# Experiment number: - 1

## **OBJECT: -** The SQL CREATE TABLE Statement.

**THEORY: -** The CREATE TABLE statement is used to create a new table in a database. The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

**Syntax:-**

CREATE TABLE table\_name (

column1 datatype,

column2 datatype,

column3 datatype

);

**SQL CREATE TABLE Example: -**

The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

Example

CREATE TABLE Persons (

PersonID int,

LastName varchar(255),

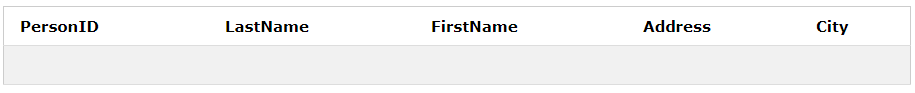
FirstName varchar(255),

Address varchar(255),

City varchar(255)

);

Result –



### Table : New Empty Table Created

# Experiment number: - 2

**OBJECT: -** The SQL INSERT INTO Statement

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

**Syntax:-**

The first way specifies both the column names and the values to be inserted:

INSERT INTO table\_name (column1, column2, column3, ...)

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

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

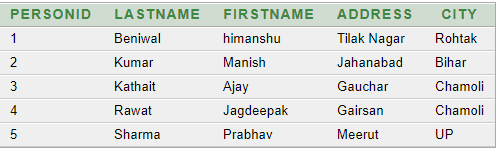
INSERT INTO table\_name VALUES (value1, value2, value3, ...);

**SQL CREATE TABLE Example: -**

Example

insert into Persons values (1,'Beniwal','himanshu','Tilak Nagar','Rohtak');

**Result –**



### Table 2: Already Created Table now contains the inserted values.

# Experiment number: - 3

**OBJECT: -** The SQL DELETE Statement

**THEORY: -** The DELETE statement is used to delete existing records in a table.**Syntax:-**

Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

**DELETE Syntax: -**

DELETE FROM table\_name

WHERE condition;

**Delete All Records**

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM table\_name;

Or:

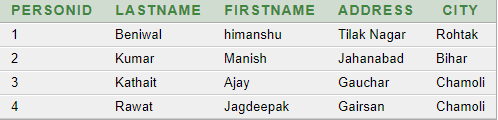
DELETE \* FROM table\_name;

**SQL DELETE Example**

The following SQL statement deletes the city "UP" from the "Persons" table:

delete from Persons where City='UP';

**Result –**



### Table 3: Values deleted (row 5) from the table.

# Experiment number: - 4

**OBJECT: -** SQL ALTER TABLE Statement

**THEORY: -** The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

**ALTER TABLE –**

ADD Column – To add a column in a table, use the following syntax:

ALTER TABLE table\_name ADD column\_name datatype;

DROP COLUMN – To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

ALTER TABLE table\_name DROP COLUMN column\_name;

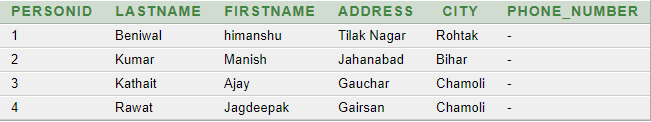
ALTER/MODIFY COLUMN – To change the data type of a column in a table, use the following syntax:

ALTER TABLE table\_name MODIFY column\_name datatype;

**Example –** We use the following SQL statement:

ALTER TABLE Persons ADD Phone\_Number number;

**Result –**



### Table 4: Column named "Phone\_Number" in the "Persons" table.

# Experiment number: - 5

**OBJECT: -** The SQL WHERE Clause

**THEORY: -** The WHERE clause is used to filter records. The WHERE clause is used to extract only those records that fulfill a specified condition.

The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement, etc.!

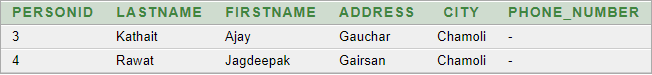
**WHERE Syntax –**

SELECT column1, column2, ... FROM table\_name WHERE condition;

**WHERE Clause Example –** The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

SELECT \* FROM Persons WHERE City='Chamoli';

**Result –**



### Table 5: Following SQL statement selects all the Persons from the City "Chamoli", in the "Persons" table.

# Experiment number: - 6

**OBJECT: -** SQL AND, OR and NOT Operators

**THEORY: -** The WHERE clause can be combined with AND, OR, and NOT operators.

* The AND and OR operators are used to filter records based on more than one condition:
* The AND operator displays a record if all the conditions separated by AND is TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.
* The NOT operator displays a record if the condition(s) is NOT TRUE.

**AND Syntax –**

SELECT column1, column2, ... FROM table\_name WHERE condition1 AND condition2;

**OR Syntax**

SELECT column1, column2, ... FROM table\_name WHERE condition1 OR condition2;

**NOT Syntax**

SELECT column1, column2, ... FROM table\_name WHERE NOT condition;

**AND Example** the following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

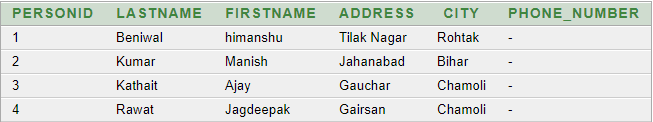
SELECT \* FROM Persons WHERE City=’UP’ AND City=’Rohtak’';

**OR Example** The following SQL statement selects all fields from "Customers" where city is "Berlin" OR "München":

SELECT \* FROM Persons WHERE City=’Rohtak’'OR City=’UP’;

**NOT Example** the following SQL statement selects all fields from "Customers" where country is NOT "Germany":

SELECT \* FROM Persons WHERE NOT City=’UP’;

**Result **

### Table 6: the following SQL statement Shows Table with not City with value “UP”.

# Experiment number: - 7

**OBJECT: -** The SQL UPDATE Statement

**THEORY: -** The UPDATE statement is used to modify the existing records in a table.

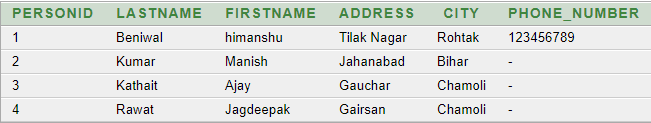
Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

**UPDATE Syntax**

UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE condition;

**UPDATE Table**

UPDATE Persons SET Phone\_Number = '123456789' WHERE City='Rohtak';

**Result** – 

### Table 7: the following SQL statement updates the first Person (City=’Rohtak’) with a new Phone number.

# Experiment number: - 8

**OBJECT: -** The SQL LIKE, NOT LIKE Operator

**THEORY: -** The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.There are two wildcards used in conjunction with the LIKE operator:

* % - The percent sign represents zero, one, or multiple characters
* \_ - The underscore represents a single character

Note: MS Access uses a question mark (?) instead of the underscore (\_). The percent sign and the underscore can also be used in combinations!

**LIKE Syntax**

SELECT column1, column2, ... FROM table\_name WHERE columnn LIKE pattern;

**NOT LIKE Syntax**

SELECT column1, column2, ... FROM table\_name WHERE column NOT LIKE pattern;

**Example –**

SELECT \* FROM Persons WHERE FirstName LIKE '%a%';

**Result –**

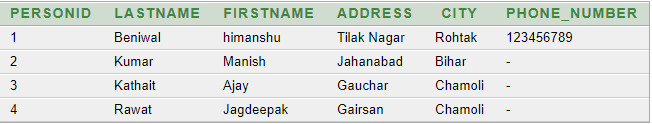


Table 8: the following SQL statement selects all Persons with a FirstName starting with "a".

# Experiment number: - 9

**OBJECT: -** SQL JOINs

**THEORY: -** A JOIN clause is used to combine rows from two or more tables, based on a related column between them. Here are the different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Return all records when there is a match in either left or right table



**LEFT JOIN Syntax**

SELECT column\_name(s) FROM table1 LEFT JOIN table2 ON table1.column\_name = table2.column\_name;

**RIGHT JOIN Syntax**

SELECT column\_name(s) FROM table1 RIGHT JOIN table2 ON table1.column\_name = table2.column\_name;

**FULL OUTER JOIN Syntax**

SELECT column\_name(s)FROM table1FULL OUTER JOIN table2 ON table1.column\_name = table2.column\_name;

**INNER JOIN Syntax**

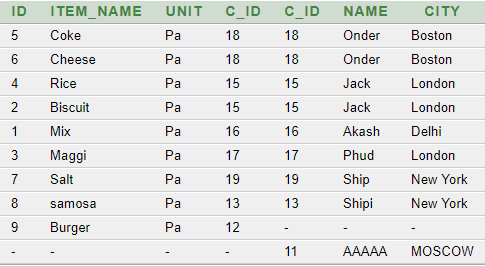
SELECT column\_name(s) FROM table1 INNER JOIN table2 ON table1.column\_name = table2.column\_name;

**SQL INNER JOIN Example**

The following SQL statement selects all orders with Company information:

select \* from Food FULL OUTER JOIN Company ON Food.c\_id = Company.c\_id;

**Result –**



### Table 9: there are records in the "Food" table that do not have matches in "Company", these orders will not be shown!

# Experiment number: - 10

**OBJECT: -** The SQL MIN(), MAX(), COUNT(), AVG() and SUM() Functions

**THEORY: -**

* The MIN() function returns the smallest value of the selected column.
* The MAX() function returns the largest value of the selected column.
* The COUNT() function returns the number of rows that matches a specified criteria.
* The AVG() function returns the average value of a numeric column.
* The SUM() function returns the total sum of a numeric column.

**MIN() Syntax**

SELECT MIN(column\_name) FROM table\_name WHERE condition;

**MAX() Syntax**

SELECT MAX(column\_name) FROM table\_name WHERE condition;

**COUNT() Syntax**

SELECT COUNT(column\_name) FROM table\_name WHERE condition;

**AVG() Syntax**

SELECT AVG(column\_name) FROM table\_name WHERE condition;

**SUM() Syntax**

SELECT SUM(column\_name) FROM table\_name WHERE condition;

**Result –**

C:\Users\Himanshu Beniwal\Pictures\Screenshots\Screenshot (277).png C:\Users\Himanshu Beniwal\Pictures\Screenshots\Screenshot (278).png C:\Users\Himanshu Beniwal\Pictures\Screenshots\Screenshot (279).png C:\Users\Himanshu Beniwal\Pictures\Screenshots\Screenshot (280).png

C:\Users\Himanshu Beniwal\Pictures\Screenshots\Screenshot (281).png

### Table 10: the following SQL statement selects MIN, MAX, COUNT, AVG, SUM.

**Experiment number: - 11**

**OBJECT: -** DROP statement.

**THEORY: -** DROP statement is used to delete a table or a column from Database.

**DROP Statement-**

DROP TABLE table\_name;

**Drop Table example –**

DROP Table Food;

**Result –**

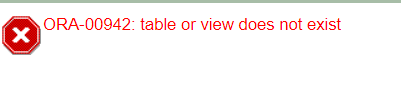


Table 11: the following SQL statement Delete the table from Database.

# Experiment number: - 12

**OBJECT: -** Creating and viewing the VIEW from a Table.

**THEORY: -** View is used as creating a copy of a table which is accessible and further operations can be performed on this view without affecting the real table.

**VIEW Statement-**

create view view\_name as select \* from Table\_Name;

**VIEW example –**

create view per as select \* from Persons;

**Result –**

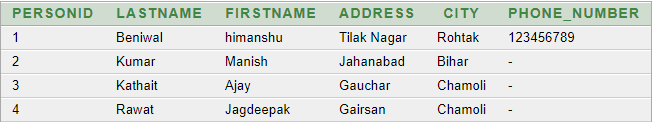


Table 12: the following SQL statement create a view from table “Persons” as ViewName of “per”.

**Experiment number: - 13**

**OBJECT: -** Defining the Cartesian product.

**THEORY: -** The CARTESIAN JOIN or CROSS JOIN returns the Cartesian product of the sets of records from two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to either True or where the join-condition is absent from the statement

**SYNTAX-**

select table1.column1, table2.column2... from table1, table2, [ table3 ]

**EXAMPLE –**

Select \* from emp,employee;

**Result –**

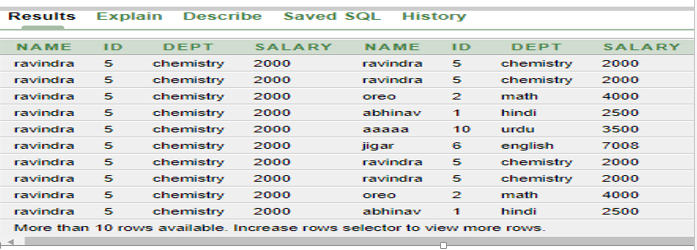


Table 12: the following SQL statement create a Cartesian Product of two tables.

**Experiment number: - 14**

**OBJECT: -** Defining Order by and Group by.

**THEORY: -**

* GROUP BY clause can be used in a select statement to collect data across multiple records and group the results by one or more columns.
* ORDER BY clause is used to sort the records in the result set for a SELECT statement.

**SYNTAX-**

* select expressions from table where condition ORDER BY expression [ ASC | DESC ];
* select expresion1, expresion2 from table name1 group by Expression1,expression;

**EXAMPLE –**

select id, sum(salary) from employee group by id,sum;

**Result –**

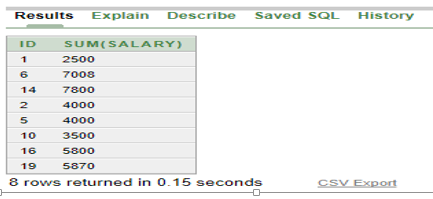
****

Table 12: the following SQL statement create a Group By, Order By.

**Experiment number: - 15**

**OBJECT: -** Defining the terms Minus, Intersect and union.

**THEORY: -**

* Each SELECT statement within the UNION must have the same number of fields in the result sets with similar data types.
* Each SQL statement within the SQL INTERSECT must have the same number of fields in the result sets with similar data types.
* Each SELECT statement within the MINUS query must have the same number of fields in the result sets with similar data types.

**SYNTAX-**

* select\* from expression1, expression2 from table name1 minus expression1, expression2 from table name 2
* select\* from expression1, expression2 from table name1 union expression1, expression2 from table name2;
* select\* from expression1, exprreession2 from table name1 intersect expression1, expression2 from table name2;

**EXAMPLE –**

select\* from emp1 intersect select\* from employee3;

**Result –**

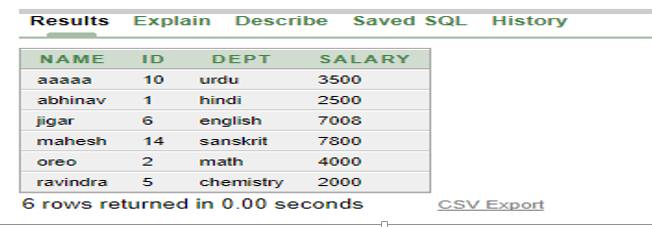


Table 12: the following SQL statement create a Intersect between Two Tables.