Ex.No.13 .05.2025	PERSONAL CONTACT MANAGER SYSTEM

AIM:

To create a Personal Contact Manager System and perform CRUD operation using Nodejs and Mysql workbench.

Personal Contact Manager System :

The **Personal Contact Manager System** is a **web-based application** designed to **track, manage, and analyse contacts**.

- **Add** new alumni records with details such as **Name**, **Email**, and **Mobile Number**
- **Update** existing alumni information
- **Delete** outdated or incorrect records

CREATING 'contact' TABLE:

```
-- Create database

CREATE DATABASE IF NOT EXISTS contact_manager;

-- Use database

USE contact_manager;

-- Create the contacts table

CREATE TABLE IF NOT EXISTS contacts (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) NOT NULL,
   phone VARCHAR(20) NOT NULL,
   address TEXT,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
  );
```

FRONT-END CODING:

index.html

(Home page)

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta name="description" content="Personal Contact Manager System" />

    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>Personal Contact Manager</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>
```

Backend (Node.js) - server.js:

```
const express = require('express');
const cors = require('cors');
const mysql = require('mysql2');
require('dotenv').config();
```

```
const { body, validationResult } = require('express-validator');

const app = express();

// Middleware
app.use(cors());
app.use(express.json());

// Database connection
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '1234567890',
  database: 'contact_manager',
});

// Connect to database
db.connect((err) => {
  if (err) {
    console.error('Error connecting to database:', err);
    return;
  }
  console.log('Connected to MySQL database');
});

// GET all contacts
app.get('/api/contacts', (req, res) => {
  db.query('SELECT * FROM contacts', (err, results) => {
    if (err) {
      res.status(500).json({ error: err.message });
      return;
    }
    res.json(results);
  });
});

// POST create a new contact
```

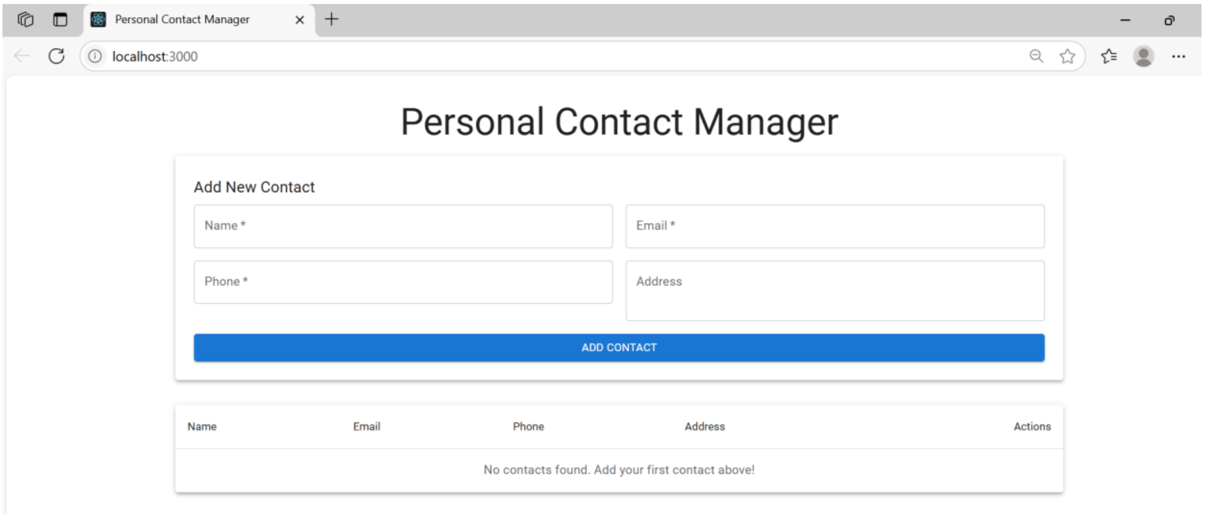
```
app.post(
  '/api/contacts',
  [
    body('name').isString().notEmpty().withMessage('Name is required'),
    body('email').isEmail().withMessage('Email is not valid'),
    body('phone').isMobilePhone().withMessage('Phone number is not valid'),
    body('address').optional().isString(),
  ],
  (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    const { name, email, phone, address } = req.body;
    const query = 'INSERT INTO contacts (name, email, phone, address) VALUES
(?, ?, ?, ?)';

    db.query(query, [name, email, phone, address], (err, result) => {
      if (err) {
        res.status(500).json({ error: err.message });
        return;
      }
      res.status(201).json({ id: result.insertId, name, email, phone, address });
    });
  }
);

// PUT update an existing contact
app.put(
  '/api/contacts/:id',
  [
    body('name').optional().isString(),
    body('email').optional().isEmail(),
    body('phone').optional().isMobilePhone(),
    body('address').optional().isString(),
  ],
  (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    const { id } = req.params;
    const { name, email, phone, address } = req.body;
    const query = 'UPDATE contacts SET name = ?, email = ?, phone = ?, address
= ? WHERE id = ?';

    db.query(query, [name, email, phone, address, id], (err, result) => {
      if (err) {
        res.status(500).json({ error: err.message });
        return;
      }
      if (result.affectedRows === 0) {
        return res.status(404).json({ message: 'Contact not found' });
      }
      res.json({ id, name, email, phone, address });
    });
  }
);

// DELETE a contact
app.delete('/api/contacts/:id', (req, res) => {
  const { id } = req.params;
  db.query('DELETE FROM contacts WHERE id = ?', [id], (err, result) => {
    if (err) {
      res.status(500).json({ error: err.message });
      return;
    }
    if (result.affectedRows === 0) {
      return res.status(404).json({ message: 'Contact not found' });
    }
    res.json({ message: 'Contact deleted successfully' });
  });
});

// Start the server
const PORT = process.env.PORT || 5001;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Package.json:

```
{
  "name": "contact-manager-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.14.3",
    "@mui/material": "^5.14.3",
    "axios": "^1.4.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

OUTPUTS:

INITIAL



localhost:3000

Personal Contact Manager

Add New Contact

Name *

Email *

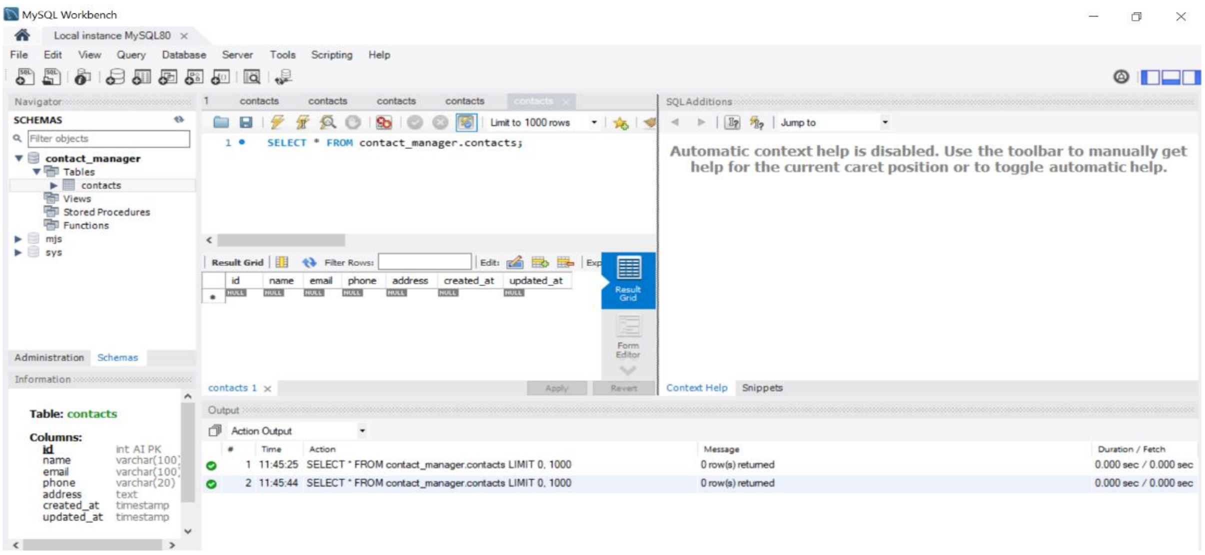
Phone *

Address

ADD CONTACT

Name	Email	Phone	Address	Actions

No contacts found. Add your first contact above!



```
SELECT * FROM contact_manager.contacts;
```

id	name	email	phone	address	created_at	updated_at
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: contacts

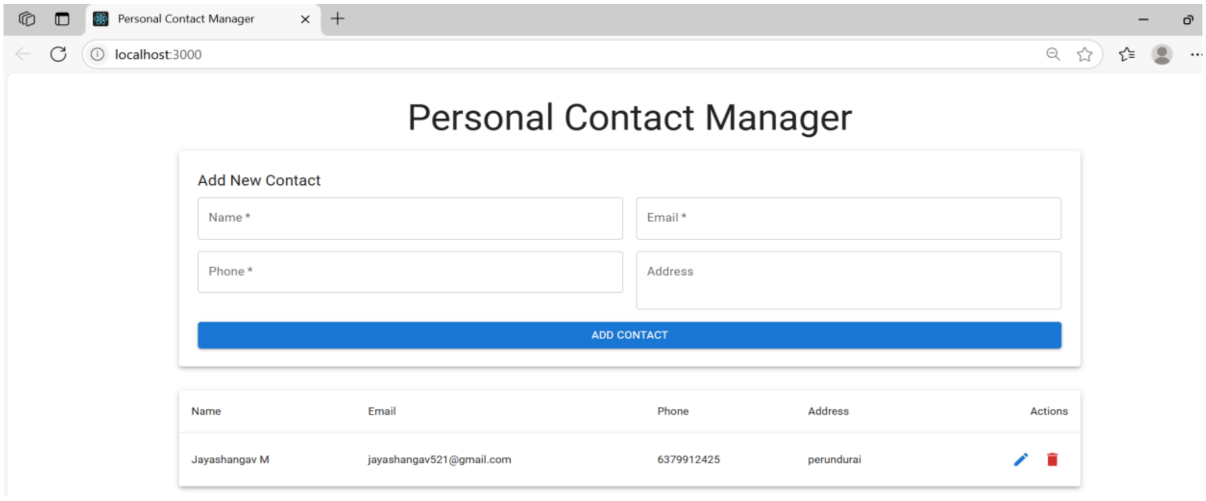
Columns:

id	int AI PK
name	varchar(100)
email	varchar(100)
phone	varchar(20)
address	text
created_at	timestamp
updated_at	timestamp

Action Output

#	Time	Action	Message	Duration / Fetch
1	11:45:25	SELECT * FROM contact_manager.contacts LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
2	11:45:44	SELECT * FROM contact_manager.contacts LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

INSERT:



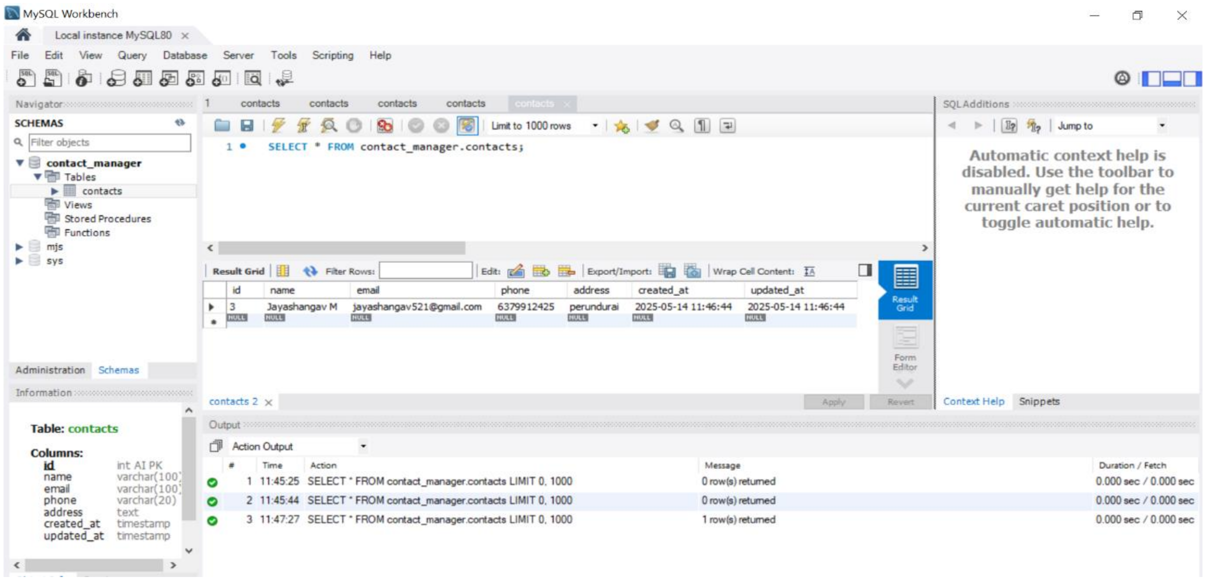
Personal Contact Manager

Add New Contact

Name * | Email * | Phone * | Address

ADD CONTACT

Name	Email	Phone	Address	Actions
Jayashangav M	jayashangav521@gmail.com	6379912425	perundurai	



```
SELECT * FROM contact_manager.contacts;
```

id	name	email	phone	address	created_at	updated_at
3	Jayashangav M	jayashangav521@gmail.com	6379912425	perundurai	2025-05-14 11:46:44	2025-05-14 11:46:44
NULL	NULL	NULL	NULL	NULL	NULL	NULL

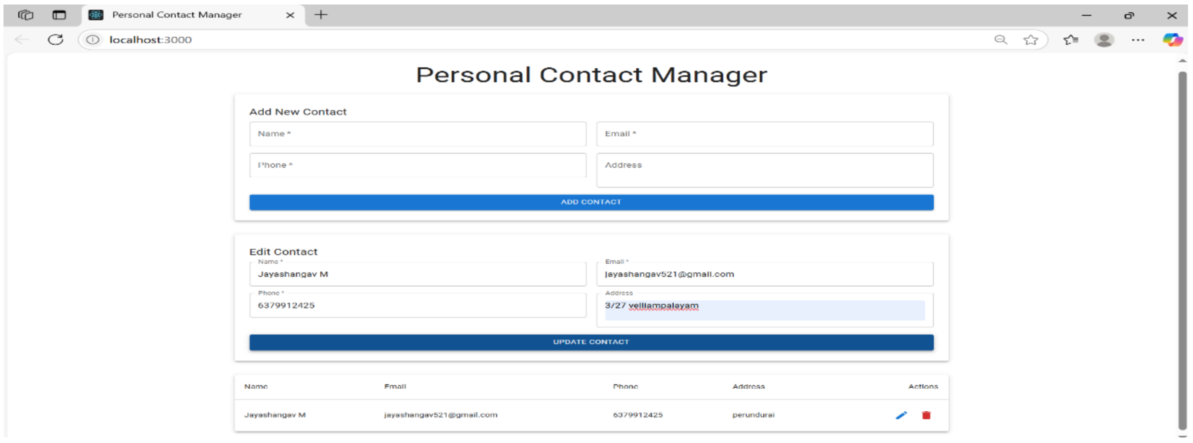
Table: contacts

Columns:

id	int AI PK
name	varchar(100)
email	varchar(100)
phone	varchar(20)
address	text
created_at	timestamp
updated_at	timestamp

```
#  Time      Action                                             Message            Duration / Fetch
1  11:45:25  SELECT * FROM contact_manager.contacts LIMIT 0, 1000  0 row(s) returned  0.000 sec / 0.000 sec
2  11:45:44  SELECT * FROM contact_manager.contacts LIMIT 0, 1000  0 row(s) returned  0.000 sec / 0.000 sec
3  11:47:27  SELECT * FROM contact_manager.contacts LIMIT 0, 1000  1 row(s) returned  0.000 sec / 0.000 sec
```

UPDATE:



Personal Contact Manager

Add New Contact

Name * | Email * | Phone * | Address

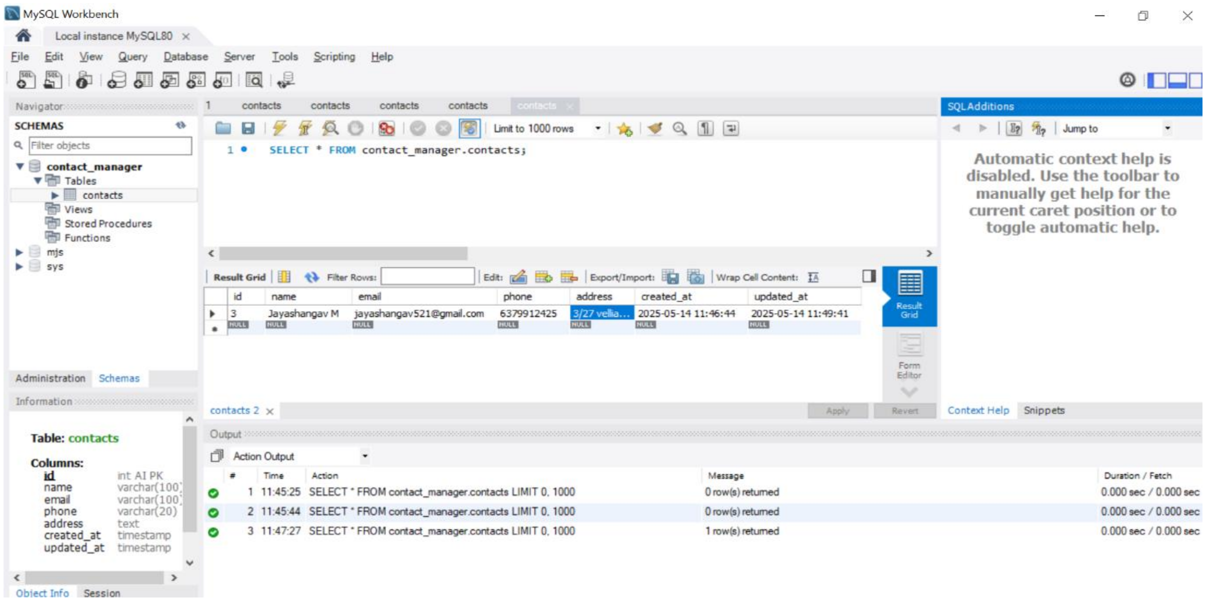
ADD CONTACT

Edit Contact

Name *: Jayashangav M	Email *: jayashangav521@gmail.com
Phone *: 6379912425	Address: 3/27 vellampalayam

UPDATE CONTACT

Name	Email	Phone	Address	Actions
Jayashangav M	jayashangav521@gmail.com	6379912425	perundurai	



```
SELECT * FROM contact_manager.contacts;
```

id	name	email	phone	address	created_at	updated_at
3	Jayashangav M	jayashangav521@gmail.com	6379912425	3/27 vellia...	2025-05-14 11:46:44	2025-05-14 11:49:41
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table: contacts

Columns:

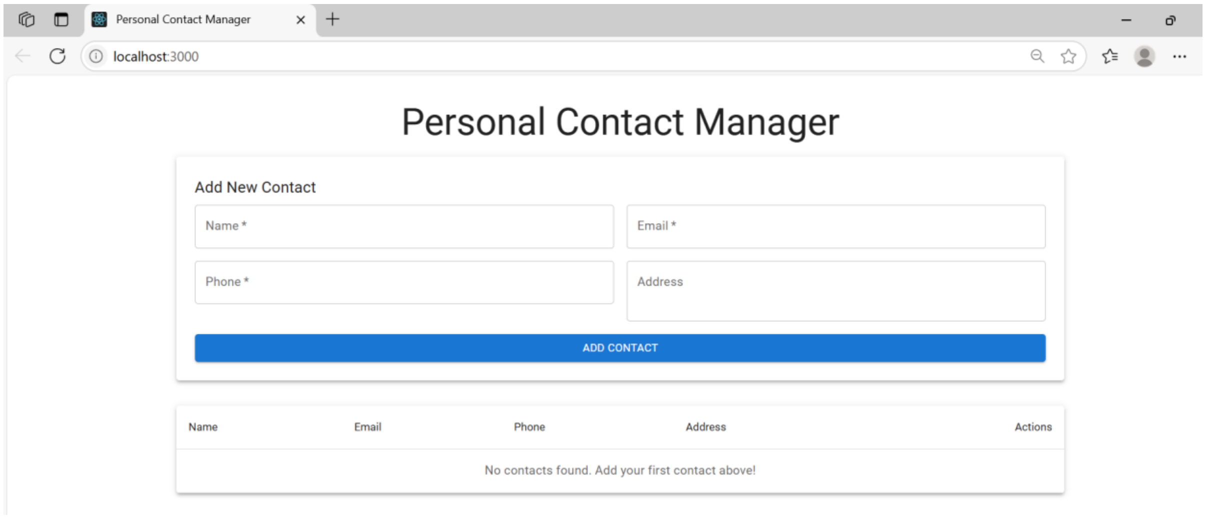
id	int AI PK
name	varchar(100)
email	varchar(100)
phone	varchar(20)
address	text
created_at	timestamp
updated_at	timestamp

Output — Action Output

#	Time	Action	Message	Duration / Fetch
1	11:45:25	SELECT * FROM contact_manager.contacts LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
2	11:45:44	SELECT * FROM contact_manager.contacts LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
3	11:47:27	SELECT * FROM contact_manager.contacts LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

DELETE:



localhost:3000

Personal Contact Manager

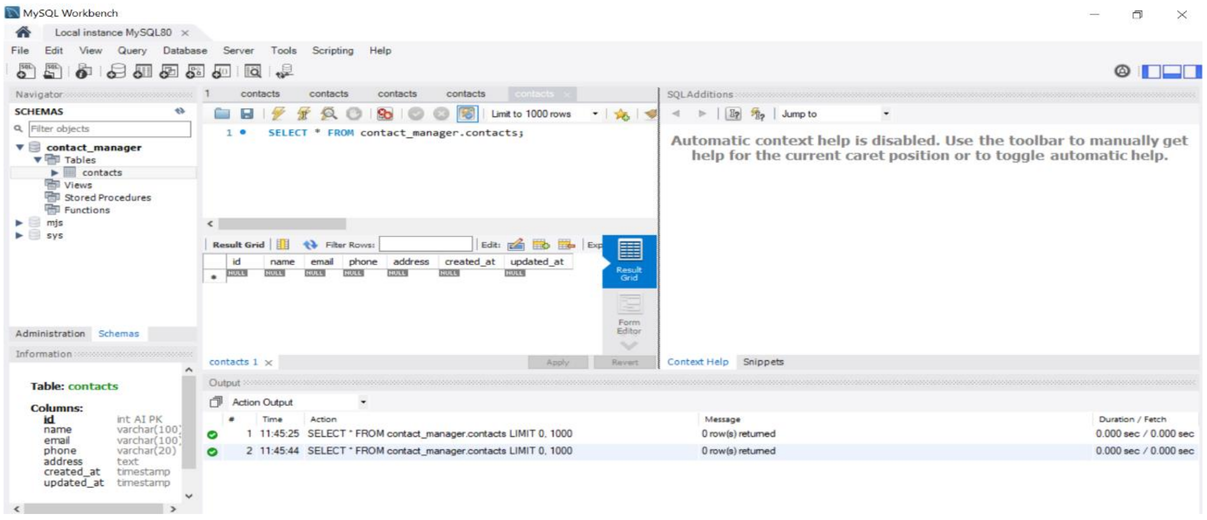
Add New Contact

Name * | Email * | Phone * | Address

ADD CONTACT

Name	Email	Phone	Address	Actions

No contacts found. Add your first contact above!



```
SELECT * FROM contact_manager.contacts;
```

id	name	email	phone	address	created_at	updated_at
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table: contacts

Columns:

id	int AI PK
name	varchar(100)
email	varchar(100)
phone	varchar(20)
address	text
created_at	timestamp
updated_at	timestamp

Output — Action Output

#	Time	Action	Message	Duration / Fetch
1	11:45:25	SELECT * FROM contact_manager.contacts LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
2	11:45:44	SELECT * FROM contact_manager.contacts LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
Aim, Algorithm, SQL, PL/SQL	30	
Execution and Result	20	
Viva	10	
Total	60	

RESULT:

Thus, a alumni database management system was created and performed CRUD operation using Node js and Mysql Workbench.