

ASSIGNMENT

By

Diya Bhat (2022a1r017)

Shoieb Ali (2022a1r014)

Ishika Razdan (2022a1r016)

Prachi Khajuria (2022a1r015)

4th Semester

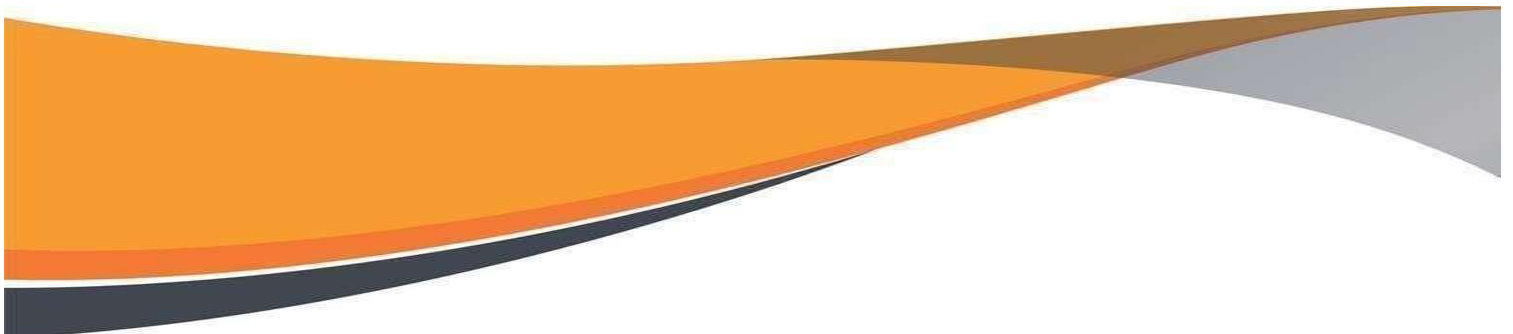
CSE Department



Model Institute of Engineering & Technology (Autonomous)

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

Jammu, India 2023



ASSIGNMENT**GROUP - 4****Subject Code:** Relational Database Management System**Due Date:** 24-05-24

Question Number	Course Outcomes	Blooms' Level	Maximum Marks	Marks Obtain
Q1, Q2, Q3	CO 3, CO 4, CO 5	3-6	20	
Total Marks			20	
Faculty Signature: Mr. Navin M Upadhyay Email: navin.cse@mietjammu.in				

Assignment 1 -- SQL as a Data Manipulation Language Using MySQL:

1. Create the tables for the Company database in your text, and populate them with data.

ANS 1: CODE :-

```
create database hnji; use hnji;
-- Create Departments Table without ManagerID foreign key CREATE TABLE Departments (
DepartmentID INT PRIMARY KEY AUTO_INCREMENT,
DepartmentName VARCHAR(100) NOT NULL
);
-- Insert data into Departments Table
INSERT INTO Departments (DepartmentName) VALUES ('Human Resources'),
('Finance'),
('Engineering');
select * from Departments;

-- Create Employees Table CREATE TABLE Employees (
EmployeeID INT PRIMARY KEY AUTO_INCREMENT, FirstName VARCHAR(50) NOT NULL,
LastName VARCHAR(50) NOT NULL,
Email VARCHAR(100) NOT NULL UNIQUE, Phone VARCHAR(15),
HireDate DATE, DepartmentID INT,
FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
-- Insert data into Employees Table
INSERT INTO Employees (FirstName, LastName, Email, Phone, HireDate, DepartmentID) VALUES
('John', 'Doe', 'john.doe@example.com', '555-1234', '2020-01-15', 1),
('Jane', 'Smith', 'jane.smith@example.com', '555-5678', '2019-03-22', 2),
('Michael', 'Brown', 'michael.brown@example.com', '555-8765', '2018-07-11', 3),
('Emily', 'Davis', 'emily.davis@example.com', '555-4321', '2021-10-05', 1); select * from Employeess;

-- Create Projects Table CREATE TABLE Projects (
ProjectID INT PRIMARY KEY AUTO_INCREMENT, ProjectName VARCHAR(100) NOT NULL,
StartDate DATE,
```

EndDate DATE

);

-- Insert data into Projects Table

```
INSERT INTO Projects (ProjectName, StartDate, EndDate) VALUES ('Project Alpha', '2023-01-01', '2023-12-31'),
```

```
('Project Beta', '2023-02-01', '2023-11-30'),
```

```
('Project Gamma', '2023-03-01', '2023-10-31');
```

```
select * from projects;
```

-- Create Salaries Table CREATE TABLE Salaries (

EmployeeID INT PRIMARY KEY,

SalaryAmount DECIMAL(10, 2) NOT NULL, EffectiveDate DATE NOT NULL,

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)

);

-- Insert data into Salaries Table

```
INSERT INTO Salaries (EmployeeID, SalaryAmount, EffectiveDate) VALUES (1, 60000.00, '2023-01-01'),
```

```
(2, 70000.00, '2023-01-01'),
```

```
(3, 80000.00, '2023-01-01'),
```

```
(4, 50000.00, '2023-01-01');
```

```
select * from salaries;
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The main area displays a list of SQL queries with line numbers 1 through 13. The queries are as follows:

```
1 create database hnji;
2 use hnji;
3 -- Create Departments Table without ManagerID foreign key
4 CREATE TABLE Departments (
5     DepartmentID INT PRIMARY KEY AUTO_INCREMENT,
6     DepartmentName VARCHAR(100) NOT NULL
7 );
8 -- Insert data into Departments Table
9 INSERT INTO Departments (DepartmentName) VALUES
10 ('Human Resources'),
11 ('Finance'),
12 ('Engineering');
13 select * from Departments;
```

Below the queries, the 'Result Grid' is visible, showing the output of the last query. It has a 'Filter Rows' field and buttons for 'Edit', 'Export/Import', and 'Wrap Cell Cor'. The result grid contains the following data:

DepartmentID	DepartmentName
1	Human Resources
2	Finance
3	Engineering
4	Human Resources
5	Finance
6	Engineering
7	Human Resources
8	Finance
9	Engineering

Limit to 1000 rows

```
14 -- Create Employees Table
15 CREATE TABLE Employees (
25 -- Insert data into Employees Table
26 INSERT INTO Employees (FirstName, LastName, Email, Phone, HireDate, DepartmentID) VALUES
27 ('John', 'Doe', 'john.doe@example.com', '555-1234', '2020-01-15', 1),
28 ('Jane', 'Smith', 'jane.smith@example.com', '555-5678', '2019-03-22', 2),
29 ('Michael', 'Brown', 'michael.brown@example.com', '555-8765', '2018-07-11', 3),
30 ('Emily', 'Davis', 'emily.davis@example.com', '555-4321', '2021-10-05', 1);
31
32 select * from Employees;
33 -- Create Projects Table
```

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

	EmployeeID	FirstName	LastName	Email	Phone	HireDate	DepartmentID
▶	1	John	Doe	john.doe@example.com	555-1234	2020-01-15	1
	2	Jane	Smith	jane.smith@example.com	555-5678	2019-03-22	2
	3	Michael	Brown	michael.brown@example.com	555-8765	2018-07-11	3
	4	Emily	Davis	emily.davis@example.com	555-4321	2021-10-05	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid Form Editor

Limit to 1000 rows

```
46 -- Create Salaries Table
47 CREATE TABLE Salaries (
48     EmployeeID INT PRIMARY KEY,
49     SalaryAmount DECIMAL(10, 2) NOT NULL,
50     EffectiveDate DATE NOT NULL,
51     FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
52 );
53 -- Insert data into Salaries Table
54 INSERT INTO Salaries (EmployeeID, SalaryAmount, EffectiveDate) VALUES
55 (1, 60000.00, '2023-01-01'),
56 (2, 70000.00, '2023-01-01'),
57 (3, 80000.00, '2023-01-01'),
58 (4, 50000.00, '2023-01-01');
59 select * from salaries;
60
```

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

	EmployeeID	SalaryAmount	EffectiveDate
▶	1	60000.00	2023-01-01
	2	70000.00	2023-01-01
	3	80000.00	2023-01-01
	4	50000.00	2023-01-01
*	NULL	NULL	NULL

Result Grid Form Editor

Limit to 1000 rows

34

CREATE TABLE Projects (
35 ProjectID INT PRIMARY KEY AUTO_INCREMENT,
36 ProjectName VARCHAR(100) NOT NULL,
37 StartDate DATE,
38 EndDate DATE
39);
40 -- Insert data into Projects Table
41 INSERT INTO Projects (ProjectName, StartDate, EndDate) VALUES
42 ('Project Alpha', '2023-01-01', '2023-12-31'),
43 ('Project Beta', '2023-02-01', '2023-11-30'),
44 ('Project Gamma', '2023-03-01', '2023-10-31');
45 select * from projects;
46 -- Create Salaries Table

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	ProjectID	ProjectName	StartDate	EndDate
▶	1	Project Alpha	2023-01-01	2023-12-31
	2	Project Beta	2023-02-01	2023-11-30
	3	Project Gamma	2023-03-01	2023-10-31
•	NULL	NULL	NULL	NULL

Result Grid

Form Editor

2. Create a simple desktop app to load, add, and delete the data from the database. [Use any language Python tk, c#, .net, etc.]

ANS 2: -

Code for creating GUI Application :

```
import tkinter as tk
from tkinter import messagebox
import sqlite3

# Function to connect to the Company database
def connect_to_database():
    conn = sqlite3.connect('company.db')
    c = conn.cursor()
    return conn, c

# Function to create the Department table if not exists
def create_department_table():
    conn, c = connect_to_database()
    c.execute("""CREATE TABLE IF NOT EXISTS Department (
        id INTEGER PRIMARY KEY,
        name TEXT)""")
    conn.commit()
    conn.close()

# Function to create the Employee table if not exists
def create_employee_table():
    conn, c = connect_to_database()
    c.execute("""CREATE TABLE IF NOT EXISTS Employee (
        id INTEGER PRIMARY KEY,
        name TEXT,
        department_id INTEGER,
        FOREIGN KEY (department_id) REFERENCES Department(id))""")
    conn.commit()
    conn.close()

# Function to populate the Department table with sample data
def populate_department_table():
```

```

conn, c = connect_to_database()
c.execute("INSERT INTO Department (name) VALUES ('HR')")
c.execute("INSERT INTO Department (name) VALUES ('Finance')")
c.execute("INSERT INTO Department (name) VALUES ('IT')")
conn.commit()
conn.close()

# Function to populate the Employee table with sample data
def populate_employee_table():
    conn, c = connect_to_database()
    c.execute("INSERT INTO Employee (name, department_id) VALUES ('John Doe', 1)")
    c.execute("INSERT INTO Employee (name, department_id) VALUES ('Jane Smith', 2)")
    c.execute("INSERT INTO Employee (name, department_id) VALUES ('Mike Johnson', 3)")
    conn.commit()
    conn.close()

# Function to load all data from the Employee table
def load_employee_data():
    conn, c = connect_to_database()
    c.execute("SELECT Employee.id, Employee.name, Department.name FROM Employee INNER
JOIN Department ON Employee.department_id = Department.id")
    rows = c.fetchall()
    conn.close()
    return rows

# Function to add data to the Employee table
def add_employee_data(name, department_id):
    conn, c = connect_to_database()
    c.execute("INSERT INTO Employee (name, department_id) VALUES (?, ?)", (name,
department_id))
    conn.commit()
    conn.close()

# Function to add department to the Department table
def add_department_data(department_name):
    conn, c = connect_to_database()
    c.execute("INSERT INTO Department (name) VALUES (?)", (department_name,))
    conn.commit()
    conn.close()

# Function to delete data from the Employee table
def delete_employee_data(id):
    conn, c = connect_to_database()
    c.execute("DELETE FROM Employee WHERE id=?", (id,))
    conn.commit()
    conn.close()

# Main tkinter window
root = tk.Tk()
root.title("Company Database App")

# Create Department and Employee tables if not exists
create_department_table()
create_employee_table()

```



```

populate_department_table()
populate_employee_table()

# Listbox to display employee data
listbox_employee = tk.Listbox(root, width=50)
listbox_employee.grid(row=0, column=0, columnspan=2, padx=10, pady=10)

# Function to update the listbox with employee data
def update_employee_listbox():
    listbox_employee.delete(0, tk.END)
    for row in load_employee_data():
        listbox_employee.insert(tk.END, row)

# Update listbox with existing employee data
update_employee_listbox()

# Entry widgets for adding employee data
entry_name = tk.Entry(root, width=30)
entry_name.grid(row=1, column=0, padx=10, pady=5)

entry_department_id = tk.Entry(root, width=10)
entry_department_id.grid(row=1, column=1, padx=5, pady=5)

# Entry widget for adding department data
entry_department_name = tk.Entry(root, width=30)
entry_department_name.grid(row=2, column=0, columnspan=2, padx=10, pady=5)

# Function to handle the 'Add Employee' button click
def add_employee_button_click():
    name = entry_name.get()
    department_id = entry_department_id.get()
    if name and department_id:
        add_employee_data(name, department_id)
        entry_name.delete(0, tk.END)
        entry_department_id.delete(0, tk.END)
        update_employee_listbox()
    else:
        messagebox.showwarning("Warning", "Please enter name and department ID.")

# Function to handle the 'Add Department' button click
def add_department_button_click():
    department_name = entry_department_name.get()
    if department_name:
        add_department_data(department_name)
        entry_department_name.delete(0, tk.END)
        update_employee_listbox() # Update listbox to reflect changes in department
    else:
        messagebox.showwarning("Warning", "Please enter a department name.")

def delete_button_click():
    try:
        selected_index = listbox_employee.curselection()[0]
        selected_id = listbox_employee.get(selected_index)[0]
        delete_employee_data(selected_id)
        update_employee_listbox()

```

```

except IndexError:
    messagebox.showwarning("Warning", "Please select an item to delete.")

# 'Add Employee' button
add_employee_button = tk.Button(root, text="Add Employee",
                                command=add_employee_button_click)
add_employee_button.grid(row=3, column=0, padx=5, pady=5)

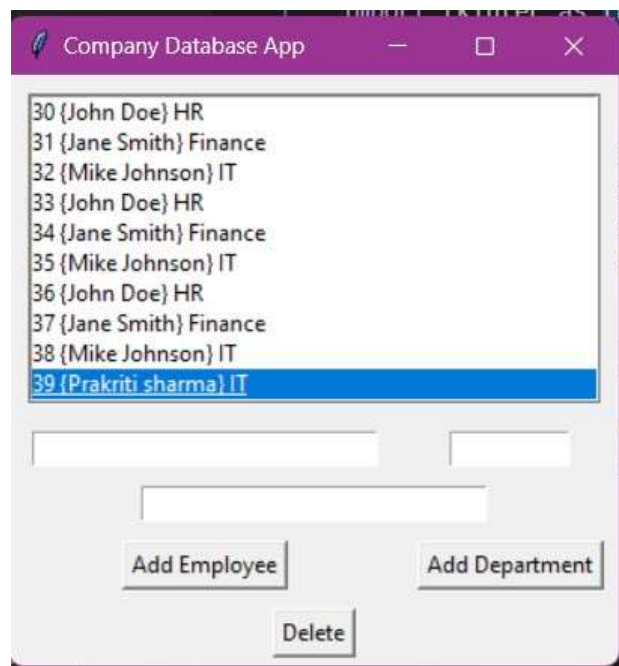
# 'Add Department' button
add_department_button = tk.Button(root, text="Add Department",
                                   command=add_department_button_click)
add_department_button.grid(row=3, column=1, padx=5, pady=5)

# 'Delete' button
delete_button = tk.Button(root, text="Delete", command=delete_button_click)
delete_button.grid(row=4, column=0, columnspan=2, padx=5, pady=5)

root.mainloop()

```

OUTPUT:



EXPLANATION: -

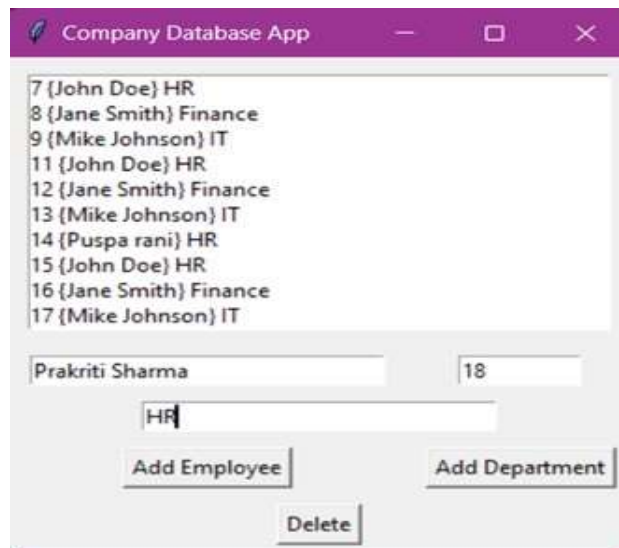
- **Functionality:** The application allows users to manage employee data within a company database. It provides features to add new employees, delete existing employees, and displays a list of employees along with their respective departments.
- **GUI Structure:** The GUI consists of a main window where users can view employee data in a listbox, add new employees using entry fields, and delete selected employees using buttons.
- **Database Interaction:** It utilizes SQLite to create and manage two tables: "Employee" and "Department". The Employee table stores employee details such as name and department ID, while the Department table stores department names.

Code Structure:

- **Database Operations Functions:** There are functions for creating tables (create_department_table() and create_employee_table()), populating tables with sample data (populate_department_table() and populate_employee_table()), loading employee data (load_employee_data()), adding new employees (add_employee_data()), and deleting employees (delete_employee_data()).
- **GUI Setup:** The Tkinter window is created (root = tk.Tk()) along with the title "Company Database App". It defines the listbox (listbox_employee) for displaying employee data and entry fields (entry_name and entry_department_id) for adding new.
- **Button Functions:** The add_button_click() and delete_button_click() functions handle the "Add" and "Delete" button clicks respectively. They retrieve input data from entry fields, perform necessary database operations, and update the listbox accordingly. Error messages are displayed if required fields are empty or if invalid data is entered.
- **Main Loop:** The root.mainloop() function initiates the Tkinter event loop, allowing the GUI to respond to user interactions.

OPERATIONS :

1. TO ADD DATA :



Company Database App

7 {John Doe} HR
8 {Jane Smith} Finance
9 {Mike Johnson} IT
11 {John Doe} HR
12 {Jane Smith} Finance
13 {Mike Johnson} IT
14 {Puspa rani} HR
15 {John Doe} HR
16 {Jane Smith} Finance
17 {Mike Johnson} IT

Prakriti Sharma 18

HR

Add Employee Add Department

Delete

2. TO DELETE DATA:-

7 {John Doe} HR
8 {Jane Smith} Finance
9 {Mike Johnson} IT
11 {John Doe} HR
12 {Jane Smith} Finance
13 {Mike Johnson} IT
14 {Puspa rani} HR
15 {John Doe} HR
16 {Jane Smith} Finance
17 {Mike Johnson} IT

3. Create a Mini Project report for the application you have created.

Mini Project Report: Company Database Application

Introduction

The Company Database Application is a graphical user interface (GUI) application developed using Python's Tkinter library and SQLite database. The primary objective of this application is to provide a user-friendly interface for managing employee data within a company. It allows users to add new employees, delete existing employees, and view a list of employees along with their respective departments.

Features

- Add New Employee: Users can enter the name of a new employee and select the corresponding department ID from a drop-down menu to add a new employee to the database.
- Delete Employee: Users can select an employee from the list and click the "Delete" button to remove the employee's record from the database.
- View Employee List: The application displays a list of employees, including their names and respective departments, in a listbox.
- Add New Department: Users can enter the name of a new department, which will be added to the Department table in the database.

Technologies Used

- Programming Language: Python
- GUI Library: Tkinter

Code Structure

The application is structured into several functions, each serving a specific purpose:

- a. Database Connection: The `'connect_to_database()'` function establishes a connection to the SQLite database file (`'company.db'`).
- b. Table Creation: The `'create_department_table()'` and `'create_employee_table()'` functions create the necessary tables (`'Department'` and `'Employee'`) if they do not already exist.
- c. Data Manipulation: The `'populate_department_table()'` and `'populate_employee_table()'` functions populate the respective tables with sample data. The `'load_employee_data()'` function retrieves employee data from the database, while `'add_employee_data()'`, `'add_department_data()'`, and `'delete_employee_data()'` functions handle adding and deleting data.
- d. GUI Setup: The main Tkinter window is created, along with a listbox for displaying employee data and entry fields for user input.
- e. Event Handling: The `'add_employee_button_click()'`, `'add_department_button_click()'`, and `'delete_button_click()'` functions handle button click events and perform the corresponding database operations.

Installation and Usage

To run the Company Database Application, follow these steps:

1. Ensure that you have Python installed on your system.
2. Clone or download the project files to your local machine.
3. Open a terminal or command prompt and navigate to the project directory.
4. Run the following command to execute the application:

```
'''  
python main.py  
'''
```

5. The application window will open, and you can start interacting with the application.

Conclusion

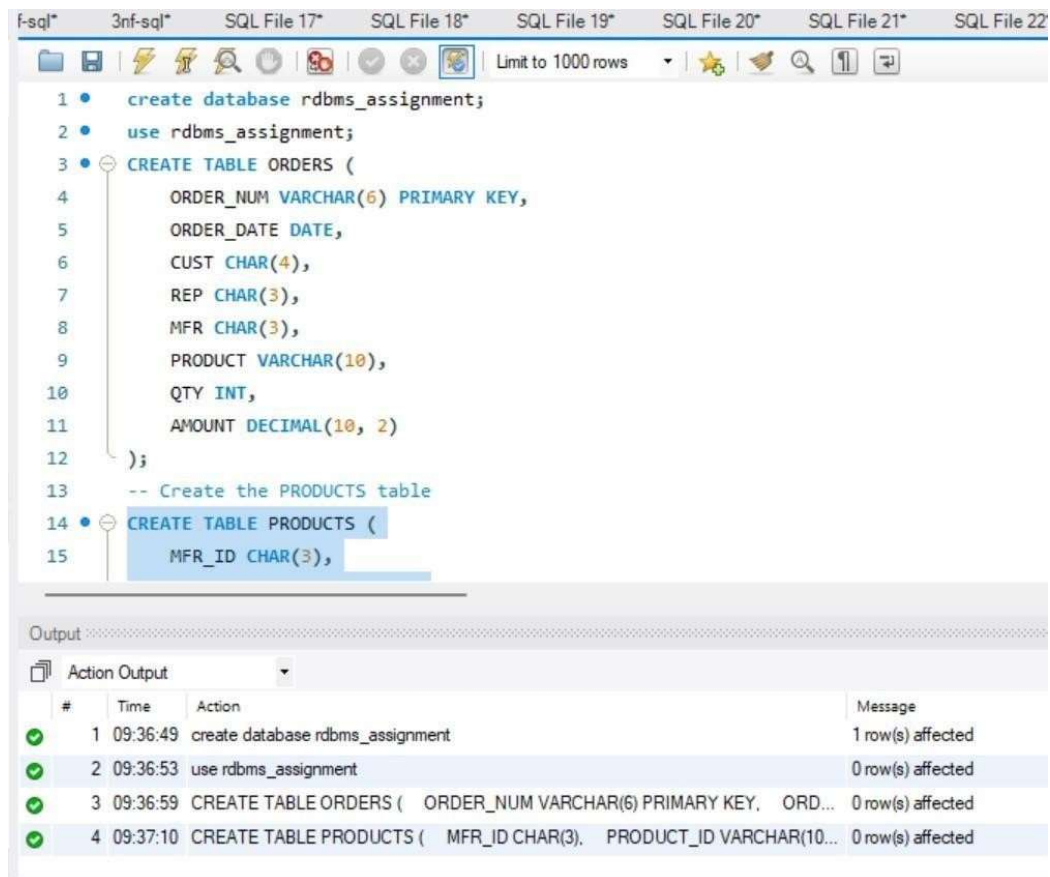
The Company Database Application provides a simple yet functional solution for managing employee data within a company. It demonstrates the integration of Python's Tkinter library with SQLite database for creating a user-friendly GUI application. Further enhancements can be made to include additional features, such as editing employee data, generating reports, or implementing user authentication.

Assignment 2 -- Practice writing Queries

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, and 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.

Creation of database:



The screenshot shows a SQL IDE window with multiple tabs. The active tab displays the following SQL code:

```
1 • create database rdbms_assignment;
2 • use rdbms_assignment;
3 • CREATE TABLE ORDERS (
4     ORDER_NUM VARCHAR(6) PRIMARY KEY,
5     ORDER_DATE DATE,
6     CUST CHAR(4),
7     REP CHAR(3),
8     MFR CHAR(3),
9     PRODUCT VARCHAR(10),
10    QTY INT,
11    AMOUNT DECIMAL(10, 2)
12 );
13 -- Create the PRODUCTS table
14 • CREATE TABLE PRODUCTS (
15     MFR_ID CHAR(3),
```

Below the code editor, the 'Output' pane is visible, showing the execution results:

#	Time	Action	Message
✓ 1	09:36:49	create database rdbms_assignment	1 row(s) affected
✓ 2	09:36:53	use rdbms_assignment	0 row(s) affected
✓ 3	09:36:59	CREATE TABLE ORDERS (ORDER_NUM VARCHAR(6) PRIMARY KEY, ORD...	0 row(s) affected
✓ 4	09:37:10	CREATE TABLE PRODUCTS (MFR_ID CHAR(3). PRODUCT_ID VARCHAR(10...	0 row(s) affected

```
f-sql* 3nf-sql* SQL File 17* SQL File 18* SQL File 19* SQL File 20* SQL File 21* SQL File 22*
Limit to 1000 rows
38     SALES DECIMAL(10, 2)
39 );
40
41 -- Create the SalesReps table
42 CREATE TABLE SalesReps (
43     emp_num CHAR(3) PRIMARY KEY,
44     name VARCHAR(20),
45     age INT,
46     rep_office CHAR(2),
47     title VARCHAR(10),
48     manager CHAR(3),
49     hire_date DATE,
50     quota DECIMAL(10, 2),
51     sales DECIMAL(10, 2)
52 );
```

Output

Action Output

#	Time	Action	Message
✓ 4	09:37:10	CREATE TABLE PRODUCTS (MFR_ID CHAR(3), PRODUCT_ID VARCHAR(...	0 row(s) affected
✓ 5	09:37:41	CREATE TABLE CUSTOMERS (CUST_NUM CHAR(4) PRIMARY KEY, COM...	0 row(s) affected
✓ 6	09:37:48	CREATE TABLE OFFICES (OFFICE CHAR(2) PRIMARY KEY, CITY VARCHA...	0 row(s) affected
✓ 7	09:37:59	CREATE TABLE SalesReps (emp_num CHAR(3) PRIMARY KEY, name VARC...	0 row(s) affected

```
f-sql* 3nf-sql* SQL File 17* SQL File 18* SQL File 19* SQL File 20* SQL File 21* SQL File 22* SQL File 23*
Limit to 1000 rows
142 ('REI', '2A44G', 'Hinge Pin', 350, 14),
143 ('REI', '2A44L', 'Left Hinge', 4500, 12),
144 ('REI', '2A44R', 'Right Hinge', 4500, 12),
145 ('REI', '2A45C', 'Ratchet Link', 79, 210);
146
147 -- Insert data into SalesReps table
148 INSERT INTO SalesReps VALUES
149 ('101', 'Dan Roberts', 45, '12', 'Sales Rep', '104', '1996-10-20', 300000, 305673),
150 ('102', 'Sue Smith', 48, '21', 'Sales Rep', '108', '1996-12-10', 350000, 474050),
151 ('103', 'Paul Cruz', 29, '12', 'Sales Rep', '104', '1997-03-01', 275000, 286775),
152 ('104', 'Bob Smith', 33, '12', 'Sales Mgr', '106', '1997-05-19', 200000, 142594),
153 ('105', 'Bill Adams', 37, '13', 'Sales Rep', '104', '1996-02-12', 350000, 367911),
154 ('106', 'Sam Clark', 52, '11', 'Vp Sales', NULL, '1998-06-14', 275000, 299912),
155 ('107', 'Nancy Angelli', 49, '22', 'Sales Rep', '108', '1996-11-14', 300000, 186042),
156 ('108', 'Larry Fitch', 62, '21', 'Sales Mgr', '106', '1999-10-12', 350000, 361865),
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
254
```


159 • `select * from salesreps;`

emp_num	name	age	rep_office	title	manager	hire_date	quota	sales
101	Dan Roberts	45	12	Sales Rep	104	1996-10-20	300000.00	305673.00
102	Sue Smith	48	21	Sales Rep	108	1996-12-10	350000.00	474050.00
103	Paul Cruz	29	12	Sales Rep	104	1997-03-01	275000.00	286775.00
104	Bob Smith	33	12	Sales Mgr	106	1997-05-19	200000.00	142594.00
105	Bill Adams	33	12	Sales Rep	104	1996-02-12	350000.00	367911.00
106	Sam Clark	52	11	Vp Sales	108	1998-06-14	275000.00	299912.00
107	Nancy Angelli	49	22	Sales Rep	108	1998-11-14	300000.00	186042.00

salesreps 1 x

Output

#	Time	Action	Message
10	09:38:57	INSERT INTO ORDERS VALUES ('112961', '1999-12-17', '2117', '106', 'REI', '2...	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0
11	09:39:16	INSERT INTO PRODUCTS VALUES ('ACI', '41002', 'Size 2 Widget', 76, 167), ...	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0
12	09:40:11	INSERT INTO SalesReps VALUES ('101', 'Dan Roberts', 45, '12', 'Sales Rep', '1...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
13	09:41:05	select * from salesreps LIMIT 0, 1000	10 row(s) returned

Queries:

(Unless otherwise instructed, you should assume the query is asking for names, not id numbers of customers, people, product or city offices; rename attributes if the meaning of the resultant table is not clear.)

1. Show the name, sales, and quota of Bill Adams.

ANS:

161 `-- 1. Show the name, sales, and quota of Bill Adams`
 162 • `SELECT name, sales, quota`
 163 `FROM SalesReps`
 164 `WHERE name = 'Bill Adams';`
 ...

name	sales	quota
Bill Adams	367911.00	350000.00

SalesReps 2 x

Output

#	Time	Action	Message
11	09:39:16	INSERT INTO PRODUCTS VALUES ('ACI', '41002', 'Size 2 Widget', 76, 167), ...	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0
12	09:40:11	INSERT INTO SalesReps VALUES ('101', 'Dan Roberts', 45, '12', 'Sales Rep', '1...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
13	09:41:05	select * from salesreps LIMIT 0, 1000	10 row(s) returned
14	09:41:43	SELECT name, sales, quota FROM SalesReps WHERE name = 'Bill Adams' LIM...	1 row(s) returned

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

ANS:

SQL File 17 SQL File 18 SQL File 19 SQL File 20 SQL File 21 SQL File 22 SQL File 23

Limit to 1000 rows

```

167 -- 2. List the company names and the product description of all the products each has ordered. Arran
168 • SELECT c.COMPANY, p.DESCRPTION
169 FROM CUSTOMERS c
170 JOIN ORDERS o ON c.CUST_NUM = o.CUST
171 JOIN PRODUCTS p ON o.PRODUCT = p.PRODUCT_ID
172 ORDER BY c.COMPANY DESC;

```

Result Grid

	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

Result 3 x Read Only

Output

Action Output

#	Time	Action	Message
✓ 12	09:40:11	INSERT INTO SalesReps VALUES ('101', 'Dan Roberts', 45, '12', 'Sales Rep', '1...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
✓ 13	09:41:05	select * from salesreps LIMIT 0, 1000	10 row(s) returned
✓ 14	09:41:43	SELECT name, sales, quota FROM SalesReps WHERE name = 'Bill Adams' LIM...	1 row(s) returned
✓ 15	09:42:04	SELECT c.COMPANY, p.DESCRPTION FROM CUSTOMERS c JOIN ORDERS o...	33 row(s) returned

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

ANS:

f-sql* 3nf-sql* SQL File 17* SQL File 18* SQL File 19* SQL File 20* SQL File 21* SQL File 22* SQL File 23*

Limit to 1000 rows

```
174 -- 3. Show the total value of the inventory on hand for each product. Arrange in descending order by
175 SELECT PRODUCT_ID, SUM(PRICE * QTY_ON_HAND) AS Total_Value
176 FROM PRODUCTS
177 GROUP BY PRODUCT_ID
178 ORDER BY Total_Value DESC;
179
```

Result Grid

	PRODUCT_ID	Total_Value
▶	4101	70000.00
	4100Y	68750.00
	2A44L	54000.00
	2A44R	54000.00
	773C	27300.00
	XK48	27202.00
	41003	24105.00

Result 4 x Read Only

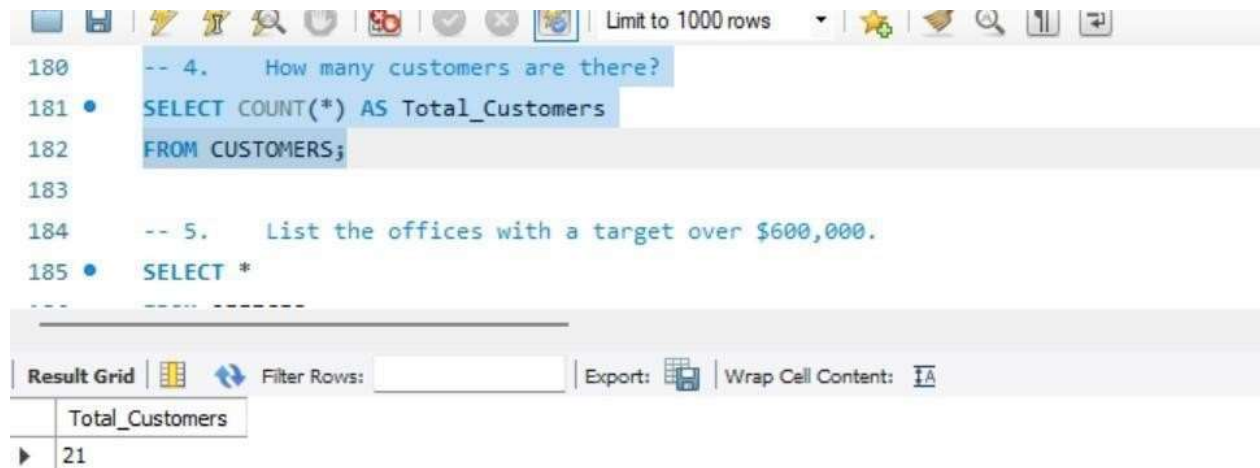
Output

Action Output

#	Time	Action	Message
✓ 13	09:41:05	select * from salesreps LIMIT 0, 1000	10 row(s) returned
✓ 14	09:41:43	SELECT name, sales, quota FROM SalesReps WHERE name = 'Bill Adams' LIM...	1 row(s) returned
✓ 15	09:42:04	SELECT c.COMPANY, p.DESCRPTION FROM CUSTOMERS c JOIN ORDERS o ...	33 row(s) returned
✓ 16	09:42:51	SELECT PRODUCT_ID, SUM(PRICE * QTY_ON_HAND) AS Total_Value FROM P...	24 row(s) returned

4. How many customers are there?

ANS:

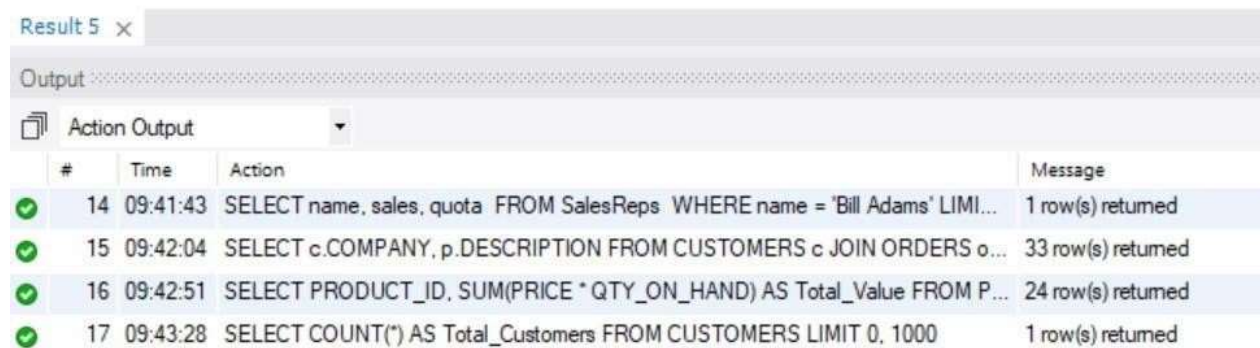


The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
180  -- 4.  How many customers are there?
181  •  SELECT COUNT(*) AS Total_Customers
182  FROM CUSTOMERS;
183
184  -- 5.  List the offices with a target over $600,000.
185  •  SELECT *
```

Below the editor is the 'Result Grid' tab. It shows a single column named 'Total_Customers' with a value of 21.

Total_Customers
21



The screenshot shows the 'Action Output' window with a table of SQL actions. The table has columns for '#', 'Time', 'Action', and 'Message'.

#	Time	Action	Message
✓ 14	09:41:43	SELECT name, sales, quota FROM SalesReps WHERE name = 'Bill Adams' LIM...	1 row(s) returned
✓ 15	09:42:04	SELECT c.COMPANY, p.DESCRPTION FROM CUSTOMERS c JOIN ORDERS o...	33 row(s) returned
✓ 16	09:42:51	SELECT PRODUCT_ID, SUM(PRICE * QTY_ON_HAND) AS Total_Value FROM P...	24 row(s) returned
✓ 17	09:43:28	SELECT COUNT(*) AS Total_Customers FROM CUSTOMERS LIMIT 0, 1000	1 row(s) returned

5. List the offices with a target over \$600,000.

ANS:

```

184 -- 5. List the offices with a target over $600,000.
185 • SELECT *
186 FROM OFFICES
187 WHERE TARGET > 600000;
188

```

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

Result Grid

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

OFFICES 6 x

Output

Action Output

#	Time	Action	Message
✓ 15	09:42:04	SELECT c.COMPANY, p.DESCRPTION FROM CUSTOMERS c JOIN ORDERS o...	33 row(s) returned
✓ 16	09:42:51	SELECT PRODUCT_ID, SUM(PRICE * QTY_ON_HAND) AS Total_Value FROM P...	24 row(s) returned
✓ 17	09:43:28	SELECT COUNT(*) AS Total_Customers FROM CUSTOMERS LIMIT 0, 1000	1 row(s) returned
✓ 18	09:43:56	SELECT * FROM OFFICES WHERE TARGET > 600000 LIMIT 0, 1000	2 row(s) returned

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

ANS:

```

189 -- 6. What is the average of all the sales people?
190 • SELECT AVG(sales) AS Average_Sales
191 FROM SalesReps;

```

Average_Sales
289353.200000

Result Grid

Filter Rows: Export: Wrap Cell Content:

Result 7 x

Output

Action Output

#	Time	Action	Message
✓ 16	09:42:51	SELECT PRODUCT_ID, SUM(PRICE * QTY_ON_HAND) AS Total_Value FROM P...	24 row(s) returned
✓ 17	09:43:28	SELECT COUNT(*) AS Total_Customers FROM CUSTOMERS LIMIT 0, 1000	1 row(s) returned
✓ 18	09:43:56	SELECT * FROM OFFICES WHERE TARGET > 600000 LIMIT 0, 1000	2 row(s) returned
✓ 19	09:44:28	SELECT AVG(sales) AS Average_Sales FROM SalesReps LIMIT 0, 1000	1 row(s) returned

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

ANS:

Limit to 1000 rows

```

193 -- 7. List orders over $25,000, including the name of the salesperson who took the order and the name
194 • SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS Customer
195 FROM ORDERS o
196 JOIN SalesReps s ON o.REP = s.emp_num
197 JOIN CUSTOMERS c ON o.CUST = c.CUST_NUM
198 WHERE o.AMOUNT > 25000;

```

Result Grid

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

Result 8 x Read Only

Output

Action Output

#	Time	Action	Message
✓ 17	09:43:28	SELECT COUNT(*) AS Total_Customers FROM CUSTOMERS LIMIT 0, 1000	1 row(s) returned
✓ 18	09:43:56	SELECT * FROM OFFICES WHERE TARGET > 600000 LIMIT 0, 1000	2 row(s) returned
✓ 19	09:44:28	SELECT AVG(sales) AS Average_Sales FROM SalesReps LIMIT 0, 1000	1 row(s) returned
✓ 20	09:45:08	SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS ...	4 row(s) returned

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

ANS:

```

200 -- 8. How many sales offices have salespeople who are over quota?
201 • SELECT COUNT(DISTINCT rep_office) AS Offices_Over_Quota
202 FROM SalesReps
203 WHERE quota < sales;
204

```

Result Grid

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

Result 9 x

Output

Action Output

#	Time	Action	Message
✓ 18	09:43:56	SELECT * FROM OFFICES WHERE TARGET > 600000 LIMIT 0, 1000	2 row(s) returned
✓ 19	09:44:28	SELECT AVG(sales) AS Average_Sales FROM SalesReps LIMIT 0, 1000	1 row(s) returned
✓ 20	09:45:08	SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS ...	4 row(s) returned
✓ 21	09:45:54	SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS ...	4 row(s) returned

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

ANS:

The screenshot shows a database IDE with a SQL editor and a results pane. The SQL editor contains the following query:

```
-- 9. Show the name, sales and office for each salesperson. Order by increasing sales.
SELECT name, sales, rep_office AS Office
FROM SalesReps
ORDER BY sales;
```

The results pane displays a table with the following data:

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

The results pane also shows an "Action Output" section with the following entries:

#	Time	Action	Message
19	09:44:28	SELECT AVG(sales) AS Average_Sales FROM SalesReps LIMIT 0, 1000	1 row(s) returned
20	09:45:08	SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS ...	4 row(s) returned
21	09:45:54	SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS ...	4 row(s) returned
22	09:46:15	SELECT name, sales, rep_office AS Office FROM SalesReps ORDER BY sales LI...	10 row(s) returned

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

ANS:

The screenshot shows a SQL IDE interface. At the top, a query is entered in a text area:

```
-- 10. List all the companies who have ordered any size widget, and the widget they ordered
SELECT DISTINCT c.COMPANY, p.DESRIPTION AS Widget_Ordered
FROM CUSTOMERS c
JOIN ORDERS o ON c.CUST_NUM = o.CUST
JOIN PRODUCTS p ON o.PRODUCT = p.PRODUCT_ID
WHERE p.DESRIPTION LIKE '%widget%';
```

Below the query, the 'Result Grid' is displayed, showing the results of the query. The grid has two columns: 'COMPANY' and 'Widget_Ordered'. The results are as follows:

COMPANY	Widget_Ordered
Acme Mfg.	Size 4 Widget
First Corp.	Size 4 Widget
Orion Corp.	Size 1 Widget
Acme Mfg.	Widget Remover
Miswest Sytems	Size 2 Widget
Peter Brothers	Size 3 Widget
JCP Inc.	Size 3 Widget

At the bottom of the screenshot, the 'Action Output' pane is visible, showing a log of SQL actions and their results:

#	Time	Action	Message
20	09:45:08	SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS ...	4 row(s) returned
21	09:45:54	SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS ...	4 row(s) returned
22	09:46:15	SELECT name, sales, rep_office AS Office FROM SalesReps ORDER BY sales LI...	10 row(s) returned
23	09:46:48	SELECT DISTINCT c.COMPANY, p.DESRIPTION AS Widget_Ordered FROM C...	12 row(s) returned

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

ANS:


```

217 -- 11. List the city, region and amount that sales are over/under target for each office.
218 • SELECT CITY, REGION, SALES - TARGET AS Difference
219 FROM OFFICES;
220

```

Result Grid

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

Result 12 x Read Only

Output

Action Output

#	Time	Action	Message
21	09:45:54	SELECT o.ORDER_NUM, o.AMOUNT, s.name AS Salesperson, c.COMPANY AS ...	4 row(s) returned
22	09:46:15	SELECT name, sales, rep_office AS Office FROM SalesReps ORDER BY sales LI...	10 row(s) returned
23	09:46:48	SELECT DISTINCT c.COMPANY, p.DESCRPTION AS Widget_Ordered FROM C...	12 row(s) returned
24	09:47:17	SELECT CITY, REGION, SALES - TARGET AS Difference FROM OFFICES LIMIT ...	5 row(s) returned

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

ANS:

```

221 -- 12. What is the total number of each part that has been ordered?
222 • SELECT PRODUCT, SUM(QTY) AS Total_Quantity_Ordered
223 FROM ORDERS
224 GROUP BY PRODUCT;
225
226

```

Result Grid

PRODUCT	Total_Quantity_Ordered
2A44L	
41004	68
2A44G	6
4100Z	15
4100Y	11
114	16
41002	64

Result 13 x

Output

Action Output

#	Time	Action	Message
22	09:46:15	SELECT name, sales, rep_office AS Office FROM SalesReps ORDER BY sales LI...	10 row(s) returned
23	09:46:48	SELECT DISTINCT c.COMPANY, p.DESCRPTION AS Widget_Ordered FROM C...	12 row(s) returned
24	09:47:17	SELECT CITY, REGION, SALES - TARGET AS Difference FROM OFFICES LIMIT ...	5 row(s) returned
25	09:48:10	SELECT PRODUCT, SUM(QTY) AS Total_Quantity_Ordered FROM ORDERS GR...	16 row(s) returned

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

ANS:

SQL File 23*

```

225 -- 13. List the salespeople, the city they work in, and the manager of the office in which they work
226 • SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager
227 FROM SalesReps s
228 JOIN OFFICES o ON s.rep_office = o.OFFICE;
229
230

```

Result Grid

Salesperson	CITY	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	HULL
Nancy Angelli	Denver	108

Result 14 x Read Only

Output

Action Output

#	Time	Action	Message
23	09:46:48	SELECT DISTINCT c.COMPANY, p.DESCRPTION AS Widget_Ordered FROM C...	12 row(s) returned
24	09:47:17	SELECT CITY, REGION, SALES - TARGET AS Difference FROM OFFICES LIMIT ...	5 row(s) returned
25	09:48:10	SELECT PRODUCT, SUM(QTY) AS Total_Quantity_Ordered FROM ORDERS GR...	16 row(s) returned
26	09:48:48	SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager FROM SalesRe...	9 row(s) returned

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

ANS:

```

231 -- 14. List all orders showing order number, amount, customer name and the customer's credit limit wher
232 • SELECT o.ORDER_NUM, o.AMOUNT, c.COMPANY AS Customer_Name, c.CREDIT_LIMIT
233 FROM ORDERS o
234 JOIN CUSTOMERS c ON o.CUST = c.CUST_NUM
235 WHERE o.AMOUNT > 20000;
236

```

Result Grid

Salesperson	CITY	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	HULL
Nancy Angelli	Denver	108

Result 15 x Read Only

Output

Action Output

#	Time	Action	Message
24	09:47:17	SELECT CITY, REGION, SALES - TARGET AS Difference FROM OFFICES LIMIT ...	5 row(s) returned
25	09:48:10	SELECT PRODUCT, SUM(QTY) AS Total_Quantity_Ordered FROM ORDERS GR...	16 row(s) returned
26	09:48:48	SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager FROM SalesRe...	9 row(s) returned
27	09:49:39	SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager FROM SalesRe...	9 row(s) returned

15. 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.

ANS:

Limit to 1000 rows

```

236 -- 15. Are there any customers who are over their credit limit? If so, list the customer, the total amount
237 • SELECT c.COMPANY AS Customer, SUM(o.AMOUNT) AS Total_Order_Amount, c.CREDIT_LIMIT
238 FROM CUSTOMERS c
239 JOIN ORDERS o ON c.CUST_NUM = o.CUST
240 GROUP BY c.COMPANY, c.CREDIT_LIMIT
241 HAVING SUM(o.AMOUNT) > CAST(c.CREDIT_LIMIT AS DECIMAL(10,2));

```

Customer	Total_Order_Amount	CREDIT_LIMIT
Orion Corp.	22100.00	20000.00
Ian & Schmidt	22500.00	20000.00
Chen Associates	31350.00	25000.00

Result 16 x Read Only

Output

Action Output

#	Time	Action	Message
✓ 25	09:48:10	SELECT PRODUCT, SUM(QTY) AS Total_Quantity_Ordered FROM ORDERS GR...	16 row(s) returned
✓ 26	09:48:48	SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager FROM SalesRe...	9 row(s) returned
✓ 27	09:49:39	SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager FROM SalesRe...	9 row(s) returned
✓ 28	09:51:03	SELECT c.COMPANY AS Customer, SUM(o.AMOUNT) AS Total_Order_Amount, c...	3 row(s) returned

16. List the salespeople with a higher quota than their manager.

ANS:

Limit to 1000 rows

```

242 -- 16. List the salespeople with a higher quota than their manager.
243 • SELECT s.name AS Salesperson
244 FROM SalesReps s
245 JOIN SalesReps m ON s.manager = m.emp_num
246 WHERE s.quota > m.quota;
247

```

Salesperson
Dan Roberts
Paul Cruz
Bill Adams
Larry Fitch
Mary Jones

Result 17 x

Output

Action Output

#	Time	Action	Message
✓ 26	09:48:48	SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager FROM SalesRe...	9 row(s) returned
✓ 27	09:49:39	SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager FROM SalesRe...	9 row(s) returned
✓ 28	09:51:03	SELECT c.COMPANY AS Customer, SUM(o.AMOUNT) AS Total_Order_Amount, c...	3 row(s) returned
✓ 29	09:51:45	SELECT s.name AS Salesperson FROM SalesReps s JOIN SalesReps m ON s.man...	5 row(s) returned

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

ANS:

The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

```
-- 17. List salespeople who work in different offices than their managers, show the name and office where  
SELECT s.name AS Salesperson, s.rep_office AS Salesperson_Office, m.rep_office AS Manager_Office  
FROM SalesReps s  
JOIN SalesReps m ON s.manager = m.emp_num  
WHERE s.rep_office != m.rep_office;
```

The results are displayed in a table with the following columns: Salesperson, Salesperson_Office, and Manager_Office. The data is as follows:

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

The bottom of the screenshot shows the 'Output' pane with the following table:

#	Time	Action	Message
27	09:49:39	SELECT s.name AS Salesperson, o.CITY, s.manager AS Manager FROM SalesRe...	9 row(s) returned
28	09:51:03	SELECT c.COMPANY AS Customer, SUM(o.AMOUNT) AS Total_Order_Amount, c...	3 row(s) returned
29	09:51:45	SELECT s.name AS Salesperson FROM SalesReps s JOIN SalesReps m ON s.man...	5 row(s) returned
30	09:52:05	SELECT s.name AS Salesperson, s.rep_office AS Salesperson_Office, m.rep_office...	4 row(s) returned

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

ANS:

The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```
-- 18. What is the total order size for each salesperson? Order by increasing sales
SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size
FROM SalesReps s
JOIN ORDERS o ON s.emp_num = o.REP
GROUP BY s.name
ORDER BY Total_Order_Size;
```

The results pane displays the following table:

Salesperson	Total_Order_Size
Sam Clark	13
Mary Jones	13
Tom Snyder	17
Paul Cruz	30
Nancy Angelli	33
Sue Smith	38
Dan Roberts	45

Below the results pane, the 'Output' section shows the execution log:

#	Time	Action	Message
28	09:51:03	SELECT c.COMPANY AS Customer, SUM(o.AMOUNT) AS Total_Order_Amount, c...	3 row(s) returned
29	09:51:45	SELECT s.name AS Salesperson FROM SalesReps s JOIN SalesReps m ON s.man...	5 row(s) returned
30	09:52:05	SELECT s.name AS Salesperson, s.rep_office AS Salesperson_Office, m.rep_office...	4 row(s) returned
31	09:52:31	SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size FROM Sales...	9 row(s) returned

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

ANS:

SQL File 23*

```

261 -- 19. List all the customers whose sales representative is a manager. Arrange increasing by company.
262 SELECT c.COMPANY AS Customer
263 FROM CUSTOMERS c
264 JOIN SalesReps s ON c.CUST_REP = s.emp_num
265 WHERE s.title = 'Sales Mgr'
266 ORDER BY c.COMPANY;

```

Result Grid

Customer
Ian & Schmidt
Miswest Sytems
Zetacorp

Result 20 x Read Only

Output

Action Output

#	Time	Action	Message
29	09:51:45	SELECT s.name AS Salesperson FROM SalesReps s JOIN SalesReps m ON s.man...	5 row(s) returned
30	09:52:05	SELECT s.name AS Salesperson, s.rep_office AS Salesperson_Office, m.rep_office...	4 row(s) returned
31	09:52:31	SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size FROM Sales...	9 row(s) returned
32	09:52:58	SELECT c.COMPANY AS Customer FROM CUSTOMERS c JOIN SalesReps s ON...	3 row(s) returned

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

ANS:

SQL File 22*

```

268 -- 20. What is the total order size for each salesperson whose orders total more than $30,000?
269 SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size
270 FROM SalesReps s
271 JOIN ORDERS o ON s.emp_num = o.REP
272 GROUP BY s.name
273 HAVING SUM(o.AMOUNT) > 30000;

```

Result Grid

Salesperson	Total_Order_Size
Sam Clark	13
Bill Adams	134
Larry Fitch	50
Nancy Angelli	33

Result 21 x Read Only

Output

Action Output

#	Time	Action	Message
30	09:52:05	SELECT s.name AS Salesperson, s.rep_office AS Salesperson_Office, m.rep_office...	4 row(s) returned
31	09:52:31	SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size FROM Sales...	9 row(s) returned
32	09:52:58	SELECT c.COMPANY AS Customer FROM CUSTOMERS c JOIN SalesReps s ON...	3 row(s) returned
33	09:53:27	SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size FROM Sales...	4 row(s) returned

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

ANS:

Limit to 1000 rows

```

275 -- 21. List the offices where the sales target for the office exceeds the sum of the individual sales
276 • SELECT o.OFFICE, o.TARGET, SUM(s.quota) AS Total_Quota
277 FROM OFFICES o
278 JOIN SalesReps s ON o.OFFICE = s.rep_office
279 GROUP BY o.OFFICE, o.TARGET
280 HAVING o.TARGET > SUM(s.quota);

```

Result Grid

	OFFICE	TARGET	Total_Quota
▶	12	800000.00	775000.00
	21	725000.00	700000.00

Result 22 x Read Only

Output

Action Output

#	Time	Action	Message
✓ 31	09:52:31	SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size FROM Sales...	9 row(s) returned
✓ 32	09:52:58	SELECT c.COMPANY AS Customer FROM CUSTOMERS c JOIN SalesReps s ON...	3 row(s) returned
✓ 33	09:53:27	SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size FROM Sales...	4 row(s) returned
✓ 34	09:54:02	SELECT o.OFFICE, o.TARGET, SUM(s.quota) AS Total_Quota FROM OFFICES o ...	2 row(s) returned

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

ANS:

f-sql* 3nf-sql* SQL File 17* SQL File 18* SQL File 19* SQL File 20* SQL File 21* SQL File 22* SQL File 23*

Limit to 1000 rows

```

282 -- 22. List the salespeople whose quotas are equal to or higher than the target of the Atlanta sales
283 • SELECT s.name AS Salesperson, s.quota, o.TARGET
284 FROM SalesReps s
285 JOIN OFFICES o ON s.rep_office = o.OFFICE
286 WHERE o.CITY = 'Atlanta' AND s.quota >= o.TARGET;
287

```

Result Grid

	Salesperson	quota	TARGET
▶	Bill Adams	350000.00	350000.00

Result 23 x Read Only

Output

Action Output

#	Time	Action	Message
✓ 32	09:52:58	SELECT c.COMPANY AS Customer FROM CUSTOMERS c JOIN SalesReps s ON...	3 row(s) returned
✓ 33	09:53:27	SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size FROM Sales...	4 row(s) returned
✓ 34	09:54:02	SELECT o.OFFICE, o.TARGET, SUM(s.quota) AS Total_Quota FROM OFFICES o ...	2 row(s) returned
✓ 35	09:54:25	SELECT s.name AS Salesperson, s.quota, o.TARGET FROM SalesReps s JOIN O...	1 row(s) returned

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

ANS:

SQL File 23*

```

288 -- 23. List the salespeople who do not work in offices managed by Larry Fitch(employee 108).
289 • SELECT s.name AS Salesperson, o.MGR AS Office_Manager
290 FROM SalesReps s
291 JOIN OFFICES o ON s.rep_office = o.OFFICE
292 WHERE o.MGR != '108';
293

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Salesperson	Office_Manager
Dan Roberts	104
Paul Cruz	104
Bob Smith	104
Bill Adams	105
Sam Clark	106
Mary Jones	106

Result 24 x Read Only

Output

Action Output

#	Time	Action	Message
✓ 33	09:53:27	SELECT s.name AS Salesperson, SUM(o.QTY) AS Total_Order_Size FROM Sales...	4 row(s) returned
✓ 34	09:54:02	SELECT o.OFFICE, o.TARGET, SUM(s.quota) AS Total_Quota FROM OFFICES o ...	2 row(s) returned
✓ 35	09:54:25	SELECT s.name AS Salesperson, s.quota, o.TARGET FROM SalesReps s JOIN O...	1 row(s) returned
✓ 36	09:55:02	SELECT s.name AS Salesperson, o.MGR AS Office_Manager FROM SalesReps s ...	6 row(s) returned

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

ANS:

```

294 -- 24. List the products for which an order of $25,000 or more has been received.
295 • SELECT DISTINCT p.DESCRPTION AS Product_Description
296 FROM PRODUCTS p
297 JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT
298 WHERE o.AMOUNT >= 25000;
299

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Product_Description
Widget Remover
500 -lb Brace
Left Hinge
Right Hinge

Result 25 x

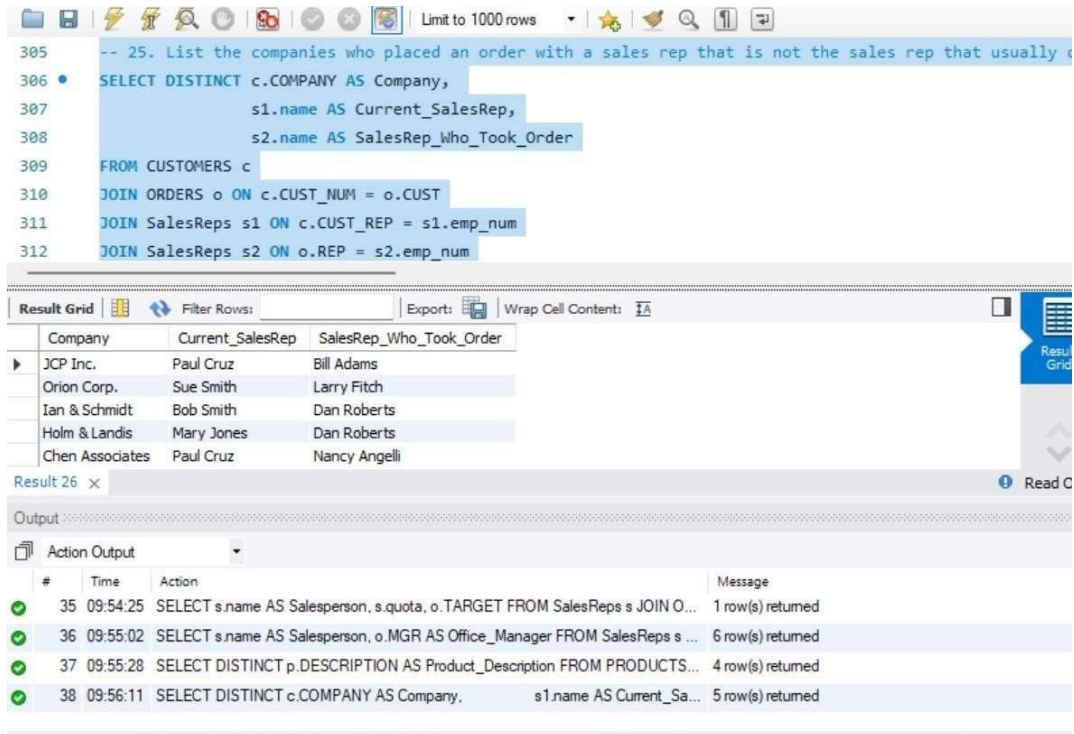
Output

Action Output

#	Time	Action	Message
✓ 34	09:54:02	SELECT o.OFFICE, o.TARGET, SUM(s.quota) AS Total_Quota FROM OFFICES o ...	2 row(s) returned
✓ 35	09:54:25	SELECT s.name AS Salesperson, s.quota, o.TARGET FROM SalesReps s JOIN O...	1 row(s) returned
✓ 36	09:55:02	SELECT s.name AS Salesperson, o.MGR AS Office_Manager FROM SalesReps s ...	6 row(s) returned
✓ 37	09:55:28	SELECT DISTINCT p.DESCRPTION AS Product_Description FROM PRODUCTS...	4 row(s) returned

25. List the companies who placed an order with a sales rep that is not the sales rep that usually calls on them. Include the names of the salesreps, indicating by attribute name who took the order.

ANS:



The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

```
-- 25. List the companies who placed an order with a sales rep that is not the sales rep that usually calls on them. Include the names of the salesreps, indicating by attribute name who took the order.
SELECT DISTINCT c.COMPANY AS Company,
               s1.name AS Current_SalesRep,
               s2.name AS SalesRep_Who_Took_Order
FROM CUSTOMERS c
JOIN ORDERS o ON c.CUST_NUM = o.CUST
JOIN SalesReps s1 ON c.CUST_REP = s1.emp_num
JOIN SalesReps s2 ON o.REP = s2.emp_num
```

The result grid shows the following data:

Company	Current_SalesRep	SalesRep_Who_Took_Order
JCP Inc.	Paul Cruz	Bill Adams
Orion Corp.	Sue Smith	Larry Fitch
Ian & Schmidt	Bob Smith	Dan Roberts
Holm & Landis	Mary Jones	Dan Roberts
Chen Associates	Paul Cruz	Nancy Angelli

The output section shows the execution of the query and the number of rows returned:

#	Time	Action	Message
35	09:54:25	SELECT s.name AS Salesperson, s.quota, o.TARGET FROM SalesReps s JOIN O...	1 row(s) returned
36	09:55:02	SELECT s.name AS Salesperson, o.MGR AS Office_Manager FROM SalesReps s ...	6 row(s) returned
37	09:55:28	SELECT DISTINCT p.DESCRPTION AS Product_Description FROM PRODUCTS...	4 row(s) returned
38	09:56:11	SELECT DISTINCT c.COMPANY AS Company, s1.name AS Current_Sa...	5 row(s) returned

Assignment 3 Database Design Assignment

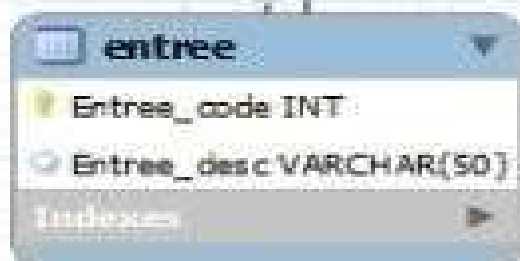
Part I -- the ER Diagram

1. Draw the ER diagram.
 - o 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.
 - o 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).

ANS:

In the below ER diagram:

- The **Member** entity has a one-to-many relationship with **Invitation**, as each member can receive multiple invitations.
- The **Invitation** entity has a one-to-many relationship with **Response**, as each invitation can have multiple responses (accept or decline).
- The **Response** entity has a one-to-many relationship with **Dinner**, as each response can be associated with multiple dinners (if the member attends multiple dinners).
- The **Dinner** entity has a one-to-one relationship with **Entree**, as each dinner is based on a single entree.
- The **Dinner** entity has a one-to-many relationship with **Dessert**, as each dinner can have multiple desserts.



2. Relational Schema (Tables): Member

- Member_num (PK)
- Member_name
- Member_address

- Member_city
- Member_zip

Invitation

- Invite_num (PK)
- Invite_date
- Dinner_date
- Dinner_code (FK references Dinner.Dinner_code)
- Member_num (FK references Member.Member_num)

Response

- Invite_num (PK, FK references Invitation.Invite_num)
- Member_num (PK, FK references Member.Member_num)
- Accept_date

Dinner

- Dinner_code (PK)
- Dinner_desc
- Entree_code (FK references Entree.Entree_code)

Entree

- Entree_code (PK)
- Entree_desc

Dessert

- Dessert_code (PK)
- Dessert_desc
- Dinner_code (FK references Dinner.Dinner_code)

Part II -- Normalization of the Universal Relation

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



In the above dependency diagram, the transitive dependencies are:

- Member_num -> Member_name, Member_address, Member_city, Member_zip
- Dinner_code -> Dinner_desc, Entree_code
- Entree_code -> Entree_desc
- Dinner_code -> Dessert_code
- Dessert_code -> Dessert_desc

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

To normalize the universal relation to the Third Normal Form (3NF), we need to remove any transitive dependencies and partial dependencies.

Step 1: Remove transitive dependencies to achieve 3NF

Member(Member_num, Member_name, Member_address, Member_city, Member_zip)
Invitation(Invite_num, Invite_date, Dinner_date, Accept_date) Dinner(Dinner_code, Dinner_desc,
Entree_code, Dessert_code) Entree(Entree_code, Entree_desc) Dessert(Dessert_code, Dessert_desc)

The resulting tables after normalization to 3NF are the same as the tables obtained from the ER diagram approach. Both methods lead to the same final database design, ensuring data integrity and minimizing redundancy.

Dependency Diagram:

Member_num -> Member_name
Member_address
Member_city
Member_zip

Invite_num -> Invite_date
Accept_date
Member_num
Dinner_date
Dinner_attend
Dinner_code

Dinner_date, -> Dinner_desc
Dinner_code

Entree_code -> Entree_desc

Dessert_code -> Dessert_desc

Member_num, -> Dinner_date
Dinner_attend Dinner_code

Transitive Dependencies:

- Invite_num -> Dinner_date -> Dinner_desc
- Invite_num -> Dinner_code -> Dinner_desc
- Invite_num -> Dinner_code -> Entree_desc

- Invite_num -> Dinner_code -> Dessert_desc
- Member_num, Dinner_attend -> Dinner_date -> Dinner_desc
- Member_num, Dinner_attend -> Dinner_code -> Dinner_desc
- Member_num, Dinner_attend -> Dinner_code -> Entree_desc
- Member_num, Dinner_attend -> Dinner_code -> Dessert_desc

Partial dependencies:

- Invite_num -> Member_num (Member_num is part of a composite key in the Invitation table)
- Invite_num -> Dinner_date (Dinner_date is part of a composite key in the Dinner table)
- Invite_num -> Dinner_code (Dinner_code is part of a composite key in the Dinner table)
- Member_num, Dinner_attend -> Member_num (Member_num is not part of the dependency but is included in the composite key).

Normalization to 3NF:

To normalize the universal connection with 3NF, we must remove the transitive and partial dependencies.

Step 1: Make separate tables for each entity (Member, Invitation, Dinner, Entree, and Dessert) and their attributes.

Code:

```
CREATE TABLE Member (
  Member_num INT NOT NULL,
  Member_name VARCHAR(255) NOT NULL,
  Member_address VARCHAR(255) NOT NULL,
  Member_city VARCHAR(255) NOT NULL,
  Member_zip VARCHAR(10) NOT NULL,
  PRIMARY KEY (Member_num)
);
```

```
CREATE TABLE Invitation (
  Invite_num INT NOT NULL,
  Invite_date DATE NOT NULL,
  Accept_date DATE,
  Member_num INT NOT NULL,
  Dinner_date DATE NOT NULL,
  Dinner_attend ENUM('Y', 'N') NOT NULL,
  PRIMARY KEY (Invite_num),
  FOREIGN KEY (Member_num) REFERENCES Member(Member_num)
);
```

```
CREATE TABLE Dinner (
  Dinner_date DATE NOT NULL,
  Dinner_code INT NOT NULL,
  Dinner_desc VARCHAR(255) NOT NULL,
  Entree_code INT NOT NULL,
  Dessert_code INT NOT NULL,
  PRIMARY KEY (Dinner_date, Dinner_code),
  FOREIGN KEY (Entree_code) REFERENCES Entree(Entree_code),
  FOREIGN KEY (Dessert_code) REFERENCES Dessert(Dessert_code)
);
```

```
CREATE TABLE Entree (
  Entree_code INT NOT NULL,
  Entree_desc VARCHAR(255) NOT NULL,
  PRIMARY KEY (Entree_code)
);
```

```
CREATE TABLE Dessert (
  Dessert_code INT NOT NULL,
  Dessert_desc VARCHAR(255) NOT NULL,
  PRIMARY KEY (Dessert_code)
);
```

Step 2: Remove partial dependencies by creating a new table for the Invitation-Dinner relationship:

Code:

```
CREATE TABLE Invitation_Dinner (
  Invite_num INT NOT NULL,
  Dinner_date DATE NOT NULL,
  Dinner_code INT NOT NULL,
  PRIMARY KEY (Invite_num),
  FOREIGN KEY (Invite_num) REFERENCES Invitation(Invite_num),
  FOREIGN KEY (Dinner_date, Dinner_code) REFERENCES Dinner(Dinner_date,
Dinner_code)
);
```

Step 3: Remove transitive dependencies by creating a new table for the Dinner-Entree and Dinner-Dessert relationships:

```
CREATE TABLE Dinner_Entree (
  Dinner_date DATE NOT NULL,
  Dinner_code INT NOT NULL,
  Entree_code INT NOT NULL,
  PRIMARY KEY (Dinner_date, Dinner_code),
  FOREIGN KEY (Dinner_date, Dinner_code) REFERENCES Dinner(Dinner_date,
Dinner_code),
  FOREIGN KEY (Entree_code) REFERENCES Entree(Entree_code)
);
```

```
CREATE TABLE Dinner_Dessert (
  Dinner_date DATE NOT NULL,
  Dinner_code INT NOT NULL,
  Dessert_code INT NOT NULL,
  PRIMARY KEY (Dinner_date, Dinner_code),
  FOREIGN KEY (Dinner_date, Dinner_code) REFERENCES Dinner(Dinner_date,
Dinner_code),
  FOREIGN KEY (Dessert_code) REFERENCES Dessert(Dessert_code)
);
```

Hence the problem solved.

