DATABASE MANAGEMENT SYSTEMS

(MS SQL)

CHAPTER - 3

Sub Queries

Karuvarai
INSPIRE. INCUBATE. INNOVATE.

MySQL Subquery:

- A MySQL subquery is a query nested within another query such as SELECT, INSERT, UPDATE or DELETE. Also, a subquery can be nested within another subquery.
- A MySQL subquery is called an inner query while the query that contains the subquery is called an outer query.
- A subquery can be used anywhere that expression is used and must be closed in **parentheses**.

MySQL Subquery Syntax:

```
SELECT column_list (s) FROM  table_name
WHERE  column_name OPERATOR
  (SELECT column_list (s)  FROM table_name [WHERE])
```

For example, the following query uses a subquery to return the employees who work in the offices located in the USA.

```
SELECT   lastName, firstName
FROM   employees
WHERE  officeCode IN
(SELECT  officeCode  FROM  offices WHERE   country = 'USA');
```

Karuvarai
INSPIRE. INCUBATE. INNOVATE.

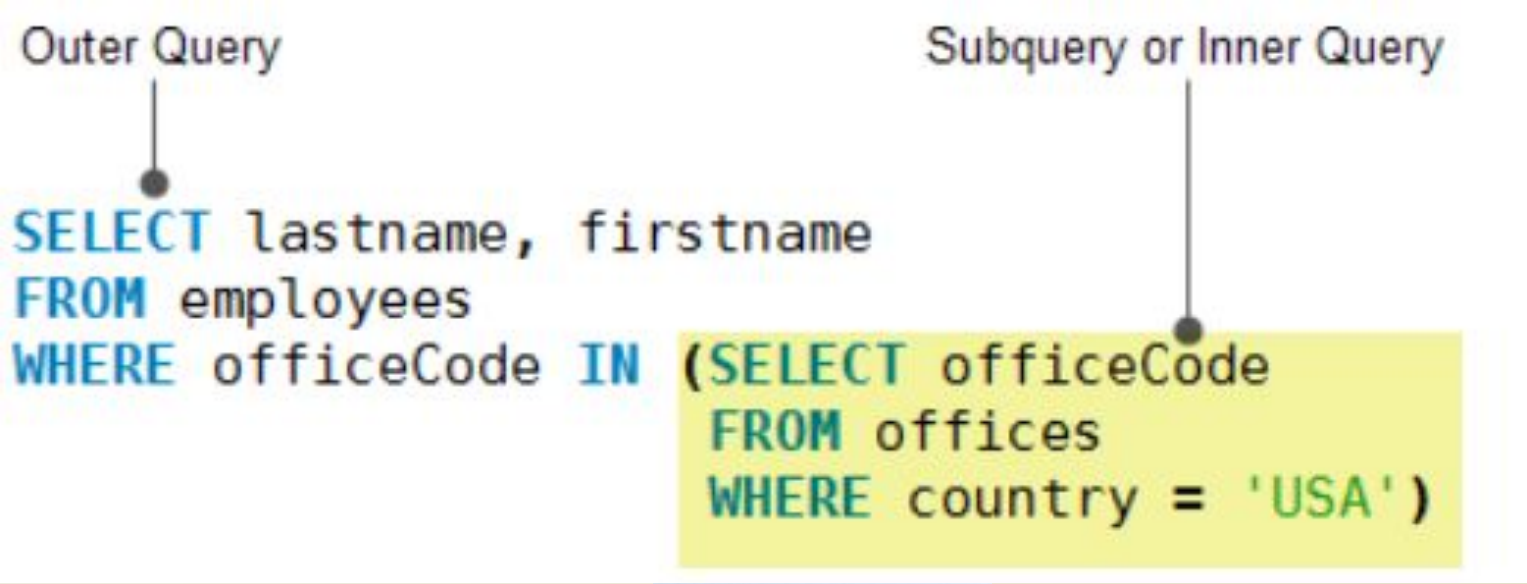
MySQL Subquery:

In this example:

The subquery returns all office codes of the offices located in the USA.

The outer query selects the last name and first name of employees who work in the offices whose office codes are in the result set returned by the subquery.

When executing the query, MySQL evaluates the subquery first and uses the result of the subquery for the outer query.



Using a MySQL subquery in the WHERE clause

MySQL subquery with comparison operators

You can use comparison operators e.g., =, >, < to compare a single value returned by the subquery with the expression in the WHERE clause.

For example, the following query returns the customer who has the highest payment.

```
SELECT   customerNumber, checkNumber, amount
FROM  payments
WHERE  amount = (SELECT MAX(amount) FROM payments);
```

Besides the = operator, you can use other comparison operators such as greater than (>), greater than or equal to (>=) less than(<), and less than or equal to (<=).

For example, you can find customers whose payments are greater than the average payment using a subquery:

```
SELECT  customerNumber,  checkNumber, amount
FROM payments
WHERE amount > (SELECT  AVG(amount) FROM  payments);
```

In this example:

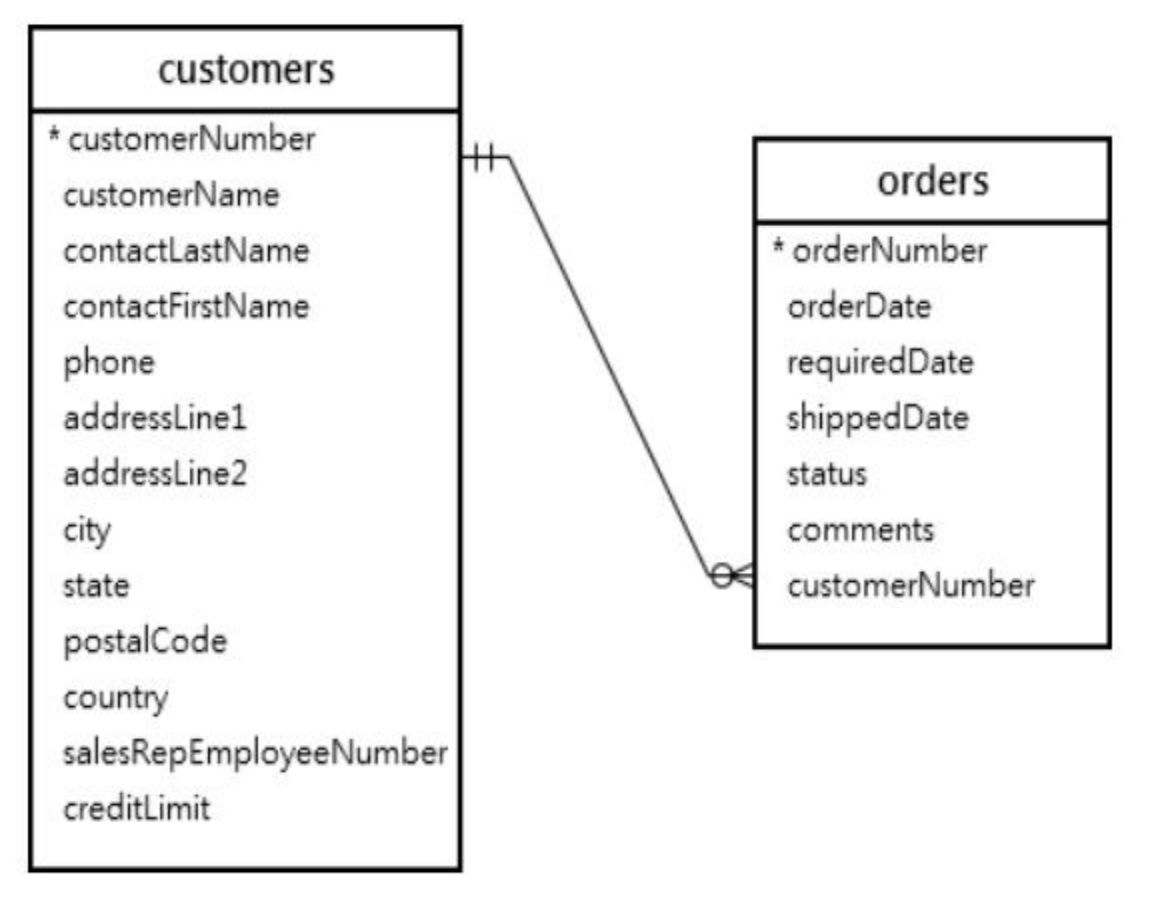
First, get the average payment by using a subquery.

Then, select the payments that are greater than the average payment returned by the subquery in the outer query

MySQL subquery with IN and NOT IN operators

If a subquery returns more than one value, you can use other operators such as IN or NOT IN operator in the WHERE clause.

See the following customers and orders tables:



For example, you can use a subquery with NOT IN operator to find the customers who have not placed any orders as follows:

```
SELECT customerName
FROM customers
WHERE customerNumber NOT IN (SELECT DISTINCT customerNumber FROM orders);
```

MySQL subquery in the FROM clause

When you use a subquery in the FROM clause, the result set returned from a subquery is used as a temporary table. This table is referred to as a derived table or materialized subquery.

The following subquery finds the maximum, minimum, and average number of items in sale orders:

```
SELECT MAX(items), MIN(items), FLOOR(AVG(items))
FROM
    (SELECT  orderNumber, COUNT(orderNumber) AS items
    FROM orderdetails
    GROUP BY orderNumber) AS lineitems;
```

Karuvarai
INSPIRE. INCUBATE. INNOVATE.
www.siet.ac.in

Example:

emp_id	emp_name	emp_age	city	income
101	Peter	32	Newyork	200000
102	Mark	32	California	300000
103	Donald	40	Arizona	1000000
104	Obama	35	Florida	5000000
105	Linklon	32	Georgia	250000
106	Kane	45	Alaska	450000
107	Adam	35	California	5000000
108	Macculam	40	Florida	350000
109	Brayan	32	Alaska	400000
110	Stephen	40	Arizona	600000
111	Alexander	45	California	70000

Karuvarai
INSPIRE. INCUBATE. INNOVATE.
www.siet.ac.in

Example:

Table: Student

Stud_ID	Name	Email	City
1	Peter	peter@javatpoint.com	Texas
2	Suzi	suzi@javatpoint.com	California
3	Joseph	joseph@javatpoint.com	Alaska
4	Andrew	andrew@javatpoint.com	Los Angeles
5	Brayan	brayan@javatpoint.com	New York

Stud_ID	Name	Email	City
1	Stephen	stephen@javatpoint.com	Texas
2	Joseph	joseph@javatpoint.com	Los Angeles
3	Peter	peter@javatpoint.com	California
4	David	david@javatpoint.com	New York
5	Maddy	maddy@javatpoint.com	Los Angeles

Example:

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM customer;
+---------+---------+------------+------+
| cust_id | name    | occupation | age  |
+---------+---------+------------+------+
|     101 | Peter   | Engineer   |   32 |
|     102 | Joseph  | Developer  |   30 |
|     103 | John    | Leader     |   28 |
|     104 | Stephen | Scientist  |   45 |
|     105 | Suzi    | Carpenter  |   26 |
|     106 | Bob     | Actor      |   25 |
|     107 | NULL    | NULL       | NULL |
+---------+---------+------------+------+
7 rows in set (0.00 sec)

mysql> SELECT * FROM Orders;
+----------+---------+-----------+------------+
| order_id | cust_id | prod_name | order_date |
+----------+---------+-----------+------------+
|        1 |     101 | Laptop    | 2020-01-10 |
|        2 |     103 | Desktop   | 2020-02-12 |
|        3 |     106 | Iphone    | 2020-02-15 |
|        4 |     104 | Mobile    | 2020-03-05 |
|        5 |     102 | TV        | 2020-03-20 |
+----------+---------+-----------+------------+
```

Joins:

- A join is a method of linking data between one (self-join) or more tables based on values of the common column between the tables.

MySQL supports the following types of joins:

- Inner join
- Left join
- Right join
- Full join

The join clause is used in the SELECT statement appeared after the FROM clause.

member_id	name
1	John
2	Jane
3	Mary
4	David
5	Amelia

committee_id	name
1	John
2	Mary
3	Amelia
4	Joe

Karuvarai
INSPIRE. INCUBATE. INNOVATE.

MySQL INNER JOIN clause

The following shows the basic syntax of the inner join clause that joins two tables table_1 and table_2:

```
SELECT column_list
FROM table_1
INNER JOIN table_2 ON join_condition;
```

The inner join clause joins two tables based on a condition which is known as a join predicate.

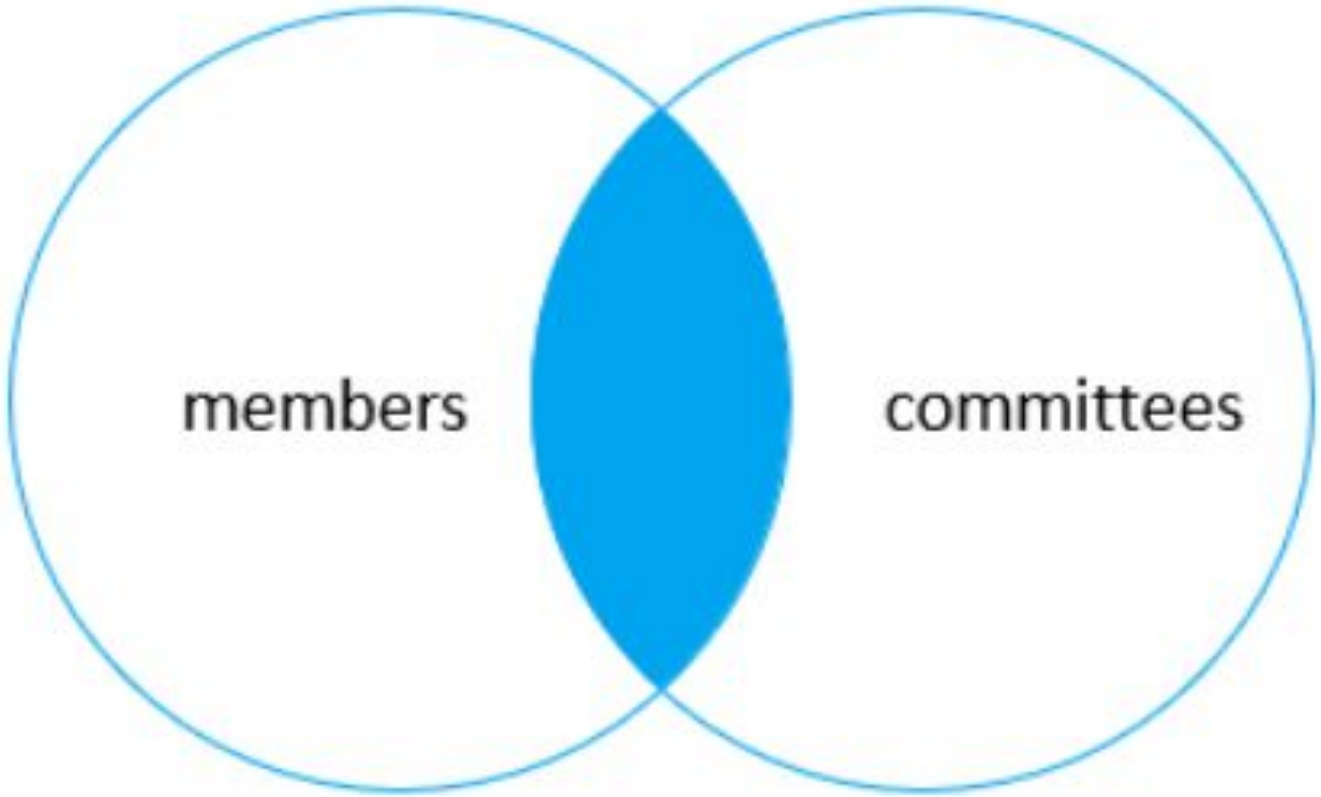
The inner join clause compares each row from the first table with every row from the second table.

If values from both rows satisfy the join condition, the inner join clause creates a new row whose column contains all columns of the two rows from both tables and includes this new row in the result set.

In other words, the inner join clause **includes only matching rows from both tables.**

Karuvarai
INSPIRE. INCUBATE. INNOVATE.
www.siet.ac.in

The following Venn diagram illustrates the inner join:



Karuvarai
INSPIRE. INCUBATE. INNOVATE.

The following statement uses an inner join clause to find members who are also the committee members:

```
SELECT
    m.member_id,
    m.name AS member,
    c.committee_id,
    c.name AS committee
FROM  members m
INNER JOIN committees c ON c.name = m.name;
```

member_id	member	committee_id	committee
1	John	1	John
3	Mary	2	Mary
5	Amelia	3	Amelia

Karuvarai
INSPIRE. INCUBATE. INNOVATE.

If both tables use the same column to match, you can use the USING clause as shown in the following query:

```
SELECT
    m.member_id,
    m.name AS member,
    c.committee_id,
    c.name AS committee
FROM
    members m
INNER JOIN committees c USING(name);
```

MySQL LEFT JOIN clause

For each row in the left table, the left join compares with every row in the right table.

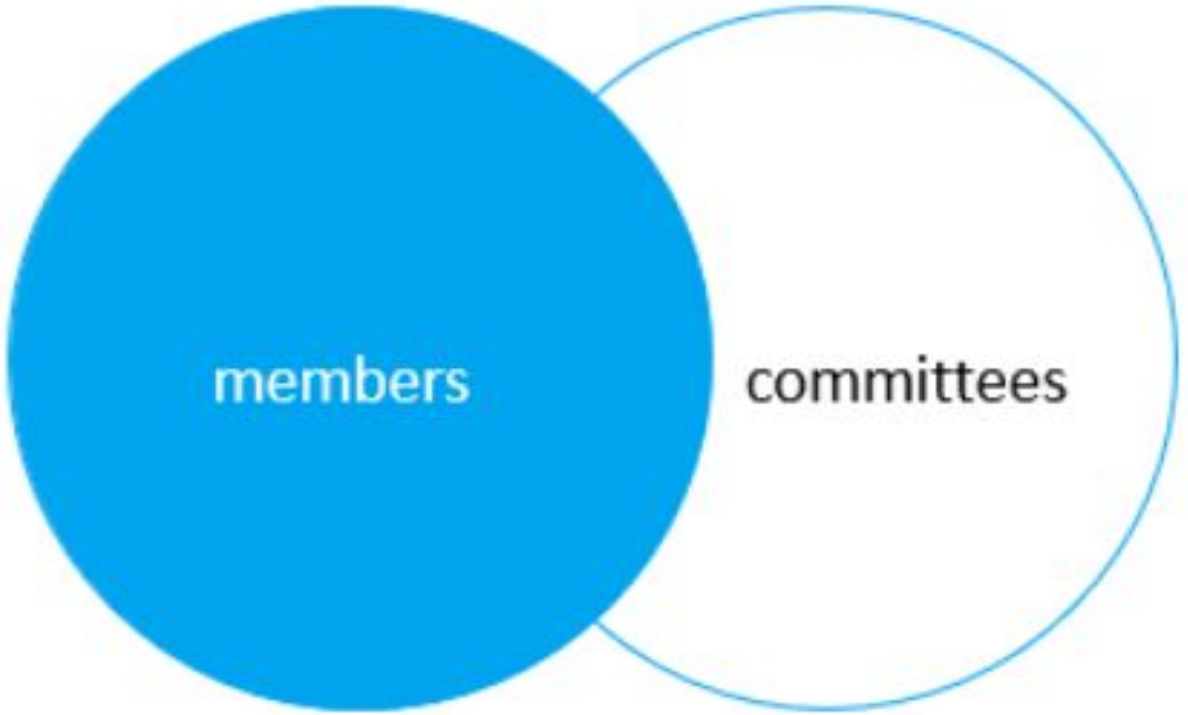
If the values in the two rows satisfy the join condition, the left join clause creates a new row whose columns contain all columns of the rows in both tables and includes this row in the result set.

If the values in the two rows are not matched, the left join clause still creates a new row whose columns contain columns of the row in the left table and NULL for columns of the row in the right table.

In other words, **the left join selects all data from the left table whether there are matching rows exist in the right table or not.**

In case there are no matching rows from the right table found, t**he left join uses NULLs for columns of the row from the right table in the result set.**

The following Venn diagram illustrates the left join:



Karuvarai
INSPIRE. INCUBATE. INNOVATE.

```
SELECT column_list
FROM table_1
LEFT JOIN table_2 ON join_condition;

SELECT column_list
FROM table_1
LEFT JOIN table_2 USING (column_name);

SELECT
    m.member_id,
    m.name AS member,
    c.committee_id,
    c.name AS committee
FROM
    members m
LEFT JOIN committees c USING(name);
```

member_id	member	committee_id	committee
1	John	1	John
2	Jane	NULL	NULL
3	Mary	2	Mary
4	David	NULL	NULL
5	Amelia	3	Amelia

Karuvarai
INSPIRE. INCUBATE. INNOVATE.
www.siet.ac.in

To find members who are not the committee members, you add a WHERE clause and IS NULL operator as follows:

```
SELECT
    m.member_id,
    m.name AS member,
    c.committee_id,
    c.name AS committee
FROM
    members m
LEFT JOIN committees c USING(name)
WHERE c.committee_id IS NULL;
```

member_id	member	committee_id	committee
2	Jane	NULL	NULL
4	David	NULL	NULL

Karuvarai
INSPIRE. INCUBATE. INNOVATE.

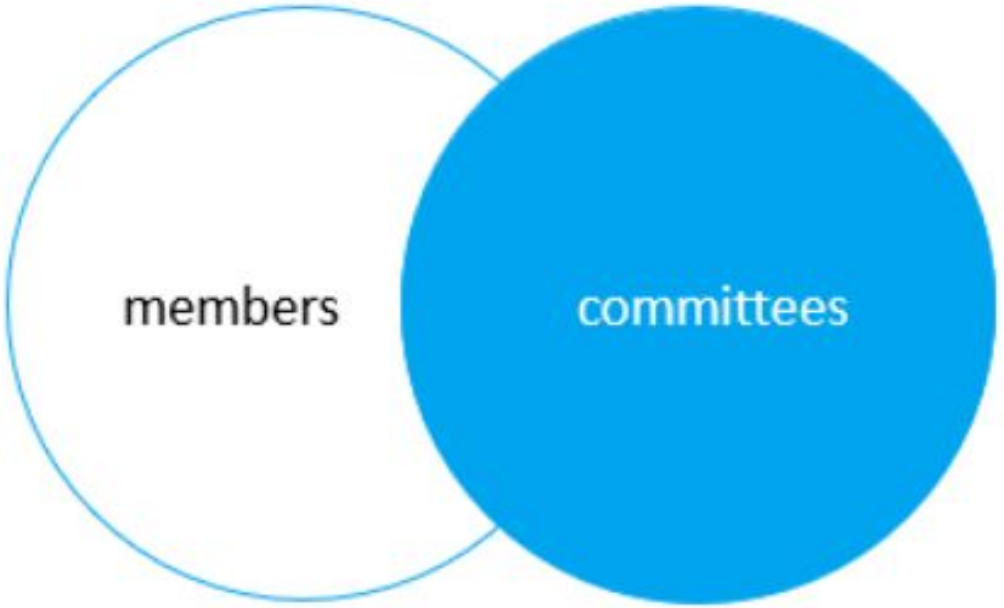
MySQL RIGHT JOIN clause

The right join clause is similar to the left join clause except that the treatment of left and right tables is reversed. The right join starts selecting data from the right table instead of the left table.

The right join clause selects all rows from the right table and matches rows in the left table. If a row from the right table does not have matching rows from the left table, the column of the left table will have NULL in the final result set.

```
SELECT column_list
FROM table_1
RIGHT JOIN table_2 ON join_condition;
```

This Venn diagram illustrates the right join:



www.siet.ac.in

```
SELECT column_list
FROM table_1
RIGHT JOIN table_2 USING (column_name);
```

To find rows in the right table that does not have corresponding rows in the left table, you also use a WHERE clause with the IS NULL operator:

```
SELECT column_list
FROM table_1
RIGHT JOIN table_2 USING (column_name)
WHERE column_table_1 IS NULL;
```

```
SELECT
    m.member_id,
    m.name AS member,
    c.committee_id,
    c.name AS committee
FROM
    members m
RIGHT JOIN committees c on c.name = m.name;
```

member_id	member	committee_id	committee
1	John	1	John
3	Mary	2	Mary
5	Amelia	3	Amelia
NULL	NULL	4	Joe

To find the committee members who are not in the members table, you use this query:

```
SELECT
    m.member_id,
    m.name AS member,
    c.committee_id,
    c.name AS committee
FROM
    members m
RIGHT JOIN committees c USING(name)
WHERE m.member_id IS NULL;
```

What is an Index in MySQL?

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns.

The users cannot see the indexes, they are just used to speed up searches/queries.

CREATE INDEX Syntax

Creates an index on a table. Duplicate values are allowed:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

CREATE UNIQUE INDEX Syntax

Creates a unique index on a table. Duplicate values are not allowed:

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

The SQL statement below creates an index named "idx_lastname" on the "LastName" column in the "Persons" table:

```
CREATE INDEX idx_lastname
ON Persons (LastName);
```

If you want to create an index on a combination of columns, you can list the column names within the parentheses, separated by commas:

```
CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

DROP INDEX Statement

The DROP INDEX statement is used to delete an index in a table.

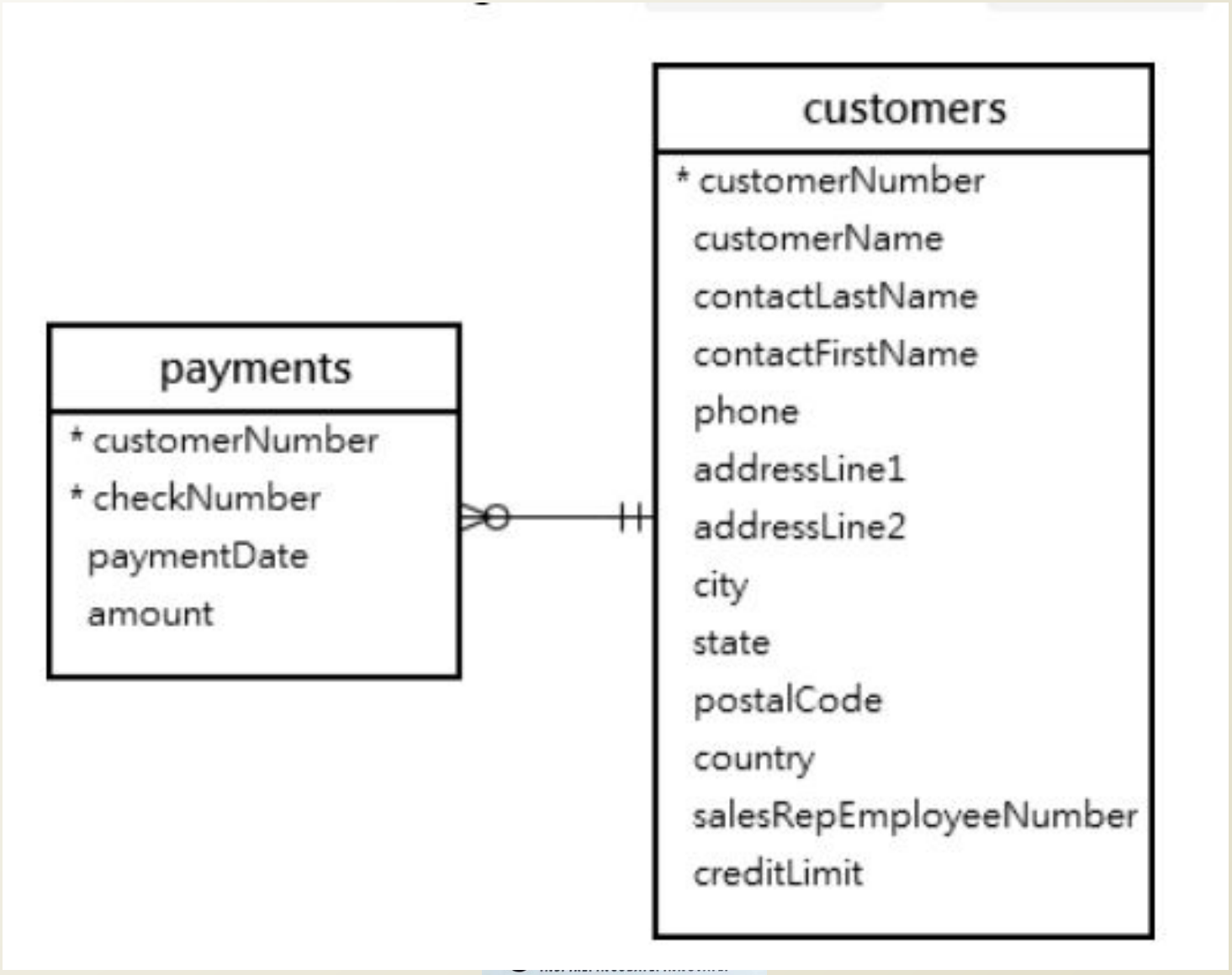
```
ALTER TABLE table_name
DROP INDEX index_name;
```

To query the index information of a table, you use the SHOW INDEXES statement as follows:

```
SHOW INDEXES FROM table_name;
```

Views:

Let’s see the following tables customers and payments



Views:

```
SELECT
    customerName,
    checkNumber,
    paymentDate,
    amount
FROM
    customers
INNER JOIN
    payments USING (customerNumber);
```

customerName	checkNumber	paymentDate	amount
Atelier graphique	HQ336336	2004-10-19	6066.78
Atelier graphique	JM555205	2003-06-05	14571.44
Atelier graphique	OM314933	2004-12-18	1676.14
Signal Gift Stores	BO864823	2004-12-17	14191.12
Signal Gift Stores	HQ55022	2003-06-06	32641.98
Signal Gift Stores	ND748579	2004-08-20	33347.88
Australian Collectors, Co.	GG31455	2003-05-20	45864.03
Australian Collectors, Co.	MA765515	2004-12-15	82261.22
Australian Collectors, Co.	NP603840	2003-05-31	7565.08
Australian Collectors, Co.	NR27552	2004-03-10	44894.74
La Rochelle Gifts	DB933704	2004-11-14	19501.82
La Rochelle Gifts	LN373447	2004-08-08	47924.19

Karuvarai
INSPIRE. INCUBATE. INNOVATE.
www.siet.ac.in

What are Views in MySQL?

VIEWS are virtual tables that do not store any data of their own but display data stored in other tables.

In other words, VIEWS are nothing but SQL Queries.

A view can contain all or a few rows from a table. A MySQL view can show data from one table or many tables.

CREATE VIEW view_name AS SELECT statement;

CREATE VIEW view_name tells MySQL server to create a view object in the database named view_name

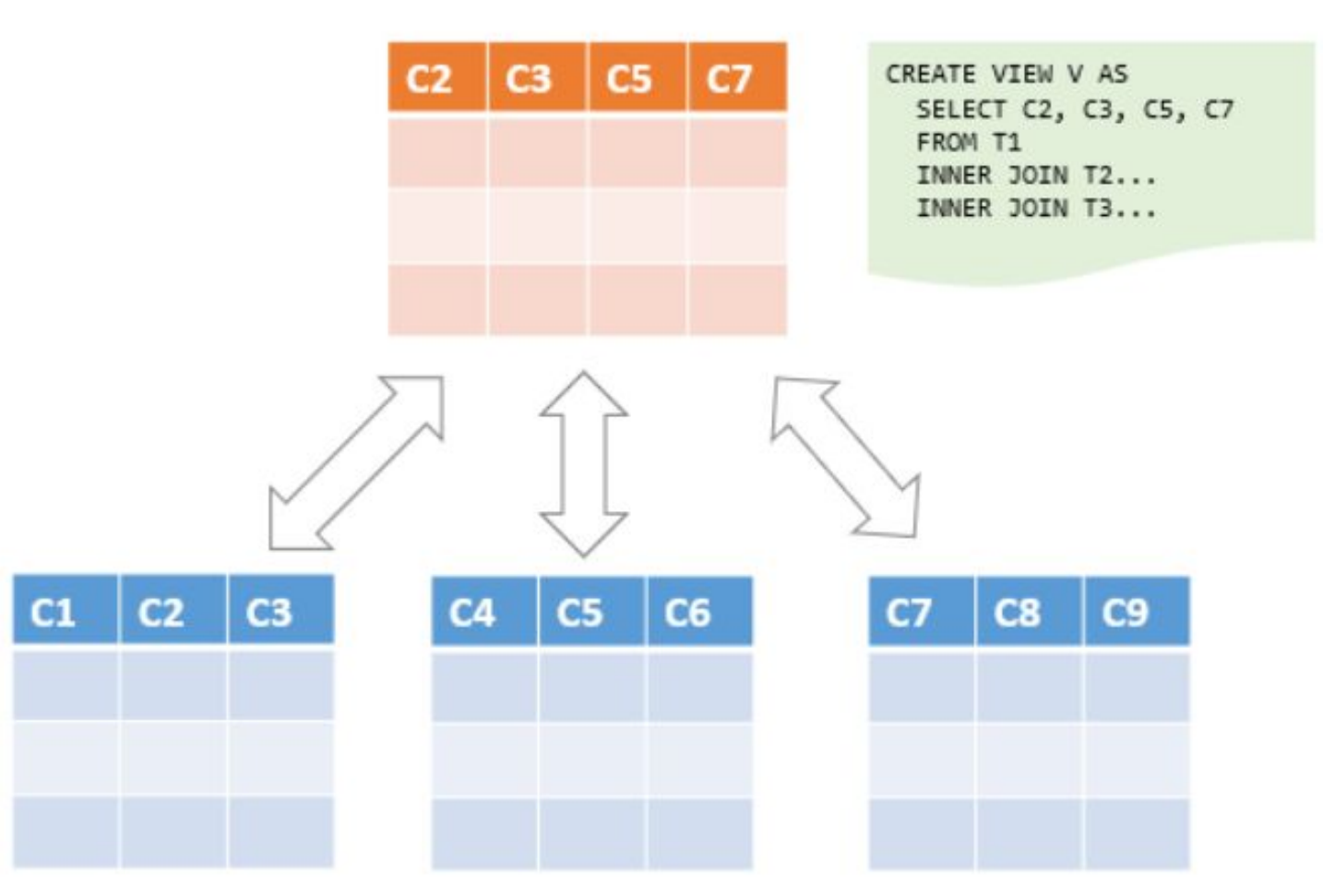
"AS SELECT statement" is the SQL statements to be packed in the MySQL Views. It can be a SELECT statement can contain data from one table or multiple tables.

```
CREATE VIEW customerPayments
AS
SELECT
    customerName,
    checkNumber,
    paymentDate,
    amount
FROM
    customers
INNER JOIN
    payments USING (customerNumber);


SELECT * FROM customerPayments;
```

Karuvarai
INSPIRE. INCUBATE. INNOVATE.

MySQL allows you to create a view based on a SELECT statement that retrieves data from one or more tables. This picture illustrates a view based on columns of multiple tables:



Dropping Views in MySQL

The DROP command can be used to delete a view from the database that is no longer required. The basic syntax to drop a view is as follows.

DROP VIEW View_Name;