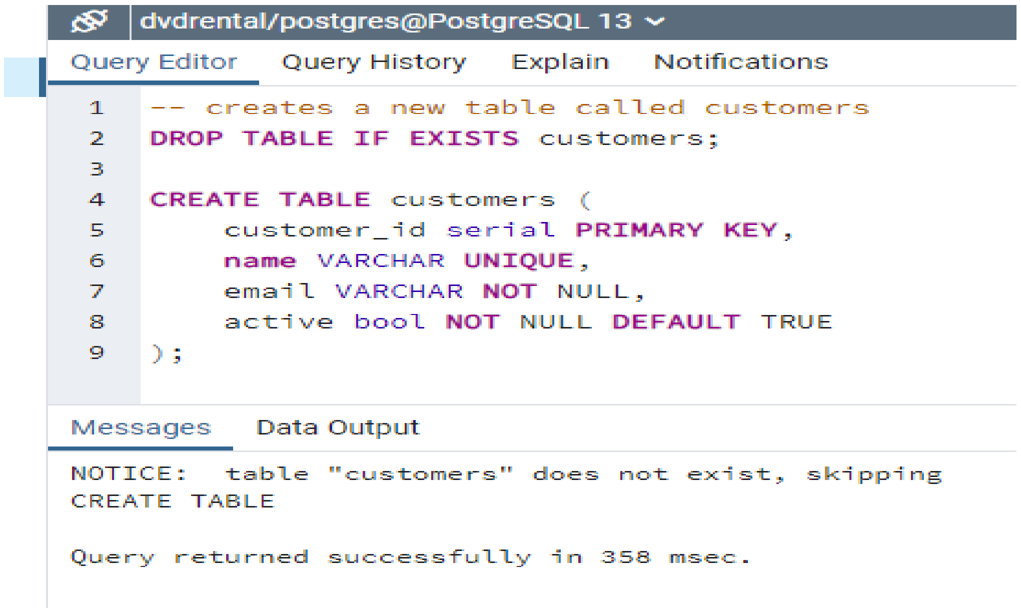
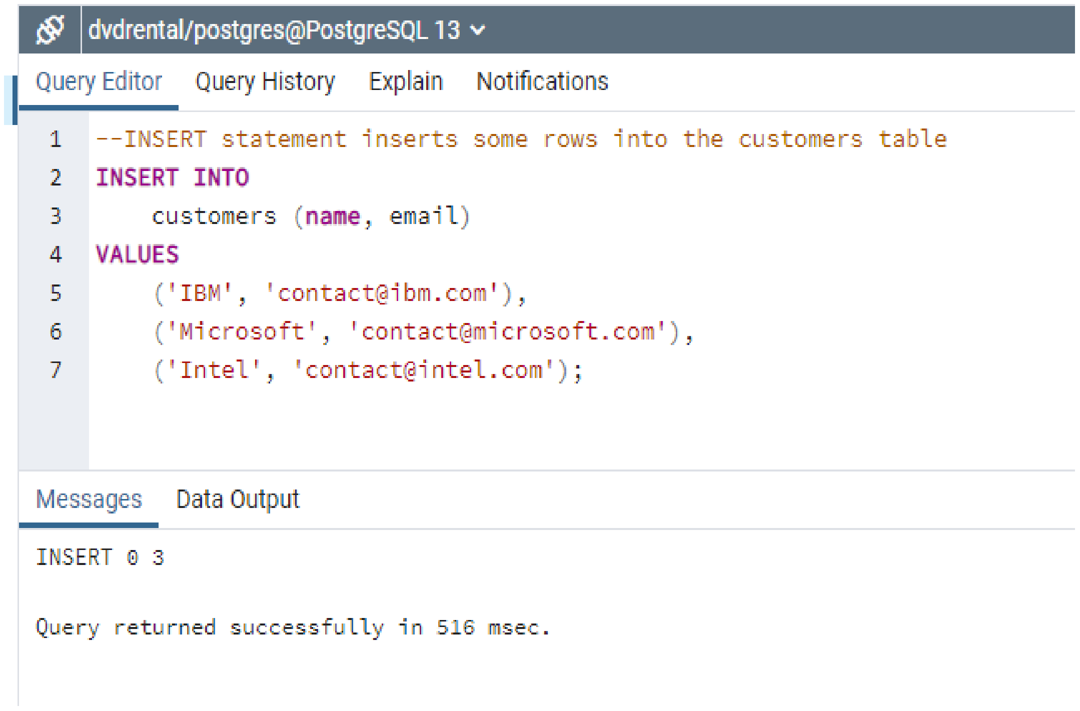
POSTGRESQL UPSERT



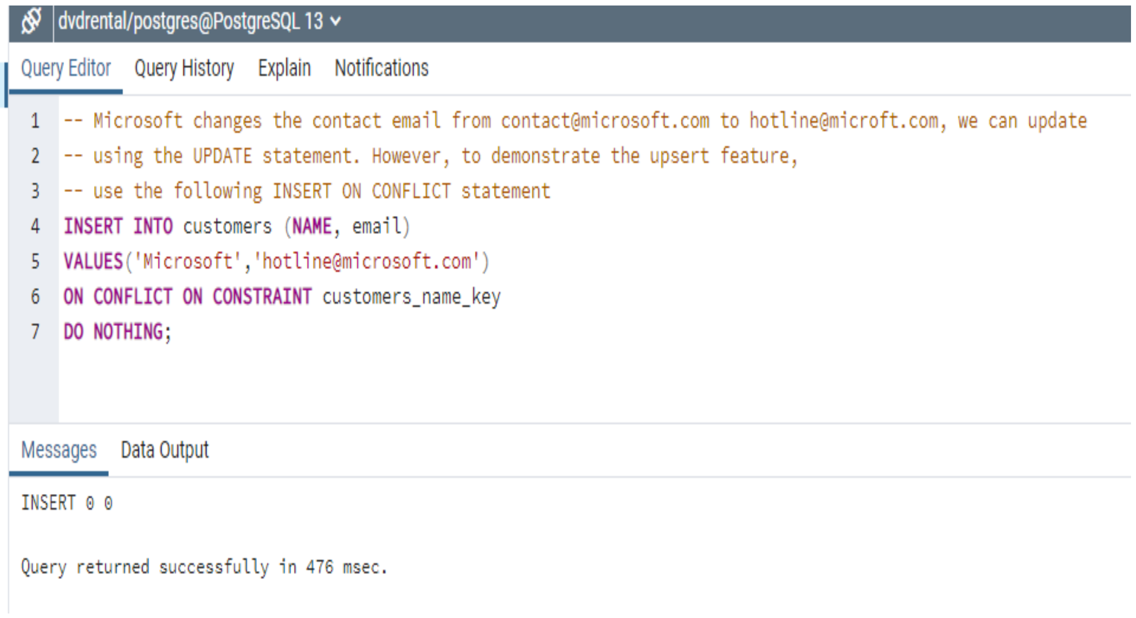
```sql
-- creates a new table called customers
DROP TABLE IF EXISTS customers;

CREATE TABLE customers (
    customer_id serial PRIMARY KEY,
    name VARCHAR UNIQUE,
    email VARCHAR NOT NULL,
    active bool NOT NULL DEFAULT TRUE
);
```

```
NOTICE:  table "customers" does not exist, skipping
CREATE TABLE

Query returned successfully in 358 msec.
```



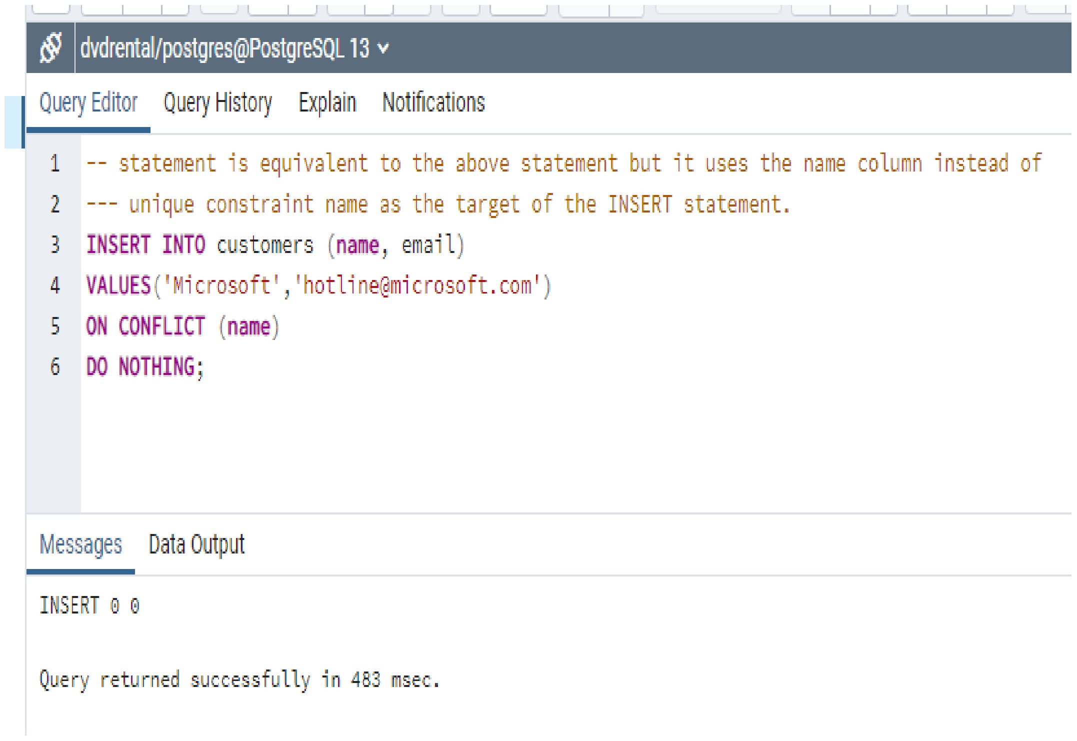
```sql
--INSERT statement inserts some rows into the customers table
INSERT INTO
    customers (name, email)
VALUES
    ('IBM', 'contact@ibm.com'),
    ('Microsoft', 'contact@microsoft.com'),
    ('Intel', 'contact@intel.com');
```

```
INSERT 0 3

Query returned successfully in 516 msec.
```



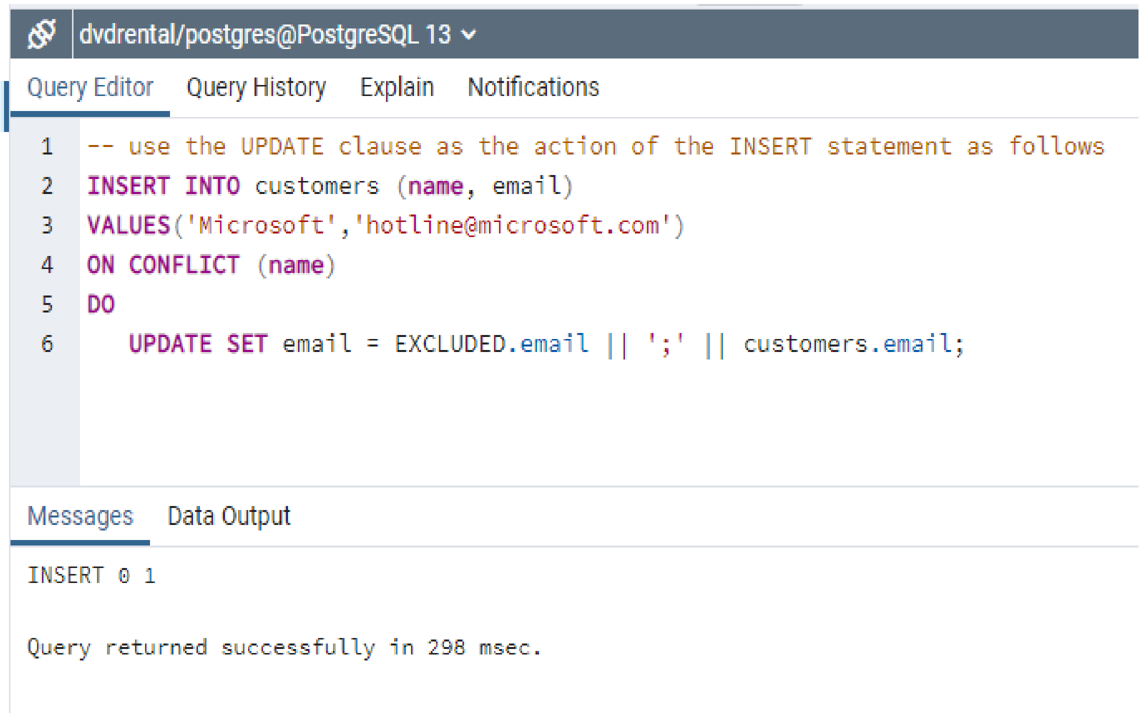
```sql
-- Microsoft changes the contact email from contact@microsoft.com to hotline@microft.com, we can update
-- using the UPDATE statement. However, to demonstrate the upsert feature,
-- use the following INSERT ON CONFLICT statement
INSERT INTO customers (NAME, email)
VALUES('Microsoft','hotline@microsoft.com')
ON CONFLICT ON CONSTRAINT customers_name_key
DO NOTHING;
```

```
INSERT 0 0

Query returned successfully in 476 msec.
```



```sql
-- statement is equivalent to the above statement but it uses the name column instead of
--- unique constraint name as the target of the INSERT statement.
INSERT INTO customers (name, email)
VALUES('Microsoft','hotline@microsoft.com')
ON CONFLICT (name)
DO NOTHING;
```

```
INSERT 0 0

Query returned successfully in 483 msec.
```



```sql
-- use the UPDATE clause as the action of the INSERT statement as follows
INSERT INTO customers (name, email)
VALUES('Microsoft','hotline@microsoft.com')
ON CONFLICT (name)
DO
   UPDATE SET email = EXCLUDED.email || ';' || customers.email;
```

```
INSERT 0 1

Query returned successfully in 298 msec.
```