ADITYA UNIVERSITY

DATABASE MANAGEMENT SYSTEMS

Unit II Relational Model

ganesh

]]Assistant Professor

Department of CSE

Aditya University

ADITYA UNIVERSITY

Relational Model Definition

Relational model is the most widely used data model. A database is a collection of one or more relations, where each relation is a table with rows and columns.

Simple tabular representation enable naive users to understand the contents of a database and ease with which complex queries can be expressed.

Definitions

- Relation schema specifies the relation name, the name of each field(column or attribute), domain of each field.

  R(F1:D1,F2:D2,…..Fn:Dn)

Ex:-Students(sid:integer,name:string,login:string,age:integer)

Field name has a domain string.

- Relation instance is a table. Tuples at a specific moment of time

SID	NAME	LOGIN	AGE
501	Ajay	ajay@cs	20
502	Ramya	ramya@ece	30
503	Rani	Rani@ece	50

Degree and Cardinality

- Degree of a relation

It represents the number of fields

Ex:- In students relation

Degree=4(fields)

- Cardinality of a relation

It represents the number of tuples

Ex:- In students relation

Cardinality=3(rows)

Importance of null values

value is missing/unknown/inapplicable

student(sid:integer,name:string,marks:integer)

```
create table student
(
sid integer,
name varchar2(20) not null,
marks integer,
constraint pk_student primary key(sid)
);

insert into student values(502,null,null);
cannot insert null into CSE23501.student.name
insert into student values(502,'suma',null);
select * from student;
```

sid	name	marks
501	ravi	20
502	suma	

:Find the name of students whose marks are empty/null.

select name from student where marks is null

name

suma

Find the name of students whose marks are not empty/null.

select name from student where marks is not null

name

ravi

update student
set marks=50
where sid=502;
select * from student;

sid	**name**	**marks**
501	ravi	20
502	suma	50

ADITYA UNIVERSITY

Domain Constraint

A domain is defined as the set of all values for an attribute.
Ex:- a domain of date is the set of all possible valid dates.
Database to prevent invalid dates being entered
30 february 2020
Data is rejected
Every attribute is bound to have a specific range of values
Ex:- age cannot be less than zero
Telephone numbers cannot contain a digit outside 0-9.
Domain constraints are tested easily by the system whenever a new data is entered into the database.

ADITYA UNIVERSITY

Integrity Constraint

Integrity constraint is a condition specified on a database schema and restricts the data that can be stored in an instance of the database.

If a database instance satisfies all the integrity constraints specified on the database schema, it is a legal instance.

A Key Constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.

ADITYA UNIVERSITY

Key Constraints

Super Key- set of one or more attributes whose values uniquely identify an row in the relation.

Candidate Key- a candidate key of a relation is a minimal super key for that relation.

One of the candidate key is selected as primary key. Primary key is a combination of not null and unique.

Primary key should be chosen such that its attributes are never or very rarely changed.

A foreign key is defined as a column in one table that references the primary key of another table. A foreign key links two tables together. The table that holds the foreign key is called a child table and that holds the primary key or unique key is called the parent table.

Enforcing Integrity Constraints

ICs are specified when a relation is created and enforced when a relation is modified.

- Insertion violates the primary key constraint cannot contain duplicate values

Insert into students(sid,name,age) values(501,'john',30);

- Insertion violates the primary key constraint cannot contain null values

Insert into students(sid,name,age) values(null,'kamala',20);

- Insertion violates domain constraint

Insert into students(sid,name,age) values(504,'udaya','ab');

- Updation violates primary key constraint

Update students
Set sid=501
Where sid=502;

ADITYA UNIVERSITY

Querying Relational data

A query is a request for data or information from a database table or combination of tables.

Ex:- Given data

Students

SID	NAME	LOGIN	AGE
501	Dave	dave@cs	19
502	John	john@cs	18
503	Smith	smith@ce	20
504	Smith	smith@math	19

Enrolled

SID	CID	Grade
501	DBMS05	A
502	OS04	B
503	CN03	C
504	UP02	A

Querying Relational data

Find all the details of students who are younger than 19

Select * from students s where s.age<19;

Find the name and login of students who are younger than 19.

Select s.name,s.login from students s where s.age<19.

Find the names and courseid of all students who obtained grade A

Select s.name, c.cid from students s,enrolled c where s.sid=c.sid and c.grade='A';

ADITYA UNIVERSITY

SQL

IBM developed the original version of sequel in 1970's. The sequel language name has changed to SQL(Structured Query Language).

SQL has several parts

1) DDL
2) DML
3) Integrity
4) View Definition
5) Transaction Control
6) Embedded SQL
7) Authorization

ADITYA UNIVERSITY

SQL Data Types

1) char(n):- A fixed-length character string with user specified length n.

2) varchar2(n):- A variable-length character string with user specified length n.

3) numeric(p,d):- number consists of p digits and d digits are to the right of the decimal point.

4) integer

5) date

DDL and DML commands

DDL Commands

1) Create
2) Alter
3) Rename
4) Truncate
5) Drop

DML Commands

1) Insert
2) Select
3) Delete
4) Update

Create Command

1) Create

It is used to create a table using schema.

Syntax:-

Create table tablename(A1 D1, A2 D2,…..AnDn,

Integrity constraint1,

.

.

Integrity constraintn)

A-Attribute in the schema of the relation

D-Domain type of values

Create Command

Ex:- students(sid:integer,name:string,age:integer)

```
Create table students
(
sid integer,                    //sid integer primary key
name varchar2(20),
age integer,
primary key(sid));

Or
Constraint p_stu primary key(sid)
```

Insert Command

To insert data into a table

Insert into tablename values(value1,value2,…..valuen);

Insert into tablename(col2,col1,col3) values(value2,value1,value3);

Ex:-

Insert into students values(501,'leela',22);

Insert into students(name,sid,age) values('suma',502,30);

Insert into students values(&sid,'&name',&age);

Enter sid: 503

Enter name: hema

Enter age: 23

Select Command

To retrieve rows selected from one or more tables.

Basic structure of SQL query consists of 3 clauses

1) Select:- It specifies the table columns that are retrieved.
2) From:- It lists the relations to be scanned.
3) Where:- It consists of predicate involving attributes of the relation.

To remove duplicates distinct can be used after select clause.

Ex:-Select distinct name from students;

Select Command

Syntax:-

Select [distinct] column1,column2 as newname from table1,table2

Where condition

Group by columnname

Having condition

Order by columname asc|desc

Ex:-To retrieve all rows in students

Select * from students;

To display student id and name

Select sid,name as stuname from students;

Alter Command

Alter command is used for altering the table structure, such as,

- to add a column to existing table
- to rename any existing column
- to change datatype of any column or to modify its size.
- to drop a column from the table.

ALTER TABLE table_name ADD(column_name1 datatype1
column_name2 datatype2);

ALTER TABLE table_name modify(column_name datatype);

ALTER TABLE table_name RENAME old_column_name TO new_column_name;

ALTER TABLE table_name drop column column_name;

Alter Command

```
create table st1
(
rollno integer,
name varchar2(20)
);
alter table st1
add constraint st_pk1 primary key(rollno);
alter table st1
drop constraint st_pk1;
```

ADITYA UNIVERSITY

Alter Command

```
create table st2
(
rollno integer,
name varchar2(20)
);

alter table st2
add constraint st_un unique(rollno);
alter table st2
drop constraint st_un;
```

Alter Command

```
alter table st2
modify rollno integer not null;
alter table st2
modify name varchar2(20) default 'ravi';
alter table st2
add marks integer;
alter table st2
add constraint c_ch check(marks<101);
alter table st2
drop constraint c_ch;
```

Alter Command

```
create table st2
(
rollno integer primary key,
name varchar2(20)
);
create table st3
(
rollno integer,
fee integer
);
alter table st3
add constraint fk_st1 foreign key(rollno) references st2(rollno);
alter table st3
drop constraint fk_st1;
```

Rename and Drop Commands

Rename is used to change the name of a table.

Syntax:-

Rename oldtablename to newtablename

Alter table tablename

Rename to newtablename

Drop is used to completely removes a table from the database. This will destroy the table structure and the data stored in it.

Syntax:-

Drop table tablename

Truncate, Delete and Update

TRUNCATE command removes all the records from a table. But this command will not destroy the table's structure.

Syntax:-

Truncate table tablename

The DELETE statement is used to delete existing records in a table.

Syntax:-

DELETE FROM table_name WHERE condition;

The Update statement is used to modify the existing records in a table.

Syntax:-

Update tablename set columname=value where condition;

ADITYA UNIVERSITY

Constraints

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table.

1) Not null
2) Unique
3) Check
4) Primary key
5) Foreign key
6) default

Not Null constraint

It indicates that a column cannot store null values. When the value of an attribute is not known, doesn't exist, not applicable.

Ex:- create table student

```
(
rollno integer not null,
name varchar2(20),
marks integer
);
```

Column rollno in student table not to accept null values. If null values are given it shows cannot insert null into student.rollno

Select name from student where rollno is null;

Select name from student where rollno is not null;

Unique constraint

Ensures that each row for a column must have a unique value. It will eliminate duplication.

```
Create table student
(
Rollno integer,
Name varchar2(20),
Marks integer,
Unique(rollno));
```

Column rollno in student table must have unique values.

Check constraint

It ensures that the value in a column meets a specific condition. If you define a check constraint on a single column it allows only certain values for that column.

```
Create table student
(
Rno integer,
Name varchar2(20),
Marks integer,
Check(marks<101));
```

Marks column include integers less than 101.

ADITYA UNIVERSITY

Primary Key

A combination of not null and unique. It must contain unique values and cannot contain null values.

```
Create table student
(
Rno integer,
Name varchar2(20),
Marks integer,
Primary key(rno));
```

Foreign Key

It represent relationships between tables. A foreign key is a column whose values are derived from the primary key of other table.

```
Create table studentad
(
Rno integer,
Fee integer,
Foreign key(Rno) references student(Rno));
```

Foreign key “rno” in studentad table refers to primary key “rno” column in student table.

Primary and Foreign Key

Studentad (referrencing relation)

RNO	Fee
501	5000
502	6000
503	7000

Foreign key

Student (referrenced relation)

RNO	NAME	MARKS
501	Padma	50
502	Ravi	80
503	Rahul	70

Primary key

ADITYA UNIVERSITY

Default

At the time of column creation a default value can be assigned to it.

```
Create table students
(
Rno integer,
Name varchar2(20),
Gender char(1) default ‘F’,
Marks integer);
```

Default value for gender column is ‘F’

On delete cascade

A foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted.

Syntax

```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,
  ...
    CONSTRAINT fk_column
     FOREIGN KEY (column1, column2, ... column_n)
     REFERENCES parent_table (column1, column2, ... column_n)
     ON DELETE CASCADE
);
```

On Delete Cascade

```
create table t1
 (
 t1_id integer,
 t1_desc varchar2(30),
 primary key(t1_id)
);
 create table t2
 (
 t1_t2_id integer,
 t2_id integer,
 t2_desc varchar2(30),
 primary key(t2_id),
 foreign key(t1_t2_id) references t1(t1_id) on delete cascade
 );
```

On Delete Cascade

```
create table t3
(
t2_t3_id integer,
t3_id integer,
t3_desc varchar2(30),
primary key(t3_id),
foreign key(t2_t3_id) references t2(t2_id) on delete cascade
);

insert into t1 values(1,'t1one');
insert into t2 values(1,1,'t2one');
insert into t3 values(1,1,'t3one');
```

On Delete Cascade

```
delete from t1 cascade;
select * from t1;
select * from t2;
select * from t3;
insert into t1 values(1,'t1one');
insert into t2 values(1,1,'t2one');
insert into t3 values(1,1,'t3one');
delete from t2 cascade;
select * from t1;
select * from t2;
select * from t3;
```

On Delete set null

It sets all the records of the column which is defined as a foreign key in the child table to Null if the corresponding record in the parent table is deleted.

```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,
  ...
    CONSTRAINT fk_column
    FOREIGN KEY (column1, column2, ... column_n)
    REFERENCES parent_table (column1, column2, ... column_n)
    ON DELETE SET NULL
);
```

On Delete set null

```
create table t1
(
t1_id integer,
t1_desc varchar2(30),
primary key(t1_id)
);
create table t2
(
t2_id integer,
t1_t2_id integer,
t2_desc varchar2(30),
primary key(t2_id),
foreign key(t1_t2_id) references t1(t1_id) on delete set NULL
);
```

On Delete set null

```
create table t3
 (
 t2_t3_id integer,
 t3_id integer,
 t3_desc varchar2(30),
 primary key(t3_id),
 foreign key(t2_t3_id) references t2(t2_id) on delete set null );

insert into t1 values(1,'t1one');
insert into t2 values(1,1,'t2one');
insert into t3 values(1,1,'t3one');
```

On Delete set null

```
delete from t1 where t1_id = 1;
select * from t1;
select * from t2;
select * from t3;
delete from t2 where t2_id=1;
select * from t2;
select * from t3;
```

ADITYA UNIVERSITY

Basic Queries

find the name,age of all sailors

select sname,age from sailors;

select sailors.sname,sailors.age from sailors;

select s.sname,s.age from sailors s;

find all sailors whose rating is greater than 7

select * from sailors s

where s.rating>7;

find all sailors whose rating is greater than 7 and age is less than 20

select * from sailors s

where s.rating>7 and s.age<20;

find the age of sailors whose name begins and ends with b and has atleast three characters

select s.age from sailors s

where s.sname like 'B_%b';

like operator

like operator

The **SQL LIKE** operator is used to retrieve the data in a column of a table, based on a specified pattern.

Syntax:

SELECT column1, column2, ... FROM table_name WHERE columnn LIKE specified_pattern;

the **%** is a wild card which represents zero, one or multiple characters.

The _underscore represents a single number or character.

ROLLNO	NAME	CITY
501	ravi	bangalore
502	suma	mumbai
503	uma	chennai
504	leela	raipur
505	mani	jaipur

ADITYA UNIVERSITY

Find all the details of students where city that have "ai" in any position

SQL> select * from studs where city like '%ai%';

select * from studs where city not like '%ai%';

Find all the details of students where city that ends with r

SQL> select * from studs where city like '%r';

Find all the details of students where city that starts with m

SQL> select * from studs where city like 'm%';

Find all the details of students where city that have "e" in third position

SQL> select * from studs where city like '__e%';

Find all the details of students where city that start with c and ends with i

SQL> select * from studs where city like 'c%i';

ADITYA UNIVERSITY

Arithmetic Operators

- The addition operator in SQL is used to add two numeric values. It is similar to the "plus" symbol in basic mathematics.
- The subtraction operator in SQL is used to subtract one numeric value from another. It is similar to the "minus" symbol in basic mathematics.
- The multiplication operator in SQL is used to perform mathematical multiplication on numeric values. It allows us to multiply two columns or numeric expressions together, resulting in a new value representing the product of the operands.
- The division operator (/) in SQL is used to perform mathematical division on numeric values. It allows us to divide one numeric operand by another, resulting in a new value representing the quotient of the division.
- The modulus operator (%) in SQL is used to find the remainder after dividing one numeric value by another. It returns the integer remainder of the division operation.

Arithmetic Operators:

```
SQL> select 4+5 as "add" from dual;
  add
---------
        9
 SQL> select 9-5 from dual;
  9-5
---------
        4
SQL> select 2*3 as mul from dual;
       MUL
---------
        6
```

ADITYA UNIVERSITY

```
SQL> select 8/4 from dual;
        8/4
----------
         2
SQL> select mod(8,4) from dual;
   MOD(8,4)
----------
         0
```

ADITYA UNIVERSITY

Logical Operators

A logical function performs a logical operation or comparison on objects and expressions and returns a boolean value

NOT	Reverses the value of any other Boolean operator.
OR	TRUE if any of the conditions separated by OR is TRUE
AND	TRUE if all the conditions separated by AND are TRUE.
BETWEEN	TRUE if the operand lies within the range of comparisons.

Logical Operators

```
SQL> select * from student;
SID                  SNAME                          AGE
-------------------- ------------------------------ ----------
501                  akash                          21
502                  thanmayi                       24

SQL> select sid,sname from student where age>20 and age<30;
SID                  SNAME
-------------------- ------------------------------
501                  akash
502                  thanmayi

SQL> select sid,sname from student where age>=21 or age<25;
 SID                  SNAME
-------------------- ------------------------------
501                  akash
502                  thanmayi
```

Logical Operators

```
SQL> select sid,sname from student where age between 20 and 23;
SID                  SNAME
-------------------- ----------------------------
501                  akash

SQL> select sid,sname from student where not age=21;
SID                  SNAME
-------------------- ----------------------------
502                  thanmayi
```

ADITYA UNIVERSITY

Comparison Operators

The **Comparison Operators** in SQL compare two different data of SQL table and check whether they are the same, greater, and lesser. The SQL comparison operators are used with the WHERE clause in the SQL queries.

Operator	Meaning
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

ADITYA UNIVERSITY

Comparison Operators

```
SQL> select sid,sname from student where age=21;
 SID                 SNAME
-------------------- ----------------------------
501                  akash
SQL> select sid,sname from student where age>=21;
SID                  SNAME
-------------------- ----------------------------
501                  akash
502                  thanmayi
SQL> select sid,sname from student where age<=30;
SID                  SNAME
-------------------- ----------------------------
501                  akash
502                  thanmayi
SQL> select sid,sname from student where age<>50;
SID                  SNAME
-------------------- ----------------------------
501                  akash
502                  thanmayi
```

String Functions

Concat
It concatenates two strings
Syntax:-CONCAT(input_string1, input_string2)
Lower
It converts a string to lowercase
Syntax:-LOWER(input_string)
Upper
Convert a string to uppercase
UPPER(input_string)
Length
Returns the length of the string
Syntax:-length(str)
ASCII
Returns the ASCII code value of a character
Syntax:-ASCII(input_string)
Chr
Convert a ASCII value to a character
Syntax:-chr(int)
Reverse
It returns the reverse order of a character string
Syntax:- REVERSE (input_string)

String Functions

Lpad

It adds left padding with the given symbol to make the string of given size.

Syntax:- lpad(input_string,size,symbol)

Rpad

It adds right padding with the given symbol to make the string of given size.

Syntax:- rpad(input_string,size,symbol)

Ltrim

It trims the given character from the left of the string.

Syntax:-ltrim(input_string,character)

Rtrim

It trims the given character from the right of the string.

Syntax:-rtrim(input_string,character)

String Functions

Initcap

The INITCAP function converts the first letter of each word in a string to uppercase, and converts any remaining characters in each word to lowercase.

Syntax:-initcap(input_string)

Instr:

Searches a string for substring using characters and return the position in the string.

Syntax:-instr(string,character)

Replace

It replaces a character with a new character.

Substr

Returns substring from a given string length

Syntax:-substr(string,startindex,length)

String Functions

```
SQL> select concat('aditya','engg') from dual;
CONCAT('AD
---------
adityaengg
 SQL> select concat(concat('aditya','engg'),'college') from dual;
 CONCAT(CONCAT('AD
----------------
adityaenggcollege
 SQL> select 'aditya'||'engg' from dual;
 'ADITYA'||
---------
adityaengg
 SQL> select lpad('aditya',15,'*')as lpad from dual;
 LPAD
--------------
*********aditya
```

String Functions

```
SQL> select rpad('aditya',15,'*')as rpad from dual;
RPAD
---------------
aditya*********
SQL> select ltrim('123123123rama123','123')from dual;
LTRIM('
-------
rama123
SQL> select rtrim('123123123rama123','123')from dual;
RTRIM('123123
-------------
123123123rama
SQL> select upper('aditya') from dual;
UPPER(
------
ADITYA
```

String Functions

```
SQL> select lower('ADITYA') from dual;
 LOWER(
------
aditya
 SQL> select length('aditya') from dual;
 LENGTH('ADITYA')
----------------
              6
 SQL> select substr('abcdefg',-3,2)from dual;
 SU
--
ef
 SQL> select instr('abab','b')from dual;
 INSTR('ABAB','B')
-----------------
               2
```

String Functions

```
SQL> select ASCII('A') from dual;
 ASCII('A')
----------
        65
 SQL> select chr(97) from dual;
 C
-
a
 SQL> select reverse('aditya') from dual;
 REVERS
------
aytida
 SQL> select initcap('adityaengg') from dual;
 INITCAP('AD
-----------
Aditya Engg
```

Date Functions

EXTRACT():

Returns a single part of a date/time.

Syntax:-EXTRACT(unit FORM date);

Add_months(sysdate,integer)

It gives the same day,n number of months away.

Last_day(sysdate)

It displays the last day of the month that certain date.

Next_day(sysdate,day)

It always returns a DATE value that represents the next weekday after the date

Months_between()

It retuns the number of moths between date1 and date2.

Date Functions

```
SQL> select sysdate from dual;

 SYSDATE
---------
07-OCT-21

 SQL> select sysdate+1 from dual;

 SYSDATE+1
---------
08-OCT-21

 SQL> select sysdate-1 from dual;

 SYSDATE-1
---------
06-OCT-21

SQL> select extract(year from sysdate) from dual;

 EXTRACT(YEARFROMSYSDATE)
------------------------
                    2021
```

Date Functions

```
SQL> select extract(month from sysdate) from dual;
 EXTRACT(MONTHFROMSYSDATE)
-------------------------
                       10
SQL> select to_char(sysdate,'yyyy/mm/dd') from dual;
 TO_CHAR(SY
----------
2021/10/07
 SQL> select to_char(sysdate,'HH:MM:SS') from dual;
 TO_CHAR(
--------
05:10:24
SQL> select months_between(to_date('09-dec-2020','dd-mm-yyyy'),to_date('09-dec-2019','dd-mm-yyyy'))
from dual;

MONTHS_BETWEEN(TO_DATE('09-DEC-2020','DD-MM-YYYY'),TO_DATE('09-DEC-2019','DD-MM-
--------------------------------------------------------------------------------
                                                                              12
```

Date Functions

SQL> select add_months(sysdate,2) from dual;

ADD_MONTH

07-DEC-21

SQL> select next_day(sysdate,'Thursday') from dual; finds the date of thursday from current system

NEXT_DAY(

14-OCT-21

SQL> select next_day('10-dec-2019','Tuesday') from dual; calculates the date of tuesday

NEXT_DAY(

17-DEC-19

SQL> select last_day(sysdate) from dual;

LAST_DAY(

31-OCT-21

ADITYA UNIVERSITY

Numeric Functions

ABS(): It returns the absolute value of a number.

CEIL(): It returns the smallest integer value that is greater than or equal to a number.

EXP(): It returns e raised to the power of number

FLOOR(): It returns the largest integer value that is less than or equal to a number.

LOG(): It returns the logarithm of a number

MOD(): It returns the remainder of n divided by m.

POWER(): It returns m raised to the nth power.

ROUND(): It returns a number rounded to a certain number of decimal places.

SIGN(): It returns a value indicating the sign of a number

SQRT(): It returns the square root of a number

TRUNCATE(): It returns 7.53635 truncated to 2 places right of the decimal point

Numeric Functions

```
SQL> select abs(19) from dual;
  ABS(19)
---------
       19
SQL> select abs(-19) from dual;
  ABS(-19)
---------
       19
 SQL> select sign(19) from dual;
  SIGN(19)
---------
        1
 SQL> select sign(-19) from dual;
 SIGN(-19)
---------
       -1
 SQL> select power(3,2) from dual;
 POWER(3,2)
---------
        9
```

Numeric Functions

```
SQL> select sqrt(9) from dual;
  SQRT(9)
---------
         3
 SQL> select ceil(2.2) from dual;
  CEIL(2.2)
---------
         3
 SQL> select ceil(-2.2) from dual;
 CEIL(-2.2)
---------
        -2
 SQL> select floor(2.2) from dual;
 FLOOR(2.2)
---------
         2
 SQL> select floor(-2.2) from dual;
 FLOOR(-2.2)
----------
```

Numeric Functions

```
SQL> select mod(150,7) from dual;
MOD(150,7)
----------
         3
SQL> select round(66.666,2) from dual;
 ROUND(66.666,2)
---------------
          66.67
SQL> select trunc(66.666,2) from dual;
TRUNC(66.666,2)
---------------
          66.66
SQL> select exp(3) from dual;
    EXP(3)
----------
20.0855369
SQL> select log(2,2) from dual;
 LOG(2,2)
----------
         1
```

ADITYA UNIVERSITY

Aggregate operators

The COUNT() function returns the number of rows that matches a specified criterion.

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

The AVG() function returns the average value of a numeric column.

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

The SUM() function returns the total sum of a numeric column.

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

The MIN() function returns the smallest value of the selected column.

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;.
```

The MAX() function returns the largest value of the selected column.

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

Queries

1) Count the number of sailors

SQL>select count(*) from sailors;

2) Find the average age of all sailors

SQL> select avg(s.age) from sailors s;

3) Find the maximum age of sailors

select max(s.age) from sailors s;

4) Find the minimum age of sailors

select min(s.age) from sailors s;

5) Find the sum of age of sailors whose age is greater than 30.

SQL> select sum(s.age) from sailors s where s.age>30;

6) Find the average age of all sailors with a rating of 10.

SQL> select avg(s.age) from sailors s where s.rating=10;

Group By, Having, Order by

Group by:

It is used with aggregate functions(COUNT,MAX,MIN,SUM,AVG) to group the resultset by one or more columns.

Having clause:

It was added to Select because the Where keyword could not be used with aggregate functions.

Order by:

It is used to sort the resultset in ascending or descending order.

Syntax:

Select column_list from table_list

Where condition

Group by column_name

Having condition

Order by column_name asc/desc

Group By, Having, Order by

select * from company;

companyn amount

wipro 5000

ibm 9000

dell 10000

wipro 4000

dell 6000

Queries

Find the sum of amount of each company.

SQL> select companyn,sum(amount) from company group by companyn;

COMPANYN SUM(AMOUNT)

---------- ----------

wipro 9000

dell 16000

ibm 9000

Group By, Having, Order by

find the count of all the rows grouped by each company name.

SQL> select companyn,count(*) from company group by companyn;

COMPANYN	COUNT(*)
wipro	2
dell	2
ibm	1

find the minimum amount of each company.

SQL> select companyn,min(amount) from company group by companyn;

COMPANYN	MIN(AMOUNT)
wipro	4000
dell	6000
ibm	9000

Group By, Having, Order by

find the max amount of each company.

SQL> select companyn,max(amount) from company group by companyn;

COMPANYN	MAX(AMOUNT)
wipro	5000
dell	10000
ibm	9000

find the count of all the rows grouped by each company name and having count greater than 1.

SQL> select companyn,count(*) from company group by companyn having count(*)>1;

COMPANYN	COUNT(*)
wipro	2
dell	2

Group By, Having, Order by

find sum of amount of each company and having sum of amount greater than 10000.

SQL> select companyn,sum(amount) from company group by companyn having sum(amount)>10000;

COMPANYN	SUM(AMOUNT)
dell	16000