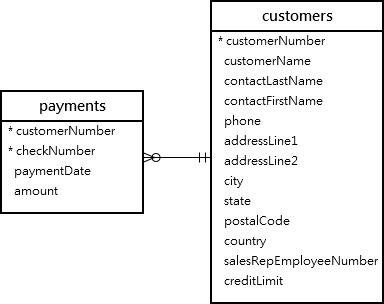
**MySQL Views**

1. In SQL, a view is a virtual table based on the result-set of an SQL statement.
2. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
3. Let’s see the following tables customers and payments from the [sample database](https://www.mysqltutorial.org/mysql-sample-database.aspx).



SELECT

customerName,

checkNumber,

paymentDate,

amount

FROM

customers

INNER JOIN

payments USING (customerNumber);

Next time, if you want to get the same information including customer name, check number, payment date, and amount, you need to issue the same query again.

A better way to do this is to save the query in the database server and assign a name to it. This named query is called a **database view,** or simply, **view**.

So, by definition we can say that a view is a named query stored in the database catalog.

To create a new view you use the [CREATE VIEW](https://www.mysqltutorial.org/create-sql-views-mysql.aspx) statement. This statement creates a view customerPayments based on the above query above:

CREATE VIEW customerPayments

AS

SELECT

customerName,

checkNumber,

paymentDate,

amount

FROM

customers

INNER JOIN

payments USING (customerNumber);

Once you execute the CREATE VIEW statement, MySQL creates the view and stores it in the database.

Now, you can reference the view as a table in SQL statements. For example, you can query data from the customerPayments view using the SELECT statement:

select \* from customerPayments;

In addition, MySQL even allows you to create a view that does not refer to any table.

For example, you can create a view called daysofweek that return 7 days of a week by executing the following query:

CREATE VIEW daysofweek (day) AS

SELECT 'Mon'

UNION

SELECT 'Tue'

UNION

SELECT 'Web'

UNION

SELECT 'Thu'

UNION

SELECT 'Fri'

UNION

SELECT 'Sat'

UNION

SELECT 'Sun';

And you can query data from the daysofweek view as follows:

SELECT \* FROM daysofweek;

**Some Practice questions:**

1. **Create a view that represents total sales per order.**

CREATE VIEW salePerOrder AS

SELECT

orderNumber,

SUM(quantityOrdered \* priceEach) total

FROM

orderDetails

GROUP by orderNumber

ORDER BY total DESC;

SELECT \* FROM salePerOrder;

### Creating a view with join example

CREATE OR REPLACE VIEW customerOrders AS

SELECT

orderNumber,

customerName,

SUM(quantityOrdered \* priceEach) total

FROM orderDetails

INNER JOIN orders o USING (orderNumber)

INNER JOIN customers USING (customerNumber)

GROUP BY orderNumber;

SELECT \* FROM customerOrders;

### Creating a view with a subquery example

Create a view that contains products whose buy prices are higher than the average price of all products.

CREATE VIEW aboveAvgProducts AS

SELECT

productCode,

productName,

buyPrice

FROM

products

WHERE

buyPrice > (

SELECT

AVG(buyPrice)

FROM

products)

ORDER BY buyPrice DESC;

SELECT \* FROM aboveAvgProducts;

### Creating a view with explicit view columns example

### Create a new view based on the customers and orders tables with explicit view columns:

CREATE VIEW customerOrderStats (

customerName ,

orderCount

)

AS

SELECT

customerName,

COUNT(orderNumber)

FROM

customers

INNER JOIN

orders USING (customerNumber)

GROUP BY customerName;

SELECT

customerName,

orderCount

FROM

customerOrderStats

ORDER BY orderCount, customerName;

**Difference between View and stored Procedure**

