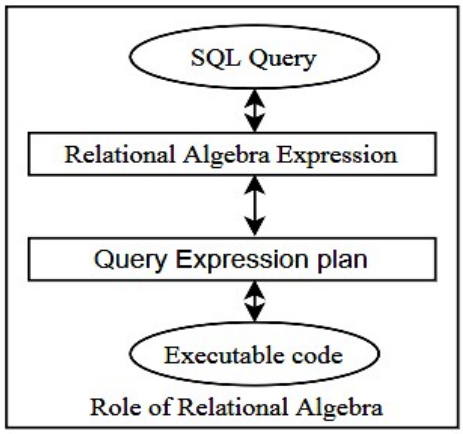
RELATIONAL ALGEBRA

RELATIONAL ALGEBRA

Relational algebra defines the basic set of operations of relational database model. A sequence of relational algebra operations forms a relational algebra expression. The result of this expression represents the result of a database query.

- ✓ A set of operators (unary or binary) that take relation instances as arguments and return new relations.
- ✓ Gives a procedural method of specifying a retrieval query
- ✓ Forms the core component of a relational query engine
- ✓ SQL queries are internally translated into RA expressions
- ✓ Provides a framework for query optimization



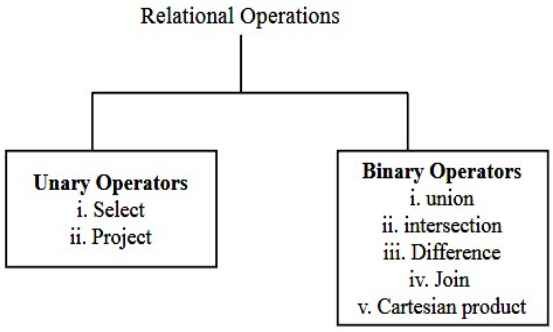
Role of Relational Algebra

Relational Operations

A collection of simple 'low-level' operations used to manipulate relations.

- ✓ It provides a procedural way to query a database.
- ✓ Input is one (or) more relations.
- ✓ Output is one relation.

They ar unary and binary operations



Select Operator(σ)

Select operator is a unary operator. It can be used to select those tuples of a relation that satisfy a given condition.

Notation: $\sigma_{\square}(r)$

σ: Select operator(read as sigma)

□: Selection condition

r: Relation name

Result is a relation with the same scheme as 'r' consisting of the tuples in r that satisfy condition

Syntax: $\boldsymbol{\sigma}_{condition}$ (relation)

Example: See the following table 'Student'

Roll	Name	Address	Age
1	Rahim	Dhaka	16
2	Karim	Chandpur	18
3	Morjina	Dhaka	32
4	Jorina	Sylhet	23

Note: Selection condition can use following oerators:<, ≤, ≥, >, =, ≠

Example:

1	**Select all record from Student table**
SQL	Select * from Student
RA	$\sigma_{(Student)}$
2	**Select all record from Student table where address is dhaka**
SQL	Select * from Student where Address="Dhaka"
RA	$\sigma_{Address='Dhaka'}$ (Student)
3	**Select all record from Student table where address is Dhaka and age more than 25**
SQL	Select * from Student where Address="Dhaka" and Age>25.
RA	$\sigma_{Address='Dhaka' \ AND \ Age>25}$ (Student)

Project Operator (Π)

The project operator is unary operator. It can be used to keep only the **required attributes** (Specific attribute selection) of a relation instance and throw away others.

1	**Select the nameof all Student from Student table**
SQL	Select Name from Student
RA	Π_{Name} (Student)
2	**Find name and age from student table**
SQL	Select Name, Age from Student
RA	$\Pi_{Name, Age}$ (Student)
	Using both select and Project
3	**Find name and address from all student those age is below 28**
SQL	Select Name,Address from Student where Age<28.

RA	$\Pi_{Name, Address}$ ($\sigma_{Age<28}$(Student))
4	**Select name and address from Student table where address is Dhaka and age more than 25**
SQL	Select Name, Address from Student where Address=”Dhaka” and Age>25.
RA	$\Pi_{Name, Address}$ ($\sigma_{Address='Dhaka' \text{ AND } Age>25}$ (Student))

More Example of Relational Algebra

Consider the database schema below:

employee (ename, street, city)
works (ename, cname, salary, jdate)
company (cname, city)
manages (ename, mname)
Note: A manager is also an employee of a company.

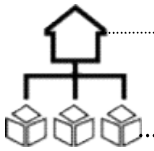
1	**Find the names of all employees who work for CTI Corporation.** SQL: select ename from works where cname = ‘CTI Corporation’; RA: Π_{ename} ($\sigma_{cname=\text{" CTI Corporation"}}$ (works))
2	**Find the names and cities of residence of all employees who work for CTI Corporation** SQL: select ename, city from employee natural join works where cname = ‘**CTI Corporation**’; RA: $\Pi_{ename, city}$ ($\sigma_{cname = \text{" CTI Corporation"}}$ (employee ⋈ works))
3	**Find the names, street address, and cities of residence of all employees who work for CTI Corporation and earn more than Tk. 5000.** SQL: select ename, street, city from employee natural join works where cname = ‘**CTI Corporation**’ and salary > 5000; RA: $\Pi_{ename, street, city}$ ($\sigma_{cname = \text{“CTI Corporation”} \wedge salary> 5000}$ (employee ⋈ works))
4	**Find the names of all employees who live in the same city and on the same street as do their managers.** SQL: select employee.ename from employee natural join manages, employee as emp where mname = emp.ename and employee.street = emp.street and employee.city = emp.city; RA: $\Pi_{employee.ename}$ ($\sigma_{mname = emp.ename \wedge employee.street = emp.street \wedge employee.city = emp.city}$ (employee ⋈ manages × emp (employee)))
5	**Find the names of all employees in this database who do not work for CTI Corporation** SQL: select ename from works where cname<> ‘**CTI Corporation’**; RA: Π_{ename} ($\sigma_{cname \neq \text{" CTI Corporation"}}$ (works))
6	**Display the average salary of each company.** SQL: select cname, avg(salary) from works. RA: $G_{AVG(salary)}$(works) [here G denoted group by]
7	**Display the average salary of each company except CTI Corporation.** SQL: select cname, avg(salary) from works where cname<> ' **CTI Corporation**' group by cname; RA: $\Pi_{cname} G_{avg(salary)}$ ($\sigma_{cname\neq \text{"Square Pharma"}}$ (works))

Translate the following query into relational algebra *[SBL ADA-2020]*

SELECT marks from STUDENTS WHERE MARKS > 80

a) $\pi_{marks}(\sigma_{marks>80}(student))$ b) $\sigma_{marks}(\pi_{marks>80}(student))$

c) $\pi_{student}(\sigma_{marks>80}(student))$ d) $\pi_{student}(\sigma_{marks>80}(mark))$ Ans:**a**



STRUCTURED QUERY LANGUAGE (SQL)

SQL (Structured Query Language)

If you want to get some information from a database file, you need a query. A query is a user request to retrieve data or information with a certin condition. The program will go through all records in the database file and select those records that satisfy the condition.The result of the query will be stored in the form of a table. It must be a single table.

Features of SQL

- ✓ SQL is a language of database. It includes database creation, deletion, fetching rows and modifying rows.
- ✓ SQL is a structured query language for storing, manipulation and retrieving data stored in relational databases.
- ✓ It allows users to create and drop databases and tables.
- ✓ It allows users to create view, functions in a database.
- ✓ Allows users to set permissions on tables and views
- ✓ The standard SQL commands to interact with relational database are CREATE, SELECT, UPDATE, INSERT, DROP and DELETE.

Query Processing in DBMS

Query Processing is the activity performed in extracting data from the database. In query processing, it takes various steps for fetching the data from the database. The steps involved are:

1. Parsing and translation
2. Optimization
3. Evaluation

1. What is the wrong statement for SQL? *[SBL ADA-2020]*

a) Non procedural language b) Input can be several table

c) Output is always a single table d) Output can be multiple table Ans: **d**

2. Which one is not the step of SQL query processing? *[SBL ADA-2020]*

a) Parsing b) Translation c) Optimization d) none Ans: **d**

Basic Structure

Basic structure of an SQL expression consists of **select, from** and **where** clauses.

- ✓ **select** clause lists attributes to be copied - corresponds to relational algebra **project**.
- ✓ **from** clause corresponds to Cartesian product - lists relations to be used.
- ✓ **where** clause corresponds to selection predicate in relational algebra.

SQL is divided into two languages

1. DML (data manipulation language)
   - ✓ SELECT: Extracts data from a database table.
   - ✓ UPDATE: Updates data in a database table.
   - ✓ DELETE: Deletes data from a database table.
   - ✓ INSERT INTO: Inserts new data into database table.
2. DDL (data definition language)
   - ✓ CREATE TABLE - creates a new database table.
   - ✓ ALTER TABLE - Alters a database table.
   - ✓ DROP TABLE - deletes a database table.
   - ✓ CREATE INDEX - Creates an index (search key).
   - ✓ DROP INDEX - Deletes an index,
   - ✓ RENAME - Changes the name of the table.

Consider the following table and learn the query

Table – EmployeeDetails

EmpId	FullName	ManagerId	DateOfJoining
121	John Snow	321	01/31/2014
321	Walter White	986	01/30/2015
421	Kuldeep Rana	876	27/11/2016

Table – EmployeeSalary

EmpId	Project	Salary
121	P1	8000
321	P2	1000
421	P1	12000

1. **Write a SQL query to fetch the count of employees working in project 'P1'.**

```
SELECTCOUNT(*) FROMEmployeeSalaryWHERE Project = 'P1';
```

Note: operates used in where condition is =, <>(not equal or !=),>,<,>=,<=

2. **Write a SQL query to fetch employee names having salary greater than or equal to 5000 an less than or equal 10000.**

```
SELECT FullName FROM EmployeeDetails WHERE EmpId IN(SELECT
EmpId FROM EmpolyeeSalary WHERE Salary BETWEEN 5000 AND
10000);
```

3. **Write a SQL query to fetch project-wise count of employees sorted by project's count in descending order.**

```
SELECT Project, count(EmpId) EmpProjectCount FROM
EmployeeSalary GROUPBY Project ORDER BY EmpProjectCount DESC;
```

4. **Write a query to fetch only the first name(string before space) from the FullName column of EmployeeDetails table.**

```
SELECTMID(FullName, 0, LOCATE(' ',FullName)) FROM
EmployeeDetails;
```

SQL Server-Using SUBSTRING

```
SELECT SUBSTRING(FullName, 0, CHARINDEX(' ',FullName)) FROM
```

```
EmployeeDetails;
```

Also, we can use LEFT which returns the left part of a string till specified number of characters.

```
SELECT LEFT(FullName, CHARINDEX(' ',FullName) - 1) FROM
EmployeeDetails;
```

5. **Write a query to fetch employee names and salary records. Return employee details even if the salary record is not present for the employee.**

```
SELECTE.FullName, S.Salary
FROM EmployeeDetails E LEFT JOIN EmployeeSalary S
ONE.EmpId = S.EmpId;
```

6. **Write a SQL query to fetch all the Employees who are also managers from EmployeeDetails table.**

```
SELECT DISTINCTE.FullName FROM EmpDetails E
INNER JOIN EmpDetails MONE.EmpID = M.ManagerID;
```

7. **Write a SQL query to fetch all employee records from EmployeeDetails table who have a salary record in EmployeeSalary table.**

```
SELECT * FROM EmployeeDetails E
WHERE EXISTS
(SELECT * FROM EmployeeSalary S WHERE  E.EmpId = S.EmpId);
```

Subquery return one row when
SELECT * FROM `student` WHERE id = (select id from student where salary =(select max(salary) from student))
Subquery return more then one row when
Subquery return more thenone row when
SELECT * FROM `student` WHERE id in (select id from student where salary<(select max(salary) from student))

8. **Write a SQL query to fetch duplicate records from a table.**

```
SELECT EmpId, Project, Salary, COUNT(*)FROM EmployeeSalary
GROUP BY EmpId, Project, Salary HAVING COUNT(*) > 1;
```

9. **Write a SQL query to remove duplicates from a table without using temporary table.**

```
DELETE FROM EmployeeSalary
WHERE EmpId IN (
SELECT EmpId
FROM EmployeeSalary
GROUP BY Project, Salary
HAVING COUNT(*) > 1));
```

Using rowId in Oracle-

```
DELETE FROM EmployeeSalary
WHERE rowed NOT IN
(SELECT MAX(rowid) FROM EmployeeSalary GROUP BY EmpId);
```

10. **Write a SQL query to fetch only odd rows from table.**

```
SELECTE.EmpId, E.Project, E.Salary
FROM (
```

```
    SELECT *, Row_Number() OVER(ORDER BY EmpId) AS RowNumber
    FROM EmployeeSalary
) E
WHEREE.RowNumber % 2 = 1
```

11. **Write a SQL query to fetch only even rows from table.**

```
SELECTE.EmpId, E.Project, E.Salary
FROM (
    SELECT *, Row_Number() OVER(ORDER BY EmpId) AS RowNumber
    FROM EmployeeSalary
) E
WHERE E.RowNumber % 2 = 0
```

12. **Write a SQL query to create a new table with data and structure copied from another table.**

```
SELECT * INTO newTable FROM EmployeeDetails;
```

13. **Write a SQL query to create an empty table with same structure as some other table.**

```
SELECT * INTO newTable FROM EmployeeDetails WHERE 1 = 0;
```

14. **Write a SQL query to fetch common records between two tables.**

```
SELECT * FROM EmployeeSalary
INTERSECT SELECT * FROM ManagerSalary
```

15. **Write a SQL query to fetch records that are present in one table but not in another table.**

```
SELECT * FROM EmployeeSalary
MINUS
SELECT * FROM ManagerSalary
```

16. **Write a SQL query to find current date-time.**

mySQL-

```
SELECTNOW();
```

SQL Server-

```
SELECTgetdate();
```

Oracle-

```
SELECT SYSDATE FROM DUAL;
```

17. **Write a SQL query to fetch all the Employees from EmployeeDetails table who joined in Year 2016.**

```
SELECT * FROM EmployeeSalary
WHERE DateOfJoining BETWEEN '01-01-2016' AND date '31-12-
2016';
```

Using YEAR in mySQL-

```
SELECT * FROM EmployeeSalary
WHERE YEAR(DateOfJoining) = '2016';
```

18. **Write a SQL query to fetch top n records?**

In mySQL using LIMIT-

```
SELECT * FROM EmployeeSalary ORDERBY Salary DESC LIMIT N
```

In SQL server using TOP command-

```
SELECT TOP N * FROM EmployeeSalary ORDER BY Salary DESC
```

In Oracle using ROWNUM-

```
SELECT * FROM (SELECT * FROM EmployeeSalary ORDER BY Salary
DESC)
WHERE ROWNUM <= 3;
```

19. **Write SQL query to find the nth highest salary from table.**

Using Top keyword (SQL Server)-

```
SELECT TOP 1 Salary
FROM (
      SELECTDISTINCT TOP N Salary
      FROM Employee
      ORDERBY Salary DESC
      )
ORDER BY Salary ASC
```

Using limit clause(mySQL)-

```
SELECT Salary FROM Employee ORDERBY Salary DESCLIMIT N-1,1;
```

20. **Write SQL query to find the 3rd highest salary from table without using TOP/limit keyword.**

```
SELECT Salary
FROM EmployeeSalary Emp1
WHERE 2 = (
                SELECT COUNT( DISTINCT ( Emp2.Salary ) )
                FROM EmployeeSalary Emp2
                WHERE Emp2.Salary > Emp1.Salary
            )
```

For nth highest salary-

```
SELECT Salary
FROM EmployeeSalary Emp1
WHERE N-1 = (
                SELECT COUNT( DISTINCT ( Emp2.Salary ) )
                FROM EmployeeSalary Emp2
                WHERE Emp2.Salary > Emp1.Salary
            )
```

21. **LIKE condition**

The LIKE operator is used to list all rows in a table whose column values match a specified pattern. It is useful when you want to search rows to match a specific pattern, or whenyou do not know the entire value. For this purpose, we use a wildcard character '%'.
The LIKE condition is used to specify a search for a pattern in a column.
A '%' sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.
Syntax: SELECT column FROM table WHERE column LIKE pattern

```
SELECT * FROM EmployeeDetails WHERE FullName LIKE'joh%'
```

```
--Finds any values that start with "joh"
SELECT * FROM EmployeeDetails  WHERE FullName LIKE'%joh'
--Finds any values that end with "joh"
SELECT * FROM EmployeeDetailsWHERE FullName LIKE '%joh%'
--Finds any values that any position with "joh"
SELECT * FROM EmployeeDetails WHERE FullName LIKE '-h%'
--Finds any values that have 'r' in the second position
```

22 **Write a query to select top ten percent roll number from grade table.**

```
Select * from grade where round((selectcount(*) from
grade)*.10)>(select count(distinct(grade.id) ) from grade
where grade.id>grade.id)
```

Oracle Database queries (Table Create ,DDL,DML,Trigger,Function)

```
create table stu_info
(
id varchar2(12),
name varchar2(12),
dept varchar2(12),
year varchar2(12),
constraint pk_001 primary key(id)
);

create table stu_det
(
id varchar2(12),
acc number,
dist varchar2(12),
income number,
constraint pk_02 primary key(id),
constraint fk_001 foreign key(id)
references stu_info(id)
);
create table stu_acc
(
id varchar2(12),
mob number,
bra varchar2(12),
bal number,
constraint pk_03 primary key(id),
constraint fk_002 foreign key(id)
references stu_det(id)
);
```

Inseret Table stu_info

```
insert into stu_info values('CE08010','Rasel','CSE','4th');
insert into stu_info values('CE09010','Faysal','CSE','3rd');
```

Inseret Table stu_det

```
insert into stu_det values('CE08010',111,'Dhaka',4500);
insert into stu_det values('CE09010',222,'Khulna',7000);
```

Inseret Table stu_acc

```
insert into stu_acc
values('CE08010',01722410854,'Dhaka',5600);
insert into stu_acc
values('CE09010',01923456789,'Khulna',2000);
```

Quaries:

1. select stu_info.id,stu_info.name,stu_acc.mob,stu_det.income,stu_acc.bal from stu_info,stu_det,stu_acc where stu_info.id=stu_det.id and stu_det.id=stu_acc.id;
2. select distinct dept,bal,income from stu_info,stu_det,stu_acc where stu_info.id=stu_det.id and stu_det.id=stu_acc.idgroup by dept,bal,income having avg(bal)>3000;
3. update stu_acc set bal=case when bal>4000 then bal*1.15 else bal*1.10 end;
4. select count(name) from stu_info where name like '%Sh%' or name like '%ra%';
5. select count(id) from stu_det where income>= 1000 and income<=5000;

Or, select count(id) as Range from stu_det where income between 1000 and 5000;

6. update stu_info set name='shafiq' where idin(select a.id from stu_acc a where a.bal>7500);

Create Function:

```
create table funn
(
id varchar2(12),
name varchar2(12),
acc number,
bal number
);
insert into funn values('CE10031','Nipu',101,100);
insert into funn values('CE10032','fuad',102,200);
insert into funn values('CE10033','Akas',103,300);
```

Function that will take id on it's parameter and return name as it's output:

```
CREATE or replace FUNCTION get_bal(id1 IN varchar2)
   RETURN varchar2
   IS
name varchar2(10);
BEGIN
   SELECT a.name
   INTO name
   FROM funn a
   WHERE a.id= id1;
RETURN(name);
 END;
```

```
select get_bal('CE10032') from dual;
```

Function that will take id id on parameter return balance

```
create table balance_info
(
id number(20),
balance number(20)
);
insert into balance_infovalues(4,5000.5478);
CREATE or replace FUNCTION get_bal(acc_no IN NUMBER)
   RETURN NUMBER
   IS
acc_balNUMBER(10,2);
BEGIN
   SELECT a.balance
   INTO acc_bal
   FROM balance_info a
   WHERE a.id = acc_no;
RETURN(acc_bal);
 END;
select get_bal(4) from dual;
```

PROCEDURE

```
create or replace procedure select_student(c_id in
varchar2,mycur out sys_refcursor,mycur1 out
sys_refcursor,mycur2 out sys_refcursor)
is
begin
open mycur for
SELECT* FROM stu_info WHERE stu_info.ID=c_id;
open mycur1 for
SELECT* FROM stu_det WHERE stu_det.ID=c_id;
open mycur2 for
SELECT* FROM stu_acc WHERE stu_acc.ID=c_id;

end;
/

set serveroutput ondeclare

testcursys_refcursor;
testcur1 sys_refcursor;
testcur2 sys_refcursor;
row stu_info%rowtype;
row1 stu_det%rowtype;
row2 stu_acc%rowtype;
```

```
begin
select_student('CE09010',testcur,testcur1,testcur2);
loop
fetch testcur into row;
fetch testcur1 into row1;
fetch testcur2 into row2;
exit when testcur%NOTFOUND;
exit when testcur1%NOTFOUND;
exit when testcur2%NOTFOUND;
dbms_output.put_line(row.id||'----'||row.name||'---
'||row1.acc||'---'||row1.dist||'---'||row2.mob||'---
'||row2.bal);
end loop;
end;
/
```

TRIGGER

```
create table stu_1
(
id varchar2(12),
name varchar2(12),
dept varchar2(12)
);
create table stu_2
(
id varchar2(12),
name varchar2(12),
dept varchar2(12)
);
CREATE OR REPLACE TRIGGER before_stu_1_name_insert
 before INSERT
  ON stu_1
 FOR EACH ROW
  BEGIN
dbms_output.put_line('id = ' || :old.id);
dbms_output.put_line('name = ' || :old.name);
dbms_output.put_line('dept = ' || :old.dept);
  INSERT INTO stu_2 (id, name, dept)VALUES (:new.id,
:new.name, :new.dept);
END before_stu_1_name_insert;
insert into stu_1 values('10033','fuad','cse');
insert into stu_1 values('10043','akas','cse');
```

TRIGGER2

```
create table myaudit(
id VARCHAR2(10) NOT NULL,
old_value VARCHAR2(30),
new_value VARCHAR2(30)
);
create table Employee(
ID VARCHAR2(2) NOT NULL,
First_Name VARCHAR2(10),
Last_Name VARCHAR2(10),
Start_Date DATE,
End_Date DATE,
Salary Number(8,2),
City VARCHAR2(10),
Description VARCHAR2(11)
);
insert into Employee values ('01','Jason','Martin',
to_date('19960725','YYYYMMDD'),
to_date('20060725','YYYYMMDD'), 1234.56, 'Toronto',
'Programmer');
insert into Employee values('02','Alison',   'Mathews',
to_date('19760321','YYYYMMDD'),
to_date('19860221','YYYYMMDD'), 6661.78,
'Vancouver','Tester');
insert into Employee values('03','James',    'Smith',
to_date('19781212','YYYYMMDD'),
to_date('19900315','YYYYMMDD'), 6544.78,
'Vancouver','Tester');
insert into Employee values('04','Celia',    'Rice',
to_date('19821024','YYYYMMDD'),
to_date('19990421','YYYYMMDD'), 2344.78,
'Vancouver','Manager');
```

TRIGGER

```
CREATE OR REPLACE TRIGGER before_employee_salary_update
  BEFORE UPDATE OF salary
  ON employee
 FOR EACH ROW WHEN (new.salary<old.salary * 0.75)
  BEGIN
dbms_output.put_line('id = ' || :old.id);
dbms_output.put_line('Old salary = ' || :old.salary);
dbms_output.put_line('New salary = ' || :new.salary);
dbms_output.put_line('The salary reduction is more than 25%');
 INSERT INTO Myaudit (id, old_value, new_value)VALUES
(:old.id, :old.salary, :new.salary);
END before_employee_salary_update;
```

Effect:

```
update employee set salary = 0;
select * form employee;
select * form myaudit;
```

Previous Year Questions

1. Table programmer has 10 records. It has a non-NULL SALARY column which is also unique. What will be the output of the SQL statement-? *[Com AP-2021]*

"SELECT COUNT(*) FROM programmer WHERE SALARY > ALL (SELECT SALARY FROM programmer)

a) 0 b) 5 c) 9 d) 10 Ans. **a**

2. Consider the following relational data table, Employee. Now find the output for the following SQL Statement

SELECT COUNT (*) FROM Employee, Employee, Employee

eid	name	dept
E101	John	HRM
E102	Lucky	Marketing
E103	Rick	Management

a) 4 b) 16 c) 32 d) 27 Ans. **d**

3. In SQL, aggregate functions can be used in the select list or the ------- clause of a select statement or subquery. They cannot be used in a-------- clause.*[SBL,RBL,BKB(AP)- 2018]*

a) Where, having b) Having, where

c) Group by. Having d) Group by. Where Ans. **b**

4. Write a query to select top 10% row from grade table. *[HBL,KB(AP)- 2018]*

5. Three table are given : *[PalliSanchay Bank(ADA)- 2018]*

Customer(cust_id, Name,Address,Sales_id), Order(Order_id,cust_id,Date,Sales_id), Salesman(Sales_id,commision)

I. Find the customer details for thoussalesman get commission greater than 12%commission.

Ii. Count the salesman by their order_id and date.

6. Which one is not unary operation in relational algebra? *[SBL ofc(IT)-2020]*

a) Selectb) Project c) Union d) Rename Ans:**c**

7. Query to find out even number row (id column) from given table. *[RAKUB-2021]*

Solution: *Select * from table where id % 2 = 0*

8. Suppose you've two tables(Employee, Department) in a Database and Employee

table three cell(emp_name, dept_id,salary) also Department table two cell(dept_id, dept_name), now update the salary 10% increase value from Department table. Give the appropriate example. *[Pubali Bank Ltd(SO)- 2018]*

9. **Write an SQL query to insert a tuple in the table: Employee(ID, Name, Designation, Salary).** *[NESCO (AM)- 2018]*

10. **Show second maximum salary from a table.** *[BPSC (AP)- 2018] [BB (AP)- 2016]*

11. **Department wise salary calculation using sql from a table.** *[BPSC (AP)- 2018]*

12. **What is the SQL query for showing only the duplicate lists in student (stdid,name,gpa) table?** *[ABL O(ICT)- 2017]*

13. **Given two tables are employee(id,name,salary, dept_id) and department(dept_id, dept_name), Write a SQL to find MAX salary and AVERAGE Salary for Specific department.** *[DBBL(Software)- 2016]*

14. **Write a program in pl/sql to find the highest paid employees from employee table and store the data in HighestPaidEmp table**. *[DBBL(Software)- 2018]*

Model Test

1. In SQL, which of the following is not a data definition language commands?
a) RENAME **b)** REVOKE **c)** GRANT **d)** UPDATE

2. Here which of the following displays the unique values of the column?
SELECT _____ *dept_name* ***FROM*** *instructor;*
a) All b) From c) Distinct d) Name

3. The _____ clause is used to list the attributes desired in the result of a query.
a) Where b) Select c) From d) Distinct

4. Which of the following statements contains an error?
a) Select * from emp where empid = 10003;
b) Select empid from emp where empid = 10006;
c) Select empid from emp;
d) Select empid where empid = 1009 and lastname = 'GELLER';

5. A __ indicates an absent value that may exist but be unknown or that may not exist at all.
a) Empty tuple b) New value c) Null value d) Old value

6. Which of the following should be used to find the mean of the salary?
SELECT _____ ***FROM*** *instructor* ***WHERE*** *dept name= 'Comp. Sci.';*
a) Mean(salary) b) Avg(salary) c) Sum(salary) d) Count(salary)

7. Which of the join operations do not preserve non matched tuples?
a) Left outer join b) Right outer join c) Inner join d) Natural join

8. What type of join is needed when you wish to include rows that do not have matching values?

a) Equi-join b) Natural join c) Outer join d) All of the mentioned

9. Functional Dependencies are the types of constraints that are based on_____

a) Key b) Key revisited c) Superset key d) None of the mentioned

10. Which forms simplifies and ensures that there are minimal data aggregates and repetitive groups:

a) 1NF b) 2NF c) 3NF d) All of the mentioned

MODEL TEST ANSWER

1	2	3	4	5	6	7	8	9	10
d	c	b	d	c	b	c	c	a	c