SQL Queries

DataBase

- A **database** is an organized collection of data, so that it can be easily accessed, update and managed.
- Four Basic Operation of Relational Database(organized data in one or more tables) is
  - CRUD

CRUD

C stand for Create

R stand for Read

U stand for Update

D stand for Delete

SQL Data Type

Data Type	Description
char(n)	Fixed width character string
varchar(n)	Variable width character string
int	Allows whole number
bool	Zero is consider as false and 1 is consider true
date	Store date only, e.g january 6,2023

Create Database

CREATING DATABASE:

CREATE DATABASE bank;

VERIFY DATABASE IS CREATED OR NOT:

SHOW DATABASES;

DESCRIBE WHICH DATABASE YOU USED:

USE bank;

CREATING TABLES:

Syntax:

```
CREATE TABLE table_name (

    column1 datatype,

    column2 datatype,

    column3 datatype,

   ....

);
```

CREATING TABLE “Employee”:

```
CREATE TABLE Employee(
emp_id INT,
first_name varchar(20),
last_name VARCHAR(20),
birth_date DATE,
sex varchar(1),
salary INT,
super_id INT,
branch_id INT,
PRIMARY KEY(emp_id)
);
```

VERIFY TABLE CREATED OR NOT:

SHOW TABLES;

SEE PROPERTIES OF “Employee” table:

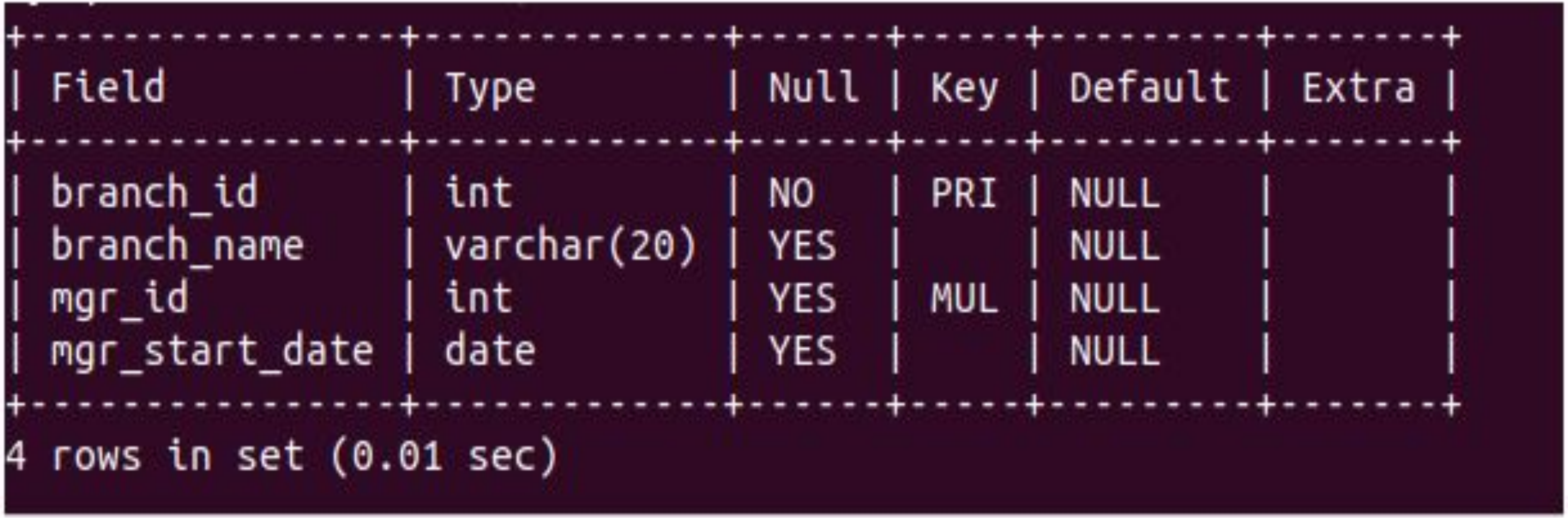
DESCRIBE Employee;

CREATE TABLE “branch” :

```
CREATE TABLE branch(
branch_id INT,
branch_name VARCHAR(20),
mgr_id INT,
mgr_start_date DATE,
PRIMARY KEY(branch_id),
FOREIGN KEY (mgr_id) REFERENCES Employee (emp_id) ON DELETE SET NULL
);
```

SEE PROPERTIES OF “branch” TABLE:

DESCRIBE branch;



```
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| branch_id      | int         | NO   | PRI | NULL    |       |
| branch_name    | varchar(20) | YES  |     | NULL    |       |
| mgr_id         | int         | YES  | MUL | NULL    |       |
| mgr_start_date | date        | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

ALTER TABLE - ADD Column

```
ALTER TABLE table_name

ADD column_name datatype;
```

MAKE CHANGES OF “Employee” table (making “super_id” and “branch_id” foreign key)

```
ALTER TABLE Employee
ADD FOREIGN KEY (super_id) REFERENCES Employee (emp_id)
ON DELETE SET NULL;

ALTER TABLE Employee
ADD FOREIGN KEY (branch_id) REFERENCES branch(branch_id)
ON DELETE SET NULL;
```

ALTER TABLE - DROP COLUMN

To delete a column in a table.

```
ALTER TABLE table_name

DROP COLUMN column_name;
```

DROP Statement

```
DROP DATABASE databasename;

DROP TABLE table_name;
```

INSERT INTO

The **INSERT INTO** statement is used to insert new records in a table.

```
INSERT INTO table_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);
```

```
INSERT INTO table_name

VALUES (value1, value2, value3, ...);
```

SELECT Statement

- **Select all column:**

```
SELECT *
FROM table_name;
```

- **Select selected column:**

```
SELECT column1, column2, ...
FROM table_name;
```

UPDATE Statement

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

BETWEEN Operator

The **BETWEEN** operator selects values within a given range. The values can be numbers, text, or dates.

The **BETWEEN** operator is inclusive: begin and end values are included.

```
SELECT column_name(s)

FROM table_name

WHERE column_name BETWEEN value1 AND value2;
```

IN Operator

The **IN** operator allows you to specify multiple values in a **WHERE** clause.

The **IN** operator is a shorthand for multiple **OR** conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

ORDER BY

The **ORDER BY** keyword is used to sort the result-set in ascending or descending order.

```
SELECT column1, column2, ...

FROM table_name

ORDER BY column1, column2, ... ASC|DESC;
```

LIMIT Clause

The **LIMIT** clause is used to specify the number of records to return.

```
SELECT column_name(s)

FROM table_name

WHERE condition

LIMIT number;
```

AND Operator , OR Operator , NOT Operator

The **WHERE** clause can contain one or many **AND**,**OR** operators.

```
SELECT column1, column2, ...

FROM table_name

WHERE condition1 AND condition2 AND condition3 ...;

WHERE condition1 OR condition2 OR condition3 ...;

WHERE NOT condition;
```

COUNT() Function , SUM() Function , AVG() Function

```
SELECT AVG(column_name)| COUNT(column_name)|
SUM(column_name)

FROM table_name

WHERE condition;
```

GROUP BY Statement

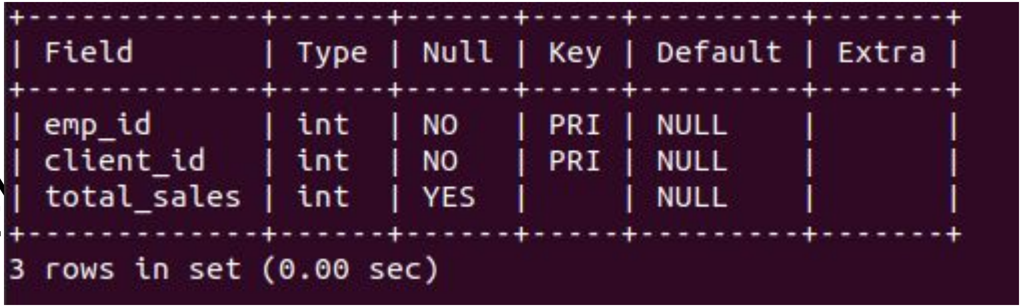
The `GROUP BY` statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The **`GROUP BY`** statement is often used with aggregate functions (`COUNT()`, `MAX()`, `MIN()`, `SUM()`, `AVG()`) to group the result-set by one or more columns.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

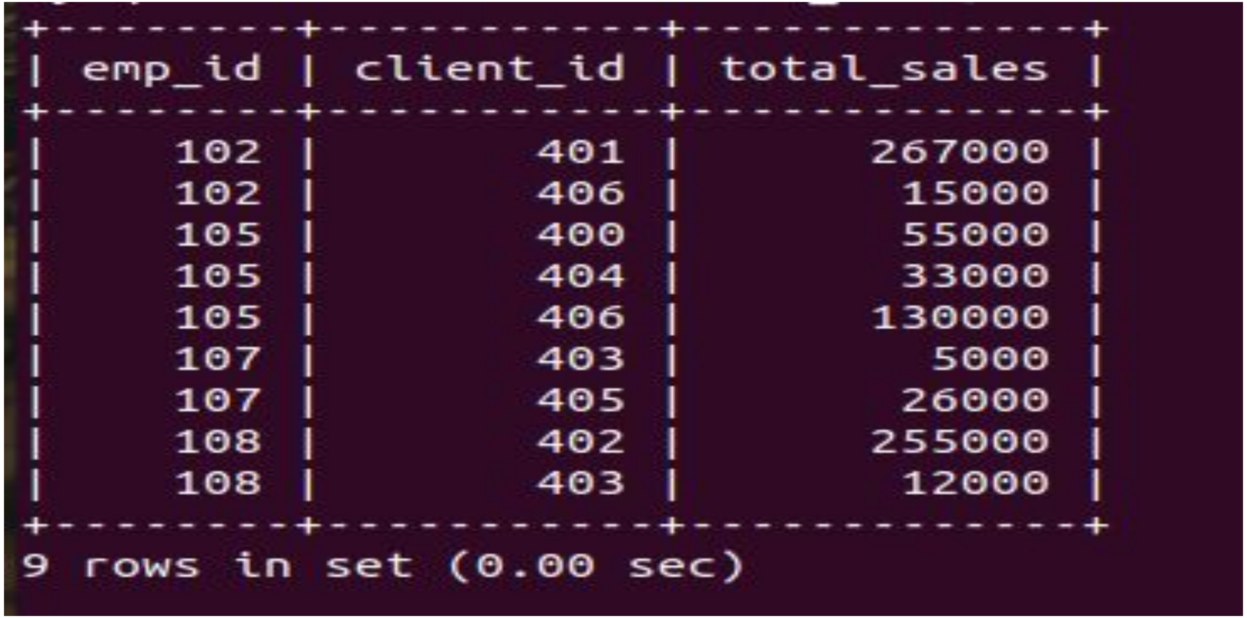
Works_with table

```
CREATE TABLE works_with (
emp_id INT,
client_id INT,
total_sales INT,
PRIMARY KEY(emp_id,client_id),
FOREIGN KEY (emp_id) REFEREN
FOREIGN KEY (client_id) REFERE
);
```



```
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| emp_id      | int  | NO   | PRI | NULL    |       |
| client_id   | int  | NO   | PRI | NULL    |       |
| total_sales | int  | YES  |     | NULL    |       |
+-------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

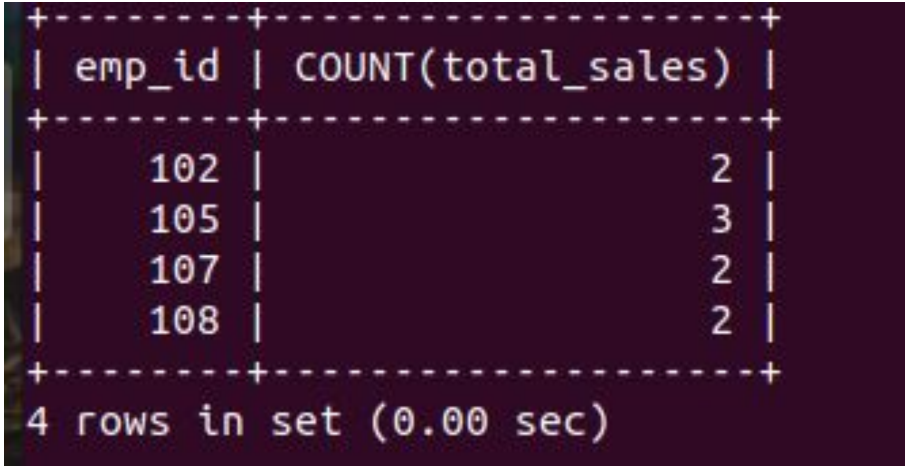
Find how many sales of each salesman:



```
+--------+-----------+-------------+
| emp_id | client_id | total_sales |
+--------+-----------+-------------+
|    102 |       401 |      267000 |
|    102 |       406 |       15000 |
|    105 |       400 |       55000 |
|    105 |       404 |       33000 |
|    105 |       406 |      130000 |
|    107 |       403 |        5000 |
|    107 |       405 |       26000 |
|    108 |       402 |      255000 |
|    108 |       403 |       12000 |
+--------+-----------+-------------+
9 rows in set (0.00 sec)
```

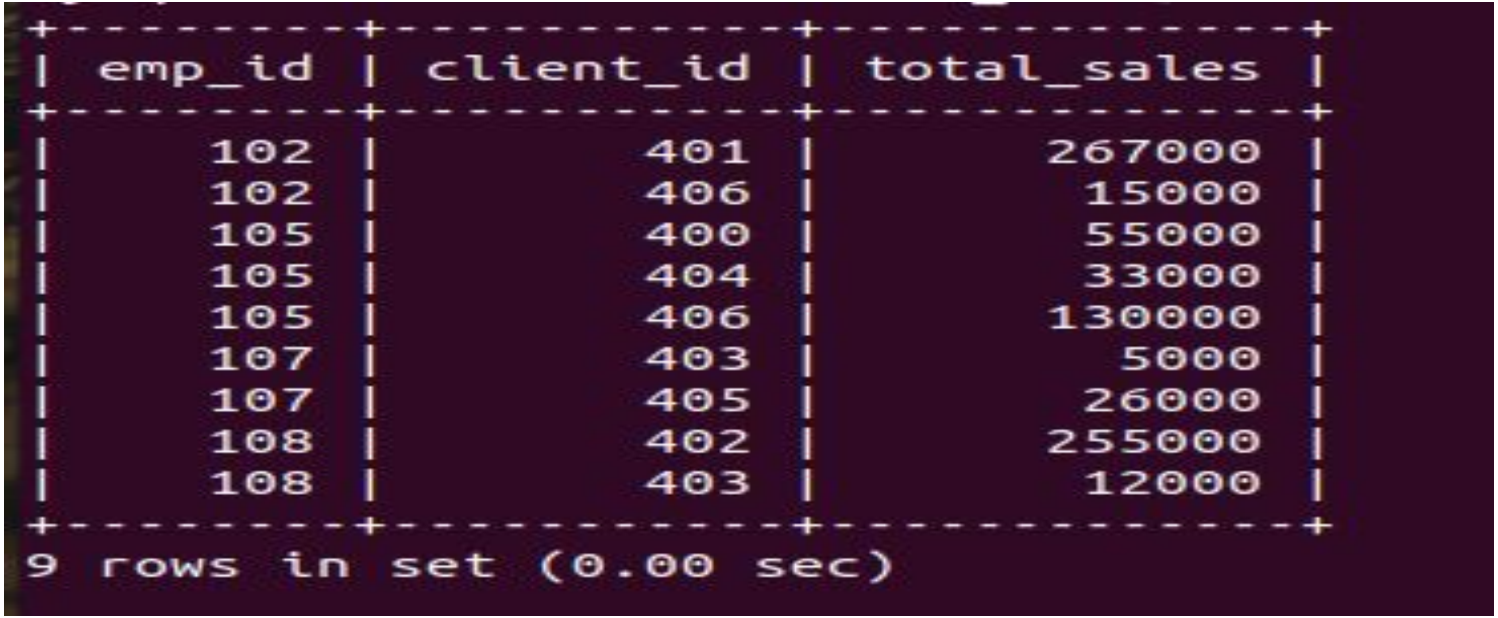
Find how many sales of each salesman:

```
SELECT emp_id,COUNT(total_sales)
FROM works_with
GROUP BY (emp_id);
```



```
+--------+--------------------+
| emp_id | COUNT(total_sales) |
+--------+--------------------+
|    102 |                  2 |
|    105 |                  3 |
|    107 |                  2 |
|    108 |                  2 |
+--------+--------------------+
4 rows in set (0.00 sec)
```

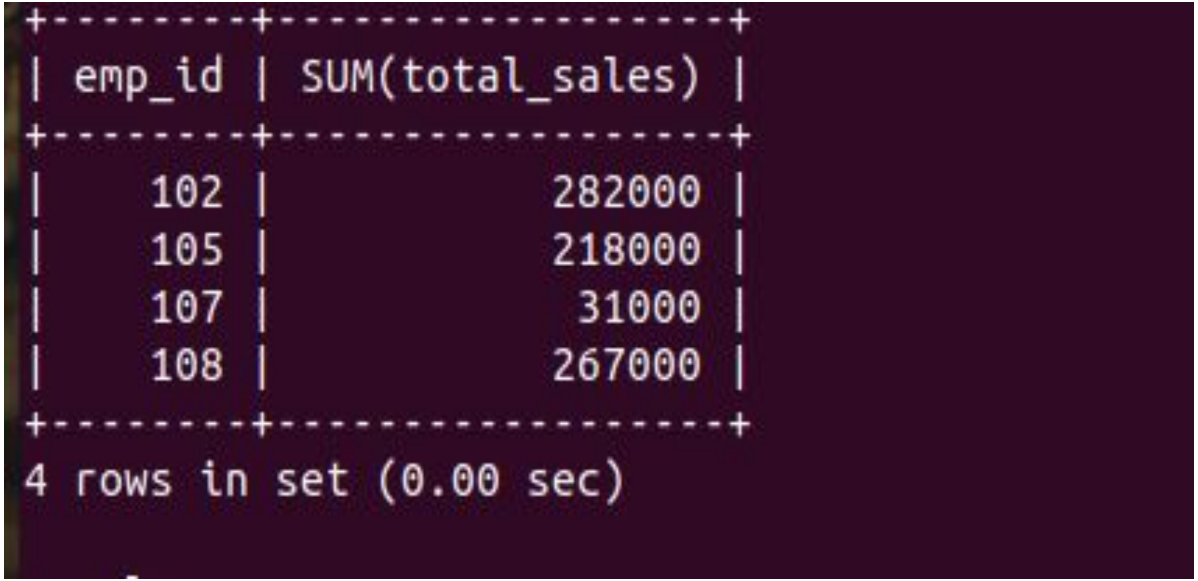
Find total sales of each salesman:



```
+--------+-----------+--------------+
| emp_id | client_id | total_sales  |
+--------+-----------+--------------+
|    102 |       401 |       267000 |
|    102 |       406 |        15000 |
|    105 |       400 |        55000 |
|    105 |       404 |        33000 |
|    105 |       406 |       130000 |
|    107 |       403 |         5000 |
|    107 |       405 |        26000 |
|    108 |       402 |       255000 |
|    108 |       403 |        12000 |
+--------+-----------+--------------+
9 rows in set (0.00 sec)
```

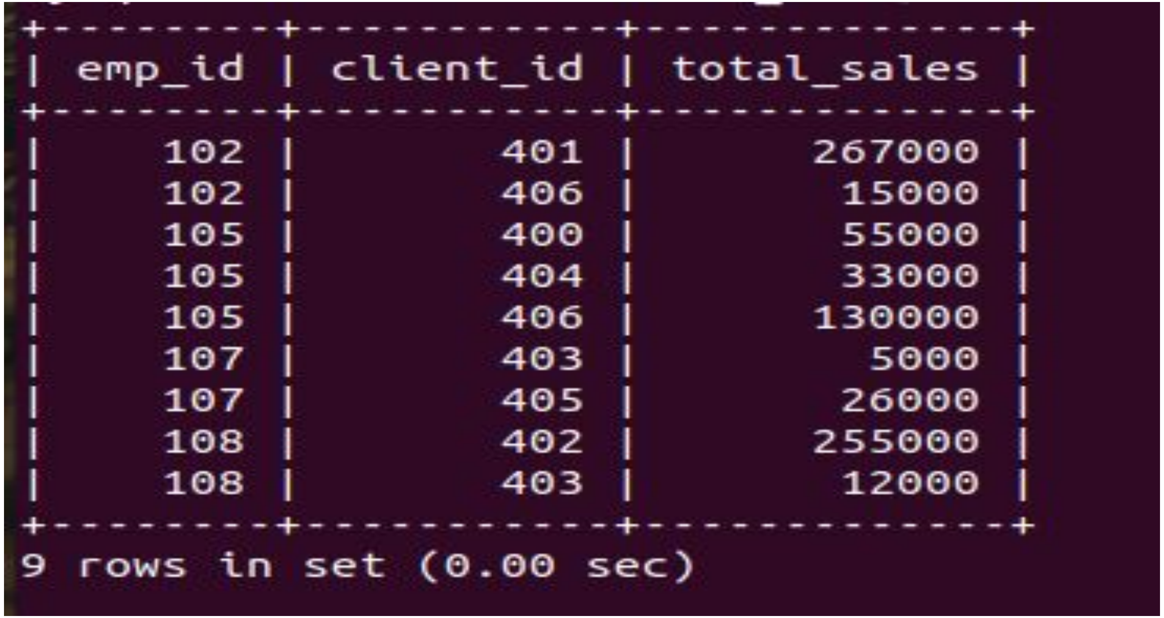
Find total sales of each salesman:

```
SELECT emp_id,SUM(total_sales)
FROM works_with
GROUP BY (emp_id);
```



```
+--------+-------------------+
| emp_id | SUM(total_sales)  |
+--------+-------------------+
|    102 |            282000 |
|    105 |            218000 |
|    107 |             31000 |
|    108 |            267000 |
+--------+-------------------+
4 rows in set (0.00 sec)
```

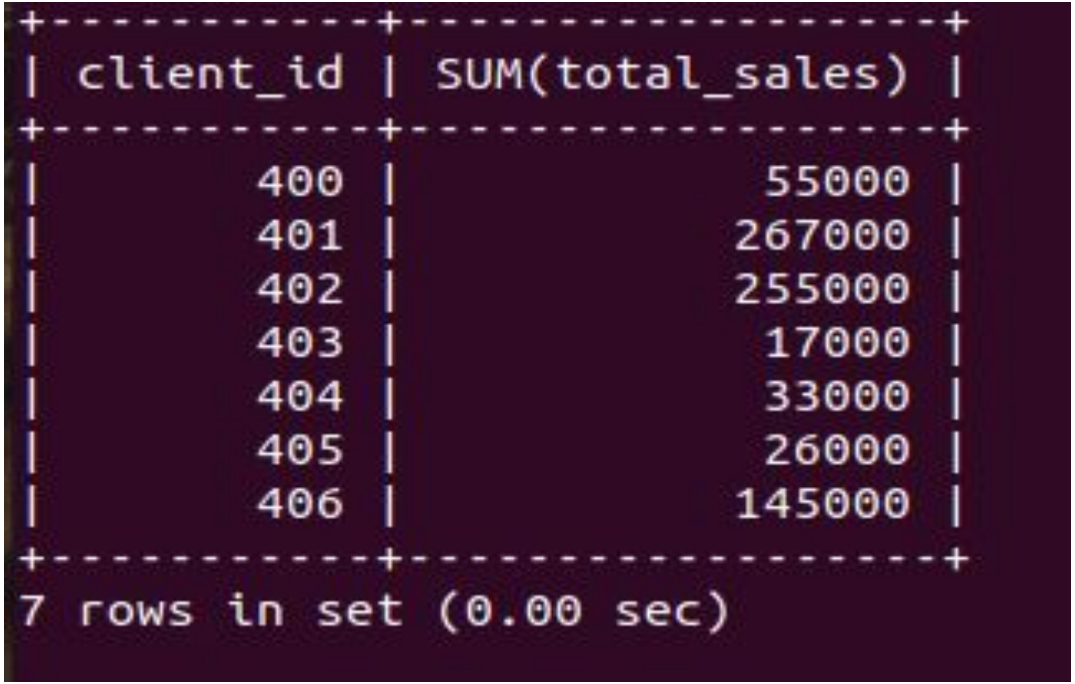
Find total amount of money spend by each client:



```
+--------+-----------+-------------+
| emp_id | client_id | total_sales |
+--------+-----------+-------------+
|    102 |       401 |      267000 |
|    102 |       406 |       15000 |
|    105 |       400 |       55000 |
|    105 |       404 |       33000 |
|    105 |       406 |      130000 |
|    107 |       403 |        5000 |
|    107 |       405 |       26000 |
|    108 |       402 |      255000 |
|    108 |       403 |       12000 |
+--------+-----------+-------------+
9 rows in set (0.00 sec)
```

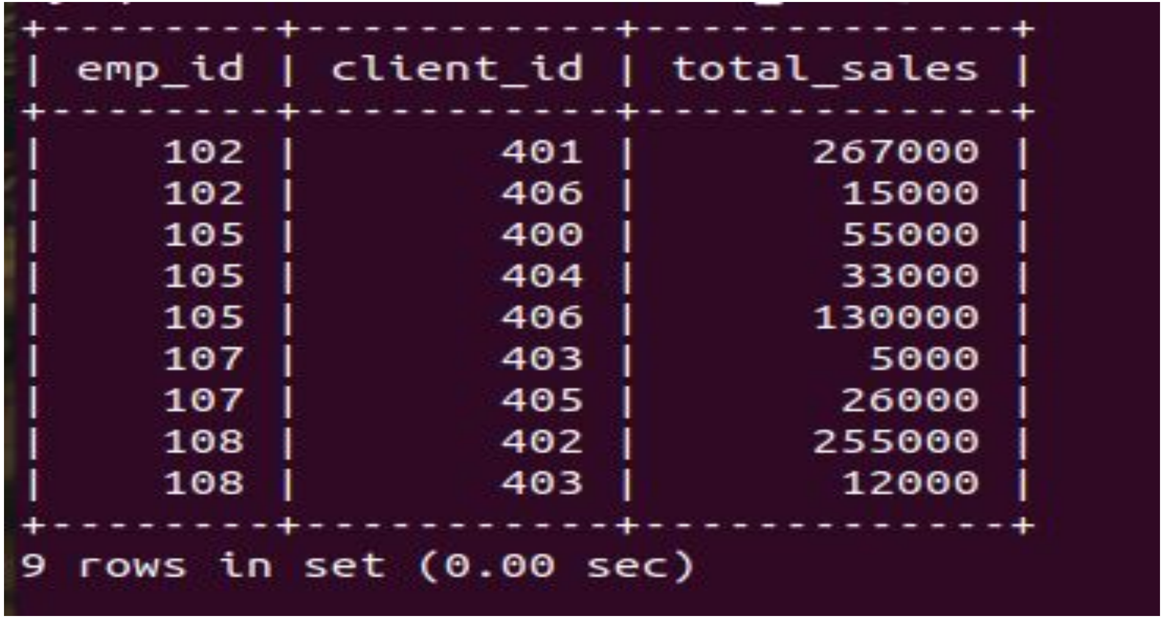
Find total amount of money spend by each client:

```
SELECT client_id,SUM(total_sales)
FROM works_with
GROUP BY client_id;
```



```
+-----------+------------------+
| client_id | SUM(total_sales) |
+-----------+------------------+
|       400 |            55000 |
|       401 |           267000 |
|       402 |           255000 |
|       403 |            17000 |
|       404 |            33000 |
|       405 |            26000 |
|       406 |           145000 |
+-----------+------------------+
7 rows in set (0.00 sec)
```

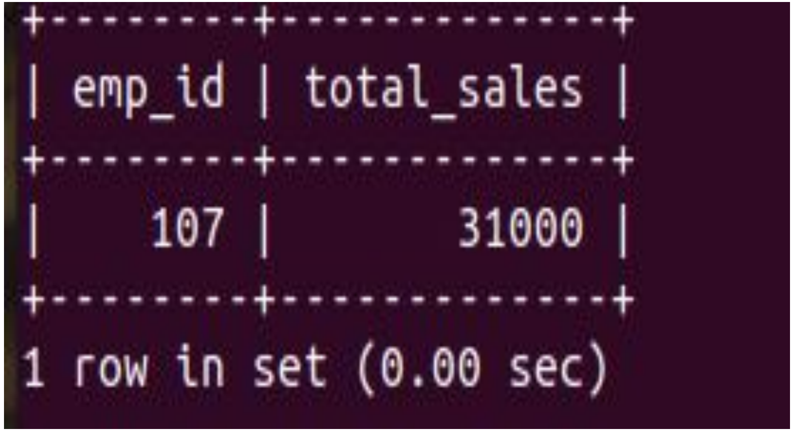
Find which employee did minimum amount of sales:



```
+--------+-----------+-------------+
| emp_id | client_id | total_sales |
+--------+-----------+-------------+
|    102 |       401 |      267000 |
|    102 |       406 |       15000 |
|    105 |       400 |       55000 |
|    105 |       404 |       33000 |
|    105 |       406 |      130000 |
|    107 |       403 |        5000 |
|    107 |       405 |       26000 |
|    108 |       402 |      255000 |
|    108 |       403 |       12000 |
+--------+-----------+-------------+
9 rows in set (0.00 sec)
```

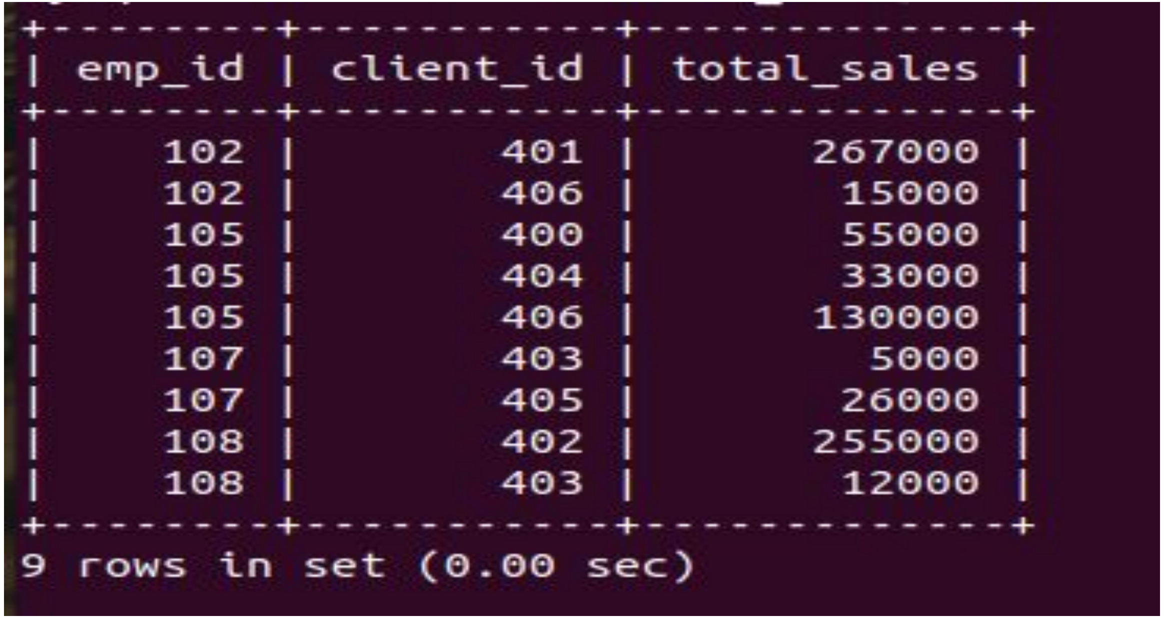
Find which employee did minimum amount of sales:

```
SELECT emp_id,SUM(total_sales) AS total_sales
FROM works_with
GROUP BY (emp_id)
ORDER BY (total_sales)
LIMIT 1;
```



```
+--------+-------------+
| emp_id | total_sales |
+--------+-------------+
|    107 |       31000 |
+--------+-------------+
1 row in set (0.00 sec)
```

Find which employee did total_sales sales greater than 2:



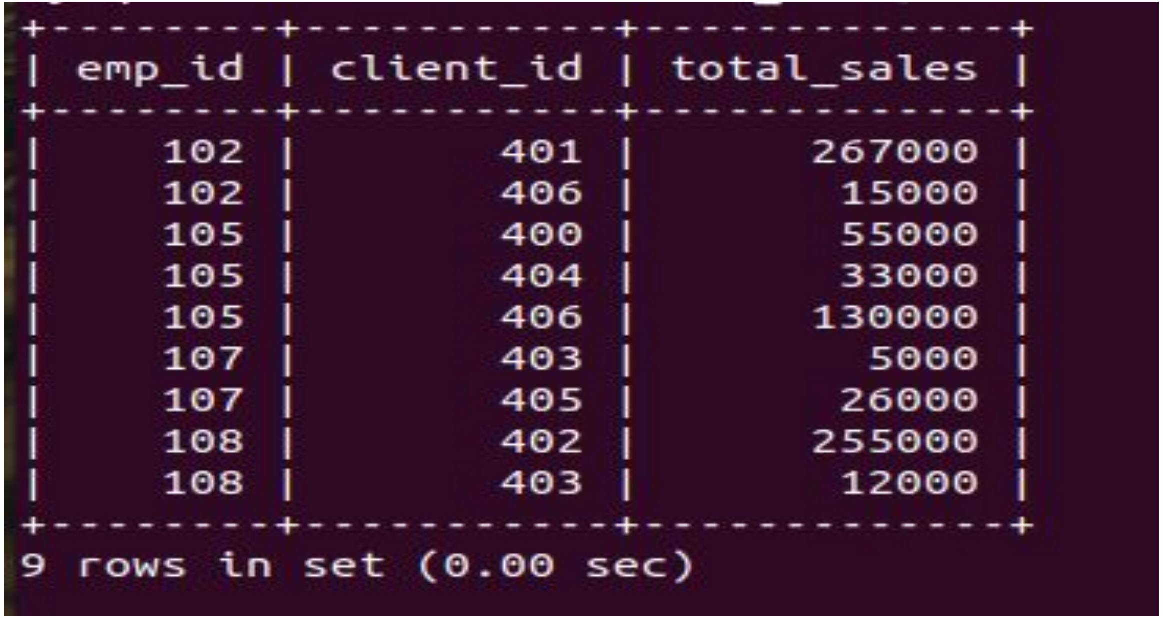
```
+--------+-----------+-------------+
| emp_id | client_id | total_sales |
+--------+-----------+-------------+
|    102 |       401 |      267000 |
|    102 |       406 |       15000 |
|    105 |       400 |       55000 |
|    105 |       404 |       33000 |
|    105 |       406 |      130000 |
|    107 |       403 |        5000 |
|    107 |       405 |       26000 |
|    108 |       402 |      255000 |
|    108 |       403 |       12000 |
+--------+-----------+-------------+
9 rows in set (0.00 sec)
```

Find which employee did total_sales sales greater than 2:

```
SELECT emp_id,SUM(total_sales) AS total_sales
FROM works_with
GROUP BY (emp_id)
HAVING COUNT(*) > 2;
```

emp_id	total_sales
105	207000

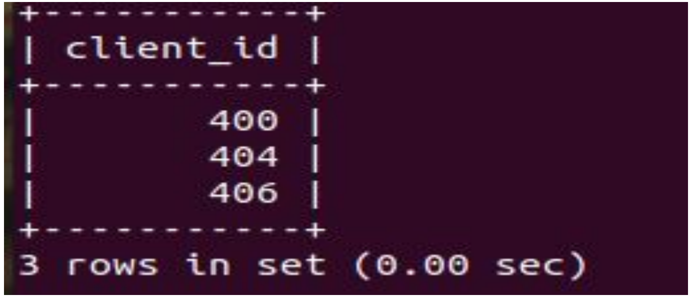
Find which employee did total_sales greater than 2 and find which client id_it works with



```
+--------+------------+--------------+
| emp_id | client_id  | total_sales  |
+--------+------------+--------------+
|    102 |        401 |       267000 |
|    102 |        406 |        15000 |
|    105 |        400 |        55000 |
|    105 |        404 |        33000 |
|    105 |        406 |       130000 |
|    107 |        403 |         5000 |
|    107 |        405 |        26000 |
|    108 |        402 |       255000 |
|    108 |        403 |        12000 |
+--------+------------+--------------+
9 rows in set (0.00 sec)
```

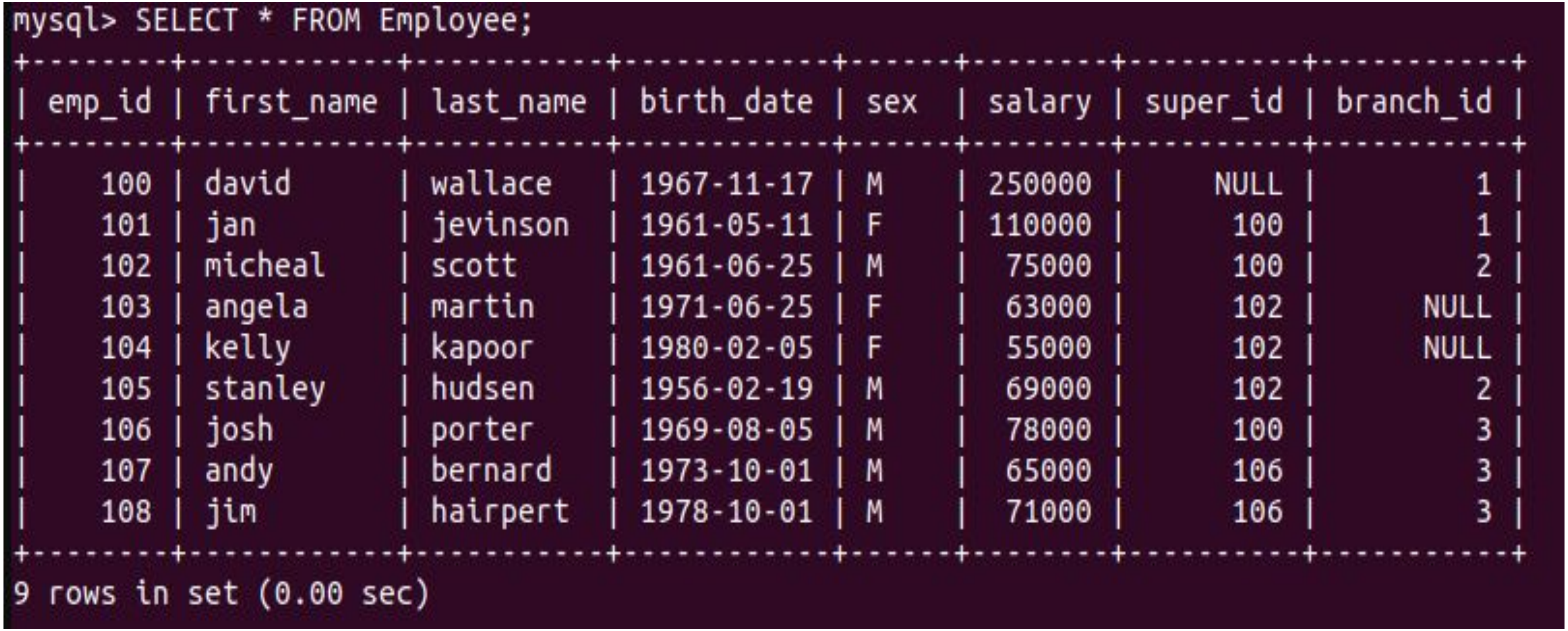
Find which employee did total sales greater than 2 and find which client_id it works with

```
SELECT client_id
FROM works_with
WHERE emp_id IN (
        SELECT emp_id
        FROM works_with
        GROUP BY (emp_id)
        HAVING COUNT(*) > 2
);
```



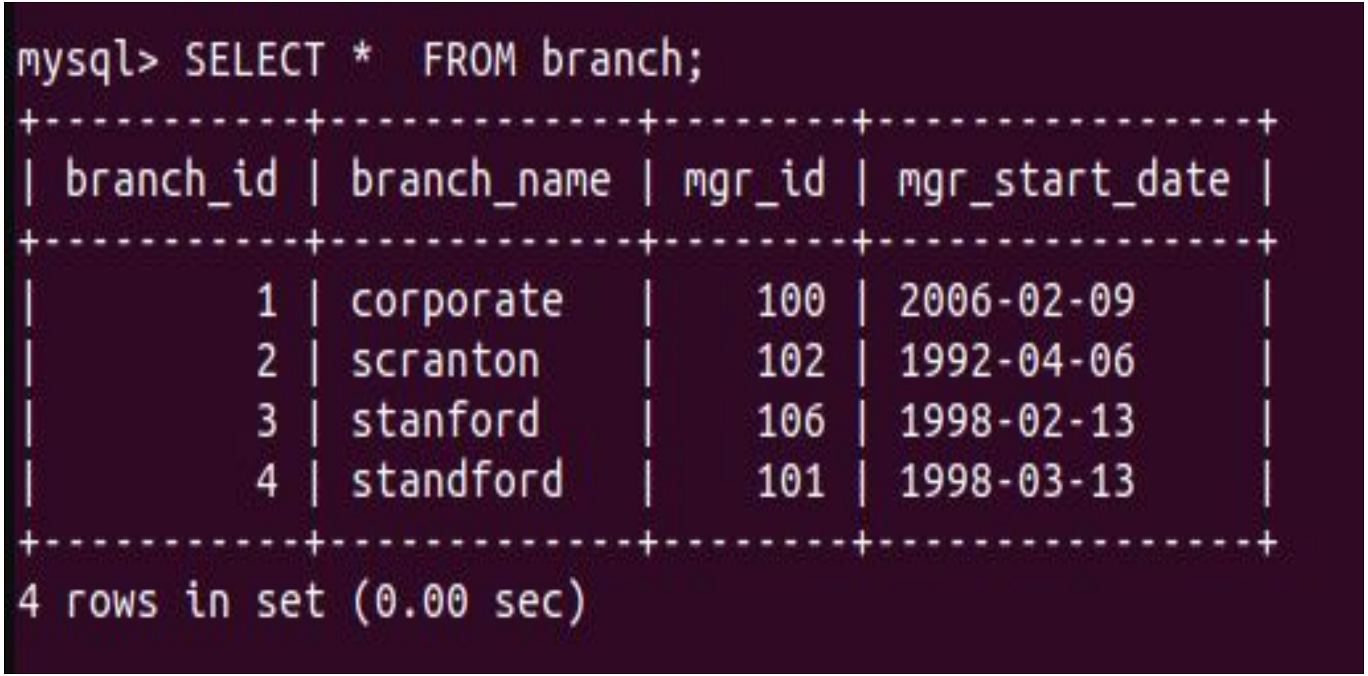
```
+-----------+
| client_id |
+-----------+
|       400 |
|       404 |
|       406 |
+-----------+
3 rows in set (0.00 sec)
```

Employee Table



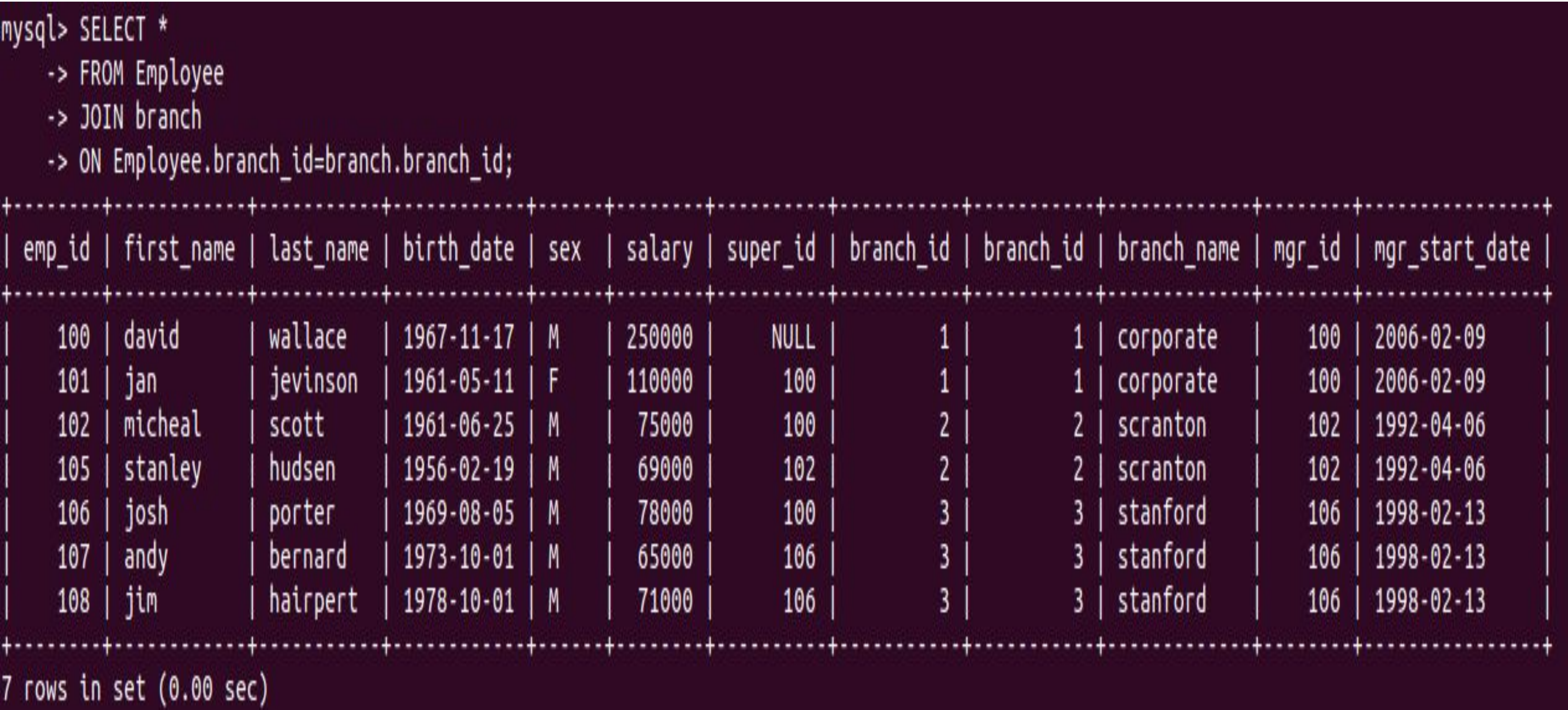
```
mysql> SELECT * FROM Employee;
+--------+------------+-----------+------------+------+--------+----------+-----------+
| emp_id | first_name | last_name | birth_date | sex  | salary | super_id | branch_id |
+--------+------------+-----------+------------+------+--------+----------+-----------+
|    100 | david      | wallace   | 1967-11-17 | M    | 250000 |     NULL |         1 |
|    101 | jan        | jevinson  | 1961-05-11 | F    | 110000 |      100 |         1 |
|    102 | micheal    | scott     | 1961-06-25 | M    |  75000 |      100 |         2 |
|    103 | angela     | martin    | 1971-06-25 | F    |  63000 |      102 |      NULL |
|    104 | kelly      | kapoor    | 1980-02-05 | F    |  55000 |      102 |      NULL |
|    105 | stanley    | hudsen    | 1956-02-19 | M    |  69000 |      102 |         2 |
|    106 | josh       | porter    | 1969-08-05 | M    |  78000 |      100 |         3 |
|    107 | andy       | bernard   | 1973-10-01 | M    |  65000 |      106 |         3 |
|    108 | jim        | hairpert  | 1978-10-01 | M    |  71000 |      106 |         3 |
+--------+------------+-----------+------------+------+--------+----------+-----------+
9 rows in set (0.00 sec)
```

Branch Table



```
mysql> SELECT *  FROM branch;
+-----------+-------------+--------+----------------+
| branch_id | branch_name | mgr_id | mgr_start_date |
+-----------+-------------+--------+----------------+
|         1 | corporate   |    100 | 2006-02-09     |
|         2 | scranton    |    102 | 1992-04-06     |
|         3 | stanford    |    106 | 1998-02-13     |
|         4 | standford   |    101 | 1998-03-13     |
+-----------+-------------+--------+----------------+
4 rows in set (0.00 sec)
```

JOIN



```
mysql> SELECT *
    -> FROM Employee
    -> JOIN branch
    -> ON Employee.branch_id=branch.branch_id;
+--------+------------+-----------+------------+------+--------+----------+-----------+-----------+-------------+--------+----------------+
| emp_id | first_name | last_name | birth_date | sex  | salary | super_id | branch_id | branch_id | branch_name | mgr_id | mgr_start_date |
+--------+------------+-----------+------------+------+--------+----------+-----------+-----------+-------------+--------+----------------+
|    100 | david      | wallace   | 1967-11-17 | M    | 250000 |     NULL |         1 |         1 | corporate   |    100 | 2006-02-09     |
|    101 | jan        | jevinson  | 1961-05-11 | F    | 110000 |      100 |         1 |         1 | corporate   |    100 | 2006-02-09     |
|    102 | micheal    | scott     | 1961-06-25 | M    |  75000 |      100 |         2 |         2 | scranton    |    102 | 1992-04-06     |
|    105 | stanley    | hudsen    | 1956-02-19 | M    |  69000 |      102 |         2 |         2 | scranton    |    102 | 1992-04-06     |
|    106 | josh       | porter    | 1969-08-05 | M    |  78000 |      100 |         3 |         3 | stanford    |    106 | 1998-02-13     |
|    107 | andy       | bernard   | 1973-10-01 | M    |  65000 |      106 |         3 |         3 | stanford    |    106 | 1998-02-13     |
|    108 | jim        | hairpert  | 1978-10-01 | M    |  71000 |      106 |         3 |         3 | stanford    |    106 | 1998-02-13     |
+--------+------------+-----------+------------+------+--------+----------+-----------+-----------+-------------+--------+----------------+
7 rows in set (0.00 sec)
```

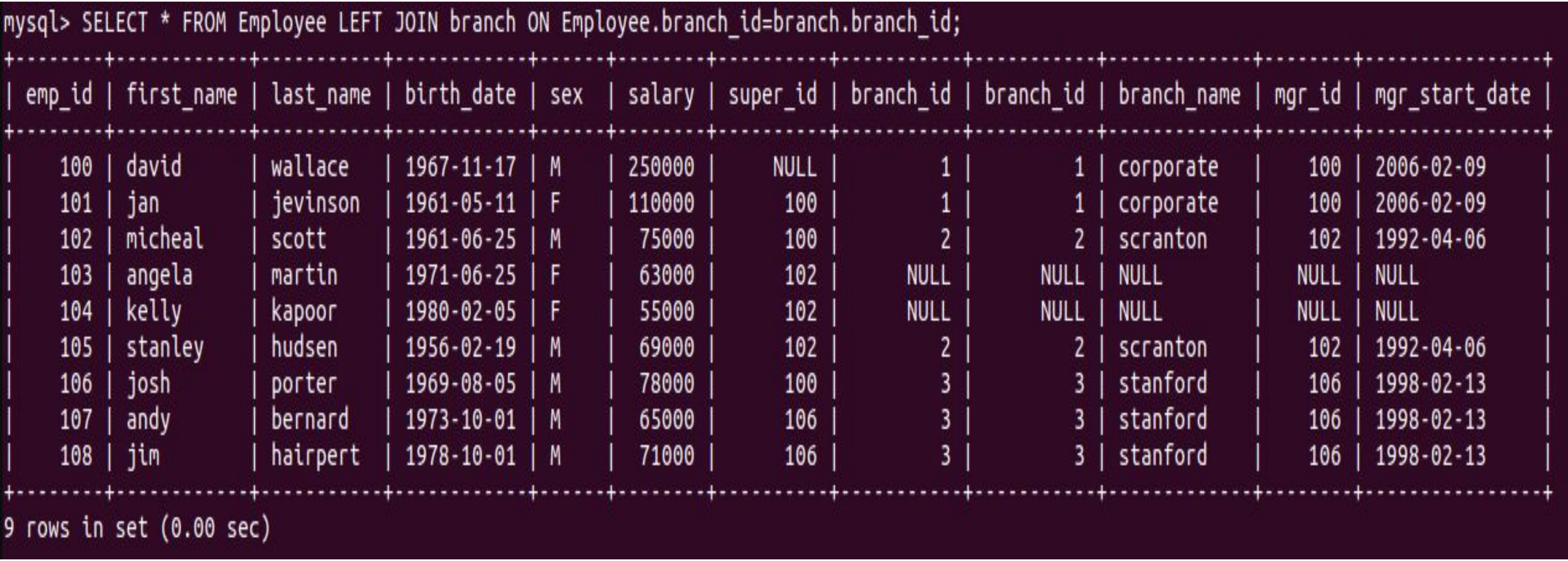
Join through RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

Rdd1=spark.read.csv("bank.csv”,sep= ";",inferSchema=True,header=True)

Rdd2=Rdd.join(RDD1,Rdd.bank_id==Rdd1.bank_id)
```

LEFT JOIN



```
mysql> SELECT * FROM Employee LEFT JOIN branch ON Employee.branch_id=branch.branch_id;
+--------+------------+-----------+------------+-----+--------+----------+-----------+-----------+-------------+--------+----------------+
| emp_id | first_name | last_name | birth_date | sex | salary | super_id | branch_id | branch_id | branch_name | mgr_id | mgr_start_date |
+--------+------------+-----------+------------+-----+--------+----------+-----------+-----------+-------------+--------+----------------+
|    100 | david      | wallace   | 1967-11-17 | M   | 250000 |     NULL |         1 |         1 | corporate   |    100 | 2006-02-09     |
|    101 | jan        | jevinson  | 1961-05-11 | F   | 110000 |      100 |         1 |         1 | corporate   |    100 | 2006-02-09     |
|    102 | micheal    | scott     | 1961-06-25 | M   |  75000 |      100 |         2 |         2 | scranton    |    102 | 1992-04-06     |
|    103 | angela     | martin    | 1971-06-25 | F   |  63000 |      102 |      NULL |      NULL | NULL        |   NULL | NULL           |
|    104 | kelly      | kapoor    | 1980-02-05 | F   |  55000 |      102 |      NULL |      NULL | NULL        |   NULL | NULL           |
|    105 | stanley    | hudsen    | 1956-02-19 | M   |  69000 |      102 |         2 |         2 | scranton    |    102 | 1992-04-06     |
|    106 | josh       | porter    | 1969-08-05 | M   |  78000 |      100 |         3 |         3 | stanford    |    106 | 1998-02-13     |
|    107 | andy       | bernard   | 1973-10-01 | M   |  65000 |      106 |         3 |         3 | stanford    |    106 | 1998-02-13     |
|    108 | jim        | hairpert  | 1978-10-01 | M   |  71000 |      106 |         3 |         3 | stanford    |    106 | 1998-02-13     |
+--------+------------+-----------+------------+-----+--------+----------+-----------+-----------+-------------+--------+----------------+
9 rows in set (0.00 sec)
```

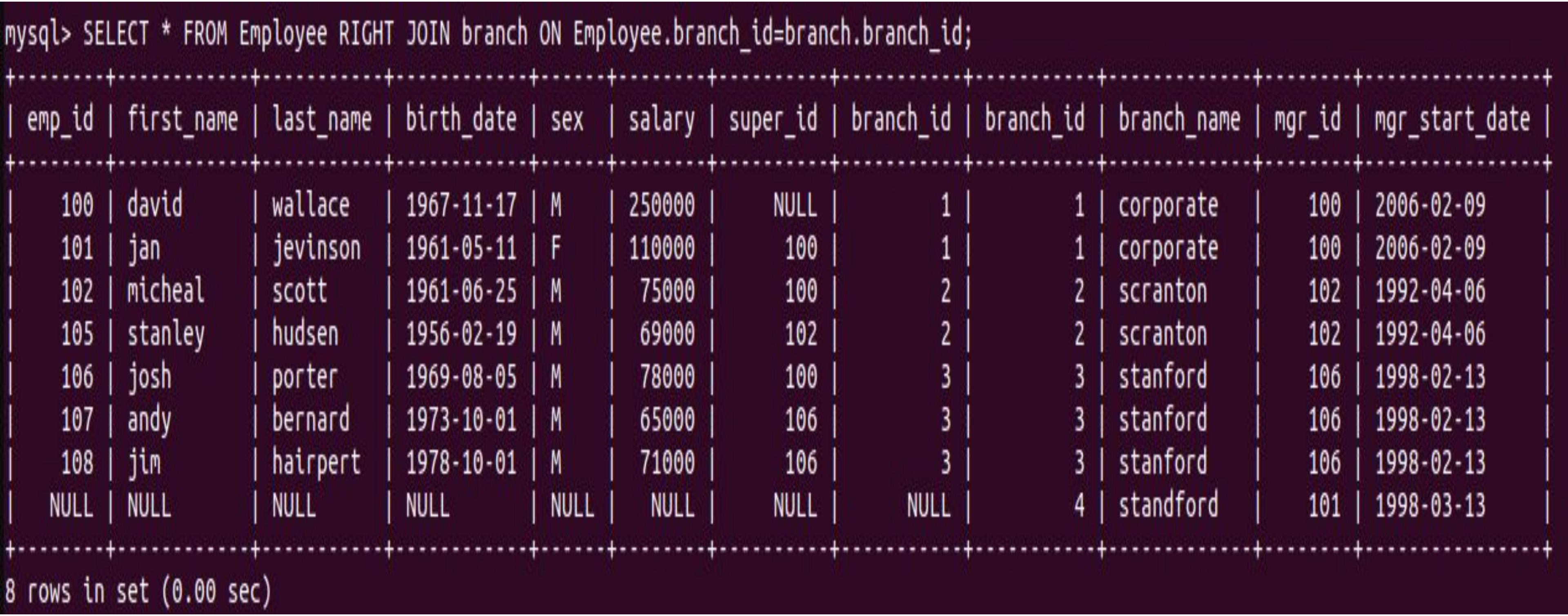
Left join through RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

Rdd1=spark.read.csv("bank.csv”,sep= ";",inferSchema=True,header=True)

Rdd2=Rdd.leftOuterJoin(RDD1,Rdd.bank_id==Rdd1.bank_id)
```

Right Join



```
mysql> SELECT * FROM Employee RIGHT JOIN branch ON Employee.branch_id=branch.branch_id;
+--------+------------+-----------+------------+------+--------+----------+-----------+-----------+-------------+--------+----------------+
| emp_id | first_name | last_name | birth_date | sex  | salary | super_id | branch_id | branch_id | branch_name | mgr_id | mgr_start_date |
+--------+------------+-----------+------------+------+--------+----------+-----------+-----------+-------------+--------+----------------+
|    100 | david      | wallace   | 1967-11-17 | M    | 250000 |     NULL |         1 |         1 | corporate   |    100 | 2006-02-09     |
|    101 | jan        | jevinson  | 1961-05-11 | F    | 110000 |      100 |         1 |         1 | corporate   |    100 | 2006-02-09     |
|    102 | micheal    | scott     | 1961-06-25 | M    |  75000 |      100 |         2 |         2 | scranton    |    102 | 1992-04-06     |
|    105 | stanley    | hudsen    | 1956-02-19 | M    |  69000 |      102 |         2 |         2 | scranton    |    102 | 1992-04-06     |
|    106 | josh       | porter    | 1969-08-05 | M    |  78000 |      100 |         3 |         3 | stanford    |    106 | 1998-02-13     |
|    107 | andy       | bernard   | 1973-10-01 | M    |  65000 |      106 |         3 |         3 | stanford    |    106 | 1998-02-13     |
|    108 | jim        | hairpert  | 1978-10-01 | M    |  71000 |      106 |         3 |         3 | stanford    |    106 | 1998-02-13     |
|   NULL | NULL       | NULL      | NULL       | NULL |   NULL |     NULL |      NULL |         4 | standford   |    101 | 1998-03-13     |
+--------+------------+-----------+------------+------+--------+----------+-----------+-----------+-------------+--------+----------------+
8 rows in set (0.00 sec)
```

Right join through RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

Rdd1=spark.read.csv("bank.csv”,sep= ";",inferSchema=True,header=True)

Rdd2=Rdd.rightOuterJoin(RDD1,Rdd.bank_id==Rdd1.bank_id)
```

Find all Employees:

```
sql_cmd= “SELECT * FROM Employee”
df=spark.sql(sql_cmd)
df.show()
```

RDD:

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

Rdd.show()
```

Find first and last name of all employees:

sql_cmd=SELECT first_name,last_name **FROM** Employee”
df=spark.sql(sql_cmd)
df.show()

RDD

Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.select(Rdd.first_name,Rdd.last_name)

rdd1.show()

Find fore name and sur name of all employees:

sql_cmd=“SELECT first_name **AS** forename,last_name **AS** surname **FROM** Employee”

df=spark.sql(sql_cmd)

df.show()

RDD

Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.select(Rdd.first_name,Rdd.last_name).alais(“forename”,”sur name”)

rdd1.show()

Find all employees ordered by salary (ascending order):

```
sql_cmd= “SELECT * FROM Employee ORDER BY salary ASC”
df=spark.sql(sql_cmd)
df.show()
```

RDD:

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.select(Rdd.salary , ascending=True)

rdd1.show()
```

Find all male employees:

```
sql_cmd= “SELECT * FROM Employee WHERE sex = 'M' ”
df=spark.sql(sql_cmd)
df.show()
```

RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.filter(Rdd.sex=’M’)

rdd1.show()
```

Find all employees from “branch 2” and “sex = F” :

sql_cmd= **“SELECT** * **FROM** Employee **WHERE** branch_id =2 **AND** sex = 'F' ”

df=spark.sql(sql_cmd)

df.show()

RDD

Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.filter(Rdd.branch_id=2 **and** Rdd.sex=’F’)

rdd1.show()

Find out all distinct gender:

sql_cmd= “SELECT DISTINCT sex **FROM** Employee”

```
df=spark.sql(sql_cmd)
df.show()
```

RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.select(Rdd.sex).distinct()

rdd1.show()
```

Find number of employee:

```
Sql_cmd = “SELECT COUNT(emp_id) FROM Employee”
df=spark.sql(sql_cmd)
df.show()
```

RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.count()

rdd1.show()
```

Find average of all employee salaries:

```
sql_cmd= “SELECT AVG(salary) FROM Employee”
df=spark.sql(sql_cmd)
df.show()
```

RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.select(avg(Rdd.salary))

rdd1.show()
```

Find sum of all employee salaries:

```
Sql_cmd = “SELECT SUM(salary) FROM Employee”
df=spark.sql(sql_cmd)
df.show()
```

RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.select(sum(Rdd.salary))

rdd1.show()
```

Find number of distinct sex in employee table:

```
sql_cmd = “SELECT COUNT(DISTINCT sex) FROM Employee”
df=spark.sql(sql_cmd)
df.show()
```

RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)

rdd1=Rdd.select(Rdd.sex).distinct().count()

rdd1.show()
```

Find how many sex in employee table:

```
sql_cmd = “SELECT COUNT(sex) ,sex FROM Employee GROUP BY sex”
df=spark.sql(sql_cmd)
df.show()
```

RDD

```
Rdd = spark.read.csv("Employee.csv”,sep= ";",inferSchema=True,header=True)
rdd1=Rdd.groupBy(Rdd.sex).count()
rdd1.show()
```

Find out total sales of each client name:

```
sql_cmd = “

SELECT client.client_name, SUM(works_with.total_sales) AS sales

FROM client

JOIN works_with

ON client.client_id = works_with.client_id

GROUP BY works_with.client_id

df=spark.sql(sql_cmd)

df.show()
```

RDD:

```
Rdd = spark.read.csv("client.csv”,sep=";",inferSchema=True,header=True)

Rdd1=
spark.read.csv(“works_with.csv”,,sep=";,inferSchema=True,header=True)

rdd2=Rdd.join(Rdd1).groupBy(Rdd1.client_id).sum(Rdd1.total_sales).select
(Rdd.client_name,Rdd1.total_sales).show()
```