**Department of Information Technology**

**Delhi Technological University**

**Database Management System (IT-202)**



Session 2020-21(IT-4th Semester Section-2)

**Submitted By: Submitted To:-**

Name: VARUN KUMAR Ms. Swati Sharda

Roll No.:- 2K19/IT/140

Index

|  |  |  |
| --- | --- | --- |
| **SNo.** | **Title** | **Page No** |
| 1 | Introduction to DML DDL command | 3 |
| 2 | Introduction to constraints | 7 |
| 3 | Lab assignment 1 | 12 |
| 4 | Lab assignment 2 | 17 |
| 5 | Introduction to Aggregate Function | 20 |
| 6 | Lab assignment 3 | 34 |
| 7 | Introduction to joins | 38 |
| 8 | Wildcard, union, intersection | 46 |
| 9 | Transaction (commit, Rollback) | 52 |
| 10 | Trigger | 55 |

**LAB – 1**

Introduction to DDL:

* DDL stands for **Data Definition Language.**
* It is a language used for defining and modifying the data and its structured
* It is used to build and modify the structure of your tables and other objects in the database.

**DDL commands are as follows,**  
**1. CREATE**  
**2. DROP**  
**3. ALTER**  
**4. RENAME**  
**5. TRUNCATE**

* These commands can be used to add, remove or modify tables within a database.
* DDL has pre-defined syntax for describing the data.

1. CREATE COMMAND

* **CREATE command** is used for creating objects in the database.
* It creates a new table.

## 2. DROP COMMAND

* **DROP command** allows to remove entire database objects from the database.
* It removes entire data structure from the database.
* It deletes a table, index or view.

## 3. ALTER COMMAND

* An **ALTER command** allows to alter or modify the structure of the database.
* It modifies an existing database object.
* Using this command, you can add additional column, drop existing column and even change the data type of columns.

## Introduction to DML:

* DML stands for **Data Manipulation Language.**
* It is a language used for selecting, inserting, deleting and updating data in a database.
* It is used to retrieve and manipulate data in a relational database.
* **DDL commands are as follows,**  
  **1. SELECT**  
  **2. INSERT**  
  **3. UPDATE**  
  **4. DELETE**
* DML performs read-only queries of data.

## 1. SELECT COMMAND

* **SELECT command** is used to retrieve data from the database.
* This command allows database users to retrieve the specific information they desire from an operational database.
* It returns a result set of records from one or more tables.

**SELECT Command has many optional clauses are as stated below:**

|  |  |
| --- | --- |
| **Clause** | **Description** |
| WHERE | It specifies which rows to retrieve. |
| GROUP BY | It is used to arrange the data into groups. |
| HAVING | It selects among the groups defined by the GROUP BY clause. |
| ORDER BY | It specifies an order in which to return the rows. |
| AS | It provides an alias which can be used to temporarily rename tables or columns. |

**Syntax:**  
**SELECT \* FROM <table\_name>;**

## 2. INSERT COMMAND

* **INSERT command** is used for inserting a data into a table.
* Using this command, you can add one or more records to any single table in a database.
* It is also used to add records to an existing code.

**Syntax:**  
**INSERT INTO <table\_name> (`column\_name1` <datatype>, `column\_name2` <datatype>, . . . , `column\_name\_n` <database>) VALUES (`value1`, `value2`, . . . , `value n`);**  
  
**3. UPDATE COMMAND**

* **UPDATE command** is used to modify the records present in existing table.
* This command updates existing data within a table.
* It changes the data of one or more records in a table.

**Syntax:**  
**UPDATE <table\_name>**  
**SET <column\_name = value>**  
**WHERE condition;**

## 4. DELETE COMMAND

* **DELETE command** is used to delete some or all records from the existing table.
* It deletes all the records from a table.
* **Syntax:**  
  **DELETE FROM <table\_name> WHERE <condition>;**  
  **If we** does not write the WHERE condition, then all rows will get deleted.

CODE :

mysql> create table CollegeId

-> (firstName varchar(30) NOT NULL,

-> lastName varchar(30) NOT NULL,

-> Roll\_No int,

-> Branch varchar(10));

mysql> insert into CollegeId(firstName,lastName,Roll\_No,Branch)

-> values(“Naveen”,”Yadav”,89,”IT”),

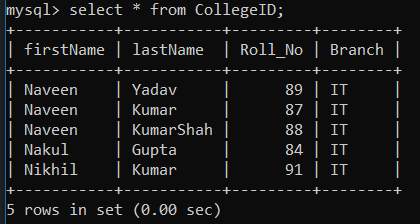
->("Naveen","Kumar",87,"IT"),

-> ("Naveen","KumarShah",88,"IT"),

-> ("Nakul","Gupta",84,"IT"),

-> ("Nikhil","Kumar",91,"IT");

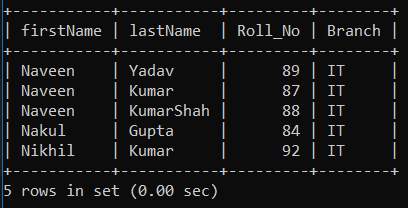
mysql> select \* from CollegeID;



mysql> update CollegeId

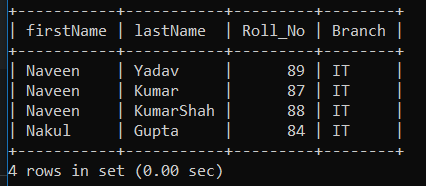
-> set Roll\_No=92

-> where Roll\_no=91;



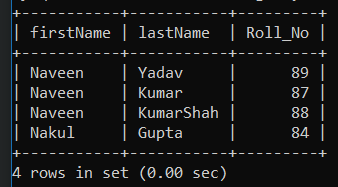
mysql> delete from CollegeID

-> where Roll\_No=92;



alter table CollegeId

-> drop column Branch;



drop table CollegeID;



**LAB – 2**

Aim: To create Table queries using the following constraints

\* Primary Key constraint \* Foreign Key constraint \*Check Constraint \* Unique Constraint \* Not null constraint.

## SQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

* [**NOT NULL**](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [**UNIQUE**](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [**PRIMARY KEY**](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [**FOREIGN KEY**](https://www.w3schools.com/sql/sql_foreignkey.asp) - Uniquely identifies a row/record in another table
* [**CHECK**](https://www.w3schools.com/sql/sql_check.asp) - Ensures that all values in a column satisfies a specific condition
* [**DEFAULT**](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column when no value is specified
* [**INDEX**](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quick

## SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

## SQL FOREIGN KEY Constraint

A FOREIGN KEY is a key used to link two tables together.

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

## SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

SQL NOT NULL Constraint

By default, a column can hold NULL values.

The NOT NULL constraint enforces a column to NOT accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

## SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different.

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

CODE :

create table CollegeId

-> (firstName varchar(30) NOT NULL,

-> lastName varchar(30) NOT NULL,

-> Roll\_No int,

-> Branch varchar(10));

insert into CollegeId(firstName,lastName,Roll\_No,Branch)

-> values("Naveen","Yadav",89,"IT"),

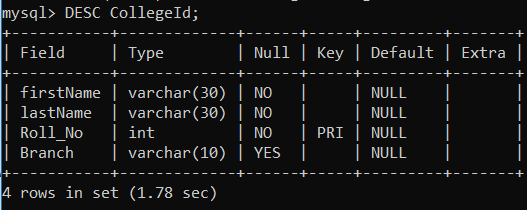
-> ("Naveen","Kumar",87,"IT"),

-> ("Naveen","KumarShah",88,"IT"),

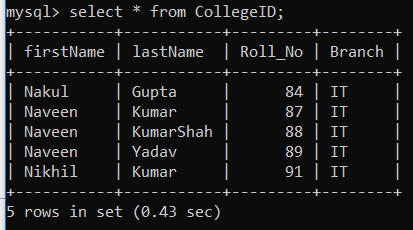
-> ("Nakul","Gupta",84,"IT"),

-> ("Nikhil","Kumar",91,"IT");

desc CollegeId

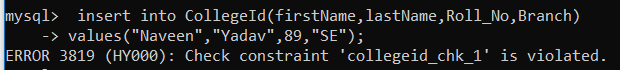


select \* from CollegeID;



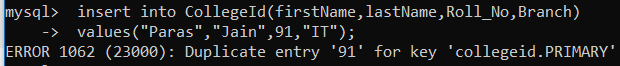
insert into CollegeId(firstName,lastName,Roll\_No,Branch)

-> values("Naveen","Yadav",89,"SE");



insert into CollegeId(firstName,lastName,Roll\_No,Branch)

-> values("Paras","Jain",91,"IT");



insert into CollegeId(firstName,lastName,Roll\_No,Branch)

-> values("Paras",,93,"IT");



create table Teacher

-> (firstName varchar(30) NOT NULL,

-> lastName varchar(30) NOT NULL,

-> Subject varchar(10),

-> TeacherID int UNIQUE,

-> Roll\_No int,

-> PRIMARY KEY (TeacherID),

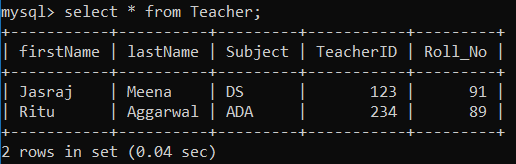
-> FOREIGN KEY (Roll\_No) REFERENCES CollegeId(Roll\_No));

mysql> insert into Teacher(firstName,lastName,Subject,TeacherID,Roll\_No)

-> values("Jasraj","Meena","DS",123,91),

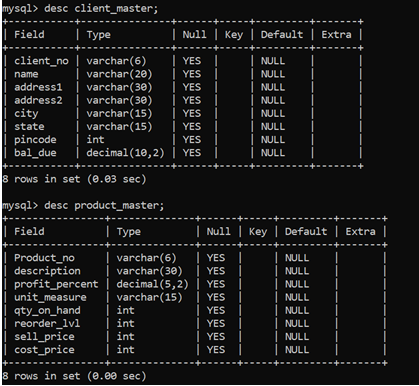
-> ("Ritu","Aggarwal","ADA",234,89);

mysql> select \* from Teacher;

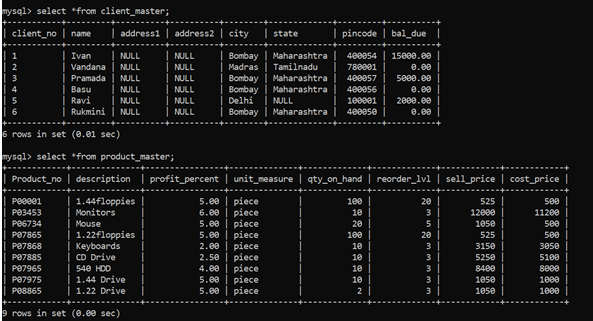


**Assignment - 1**

**Created client\_master and product\_master table:**

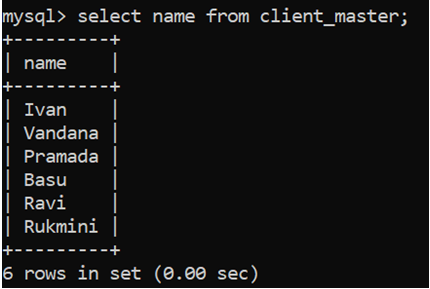


**Inserted data into Client\_master and Product\_master table:**

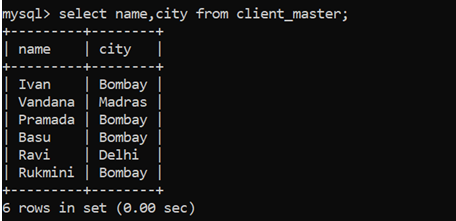


**Solutions of the questions =>**

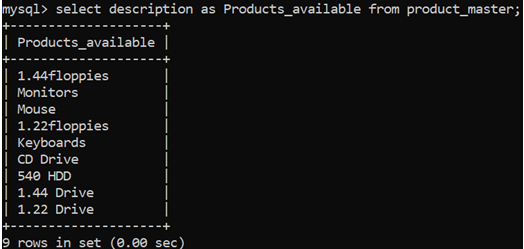
1. **Find out the names of all the clients.**



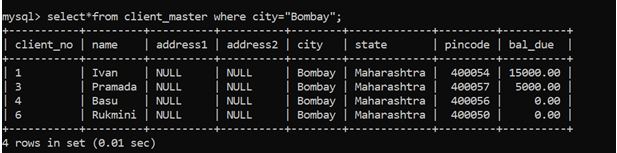
1. **Retrieve the list of names and cities of all the clients.**



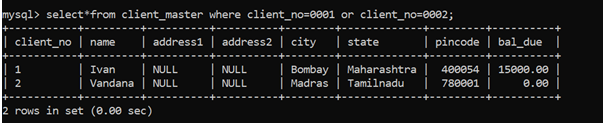
1. **List the various products available from the product\_master table.**



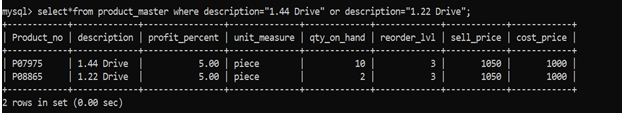
1. **List all the clients who are located in Bombay.**



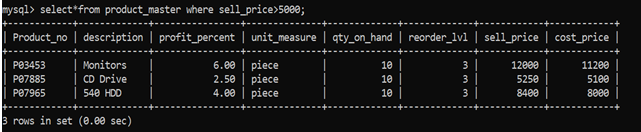
1. **Display the information for client no 0001 and 0002.**



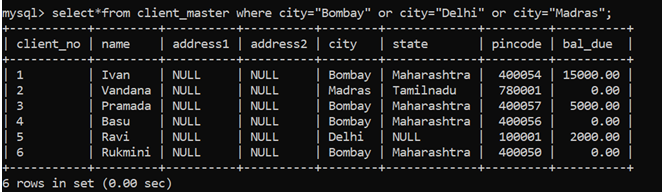
1. **Find the products with description as ‘1.44 Drive’ and ‘1.22 Drive’.**



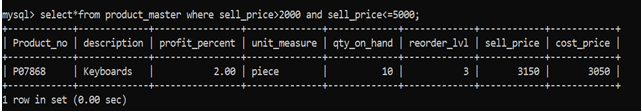
1. **Find all the products whose sell price is greater than 5000.**



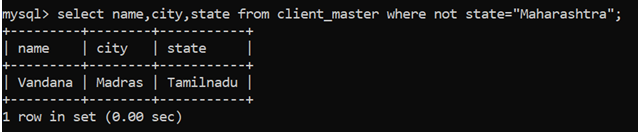
1. **Find the list of all clients who stay in in city ‘Bombay’ or city ‘Delhi’ or ‘Madras’.**



1. **Find the product whose selling price is greater than 2000 and less than or equal to 5000.**



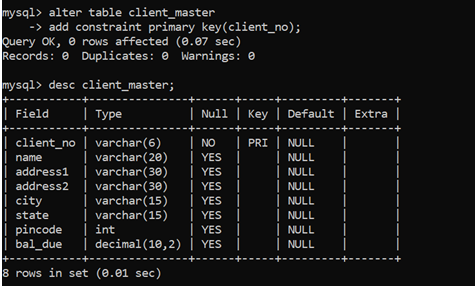
1. **List the name, city and state of clients not in the state of ‘Maharashtra’.**



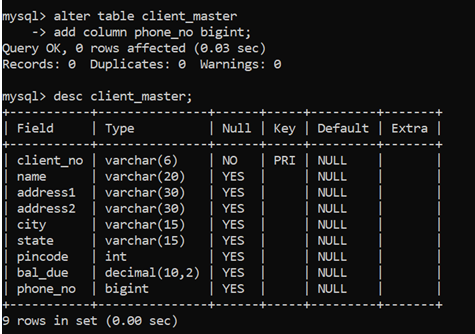
**Assignment -2**

**Objective – Answer the following Questions**

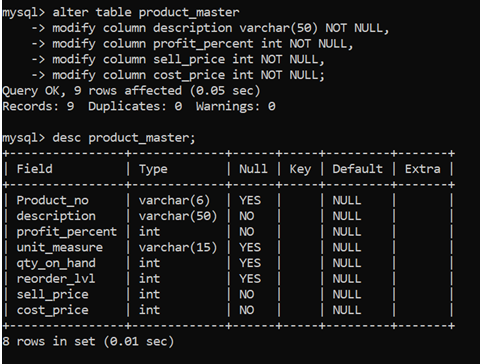
**Q1. Make the primary key to client\_no in client\_master**.



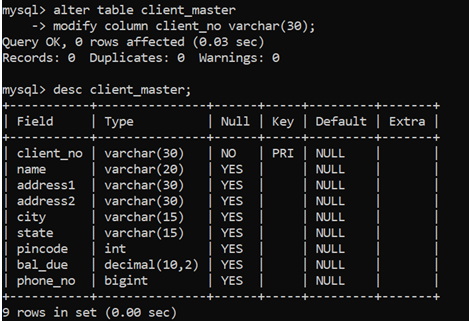
**Q2. Add a new column phone\_no in the client\_master table.**



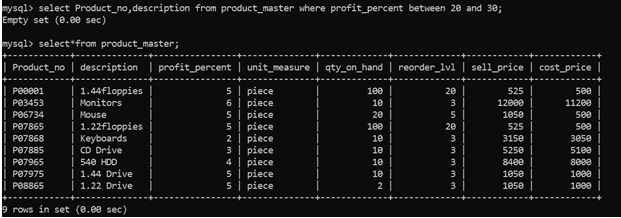
**Q3. Add the not null constraint in the product\_master table with the columns description, profit percent , sell price and cost price.**



**Q4. Change the size of client\_no field in the client\_master table.**



**Q5. Select product\_no, description where profit percent is between 20 and 30 both inclusive.**



**As no value matches the given constraint of profit\_percent between 20 and 30 hence the output is an empty set**

**LAB – 3**

**AGGREGATE FUNCTIONS**

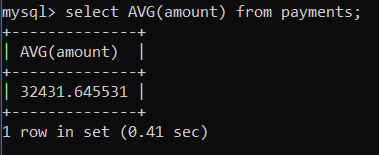
COUNT – counts the number of elements in the group defined

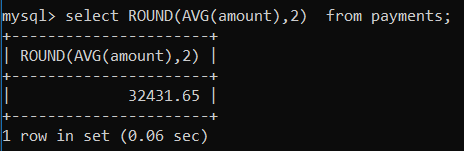
SUM – calculates the sum of the given attribute/expression in the group defined

AVG – calculates the average value of the given attribute/expression in the group defined

MIN – finds the minimum in the group defined

MAX – finds the maximum in the group defined





mysql> select customerNumber, Round(AVG(amount),2) from payments group by customerNumber;

+----------------+----------------------+

| customerNumber | Round(AVG(amount),2) |

+----------------+----------------------+

| 103 | 7438.12 |

| 112 | 26726.99 |

| 114 | 45146.27 |

| 119 | 38983.23 |

| 121 | 26056.20 |

| 124 | 64909.80 |

| 128 | 18984.44 |

| 129 | 22236.85 |

| 131 | 35879.98 |

| 141 | 55056.84 |

| 144 | 21840.33 |

| 145 | 26861.63 |

| 146 | 43435.12 |

| 148 | 39062.76 |

| 151 | 44478.49 |

| 157 | 49254.63 |

| 161 | 26136.31 |

| 166 | 35140.19 |

| 167 | 48781.24 |

| 171 | 30890.85 |

| 172 | 28851.17 |

| 173 | 16099.35 |

| 175 | 31808.21 |

| 177 | 31180.61 |

| 181 | 24165.88 |

| 186 | 31848.82 |

| 187 | 49470.03 |

| 189 | 24949.14 |

| 198 | 7184.75 |

| 201 | 30583.59 |

| 202 | 35061.10 |

| 204 | 27788.63 |

| 205 | 31267.77 |

| 209 | 25286.44 |

| 211 | 45480.79 |

| 216 | 22840.16 |

| 219 | 3959.30 |

| 227 | 44954.90 |

| 233 | 22992.56 |

| 239 | 80375.24 |

| 240 | 35891.88 |

| 242 | 20161.12 |

| 249 | 41111.62 |

| 250 | 22553.06 |

| 256 | 29438.21 |

| 259 | 44611.57 |

| 260 | 33406.00 |

| 276 | 34258.56 |

| 278 | 42509.90 |

| 282 | 30551.87 |

| 286 | 45272.69 |

| 298 | 54388.96 |

| 299 | 34529.52 |

| 311 | 31902.05 |

| 314 | 31126.93 |

| 319 | 39216.08 |

| 320 | 33957.51 |

| 321 | 66170.39 |

| 323 | 38655.52 |

| 324 | 26852.24 |

| 328 | 19140.76 |

| 333 | 18396.72 |

| 334 | 34632.25 |

| 339 | 28969.67 |

| 344 | 23375.57 |

| 347 | 20753.10 |

| 350 | 23849.18 |

| 353 | 31745.80 |

| 357 | 28331.19 |

| 362 | 16766.74 |

| 363 | 38816.43 |

| 379 | 24511.22 |

| 381 | 7304.30 |

| 382 | 28353.33 |

| 385 | 29156.10 |

| 386 | 45071.66 |

| 398 | 26387.18 |

| 406 | 28812.32 |

| 412 | 33352.47 |

| 415 | 31310.09 |

| 424 | 23071.44 |

| 447 | 16655.93 |

| 448 | 38388.22 |

| 450 | 59551.38 |

| 452 | 17020.00 |

| 455 | 35189.33 |

| 456 | 14615.22 |

| 458 | 37480.03 |

| 462 | 29542.50 |

| 471 | 22460.38 |

| 473 | 12679.16 |

| 475 | 21874.36 |

| 484 | 25493.93 |

| 486 | 25908.86 |

| 487 | 21285.19 |

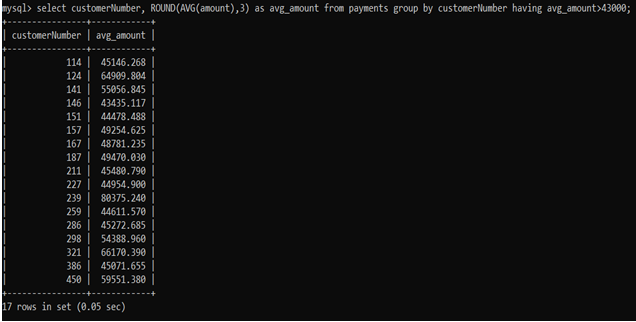
| 489 | 14793.08 |

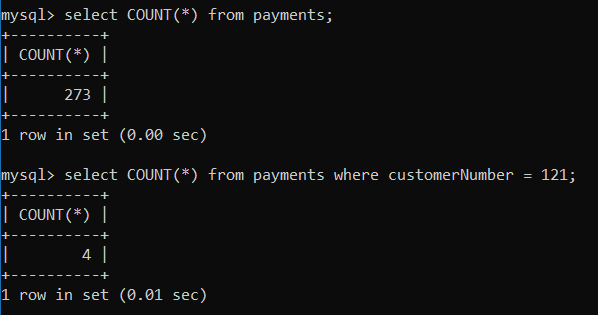
| 495 | 32770.87 |

| 496 | 38165.73 |

+----------------+----------------------+

98 rows in set (0.11 sec)





mysql> select customerNumber, COUNT(\*) as count from payments group by customerNumber;

+----------------+-------+

| customerNumber | count |

+----------------+-------+

| 103 | 3 |

| 112 | 3 |

| 114 | 4 |

| 119 | 3 |

| 121 | 4 |

| 124 | 9 |

| 128 | 4 |

| 129 | 3 |

| 131 | 3 |

| 141 | 13 |

| 144 | 2 |

| 145 | 4 |

| 146 | 3 |

| 148 | 4 |

| 151 | 4 |

| 157 | 2 |

| 161 | 4 |

| 166 | 3 |

| 167 | 2 |

| 171 | 2 |

| 172 | 3 |

| 173 | 2 |

| 175 | 3 |

| 177 | 2 |

| 181 | 3 |

| 186 | 3 |

| 187 | 3 |

| 189 | 2 |

| 198 | 3 |

| 201 | 2 |

| 202 | 2 |

| 204 | 2 |

| 205 | 3 |

| 209 | 3 |

| 211 | 1 |

| 216 | 3 |

| 219 | 2 |

| 227 | 2 |

| 233 | 3 |

| 239 | 1 |

| 240 | 2 |

| 242 | 3 |

| 249 | 2 |

| 250 | 3 |

| 256 | 2 |

| 259 | 2 |

| 260 | 2 |

| 276 | 4 |

| 278 | 3 |

| 282 | 3 |

| 286 | 2 |

| 298 | 2 |

| 299 | 2 |

| 311 | 3 |

| 314 | 2 |

| 319 | 2 |

| 320 | 3 |

| 321 | 2 |

| 323 | 4 |

| 324 | 3 |

| 328 | 2 |

| 333 | 3 |

| 334 | 3 |

| 339 | 2 |

| 344 | 2 |

| 347 | 2 |

| 350 | 3 |

| 353 | 4 |

| 357 | 2 |

| 362 | 2 |

| 363 | 3 |

| 379 | 3 |

| 381 | 4 |

| 382 | 3 |

| 385 | 3 |

| 386 | 2 |

| 398 | 4 |

| 406 | 3 |

| 412 | 2 |

| 415 | 1 |

| 424 | 3 |

| 447 | 3 |

| 448 | 2 |

| 450 | 1 |

| 452 | 3 |

| 455 | 2 |

| 456 | 2 |

| 458 | 3 |

| 462 | 3 |

| 471 | 2 |

| 473 | 2 |

| 475 | 2 |

| 484 | 2 |

| 486 | 3 |

| 487 | 2 |

| 489 | 2 |

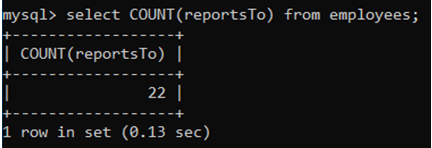
| 495 | 2 |

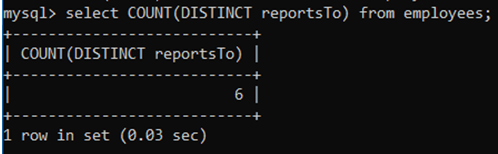
| 496 | 3 |

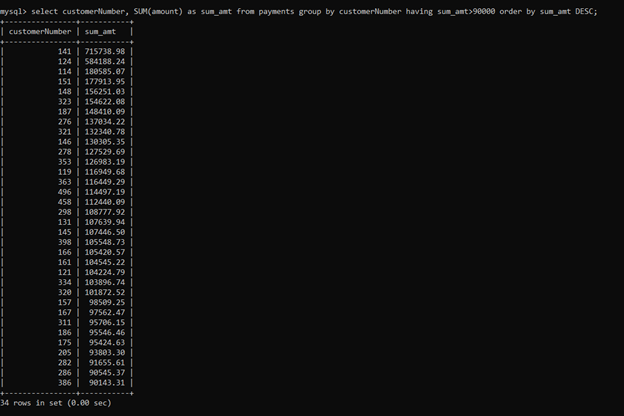
+----------------+-------+

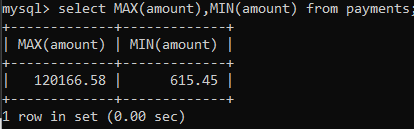
98 rows in set (0.00 sec)











mysql> select customerNumber, MAX(amount) from payments group by customerNumber;

+----------------+-------------+

| customerNumber | MAX(amount) |

+----------------+-------------+

| 103 | 14571.44 |

| 112 | 33347.88 |

| 114 | 82261.22 |

| 119 | 49523.67 |

| 121 | 50218.95 |

| 124 | 111654.40 |

| 128 | 33820.62 |

| 129 | 26248.78 |

| 131 | 50025.35 |

| 141 | 120166.58 |

| 144 | 36005.71 |

| 145 | 53959.21 |

| 146 | 49614.72 |

| 148 | 105743.00 |

| 151 | 58841.35 |

| 157 | 63357.13 |

| 161 | 50743.65 |

| 166 | 44160.92 |

| 167 | 85024.46 |

| 171 | 42783.81 |

| 172 | 51209.58 |

| 173 | 20355.24 |

| 175 | 42044.77 |

| 177 | 47177.59 |

| 181 | 44400.50 |

| 186 | 37602.48 |

| 187 | 52825.29 |

| 189 | 32538.74 |

| 198 | 9658.74 |

| 201 | 37258.94 |

| 202 | 36527.61 |

| 204 | 51152.86 |

| 205 | 50342.74 |

| 209 | 36069.26 |

| 211 | 45480.79 |

| 216 | 40473.86 |

| 219 | 4465.85 |

| 227 | 53745.34 |

| 233 | 29070.38 |

| 239 | 80375.24 |

| 240 | 46788.14 |

| 242 | 33818.34 |

| 249 | 48298.99 |

| 250 | 26311.63 |

| 256 | 53116.99 |

| 259 | 61234.67 |

| 260 | 37527.58 |

| 276 | 41554.73 |

| 278 | 52151.81 |

| 282 | 35806.73 |

| 286 | 47411.33 |

| 298 | 61402.00 |

| 299 | 36798.88 |

| 311 | 46770.52 |

| 314 | 45352.47 |

| 319 | 42339.76 |

| 320 | 52548.49 |

| 321 | 85559.12 |

| 323 | 75020.13 |

| 324 | 37455.77 |

| 328 | 31102.85 |

| 333 | 23936.53 |

| 334 | 45785.34 |

| 339 | 34606.28 |

| 344 | 31428.21 |

| 347 | 21053.69 |

| 350 | 50824.66 |

| 353 | 49705.52 |

| 357 | 36442.34 |

| 362 | 18473.71 |

| 363 | 55425.77 |

| 379 | 32680.31 |

| 381 | 14379.90 |

| 382 | 42813.83 |

| 385 | 51001.22 |

| 386 | 51619.02 |

| 398 | 48927.64 |

| 406 | 49165.16 |

| 412 | 35034.57 |

| 415 | 31310.09 |

| 424 | 25505.98 |

| 447 | 26304.13 |

| 448 | 48809.90 |

| 450 | 59551.38 |

| 452 | 27121.90 |

| 455 | 38139.18 |

| 456 | 27550.51 |

| 458 | 57131.92 |

| 462 | 48355.87 |

| 471 | 35505.63 |

| 473 | 17746.26 |

| 475 | 36070.47 |

| 484 | 47513.19 |

| 486 | 45994.07 |

| 487 | 29997.09 |

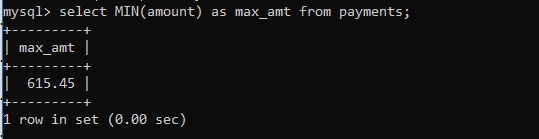
| 489 | 22275.73 |

| 495 | 59265.14 |

| 496 | 52166.00 |

+----------------+-------------+

98 rows in set (0.00 sec)

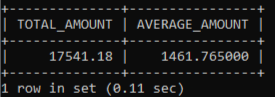


**Assignment -3**

***QUES 1) Write a SQL statement to find the total and average purchase amount of all orders.***

*ANS 1 - SELECT SUM (purch\_amt), AVG(purch\_amt)*

*FROM orders;*



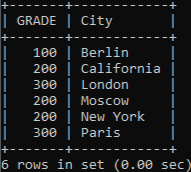
***QUES 2) Write a SQL statement which selects the highest grade for each of the cities of the customers***

*.*

*ANS 2 - SELECT city,MAX(grade)*

*FROM customer*

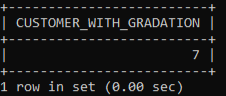
*GROUP BY city;*



***QUES 3) Write a SQL statement find the number of customers who gets at least a gradation for his/her performance.***

*ANS 3 - SELECT COUNT ( ALL grade )*

*FROM customer;*

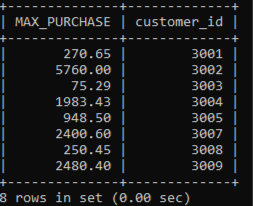


***QUES 4) Write a SQL statement to find the highest purchase amount ordered by the each customer with their ID and highest purchase amount.***

*ANS 4 - SELECT customer\_id,MAX(purch\_amt)*

*FROM orders*

*GROUP BY customer\_id;*

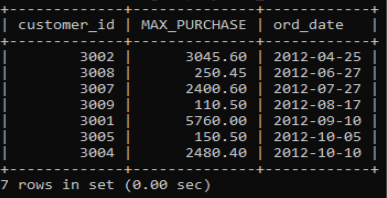


***QUES 5) Write a SQL statement to find the highest purchase amount ordered by the each customer on a particular date with their ID, order date and highest purchase amount.***

*ANS 5 - SELECT customer\_id,ord\_date,MAX(purch\_amt)*

*FROM orders*

*GROUP BY customer\_id,ord\_date;*



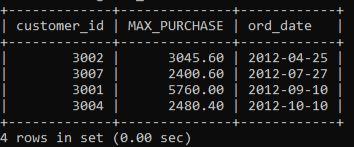
***QUES 6) Write a SQL statement to find the highest purchase amount with their ID and order date, for those customers who have a higher purchase amount in a day is within the range 2000 and 6000.***

*ANS 6 - SELECT customer\_id,ord\_date,MAX(purch\_amt)*

*FROM orders*

*GROUP BY customer\_id,ord\_date*

*HAVING MAX(purch\_amt) BETWEEN 2000 AND 6000;*



***QUES 7) Write a SQL statement to display customer details (ID and purchase amount) whose IDs are within the range 3002 and 3007 and highest purchase amount is more than 1000.***

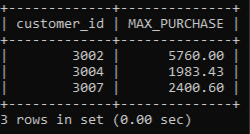
*ANS 7 - SELECT customer\_id,MAX(purch\_amt)*

*FROM orders*

*WHERE customer\_id BETWEEN 3002 and 3007*

*GROUP BY customer\_id*

*HAVING MAX(purch\_amt)>1000;*



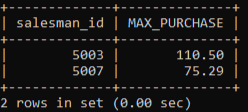
***QUES 8) Write a SQL statement to find the highest purchase amount on a date '2012-08-17' for each salesman with their ID.***

*ANS 8 - SELECT salesman\_id,MAX(purch\_amt)*

*FROM orders*

*WHERE ord\_date = '2012-08-17'*

*GROUP BY salesman\_id;*

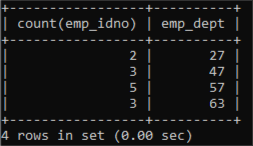


***QUES 10) Write a query in SQL to find the number of employees in each department along with the department code.***

*ANS 10 – SELECT emp\_dept, COUNT(\*)*

*FROM emp\_details*

*GROUP BY emp\_dept;*



**LAB – 4**

**SQL Join :**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

**Types of joins :**

a) **Inner Join:** The INNER JOIN keyword selects records that have matching values in both tables.

Syntax:

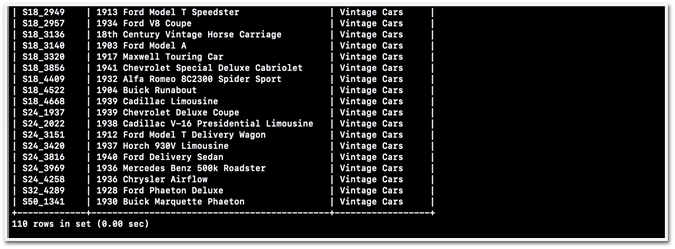
SELECT column\_name(s)

FROM table1

INNER JOIN table2

ON table1.column\_name = table2.column\_name;





b) **Left Join :**  The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

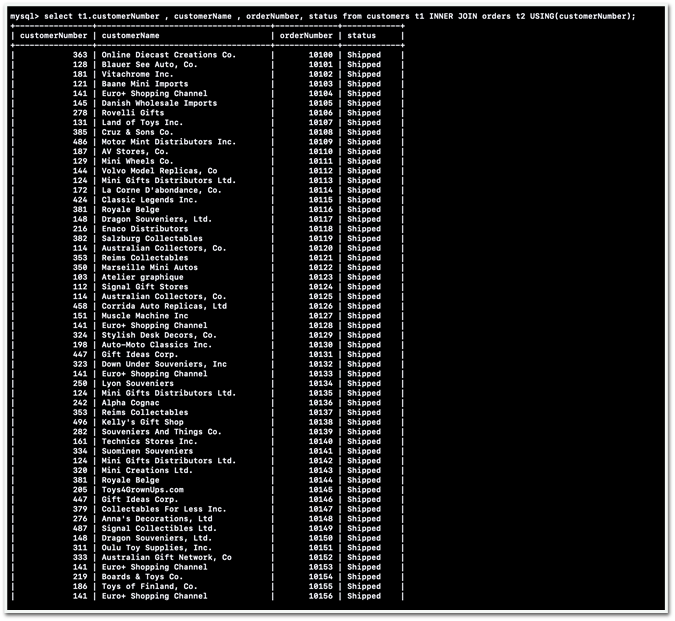
Syntax:

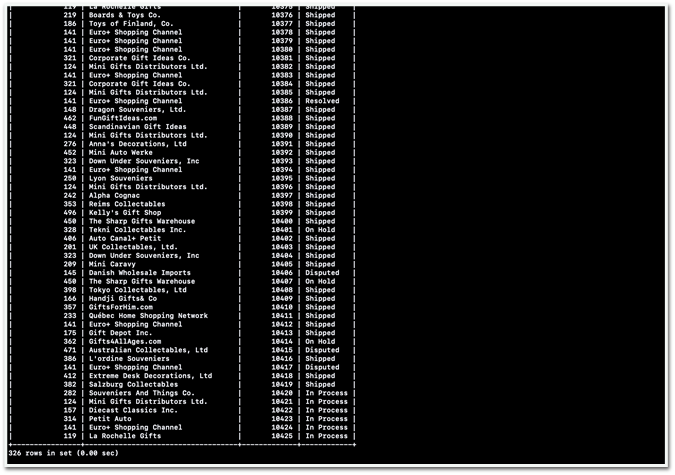
SELECT column\_name(s)

FROM table1

LEFT JOIN table2

ON table1.column\_name = table2.column\_name;





c) **Right Join :** The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

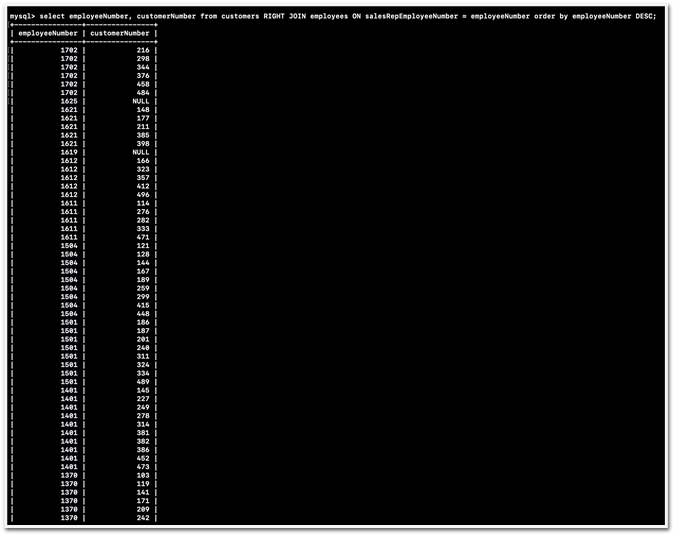
Syntax:

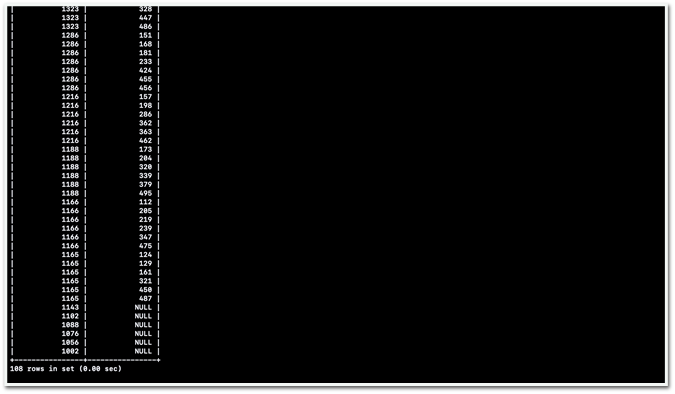
SELECT column\_name(s)

FROM table1

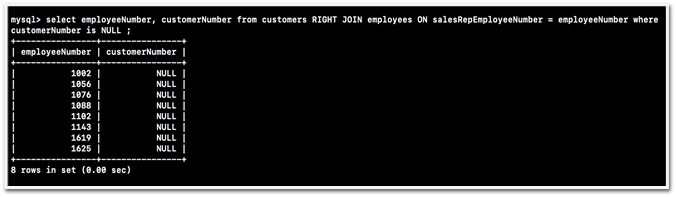
RIGHT JOIN table2

ON table1.column\_name = table2.column\_name;



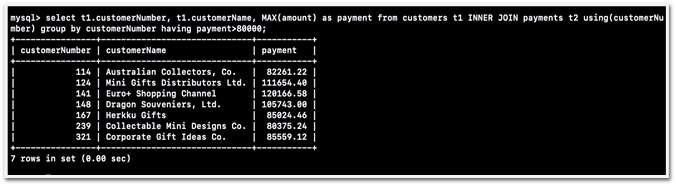


Some Other queries:

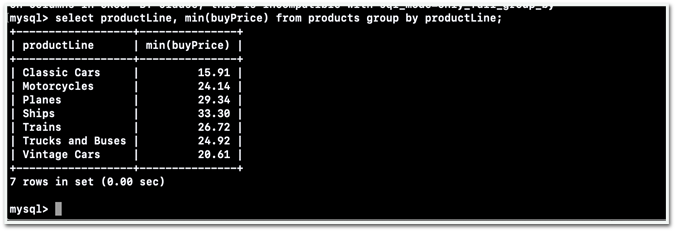


—> display employeeNumber , customerNumber by joining customer and employee table where customer number is null

—> largest payment of each customer (display customer name , customer number, amount)



then apply condition to display only those customers having amount >80000



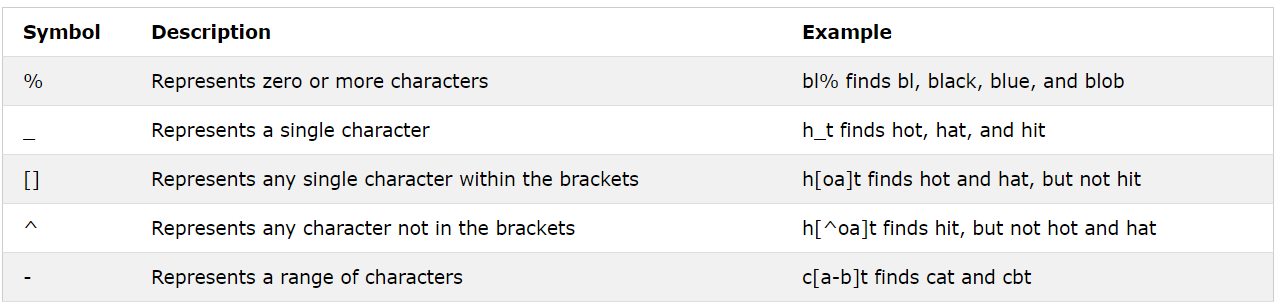
—> select lowest priced product in every product line

**LAB – 5**

**Wildcard**

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the [LIKE](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.



**Union**

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

To use this UNION clause, each SELECT statement must have

* The same number of columns selected
* The same number of column expressions
* The same data type and
* Have them in the same order

But they need not have to be in the same length.

## Syntax

The basic syntax of a **UNION** clause is as follows −

SELECT column1 [, column2 ]FROM table1 [, table2 ][WHERE condition]UNIONSELECT column1 [, column2 ]FROM table1 [, table2 ][WHERE condition]

**INTERSECTION**

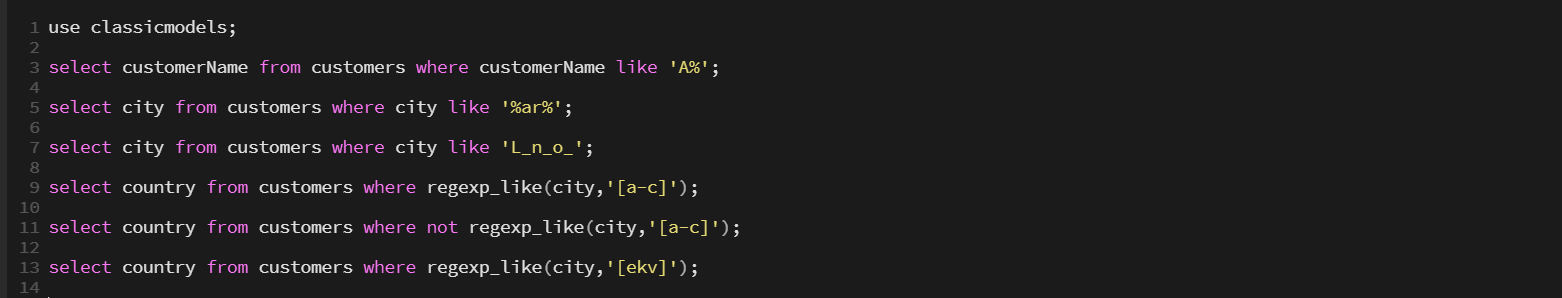
The SQL INTERSECT clause/operator is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. This means INTERSECT returns only common rows returned by the two SELECT statements.

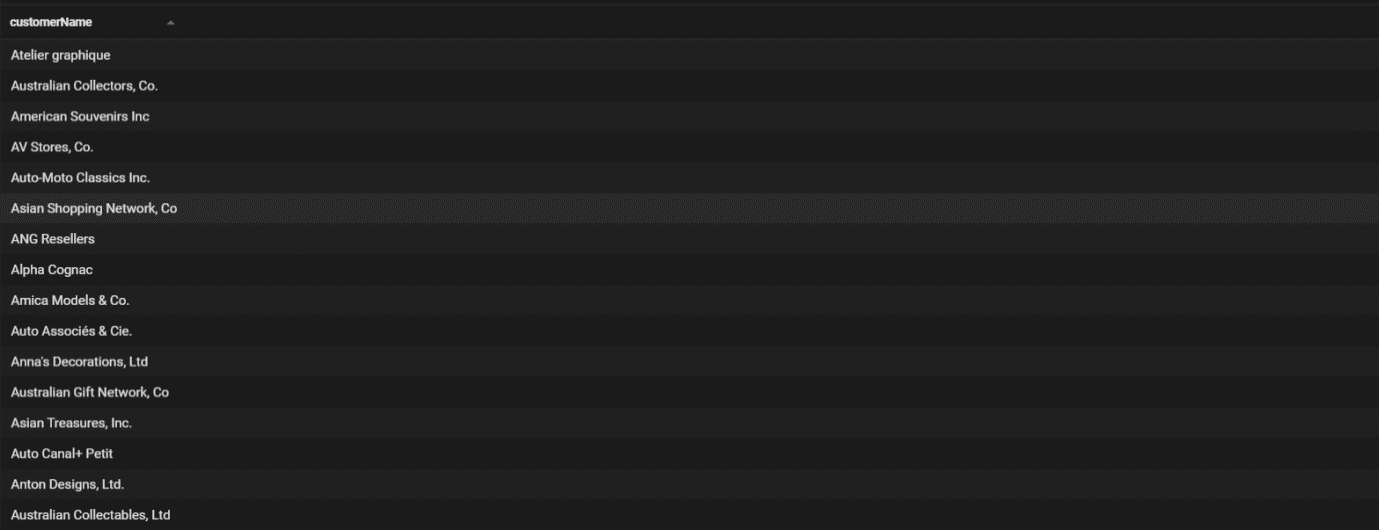
## Syntax

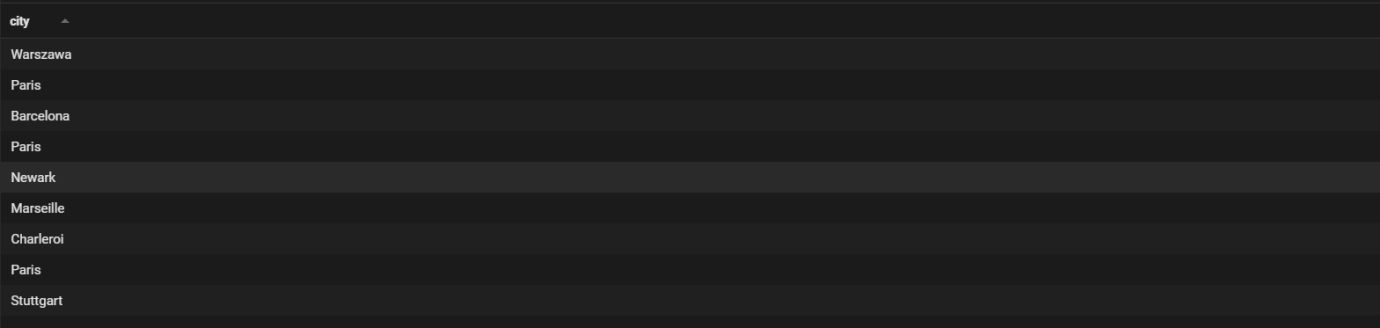
The basic syntax of **INTERSECT** is as follows.

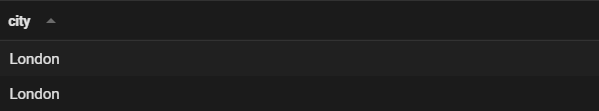
SELECT column1 [, column2 ]FROM table1 [, table2 ][WHERE condition]INTERSECTSELECT column1 [, column2 ]FROM table1 [, table2 ][WHERE condition]

**Wildcards**

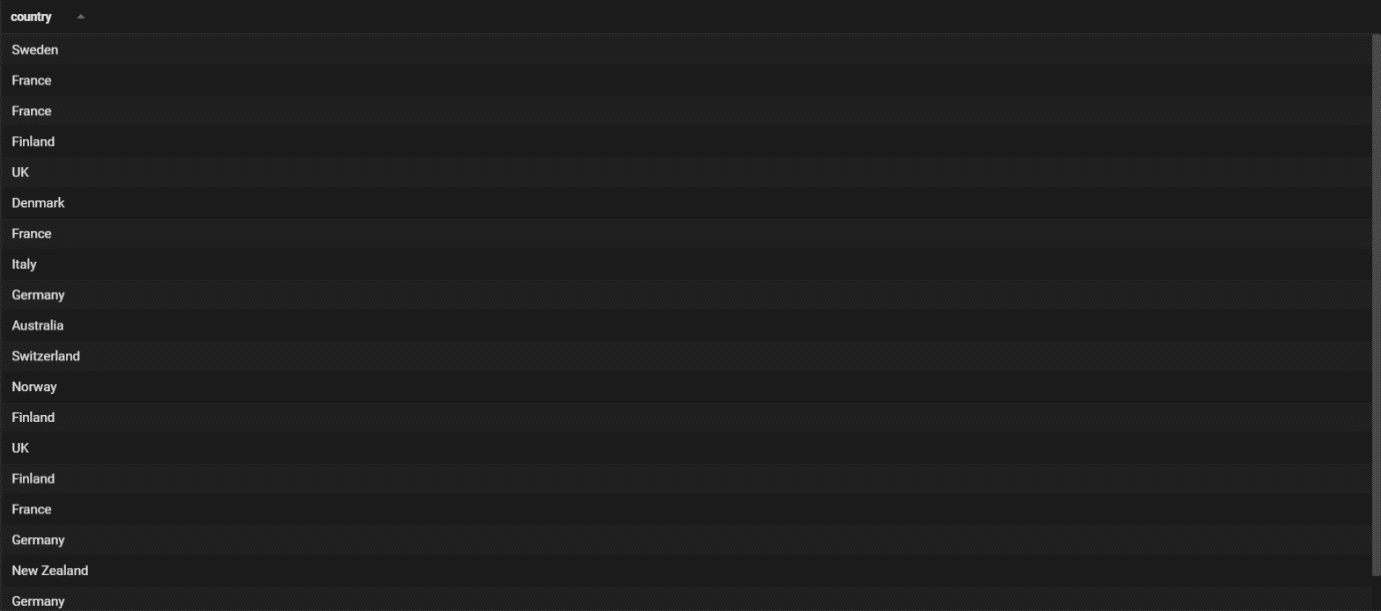


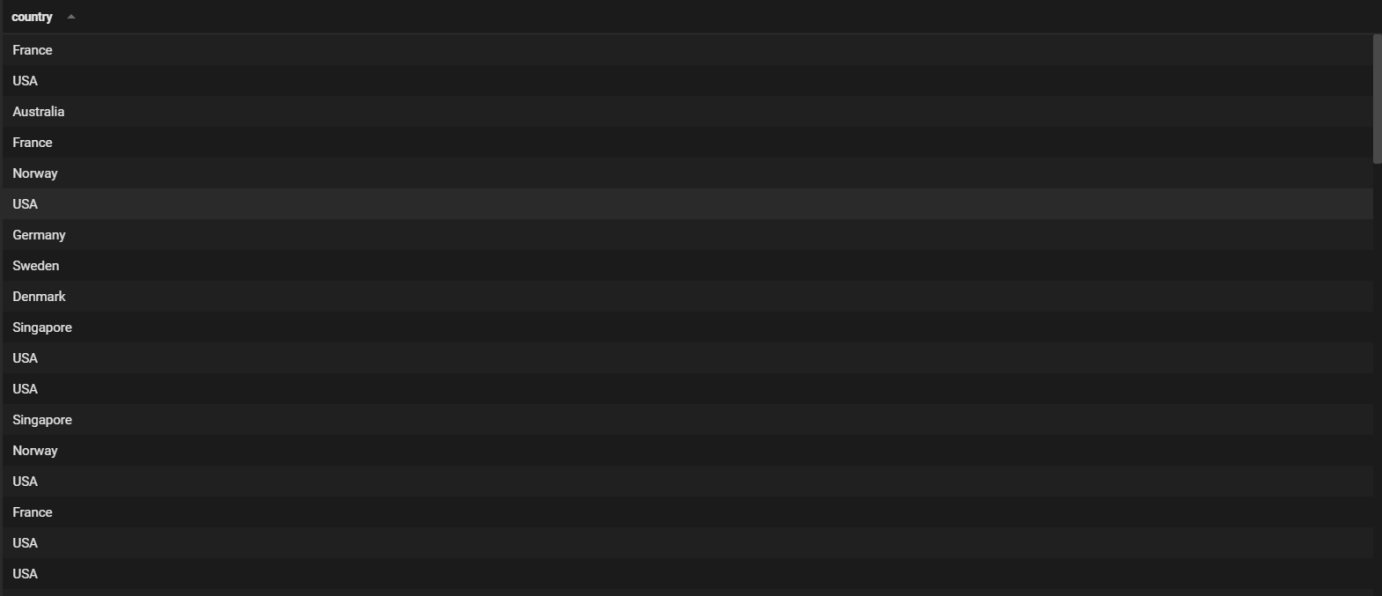






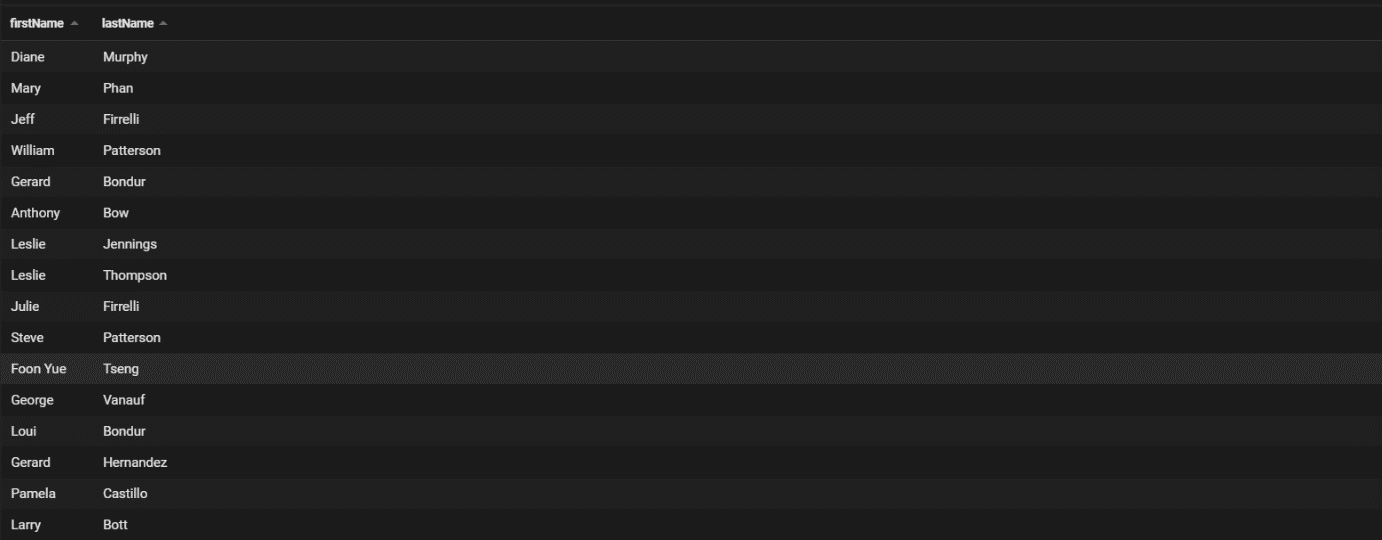


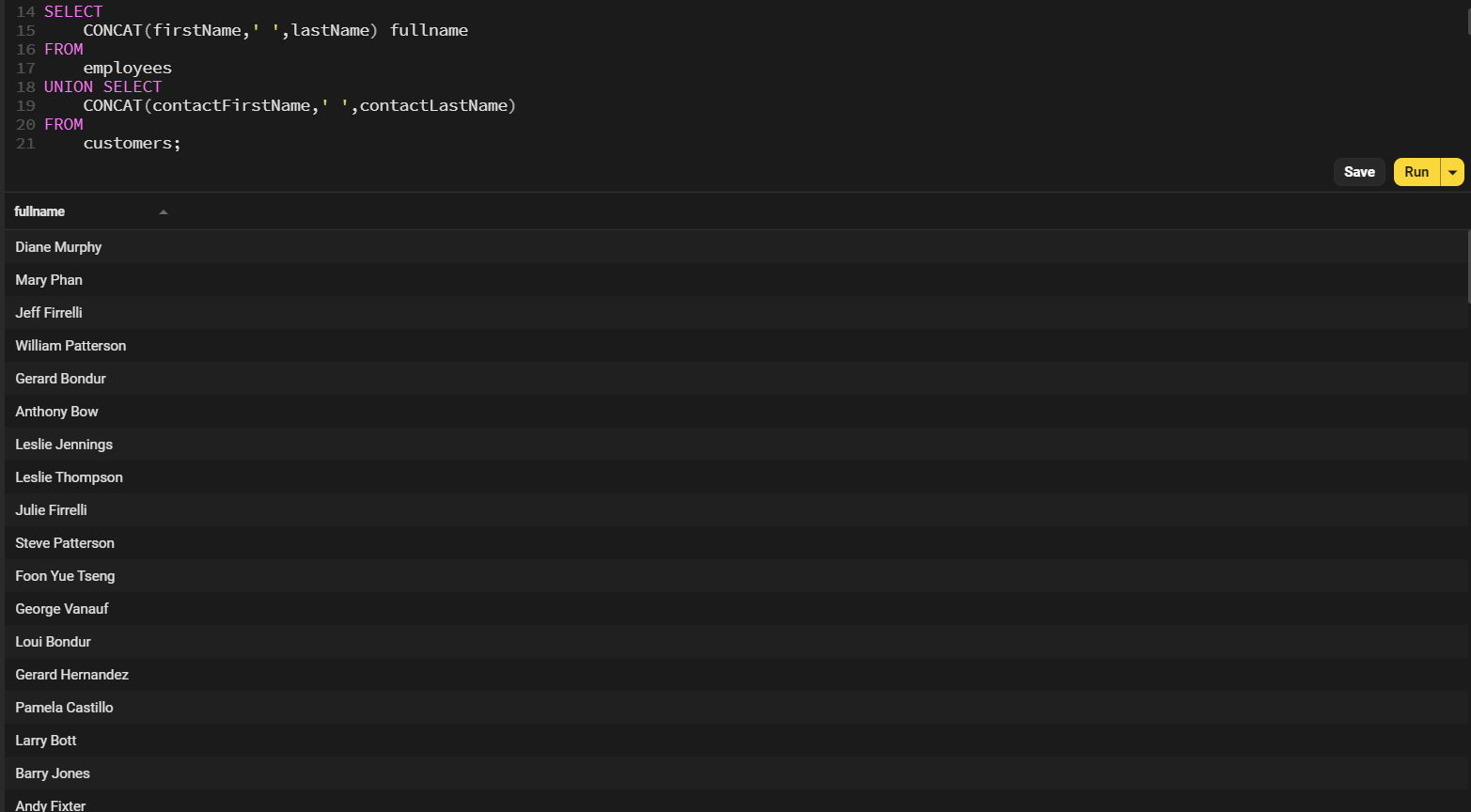


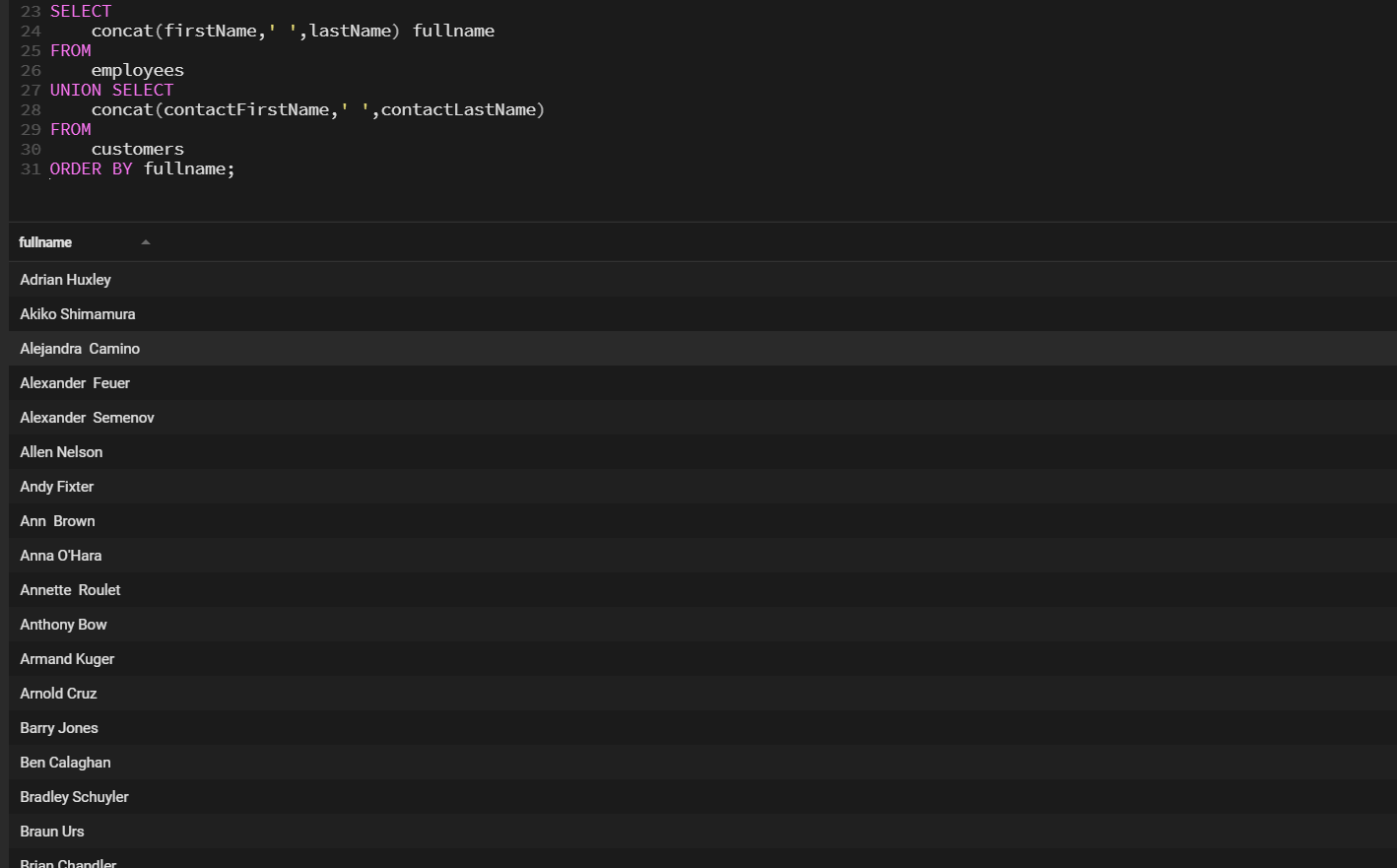


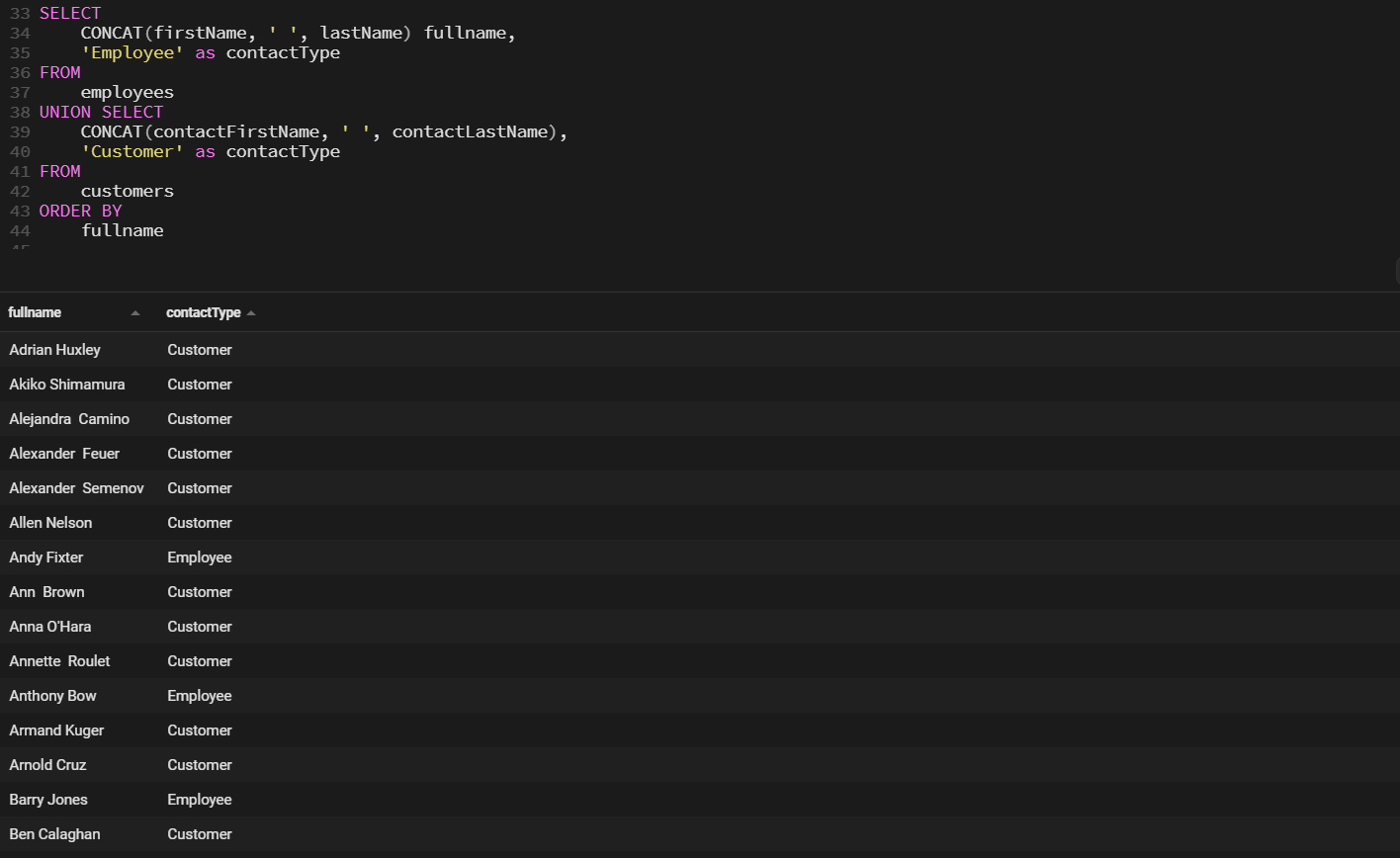
**Union**











**LAB – 6**

**Transaction (commit, Rollback)**

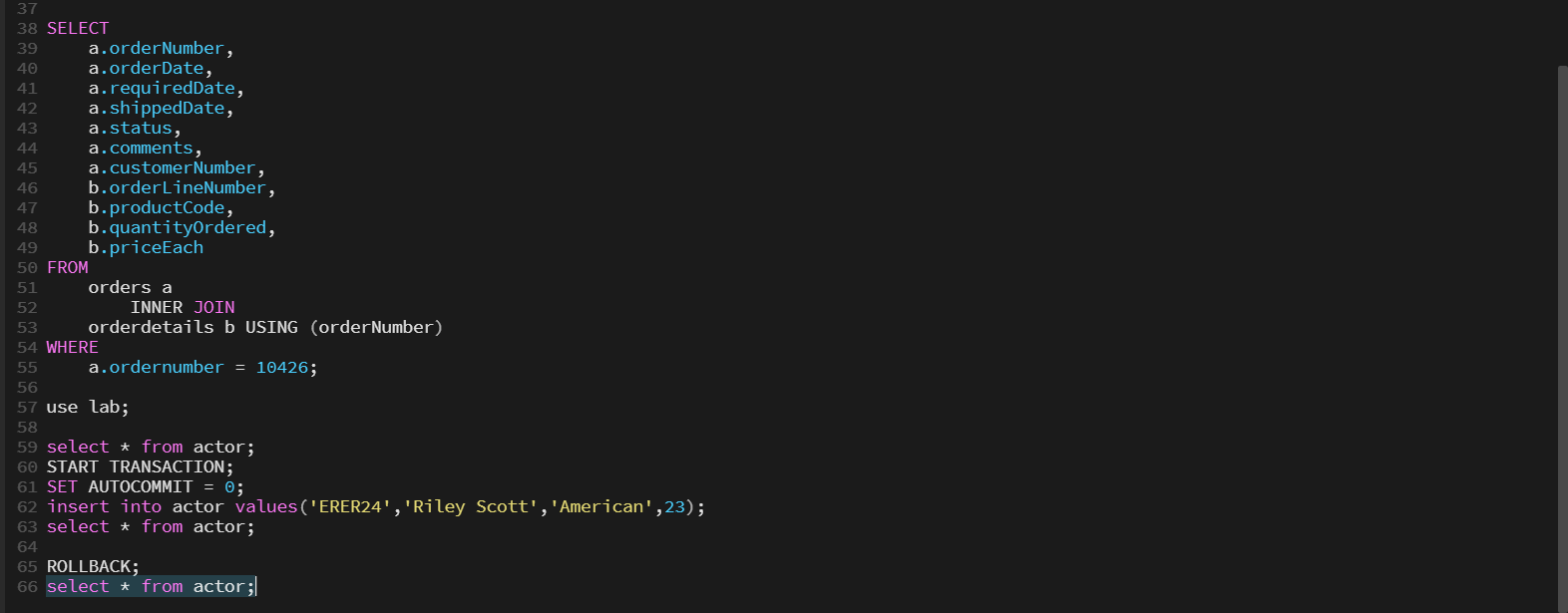
**What are Transactions?**

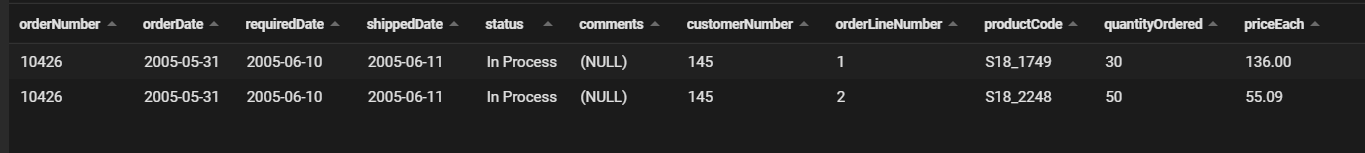
Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: **success** or **failure**.

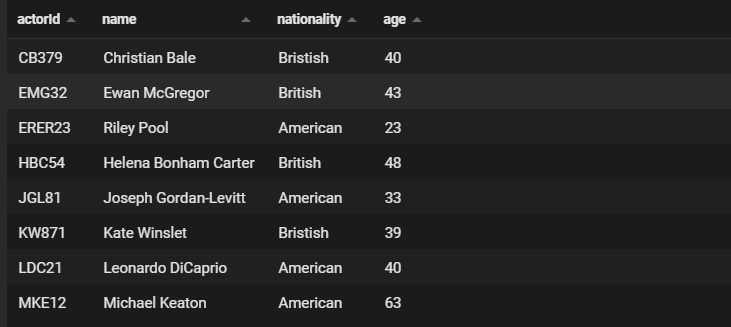
**COMMIT:** If everything is in order with all statements within a single transaction, all changes are recorded together in the database is called **committed**. The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.

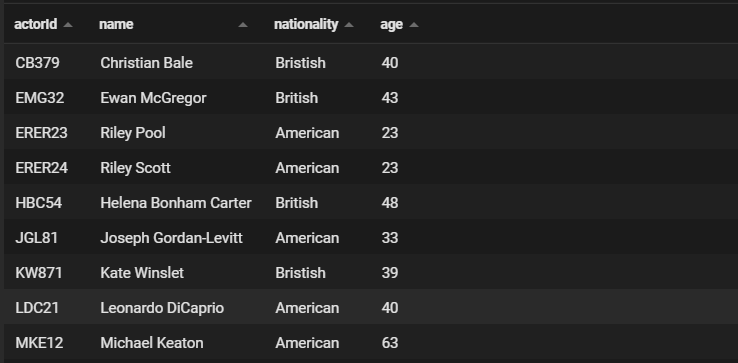
**ROLLBACK:** If any error occurs with any of the SQL grouped statements, all changes need to be aborted. The process of reversing changes is called **rollback**. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

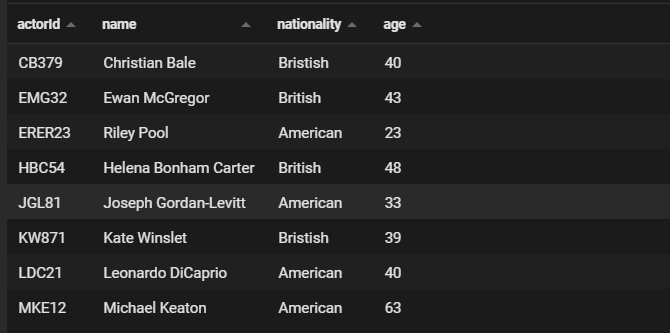












**LAB – 7**

**Trigger**

**Trigger:** A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

**Syntax:**

create trigger [trigger\_name] [before | after] {insert | update | delete} on [table\_name] [for each row] [trigger\_body]

