Department of Information Technology
Delhi Technological University

Database Management System (IT-202)

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

Submitted By:
Name: VARUN KUMAR
Roll No.:- 2K19/IT/140

Submitted To:-
Ms. Swati Sharda

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;

```
5 rows in set (0.00 sec)
```

mysql> update CollegeId

-> set Roll_No=92

-> where Roll_no=91;

```
5 rows in set (0.00 sec)
```

mysql> delete from CollegeID

-> where Roll_No=92;

```
4 rows in set (0.00 sec)
```

alter table CollegeId

-> drop column Branch;

```
4 rows in set (0.00 sec)
```

drop table CollegeID;

```
mysql>
```

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** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - 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

```
mysql> DESC CollegeId;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| firstName | varchar(30) | NO   |     | NULL    |       |
| lastName  | varchar(30) | NO   |     | NULL    |       |
| Roll_No   | int         | NO   | PRI | NULL    |       |
| Branch    | varchar(10) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (1.78 sec)
```

```
select * from CollegeID;
```

```
mysql> select * from CollegeID;
+-----------+-----------+---------+--------+
| firstName | lastName  | Roll_No | Branch |
+-----------+-----------+---------+--------+
| Nakul     | Gupta     |      84 | IT     |
| Naveen    | Kumar     |      87 | IT     |
| Naveen    | KumarShah |      88 | IT     |
| Naveen    | Yadav     |      89 | IT     |
| Nikhil    | Kumar     |      91 | IT     |
+-----------+-----------+---------+--------+
5 rows in set (0.43 sec)
```

insert into CollegeId(firstName,lastName,Roll_No,Branch)

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

```
mysql>  insert into CollegeId(firstName,lastName,Roll_No,Branch)
    -> values("Naveen","Yadav",89,"SE");
ERROR 3819 (HY000): Check constraint 'collegeid_chk_1' is violated.
```

insert into CollegeId(firstName,lastName,Roll_No,Branch)

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

```
mysql>  insert into CollegeId(firstName,lastName,Roll_No,Branch)
    ->  values("Paras","Jain",91,"IT");
ERROR 1062 (23000): Duplicate entry '91' for key 'collegeid.PRIMARY'
```

insert into CollegeId(firstName,lastName,Roll_No,Branch)

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

```
mysql>  insert into CollegeId(firstName,Roll_No,Branch)
    -> values("Paras",93,"IT");
ERROR 1364 (HY000): Field 'lastName' doesn't have a default value
```

```
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;
```

```
mysql> select * from Teacher;
+-----------+----------+---------+-----------+---------+
| firstName | lastName | Subject | TeacherID | Roll_No |
+-----------+----------+---------+-----------+---------+
| Jasraj    | Meena    | DS      |       123 |      91 |
| Ritu      | Aggarwal | ADA     |       234 |      89 |
+-----------+----------+---------+-----------+---------+
2 rows in set (0.04 sec)
```

Assignment - 1

Created client_master and product_master table:

```
mysql> desc client_master;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| client_no | varchar(6)   | YES  |     | NULL    |       |
| name      | varchar(20)  | YES  |     | NULL    |       |
| address1  | varchar(30)  | YES  |     | NULL    |       |
| address2  | varchar(30)  | YES  |     | NULL    |       |
| city      | varchar(15)  | YES  |     | NULL    |       |
| state     | varchar(15)  | YES  |     | NULL    |       |
| pincode   | int          | YES  |     | NULL    |       |
| bal_due   | decimal(10,2)| YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
8 rows in set (0.03 sec)

mysql> desc product_master;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Product_no    | varchar(6)  | YES  |     | NULL    |       |
| description   | varchar(30) | YES  |     | NULL    |       |
| profit_percent| decimal(5,2)| YES  |     | NULL    |       |
| unit_measure  | varchar(15) | YES  |     | NULL    |       |
| qty_on_hand   | int         | YES  |     | NULL    |       |
| reorder_lvl   | int         | YES  |     | NULL    |       |
| sell_price    | int         | YES  |     | NULL    |       |
| cost_price    | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Inserted data into Client_master and Product_master table:

```
mysql> select *from client_master;
+-----------+---------+----------+----------+---------+-------------+---------+----------+
| client_no | name    | address1 | address2 | city    | state       | pincode | bal_due  |
+-----------+---------+----------+----------+---------+-------------+---------+----------+
| 1         | Ivan    | NULL     | NULL     | Bombay  | Maharashtra |  400054 | 15000.00 |
| 2         | Vandana | NULL     | NULL     | Madras  | Tamilnadu   |  780001 |     0.00 |
| 3         | Pramada | NULL     | NULL     | Bombay  | Maharashtra |  400057 |  5000.00 |
| 4         | Basu    | NULL     | NULL     | Bombay  | Maharashtra |  400056 |     0.00 |
| 5         | Ravi    | NULL     | NULL     | Delhi   | NULL        |  100001 |  2000.00 |
| 6         | Rukmini | NULL     | NULL     | Bombay  | Maharashtra |  400050 |     0.00 |
+-----------+---------+----------+----------+---------+-------------+---------+----------+
6 rows in set (0.01 sec)

mysql> select *from product_master;
+------------+-------------+----------------+--------------+-------------+-------------+------------+------------+
| Product_no | description | profit_percent | unit_measure | qty_on_hand | reorder_lvl | sell_price | cost_price |
+------------+-------------+----------------+--------------+-------------+-------------+------------+------------+
| P00001     | 1.44floppies |           5.00 | piece        |         100 |          20 |        525 |        500 |
| P03453     | Monitors    |           6.00 | piece        |          10 |           3 |      12000 |      11200 |
| P06734     | Mouse       |           5.00 | piece        |          20 |           5 |       1050 |        500 |
| P07865     | 1.22floppies |           5.00 | piece        |         100 |          20 |        525 |        500 |
| P07868     | Keyboards   |           2.00 | piece        |          10 |           3 |       3150 |       3050 |
| P07885     | CD Drive    |           2.50 | piece        |          10 |           3 |       5250 |       5100 |
| P07965     | 540 HDD     |           4.00 | piece        |          10 |           3 |       8400 |       8000 |
| P07975     | 1.44 Drive  |           5.00 | piece        |          10 |           3 |       1050 |       1000 |
| P08865     | 1.22 Drive  |           5.00 | piece        |           2 |           3 |       1050 |       1000 |
+------------+-------------+----------------+--------------+-------------+-------------+------------+------------+
9 rows in set (0.00 sec)
```

Solutions of the questions =>

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

```
mysql> select name from client_master;
+---------+
| name    |
+---------+
| Ivan    |
| Vandana |
| Pramada |
| Basu    |
| Ravi    |
| Rukmini |
+---------+
6 rows in set (0.00 sec)
```

2) **Retrieve the list of names and cities of all the clients.**

```
mysql> select name,city from client_master;
+---------+--------+
| name    | city   |
+---------+--------+
| Ivan    | Bombay |
| Vandana | Madras |
| Pramada | Bombay |
| Basu    | Bombay |
| Ravi    | Delhi  |
| Rukmini | Bombay |
+---------+--------+
6 rows in set (0.00 sec)
```

3) List the various products available from the product_master table.

```
mysql> select description as Products_available from product_master;
+--------------------+
| Products_available |
+--------------------+
| 1.44floppies       |
| Monitors           |
| Mouse              |
| 1.22floppies       |
| Keyboards          |
| CD Drive           |
| 540 HDD            |
| 1.44 Drive         |
| 1.22 Drive         |
+--------------------+
9 rows in set (0.00 sec)
```

4) List all the clients who are located in Bombay.

```
mysql> select*from client_master where city="Bombay";
+-----------+---------+----------+----------+--------+-------------+---------+----------+
| client_no | name    | address1 | address2 | city   | state       | pincode | bal_due  |
+-----------+---------+----------+----------+--------+-------------+---------+----------+
| 1         | Ivan    | NULL     | NULL     | Bombay | Maharashtra |  400054 | 15000.00 |
| 3         | Pramada | NULL     | NULL     | Bombay | Maharashtra |  400057 |  5000.00 |
| 4         | Basu    | NULL     | NULL     | Bombay | Maharashtra |  400056 |     0.00 |
| 6         | Rukmini | NULL     | NULL     | Bombay | Maharashtra |  400050 |     0.00 |
+-----------+---------+----------+----------+--------+-------------+---------+----------+
4 rows in set (0.01 sec)
```

5) Display the information for client no 0001 and 0002.

```
mysql> select*from client_master where client_no=0001 or client_no=0002;
+-----------+---------+----------+----------+--------+-------------+---------+----------+
| client_no | name    | address1 | address2 | city   | state       | pincode | bal_due  |
+-----------+---------+----------+----------+--------+-------------+---------+----------+
| 1         | Ivan    | NULL     | NULL     | Bombay | Maharashtra |  400054 | 15000.00 |
| 2         | Vandana | NULL     | NULL     | Madras | Tamilnadu   |  780001 |     0.00 |
+-----------+---------+----------+----------+--------+-------------+---------+----------+
2 rows in set (0.00 sec)
```

6) Find the products with description as ‘1.44 Drive’ and ‘1.22 Drive’.

```
mysql> select*from product_master where description="1.44 Drive" or description="1.22 Drive";
+-----------+-------------+----------------+--------------+-------------+------------+------------+------------+
| Product_no | description | profit_percent | unit_measure | qty_on_hand | reorder_lvl | sell_price | cost_price |
+-----------+-------------+----------------+--------------+-------------+------------+------------+------------+
| P07975     | 1.44 Drive  |           5.00 | piece        |          10 |          3 |       1050 |       1000 |
| P08865     | 1.22 Drive  |           5.00 | piece        |           2 |          3 |       1050 |       1000 |
+-----------+-------------+----------------+--------------+-------------+------------+------------+------------+
2 rows in set (0.00 sec)
```

7) Find all the products whose sell price is greater than 5000.

```
mysql> select*from product_master where sell_price>5000;
+-----------+-------------+----------------+--------------+-------------+------------+------------+------------+
| Product_no | description | profit_percent | unit_measure | qty_on_hand | reorder_lvl | sell_price | cost_price |
+-----------+-------------+----------------+--------------+-------------+------------+------------+------------+
| P03453     | Monitors    |           6.00 | piece        |          10 |          3 |      12000 |      11200 |
| P07885     | CD Drive    |           2.50 | piece        |          10 |          3 |       5250 |       5100 |
| P07965     | 540 HDD     |           4.00 | piece        |          10 |          3 |       8400 |       8000 |
+-----------+-------------+----------------+--------------+-------------+------------+------------+------------+
3 rows in set (0.00 sec)
```

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

```
mysql> select*from client_master where city="Bombay" or city="Delhi" or city="Madras";
+-----------+---------+----------+----------+--------+-------------+---------+----------+
| client_no | name    | address1 | address2 | city   | state       | pincode | bal_due  |
+-----------+---------+----------+----------+--------+-------------+---------+----------+
| 1         | Ivan    | NULL     | NULL     | Bombay | Maharashtra |  400054 | 15000.00 |
| 2         | Vandana | NULL     | NULL     | Madras | Tamilnadu   |  780001 |     0.00 |
| 3         | Pramada | NULL     | NULL     | Bombay | Maharashtra |  400057 |  5000.00 |
| 4         | Basu    | NULL     | NULL     | Bombay | Maharashtra |  400056 |     0.00 |
| 5         | Ravi    | NULL     | NULL     | Delhi  | NULL        |  100001 |  2000.00 |
| 6         | Rukmini | NULL     | NULL     | Bombay | Maharashtra |  400050 |     0.00 |
+-----------+---------+----------+----------+--------+-------------+---------+----------+
6 rows in set (0.00 sec)
```

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

```
mysql> select*from product_master where sell_price>2000 and sell_price<=5000;
+------------+-------------+---------------+--------------+-------------+-------------+------------+------------+
| Product_no | description | profit_percent | unit_measure | qty_on_hand | reorder_lvl | sell_price | cost_price |
+------------+-------------+---------------+--------------+-------------+-------------+------------+------------+
| P07868     | Keyboards   |          2.00 | piece        |          10 |           3 |       3150 |       3050 |
+------------+-------------+---------------+--------------+-------------+-------------+------------+------------+
1 row in set (0.00 sec)
```

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

```
mysql> select name,city,state from client_master where not state="Maharashtra";
+---------+--------+-----------+
| name    | city   | state     |
+---------+--------+-----------+
| Vandana | Madras | Tamilnadu |
+---------+--------+-----------+
1 row in set (0.00 sec)
```

Assignment -2

<u>Objective</u> – Answer the following Questions

Q1. Make the primary key to client_no in client_master.

```
mysql> alter table client_master
    -> add constraint primary key(client_no);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc client_master;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| client_no | varchar(6)   | NO   | PRI | NULL    |       |
| name      | varchar(20)  | YES  |     | NULL    |       |
| address1  | varchar(30)  | YES  |     | NULL    |       |
| address2  | varchar(30)  | YES  |     | NULL    |       |
| city      | varchar(15)  | YES  |     | NULL    |       |
| state     | varchar(15)  | YES  |     | NULL    |       |
| pincode   | int          | YES  |     | NULL    |       |
| bal_due   | decimal(10,2)| YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
8 rows in set (0.01 sec)
```

Q2. Add a new column phone_no in the client_master table.

```
mysql> alter table client_master
    -> add column phone_no bigint;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc client_master;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| client_no | varchar(6)   | NO   | PRI | NULL    |       |
| name      | varchar(20)  | YES  |     | NULL    |       |
| address1  | varchar(30)  | YES  |     | NULL    |       |
| address2  | varchar(30)  | YES  |     | NULL    |       |
| city      | varchar(15)  | YES  |     | NULL    |       |
| state     | varchar(15)  | YES  |     | NULL    |       |
| pincode   | int          | YES  |     | NULL    |       |
| bal_due   | decimal(10,2)| YES  |     | NULL    |       |
| phone_no  | bigint       | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

Q3. Add the not null constraint in the product_master table with the columns description, profit percent , sell price and cost price.

```
mysql> alter table product_master
    -> modify column description varchar(50) NOT NULL,
    -> modify column profit_percent int NOT NULL,
    -> modify column sell_price int NOT NULL,
    -> modify column cost_price int NOT NULL;
Query OK, 9 rows affected (0.05 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> desc product_master;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| Product_no     | varchar(6)  | YES  |     | NULL    |       |
| description    | varchar(50) | NO   |     | NULL    |       |
| profit_percent | int         | NO   |     | NULL    |       |
| unit_measure   | varchar(15) | YES  |     | NULL    |       |
| qty_on_hand    | int         | YES  |     | NULL    |       |
| reorder_lvl    | int         | YES  |     | NULL    |       |
| sell_price     | int         | NO   |     | NULL    |       |
| cost_price     | int         | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
8 rows in set (0.01 sec)
```

Q4. Change the size of client_no field in the client_master table.

```
mysql> alter table client_master
    -> modify column client_no varchar(30);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc client_master;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| client_no | varchar(30)   | NO   | PRI | NULL    |       |
| name      | varchar(20)   | YES  |     | NULL    |       |
| address1  | varchar(30)   | YES  |     | NULL    |       |
| address2  | varchar(30)   | YES  |     | NULL    |       |
| city      | varchar(15)   | YES  |     | NULL    |       |
| state     | varchar(15)   | YES  |     | NULL    |       |
| pincode   | int           | YES  |     | NULL    |       |
| bal_due   | decimal(10,2) | YES  |     | NULL    |       |
| phone_no  | bigint        | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

Q5. Select product_no, description where profit percent is between 20 and 30 both inclusive.

```
mysql> select Product_no,description from product_master where profit_percent between 20 and 30;
Empty set (0.00 sec)

mysql> select*from product_master;
+------------+-------------+---------------+-------------+------------+------------+------------+------------+
| Product_no | description | profit_percent | unit_measure | qty_on_hand | reorder_lvl | sell_price | cost_price |
+------------+-------------+---------------+-------------+------------+------------+------------+------------+
| P00001     | 1.44floppies |             5 | piece       |        100 |          20 |        525 |        500 |
| P03453     | Monitors    |             6 | piece       |         10 |           3 |      12000 |      11200 |
| P06734     | Mouse       |             5 | piece       |         20 |           5 |       1050 |        500 |
| P07865     | 1.22floppies |             5 | piece       |        100 |          20 |        525 |        500 |
| P07868     | Keyboards   |             2 | piece       |         10 |           3 |       3150 |       3050 |
| P07885     | CD Drive    |             3 | piece       |         10 |           3 |       5250 |       5100 |
| P07965     | 540 HDD     |             4 | piece       |         10 |           3 |       8400 |       8000 |
| P07975     | 1.44 Drive  |             5 | piece       |         10 |           3 |       1050 |       1000 |
| P08865     | 1.22 Drive  |             5 | piece       |          2 |           3 |       1050 |       1000 |
+------------+-------------+---------------+-------------+------------+------------+------------+------------+
9 rows in set (0.00 sec)
```

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 AVG(amount) from payments;
+--------------+
| AVG(amount)  |
+--------------+
| 32431.645531 |
+--------------+
1 row in set (0.41 sec)
```

```
mysql> select ROUND(AVG(amount),2)  from payments;
+----------------------+
| ROUND(AVG(amount),2) |
+----------------------+
|             32431.65 |
+----------------------+
1 row in set (0.06 sec)
```

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, ROUND(AVG(amount),3) as avg_amount from payments group by customerNumber having avg_amount>43000;
+---------------+------------+
| customerNumber | avg_amount |
+---------------+------------+
|           114 |  45146.268 |
|           124 |  64909.804 |
|           141 |  55056.845 |
|           146 |  43435.117 |
|           151 |  44478.488 |
|           157 |  49254.625 |
|           167 |  48781.235 |
|           187 |  49470.030 |
|           211 |  45480.790 |
|           227 |  44954.900 |
|           239 |  80375.240 |
|           259 |  44611.570 |
|           286 |  45272.685 |
|           298 |  54388.960 |
|           321 |  66170.390 |
|           386 |  45071.655 |
|           450 |  59551.380 |
+---------------+------------+
17 rows in set (0.05 sec)
```

```
mysql> select COUNT(*) from payments;
+----------+
| COUNT(*) |
+----------+
|      273 |
+----------+
1 row in set (0.00 sec)

mysql> select COUNT(*) from payments where customerNumber = 121;
+----------+
| COUNT(*) |
+----------+
|        4 |
+----------+
1 row in set (0.01 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 |

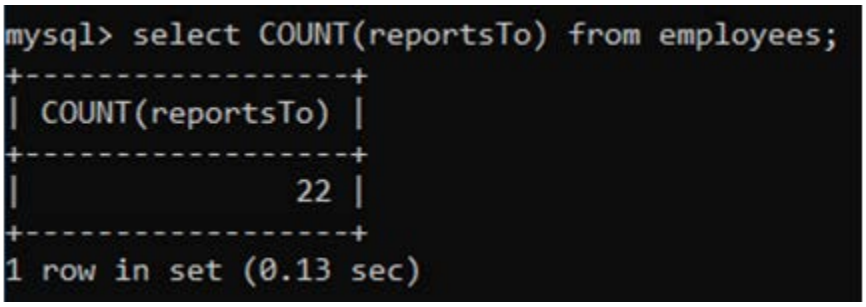
| 489 | 2 |

| 495 | 2 |

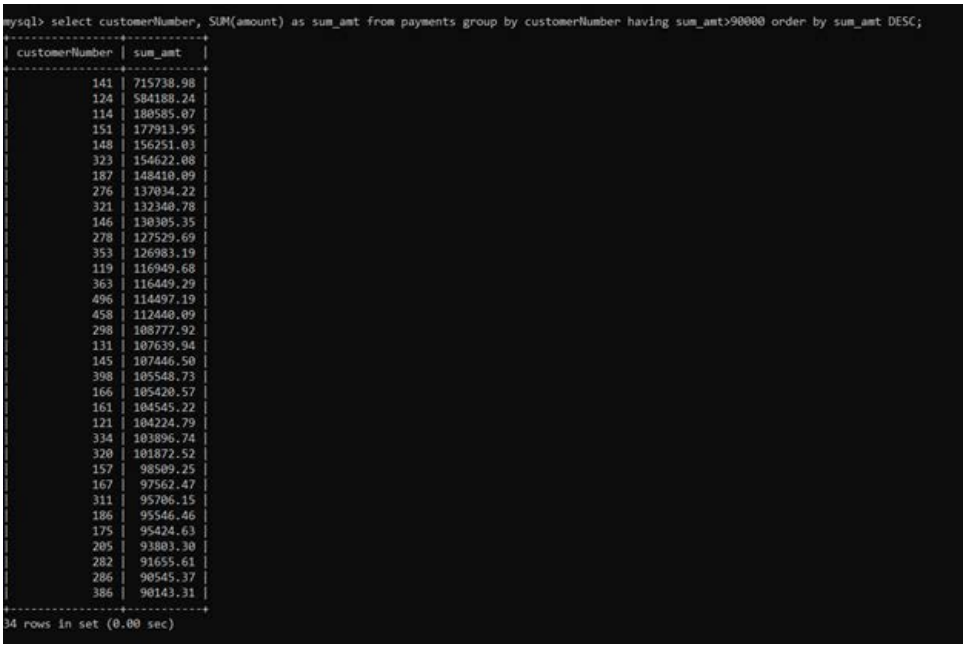
| 496 | 3 |

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

98 rows in set (0.00 sec)



```
mysql> select COUNT(DISTINCT reportsTo) from employees;
+----------------------------+
| COUNT(DISTINCT reportsTo) |
+----------------------------+
|                          6 |
+----------------------------+
1 row in set (0.03 sec)
```



```
mysql> select customerNumber, SUM(amount) as sum_amt from payments group by customerNumber having sum_amt>90000 order by sum_amt DESC;
+----------------+----------+
| customerNumber | sum_amt  |
+----------------+----------+
|            141 | 715738.98 |
|            124 | 584188.24 |
|            114 | 180585.07 |
|            151 | 177913.95 |
|            148 | 156251.03 |
|            323 | 154622.08 |
|            187 | 148410.09 |
|            276 | 137034.22 |
|            321 | 132340.78 |
|            146 | 130305.35 |
|            278 | 127529.69 |
|            353 | 126983.19 |
|            119 | 116949.68 |
|            363 | 116449.29 |
|            496 | 114497.19 |
|            458 | 112440.09 |
|            298 | 108777.92 |
|            131 | 107639.94 |
|            145 | 107446.50 |
|            398 | 105548.73 |
|            166 | 105420.57 |
|            161 | 104545.22 |
|            121 | 104224.79 |
|            334 | 103896.74 |
|            320 | 101872.52 |
|            157 |  98509.25 |
|            167 |  97562.47 |
|            311 |  95706.15 |
|            186 |  95546.46 |
|            175 |  95424.63 |
|            205 |  93803.30 |
|            282 |  91655.61 |
|            286 |  90545.37 |
|            386 |  90143.31 |
+----------------+----------+
34 rows in set (0.00 sec)
```

```
mysql> select MAX(amount),MIN(amount) from payments;
+--------------+--------------+
| MAX(amount) | MIN(amount) |
+--------------+--------------+
|   120166.58 |      615.45 |
+--------------+--------------+
1 row 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 |

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

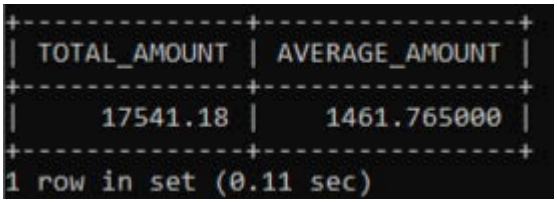
98 rows in set (0.00 sec)

```
mysql> select MIN(amount) as max_amt from payments;
+---------+
| max_amt |
+---------+
|  615.45 |
+---------+
1 row in set (0.00 sec)
```

Assignment -3

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

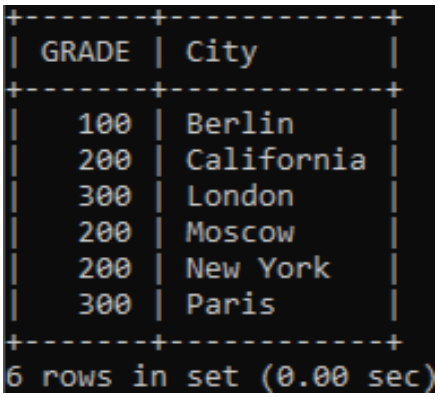
ANS 1 - SELECT SUM (purch_amt), AVG(purch_amt)
FROM orders;



```
+--------------+----------------+
| TOTAL_AMOUNT | AVERAGE_AMOUNT |
+--------------+----------------+
|     17541.18 |    1461.765000 |
+--------------+----------------+
1 row in set (0.11 sec)
```

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

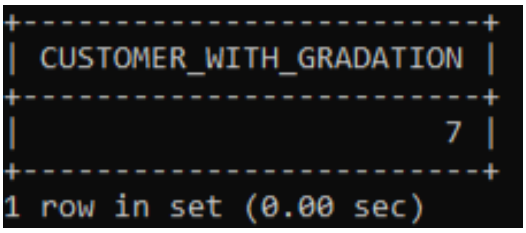
ANS 2 - SELECT city,MAX(grade)
FROM customer
GROUP BY city;



```
+-------+------------+
| GRADE | City       |
+-------+------------+
|   100 | Berlin     |
|   200 | California |
|   300 | London     |
|   200 | Moscow     |
|   200 | New York   |
|   300 | Paris      |
+-------+------------+
6 rows in set (0.00 sec)
```

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;

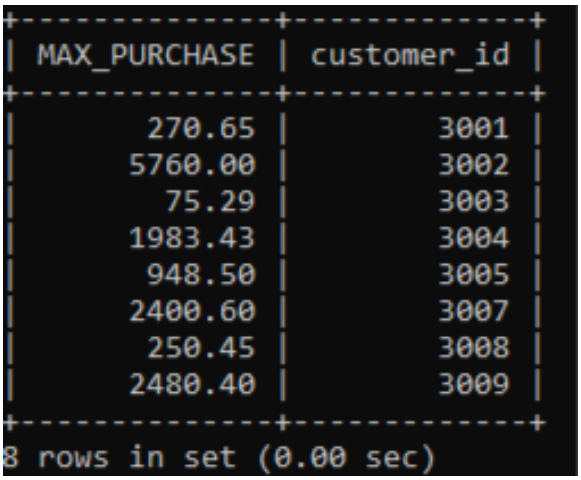


```
+-------------------------+
| CUSTOMER_WITH_GRADATION |
+-------------------------+
|                       7 |
+-------------------------+
1 row in set (0.00 sec)
```

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;

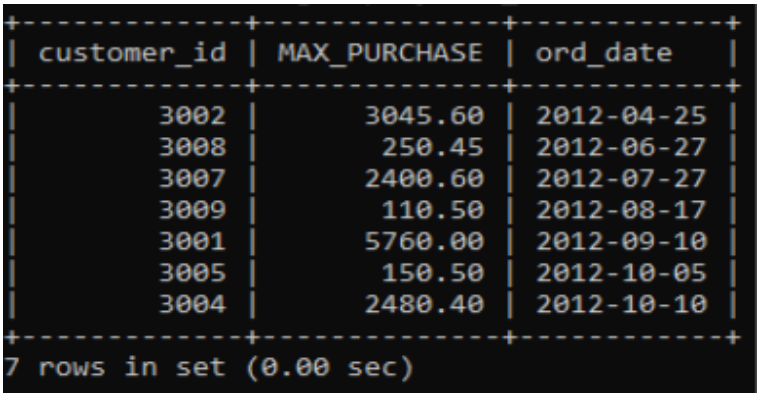


MAX_PURCHASE	customer_id
270.65	3001
5760.00	3002
75.29	3003
1983.43	3004
948.50	3005
2400.60	3007
250.45	3008
2480.40	3009

8 rows in set (0.00 sec)

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;



customer_id	MAX_PURCHASE	ord_date
3002	3045.60	2012-04-25
3008	250.45	2012-06-27
3007	2400.60	2012-07-27
3009	110.50	2012-08-17
3001	5760.00	2012-09-10
3005	150.50	2012-10-05
3004	2480.40	2012-10-10

7 rows in set (0.00 sec)

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;

customer_id	MAX_PURCHASE	ord_date
3002	3045.60	2012-04-25
3007	2400.60	2012-07-27
3001	5760.00	2012-09-10
3004	2480.40	2012-10-10

4 rows in set (0.00 sec)

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;

customer_id	MAX_PURCHASE
3002	5760.00
3004	1983.43
3007	2400.60

3 rows in set (0.00 sec)

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;

salesman_id	MAX_PURCHASE
5003	110.50
5007	75.29

2 rows in set (0.00 sec)

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;

count(emp_idno)	emp_dept
2	27
3	47
5	57
3	63

4 rows in set (0.00 sec)

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;
```

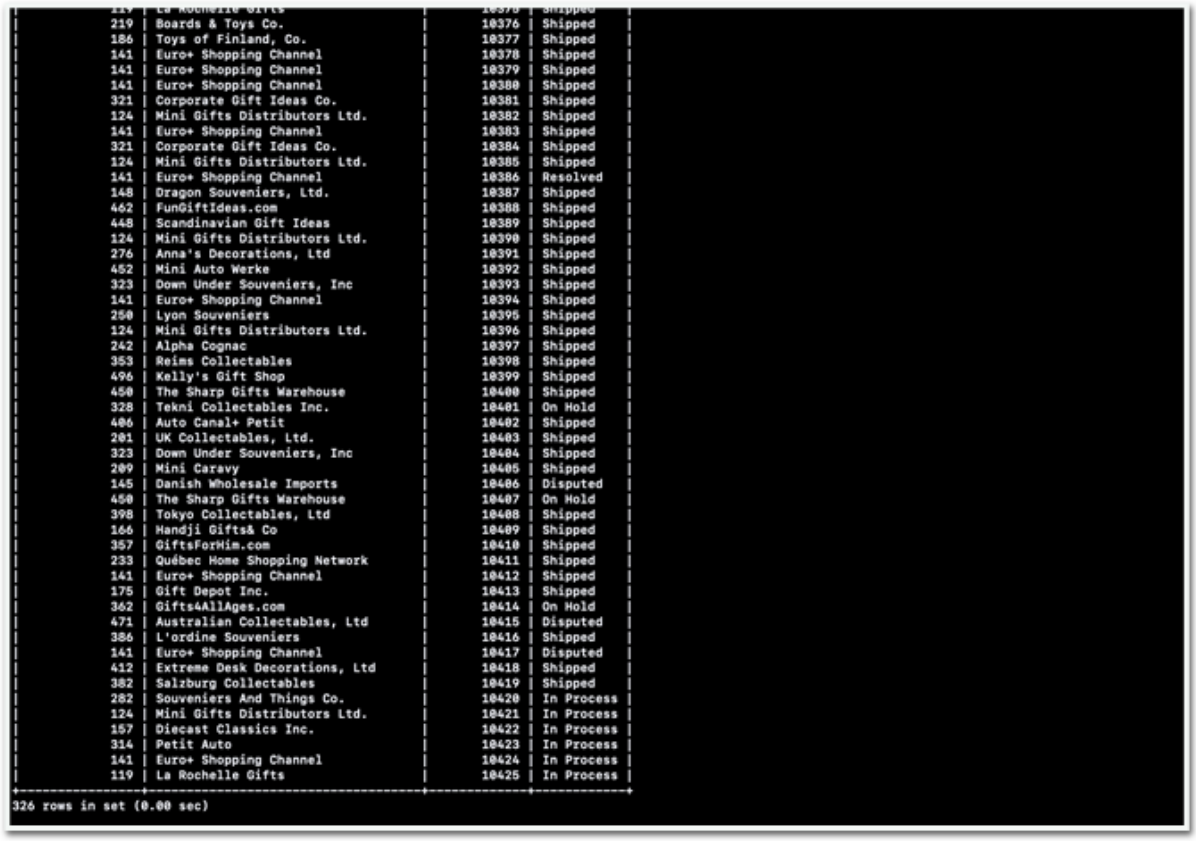
```
mysql> select productCode, productName,t1.productLine from products t1 INNER JOIN productLines t2 USING (productLine);
+-------------+----------------------------------------+-----------------+
| productCode | productName                            | productLine     |
+-------------+----------------------------------------+-----------------+
| S10_1949    | 1952 Alpine Renault 1300               | Classic Cars    |
| S10_4757    | 1972 Alfa Romeo GTA                    | Classic Cars    |
| S10_4962    | 1962 LanciaA Delta 16V                 | Classic Cars    |
| S12_1099    | 1968 Ford Mustang                      | Classic Cars    |
| S12_1108    | 2001 Ferrari Enzo                      | Classic Cars    |
| S12_3148    | 1969 Corvair Monza                     | Classic Cars    |
| S12_3380    | 1968 Dodge Charger                     | Classic Cars    |
| S12_3891    | 1969 Ford Falcon                       | Classic Cars    |
| S12_3990    | 1970 Plymouth Hemi Cuda                | Classic Cars    |
| S12_4675    | 1969 Dodge Charger                     | Classic Cars    |
| S18_1129    | 1993 Mazda RX-7                        | Classic Cars    |
| S18_1589    | 1965 Aston Martin DB5                  | Classic Cars    |
| S18_1889    | 1948 Porsche 356-A Roadster            | Classic Cars    |
| S18_1984    | 1995 Honda Civic                       | Classic Cars    |
| S18_2238    | 1998 Chrysler Plymouth Prowler         | Classic Cars    |
| S18_2870    | 1999 Indy 500 Monte Carlo SS           | Classic Cars    |
| S18_3232    | 1992 Ferrari 360 Spider red            | Classic Cars    |
| S18_3233    | 1985 Toyota Supra                      | Classic Cars    |
| S18_3278    | 1969 Dodge Super Bee                   | Classic Cars    |
| S18_3482    | 1976 Ford Gran Torino                  | Classic Cars    |
| S18_3685    | 1948 Porsche Type 356 Roadster         | Classic Cars    |
| S18_4027    | 1970 Triumph Spitfire                  | Classic Cars    |
| S18_4721    | 1957 Corvette Convertible              | Classic Cars    |
| S18_4933    | 1957 Ford Thunderbird                  | Classic Cars    |
| S24_1046    | 1970 Chevy Chevelle SS 454             | Classic Cars    |
| S24_1444    | 1970 Dodge Coronet                     | Classic Cars    |
| S24_1628    | 1966 Shelby Cobra 427 S/C              | Classic Cars    |
| S24_2766    | 1949 Jaguar XK 120                     | Classic Cars    |
| S24_2840    | 1958 Chevy Corvette Limited Edition    | Classic Cars    |
| S24_2887    | 1952 Citroen-15CV                      | Classic Cars    |
| S24_2972    | 1982 Lamborghini Diablo                | Classic Cars    |
| S24_3191    | 1969 Chevrolet Camaro Z28              | Classic Cars    |
| S24_3371    | 1971 Alpine Renault 1600s              | Classic Cars    |
| S24_3432    | 2002 Chevy Corvette                    | Classic Cars    |
| S24_3856    | 1956 Porsche 356A Coupe                | Classic Cars    |
| S24_4048    | 1992 Porsche Cayenne Turbo Silver      | Classic Cars    |
| S24_4620    | 1961 Chevrolet Impala                  | Classic Cars    |
| S700_2824   | 1982 Camaro Z28                        | Classic Cars    |
| S10_1678    | 1969 Harley Davidson Ultimate Chopper  | Motorcycles     |
| S10_2016    | 1996 Moto Guzzi 1100i                  | Motorcycles     |
| S10_4698    | 2003 Harley-Davidson Eagle Drag Bike   | Motorcycles     |
| S12_2823    | 2002 Suzuki XREO                       | Motorcycles     |
| S18_2625    | 1936 Harley Davidson El Knucklehead    | Motorcycles     |
| S18_3782    | 1957 Vespa GS150                       | Motorcycles     |
| S24_1578    | 1997 BMW R 1100 S                      | Motorcycles     |
| S24_2000    | 1960 BSA Gold Star DBD34               | Motorcycles     |
| S24_2360    | 1982 Ducati 900 Monster                | Motorcycles     |
| S32_1374    | 1997 BMW F650 ST                       | Motorcycles     |
| S32_2206    | 1982 Ducati 996 R                      | Motorcycles     |
| S32_4485    | 1974 Ducati 350 Mk3 Desmo              | Motorcycles     |
| S50_4713    | 2002 Yamaha YZR M1                     | Motorcycles     |
| S18_1662    | 1980s Black Hawk Helicopter            | Planes          |
| S18_2581    | P-51-D Mustang                         | Planes          |
| S24_1785    | 1928 British Royal Navy Airplane       | Planes          |
| S24_2841    | 1900s Vintage Bi-Plane                 | Planes          |
| S24_3949    | Corsair F4U ( Bird Cage)               | Planes          |
| S24_4278    | 1900s Vintage Tri-Plane                | Planes          |
| S18_2949    | 1913 Ford Model T Speedster            | Vintage Cars    |
| S18_2957    | 1934 Ford V8 Coupe                     | Vintage Cars    |
| S18_3136    | 18th Century Vintage Horse Carriage    | Vintage Cars    |
| S18_3140    | 1903 Ford Model A                      | Vintage Cars    |
| S18_3320    | 1917 Maxwell Touring Car               | Vintage Cars    |
| S18_3856    | 1941 Chevrolet Special Deluxe Cabriolet | Vintage Cars   |
| S18_4409    | 1932 Alfa Romeo 8C2300 Spider Sport    | Vintage Cars    |
| S18_4522    | 1904 Buick Runabout                    | Vintage Cars    |
| S18_4668    | 1939 Cadillac Limousine                | Vintage Cars    |
| S24_1937    | 1939 Chevrolet Deluxe Coupe            | Vintage Cars    |
| S24_2022    | 1938 Cadillac V-16 Presidential Limousine | Vintage Cars |
| S24_3151    | 1912 Ford Model T Delivery Wagon       | Vintage Cars    |
| S24_3420    | 1937 Horch 930V Limousine              | Vintage Cars    |
| S24_3816    | 1940 Ford Delivery Sedan               | Vintage Cars    |
| S24_3969    | 1936 Mercedes Benz 500k Roadster       | Vintage Cars    |
| S24_4258    | 1936 Chrysler Airflow                  | Vintage Cars    |
| S32_4289    | 1928 Ford Phaeton Deluxe               | Vintage Cars    |
| S50_1341    | 1930 Buick Marquette Phaeton           | Vintage Cars    |
+-------------+----------------------------------------+-----------------+
110 rows in set (0.00 sec)
```

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;
```

```
mysql> select t1.customerNumber , customerName , orderNumber, status from customers t1 INNER JOIN orders t2 USING(customerNumber);
+----------------+----------------------------------+-------------+------------+
| customerNumber | customerName                     | orderNumber | status     |
+----------------+----------------------------------+-------------+------------+
|            363 | Online Diecast Creations Co.     |       10100 | Shipped    |
|            128 | Blauer See Auto, Co.             |       10101 | Shipped    |
|            181 | Vitachrome Inc.                  |       10102 | Shipped    |
|            121 | Baane Mini Imports               |       10103 | Shipped    |
|            141 | Euro+ Shopping Channel           |       10104 | Shipped    |
|            145 | Danish Wholesale Imports         |       10105 | Shipped    |
|            278 | Rovelli Gifts                    |       10106 | Shipped    |
|            131 | Land of Toys Inc.                |       10107 | Shipped    |
|            385 | Cruz & Sons Co.                  |       10108 | Shipped    |
|            486 | Motor Mint Distributors Inc.     |       10109 | Shipped    |
|            187 | AV Stores, Co.                   |       10110 | Shipped    |
|            129 | Mini Wheels Co.                  |       10111 | Shipped    |
|            144 | Volvo Model Replicas, Co         |       10112 | Shipped    |
|            124 | Mini Gifts Distributors Ltd.     |       10113 | Shipped    |
|            172 | La Corne D'abondance, Co.        |       10114 | Shipped    |
|            424 | Classic Legends Inc.             |       10115 | Shipped    |
|            381 | Royale Belge                     |       10116 | Shipped    |
|            148 | Dragon Souveniers, Ltd.          |       10117 | Shipped    |
|            216 | Enaco Distributors               |       10118 | Shipped    |
|            382 | Salzburg Collectables            |       10119 | Shipped    |
|            114 | Australian Collectors, Co.       |       10120 | Shipped    |
|            353 | Reims Collectables               |       10121 | Shipped    |
|            350 | Marseille Mini Autos             |       10122 | Shipped    |
|            103 | Atelier graphique                |       10123 | Shipped    |
|            112 | Signal Gift Stores               |       10124 | Shipped    |
|            114 | Australian Collectors, Co.       |       10125 | Shipped    |
|            458 | Corrida Auto Replicas, Ltd       |       10126 | Shipped    |
|            151 | Muscle Machine Inc               |       10127 | Shipped    |
|            141 | Euro+ Shopping Channel           |       10128 | Shipped    |
|            324 | Stylish Desk Decors, Co.         |       10129 | Shipped    |
|            198 | Auto-Moto Classics Inc.          |       10130 | Shipped    |
|            447 | Gift Ideas Corp.                 |       10131 | Shipped    |
|            323 | Down Under Souveniers, Inc       |       10132 | Shipped    |
|            141 | Euro+ Shopping Channel           |       10133 | Shipped    |
|            250 | Lyon Souveniers                  |       10134 | Shipped    |
|            124 | Mini Gifts Distributors Ltd.     |       10135 | Shipped    |
|            242 | Alpha Cognac                     |       10136 | Shipped    |
|            353 | Reims Collectables               |       10137 | Shipped    |
|            496 | Kelly's Gift Shop                |       10138 | Shipped    |
|            282 | Souveniers And Things Co.        |       10139 | Shipped    |
|            161 | Technics Stores Inc.             |       10140 | Shipped    |
|            334 | Suominen Souveniers              |       10141 | Shipped    |
|            124 | Mini Gifts Distributors Ltd.     |       10142 | Shipped    |
|            320 | Mini Creations Ltd.              |       10143 | Shipped    |
|            381 | Royale Belge                     |       10144 | Shipped    |
|            205 | Toys4GrownUps.com                |       10145 | Shipped    |
|            447 | Gift Ideas Corp.                 |       10146 | Shipped    |
|            379 | Collectables For Less Inc.       |       10147 | Shipped    |
|            276 | Anna's Decorations, Ltd          |       10148 | Shipped    |
|            487 | Signal Collectibles Ltd.         |       10149 | Shipped    |
|            148 | Dragon Souveniers, Ltd.          |       10150 | Shipped    |
|            311 | Oulu Toy Supplies, Inc.          |       10151 | Shipped    |
|            333 | Australian Gift Network, Co      |       10152 | Shipped    |
|            141 | Euro+ Shopping Channel           |       10153 | Shipped    |
|            219 | Boards & Toys Co.                |       10154 | Shipped    |
|            186 | Toys of Finland, Co.             |       10155 | Shipped    |
|            141 | Euro+ Shopping Channel           |       10156 | Shipped    |
```



219	Boards & Toys Co.	10376	Shipped
186	Toys of Finland, Co.	10377	Shipped
141	Euro+ Shopping Channel	10378	Shipped
141	Euro+ Shopping Channel	10379	Shipped
141	Euro+ Shopping Channel	10380	Shipped
321	Corporate Gift Ideas Co.	10381	Shipped
124	Mini Gifts Distributors Ltd.	10382	Shipped
141	Euro+ Shopping Channel	10383	Shipped
321	Corporate Gift Ideas Co.	10384	Shipped
124	Mini Gifts Distributors Ltd.	10385	Shipped
141	Euro+ Shopping Channel	10386	Resolved
148	Dragon Souveniers, Ltd.	10387	Shipped
462	FunGiftIdeas.com	10388	Shipped
448	Scandinavian Gift Ideas	10389	Shipped
124	Mini Gifts Distributors Ltd.	10390	Shipped
276	Anna's Decorations, Ltd	10391	Shipped
452	Mini Auto Werke	10392	Shipped
323	Down Under Souveniers, Inc	10393	Shipped
141	Euro+ Shopping Channel	10394	Shipped
250	Lyon Souveniers	10395	Shipped
124	Mini Gifts Distributors Ltd.	10396	Shipped
242	Alpha Cognac	10397	Shipped
353	Reims Collectables	10398	Shipped
496	Kelly's Gift Shop	10399	Shipped
450	The Sharp Gifts Warehouse	10400	Shipped
328	Tekni Collectables Inc.	10401	On Hold
406	Auto Canal+ Petit	10402	Shipped
201	UK Collectables, Ltd.	10403	Shipped
323	Down Under Souveniers, Inc	10404	Shipped
209	Mini Caravy	10405	Shipped
145	Danish Wholesale Imports	10406	Disputed
450	The Sharp Gifts Warehouse	10407	On Hold
398	Tokyo Collectables, Ltd	10408	Shipped
166	Handji Gifts& Co	10409	Shipped
357	GiftsForHim.com	10410	Shipped
233	Québec Home Shopping Network	10411	Shipped
141	Euro+ Shopping Channel	10412	Shipped
175	Gift Depot Inc.	10413	Shipped
362	Gifts4AllAges.com	10414	On Hold
471	Australian Collectables, Ltd	10415	Disputed
386	L'ordine Souveniers	10416	Shipped
141	Euro+ Shopping Channel	10417	Disputed
412	Extreme Desk Decorations, Ltd	10418	Shipped
382	Salzburg Collectables	10419	Shipped
282	Souveniers And Things Co.	10420	In Process
124	Mini Gifts Distributors Ltd.	10421	In Process
157	Diecast Classics Inc.	10422	In Process
314	Petit Auto	10423	In Process
141	Euro+ Shopping Channel	10424	In Process
119	La Rochelle Gifts	10425	In Process

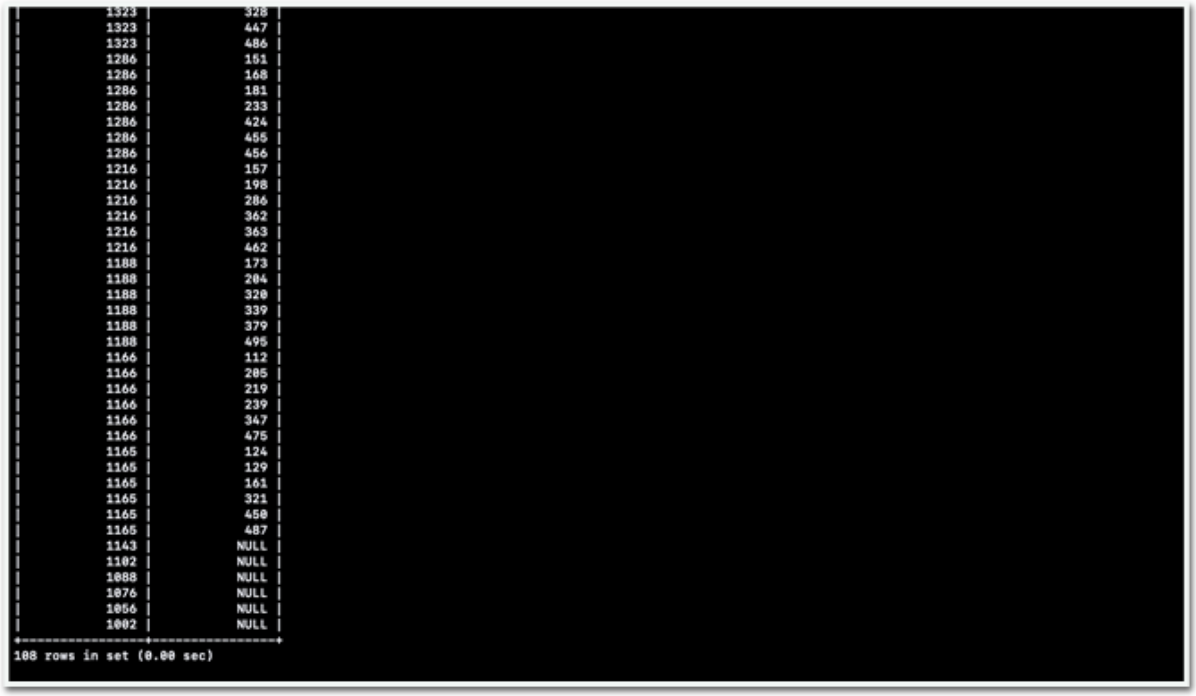
326 rows in set (0.00 sec)

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:

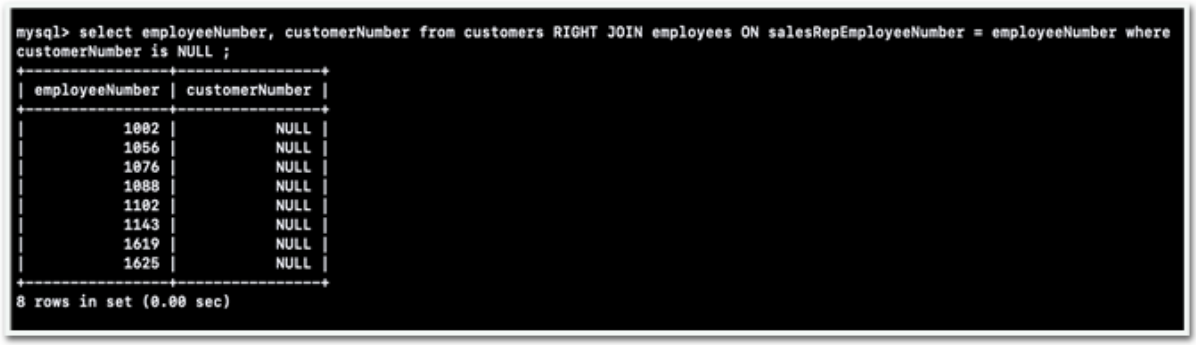
```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

```
mysql> select employeeNumber, customerNumber from customers RIGHT JOIN employees ON salesRepEmployeeNumber = employeeNumber order by employeeNumber DESC;
+----------------+----------------+
| employeeNumber | customerNumber |
+----------------+----------------+
|           1702 |            216 |
|           1702 |            298 |
|           1702 |            344 |
|           1702 |            376 |
|           1702 |            458 |
|           1702 |            484 |
|           1625 |           NULL |
|           1621 |            148 |
|           1621 |            177 |
|           1621 |            211 |
|           1621 |            385 |
|           1621 |            398 |
|           1619 |           NULL |
|           1612 |            166 |
|           1612 |            323 |
|           1612 |            357 |
|           1612 |            412 |
|           1612 |            496 |
|           1611 |            114 |
|           1611 |            276 |
|           1611 |            282 |
|           1611 |            333 |
|           1611 |            471 |
|           1504 |            121 |
|           1504 |            128 |
|           1504 |            144 |
|           1504 |            167 |
|           1504 |            189 |
|           1504 |            259 |
|           1504 |            299 |
|           1504 |            415 |
|           1504 |            448 |
|           1501 |            186 |
|           1501 |            187 |
|           1501 |            201 |
|           1501 |            240 |
|           1501 |            311 |
|           1501 |            324 |
|           1501 |            334 |
|           1501 |            489 |
|           1401 |            145 |
|           1401 |            227 |
|           1401 |            249 |
|           1401 |            278 |
|           1401 |            314 |
|           1401 |            381 |
|           1401 |            382 |
|           1401 |            386 |
|           1401 |            452 |
|           1401 |            473 |
|           1370 |            103 |
|           1370 |            119 |
|           1370 |            141 |
|           1370 |            171 |
|           1370 |            209 |
|           1370 |            242 |
```



```
|          1323 |           328 |
|          1323 |           447 |
|          1323 |           486 |
|          1286 |           151 |
|          1286 |           168 |
|          1286 |           181 |
|          1286 |           233 |
|          1286 |           424 |
|          1286 |           455 |
|          1286 |           456 |
|          1216 |           157 |
|          1216 |           198 |
|          1216 |           286 |
|          1216 |           362 |
|          1216 |           363 |
|          1216 |           462 |
|          1188 |           173 |
|          1188 |           204 |
|          1188 |           320 |
|          1188 |           339 |
|          1188 |           379 |
|          1188 |           495 |
|          1166 |           112 |
|          1166 |           205 |
|          1166 |           219 |
|          1166 |           239 |
|          1166 |           347 |
|          1166 |           475 |
|          1165 |           124 |
|          1165 |           129 |
|          1165 |           161 |
|          1165 |           321 |
|          1165 |           450 |
|          1165 |           487 |
|          1143 |          NULL |
|          1102 |          NULL |
|          1088 |          NULL |
|          1076 |          NULL |
|          1056 |          NULL |
|          1002 |          NULL |
+---------------+---------------+
108 rows in set (0.00 sec)
```

Some Other queries:



```
mysql> select employeeNumber, customerNumber from customers RIGHT JOIN employees ON salesRepEmployeeNumber = employeeNumber where
customerNumber is NULL ;
+----------------+----------------+
| employeeNumber | customerNumber |
+----------------+----------------+
|           1002 |           NULL |
|           1056 |           NULL |
|           1076 |           NULL |
|           1088 |           NULL |
|           1102 |           NULL |
|           1143 |           NULL |
|           1619 |           NULL |
|           1625 |           NULL |
+----------------+----------------+
8 rows in set (0.00 sec)
```

—> 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)

```
mysql> select t1.customerNumber, t1.customerName, MAX(amount) as payment from customers t1 INNER JOIN payments t2 using(customerNu
mber) group by customerNumber having payment>80000;
+----------------+----------------------------+-----------+
| customerNumber | customerName               | payment   |
+----------------+----------------------------+-----------+
|            114 | Australian Collectors, Co. |  82261.22 |
|            124 | Mini Gifts Distributors Ltd. | 111654.40 |
|            141 | Euro+ Shopping Channel     | 120166.58 |
|            148 | Dragon Souveniers, Ltd.    | 105743.00 |
|            167 | Herkku Gifts               |  85024.46 |
|            239 | Collectable Mini Designs Co. |  80375.24 |
|            321 | Corporate Gift Ideas Co.   |  85559.12 |
+----------------+----------------------------+-----------+
7 rows in set (0.00 sec)
```

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

```
mysql> select productLine, min(buyPrice) from products group by productLine;
+------------------+---------------+
| productLine      | min(buyPrice) |
+------------------+---------------+
| Classic Cars     |         15.91 |
| Motorcycles      |         24.14 |
| Planes           |         29.34 |
| Ships            |         33.30 |
| Trains           |         26.72 |
| Trucks and Buses |         24.92 |
| Vintage Cars     |         20.61 |
+------------------+---------------+
7 rows in set (0.00 sec)

mysql>
```

—> 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 operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents a range of characters	c[a-b]t finds cat and cbt

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.

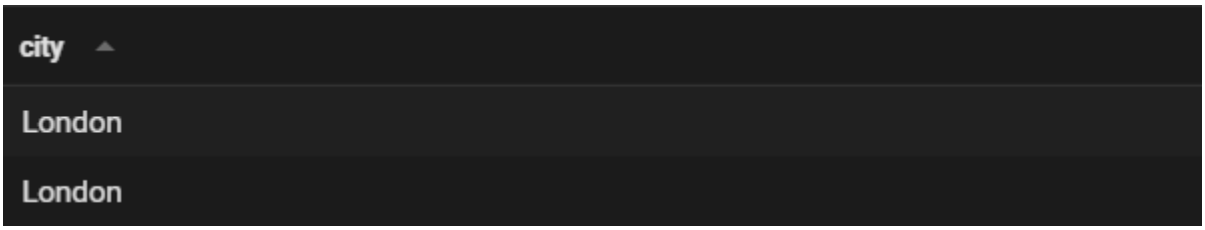
SELECT column1 [, column2]FROM table1 [, table2][WHERE condition]INTERSECTSELECT column1 [, column2]FROM table1 [, table2][WHERE condition]

Wildcards

```
use classicmodels;

select customerName from customers where customerName like 'A%';

select city from customers where city like '%ar%';

select city from customers where city like 'L_n_o_';

select country from customers where regexp_like(city,'[a-c]');

select country from customers where not regexp_like(city,'[a-c]');

select country from customers where regexp_like(city,'[ekv]');
```

customerName
Atelier graphique
Australian Collectors, Co.
American Souvenirs Inc
AV Stores, Co.
Auto-Moto Classics Inc.
Asian Shopping Network, Co
ANG Resellers
Alpha Cognac
Amica Models & Co.
Auto Associés & Cie.
Anna's Decorations, Ltd
Australian Gift Network, Co
Asian Treasures, Inc.
Auto Canal+ Petit
Anton Designs, Ltd.
Australian Collectables, Ltd

city
Warszawa
Paris
Barcelona
Paris
Newark
Marseille
Charleroi
Paris
Stuttgart



city
London
London

country
France
USA
Australia
France
Norway
USA
Poland
Germany
USA
USA
Spain
Denmark
Singapore
USA
USA
USA
Singapore
Norway
USA

country
Sweden
France
France
Finland
UK
Denmark
France
Italy
Germany
Australia
Switzerland
Norway
Finland
UK
Finland
France
Germany
New Zealand
Germany

country
France
USA
Australia
France
Norway
USA
Germany
Sweden
Denmark
Singapore
USA
USA
Singapore
Norway
USA
France
USA
USA

Union

```
use classicmodels;
SELECT
    firstName,
    lastName
FROM
    employees
UNION
SELECT
    contactFirstName,
    contactLastName
FROM
    customers;
```

firstName	lastName
Diane	Murphy
Mary	Phan
Jeff	Firrelli
William	Patterson
Gerard	Bondur
Anthony	Bow
Leslie	Jennings
Leslie	Thompson
Julie	Firrelli
Steve	Patterson
Foon Yue	Tseng
George	Vanauf
Loui	Bondur
Gerard	Hernandez
Pamela	Castillo
Larry	Bott

```sql
SELECT
    CONCAT(firstName,' ',lastName) fullname
FROM
    employees
UNION SELECT
    CONCAT(contactFirstName,' ',contactLastName)
FROM
    customers;
```

Save Run

fullname
Diane Murphy
Mary Phan
Jeff Firrelli
William Patterson
Gerard Bondur
Anthony Bow
Leslie Jennings
Leslie Thompson
Julie Firrelli
Steve Patterson
Foon Yue Tseng
George Vanauf
Loui Bondur
Gerard Hernandez
Pamela Castillo
Larry Bott
Barry Jones
Andy Fixter

```sql
SELECT
    concat(firstName,' ',lastName) fullname
FROM
    employees
UNION SELECT
    concat(contactFirstName,' ',contactLastName)
FROM
    customers
ORDER BY fullname;
```

fullname
Adrian Huxley
Akiko Shimamura
Alejandra Camino
Alexander Feuer
Alexander Semenov
Allen Nelson
Andy Fixter
Ann Brown
Anna O'Hara
Annette Roulet
Anthony Bow
Armand Kuger
Arnold Cruz
Barry Jones
Ben Calaghan
Bradley Schuyler
Braun Urs
Brian Chandler

```sql
SELECT
    CONCAT(firstName, ' ', lastName) fullname,
    'Employee' as contactType
FROM
    employees
UNION SELECT
    CONCAT(contactFirstName, ' ', contactLastName),
    'Customer' as contactType
FROM
    customers
ORDER BY
    fullname
```

fullname	contactType
Adrian Huxley	Customer
Akiko Shimamura	Customer
Alejandra Camino	Customer
Alexander Feuer	Customer
Alexander Semenov	Customer
Allen Nelson	Customer
Andy Fixter	Employee
Ann Brown	Customer
Anna O'Hara	Customer
Annette Roulet	Customer
Anthony Bow	Employee
Armand Kuger	Customer
Arnold Cruz	Customer
Barry Jones	Employee
Ben Calaghan	Customer

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.

```
use classicmodels;

START TRANSACTION;


SELECT
    @orderNumber:=MAX(orderNUmber)+1
FROM
    orders;


INSERT INTO orders(orderNumber,
                   orderDate,
                   requiredDate,
                   shippedDate,
                   status,
                   customerNumber)
VALUES(@orderNumber,
       '2005-05-31',
       '2005-06-10',
       '2005-06-11',
       'In Process',
        145);


INSERT INTO orderdetails(orderNumber,
                         productCode,
                         quantityOrdered,
                         priceEach,
                         orderLineNumber)
VALUES(@orderNumber,'S18_1749', 30, '136', 1),
      (@orderNumber,'S18_2248', 50, '55.09', 2);


```

```sql
SELECT
    a.orderNumber,
    a.orderDate,
    a.requiredDate,
    a.shippedDate,
    a.status,
    a.comments,
    a.customerNumber,
    b.orderLineNumber,
    b.productCode,
    b.quantityOrdered,
    b.priceEach
FROM
    orders a
        INNER JOIN
    orderdetails b USING (orderNumber)
WHERE
    a.ordernumber = 10426;

use lab;

select * from actor;
START TRANSACTION;
SET AUTOCOMMIT = 0;
insert into actor values('ERER24','Riley Scott','American',23);
select * from actor;

ROLLBACK;
select * from actor;
```

orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumber	orderLineNumber	productCode	quantityOrdered	priceEach
10426	2005-05-31	2005-06-10	2005-06-11	In Process	(NULL)	145	1	S18_1749	30	136.00
10426	2005-05-31	2005-06-10	2005-06-11	In Process	(NULL)	145	2	S18_2248	50	55.09

actorId	name	nationality	age
CB379	Christian Bale	Bristish	40
EMG32	Ewan McGregor	British	43
ERER23	Riley Pool	American	23
HBC54	Helena Bonham Carter	British	48
JGL81	Joseph Gordan-Levitt	American	33
KW871	Kate Winslet	Bristish	39
LDC21	Leonardo DiCaprio	American	40
MKE12	Michael Keaton	American	63

actorId	name	nationality	age
CB379	Christian Bale	Bristish	40
EMG32	Ewan McGregor	British	43
ERER23	Riley Pool	American	23
ERER24	Riley Scott	American	23
HBC54	Helena Bonham Carter	British	48
JGL81	Joseph Gordan-Levitt	American	33
KW871	Kate Winslet	Bristish	39
LDC21	Leonardo DiCaprio	American	40
MKE12	Michael Keaton	American	63

actorId	name	nationality	age
CB379	Christian Bale	Bristish	40
EMG32	Ewan McGregor	British	43
ERER23	Riley Pool	American	23
HBC54	Helena Bonham Carter	British	48
JGL81	Joseph Gordan-Levitt	American	33
KW871	Kate Winslet	Bristish	39
LDC21	Leonardo DiCaprio	American	40
MKE12	Michael Keaton	American	63

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]

```
use classicmodels;

CREATE TABLE employees_audit (
    id INT AUTO_INCREMENT PRIMARY KEY,
    employeeNumber INT NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    changedat DATETIME DEFAULT NULL,
    action VARCHAR(50) DEFAULT NULL
);

CREATE TRIGGER before_employee_update
    BEFORE UPDATE ON employees
    FOR EACH ROW
 INSERT INTO employees_audit
 SET action = 'update',
     employeeNumber = OLD.employeeNumber,
     lastname = OLD.lastname,
     changedat = NOW();

SHOW TRIGGERS;

UPDATE employees
SET
    lastName = 'Phan'
WHERE
    employeeNumber = 1056;

SELECT * FROM employees_audit;
```

Trigger	Event	Table	Statement	Timing	Created	sql_mode
before_employee_update	UPDATE	employees	INSERT INTO employees_audit ↩ SET action = 'update', ↩ employeeNumber = OLD.employeeNumber, ↩ lastname = OLD.lastname, ↩ changedat = NOW()	BEFORE	2021-04-18 14:37:34.60	IGNORE_SPACE,ST

id	employeeNumber	lastname	changedat	acti...
1	1056	Phan	2021-04-18 14:38:29	update
2	1056	Phan	2021-04-18 14:38:43	update