

ASSIGNMENT-2

NAME: CH POOJA

REG. NO.: 20BCE7630

Creating a Table:

CODE:

```
CREATE TABLE Employee(
```

```
    EmpId int,
```

```
    LastName varchar(255),
```

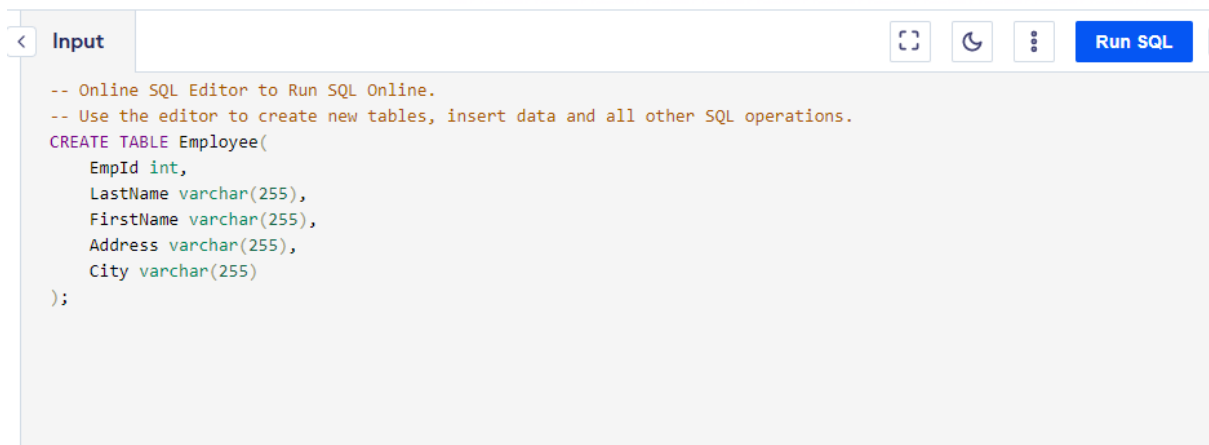
```
    FirstName varchar(255),
```

```
    Address varchar(255),
```

```
    City varchar(255)
```

```
);
```

SCREENSHOT:



The screenshot shows an online SQL editor interface. At the top, there is a tab labeled 'Input' and a 'Run SQL' button. The main text area contains the following SQL code:

```
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.
CREATE TABLE Employee(
    EmpId int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

OUTPUT:

Employee				
EmpId	LastName	FirstName	Address	City
empty				

Inserting Values in the table:

CODE:

```
INSERT INTO Employee(EmpId,LastName,FirstName,ADDRESS,City)
VALUES (1, 'pooja', 'Khushi', 'India', 'Delhi' );

INSERT INTO Employee (EmpId,LastName,FirstName,ADDRESS,City)
VALUES (2, 'vivek', 'Y', 'India', 'Mumbai' );
```

```
INSERT INTO Employee
VALUES (3, 'Krishna', 'C', 'India', 'Chennai' );
```

SCREENSHOT:

A screenshot of a SQL IDE interface. The top bar has a back arrow, the word 'Input', and icons for full screen, dark mode, and a menu. A 'Run SQL' button is on the right. The main area contains the following SQL code:

```
INSERT INTO Employee(EmpId,LastName,FirstName,ADDRESS,City)
VALUES (1, 'pooja', 'Khushi', 'India', 'Delhi' );
INSERT INTO Employee (EmpId,LastName,FirstName,ADDRESS,City)
VALUES (2, 'vivek', 'Y', 'India', 'Mumbai' );

INSERT INTO Employee
VALUES (3, 'Krishna', 'C', 'India', 'Chennai' );
|
```

OUTPUT:

Employee

EmpId	LastName	FirstName	Address	City
1	pooja	Khushi	India	Delhi
2	vivek	Y	India	Mumbai
3	Krishna	C	India	Chennai

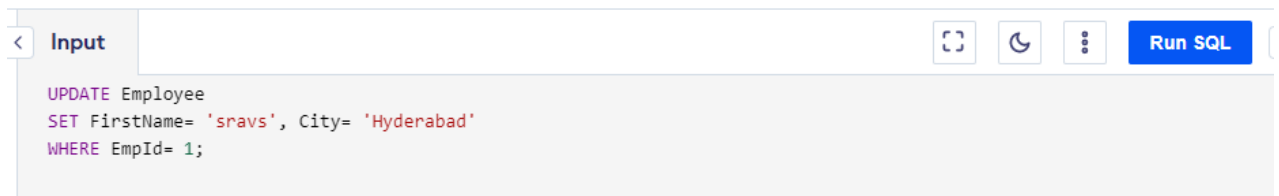
Updating a Table:

a) Updating single row:

CODE:

```
UPDATE Employee  
SET FirstName= 'sravs', City= 'Hyderabad'  
WHERE EmpId= 1;
```

SCREENSHOT:



OUTPUT:

Employee

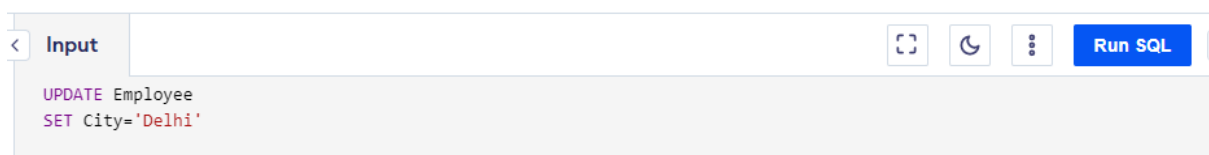
EmpId	LastName	FirstName	Address	City
1	pooja	sravs	India	Hyderabad
2	vivek	Y	India	Mumbai
3	Krishna	C	India	Chennai

b) Updating multiple rows:

CODE:

```
UPDATE Employee  
SET City='Delhi'
```

SCREENSHOT:



OUTPUT:

Employee				
EmpId	LastName	FirstName	Address	City
1	pooja	sravs	India	Delhi
2	vivek	Y	India	Delhi
3	Krishna	C	India	Delhi

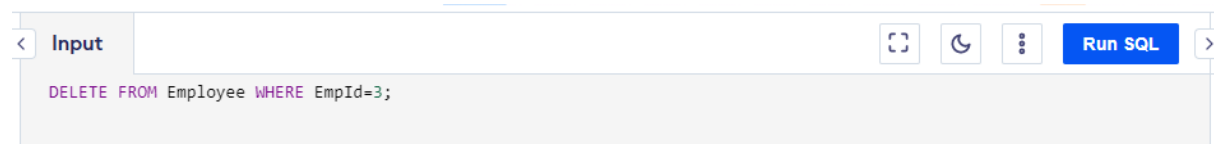
Deleting:

a) Deleting single row:

CODE:

DELETE FROM Employee WHERE EmpId=3;

SCREENSHOT:



OUTPUT:

Employee				
EmpId	LastName	FirstName	Address	City
1	pooja	sravs	India	Delhi
2	vivek	Y	India	Delhi

b) Deleting all records:

CODE:

DELETE From Employee

SCREENSHOT:

<

Input

Run SQL

>

DELETE From Employee

OUTPUT:

Employee				
EmpId	LastName	FirstName	Address	City
empty				

[illegible]

Creating tables and performing Joins:

Creating a Tables:

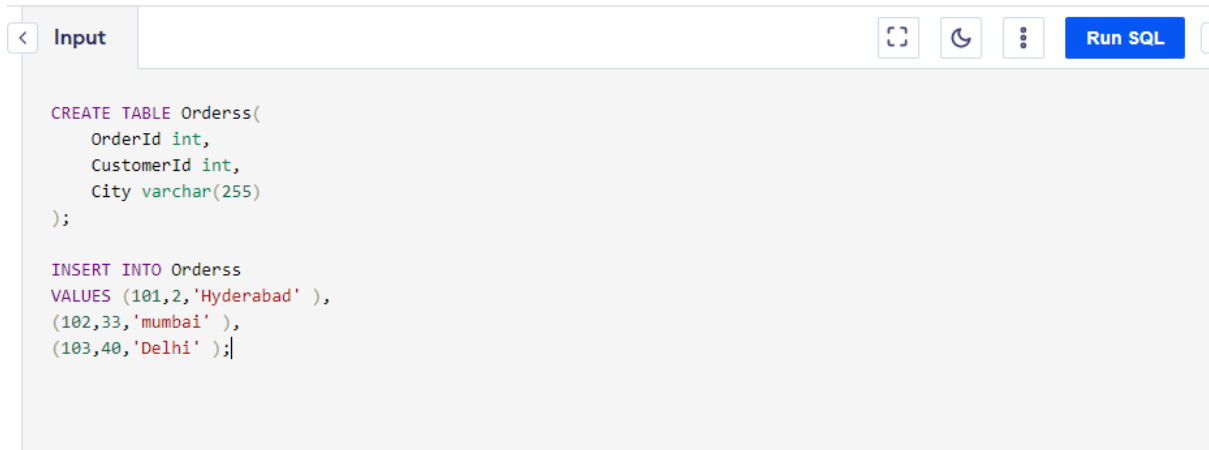
TABLE 1:

CODE:

```
CREATE TABLE Orderss(
    OrderId int,
    CustomerId int,
    City varchar(255)
);
```

```
INSERT INTO Orderss
VALUES (101,2,'Hyderabad' ),
(102,33,'mumbai' ),
(103,40,'Delhi' );
```

SCREENSHOT:



The screenshot shows a SQL IDE interface with a tab labeled 'Input'. The SQL code entered is as follows:

```
CREATE TABLE Orderss(  
    OrderId int,  
    CustomerId int,  
    City varchar(255)  
);  
  
INSERT INTO Orderss  
VALUES (101,2,'Hyderabad' ),  
(102,33,'mumbai' ),  
(103,40,'Delhi' );|
```

At the top right of the IDE, there are icons for full screen, dark mode, and a menu, along with a blue 'Run SQL' button.

OUTPUT:

Orderss

OrderId	CustomerId	City
101	2	Hyderabad
102	33	mumbai
103	40	Delhi

TABLE 2:

CODE:



```
CREATE TABLE Customerss(  
    CustomerId int,  
    CustomerName varchar(255),  
    LastName varchar(255),  
    Country varchar(255)  
);
```

```
INSERT INTO Customerss
```

```
VALUES (1,'pooja', 'CH','India' ),  
(2,'vivek', 'Y','Australia' ),  
(3,'krishna', 'p','USA' ),  
(4,'ruha', 'G','India' );
```

SCREENSHOT:

Input



Run SQL

```
CREATE TABLE Customerss(  
    CustomerId int,  
    CustomerName varchar(255),  
    LastName varchar(255),  
    Country varchar(255)  
);  
  
INSERT INTO Customerss  
VALUES (1,'pooja', 'CH','India' ),  
(2,'vivek', 'Y','Australia' ),  
(3,'krishna', 'p','USA' ),  
(4,'ruha', 'G','India' );
```

OUTPUT:

Customerss

CustomerId	CustomerName	LastName	Country
1	pooja	CH	India
2	vivek	Y	Australia
3	krishna	P	USA
4	ruha	G	India

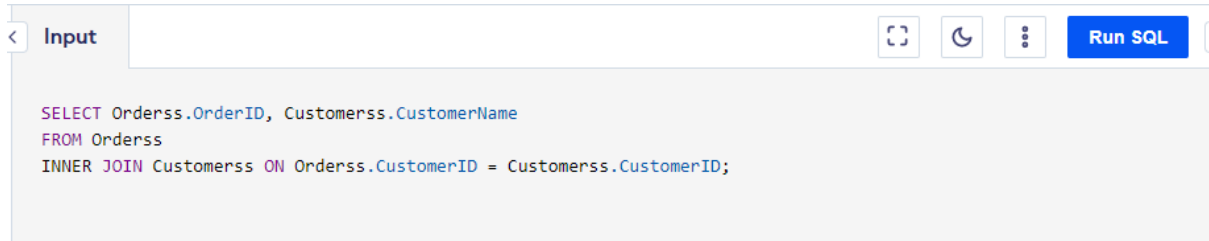
Performing Joins:

Inner Join:

CODE:

```
SELECT Orderss.OrderID, Customerss.CustomerName  
FROM Orderss  
INNER JOIN Customerss ON Orderss.CustomerID = Customerss.CustomerID;
```

SCREENSHOT:



A screenshot of a SQL query editor interface. The top bar has a tab labeled 'Input', a refresh icon, a settings icon, and a blue 'Run SQL' button. The main area contains the following SQL query:

```
SELECT Orderss.OrderID, Customerss.CustomerName
FROM Orderss
INNER JOIN Customerss ON Orderss.CustomerID = Customerss.CustomerID;
```

OUTPUT:

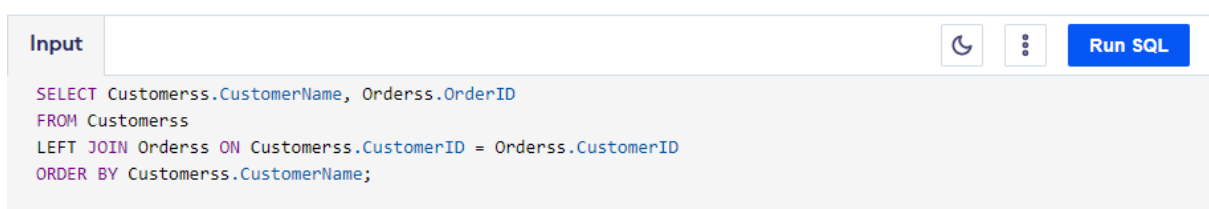
Output	
OrderId	CustomerName
101	vivek

Left Join:

CODE:

```
SELECT Customerss.CustomerName, Orderss.OrderID
FROM Customerss
LEFT JOIN Orderss ON Customerss.CustomerID = Orderss.CustomerID
ORDER BY Customerss.CustomerName;
```

SCREENSHOT:



A screenshot of a SQL query editor interface. The top bar has a tab labeled 'Input', a refresh icon, a settings icon, and a blue 'Run SQL' button. The main area contains the following SQL query:

```
SELECT Customerss.CustomerName, Orderss.OrderID
FROM Customerss
LEFT JOIN Orderss ON Customerss.CustomerID = Orderss.CustomerID
ORDER BY Customerss.CustomerName;
```


OUTPUT:

Output	
CustomerName	OrderId
krishna	
pooja	
vivek	101

Right Join:

CODE:

```
SELECT Orderss.CustomerId,Customerss.CustomerName,Customerss.LastName
FROM Orderss
RIGHT JOIN Customerss
ON Orderss.CustomerId = Customerss.CustomerId
ORDER BY Orderss.OrderID;
```

SCREENSHOT:

```
1 |
2 | SELECT Orderss.CustomerId,Customerss.CustomerName,Customerss.LastName
3 | FROM Orderss
4 | RIGHT JOIN Customerss
5 | ON Orderss.CustomerId = Customerss.CustomerId
6 | ORDER BY Orderss.OrderID;
7 |
```

OUTPUT:

Output		
CustomerId	CustomerName	LastName
NULL	pooja	CH
NULL	krishna	p
NULL	rucha	G
2	vivek	Y

[Execution complete with exit code 0]

Full outer Join:

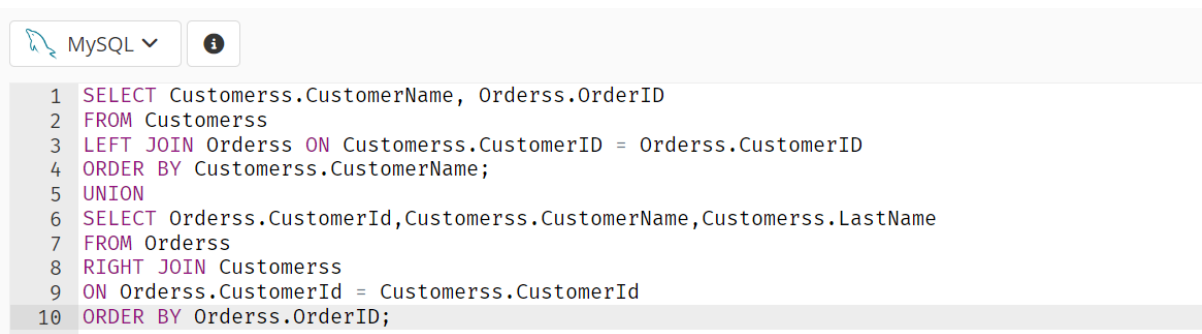
CODE:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;

UNION

SELECT Orders.CustomerId, Customers.CustomerName, Customers.LastName
FROM Orders
RIGHT JOIN Customers
ON Orders.CustomerId = Customers.CustomerId
ORDER BY Orders.OrderID;
```

SCREENSHOT:

A screenshot of a MySQL query editor interface. At the top, there is a toolbar with a MySQL logo and a dropdown menu showing 'MySQL'. Below the toolbar, the SQL query is displayed in a monospaced font, with line numbers 1 through 10 on the left. The query is a full outer join between Customers and Orders tables.

```
1 SELECT Customers.CustomerName, Orders.OrderID
2 FROM Customers
3 LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
4 ORDER BY Customers.CustomerName;
5 UNION
6 SELECT Orders.CustomerId, Customers.CustomerName, Customers.LastName
7 FROM Orders
8 RIGHT JOIN Customers
9 ON Orders.CustomerId = Customers.CustomerId
10 ORDER BY Orders.OrderID;
```

OUTPUT:

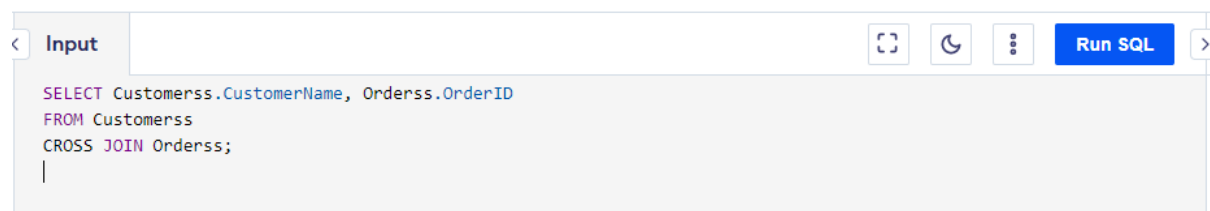
Output	
CustomerName	OrderID
krishna	NULL
pooja	NULL
rucha	NULL
vivek	101

Cross Join:

CODE:

```
SELECT Customerss.CustomerName, Orderss.OrderID  
FROM Customerss  
CROSS JOIN Orderss;
```

SCREENSHOT:



OUTPUT:

Output	
CustomerName	OrderId
pooja	101
pooja	102
pooja	103
vivek	101
vivek	102
vivek	103
krishna	101
krishna	102

Self Join:

```
SELECT A.CustomerName AS C1, B.CustomerName AS C2  
FROM Customerss A , Customerss B  
WHERE A.CustomerID <> B.CustomerID  
AND A.Country = B.Country  
ORDER BY A.Country;
```

SCREENSHOT:



OUTPUT:

Output	
C1	C2
pooja	ruch
ruch	pooja

[illegible]

MongoDB:

CREATING AND INSERTING:

a) Inserting one document:

CODE:

```
db.monuments.insertOne({
  "name": "The Pyramids of Giza",
  "city": "Giza",
  "country": "Egypt",
  "gps": {
    "lat": 29.976480,
    "lng": 31.131302
  }
})
```

)

SCREENSHOT:

```
MongoDB ▼ ⓘ
1 db.monuments.insertOne(
2   {
3     "name": "The Pyramids of Giza",
4     "city": "Giza",
5     "country": "Egypt",
6     "gps": {
7       "lat": 29.976480,
8       "lng": 31.131302
9     }
10  }
11 )
```

OUTPUT:

```
Output

mycompiler_mongodb> ... .. {
  acknowledged: true,
  insertedId: ObjectId("6473aba0526997f71675f4d6")
}
mycompiler_mongodb>

[Execution complete with exit code 0]
```

b) Inserting Many:

CODE:

```
db.monuments.insertMany([

  {"name": "The Valley of the Kings", "city": "Luxor", "country": "Egypt", "gps": { "lat": 25.746424,
  "lng": 32.605309 }},

  {"name": "Arc de Triomphe", "city": "Paris", "country": "France", "gps": { "lat": 48.873756, "lng":
  2.294946 }},

  {"name": "The Eiffel Tower", "city": "Paris", "country": "France", "gps": { "lat": 48.858093, "lng":
  2.294694 }},

  {"name": "Acropolis", "city": "Athens", "country": "Greece", "gps": { "lat": 37.970833, "lng":
  23.726110 }},

  {"name": "The Great Wall of China", "city": "Huairou", "country": "China", "gps": { "lat": 40.431908,
  "lng": 116.570374 }},

  {"name": "The Statue of Liberty", "city": "New York", "country": "USA", "gps": { "lat": 40.689247,
  "lng": -74.044502 }}

])
```

SCREENSHOT:

```
MongoDB ▼ ⓘ
1 db.monuments.insertMany([
2   { "name": "The Valley of the Kings", "city": "Luxor", "country": "Egypt", "gps": { "lat": 25.7464, "lng": 32.7022 } },
3   { "name": "Arc de Triomphe", "city": "Paris", "country": "France", "gps": { "lat": 48.873756, "lng": 2.295037 } },
4   { "name": "The Eiffel Tower", "city": "Paris", "country": "France", "gps": { "lat": 48.858093, "lng": 2.294491 } },
5   { "name": "Acropolis", "city": "Athens", "country": "Greece", "gps": { "lat": 37.970833, "lng": 7.796667 } },
6   { "name": "The Great Wall of China", "city": "Huairou", "country": "China", "gps": { "lat": 40.422222, "lng": 117.027778 } },
7   { "name": "The Statue of Liberty", "city": "New York", "country": "USA", "gps": { "lat": 40.689244, "lng": -74.044503 } },
8 ])
```

OUTPUT:

Output

```
mycompiler_mongodb> ... .. {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6473ac53e33b266674923789"),
    '1': ObjectId("6473ac53e33b26667492378a"),
    '2': ObjectId("6473ac53e33b26667492378b"),
    '3': ObjectId("6473ac53e33b26667492378c"),
    '4': ObjectId("6473ac53e33b26667492378d"),
    '5': ObjectId("6473ac53e33b26667492378e")
  }
}
mycompiler_mongodb>

[Execution complete with exit code 0]
```

UPDATING:

a) Single document:

CODE:

```
db.monuments.updateOne(
  { "name": "Arc de Triomphe" },
  {
    $set: { "name": "Arc de Triomphe de l'Étoile" }
  }
)
```

SCREENSHOT:

```
MongoDB ▼ ⓘ  
1 db.monuments.updateOne(  
2   { "name": "Arc de Triomphe" },  
3   {  
4     $set: { "name": "Arc de Triomphe de l'Étoile" }  
5   }  
6 )
```

OUTPUT:

Output

```
mycompiler_mongodb> ... .. {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 0,  
  modifiedCount: 0,  
  upsertedCount: 0  
}  
mycompiler_mongodb>  
  
[Execution complete with exit code 0]
```

b) Updating many:

CODE:

```
db.monuments.updateMany(  
  {},  
  {  
    $set: { "editor": "Sammy" }  
  }  
)
```

SCREENSHOT:

```
MongoDB ⓘ  
1 db.monuments.updateMany(  
2   { },  
3   {  
4     $set: { "editor": "Sammy" }  
5   }  
6 )
```

OUTPUT:

Output

```
mycompiler_mongodb> ... .. {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 0,  
  modifiedCount: 0,  
  upsertedCount: 0  
}  
mycompiler_mongodb>  
  
[Execution complete with exit code 0]
```

Deleting:

a) Deleting one document:

CODE:

```
db.monuments.deleteOne(  
  { "name": "Arc de Triomphe de l'Étoile" }  
)
```

SCREENSHOT:

```
MongoDB ⓘ  
1 db.monuments.deleteOne(  
2   { "name": "Arc de Triomphe de l'Étoile" }  
3 )
```


OUTPUT:

Output

```
mycompiler_mongodb> ... .. { acknowledged: true, deletedCount: 0 }
mycompiler_mongodb>

[Execution complete with exit code 0]
```

b) Deleting many:

CODE:

```
db.monuments.deleteMany(
  { "editor": "Sammy" }
)
```

SCREENSHOT:



OUTPUT:

```
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb> ... .. { acknowledged: true, deletedCount: 6 }
mycompiler_mongodb>

[Execution complete with exit code 0]
```
