**Antony Roshan**

**MICROSOFT SQL SERVER**

**Create Database**

create database test;

**Use a Database**

use test;

**Drop a database**

drop database test;

**Create a table**

create table employee(

id int primary key,

Name varchar(20),

salary int,

age int,

gender varchar(20),

dept varchar(20)

);

**Insert command**

Insert into employee values(

1,'Roshan',150000,22,'Male','Manage');

Insert into employee values(

2,'Joseph',125000,24,'Male','Vice president');

Insert into employee values(

3,'Arya',65000,29,'Female','Support');

Insert into employee values(

4,'Rahul',110000,35,'Male','Analytics');

Insert into employee values(

5,'Amala',75000,27,'Female','Support');

Insert into employee values(

6,'Alen',95000,55,'Male','Operations')

**select statement**

select Name from employee;

|  |
| --- |
| Name |
| Roshan |
| Joseph |
| Arya |
| Rahul |
| Amala |
| Alen |

select Age from employee;

|  |
| --- |
| Age |
| 22 |
| 24 |
| 29 |
| 35 |
| 27 |
| 55 |

select Name,Gender,Salary from employee;

|  |  |  |
| --- | --- | --- |
| Name | Gender | Salary |
| Roshan | Male | 150000 |
| Joseph | Male | 125000 |
| Arya | Female | 65000 |
| Rahul | Male | 110000 |
| Amala | Female | 75000 |
| Alen | Male | 95000 |

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 3 | Arya | 65000 | 29 | Female | Support |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 5 | Amala | 75000 | 27 | Female | Support |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Select distinct value**

select distinct Gender from employee;

|  |
| --- |
| Gender |
| Female |
| Male |

**Where clause**

select \* from employee where gender = 'female';

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 3 | Arya | 65000 | 29 | Female | Support |
| 5 | Amala | 75000 | 27 | Female | Support |

select \* from employee where salary >100000;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |

**And operator**

select \* from employee where salary > 100000 AND age < 25;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |

select \* from employee where dept = 'Support' and salary>70000;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 5 | Amala | 75000 | 27 | Female | Support |

**OR operator**

select \* from employee where salary > 100000 OR age <25;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |

select \* from employee where dept='Support' or salary>70000;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 3 | Arya | 65000 | 29 | Female | Support |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 5 | Amala | 75000 | 27 | Female | Support |
| 6 | Alen | 95000 | 55 | Male | Operations |

**NOT operator**

select \* from employee where not gender = 'Female';

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |

select \* from employee where not age > 28;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 5 | Amala | 75000 | 27 | Female | Support |

**Like operator**

select \* from employee where name like 'R%';

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 4 | Rahul | 110000 | 35 | Male | Analytics |

select \* from employee where age like '5\_';

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Between operator**

select \* from employee where age between 25 and 35;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 3 | Arya | 65000 | 29 | Female | Support |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 5 | Amala | 75000 | 27 | Female | Support |

select \* from employee where salary between 90000 and 120000;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Aggregate commands**

**Min() function**

select min(age) from employee;

|  |
| --- |
| (No column name) |
| 22 |

select min(salary) from employee;

|  |
| --- |
| (No column name) |
| 65000 |

**Max() function**

select max(age) from employee;

|  |
| --- |
| (No column name) |
| 55 |

select max(salary) from employee;

|  |
| --- |
| (No column name) |
| 150000 |

**Count() function**

select count(\*) from employee where gender= 'Male';

|  |
| --- |
| (No column name) |
| 4 |

select count(\*) from employee where gender= 'Female';

|  |
| --- |
| (No column name) |
| 2 |

**Sum() function**

select sum(salary) from employee;

|  |
| --- |
| (No column name) |
| 620000 |

**Avg() function**

select avg(age) from employee;

|  |
| --- |
| (No column name) |
| 32 |

**String fucntions**

**ltrim()**

select ' Avenger'

|  |
| --- |
| (No column name) |
| Avenger |

select ltrim(' Avenger')

|  |
| --- |
| (No column name) |
| Avenger |

**lower()**

select lower('AVENGER')

|  |
| --- |
| (No column name) |
| avenger |

**upper()**

select upper('earth')

|  |
| --- |
| (No column name) |
| EARTH |

**reverse()**

select reverse('hello')

|  |
| --- |
| (No column name) |
| olleh |

**substring()**

select substring('Earth is in Milkyway Galaxy',13,8)

|  |
| --- |
| (No column name) |
| Milkyway |

**Order by syntax**

select \* from employee order by salary;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 3 | Arya | 65000 | 29 | Female | Support |
| 5 | Amala | 75000 | 27 | Female | Support |
| 6 | Alen | 95000 | 55 | Male | Operations |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 1 | Roshan | 150000 | 22 | Male | Manage |

select \* from employee order by salary DESC;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |
| 5 | Amala | 75000 | 27 | Female | Support |
| 3 | Arya | 65000 | 29 | Female | Support |

**Top clause syntax**

select TOP 3 \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 3 | Arya | 65000 | 29 | Female | Support |

select TOP 3 \* from employee order by age DESC;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 6 | Alen | 95000 | 55 | Male | Operations |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 3 | Arya | 65000 | 29 | Female | Support |

**Group by syntax**

select avg(salary) from employee group by gender;

|  |
| --- |
| (No column name) |
| 70000 |
| 120000 |

select avg(age),dept from employee group by dept order by avg(age) desc;

|  |  |
| --- | --- |
| (No column name) | dept |
| 55 | Operations |
| 35 | Analytics |
| 28 | Support |
| 24 | Vice president |
| 22 | Manage |

**Having clause syntax**

select dept,avg(salary) as avg\_salary

from employee

group by dept

having avg(salary) > 100000;

|  |  |
| --- | --- |
| dept | avg\_salary |
| Analytics | 110000 |
| Manage | 150000 |
| Vice president | 125000 |

**Update statement syntax**

update employee set age=42 where name = 'Alen';

update employee set dept='Tech' where gender='Female';

update employee set salary=50000;

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 50000 | 22 | Male | Manage |
| 2 | Joseph | 50000 | 24 | Male | Vice president |
| 3 | Arya | 50000 | 29 | Female | Tech |
| 4 | Rahul | 50000 | 35 | Male | Analytics |
| 5 | Amala | 50000 | 27 | Female | Tech |
| 6 | Alen | 50000 | 42 | Male | Operations |

**Delete statement syntax**

delete from employee where age=29;

delete from employee where Name='Rahul';

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 50000 | 22 | Male | Manage |
| 2 | Joseph | 50000 | 24 | Male | Vice president |
| 5 | Amala | 50000 | 27 | Female | Tech |
| 6 | Alen | 50000 | 42 | Male | Operations |

**Truncate statement syntax**

truncate table employee;

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |

**Create table department**

create table department(

did int primary key,

dname varchar(20),

location varchar(20));

insert into department values(1,'Manage','New York');

insert into department values(2,'Sales','Texas');

insert into department values(3,'Analytics','Dallas');

insert into department values(4,'Content','Mumbai');

insert into department values(5,'Support','Texas');

insert into department values(6,'Finance','Boston');

select \* from department;

|  |  |  |
| --- | --- | --- |
| did | dname | location |
| 1 | Manage | New York |
| 2 | Sales | Texas |
| 3 | Analytics | Dallas |
| 4 | Content | Mumbai |
| 5 | Support | Texas |
| 6 | Finance | Boston |

**Join commands**

**Inner Join**

select employee.Name,employee.dept,department.dname,department.location

from employee

INNER JOIN department ON employee.dept=department.dname;

|  |  |  |  |
| --- | --- | --- | --- |
| Name | dept | dname | location |
| Roshan | Manage | Manage | New York |
| Arya | Support | Support | Texas |
| Rahul | Analytics | Analytics | Dallas |
| Amala | Support | Support | Texas |

**Left Join**

select employee.Name,employee.dept,department.dname,department.location

from employee

LEFT JOIN department

ON employee.dept=department.dname;

|  |  |  |  |
| --- | --- | --- | --- |
| Name | dept | dname | location |
| Roshan | Manage | Manage | New York |
| Joseph | Vice president | NULL | NULL |
| Arya | Support | Support | Texas |
| Rahul | Analytics | Analytics | Dallas |
| Amala | Support | Support | Texas |
| Alen | Operations | NULL | NULL |

**Right Join**

select employee.Name,employee.dept,department.dname,department.location

from employee

RIGHT JOIN department

ON employee.dept=department.dname;

|  |  |  |  |
| --- | --- | --- | --- |
| Name | dept | dname | location |
| Roshan | Manage | Manage | New York |
| NULL | NULL | Sales | Texas |
| Rahul | Analytics | Analytics | Dallas |
| NULL | NULL | Content | Mumbai |
| Arya | Support | Support | Texas |
| Amala | Support | Support | Texas |
| NULL | NULL | Finance | Boston |

**Full Join**

select employee.Name,employee.dept,department.dname,department.location

from employee

FULL JOIN department

ON employee.dept=department.dname;

|  |  |  |  |
| --- | --- | --- | --- |
| Name | dept | dname | location |
| Roshan | Manage | Manage | New York |
| Joseph | Vice president | NULL | NULL |
| Arya | Support | Support | Texas |
| Rahul | Analytics | Analytics | Dallas |
| Amala | Support | Support | Texas |
| Alen | Operations | NULL | NULL |
| NULL | NULL | Sales | Texas |
| NULL | NULL | Content | Mumbai |
| NULL | NULL | Finance | Boston |

**Update using Join**

update employee

set age=age+3

from employee

JOIN department

on employee.dept=department.dname

where location='Texas';

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 3 | Arya | 65000 | 32 | Female | Support |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 5 | Amala | 75000 | 30 | Female | Support |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Delete using Join**

delete employee

from employee

join department on employee.dept=department.dname

where location='Texas';

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Create student1 table**

create table student1(id int primary key,name varchar(20),marks int);

insert into student1 values(1,'Tom',45);

insert into student1 values(2,'John',87);

insert into student1 values(3,'Siya',73);

insert into student1 values(4,'Alen',92);

insert into student1 values(5,'Jeff',65);

select \* from student1;

|  |  |  |
| --- | --- | --- |
| id | name | marks |
| 1 | Tom | 45 |
| 2 | John | 87 |
| 3 | Siya | 73 |
| 4 | Alen | 92 |
| 5 | Jeff | 65 |

**create student2 table**

create table student2(id int primary key,name varchar(20),marks int);

insert into student2 values(4,'Alen',92)

insert into student2 values(5,'Jeff',65);

insert into student2 values(6,'Mike',59);

select \* from student2;

|  |  |  |
| --- | --- | --- |
| id | name | marks |
| 4 | Alen | 92 |
| 5 | Jeff | 65 |
| 6 | Mike | 59 |

**Union operator**

select \* from student1

union

select \* from student2

|  |  |  |
| --- | --- | --- |
| id | name | marks |
| 1 | Tom | 45 |
| 2 | John | 87 |
| 3 | Siya | 73 |
| 4 | Alen | 92 |
| 5 | Jeff | 65 |
| 6 | Mike | 59 |

**Union all operator**

select \* from student1

union all

select \* from student2

|  |  |  |
| --- | --- | --- |
| id | name | marks |
| 1 | Tom | 45 |
| 2 | John | 87 |
| 3 | Siya | 73 |
| 4 | Alen | 92 |
| 5 | Jeff | 65 |
| 4 | Alen | 92 |
| 5 | Jeff | 65 |
| 6 | Mike | 59 |

**Except operator**

select \* from student1

except

select \* from student2

|  |  |  |
| --- | --- | --- |
| id | name | marks |
| 1 | Tom | 45 |
| 2 | John | 87 |
| 3 | Siya | 73 |

**Intersect operator**

select \* from student1

intersect

select \* from student2

|  |  |  |
| --- | --- | --- |
| id | name | marks |
| 4 | Alen | 92 |
| 5 | Jeff | 65 |

**Views**

create view male\_employee AS

select \* from employee

where gender='Male';

select \* from male\_employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Drop a view**

drop view male\_employee;

**Alter Table**

alter table employee add dob date;

select \* from employee;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept | dob |
| 1 | Roshan | 150000 | 22 | Male | Manage | NULL |
| 2 | Joseph | 125000 | 24 | Male | Vice president | NULL |
| 4 | Rahul | 110000 | 35 | Male | Analytics | NULL |
| 6 | Alen | 95000 | 55 | Male | Operations | NULL |

**Drop column**

alter table employee drop column dob;

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |

**create table source**

CREATE TABLE Source(

pid INT,

pname VARCHAR(50),

price DECIMAL(9,2))

INSERT INTO Source VALUES(1,'Table',100)

INSERT INTO Source VALUES(2,'Desk',80)

INSERT INTO Source VALUES(3,'Chair',50)

INSERT INTO Source VALUES(4,'Computer',300)

SELECT \* FROM Source

|  |  |  |
| --- | --- | --- |
| pid | pname | price |
| 1 | Table | 100.00 |
| 2 | Desk | 80.00 |
| 3 | Chair | 50.00 |
| 4 | Computer | 300.00 |

**create table Target**

CREATE TABLE Target(

pid INT,

pname VARCHAR(50),

price DECIMAL(9,2)

)

INSERT INTO Target VALUES(1,'Table',100)

INSERT INTO Target VALUES(2,'Desk',180)

INSERT INTO Target VALUES(5,'Bed',50)

INSERT INTO Target VALUES(6,'Cupboard',300)

SELECT \* FROM Target

|  |  |  |
| --- | --- | --- |
| pid | pname | price |
| 1 | Table | 100.00 |
| 2 | Desk | 180.00 |
| 5 | Bed | 50.00 |
| 6 | Cupboard | 300.00 |

**Merge**

MERGE Target AS T

USING SOURCE as S

ON T.pid=s.pid

WHEN MATCHED

THEN update set T.price=S.price

WHEN NOT MATCHED BY TARGET

THEN insert (pid,pname,price)

VALUES(S.pid,S.pname,S.price)

WHEN NOT MATCHED BY SOURCE

THEN delete;

select \* from Target

|  |  |  |
| --- | --- | --- |
| pid | pname | price |
| 1 | Table | 100.00 |
| 2 | Desk | 80.00 |
| 3 | Chair | 50.00 |
| 4 | Computer | 300.00 |

select \* from Source

|  |  |  |
| --- | --- | --- |
| pid | pname | price |
| 1 | Table | 100.00 |
| 2 | Desk | 80.00 |
| 3 | Chair | 50.00 |
| 4 | Computer | 300.00 |

**User defined fuctions**

**Scalar valued functions**

create function add\_five(@num as int)

RETURNS int

AS

BEGIN

RETURN(

@num+5)

end

select dbo.add\_five(5);

|  |
| --- |
| (No column name) |
| 10 |

**Table valued function**

create function select\_gender(@gender as varchar(20))

RETURNS table

AS RETURN(

select \* from employee where gender=@gender)

select \* from dbo.select\_gender('Male');

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Create temporary table**

create table #student(

sid int,

sname varchar(20));

insert into #student values(1,'sam');

select \* from #student;

|  |  |
| --- | --- |
| sid | sname |
| 1 | sam |

**Case statement**

select

case

when 10>20 then '10 is grester than 20'

when 10<20 then '10 is less than 20'

else '10 is equal to 10'

end

|  |
| --- |
| (No column name) |
| 10 is less than 20 |

select \*,grade=

case

when salary<90000 then 'C'

when salary<120000 then 'B'

else 'A'

end

from employee

go

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept | grade |
| 1 | Roshan | 150000 | 22 | Male | Manage | A |
| 2 | Joseph | 125000 | 24 | Male | Vice president | A |
| 4 | Rahul | 110000 | 35 | Male | Analytics | B |
| 6 | Alen | 95000 | 55 | Male | Operations | B |

**IIF function**

select

iif(10>20,'10 is grater than 20','10 is less than 20');

(No column name)

10 is less than 20

select id,Name,age, iif(age>30,'Old employee','Young employee') as employee\_generation from employee;

|  |  |  |  |
| --- | --- | --- | --- |
| id | Name | age | employee\_generation |
| 1 | Roshan | 22 | Young employee |
| 2 | Joseph | 24 | Young employee |
| 4 | Rahul | 35 | Old employee |
| 6 | Alen | 55 | Old employee |

**Stored procedure in sql**

create procedure employee\_age

as

select age from employee

go

exec employee\_age;

|  |
| --- |
| age |
| 22 |
| 24 |
| 35 |
| 55 |

create procedure employee\_details

as

select \* from employee

go

exec employee\_details;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Stored procedure with parameter syntax**

create procedure employee\_gender @gender varchar(20)

as

select \* from employee

where gender=@gender

go

exec employee\_gender @gender='Male';

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Exception handling**

declare @val1 int;

declare @val2 int;

begin try

set @val1=8;

set @val2=@val1/0;

end try

begin catch

print error\_message()

end catch

Divide by zero error encountered.

**Example 2**

begin try

select salary+name from employee

end try

begin catch

print 'Cannot add a numerical with a string value'

end catch

go

(0 rows affected)

Cannot add a numerical with a string value

**Transcation in sql**

begin transaction

update employee set age=30 where name='Arya'

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 3 | Arya | 65000 | 30 | Female | Support |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 5 | Amala | 75000 | 27 | Female | Support |
| 6 | Alen | 95000 | 55 | Male | Operations |

rollback transaction

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 3 | Arya | 65000 | 29 | Female | Support |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 5 | Amala | 75000 | 27 | Female | Support |
| 6 | Alen | 95000 | 55 | Male | Operations |

commit transaction

**Example 2**

begin try

begin transaction

update employee set salary=77777 where gender='Male'

update employee set salary=55555/0 where gender='Female'

commit transaction

end try

begin catch

rollback transaction

print 'Transaction rollback'

end catch

(4 rows affected)

(0 rows affected)

Transaction rollback

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 150000 | 22 | Male | Manage |
| 2 | Joseph | 125000 | 24 | Male | Vice president |
| 3 | Arya | 65000 | 29 | Female | Support |
| 4 | Rahul | 110000 | 35 | Male | Analytics |
| 5 | Amala | 75000 | 27 | Female | Support |
| 6 | Alen | 95000 | 55 | Male | Operations |

**Example 3**

begin try

begin transaction

update employee set salary=77777 where gender='Male'

update employee set salary=55555 where gender='Female'

commit transaction

end try

begin catch

rollback transaction

print 'Transaction rollback'

end catch

select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Name | salary | age | gender | dept |
| 1 | Roshan | 77777 | 22 | Male | Manage |
| 2 | Joseph | 77777 | 24 | Male | Vice president |
| 3 | Arya | 55555 | 29 | Female | Support |
| 4 | Rahul | 77777 | 35 | Male | Analytics |
| 5 | Amala | 55555 | 27 | Female | Support |
| 6 | Alen | 77777 | 55 | Male | Operations |

**DATA BASE ADMINISTRATOR**

**Roles of Data Base Administrator**

* Software installation and maintenance
* Taking care of Database backup and recovery
* Maintaining security of the database
* Take care of access control for different users
* Monitor databases for performance issues

**Types of DBA**

* **Production DBA**
* Maintain databases in an organization
* Take over after applications have been created
* **Application DBA**
* Focus on specific business application
* Expert in SQL
* Implement application using SQL
* **Development DBA**
  + Focus on tasks related to building an effective useable database environment
  + Skilled in data modelling and normalization
  + Promote data integrity
* **UAT DBA**
  + Under Application Testing Database
  + Completely responsible for developing UAT database
* **Data warehouse DBA**
  + Responsible for analysing and understanding the data provided
  + To study and track patterns

**MS SQL Server**

* It is an ORDBMS
* It is both GUI and command based software

**Editions of SQL Server**

* Enterprise Edition
* Standard Edition
* Web Edition
* Developer Edition
* Express Edition

**Instances in SQL Server**

* An instance is a collection of SQL server databases run by a single SQL server service
* A computer can host only one default instance. All other instances must be named

**SQL Server System Databases**

* **Master** 
  + Records all system level information for SQL Server system
  + SQL servers store information of all databases
* **Model**
  + Used as template for new user database created
  + Can change most database properties, create users, stored procedures, tables, views etc…
* **MSDB**
  + Used by SQL server agent for scheduling alerts and job
  + Holds backup history
* **TempDB**
* Tempdb is the workhorse of the system databases
* Workspace to store the intermediate results of query processing and sorting

**Backup & Restore of databases**

* Restore & backup to make sure that there is no data loss
* The system database to always backup includes msdb, master and model

**Recovery Models**

* **Simple**
* Simple recovery model maintain only a minimum amount of information in the SQL server transaction log file
* Truncates transaction log files when it has reached checkpoints so that the space can be reused
* The data is recoverable to the most recent full database or deferential backup
* **Full**
* Full recovery model logs every transaction and maintains it there until a transaction log backup is taken
* You can devise a disaster recovery plan that includes a combination of full and transaction log backups
* You can recover to an arbitrary point in point from transaction log backup and hence no work is lost
* **Bulk Logged**
* In this model there is certain bulk operations such as BULK INSERT, CREATE INDEX etc…
* To run batch jobs
* It is used when you perform routine maintenance
* Reduce log space usage

**Types of Backup Models**

* **Full Backups**
* A full database backup backs up the whole database
* The primary advantage to performing a full backup during every operation is that a complete copy of all data is available with a single set of media
* **Differential Backups**
* A differential backup is a type of backup that copies all the data that has changed since the last full backup
* The full backup upon which a differential backup is based is known as the base of the differential

* **File Backups & Filegroup Backups**
* When the database size and performance requirements make a full database backup impractical, you can create a file backup instead
* A file backup contains all the data in or more files(or filegroups)
* **Partial Backups**
* A partial backup resembles a full database backup, except that it does not contain all the filegroups.
* A partial backup will have inside it all of the data in the primary filegroup, every read/write filegroup and possibly read-only files which can be specified optionally
* **Copy-Only Backups**
* Copy Only backup is an independent backup and it does not break the chain of database backups
* In other words, it does not disturb the original backup workflow, but just creates a copy of the original database independently
* **Mirror Backups**
* Mirrored backups are actually identical copies of the same backup
* Administrators usually store them on multiple backup devices, thus increasing the reliability of the backups, and greatly reducing the probability of data loss
* **Transaction Log Backups**
* A transaction log backup allows you to backup the active part of the transaction log
* So, after you issue a "Full" or "Differential" backup, the transaction log backup will have any transactions that were created after those other backups completed

**Point in Time Recovery**

* Point-in-time recovery allows to restore a database into a state it was in any point of time
* This type of recovery is applicable only to databases that run under the full or bulk-logged recovery model

**Dynamic Management Views**

Dynamic Management Views return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance

**Properties of Dynamic Management Views**

* Dynamic Management Views are actually composed of both views & table-valued functions. Some apply to the entire server and are stored in the master database. Others are specific to each database
* All dynamic management views and functions exist in the sys schema and follow this naming convention dm \*

**Categories of DMV’s**

**Database Related DMV’s**

**1) sys.dm\_db\_file\_space\_usage**

USE tempdb;

GO

SELECT SUM(unallocated\_extent\_page\_count) AS [free pages], (SUM(unallocated\_extent\_page\_count)\*1.0/128) AS [free space in MB]

FROM sys.dm\_db\_file\_space\_usage;

|  |  |
| --- | --- |
| free pages | free space in MB |
| 576 | 4.500000 |

**2) sys.dm\_db\_log\_space\_usage**

USE tempdb;

GO

SELECT (total\_log\_size\_in\_bytes -

used\_log\_space\_in\_bytes)\*1.0/1024/1024 AS [free log space in MB]

FROM sys.dm\_db\_log\_space\_usage;

|  |
| --- |
| free log space in MB |
| 7.38671875000 |

**3) sys.dm\_db\_partition\_stats**

USE test;

GO

SELECT \* FROM sys.dm\_db\_partition\_stats;

GO

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| partition\_id | object\_id | index\_id | partition\_number | in\_row\_data\_page\_count | in\_row\_used\_page\_count | in\_row\_reserved\_page\_count | lob\_used\_page\_count | lob\_reserved\_page\_count | row\_overflow\_used\_page\_count | row\_overflow\_reserved\_page\_count | used\_page\_count | reserved\_page\_count | row\_count |
| 196608 | 3 | 1 | 1 | 18 | 20 | 25 | 0 | 0 | 0 | 0 | 20 | 25 | 1508 |
| 327680 | 5 | 1 | 1 | 3 | 5 | 5 | 0 | 0 | 0 | 0 | 5 | 5 | 192 |
| 281474977103872 | 6 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SQL Server OS Related DMV’s**

**1) sys.dm\_os\_buffer\_descriptors**

SELECT COUNT(\*)AS cached\_pages\_count

,CASE database\_id

WHEN 32767 THEN 'ResourceDb'

ELSE db\_name(database\_id)

END AS database\_name

FROM sys.dm\_os\_buffer\_descriptors

GROUP BY DB\_NAME(database\_id),database\_id

ORDER BY cached\_pages\_count DESC;

|  |  |
| --- | --- |
| cached\_pages\_count | database\_name |
| 2414 | ResourceDb |
| 207 | test |
| 186 | master |
| 168 | tempdb |
| 61 | msdb |
| 6 | model |

**2) sys.dm\_os\_threads**

Returns a list of all SQL Server Operating System threads that are running under the SQL Server process

SELECT \*

FROM sys.dm\_os\_threads

WHERE started\_by\_sqlservr = 0;

**Execution Related DMV’s**

**1) sys.dm\_exec\_connections**

Returns information about the connections established to this instance of SQL Server and the details of each connection

SELECT

c.session\_id, c.net\_transport, c.encrypt\_option, c.auth scheme, s.host\_name, s.program\_name, s.client\_interface\_name, s.login\_name, s.nt\_domain, s.nt\_user\_name, s.original\_login\_name, c.connect\_time,

s.login\_time

FROM sys.dm\_exec\_connections AS c

JOIN sys.dm\_exec\_sessions AS s

ON c.session\_id = s.session\_id

WHERE c.session\_id = @@SPID;

**Activity Monitor**

Activity Monitor displays information about SQL Server processes and how these processes affect the current instance of SQL Server

Activity Monitor is a tabbed document window with the following expandable and collapsible panes:

**Overview, Processes, Resource Waits, Data File I/O, Recent Expensive Queries & Active Expensive Queries**

**System/Performance Monitor**

Sometimes there are issues outside of SQL Server that may be causing performance issues and this is where we can use Performance Monitor. With the help of Performance Monitor we can view SQL Server Objects, performance counters, and the behaviour of other objects such as processors, memory, cache, & threads

Using Performance Monitor you can:

* View and change charts to reflect current activity, & show counter values that are updated at a frequency that the user defines
* Add system alerts that list an event in the alert log and can notify you by issuing a network alert
* Run a pre-defined application every time a counter value goes over or under a user-defined value

**Tracing SQL Server Activity**

When working with long and complex SQL queries, it becomes extremely important to monitor them and do a thorough and accurate analysis of all the tasks, so that it becomes easy to troubleshoot the data. This is where we can trace all the ongoing tasks and understand about the exceptions properly.

**SQL Server Profiler**

SQL Server Profiler is a tool for tracing, recreating, and troubleshooting problems in MS SQL Server. The profiler lets developers and database administrators create and handle traces and replay and analyse trace results.

**Ensuring Data Integrity**

Data Integrity is used to maintain accuracy and consistency of data in a table

* Correct
* Error-free
* Usefull

**Categories of Data Integrity**

* Entity Integrity
* Primary Key
* Unique key
* Not null
* Referential Integrity
* Foreign key
* Domain Integrity
* Check
* Default
* User-Defined Data Integrity
* Triggers

**Index**

An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view.

**Some advantages of indexes:**

* Reduce I/O operations on the disk
* Consume fewer resources through improved query performance
* Helpful in SELECT, UPDATE, DELETE, or MERGE statements

**Types of Index**

* **Clustered Index**

Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition.

* **Non-Clustered Index**

Non-clustered indexes have a structure separate from the non-clustered index key values and each key value entry has a pointer to the data row that contains the key value

**Fragmentation**

As data is modified in a database, the database and its indexes become fragmented. As indexes become fragmented, ordered data retrieval becomes less efficient and reduces database performance

**Remedy for Fragmentation**

* **Rebuilding Index**

Rebuilding an index drops and re-creates the index. This removes fragmentation, reclaims disk space by compacting the pages, and reorders the index rows in contiguous pages.

* **Re-organising Index**

Reorganizing an index uses minimal system resources. It defragments the leaf level of clustered and non-clustered indexes on tables and views by physically reordering the leaf-level pages to match the logical, left to right, order of the leaf nodes