Performance Optimization and Best Practices in SQL

Learning Objectives

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

- Implement an execution plan in SQL
- Compare VARCHAR, CHAR, and NVARCHAR
- List the various index guidelines
- Outline queries with SQL indexes

Execution Plan in SQL

Execution Plan in SQL

Execution plan

- Unlike other database solutions, MySQL does not generate byte-code to execute a query result. Instead, the query execution plan is used.
- The query execution plan is a list of instructions that the query execution must follow to produce the query result.
- It converts the source code (SQL query) into an executable program.

Execution Plan in SQL

Problem Scenario:
A data analyst of a company wants to check the execution plan for a query to view the performance and cost of the query.

Objective:
View the execution plan and create an index on the query to improve efficiency of the query.

Execution Plan in SQL

Creating exe_plan

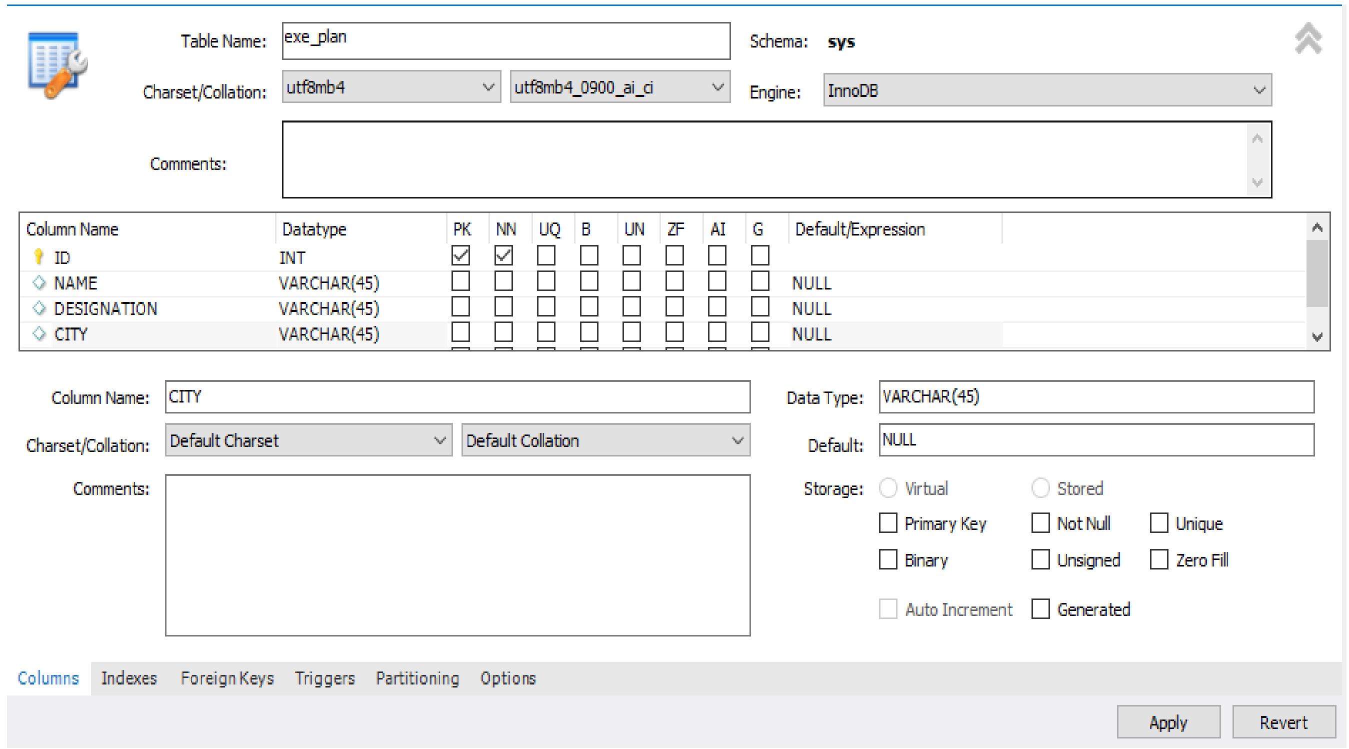


Table Name: exe_plan

Schema: sys

Charset/Collation: utf8mb4 | utf8mb4_0900_ai_ci

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT	☑	☑	□	□	□	□	□	□	
NAME	VARCHAR(45)	□	□	□	□	□	□	□	□	NULL
DESIGNATION	VARCHAR(45)	□	□	□	□	□	□	□	□	NULL
CITY	VARCHAR(45)	□	□	□	□	□	□	□	□	NULL

Column Name: CITY

Data Type: VARCHAR(45)

Charset/Collation: Default Charset | Default Collation

Default: NULL

Comments:

Storage: ○ Virtual ○ Stored

□ Primary Key □ Not Null □ Unique

□ Binary □ Unsigned □ Zero Fill

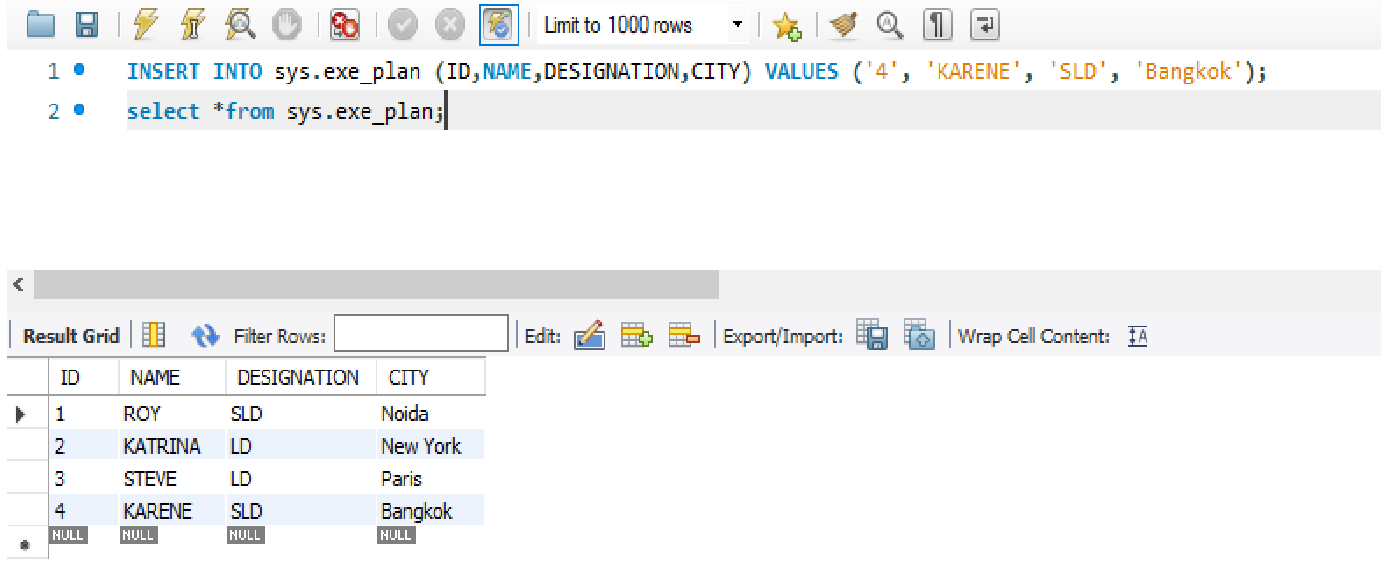
□ Auto Increment □ Generated

QUERY

```
CREATE TABLE exe_plan (
    ID int,
    NAME varchar(255),
    DESIGNATION varchar(255),
    CITY varchar(255),
);
```

Execution Plan in SQL

Inserting data in a table



```
INSERT INTO sys.exe_plan (ID,NAME,DESIGNATION,CITY) VALUES ('4', 'KARENE', 'SLD', 'Bangkok');
select *from sys.exe_plan;
```

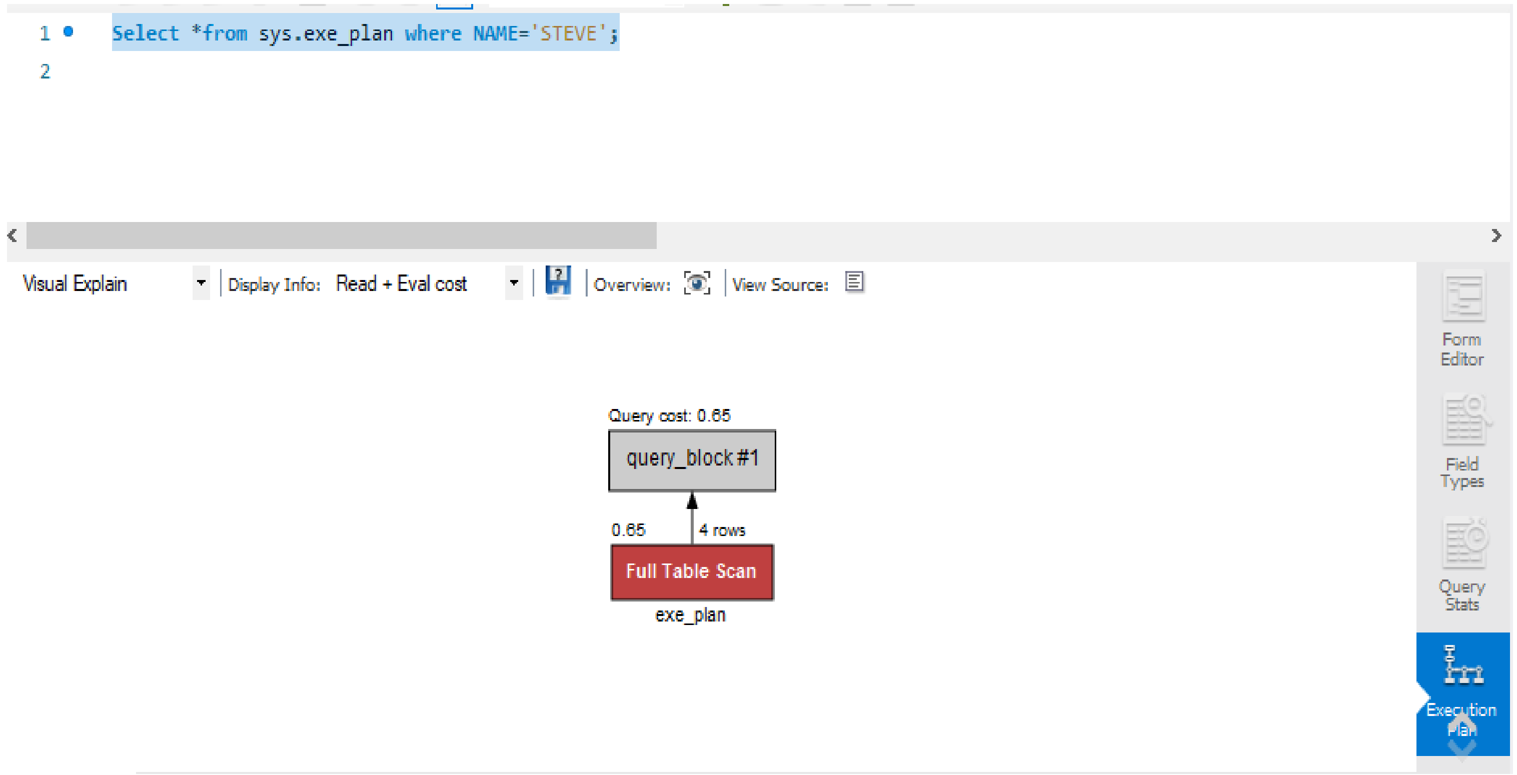
ID	NAME	DESIGNATION	CITY
1	ROY	SLD	Noida
2	KATRINA	LD	New York
3	STEVE	LD	Paris
4	KARENE	SLD	Bangkok
NULL	NULL	NULL	NULL

QUERY

```
INSERT INTO sys.exe_plan
(ID,NAME,DESIGNATION,CITY) VALUES (‘3',
‘STEVE', 'LD', ‘Paris');

Select *from sys.exe_plan;
```

Execution Plan in SQL



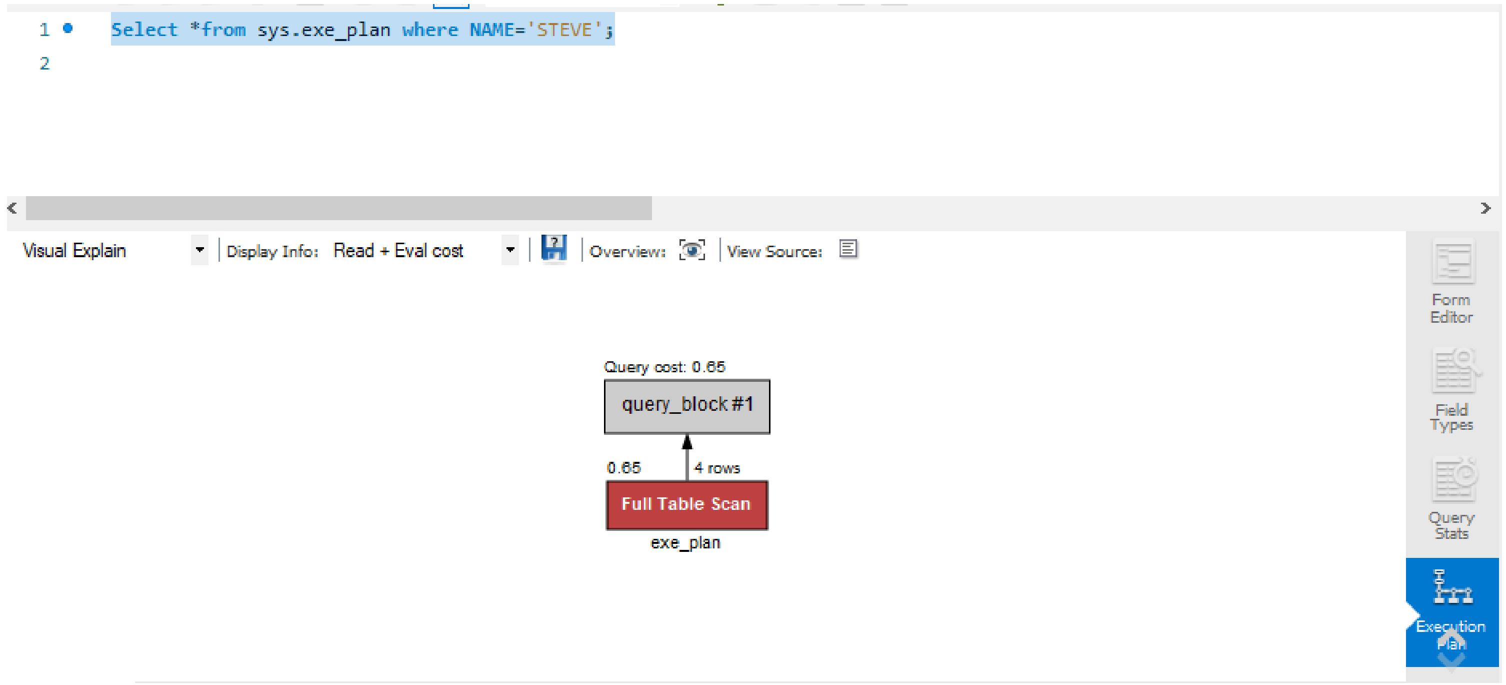
```
Select *from sys.exe_plan where NAME='STEVE';
```

Visual Explain | Display Info: Read + Eval cost

QUERY

```
select *from sys.exe_plan where
NAME='STEVE';
```

Execution Plan in SQL



```
Select *from sys.exe_plan where NAME='STEVE';
```

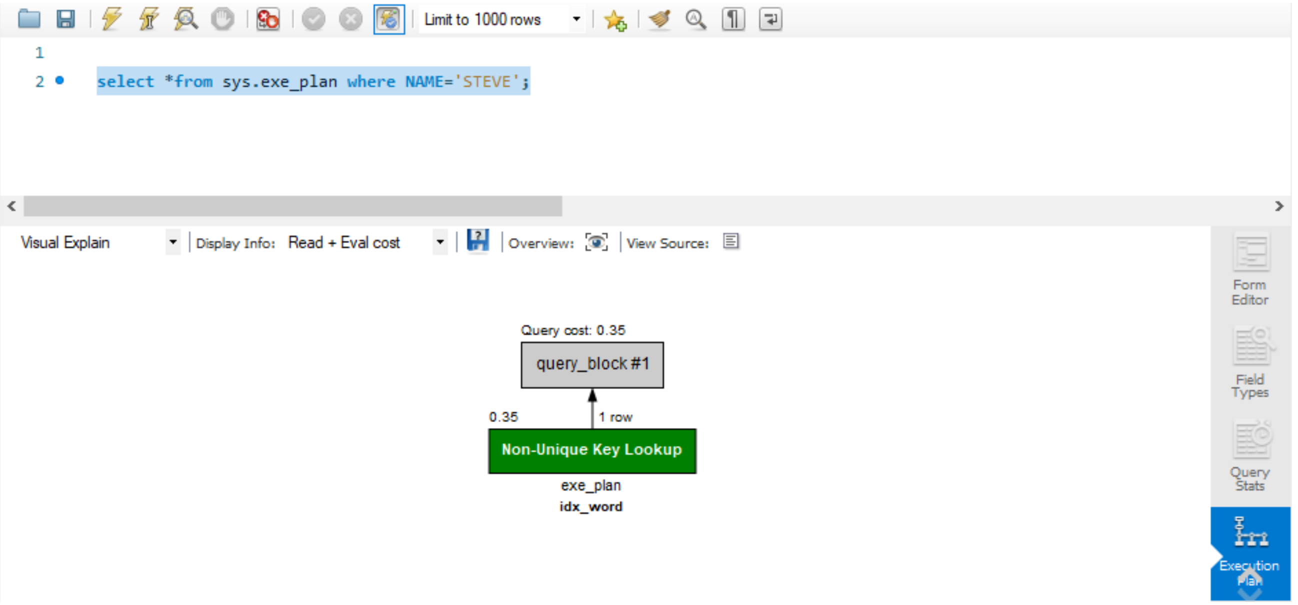
Visual Explain | Display Info: Read + Eval cost | Overview: | View Source:

- The image shows the execution plan, and the red box which is in the image is due to the Performance and high cost of the query.

- Query cost is a metric used in MySQL to determine how expensive a query is in terms of the overall cost of query execution.

Execution Plan in SQL

Creating an index to enhance the query performance



Limit to 1000 rows

```
select *from sys.exe_plan where NAME='STEVE';
```

Visual Explain

Display Info: Read + Eval cost

Overview:

View Source:

Query cost: 0.35

query_block #1

0.35

1 row

Non-Unique Key Lookup

exe_plan

idx_word

QUERY

```
create index idx_word on exe_plan(NAME);

select *from sys.exe_plan where NAME='STEVE';
```

Difference Between CHAR, VARCHAR, and NVARCHAR

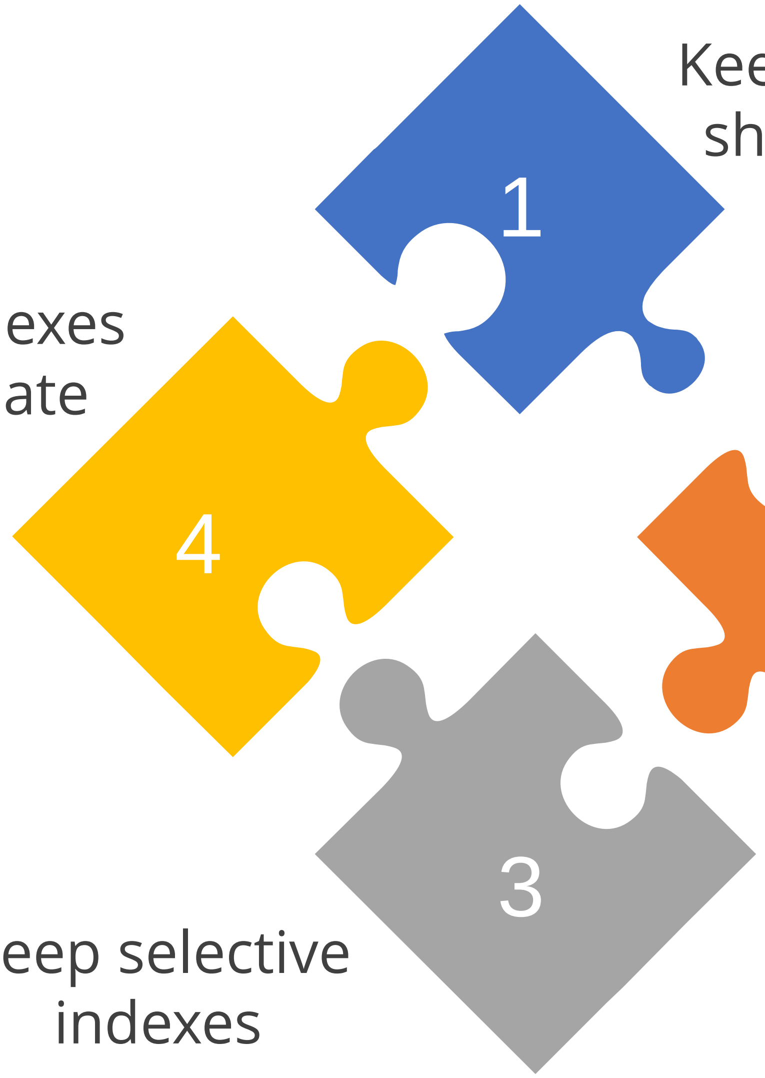
Difference Between CHAR, VARCHAR, and NVARCHAR

CHAR	VARCHAR	NVARCHAR
• It's a data type with a set length.	• It's a data type with a changeable length.	• It's a data type with a changeable length.
• Non-Unicode characters are stored here.	• Non-Unicode characters are stored here.	• Unicode characters are stored here.
• Each character is given one byte of space.	• Each character takes one byte of memory.	• Each character takes two bytes of memory.

Index Guidelines

Index Guidelines

Choosing the right columns and types for an index is a crucial part of building a useful index.



Keep index keys as short as possible

Keep separate index keys

Keep selective indexes

Keep indexes up-to-date

Index Guidelines

Keep index keys as short as possible

The larger an index key gets, the harder it is for a database to use it.
An integer key is smaller than a character field that can carry 100 characters.
Keep clustered indexes as short as possible.

Index Guidelines

Keep separate index keys

Indexes with a limited percentage of duplicated values are the most effective. With a decent index, the database will be able to ignore as many records as possible.

Index Guidelines

Keep selective indexes

A selective index has a lot of unique values. A unique index is the most selective of all the indexes, because there are no duplicate values.

Index Guidelines

Keeping indexes up-to-date

You will need to see existing indexes as well as delete or rename them, in addition to generating new ones. As the schema or even naming standards change, this is part of the database's continual maintenance cycle.

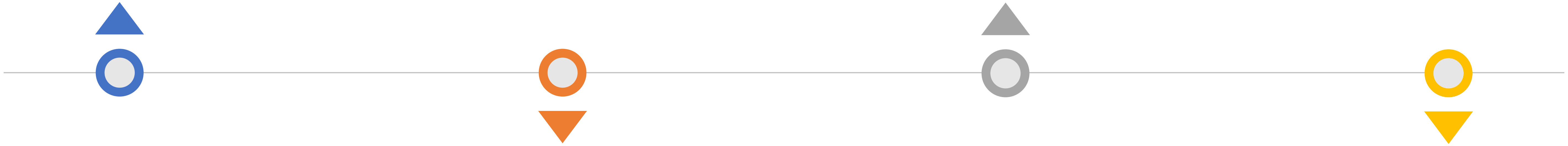
Creating a Clustered Index in SQL

Clustered Index in SQL

- A clustered index is an index that reorders the actual storage of entries in a table.
- Each table can only have one clustered index.

Clustered Index in SQL

Following are the essential characteristics of a clustered index:



It allows us to store both data and indexes at the same time.

It only has one manner of storing data, which is dependent on the key values.

It's an excellent choice for range or group queries that return min, max, or count values.

It always takes one or more columns to create an index.

Covering Queries With Indexes in SQL

A covered query is a query where all the columns in the query's result set are pulled from non-clustered indexes.

The careful placement of indexes transforms a query into a covered query.

Clustered Index in SQL

Problem Scenario:

A data analyst wants to sort a table in order, with the help of a clustered index. Create a primary key which acts as a clustered index in that table.

Objective:

Implement the clustered index to obtain the result.

Instructions:

Refer the emp_data table created before and perform the objectives.

Table Description

Field Name	Description
EMP_ID	Employee ID
FIRST_NAME	First name of the employee
LAST_NAME	Last name of the employee
GENDER	Gender of the employee (M/F)
ROLE	Designation of the employee (Junior, Senior, Lead, and Associate Data Scientist)
DEPT	Name of the department (Retail, Finance, Automotive, and Healthcare)

Table Description

Field Name	Description
EXP	Experience of the employee
COUNTRY	Country where the employee lives
CONTINENT	Continent based on the country
SALARY	Salary of the employee per month
EMP_RATING	Rating for the employee (1: Not Achieving Any Goals, 2: Below Expectations, 3: Meeting Expectations, 4: Excellent Performance, 5: Overachiever
MANAGER_ID	It is the employee ID for the manager

Clustered Index in SQL

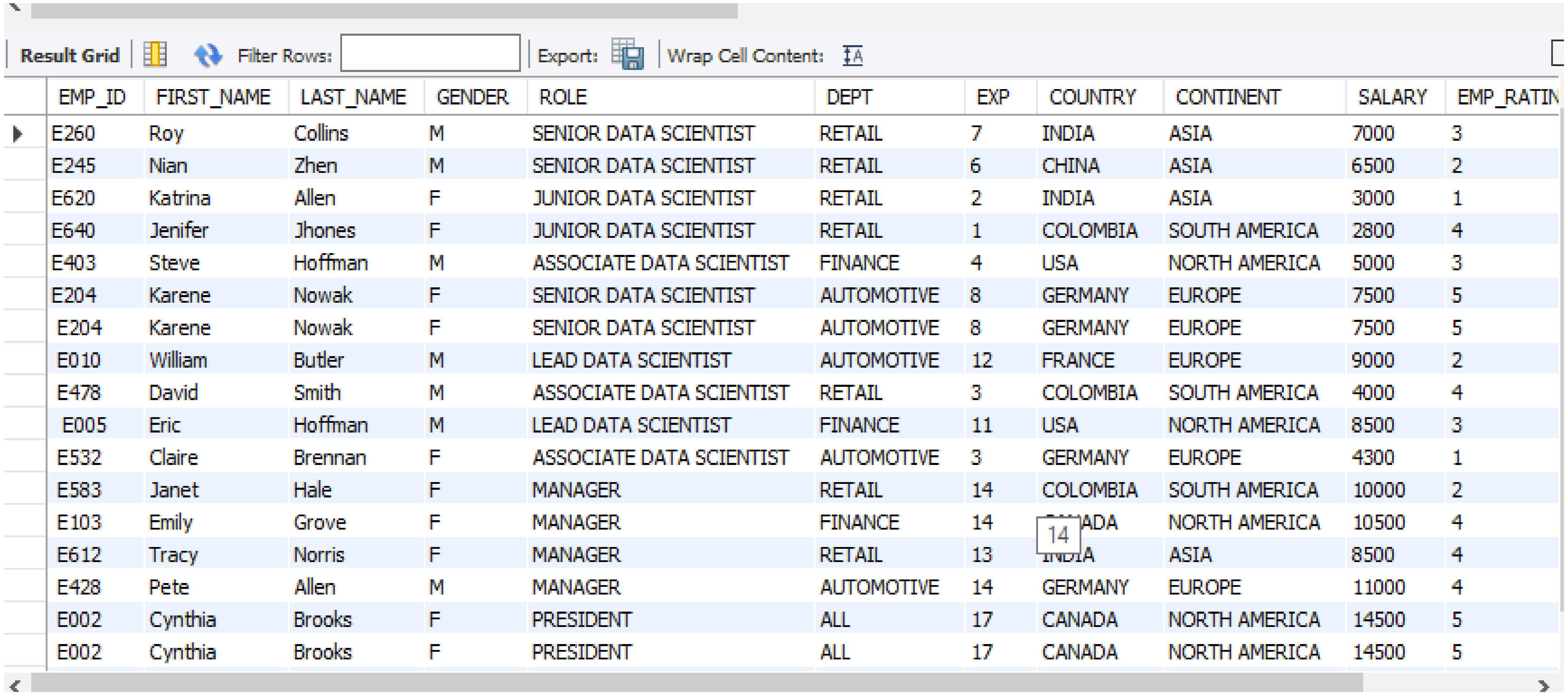
In the query below, the PRIMARY KEY is a clustered index.

QUERY

```
CREATE TABLE
emp_data( `EMP_ID` Varchar NOT NULL,    `FIRST_NAME` varchar(45) DEFAULT NULL,    `
LAST_NAME` varchar(3) DEFAULT NULL,    `GENDER` varchar(20) DEFAULT NULL,   `ROLE`
 varchar(25) DEFAULT NULL,   `DEPT` varchar(25) DEFAULT NULL,   `EXP` varchar(25)
DEFAULT NULL,  `COUNTRY` varchar(25) DEFAULT NULL,    `CONTINENT` varchar(25) DEFA
ULT NULL, `SALARY` INT  DEFAULT NULL,    `EMP_RATING` INT
DEFAULT NULL, `MANAGER_ID` varchar(25) DEFAULT NULL,      PRIMARY KEY (`EMP_ID`)//
clustered index    UNIQUE KEY `SALARY` (`SALARY`) ;
```

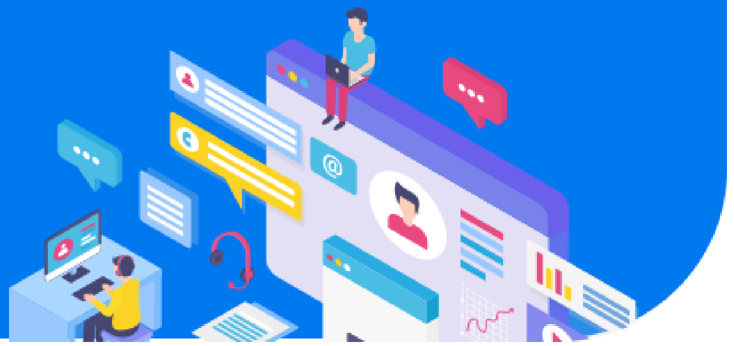
Clustered Index in SQL

Output



EMP_ID	FIRST_NAME	LAST_NAME	GENDER	ROLE	DEPT	EXP	COUNTRY	CONTINENT	SALARY	EMP_RATIN
E260	Roy	Collins	M	SENIOR DATA SCIENTIST	RETAIL	7	INDIA	ASIA	7000	3
E245	Nian	Zhen	M	SENIOR DATA SCIENTIST	RETAIL	6	CHINA	ASIA	6500	2
E620	Katrina	Allen	F	JUNIOR DATA SCIENTIST	RETAIL	2	INDIA	ASIA	3000	1
E640	Jenifer	Jhones	F	JUNIOR DATA SCIENTIST	RETAIL	1	COLOMBIA	SOUTH AMERICA	2800	4
E403	Steve	Hoffman	M	ASSOCIATE DATA SCIENTIST	FINANCE	4	USA	NORTH AMERICA	5000	3
E204	Karene	Nowak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE	7500	5
E204	Karene	Nowak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE	7500	5
E010	William	Butler	M	LEAD DATA SCIENTIST	AUTOMOTIVE	12	FRANCE	EUROPE	9000	2
E478	David	Smith	M	ASSOCIATE DATA SCIENTIST	RETAIL	3	COLOMBIA	SOUTH AMERICA	4000	4
E005	Eric	Hoffman	M	LEAD DATA SCIENTIST	FINANCE	11	USA	NORTH AMERICA	8500	3
E532	Claire	Brennan	F	ASSOCIATE DATA SCIENTIST	AUTOMOTIVE	3	GERMANY	EUROPE	4300	1
E583	Janet	Hale	F	MANAGER	RETAIL	14	COLOMBIA	SOUTH AMERICA	10000	2
E103	Emily	Grove	F	MANAGER	FINANCE	14	CANADA	NORTH AMERICA	10500	4
E612	Tracy	Norris	F	MANAGER	RETAIL	13	INDIA	ASIA	8500	4
E428	Pete	Allen	M	MANAGER	AUTOMOTIVE	14	GERMANY	EUROPE	11000	4
E002	Cynthia	Brooks	F	PRESIDENT	ALL	17	CANADA	NORTH AMERICA	14500	5
E002	Cynthia	Brooks	F	PRESIDENT	ALL	17	CANADA	NORTH AMERICA	14500	5

Assisted Practice: Covering Query



Duration: 15 min

Problem Statement: You are required to create the temperature as unique, add new index columns, and view the index of the weather table for data integrity.

Assisted Practice: Covering Query

Steps to be performed:

Step 1 : Creating the weather table and inserting values in it:

CREATE

```
CREATE TABLE weather (    temp INT NOT NULL,    windspeed varchar(45) NOT
NULL,    vapour varchar(45) NOT NULL,    climate varchar(45) NOT NULL  );
```

-

INSERT

```
INSERT INTO weather(temp,windspeed,vapour,climate) VALUES
('35','120km/hr','21','summer');
```

Assisted Practice: Covering Query

Step 2 : Querying to create unique index:

QUERY

```
CREATE UNIQUE INDEX temp_index ON weather(temp);
```

Step 3 : Querying to add new index:

QUERY

```
ALTER TABLE weather ADD INDEX index_col1_col2(windspeed,vapour,climate);
```

Assisted Practice: Covering Query

Step 3 : Querying to show index:

QUERY

```
SHOW INDEX FROM weather;
```

Output :

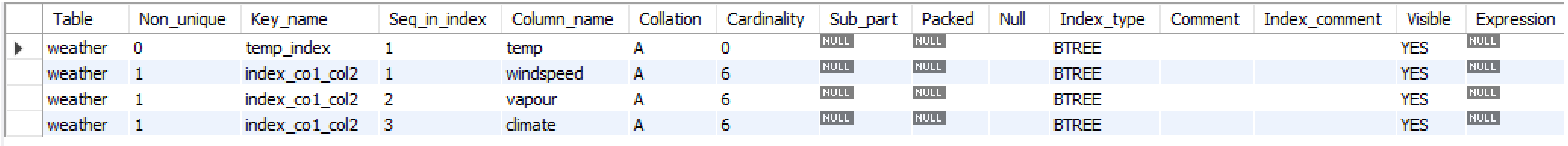


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
weather	0	temp_index	1	temp	A	0	NULL	NULL		BTREE			YES	NULL
weather	1	index_co1_col2	1	windspeed	A	6	NULL	NULL		BTREE			YES	NULL
weather	1	index_co1_col2	2	vapour	A	6	NULL	NULL		BTREE			YES	NULL
weather	1	index_co1_col2	3	climate	A	6	NULL	NULL		BTREE			YES	NULL

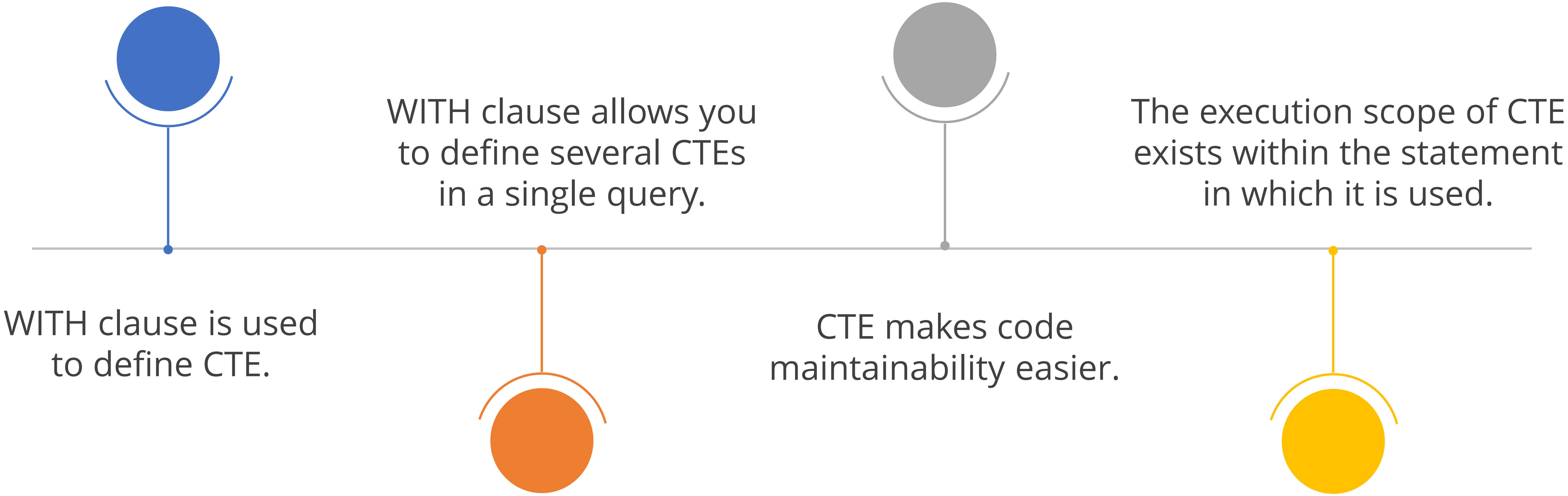
Common Table Expression

Common Table Expression

Each statement or query in MySQL generates a temporary result or relation. CREATE, INSERT, SELECT, UPDATE, DELETE, and other statements employ a common table expression (CTE) to name the temporary results sets that exist within the execution scope of that statement.

Common Table Expression

Following are some of the most important aspects of CTE:



WITH clause is used to define CTE.

WITH clause allows you to define several CTEs in a single query.

CTE makes code maintainability easier.

The execution scope of CTE exists within the statement in which it is used.

Common Table Expression

The basic syntax of CTE in MySQL is as follows:

SYNTAX

```
WITH cte_name (column_names) AS (query)
SELECT * FROM cte_name;
```

Common Table Expression

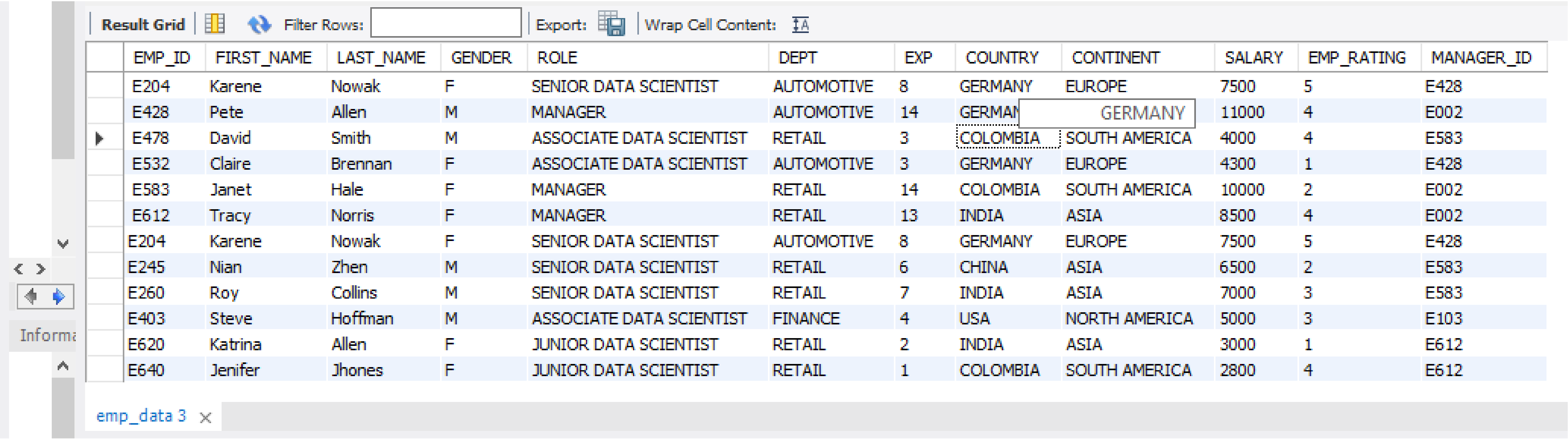
Problem Scenario:
A data analyst wants to apply a CTE to the table with first name, last name, and the country of the person whose salary is greater than or equal to 4000.

Objective:
Implement the CTE to obtain the required result.

Instructions:
Refer the emp_data table which was created and shown before. Perform the objectives mentioned above.

Common Table Expression

Following **emp_data** table is used to show the CTE:



EMP_ID	FIRST_NAME	LAST_NAME	GENDER	ROLE	DEPT	EXP	COUNTRY	CONTINENT	SALARY	EMP_RATING	MANAGER_ID
E204	Karene	Nowak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE	7500	5	E428
E428	Pete	Allen	M	MANAGER	AUTOMOTIVE	14	GERMAN	GERMANY	11000	4	E002
E478	David	Smith	M	ASSOCIATE DATA SCIENTIST	RETAIL	3	COLOMBIA	SOUTH AMERICA	4000	4	E583
E532	Claire	Brennan	F	ASSOCIATE DATA SCIENTIST	AUTOMOTIVE	3	GERMANY	EUROPE	4300	1	E428
E583	Janet	Hale	F	MANAGER	RETAIL	14	COLOMBIA	SOUTH AMERICA	10000	2	E002
E612	Tracy	Norris	F	MANAGER	RETAIL	13	INDIA	ASIA	8500	4	E002
E204	Karene	Nowak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE	7500	5	E428
E245	Nian	Zhen	M	SENIOR DATA SCIENTIST	RETAIL	6	CHINA	ASIA	6500	2	E583
E260	Roy	Collins	M	SENIOR DATA SCIENTIST	RETAIL	7	INDIA	ASIA	7000	3	E583
E403	Steve	Hoffman	M	ASSOCIATE DATA SCIENTIST	FINANCE	4	USA	NORTH AMERICA	5000	3	E103
E620	Katrina	Allen	F	JUNIOR DATA SCIENTIST	RETAIL	2	INDIA	ASIA	3000	1	E612
E640	Jenifer	Jhones	F	JUNIOR DATA SCIENTIST	RETAIL	1	COLOMBIA	SOUTH AMERICA	2800	4	E612

Common Table Expression

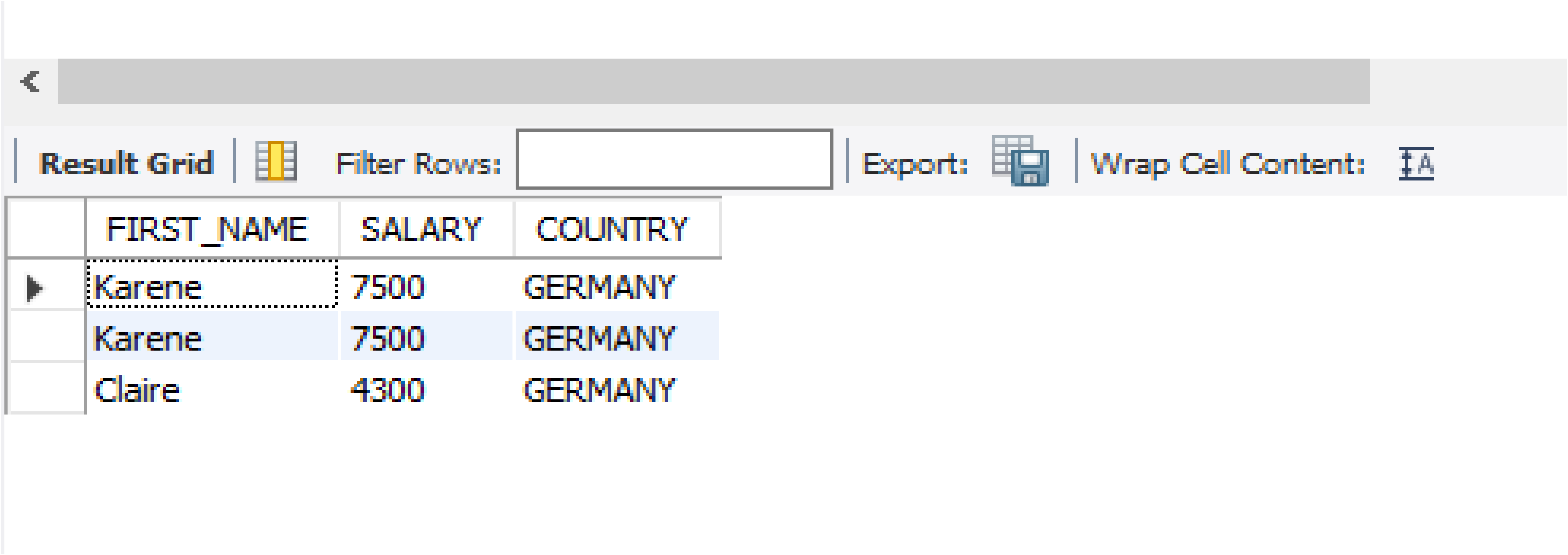
CTE QUERY

```
 WITH emp_in_Germany AS
 (
  SELECT * FROM sys . emp_data WHERE COUNTRY = 'GERMANY'
  )

 SELECT  FIRST_NAME , SALARY , COUNTRY from emp_in_Germany    WHERE SALARY >= '4000’ order
by FIRST_NAME;
```

Common Table Expression

Output:



FIRST_NAME	SALARY	COUNTRY
Karene	7500	GERMANY
Karene	7500	GERMANY
Claire	4300	GERMANY

Assisted Practice: Common Table Expressions

Duration: 20 mins

Problem statement: You've been asked to train a junior analyst on Common Table Expressions (CTE). You've been given a predefined problem to use. Using the data below, you need to guide the junior analyst to write a query using CTE to fetch the count of unique policy number for policies in India.

CUST_ID	POLICY_NO	COUNTRY_ID	CLAIM_AMT
C1	PO21	1	1000
C2	PO22	1	3200
C3	PO33	2	5000
C4	PO44	3	4500
C5	PO55	2	1200
C6	PO44	3	4500
C7	PO55	2	1200

Claim table

COUNTRY_ID	NAME
1	US
2	India
3	UK

Country table

Assisted Practice: Common Table Expressions

Steps to be performed:

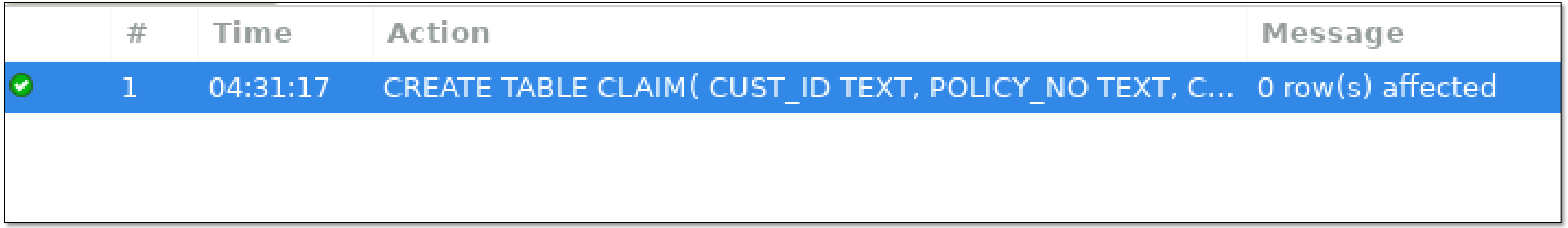
Step 01: Create the **CLAIM** table

SQL Query

```
CREATE TABLE CLAIM(
CUST_ID TEXT,
POLICY_NO TEXT,
COUNTRY_ID INT,
CLAIM_AMT INT
);
```

Assisted Practice: Common Table Expressions

Output:



```
     #   Time       Action                                                 Message
✓    1   04:31:17   CREATE TABLE CLAIM( CUST_ID TEXT, POLICY_NO TEXT, C...  0 row(s) affected
```

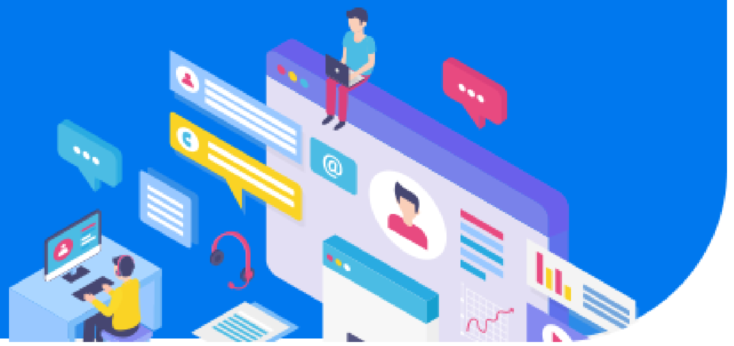
Assisted Practice: Common Table Expressions

Step 02: Create the **COUNTRY** table

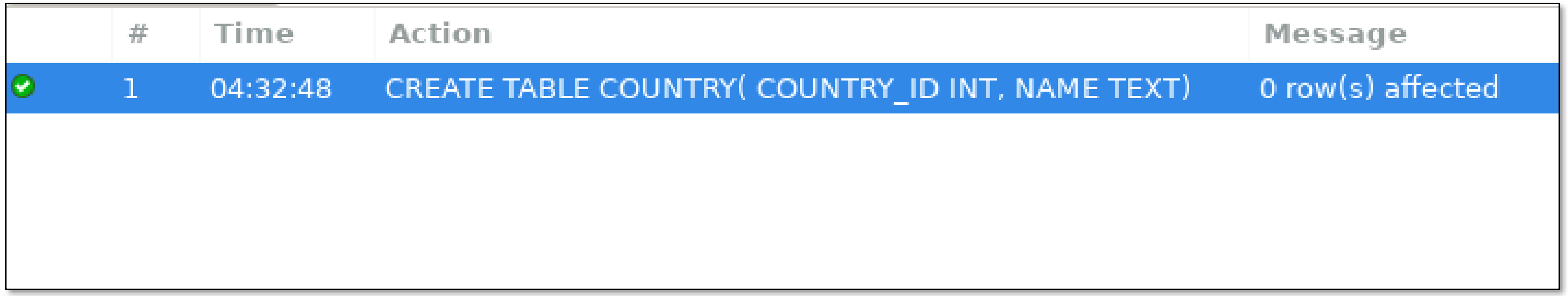
SQL Query

```
CREATE TABLE COUNTRY(
COUNTRY_ID INT,
NAME TEXT);
```

Assisted Practice: Common Table Expressions



Output:



	#	Time	Action	Message
✓	1	04:32:48	CREATE TABLE COUNTRY(COUNTRY_ID INT, NAME TEXT)	0 row(s) affected

Assisted Practice: Common Table Expressions

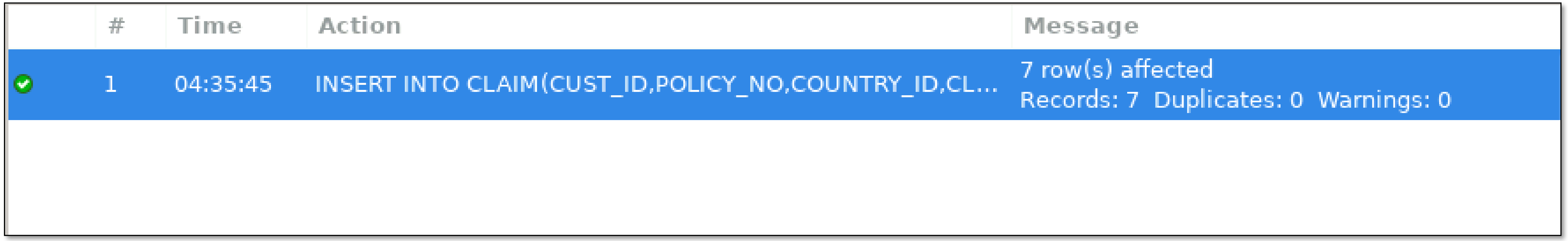
Step 03: Insert records in the **CLAIM** table

SQL Query

```
INSERT INTO CLAIM(CUST_ID,POLICY_NO,COUNTRY_ID,CLAIM_AMT)
VALUES("C","PO21",1,1000),
("C2","PO22",1,3200),
("C3","PO33",2,5000),
("C4","PO44",3,4500),
("C5","PO55",2,1200),
("C6","PO44",3,4500),
("C7","PO55",2,1200);
```

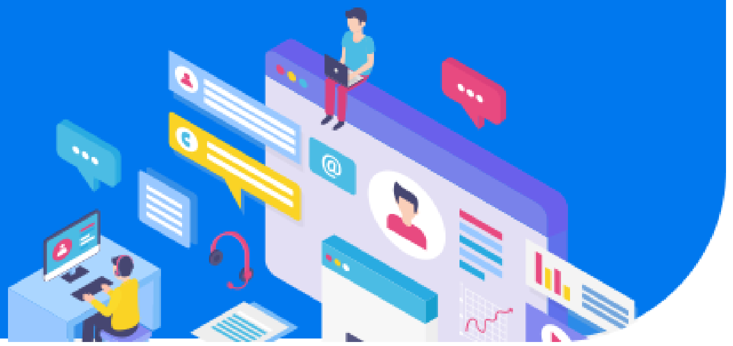
Assisted Practice: Common Table Expressions

Output:



	#	Time	Action	Message
✓	1	04:35:45	INSERT INTO CLAIM(CUST_ID,POLICY_NO,COUNTRY_ID,CL...	7 row(s) affected Records: 7 Duplicates: 0 Warnings: 0

Assisted Practice: Common Table Expressions

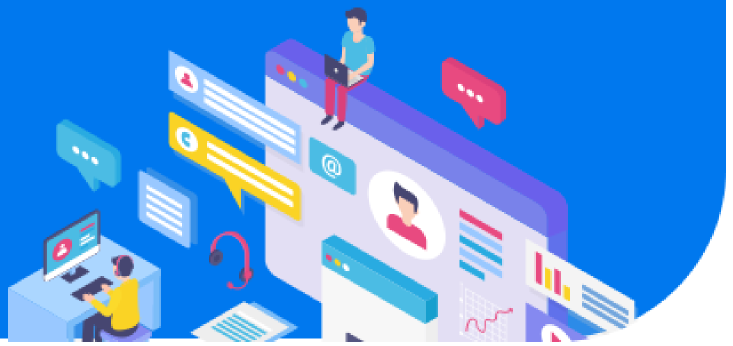


Step 04: Insert records in the **COUNTRY** table

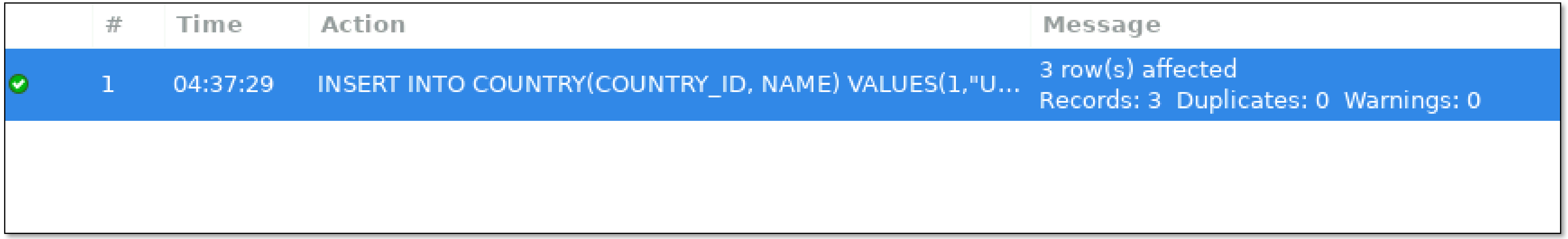
SQL Query

```
INSERT INTO COUNTRY(COUNTRY_ID, NAME)
VALUES(1,"US"),
(2,"India"),
(3,"UK");
```

Assisted Practice: Common Table Expressions



Output:



	#	Time	Action	Message
✓	1	04:37:29	INSERT INTO COUNTRY(COUNTRY_ID, NAME) VALUES(1,"U...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0

Assisted Practice: Common Table Expressions

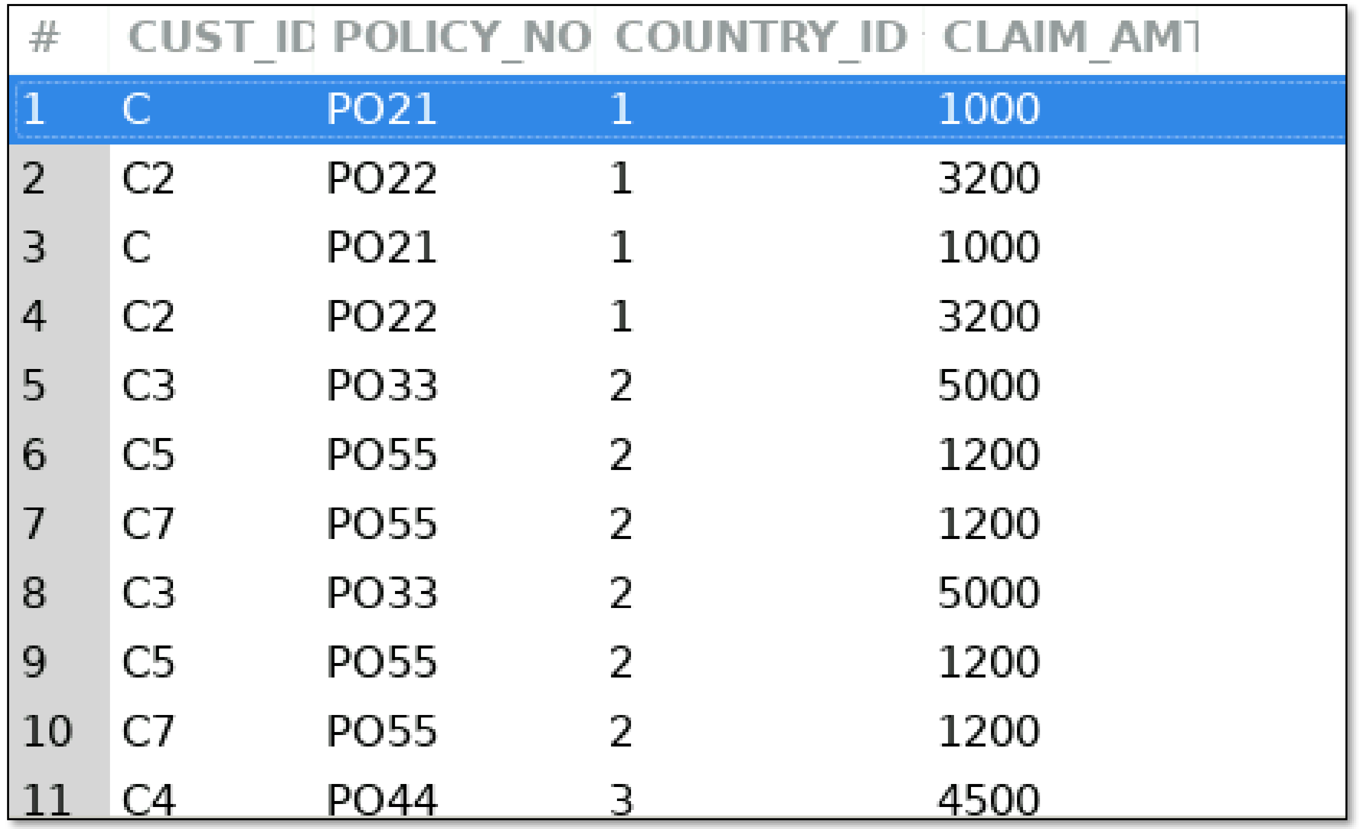
Step 05: Display the **CLAIM** and **COUNTRY** tables

SQL Query

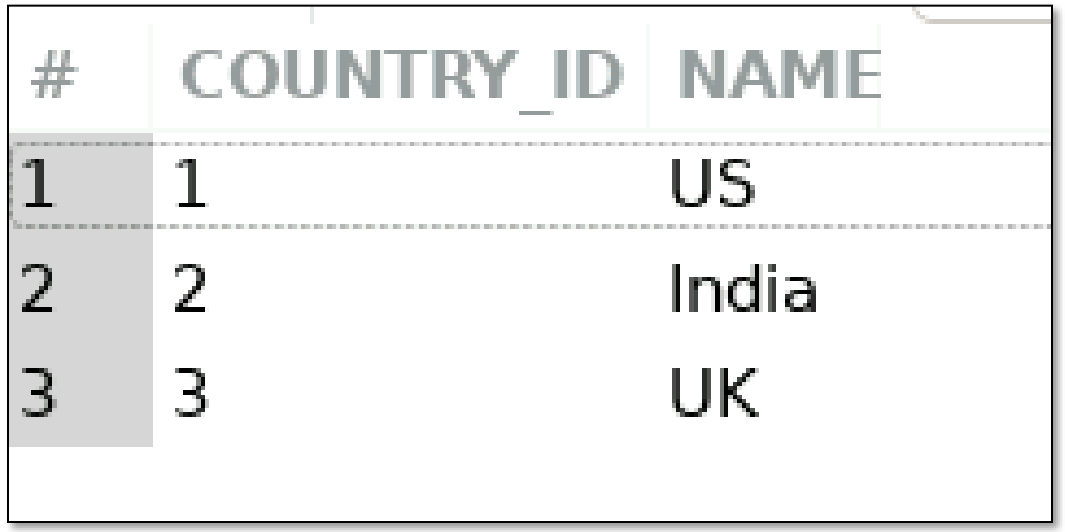
```
SELECT * FROM CLAIM;
SELECT * FROM COUNTRY;
```

Assisted Practice: Common Table Expressions

Output:

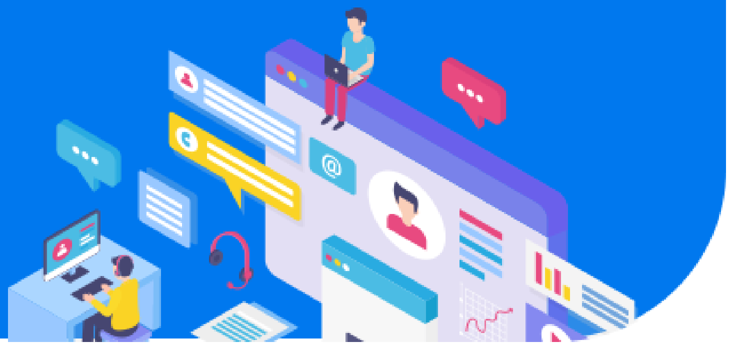


#	CUST_ID	POLICY_NO	COUNTRY_ID	CLAIM_AMT
1	C	PO21	1	1000
2	C2	PO22	1	3200
3	C	PO21	1	1000
4	C2	PO22	1	3200
5	C3	PO33	2	5000
6	C5	PO55	2	1200
7	C7	PO55	2	1200
8	C3	PO33	2	5000
9	C5	PO55	2	1200
10	C7	PO55	2	1200
11	C4	PO44	3	4500



#	COUNTRY_ID	NAME
1	1	US
2	2	India
3	3	UK

Assisted Practice: Common Table Expressions



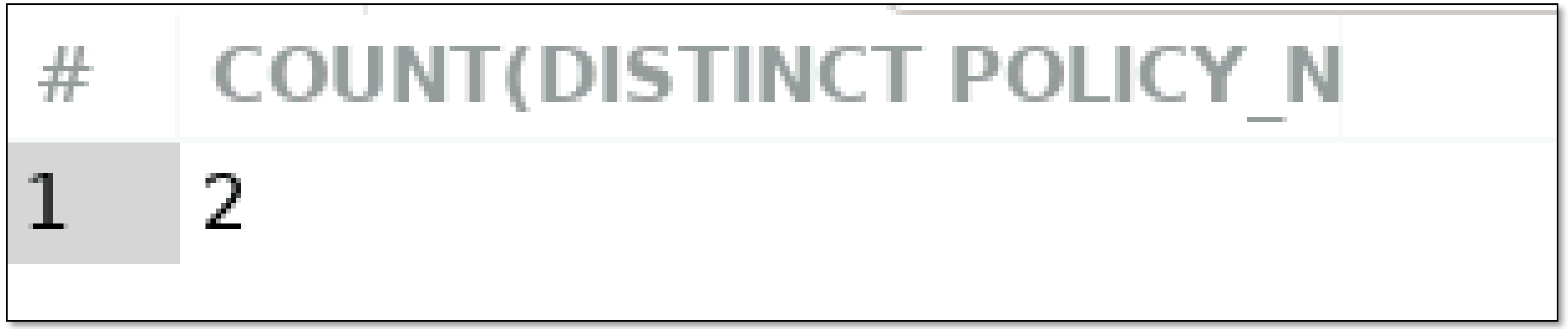
Step 06: Write a query using CTE to fetch the count of unique policy numbers in India

SQL Query

```
WITH India_ID AS
    (SELECT COUNTRY_ID
    FROM COUNTRY
    WHERE NAME = 'India')
SELECT COUNT(DISTINCT POLICY_NO)
FROM CLAIM
WHERE COUNTRY_ID IN (SELECT COUNTRY_ID FROM India_ID)
```

Assisted Practice: Common Table Expressions

Output:



#	COUNT(DISTINCT POLICY_N
1	2

SQL Best Practices

SQL Best Practices

SQL Best Practices

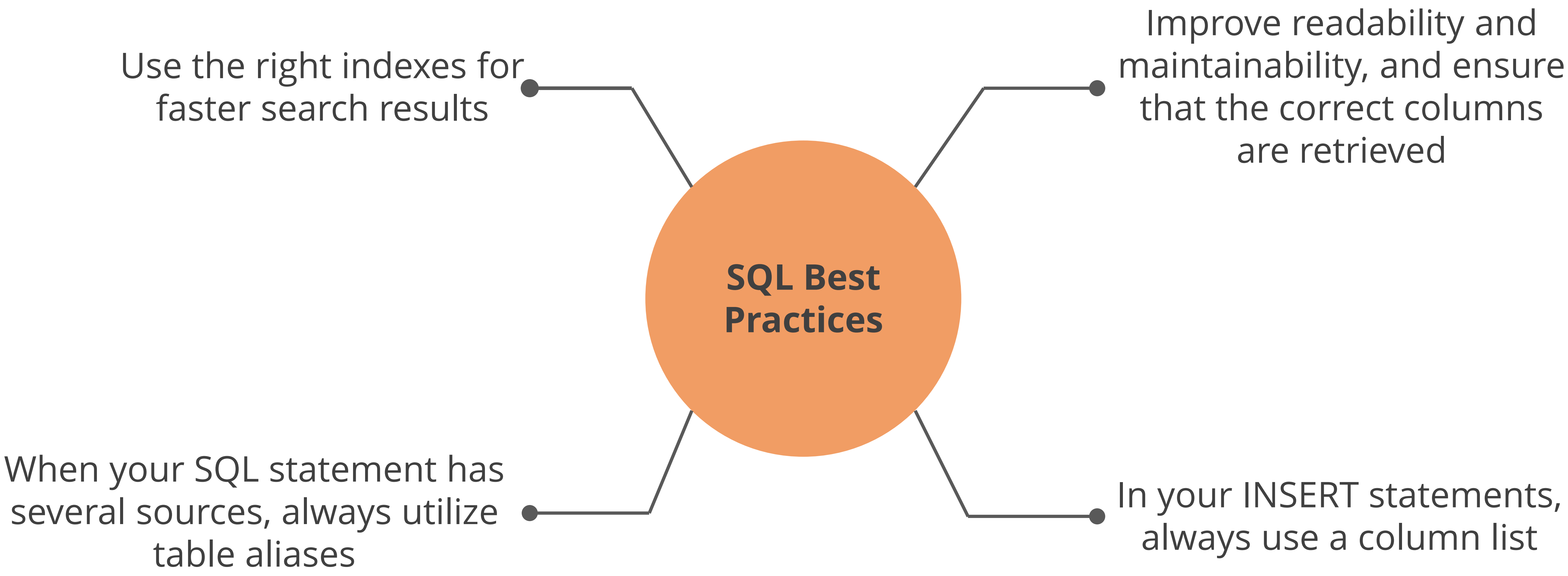
Always check for NULLS in your data

Don’t use a query again if it does not serve its entire function

Always keep an eye on the execution plan and track of the time costs

Avoid sub queries, and do joins or write functions if required

SQL Best Practices



SQL Best Practices

Use the right indexes for faster search results

Improve readability and maintainability, and ensure that the correct columns are retrieved

When your SQL statement has several sources, always utilize table aliases

In your INSERT statements, always use a column list

Knowledge Check

Knowledge Check

1

Which of the following is one of the best practices in SQL?

A. Use the right indexes for slower search results

B. Make subqueries, and do not make joins or write functions

C. Always check for NULLS in your data

D. Use a query again if it does not serve its entire function

Knowledge Check

1

Which of the following is one of the best practices in SQL?

A. Use the right indexes for slower search results

B. Make subqueries, and do not make joins or write functions

C. Always check for NULLS in your data

D. Use a query again if it does not serve its entire function

The correct answer is **C**

Always check for NULLS in your data. This is one of the main SQL practices.

Knowledge Check

2

Which of the following stores Unicode characters?

A. CHAR

B. VARCHAR

C. NVARCHAR

D. None of the above

Knowledge Check

2

Which of the following stores Unicode characters?

A. CHAR

B. VARCHAR

C. NVARCHAR

D. None of the above

The correct answer is **C**

Unicode characters are stored in NVARCHAR.

Knowledge Check

3

Which of the following are index guidelines? Select all that apply.

A. Keeping index keys as short as possible

B. Keeping separate index keys

C. Not keeping indexes up-to-date

D. Keeping a single key for indexes

Knowledge Check

3

Which of the following are index guidelines? Select all that apply.

A. Keeping index keys as short as possible

B. Keeping separate index keys

C. Not keeping indexes up-to-date

D. Keeping a single key for indexes

The correct answer are **A and B**

Index guidelines are keeping separate index keys and making them as short as possible.

Lesson-End Project: Airline Customer Expense Analysis

Problem statement:

You are working for an airline company and this company has organized an SQL hackathon. You have been chosen to represent the LOYALTY team in the hackathon.

Objective:

The challenge is to overcome the constraint of utilizing CTE to assess the time period and customers based on spend patterns.

This analysis will help the company design attractive campaigns for low- and high-spend customers, decide in which quarter deals should be completed, etc.

Lesson-End Project: Airline Customer Expense Analysis

Tasks to be performed:

Step 01: Create a table containing the columns **CustID**, **CustNAME**, **Quarter**, and **TravelSpend** and name it "**customer_spend**"

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

Step 03: Find the total spend across each quarter

Step 04: Find the average spend across all quarters

Lesson-End Project: Airline Customer Expense Analysis

Tasks to be performed:

Step 05: Using CTE, find the specific quarter(s) when the spend was better than the average spend across all quarters

Step 06: Using CTE, find the list of customers who have spent less than the average spend of all customers

Step 07: Using CTE, find the list of customers who have spent more than the average spend of all customers

Key Takeaways

- Clustered index always uses one or more columns for creating an index.
- Choosing the right columns and types for an index is a crucial part of building a useful index.
- A covered query is a query where all the columns in the query's result set are pulled from non-clustered indexes.
- The query execution plan is a list of instructions that the query execution must follow to produce the query result.