INSERT STATEMENT :

class code :

```
-- Use the database
USE employees;

-- Insert a new employee record into the employees table
INSERT INTO employees (emp_no, birth_date, first_name, last_name, gende
r, hire_date)
VALUES (2001, '1990-05-15', 'John', 'Doe', 'M', '2020-01-10');

-- Insert data into employees only if emp_no doesn't exist already
INSERT INTO employees (emp_no, birth_date, first_name, last_name, gende
r, hire_date)
SELECT 2001, '1990-05-15', 'John', 'Doe', 'M', '2020-01-10'
WHERE NOT EXISTS (
    SELECT 1
    FROM employees
    WHERE emp_no = 2001
);

-- Insert another employee record into the employees table
INSERT INTO employees (first_name, last_name, gender, emp_no, birth_dat
e, hire_date)
VALUES ('John', 'Doe', 'M', 2002, '1990-05-15', '2020-01-10');

-- Insert multiple employee records into the employees table
INSERT INTO employees (emp_no, birth_date, first_name, last_name, gende
r, hire_date)
VALUES
(2003, '1985-03-25', 'Alice', 'Smith', 'F', '2018-07-19'),
(2043, '1993-09-12', 'Bob', 'Johnson', 'M', '2019-11-30'),
(2005, '1997-01-04', 'Clara', 'Brown', 'F', '2021-06-01');
```

```
-- Delete specific employee records by emp_no
DELETE FROM employees
WHERE emp_no IN (2001, 2002, 2003);

-- Delete multiple records based on emp_no
DELETE FROM employees
WHERE emp_no IN (2001, 2002, 2003, 2043, 2005);

-- Insert a record into employees (Duplicate operation)
INSERT INTO employees
VALUES ('John', 'Doe', 'M', 2001, '1990-05-15', '2020-01-10');

-- Creating the employees1 table
CREATE TABLE employees1 (
    emp_no INT PRIMARY KEY,
    first_name VARCHAR(14),
    last_name VARCHAR(16),
    gender ENUM('M', 'F'),
    birth_date DATE,
    hire_date DATE
);

-- Creating the employees1_backup table
CREATE TABLE employees1_backup (
    emp_no INT PRIMARY KEY,
    first_name VARCHAR(14),
    last_name VARCHAR(16),
    gender ENUM('M', 'F'),
    birth_date DATE,
    hire_date DATE
);

-- Insert multiple records into employees1
INSERT INTO employees1 (emp_no, first_name, last_name, gender, birth_da
te, hire_date)
VALUES
(1001, 'John', 'Doe', 'M', '1990-05-15', '2020-01-10'),
(1002, 'Alice', 'Smith', 'F', '1985-03-25', '2018-07-19'),
(1003, 'Bob', 'Johnson', 'M', '1993-09-12', '2019-11-30'),
(1004, 'Emily', 'Davis', 'F', '1988-02-20', '2021-06-15'),
(1005, 'Michael', 'Williams', 'M', '1982-12-05', '2017-11-22');
```

```
-- Insert data into employees1_backup table, checking for duplicates ba
sed on emp_no
INSERT INTO employees1_backup (emp_no, birth_date, first_name, last_nam
e, gender, hire_date)
SELECT emp_no, birth_date, first_name, last_name, gender, hire_date
FROM employees1
WHERE NOT EXISTS (
    SELECT 1
    FROM employees1_backup
    WHERE employees1_backup.emp_no = employees1.emp_no
);

-- Drop columns from employees1 table
ALTER TABLE employees1
DROP COLUMN first_name,
DROP COLUMN last_name;

-- Drop multiple columns from employees1 table
ALTER TABLE employees1
DROP COLUMN first_name,
DROP COLUMN last_name,
DROP COLUMN gender,
DROP COLUMN hire_date;

-- Drop the employees1 table
DROP TABLE IF EXISTS employees1;

-- Drop both employees1 and employees1_backup tables
DROP TABLE IF EXISTS employees1, employees1_backup;

-- Display all records from the employees table
SELECT * FROM employees;
```

Syntax for INSERT INTO

```
INSERT INTO employees (emp_no, birth_date, first_name, last_name, gende
r, hire_date)
```

```
VALUES (value1, value2, 'value3', 'value4', 'value5', value6);
```

Example

Let's insert a record into your `employees` table:

```
INSERT INTO employees (emp_no, birth_date, first_name, last_name, gende
r, hire_date)
VALUES (1001, '1990-05-15', 'John', 'Doe', 'M', '2020-01-10');
```

IF Data already exists:

Method 1: Using `NOT EXISTS` to Prevent Duplicates

Alternatively, if you want to ensure there is no duplicate `emp_no` before inserting, you can use the `NOT EXISTS` clause:

```
INSERT INTO employees (emp_no, birth_date, first_name, last_name, gende
r, hire_date)
SELECT 2001, '1990-05-15', 'John', 'Doe', 'M', '2020-01-10'
WHERE NOT EXISTS (
    SELECT 1
    FROM employees
    WHERE emp_no = 2001
);
```

Explanation of `NOT EXISTS` Approach:

- The `SELECT 1` inside the `NOT EXISTS` clause checks whether there is already an employee with the same `emp_no = 2001` in the `employees` table.
- If the `emp_no = 2001` does not exist, it will insert the new data.
- If the `emp_no = 2001` already exists, the `NOT EXISTS` condition will prevent the insertion from happening, and no data will be added.

How to delete any row from the database:

Step 1: Delete the Record Using `emp_no`

Since you inserted a record with `emp_no = 2001`, you can delete it using the following SQL query:

```
sql
Copy code
DELETE FROM employees
WHERE emp_no = 2001;
```

This query will delete the record where the `emp_no` is `2001` from the `employees` table.

Step 2: Verify the Deletion

After running the `DELETE` command, you can verify that the record has been deleted by running:

```
sql
Copy code
SELECT * FROM employees
WHERE emp_no = 2001;
```

Inserting Multiple Records

To insert multiple records at once, you can use the following syntax:

```
sql
Copy code
INSERT INTO employees (emp_no, birth_date, first_name, last_name, gende
r, hire_date)
VALUES
(1002, '1985-03-25', 'Alice', 'Smith', 'F', '2018-07-19'),
(1003, '1993-09-12', 'Bob', 'Johnson', 'M', '2019-11-30'),
(1004, '1997-01-04', 'Clara', 'Brown', 'F', '2021-06-01');
```

Example: Default Order

For the table `employees`, the default column order is:

`(emp_no, birth_date, first_name, last_name, gender, hire_date)`

If you use:

```
INSERT INTO employees (emp_no, birth_date, first_name, last_name, gende
r, hire_date)
VALUES (1001, '1990-05-15', 'John', 'Doe', 'M', '2020-01-10');
```

- `emp_no` → `1001`
- `birth_date` → `'1990-05-15'`
- `first_name` → `'John'`
- `last_name` → `'Doe'`
- `gender` → `'M'`
- `hire_date` → `'2020-01-10'`

Changing the Order of Columns

If you change the order of the column names in the `INSERT INTO` statement, MySQL will store the values based on **your custom order**, not the table's default order.

Example: Changing the Order

```
INSERT INTO employees (first_name, last_name, gender, emp_no, birth_dat
e, hire_date)
VALUES ('John', 'Doe', 'M', 1001, '1990-05-15', '2020-01-10');
```

Here, the mapping will be:

- `first_name` → `'John'`
- `last_name` → `'Doe'`
- `gender` → `'M'`
- `emp_no` → `1001`
- `birth_date` → `'1990-05-1`
- `hire_date` → `'2020-01-10'`

This will insert the data correctly into the respective columns **even though the order of columns is different from the table's default order**.

Incorrect Mapping Example

If you provide values **without reordering the column names explicitly**, MySQL will assume the table's default column order. For example:

```
sql
Copy code
INSERT INTO employees
VALUES ('John', 'Doe', 'M', 1001, '1990-05-15', '2020-01-10');
```

Here, MySQL assumes:

- `emp_no` → `'John'` ❌
- `birth_date` → `'Doe'` ❌
- `first_name` → `'M'` ❌
- `last_name` → `1001` ❌
- `gender` → `'1990-05-15'` ❌
- `hire_date` → `'2020-01-10'` ✅

This will result in **errors** (data type mismatch) or **wrong data storage**.

DUPLICATE TABLES:

Step 1: Create the `employees` and `employees_backup` Tables

First, you need to create both the source table (`employees`) and the target table (`employees_backup`):

```
CREATE TABLE employees (
    emp_no INT PRIMARY KEY,
    first_name VARCHAR(14),
    last_name VARCHAR(16),
    gender ENUM('M', 'F'),
    birth_date DATE,
```

```
    hire_date DATE
);

CREATE TABLE employees_backup (
    emp_no INT PRIMARY KEY,
    first_name VARCHAR(14),
    last_name VARCHAR(16),
    gender ENUM('M', 'F'),
    birth_date DATE,
    hire_date DATE
);
```

Step 2: Insert Data into the `employees` Table

Now, we will insert some sample data into the `employees` table:

```
INSERT INTO employees (emp_no, first_name, last_name, gender, birth_dat
e, hire_date)
VALUES
(1001, 'John', 'Doe', 'M', '1990-05-15', '2020-01-10'),
(1002, 'Alice', 'Smith', 'F', '1985-03-25', '2018-07-19'),
(1003, 'Bob', 'Johnson', 'M', '1993-09-12', '2019-11-30'),
(1004, 'Emily', 'Davis', 'F', '1988-02-20', '2021-06-15'),
(1005, 'Michael', 'Williams', 'M', '1982-12-05', '2017-11-22');
```

The `employees` table now contains the following data:

emp_no	first_name	last_name	gender	birth_date	hire_date
1001	John	Doe	M	1990-05-15	2020-01-10
1002	Alice	Smith	F	1985-03-25	2018-07-19
1003	Bob	Johnson	M	1993-09-12	2019-11-30
1004	Emily	Davis	F	1988-02-20	2021-06-15
1005	Michael	Williams	M	1982-12-05	2017-11-22

Step 3: Insert Data into the `employees_backup` Table (Avoiding Duplicates)

Now, to insert data from `employees` into `employees_backup`, ensuring no duplicates based on `emp_no`, we use the following SQL query:

```
INSERT INTO employees_backup (emp_no, birth_date, first_name, last_nam
e, gender, hire_date)
SELECT emp_no, birth_date, first_name, last_name, gender, hire_date
FROM employees
WHERE NOT EXISTS (
    SELECT 1
    FROM employees_backup
    WHERE employees_backup.emp_no = employees.emp_no
);
```

This query will insert all records from the `employees` table into the `employees_backup` table, **but only if the `emp_no` does not already exist** in `employees_backup`.

Step 4: Verify the Data in `employees_backup`

After running the `INSERT INTO ... SELECT` query, the `employees_backup` table will have the same data as the `employees` table (since it was initially empty). To verify:

```
sql
Copy code
SELECT * FROM employees_backup;
```

This will output the following data:

emp_no	first_name	last_name	gender	birth_date	hire_date
1001	John	Doe	M	1990-05-15	2020-01-10
1002	Alice	Smith	F	1985-03-25	2018-07-19
1003	Bob	Johnson	M	1993-09-12	2019-11-30
1004	Emily	Davis	F	1988-02-20	2021-06-15
1005	Michael	Williams	M	1982-12-05	2017-11-22

Step 5: Insert a New Employee and Verify Duplicates Are Prevented

Now, if you try to insert the same employee again, it will not be inserted due to the `NOT EXISTS` condition.

```
sql
Copy code
```

```
INSERT INTO employees_backup (emp_no, birth_date, first_name, last_nam
e, gender, hire_date)
SELECT emp_no, birth_date, first_name, last_name, gender, hire_date
FROM employees
WHERE NOT EXISTS (
    SELECT 1
    FROM employees_backup
    WHERE employees_backup.emp_no = employees.emp_no
);
```

If you run the above query again, no data will be inserted because all `emp_no` values in `employees_backup` already exist. To confirm:

```
sql
Copy code
SELECT * FROM employees_backup;
```

It will still show the same data as before, with no duplicates:

emp_no	first_name	last_name	gender	birth_date	hire_date
1001	John	Doe	M	1990-05-15	2020-01-10
1002	Alice	Smith	F	1985-03-25	2018-07-19
1003	Bob	Johnson	M	1993-09-12	2019-11-30
1004	Emily	Davis	F	1988-02-20	2021-06-15
1005	Michael	Williams	M	1982-12-05	2017-11-22

Conclusion

The `INSERT INTO ... SELECT` with `NOT EXISTS` ensures that no duplicate records are inserted into the `employees_backup` table. This is useful for preventing duplication when transferring data from one table to another.

If you'd like further clarification or an extension of this example, feel free to ask!