Working with Views

Learning Objectives

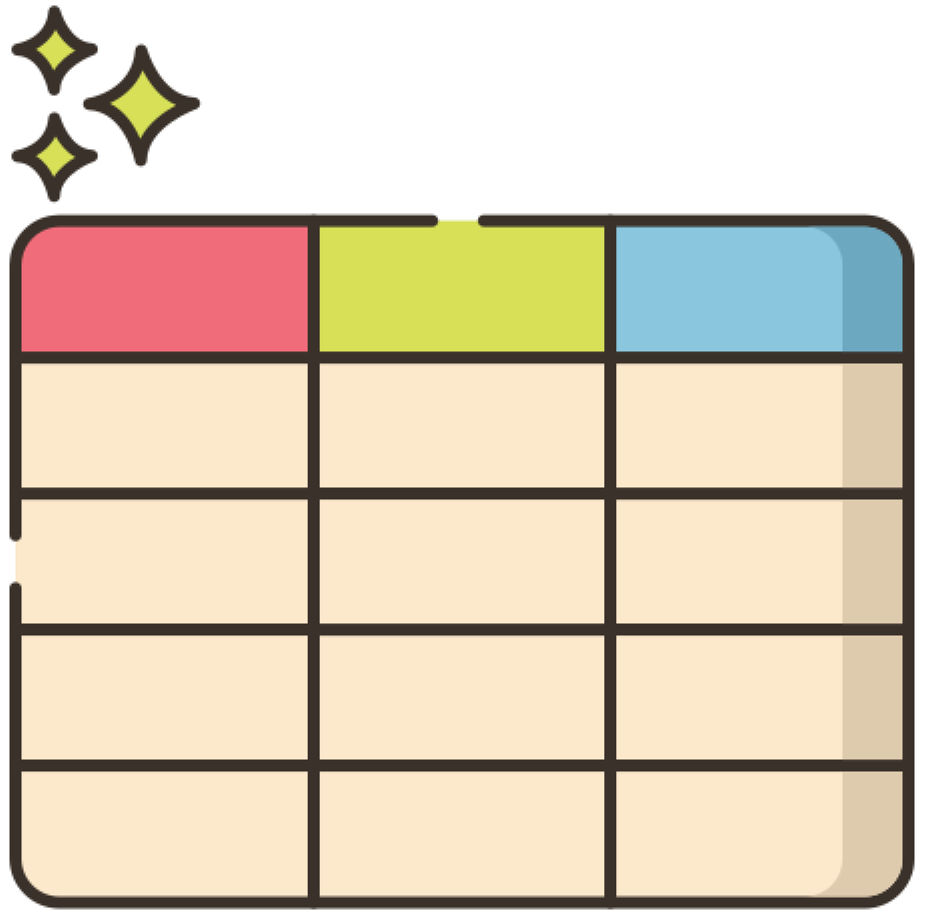
By the end of this lesson, you will be able to:

- Illustrate SQL views
- List the view manipulation methods
- Utilize the view process algorithms
- Implement CHECK constraints on views

SQL Views

SQL Views

In SQL, a view refers to a virtual table. It can be created by selecting fields from one or more tables.



SYNTAX

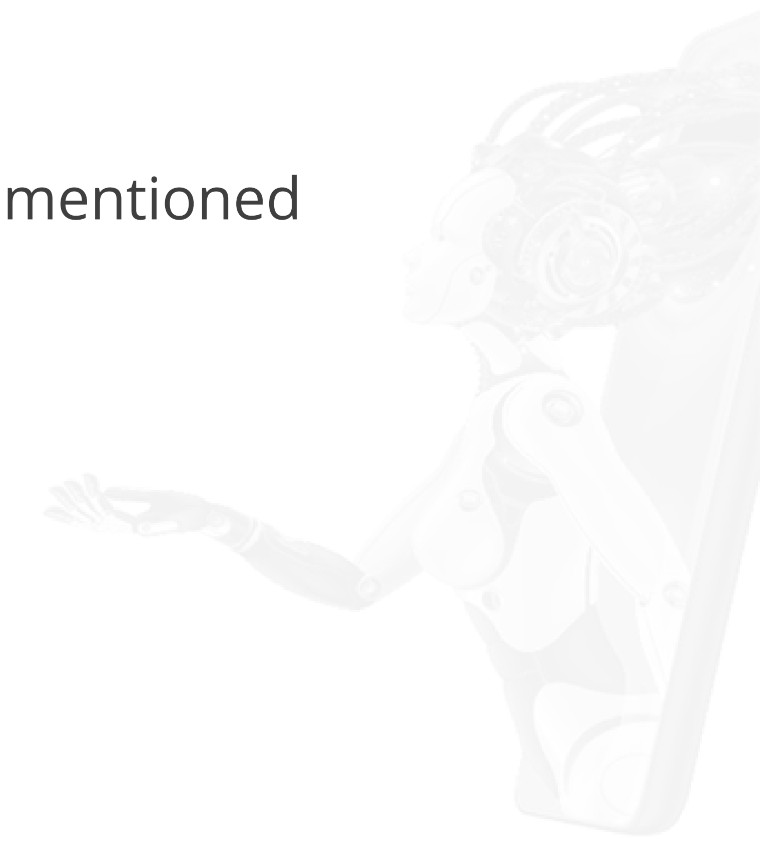
```
CREATE VIEW view_name AS
SELECT column1, column2, column3,….
FROM table_name
WHERE condition…
```

View Manipulation Methods

Creating a View From a Single Table: Example

Problem Scenario: You are a data analyst in your company, and you are asked to create a temporary table of employees with salary more than 22000.

Objective: Use the view command to create a temporary table for the condition mentioned above.



Creating a View From a Single Table: Example

Step 1: Create a table named **employee records** with the following data:

Emp_ID	Emp_F_Name	Emp_L_Name	Emp_Salary	Emp_Location
1134	Mark	Jacobs	20000	New York
1256	John	Barter	25000	California
1277	Michael	Scar	22000	San Francisco
1300	Dan	Harris	30000	Texas

Creating a View From a Single Table: Example

Step 2: Use the following view syntax to create a temporary table of employees with salary more than 22000.

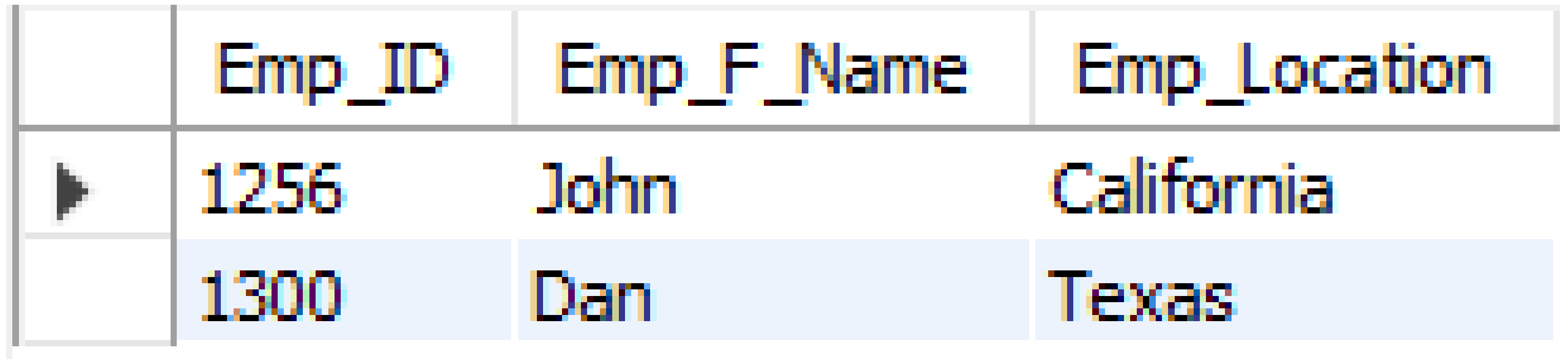
QUERY

```
CREATE VIEW Employee_View AS
SELECT Emp_ID, Emp_F_Name, Emp_Location
FROM Employee_Records
WHERE Emp_Salary>22000;
```

Creating a View From a Single Table: Example

To display the created view, enter:
SELECT * FROM Employee_Details;

Output:



	Emp_ID	Emp_F_Name	Emp_Location
▶	1256	John	California
	1300	Dan	Texas

Deleting or Dropping a View

DROP statement allows you to delete or drop a created view.

SYNTAX

```
DROP VIEW view_name;
```

Updating or Modifying a View

CREATE OR REPLACE VIEW statement allows you to update a created view.

SYNTAX

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2,...
FROM table_name
WHERE condition;
```

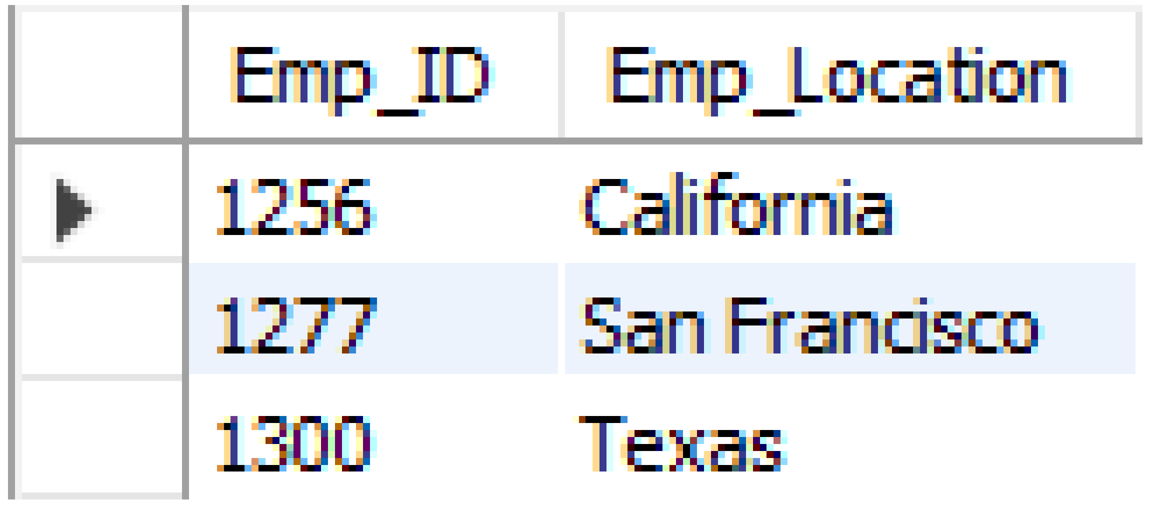
Updating or Modifying a View: Example

Suppose you want to display different columns in the same view name created earlier, then you can use the replace view command.

QUERY

```
CREATE OR REPLACE VIEW Emp_VIew AS
SELECT Emp_ID, Emp_Location
FROM Employee_Records
WHERE Emp_Salary > 20000;
```

Output:



	Emp_ID	Emp_Location
▶	1256	California
	1277	San Francisco
	1300	Texas

Altering a View

ALTER VIEW command allows you to change the SQL statements present in a view.

SYNTAX

```
ALTER VIEW view_name AS
SELECT column1, column 2
FROM table_name
WHERE condition;
```

ALTER VIEW is similar to **replace view** seen in the earlier slides.

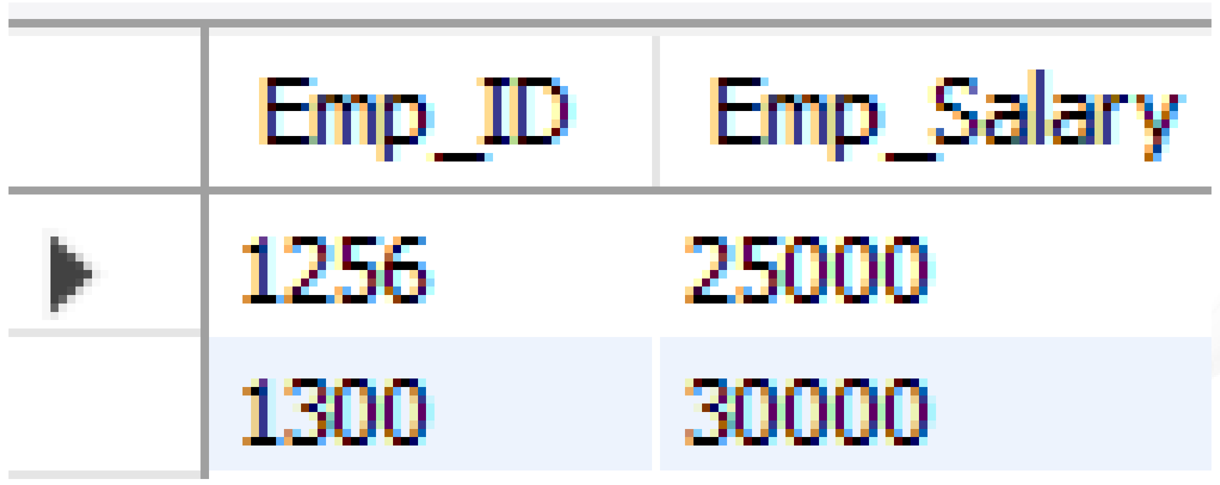
Altering a View: Example

If you want to change the columns of a created Emp_View, then you use the alter command.

QUERY

```
ALTER VIEW Emp_View AS
SELECT Emp_ID, Emp_Salary
FROM Emp_Records
WHERE Emp_Salary>22000;
```

Output:



	Emp_ID	Emp_Salary
▶	1256	25000
	1300	30000

Renaming a View

Renaming tables and views in MySQL use the same namespace. Therefore, you can use the RENAME_TABLE statement to rename a view.

SYNTAX

```
RENAME TABLE present_view_name TO
new_view_name;
```

Renaming a View: Example

Consider the same employee records table and its view **Emp view.** If you want to rename the view name that focuses more on the salary aspect, you can use the following syntax:

QUERY

```
RENAME TABLE Emp_view TO Emp_Salary_View;
```

Now, the output remains the same as the earlier view, but the view name must be changed to view the results.

Replacing a View

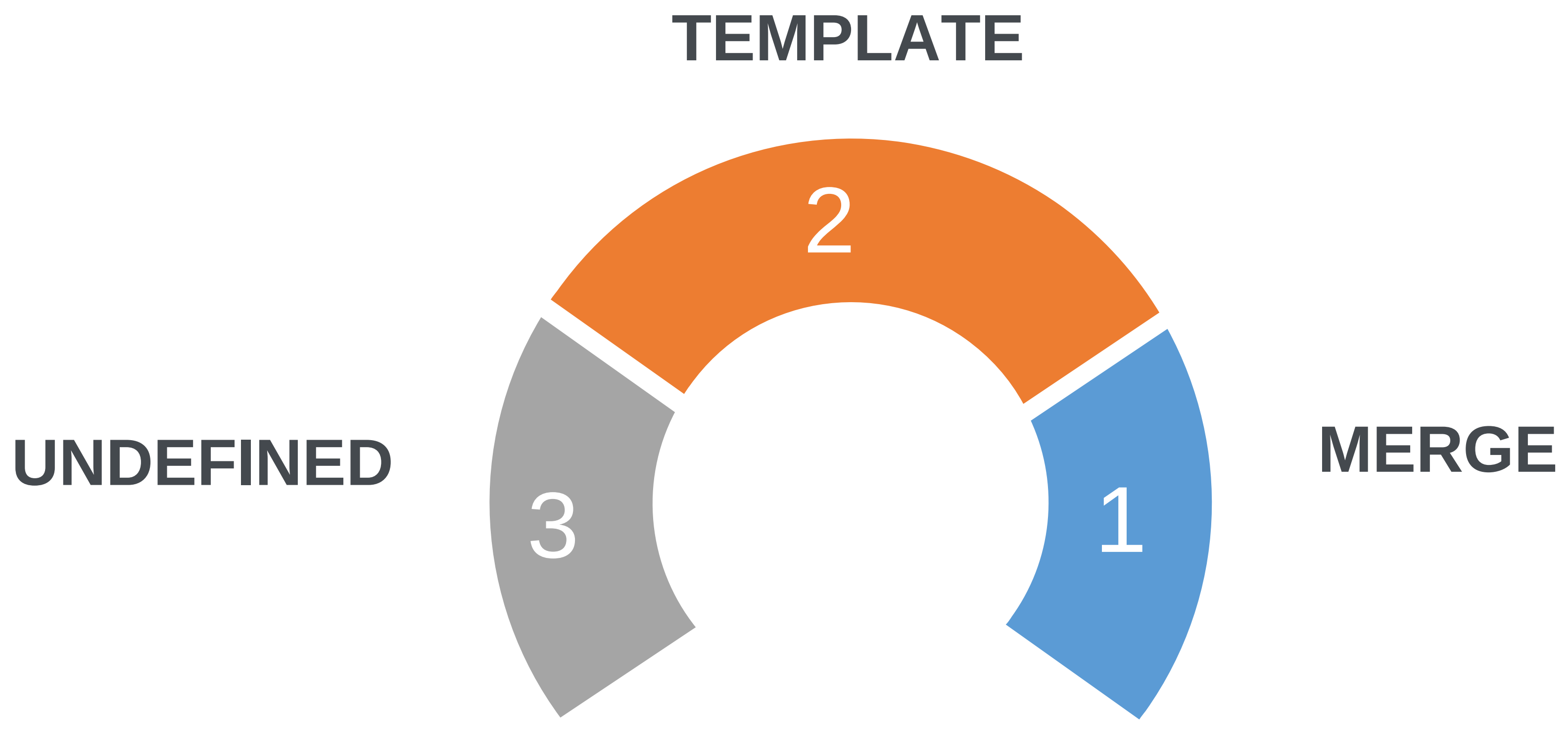
REPLACE VIEW allows you to replace an existing view with a newly specified view.

SYNTAX

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2,...
FROM table_name
WHERE condition;
```

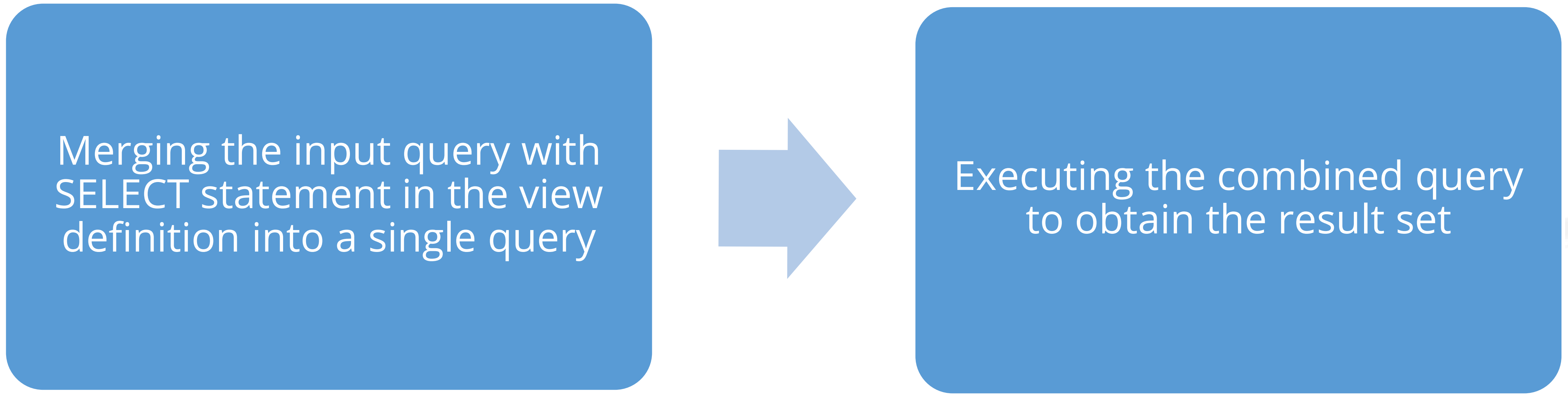
View Processing Algorithms

In CREATE VIEW command, there is an optional clause called ALGORITHM. This specifies how the view must be processed.



Workflow of View Processing Algorithms

In MERGE, the text of a statement is merged in such a way that parts of the view definition replace the corresponding parts of the statement.


Merging the input query with SELECT statement in the view definition into a single query
Executing the combined query to obtain the result set

Workflow of View Processing Algorithms (Merge): Example

Problem Statement: You are the junior DB administrator in your organization, and your manager has asked you to change the column headers of the employee table and create a new **view**.

Objective: Create a **view** with the merge algorithm and specify the column headers with the changes.

Workflow of View Processing Algorithms (Merge): Example

Step 1: Create a **view** with the algorithm as merge. Include column headers that must be merged or changed in the **view**.

QUERY

```
CREATE ALGORITHM = MERGE VIEW E_VIEW (Emp_Name,
R_name, D_name, Ex_p)
AS
SELECT Emp_Name, Role_name, Dept, Experience
FROMEmp_Table;
```

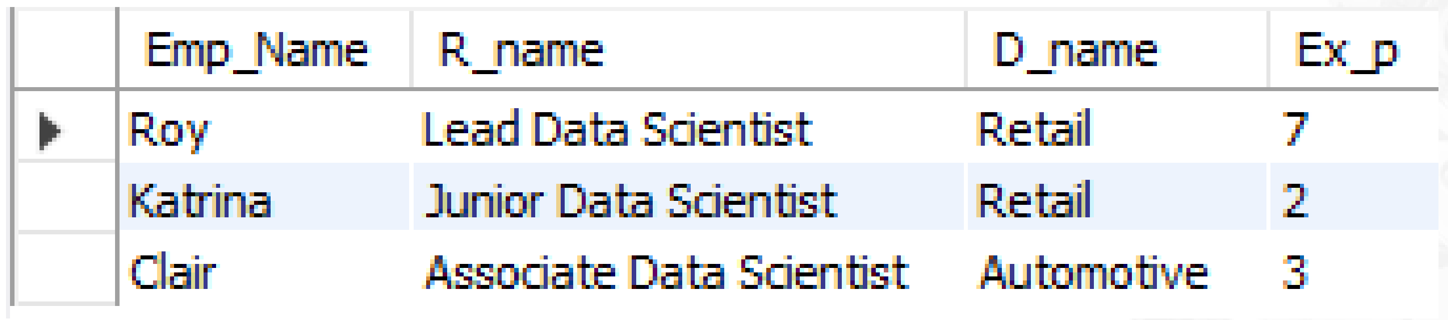
Workflow of View Processing Algorithms (Merge): Example

Step 2: Use the select command to extract the data that satisfies the given condition.

QUERY

```
SELECT * FROM E_VIEW WHERE Emp_Name LIKE '%R%';
```

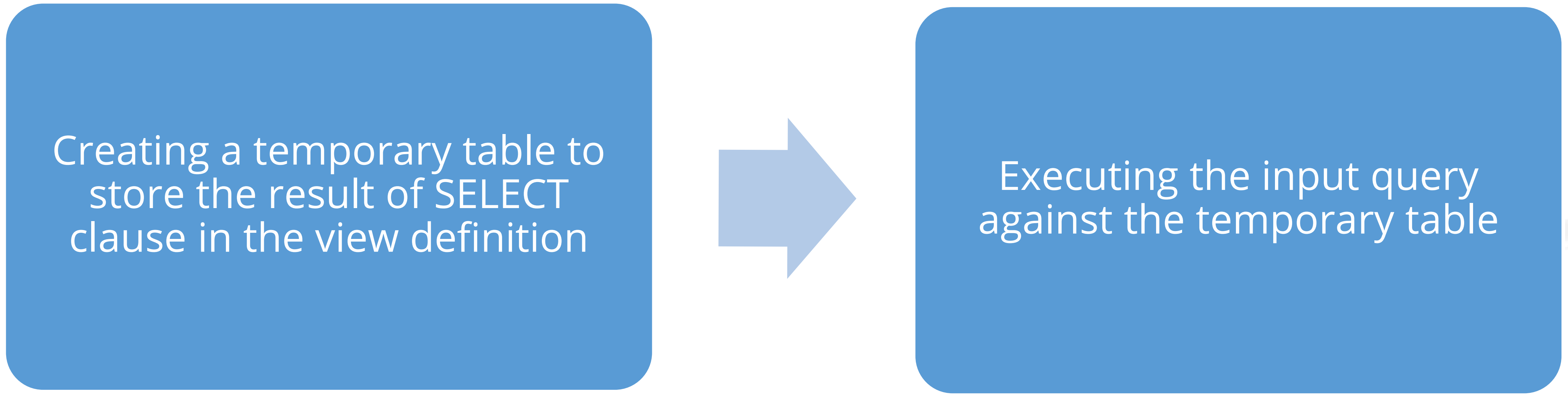
Output:



Emp_Name	R_name	D_name	Ex_p
Roy	Lead Data Scientist	Retail	7
Katrina	Junior Data Scientist	Retail	2
Clair	Associate Data Scientist	Automotive	3

Workflow of View Processing Algorithms

In TEMPTABLE, the results from the view are retrieved into a temporary table, which is then used to execute the statement.


Creating a temporary table to store the result of SELECT clause in the view definition
Executing the input query against the temporary table

The TEMPTABLE algorithm is less efficient than the MERGE algorithm, as MySQL creates a temporary table to store data from the base table.

Workflow of View Processing Algorithms

UNDEFINED is the default algorithm applied when you create a view without specifying the ALGORITHM clause.

MySQL decides to go either with MERGE or TEMPTABLE but prefers MERGE as it is more efficient.

Updatable View

Updatable in MySQL refers to the ability of executing UPDATE and DELETE queries in the database view.

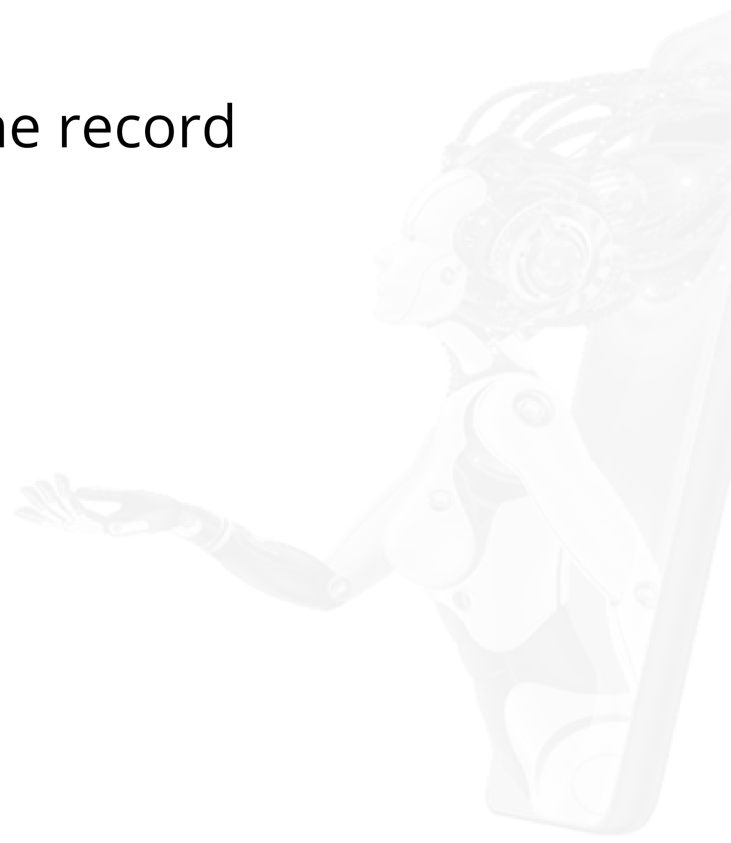
To create an updatable view, you must have the following:

- Any aggregation function
- Subqueries in SELECT and WHERE clause
- UNION or UNION ALL
- Outer or Left Joins
- Having and Group By clauses

Updatable View: Example

Problem Statement: You are a junior DB administrator in your organization. After appraisals, your designation has been changed to Lead Data Scientist. Your Manager has asked you to update the role change in the created view.

Objective: Create a view for employees, using the employee table, and update the record for the employee named Roy.



Updatable View: Example

Step 1: Create a view called *role name after appraisal*

QUERY

```
CREATE VIEW Role_Name_After_Appraisal AS
SELECT Emp_ID, Emp_Name, Role_Name
FROM Emp_Table;
```

Updatable View: Example

Step 2: Update the role name of *Roy* using the following code

QUERY

```
UPDATE Role_Name_After_Appraisal
SE TRole_Name = "Lead Data Scientist‟
WHERE
Emp_Name = "Roy";
```

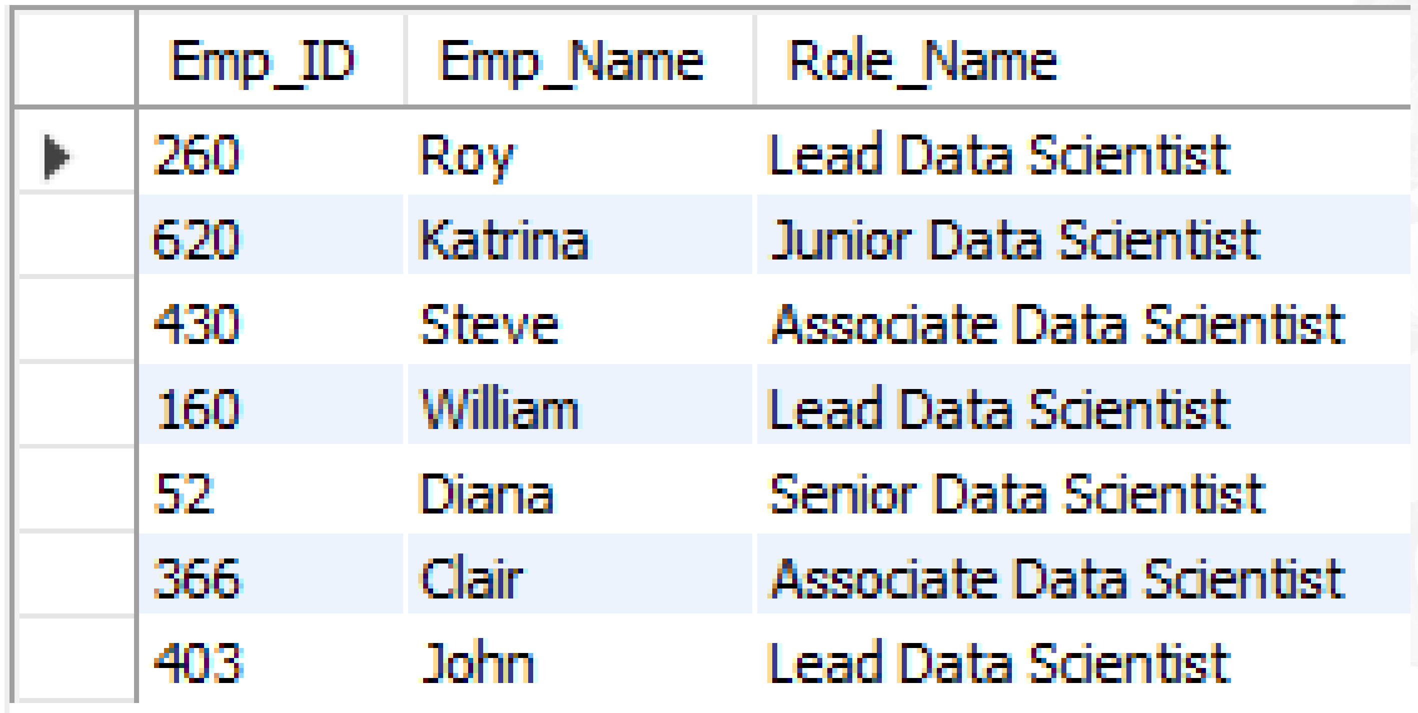
Updatable View: Example

Step 3: Use the *select view* command as shown below to view the changes

QUERY

```
SELECT * FROM Role_Name_After_Appraisal;
```

Output:



Emp_ID	Emp_Name	Role_Name
260	Roy	Lead Data Scientist
620	Katrina	Junior Data Scientist
430	Steve	Associate Data Scientist
160	William	Lead Data Scientist
52	Diana	Senior Data Scientist
366	Clair	Associate Data Scientist
403	John	Lead Data Scientist

Creating Views Using WITH CHECK OPTION

Is an optional clause

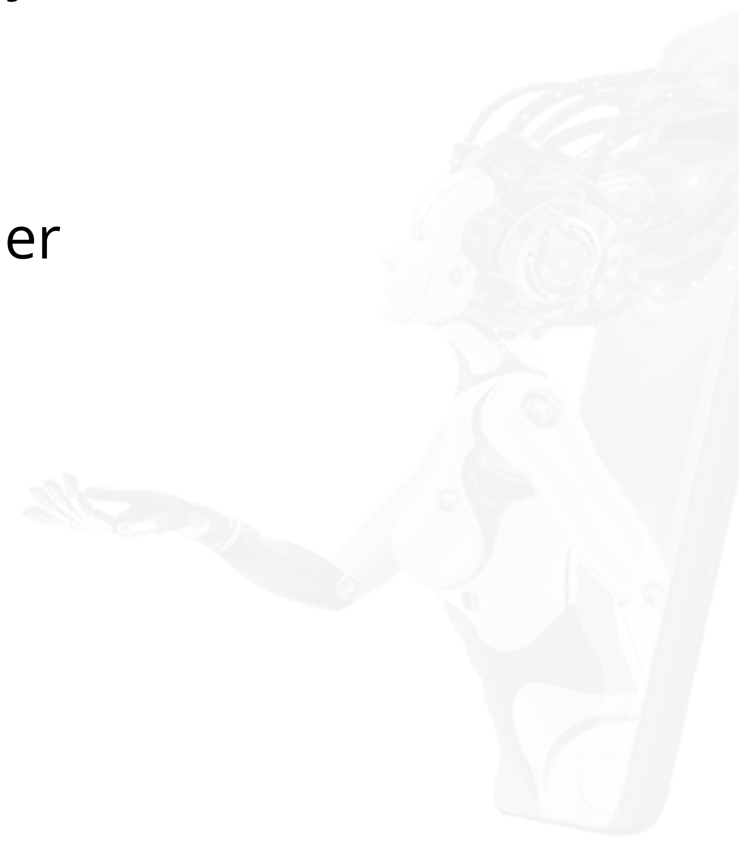
Specifies the level of checking when data is inserted or updated through a view

- If specified, every row that is inserted or updated through the view must conform to the definition of the view.
- If it is not specified, insert and update operations that are performed on the view are not checked for conformance to view the definition.

Creating Views Using WITH CHECK OPTION: Example

Problem Statement: You are the junior DB administrator in your organization, and your Manger has asked you to create a view that displays the Lead Data Scientist role from the employee table created earlier. The view must only allow the addition of employee with the Lead designation. Any other designation entered must prompt an error.

Objective: Create a view using the *WITH CHECK OPTION* to avoid addition of other designations.



Creating Views Using WITH CHECK OPTION: Example

Step 1: Create a view using the *WITH CHECK OPTION* to avoid addition of other designations.

QUERY

```
DELIMITER &&
CREATE OR REPLACE VIEW Lead_DS AS SELECT Emp_ID,
Emp_Name, Role_name, Dept, Experience
FROM Emp_Table
WHERE Role_name LIKE "%Lead‟
WITH CHECK OPTION;
END &&
```

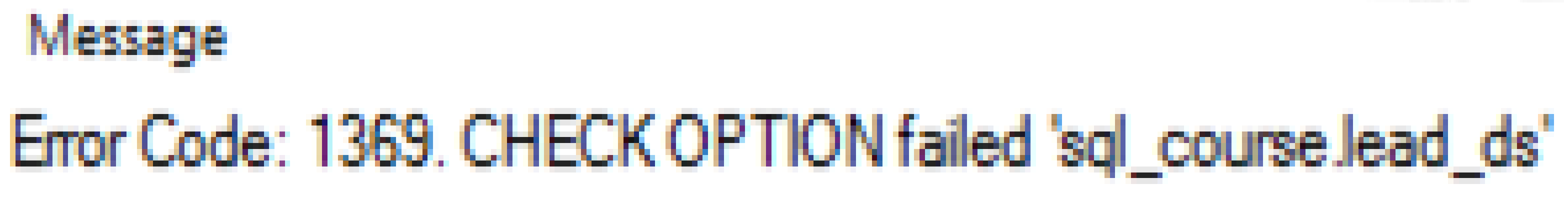
Creating Views Using WITH CHECK OPTION: Example

Step 2: Insert a record of an employee whose designation is not Lead Data Scientist.

QUERY

```
INSERT INTO Lead_DS (Emp_ID, Emp_Name, Role_name,
Dept, Experience) values (333, "Mike", "Senior
Data Scientist","Finance",6);
```

Output:



```
Message
Error Code: 1369. CHECK OPTION failed 'sql_course.lead_ds'
```

When you insert the above values, you get the error message shown above.

Creating Views Using WITH CASCADED CHECK OPTION

This clause specifies that every row that is inserted or updated through a view must conform to the definition of the view.

A row cannot be retrieved through the view if it does not conform to the definition of view.

If the keyword CASCADED is not specified with the WITH CHECK OPTION, then it is by default taken as CASCADED.

Creating Views Using WITH CASCADED CHECK OPTION: Example

QUERY

```
CREATE TABLE T1 (C INT);
CREATE VIEW V1 AS SELECT C
FROM T1 WHERE C >10;
INSERT INTO V1 (C) VALUES (5);
```

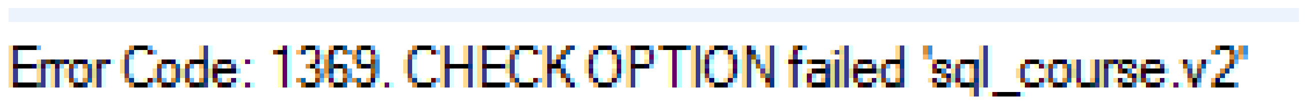
Without **WITH CHECK OPTION**

QUERY

```
CREATE VIEW V2
AS
SELECT C FROM V1
WITH CASCADED CHECK OPTION;
INSERT INTO V2(C) VALUES (5);
```

With **CASCADED CHECK OPTION**

Output:



```
Error Code: 1369. CHECK OPTION failed 'sql_course.v2'
```

Creating Views Using WITH CASCADED CHECK OPTION: Example

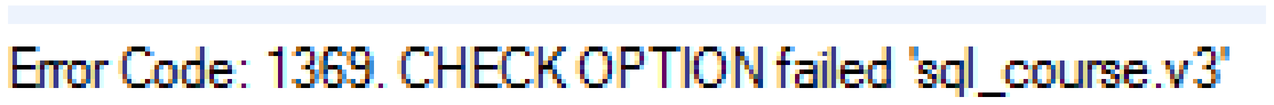
QUERY

```
CREATE VIEW V3
AS
SELECT C
FROM V2
WHERE C < 20;
INSERT INTO V3(C) VALUES (8);
```

QUERY

```
INSERT INTO V3(C) VALUES (30);
```

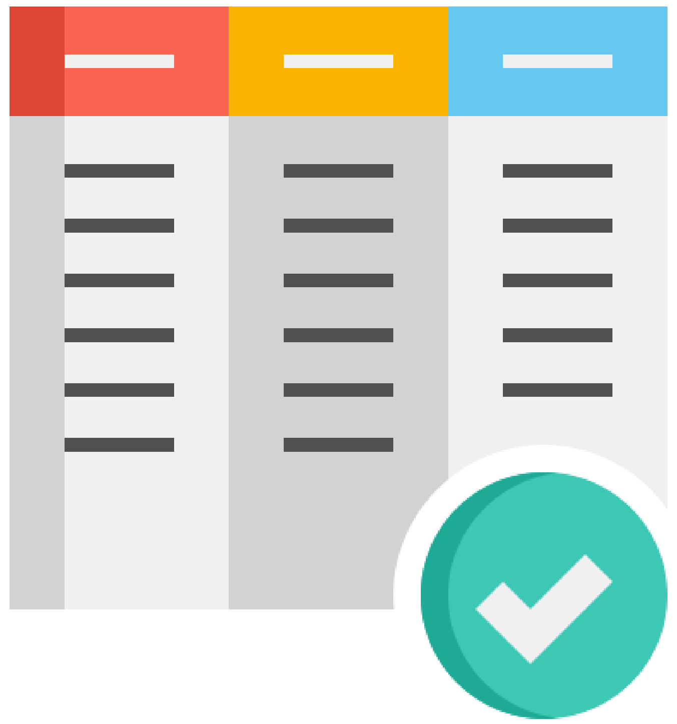
Output:



```
Error Code: 1369. CHECK OPTION failed 'sql_course.v3'
```

Creating Views Using WITH LOCAL CHECK OPTION

WITH LOCAL CHECK OPTION clause is same as WITH CASCADED CHECK OPTION clause, except that you can update a row in such a way that it cannot be retrieved through the view.



This occurs when the view is directly or indirectly dependent on a view that is defined without a WITH CHECK OPTION.

Creating Views Using WITH LOCAL CHECK OPTION: Example

QUERY

```
ALTER VIEW V2 AS
SELECT C
FROM V1
WITH LOCAL CHECK OPTION;
```

QUERY

```
INSERT INTO V2(C) VALUES (5);
```

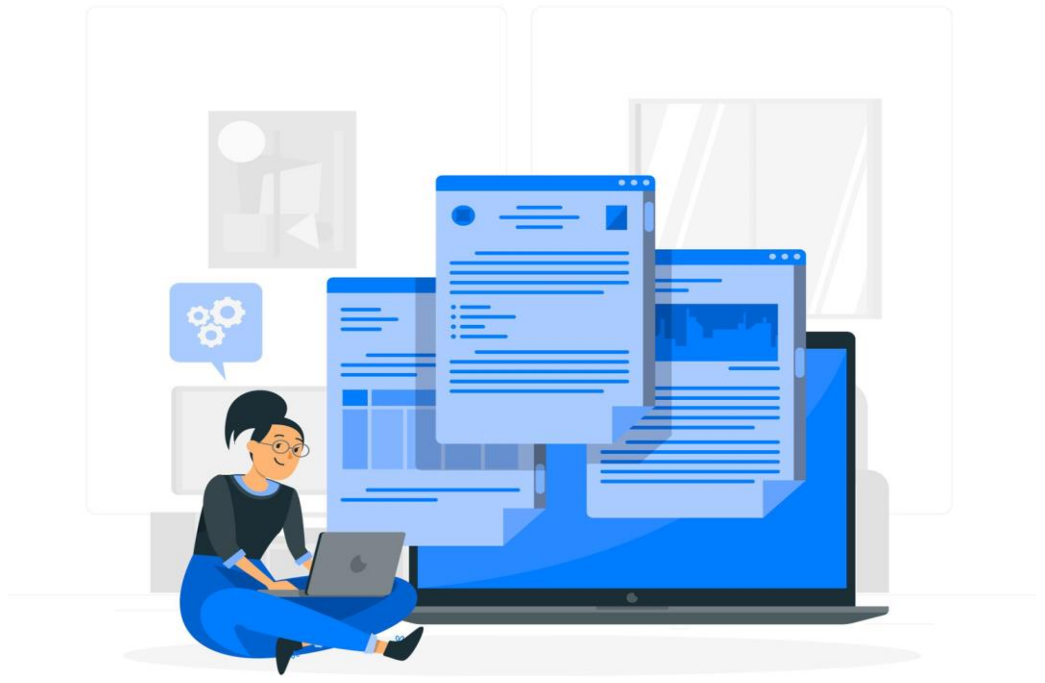
QUERY

```
INSERT INTO V3(C) VALUES (8);
```

For views using WITH LOCAL CHECK OPTION, MySQL checks the rules of views that have a WITH LOCAL CHECK OPTION and a WITH CASCADED CHECK OPTION.

Show Views

In MySQL, views are treated as tables with the type as VIEW. To list or show all the views in the selected database, you need to use the SHOW FULL TABLES command.



SYNTAX

```
SHOW FULL TABLES WHERE table_type = ‘VIEW’;
```

Assisted Practice: VIEWS

Duration: 15 minutes

Problem Statement: Design a VIEW in MySQL which displays the employee’s name, location, and project name from two different tables: data_scientist and project.

Assisted Practice: Views One

Steps to be performed:

1. Create a database with a suitable name and then create a table named **data_scientist** with multiple columns named **emp_code**, **name**, **location, time, and designation.**

TABLE CREATION:

```
CREATE TABLE `data_scientist`(
`emp_code` int NOT NULL,
`name` varchar(255) NOT NULL,
`location` varchar(255) NOT NULL,
`time` int NOT NULL,
`designation` varchar(45) NOT NULL,
PRIMARY KEY (`emp_code`));
```

Assisted Practice: Views One

2. Insert values in the **data_scientist** table.

VALUE INSERTION:

```
INSERT INTO `sys`.`data_scientist` (`emp_code`, `name`, `location`, `experience`,
`desgination`) VALUES ('01', 'ram', 'India', '1', 'jr');
INSERT INTO `sys`.`data_scientist` (`emp_code`, `name`, `location`, `experience`,
`desgination`) VALUES ('02', 'robert', 'USA', '6', 'lead');
INSERT INTO `sys`.`data_scientist` (`emp_code`, `name`, `location`, `experience`,
`desgination`) VALUES ('03', 'tim', 'china', '2', 'Sr');
INSERT INTO `sys`.`data_scientist` (`emp_code`, `name`, `location`, `experience`,
`desgination`) VALUES ('04', 'william', 'paris', '5', 'lead');
INSERT INTO `sys`.`data_scientist` (`emp_code`, `name`, `location`, `experience`,
`desgination`) VALUES ('05', 'maggie', 'UK', '9', 'lead');
```

Assisted Practice: Views One

3. Create a table named **project** with multiple columns named **emp_code**, **project_name, and project_status.**

TABLE CREATION:

```
CREATE TABLE `data_scientist` (
`emp_code` int NOT NULL,
`name` varchar(255) NOT NULL,
`location` varchar(255) NOT NULL,
`time` int NOT NULL,
`designation` varchar(45) NOT NULL,
PRIMARY KEY (`emp_code`)
```

Assisted Practice: Views One

4. Insert values in the **project_name** table.

VALUE INSERTION:

```
INSERT INTO `sys`.`project` (`emp_code`, `project_name`, `project_status`) VALUES
('01', 'C++', 'DONE');
INSERT INTO `sys`.`project` (`emp_code`, `project_name`, `project_status`) VALUES
('02', 'C', 'DONE');
INSERT INTO `sys`.`project` (`emp_code`, `project_name`, `project_status`) VALUES
('03', 'JAVA', 'DONE');
INSERT INTO `sys`.`project` (`emp_code`, `project_name`, `project_status`) VALUES
('04', 'MySQL', 'DONE');
INSERT INTO `sys`.`project` (`emp_code`, `project_name`, `project_status`) VALUES
('05', 'Python', 'DONE');
```

Assisted Practice: Views One

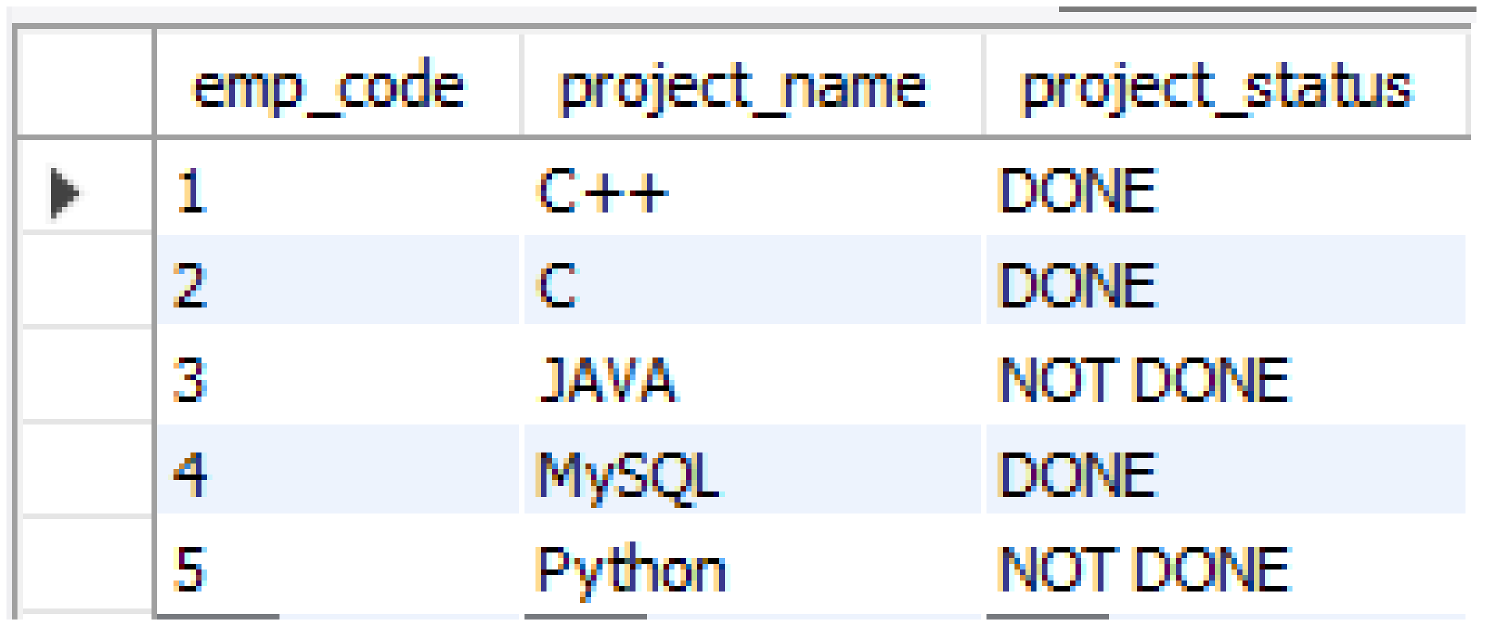
5. Write a query for creating a VIEW named as display for displaying the desired contents of both the tables.

VIEW Creation:

```
DROP VIEW display;
CREATE VIEW display AS
SELECT data_scientist.name , data_scientist.location , project.project_name
FROM data_scientist , project WHERE data_scientist.emp_code = project.emp_code ;
SELECT * FROM display;
```

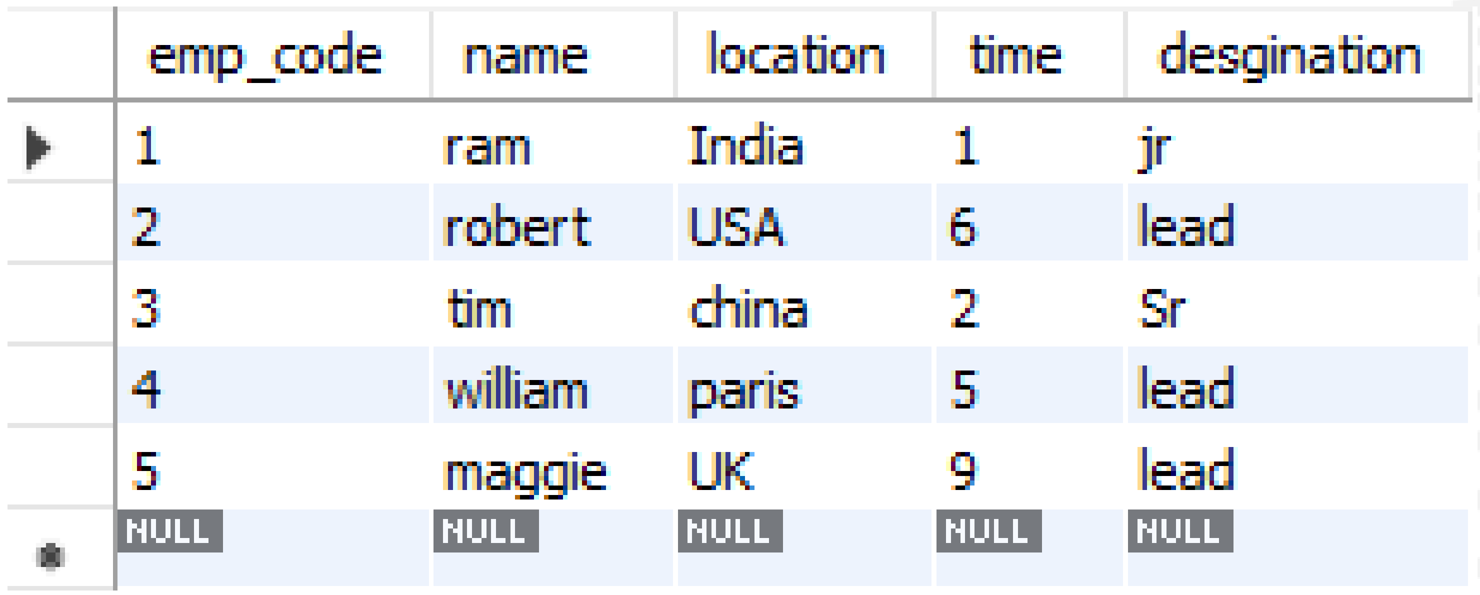
Assisted Practice: Views One - Lab Output

Table project:



emp_code	project_name	project_status
1	C++	DONE
2	C	DONE
3	JAVA	NOT DONE
4	MySQL	DONE
5	Python	NOT DONE

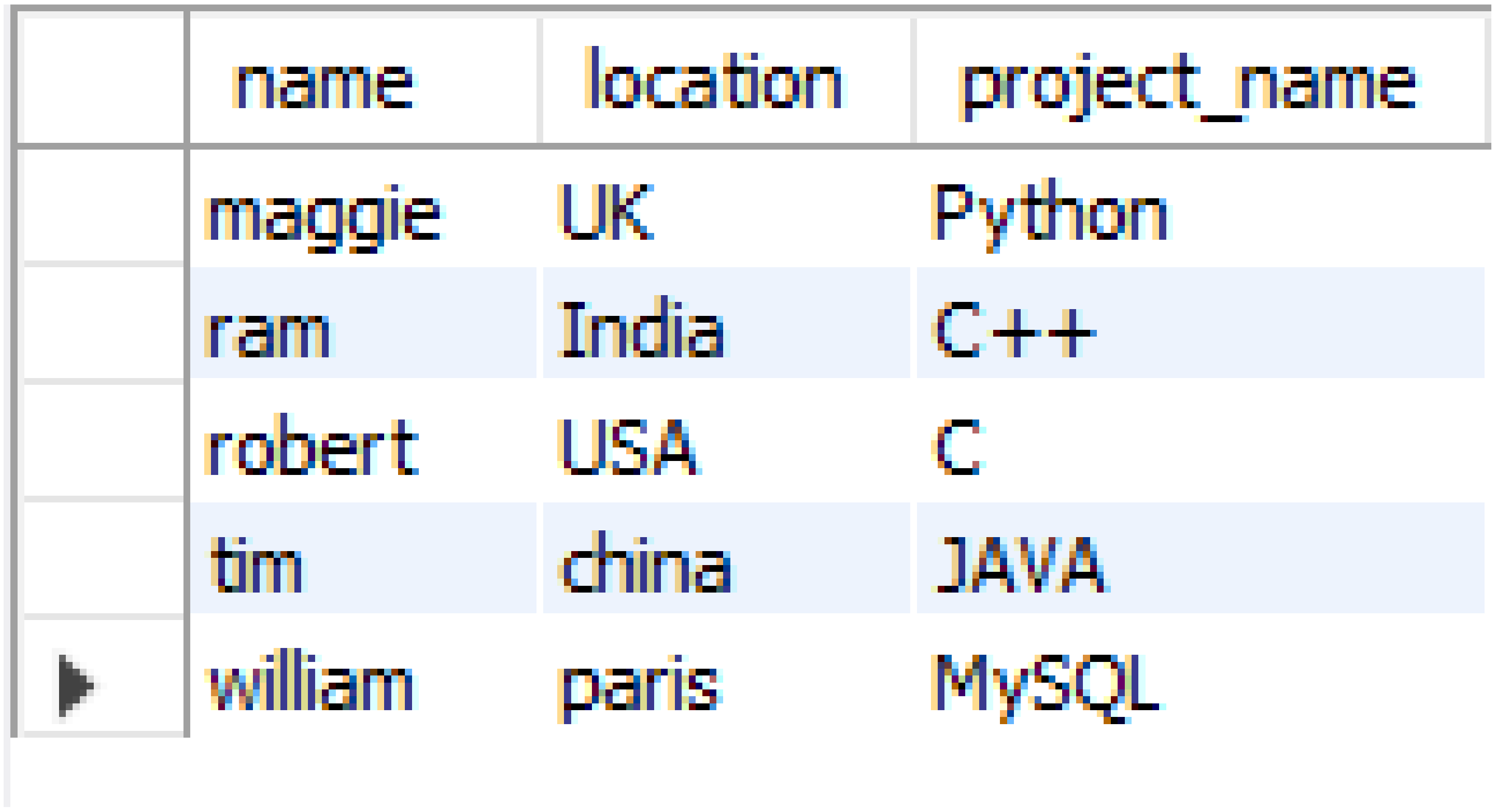
Table data_scientist:



emp_code	name	location	time	desgination
1	ram	India	1	jr
2	robert	USA	6	lead
3	tim	china	2	Sr
4	william	paris	5	lead
5	maggie	UK	9	lead
NULL	NULL	NULL	NULL	NULL

Assisted Practice: Views One - Lab Output

Output of VIEW display:



name	location	project_name
maggie	UK	Python
ram	India	C++
robert	USA	C
tim	china	JAVA
william	paris	MySQL

Assisted Practice: Views Two

Duration: 20 mins

Problem statement: As an SQL expert, you have been asked to analyze the customer purchase data preferably using VIEWS so that the other concerned users only have access to the data they need, while protecting other data in the same table.

Assisted Practice: Views Two

Steps to be performed:

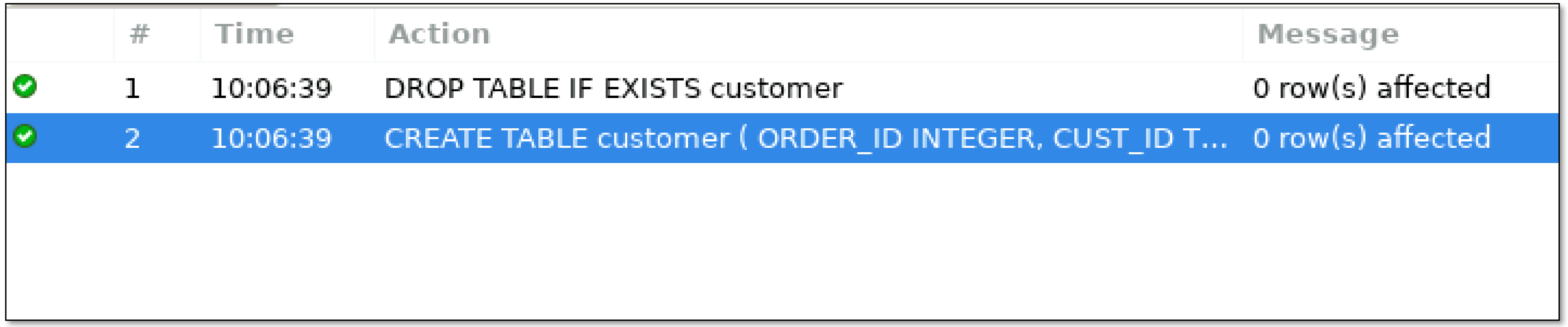
Step 01: Create a table named **“customer”** containing the columns **ORDER_ID, ORDER_DATE, CUST_ID, PROD_ID, UNIT_QTY,** and **WEIGHT**

CREATE

```
DROP TABLE IF EXISTS customer;
CREATE TABLE customer (
ORDER_ID INTEGER,
CUST_ID TEXT,
PROD_ID TEXT,
UNIT_QTY INT,
WEIGHT DOUBLE);
```

Assisted Practice: Views Two

Output:



	#	Time	Action	Message
✓	1	10:06:39	DROP TABLE IF EXISTS customer	0 row(s) affected
✓	2	10:06:39	CREATE TABLE customer (ORDER_ID INTEGER, CUST_ID T...	0 row(s) affected

Assisted Practice: Views Two

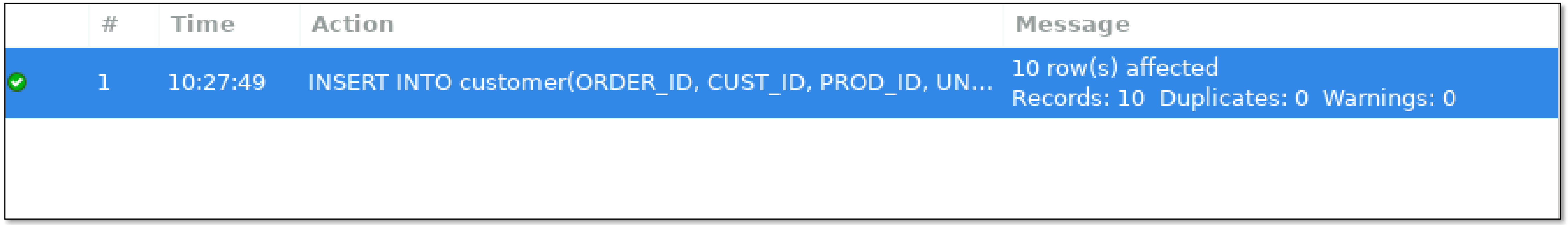
Step 02: Insert values in the **customer** table

SQL Query

```
INSERT INTO customer(ORDER_ID, CUST_ID, PROD_ID, UNIT_QTY, WEIGHT)
VALUES ('1447296447','V55555_53','1700106',808,14.3),
('1447158015','V55555_54','1700106',3188,87.94),
('1447138899','V55555_53','1700107',2331,61.2),
('1447363528','V55555_55','1700109',847,16.16),
('1447363981','V55555_53','1700108',2163,52.34),
('1447351441','V55555_56','1700106',3332,92.8),
('1447320236','V55555_57','1700107',1782,46.9),
('1447158019','V55555_57','1700107',427,2.86),
('1447219341','V55555_58','1700109',1291,26.6),
('1447398416','V55555_59','1700108',2294,62.2);
```

Assisted Practice: Views Two

Output:



	#	Time	Action	Message
✓	1	10:27:49	INSERT INTO customer(ORDER_ID, CUST_ID, PROD_ID, UN...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0

Assisted Practice: Views Two

Step 03: Write a query to create a **VIEW** using the **customer** table capturing the details of customers who have purchased more than 2000 units

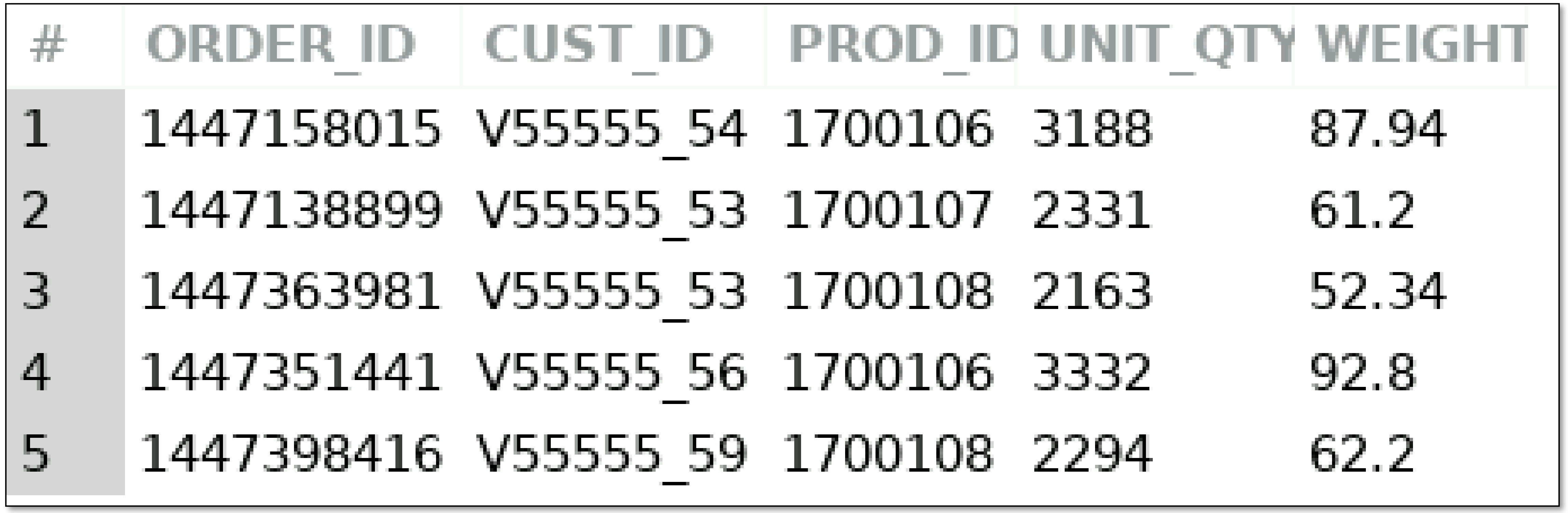
SQL Query

```
CREATE VIEW C1 AS
SELECT * FROM customer WHERE UNIT_QTY>2000;

SELECT * FROM C1;
```

Assisted Practice: Views Two

Output:



#	ORDER_ID	CUST_ID	PROD_ID	UNIT_QTY	WEIGHT
1	1447158015	V55555_54	1700106	3188	87.94
2	1447138899	V55555_53	1700107	2331	61.2
3	1447363981	V55555_53	1700108	2163	52.34
4	1447351441	V55555_56	1700106	3332	92.8
5	1447398416	V55555_59	1700108	2294	62.2

Assisted Practice: Views Two

Step 04: Write a query to create a V**IEW** using the **VIEW** created in previous step with the columns **CUST_ID** and **UNIT_QTY** and use **CHECK OPTION** to ensure records where the quantity is greater than 3000 are not allowed

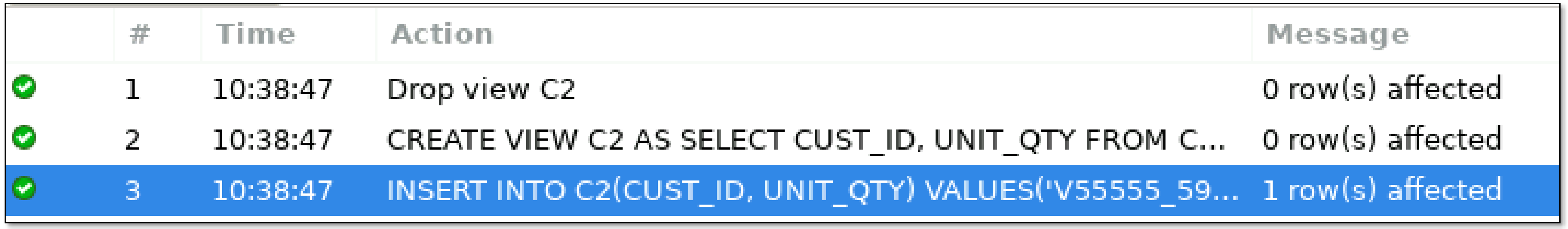
SQL Query

```
CREATE VIEW C2 AS
SELECT CUST_ID, UNIT_QTY FROM C1 WHERE UNIT_QTY<=3000
WITH CASCADED CHECK OPTION;

INSERT INTO C2(CUST_ID, UNIT_QTY) VALUES('V55555_59',2200);
```

Assisted Practice: Views Two

Output:



	#	Time	Action	Message
✓	1	10:38:47	Drop view C2	0 row(s) affected
✓	2	10:38:47	CREATE VIEW C2 AS SELECT CUST_ID, UNIT_QTY FROM C...	0 row(s) affected
✓	3	10:38:47	INSERT INTO C2(CUST_ID, UNIT_QTY) VALUES('V55555_59...	1 row(s) affected

Assisted Practice: Views Two

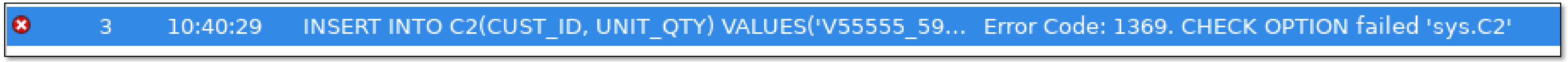
Step 05: Write an insert query to add a record value to the **VIEW** created in step 04, with CUST_ID and UNIT_QTY values. Keep the UNIT_QTY values at 1000, which violates the view's criteria, which only allows values more than 2000 and less than or equal to 3000.

Test Case 01:

```
INSERT INTO C2(CUST_ID, UNIT_QTY) VALUES('V55555_59',1000);
```

Assisted Practice: Views Two

Output:



```
3    10:40:29    INSERT INTO C2(CUST_ID, UNIT_QTY) VALUES('V55555_59...    Error Code: 1369. CHECK OPTION failed 'sys.C2'
```

Assisted Practice: Views Two

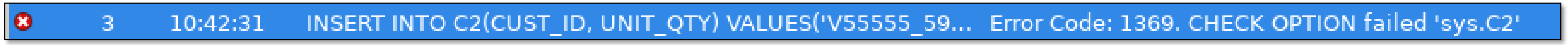
Step 06: Write an insert query to add a record value to the **VIEW** created in step 04, with CUST_ID and UNIT_QTY values. Keep the UNIT_QTY values at 3500, which violates the view's criteria, which only allows values more than 2000 and less than or equal to 3000.

Test Case 02:

```
INSERT INTO C2(CUST_ID, UNIT_QTY) VALUES('V55555_59',3500);
```

Assisted Practice: Views Two

Output:



```
3   10:42:31   INSERT INTO C2(CUST_ID, UNIT_QTY) VALUES('V55555_59...   Error Code: 1369. CHECK OPTION failed 'sys.C2'
```

Knowledge Check

Knowledge Check

1

Which of the following commands is used to display all the views?

A. SHOW VIEWS

B. DISPLAY VIEWS

C. SHOW FULL TABLES

D. SHOW ALL VIEWS

Knowledge Check
1

Which of the following commands is used to display all the views?

A. SHOW VIEWS

B. DISPLAY VIEWS

C. SHOW FULL TABLES

D. SHOW ALL VIEWS

The correct answer is **C**

SHOW FULL TABLES displays all the views created in a database.

Knowledge Check

2

What cannot be done on a view?

A. DISPLAY

B. FILTER

C. INDEX

D. DROP

Knowledge Check

2

What cannot be done on a view?

A. DISPLAY

B. FILTER

C. INDEX

D. DROP

The correct answer is **C**

Views are virtual tables in MySQL. The creation of indexes on a view is not possible. However, they can be used for the views that are processed using the merge algorithm.

Knowledge Check

3

A view can be created for ___________

A. One table

B. Many tables

C. Another view

D. All of the above

Knowledge Check

3

A view can be created for ___________

A. One table

B. Many tables

C. Another view

D. All of the above

The correct answer is **D**

Views can be generated using a single table, multiple tables, or an existing view.

Knowledge Check

4

Which view option ensures that all UPDATE and INSERT satisfy the condition(s) specified in the view definition?

A. UNCHECK

B. WITH CHECK

C. CHECK

D. WITH

Knowledge Check
4

Which view option ensures that all UPDATE and INSERT satisfy the condition(s) specified in the view definition?

A. UNCHECK

B. WITH CHECK

C. CHECK

D. WITH

The correct answer is **B**

The purpose of the WITH CHECK OPTION is to ensure that all the UPDATE and INSERT satisfy the condition(s) specified in the view definition.

Knowledge Check

5

Which of the following statements is false about views?

A. Database views are created using the CREATE VIEW statement.

B. To create a view, a user must have the appropriate system privilege according to the specific implementation.

C. One view can be used in the expression defining another view.

D. We can update a view if it has multiple database relations in the FROM clause.

Knowledge Check

5

Which of the following statements is false about views?

A. Database views are created using the CREATE VIEW statement.

B. To create a view, a user must have the appropriate system privilege according to the specific implementation.

C. One view can be used in the expression defining another view.

D. We can update a view if it has multiple database relations in the FROM clause.

The correct answer is **D**

We can only update a view if the FROM clause has a single database relation.

Knowledge Check

6

Can we insert and delete rows in a view?

A. Yes

B. No

C. Rows of data can be inserted but cannot be deleted

D. Rows of data can be deleted but cannot be inserted

Knowledge Check

6

Can we insert and delete rows in a view?

A. Yes

B. No

C. Rows of data can be inserted but cannot be deleted

D. Rows of data can be deleted but cannot be inserted

The correct answer is **A**

Yes, we can insert and delete rows in a view.

Lesson-End Project: Freight Company Analysis

Problem statement:

As an analyst of a port inspection team at a freight company, you've been asked to perform analyses using VIEWS.

Objective:

To analyze the transportation of orders of various quantities across different ports

Lesson-End Project: Freight Company Analysis

Tasks to be performed:

Step 01: Upload the **FreightRates.csv** dataset to the lab

Step 02: Import **freightRates.csv** into the database as a table

Step 03: Create a VIEW and name it **FR_GROUND** and use it to display the record count of the **FreightRates** table for the orders transported via *GROUND*

Step 04: Create a VIEW, name it **FR_AIR,** and use it to display the record count of the **FreightRates** table for the orders transported via *AIR*

Lesson-End Project: Freight Company Analysis

Tasks to be performed:

Step 05: Create a VIEW and name it **FR_V1**, with **mode_dsc** and the **average rate** for each type of value in the **mode_dsc** column

Step 06: Alter the VIEW **FR_V1** by replacing the column **mode_dsc** with **carrier**

Step 07: Rename the VIEW **FR_V1** as **FR_GROUPED_DATA**

Step 08: Create a VIEW with the **carrier** and **mode_dsc** columns using **WITH CHECK** option to make sure that **mode_dsc** accepts no entry other than *AIR* or *GROUND*

Key Takeaways

- SQL views are created by selecting fields from one or more tables.
- ALTER VIEW command allows you to change the SQL statements present in a view.
- MERGE, TEMPLATE, and UNDEFINED are the three types of view algorithms.
- The WITH CHECK OPTION specifies the level of checking when data is entered or modified using a view.