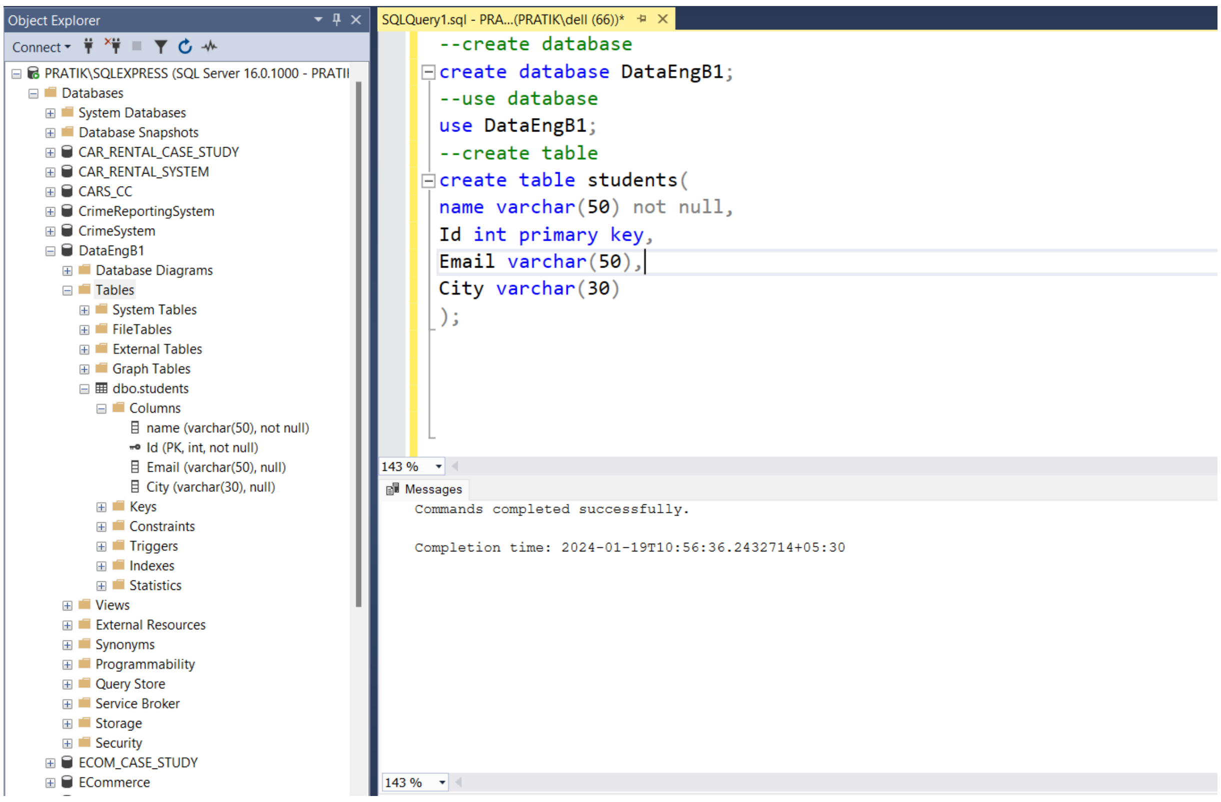
Assignment No: 3

Name: Pratik Wani

- SQL Queries

  - Here I created a database named as 'DataEndB1' 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

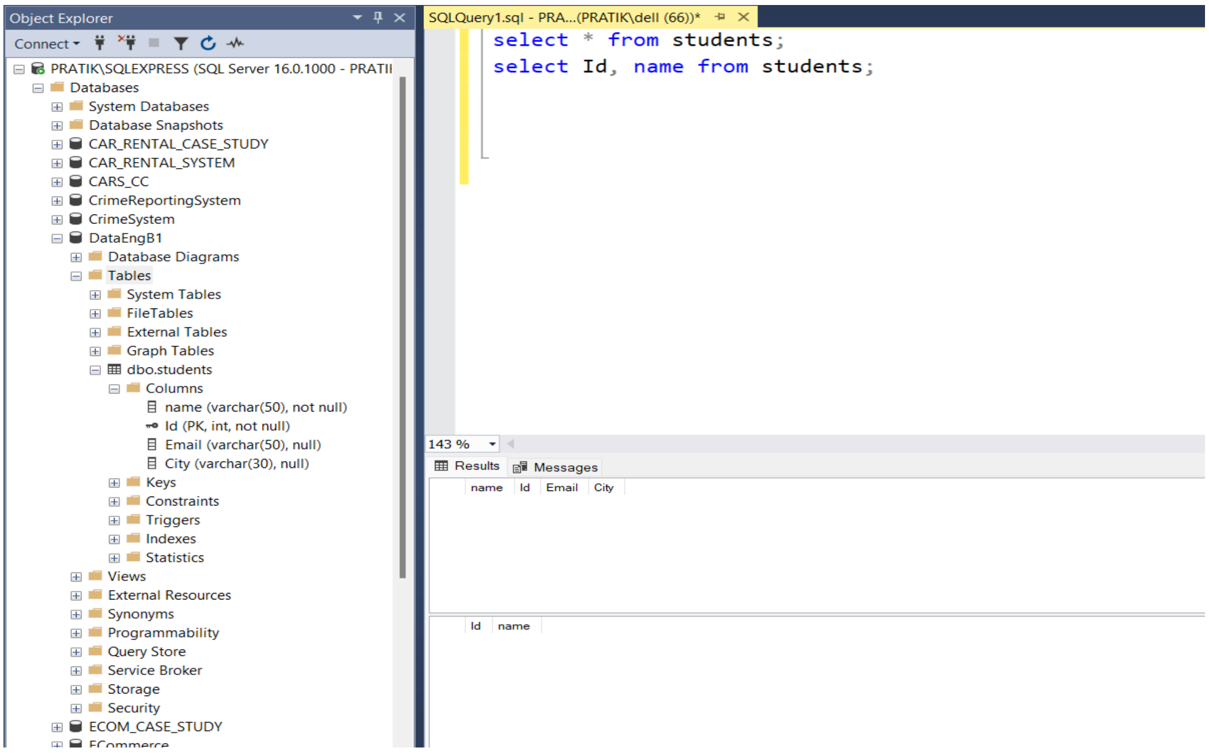


```
--create database
create database DataEngB1;
--use database
use DataEngB1;
--create table
create table students(
name varchar(50) not null,
Id int primary key,
Email varchar(50),
City varchar(30)
);
```

```
Commands completed successfully.

Completion time: 2024-01-19T10:56:36.2432714+05:30
```

Object Explorer
SQLQuery1.sql - PRA...(PRATIK\dell (66))*
PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATII
Databases
DataEngB1
Tables
dbo.students
Columns
name (varchar(50), not null)
Id (PK, int, not null)
Email (varchar(50), null)
City (varchar(30), null)
Messages
143 %

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

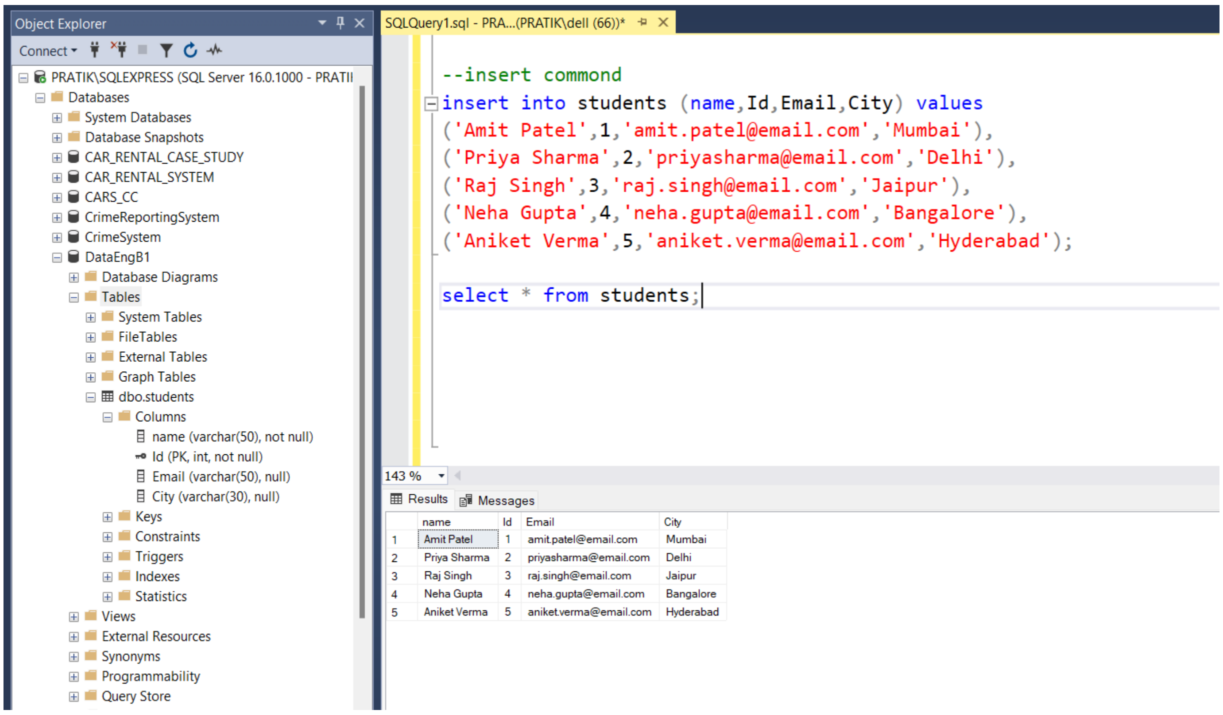


```
select * from students;
select Id, name from students;
```

name	Id	Email	City

Id	name

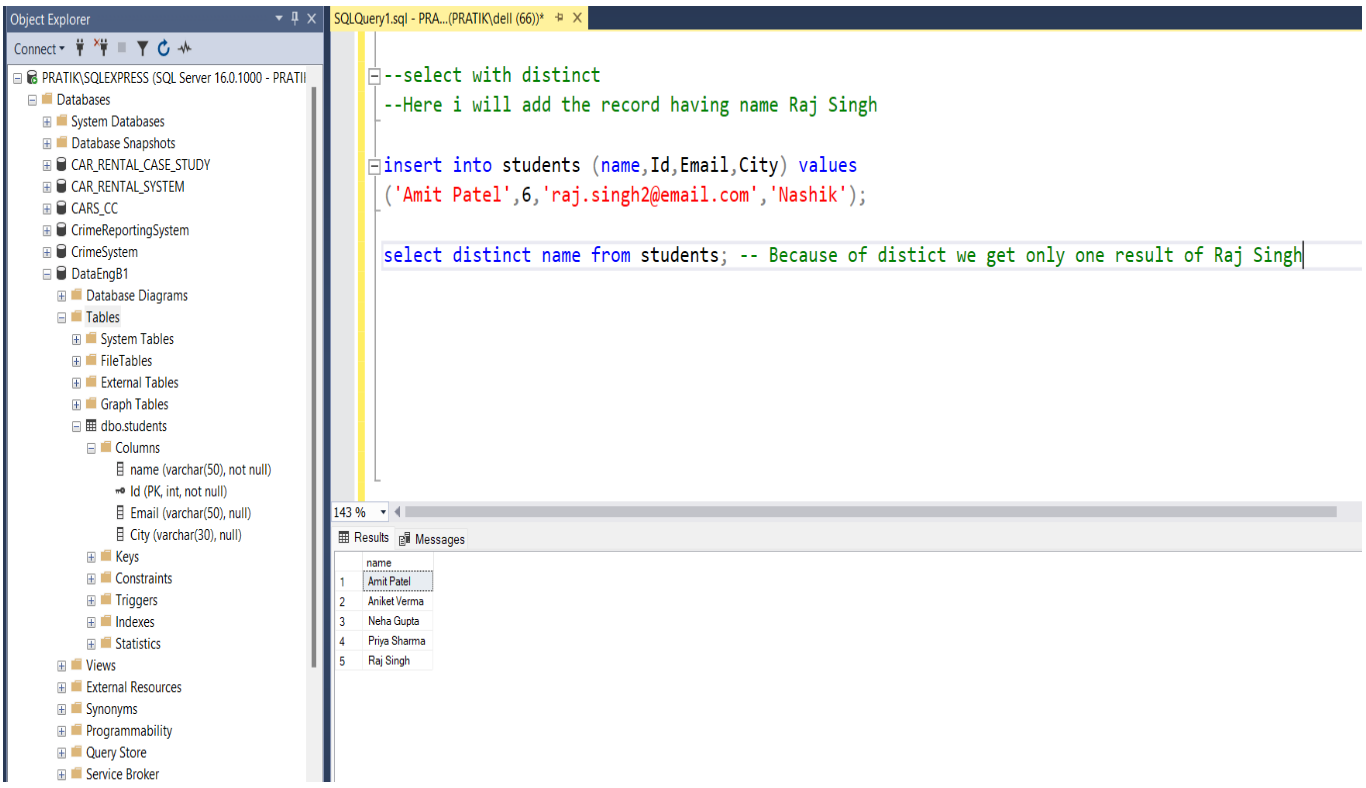
- Inserting dummy records:
    - Insert is DML command



```
--insert commond
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;
```

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

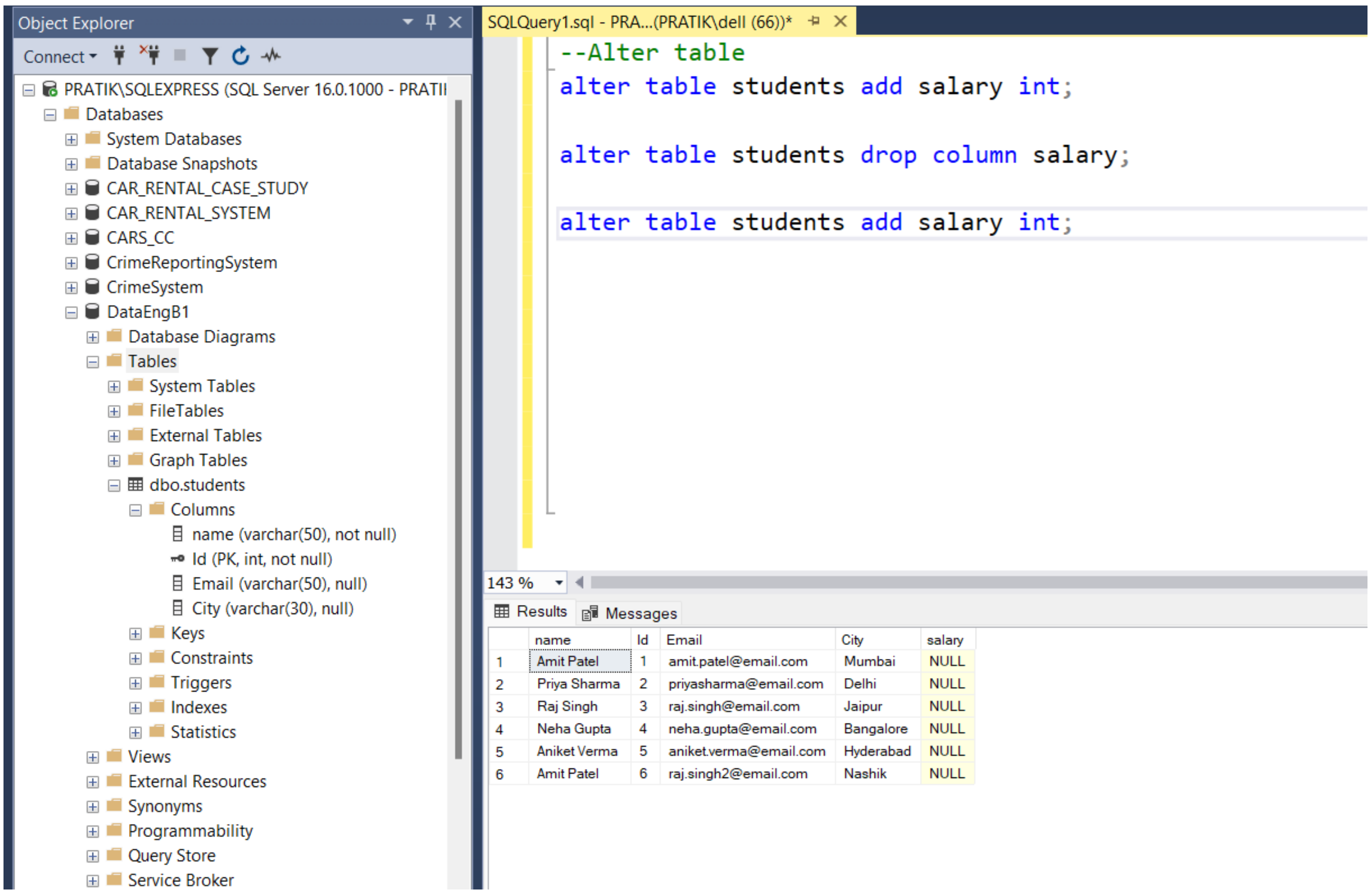
- Select with distinct:
    - If we use distinct with select then it will removed the duplicate records from result set



```
--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
```

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

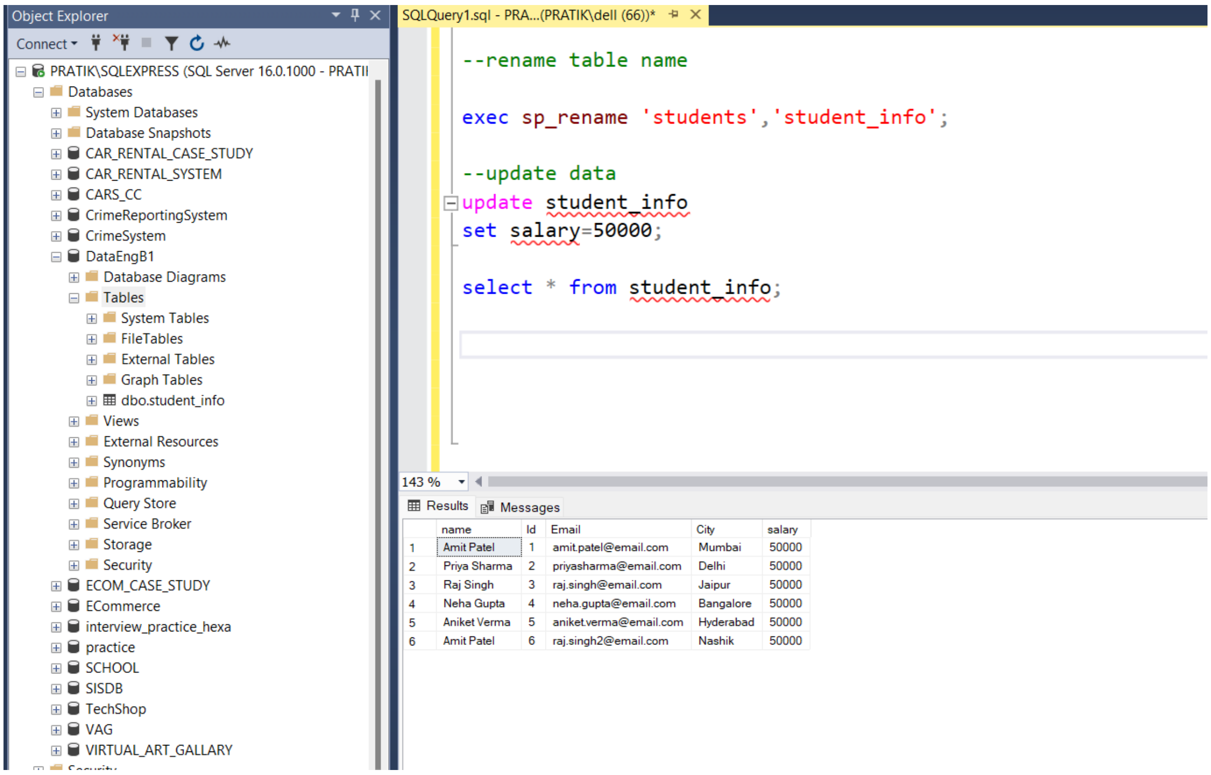
- Alter table:
    - Alter command is used to add new column in table, Modify the column in table, Drop the column in table and to add the constraint to any column in a table which is already created
    - It is DDL command



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

alter table students drop column salary;

alter table students add salary int;
```

	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

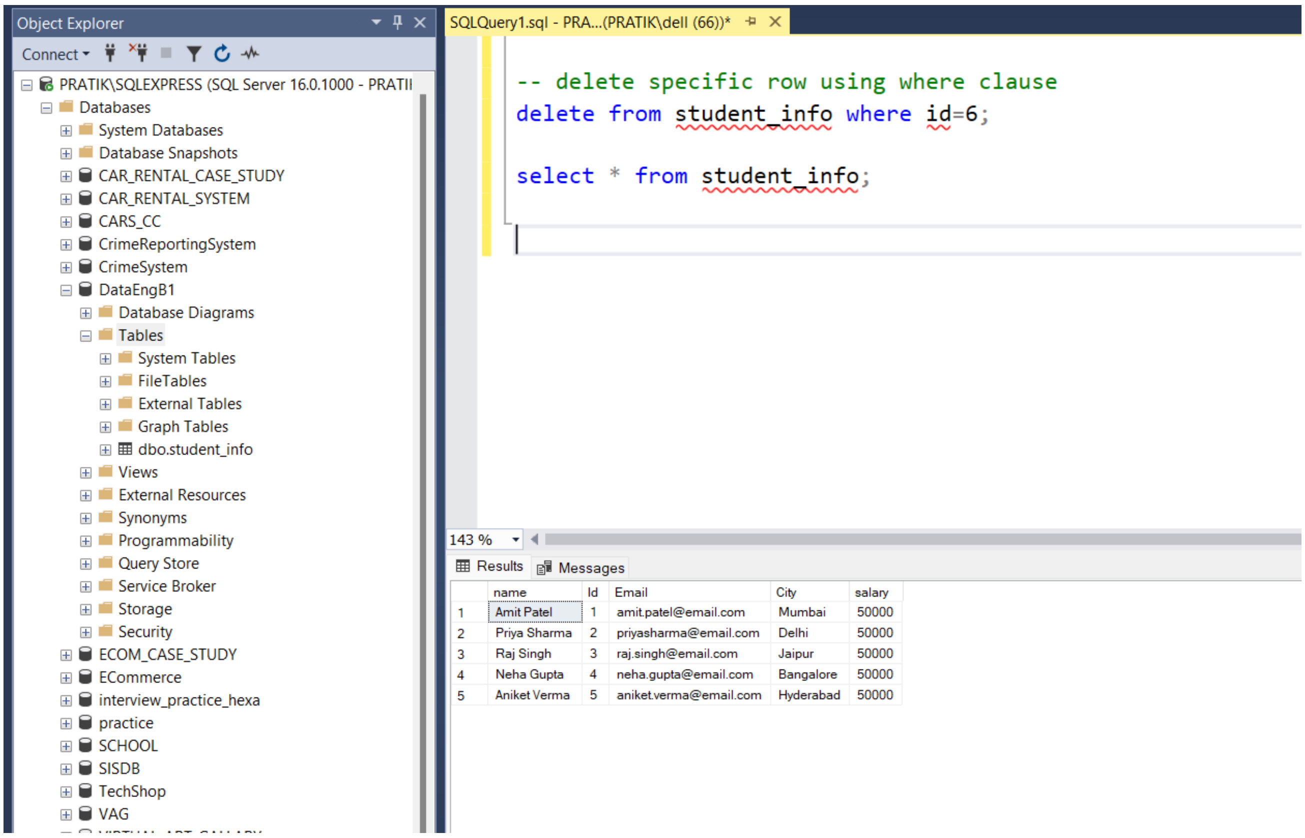
- Renaming the table and updating the values:
    - In SQL server to rename the database or table we use EXEC sp_rename command. It is a DDL command
    - To update any column values in column we used Update and Set command It is DML command



```
--rename table name

exec sp_rename 'students','student_info';

--update data
update student_info
set salary=50000;

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	Amit Patel	6	raj.singh2@email.com	Nashik	50000

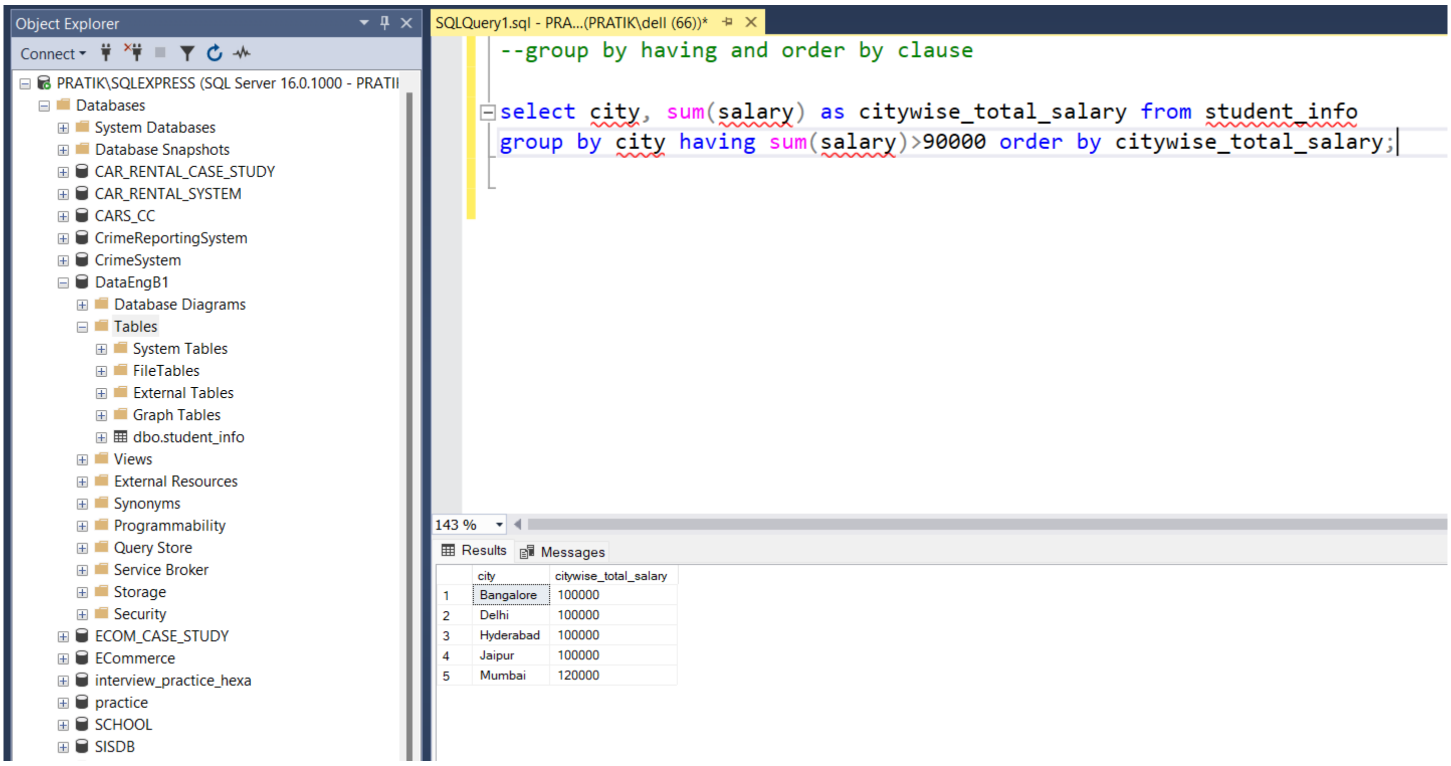
- Delete and where:
    - Delete is used to delete the specific record from the table and it is DML command
    - To choose the specific record we use the where clause



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

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

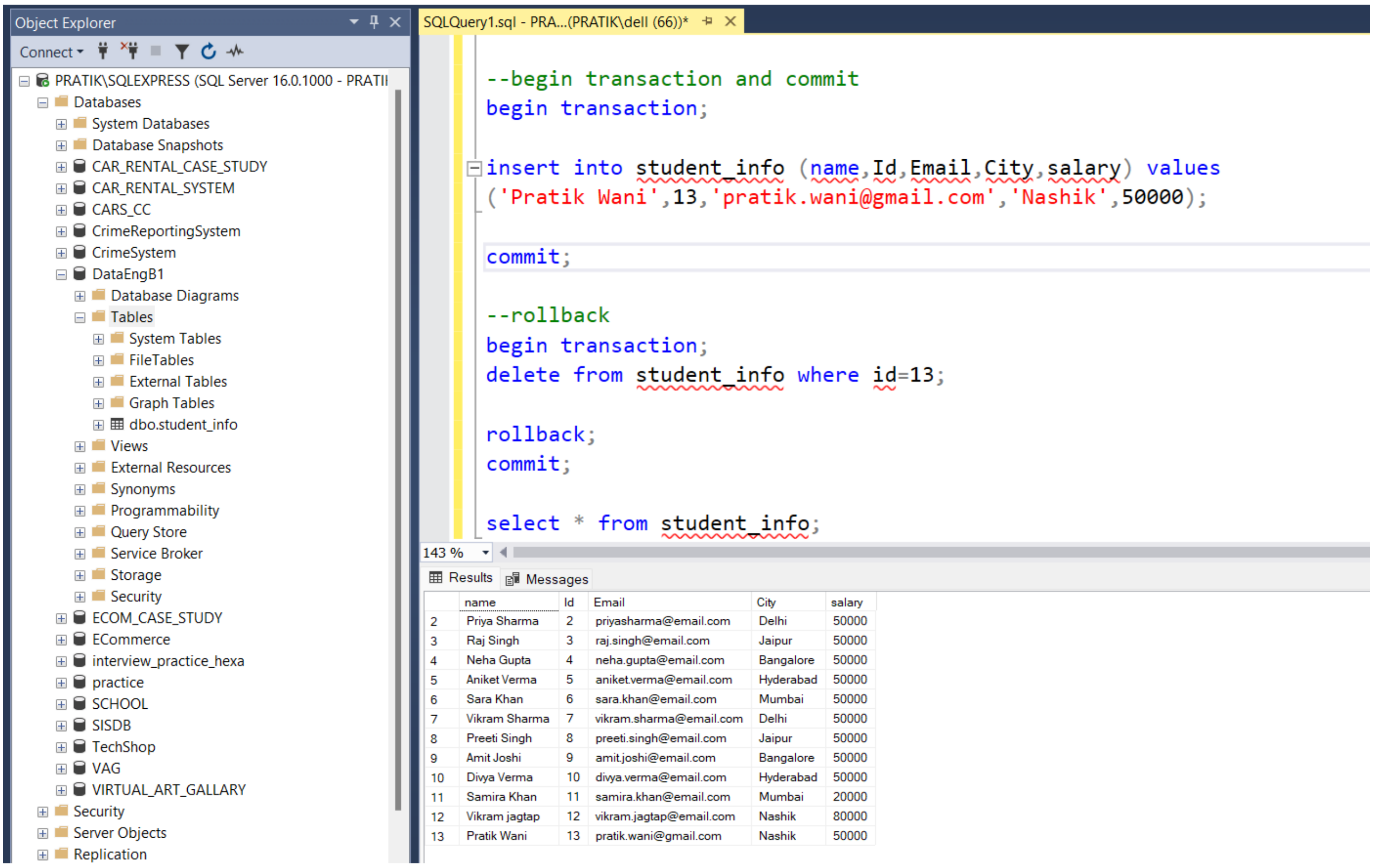
- **Group by Having and Order by:**
  - Group by is used to group the records based on some criteria like we can group the data city wise, salary wise.
  - Having clause is used with Group by clause we can perform aggregate functions using Having clause
  - Where is different from Having as we cant perform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



```
--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;
```

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

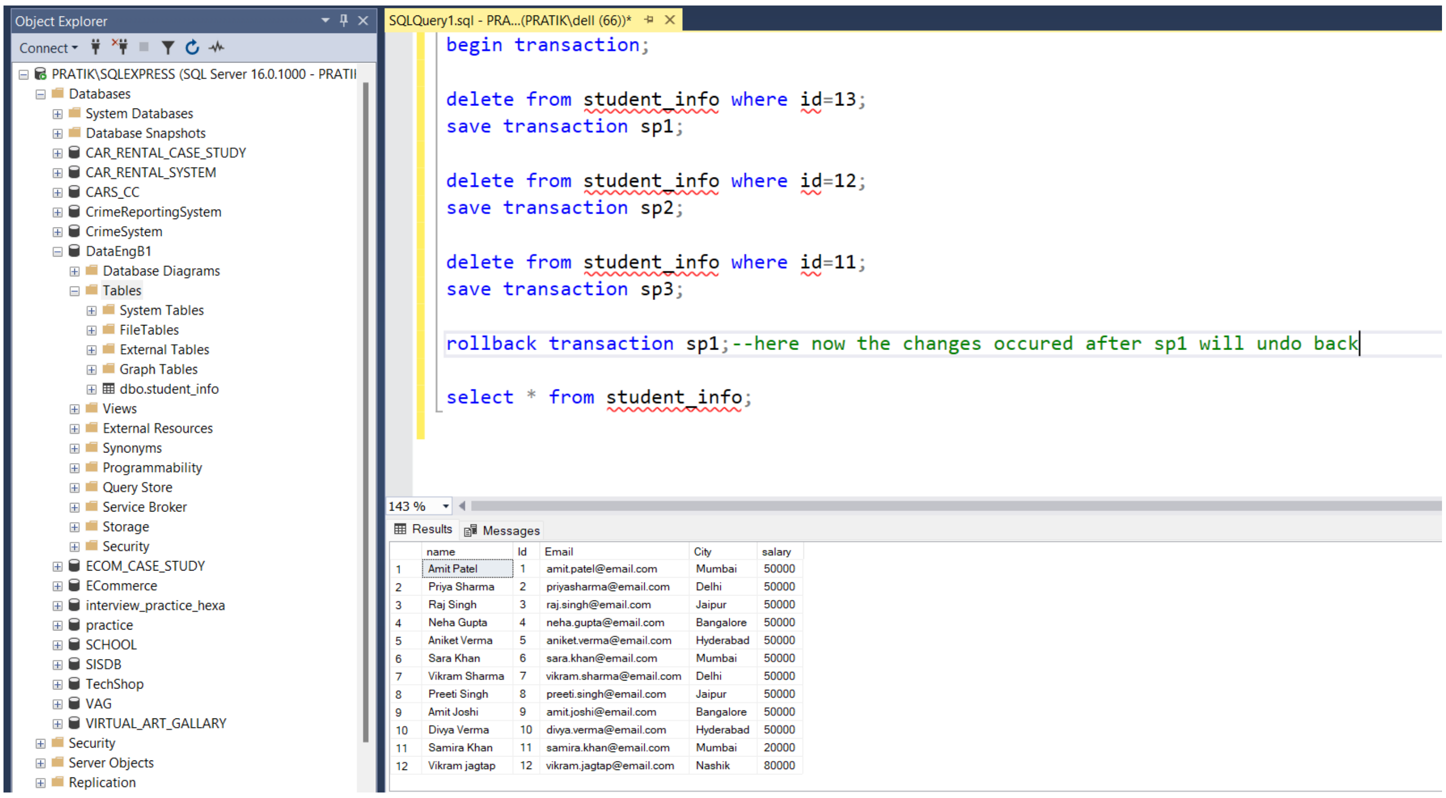
- Begin Transaction, Commit and Rollback:
    - Transactional control commands are only used with the DML Commands such as - INSERT, UPDATE and DELETE.
    - Begin Transaction is used to begin the new transaction block
    - Commit is used to save the changes.
    - Rollback is used to roll back the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



```sql
--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;
```

	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	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
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 creates points within the groups of transactions in which to ROLLBACK.
    - To rollback to specific save point we use Rollback Transaction save point_name

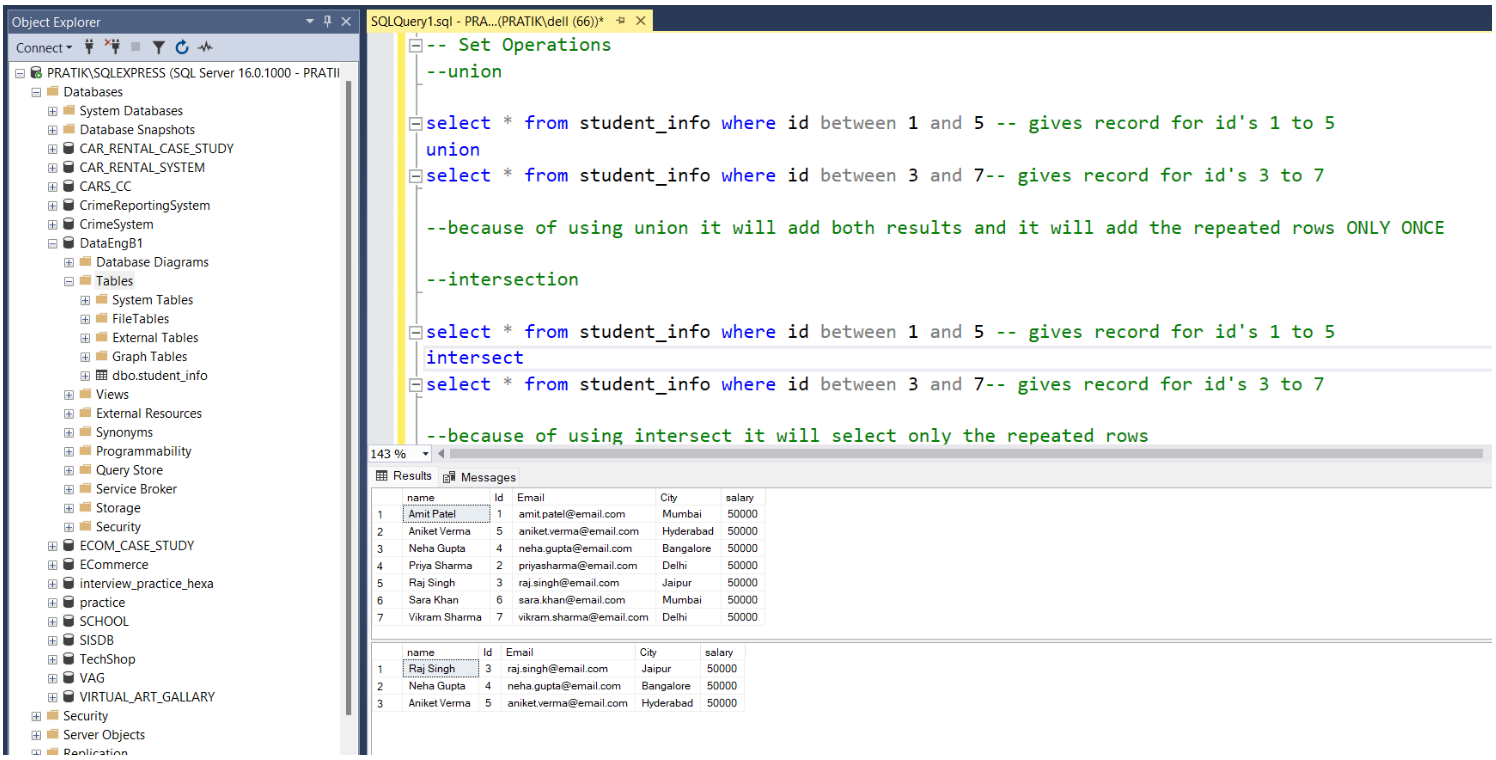


```sql
begin transaction;

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 occured 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.

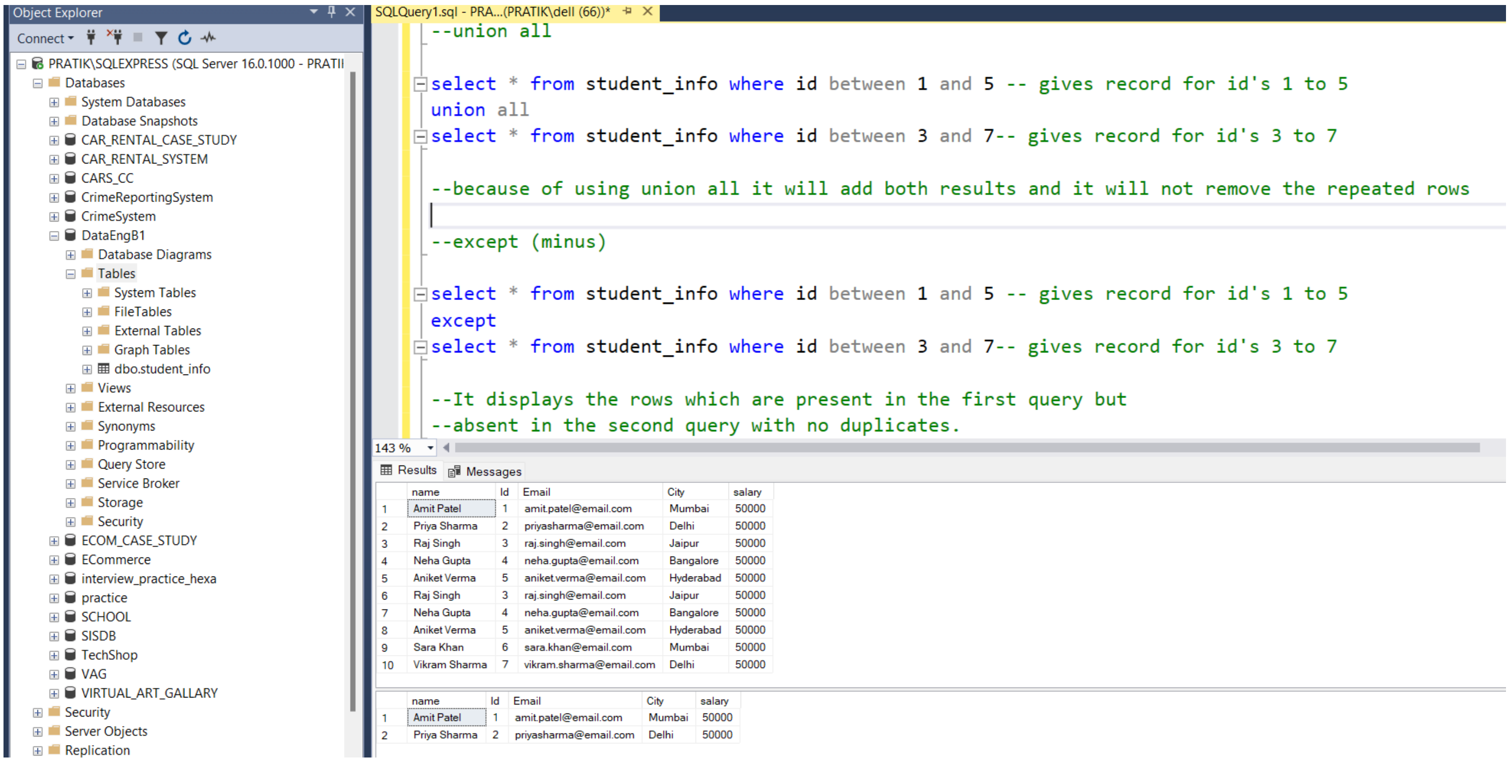


```sql
-- Set Operations
--union

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
```

	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

	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.



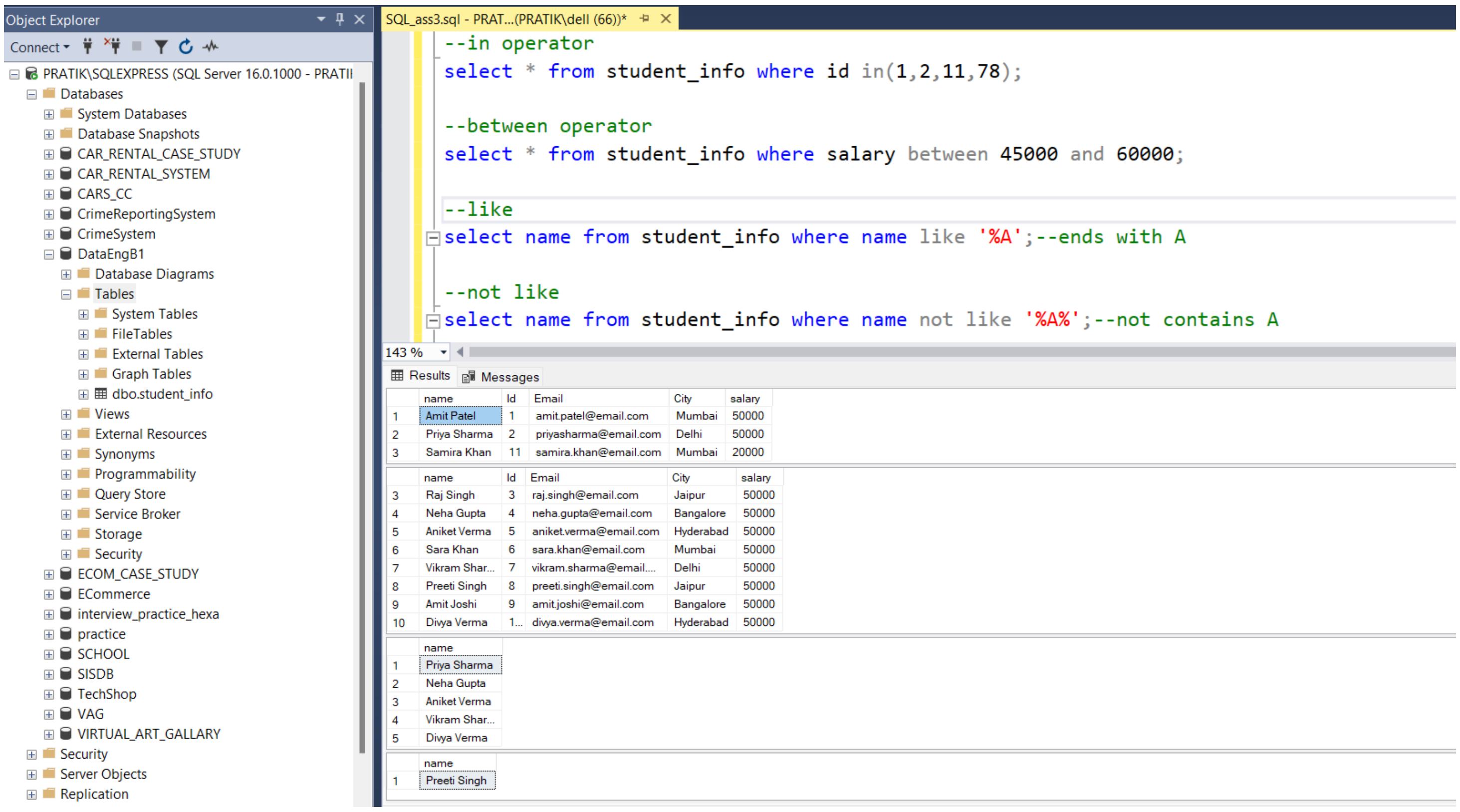
```sql
--union all

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.
```

	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

- <u>Operators:</u>

  - 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.



```
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_GALLARY
  Security
  Server Objects
  Replication
```

```sql
--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
```

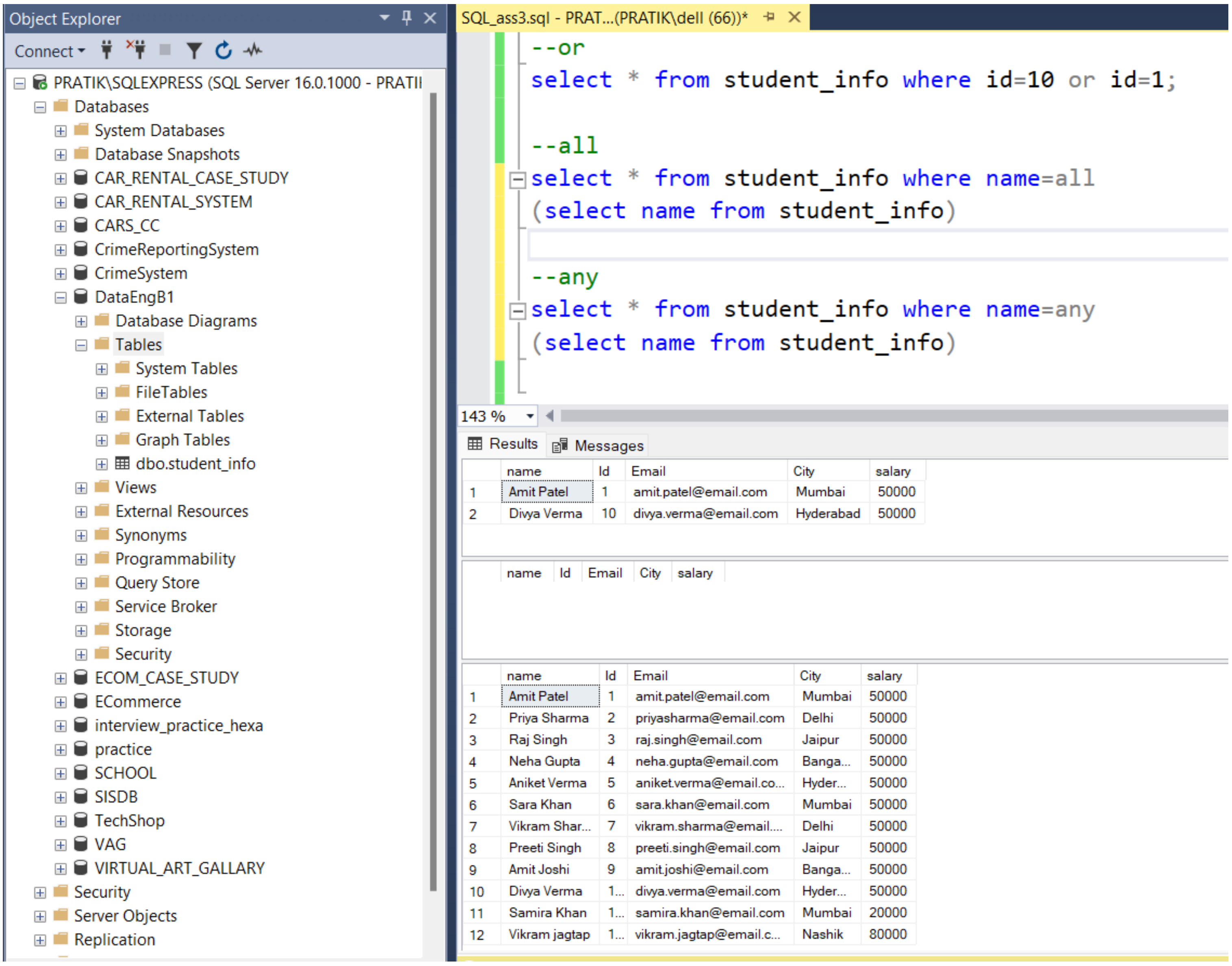
	name	Id	Email	City	salary
1	Amit Patel	1	amit.patel@email.com	Mumbai	50000
2	Priya Sharma	2	priyasharma@email.com	Delhi	50000
3	Samira Khan	11	samira.khan@email.com	Mumbai	20000

	name	Id	Email	City	salary
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 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	Bangalore	50000
10	Divya Verma	1...	divya.verma@email.com	Hyderabad	50000

	name
1	Priya Sharma
2	Neha Gupta
3	Aniket Verma
4	Vikram Shar...
5	Divya Verma

	name
1	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.



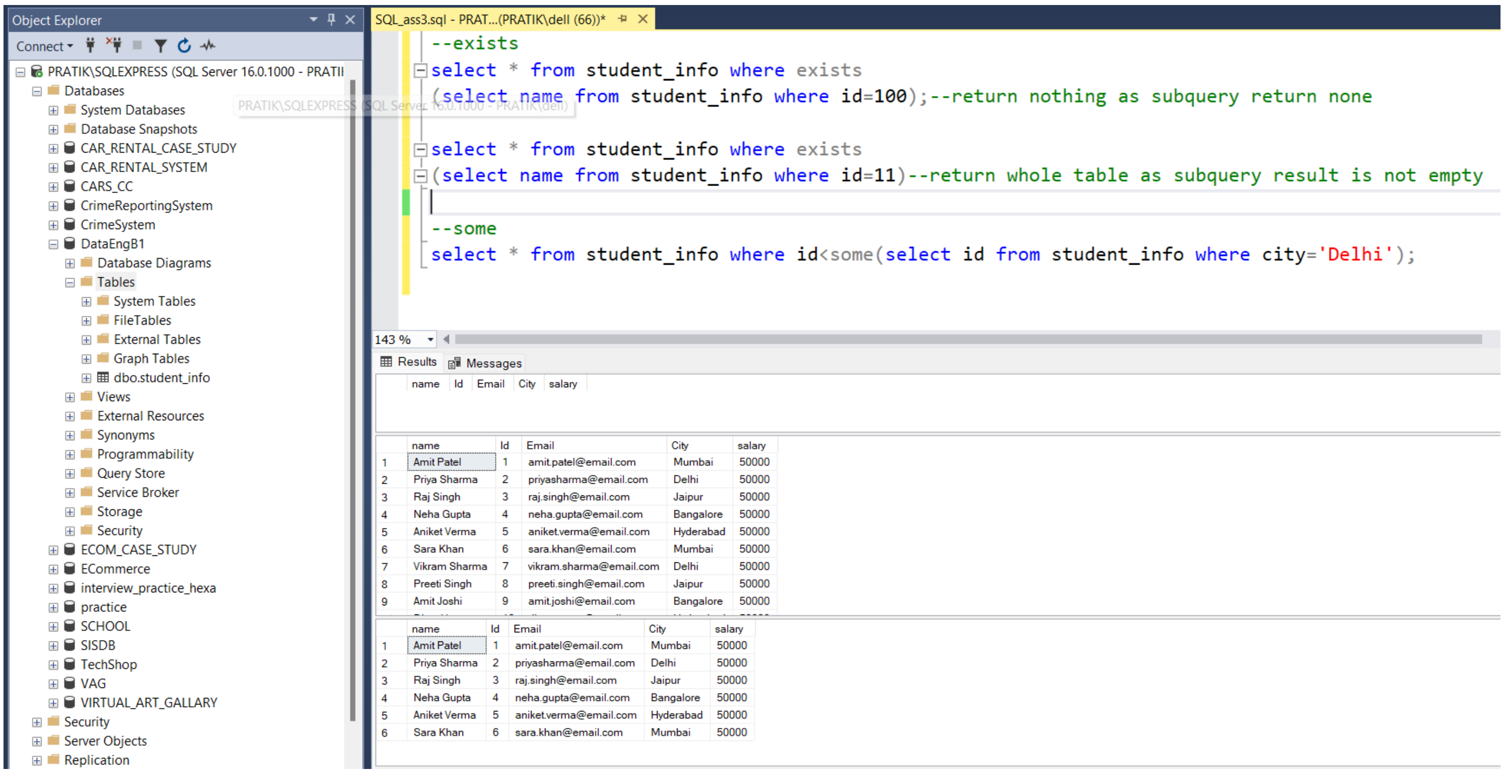
```
--or
select * from student_info where id=10 or id=1;

--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)
```

	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

	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.



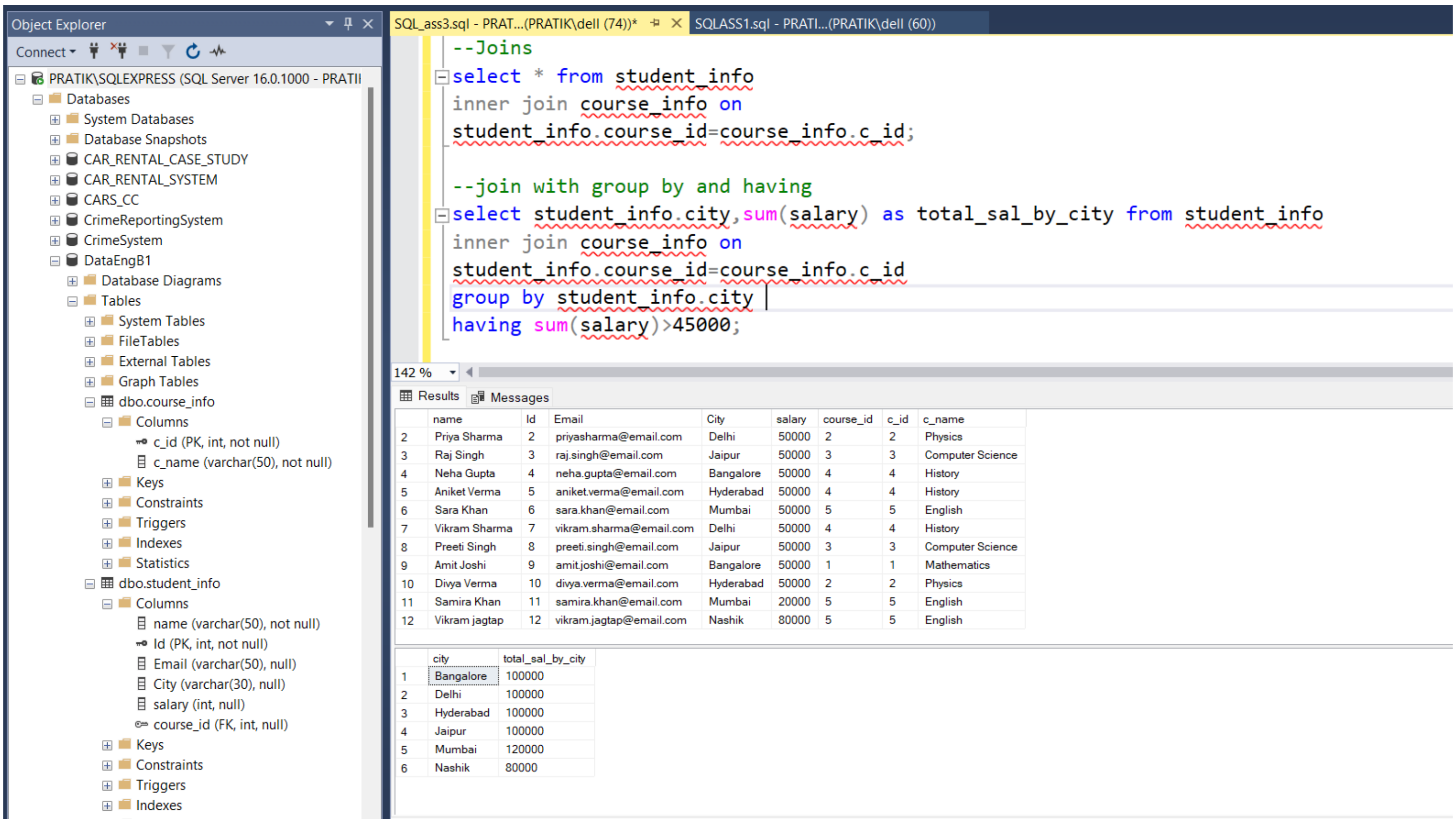
```sql
--exists
select * from student_info where exists
(select name from student_info where id=100);--return nothing as subquery return none

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

	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.


Object Explorer
Connect
SQL_ass3.sql - PRAT...(PRATIK\dell (74))*
SQLASS1.sql - PRATI...(PRATIK\dell (60))
142 %
Results
Messages

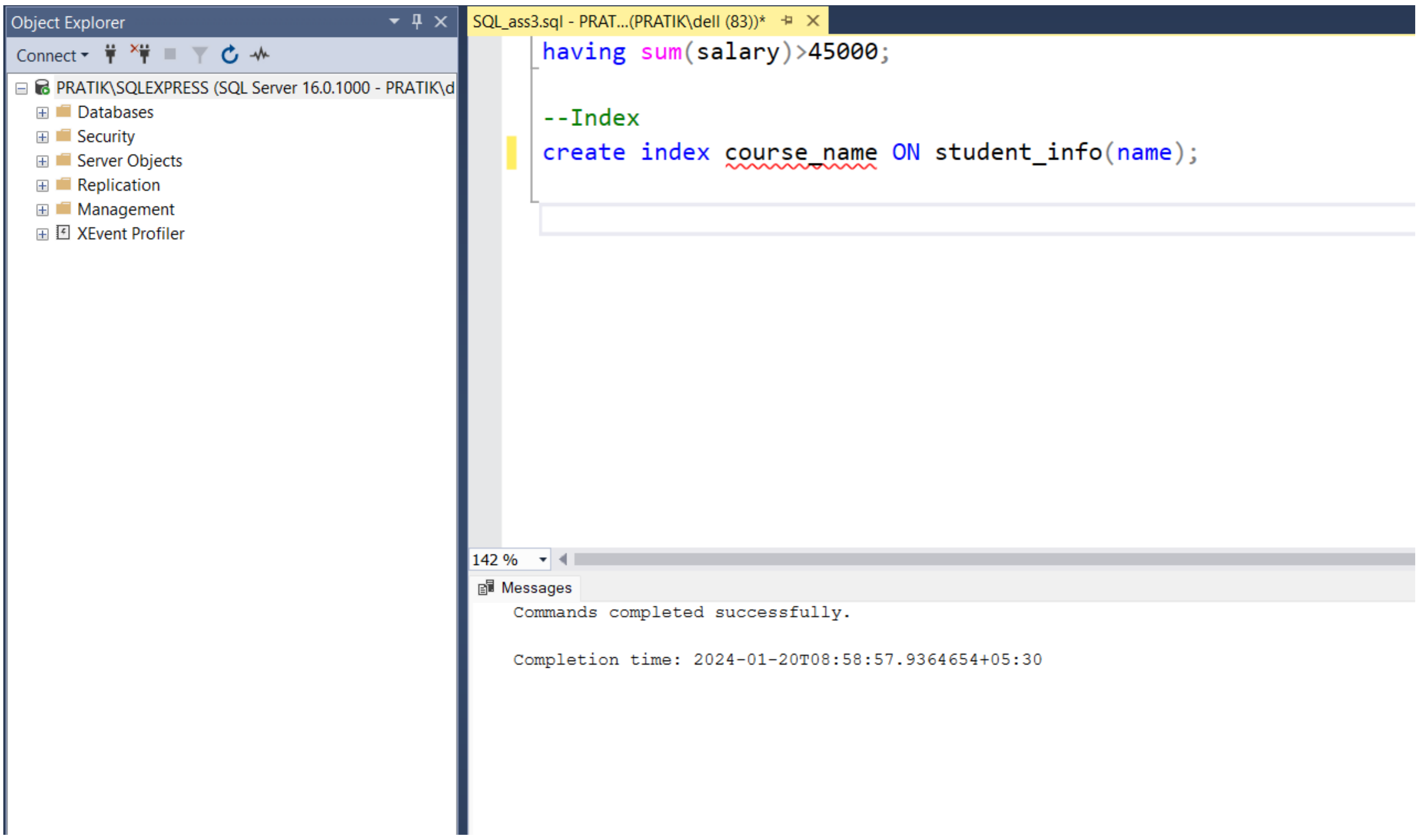
```
PRATIK\SQLEXPRESS (SQL Server 16.0.1000 - PRATII
  Databases
    System Databases
    Database Snapshots
    CAR_RENTAL_CASE_STUDY
    CAR_RENTAL_SYSTEM
    CARS_CC
    CrimeReportingSystem
    CrimeSystem
    DataEngB1
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.course_info
          Columns
            c_id (PK, int, not null)
            c_name (varchar(50), not null)
          Keys
          Constraints
          Triggers
          Indexes
          Statistics
        dbo.student_info
          Columns
            name (varchar(50), not null)
            Id (PK, int, not null)
            Email (varchar(50), null)
            City (varchar(30), null)
            salary (int, null)
            course_id (FK, int, null)
          Keys
          Constraints
          Triggers
          Indexes
```

```sql
--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;
```

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

	city	total_sal_by_city
1	Bangalore	100000
2	Delhi	100000
3	Hyderabad	100000
4	Jaipur	100000
5	Mumbai	120000
6	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



```
having sum(salary)>45000;

--Index
create index course_name ON student_info(name);
```

```
Commands completed successfully.

Completion time: 2024-01-20T08:58:57.9364654+05:30
```