Implementing SQL Queries Using MySQL - II

Session 8

Objectives

- *Explain the use of keys in a table*
- *Explain the use of indexes in a table*
- *Explain modification of tables*
- *Explain the use of the ORDER BY command*
- *Explain the use of the GROUP BY command*

Defining Keys

- Keys are columns that uniquely identify information present in a table
- Some of the different types of keys include primary key, composite key, and foreign key

- **Primary key**
  - A primary key is used to uniquely identify each row in a table
  - A primary key specifies that there cannot be a column in a table that contains two similar values
  - The primary key does not allow blank or `NULL` values for the column
- You must remember the following rules while defining a primary key:
  - Contains a value and cannot contain a null value
  - Is unique for each record

Defining Keys

- The syntax for defining a primary key while creating a table is:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name col_name
column definition PRIMARY KEY (fieldname);
```

where,

`CREATE` – adds an object to the database

`TABLE` – adds a table to the database

`tbl_name` – specifies the name for the table to add to the database

`PRIMARY KEY` – creates a primary key on the specified column

`fieldname` – specifies the name of the column to define the primary key

Defining Keys

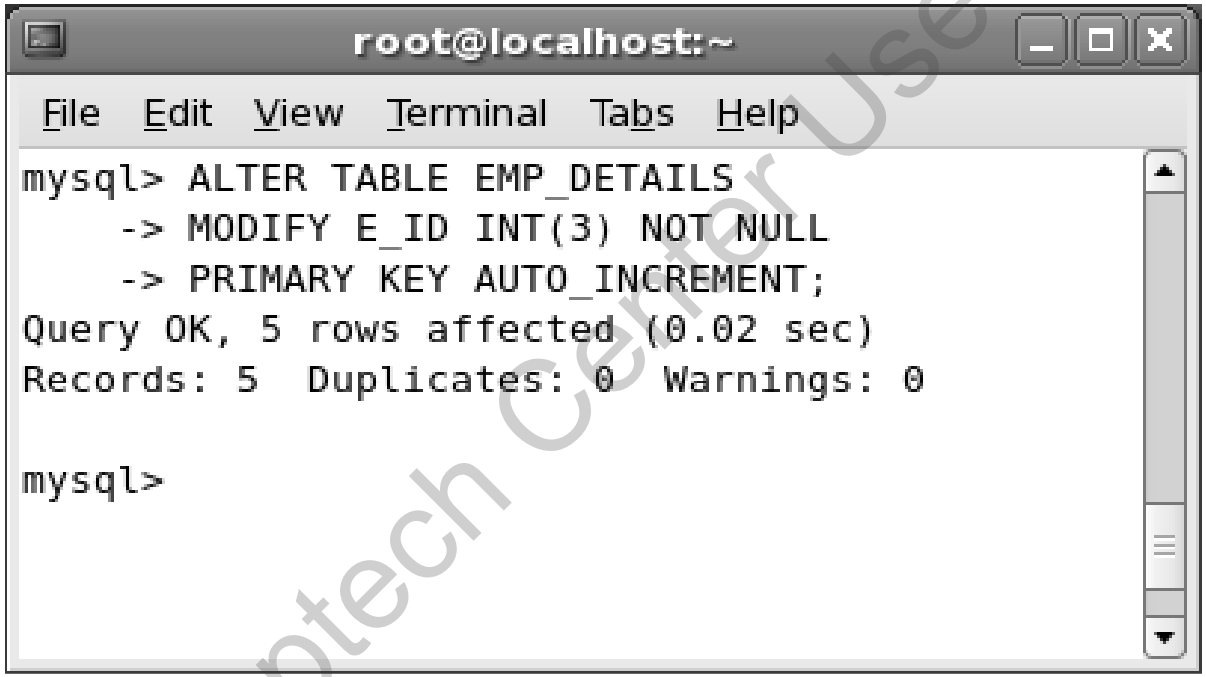
- MySQL provides different ways to define a primary key for a table:
  - You can define the primary key on a table while creating it
  - You can also modify an existing table structure and define a primary key using the `ALTER TABLE` command
  - You can modify the existing primary key on a table and specify conditions for the primary key

Defining Keys

- For example, You have already created a primary key for the `EMP_DETAILS` table
- You will now modify this key as `NOT NULL` and `AUTO_INCREMENT` so that this primary key does not accept or contain a NULL value
- To modify the primary key as `NOT NULL` and `AUTO_INCREMENT`, enter the following command at the command prompt:

```
ALTER TABLE EMP_DETAILS MODIFY E_ID INT(3)NOT
NULL PRIMARY KEY AUTO_INCREMENT;
```

Figure displays the output of the command



- **Composite Key**
  - When you use more than one column to define a primary key, then this type of a primary key is known as a composite key
  - The rules for defining a composite key are the same as the primary key

- **Foreign Key**
  - A foreign key is used to establish relationship between two tables
  - A foreign key is a column in a table that has a corresponding primary key column in a different table
  - The main goal of foreign key is to maintain referential integrity of the data
  - You can define a column as a foreign key only if both the tables have `InnoDB` as their storage engine
  - By default, MySQL assigns the `MyISAM` storage engine when you create a table
  - you can change the storage engine of the table using the `ALTER TABLE` command

Defining Keys

- The advantages of using `InnoDB` as the storage engine are:
  - Cancels `INSERT` and `UPDATE` commands that attempt to add records to the child table without having a corresponding record in the parent table
  - Allows the foreign key to reference a group of columns in the parent table
- To modify the storage engine of the `EMP_DETAILS` to `InnoDB`, enter the following command at the command prompt:

```
ALTER TABLE EMP_DETAILS ENGINE=INNODB;
```

- To modify the `EMP_SALARY` storage engine to `InnoDB`, enter the following command at the command prompt:

```
ALTER TABLE EMP_SALARY ENGINE=INNODB;
```

Defining Keys

- The syntax to define a foreign key is:

```
ALTER TABLE tbl_name ADD [Constraint symbol] FOREIGN KEY
[index_name](index_col_name,...) REFERENCES table_name
(index_col_name,...) [ON DELETE {CASCADE|SET NULL|NO
ACTION|RESTRICT}] [ON UPDATE {CASCADE|SET NULL|NO
ACTION|RESTRICT}];
```

where,

`ALTER TABLE` – modifies the table structure

`tbl_name` – specifies the name of the table to edit

`ADD` – appends an object to the table structure

`FOREIGN KEY` – defines the foreign key to the column

`index_name` – specifies the name of the index

`index_col_name` – specifies the name of the column defined as index

`REFERENCES` – defines the relationship with a primary key of another table

The options for defining a foreign key are listed in the table

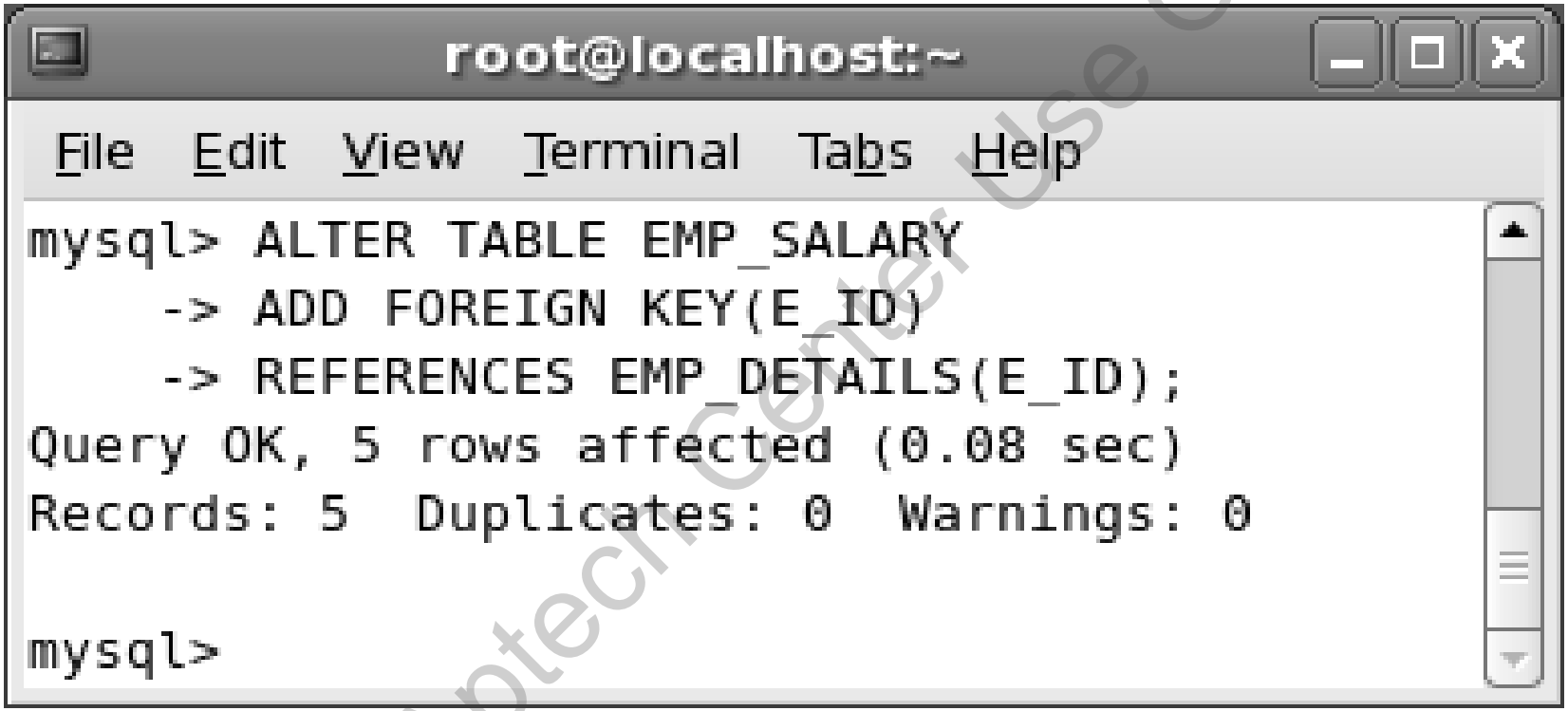
Option	Description
ON DELETE	Deletes the child row if the related parent row is deleted
ON UPDATE	Modifies the child row if the related parent row is modified
CASCADE	Delete or update the row in the parent table and automatically delete or update corresponding rows in the child table
SET NULL	Deletes or modifies the row in the parent table and sets the corresponding row in the child table to NULL provided NOT NULL is not specified in the foreign key column
NO ACTION	Prevents a delete or update to the primary key if a foreign key value exists
RESTRICT	Prevents the deletion or updation of row in the parent table if a dependent child row exits

Defining Keys

- To define a foreign key, enter the following command at the command prompt:

```
ALTER TABLE EMP_SALARY
ADD FOREIGN KEY(E_ID)
REFERENCES EMP_DETAILS(E_ID);
```

Figure displays the output of the command



Creating Indexes

- You can use indexes to improve the search characteristics of the database
- Indexing enables the search operation on a database to be faster
- MySQL creates separate files to store and track indexes when you define an index for a table
- MySQL allows upto 64 indexes for each table
- Each index allows upto 16 columns to be included

- The disadvantages of indexing are:
  - The process of altering data in a database becomes slow
  - The process such as update, insert, and delete require more time to execute
  - MySQL requires this additional time to record the changes in the index file as well

Creating Indexes

- You can index the field that:
  - Is used in the `WHERE` clause of query
  - Is used in `ORDER BY` clause of the query
  - Is having a unique value
- If you do not specify an index key, MySQL automatically indexes the primary key

- The syntax to create an index for a table is:

```
CREATE INDEX index_name ON
tablename(column1,column2,..., columnN);
```

where,

`CREATE INDEX` – appends an index to the

`tableindex_name` – specifies the name for the index

`tablename` – specifies the name of the table to create the index

`column1` – specifies the names of the columns to be indexed in the table

Creating Indexes

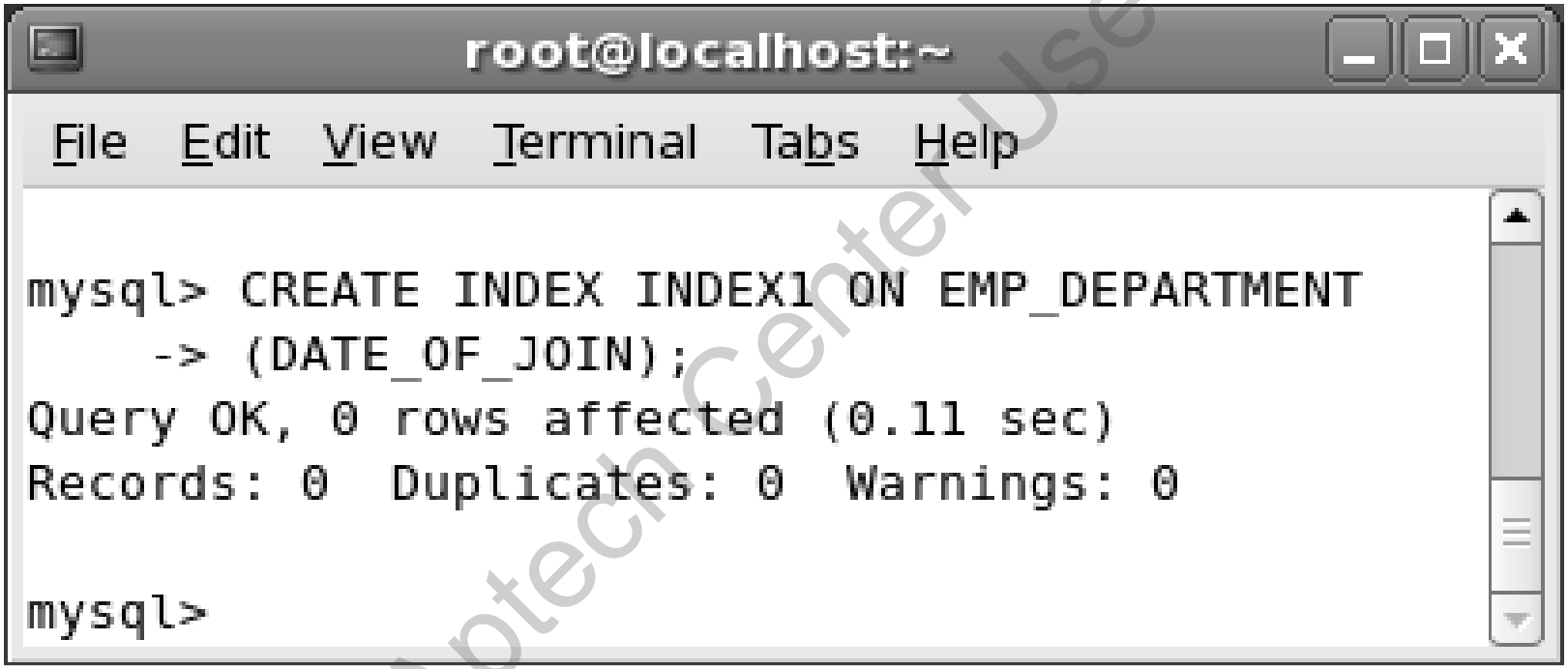
- To create an index named INDEX1 on the columns ID and NAME while creating a SAMPLE table, enter the following command at the command prompt:

```
CREATE TABLE SAMPLE (ID INT(2)NOT NULL,NAME
CHAR(10),INDEX INDEX1(ID,NAME), UNIQUE
INDEX(ID));
```

- You will now create an index on the DATE_OF_JOIN field in the EMP_DEPARTMENT table to search data
- To create an index, INDEX1 on EMP_DEPARTMENT, enter the following command at the command prompt:

```
CREATE INDEX INDEX1 ON
EMP_DEPARTMENT(DATE_OF_JOIN);
```

Figure displays the output of the command



Inserting Data into a Table

- MySQL provides the `INSERT` command to add new records into a table
- MySQL appends the newly inserted record after the last record in the table if the table already contains records
- The syntax to insert data into a table is:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO] tbl_name
[(col_name,...)] VALUES ({expression |
DEFAULT},...),(...),...[ON DUPLICATE KEY UPDATE
col_name=expression,...];
```

where,

`INSERT` – adds new row to an existing table

`tbl_name` – specifies the name of the table where MySQL will append the row

`col_name` – specifies the name of the column where MySQL will add the data

`VALUES` – defines the data that will be added to the table

The options for the INSERT command are listed in table

Option	Description
LOW_PRIORITY	Postpones addition of records till all operations (reading) from the client have completed
DELAYED	Delays the insertion against all incoming SELECT statements
IGNORE	Overrides the error conditions for columns and adds the record to the table
INTO	Specifies the name of the table into which the columns are to be inserted
col_name	Specifies the name of the columns to insert values
expression	Includes a mathematical expression for a column
DEFAULT	Stores the default value into the specified columns
ON DUPLICATE KEY UPDATE col_name=expression	Updates the existing record if the addition of a new record creates a duplicate value in the PRIMARY KEY column

Inserting Data into a Table

- MySQL allows you to insert values for specific column names
- All other columns, if not specified, will contain a `NULL` value
- You have created columns to contain default data, then you can use the `DEFAULT` clause of the `INSERT` command to specify a particular column name and the value for that column
- An error occurs if the column does not have a default value

Inserting Data into a Table

- To insert rows in EMP_DEPARTMENT, enter the following command at the command prompt:

```
INSERT INTO EMP_DEPARTMENT (E_ID, D_NAME,
DATE_OF_JOIN, DESIGNATION) VALUES (108, 'MARKETING',
'2000-05-15', 'MANAGER');
```

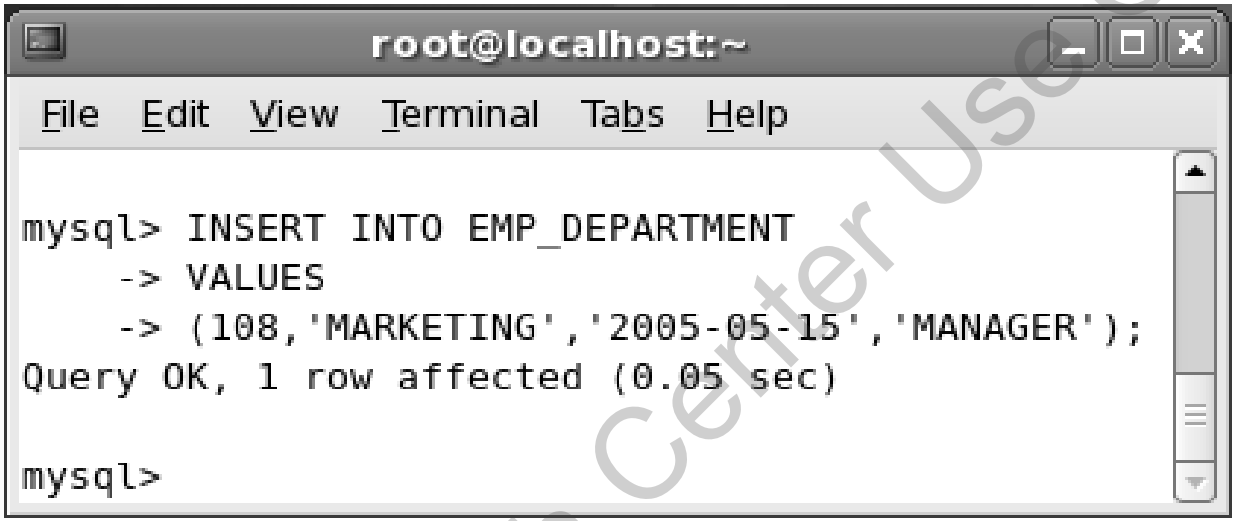
Inserting Data into a Table

- You can also enter the following command at the command prompt:

```
INSERT INTO EMP_DEPARTMENT VALUES (108,
'MARKETING', '2000-05-15 ', 'MANAGER');
```

- This command will work only if values for all the columns are specified following the `VALUES` keyword

Figure displays the output of the command

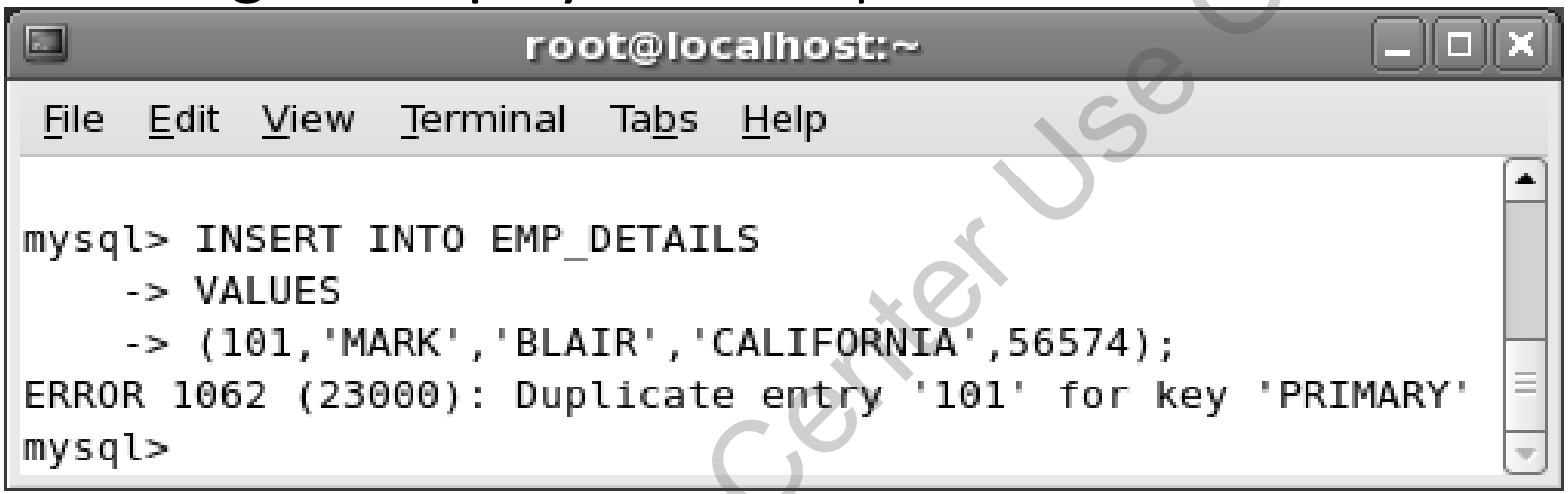


Inserting Data into a Table

- You had earlier defined the primary key of `EMP_DEPARTMENT` table as `NOT NULL` and `AUTO_INCREMENT`
- Now you can try to insert a row having the `E_ID` that already exists, by entering the following command at the command prompt:

```
INSERT INTO EMP_DETAILS VALUES
(101,'MARK','BLAIR','CALIFORNIA', 56574);
```

Figure displays the output of the command



Inserting Data into a Table

- A different form of the `INSERT` command enables you to set the column to a specific value
- The syntax is as follows:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name SET col_name = {expression | DEFAULT},...
[ON DUPLICATE KEY UPDATE col_name=expression,...];
```

where,

`INSERT` – adds data to the table

`tbl_name` – specifies the name of the table

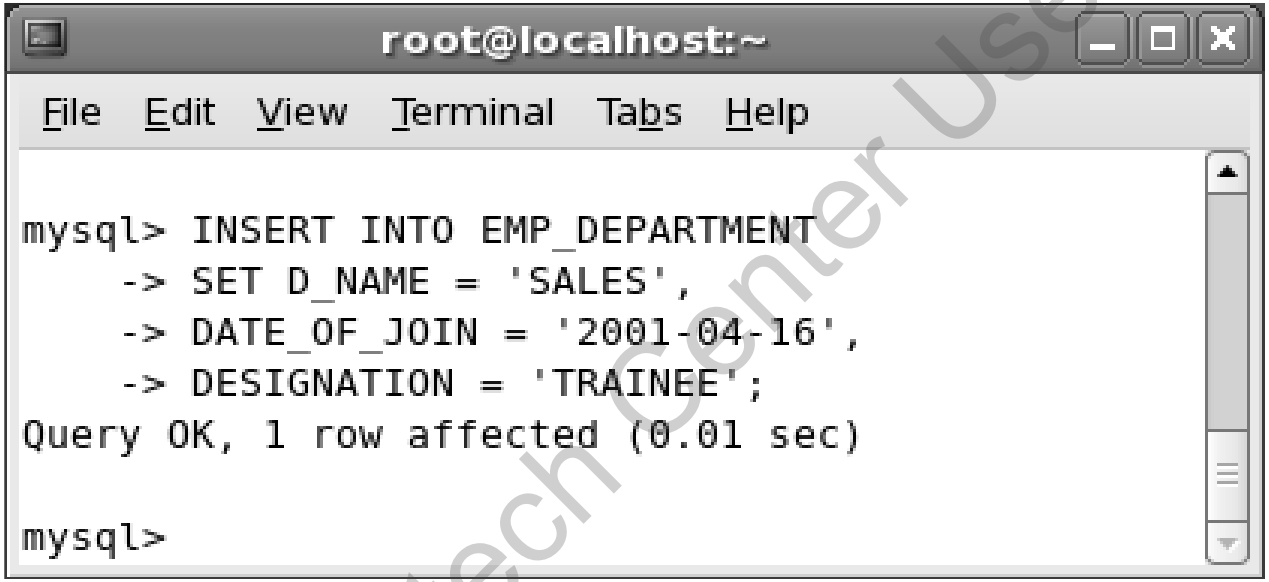
`SET` – specifies the column names explicitly to which the values will be added

Inserting Data into a Table

- To insert rows in EMP_DEPARTMENT, enter the following command at the command prompt:

```
INSERT INTO EMP_DEPARTMENT SET D_NAME = 'SALES',
DATE_OF_JOIN = '2001-04-16', DESIGNATION= 'TRAINEE';
```

Figure displays the output of the command



Inserting Data into a Table

- The INSERT command enables you to add data to a table returned from a SELECT query

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY][IGNORE][INTO]
tbl_name [(col_name,...)] SELECT ...;
```

where,

INSERT – adds data into the table

tbl_name – specifies the name of the table to add data

col_name – specifies the name of the column

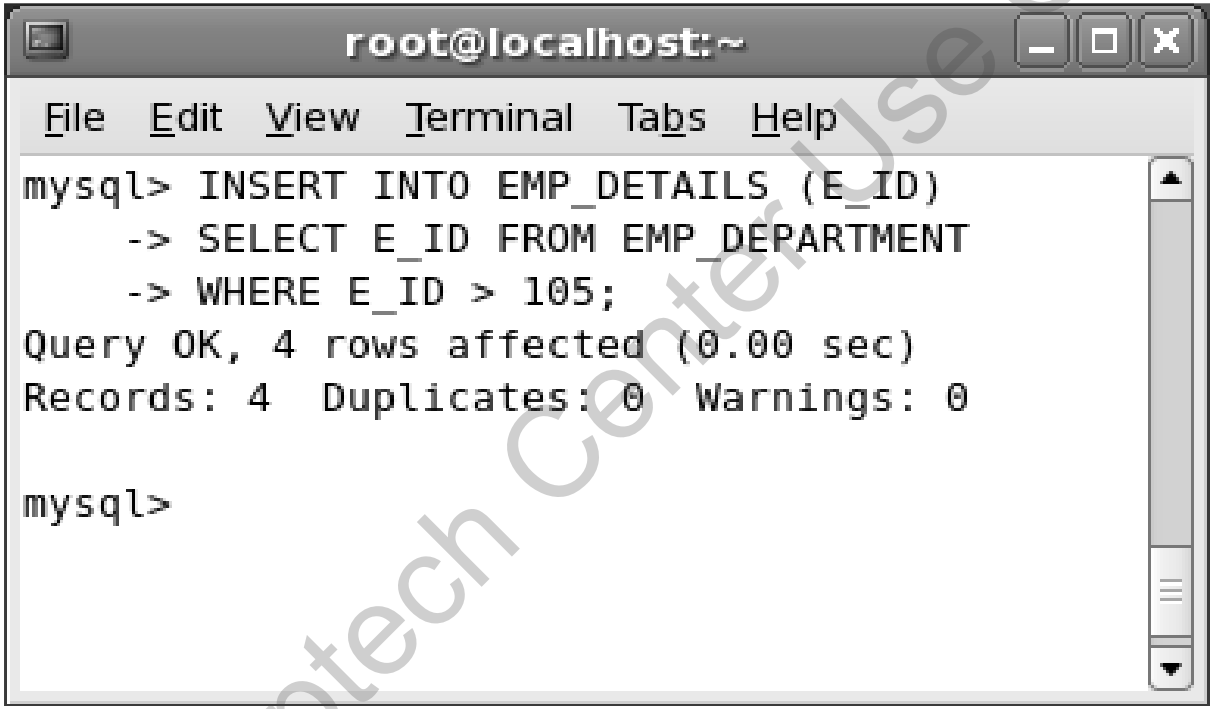
SELECT – defines the data to be added in the table specified in the tbl_name clause

Inserting Data into a Table

- To insert E_ID in the EMP_DETAILS table where the E_ID is retrieved from the EMP_DEPARTMENT table using the SELECT command, enter the following command at the command prompt:

```
INSERT INTO EMP_DETAILS (E_ID)
SELECT E_ID FROM EMP_DEPARTMENT
WHERE E_ID > 105;
```

Figure displays the output of the command



Updating Data in a Table

- MySQL provides the `UPDATE` command to change or modify data within a table
- The syntax to update a table is:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name SET
column=value,... [WHERE clause][LIMIT n];
```

where,

`UPDATE` – modifies the column data

`tbl_name` – specifies the name of the table to modify

`SET` – defines the value to change

`column` – specifies the name of the column

`value` – defines the new value to be entered for the column

The options for the `UPDATE` command are listed in table

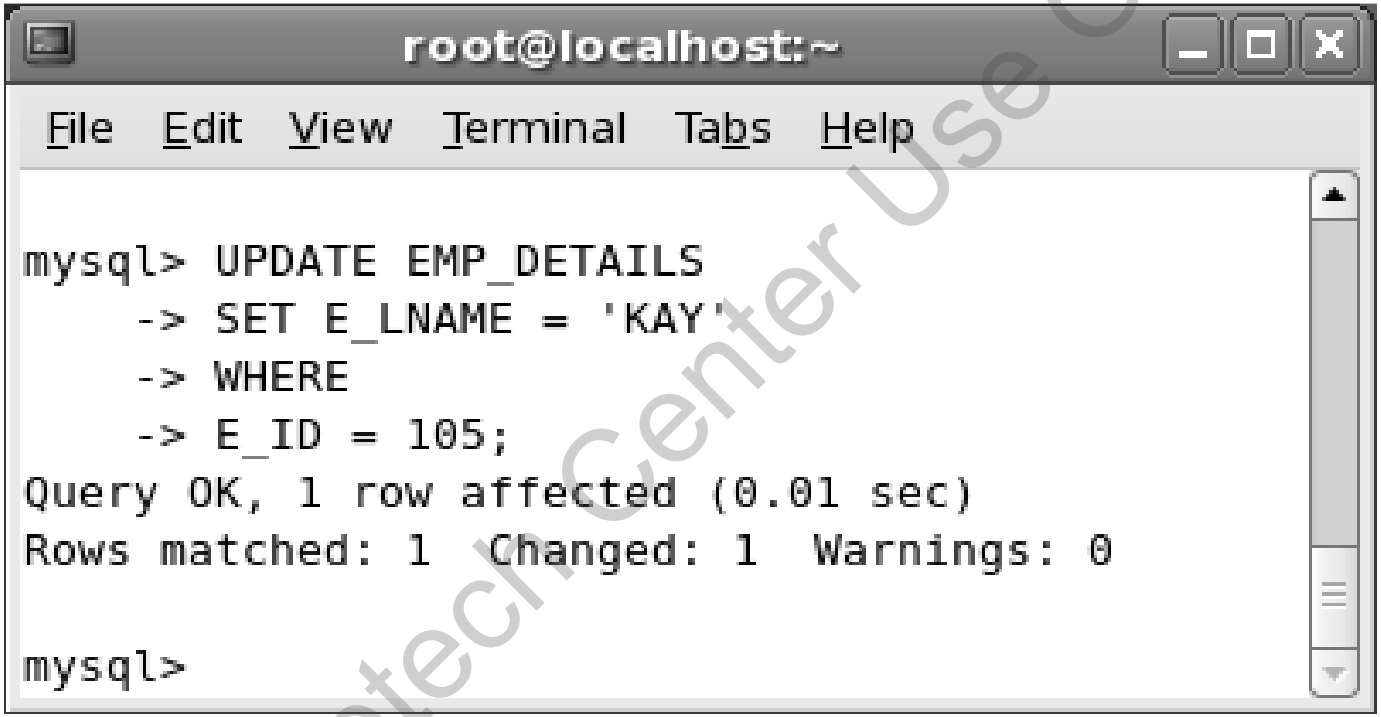
Option	Description
LOW_PRIORITY	Assigns a low priority to the update against all other commands
IGNORE	Updates a row even if there is an error
WHERE	Specifies the rows to be updated based on the specific condition
LIMIT	Specifies a limit on the number of rows that can be updated

Updating Data in a Table

- To update a row in EMP_DETAILS table, enter the following command at the command prompt

```
UPDATE EMP_DETAILS SET E_LNAME = 'KAY' WHERE E_ID =
105;
```

Figure displays the output of the command

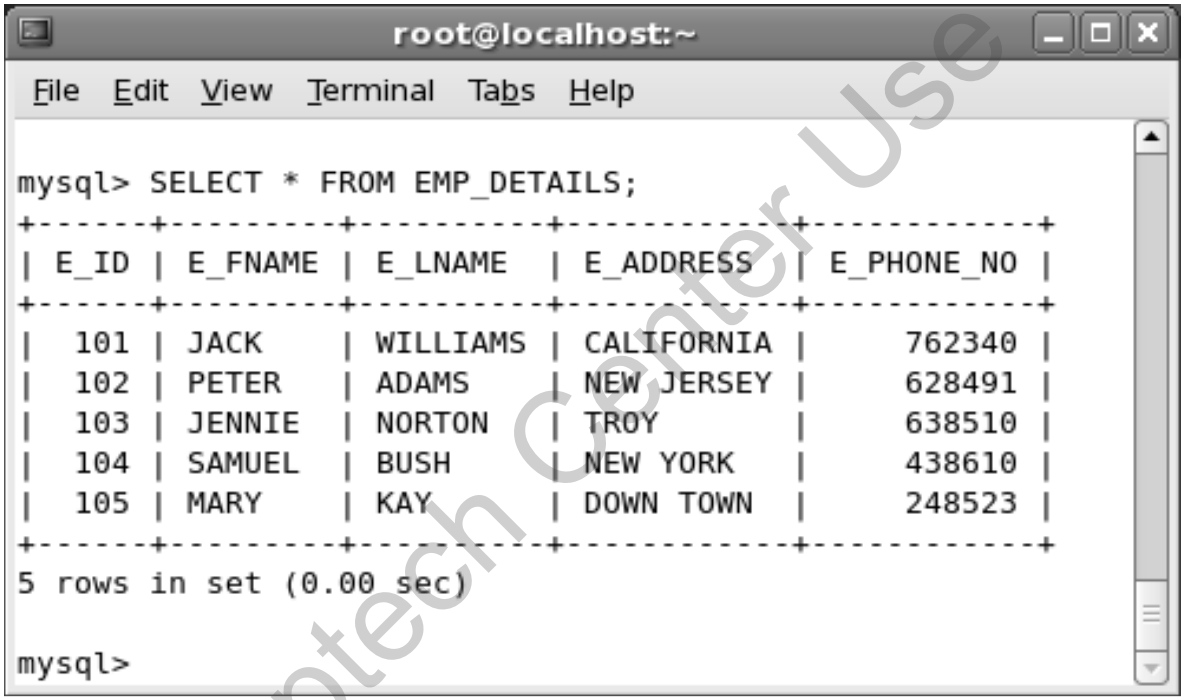


Updating Data in a Table

- When you include the `ORDER BY` clause in an `UPDATE` command, MySQL will update the rows in the order specified in the `ORDER BY` clause
- You can include the `ORDER BY` clause in an `UPDATE` command to modify records in a table sorted on specific conditions
- You can now use the `SELECT` command to view the updated data in the table
- To view the updated results, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS;
```

Figure displays the output of the command



Updating Data in a Table

- To update a row in EMP_DETAILS table, enter the following command at the command prompt

```
UPDATE EMP_DETAILS SET E_LNAME = 'KAY' WHERE E_ID =
105;
```

Replacing Data in a Table

- MySQL provides the `REPLACE` command to substitute data in the table
- The `REPLACE` command is similar to the `INSERT` command, except that it replaces the old value of a column with a new value
- The `REPLACE` command will delete the old row only if the new row satisfies the conditions of the column definitions
- If the value in the `REPLACE` command does not match the existing record then MySQL executes it as an `INSERT` statement

Replacing Data in a Table

- The syntax for replacing data in a table is:

```
REPLACE [DELAYED|LOW_PRIORITY] INTO tbl_name
[(column,…)] VALUES (value,...);
```

where,

`REPLACE` – substitutes the column data

`tbl_name` – specifies the name of the table to modify

`value` – defines the new value for the column

The options of the REPLACE command are listed in table

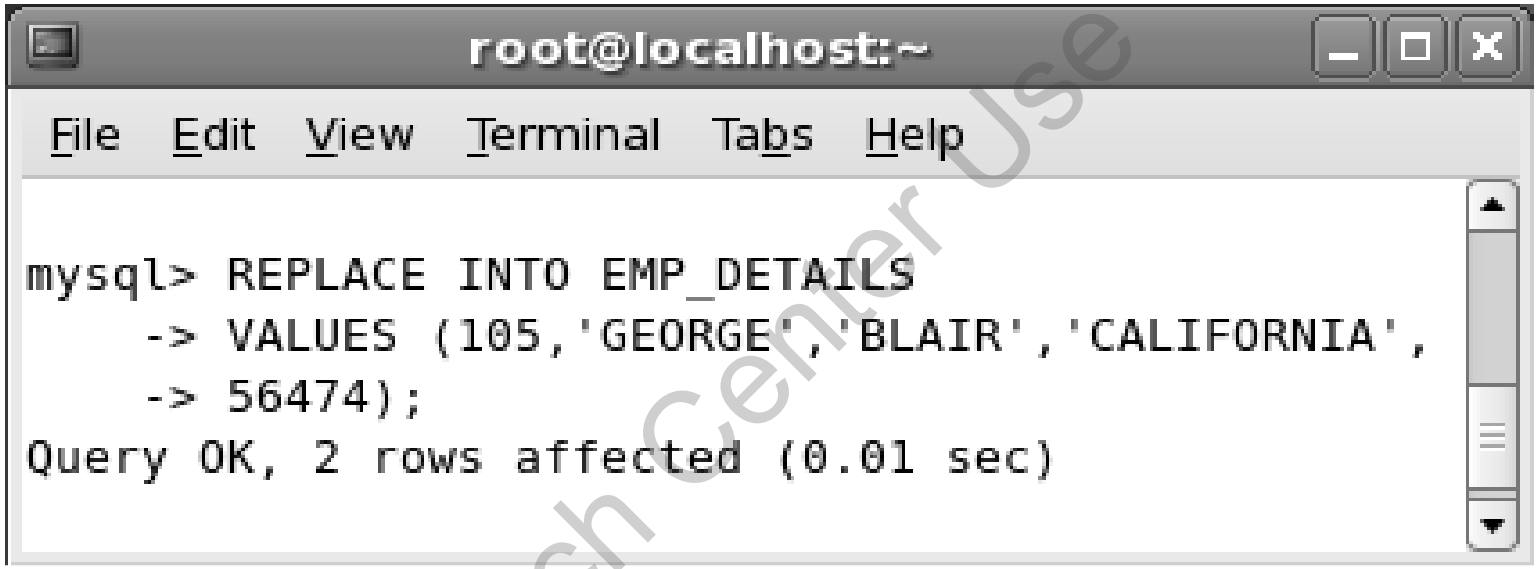
Option	Description
DELAYED	Postpones the execution of the command until no client is accessing the table
LOW_PRIORITY	Specifies the rows to be inserted into a buffer
column	Specifies the column names for which the values are to be replace
value	Defines the new values for the columns

Replacing Data in a Table

- To replace a row in EMP_DETAILS, enter the following command at the command prompt:

```
REPLACE INTO EMP_DETAILS
VALUES(105,'GEORGE','BLAIR','CALIFORNIA',56474
);
```

Figure displays the output of the command

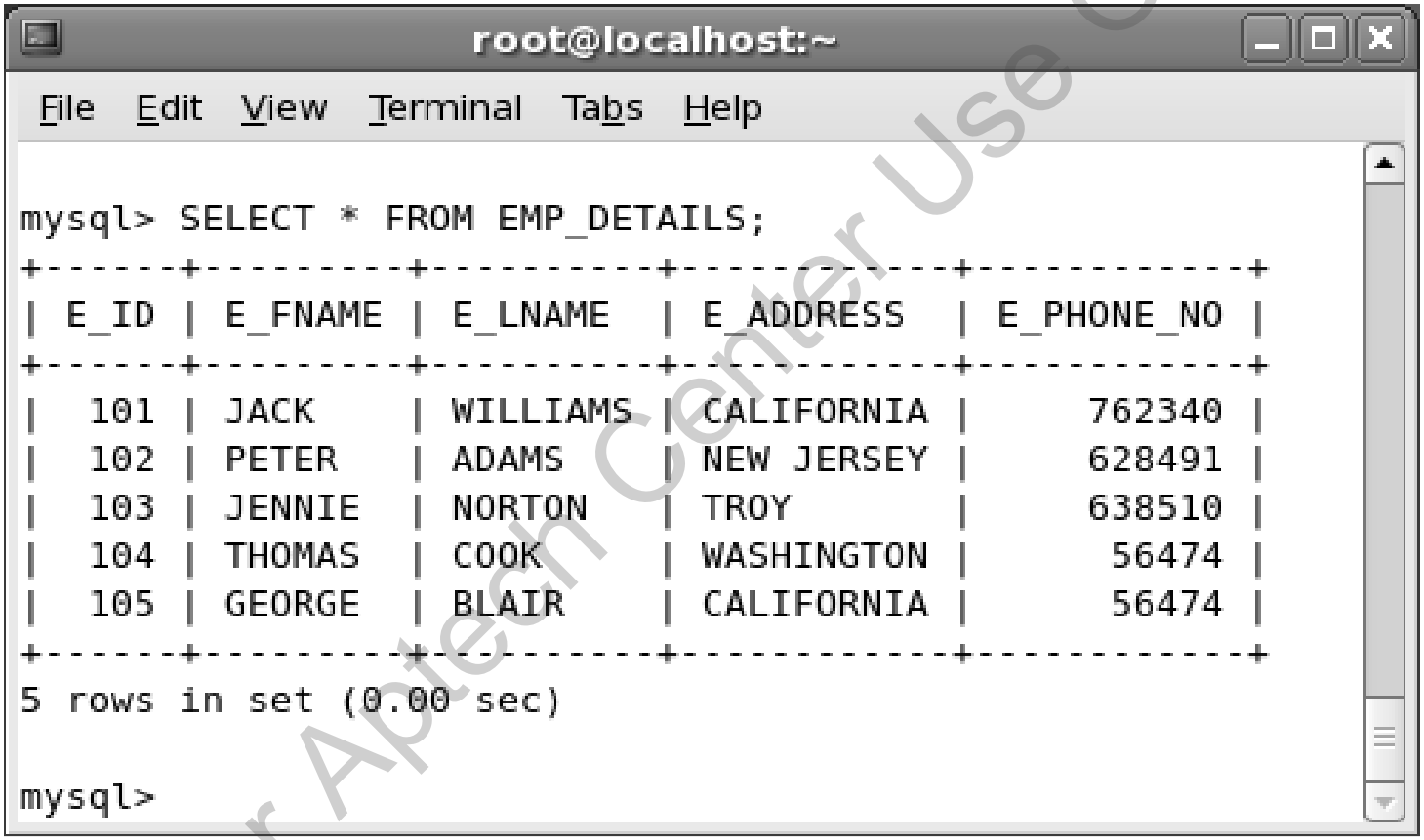


Replacing Data in a Table

- MySQL displays the number of rows affected by the `REPLACE` query
- If MySQL returns the number of rows affected as one for a single row modification, it means that a row has been added and existing rows have not been deleted
- When the number of rows affected is more than one, it means that an existing record has been deleted and a new record has been added to the table
- To view the data in the `EMP_DETAILS` table, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS;
```

Figure displays the output of the command



- You can also use the SELECT query within a REPLACE command. The syntax to include a SELECT query within a REPLACE command is:

```
REPLACE [DELAYED|LOW_PRIORITY] INTO tbl_name
[(column,...)] SELECT select_clause;
```

where,

REPLACE – substitutes the column data

tbl_name – specifies the name of the table to modify

SELECT – executes the select query specified in the clause

Replacing Data in a Table

- The REPLACE command also enables you to specify the column names and the values together
- MySQL assigns a NULL value to the columns if you fail to specify a value in the REPLACE command

```
REPLACE [DELAYED|LOW_PRIORITY] INTO tbl_name SET
column=value, column=value,...;
```

where,

REPLACE – substitutes the column data

tbl_name – specifies the name of the table to modify

SET – defines the new value to be entered in the column

column – specifies the name of the column

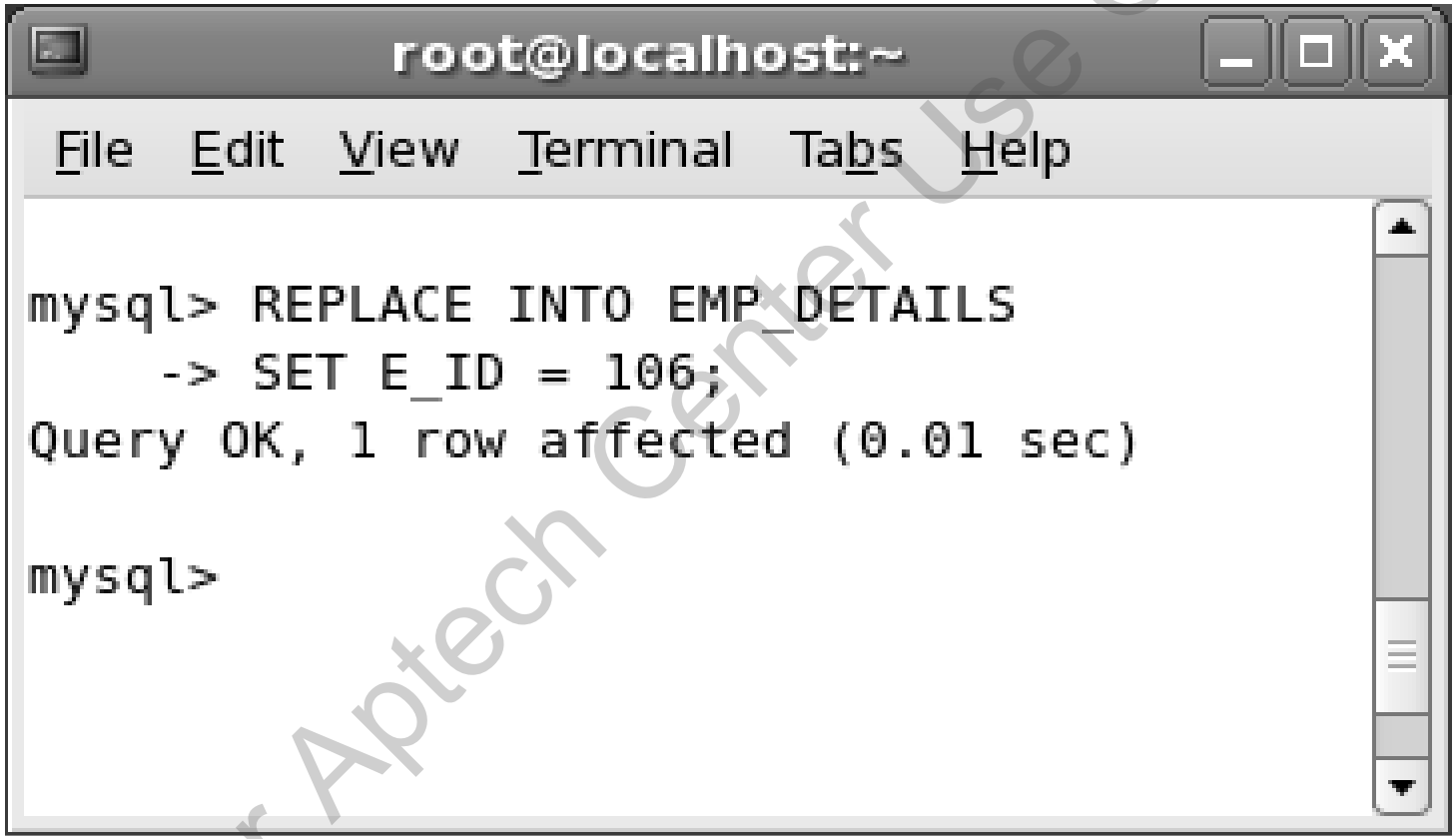
value – defines the new value for the specified column

Replacing Data in a Table

- For example, to add a new record in the EMP_DETAILS table, where you only specify the E_ID, enter the following command at the command prompt:

```
REPLACE INTO EMP_DETAILS
SET E_ID = 106;
```

Figure displays the output of the command

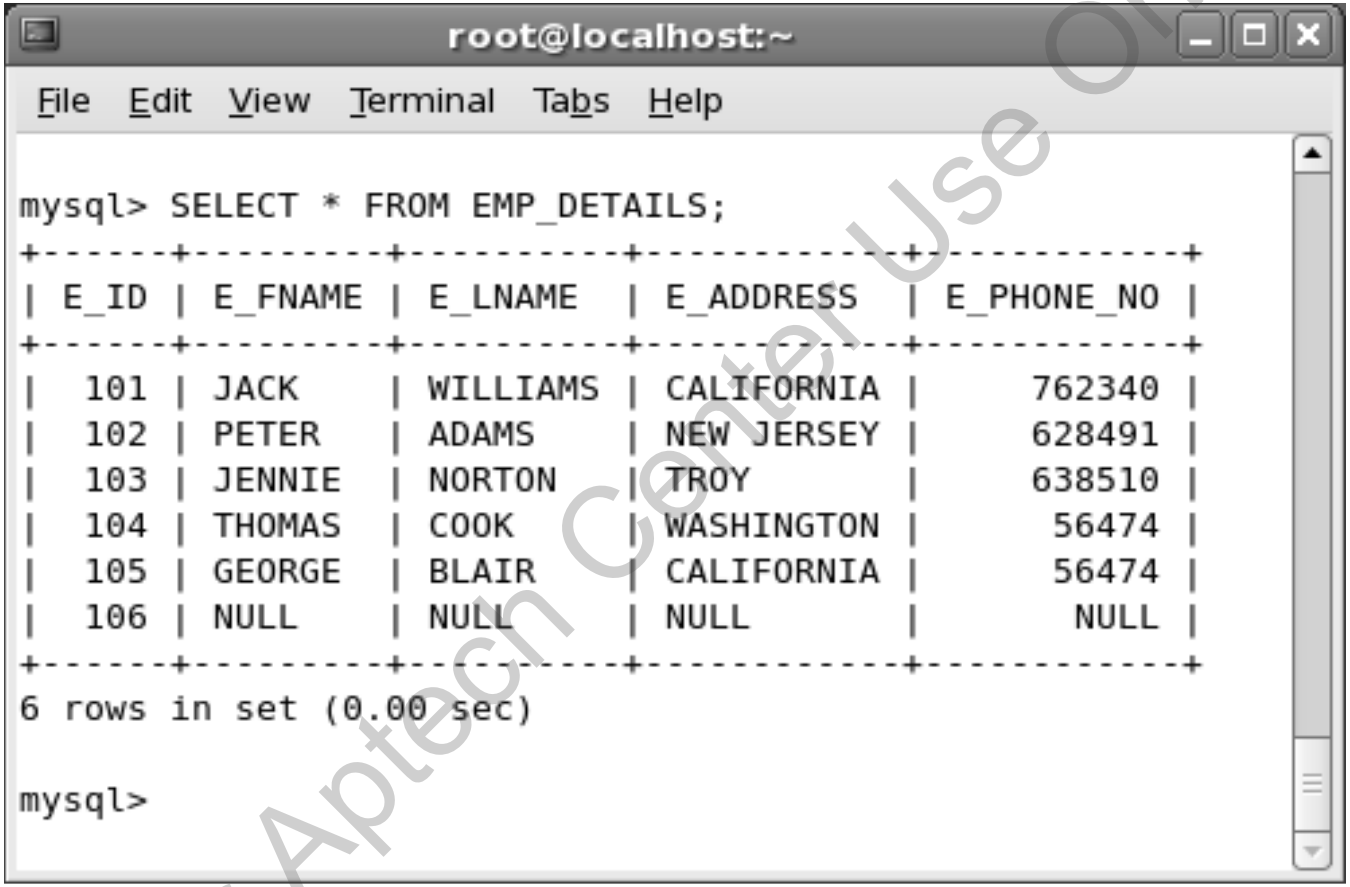


Replacing Data in a Table

- In this command, you have specified a value only for the `E_ID` field
- MySQL will assign `NULL` values to the other columns in the table for this record
- To view the `NULL` values for this record, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS;
```

Figure displays the output of the command



Deleting Data from a Table

- MySQL provides the `DELETE` command to remove rows from a table
- You can choose to remove selected rows of a table or the entire table
- The syntax to delete rows from a table is:

```
DELETE [LOW_PRIORITY|QUICK] [IGNORE} FROM tbl_name [WHERE
clause] [ORDER BY column,...] [LIMIT n];
```

where,

`DELETE` – removes data from the table

`tbl_name` – specifies the name of the table from which rows are to be deleted

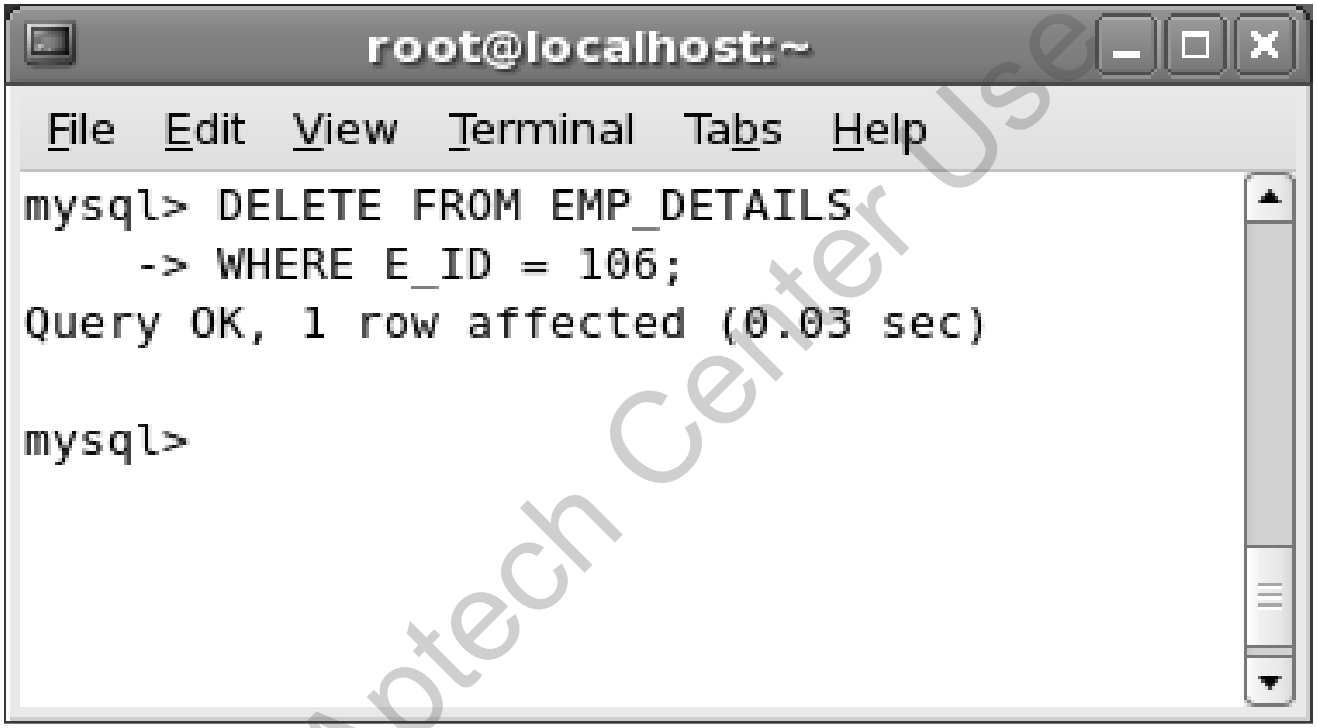
The options for the `DELETE` command are provided in table

Option	Description
LOW_PRIORITY	Postpones deletion until all other operations on the table have been completed by the client such as reading
QUICK	Suspends the merging of indexes during execution
IGNORE	Ignores error while deleting a row
WHERE	Deletes rows that satisfy the specified conditions
ORDER BY	Deletes data in the order specified
LIMIT n	Specifies the maximum number of rows to be deleted in a single attempt

Deleting Data from a Table

- If you use the `DELETE` command without the `WHERE` clause, the entire table will be deleted
- If you specify the `WHERE` clause in the `DELETE` command, the system prompts a message displaying the number of rows that will be deleted
- If no `WHERE` clause is used, all the rows are deleted
- MySQL returns a `'0'` value, when the entire table is deleted because MySQL cannot identify the number of rows to be deleted

Figure displays the output of the command



Using the ORDER BY Clause

- You use the `SELECT` command to retrieve data from a database
- MySQL displays the `SELECT` command result in the order in which the data is inserted in the table
- You can also sort the output on more than one column in ascending or descending order
- You will use the `ORDER BY` clause with `SELECT` command to sort the data
- The use of `DISTINCT` keyword causes only one row of data to be displayed for every group of rows that is identical

Using the ORDER BY Clause

- The syntax for `ORDER BY` is:

```
SELECT column_name FROM tbl_name ORDER BY column
[ASC|DESC] [,column2[ASC|DESC],...];
```

where,

`SELECT` – retrieves data from the table

`column_name` – specifies the name for the column to retrieve data from

`tbl_name` – specifies the name of the table that contains the column and data

`ORDER BY` – sorts the column data in the specified order

`column` – specifies the name of the column

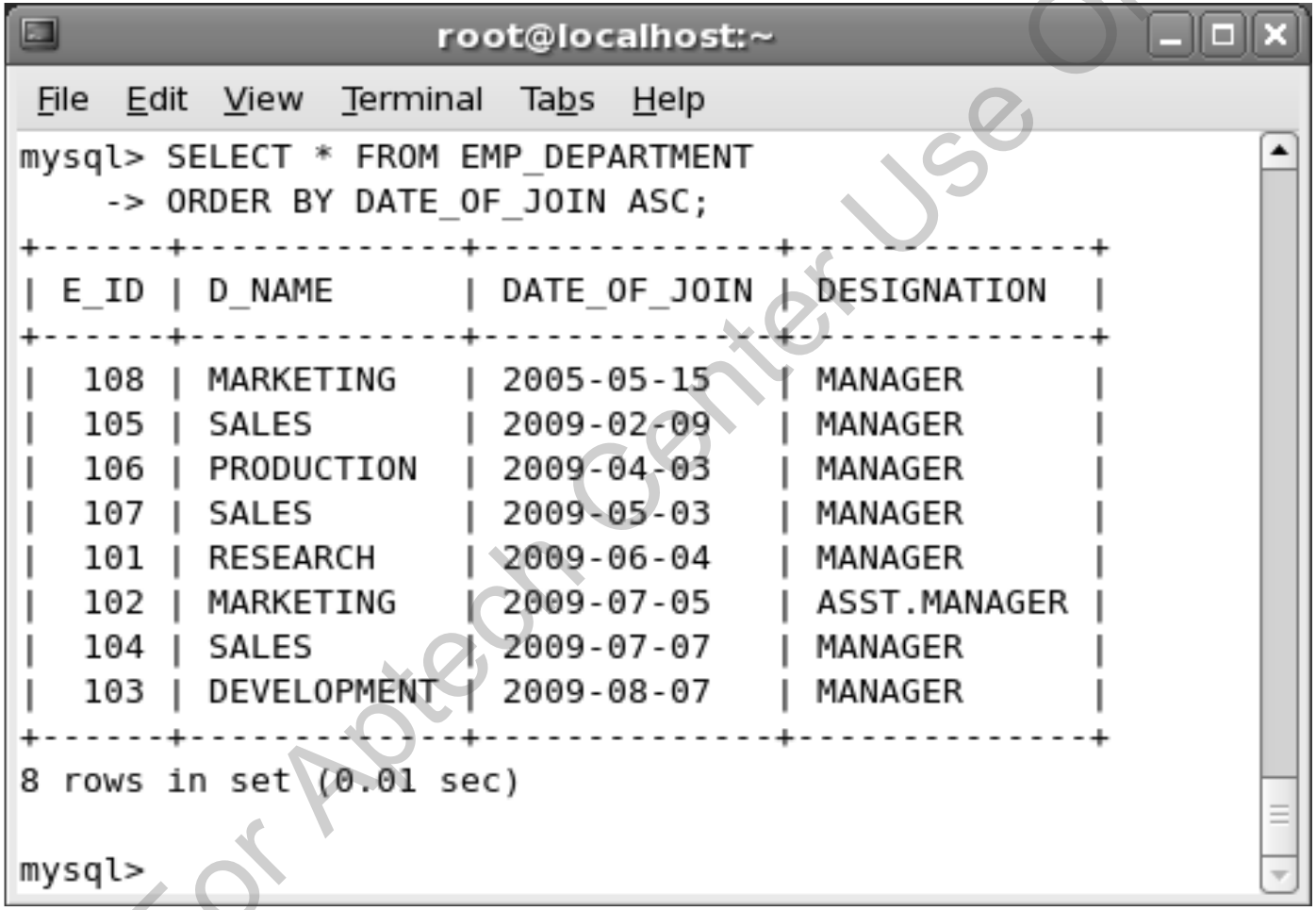
`ASC|DESC` – specifies the sort direction for data. MySQL supports types of sorting

Using the ORDER BY Clause

- To organize the output of EMP_DEPARMENT table on the DATE_OF_JOIN field in an ascending order, enter the following command at the command prompt:

```
SELECT * FROM EMP_DEPARTMENT ORDER BY DATE_OF_JOIN
ASC;
```

Figure displays the output of the command

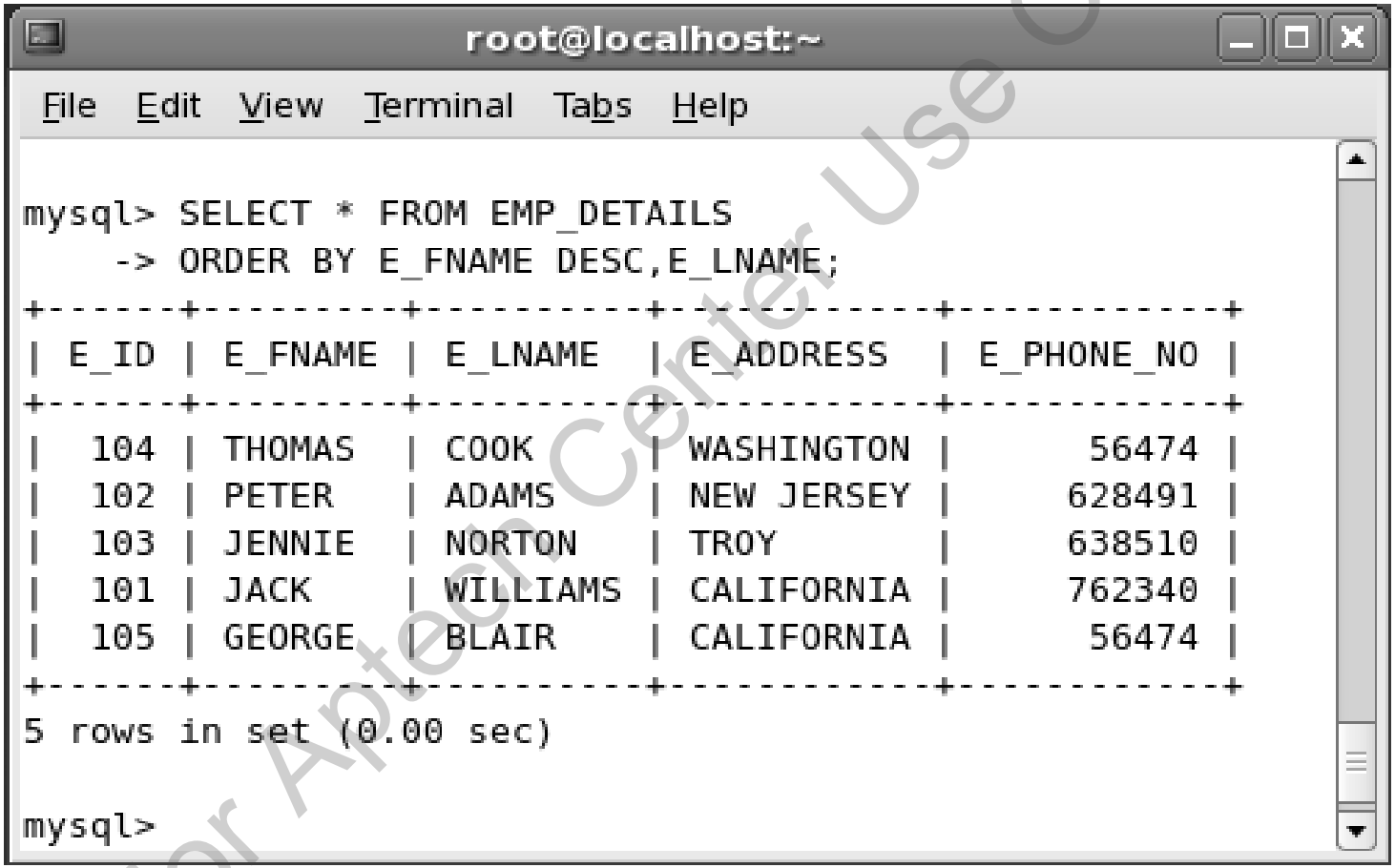


Using the ORDER BY Clause

- You will now sort the data in the EMP_DETAILS table on the E_FNAME and E_LNAME columns
- You will sort data from the E_FNAME column in the descending order
- To sort the output of EMP_DETAILS table on the E_FNAME field in the descending order and then by E_LNAME field, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS ORDER BY E_FNAME DESC,
E_LNAME;
```

Figure displays the output of the command



Using the GROUP BY Clause

- MySQL provides the `GROUP BY` command to group rows with similar values for a specific column into a single row
- This will enable you to execute tasks on these grouped rows
- The syntax for grouping data is:

```
SELECT column_name FROM tbl_name GROUP BY column
[,column2...];
```

where,

`SELECT` – retrieves data from the table

`column_name` – specifies the column name to retrieve data from

`tbl_name` – specifies the name of the table that contains the column and data

`GROUP BY` – categorizes the data

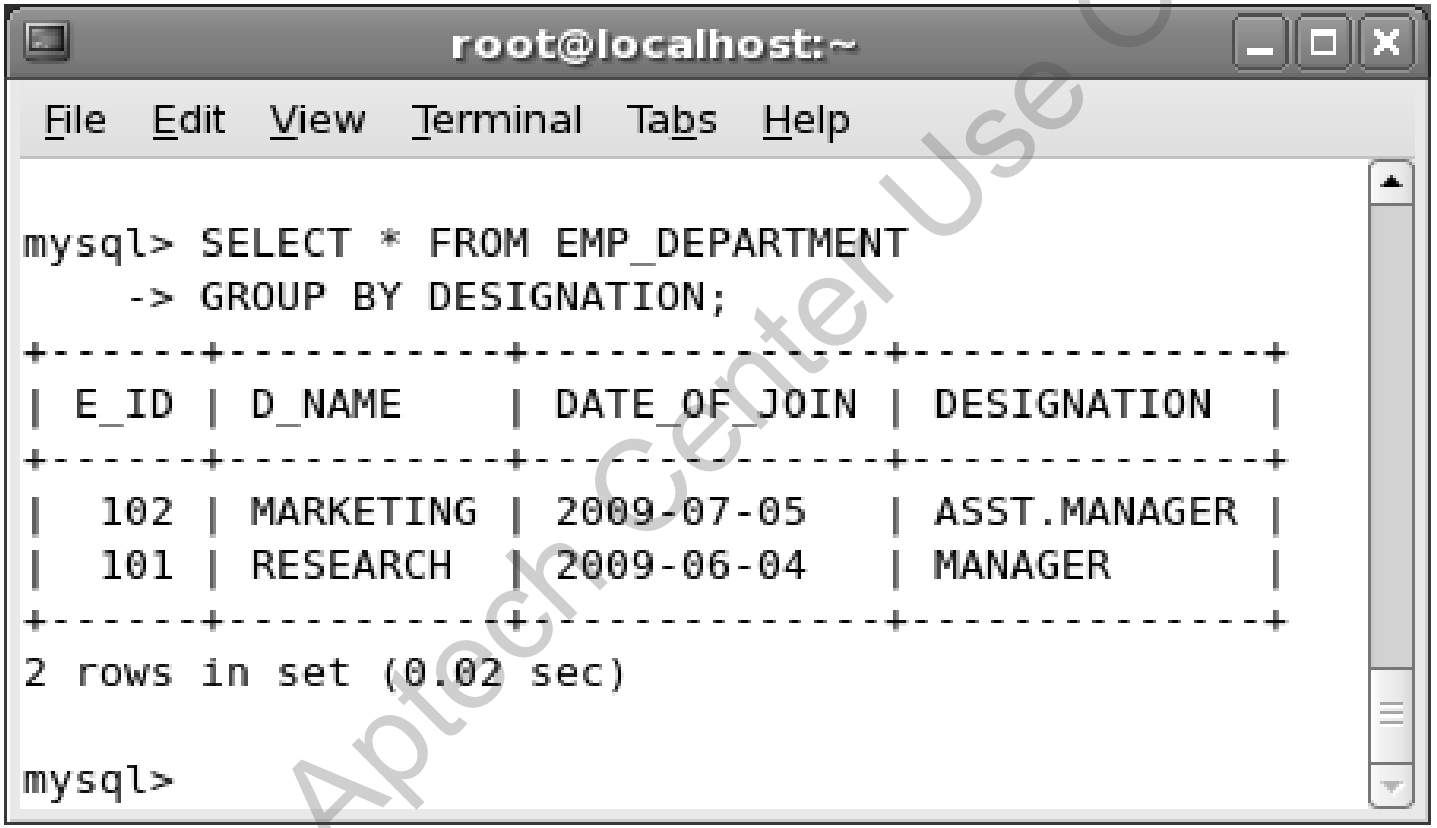
`column` – specifies the name of the column in the table

Using the GROUP BY Clause

- You can group the employees based on their designations from the EMP_DEPARTMENT table on the DESIGNATION column
- To group the output of EMP_DEPARTMENT table on the DESIGNATION field, enter the following command at the command prompt:

```
SELECT * FROM EMP_DEPARTMENT GROUP BY
DESIGNATION;
```

Figure displays the output of the command



Summary

- Keys help to identify a record in the table
- A primary key uniquely identifies the records in a table
- A foreign key in a table must have a corresponding primary key in a different table
- A primary key and a foreign key can be defined while creating a table
- The ALTER TABLE command enables to define keys in a table
- The INSERT command allows you to add a row in a table
- The UPDATE command allows you to modify the values present in the columns of a table

Summary

- MySQL enables you to substitute values in a column using the REPLACE command
- MySQL provides the DELETE command to remove unwanted or obsolete rows in a table
- The ORBER BY clause can be used with the SELECT command to sort the data
- Grouping categorizes rows with similar values for a specific column into a single row to execute combined data processing operations