ASSIGNMENT

By

Aarush Wali

2022A6R002

Semester – 4th

Department of Computer Science and Engineering

(Artificial Intelligence and Machine Learning)

Model Institute of Engineering & Technology (Autonomous)

(Permanently Affiliated to the University of Jammu, Accredited by NAAC with "A" Grade)

Jammu, India

2024

ASSIGNMENT

Subject Name: Database Management System

Subject Code: COM-402

Due Date: 18-05-2024

Question Number	**Course Outcomes**	**Bloom's Level**	**Maximum Marks**
Q1	CO3, CO4, CO5	6	20
Total Marks			20
Faculty Signature: Email: navin.cse@mietjammu.in			

Q. No.	**Question**	**BL**	**CO**	**Marks**	**Total Marks**
1	Using MySQL: 1. Create the tables for the Company database in your text, and populate with data. 2. Create a simple desktop app to load, add and delete the data from database. [Use any language Python tk, c#, .net etc.] 3. Create a Mini Project report for the application you have created.	6	3,4,5	20	20

Abstract

This project presents the development of a Company Database GUI Application using Python, Tkinter, and MySQL Connector to address the challenges associated with managing extensive contact and company data in today's digital age. With the proliferation of digital communication channels and the exponential growth of contact databases, the need for streamlined organization and efficient data management has become increasingly critical. The application aims to provide users with an intuitive interface to add, update, delete, and view company records seamlessly. By leveraging Python's versatility, Tkinter's GUI capabilities, and MySQL Connector's robust database interaction, the application ensures data integrity and security while enhancing operational efficiency. Additionally, the integration of the Pillow library allows for effective image handling and display within the application. Embodying the motto of "innovating tomorrow's solutions today," the project lays the foundation for future enhancements, including advanced data analytics and integration with other business tools. Through its user-centric design and commitment to meeting modern communication demands, the Company Database GUI Application emerges as a pivotal solution for businesses seeking to optimize their data management processes and facilitate efficient communication in an interconnected world.

TABLE OF CONTENT

1. Introduction

In our increasingly interconnected world, the efficiency of communication is fundamentally tied to effective contact management. As digital communication channels continue to multiply and contact databases grow exponentially, the necessity for streamlined organization becomes more critical. Addressing this need, the Tech Solutions Inc. Desktop Application emerges as a pivotal solution. This project is specifically designed to navigate and simplify the complexities associated with managing extensive arrays of contact details.

*1.1 **<u>Background</u>***

The rapid expansion of digital communication has led to a significant increase in the volume and complexity of contact information. Traditional methods of managing this data are insufficient and prone to errors. To address this, we developed a Company Database GUI Application using Python, Tkinter, and MySQL Connector. This application provides an intuitive interface for managing company data, allowing users to add, update, and delete records efficiently. By leveraging these technologies, our project aims to streamline data management processes, ensuring reliable and secure handling of information in today's fast-paced, interconnected environment. This app ensures that users can maintain seamless and efficient communication. Embodying the motto of "innovating tomorrow's solutions today," Tech Solutions Inc. is dedicated to providing advanced tools that meet the evolving demands of modern communication, making it an indispensable asset in any professional or personal setting.

*1.2 **<u>Problem Statement</u>***

In today's digital age, managing large volumes of contact and company data has become increasingly complex and error-prone with traditional methods. Organizations struggle with inefficiencies, data inconsistencies, and security risks due to outdated manual record-keeping and basic spreadsheet use. This lack of effective data management hinders operational efficiency and decision-making. To address these challenges, there is a need for a robust, user-friendly solution that can streamline data handling processes, ensure data integrity, and provide secure, efficient access to information. Our project aims to develop a Company Database GUI Application to meet these critical needs effectively.

1.3 <u>Objective</u>

The objective of this project is to create a user-friendly Company Database GUI Application using Python, Tkinter, and MySQL Connector to enhance the management of company data. The application aims to provide an intuitive interface for users to easily add, update, delete, and view records, ensuring data integrity and security through robust interaction with a MySQL database. By reducing errors and inefficiencies associated with traditional data management methods, the application seeks to enhance operational efficiency and facilitate seamless communication. Additionally, it lays the groundwork for future enhancements, such as advanced data analytics and integration with other business tools.

1.4 <u>Overview of the Technologies Used</u>

The project employs a combination of powerful technologies to develop the Company Database GUI Application. Python, renowned for its versatility and ease of use, serves as the core programming language, providing a robust foundation for the application's development. Tkinter, Python's standard GUI toolkit, facilitates the creation of an intuitive and user-friendly interface, enabling seamless interaction with the application's functionalities. MySQL, a popular relational database management system, is utilized for efficient data storage and retrieval, while MySQL Connector ensures smooth communication between the Python application and the MySQL database.

Additionally, the project leverages various Python libraries to enhance functionality and performance. The io module enables efficient handling of input-output operations, ensuring smooth data flow within the application. The requests library facilitates HTTP requests, allowing the application to interact with web services and APIs seamlessly. Moreover, the Pillow library is utilized for image processing tasks, enabling the application to handle image data effectively.

By integrating these technologies, the project aims to deliver a comprehensive solution for managing company data efficiently and securely, providing a seamless user experience.

2. Literature Overview

The literature overview for this project report encompasses the exploration of existing research, methodologies, and technologies relevant to the development of a Company Database GUI

Application. Key areas of focus include studies on data management practices, GUI development frameworks, database systems, and relevant Python libraries.

Research into data management practices elucidates the challenges and best practices associated with handling extensive contact and company data in today's digital landscape. Studies on GUI development frameworks, particularly those utilizing Python and Tkinter, provide insights into designing intuitive and user-friendly interfaces for efficient data interaction. Additionally, investigations into database systems such as MySQL offer valuable insights into optimizing data storage, retrieval, and security.

Furthermore, literature on relevant Python libraries, including MySQL Connector, io, requests, and Pillow, provides foundational knowledge for leveraging these tools in the development of the application. By synthesizing insights from existing literature, this project aims to build upon established methodologies and technologies to create a robust and effective solution for managing company data.

Through a comprehensive review of relevant literature, this project seeks to identify gaps, challenges, and opportunities in the field, informing the development process and contributing to the advancement of knowledge in database management and GUI application development.

3. System Design

System design ensures the effective implementation of the Tech Solutions Inc. Web Application, providing users with a robust platform for managing their contact details efficiently.

The Company Database GUI Application follows a traditional three-tier architecture, ensuring modularity and scalability. The architecture consists of:

3.1.1 <u>Presentation Layer</u>

Implemented using Tkinter, the presentation layer provides the graphical user interface (GUI) for users to interact with the application. Tkinter's widgets and layout management tools create an intuitive and user-friendly interface, enabling seamless navigation and data manipulation.

3.1.2 <u>Business Logic Layer</u>

The business logic layer is encapsulated within the application's core Python classes and functions, handling essential functionalities such as CRUD (Create, Read, Update, Delete) operations, data validation, and business rules. This layer ensures that all business processes are executed correctly and efficiently.

3.1.3 <u>Data Access Layer</u>

The data access layer interacts with the MySQL database through the MySQL Connector library, facilitating seamless data retrieval and manipulation. This layer ensures secure and efficient communication between the application and the database, maintaining data integrity and supporting complex queries and transactions.

Additionally, the Pillow library is integrated into the application to handle image processing tasks, enabling the display and management of images within the GUI. This comprehensive architecture supports the application's goal of providing a robust and efficient solution for managing company data.

3.2 <u>Database Schema Design</u>

The database schema is designed to efficiently store and manage employee details. The schema consists of a single table named "details" with the following columns:

3.2.1 ID (INT, Primary Key): Unique identifier for each employee record.

3.2.2 name (VARCHAR): Stores the name of the employee.

3.2.3 position (VARCHAR): Stores the position of the employee.

3.2.4 phone_number (VARCHAR): Stores the contact number of the employee.

3.2.5 gender (VARCHAR): Stores the gender of the employee.

3.2.6 photo (VARCHAR): Stores the file path or URL of an optional image associated with the employee.

3.2.7 *department (VARCHAR):* Stores the department of the employee.

3.2.8 salary (FLOAT): Stores the salary of the employee.

3.3 <u>GUI Design</u>

The GUI design of the Tech Solutions Inc. application emphasizes user-friendliness and intuitive navigation, incorporating various elements to enhance usability. The main interface comprises:

3.3.1 <u>Data Grid View Control</u>

Utilized to display contact details in a tabular format, facilitating easy access and navigation of contact records.

3.3.2 <u>User Interaction Elements</u>

1. Buttons: Add, Update, Delete, and Search buttons enable users to perform CRUD operations and search for specific contacts.
2. Textboxes: Labeled input fields for entering contact details such as first name, last name, and contact number.
3. Picture boxes: Optional image display for visual representation of contacts, enhancing user engagement.
4. Labels: Static text elements used to provide descriptions or instructions to users

3.3.3 <u>Search Functionality:</u>

Search bar allows users to search for contacts by entering the first name in the search field, providing quick access to specific contacts.

By integrating these elements, the Tech Solutions Inc. GUI design ensures a seamless and efficient user experience, empowering users to manage their contact details with ease and precision.

3.3.4 <u>Employee Image Popup Functionality:</u>

The application features a functionality where selecting an employee's name displays their image, which is extracted from their LinkedIn profile. This is achieved by integrating the Pillow library, which handles image processing within the GUI. When a user selects an employee from the list, the application sends a request to LinkedIn using the requests library to fetch the employee's profile image. The image is then processed and displayed in the application interface, providing a visual representation of the employee alongside their contact information. This feature enhances the user experience by offering a more personalized and informative data management tool.

3.4 <u>Description of Tools and Technologies Used for Development</u>

By leveraging the listed tools and technologies, the Company Database GUI Application project achieves a comprehensive and efficient development process, enabling the creation of a robust and user-friendly data management application.

3.4.1 <u>Python Programming Language:</u>

Python is used for implementing the application logic and GUI components. Its versatility and extensive library support make it ideal for developing a complex application efficiently.

3.4.2 <u>Tkinter:</u>

Tkinter is employed for designing and developing the graphical user interface (GUI). It facilitates the creation of user-friendly and intuitive interfaces, ensuring a seamless user experience.

3.4.3 <u>MySQL Database:</u>

MySQL is utilized as the backend storage solution for storing and managing company data records. The database management is facilitated through MySQL Server, ensuring reliable and efficient data handling.

3.4.4 <u>MySQL Connector:</u>

MySQL Connector is integrated within the development environment for connecting to and interacting with the MySQL database. It ensures seamless data retrieval and manipulation, maintaining data integrity and security.

3.4.5 <u>Pillow Library:</u>

The Pillow library is used for image processing tasks. It enables the application to fetch and display employee images from LinkedIn profiles, enhancing the visual representation of data.

3.4.6 *Requests Library:*

The requests library is utilized to handle HTTP requests, specifically for fetching images from LinkedIn profiles. This integration allows for real-time retrieval of employee images, enriching the user interface.

3.4.7 *io Module:*

The io module is used for handling input and output operations within the application. It ensures efficient data flow and processing, contributing to the overall performance and reliability of the system.

The Company Database GUI Application employs a structured three-tier architecture for robustness, with a user-friendly GUI designed to streamline data management. Utilizing MySQL database and Python programming, it ensures efficient data handling and seamless user interaction, marking a significant step towards effective data management.

4. Implementation

This section discusses the implementation of the Company Database GUI Application, focusing on user satisfaction and enhanced data management using Python, Tkinter, MySQL Connector, and MySQL Server.

4.1 *Requirement Analysis:*

Through the creation of the Company Database GUI Application, our primary objective is to deliver a comprehensive solution that empowers users to efficiently manage their company and contact details. To achieve this goal, we prioritize the following objectives:

4.1.1 *Seamless and Intuitive Platform:*

The application aims to provide users with a seamless and intuitive platform for managing their company data. By leveraging a user-friendly design approach with Tkinter, we streamline user interactions and minimize the learning curve, ensuring that users can easily navigate the application and perform tasks without encountering unnecessary complexities.

4.1.2 Enhanced User Satisfaction and Productivity:

Central to our approach is the goal of enhancing user satisfaction and productivity. By prioritizing a user-friendly design, we create an interface that not only meets the functional requirements of data management but also delights users with its simplicity and ease of use. This empowerment boosts productivity and overall user satisfaction.

4.1.3 Data Integrity and Accessibility:

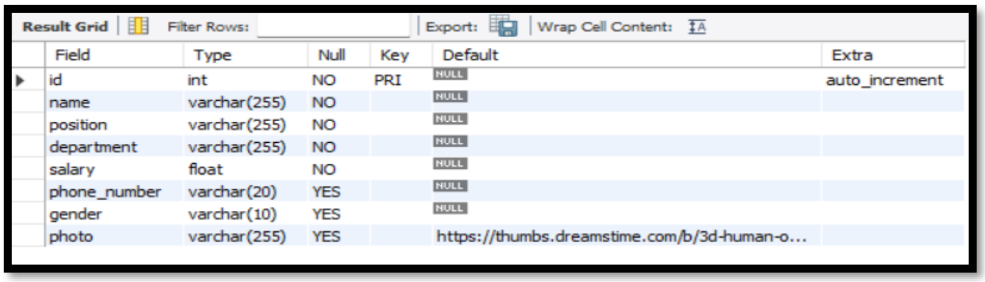
We recognize the critical importance of ensuring data integrity and accessibility in managing company data. To address this, we commit to creating and maintaining a secure MySQL database that safeguards records against unauthorized access, data loss, and corruption. By implementing robust security measures and adhering to best practices in database management, we aim to install confidence in users regarding the reliability and integrity of their data.

4.1.4 Seamless Interaction with Data:

Integration of the database using a GUI application is essential for facilitating seamless interaction with company data. By integrating database functionality directly into the application's graphical user interface through MySQL Connector, we empower users to perform Create, Read, Update, and Delete (CRUD) operations with ease and efficiency. Whether adding new records, updating existing information, or searching for specific entries, users can interact with their data seamlessly, without the need for specialized database management skills.

By implementing these requirements, the Company Database GUI Application strives to provide a robust, secure, and user-friendly environment for efficient data management.

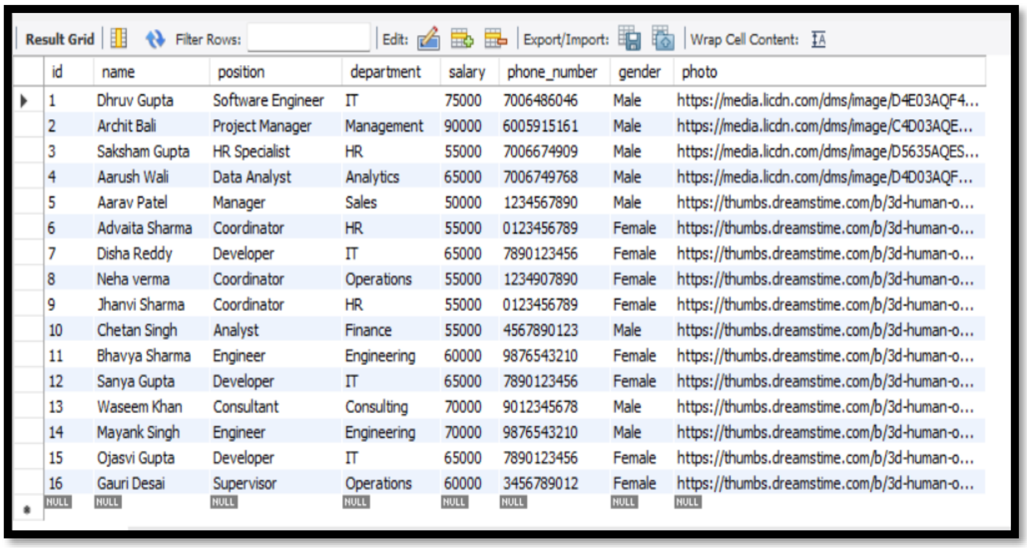
4.2 __Design Phase:__



Result Grid
Filter Rows:
Export:
Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
position	varchar(255)	NO		NULL	
department	varchar(255)	NO		NULL	
salary	float	NO		NULL	
phone_number	varchar(20)	YES		NULL	
gender	varchar(10)	YES		NULL	
photo	varchar(255)	YES		https://thumbs.dreamstime.com/b/3d-human-o...	

__Figure 1 : Table details(datatype)__



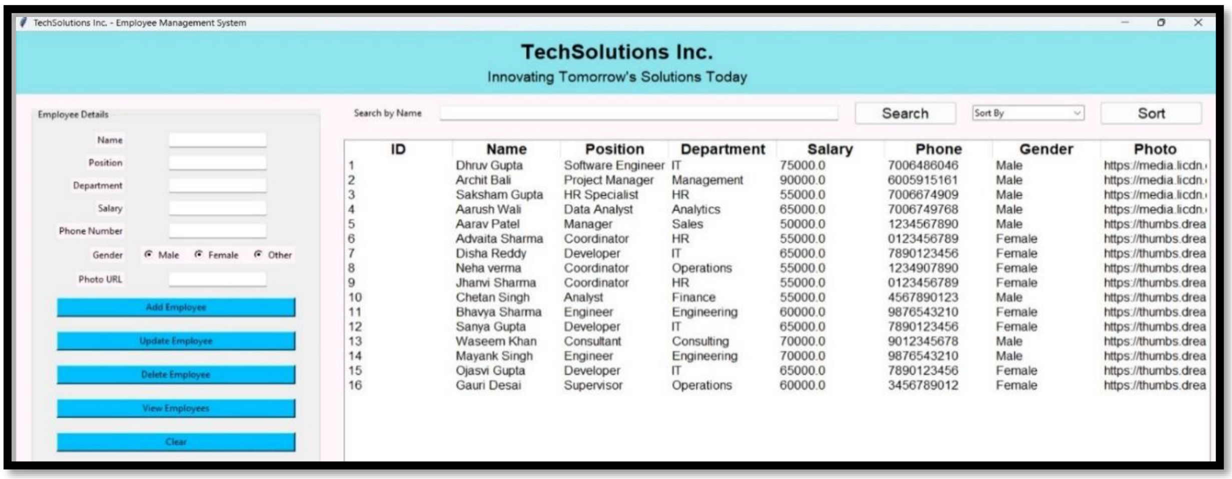
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:

id	name	position	department	salary	phone_number	gender	photo
1	Dhruv Gupta	Software Engineer	IT	75000	7006486046	Male	https://media.licdn.com/dms/image/D4E03AQF4...
2	Archit Bali	Project Manager	Management	90000	6005915161	Male	https://media.licdn.com/dms/image/C4D03AQE...
3	Saksham Gupta	HR Specialist	HR	55000	7006674909	Male	https://media.licdn.com/dms/image/D5635AQES...
4	Aarush Wali	Data Analyst	Analytics	65000	7006749768	Male	https://media.licdn.com/dms/image/D4D03AQF...
5	Aarav Patel	Manager	Sales	50000	1234567890	Male	https://thumbs.dreamstime.com/b/3d-human-o...
6	Advaita Sharma	Coordinator	HR	55000	0123456789	Female	https://thumbs.dreamstime.com/b/3d-human-o...
7	Disha Reddy	Developer	IT	65000	7890123456	Female	https://thumbs.dreamstime.com/b/3d-human-o...
8	Neha verma	Coordinator	Operations	55000	1234907890	Female	https://thumbs.dreamstime.com/b/3d-human-o...
9	Jhanvi Sharma	Coordinator	HR	55000	0123456789	Female	https://thumbs.dreamstime.com/b/3d-human-o...
10	Chetan Singh	Analyst	Finance	55000	4567890123	Male	https://thumbs.dreamstime.com/b/3d-human-o...
11	Bhavya Sharma	Engineer	Engineering	60000	9876543210	Female	https://thumbs.dreamstime.com/b/3d-human-o...
12	Sanya Gupta	Developer	IT	65000	7890123456	Female	https://thumbs.dreamstime.com/b/3d-human-o...
13	Waseem Khan	Consultant	Consulting	70000	9012345678	Male	https://thumbs.dreamstime.com/b/3d-human-o...
14	Mayank Singh	Engineer	Engineering	70000	9876543210	Male	https://thumbs.dreamstime.com/b/3d-human-o...
15	Ojasvi Gupta	Developer	IT	65000	7890123456	Female	https://thumbs.dreamstime.com/b/3d-human-o...
16	Gauri Desai	Supervisor	Operations	60000	3456789012	Female	https://thumbs.dreamstime.com/b/3d-human-o...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 2 : Table details Structure

__Main GUI Web Page:__

When we open the GUI web app, the first page that appears is as given below



ID	Name	Position	Department	Salary	Phone	Gender	Photo
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male	https://media.licdn.
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male	https://media.licdn.
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male	https://media.licdn.
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.drea
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.drea
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female	https://thumbs.drea
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female	https://thumbs.drea
9	Jhanvi Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.drea
10	Chetan Singh	Analyst	Finance	55000.0	4567890123	Male	https://thumbs.drea
11	Bhavya Sharma	Engineer	Engineering	60000.0	9876543210	Female	https://thumbs.drea
12	Sanya Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.drea
13	Waseem Khan	Consultant	Consulting	70000.0	9012345678	Male	https://thumbs.drea
14	Mayank Singh	Engineer	Engineering	70000.0	9876543210	Male	https://thumbs.drea
15	Ojasvi Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.drea
16	Gauri Desai	Supervisor	Operations	60000.0	3456789012	Female	https://thumbs.drea

Figure 3: Application Front View

Through meticulous planning and execution in the design phase, we ensured that our Tech Solution Inc. Web Application embodies efficiency, usability, and data integrity at its core.

4.4 Code:

```
import tkinter as tk
from tkinter import messagebox, ttk
from PIL import Image, ImageTk
import mysql.connector
import requests
from io import BytesIO

# Database setup
def create_database():
    conn = mysql.connector.connect(
        host='localhost',
        user='root',
        password='NC5ZQY@123',
        database='TechSolutions_Inc'
    )
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS employees (
            id INT AUTO_INCREMENT PRIMARY KEY,
            name VARCHAR(255) NOT NULL,
            position VARCHAR(255) NOT NULL,
            department VARCHAR(255) NOT NULL,
            salary FLOAT NOT NULL,
            phone_number VARCHAR(20),
            gender VARCHAR(10),
            photo VARCHAR(255) DEFAULT 'https://thumbs.dreamstime.com/b/3d-
human-ok-16682223.jpg'
        )
    ''')

    conn.commit()
    conn.close()
```

```python
create_database()

# GUI Application
def add_employee():
    name = name_entry.get()
    position = position_entry.get()
    department = department_entry.get()
    salary = salary_entry.get()
    phone_number = phone_entry.get()
    gender = gender_var.get()
    photo = photo_entry.get()

    if not name or not position or not department or not salary or not
phone_number or not gender:
        messagebox.showerror("Error", "All fields except photo are required")
        return

    try:
        salary = float(salary)
    except ValueError:
        messagebox.showerror("Error", "Salary must be a number")
        return

    if not photo:
        photo = 'https://thumbs.dreamstime.com/b/3d-human-ok-16682223.jpg'

    conn = mysql.connector.connect(
        host='localhost',
        user='root',
        password='NC5ZQY@123',
        database='TechSolutions_Inc'
    )
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO employees (name, position, department, salary,
phone_number, gender, photo)
        VALUES (%s, %s, %s, %s, %s, %s, %s)
    ''', (name, position, department, salary, phone_number, gender, photo))

    conn.commit()
    conn.close()
    messagebox.showinfo("Success", "Employee added successfully")
    clear_entries()
    view_employees()

def clear_entries():
    name_entry.delete(0, tk.END)
    position_entry.delete(0, tk.END)
    department_entry.delete(0, tk.END)
    salary_entry.delete(0, tk.END)
    phone_entry.delete(0, tk.END)
    gender_var.set("")
    photo_entry.delete(0, tk.END)
    photo_label.config(image=None)  # Clear the photo label

def view_employees(order_by=None):
    for row in employee_tree.get_children():
        employee_tree.delete(row)

    conn = mysql.connector.connect(
        host='localhost',
        user='root',
        password='NC5ZQY@123',
        database='TechSolutions_Inc'
    )
    cursor = conn.cursor()
    query = 'SELECT * FROM employees'
    if order_by:
        query += f' ORDER BY {order_by}'
    cursor.execute(query)
    rows = cursor.fetchall()
    conn.close()

    for row in rows:
        employee_tree.insert("", tk.END, values=row)

def delete_employee():
    selected_item = employee_tree.selection()
    if not selected_item:
        messagebox.showerror("Error", "Please select an employee to delete")
        return

    employee_id = employee_tree.item(selected_item[0])['values'][0]

    conn = mysql.connector.connect(
        host='localhost',
        user='root',
        password='NC5ZQY@123',
        database='TechSolutions_Inc'
    )
    cursor = conn.cursor()
    cursor.execute('DELETE FROM employees WHERE id=%s', (employee_id,))
    conn.commit()
    conn.close()
    messagebox.showinfo("Success", "Employee deleted successfully")
    view_employees()

def update_employee():
    selected_item = employee_tree.selection()
    if not selected_item:
        messagebox.showerror("Error", "Please select an employee to update")
        return

    name = name_entry.get()
    position = position_entry.get()
    department = department_entry.get()
    salary = salary_entry.get()
    phone_number = phone_entry.get()
    gender = gender_var.get()
    photo = photo_entry.get()

    if not name or not position or not department or not salary or not
phone_number or not gender:
        messagebox.showerror("Error", "All fields except photo are required")
        return

    try:
        salary = float(salary)
    except ValueError:
        messagebox.showerror("Error", "Salary must be a number")
        return

    if not photo:
        photo = 'https://thumbs.dreamstime.com/b/3d-human-ok-16682223.jpg'

    employee_id = employee_tree.item(selected_item[0])['values'][0]

    conn = mysql.connector.connect(
        host='localhost',
        user='root',
        password='NC5ZQY@123',
        database='TechSolutions_Inc'
    )
    cursor = conn.cursor()
    cursor.execute('''
        UPDATE employees
        SET name=%s, position=%s, department=%s, salary=%s, phone_number=%s,
gender=%s, photo=%s
        WHERE id=%s
    ''', (name, position, department, salary, phone_number, gender, photo,
employee_id))

    conn.commit()
    conn.close()
    messagebox.showinfo("Success", "Employee updated successfully")
    view_employees()
```

```python
def search_employees():
    search_query = search_entry.get()
    if not search_query:
        view_employees()
        return

    for row in employee_tree.get_children():
        employee_tree.delete(row)

    conn = mysql.connector.connect(
        host='localhost',
        user='root',
        password='NC5ZQY@123',
        database='TechSolutions_Inc'
    )
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM employees WHERE name LIKE %s',
(f'%{search_query}%',))
    rows = cursor.fetchall()
    conn.close()

    for row in rows:
        employee_tree.insert("", tk.END, values=row)

def add_button_hover_effects(button):
    def on_enter(event):
        button['bg'] = '#006dbf'
    def on_leave(event):
        button['bg'] = button_bg

    button.bind("<Enter>", on_enter)
    button.bind("<Leave>", on_leave)

app = tk.Tk()
app.title("TechSolutions Inc. - Employee Management System")

# Set window size
app.geometry("1000x600")

# Styling
style = ttk.Style()
style.configure("TLabel", font=("Helvetica", 14))
style.configure("TButton", font=("Helvetica", 14))
style.configure("TEntry", font=("Helvetica", 14))
style.configure("Treeview.Heading", font=("Helvetica", 14, "bold"))
style.configure("Treeview", font=("Helvetica", 12))

# Colors
bg_color = "#fff7fc"
```

```
header_bg = "#8ee5ee"
header_fg = "#000000"
button_bg = "#00bfff"
button_fg = "#000000"

app.configure(bg=bg_color)

# Company Header
header_frame = tk.Frame(app, bg=header_bg, pady=10)
header_frame.pack(fill=tk.X)

company_name = tk.Label(header_frame, text="TechSolutions Inc.",
font=("Helvetica", 20, "bold"), bg=header_bg, fg=header_fg)
company_name.pack()

tagline = tk.Label(header_frame, text="Innovating Tomorrow's Solutions Today",
font=("Helvetica", 14), bg=header_bg, fg=header_fg)
tagline.pack()

# Input Form in a labeled frame
input_frame = ttk.LabelFrame(app, text="Employee Details", padding=(20, 10))
input_frame.pack(side=tk.LEFT, fill=tk.Y, padx=20, pady=20)


tk.Label(input_frame, text="Name", bg=bg_color).grid(row=0, column=0, padx=10,
pady=5, sticky="e")
tk.Label(input_frame, text="Position", bg=bg_color).grid(row=1, column=0,
padx=10, pady=5, sticky="e")
tk.Label(input_frame, text="Department", bg=bg_color).grid(row=2, column=0,
padx=10, pady=5, sticky="e")
tk.Label(input_frame, text="Salary", bg=bg_color).grid(row=3, column=0,
padx=10, pady=5, sticky="e")
tk.Label(input_frame, text="Phone Number", bg=bg_color).grid(row=4, column=0,
padx=10, pady=5, sticky="e")
tk.Label(input_frame, text="Gender", bg=bg_color).grid(row=5, column=0,
padx=10, pady=5, sticky="e")
tk.Label(input_frame, text="Photo URL", bg=bg_color).grid(row=6, column=0,
padx=10, pady=5, sticky="e")

name_entry = ttk.Entry(input_frame)
position_entry = ttk.Entry(input_frame)
department_entry = ttk.Entry(input_frame)
salary_entry = ttk.Entry(input_frame)
phone_entry = ttk.Entry(input_frame)
gender_var = tk.StringVar()
```

```
gender_frame = tk.Frame(input_frame, bg=bg_color)
gender_male = tk.Radiobutton(gender_frame, text="Male", variable=gender_var,
value="Male", bg=bg_color)
gender_female = tk.Radiobutton(gender_frame, text="Female",
variable=gender_var, value="Female", bg=bg_color)
gender_other = tk.Radiobutton(gender_frame, text="Other", variable=gender_var,
value="Other", bg=bg_color)
photo_entry = ttk.Entry(input_frame)

name_entry.grid(row=0, column=1, padx=10, pady=5)
position_entry.grid(row=1, column=1, padx=10, pady=5)
department_entry.grid(row=2, column=1, padx=10, pady=5)
salary_entry.grid(row=3, column=1, padx=10, pady=5)
phone_entry.grid(row=4, column=1, padx=10, pady=5)
gender_frame.grid(row=5, column=1, padx=10, pady=5, sticky="w")
gender_male.pack(side=tk.LEFT)
gender_female.pack(side=tk.LEFT, padx=(10, 0))
gender_other.pack(side=tk.LEFT, padx=(10, 0))
photo_entry.grid(row=6, column=1, padx=10, pady=5)

add_button = tk.Button(input_frame, text="Add Employee", command=add_employee,
bg=button_bg, fg=button_fg)
add_button.grid(row=7, column=0, columnspan=2, padx=10, pady=10, sticky="we")
add_button_hover_effects(add_button)

update_button = tk.Button(input_frame, text="Update Employee",
command=update_employee, bg=button_bg, fg=button_fg)
update_button.grid(row=8, column=0, columnspan=2, padx=10, pady=10,
sticky="we")
add_button_hover_effects(update_button)

delete_button = tk.Button(input_frame, text="Delete Employee",
command=delete_employee, bg=button_bg, fg=button_fg)
delete_button.grid(row=9, column=0, columnspan=2, padx=10, pady=10,
sticky="we")
add_button_hover_effects(delete_button)


view_button = tk.Button(input_frame, text="View Employees",
command=view_employees, bg=button_bg, fg=button_fg)
view_button.grid(row=10, column=0, columnspan=2, padx=10, pady=10,
sticky="we")
add_button_hover_effects(view_button)

# Clear Button
clear_button = tk.Button(input_frame, text="Clear", command=clear_entries,
bg=button_bg, fg=button_fg)
```

```
clear_button.grid(row=11, column=0, columnspan=2, padx=10, pady=10,
sticky="we")
add_button_hover_effects(clear_button)

# Search and Sort Frame
search_sort_frame = tk.Frame(app, padx=10, pady=10, bg=bg_color)
search_sort_frame.pack(fill=tk.X)

tk.Label(search_sort_frame, text="Search by Name",
bg=bg_color).pack(side=tk.LEFT, padx=10)
search_entry = ttk.Entry(search_sort_frame)
search_entry.pack(side=tk.LEFT, padx=10, fill=tk.X, expand=True)
search_button = ttk.Button(search_sort_frame, text="Search",
command=search_employees)
search_button.pack(side=tk.LEFT, padx=10)
add_button_hover_effects(search_button)

# Sort dropdown
sort_by_var = tk.StringVar()
sort_by_dropdown = ttk.Combobox(search_sort_frame, textvariable=sort_by_var,
values=["Name", "Salary"], state="readonly")
sort_by_dropdown.set("Sort By")
sort_by_dropdown.pack(side=tk.LEFT, padx=10)

def sort_employees():
    order_by = sort_by_var.get().lower()
    if order_by == "name":
        view_employees(order_by="name")
    elif order_by == "salary":
        view_employees(order_by="salary")

sort_button = ttk.Button(search_sort_frame, text="Sort",
command=sort_employees)
sort_button.pack(side=tk.LEFT, padx=10)
add_button_hover_effects(sort_button)

# Employee List
list_frame = tk.Frame(app, padx=10, pady=10, bg=bg_color)
list_frame.pack(fill=tk.BOTH, expand=True)

columns = ("ID", "Name", "Position", "Department", "Salary", "Phone",
"Gender", "Photo")
employee_tree = ttk.Treeview(list_frame, columns=columns, show="headings")

for col in columns:
    employee_tree.heading(col, text=col)
    employee_tree.column(col, minwidth=0, width=100)

employee_tree.pack(fill=tk.BOTH, expand=True)
```

```
# Image Display
photo_label = tk.Label(list_frame, bg=bg_color)
photo_label.pack(pady=10)

# Function to display image and populate entry fields when an employee is
selected
def show_employee_image(event):
    selected_item = employee_tree.selection()
    if not selected_item:
        return

    employee_data = employee_tree.item(selected_item[0])['values']
    name_entry.delete(0, tk.END)
    name_entry.insert(0, employee_data[1])
    position_entry.delete(0, tk.END)
    position_entry.insert(0, employee_data[2])
    department_entry.delete(0, tk.END)
    department_entry.insert(0, employee_data[3])
    salary_entry.delete(0, tk.END)
    salary_entry.insert(0, str(employee_data[4]))
    phone_entry.delete(0, tk.END)
    phone_entry.insert(0, employee_data[5])
    gender_var.set(employee_data[6])
    photo_entry.delete(0, tk.END)
    photo_entry.insert(0, employee_data[7])

    employee_photo_url = employee_data[7]

    try:
        response = requests.get(employee_photo_url)
        response.raise_for_status()
        image_data = Image.open(BytesIO(response.content))
        image_data.thumbnail((200, 200))
        photo = ImageTk.PhotoImage(image_data)
        photo_label.config(image=photo)
        photo_label.image = photo
    except requests.exceptions.RequestException as e:
        messagebox.showerror("Error", f"Failed to load image: {e}")
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {e}")
```

```
# Bind the show_employee_image function to the Treeview selection event
employee_tree.bind("<<TreeviewSelect>>", show_employee_image)

# Initialize with employee data
view_employees()

app.mainloop()
```

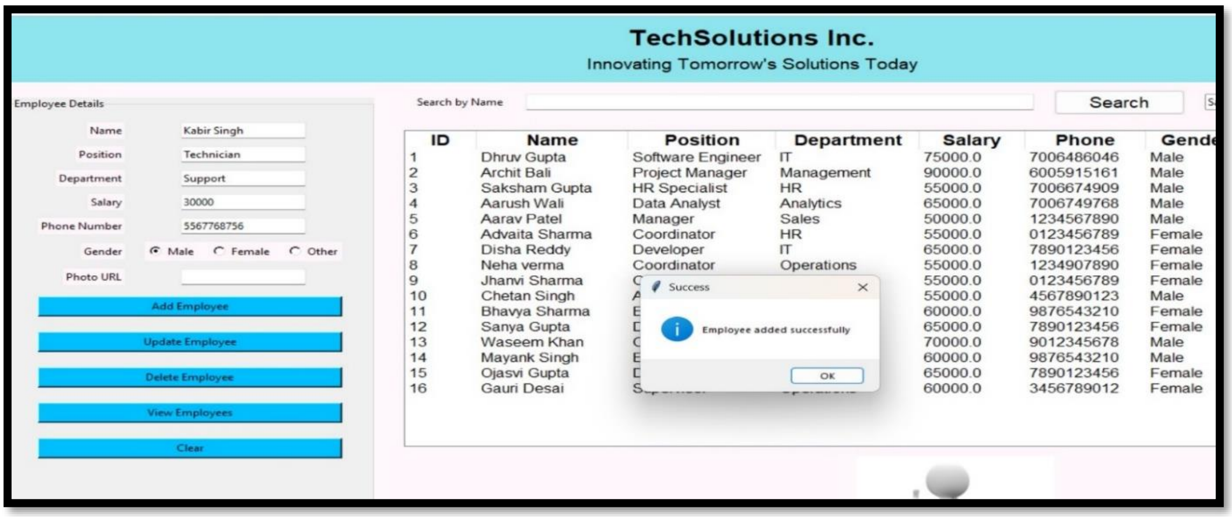
4.5 __Testing__

Here's how the testing process incorporates the operations- adding, updating, deleting, sorting, searching, checking employee's picture. :-

4.4.1 __Addition of Employees:__

During testing, we meticulously validate the application's ability to add new candidates to the database.

__Adding Data:__



Employee Details

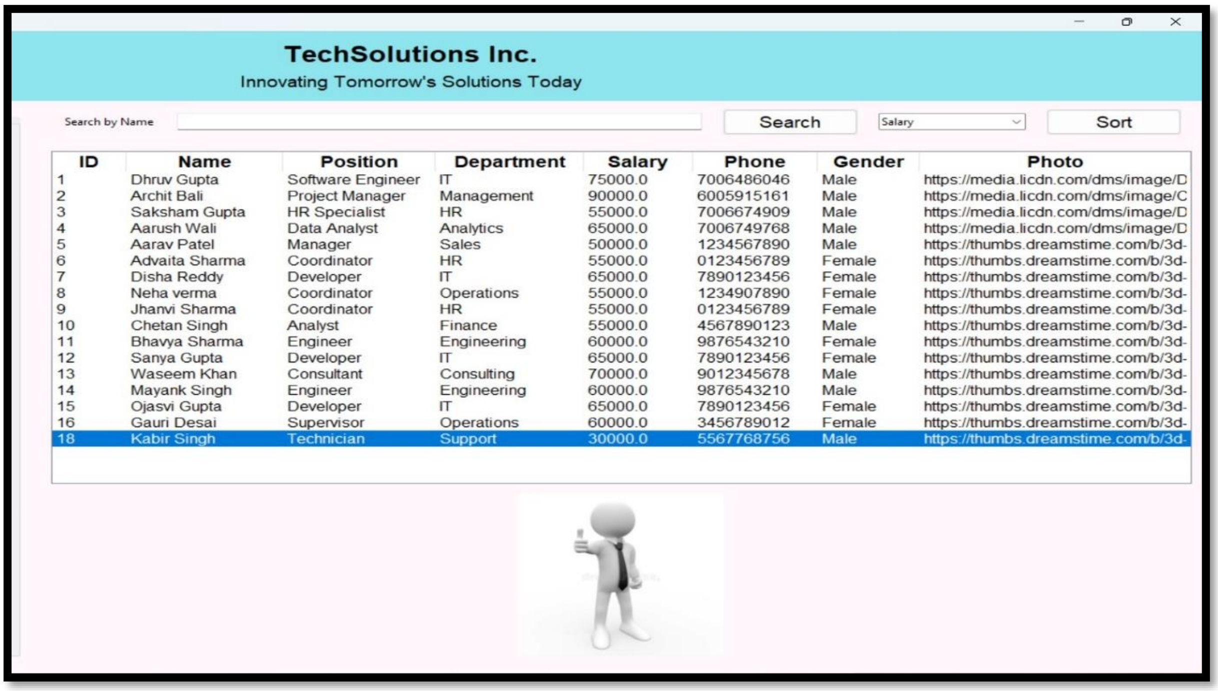
Name	Kabir Singh
Position	Technician
Department	Support
Salary	30000
Phone Number	5567768756
Gender	[✓ Male] Female Other
Photo URL	

ID	Name	Position	Department	Salary	Phone	Gende
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female
9	Jhanvi Sharma	C		55000.0	0123456789	Female
10	Chetan Singh	A		55000.0	4567890123	Male
11	Bhavya Sharma	E		60000.0	9876543210	Female
12	Sanya Gupta	D		65000.0	7890123456	Female
13	Waseem Khan	C		70000.0	9012345678	Male
14	Mayank Singh	E		60000.0	9876543210	Male
15	Ojasvi Gupta	D		65000.0	7890123456	Female
16	Gauri Desai	S[illegible]	[illegible]	60000.0	3456789012	Female

Employee added successfully

__Figure 4: Addition (in Procedure)__

<u>After adding Data:</u>



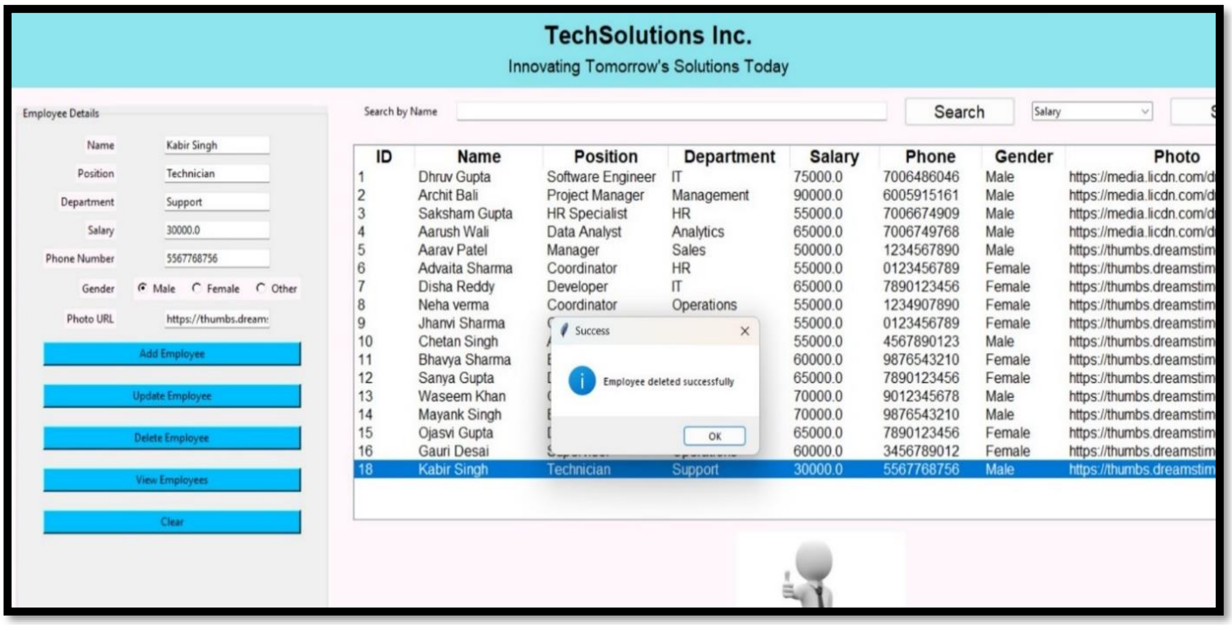
ID	Name	Position	Department	Salary	Phone	Gender	Photo
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male	https://media.licdn.com/dms/image/D
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male	https://media.licdn.com/dms/image/C
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male	https://media.licdn.com/dms/image/D
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.com/dms/image/D
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.dreamstime.com/b/3d-
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female	https://thumbs.dreamstime.com/b/3d-
9	Jhanvi Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
10	Chetan Singh	Analyst	Finance	55000.0	4567890123	Male	https://thumbs.dreamstime.com/b/3d-
11	Bhavya Sharma	Engineer	Engineering	60000.0	9876543210	Female	https://thumbs.dreamstime.com/b/3d-
12	Sanya Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
13	Waseem Khan	Consultant	Consulting	70000.0	9012345678	Male	https://thumbs.dreamstime.com/b/3d-
14	Mayank Singh	Engineer	Engineering	60000.0	9876543210	Male	https://thumbs.dreamstime.com/b/3d-
15	Ojasvi Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
16	Gauri Desai	Supervisor	Operations	60000.0	3456789012	Female	https://thumbs.dreamstime.com/b/3d-
18	Kabir Singh	Technician	Support	30000.0	5567768756	Male	https://thumbs.dreamstime.com/b/3d-

<u>Figure 5: After Addition</u>

4.4.2 <u>Deletion of Contacts:</u>

Testing encompasses the application's ability to delete unwanted employees records from the database. Test cases verify the deletion process for individual contact entries and bulk deletion of multiple records.

Deletion in process:



Name	Kabir Singh
Position	Technician
Department	Support
Salary	30000.0
Phone Number	5567768756
Gender	[✓ Male] Female Other
Photo URL	https://thumbs.dream:

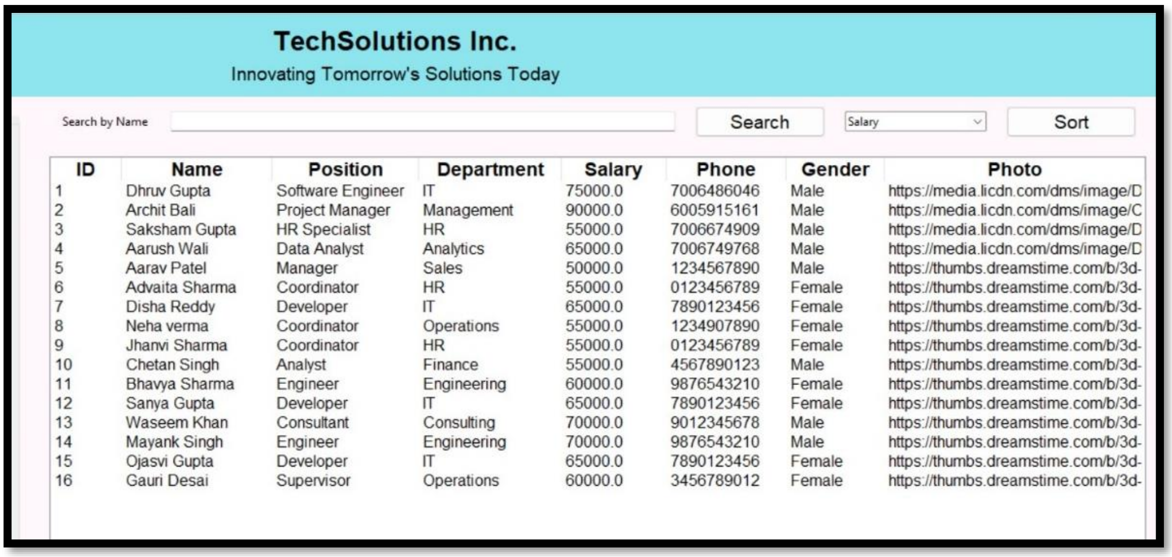
Sort: Salary

ID	Name	Position	Department	Salary	Phone	Gender	Photo
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male	https://media.licdn.com/d
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male	https://media.licdn.com/d
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male	https://media.licdn.com/d
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.com/d
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.dreamstim
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstim
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstim
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female	https://thumbs.dreamstim
9	Jhanvi Sharma			55000.0	0123456789	Female	https://thumbs.dreamstim
10	Chetan Singh			55000.0	4567890123	Male	https://thumbs.dreamstim
11	Bhavya Sharma			60000.0	9876543210	Female	https://thumbs.dreamstim
12	Sanya Gupta			65000.0	7890123456	Female	https://thumbs.dreamstim
13	Waseem Khan			70000.0	9012345678	Male	https://thumbs.dreamstim
14	Mayank Singh			70000.0	9876543210	Male	https://thumbs.dreamstim
15	Ojasvi Gupta			65000.0	7890123456	Female	https://thumbs.dreamstim
16	Gauri Desai			60000.0	3456789012	Female	https://thumbs.dreamstim
18	Kabir Singh	Technician	Support	30000.0	5567768756	Male	https://thumbs.dreamstim

Employee deleted successfully

Figure 6: Deletion

After Deletion of Data:



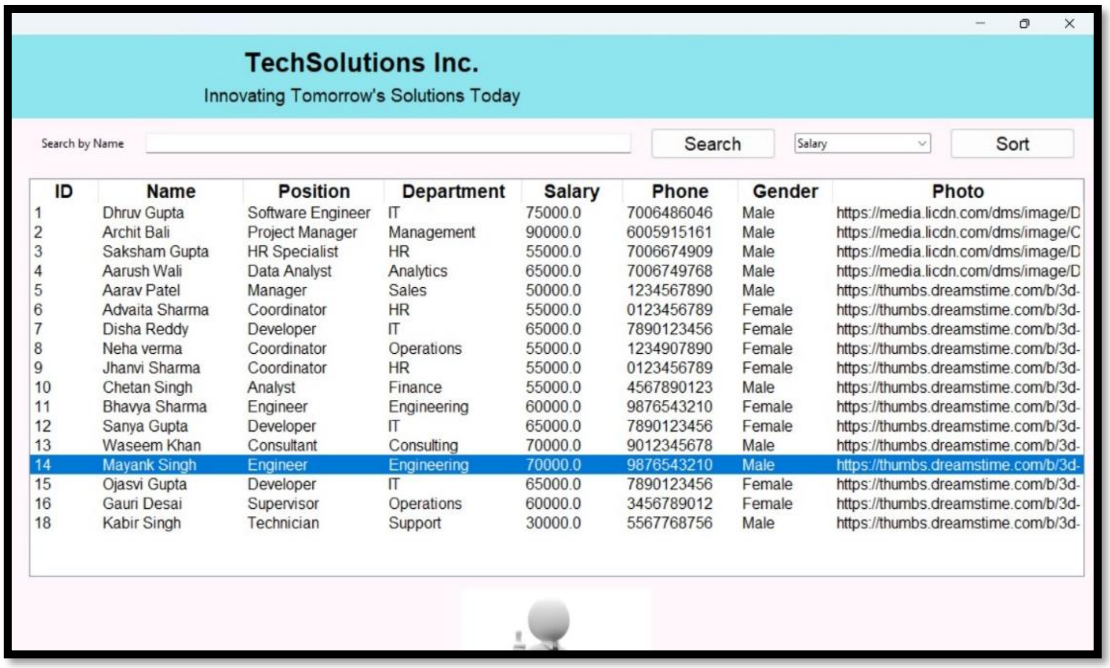
Salary

ID	Name	Position	Department	Salary	Phone	Gender	Photo
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male	https://media.licdn.com/dms/image/D
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male	https://media.licdn.com/dms/image/C
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male	https://media.licdn.com/dms/image/D
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.com/dms/image/D
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.dreamstime.com/b/3d-
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female	https://thumbs.dreamstime.com/b/3d-
9	Jhanvi Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
10	Chetan Singh	Analyst	Finance	55000.0	4567890123	Male	https://thumbs.dreamstime.com/b/3d-
11	Bhavya Sharma	Engineer	Engineering	60000.0	9876543210	Female	https://thumbs.dreamstime.com/b/3d-
12	Sanya Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
13	Waseem Khan	Consultant	Consulting	70000.0	9012345678	Male	https://thumbs.dreamstime.com/b/3d-
14	Mayank Singh	Engineer	Engineering	70000.0	9876543210	Male	https://thumbs.dreamstime.com/b/3d-
15	Ojasvi Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
16	Gauri Desai	Supervisor	Operations	60000.0	3456789012	Female	https://thumbs.dreamstime.com/b/3d-

Figure 7: After Deletion

4.4.3 Updation:

Here the employee's salary has been updated by Rs 10,000 in the details table, as shown in the figure 8.



TechSolutions Inc.

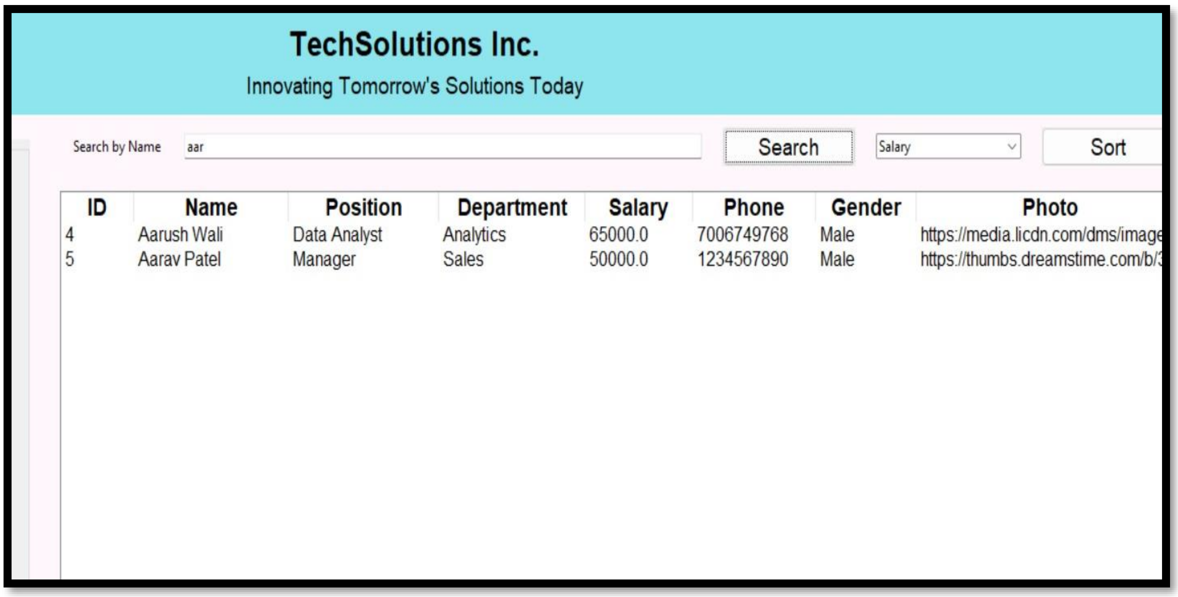
Innovating Tomorrow's Solutions Today

ID	Name	Position	Department	Salary	Phone	Gender	Photo
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male	https://media.licdn.com/dms/image/D
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male	https://media.licdn.com/dms/image/C
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male	https://media.licdn.com/dms/image/D
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.com/dms/image/D
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.dreamstime.com/b/3d-
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female	https://thumbs.dreamstime.com/b/3d-
9	Jhanvi Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
10	Chetan Singh	Analyst	Finance	55000.0	4567890123	Male	https://thumbs.dreamstime.com/b/3d-
11	Bhavya Sharma	Engineer	Engineering	60000.0	9876543210	Female	https://thumbs.dreamstime.com/b/3d-
12	Sanya Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
13	Waseem Khan	Consultant	Consulting	70000.0	9012345678	Male	https://thumbs.dreamstime.com/b/3d-
14	Mayank Singh	Engineer	Engineering	70000.0	9876543210	Male	https://thumbs.dreamstime.com/b/3d-
15	Ojasvi Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
16	Gauri Desai	Supervisor	Operations	60000.0	3456789012	Female	https://thumbs.dreamstime.com/b/3d-
18	Kabir Singh	Technician	Support	30000.0	5567768756	Male	https://thumbs.dreamstime.com/b/3d-

Figure 8: After Updating

4.4.4 Search By name:

Employee's name can be searched through search bar, leading to time management and efficient data accessibility.



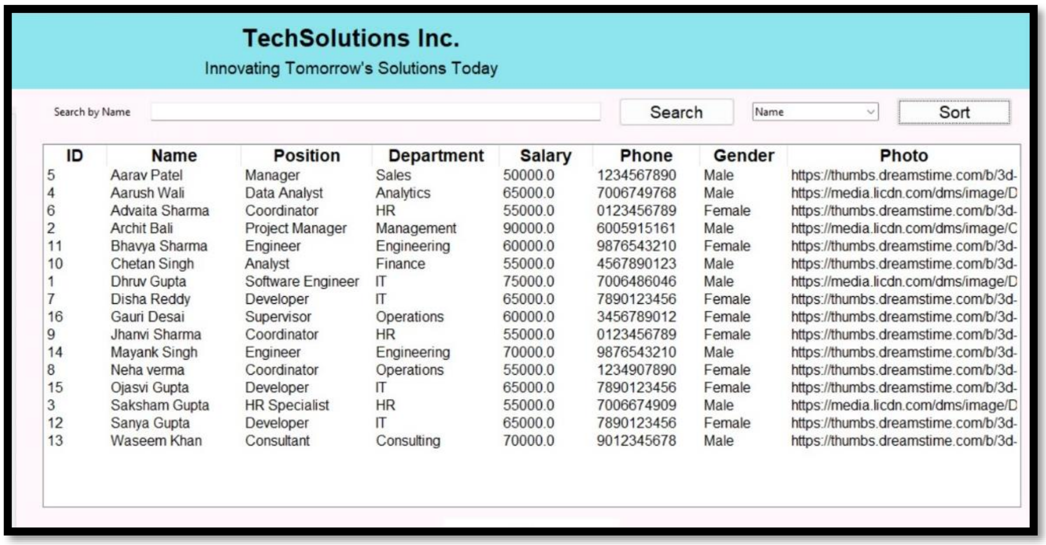
Search by Name: aar

ID	Name	Position	Department	Salary	Phone	Gender	Photo
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.com/dms/image
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.dreamstime.com/b/3

Figure 9: Search process

4.4.5 *Sorting of Employees (By name, salary):*

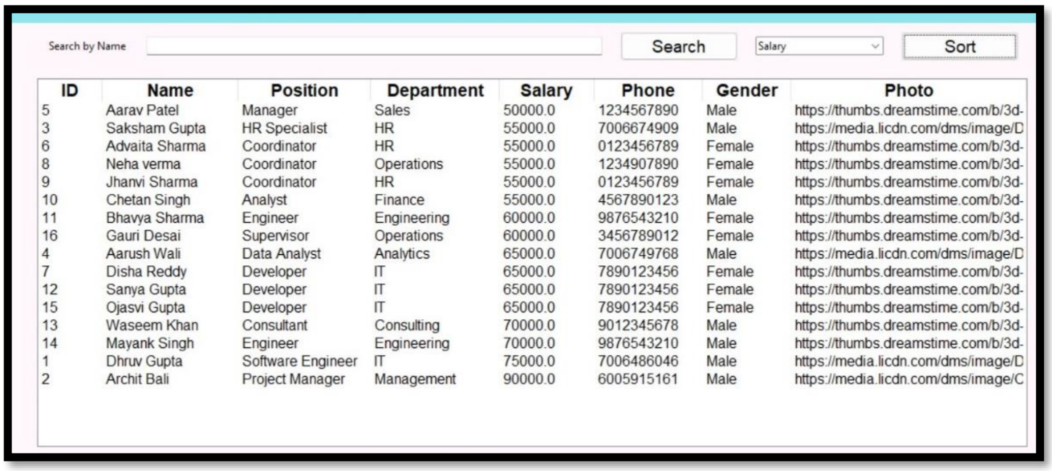
Sorting (By name)-



ID	Name	Position	Department	Salary	Phone	Gender	Photo
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.dreamstime.com/b/3d-
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.com/dms/image/D
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male	https://media.licdn.com/dms/image/C
11	Bhavya Sharma	Engineer	Engineering	60000.0	9876543210	Female	https://thumbs.dreamstime.com/b/3d-
10	Chetan Singh	Analyst	Finance	55000.0	4567890123	Male	https://thumbs.dreamstime.com/b/3d-
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male	https://media.licdn.com/dms/image/D
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
16	Gauri Desai	Supervisor	Operations	60000.0	3456789012	Female	https://thumbs.dreamstime.com/b/3d-
9	Jhanvi Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
14	Mayank Singh	Engineer	Engineering	70000.0	9876543210	Male	https://thumbs.dreamstime.com/b/3d-
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female	https://thumbs.dreamstime.com/b/3d-
15	Ojasvi Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male	https://media.licdn.com/dms/image/D
12	Sanya Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
13	Waseem Khan	Consultant	Consulting	70000.0	9012345678	Male	https://thumbs.dreamstime.com/b/3d-

Figure 10: Sorting By Name

Sorting (By salary)-

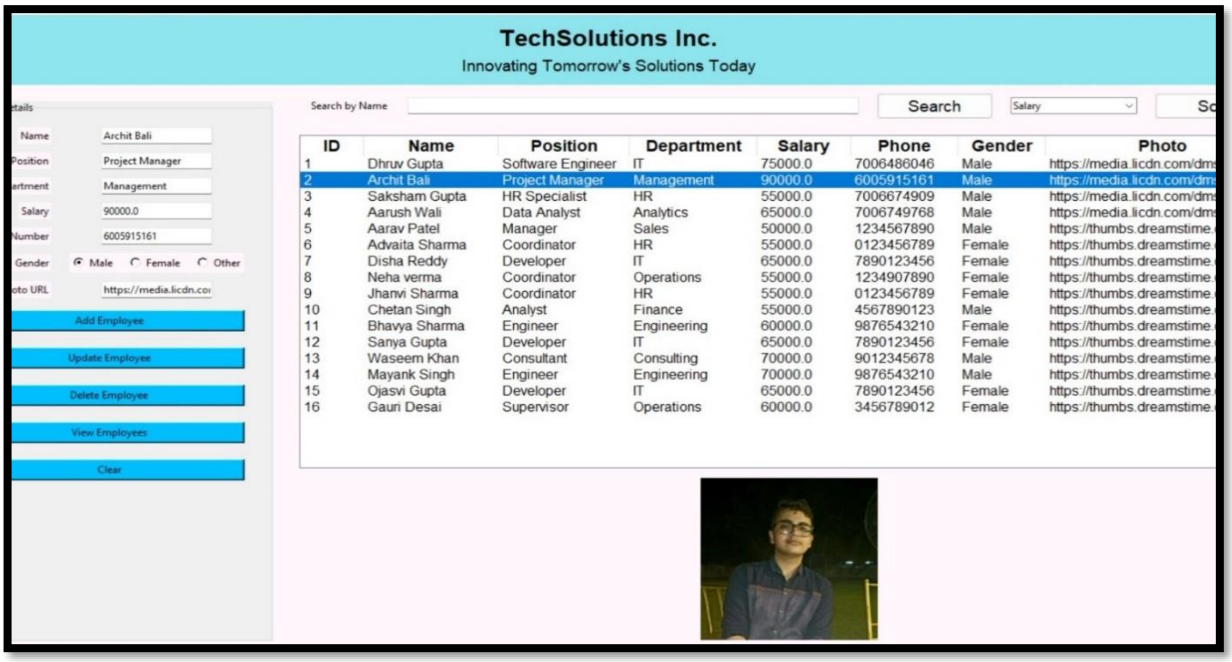


ID	Name	Position	Department	Salary	Phone	Gender	Photo
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.dreamstime.com/b/3d-
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male	https://media.licdn.com/dms/image/D
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female	https://thumbs.dreamstime.com/b/3d-
9	Jhanvi Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.com/b/3d-
10	Chetan Singh	Analyst	Finance	55000.0	4567890123	Male	https://thumbs.dreamstime.com/b/3d-
11	Bhavya Sharma	Engineer	Engineering	60000.0	9876543210	Female	https://thumbs.dreamstime.com/b/3d-
16	Gauri Desai	Supervisor	Operations	60000.0	3456789012	Female	https://thumbs.dreamstime.com/b/3d-
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.com/dms/image/D
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
12	Sanya Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
15	Ojasvi Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.com/b/3d-
13	Waseem Khan	Consultant	Consulting	70000.0	9012345678	Male	https://thumbs.dreamstime.com/b/3d-
14	Mayank Singh	Engineer	Engineering	70000.0	9876543210	Male	https://thumbs.dreamstime.com/b/3d-
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male	https://media.licdn.com/dms/image/D
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male	https://media.licdn.com/dms/image/C

Figure 11: Sorting By Salary

4.4.6 *Checking employee's image:*

When a user selects an employee from the list, the application sends a request to LinkedIn using the requests library to fetch the employee's profile image. The image is then processed and displayed in the application interface, providing a visual representation of the employee alongside their contact information.



Name	Archit Bali
osition	Project Manager
artment	Management
Salary	90000.0
Number	6005915161
Gender	[✓ Male] Female Other
oto URL	https://media.licdn.coı

ID	Name	Position	Department	Salary	Phone	Gender	Photo
1	Dhruv Gupta	Software Engineer	IT	75000.0	7006486046	Male	https://media.licdn.com/dms
2	Archit Bali	Project Manager	Management	90000.0	6005915161	Male	https://media.licdn.com/dms
3	Saksham Gupta	HR Specialist	HR	55000.0	7006674909	Male	https://media.licdn.com/dms
4	Aarush Wali	Data Analyst	Analytics	65000.0	7006749768	Male	https://media.licdn.com/dms
5	Aarav Patel	Manager	Sales	50000.0	1234567890	Male	https://thumbs.dreamstime.
6	Advaita Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.
7	Disha Reddy	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.
8	Neha verma	Coordinator	Operations	55000.0	1234907890	Female	https://thumbs.dreamstime.
9	Jhanvi Sharma	Coordinator	HR	55000.0	0123456789	Female	https://thumbs.dreamstime.
10	Chetan Singh	Analyst	Finance	55000.0	4567890123	Male	https://thumbs.dreamstime.
11	Bhavya Sharma	Engineer	Engineering	60000.0	9876543210	Female	https://thumbs.dreamstime.
12	Sanya Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.
13	Waseem Khan	Consultant	Consulting	70000.0	9012345678	Male	https://thumbs.dreamstime.
14	Mayank Singh	Engineer	Engineering	70000.0	9876543210	Male	https://thumbs.dreamstime.
15	Ojasvi Gupta	Developer	IT	65000.0	7890123456	Female	https://thumbs.dreamstime.
16	Gauri Desai	Supervisor	Operations	60000.0	3456789012	Female	https://thumbs.dreamstime.

Figure 12: Employee's Image Retrieval

When we select the employee's name it shows their linked in's profile picture which makes it easier to recognize them (as shown in figure 12).

By incorporating visual documentation, such as screenshots or screencasts, into the testing process, we provide comprehensive evidence of the CRUD functionalities in action. This visual representation not only facilitates effective communication among team members but also serves as valuable documentation for stakeholders, demonstrating the robustness and usability of the Tech Solutions Inc. web Application.

5. Results and Discussion

5.1 Presentation of the Final GUI Application:

The final GUI application of the Company Database GUI Application embodies a user-friendly interface designed to streamline company and contact data management tasks. Featuring intuitive navigation, responsive controls, and visually appealing design elements, the application offers users a seamless experience for interacting with their database. Integration with the Pillow library allows for the display of employee images, enhancing the visual appeal and functionality of the interface.

5.2 Achieved Results in Comparison to Project Objectives:

5.2.1 Objective 1: Provide a Seamless and Intuitive Platform:

The GUI application successfully delivers on this objective by offering a user-friendly platform for managing employee details efficiently. With clear navigation and intuitive design, users can easily perform various operations such as adding, updating, deleting, and viewing records with minimal effort.

5.2.2 Objective 2: Enhance User Satisfaction and Productivity:

The application's emphasis on usability and responsiveness contributes to enhanced user satisfaction and productivity. By prioritizing a user-friendly design approach, the application minimizes the learning curve and empowers users to accomplish tasks effortlessly. The integration of sorting and search functionalities further enhances productivity by allowing users to quickly find and organize data according to their needs.

5.2.3 Objective 3: Ensure Data Integrity and Accessibility:

The integration of robust database functionalities ensures data integrity and accessibility. Utilizing MySQL for secure storage and MySQL Connector for efficient data retrieval, the application guarantees the reliability of contact records. Users can trust the accuracy and security of their data, knowing it is protected against unauthorized access, data loss, and corruption. The ability to update information and perform CRUD operations seamlessly also contributes to maintaining up-to-date and accurate data.

Through these results, the Company Database GUI Application meets its project objectives, providing a robust, secure, and user-friendly solution for managing company and contact data efficiently.

5.3 Future Enhancements

5.3.1 Integration of Advanced Data Analytics:

Integrating advanced data analytics capabilities into the application can provide valuable insights into employee performance, trends, and patterns. By leveraging data visualization techniques and statistical analysis, users can gain deeper insights into their company data, enabling informed decision-making and strategic planning.

5.3.2 Implementation of Real-time Collaboration Features:

Enhancing the application with real-time collaboration features enables users to collaborate on projects and share information seamlessly. Features such as live chat, document sharing, and collaborative editing enhance team collaboration and communication, fostering a more dynamic and productive work environment.

5.3.3 <u>Integration with Third-party APIs:</u>

Integrating with third-party APIs allows the application to access external data sources and services, expanding its functionality and capabilities. Integration with APIs such as LinkedIn, Google Contacts, or Microsoft Outlook can streamline data import/export processes, enhance data accuracy, and provide additional functionalities for users.

5.3.4 <u>Mobile Application Development:</u>

Developing a mobile application companion to the desktop application extends its reach and accessibility, allowing users to manage their company data on the go. A mobile application offers flexibility and convenience, enabling users to access, update, and interact with their data from anywhere, anytime.

6. Conclusion

The completion of the Tech Solution Inc. Desktop Application marks a significant achievement in the realm of contact management software. Throughout the development process, our project has remained steadfast in its commitment to delivering a solution that prioritizes user satisfaction, data integrity, and accessibility.

With its intuitive user interface and comprehensive features, including CRUD functionalities and seamless integration with the MySQL database, the Tech Solution Inc. Desktop Application simplifies the complexities of contact management. Users can effortlessly add, view, update, and delete contact records, thanks to the application's user-friendly design and adherence to best practices in GUI development and database management.

Looking towards the future, there are numerous opportunities for further enhancement and refinement. Implementation of user authentication and authorization mechanisms will bolster security, while refining UI/UX design elements will ensure a more engaging and efficient user experience. Additionally, integrating robust data validation mechanisms will enhance the reliability and accuracy of the application's data management capabilities.

Overall, the Tech Solution Inc. Desktop Application stands as a testament to our dedication to innovation and excellence in software development. By addressing the diverse needs of users and leveraging emerging technologies, we have created a valuable tool for efficient and organized contact management. As technology continues to evolve, we remain committed to evolving alongside it, ensuring that the Tech Solution Inc. Desktop Application remains a relevant and indispensable asset for businesses and individuals alike.

7. References:

1. Date, C. J. (2003). An Introduction to Database Systems (8th ed.). Addison-Wesley.
2. Campbell, M. (2019). Tkinter GUI Application Development Blueprints: Master GUI programming in Tkinter as you design, implement, and deliver 10 real-world applications. Packt Publishing.
3. Rob, P., & Coronel, C. (2015). Database Systems: Design, Implementation, and Management (12th ed.). Cengage Learning.
4. Dawson, M. W. (2018). Python Programming for the Absolute Beginner (4th ed.). Cengage Learning.
5. Welling, L., & Thomson, L. (2016). PHP and MySQL Web Development (5th ed.). Addison-Wesley.
6. Sweigart, A. (2019). Automate the Boring Stuff with Python: Practical Programming for Total Beginners (2nd ed.). No Starch Press.
7. Reddy, K. R. (2018). Mastering Python Networking: Your one-stop solution to using Python for network automation, DevOps, and Test-Driven Development (2nd ed.). Packt Publishing.
8. Lecky-Thompson, G. (2017). Python GUI Programming Cookbook: Develop functional and responsive user interfaces with tkinter and PyQt5. Packt Publishing.
9. High, P. (2018). MySQL Connector/Python Revealed: SQL and NoSQL data storage using MySQL for Python programming. Apress.
10. Heath, D. (2015). Tkinter By Example. Packt Publishing.

DRIVE LINK FOR PROJECT VIDEO:

https://drive.google.com/file/d/1tIDok1ZJM5wezQjjSH2xCm5DWXa_SN9B/view?usp=sharing

GITHUB LINK FOR PROJECT:

https://github.com/dhruvgupta9431/dbms/blob/main/company.py

Assignment 2 -- Practice writing Queries

The Order-Processing Database

The file mentioned (see appendix) above contains the SQL table creation commands, and the data to be inserted into the tables. You should be able to quickly create the database, then create the tables from the file and enter the data.

The database for this assignment contains data that supports a simple order processing application for a small distribution company. It consists of five tables:

1. The **CUSTOMERS** table stores data about each customer, such as the company name, credit limit, and the salesperson who calls on the customer.
2. The **SALESREP** table stores the employee number, name, age, year-to-date sales and other data about each salesperson.
3. The **OFFICES** table stores data about each of the five sales offices including the city where the office is located, the sales region to which it belongs, an so on.
4. The **ORDERS** table keeps track of every order placed by a customer, identifying the salesperson who took the order (not necessarily the salesperson who calls on the customer), the product ordered, the quantity and amount of the order, and so on. For simplicity, each order is for only one product.
5. The **PRODUCTS** table stores data about each product available for sale, such as the manufacturer, product number, description, and price.

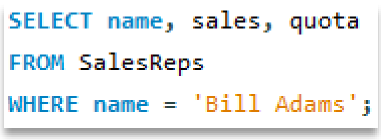
The Code is as follows-

```
Create DATABASE Intermediate;
USE Intermediate;
CREATE TABLE ORDERS(
   ORDER_NUM VARCHAR(6) PRIMARY KEY,
   ORDER_DATE VARCHAR(20),
   CUST CHAR(4),
   REP CHAR(3),
   MFR CHAR(3),
   PRODUCT VARCHAR(10),
   QTY INT,
   AMOUNT DECIMAL(10, 2)
);

CREATE TABLE PRODUCTS(
   MFR_ID CHAR(3),
   PRODUCT_ID VARCHAR(10),
   DESCRIPTION VARCHAR(20),
   PRICE DECIMAL(10, 2), -- Using DECIMAL for monetary values
   QTY_ON_HAND INT,
   CONSTRAINT pk_products PRIMARY KEY (MFR_ID, PRODUCT_ID)
);

CREATE TABLE CUSTOMERS(
   CUST_NUM CHAR(4) PRIMARY KEY,
   COMPANY VARCHAR(20),
   CUST_REP CHAR(3),
   CREDIT_LIMIT DECIMAL(10, 2) -- Assuming 10 digits with 2 decimal places for
money values
);

CREATE TABLE OFFICES(
   OFFICE CHAR(2) PRIMARY KEY,
   CITY VARCHAR(20),
   REGION VARCHAR(10),
   MGR CHAR(3),
   TARGET DECIMAL(10, 2), -- Adjust precision and scale as needed
   SALES DECIMAL(10, 2)   -- Adjust precision and scale as needed
);

CREATE TABLE SalesReps(
   emp_num CHAR(3) PRIMARY KEY,
   name VARCHAR(20),
   age INT,
   rep_office CHAR(2),
   title VARCHAR(10),
   manager CHAR(3),
   hire_date VARCHAR(20),
   quota DECIMAL(10, 2), -- Adjust precision and scale as needed for monetary values
   sales DECIMAL(10, 2) -- Adjust precision and scale as needed for monetary values
);

INSERT INTO CUSTOMERS VALUES('2101', 'Jones Mfg.', '106', '65000');
INSERT INTO CUSTOMERS VALUES('2102', 'First Corp.', '101', '65000');
INSERT INTO CUSTOMERS VALUES('2103', 'Acme Mfg.', '105', '50000');
INSERT INTO CUSTOMERS VALUES('2105', 'AAA Investments','101', '45000' );
INSERT INTO CUSTOMERS VALUES('2106', 'Fred Lewis Corp.', '102', '65000');
INSERT INTO CUSTOMERS VALUES('2107', 'Ace International', '110', '35000');
INSERT INTO CUSTOMERS VALUES('2108', 'Holm & Landis', '109', '55000');
INSERT INTO CUSTOMERS VALUES('2109', 'Chen Associates', '103', '25000');
INSERT INTO CUSTOMERS VALUES('2111', 'JCP Inc.', '103', '50000');
INSERT INTO CUSTOMERS VALUES('2112', 'Zetacorp', '108', '50000');
INSERT INTO CUSTOMERS VALUES('2113', 'Ian & Schmidt', '104', '20000');
INSERT INTO CUSTOMERS VALUES('2114', 'Orion Corp.', '102', '20000');
INSERT INTO CUSTOMERS VALUES('2115', 'Smithson Corp.', '101', '20000');
INSERT INTO CUSTOMERS VALUES('2117', 'J.P. Sinclair', '106', '35000');
INSERT INTO CUSTOMERS VALUES('2118', 'Miswest Sytems', '108', '60000');
INSERT INTO CUSTOMERS VALUES('2119', 'Solomon Inc.', '109', '25000');
INSERT INTO CUSTOMERS VALUES('2120', 'Rico Enterprises', '102', '50000');
INSERT INTO CUSTOMERS VALUES('2121', 'QMA Assoc.', '103', '54000');
INSERT INTO CUSTOMERS VALUES('2122', 'Three-Way Lines', '105', '30000');
INSERT INTO CUSTOMERS VALUES('2123', 'Carter & sons', '102', '40000');
INSERT INTO CUSTOMERS VALUES('2124', 'Peter Brothers', '107', '40000');

INSERT INTO OFFICES VALUES ('11','New York','Eastern','106','575000','692637');
INSERT INTO OFFICES VALUES ('12','Chicago','Eastern','104','800000','735042');
INSERT INTO OFFICES VALUES ('13','Atlanta','Eastern','105','350000','367911');
INSERT INTO OFFICES VALUES ('21','Los Angeles','Western','108','725000','835915');
INSERT INTO OFFICES VALUES ('22','Denver','Western','108','300000','186042');

INSERT INTO ORDERS VALUES ('112961','17-Dec-
99','2117','106','REI','2A44L',7,'31500');
INSERT INTO ORDERS VALUES ('112963','17-Dec-
99','2103','105','ACI','41004',28,'3276');
INSERT INTO ORDERS VALUES ('112968','12-Oct-
99','2102','101','ACI','41004',34,'3978');
INSERT INTO ORDERS VALUES ('112975','12-Oct-
99','2111','103','REI','2A44G',6,'2100');
INSERT INTO ORDERS VALUES ('112979','12-Oct-
99','2114','102','ACI','4100Z',6,'15000');
INSERT INTO ORDERS VALUES ('112983','27-Dec-99','2103','105','ACI','41004',6,'702');
INSERT INTO ORDERS VALUES ('112987','31-Dec-
99','2103','105','ACI','4100Y',11,'27500');
INSERT INTO ORDERS VALUES ('112989','03-Jan-00','2101','106','FEA','114',6,'1458');
INSERT INTO ORDERS VALUES ('112992','04-Nov-99','2118','108','ACI','41002',10,'760');
INSERT INTO ORDERS VALUES ('112993','04-Jan-
99','2106','102','REI','2A45C',24,'1896');
INSERT INTO ORDERS VALUES ('112997','08-Jan-00','2124','107','BIC','41003',1,'652');
INSERT INTO ORDERS VALUES ('113003','25-Jan-00','2108','109','IMM','779C',3,'5625');
INSERT INTO ORDERS VALUES ('113007','08-Jan-00','2112','108','IMM','773C',3,'2925');
INSERT INTO ORDERS VALUES ('113012','11-Jan-
00','2111','105','ACI','41003',35,'3745');
INSERT INTO ORDERS VALUES ('113013','14-Jan-00','2118','108','BIC','41003',1,'652');
INSERT INTO ORDERS VALUES ('113024','20-Jan-
00','2114','108','QSA','XK47',20,'7100');
INSERT INTO ORDERS VALUES ('113027','22-Jan-
00','2103','105','ACI','41002',54,'4104');
INSERT INTO ORDERS VALUES ('113034','29-Jan-00','2107','110','REI','2A45C',8,'632');
INSERT INTO ORDERS VALUES ('113036','30-Jan-
00','2107','110','ACI','4100Z',9,'22500');
INSERT INTO ORDERS VALUES ('113042','02-Feb-
00','2113','101','REI','2A44R',5,'22500');
INSERT INTO ORDERS VALUES ('113045','02-Feb-
00','2112','108','REI','2A44R',10,'45000');
INSERT INTO ORDERS VALUES ('113048','10-Feb-00','2120','102','IMM','779C',2,'3750');
INSERT INTO ORDERS VALUES ('113049','10-Feb-00','2118','108','QSA','XK47',2,'776');
INSERT INTO ORDERS VALUES ('113051','10-Feb-00','2118','108','QSA','XK47',4,'1420');
INSERT INTO ORDERS VALUES ('113055','15-Feb-00','2108','101','ACI','4100X',6,'150');
INSERT INTO ORDERS VALUES ('113057','18-Feb-00','2111','103','ACI','4100X',24,'600');
INSERT INTO ORDERS VALUES ('113058','23-Feb-00','2108','109','FEA','112',10,'1480');
INSERT INTO ORDERS VALUES ('113062','24-Feb-00','2124','107','FEA','114',10,'2430');
INSERT INTO ORDERS VALUES ('113065','27-Feb-00','2106','102','QSA','XK47',6,'2130');
INSERT INTO ORDERS VALUES ('113069','02-Mar-
00','2109','107','IMM','775C',22,'31350');

INSERT INTO PRODUCTS VALUES ('ACI','41002','Size 2 Widget', '76',167);
INSERT INTO PRODUCTS VALUES ('ACI','41003','Size 3 Widget','107',207);
INSERT INTO PRODUCTS VALUES ('ACI','41004','Size 4 Widget','117',139);
INSERT INTO PRODUCTS VALUES ('ACI','4100X','Widget Adjuster','25',37);
INSERT INTO PRODUCTS VALUES ('ACI','4100Y','Widget Remover','2750',25);
INSERT INTO PRODUCTS VALUES ('ACI','4100Z','Size 1 Widget','55',277);
INSERT INTO PRODUCTS VALUES ('ACI','4101','Widget Intaller','2500',28);
INSERT INTO PRODUCTS VALUES ('BIC','41003','Handle','652',3);
INSERT INTO PRODUCTS VALUES ('BIC','41089','Retainer','225',78);
INSERT INTO PRODUCTS VALUES ('BIC','41675','Plate','180',0);
INSERT INTO PRODUCTS VALUES ('FEA','112','Housing','148',115);
INSERT INTO PRODUCTS VALUES ('FEA','114','Motor Mount','243',5);
INSERT INTO PRODUCTS VALUES ('IMM','773C','300-lb Brace','975',28);
INSERT INTO PRODUCTS VALUES ('IMM','775C','500 -lb Brace','1425',5);
INSERT INTO PRODUCTS VALUES ('IMM','779C','900 -lb Brace','1875',9);
INSERT INTO PRODUCTS VALUES ('IMM','887H','Brace Holder','54',223);
INSERT INTO PRODUCTS VALUES ('IMM','887P','Brace Pin','250',24);
INSERT INTO PRODUCTS VALUES ('IMM','887X','Brace Retainer','475',32);
INSERT INTO PRODUCTS VALUES ('QSA','XK47','Reducer','355',15);
INSERT INTO PRODUCTS VALUES ('QSA','XK48','Reducer','134',203);
INSERT INTO PRODUCTS VALUES ('QSA','XK48A','Reducer','177',37);
INSERT INTO PRODUCTS VALUES ('REI','2A44G','Hinge Pin','350',14);
INSERT INTO PRODUCTS VALUES ('REI','2A44L','Left Hinge','4500',12);
INSERT INTO PRODUCTS VALUES ('REI','2A44R','Right Hinge','4500',12);
INSERT INTO PRODUCTS VALUES ('REI','2A45C','Ratchet Link','79',210);

INSERT INTO SALESREPS VALUES ('101','Dan Roberts',45,'12','Sales Rep', '104', '1996-
10-20','300000','305673');
INSERT INTO SALESREPS VALUES ('102','Sue Smith',48,'21','Sales Rep','108','10-Dec-
96', '350000','474050');
INSERT INTO SALESREPS VALUES ('103','Paul Cruz',29,'12','Sales Rep','104','01-Mar-
97', '275000','286775');
INSERT INTO SALESREPS VALUES ('104','Bob Smith',33,'12','Sales Mrg','106','19-May-
97', '200000','142594');
INSERT INTO SALESREPS VALUES ('105','Bill Adams',37,'13','Sales Rep','104','12-Feb-
96', '350000','367911');
INSERT INTO SALESREPS VALUES ('106','Sam Clark',52,'11','Vp Sales',null,'14-Jun-98',
'275000','299912');
INSERT INTO SALESREPS VALUES ('107','Nancy Angelli',49,'22','Sales Rep','108','14-
Nov-98','300000','186042');
INSERT INTO SALESREPS VALUES ('108','Larry Fitch',62,'21','Sales Mrg', '106','12-Oct-
99', '350000','361865');
INSERT INTO SALESREPS VALUES ('109','Mary Jones',31,'11','Sales Rep',  '106', '12-Oct-
99','300000','392725');
```

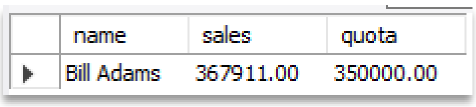
INSERT INTO SALESREPS VALUES ('110','Tom Snyder',41,NULL,'Sales Rep', '101','13-Jan-00', NULL,'75985');

select * from customers;
select * from offices;
select * from orders;
select * from products;
select * from salesreps;

Q1. Show the name, sales, and quota of Bill Adams?

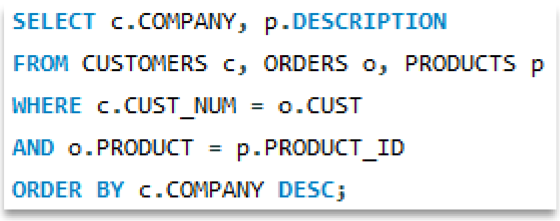


```
SELECT name, sales, quota
FROM SalesReps
WHERE name = 'Bill Adams';
```

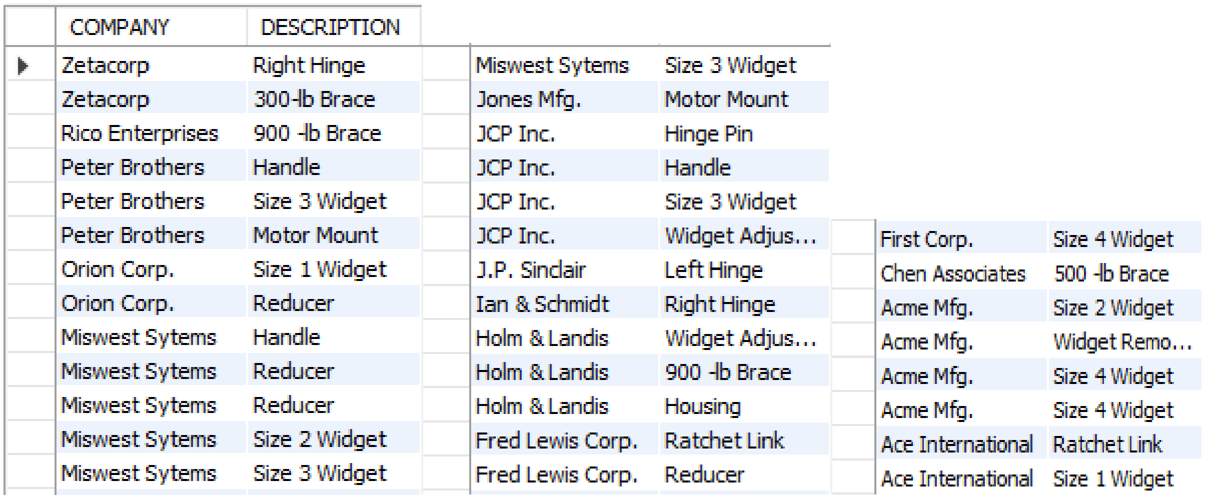


name	sales	quota
Bill Adams	367911.00	350000.00

Q2. List the company names and the product description of all the products each has ordered. Arrange descending by company?

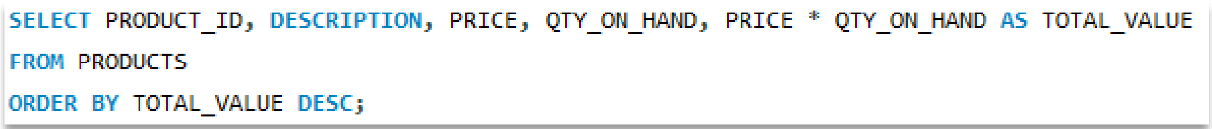


```
SELECT c.COMPANY, p.DESCRIPTION
FROM CUSTOMERS c, ORDERS o, PRODUCTS p
WHERE c.CUST_NUM = o.CUST
AND o.PRODUCT = p.PRODUCT_ID
ORDER BY c.COMPANY DESC;
```

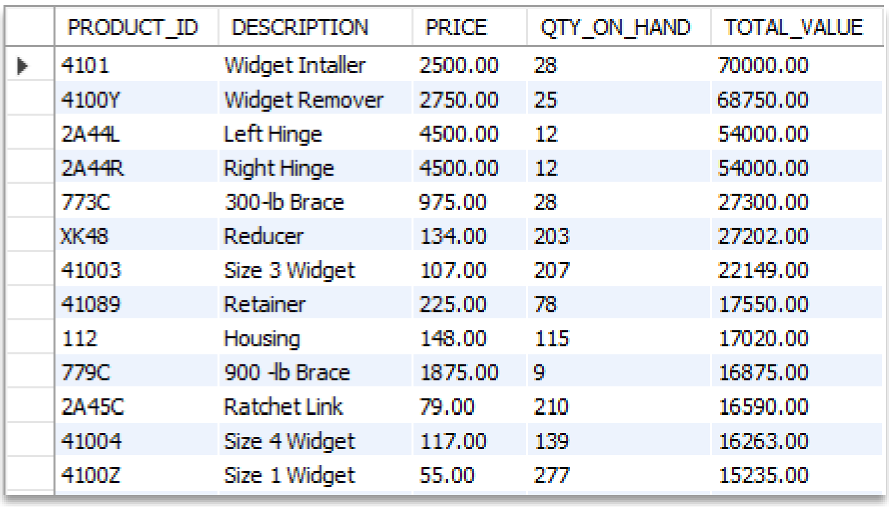


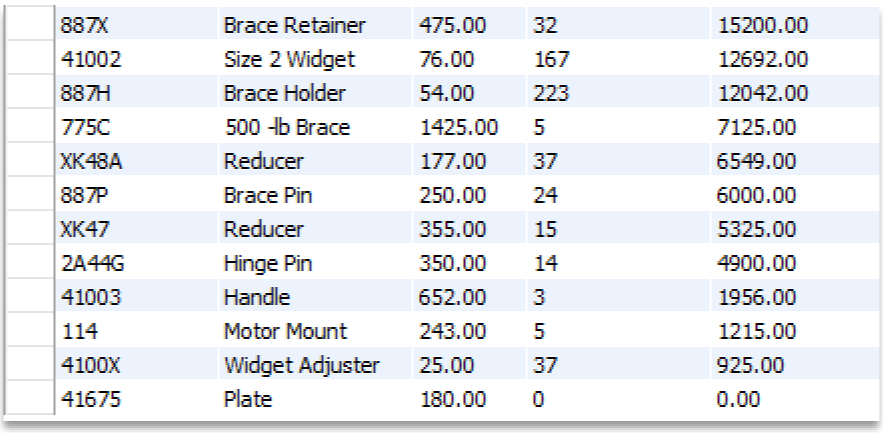
COMPANY	DESCRIPTION
Zetacorp	Right Hinge
Zetacorp	300-lb Brace
Rico Enterprises	900 -lb Brace
Peter Brothers	Handle
Peter Brothers	Size 3 Widget
Peter Brothers	Motor Mount
Orion Corp.	Size 1 Widget
Orion Corp.	Reducer
Miswest Sytems	Handle
Miswest Sytems	Reducer
Miswest Sytems	Reducer
Miswest Sytems	Size 2 Widget
Miswest Sytems	Size 3 Widget
Miswest Sytems	Size 3 Widget
Jones Mfg.	Motor Mount
JCP Inc.	Hinge Pin
JCP Inc.	Handle
JCP Inc.	Size 3 Widget
JCP Inc.	Widget Adjus...
J.P. Sinclair	Left Hinge
Ian & Schmidt	Right Hinge
Holm & Landis	Widget Adjus...
Holm & Landis	900 -lb Brace
Holm & Landis	Housing
Fred Lewis Corp.	Ratchet Link
Fred Lewis Corp.	Reducer
First Corp.	Size 4 Widget
Chen Associates	500 -lb Brace
Acme Mfg.	Size 2 Widget
Acme Mfg.	Widget Remo...
Acme Mfg.	Size 4 Widget
Acme Mfg.	Size 4 Widget
Ace International	Ratchet Link
Ace International	Size 1 Widget

Q3. Show the total value of the inventory on hand for each product. Arrange in descending order by total value?



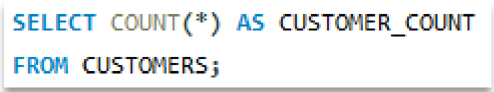
```
SELECT PRODUCT_ID, DESCRIPTION, PRICE, QTY_ON_HAND, PRICE * QTY_ON_HAND AS TOTAL_VALUE
FROM PRODUCTS
ORDER BY TOTAL_VALUE DESC;
```



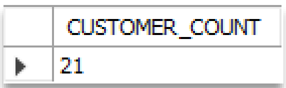


PRODUCT_ID	DESCRIPTION	PRICE	QTY_ON_HAND	TOTAL_VALUE
4101	Widget Intaller	2500.00	28	70000.00
4100Y	Widget Remover	2750.00	25	68750.00
2A44L	Left Hinge	4500.00	12	54000.00
2A44R	Right Hinge	4500.00	12	54000.00
773C	300-lb Brace	975.00	28	27300.00
XK48	Reducer	134.00	203	27202.00
41003	Size 3 Widget	107.00	207	22149.00
41089	Retainer	225.00	78	17550.00
112	Housing	148.00	115	17020.00
779C	900 -lb Brace	1875.00	9	16875.00
2A45C	Ratchet Link	79.00	210	16590.00
41004	Size 4 Widget	117.00	139	16263.00
4100Z	Size 1 Widget	55.00	277	15235.00
887X	Brace Retainer	475.00	32	15200.00
41002	Size 2 Widget	76.00	167	12692.00
887H	Brace Holder	54.00	223	12042.00
775C	500 -lb Brace	1425.00	5	7125.00
XK48A	Reducer	177.00	37	6549.00
887P	Brace Pin	250.00	24	6000.00
XK47	Reducer	355.00	15	5325.00
2A44G	Hinge Pin	350.00	14	4900.00
41003	Handle	652.00	3	1956.00
114	Motor Mount	243.00	5	1215.00
4100X	Widget Adjuster	25.00	37	925.00
41675	Plate	180.00	0	0.00

Q4. How many customers are there?

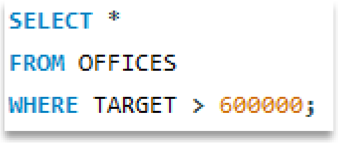


```
SELECT COUNT(*) AS CUSTOMER_COUNT
FROM CUSTOMERS;
```

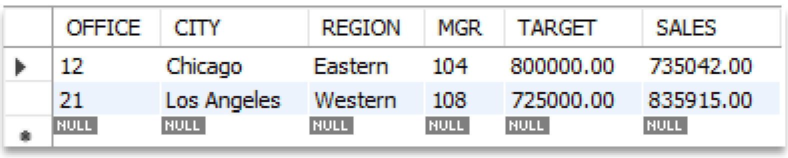


CUSTOMER_COUNT
21

Q5. List the offices with a target over $600,000?

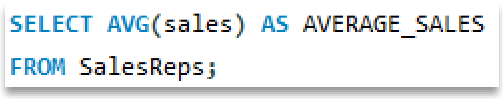


```sql
SELECT *
FROM OFFICES
WHERE TARGET > 600000;
```

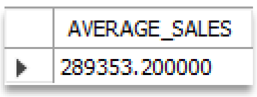


	OFFICE	CITY	REGION	MGR	TARGET	SALES
▶	12	Chicago	Eastern	104	800000.00	735042.00
	21	Los Angeles	Western	108	725000.00	835915.00
*	NULL	NULL	NULL	NULL	NULL	NULL

Q6. What is the average of all the sales people?

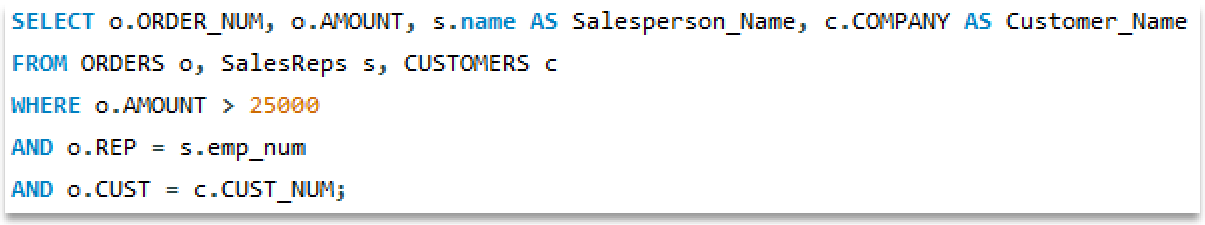


```sql
SELECT AVG(sales) AS AVERAGE_SALES
FROM SalesReps;
```

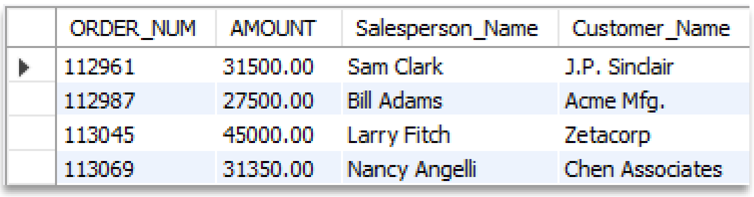


	AVERAGE_SALES
▶	289353.200000

Q7. List orders over $25,000, including the name of the salesperson who took the order and the name of the customer who placed it?

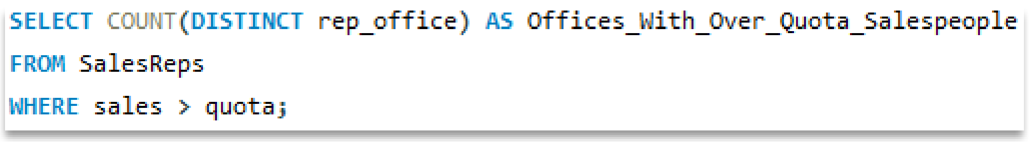


```sql
SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson_Name, c.COMPANY AS Customer_Name
FROM ORDERS o, SalesReps s, CUSTOMERS c
WHERE o.AMOUNT > 25000
AND o.REP = s.emp_num
AND o.CUST = c.CUST_NUM;
```

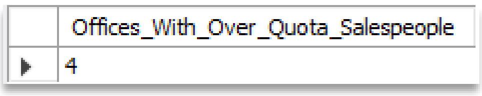


	ORDER_NUM	AMOUNT	Salesperson_Name	Customer_Name
▶	112961	31500.00	Sam Clark	J.P. Sinclair
	112987	27500.00	Bill Adams	Acme Mfg.
	113045	45000.00	Larry Fitch	Zetacorp
	113069	31350.00	Nancy Angelli	Chen Associates

Q8. How many sales offices have salespeople who are over quota?

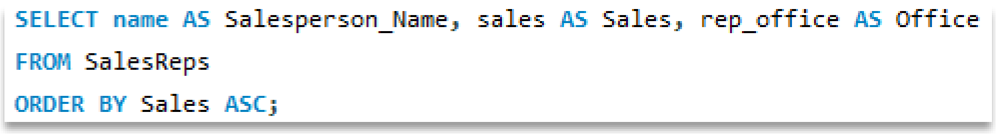


```sql
SELECT COUNT(DISTINCT rep_office) AS Offices_With_Over_Quota_Salespeople
FROM SalesReps
WHERE sales > quota;
```

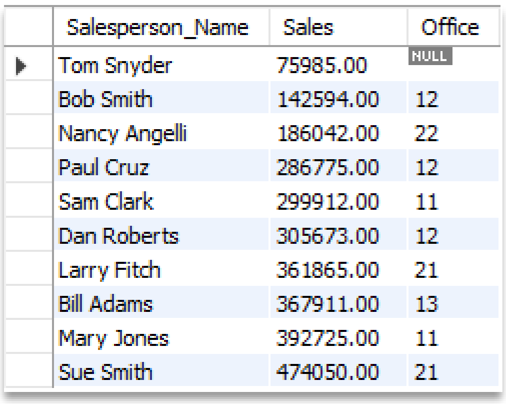


	Offices_With_Over_Quota_Salespeople
▶	4

Q9. Show the name, sales and office for each salesperson. Order by increasing sales?

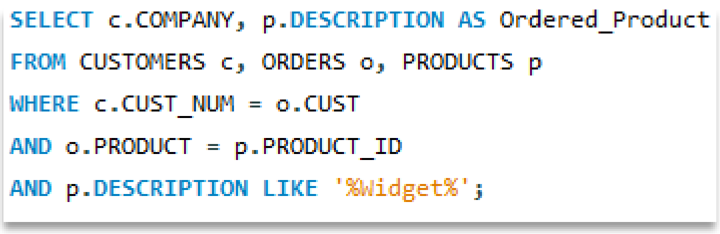


```
SELECT name AS Salesperson_Name, sales AS Sales, rep_office AS Office
FROM SalesReps
ORDER BY Sales ASC;
```

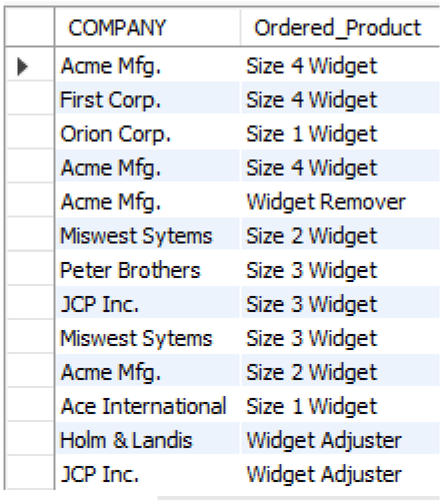


	Salesperson_Name	Sales	Office
▶	Tom Snyder	75985.00	NULL
	Bob Smith	142594.00	12
	Nancy Angelli	186042.00	22
	Paul Cruz	286775.00	12
	Sam Clark	299912.00	11
	Dan Roberts	305673.00	12
	Larry Fitch	361865.00	21
	Bill Adams	367911.00	13
	Mary Jones	392725.00	11
	Sue Smith	474050.00	21

Q10. List all the companies who have ordered any size widget, and the widget they ordered?

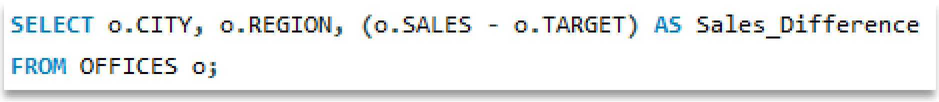


```
SELECT c.COMPANY, p.DESCRIPTION AS Ordered_Product
FROM CUSTOMERS c, ORDERS o, PRODUCTS p
WHERE c.CUST_NUM = o.CUST
AND o.PRODUCT = p.PRODUCT_ID
AND p.DESCRIPTION LIKE '%Widget%';
```

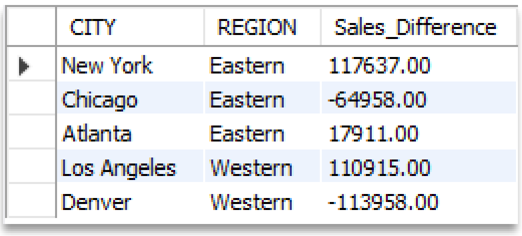


	COMPANY	Ordered_Product
▶	Acme Mfg.	Size 4 Widget
	First Corp.	Size 4 Widget
	Orion Corp.	Size 1 Widget
	Acme Mfg.	Size 4 Widget
	Acme Mfg.	Widget Remover
	Miswest Sytems	Size 2 Widget
	Peter Brothers	Size 3 Widget
	JCP Inc.	Size 3 Widget
	Miswest Sytems	Size 3 Widget
	Acme Mfg.	Size 2 Widget
	Ace International	Size 1 Widget
	Holm & Landis	Widget Adjuster
	JCP Inc.	Widget Adjuster

Q11. List the city, region and amount that sales are over/under target for each office?

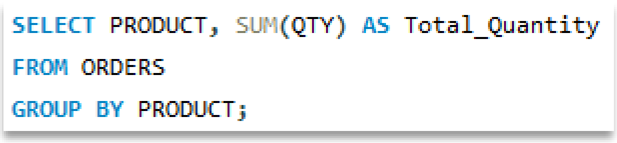


```
SELECT o.CITY, o.REGION, (o.SALES - o.TARGET) AS Sales_Difference
FROM OFFICES o;
```

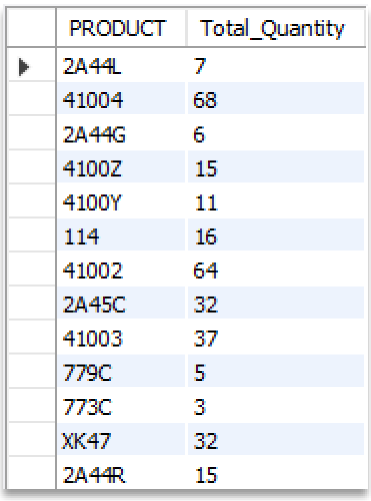


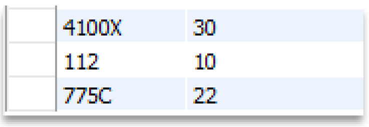
CITY	REGION	Sales_Difference
New York	Eastern	117637.00
Chicago	Eastern	-64958.00
Atlanta	Eastern	17911.00
Los Angeles	Western	110915.00
Denver	Western	-113958.00

Q12. What is the total number of each part that has been ordered?



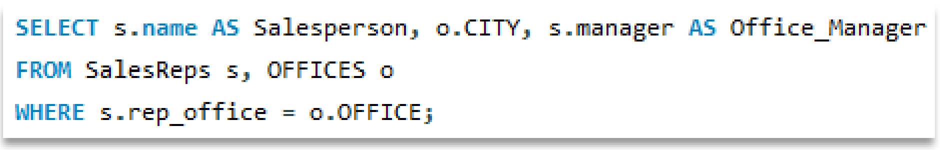
```
SELECT PRODUCT, SUM(QTY) AS Total_Quantity
FROM ORDERS
GROUP BY PRODUCT;
```



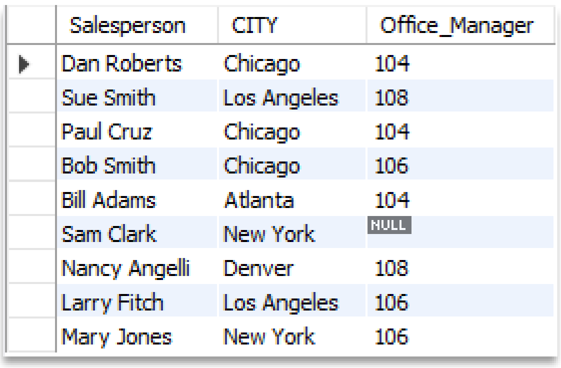


PRODUCT	Total_Quantity
2A44L	7
41004	68
2A44G	6
4100Z	15
4100Y	11
114	16
41002	64
2A45C	32
41003	37
779C	5
773C	3
XK47	32
2A44R	15
4100X	30
112	10
775C	22

Q13. List the salespeople, the city they work in, and the manager of the office in which they work?

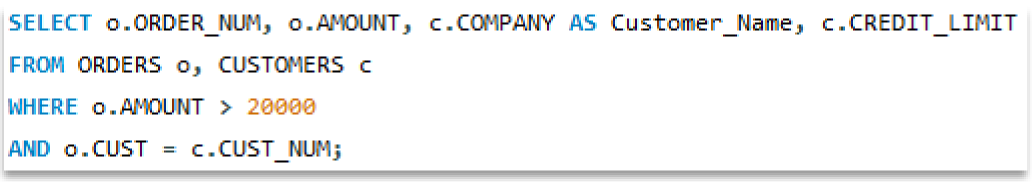


```
SELECT s.name AS Salesperson, o.CITY, s.manager AS Office_Manager
FROM SalesReps s, OFFICES o
WHERE s.rep_office = o.OFFICE;
```

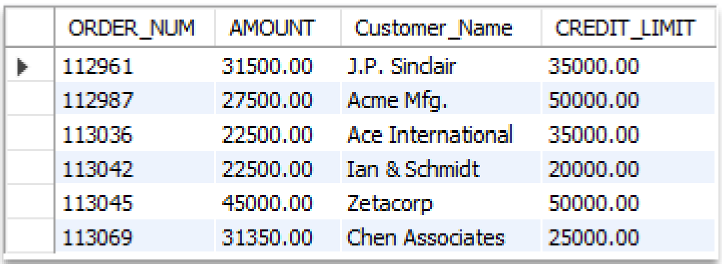


Salesperson	CITY	Office_Manager
Dan Roberts	Chicago	104
Sue Smith	Los Angeles	108
Paul Cruz	Chicago	104
Bob Smith	Chicago	106
Bill Adams	Atlanta	104
Sam Clark	New York	NULL
Nancy Angelli	Denver	108
Larry Fitch	Los Angeles	106
Mary Jones	New York	106

Q14. List all orders showing order number, amount, customer name and the customer's credit limit where the order was greater than $20,000?

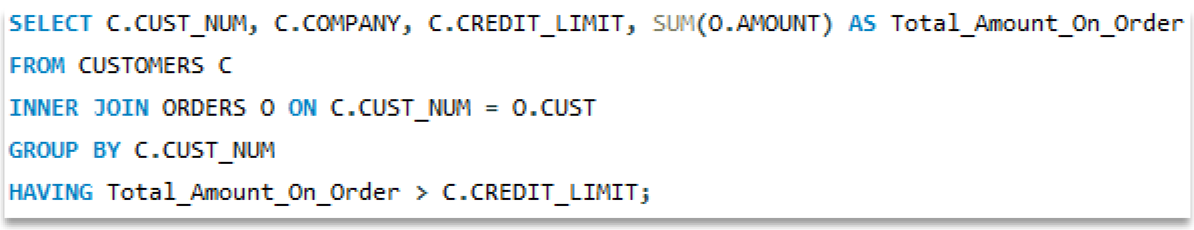


```
SELECT o.ORDER_NUM, o.AMOUNT, c.COMPANY AS Customer_Name, c.CREDIT_LIMIT
FROM ORDERS o, CUSTOMERS c
WHERE o.AMOUNT > 20000
AND o.CUST = c.CUST_NUM;
```

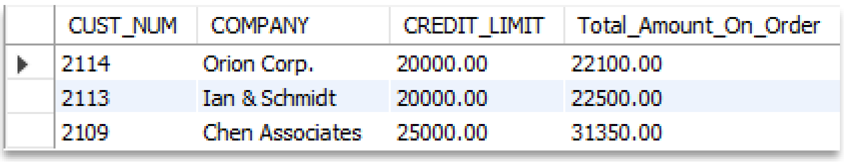


ORDER_NUM	AMOUNT	Customer_Name	CREDIT_LIMIT
112961	31500.00	J.P. Sinclair	35000.00
112987	27500.00	Acme Mfg.	50000.00
113036	22500.00	Ace International	35000.00
113042	22500.00	Ian & Schmidt	20000.00
113045	45000.00	Zetacorp	50000.00
113069	31350.00	Chen Associates	25000.00

Q15. Are there any customers who are over their credit limit? If so, list the customer, the total amount the customer has on order, and the credit limit?

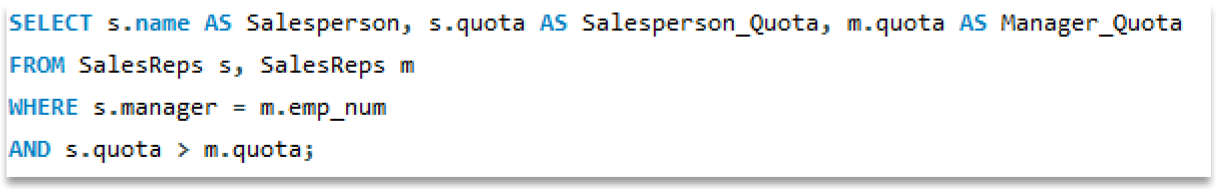


```
SELECT C.CUST_NUM, C.COMPANY, C.CREDIT_LIMIT, SUM(O.AMOUNT) AS Total_Amount_On_Order
FROM CUSTOMERS C
INNER JOIN ORDERS O ON C.CUST_NUM = O.CUST
GROUP BY C.CUST_NUM
HAVING Total_Amount_On_Order > C.CREDIT_LIMIT;
```

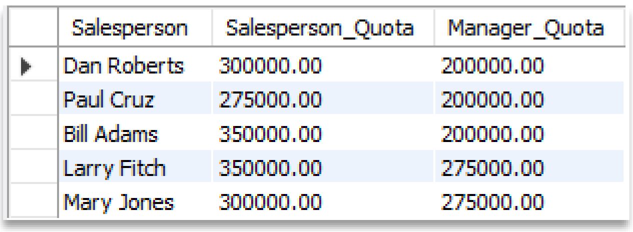


CUST_NUM	COMPANY	CREDIT_LIMIT	Total_Amount_On_Order
2114	Orion Corp.	20000.00	22100.00
2113	Ian & Schmidt	20000.00	22500.00
2109	Chen Associates	25000.00	31350.00

Q16. List the salespeople with a higher quota than their manager?

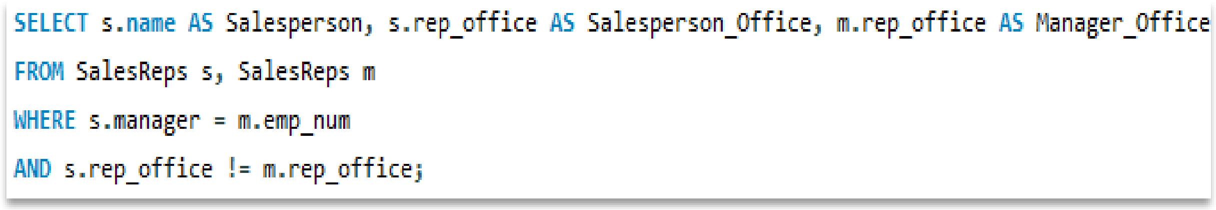


```
SELECT s.name AS Salesperson, s.quota AS Salesperson_Quota, m.quota AS Manager_Quota
FROM SalesReps s, SalesReps m
WHERE s.manager = m.emp_num
AND s.quota > m.quota;
```

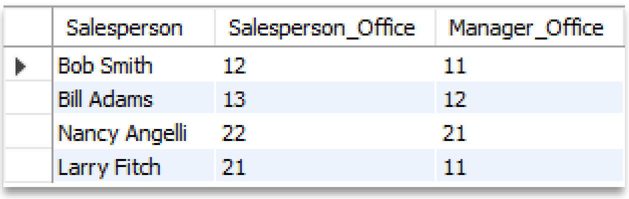


Salesperson	Salesperson_Quota	Manager_Quota
Dan Roberts	300000.00	200000.00
Paul Cruz	275000.00	200000.00
Bill Adams	350000.00	200000.00
Larry Fitch	350000.00	275000.00
Mary Jones	300000.00	275000.00

Q17. List salespeople who work in different offices than their managers, show the name and office where each work?

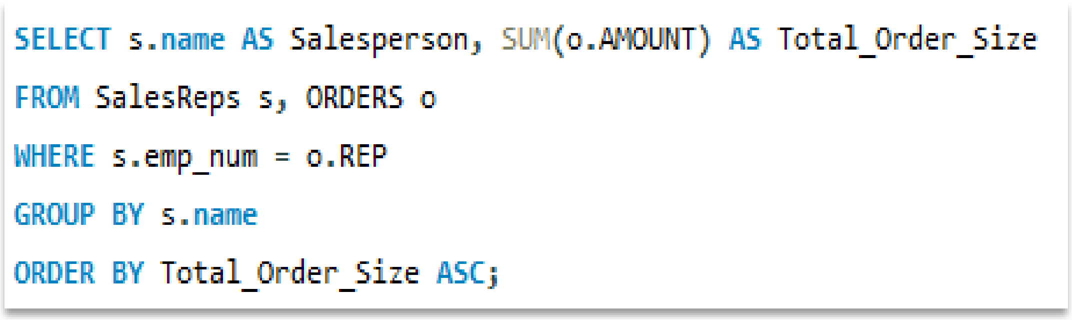


```
SELECT s.name AS Salesperson, s.rep_office AS Salesperson_Office, m.rep_office AS Manager_Office
FROM SalesReps s, SalesReps m
WHERE s.manager = m.emp_num
AND s.rep_office != m.rep_office;
```

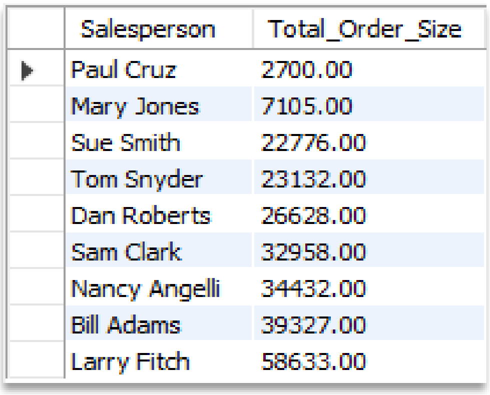


Salesperson	Salesperson_Office	Manager_Office
Bob Smith	12	11
Bill Adams	13	12
Nancy Angelli	22	21
Larry Fitch	21	11

Q18. What is the total order size for each salesperson? Order by increasing sales?

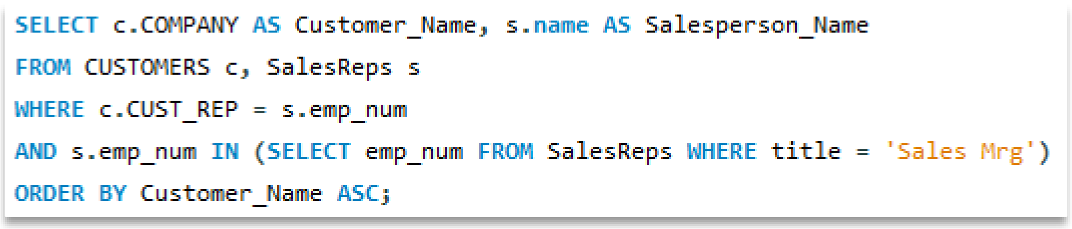


```
SELECT s.name AS Salesperson, SUM(o.AMOUNT) AS Total_Order_Size
FROM SalesReps s, ORDERS o
WHERE s.emp_num = o.REP
GROUP BY s.name
ORDER BY Total_Order_Size ASC;
```

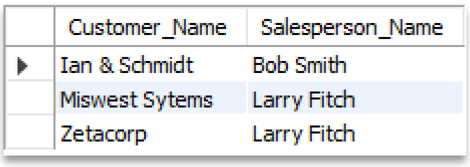


Salesperson	Total_Order_Size
Paul Cruz	2700.00
Mary Jones	7105.00
Sue Smith	22776.00
Tom Snyder	23132.00
Dan Roberts	26628.00
Sam Clark	32958.00
Nancy Angelli	34432.00
Bill Adams	39327.00
Larry Fitch	58633.00

Q19. List all the customers whose sales representative is a manager. Arrange increasing by company?

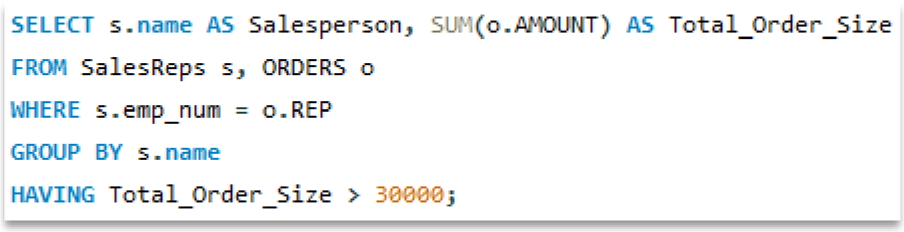


```
SELECT c.COMPANY AS Customer_Name, s.name AS Salesperson_Name
FROM CUSTOMERS c, SalesReps s
WHERE c.CUST_REP = s.emp_num
AND s.emp_num IN (SELECT emp_num FROM SalesReps WHERE title = 'Sales Mrg')
ORDER BY Customer_Name ASC;
```

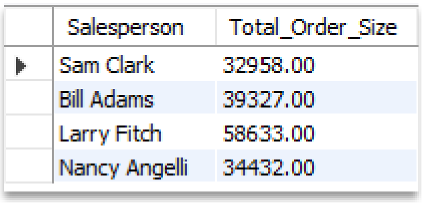


Customer_Name	Salesperson_Name
Ian & Schmidt	Bob Smith
Miswest Sytems	Larry Fitch
Zetacorp	Larry Fitch

Q20. What is the total order size for each salesperson whose orders total more than $30,000?

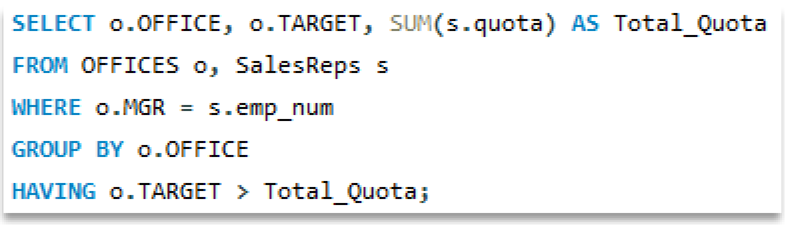


```sql
SELECT s.name AS Salesperson, SUM(o.AMOUNT) AS Total_Order_Size
FROM SalesReps s, ORDERS o
WHERE s.emp_num = o.REP
GROUP BY s.name
HAVING Total_Order_Size > 30000;
```

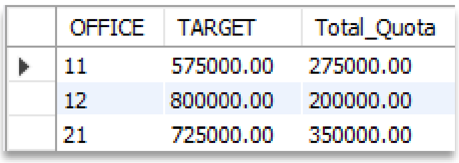


Salesperson	Total_Order_Size
Sam Clark	32958.00
Bill Adams	39327.00
Larry Fitch	58633.00
Nancy Angelli	34432.00

Q21. List the offices where the sales target for the office exceeds the sum of the individual sales people's quotas?

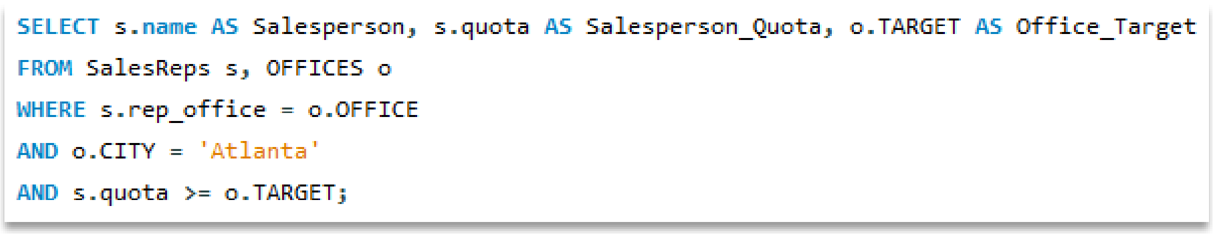


```sql
SELECT o.OFFICE, o.TARGET, SUM(s.quota) AS Total_Quota
FROM OFFICES o, SalesReps s
WHERE o.MGR = s.emp_num
GROUP BY o.OFFICE
HAVING o.TARGET > Total_Quota;
```

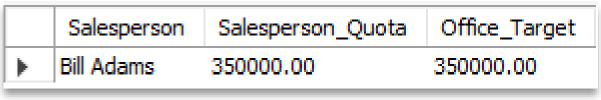


OFFICE	TARGET	Total_Quota
11	575000.00	275000.00
12	800000.00	200000.00
21	725000.00	350000.00

Q22. List the salespeople whose quotas are equal to or higher than the target of the Atlanta sales office?

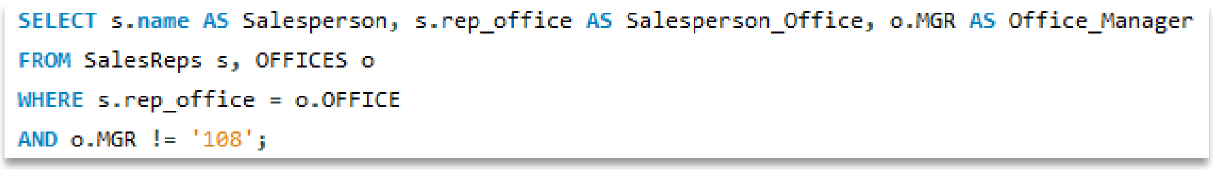


```sql
SELECT s.name AS Salesperson, s.quota AS Salesperson_Quota, o.TARGET AS Office_Target
FROM SalesReps s, OFFICES o
WHERE s.rep_office = o.OFFICE
AND o.CITY = 'Atlanta'
AND s.quota >= o.TARGET;
```

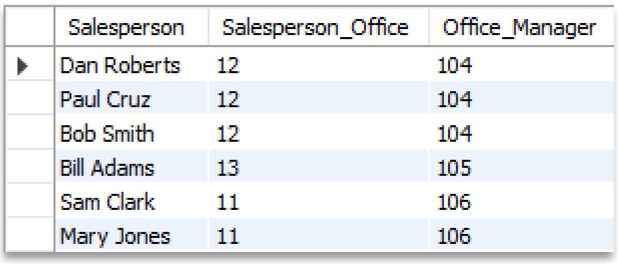


	Salesperson	Salesperson_Quota	Office_Target
▶	Bill Adams	350000.00	350000.00

Q23. List the salespeople who do not work in offices managed by Larry Fitch (employee 108)?

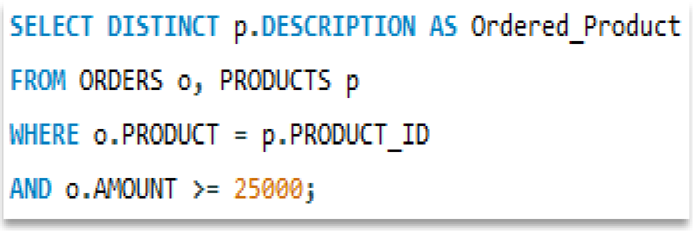


```
SELECT s.name AS Salesperson, s.rep_office AS Salesperson_Office, o.MGR AS Office_Manager
FROM SalesReps s, OFFICES o
WHERE s.rep_office = o.OFFICE
AND o.MGR != '108';
```

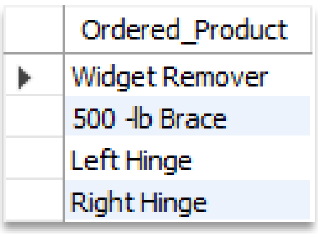


	Salesperson	Salesperson_Office	Office_Manager
▶	Dan Roberts	12	104
	Paul Cruz	12	104
	Bob Smith	12	104
	Bill Adams	13	105
	Sam Clark	11	106
	Mary Jones	11	106

Q24. List the products for which an order of $25,000 or more has been received?

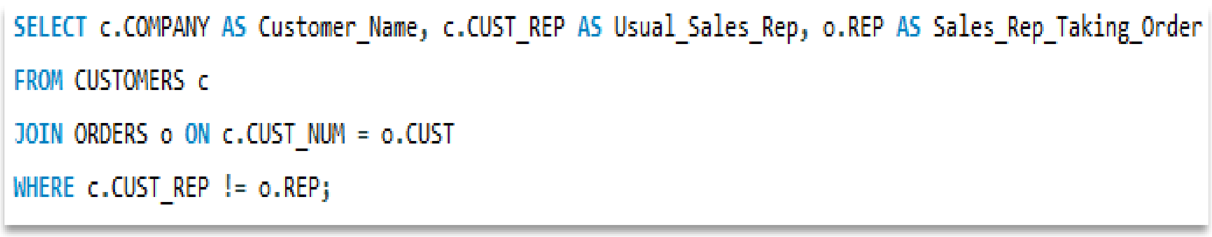


```
SELECT DISTINCT p.DESCRIPTION AS Ordered_Product
FROM ORDERS o, PRODUCTS p
WHERE o.PRODUCT = p.PRODUCT_ID
AND o.AMOUNT >= 25000;
```

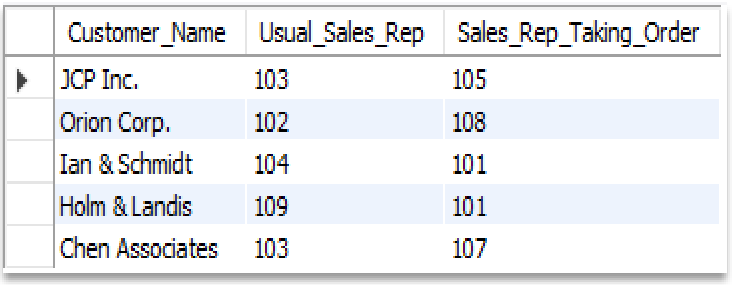


	Ordered_Product
▶	Widget Remover
	500 -lb Brace
	Left Hinge
	Right Hinge

Q25. List the companies who placed an order with a sales rep that is not the sales rep that usually calls on them?



```
SELECT c.COMPANY AS Customer_Name, c.CUST_REP AS Usual_Sales_Rep, o.REP AS Sales_Rep_Taking_Order
FROM CUSTOMERS c
JOIN ORDERS o ON c.CUST_NUM = o.CUST
WHERE c.CUST_REP != o.REP;
```



Customer_Name	Usual_Sales_Rep	Sales_Rep_Taking_Order
JCP Inc.	103	105
Orion Corp.	102	108
Ian & Schmidt	104	101
Holm & Landis	109	101
Chen Associates	103	107

Assignment 3: Database Design Assignment

Suppose you are given the following business rules and first attempt at a DB design to form the basis for your database design of a problem. The database must enable the manager of a company dinner club to mail invitations to the club's members, to plan the meals, to keep track of who attends the dinners, what dinners were served when and so on.

- Every member receives an invitation to each dinner.
- Members must respond to the invitation if they intend to attend the dinner.
- A dinner is based on a single entree, but an entree may be used as the basis for many dinners. For example, a dinner might be composed of a fish entree, rice and corn. Or the dinner might be composed of a fish entree, baked potato and string beans.

Because the manager is not a database expert, the first attempt at creating the database uses the following structure (all the attributes in one relation):

Attribute name	Sample value
Member_num	214
Member_name	Alice B.VanderVoort
Member_address	325 Meadow Park
Member_city	Murkywater
Member_zip	59759
Invite_num	8
Invite_date	8/1/97
Accept_date	8/9/97
Dinner_date	8/23/97
Dinner_attend	Y
Dinner_code	5
Dinner_desc	Sea Delight
Entrée_code	3
Entrée_desc	Stuffed Crab
Dessert_code	8
Dessert_desc	Chocolate mousse with Rasberry sauce

You have two tools to design a database, the ER diagram and normalization from a universal relation. Use both tools as specified below. Hopefully you will come up with the same set of tables.

Part I -- the ER Diagram

1. Draw the ER diagram.

- Remember your first pass at designing this way is to include all the attributes of each entity, then find the relationships by attributes that refer to other entities.
- Be sure to include whether the relationships are partial or total, and the cardinality ratio.

2. Map your ER diagram to a relational schema (tables).

Part II -- Normalization of the Universal Relation

1. Given the above structure, draw its dependency diagram. Label all transitive and/or partial dependencies.

2. Normalize the diagram above to produce dependency diagrams in are in 3NF.

Introduction

This assignment details the design process for a database system to manage a company's dinner club. The system will enable the club manager to:

- Send invitations to members
- Plan meals, including entrees and optionally desserts
- Track member attendance at dinners

Part I -- the ER Diagram

Entity: Member Table

Attributes:

- Member_num (Primary Key)

- Member_name
- Member_address
- Member_city
- Member_zip

Entity: Invitation Table

Attributes:

- Invite_num (Primary Key)
- Invite_date
- Member_num (Foreign Key referencing Member Table)

Entity: Entree Table

Attributes:

- Entree_code (Primary Key)
- Entree_desc

Entity: Dinner Table

Attributes:

- Dinner_code (Primary Key)
- Dinner_date
- Dinner_desc
- Entree_code (Foreign Key referencing Entree Table)

Entity: Dessert Table

Attributes:

- Dessert_code (Primary Key)
- Dessert_desc

Relationship:

Member □ Invitation:

- Each record in the Member table represents a unique member.

- Each record in the Invitation table represents a specific invitation sent to a member.
- The Member_num attribute in the Invitation table is a foreign key that references the Member_num attribute in the Member table.

This shows that Member □ Invitation has one-to-many relationship, where one member can have multiple invitations associated with them in the Invitation table.

<u>Entree□ Dinner:</u>

- Each record in the Dinner table represents a unique dinner event.
- Each record in the Entree table represents a specific entree that can be served at dinners.
- The Entree_code attribute in the Dinner table serves as a foreign key that references the Entree_code attribute in the Entree table.

This shows that Entrée□ Dinner has one-to-many relationship where each dinner can have one entree associated with it in the Entree table, and each entree can be served at multiple dinners.

<u>Dinner□ Dessert</u>:

- Each record in the Dinner table represents a unique dinner event.
- Each record in the Dessert table represents a specific dessert that can be served at dinners.
- The Dessert_code attribute in the Dinner table serves as a foreign key that references the Dessert_code attribute in the Dessert table.

This shows that Dinner□ Dessert has one-to-one relationship where each dinner event can have only one dessert associated with it in the Dessert table, and each dessert can be served at only one dinner event.

QUERY:

```
CREATE DATABASE companydinner;
USE companydinner;
CREATE TABLE member (
member_num INT PRIMARY KEY,
member_name VARCHAR(20),
member_address VARCHAR(20),
member_city VARCHAR(20),
member_zip INT
);
CREATE TABLE invitation (
invite_num INT PRIMARY KEY,
invite_date DATE,
member_num INT,
FOREIGN KEY (member_num) REFERENCES member(member_num)
);
CREATE TABLE Entree (
entree_code INT PRIMARY KEY,
entree_desc VARCHAR(20),
);
CREATE TABLE dinner (
dinner_code INT PRIMARY KEY,
dinner_date DATE,
dinner_desc VARCHAR(20),
entrée_code INT,
FOREIGN KEY (entrée_code) REFERENCES Entree(entrée_code)
);
CREATE TABLE Dessert (
dessert_code INT PRIMARY KEY,
dessert_desc VARCHAR(20), );
```

ER Diagram:

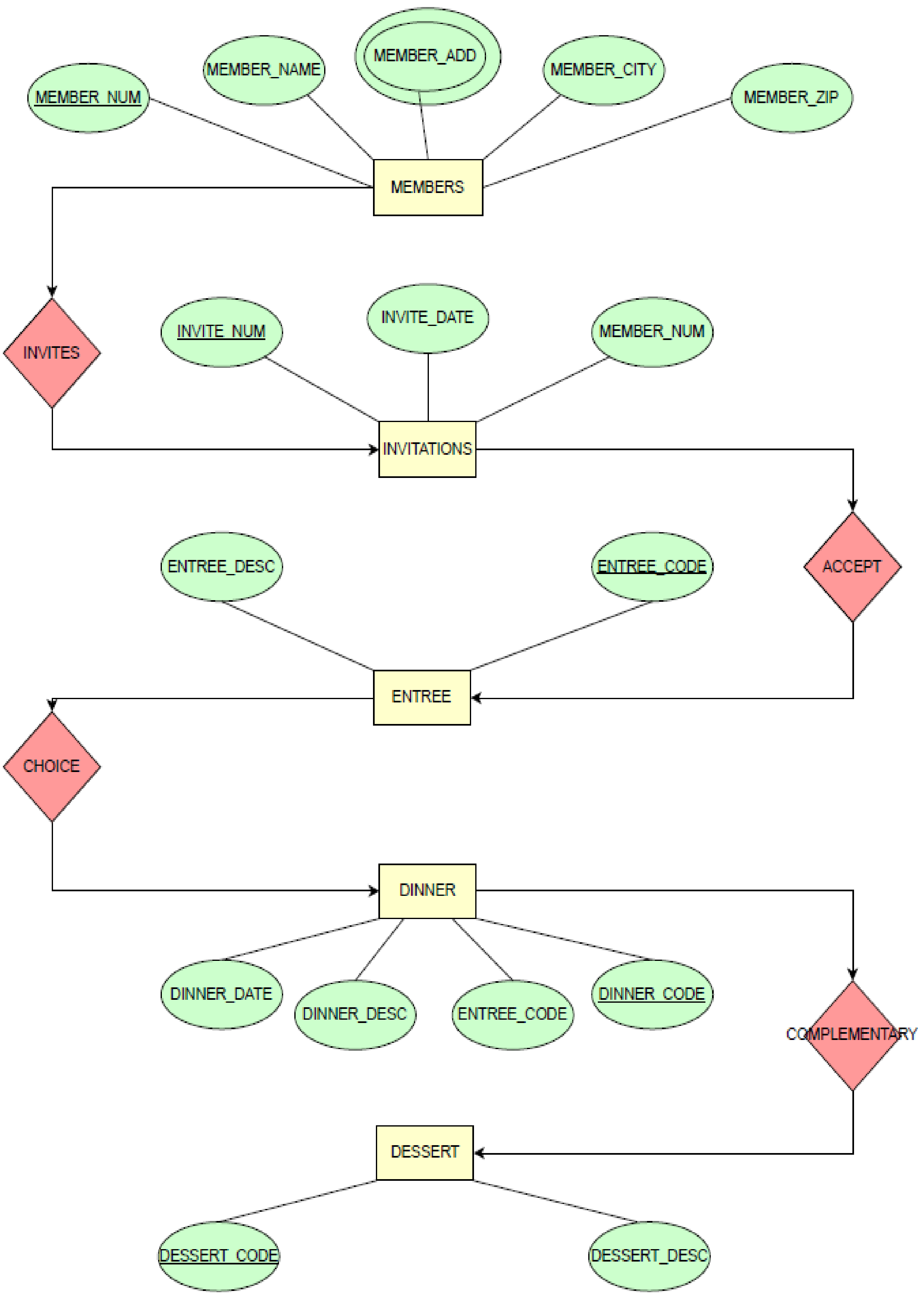


FIG 1: ER Diagram

Part II -- Normalization of the Universal Relation

Normalization in Database Management Systems (DBMS) is a systematic process of organizing data in a database to minimize redundancy and improve data integrity. The primary goals of normalization are to eliminate duplicate data, ensure data dependencies make sense, and simplify the structure of the tables to enhance data management and reduce the potential for anomalies.

1. Dependency Diagram

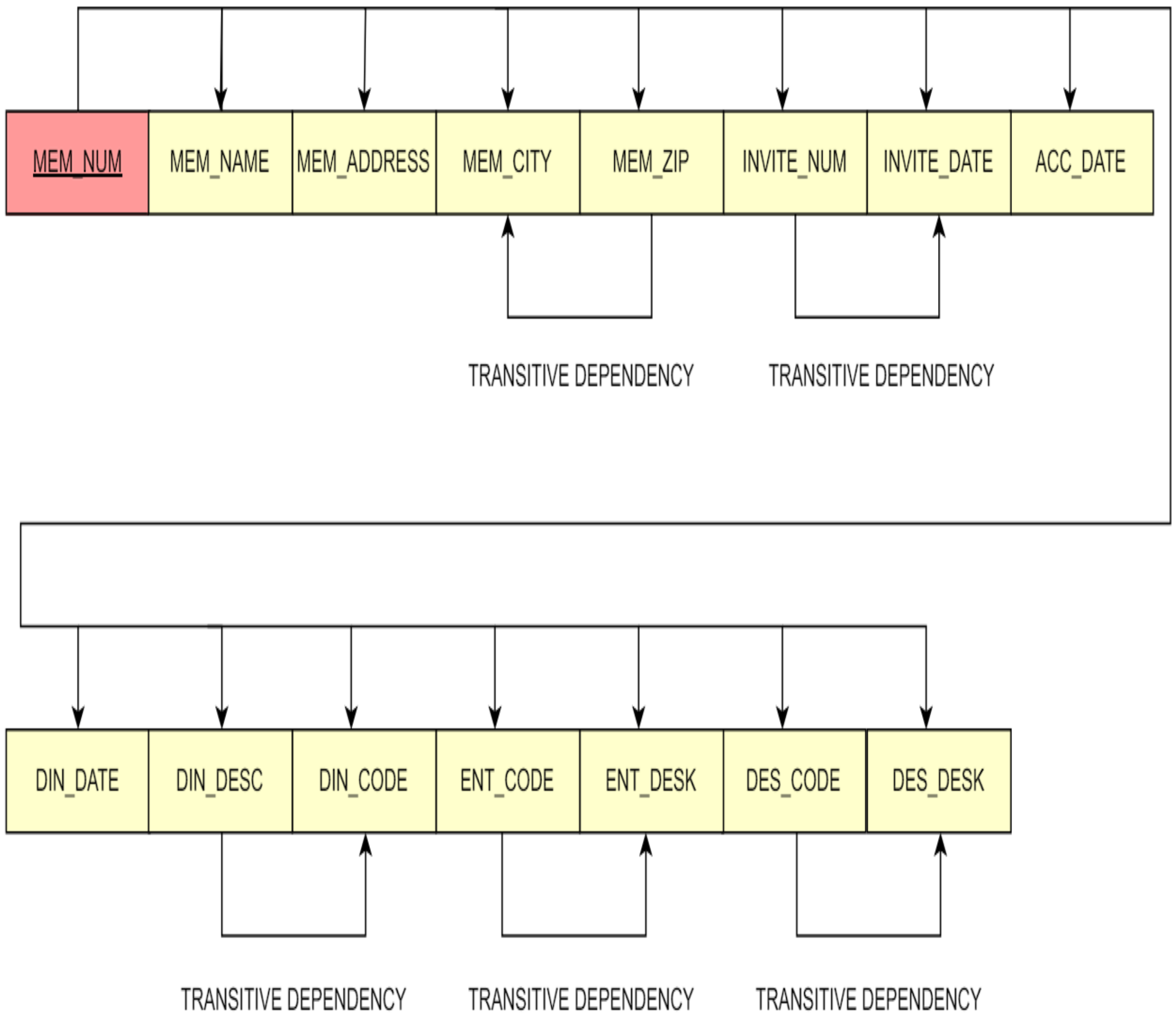


FIG 2: Dependency Diagram

Normalize the diagram above to produce dependency diagrams in are in 3NF.

A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3F).

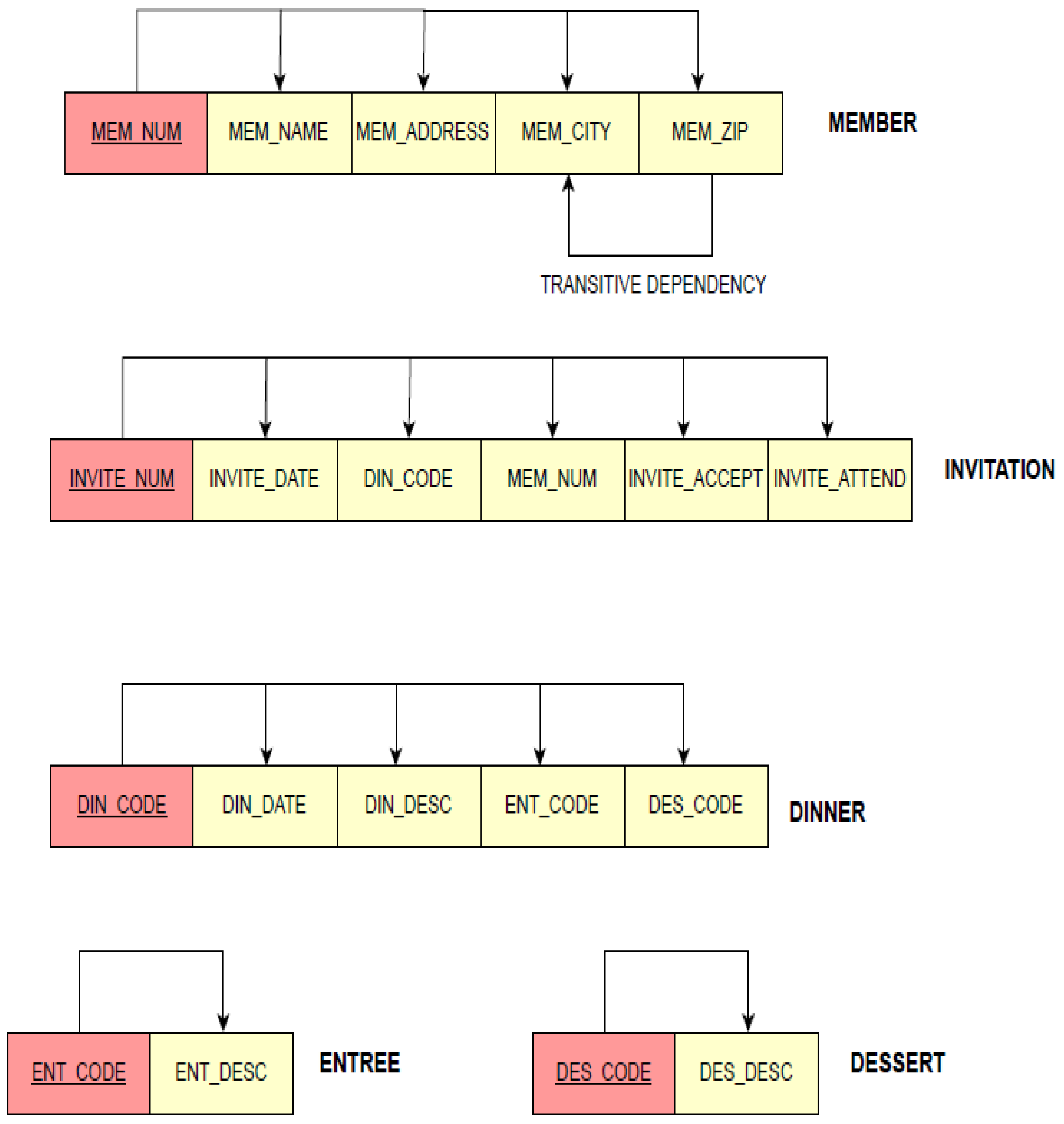


Fig 3: Normalize

For instance, the (composite) INVITATION and DINNER entities allow tracking of who was sent an invitation on what date (INVITE_DATE), the date of the dinner event (DIN_DATE), the specific dinner (DIN_CODE) to be served, who (MEM_NUM) accepted the invitation (INVITE_ACCEPT), and who actually attended (INVITE_ATTEND). The INVITE_ACCEPT and INVITE_ATTEND attributes can be simple Y/N values, with default values set to N to avoid nulls. This setup simplifies obtaining the number of acceptances for a given dinner by a certain date, thereby aiding the catering service in better planning the event.

The relational schemas are as follows:

- MEMBER(MEM_NUM, MEM_NAME, MEM_ADDRESS, MEM_CITY, MEM_ZIP)
- INVITATION(INVITE_NUM, INVITE_DATE, DIN_CODE, MEM_NUM, INVITE_ACCEPT, INVITE_ATTEND)
- ENTRÉE(ENT_CODE, ENT_DESCRIPTION)
- DINNER(DIN_CODE, DIN_DATE, DIN_DESCRIPTION, ENT_CODE, DES_CODE)
- DESSERT(DES_CODE, DES_DESCRIPTION)

Conclusion:

Design illustrated in Fig:3 demonstrates the effectiveness of the database decomposition. The composite INVITATION and DINNER entities provide a clear framework for tracking invitation details, including the invite date (INVITE_DATE), dinner date (DIN_DATE), specific dinner code (DIN_CODE), member number (MEM_NUM), acceptance status (INVITE_ACCEPT), and attendance status (INVITE_ATTEND). By setting default values for INVITE_ACCEPT and INVITE_ATTEND to 'N', the design avoids null values, thereby simplifying the process of tracking acceptances and attendance. This streamlined approach facilitates better planning for catering services by making it easy to determine the number of acceptances for any given dinner.