|  |  |
| --- | --- |
| **Practical No: 1** | **Roll No:** |
| **Objective: Creating database tables and using data types.  Create table • Modify table • Drop table** | **Date:** |

**1. Create Database**

CREATE DATABASE MCA\_RollNo;

**2. Drop Database**

DROP DATABASE MCA\_RollNo;

**3. Create table –**

CREATE TABLE STUDENT

(

RollNo INT IDENTITY(1,1),

Name VARCHAR(50),

Notes text,

Photo image,

JoiningDate date

);

**4. Modify Table**

**4.1 Add Column**

ALTER TABLE Student

ADD PreviousSchool VARCHAR (255) NOT NULL;

**4.2 Change data type**

CREATE TABLE t1 (c INT);

INSERT INTO t1 VALUES (1);

ALTER TABLE t1 ALTER COLUMN c VARCHAR (2);

INSERT INTO t1 VALUES ('@');

ALTER TABLE t1 ALTER COLUMN c INT;

Conversion failed when converting the varchar value '@' to data type int.

**4.3 Change data Size**

CREATE TABLE t2 (c VARCHAR(10));

ALTER TABLE t2 ALTER COLUMN c VARCHAR (50);

ALTER TABLE t2 ALTER COLUMN c VARCHAR (5);

String or binary data would be truncated.

**4.4 Add a NOT NULL constraint to a nullable column**

CREATE TABLE t3 (c VARCHAR(50));

INSERT INTO t3 VALUES ('Some value'),

INSERT INTO t3 VALUES (NULL);

UPDATE t3

SET c = ''

WHERE

c IS NULL;

Now add NULL Constraint

ALTER TABLE t3 ALTER COLUMN c VARCHAR (100) NOT NULL;

**4.5 Drop Column**

CREATE TABLE Product(

ProductNo int,

price DEC(10,2) NOT NULL CONSTRAINT ck\_positive\_price CHECK(price >= 0),

discount DEC(10,2) NOT NULL

);

ALTER TABLE Product DROP COLUMN price;

error: The object 'ck\_positive\_price' is dependent on column 'price'.

ALTER TABLE Product DROP CONSTRAINT ck\_positive\_price;

ALTER TABLE Product DROP COLUMN price, discount; (Deleting 2 columns)

**5. Computed Columns**

CREATE TABLE persons

(

person\_id INT PRIMARY KEY IDENTITY,

first\_name NVARCHAR(100) NOT NULL,

last\_name NVARCHAR(100) NOT NULL,

dob DATE

);

CREATE TABLE persons

(

person\_id INT PRIMARY KEY IDENTITY,

first\_name NVARCHAR(100) NOT NULL,

last\_name NVARCHAR(100) NOT NULL,

dob DATE

);

SELECT

person\_id,

first\_name + ' ' + last\_name AS full\_name,

dob

FROM

persons

ALTER TABLE persons ADD full\_name AS (first\_name + ' ' + last\_name);

(every time computes)

ALTER TABLE persons ADD full\_name AS (first\_name + ' ' + last\_name) PERSISTED;

(physically stores data)

**6. Rename Table**

EXEC sp\_rename 'Students', 'NewStudent';

**7. Drop Table**

DROP TABLE [IF EXISTS] Department;

Could not drop object 'Department' because it is referenced by a FOREIGN KEY constraint.

Code language: SQL (Structured Query Language) (sql)

DROP TABLE Employee, Department;

**8.TRUNCATE Table**

TRUNCATE TABLE Student;

Difference Between Delete and Truncate

1. Where clause present with delete , not with Truncate

2. Transaction Log

3. Identity reset

|  |  |
| --- | --- |
| **Practical No: 2** | **Roll No:** |
| **Objective: Data Manipulation - Adding/Modify/Delete data using Insert/ Update/ Delete** | **Date:** |

**Create Table –**

CREATE TABLE Employee

(

Id INT,

Name VARCHAR(50),

Salary MONEY

);

**1. Adding data with Insert –**

INSERT INTO Employee VALUES (1001, 'Leena Lele', 20000);

INSERT INTO Employee VALUES (1002, 'Pawan Pawar', 30000);

INSERT INTO Employee VALUES (1003, 'Sunil Nikumbh', 40000);

(Show result)

Try Multiple Inserts

**2. Modify data with Update –**

UPDATE Employee SET SALARY=15000 WHERE Id=1003;

(Show result)

**3. Deleting records with Delete –**

DELETE FROM Employee WHERE Id=1002;  
(Show result)

|  |  |
| --- | --- |
| **Practical No: 3** | **Roll No:** |
| **Objective: Implementing the different types of Constraints.  NULL and NOT NULL, Primary Key Constraint, Foreign Key Constraint, Unique Constraint, Check Constraint, Default Constraint** | **Date:** |

**Create Table Department –**

Create Table Department (

Id int PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

HOD VARCHAR(50) NOT NULL

);

**Create Table Employee –**

CREATE TABLE Employee

(

Id NUMERIC(4) PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Email VARCHAR(30) UNIQUE,

SALARY MONEY CHECK(SALARY >=5000),

City VARCHAR(255) DEFAULT 'Jalgaon'

DepartmentId INT FOREIGN KEY REFERENCES Department(Id)

);

(Try Different checks, default values, create 2 tables, try inserting duplicate vaules in Primary key, Unique key, insert a foreign key column value which is absent in department, multiple column check constraint)

Add Constraint and Remove Constraint

|  |  |  |
| --- | --- | --- |
|  | Example 1 | Example 2 |
| Add | ALTER TABLE Employee  ADD CONSTRAINT city\_default  DEFAULT 'Jalgaon' FOR City; | ALTER TABLE Products  ADD CONSTRAINT positive\_price CHECK(unit\_price > 0); |
| Remove | ALTER TABLE Employee  ALTER COLUMN City DROP DEFAULT; | ALTER TABLE Products  DROP CONSTRAINT positive\_price; |

Referential Integrity

Foreign Key Example

(Update On , Delete On)

|  |  |
| --- | --- |
| **Practical No: 4** | **Roll No:** |
| **Objective: Retrieving Data Using following clauses.**  **• Simple Select clause**  **• Accessing specific data with Where**  **• Ordered By**  **• Distinct**  **• Group By** | **Date:** |

**Create Table Employee –**

(Write create table query)

**Adding data with Insert –**

(Write insertinto Statements)

**Simple Select Clause –**

SELECT \* FROM Employee;

SELECT Id, Name, City, Salary, Department from Employee;

SELECT TOP 5 Id, Name, City, Salary, Department from Employee ORDER BY Salary DESC;

(Show Result After every Query)

**Accessing specific data with Where –**

SELECT \* FROM Employee WHERE Salary <= 30000;

SELECT \* FROM Employee WHERE Salary >= 15000 AND Salary <= 30000;

SELECT \* FROM Employee WHERE Department = 'CS' OR Department = 'Management';

SELECT \* FROM Employee WHERE Department in ( 'CS','Management', 'Physics');

SELECT \* FROM Employee WHERE Salary BETWEEN 15000 AND 30000;

SELECT \* FROM Employee WHERE Phone IS NULL;

SELECT \* FROM Employee WHERE Phone IS NOT NULL;

SELECT \* FROM Employee WHERE first\_name LIKE 'p%';

SELECT \* FROM Employee WHERE first\_name LIKE '%s';

SELECT \* FROM Employee WHERE first\_name LIKE 'l%a';

SELECT \* FROM Employee WHERE first\_name LIKE '\_eena';

SELECT \* FROM Employee WHERE first\_name LIKE '[A-P]%';

SELECT \* FROM Employee WHERE first\_name LIKE '[^A-P]%';

(Simillarly write NOT LIKE queries)

SELECT \* FROM Employee WHERE first\_name NOT LIKE 'p%';

SELECT \* FROM Employee ORDER BY Salary;

SELECT \* FROM Employee ORDER BY Salary DESC;

SELECT Distinct(City) FROM Employee;

SELECT COUNT(\*), Department FROM Employee GROUP BY Department;

SELECT COUNT(\*), City FROM Employee GROUP BY City;

SELECT COUNT(\*), City FROM Employee GROUP BY City HAVING count(\*) > 2;

(Show result after every query)

|  |  |
| --- | --- |
| **Practical No: 5** | **Roll No:** |
| **Objective: Aggregate Functions - AVG, COUNT, MAX, MIN, SUM, CUBE** | **Date:** |

Create table Customers (id, name, age, purchasedAmount)

(Write query)

Insert into Statements

**COUNT(Column Name) –**

SELECT COUNT(\*)FROM Customers;

**MIN(Column Name)**

SELECT M**IN**(Age) FROM Customers;

**MAX(Column Name)**

SELECT MAX(Age) FROM Customers;

**AVG(Column Name)**

SELECT AVG(Age) FROM Customers;

**SUM(Column Name)**

SELECT SUM(purchasedAmount) FROM Customers;

**CUBE Example to be added**

|  |  |
| --- | --- |
| **Practical No: 6** | **Roll No:** |
| **Objective: Implementing all String functions.** | **Date:** |

**Notes:**

1. ASCII stands for American Standard Code for Information Interchange. It serves as a character encoding standard for modern computers. See the Printable characters section of ASCII for a list of ASCII characters.

ASCII is a 7-bit character set. Extended ASCII or High ASCII is an 8-bit character set that is not handled by the ASCII function.

1. Unicode is an international character encoding standard that provides a unique number for every character across languages and scripts, making almost all characters accessible across platforms, programs, and devices.

|  |  |  |
| --- | --- | --- |
|  | Format  Query | Result |
| 1 | ASCII ( character\_expression ) , returns integer SELECT ASCII('A'); | 65 |
| 2 | CHAR ( integer\_expression ) , returns string  SELECT CHAR(65)   example: SELECT FirstName + ' ' + LastName, + CHAR(13) + EmailAddress | A |
| 3 | CHARINDEX ( expressionToFind , expressionToSearch [ , start\_location ] )  returns integer  SELECT CHARINDEX('ban','I like banana’); |  |
| 4 | CONCAT ( string\_value1, string\_value2 [, string\_valueN ] )  SELECT CONCAT ( 'Happy ', 'Birthday Prashant! Today is', 25, '/', '1/2023' ) AS Result; | Happy Birthday Prashant! Today is 25/1/2023 |
| 5 | CONCAT\_WS ( separator, argument1, argument2 [, argumentN]... )  SELECT CONCAT ( '-', 'option1', 'option2', 'option 3' ) AS Result; |  |
| 6 | LEFT ( character\_expression , integer\_expression )  SELECT LEFT('abcdefg',2);  SELECT LEFT(Name, 5) FROM Product; | ab |
| 7 | RIGHT ( character\_expression , integer\_expression )  SELECT RIGHT('abcdefg',2);  SELECT RIGHT(Name, 5) FROM Product ; | fg |
| 8 | LEN ( string\_expression )  SELECT LEN(FirstName) AS Length, FirstName FROM Customers; |  |
| 9 | LOWER ( character\_expression )  SELECT LOWER(FirstName) FROM Customers; |  |
| 10 | LTRIM ( character\_expression , [ characters ] )  SELECT LTRIM(' Five spaces are at the beginning of this string.'); |  |
| 11 | RTRIM ( character\_expression , [ characters ] )  SELECT RTRIM('Five spaces are at the end of this string. '); |  |
| 12 | NCHAR ( integer\_expression )  Returns the Unicode character with the specified integer code, as defined by the Unicode standard. |  |
| 13 | PATINDEX ( '%pattern%' , expression ); RETURNS int  SELECT PATINDEX('%ter%', 'interesting data'); |  |
| 14 | REPLACE ( string\_expression , string\_pattern , string\_replacement )  SELECT REPLACE('leena is crazy','l','t'); |  |
| 15 | REVERSE ( string\_expression )  SELECT FirstName, REVERSE(FirstName) AS Reverse FROM Person |  |
| 16 | SPACE ( integer\_expression )  SELECT FirstName + SPACE(2) + LastName FROM Persons |  |
| 17 | STUFF ( character\_expression , start , length , replace\_with\_expression )  SELECT STUFF('abcdef', 2, 3, 'ijklmn'); | aijklmnef    (1 row(s) affected) |
| 18 | SUBSTRING ( expression ,start , length )  SELECT name, SUBSTRING(name, 1, 1) AS Initial |  |
| 19 | TRIM ( [ LEADING | TRAILING | BOTH ] [characters FROM ] string ) |  |
| 20 | UPPER ( character\_expression ) SELECT UPPER(LastName) FROM Customers;  SELECT UPPER('leena'); |  |
| 21 | SELECT REPLICATE('wow! ',3) | Wow! Wow! Wow! |
| 22 | UNICODE ( 'ncharacter\_expression' ) Returns the integer value, as defined by the Unicode standard, for the first character of the input expression.  SELECT UNICODE('P') |  |

|  |  |
| --- | --- |
| **Practical No: 7** | **Roll No:** |
| **Objective: Implementing all Date functions.** | **Date:** |

|  |  |
| --- | --- |
| **1** | Return the current date and time:  SELECT CURRENT\_TIMESTAMP; |
| **2** | DATEADD(interval, number, date)  example  Add two months to a date, then return the date:  SELECT DATEADD(month, 2, '2022/01/25') AS DateAdd;  Add one year to a date, then return the date:  SELECT DATEADD(year, 1, '2022/01/25') AS DateAdd; |
| **3** | The DATEDIFF() function returns the difference between two dates.  DATEDIFF(interval, date1, date2)  SELECT DATEDIFF(month, '2017/08/25', '2022/08/25') AS DateDiff;  SELECT DATEDIFF(hour, '2022/08/25 07:00', '2022/08/25 12:45') AS DateDiff; |
| **4** | Return a specified part of a date, returns String  DATENAME(interval, date)  SELECT DATENAME(yy, '2017/08/25') AS DatePartString; |
| **5** | The DATEPART() function returns a specified part of a date, returns integer  SELECT DATEPART(yy, '2017/08/25') AS DatePartInt; |
| **6** | Return the day of the month for a date:  DAY(date)  SELECT DAY('2017/08/13 09:08') AS DayOfMonth; |
| **7** | Return the month part of a date:  SELECT MONTH('2017/08/25') AS Month; |
| **8** | Return the current database system date and time:  SELECT GETDATE(); |
| **9** | Return the date and time of the SQL Server:  SELECT SYSDATETIME() AS SysDateTime; |
| **10** | Return the year part of a date:  SELECT YEAR('2017/08/25') AS Year; |
| **11** | Return the current UTC date and time:  SELECT GETUTCDATE(); |
| **12** | Check if the expression is a valid date:  SELECT ISDATE('2022-01-25'); |

|  |  |
| --- | --- |
| **Practical No: 8** | **Roll No:** |
| **Objective: implementing use of UNION, INTERSECTION, SET DIFFERENCE.** | **Date:** |

create table School! (

firstName varchar(50),

lastName varchar(50),

courseName varchar(50)

);

create table School2 (

firstName varchar(50),

lastName varchar(50),

courseName varchar(50)

);

INSERT INTO School1 VALUES ('leena','lele', 'MCA' );

INSERT INTO School1 VALUES ('Meena','Kale', 'MSc' );

INSERT INTO School1 VALUES ('Gaurav','Sonar', 'MCom' );

INSERT INTO School1 VALUES ('Teena','Chowdhari', 'MBA' );

INSERT INTO School2 VALUES ('Bina','Soman', 'MCA' );

INSERT INTO School2 VALUES ('Sameer','Kale', 'MSc' );

INSERT INTO School2 VALUES ('Gaurav','Sonar', 'MCom' );

**UNION** - Combines Both results excluding duplicate row

SELECT firstName, lastName FROM school1

UNION

SELECT firstName, lastName FROM school2

**UNION ALL** - Combines Both results INCLUDING duplicate row

SELECT firstName, lastName FROM school1

UNION ALL

SELECT firstName, lastName FROM school2

**Intersection :** Combine result sets of two input queries and return the distinct rows that appear in both inputs.

SELECT firstName, lastName FROM school1

INTERSECT

SELECT firstName, lastName FROM school2

**Set Difference Operations:**

SELECT courseName FROM school1 EXCEPT SELECT courseName FROM school2;

|  |  |
| --- | --- |
| **Practical No: 9** | **Roll No:** |
| **Objective: Implement Nested Queries & all types of JOIN operation** | **Date:** |

* **Create Tables**

CREATE TABLE Employee

(

Id INT,

Name VARCHAR(50),

Salary MONEY,

DepartmentId INT

);

INSERT INTO Employee VALUES (1001, 'Leena Lele', 20000, 101);

INSERT INTO Employee VALUES (1002, 'Pawan Pawar', 30000, 102);

INSERT INTO Employee VALUES (1003, 'Sunil Nikumbh', 40000, 103);

INSERT INTO Employee VALUES (1003, 'Shankar Marathe', 40000, 106);

CREATE TABLE Department

(

Id INT PRIMARY KEY,

Name VARCHAR(30) NOT NULL UNIQUE,

Location VARCHAR(50) DEFAULT 'MUMBAI',

CONSTRAINT locationChk CHECK (Location IN('MUMBAI','DELHI','BANGLORE'))

);

INSERT INTO Department VALUES (101,'TESTING','BANGLORE') ;

INSERT INTO Department VALUES (102,'CODING','MUMBAI') ;

INSERT INTO Department VALUES (103,'MAINTAINANCE','DELHI') ;

INSERT INTO Department VALUES (104,'DEPLOYMENT','MUMBAI');

**NESTED QUERIES –**

SELECT \* FROM Employee WHERE departmentId NOT IN ( SELECT Id FROM Department WHERE name='TESTING');

**INNER JOIN –**

SELECT e.id, e.departmentId, e.name, e.salary, d.name, d.location FROM Employee E INNER JOIN Department D

ON E.departmentId = D.Id;

**LEFT JOIN –**

SELECT e.id, e.departmentId, e.name, e.salary, d.name, d.location FROM Employee E LEFT JOIN Department D

ON E.departmentId = D.Id;

**RIGHT JOIN –**

SELECT e.id, e.departmentId, e.name, e.salary, d.name, d.location FROM Employee E

RIGHT JOIN Department D

ON E.departmentId = D.Id;

**FULL JOIN –**

SELECT e.id, e.departmentId, e.name, e.salary, d.name, d.location FROM Employee E FULL JOIN Department D

ON E.departmentId = D.Id;

**CROSS JOIN –**

**SELF JOIN –**

|  |  |
| --- | --- |
| **Practical No: 10** | **Roll No:** |
| **Objective: Performing different operations on a view.** | **Date:** |

**CREATE Table Teacher, Department**

Create View

CREATE VIEW teachers\_data (teacherid, teachername, teacherdept, deptid, deptname, deptcapacity)

AS

SELECT

t.id, t.name, t.department, d.id, d.name, d.capacity

FROM

teacher t

INNER JOIN department d

ON t.department = d.id;

**Alter View**

ALTER VIEW teachers\_data (teacherid, teachername, teacherdept, deptid, deptname)

AS

SELECT

t.id, t.name, t.department, d.id, d.name

FROM

teacher t

INNER JOIN department d

ON t.department = d.id;

**Insert into View**

INSERT INTO teachers\_data (teacherid, teachername, teacherdept)

values (4,'kale',103);

**Update View**

update teachers\_data

set teachername='gore' where teacherid=4;

**Delete from View**

delete from teachers\_data where teacherid=4;

**Drop View –**

DROP VIEW teachers\_data;