for "answers.sql".
- Query Editor:** Contains the following SQL code:

```
25 -- Find all flights with capacity between 40 to 70
26 SELECT * FROM flight where tickets_available BETWEEN 40 AND 70;
27 SELECT * FROM flight1 where tickets_available BETWEEN 40 AND 70;
28 SELECT * FROM flight2 where tickets_available BETWEEN 40 AND 70;
```
- Data Output:** Displays the results of the query, which are identical to the results shown in the first screenshot.
- Bottom Status:** Shows "Total rows: 4 of 4", "Query complete 00:00:02.996", and a message "Successfully run. Total query runtime: 2 secs 996 msec. 4 rows affected."

Result From original table



The screenshot shows the pgAdmin interface with the following details:

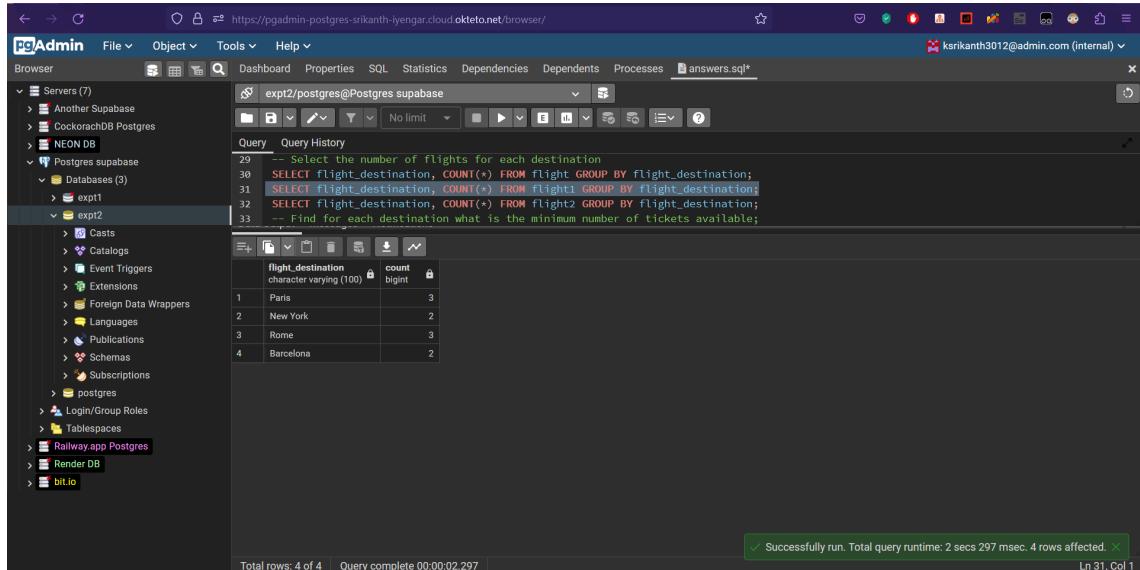
- Servers:** Postgres supabase, NEON DB.
- Databases:** exp1, exp2.
- Query Editor:** The query is run against the exp2 database. The results show 8 rows of flight information.
- Table Headers:** flight_id [PK] integer, flight_name character varying(100), flight_source character varying(100), flight_destination character varying(100), tickets_available integer.
- Table Data:**

flight_id	flight_name	flight_source	flight_destination	tickets_available
1	Continent Cruiser	Berlin	Rome	70
2	Mediterranean Flyer	Rome	Madrid	60
3	Iberian Explorer	Madrid	Barcelona	50
4	Nordic Adventurer	Barcelona	Amsterdam	40
5	España Express	Rome	Madrid	40
6	Dutch Delight	Madrid	Barcelona	50
7	Viking Venture	Barcelona	Amsterdam	60
8	Big Apple Bounce	Amsterdam	New York	70
- Message Bar:** Successfully run. Total query runtime: 2 secs 360 msec. 8 rows affected.

Query 8

```
-- Select the number of flights for each destination
SELECT flight_destination, COUNT(*) FROM flight GROUP BY flight_destination;
SELECT flight_destination, COUNT(*) FROM flight1 GROUP BY flight_destination;
SELECT flight_destination, COUNT(*) FROM flight2 GROUP BY flight_destination;
```

Result From table 1



The screenshot shows the pgAdmin interface with the following details:

- Servers:** Postgres supabase, NEON DB.
- Databases:** exp1, exp2.
- Query Editor:** The query is run against the exp2 database. The results show 4 rows of flight destination counts.
- Table Headers:** flight_destination character varying(100), count bigint.
- Table Data:**

flight_destination	count
Paris	3
New York	2
Rome	3
Barcelona	2
- Message Bar:** Successfully run. Total query runtime: 2 secs 297 msec. 4 rows affected.

Result From table 2

The screenshot shows the pgAdmin 4 interface with the 'Answers' tab selected. The left sidebar shows a tree view of databases and tables. The main area contains a query editor with the following SQL code:

```

-- Select the number of flights for each destination
SELECT flight_destination, COUNT(*) FROM flight GROUP BY flight_destination;
SELECT flight_destination, COUNT(*) FROM flight1 GROUP BY flight_destination;
SELECT flight_destination, COUNT(*) FROM flight2 GROUP BY flight_destination;
-- Find for each destination what is the minimum number of tickets available;

```

Below the code, a results grid displays the data:

flight_destination	count
Berlin	3
London	3
Amsterdam	2
Madrid	2

Total rows: 4 of 4 Query complete 00:00:01.939 Ln 32, Col 1

Result From original table

The screenshot shows the pgAdmin 4 interface with the 'Answers' tab selected. The left sidebar shows a tree view of databases and tables. The main area contains a query editor with the same SQL code as the previous screenshot.

Below the code, a results grid displays the data:

flight_destination	count
Paris	3
New York	2
Rome	3
Berlin	3
Barcelona	2
London	3
Amsterdam	2
Madrid	2

Total rows: 8 of 8 Query complete 00:00:01.871 Successfully run. Total query runtime: 1 secs 871 msec. 8 rows affected. Ln 30, Col 1

Query 9

```

-- Find for each destination what is the minimum number of tickets available;
SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight GROUP BY
flight_destination;
SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight1 GROUP BY
flight_destination;
SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight2 GROUP BY
flight_destination;

```

Result From table 1

The screenshot shows the PgAdmin 4 interface with a dark theme. The left sidebar lists several databases, including 'exp1' and 'exp2'. The current database is 'exp2'. The main area displays a SQL query and its results. The query is as follows:

```
33 -- Find for each destination what is the minimum number of tickets available;
34 SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight GROUP BY flight_destination;
35 SELECT flight_destination, MIN(tickets_available) AS Tickets FROM flight1 GROUP BY flight_destination;
36 SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight2 GROUP BY flight_destination;
37 -- Find the number of tickets available for all the destination
```

The results table has two columns: 'flight_destination' and 'tickets'. The data is:

flight_destination	tickets
Paris	10
New York	30
Rome	30
Barcelona	50

At the bottom, a green status bar indicates: 'Successfully run. Total query runtime: 2 secs 486 msec. 4 rows affected.' and 'Ln 35, Col 1'.

Result From table 2

The screenshot shows the PgAdmin 4 interface with a dark theme. The left sidebar lists several databases, including 'exp1' and 'exp2'. The current database is 'exp2'. The main area displays a SQL query and its results. The query is identical to the one in the previous screenshot:

```
33 -- Find for each destination what is the minimum number of tickets available;
34 SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight GROUP BY flight_destination;
35 SELECT flight_destination, MIN(tickets_available) AS Tickets FROM flight1 GROUP BY flight_destination;
36 SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight2 GROUP BY flight_destination;
37 -- Find the number of tickets available for all the destination
```

The results table has two columns: 'flight_destination' and 'tickets'. The data is:

flight_destination	tickets
Berlin	20
London	20
Amsterdam	40
Madrid	40

At the bottom, a green status bar indicates: 'Successfully run. Total query runtime: 1 secs 836 msec. 4 rows affected.' and 'Ln 35, Col 104'.

Result From original table

The screenshot shows the pgAdmin interface with the 'exp1' database selected. The query window contains the following SQL code:

```
33 -- Find for each destination what is the minimum number of tickets available;
34 SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight GROUP BY flight_destination;
35 SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight1 GROUP BY flight_destination;
36 SELECT flight_destination, MIN(tickets_available) AS tickets FROM flight2 GROUP BY flight_destination;
```

The results table shows the minimum number of tickets available for each destination:

flight_destination	tickets
Paris	10
New York	30
Rome	30
Berlin	20
Barcelona	50
London	20
Amsterdam	40
Madrid	40

Total rows: 8 of 8 Query complete 00:00:07.413 Ln 33, Col 79

Query 10

```
-- Find the number of tickets available for all the destination
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM
flight GROUP BY flight_destination;
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM
flight1 GROUP BY flight_destination;
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM
flight2 GROUP BY flight_destination;
```

Result From table 1

The screenshot shows the pgAdmin interface with the 'exp1' database selected. The query window contains the following SQL code:

```
37 -- Find the number of tickets available for all the destination
38 SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM flight GROUP BY flight_destination;
39 SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM flight1 GROUP BY flight_destination;
40 SELECT flight_destination AS destination, SUM(tickets.available) AS tickets FROM flight2 GROUP BY flight_destination;
```

The results table shows the total number of tickets available for each destination:

destination	tickets
Paris	190
New York	100
Rome	210
Barcelona	100

✓ Successfully run. Total query runtime: 2 secs 359 msec. 4 rows affected. Ln 39, Col 1

Total rows: 4 of 4 Query complete 00:00:02.399

Result From table 2

```

-- Find the number of tickets available for all the destination
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM flight GROUP BY flight_destination;
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM flight1 GROUP BY flight_destination;
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM flight2 GROUP BY flight_destination;

```

destination	tickets
Berlin	200
London	200
Amsterdam	100
Madrid	100

Total rows: 4 of 4 Query complete 00:00:01.859 Successfully run. Total query runtime: 1 secs 859 msec. 4 rows affected. Ln 40, Col 1

Result From original table

```

-- Find the number of tickets available for all the destination
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM flight GROUP BY flight_destination;
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM flight1 GROUP BY flight_destination;
SELECT flight_destination AS destination, SUM(tickets_available) AS tickets FROM flight2 GROUP BY flight_destination;

```

destination	tickets
Paris	190
New York	100
Rome	210
Berlin	200
Barcelona	100
London	200
Amsterdam	100
Madrid	100

Total rows: 8 of 8 Query complete 00:00:01.820 Successfully run. Total query runtime: 1 secs 820 msec. 8 rows affected. Ln 38, Col 1

PART B

- Creating a procedure which will insert the table if the hash value is even or odd

```

CREATE OR REPLACE PROCEDURE insert_flight (
    flight_id integer,
    flight_name varchar(100),
    flight_source varchar(100),
    flight_destination varchar(100),
    tickets_available integer
)
LANGUAGE plpgsql AS $$

DECLARE

```

```
hash int;
BEGIN
    hash := hashtext(flight_source);
    RAISE NOTICE 'Value of hash is :- %', hash;
    IF MOD(ABS(hash), 2) = 0 THEN
        RAISE NOTICE 'Inserting the record in table 1';
        INSERT INTO dummy_flight1 VALUES(flight_id, flight_name, flight_source,
flight_destination, tickets_available);
    END IF;
    IF MOD(ABS(hash), 2) = 1 THEN
        RAISE NOTICE 'Inserting the record in table 2';
        INSERT INTO dummy_flight2 VALUES(flight_id, flight_name, flight_source,
flight_destination, tickets_available);
    END IF;
END;$$
```

Inserting 4 records and it gets automatically added to the respective table

The screenshot shows the pgAdmin 4 interface. The left sidebar lists several databases, including 'expt1' and 'expt2'. The current session is connected to 'expt2/postgres@Postgres supabase'. The main window displays a SQL query editor with the following code:

```
1 call insert_flight('31', 'Jet Airways', 'London', 'Dubai', 10);
2 call insert_flight('31', 'Jet Airways', 'Mumbai', 'London', 10);
3
4
5
6 select * from dummy_flight2;
```

The 'Data Output' tab shows the results of the last query:

```
Query returned successfully in 1 secs 80 msec.
```

A status bar at the bottom indicates:

Total rows: 1 of 1 Query complete 00:00:01.080

Ln 1, Col 1

This screenshot is nearly identical to the one above, showing the same pgAdmin 4 interface and session details. The SQL query is identical:

```
1 call insert_flight('31', 'Jet Airways', 'London', 'Dubai', 10);
2 call insert_flight('41', 'Jet Airways', 'Mumbai', 'London', 10);
3
4
5
6 select * from dummy_flight2;
```

The 'Data Output' tab shows the results of the last query:

```
Query returned successfully in 1 secs 133 msec.
```

A status bar at the bottom indicates:

Total rows: 1 of 1 Query complete 00:00:01.133

Ln 2, Col 1

PGAdmin File Object Tools Help

Browser https://pgadmin-postgres-srikanth-iyengar.cloud.okteto.net/browser/ ksrikanth3012@admin.com (internal)

Query History

```
1 call insert_flight('31', 'Jet Airways', 'London', 'Dubai', 10);
2 call insert_flight('41', 'Jet Airways', 'Mumbai', 'London', 10);
3 call insert_flight('51', 'King Fisher', 'Banglore', 'Washington DC');
4
5
6 select * from dummy_flight2;
```

Data Output Messages Notifications

NOTICE: Value of hash is :- -334786941
NOTICE: Inserting the record in table 2
CALL

Query returned successfully in 1 secs 216 msec.

Total rows: 1 of 1 Query complete 00:00:01.216 Ln 3, Col 1

PGAdmin File Object Tools Help

Browser https://pgadmin-postgres-srikanth-iyengar.cloud.okteto.net/browser/ ksrikanth3012@admin.com (internal)

Query History

```
1 call insert_flight('31', 'Jet Airways', 'London', 'Dubai', 10);
2 call insert_flight('41', 'Jet Airways', 'Mumbai', 'London', 10);
3 call insert_flight('51', 'King Fisher', 'Banglore', 'Washington DC');
4 call insert_flight('61', 'King Fisher', 'London', 'Washington DC');
5
6
7 select * from dummy_flight2;
```

Data Output Messages Notifications

NOTICE: Value of hash is :- -221836414
NOTICE: Inserting the record in table 1
CALL

Query returned successfully in 877 msec.

Total rows: 1 of 1 Query complete 00:00:00.877 Ln 4, Col 1

```

call insert_flight('31', 'Jet Airways', 'London', 'Dubai', 10);
call insert_flight('41', 'Jet Airways', 'Mumbai', 'London', 10);
call insert_flight('51', 'Kind Fisher', 'Banglore', 'Washington DC', 10);
call insert_flight('61', 'Kind Fisher', 'London', 'Washington DC', 10);
select * from dummy_flight1;
select * from dummy_flight2;

```

```

call insert_flight('31', 'Jet Airways', 'London', 'Dubai', 10);
call insert_flight('41', 'Jet Airways', 'Mumbai', 'London', 10);
call insert_flight('51', 'Kind Fisher', 'Banglore', 'Washington DC', 10);
call insert_flight('61', 'Kind Fisher', 'London', 'Washington DC', 10);
select * from dummy_flight1;
select * from dummy_flight2;

```

Successfully run. Total query runtime: 3 secs 117 msec. 2 rows affected.

Conclusion

Learned how to design a distributed database by applying the concept of horizontal fragmentation in PostgreSQL.