Manipulating Data

Let us understand how we can manipulate data for a partitioned table.

- We can insert data using the table (most preferred way).
- As we define table for each partition, we can insert data using table created for specific partition.
- In the case of `users_part` partitioned table, we can either use table name`users_part` or partition name `users_part_u` to insert records with user_role **'U'**.

```
CREATE TABLE users_part_u
PARTITION OF users_part
FOR VALUES IN ('U')
```

- As part of the update, if we change the value in a partitioned column which will result in changing partition, then internally data from one partition will be moved to other.
- We can delete the data using the table or the table created for each partition (either by using table name `users_part` or partitions such as `users_part_u`, `users_part_a` etc

Note

DML is same irrespective of the partitioning strategy. This applies to all 3 partitioning strategies - **list**, **range** as well as **hash**.

```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

```
%env DATABASE_URL=postgresql://itversity_sms_user:sms_password@localhost:5432/itversity_sms_db
```

```
env:
DATABASE_URL=postgresql://itversity_sms_user:sms_password@localhost:5432/itversity_sms_db
```

```
%%sql

TRUNCATE TABLE users_part
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
Done.
```

```
[]
```

```
%%sql

INSERT INTO users_part (user_first_name, user_last_name, user_email_id, user_role)
VALUES
    ('Scott', 'Tiger', 'scott@tiger.com', 'U'),
    ('Donald', 'Duck', 'donald@duck.com', 'U'),
    ('Mickey', 'Mouse', 'mickey@mouse.com', 'U')
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
3 rows affected.
```

```
[]
```

```
%%sql

SELECT * FROM users_part_u
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
3 rows affected.
```

user_id	user_first_name	user_last_name	user_email_id	user_email_validated	user_passw
5	Scott	Tiger	scott@tiger.com	False	N
6	Donald	Duck	donald@duck.com	False	N
7	Mickey	Mouse	mickey@mouse.com	False	N

```
%%sql

INSERT INTO users_part_a (user_first_name, user_last_name, user_email_id, user_role)
VALUES
    ('Matt', 'Clarke', 'matt@clarke.com', 'A')
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
1 rows affected.
```

```
[]
```

```
%%sql

SELECT * FROM users_part
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
4 rows affected.
```

user_id	user_first_name	user_last_name	user_email_id	user_email_validated	user_passw
8	Matt	Clarke	matt@clarke.com	False	N
5	Scott	Tiger	scott@tiger.com	False	N
6	Donald	Duck	donald@duck.com	False	N
7	Mickey	Mouse	mickey@mouse.com	False	N

```
%%sql

UPDATE users_part SET
    user_role = 'A'
WHERE user_email_id = 'donald@duck.com'
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
1 rows affected.
```

```
[]
```

```
%%sql

SELECT * FROM users_part_a
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
2 rows affected.
```

user_id	user_first_name	user_last_name	user_email_id	user_email_validated	user_passwor
8	Matt	Clarke	matt@clarke.com	False	Non
6	Donald	Duck	donald@duck.com	False	Non

```
%%sql

DELETE FROM users_part WHERE user_email_id = 'donald@duck.com'
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
1 rows affected.
```

```
[]
```

```
%%sql

DELETE FROM users_part_u WHERE user_email_id = 'mickey@mouse.com'
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
1 rows affected.
```

```
[]
```

```
%%sql

SELECT * FROM users_part
```

```
 * postgresql://itversity_sms_user:***@localhost:5432/itversity_sms_db
2 rows affected.
```

user_id	user_first_name	user_last_name	user_email_id	user_email_validated	user_passwor
8	Matt	Clarke	matt@clarke.com	False	None
5	Scott	Tiger	scott@tiger.com	False	None

By Durga Gadiraju