

SQL ASSIGNMENT 1



Submitted by: Navyatha Godla

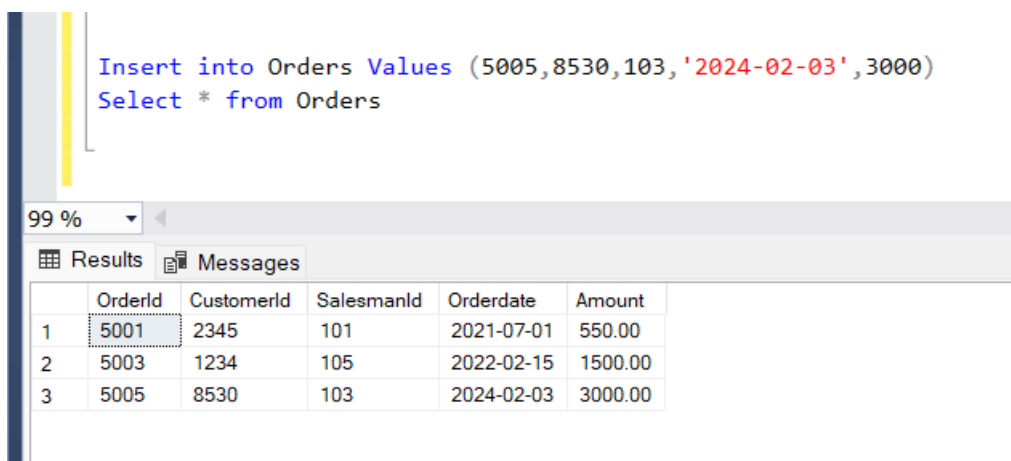
Submitted on: 05/02/2024.

Task: 1. Insert a new record in your Orders table.

Code:

```
Insert into Orders Values  
(5005,8530,103,'2024-02-03',3000)  
Select * from Orders
```

Result:

A screenshot of a SQL query execution interface. At the top, the query is displayed: "Insert into Orders Values (5005,8530,103,'2024-02-03',3000)" followed by "Select * from Orders". Below the query, there are tabs for "Results" and "Messages". The "Results" tab is active, showing a table with 6 columns: OrderId, CustomerId, SalesmanId, Orderdate, and Amount. The table contains three rows of data. The first row has OrderId 5001, CustomerId 2345, SalesmanId 101, Orderdate 2021-07-01, and Amount 550.00. The second row has OrderId 5003, CustomerId 1234, SalesmanId 105, Orderdate 2022-02-15, and Amount 1500.00. The third row has OrderId 5005, CustomerId 8530, SalesmanId 103, Orderdate 2024-02-03, and Amount 3000.00. The first row is highlighted with a dashed border.

	OrderId	CustomerId	SalesmanId	Orderdate	Amount
1	5001	2345	101	2021-07-01	550.00
2	5003	1234	105	2022-02-15	1500.00
3	5005	8530	103	2024-02-03	3000.00

Task: 2. Add Primary key constraint for SalesmanId column in Salesman table. Add default constraint for City column in Salesman table. Add Foreign key constraint for SalesmanId column in Customer table. Add not null constraint in Customer_name column for the Customer table.

Code:

```
Alter table Salesman alter Column SalesmanId
int not null;
```

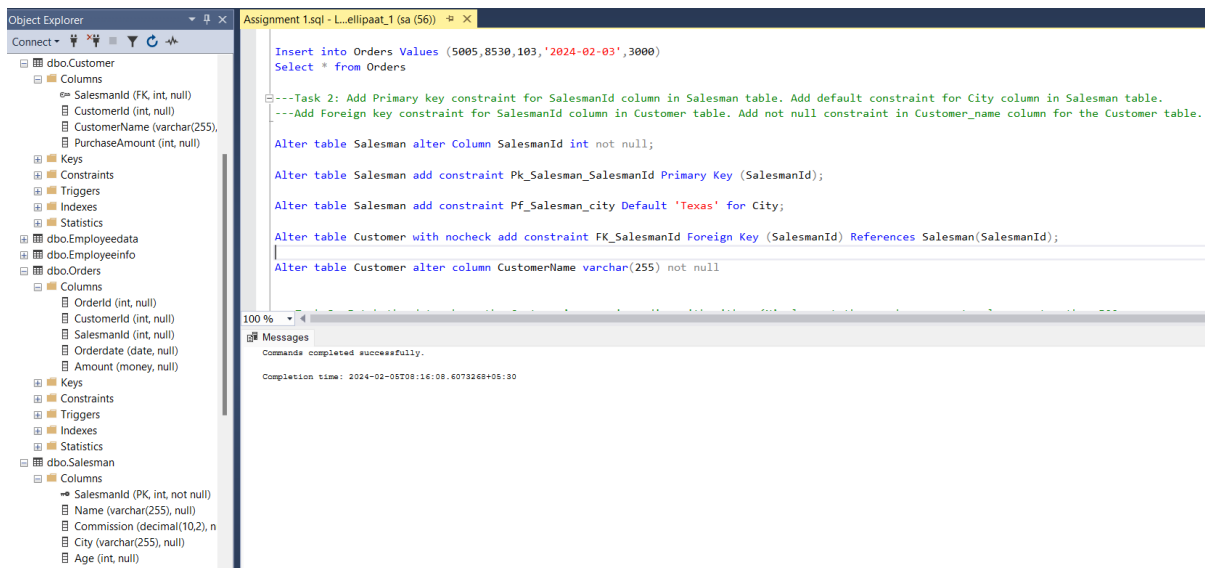
```
Alter table Salesman add constraint
Pk_Salesman_SalesmanId Primary Key
(SalesmanId);
```

```
Alter table Salesman add constraint
Pf_Salesman_city Default 'Texas' for City;
```

```
Alter table Customer with nocheck add
constraint FK_SalesmanId Foreign Key
(SalesmanId) References Salesman(SalesmanId);
```

```
Alter table Customer alter column
CustomerName varchar(255) not null
```

Results:



Object Explorer

Connect - [Server] [Database] [Schema] [Table]

dbo.Customer

- Columns
 - SalesmanId (FK, int, null)
 - CustomerId (int, null)
 - CustomerName (varchar(255), null)
 - PurchaseAmount (int, null)
- Keys
- Constraints
- Triggers
- Indexes
- Statistics

dbo.EmployeeData

dbo.EmployeeInfo

dbo.Orders

- Columns
 - OrderId (int, null)
 - CustomerId (int, null)
 - SalesmanId (int, null)
 - OrderDate (date, null)
 - Amount (money, null)
- Keys
- Constraints
- Triggers
- Indexes
- Statistics

dbo.Salesman

- Columns
 - SalesmanId (PK, int, not null)
 - Name (varchar(255), null)
 - Commission (decimal(10,2), null)
 - City (varchar(255), null)
 - Age (int, null)
- Keys
- Constraints
- Triggers
- Indexes
- Statistics

Assignment 1.sql - L..ellipaat_1 (sa (56))

```
Insert into Orders Values (5005,8530,103,'2024-02-03',3000)
Select * from Orders

----Task 2: Add Primary key constraint for SalesmanId column in Salesman table. Add default constraint for City column in Salesman table.
----Add Foreign key constraint for SalesmanId column in Customer table. Add not null constraint in Customer_name column for the Customer table.

Alter table Salesman alter Column SalesmanId int not null;

Alter table Salesman add constraint Pk_Salesman_SalesmanId Primary Key (SalesmanId);

Alter table Salesman add constraint Pf_Salesman_city Default 'Texas' for City;

Alter table Customer with nocheck add constraint FK_SalesmanId Foreign Key (SalesmanId) References Salesman(SalesmanId);

Alter table Customer alter column CustomerName varchar(255) not null
```

100 %

Messages

Commands completed successfully.

Completion time: 2024-02-03T09:16:08.6073268+05:30

Task: 3. Fetch the data where the Customer's name is ending with 'N' also get the purchase amount value greater than 500.

Code:

```
Select * from Customer where CustomerName  
like '%N' and PurchaseAmount>500
```

Result:

---Task 3: Fetch the data where the Customer's name is ending with either 'N' also get

```
Select * from Customer where CustomerName like '%N' and PurchaseAmount>500
```

---Task 4: Using SET operators, retrieve the first result with unique SalesmanId value

Results			
Messages			
SalesmanId	CustomerId	CustomerName	PurchaseAmount

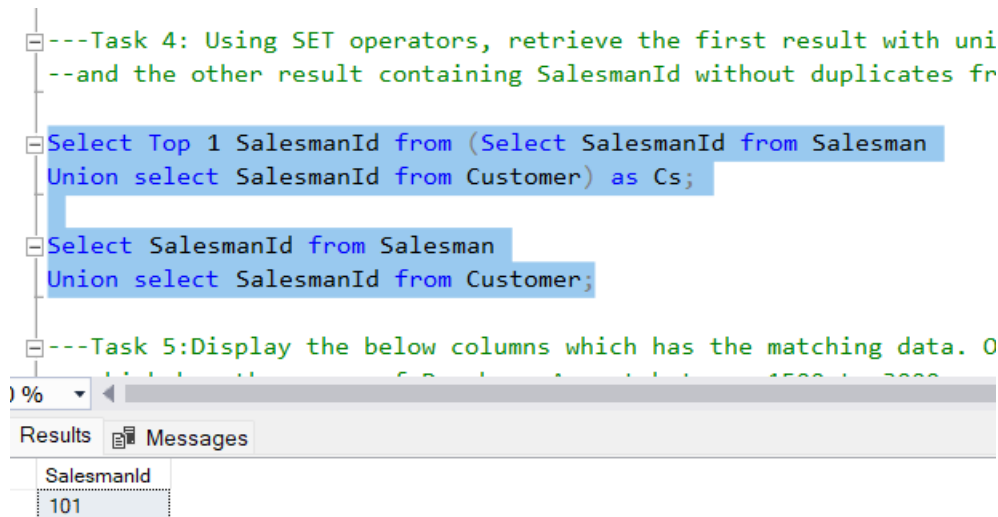
Task: 4. Using SET operators, retrieve the first result with unique SalesmanId values from two tables, and the other result containing SalesmanId without duplicates from two tables.

Code:

```
Select Top 1 SalesmanId from (Select  
SalesmanId from Salesman  
Union select SalesmanId from Customer) as Cs;
```

```
Select SalesmanId from Salesman  
Union select SalesmanId from Customer;
```

Result:

A screenshot of a SQL query execution interface. It shows a query editor with two queries. The first query is highlighted in blue and is: `Select Top 1 SalesmanId from (Select SalesmanId from Salesman Union select SalesmanId from Customer) as Cs;`. The second query is also highlighted in blue and is: `Select SalesmanId from Salesman Union select SalesmanId from Customer;`. Below the queries, there is a section for "Results" and "Messages". The "Results" tab is active, showing a table with one column "SalesmanId" and one row with the value "101".

---Task 4: Using SET operators, retrieve the first result with uni
--and the other result containing SalesmanId without duplicates fr

```
Select Top 1 SalesmanId from (Select SalesmanId from Salesman  
Union select SalesmanId from Customer) as Cs;
```

```
Select SalesmanId from Salesman  
Union select SalesmanId from Customer;
```

---Task 5: Display the below columns which has the matching data. 0

SalesmanId
101

SalesmanId
101
102
103
104
105
107
110

Task: 5. Display the below columns which has the matching data. Orderdate, Salesman Name, Customer Name, Commission, and City which has the range of Purchase Amount between 1500 to 3000.

Code:

```
Select o.Orderdate, s.Name as SalesmanName,  
c.CustomerName, s.Commission, s.city  
from Orders o  
join Salesman s on o.SalesmanId=s.SalesmanId  
join Customer c on o.CustomerId=c.CustomerId  
Where o.Amount between 1500 and 3000
```

Result:

```
Select o.Orderdate, s.Name as SalesmanName, c.CustomerName, s.Commission, s.city  
from Orders o  
join Salesman s on o.SalesmanId=s.SalesmanId  
join Customer c on o.CustomerId=c.CustomerId  
Where o.Amount between 1500 and 3000 |
```

---Task 6: Using right join fetch all the results from Salesman and Orders table

```
Select * from Salesman
```

results

Messages

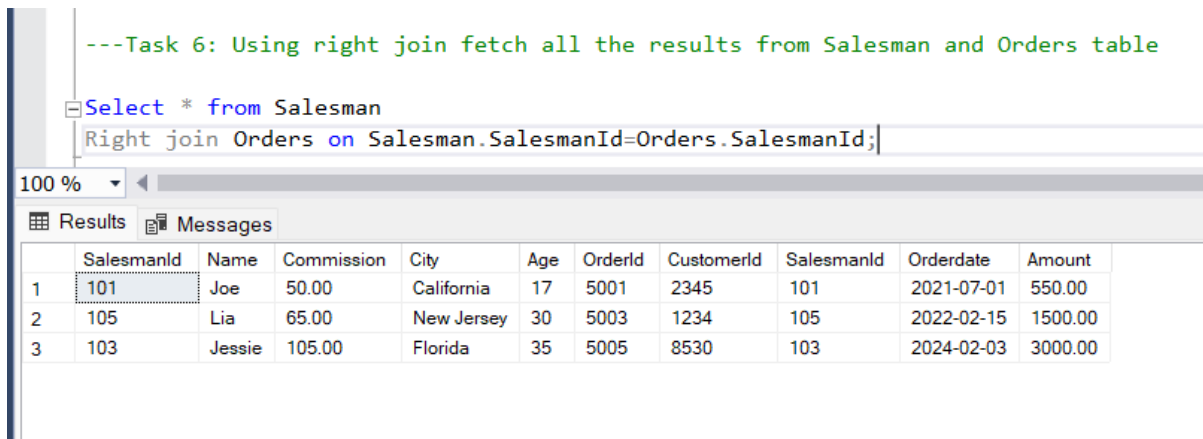
Orderdate	SalesmanName	CustomerName	Commission	city
-----------	--------------	--------------	------------	------

Task: 6. Using right join fetch all the results from Salesman and Orders table

Code:

```
Select * from Salesman  
Right join Orders on  
Salesman.SalesmanId=Orders.SalesmanId;
```

Result:

A screenshot of a SQL development environment. The top pane shows a SQL query: `---Task 6: Using right join fetch all the results from Salesman and Orders table` followed by `Select * from Salesman` and `Right join Orders on Salesman.SalesmanId=Orders.SalesmanId;`. The bottom pane shows the results of the query in a table with 11 columns: SalesmanId, Name, Commission, City, Age, OrderId, CustomerId, SalesmanId, Orderdate, and Amount. There are three rows of data.

---Task 6: Using right join fetch all the results from Salesman and Orders table										
<code>Select * from Salesman</code> <code>Right join Orders on Salesman.SalesmanId=Orders.SalesmanId;</code>										
100 %										
Results	Messages									
	SalesmanId	Name	Commission	City	Age	OrderId	CustomerId	SalesmanId	Orderdate	Amount
1	101	Joe	50.00	California	17	5001	2345	101	2021-07-01	550.00
2	105	Lia	65.00	New Jersey	30	5003	1234	105	2022-02-15	1500.00
3	103	Jessie	105.00	Florida	35	5005	8530	103	2024-02-03	3000.00