Week 4 - Database Technologies Assignment

1)List the type of joins. Explain every join type with an equivalent SQL statement.

1A)Types of SQL JOINS explained with examples:

JOINS fundamentals

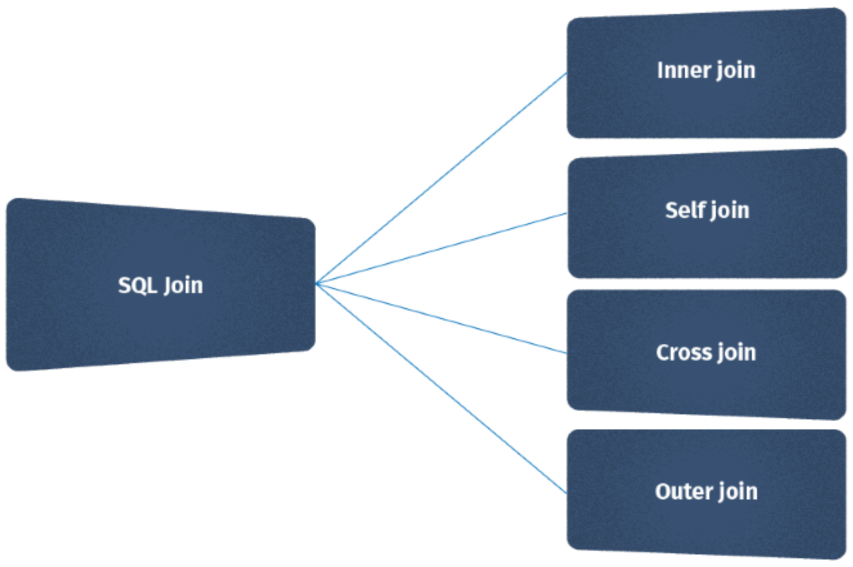
In relational databases, such as SQL Server, Oracle, MySQL, and others, data is stored in multiple tables that are related to each other with a common key value. Accordingly, there is a constant need to extract records from two or more tables into a results table based on some condition. In SQL Server, this can be easily accomplished with the SQL JOIN clause

JOIN is an SQL clause used to query and access data from multiple tables, based on logical relationships between those tables.
In other words, JOINS indicate how SQL Server should use data from one table to select the rows from another table

Different types of JOINS in SQL Server

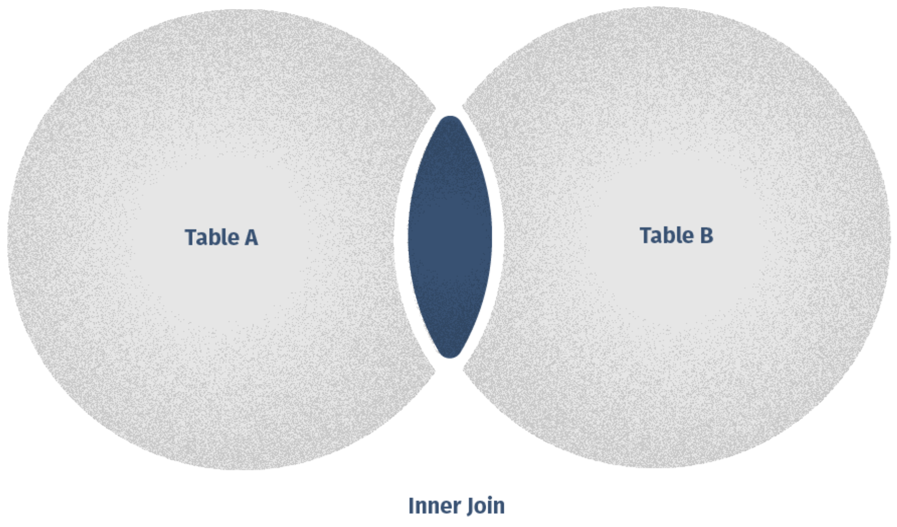
- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- SELF JOIN
- CROSS JOIN

- **SQL INNER JOIN** creates a result table by combining rows that have matching values in two or more tables.
- **SQL LEFT OUTER JOIN** includes in a result table unmatched rows from the table that is specified before the LEFT OUTER JOIN clause.
- **SQL RIGHT OUTER JOIN** creates a result table and includes into it all the records from the right table and only matching rows from the left table.
- **SQL SELF JOIN** joins the table to itself and allows comparing rows within the same table.
- **SQL CROSS JOIN** creates a result table containing paired combination of each row of the first table with each row of the second table.

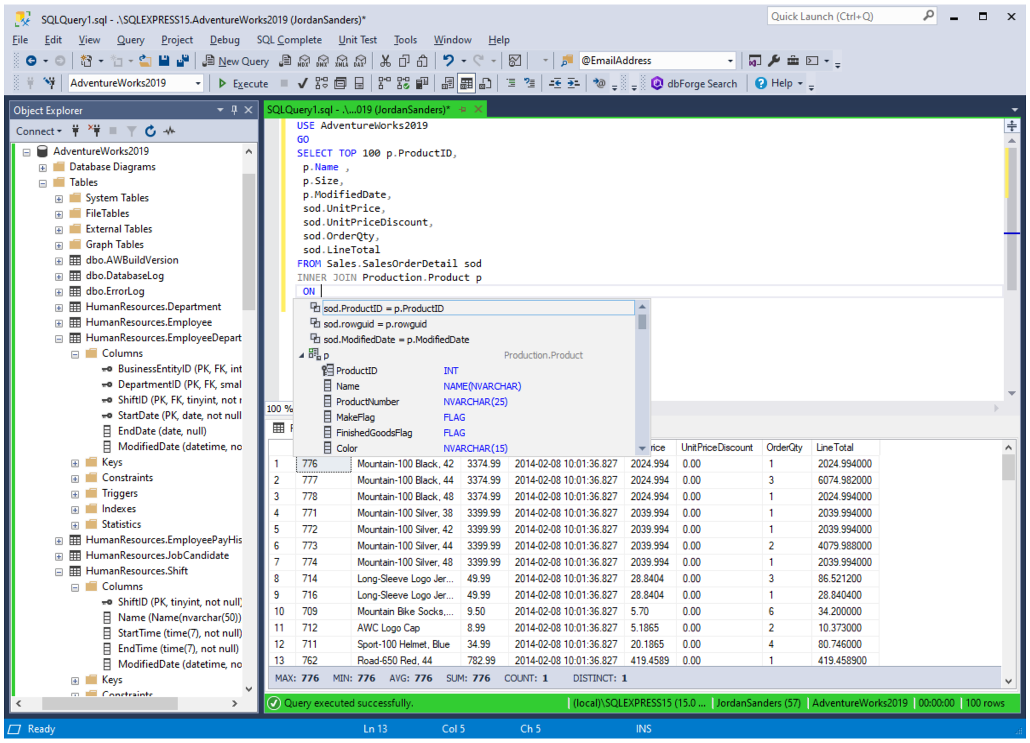


INNER JOIN

INNER JOIN statement returns only those records or rows that have matching values and is used to retrieve data that appears in both tables



In our example, we want to extract data from the Sales.SalesOrderDetail and Production.Product tables that are aliased with SOD for Sales.SalesOrderDetail and P for Production.Product. In the JOIN statement, we match records in those columns. Make notice, how code suggestions work in SQL Complete



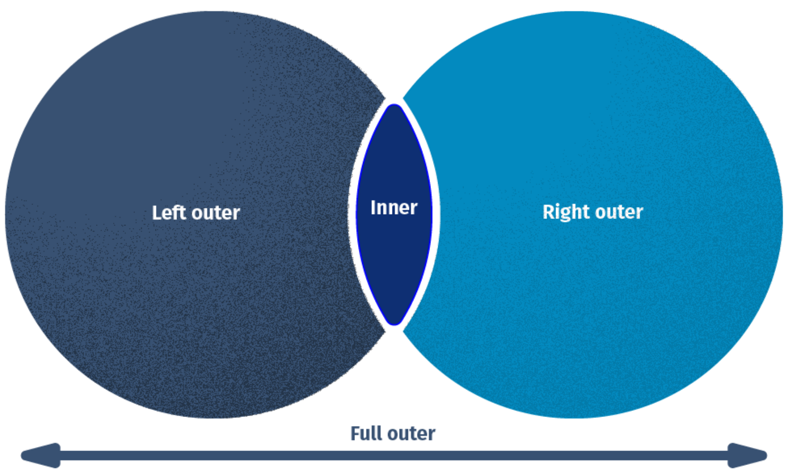
```
USE AdventureWorks2019
GO
SELECT TOP 100 p.ProductID,
 p.Name ,
 p.Size,
 p.ModifiedDate,
 sod.UnitPrice,
 sod.UnitPriceDiscount,
 sod.OrderQty,
 sod.LineTotal
FROM Sales.SalesOrderDetail sod
INNER JOIN Production.Product p
 ON
```

					rice	UnitPriceDiscount	OrderQty	LineTotal
1	776	Mountain-100 Black, 42	3374.99	2014-02-08 10:01:36.827	2024.994	0.00	1	2024.994000
2	777	Mountain-100 Black, 44	3374.99	2014-02-08 10:01:36.827	2024.994	0.00	3	6074.982000
3	778	Mountain-100 Black, 48	3374.99	2014-02-08 10:01:36.827	2024.994	0.00	1	2024.994000
4	771	Mountain-100 Silver, 38	3399.99	2014-02-08 10:01:36.827	2039.994	0.00	1	2039.994000
5	772	Mountain-100 Silver, 42	3399.99	2014-02-08 10:01:36.827	2039.994	0.00	1	2039.994000
6	773	Mountain-100 Silver, 44	3399.99	2014-02-08 10:01:36.827	2039.994	0.00	2	4079.988000
7	774	Mountain-100 Silver, 48	3399.99	2014-02-08 10:01:36.827	2039.994	0.00	1	2039.994000
8	714	Long-Sleeve Logo Jer...	49.99	2014-02-08 10:01:36.827	28.8404	0.00	3	86.521200
9	716	Long-Sleeve Logo Jer...	49.99	2014-02-08 10:01:36.827	28.8404	0.00	1	28.840400
10	709	Mountain Bike Socks,...	9.50	2014-02-08 10:01:36.827	5.70	0.00	6	34.200000
11	712	AWC Logo Cap	8.99	2014-02-08 10:01:36.827	5.1865	0.00	2	10.373000
12	711	Sport-100 Helmet, Blue	34.99	2014-02-08 10:01:36.827	20.1865	0.00	4	80.746000
13	762	Road-650 Red, 44	782.99	2014-02-08 10:01:36.827	419.4589	0.00	1	419.458900

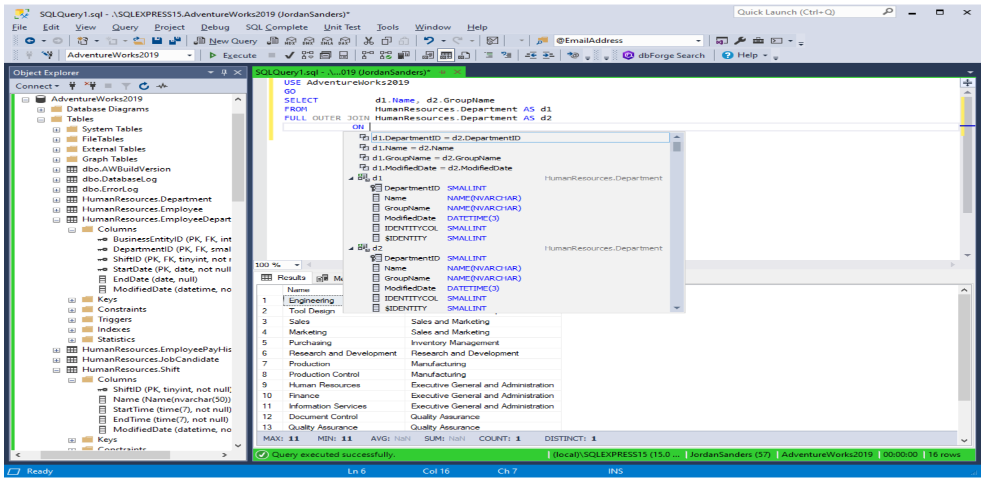
```
MAX: 776   MIN: 776   AVG: 776   SUM: 776   COUNT: 1   DISTINCT: 1
Query executed successfully.   (local)\SQLEXPRESS15 (15.0 ... | JordanSanders (57) | AdventureWorks2019 | 00:00:00 | 100 rows
```

OUTER JOIN

When applying an SQL INNER JOIN, the output returns only matching rows from the stated tables. In contrast, if you use an SQL OUTER JOIN, it will retrieve not only the matching rows but also the unmatched rows as well.



The FULL OUTER JOIN returns a result that includes rows from both left and right tables. In case, no matching rows exist for the row in the left table, the columns of the right table will have nulls. Correspondingly, the column of the left table will have nulls if there are no matching rows for the row in the right table.



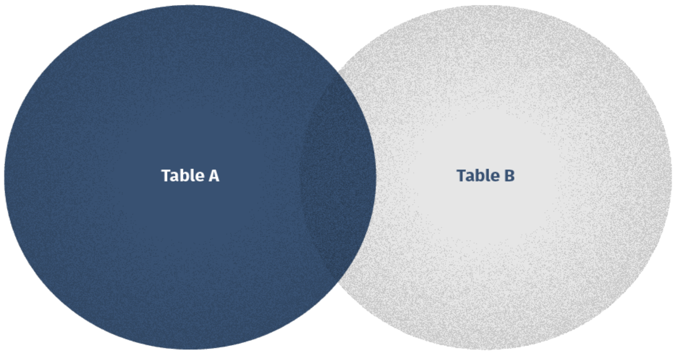
```
USE AdventureWorks2019
GO
SELECT          d1.Name, d2.GroupName
FROM            HumanResources.Department AS d1
FULL OUTER JOIN HumanResources.Department AS d2
          ON
```

```
d1.DepartmentID = d2.DepartmentID
d1.Name = d2.Name
d1.GroupName = d2.GroupName
d1.ModifiedDate = d2.ModifiedDate
d1                               HumanResources.Department
   DepartmentID   SMALLINT
   Name           NAME(NVARCHAR)
   GroupName      NAME(NVARCHAR)
   ModifiedDate   DATETIME(3)
   IDENTITYCOL    SMALLINT
   $IDENTITY      SMALLINT
d2                               HumanResources.Department
   DepartmentID   SMALLINT
   Name           NAME(NVARCHAR)
   GroupName      NAME(NVARCHAR)
   ModifiedDate   DATETIME(3)
   IDENTITYCOL    SMALLINT
   $IDENTITY      SMALLINT
```

```
    Name
1   Engineering
2   Tool Design
3   Sales                      Sales and Marketing
4   Marketing                  Sales and Marketing
5   Purchasing                 Inventory Management
6   Research and Development   Research and Development
7   Production                 Manufacturing
8   Production Control         Manufacturing
9   Human Resources            Executive General and Administration
10  Finance                    Executive General and Administration
11  Information Services       Executive General and Administration
12  Document Control           Quality Assurance
13  Quality Assurance          Quality Assurance
MAX: 11   MIN: 11   AVG: NaN   SUM: NaN   COUNT: 1   DISTINCT: 1
Query executed successfully.   (local)\SQLEXPRESS15 (15.0 ...   JordanSanders (57)   AdventureWorks2019   00:00:00   16 rows
Ready   Ln 6   Col 16   Ch 7   INS
```

LEFT OUTER JOIN

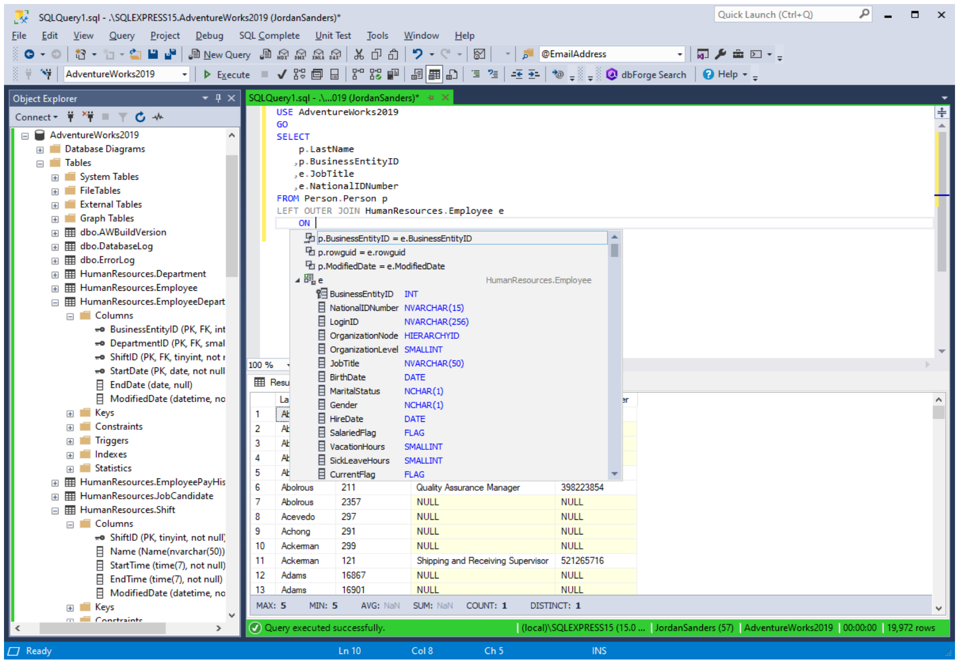
The LEFT OUTER JOIN gives the output of the matching rows between both tables. In case, no records match from the left table, it shows those records with null values.



Left Outer Join

In our example, we want to join the tables Person.Person and HumanResources.Employee to retrieve a list of all Person LastNames, but also show JobTitle if the Person is an Employee.

In the output, in case, there are no employees matching BusinessEntityID, NULL values will be listed in the corresponding rows for NationalIDNumber and JobTitle.



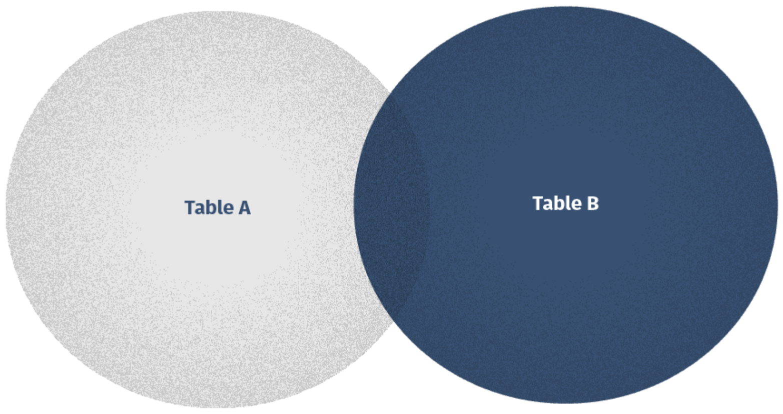
```
USE AdventureWorks2019
GO
SELECT
    p.LastName
   ,p.BusinessEntityID
   ,e.JobTitle
   ,e.NationalIDNumber
FROM Person.Person p
LEFT OUTER JOIN HumanResources.Employee e
    ON
```

```
6   Abolrous   211     Quality Assurance Manager             398223854
7   Abolrous   2357    NULL                                  NULL
8   Acevedo    297     NULL                                  NULL
9   Achong     291     NULL                                  NULL
10  Ackerman   299     NULL                                  NULL
11  Ackerman   121     Shipping and Receiving Supervisor     521265716
12  Adams      16867   NULL                                  NULL
13  Adams      16901   NULL                                  NULL
```

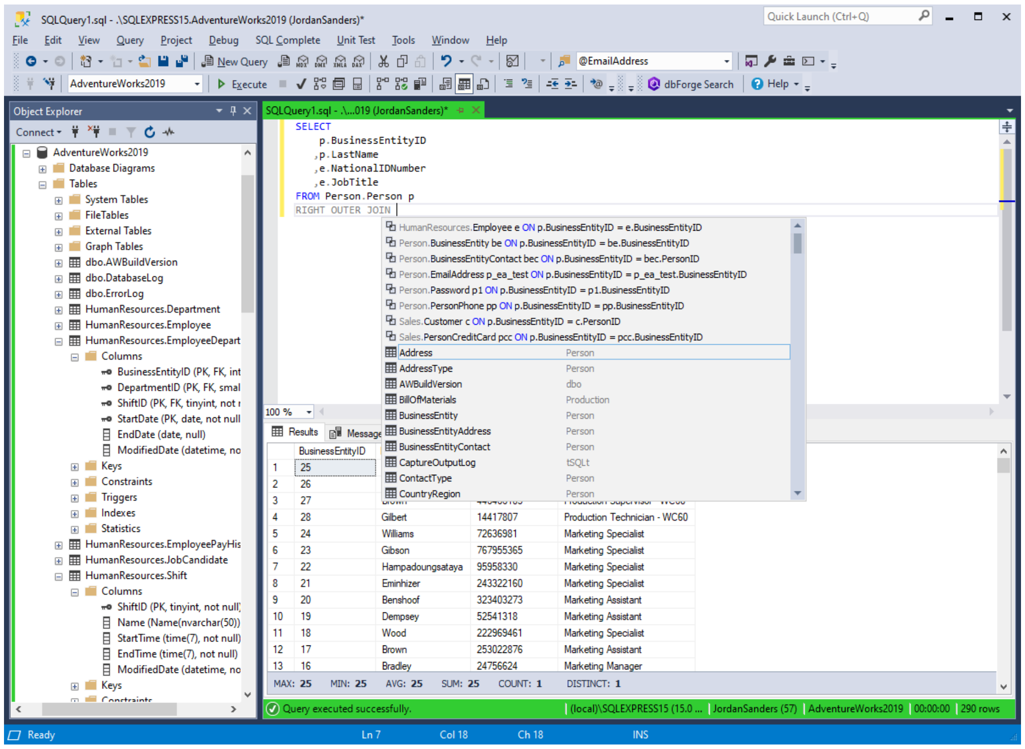
Query executed successfully.

RIGHT OUTER JOIN

The RIGHT OUTER JOIN works by the same principle as the LEFT OUTER JOIN. The RIGHT OUTER JOIN selects data from the right table (Table B) and matches this data with the rows from the left table (Table A). The RIGHT JOIN returns a result set that includes all rows in the right table, whether or not they have matching rows from the left table. In case, a row in the right table does not have any matching rows in the left table, the column of the left table in the result set will have nulls.



Right Outer Join



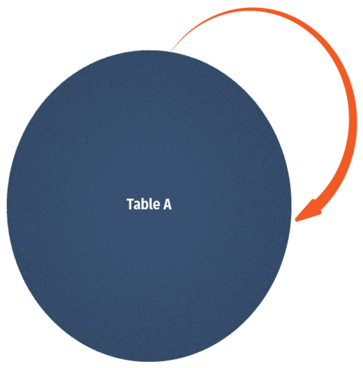
```sql
SELECT
    p.BusinessEntityID
   ,p.LastName
   ,e.NationalIDNumber
   ,e.JobTitle
FROM Person.Person p
RIGHT OUTER JOIN 
```

```
HumanResources.Employee e ON p.BusinessEntityID = e.BusinessEntityID
Person.BusinessEntity be ON p.BusinessEntityID = be.BusinessEntityID
Person.BusinessEntityContact bec ON p.BusinessEntityID = bec.PersonID
Person.EmailAddress p_ea_test ON p.BusinessEntityID = p_ea_test.BusinessEntityID
Person.Password p1 ON p.BusinessEntityID = p1.BusinessEntityID
Person.PersonPhone pp ON p.BusinessEntityID = pp.BusinessEntityID
Sales.Customer c ON p.BusinessEntityID = c.PersonID
Sales.PersonCreditCard pcc ON p.BusinessEntityID = pcc.BusinessEntityID
Address                  Person
AddressType              Person
AWBuildVersion           dbo
BillOfMaterials          Production
BusinessEntity           Person
BusinessEntityAddress    Person
BusinessEntityContact    Person
CaptureOutputLog         tSQLt
ContactType              Person
CountryRegion            Person
```

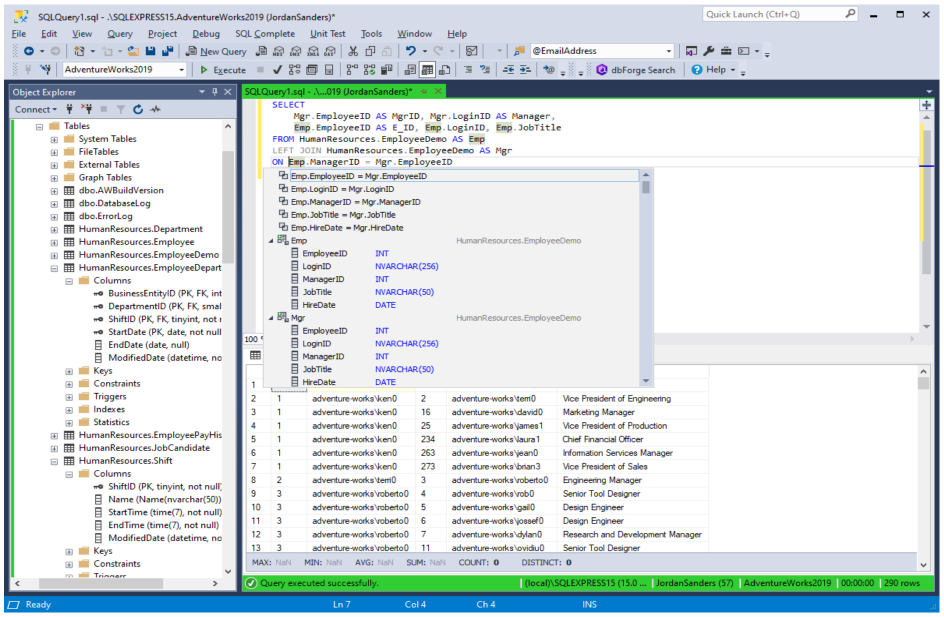
```
    BusinessEntityID
1   25
2   26
3   27
4   28    Gilbert            14417807    Production Technician - WC60
5   24    Williams           72636981    Marketing Specialist
6   23    Gibson             767955365   Marketing Specialist
7   22    Hampadoungsataya   95958330    Marketing Specialist
8   21    Eminhizer          243322160   Marketing Specialist
9   20    Benshoof           323403273   Marketing Assistant
10  19    Dempsey            52541318    Marketing Assistant
11  18    Wood               222969461   Marketing Specialist
12  17    Brown              253022876   Marketing Assistant
13  16    Bradley            24756624    Marketing Manager
MAX: 25   MIN: 25   AVG: 25   SUM: 25   COUNT: 1   DISTINCT: 1
Query executed successfully.  (local)\SQLEXPRESS15 (15.0 ...  JordanSanders (57)  AdventureWorks2019  00:00:00  290 rows
```

SELF JOIN

The SELF JOIN allows you to join a table to itself. This implies that each row of the table is combined with itself and with every other row of the table. The SELF JOIN can be viewed as a join of two copies of the same table. The table is not actually copied, but SQL performs the command as though it were. This is accomplished by using table name aliases to give each instance of the table a separate name. It is most useful for extracting hierarchical data or comparing rows within the same table.



In our example, we want to retrieve a list of all the territories and the salespeople working in them from the Sales.SalesPerson table



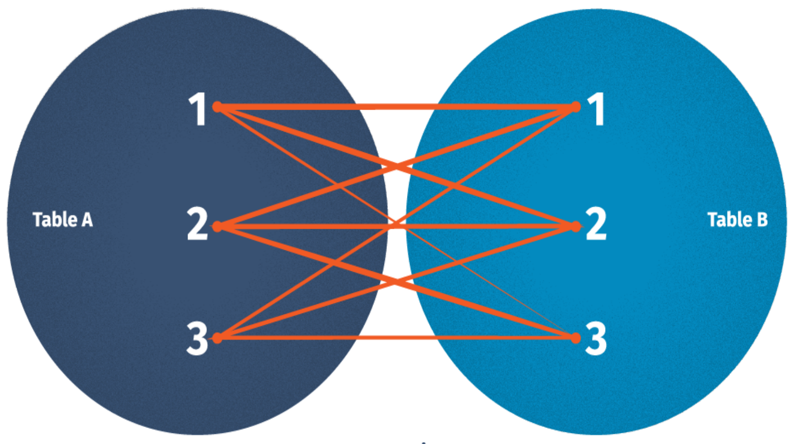
```sql
SELECT
    Mgr.EmployeeID AS MgrID, Mgr.LoginID AS Manager,
    Emp.EmployeeID AS E_ID, Emp.LoginID, Emp.JobTitle
FROM HumanResources.EmployeeDemo AS Emp
LEFT JOIN HumanResources.EmployeeDemo AS Mgr
ON Emp.ManagerID = Mgr.EmployeeID
```

```
Emp.EmployeeID = Mgr.EmployeeID
Emp.LoginID = Mgr.LoginID
Emp.ManagerID = Mgr.ManagerID
Emp.JobTitle = Mgr.JobTitle
Emp.HireDate = Mgr.HireDate
Emp                          HumanResources.EmployeeDemo
    EmployeeID   INT
    LoginID      NVARCHAR(256)
    ManagerID    INT
    JobTitle     NVARCHAR(50)
    HireDate     DATE
Mgr                          HumanResources.EmployeeDemo
    EmployeeID   INT
    LoginID      NVARCHAR(256)
    ManagerID    INT
    JobTitle     NVARCHAR(50)
    HireDate     DATE
```

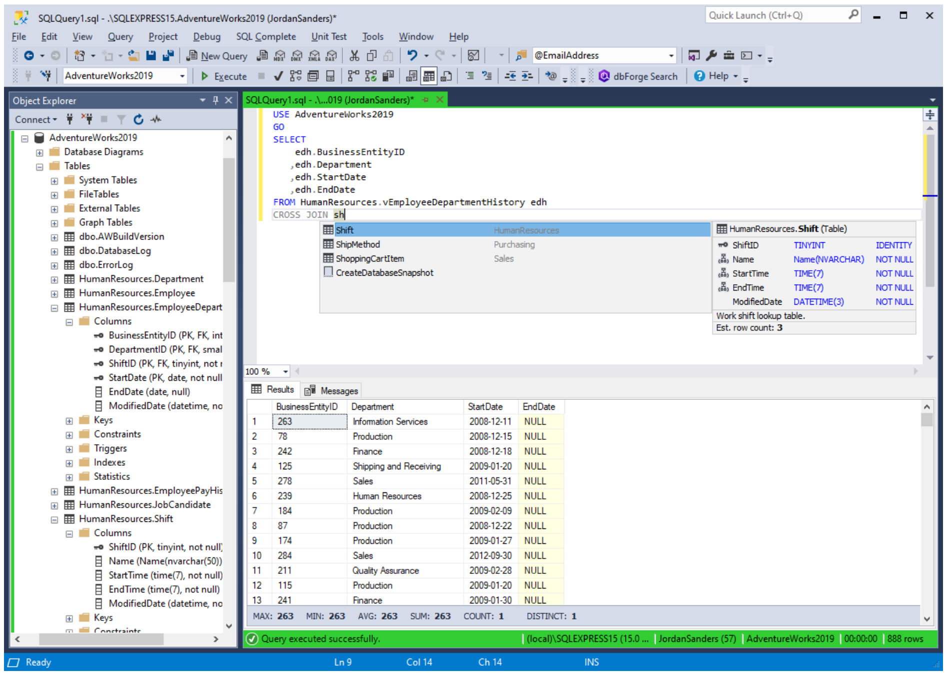
```
1
2   1   adventure-works\ken0      2     adventure-works\terri0     Vice President of Engineering
3   1   adventure-works\ken0      16    adventure-works\david0     Marketing Manager
4   1   adventure-works\ken0      25    adventure-works\james1     Vice President of Production
5   1   adventure-works\ken0      234   adventure-works\laura1     Chief Financial Officer
6   1   adventure-works\ken0      263   adventure-works\jean0      Information Services Manager
7   1   adventure-works\ken0      273   adventure-works\brian3     Vice President of Sales
8   2   adventure-works\terri0    3     adventure-works\roberto0   Engineering Manager
9   3   adventure-works\roberto0  4     adventure-works\rob0       Senior Tool Designer
10  3   adventure-works\roberto0  5     adventure-works\gail0      Design Engineer
11  3   adventure-works\roberto0  6     adventure-works\jossef0    Design Engineer
12  3   adventure-works\roberto0  7     adventure-works\dylan0     Research and Development Manager
13  3   adventure-works\roberto0  11    adventure-works\ovidiu0    Senior Tool Designer
MAX: NaN   MIN: NaN   AVG: NaN   SUM: NaN   COUNT: 0   DISTINCT: 0
Query executed successfully.  | (local)\SQLEXPRESS15 (15.0 ... | JordanSanders (57) | AdventureWorks2019 | 00:00:00 | 290 rows
Ready   Ln 7   Col 4   Ch 4   INS
```

CROSS JOIN

The CROSS JOIN command in SQL, also known as a cartesian join, returns all combinations of rows from each table. Envision that you need to find all combinations of size and color. In that case, a CROSS JOIN will be an asset. Note, that this join does not need any condition to join two tables. In fact, CROSS JOIN joins every row from the first table with every row from the second table and its result comprises all combinations of records in two tables.



Cross Join



```sql
USE AdventureWorks2019
GO
SELECT
    edh.BusinessEntityID
   ,edh.Department
   ,edh.StartDate
   ,edh.EndDate
FROM HumanResources.vEmployeeDepartmentHistory edh
CROSS JOIN sh
```

	BusinessEntityID	Department	StartDate	EndDate
1	263	Information Services	2008-12-11	NULL
2	78	Production	2008-12-15	NULL
3	242	Finance	2008-12-18	NULL
4	125	Shipping and Receiving	2009-01-20	NULL
5	278	Sales	2011-05-31	NULL
6	239	Human Resources	2008-12-25	NULL
7	184	Production	2009-02-09	NULL
8	87	Production	2008-12-22	NULL
9	174	Production	2009-01-27	NULL
10	284	Sales	2012-09-30	NULL
11	211	Quality Assurance	2009-02-28	NULL
12	115	Production	2009-01-20	NULL
13	241	Finance	2009-01-30	NULL

MAX: 263 MIN: 263 AVG: 263 SUM: 263 COUNT: 1 DISTINCT: 1

Query executed successfully. | (local)\SQLEXPRESS15 (15.0 ... | JordanSanders (57) | AdventureWorks2019 | 00:00:00 | 888 rows

2)List and explain the set operations

2A)<u>SQL Set Operations:</u>

The SQL Set operation is used to combine the two or more SQL SELECT statements.

Types of Set Operations

1. Union
2. UnionAll
3. Intersect
4. Minus

Union
1
Union All
2
Set Operation
3
Intersect
4
Minus

1.Union

- The SQL Union operation is used to combine the result of two or more SQL SELECT queries.

- In the union operation, all the number of datatype and columns must be same in both the tables on which UNION operation is being applied.

- ❑ The union operation eliminates the duplicate rows from its resultset

Syntax

```
SELECT column_name FROM table1
UNION
SELECT column_name FROM table2
```

Example:

The First table

ID	NAME
1	Jack
2	Harry

3	Jackson

The Second table

ID	**NAME**
3	Jackson
4	Stephan
5	David

Union SQL query will be:

```
SELECT * FROM First
UNION
SELECT * FROM Second;
```

The resultset table will look like:

ID	**NAME**
1	Jack

2	Harry
3	Jackson
4	Stephan
5	David

2.Union All

Union All operation is equal to the Union operation. It returns the set without removing duplication and sorting the data.

Syntax:

```
SELECT column_name FROM table1
UNION ALL
SELECT column_name FROM table2;
```

Example: Using the above First and Second table.

Union All query will be like:

```
SELECT * FROM First
UNION ALL
SELECT * FROM Second;
```

The resultset table will look like:

ID	**NAME**
1	Jack
2	Harry
3	Jackson
4	Stephan
5	David

3.Intersect

- ☐It is used to combine two SELECT statements. The Intersect operation returns the common rows from both the SELECT statements.

- ☐In the Intersect operation, the number of datatype and columns must be the same.

- ☐It has no duplicates and it arranges the data in ascending order by default.

Syntax

```
SELECT column_name FROM table1
INTERSECT
SELECT column_name FROM table2;
```

Example:

Using the above First and Second table.

Intersect query will be:

```
SELECT * FROM First
INTERSECT
SELECT * FROM Second;
```

The resultset table will look like:

ID	**NAME**
3	Jackson

4. Minus

- ☐It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- ☐It has no duplicates and data arranged in ascending order by default.

Syntax:

```
SELECT column_name FROM table1
MINUS
SELECT column_name FROM table2;
```

Example

Using the above First and Second table.

Minus query will be:

```
SELECT * FROM First
MINUS
SELECT * FROM Second;
```

The resultset table will look like:

ID	**NAME**
1	Jack
2	Harry

3)How to rename a column?

3A)Rename Columns in SQL Server:

SQL Server allows us to change the column whenever we need. We will rename the table columns when the column name is non-meaningful or does not fulfill the purpose of its creation. It must ensure that we have ALTER permission on the object before changing the column's name.

Limitations and Restrictions:

SQL Server has some restrictions while changing the column name because when we rename a column, it does not imply that all references to that column will be renamed as well. We must manually modify all objects that belong to the renamed column.

For example, if we want to change the column of a table that is also referenced in a trigger, it is required to modify the trigger for reflecting the new column name as well. The sys.sql_expression_dependencies can be used for listing all dependencies on the object before changing the name.

We can rename the table columns in SQL Server using mainly two ways:

1.Transact-SQL
2.SQL Server Management Studio (SSMS)

<u>**1.Transact-SQL**</u>

SQL Server provides a standard stored procedure called SP_RENAME for changing the name of a user-created object in the current database. The user-created object can be a table, column, index, alias data type, etc.

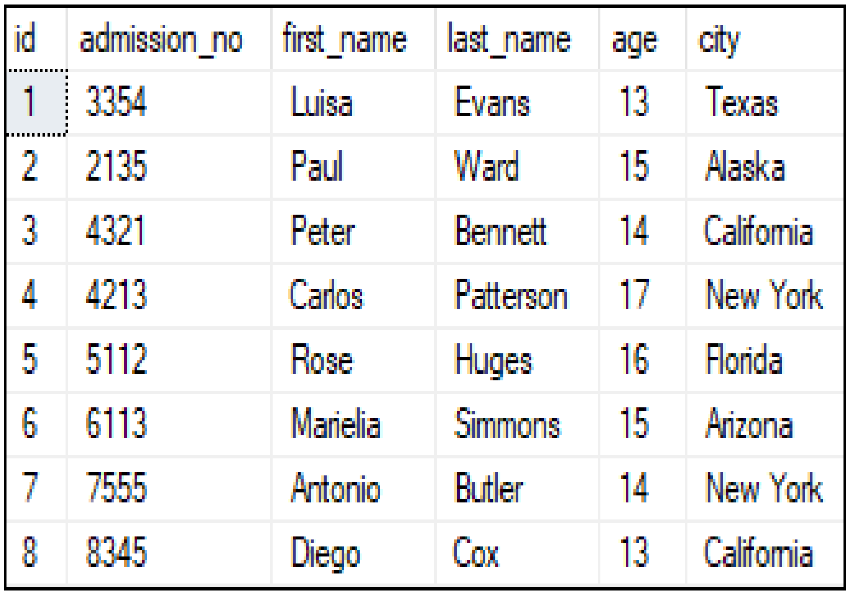
Scripts and stored procedures may be broken when we change some portion of an object's name. We advise you to drop the object and re-create it with the new name rather than using this statement to change the name of stored procedures, triggers, user-defined functions, or views.

The syntax to change the column name using this approach is:

```
EXEC SP_RENAME '[Table Name].[Old
Column Name]', '[New Column Name]',
COLUMN
```

Rename Column Example

The following example demonstrates the SQL Server rename column using the SP_RENAME stored procedure. To do this, we will take a 'student' table that contains the below data:



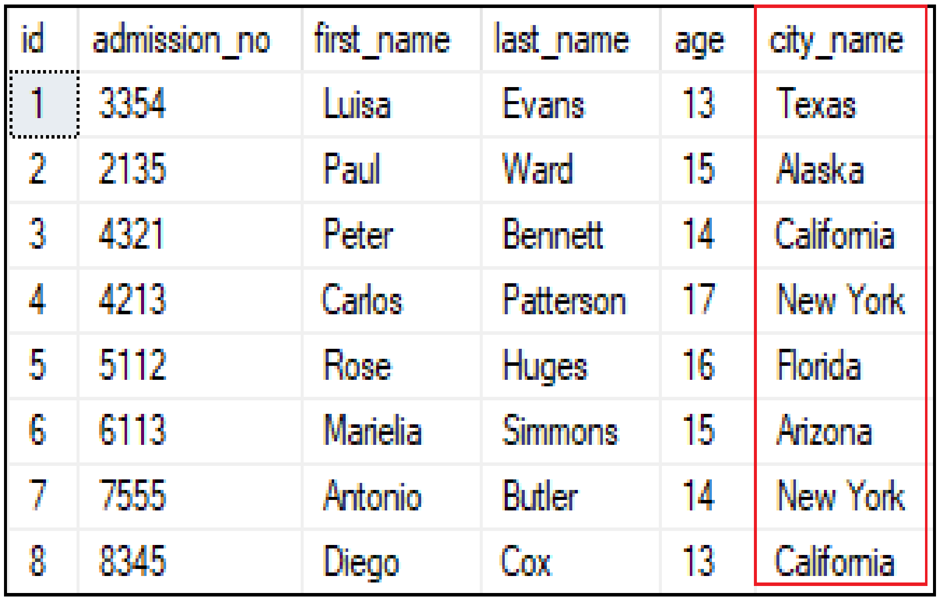
id	admission_no	first_name	last_name	age	city
1	3354	Luisa	Evans	13	Texas
2	2135	Paul	Ward	15	Alaska
3	4321	Peter	Bennett	14	California
4	4213	Carlos	Patterson	17	New York
5	5112	Rose	Huges	16	Florida
6	6113	Marielia	Simmons	15	Arizona
7	7555	Antonio	Butler	14	New York
8	8345	Diego	Cox	13	California

If we want to change the 'city' column with the new name 'city_name' of this table, we can use the above-specified SQL Server syntax or stored procedure as follows:

```
EXEC SP_RENAME 'Student.city', 'city_name',
'COLUMN'
```

After executing this script, we will get the following message:

When we verify the 'student' table, we can see that the column name of 'city' is changed successfully:



id	admission_no	first_name	last_name	age	city_name
1	3354	Luisa	Evans	13	Texas
2	2135	Paul	Ward	15	Alaska
3	4321	Peter	Bennett	14	California
4	4213	Carlos	Patterson	17	New York
5	5112	Rose	Huges	16	Florida
6	6113	Marielia	Simmons	15	Arizona
7	7555	Antonio	Butler	14	New York
8	8345	Diego	Cox	13	California

2.SQL Server Management Studio (SSMS)

SSMS is a windows software tool used to connect and work with our SQL Server from a graphical interface instead of using the command line. The management studio allows us to rename the table columns in the following ways:

- Rename a column using Object Explorer
- Double click on the column name.
- Rename a column using Table Designer.

Let us discuss each of them in detail.

Rename a column using Object Explorer:

The following steps are used to change the column name using Object Explorer:

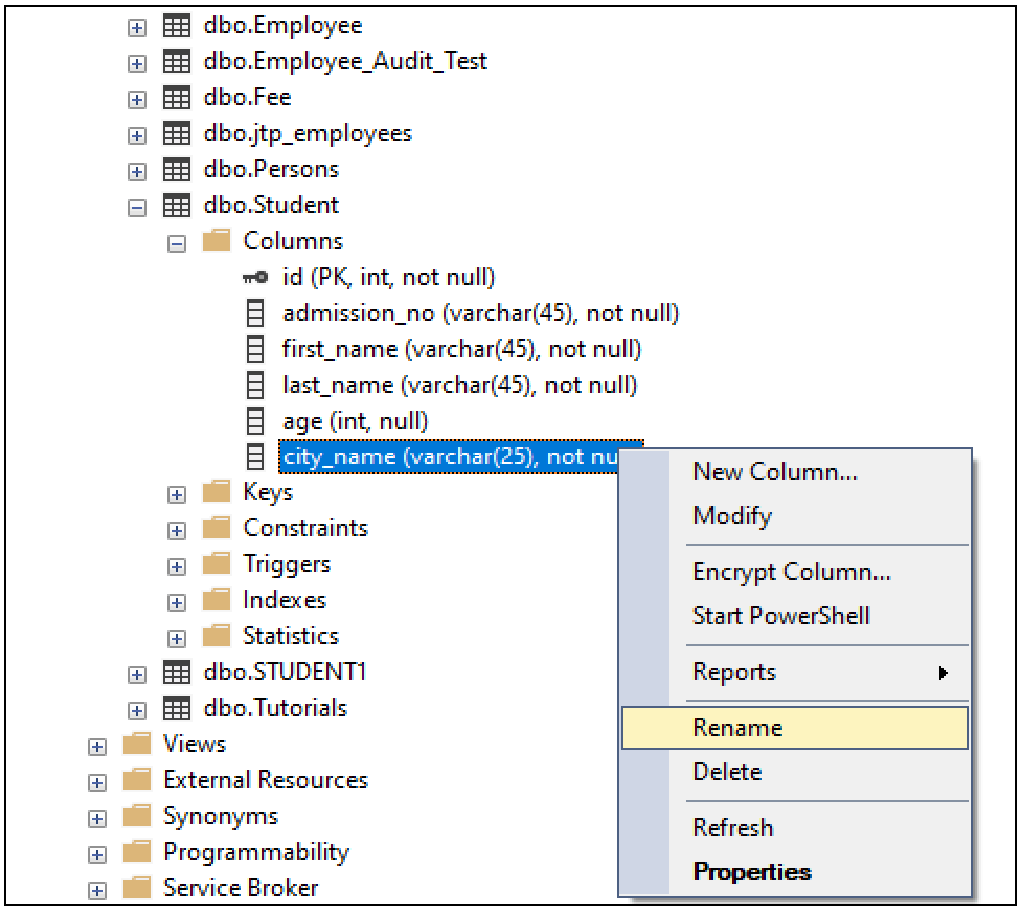
Step 1: Go to the Databases -> Tables menu and expand it.

Step 2: Select the desired table and expand it.

Step 3: Select the Columns and expand them.

Step 4: Right-click on the column name you want to change and choose the Rename option.

The below image explains all the above steps where we have chosen the 'student' table:



Step 5: Type a new name for your selected column.

Step 6: Refresh the database or table to finish the renaming steps.

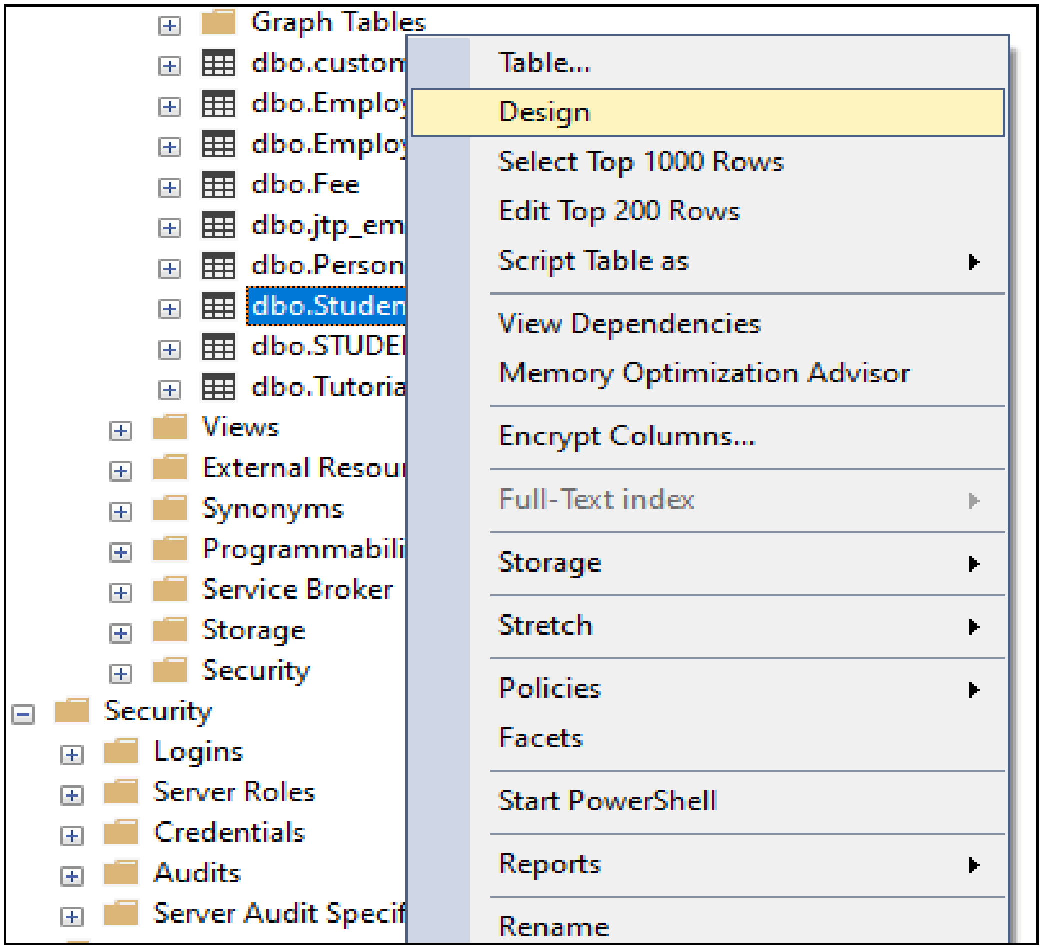
Step 7: Execute the SELECT statement to verify the changed column name.

Rename a column using Table Designer

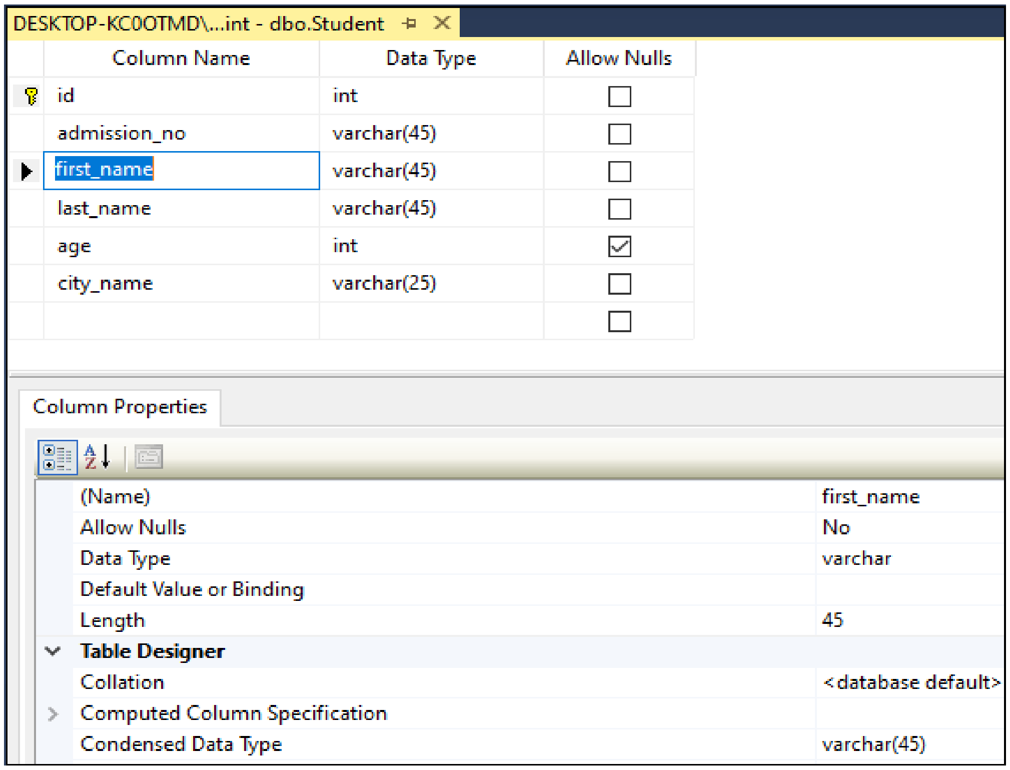
The following steps are used to change the column name using Object Explorer:

Step 1: Go to the Databases -> Tables menu and expand it.

Step 2: Select the desired table in which you want to change the column name, right-click on it, and choose the Design option from the context menu.



Step 3: Once we select the Design option, we will see the table in design mode like the below image.



Column Name	Data Type	Allow Nulls
id	int	☐
admission_no	varchar(45)	☐
first_name	varchar(45)	☐
last_name	varchar(45)	☐
age	int	✓
city_name	varchar(25)	☐
		☐

Column Properties

(Name)	first_name
Allow Nulls	No
Data Type	varchar
Default Value or Binding	
Length	45
Table Designer	
Collation	<database default>
Computed Column Specification	
Condensed Data Type	varchar(45)

Step 4: Select the column that you want to change and type the new name for this column. Let me rename the first_name column to f_name.

Step 5: To save the changes you made in the design window, click the Close button and then the Yes button in the displayed pop-up window.

We can save the changes made in the design window in another way by navigating to the File menu and click on the Save table name or press CTRL+S on the keyboard.

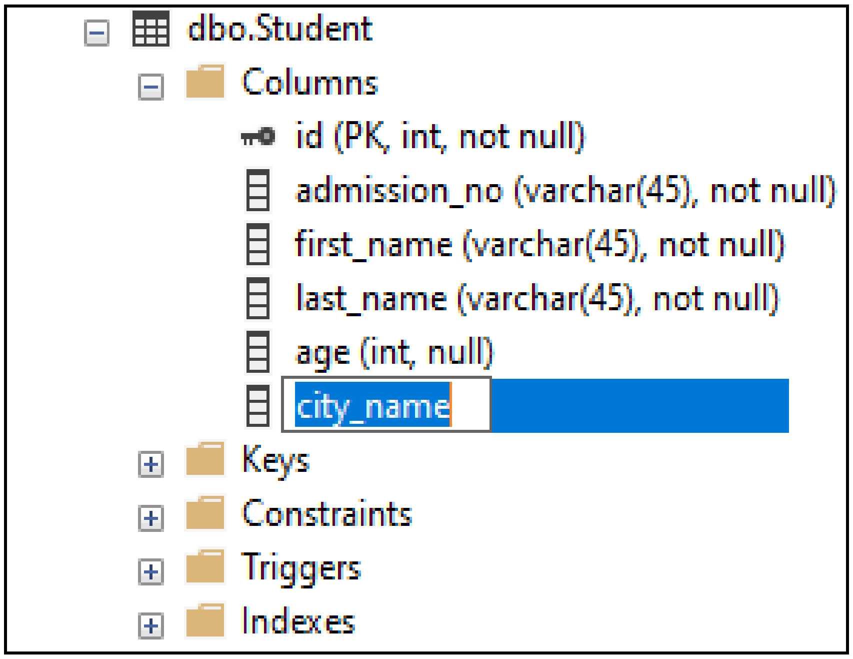
Step 6: Execute the SELECT statement to verify the changed column name

Double click on the column name

We can use the following steps to rename the column using a double click:

Step 1: Go to the Databases -> Tables -> Columns.

Step 2: Select the column name that you want to change and double-click. The below image explains it more clearly:


dbo.Student
Columns
id (PK, int, not null)
admission_no (varchar(45), not null)
first_name (varchar(45), not null)
last_name (varchar(45), not null)
age (int, null)
city_name
Keys
Constraints
Triggers
Indexes

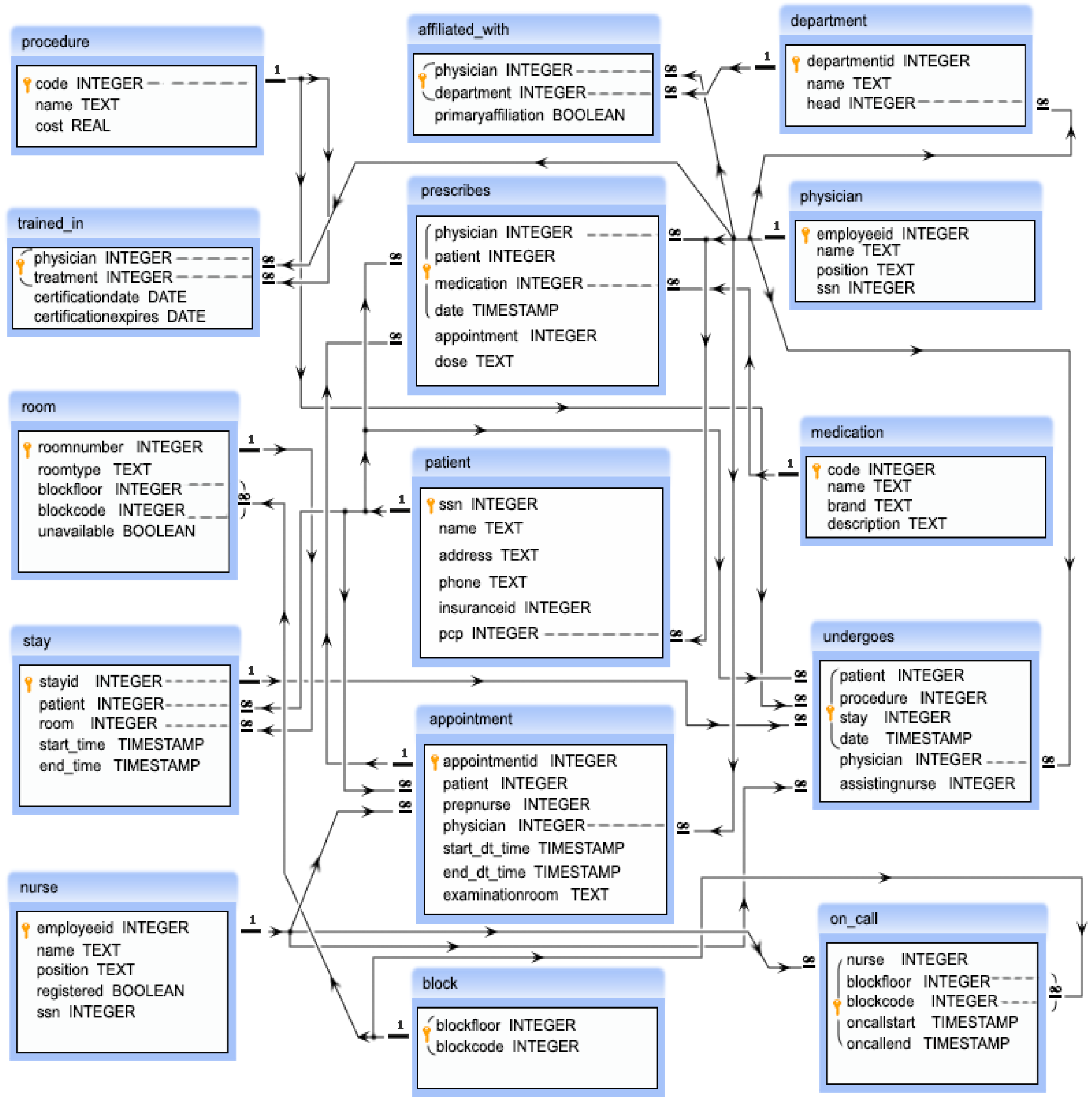
Step 3: Type a new name for your chosen column.

Step 4: Refresh the database or table to finish the renaming steps.

Step 5: Execute the SELECT statement to verify the changed column name.

4)Assume you need to create a large table for collecting patient information and retrieving the available doctors for them in a multi-speciality hospital • Create a table with at least 20 attributes • Assign primary key for a specific column •Generate 20 tuples. • Now select the doctors and their specialisation: o match patients with doctors' specialisation and retrieve

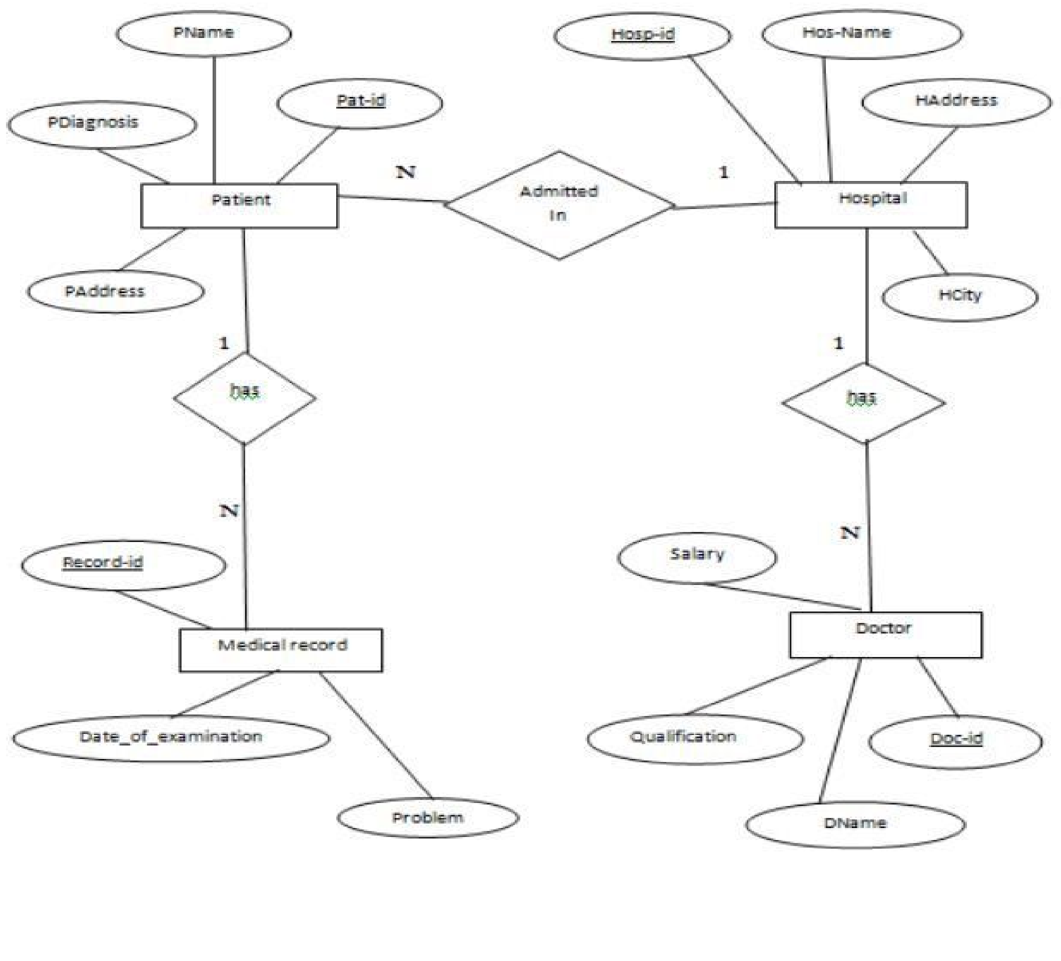
4A)



Assignment 2: Use the ER diagram for Healthcare Industry which yu have created

in previous assignment and construct a table consists of "CORONA" Prevention entities along with suitable attributes. Assume the various constraints and cardinality ratios among entities.
1.Now retrieve patients location wise
2. Now retrieve patients age wise
3. Now retrieve patients gender wise
Now retrieve patients with other diseases wise
Note: Generate 20 matching records.

A)


PName
Pat-id
PDiagnosis
Patient
PAddress
N
Admitted In
1
Hosp-id
Hos-Name
HAddress
Hospital
HCity
1
has
N
Record-id
Medical record
Date_of_examination
Problem
1
has
N
Salary
Doctor
Qualification
Doc-id
DName