ASSIGNMENT CAPSTONE

CASPTONE PROJECT-2

By

Panjal Chandrakant Yakswami

**DATA SCIENCE**

**VERTOCITY**

# CASE-1:

**Problem Statement-1:**

|  |
| --- |
| * **Create database “Assignment” and use it for upcoming problems.** |

* **DATABASE:**
* It is an organized place where the data is collected and stored.
* SQL has Relational Database Management System model.
* **QUERY:**

|  |
| --- |
| CREATE DATABASE CAPSTONE\_2;  USE CAPSTONE\_2; |

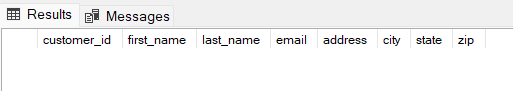
**Problem Statement-2:**

|  |
| --- |
| * **Create a customer table which comprises of these columns – ‘customer\_id’, ‘first\_name’, ‘last\_name’, ‘email’, ‘address’, ‘city’, ‘state’ and ‘zip.** |

* **CREATE:**
  + It is a DDL command which changes the structure of the table.
  + It is used to make a table in the database.
* **QUERY:**

|  |
| --- |
| CREATE TABLE customer(  customer\_id INT,  first\_name varchar(25),  last\_name varchar(25),  email varchar(30),  address varchar(30),  city varchar(20),  state varchar(20),  zip int); |

* **RESULT:**

****

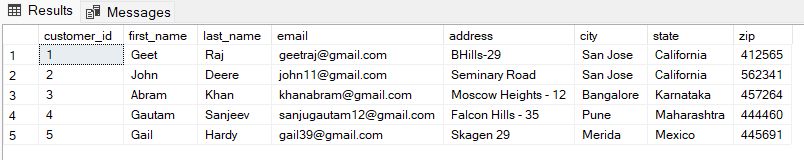
**Problem Statement-3:**

|  |
| --- |
| * **Insert 5 new records into the table as mentioned below:** |

* **QUERY:**

|  |
| --- |
| INSERT INTO customer  values (1,'Geet','Raj','geetraj@gmail.com','BHills-29','San Jose','California',412565),  (2,'John','Deere','john11@gmail.com','Seminary Road','San Jose','California',562341),  (3,'Abram','Khan','khanabram@gmail.com','Moscow Heights - 12','Bangalore','Karnataka',457264),  (4,'Gautam','Sanjeev','sanjugautam12@gmail.com','Falcon Hills - 35','Pune','Maharashtra',444460),  (5,'Gail','Hardy','gail39@gmail.com','Skagen 29','Merida','Mexico',445691);  SELECT \* FROM customer; |

* **RESULT:**

****

* **INSERT:**
  + It is a DML (Data Manipulation Language).
  + It is used to insert the values any data into the table and it cannot change the structure of the table.

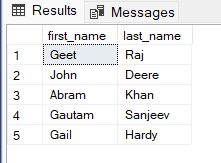
**Problem Statement-4:**

|  |
| --- |
| * **Select only the ‘first\_name’ and ‘last\_name’ columns from the customer table.** |

* **QUERY:**

|  |
| --- |
| SELECT first\_name, last\_name from customer; |

* **RESULT:**

****

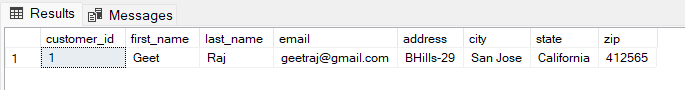
**Problem Statement-5:**

|  |
| --- |
| * **Select those records where ‘first\_name’ starts with “G” and city is ‘San Jose’.** |

* **QUERY:**

|  |
| --- |
| SELECT \* FROM customer  WHERE first\_name like 'G%' and city ='San Jose'; |

* **RESULT:**

****

# CASE-2:

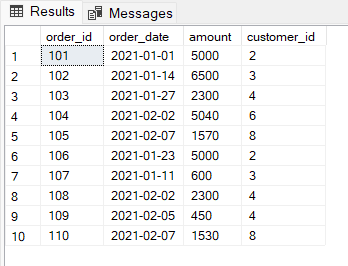
**Problem Statement-1:**

|  |
| --- |
| * **Create an ‘Orders’ table which comprises of these columns – ‘order\_id’, ‘order\_date’, ‘amount’, ‘customer\_id’ and insert.** |

* **QUERY:**

|  |
| --- |
| CREATE TABLE Orders (  order\_id INT,  order\_date DATE,  amount INT,  customer\_id INT);  INSERT INTO Orders  VALUES (101, '01/01/2021', 5000, 2),  (102, '01/14/2021', 6500, 3),  (103, '01/27/2021', 2300, 4),  (104, '02/02/2021', 5040, 6),  (105, '02/07/2021', 1570, 8),  (106, '01/23/2021', 5000, 2),  (107, '01/11/2021', 600, 3),  (108, '02/02/2021', 2300, 4),  (109, '02/05/2021', 450, 4),  (110, '02/07/2021', 1530, 8);  SELECT \* FROM Orders; |

* **RESULT:**

****

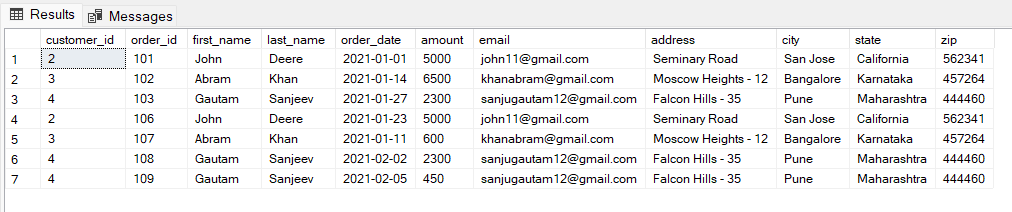
**Problem Statement-2:**

|  |
| --- |
| * **Make an inner join on ‘customer’ and ‘Orders’ tables on the ‘customer\_id’ column.** |

* **JOINS:**
  + Join is used to combine two or more tables with each other.
  + Following are the types of join used in SQL:
    - Inner Join
    - Left Join
    - Right Join
    - Full Join
    - Cross Join or Cartesian Join
* **QUERY:**

|  |
| --- |
| SELECT c.customer\_id, o.order\_id, c.first\_name, c.last\_name, o.order\_date, o.amount,  c.email, c.address, c.city, c.state, c.zip  FROM customer c  INNER JOIN Orders o  ON c.customer\_id = o.customer\_id; |

* **RESULT:**

****

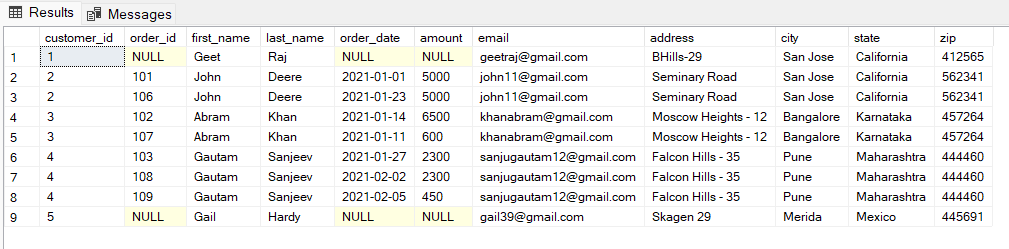
**Problem Statement-3(A):**

|  |
| --- |
| * **Make a left join on ‘customer’ and ‘Orders’ tables on the ‘customer\_id’ column.** |

* **QUERY:**

|  |
| --- |
| SELECT c.customer\_id, o.order\_id, c.first\_name, c.last\_name, o.order\_date, o.amount,  c.email, c.address, c.city, c.state, c.zip  FROM customer c  LEFT JOIN Orders o  ON c.customer\_id = o.customer\_id; |

* **RESULT:**

****

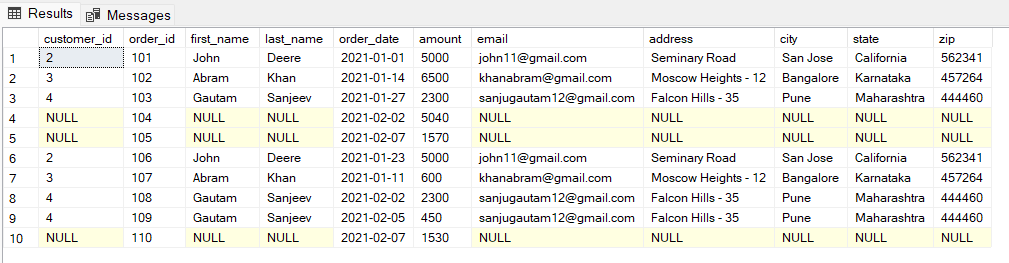
**Problem Statement-3(B):**

|  |
| --- |
| * **Make a right join on ‘customer’ and ‘Orders’ tables on the ‘customer\_id’ column.** |

* **QUERY:**

|  |
| --- |
| SELECT c.customer\_id, o.order\_id, c.first\_name, c.last\_name, o.order\_date, o.amount,  c.email, c.address, c.city, c.state, c.zip  FROM customer c  RIGHT JOIN Orders o  ON c.customer\_id = o.customer\_id; |

* **RESULT:**

****

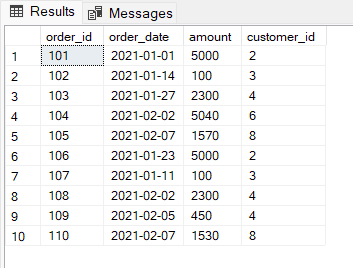
**Problem Statement-4:**

|  |
| --- |
| * **Update ‘Orders’ table, set the amount to be 100 where ‘customer\_id’ is 3.** |

* **UPDATE:**
  + It is a DML command which is used to update any already existing data in the table.
  + This command also do not change the structure of the table.
* **QUERY:**

|  |
| --- |
| UPDATE Orders  SET amount = 100  WHERE customer\_id = 3;  SELECT \* FROM Orders; |

* **RESULT:**

****

# CASE-3:

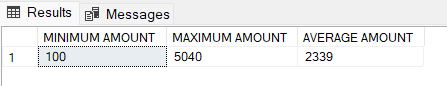
**Problem Statement-1:**

|  |
| --- |
| * **Use the inbuilt functions and find the minimum, maximum and average amount from the orders table.** |

* **In-Built Function:**
  + These are the functions which are already provided by the SQL server.
  + These are basically aggregate function, scalar function or table function.
  + There are “String Function”, “DateTime Function”, “Numeric Function” and “Conversion Function.
* **QUERY:**

|  |
| --- |
| SELECT MIN(amount) AS 'MINIMUM AMOUNT', MAX(amount) AS 'MAXIMUM AMOUNT',  AVG(amount) AS 'AVERAGE AMOUNT' FROM Orders; |

* **RESULT:**

****

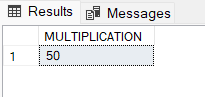
**Problem Statement-2:**

|  |
| --- |
| * **Create a user-defined function “PROD”, which will multiply the given number with 10.** |

* **USER-DEFINED FUNCTION:**
  + These functions are defined by the user for faster execution.
  + It generally accepts some parameters, perform complex calculations and gives the results.
  + Following are the types of User-Defined Function:
    - Scalar Function
    - Table Function
* **QUERY:**

|  |
| --- |
| CREATE FUNCTION PROD (@NUM INT)  RETURNS INT  AS  BEGIN  RETURN(SELECT @NUM\*10)  END;  SELECT DBO.PROD (5) AS 'MULTIPLICATION'; |

* **RESULT:**

****

**Problem Statement-3:**

|  |
| --- |
| * **Use the case statement to check if 100 is less than 200, greater than 200 or equal to 200 and print the corresponding value.** |

* **QUERY:**

|  |
| --- |
| CREATE FUNCTION COMPARISON (@N INT)  RETURNS VARCHAR(30)  AS  BEGIN  DECLARE @COMP AS VARCHAR(30) = CASE  WHEN @N<200 THEN CONCAT (@N, ' is less than 200.')  WHEN @N=200 THEN CONCAT (@N, ' equal to 200.')  WHEN @N>200 THEN CONCAT (@N, ' is greater than 200.')  END;  RETURN @COMP  END;  PRINT DBO.COMPARISON (100); |

* **RESULT:**

****

# CASE-4:

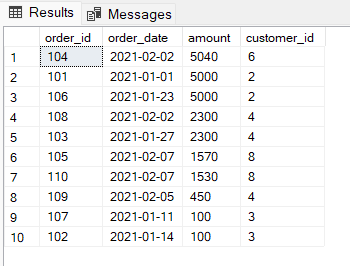
**Problem Statement-1:**

|  |
| --- |
| * **Arrange the ‘Orders’ dataset in decreasing order of amount.** |

* **ORDER BY:**
  + This command works like sorting the data based through ascending or descending trend.
  + By default the SQL server takes ascending form.
* **QUERY:**

|  |
| --- |
| SELECT \* FROM Orders  ORDER BY (amount) DESC; |

* **RESULT:**

****

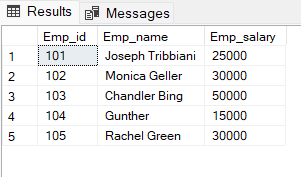
**Problem Statement-2:**

|  |
| --- |
| * **Create a table with name ‘Employee\_details1’ and comprising of these columns- ‘Emp\_id’, ‘Emp\_name’, ‘Emp\_salary’. Enter details as below.** |

* **QUERY:**

|  |
| --- |
| CREATE TABLE Employee\_details1 (  Emp\_id INT,  Emp\_name VARCHAR(40),  Emp\_salary int);  INSERT INTO Employee\_details1  VALUES (101, 'Joseph Tribbiani', 25000),  (102, 'Monica Geller', 30000),  (103, 'Chandler Bing', 50000),  (104, 'Gunther', 15000),  (105, 'Rachel Green', 30000);  SELECT \* FROM Employee\_details1; |

* **RESULT:**

****

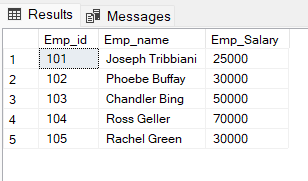
**Problem Statement-3:**

|  |
| --- |
| * **Create a table with name ‘Employee\_details2’ and comprising of these columns- ‘Emp\_id’, ‘Emp\_name’, ‘Emp\_salary’. Enter details as below.** |

* **QUERY:**

|  |
| --- |
| CREATE TABLE Employee\_details2 (  Emp\_id INT,  Emp\_name VARCHAR(40),  Emp\_Salary INT);    INSERT INTO Employee\_details2  VALUES (101, 'Joseph Tribbiani', 25000),  (102, 'Phoebe Buffay', 30000),  (103, 'Chandler Bing', 50000),  (104, 'Ross Geller', 70000),  (105, 'Rachel Green', 30000);  SELECT \* FROM Employee\_details2; |

* **RESULT:**

****

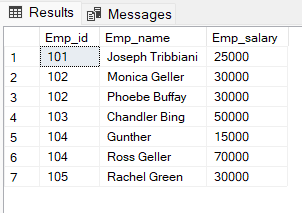
**Problem Statement-4:**

|  |
| --- |
| * **Apply the Union Operator on these two tables.** |

* **QUERY:**

|  |
| --- |
| SELECT \* FROM Employee\_details1  UNION  SELECT \* FROM Employee\_details2; |

* **RESULT:**

****

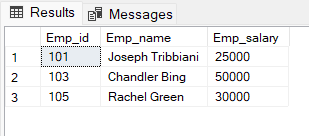
**Problem Statement-5:**

|  |
| --- |
| * **Apply the Intersect Operator on these two tables.** |

* **QUERY:**

|  |
| --- |
| SELECT \* FROM Employee\_details1  INTERSECT  SELECT \* FROM Employee\_details2; |

* **RESULT:**

****

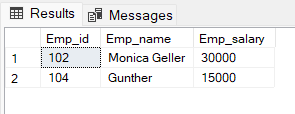
**Problem Statement-6:**

|  |
| --- |
| * **Apply the Except Operator on these two tables.** |

* **QUERY:**

|  |
| --- |
| SELECT \* FROM Employee\_details1  EXCEPT  SELECT \* FROM Employee\_details2; |

* **RESULT:**

****

# CASE-5:

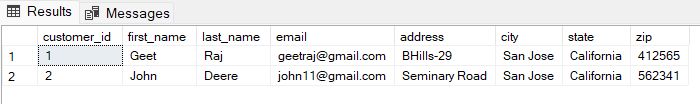
**Problem Statement-1:**

|  |
| --- |
| * **Create a view named ‘customer\_san\_jose’ which comprises of only those customers who are from San Jose.** |

* **QUERY:**

|  |
| --- |
| CREATE VIEW customer\_san\_jose  AS  SELECT \* FROM customer  WHERE city='San Jose';  SELECT \* FROM customer\_san\_jose; |

* **RESULT:**

****

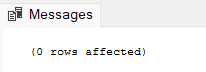
**Problem Statement-2:**

|  |
| --- |
| * **Inside a transaction, update the first\_name of the customer to Francis, where the last name is Jordan.** |

* **QUERY:**

|  |
| --- |
| BEGIN TRANSACTION  UPDATE customer  SET first\_name= 'Francis'  WHERE last\_name= 'Jordan'; |

* **RESULT:**

****

**Problem Statement-2(A):**

|  |
| --- |
| * **Rollback the Transaction.** |

* **QUERY:**

|  |
| --- |
| ROLLBACK TRANSACTION; |

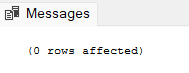
**Problem Statement-2(B):**

|  |
| --- |
| * **Set the first name of customer to Alex where the last name is Jordan.** |

* **QUERY:**

|  |
| --- |
| UPDATE customer  SET first\_name= 'Alex'  WHERE last\_name= 'Jordan'; |

* **RESULT:**

****

**Thank You…**