

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

EDA - 603 Major Assignment 1 – DB Modeling

Parul Bhutani Wadhwa (000504532)

Saskatchewan Polytechnic

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

Abstract

Normalization is a design approach which is used to create structured organization of data by reducing data redundancy and improving data integrity. It ensures that the dataset is saved only once to manage and update it easily.

Normalization is achieved by dividing a huge dataset into smaller and more manageable tables, which are connected to each other. By adding constraints, data redundancy is reduced, duplicate data is removed, and data consistency is improved.

Normalization levels, commonly known as normal forms, vary from first normal form (1NF) to fifth normal form (5NF) (5NF). To ensure that the database is properly normalized, all the levels must follow a set of rules and principles. The first normal form (1NF) needs atomic values in each column of a table, implying that the data cannot be split down into smaller components. Each non-key column in second normal form (2NF) must be completely dependent on the primary key. Each non-key column must be independent of the other non-key columns in the third normal form (3NF).

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

Table of Contents

Normalization.....	4
Process of Normalization	4
Questions	6
Entity Relationship Diagram	8
Queries	8

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

Normalization

The technique of structuring data in a database to reduce redundancy and increase data integrity is known as normalization.

Process of Normalization

The following steps are involved in the normalization process:

1. Determine entities and attributes: In the first stage we will identify which entities and characteristics will be stored in the database. In the employee excel sheet, the entities are employee, employee roles, committee, employee committee, and supervisor, and the attributes are EmpId, SinNo, FirstName, LastName, Street, City, Province, Postal, Birth Date, HiringDate, Inc. Tax, Hours, OT, Person Hours Worked, Supervisor, Supervisor Cell, Committee Id, Committee Name, and Meeting Night.
2. We will create distinct tables for each entity once the entities and attributes are decided. Each table should have a primary key that identifies each unique record in the table.
3. Identify relationships: Now we'll build the connections between the tables. For example, a supervisor may have numerous employees, which means there's a one-to-many link between the supervisor and employee tables. Also, an employee can be a member of multiple Committees, stating a one-to-many relationship between the person and the Committee table.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

4. Creating 2NF: Now identify each non-key column is completely dependent on the primary key and the dependent columns are shifted to a separate table in case of any partial dependencies.

The 2NF contains below tables:

Employee Table	Payroll Table	Job Table	Committee Table	EmployeeCommittee Involvement
EmpID, Last, First, Street, City, Prov, Postal, SIN, Birth Date	EmpID, Pay_Week_End_Date, Days_available, Hours, OT, Person_Hours_Worked	Empld, Job_Code, Position, Payrate, Inc Tax, Hire Date, Supervisor, Supervisor cell	ComId, ComName, MeetingNight	compositelid, EmpId, Committee Id

5. Creating 3NF: Now, we must ensure that each non-key column is independent of the others and the dependent columns are moved to a separate table when there are any transitive dependencies.

Employee Table (Employee_ID, Last, First, Street, City, Prov, Postal, SIN Birth Date)
Payroll Table (EmpID, Pay_Week_End_Date, Days_Available, Hours, OT)
Job Table (Empld, Job_Code, Inc.Tax, Hire Date)
Committees Table (Com_Id, Com_Name, Meeting_Night)
Supervisor Table (sId, Sname, Scell)
Position Table (job code, Position, payrate, SId)
EmployeeCommitteeInvolvement (Compositelid, EmpID, Committeelid)

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

Questions

- 1) There is a concern that the spreadsheet will get out of hand, as the number of pay periods increase, how does your 3NF solve this issue?**
 - ➔ In the 3NF we have created a separate table “Payroll Table” which will take care of the employee incomes for every Pay Period. So for every new pay period the data can be inserted to this table and it can be handled easily.
- 2) Supervisors often manage several departments, for example, Chad Long is managing butchers and bakers, how did you address this problem?**
 - ➔ I have created a separate table “Position Table” for various departments and “Supervisor Table” for all the supervisor details and created a composite key “supervisorId” which becomes the primary key for the table. In “Position Table” every unique department is linked to supervisor Id. If the user wants to get details of supervisor through any department, he can use inner join in position table with supervisor table based on supervisor Id as foreign key. Hence, one can find all the departments linked to one supervisor and viz-a-viz.
- 3) Employees often serve on multiple committees at once, how did you address this?**
 - ➔ I have created a table “Employee Committee Involvement”. In this table each employee id is linked with its committee id. Also, a composite key “EC_Id” is created which is unique for all the records and serves as the primary key in the table. Also, even if an employee is serving multiple committees, then committee meetings happen on different days which will not create any clashes.
- 4) Person hours worked is calculated (it represents information, not data) for all hours worked by all employees for that week, is there a better way to handle this rather than record it in the database? Implement your solution in the queries section.**
 - ➔ Person hours worked is calculated by adding the hours of all the employees in one pay period date. So, this field does not need to be stored in a database. Rather, we can create a procedure and it can be called whenever we need this field.
 - ➔ Create Procedure PersonHoursWorked

As

BEGIN

```
  Select PayWeekendDate, sum(HoursWorked) as totalHoursWorked
  From employeePayroll
  Where PayWeekendDate != ''
  Group by PayWeekendDate;
```

END;

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

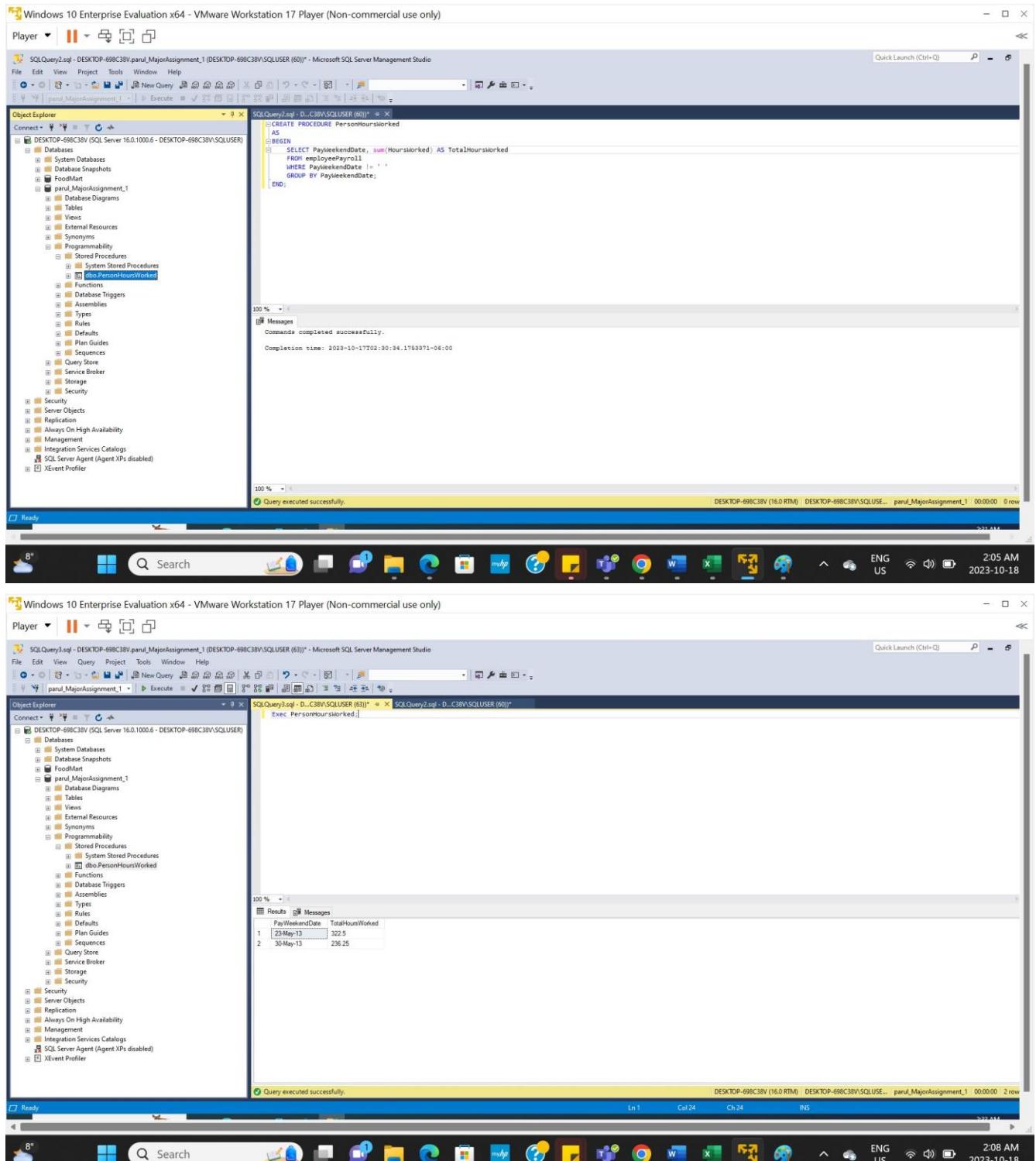


Fig 1. Store Procedure to Calculate Person Hours Worked and its result

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

Entity Relationship Diagram

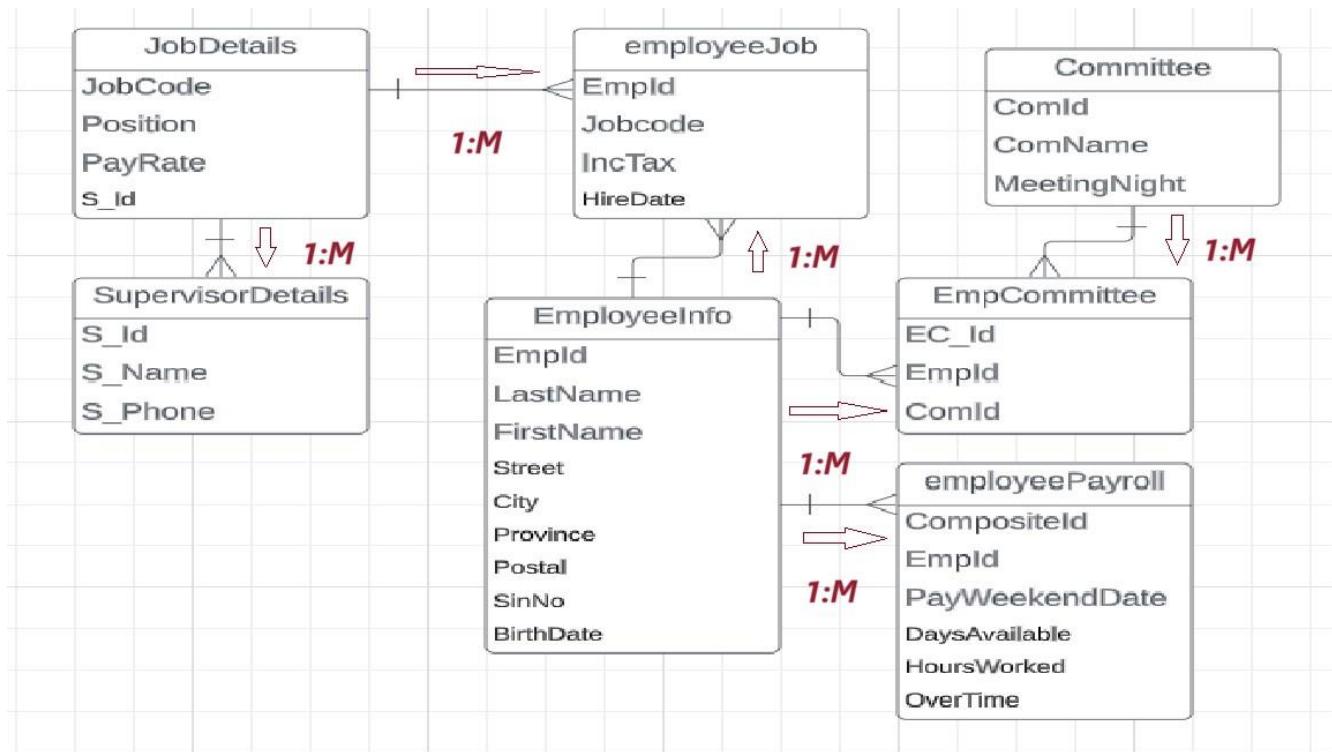


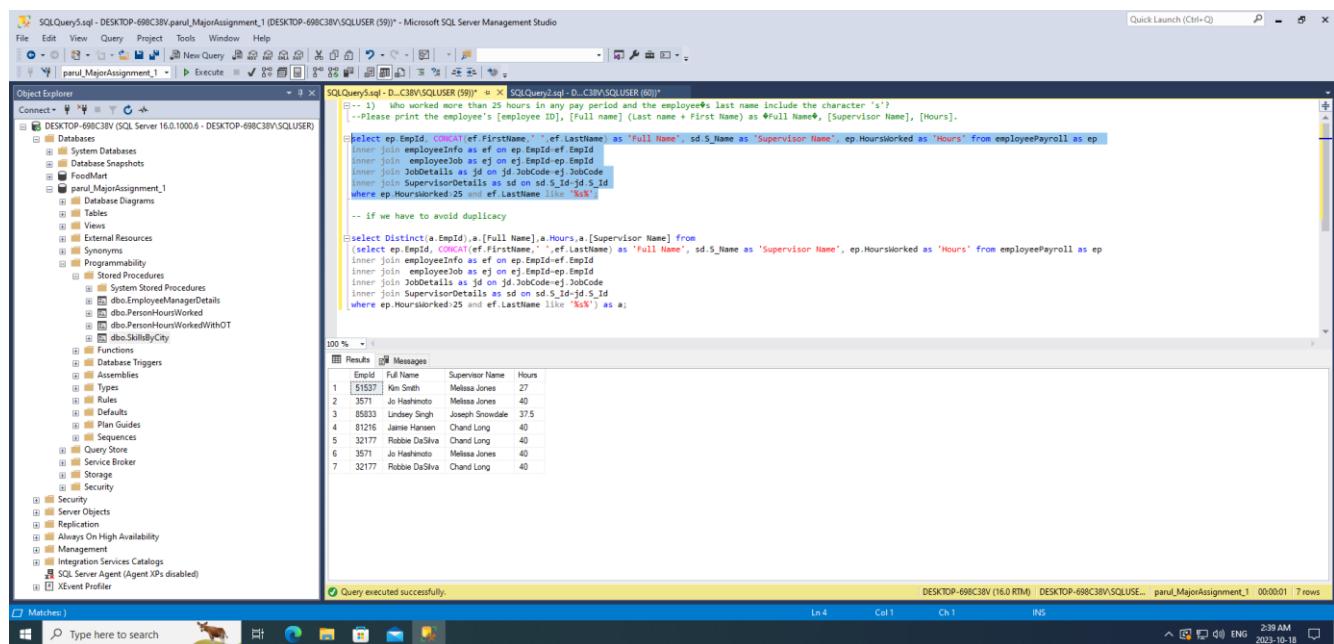
Fig 2. ER Diagram of 3NF.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

Queries

1) Who worked more than 25 hours in any pay period?

→ select ep.EmpId, CONCAT(ef.FirstName, ' ', ef.LastName) as 'Full Name', sd.S_Name as 'Supervisor Name', ep.HoursWorked as 'Hours' from employeePayroll as ep
inner join employeeInfo as ef on ep.EmpId=ef.EmpId
inner join employeeJob as ej on ej.EmpId=ep.EmpId
inner join JobDetails as jd on jd.JobCode=ej.JobCode
inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id
where ep.HoursWorked>25 and ef.LastName like '%s%';



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'parul_MajorAssignment_1' database. The central pane displays a query window with the following SQL code and results:

```
-->1) Who worked more than 25 hours in any pay period and the employee's last name include the character 's'?  
-->Please print the employee's [employee ID], [full name] (last name + First Name) as [Full Name], [Supervisor Name], [Hours].  
select ep.EmpId, ef.FirstName + ' ' + ef.LastName as 'Full Name', sd.S_Name as 'Supervisor Name', ep.HoursWorked as 'Hours' from employeePayroll as ep  
inner join employeeInfo as ef on ep.EmpId=ef.EmpId  
inner join employeeJob as ej on ej.EmpId=ep.EmpId  
inner join JobDetails as jd on jd.JobCode=ej.JobCode  
inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id  
where ep.HoursWorked>25 and ef.LastName like '%s%'  
-- If we have to avoid duplicacy  
select Distinct(a.EmpId),a.[Full Name],a.Hours,a.[Supervisor Name] from  
(select ep.EmpId,ef.FirstName + ' ' + ef.LastName as 'Full Name', sd.S_Name as 'Supervisor Name', ep.HoursWorked as 'Hours' from employeePayroll as ep  
inner join employeeInfo as ef on ep.EmpId=ef.EmpId  
inner join employeeJob as ej on ej.EmpId=ep.EmpId  
inner join JobDetails as jd on jd.JobCode=ej.JobCode  
inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id  
where ep.HoursWorked>25 and ef.LastName like '%s%' ) as a
```

The results grid shows the following data:

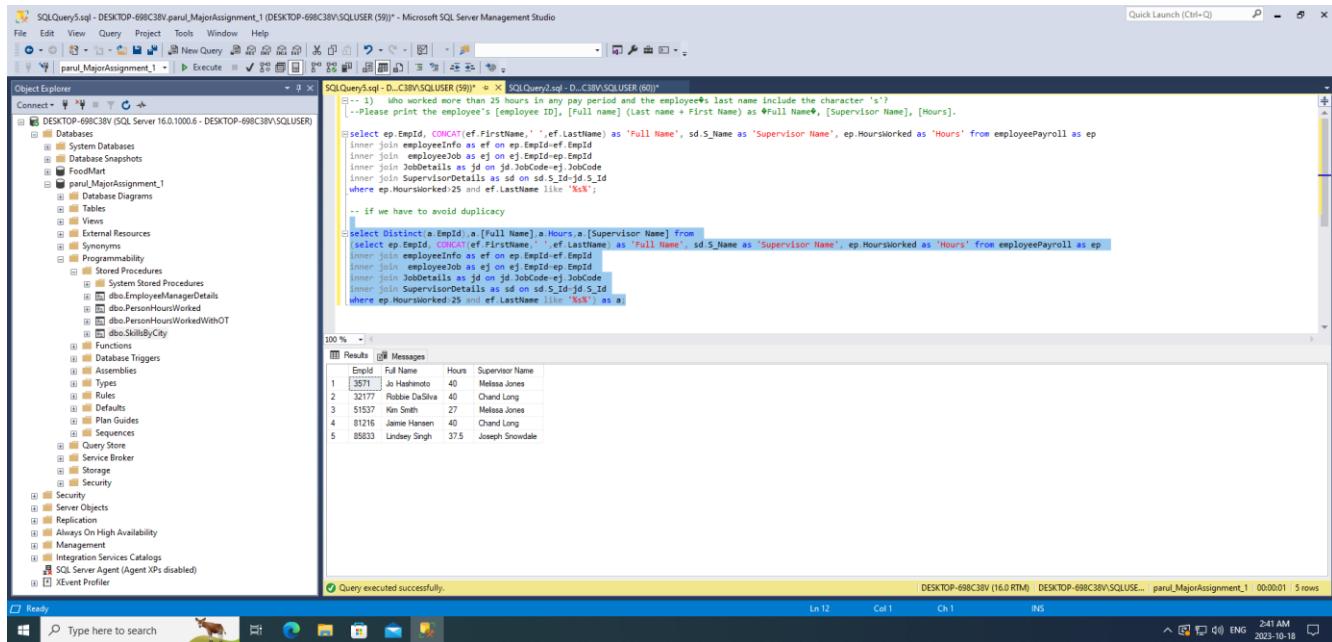
EmpId	Full Name	Supervisor Name	Hours
515537	Ken Smith	Melissa Jones	27
3571	Jo Harimoto	Melissa Jones	40
85833	Lindsay Singh	Joseph Snowdale	37.5
81216	Jeanne Hansen	Chand Long	40
32177	Robbie DaSilva	Chand Long	40
3571	Jo Harimoto	Melissa Jones	40
32177	Robbie DaSilva	Chand Long	40

At the bottom of the results grid, it says 'Query executed successfully.'

Fig 3. Employee who worked more than 35 hours (with duplicate records).

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

→ select Distinct(a.EmpId),a.[Full Name],a.Hours,a.[Supervisor Name] from
 (select ep.EmpId, CONCAT(ep.FirstName,' ',ep.LastName) as 'Full Name', sd.S_Name as 'Supervisor Name', ep.HoursWorked as 'Hours' from employeePayroll as ep
 inner join employeeInfo as ef on ep.EmpId=ef.EmpId
 inner join employeeJob as ej on ej.EmpId=ep.EmpId
 inner join JobDetails as jd on jd.JobCode=ej.JobCode
 inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id
 where ep.HoursWorked>25 and ef.LastName like '%s%'') as a;



```

--> select Distinct(a.EmpId),a.[Full Name],a.Hours,a.[Supervisor Name] from
--> (select ep.EmpId, CONCAT(ep.FirstName,' ',ep.LastName) as 'Full Name', sd.S_Name as 'Supervisor Name', ep.HoursWorked as 'Hours' from employeePayroll as ep
--> inner join employeeInfo as ef on ep.EmpId=ef.EmpId
--> inner join employeeJob as ej on ej.EmpId=ep.EmpId
--> inner join JobDetails as jd on jd.JobCode=ej.JobCode
--> inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id
--> where ep.HoursWorked>25 and ef.LastName like '%s%'') as a;

--> if we have to avoid duplication
--> select Distinct(a.EmpId),a.[Full Name],a.Hours,a.[Supervisor Name] from
--> (select ep.EmpId, CONCAT(ep.FirstName,' ',ep.LastName) as 'Full Name', sd.S_Name as 'Supervisor Name', ep.HoursWorked as 'Hours' from employeePayroll as ep
--> inner join employeeInfo as ef on ep.EmpId=ef.EmpId
--> inner join employeeJob as ej on ej.EmpId=ep.EmpId
--> inner join JobDetails as jd on jd.JobCode=ej.JobCode
--> inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id
--> where ep.HoursWorked>25 and ef.LastName like '%s%' as a

1 2571 Jo Harimoto 40 Melinda Jones
2 32177 Robbie DuGiva 40 Chand Long
3 51537 Kim Smith 27 Melinda Jones
4 81216 Jamie Hansen 40 Chand Long
5 85633 Lindsey Singh 37.5 Joseph Snowdale

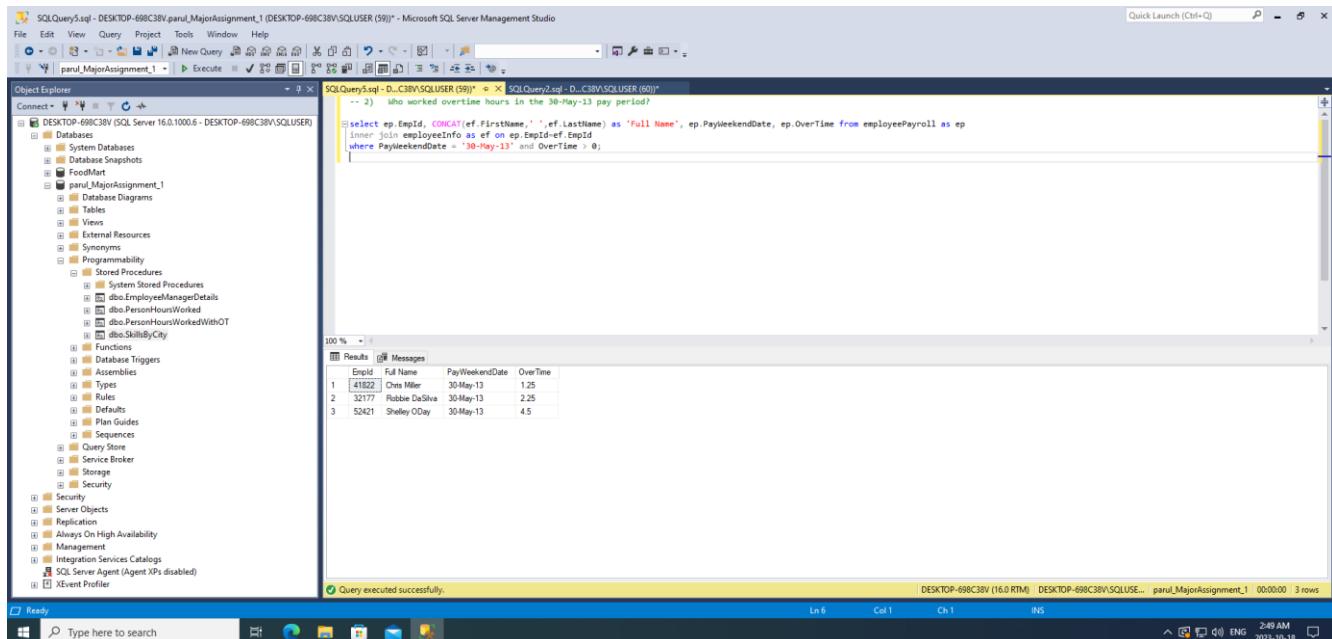
```

Fig 4. Employees who worked more than 25 hours (without duplicate records).

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

2) Who worked overtime hours in the 30-May-13 pay period?

→ select ep.EmpId, CONCAT(ef.FirstName, ' ', ef.LastName) as 'Full Name', ep.PayWeekendDate, ep.OverTime from employeePayroll as ep
inner join employeeInfo as ef on ep.EmpId=ef.EmpId
where PayWeekendDate = '30-May-13' and OverTime > 0;



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'parul_MajorAssignment_1' database. The 'parul_MajorAssignment_1' database node is expanded, showing tables like 'FoodMart', 'dbo.EmployeeManagedDetails', 'dbo.EmployeeHoursWorked', 'dbo.PersonHoursWorkedWithOT', and 'dbo.SkillByCity'. The 'Results' tab in the center displays the query results for employees who worked overtime in the 30-May-13 pay period. The results are as follows:

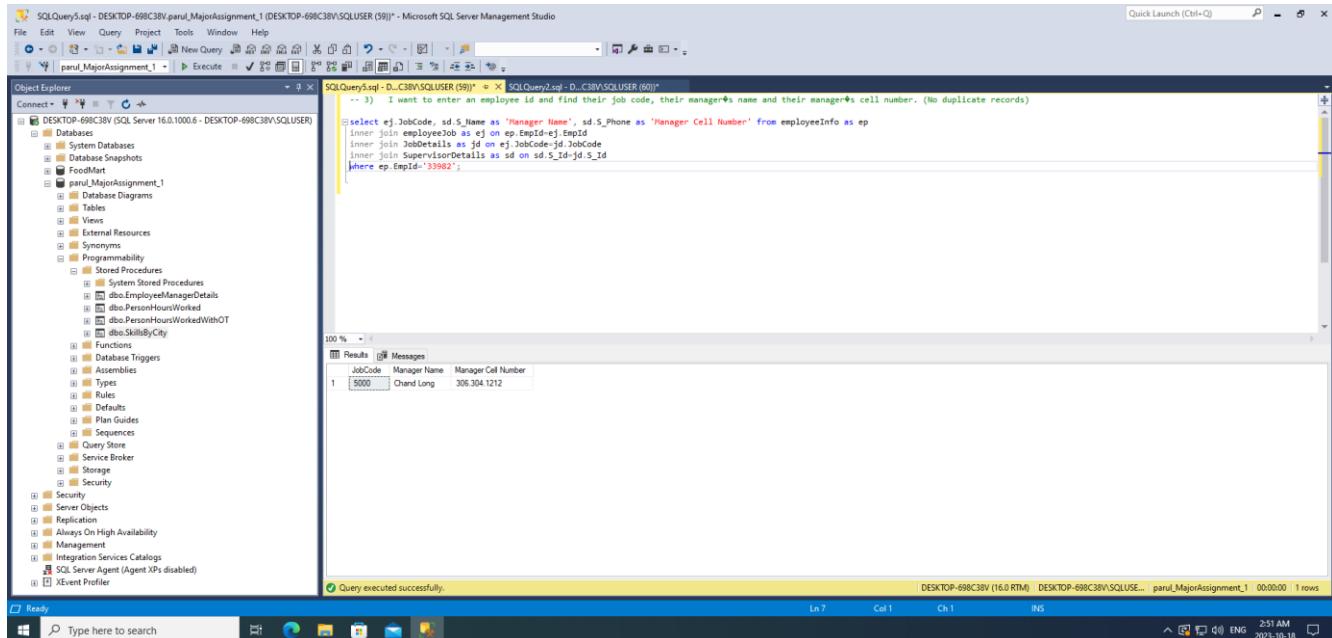
EmpId	Full Name	PayWeekendDate	OverTime
41822	Chris Miller	30-May-13	1.25
32177	Rubbie DaSilva	30-May-13	2.25
52421	Shelley O'Day	30-May-13	4.5

Fig 5. Employee who worked Overtime for the pay period 30-May-13.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

3) I want to enter an employee id and find their job code, their manager's name and their manager's cell number

- select ej.JobCode, sd.S_Name as 'Manager Name', sd.S_Phone as 'Manager Cell Number'
from employeeInfo as ep
inner join employeeJob as ej on ep.EmpId=ej.EmpId
inner join JobDetails as jd on ej.JobCode=jd.JobCode
inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id
where ep.EmpId='33982';



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'parul_MajorAssignment_1' with various objects like tables, stored procedures, and functions. The main window displays a query results grid with the following data:

JobCode	Manager Name	Manager Cell Number
5000	Chand Long	306.304.1212

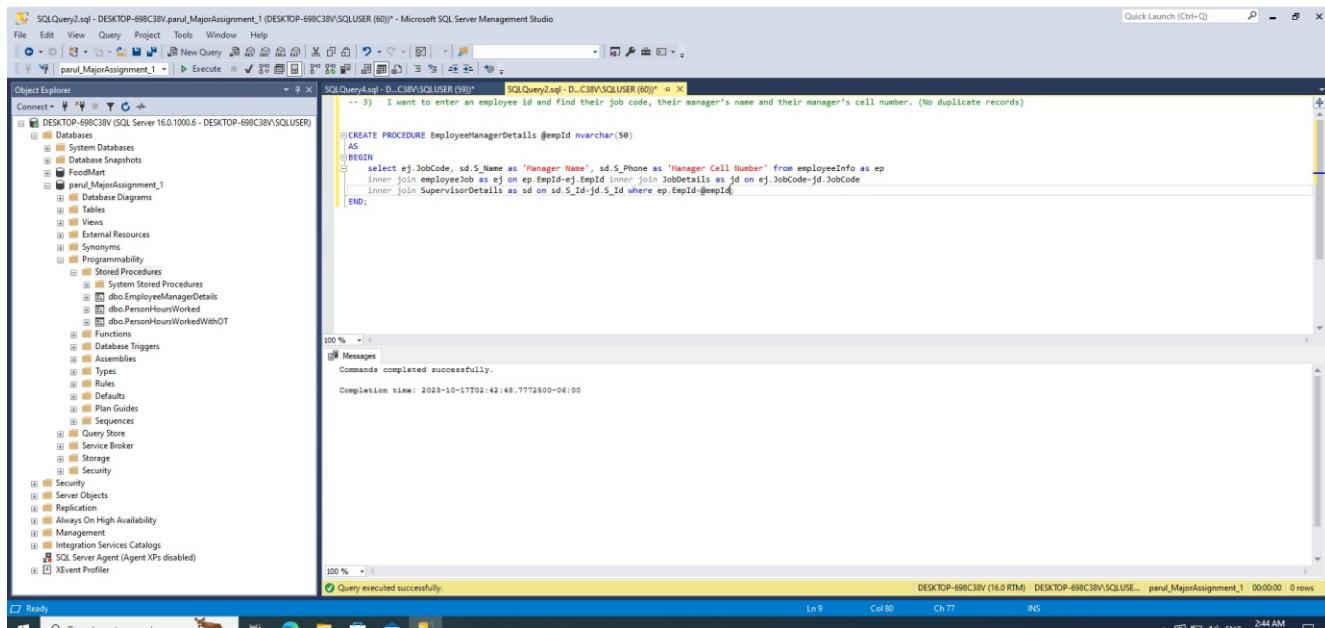
The status bar at the bottom indicates the query was executed successfully.

Fig 5. Query for retrieving JobCode, Managers details based on EmployeeId.

Also, this can be resolved using procedure:

- Create Procedure EmployeeManagerDetails @empId nvarchar(50)
as
BEGIN
 select ej.JobCode, sd.S_Name as 'Manager Name', sd.S_Phone as 'Manager Cell Number'
 from employeeInfo as ep
 inner join employeeJob as ej on ep.EmpId=ej.EmpId
 inner join JobDetails as jd on ej.JobCode=jd.JobCode
 inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id
 where ep.EmpId=@empId;
END;
→ Exec EmployeeManagerDetails @empId='33982';

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)



SQLQuery2.sql - DESKTOP-698C38V.parul_MajorAssignment_1 (DESKTOP-698C38V\SQLUSER (60)) - Microsoft SQL Server Management Studio

Object Explorer

SQLQuery4.sql - DESKTOP-698C38V\SQLUSER (59)* - Microsoft SQL Server Management Studio

CREATE PROCEDURE EmployeeManagerDetails @empId nvarchar(50)

AS

BEGIN

select ej.JobCode, sd.S_Name as 'Manager Name', sd.S_Phone as 'Manager Cell Number' from employeeInfo as ep

inner join employeeJob as ej on ep.EmpId=ej.EmpId inner join JobDetails as jd on ej.JobCode=jd.JobCode

inner join SupervisorDetails as sd on sd.S_Id=jd.S_Id where ep.EmpId=@empId

END;

Messages

Commands completed successfully.

Completion time: 2023-10-17T02:42:49.7772500-06:00

Query executed successfully.

DESKTOP-698C38V (16.0 RTM) DESKTOP-698C38V\SQLUSER... parul_MajorAssignment_1 00:00:00 0 rows

Ready

SQLQuery4.sql - DESKTOP-698C38V.parul_MajorAssignment_1 (DESKTOP-698C38V\SQLUSER (59)) - Microsoft SQL Server Management Studio

Object Explorer

SQLQuery2.sql - DESKTOP-698C38V\SQLUSER (60)* - Microsoft SQL Server Management Studio

exec EmployeeManagerDetails @empId='33982';

Results

JobCode	Manager Name	Manager Cell Number
1	5000	306.304.1212

Query executed successfully.

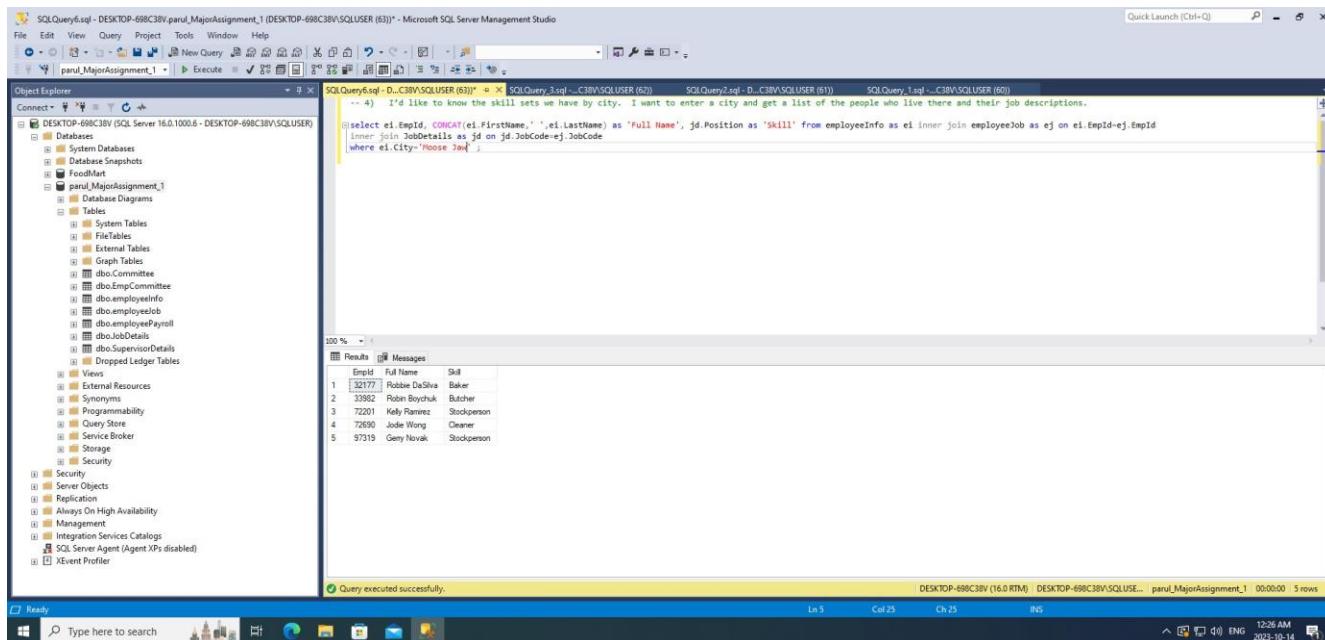
DESKTOP-698C38V (16.0 RTM) DESKTOP-698C38V\SQLUSER... parul_MajorAssignment_1 00:00:00 1 rows

Fig 6. Procedure for retrieving JobCode, Managers details based on EmployeeId with results.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

4) I'd like to know the skill sets we have by city. I want to enter a city and get a list of the people who live there and their job descriptions.

→ select ei.EmpId, CONCAT(ei.FirstName, ',', ei.LastName) as 'Full Name', jd.Position as 'Skill'
from employeeInfo as ei inner join employeeJob as ej on ei.EmpId=ej.EmpId
inner join JobDetails as jd on jd.JobCode=ej.JobCode
where ei.City='Moose Jaw' ;



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'parul_MajorAssignment_1' with various tables and objects. The central pane displays a query results grid with the following data:

Empld	Full Name	Skill
1	32177 Robbie DaSilva	Baker
2	33962 Robin Boychuk	Butcher
3	72201 Kelly Ramirez	Stockperson
4	72690 Jodie Wong	Cleaner
5	97319 Genny Novak	Stockperson

The status bar at the bottom indicates 'Query executed successfully.' and shows the date and time as 2023-10-14 12:26 AM.

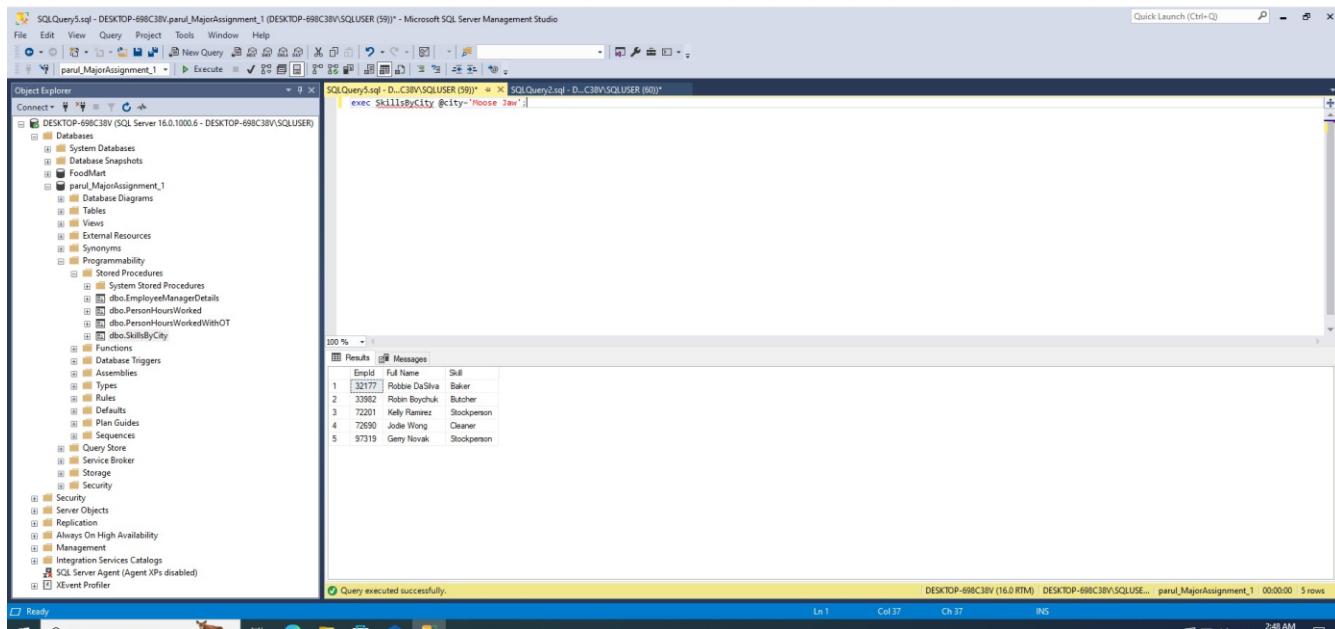
Fig 7. Query for retrieving Person Name and Job Description based on City.

This can also be resolved using a stored procedure.

→ CREATE PROCEDURE SkillsByCity @city nvarchar(50)
AS
BEGIN
select ei.EmpId, CONCAT(ei.FirstName, ',', ei.LastName) as 'Full Name', jd.Position as 'Skill'
from employeeInfo as ei inner join employeeJob as ej on ei.EmpId=ej.EmpId
inner join JobDetails as jd on jd.JobCode=ej.JobCode
where ei.City=@city
END;

→ Exec SkillsByCity @city='Moose Jaw';

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'parul_MajorAssignment_1' database and its tables. The Results tab on the right displays the output of a stored procedure named 'SkillsByCity'. The query executed was 'exec SkillsByCity @city='Moose Jaw''. The results are as follows:

Empld	Full Name	Skill	
1	32177	Robbie DaSilva	Baker
2	33982	Robin Boychuk	Butcher
3	72201	Kelly Ramirez	Stockperson
4	72890	Jodie Wong	Cleaner
5	97319	Gerry Novak	Stockperson

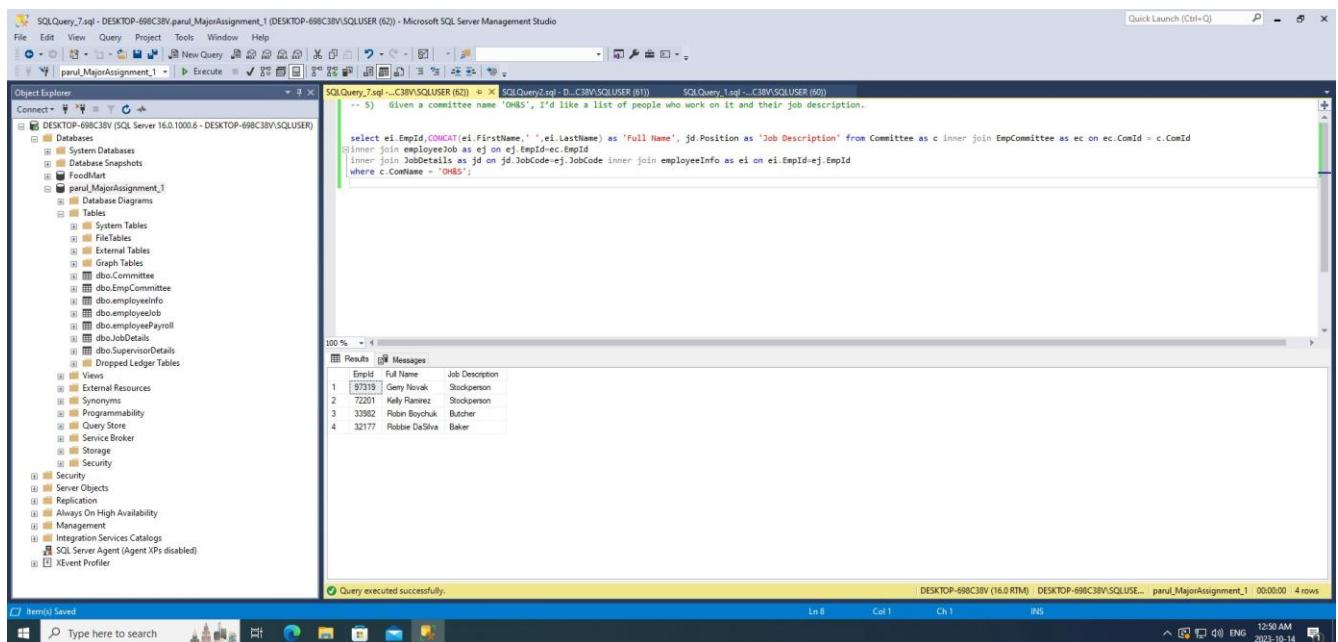
Below the results, a message box indicates 'Query executed successfully.'

Fig 8. Procedure for retrieving Person Name and Job Description based on City with results.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

5) Given a committee name 'OH&S', I'd like a list of people who work on it and their job description.

→ select ei.EmpId,CONCAT(ei.FirstName,' ',ei.LastName) as 'Full Name', jd.Position as 'Job Description' from Committee as c inner join EmpCommittee as ec on ec.ComId = c.ComId inner join employeeJob as ej on ej.EmpId=ec.EmpId inner join JobDetails as jd on jd.JobCode=ej.JobCode inner join employeeInfo as ei on ei.EmpId=ej.EmpId where c.ComName = 'OH&S';



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'parul_MajorAssignment_1' database and its tables: Employee, EmployeeJob, JobDetails, and EmployeeInfo. The SQL Query window on the right contains the following T-SQL code:

```
-- 5) Given a committee name 'OH&S', I'd like a list of people who work on it and their job description.

select ei.EmpId,CONCAT(ei.FirstName,' ',ei.LastName) as 'Full Name', jd.Position as 'Job Description' from Committee as c inner join EmpCommittee as ec on ec.ComId = c.ComId
inner join employeeJob as ej on ej.EmpId=ec.EmpId
inner join JobDetails as jd on jd.JobCode=ej.JobCode inner join employeeInfo as ei on ei.EmpId=ej.EmpId
where c.ComName = 'OH&S';
```

The results grid shows the following data:

EmpId	Full Name	Job Description
1	Gerry Novak	Stockperson
2	Kelly Ramirez	Stockperson
3	Robin Boychuk	Butcher
4	Robbie DaSilva	Baker

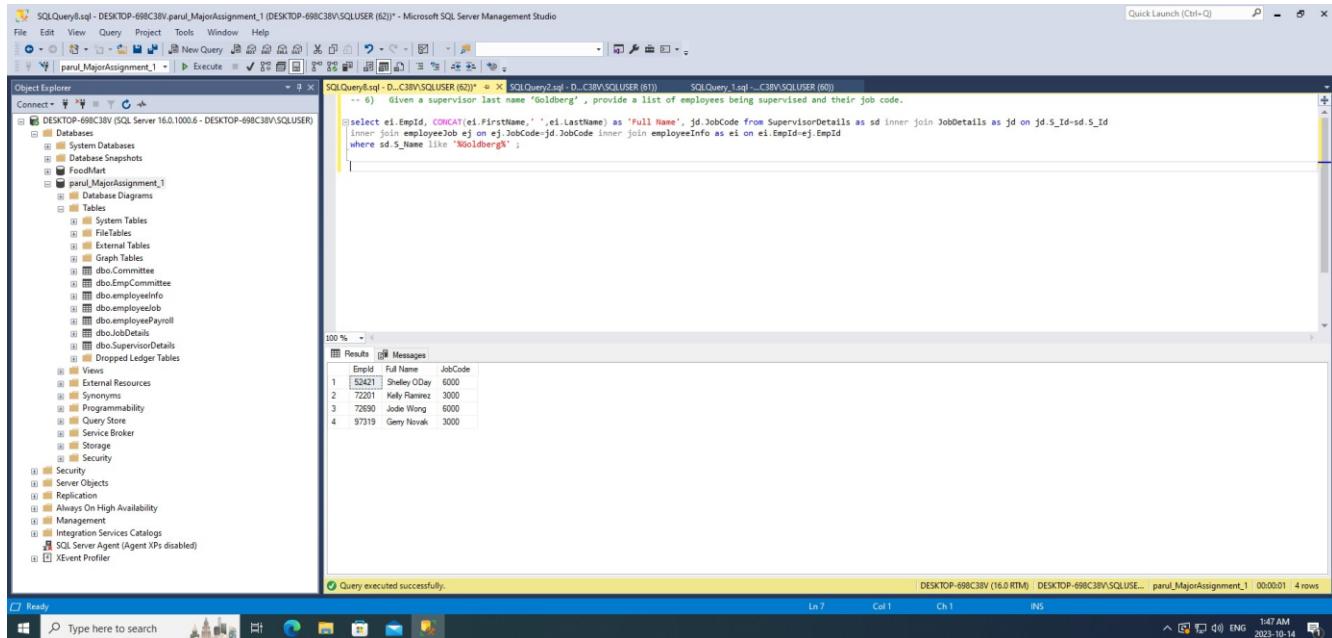
At the bottom of the interface, a message indicates: "Query executed successfully." The status bar shows the session details: DESKTOP-698C38V (16.0 RTM) - DESKTOP-698C38V\SQLUSER - parul_MajorAssignment_1 00:00:00 4 rows. The date and time are also displayed as 2023-10-14 12:50 AM.

Fig 9. Query to retrieve Person Name and Job Description based on Committee Name.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

6) Given a supervisor last name ‘Goldberg’ , provide a list of employees being supervised and their job code.

→ select ei.EmpId, CONCAT(ei.FirstName, ' ', ei.LastName) as 'Full Name', jd.JobCode from SupervisorDetails as sd inner join JobDetails as jd on jd.S_Id=sd.S_Id inner join employeeJob ej on ej.JobCode=jd.JobCode inner join employeeInfo as ei on ei.EmpId=ej.EmpId where sd.S_Name like '%Goldberg%' ;



The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the parul_MajorAssignment_1 database and its tables: EmployeeInfo, EmployeeJob, JobDetails, and SupervisorDetails. The Results tab on the right displays the output of the following query:

```
-- 6) Given a supervisor last name 'Goldberg' , provide a list of employees being supervised and their job code.
select ei.EmpId, CONCAT(ei.FirstName, ' ', ei.LastName) as 'Full Name', jd.JobCode from SupervisorDetails as sd inner join JobDetails as jd on jd.S_Id=sd.S_Id
inner join employeeJob ej on ej.JobCode=jd.JobCode inner join employeeInfo as ei on ei.EmpId=ej.EmpId
where sd.S_Name like '%Goldberg%' ;
```

The results table shows four rows of data:

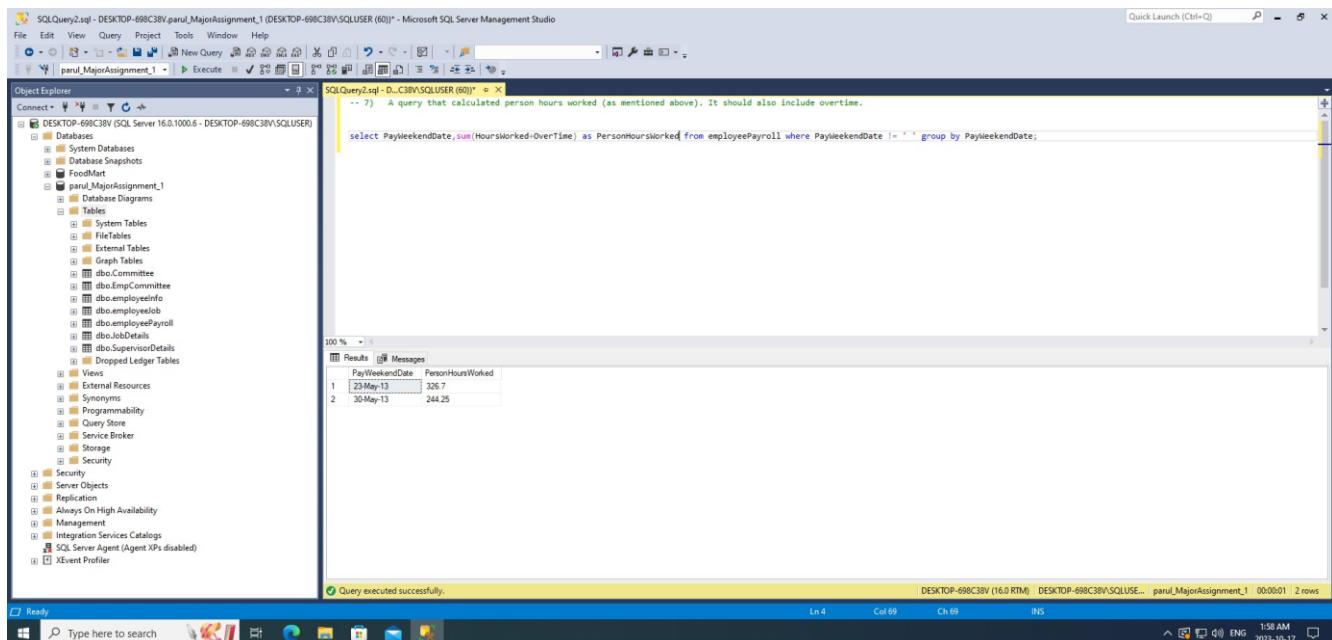
EmpId	Full Name	JobCode
1	Shelley O'Day	6000
2	Kelly Rainier	3000
3	Joe Wong	6000
4	Gerry Novak	3000

Fig 10. Query to retrieve Employee List along with Job Code based on Supervisor LastName.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

7) A query that calculated person hours worked (as mentioned above). It should also include overtime.

→ Select PayWeekendDate, sum(HoursWorked+OverTime) as PersonHoursWorked from employeePayroll where PayWeekendDate != '' group by PayWeekendDate;



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'parul_MajorAssignment_1' database and its tables: employeePayroll, employeeInfo, employeeJob, jobDetails, supervisorDetails, and others. The SQL Query window in the center contains the following query:

```
select PayWeekendDate, sum(HoursWorked+OverTime) as PersonHoursWorked from employeePayroll where PayWeekendDate != '' group by PayWeekendDate;
```

The Results pane on the right displays the output of the query:

PayWeekendDate	PersonHoursWorked
23-May-13	326.7
30-May-13	244.25

Below the results, a message indicates: "Query executed successfully." The status bar at the bottom shows the session details: DESKTOP-698C38V (16.0 RTM) | DESKTOP-698C38V\SQLUSER | parul_MajorAssignment_1 | 00:00:01 | 2 rows.

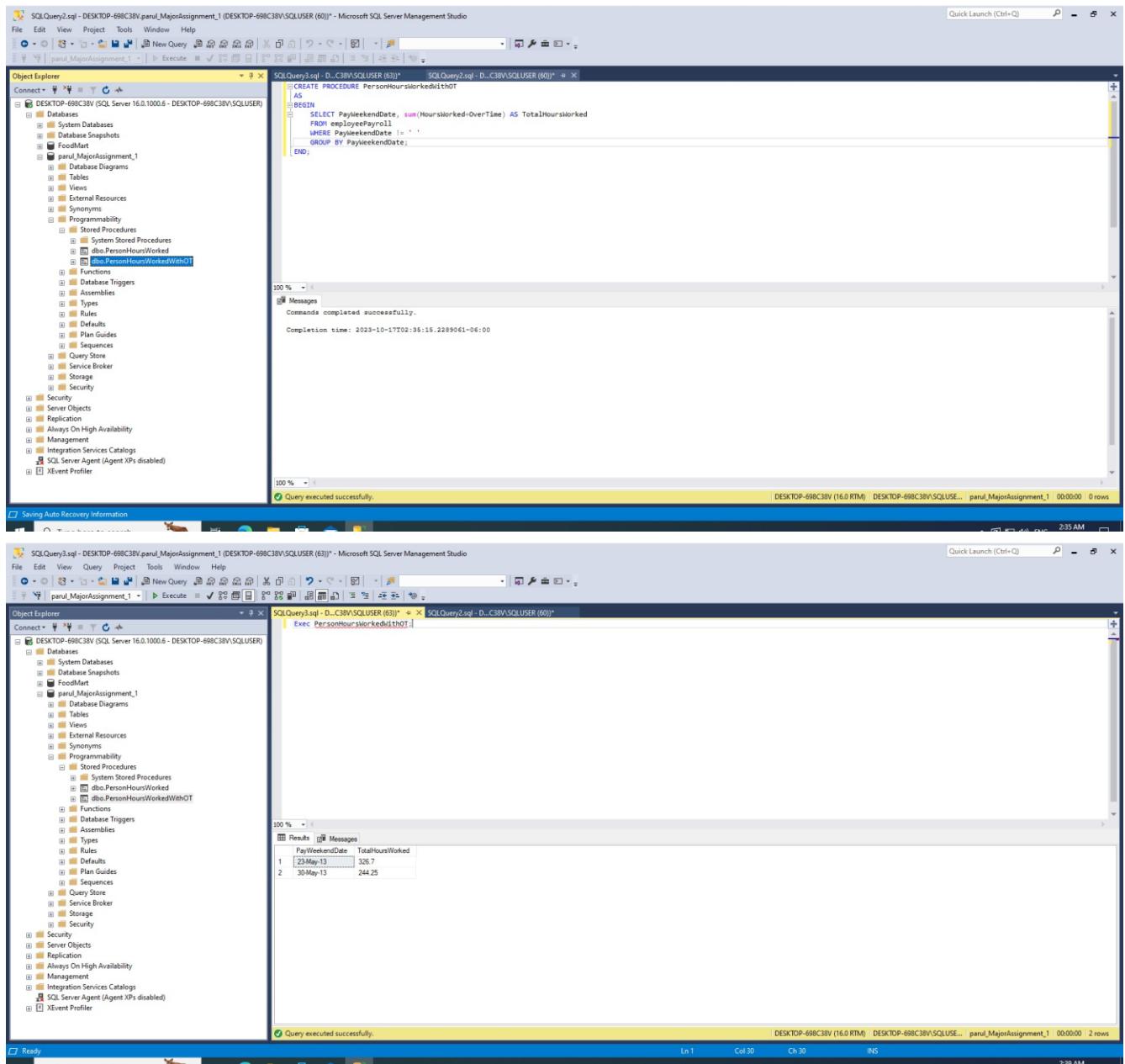
Fig 11. Query to calculate Person Hours Worked Including Overtime.

This can also be resolved using a stored procedure:

→ Create Procedure PersonHoursWorkedWithOT
AS
BEGIN
SELECT PayWeekendDate, sum(HoursWorked+OverTime) AS
TotalHoursWorked FROM employeePayroll where PayWeekendDate != ''
group by PayWeekendDate;
END;

→ Exec PersonHoursWorkedWithOT;

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)



The screenshot shows two windows of Microsoft SQL Server Management Studio (SSMS) running on a Windows 10 desktop.

Top Window (Object Explorer): Shows the database structure for 'parul_MajorAssignment_1'. The 'Stored Procedures' node is expanded, showing 'dbo.PersonHoursWorked' and 'dbo.PersonHoursWorkedWithOT'.

Top Window (SQL Query): Displays the creation script for the stored procedure 'PersonHoursWorkedWithOT'.

```

CREATE PROCEDURE PersonHoursWorkedWithOT
AS
BEGIN
    SELECT PayWeekendDate, sum(HoursWorked+Overtime) AS TotalHoursWorked
    FROM employeePayroll
    WHERE PayWeekendDate = *
    GROUP BY PayWeekendDate;
END;
  
```

Bottom Window (Object Explorer): Shows the database structure for 'DESKTOP-698C38V'. The 'Stored Procedures' node is expanded, showing 'dbo.PersonHoursWorked' and 'dbo.PersonHoursWorkedWithOT'.

Bottom Window (SQL Query): Displays the execution of the stored procedure 'PersonHoursWorkedWithOT' and its results.

```

EXEC PersonHoursWorkedWithOT;
  
```

Results Grid:

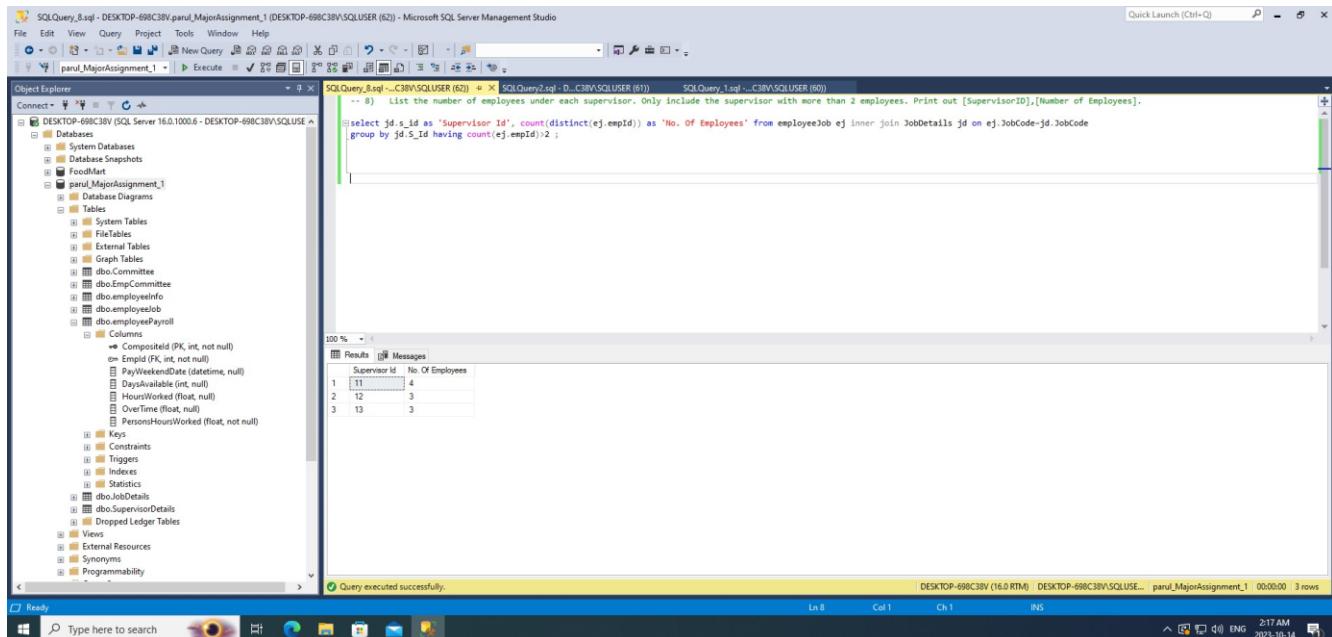
PayWeekendDate	TotalHoursWorked
23-May-13	326.7
30-May-13	244.25

Fig 12. Procedure to calculate Person Hours Worked Including OverTime with results.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

8) List the number of employees under each supervisor. Only include the supervisor with more than 2 employees. Print out [SupervisorID],[Number of Employees].

→ select jd.s_id as 'Supervisor Id', count(distinct(ej.empId)) as 'No. Of Employees' from employeeJob ej inner join JobDetails jd on ej.JobCode=jd.JobCode group by jd.S_Id having count(ej.empId)>2 ;



```
-- 8) List the number of employees under each supervisor. Only include the supervisor with more than 2 employees. Print out [SupervisorID],[Number of Employees].  
select jd.s_id as 'Supervisor Id', count(distinct(ej.empId)) as 'No. Of Employees' from employeeJob ej inner join JobDetails jd on ej.JobCode=jd.JobCode  
group by jd.S_Id having count(ej.empId)>2 ;
```

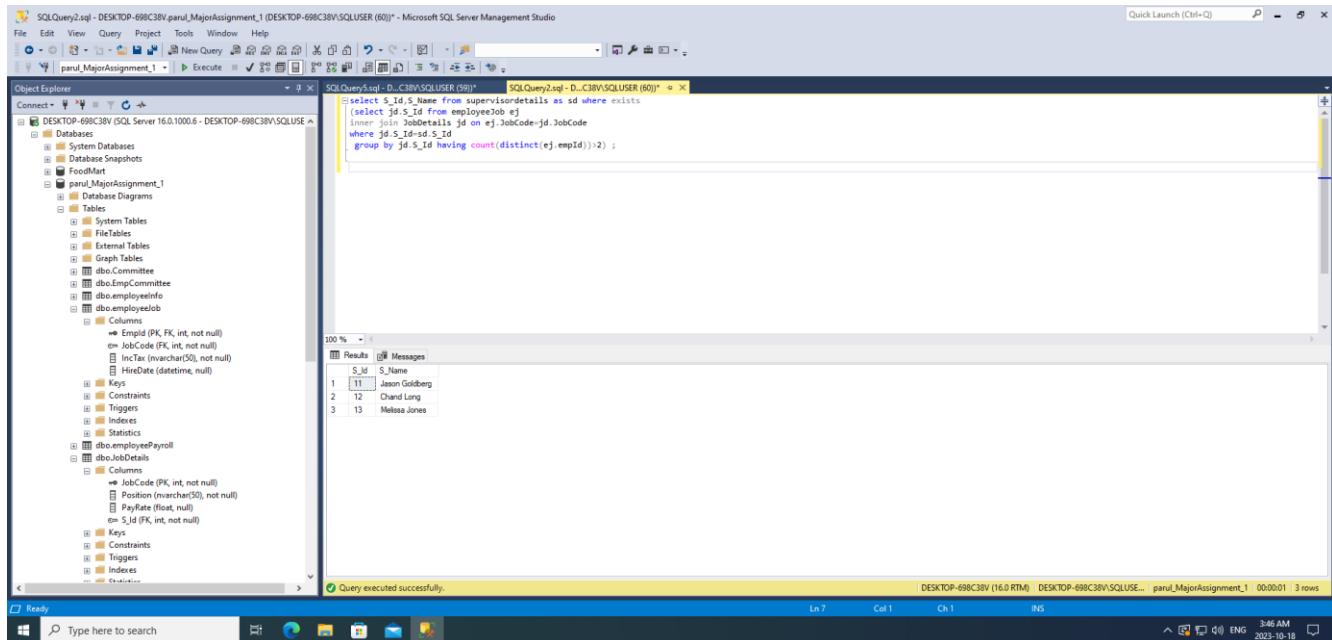
Supervisor Id	No. Of Employees
11	4
12	3
13	3

Fig 13. Query to list employees under each supervisor.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

9) List the supervisor's ID and name, who supervises more than 2 employees. (hint: using EXISTS clause)

→ select S_Id, S_Name from supervisordetails as sd where exists
(select jd.S_Id from employeeJob ej
inner join JobDetails jd on ej.JobCode=jd.JobCode
where jd.S_Id=sd.S_Id
group by jd.S_Id having count(distinct(ej.empId))>2) ;



The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure, including the 'parul_MajorAssignment_1' database and its tables: employeeJob, JobDetails, supervisordetails, and employeePayroll. The 'supervisordetails' table is selected. On the right, the 'Results' tab shows the output of the following query:

```
select S_Id, S_Name from supervisordetails as sd where exists
(select jd.S_Id from employeeJob ej
inner join JobDetails jd on ej.JobCode=jd.JobCode
where jd.S_Id=sd.S_Id
group by jd.S_Id having count(distinct(ej.empId))>2) ;
```

The results table shows three rows:

S_Id	S_Name
11	Jason Goldberg
12	Chand Ling
13	Melissa Jones

At the bottom of the interface, the status bar indicates: DESKTOP-698C38V (16.0 RTM) | DESKTOP-698C38V\SQLUSER | parul_MajorAssignment_1 | 00:00:01 | 3 rows | 3:46 AM | ENG | 2023-10-18.

Fig 14. Query to list supervisors with more than two employees.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)

10) Create a view to show the employee ID, employee's full name, committee ID, meeting night, committee name, supervisor id and supervisor's name, where the condition is the meeting night is 'Tues'. No duplicate records.

```
➔ Create view CommitteeMeetingForDay as
select ei.EmpId,Concat(ei.FirstName, ' ', ei.LastName) as 'Full Name', c.ComId as 'Committee
Id',
c.MeetingNight, c.ComName as 'Committee Name', sd.S_Id as 'Supervisor Id',sd.s_Name as
'Supervisor Name'
from committee c
inner join EmpCommittee ec on c.ComId=ec.ComId
inner join employeeInfo ei on ei.EmpId=ec.EmpId
inner join employeeJob ej on ej.EmpId=ei.EmpId
inner join JobDetails jd on jd.JobCode=ej.JobCode
inner join SupervisorDetails sd on sd.S_Id=jd.S_Id;
➔ Select * from CommitteeMeetingForDay where Meetingnight='Mon';
```

→ Select * from CommitteeMeetingForDay where Meetingnight='Mon';

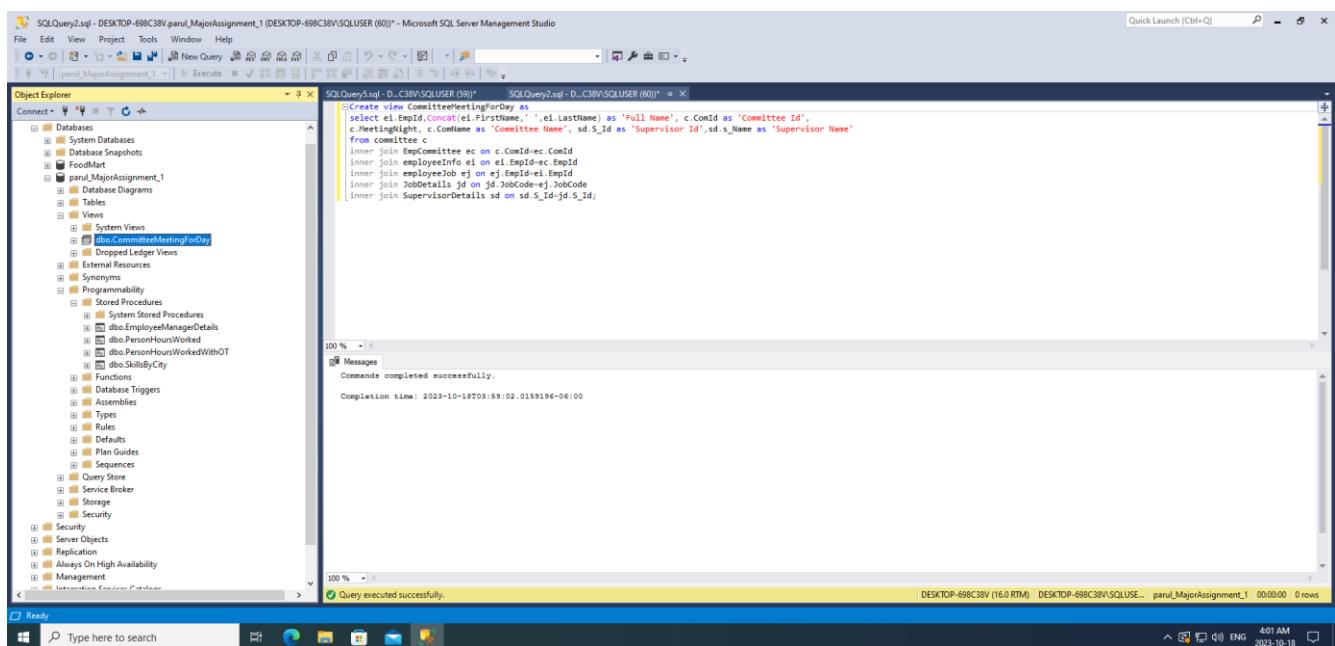
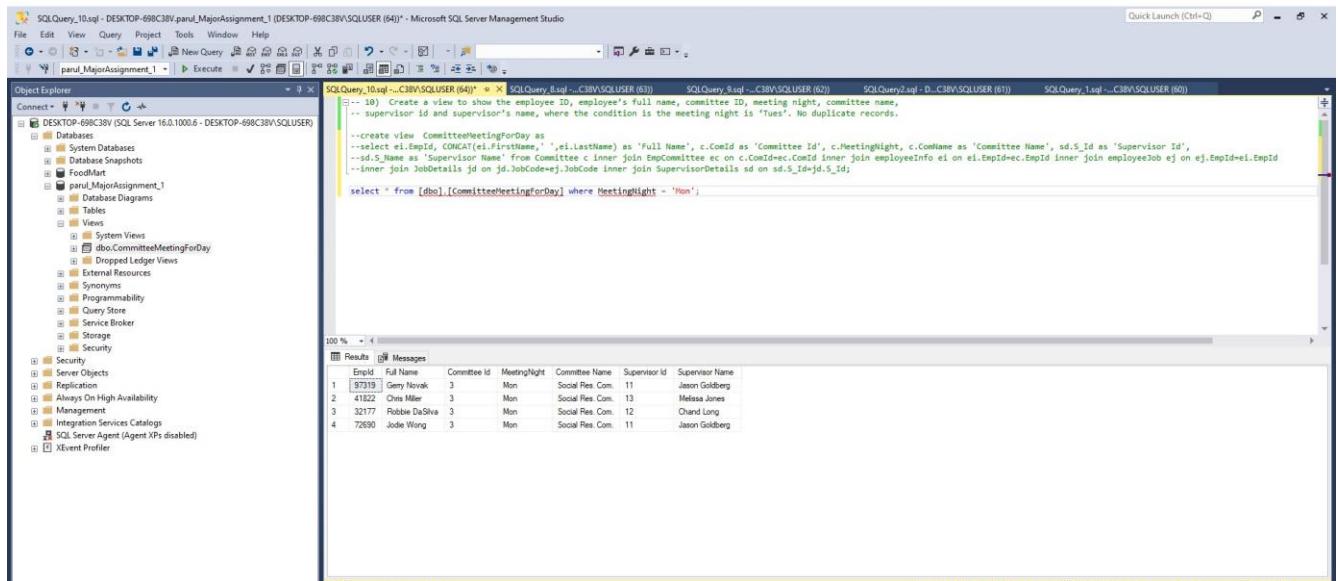


Fig 15. View to show details using MeetingNight filter.

ASSIGNMENT 1 – DATABASE MODELING (NORMALIZATION)



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'parul_MajorAssignment_1' under 'Databases'. The 'Views' node is expanded, showing a view named 'CommitteeMeetingForDay'. The 'Script' node for this view is selected, displaying the T-SQL code for creating the view. The 'Results' tab on the right displays the output of a query that selects data from the 'CommitteeMeetingForDay' view, showing four rows of results.

```
--> 18) Create a view to show the employee ID, employee's full name, committee ID, meeting night, committee name, supervisor id and supervisor's name, where the condition is the meeting night is 'Tues'. No duplicate records.
--create view CommitteeMeetingForDay as
--select ei.EmpId, CONCAT(ei.FirstName, ' ', ei.LastName) as 'Full Name', c.ComId as 'Committee Id', c.MeetingNight, c.CommitteeName as 'Committee Name', sd.S_Id as 'Supervisor Id',
--sd.S_Name as 'Supervisor Name' from Committee c inner join EmpCommittee ec on c.ComId=ec.ComId inner join employeeInfo ei on ei.EmpId=ec.EmpId inner join employeeJob ej on ej.EmpId=ei.EmpId
--inner join JobDetails jd on jd.JobCode=ej.JobCode inner join SupervisorDetails sd on sd.S_Id=jd.S_Id;
--select * from [dbo].[CommitteeMeetingForDay] where MeetingNight = 'Mon';



| EmpId | Full Name      | Committee Id | MeetingNight | Committee Name   | Supervisor Id | Supervisor Name |
|-------|----------------|--------------|--------------|------------------|---------------|-----------------|
| 1     | Gerry Novak    | 3            | Mon          | Social Res. Com. | 11            | Jason Goldberg  |
| 2     | Chris Miller   | 3            | Mon          | Social Res. Com. | 13            | Melissa Jones   |
| 3     | Robbie DaSilva | 3            | Mon          | Social Res. Com. | 12            | Chand Long      |
| 4     | Jodie Wring    | 3            | Mon          | Social Res. Com. | 11            | Jason Goldberg  |


```

Fig 16. Query to fetch details from the View.