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.**

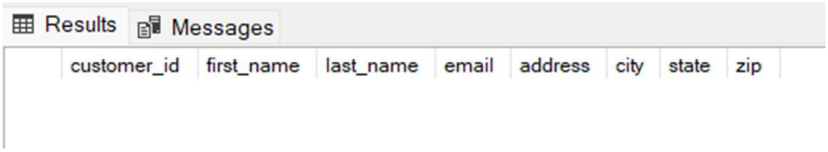
➢ CREATE:

- 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:**

Results | Messages

	customer_id	first_name	last_name	email	address	city	state	zip
1	1	Geet	Raj	geetraj@gmail.com	BHills-29	San Jose	California	412565
2	2	John	Deere	john11@gmail.com	Seminary Road	San Jose	California	562341
3	3	Abram	Khan	khanabram@gmail.com	Moscow Heights - 12	Bangalore	Karnataka	457264
4	4	Gautam	Sanjeev	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
5	5	Gail	Hardy	gail39@gmail.com	Skagen 29	Merida	Mexico	445691

➢ 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:**

Results | Messages

	first_name	last_name
1	Geet	Raj
2	John	Deere
3	Abram	Khan
4	Gautam	Sanjeev
5	Gail	Hardy

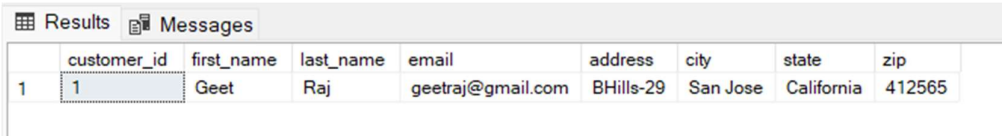
Problem Statement-5:

❖ **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:**



Results | Messages

	customer_id	first_name	last_name	email	address	city	state	zip
1	1	Geet	Raj	geetraj@gmail.com	BHills-29	San Jose	California	412565

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:**

Results | Messages

	order_id	order_date	amount	customer_id
1	101	2021-01-01	5000	2
2	102	2021-01-14	6500	3
3	103	2021-01-27	2300	4
4	104	2021-02-02	5040	6
5	105	2021-02-07	1570	8
6	106	2021-01-23	5000	2
7	107	2021-01-11	600	3
8	108	2021-02-02	2300	4
9	109	2021-02-05	450	4
10	110	2021-02-07	1530	8

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:**

Results | Messages

	customer_id	order_id	first_name	last_name	order_date	amount	email	address	city	state	zip
1	2	101	John	Deere	2021-01-01	5000	john11@gmail.com	Seminary Road	San Jose	California	562341
2	3	102	Abram	Khan	2021-01-14	6500	khanabram@gmail.com	Moscow Heights - 12	Bangalore	Karnataka	457264
3	4	103	Gautam	Sanjeev	2021-01-27	2300	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
4	2	106	John	Deere	2021-01-23	5000	john11@gmail.com	Seminary Road	San Jose	California	562341
5	3	107	Abram	Khan	2021-01-11	600	khanabram@gmail.com	Moscow Heights - 12	Bangalore	Karnataka	457264
6	4	108	Gautam	Sanjeev	2021-02-02	2300	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
7	4	109	Gautam	Sanjeev	2021-02-05	450	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460

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:**

Results | Messages

	customer_id	order_id	first_name	last_name	order_date	amount	email	address	city	state	zip
1	1	NULL	Geet	Raj	NULL	NULL	geetraj@gmail.com	BHills-29	San Jose	California	412565
2	2	101	John	Deere	2021-01-01	5000	john11@gmail.com	Seminary Road	San Jose	California	562341
3	2	106	John	Deere	2021-01-23	5000	john11@gmail.com	Seminary Road	San Jose	California	562341
4	3	102	Abram	Khan	2021-01-14	6500	khanabram@gmail.com	Moscow Heights - 12	Bangalore	Karnataka	457264
5	3	107	Abram	Khan	2021-01-11	600	khanabram@gmail.com	Moscow Heights - 12	Bangalore	Karnataka	457264
6	4	103	Gautam	Sanjeev	2021-01-27	2300	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
7	4	108	Gautam	Sanjeev	2021-02-02	2300	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
8	4	109	Gautam	Sanjeev	2021-02-05	450	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
9	5	NULL	Gail	Hardy	NULL	NULL	gail39@gmail.com	Skagen 29	Merida	Mexico	445691

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:**

Results | Messages

	customer_id	order_id	first_name	last_name	order_date	amount	email	address	city	state	zip
1	2	101	John	Deere	2021-01-01	5000	john11@gmail.com	Seminary Road	San Jose	California	562341
2	3	102	Abram	Khan	2021-01-14	6500	khanabram@gmail.com	Moscow Heights - 12	Bangalore	Karnataka	457264
3	4	103	Gautam	Sanjeev	2021-01-27	2300	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
4	NULL	104	NULL	NULL	2021-02-02	5040	NULL	NULL	NULL	NULL	NULL
5	NULL	105	NULL	NULL	2021-02-07	1570	NULL	NULL	NULL	NULL	NULL
6	2	106	John	Deere	2021-01-23	5000	john11@gmail.com	Seminary Road	San Jose	California	562341
7	3	107	Abram	Khan	2021-01-11	600	khanabram@gmail.com	Moscow Heights - 12	Bangalore	Karnataka	457264
8	4	108	Gautam	Sanjeev	2021-02-02	2300	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
9	4	109	Gautam	Sanjeev	2021-02-05	450	sanjugautam12@gmail.com	Falcon Hills - 35	Pune	Maharashtra	444460
10	NULL	110	NULL	NULL	2021-02-07	1530	NULL	NULL	NULL	NULL	NULL

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:**

Results | Messages

	order_id	order_date	amount	customer_id
1	101	2021-01-01	5000	2
2	102	2021-01-14	100	3
3	103	2021-01-27	2300	4
4	104	2021-02-02	5040	6
5	105	2021-02-07	1570	8
6	106	2021-01-23	5000	2
7	107	2021-01-11	100	3
8	108	2021-02-02	2300	4
9	109	2021-02-05	450	4
10	110	2021-02-07	1530	8

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:**

Results | Messages

	MINIMUM AMOUNT	MAXIMUM AMOUNT	AVERAGE AMOUNT
1	100	5040	2339

Problem Statement-2:

❖ **Create a user-defined function "PROD", which will multiply the given number with 10.**

➢ **USER-DEFINED FUNCTION:**

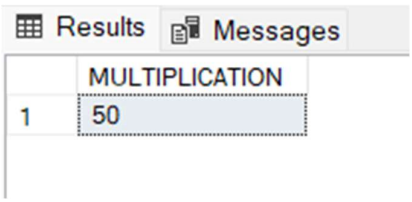
- These functions are defined by the user for faster execution.
- It generally accepts some parameters, perform complex calculations and gives the results.
- Following are the types of User-Defined Function:
  - Scalar Function
  - Table Function

- **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:**

Results | Messages

	order_id	order_date	amount	customer_id
1	104	2021-02-02	5040	6
2	101	2021-01-01	5000	2
3	106	2021-01-23	5000	2
4	108	2021-02-02	2300	4
5	103	2021-01-27	2300	4
6	105	2021-02-07	1570	8
7	110	2021-02-07	1530	8
8	109	2021-02-05	450	4
9	107	2021-01-11	100	3
10	102	2021-01-14	100	3

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:**

Results | Messages

	Emp_id	Emp_name	Emp_salary
1	101	Joseph Tribbiani	25000
2	102	Monica Geller	30000
3	103	Chandler Bing	50000
4	104	Gunther	15000
5	105	Rachel Green	30000

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:**

Results | Messages

	Emp_id	Emp_name	Emp_Salary
1	101	Joseph Tribbiani	25000
2	102	Phoebe Buffay	30000
3	103	Chandler Bing	50000
4	104	Ross Geller	70000
5	105	Rachel Green	30000

Problem Statement-4:

❖ **Apply the Union Operator on these two tables.**

- **QUERY:**

```
SELECT * FROM Employee_details1
UNION
SELECT * FROM Employee_details2;
```

- **RESULT:**

Results | Messages

	Emp_id	Emp_name	Emp_salary
1	101	Joseph Tribbiani	25000
2	102	Monica Geller	30000
3	102	Phoebe Buffay	30000
4	103	Chandler Bing	50000
5	104	Gunther	15000
6	104	Ross Geller	70000
7	105	Rachel Green	30000

Problem Statement-5:

❖ **Apply the Intersect Operator on these two tables.**

- **QUERY:**

```
SELECT * FROM Employee_details1
INTERSECT
SELECT * FROM Employee_details2;
```

- **RESULT:**

Results | Messages

	Emp_id	Emp_name	Emp_salary
1	101	Joseph Tribbiani	25000
2	103	Chandler Bing	50000
3	105	Rachel Green	30000

Problem Statement-6:

❖ **Apply the Except Operator on these two tables.**

- **QUERY:**

```
SELECT * FROM Employee_details1
EXCEPT
SELECT * FROM Employee_details2;
```

- **RESULT:**

Results | Messages

	Emp_id	Emp_name	Emp_salary
1	102	Monica Geller	30000
2	104	Gunther	15000

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:**

Results | Messages

	customer_id	first_name	last_name	email	address	city	state	zip
1	1	Geet	Raj	geetraj@gmail.com	BHills-29	San Jose	California	412565
2	2	John	Deere	john11@gmail.com	Seminary Road	San Jose	California	562341

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:**

Messages

```
(0 rows affected)
```

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…