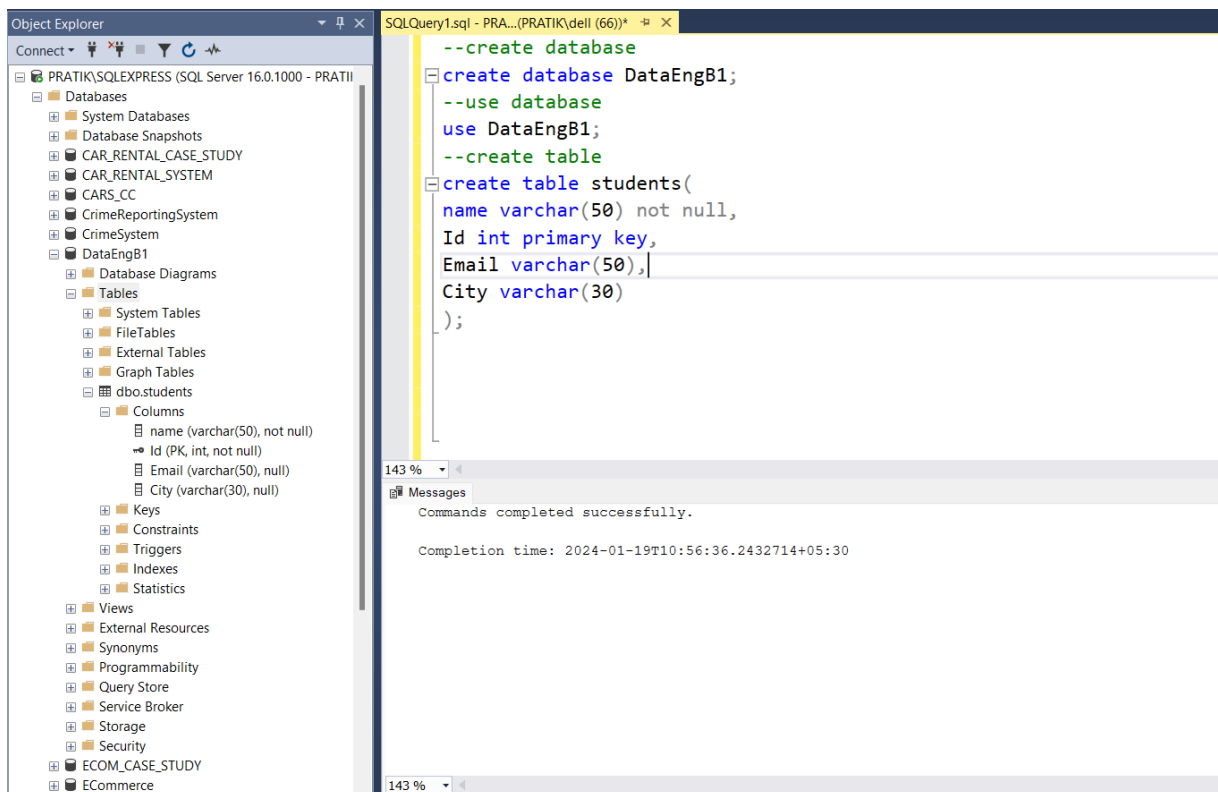


Assignment No: 3

Name: Pratik Wani

- SQL Queries
 - Here I created a database named as 'DataEngB1' and table named as 'Students' and performed all the DDL, DML, TCL, Set Operations and different operators like And, Or etc. Queries on Student table.
 - For join I created new table named as 'Course_info'
- Create database, use database and create table:
 - Create is a DDL command
 - Creating a database doesn't automatically set it as the active database, so we need USE command
 - While Creating Table we need to give all the column names there data types and constraints



- Select all columns and specific columns:

- Using * we can fetch all the columns
- Select is a DML command

The screenshot shows the SQL Server Enterprise Manager interface on the left, displaying the database structure for 'PRATIK\SQLEXPRESS'. The 'dbo.students' table is expanded, showing columns: name (varchar(50), not null), Id (PK, int, not null), Email (varchar(50), null), and City (varchar(30), null). On the right, the SQL Query window shows the following queries:

```
select * from students;
select Id, name from students;
```

The Results pane at the bottom shows the output of the first query, displaying all columns for the 'students' table:

name	Id	Email	City
Amit Patel	1	amit.patel@email.com	Mumbai
Priya Sharma	2	priyasharma@email.com	Delhi
Raj Singh	3	raj.singh@email.com	Jaipur
Neha Gupta	4	neha.gupta@email.com	Bangalore
Aniket Verma	5	aniket.verma@email.com	Hyderabad

- Inserting dummy records:

- Insert is DML command

The screenshot shows the SQL Server Enterprise Manager interface on the left, displaying the database structure for 'PRATIK\SQLEXPRESS'. The 'dbo.students' table is expanded, showing columns: name (varchar(50), not null), Id (PK, int, not null), Email (varchar(50), null), and City (varchar(30), null). On the right, the SQL Query window shows the following queries:

```
--insert command
insert into students (name,Id,Email,City) values
('Amit Patel',1,'amit.patel@email.com','Mumbai'),
('Priya Sharma',2,'priyasharma@email.com','Delhi'),
('Raj Singh',3,'raj.singh@email.com','Jaipur'),
('Neha Gupta',4,'neha.gupta@email.com','Bangalore'),
('Aniket Verma',5,'aniket.verma@email.com','Hyderabad');

select * from students;
```

The Results pane at the bottom shows the output of the second query, displaying all columns for the 'students' table:

name	Id	Email	City
Amit Patel	1	amit.patel@email.com	Mumbai
Priya Sharma	2	priyasharma@email.com	Delhi
Raj Singh	3	raj.singh@email.com	Jaipur
Neha Gupta	4	neha.gupta@email.com	Bangalore
Aniket Verma	5	aniket.verma@email.com	Hyderabad

- Select with distinct:

- If we use distinct with select then it will removed the duplicate records from result set

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATIK)'. The 'dbo.students' table is expanded, showing columns: name (varchar(50), not null), Id (PK, int, not null), Email (varchar(50), null), and City (varchar(30), null). The main window shows a SQL query in 'SQLQuery1.sql' with the following text:

```
--select with distinct
--Here i will add the record having name Raj Singh

insert into students (name,Id,Email,City) values
('Amit Patel',6,'raj.singh2@email.com','Nashik');

select distinct name from students; -- Because of distict we get only one result of Raj Singh
```

Below the query editor, the 'Results' pane shows the output of the query. It contains a table with one column, 'name', and five rows of data:

	name
1	Amit Patel
2	Aniket Verma
3	Neha Gupta
4	Priya Sharma
5	Raj Singh

- **Alter table:**

- Alter command is used to add new column in table, Modify the column in table, Drop the column in table and to add the constraint to any column in a table which is already created
- It is DDL command

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATIK)'. The 'Tables' folder is expanded, showing the 'dbo.students' table. The 'Columns' section for 'students' lists: 'name (varchar(50), not null)', 'Id (PK, int, not null)', 'Email (varchar(50), null)', and 'City (varchar(30), null)'. The 'Keys' section shows a primary key on 'Id'. The 'Constraints', 'Triggers', 'Indexes', and 'Statistics' sections are also visible.

On the right, the SQL Query window shows the following commands:

```
--Alter table
alter table students add salary int;

alter table students drop column salary;

alter table students add salary int;
```

Below the query window, the 'Results' tab shows the data for the 'students' table:

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	NULL
2	Priya Sharma	2	priyasharma@email.com	Delhi	NULL
3	Raj Singh	3	raj.singh@email.com	Jaipur	NULL
4	Neha Gupta	4	neha.gupta@email.com	Bangalore	NULL
5	Aniket Verma	5	aniket.verma@email.com	Hyderabad	NULL
6	Amit Patel	6	raj.singh2@email.com	Nashik	NULL

- Renaming the table and updating the values:
 - In SQL server to rename the database or table we use EXEC sp_rename command. It is a DDL command
 - To update any column values in column we used Update and Set command
It is DML command

The screenshot shows the SQL Server Enterprise Manager interface on the left and a SQL Query window on the right. The Object Explorer on the left shows the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATII)'. The SQL Query window contains the following T-SQL code:

```
--rename table name  
  
exec sp_rename 'students', 'student_info';  
  
--update data  
update student_info  
set salary=50000;  
  
select * from student_info;
```

Below the query window, the 'Results' tab shows the output of the query, displaying a table with 6 rows and 5 columns: name, Id, Email, City, and salary.

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Raj Singh	3	raj.singh@email.com	Jaipur	50000
4	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
5	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
6	Amit Patel	6	raj.singh2@email.com	Nashik	50000

- Delete and where:

- Delete is used to delete the specific record from the table and it is DML command
- To choose the specific record we use the where clause

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATIK)'. The 'Tables' folder is expanded, showing 'dbo.student_info'. The main query window, titled 'SQLQuery1.sql - PRA...(PRATIK\dell (66))', contains the following SQL code:

```
-- delete specific row using where clause
delete from student_info where id=6;

select * from student_info;
```

Below the query window, the 'Results' tab is active, displaying a table with 5 rows and 5 columns: name, Id, Email, City, and salary. The first row, 'Amit Patel', is highlighted.

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Raj Singh	3	raj.singh@email.com	Jaipur	50000
4	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
5	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000

- Group by Having and Order by:

- Group by is used to group the records based on some criteria like we can group the data city wise, salary wise.
- Having clause is used with Group by clause we can perform aggregate functions using Having clause
- Where is different from Having as we cant perform the aggregate conditions using Where clause.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATIK)'. The 'Tables' folder is expanded, showing 'dbo.student_info'. The main window shows a SQL query in 'SQLQuery1.sql' with the following text:

```
--group by having and order by clause  
  
select city, sum(salary) as citywise_total_salary from student_info  
group by city having sum(salary)>90000 order by citywise_total_salary;
```

Below the query editor, the 'Results' tab is active, showing a table with 5 rows and 2 columns: 'city' and 'citywise_total_salary'.

	city	citywise_total_salary
1	Bangalore	100000
2	Delhi	100000
3	Hyderabad	100000
4	Jaipur	100000
5	Mumbai	120000

■ Begin Transaction, Commit and Rollback:

- Transactional control commands are only used with the DML Commands such as - INSERT, UPDATE and DELETE.
- Begin Transaction is used to begin the new transaction block
- Commit is used to save the changes.
- Rollback is used to roll back the changes

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATIK)'. The 'Tables' folder is expanded, showing 'dbo.student_info'. The main window shows a SQL script in 'SQLQuery1.sql' with the following commands:

```
--begin transaction and commit
begin transaction;

insert into student_info (name,Id,Email,City,salary) values
('Pratik Wani',13,'pratik.wani@gmail.com','Nashik',50000);

commit;

--rollback
begin transaction;
delete from student_info where id=13;

rollback;
commit;

select * from student_info;
```

Below the script, the 'Results' tab shows the output of the final 'select' statement. The table has 5 columns: name, Id, Email, City, and salary. It contains 13 rows of data, including the newly inserted record for Pratik Wani.

	name	Id	Email	City	salary
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Raj Singh	3	raj.singh@email.com	Jaipur	50000
4	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
5	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
6	Sara Khan	6	sara.khan@email.com	Mumbai	50000
7	Vikram Sharma	7	vikram.sharma@email.com	Delhi	50000
8	Preeti Singh	8	preeti.singh@email.com	Jaipur	50000
9	Amit Joshi	9	amit.joshi@email.com	Bangalore	50000
10	Diya Verma	10	diya.verma@email.com	Hyderabad	50000
11	Samira Khan	11	samira.khan@email.com	Mumbai	20000
12	Vikram jagtap	12	vikram.jagtap@email.com	Nashik	80000
13	Pratik Wani	13	pratik.wani@gmail.com	Nashik	50000

- Save Transaction and Rollback Transaction (save point and rollback to):
 - Save Transaction is used to create points within the groups of transactions in which to ROLLBACK.
 - To rollback to specific save point we use Rollback Transaction save_point_name

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATIK)'. The main window shows a SQL query script in 'SQLQuery1.sql'. The script performs a series of deletions on the 'student_info' table, saving each step as a transaction point (sp1, sp2, sp3), and then rolls back to sp1. The results pane at the bottom shows the current state of the 'student_info' table after the rollback.

```
begin transaction;

delete from student_info where id=13;
save transaction sp1;

delete from student_info where id=12;
save transaction sp2;

delete from student_info where id=11;
save transaction sp3;

rollback transaction sp1;--here now the changes occurred after sp1 will undo back

select * from student_info;
```

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Raj Singh	3	raj.singh@email.com	Jaipur	50000
4	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
5	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
6	Sara Khan	6	sara.khan@email.com	Mumbai	50000
7	Vikram Sharma	7	vikram.sharma@email.com	Delhi	50000
8	Preeti Singh	8	preeti.singh@email.com	Jaipur	50000
9	Amit Joshi	9	amit.joshi@email.com	Bangalore	50000
10	Divya Verma	10	divya.verma@email.com	Hyderabad	50000
11	Samira Khan	11	samira.khan@email.com	Mumbai	20000
12	Vikram Jagtap	12	vikram.jagtap@email.com	Nashik	80000

Set operations:

■ Union and Intersect:

- UNION will be used to combine the result of two select statements.
- Duplicate rows will be eliminated from the results obtained after performing the UNION operation
- It is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements.

The screenshot shows the SQL Server Enterprise Manager interface. On the left is the Object Explorer showing the database structure. The main window displays a SQL query script with comments explaining the UNION and INTERSECT operations. Below the script, the Results pane shows two tables of data.

SQL Query Script:

```
-- Set Operations
--union
select * from student_info where id between 1 and 5 -- gives record for id's 1 to 5
union
select * from student_info where id between 3 and 7-- gives record for id's 3 to 7

--because of using union it will add both results and it will add the repeated rows ONLY ONCE

--intersection
select * from student_info where id between 1 and 5 -- gives record for id's 1 to 5
intersect
select * from student_info where id between 3 and 7-- gives record for id's 3 to 7

--because of using intersect it will select only the repeated rows
```

Results - UNION:

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
3	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
4	Priya Sharma	2	priyasharma@email.com	Delhi	50000
5	Raj Singh	3	raj.singh@email.com	Jaipur	50000
6	Sara Khan	6	sara.khan@email.com	Mumbai	50000
7	Vikram Sharma	7	vikram.sharma@email.com	Delhi	50000

Results - INTERSECT:

	name	Id	Email	City	salary
1	Raj Singh	3	raj.singh@email.com	Jaipur	50000
2	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
3	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000

- Union all and Except (Minus):

- UNION ALL operator combines all the records from both the queries.
- Duplicate rows will be not be eliminated from the results obtained after performing the UNION ALL operation.
- It displays the rows which are present in the first query but absent in the second query with no duplicates.

The screenshot shows the SQL Server Enterprise Manager interface. On the left is the Object Explorer showing the database structure. The main window displays a SQL query in a query editor. The query uses UNION ALL and EXCEPT operators. Below the query editor, the Results pane shows the output of the queries as two tables.

SQL Query:

```
--union all
select * from student_info where id between 1 and 5 -- gives record for id's 1 to 5
union all
select * from student_info where id between 3 and 7-- gives record for id's 3 to 7

--because of using union all it will add both results and it will not remove the repeated rows

--except (minus)
select * from student_info where id between 1 and 5 -- gives record for id's 1 to 5
except
select * from student_info where id between 3 and 7-- gives record for id's 3 to 7

--It displays the rows which are present in the first query but
--absent in the second query with no duplicates.
```

Results:

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Raj Singh	3	raj.singh@email.com	Jaipur	50000
4	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
5	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
6	Raj Singh	3	raj.singh@email.com	Jaipur	50000
7	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
8	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
9	Sara Khan	6	sara.khan@email.com	Mumbai	50000
10	Vikram Sharma	7	vikram.sharma@email.com	Delhi	50000

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000

- Operators:

- In, Between, And, Like, Not like:

- In - TRUE if the operand is equal to one of a list of expressions.
- Between - TRUE if the operand is within a range.
- Like - TRUE if the operand matches a pattern.
- Not - Reverses the value of any other Boolean operator.

The screenshot displays the SQL Server Enterprise Manager interface on the left, showing the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATIK)'. The 'Tables' folder is expanded, showing 'dbo.student_info'. The main window shows the SQL Query Editor with the following queries:

```
--in operator
select * from student_info where id in(1,2,11,78);

--between operator
select * from student_info where salary between 45000 and 60000;

--like
select name from student_info where name like '%A';--ends with A

--not like
select name from student_info where name not like '%A%';--not contains A
```

The Results pane shows the output of the queries. The first query returns 3 rows, the second returns 10 rows, and the third returns 5 rows.

name	Id	Email	City	salary
Amit Patel	1	amit.patel@email.com	Mumbai	50000
Priya Sharma	2	priyasharma@email.com	Delhi	50000
Samira Khan	11	samira.khan@email.com	Mumbai	20000

name	Id	Email	City	salary
Raj Singh	3	raj.singh@email.com	Jaipur	50000
Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
Sara Khan	6	sara.khan@email.com	Mumbai	50000
Vikram Shar...	7	vikram.sharma@email...	Delhi	50000
Preeti Singh	8	preeti.singh@email.com	Jaipur	50000
Amit Joshi	9	amit.joshi@email.com	Bangalore	50000
Divya Verma	1...	divya.verma@email.com	Hyderabad	50000

name
Priya Sharma
Neha Gupta
Aniket Verma
Vikram Shar...
Divya Verma

name
Preeti Singh

- Or, All, Any:

- Or - TRUE if either Boolean expression is TRUE.
- All - TRUE if all of a set of comparisons are TRUE.
- Any - TRUE if any one of a set of comparisons is TRUE.

The screenshot shows the SQL Server Enterprise Manager interface. On the left is the Object Explorer showing the database structure. The main window displays a SQL query in the 'SQL_ass3.sql' file. The query uses the 'OR', 'ALL', and 'ANY' operators to filter data from the 'student_info' table. Below the query, the 'Results' pane shows the output of the queries, displaying columns: name, Id, Email, City, and salary.

Object Explorer: PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATII)

- Databases
 - System Databases
 - Database Snapshots
 - CAR_RENTAL_CASE_STUDY
 - CAR_RENTAL_SYSTEM
 - CARS_CC
 - CrimeReportingSystem
 - CrimeSystem
 - DataEngB1
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.student_info
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Service Broker
 - Storage
 - Security
- ECOM_CASE_STUDY
- ECommerce
- interview_practice_hexa
- practice
- SCHOOL
- SISDB
- TechShop
- VAG
- VIRTUAL_ART_GALLERY
- Security
- Server Objects
- Replication

SQL Query:

```
--or
select * from student_info where id=10 or id=1;

--all
select * from student_info where name=all
(select name from student_info)

--any
select * from student_info where name=any
(select name from student_info)
```

Results:

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Divya Verma	10	divya.verma@email.com	Hyderabad	50000

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Raj Singh	3	raj.singh@email.com	Jaipur	50000
4	Neha Gupta	4	neha.gupta@email.com	Banga...	50000
5	Aniket Verma	5	aniket.verma@email.co...	Hyder...	50000
6	Sara Khan	6	sara.khan@email.com	Mumbai	50000
7	Vikram Shar...	7	vikram.sharma@email....	Delhi	50000
8	Preeti Singh	8	preeti.singh@email.com	Jaipur	50000
9	Amit Joshi	9	amit.joshi@email.com	Banga...	50000
10	Divya Verma	1...	divya.verma@email.com	Hyder...	50000
11	Samira Khan	1...	samira.khan@email.com	Mumbai	20000
12	Vikram jagtap	1...	vikram.jagtap@email.c...	Nashik	80000

- Exists and Some:

- Exists - TRUE if a subquery contains any rows.
- Some - It is issued with comparison operators (<,>,<=,>=) to compare the value with the result of a subquery.

The screenshot displays the SQL Server Enterprise Manager interface on the left and a SQL Query window on the right. The Enterprise Manager shows a tree view of the database structure, including databases, tables, and views. The SQL Query window contains two queries: one using the EXISTS operator and another using the SOME operator. Below the queries, the Results pane shows the output of the queries, which is a table with columns: name, Id, Email, City, and salary. The table contains 9 rows of data.

```
--exists
select * from student_info where exists
(select name from student_info where id=100);--return nothing as subquery return none

select * from student_info where exists
(select name from student_info where id=11)--return whole table as subquery result is not empty

--some
select * from student_info where id<some(select id from student_info where city='Delhi');
```

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Raj Singh	3	raj.singh@email.com	Jaipur	50000
4	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
5	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
6	Sara Khan	6	sara.khan@email.com	Mumbai	50000
7	Vikram Sharma	7	vikram.sharma@email.com	Delhi	50000
8	Preeti Singh	8	preeti.singh@email.com	Jaipur	50000
9	Amit Joshi	9	amit.joshi@email.com	Bangalore	50000

	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Raj Singh	3	raj.singh@email.com	Jaipur	50000
4	Neha Gupta	4	neha.gupta@email.com	Bangalore	50000
5	Aniket Verma	5	aniket.verma@email.com	Hyderabad	50000
6	Sara Khan	6	sara.khan@email.com	Mumbai	50000

- Join and Joins with Group by and Having:

- Join is used to query across multiple tables and retrieve the data which we need but the conditions is that there must be at least one matching column in both the tables
- We can also use Group by and Having with join to Group the records and apply some condition on group data.

The screenshot displays the SQL Server Enterprise Manager interface on the left, showing the database structure for 'PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATIK)'. The 'dbo' schema is expanded, showing tables 'student_info' and 'course_info'. The 'student_info' table structure is detailed, including columns: name (varchar(50), not null), Id (PK, int, not null), Email (varchar(50), null), City (varchar(30), null), salary (int, null), and course_id (FK, int, null). The 'course_info' table structure is also shown, including columns: course_id (PK, int, not null), c_name (varchar(50), not null), and salary (int, null).

The SQL Query window on the right shows two queries. The first query is a simple inner join between 'student_info' and 'course_info' on the 'course_id' column. The second query is a more complex join with group by and having clauses, selecting the city and the sum of salaries, grouped by city, and filtered by a sum of salaries greater than 45000.

```
--Joins
select * from student_info
inner join course_info on
student_info.course_id=course_info.c_id;

--join with group by and having
select student_info.city,sum(salary) as total_sal_by_city from student_info
inner join course_info on
student_info.course_id=course_info.c_id
group by student_info.city
having sum(salary)>45000;
```

The Results window shows the output of the second query, displaying a table with two columns: 'city' and 'total_sal_by_city'. The results are as follows:

city	total_sal_by_city
Bangalore	100000
Delhi	100000
Hyderabad	100000
Jaipur	100000
Mumbai	120000
Nashik	80000

- Index:

- Index will boost the performance of the retrieval
- An index basically hints to the database that a particular column is important, and should be used to help sort and filter through the data

