CMR INSTITUTE OF TECHNOLOGY : : HYDERABAD

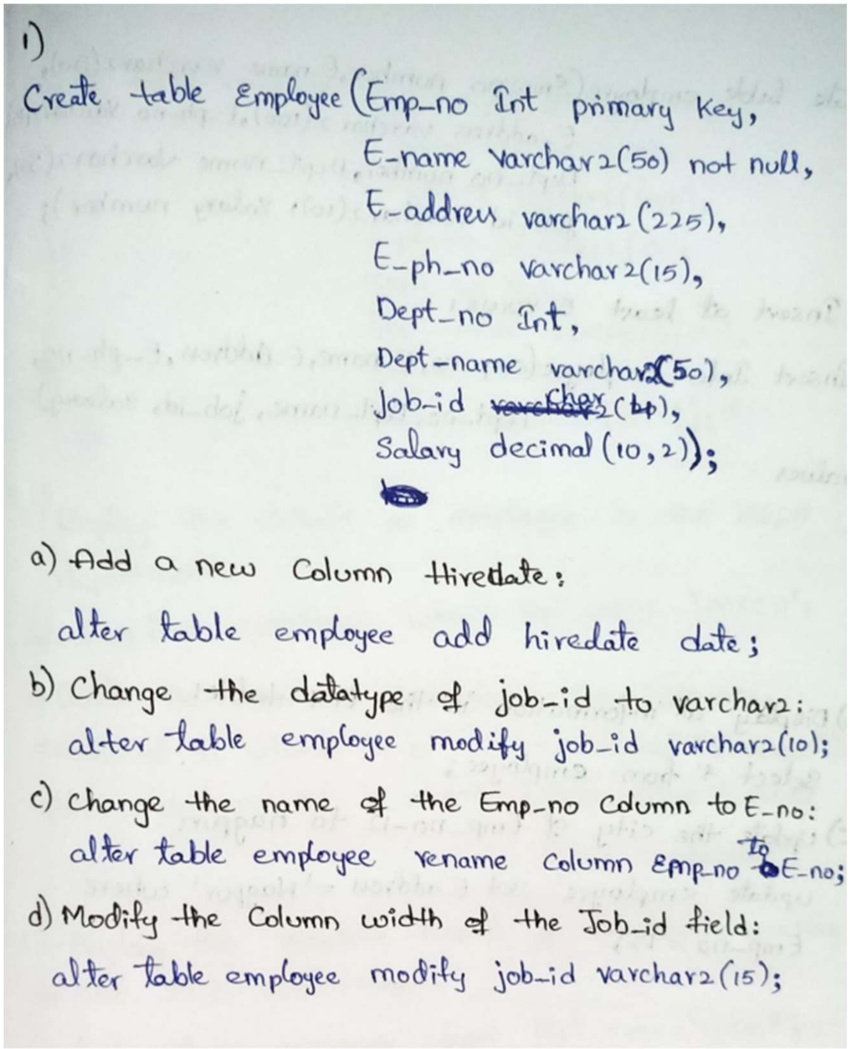
UGC AUTONOMOUS

II–B.Tech.–III–Semester

Database Management System

Lab External Examination 30th Jan’ 2024

1. Create a table EMPLOYEE with following schema:
   (Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name, Job_id, Salary)
   a. Add a new column; HIREDATE to the existing relation.
   b. Change the datatype of JOB_ID from char to varchar2.
   c. Change the name of column/field Emp_no to E_no.
   d. Modify the column width of the job field of emp table.
   e. Write the differences between nested query and correlated nested query with an example.



1)

```
Create table Employee (Emp_no Int primary Key,
                       E-name Varchar2(50) not null,
                       E-address varchar2 (225),
                       E-ph_no varchar2(15),
                       Dept_no Int,
                       Dept-name varchar2(50),
                       Job-id ~~varchar2~~ char (10),
                       Salary decimal (10,2));
```

a) Add a new Column Hiredate:

```
alter table employee add hiredate date;
```

b) Change the datatype of job-id to varchar2:

```
alter table employee modify job-id varchar2(10);
```

c) Change the name of the Emp-no Column to E-no:

```
alter table employee rename Column Emp-no to E-no;
```

d) Modify the Column width of the Job-id field:

```
alter table employee modify job-id varchar2(15);
```

e.) Differences Between Nested Query and Correlated Nested Query

Aspect	Nested Query	Correlated Nested Query
Definition	A subquery that is executed independently of the outer query.	A subquery that references columns from the outer query and executes repeatedly for each row of the outer query.
Execution	The subquery is executed once, and its result is used by the outer query.	The subquery is executed multiple times, once for each row processed by the outer query.
Dependency	Does not depend on the outer query's data.	Depends on the outer query's data for its execution.
Performance	Generally faster because the subquery is executed only once.	Can be slower due to repeated execution of the subquery for each row of the outer query.
Use Case	Typically used to filter or aggregate data independently of the outer query.	Used when the subquery needs to dynamically evaluate data based on each row of the outer query.

Examples

Nested Query Example:

Find employees who earn more than the average salary of all employees:

```
SELECT employee_id, name, salary
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
);
```

- **Explanation**: The subquery calculates the average salary of all employees. The outer query then retrieves employees whose salaries are greater than this value. The subquery runs once.

Correlated Nested Query Example:

Find employees who earn more than the average salary of their department:

```
SELECT e1.employee_id, e1.name, e1.salary, e1.department_id
FROM employees e1
WHERE e1.salary > (
    SELECT AVG(e2.salary)
    FROM employees e2
    WHERE e1.department_id = e2.department_id
);
```

- **Explanation**: The subquery calculates the average salary for each department. It refers to the `department_id` of the current row (`e1.department_id`) in the outer query. The subquery executes once for every row in the outer query.

2. Create a table EMPLOYEE with following schema: (Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name, Job_id, Salary)
   Write SQL queries for following question:
   a. Insert a least 5 rows in the table.
   b. Display all the information of EMP table.
   c. Update the city of Emp_no-12 with current city as Nagpur.
   d. List out different SQL commands? Write their basic structure

a. Create the EMPLOYEE Table and Insert Rows

```
CREATE TABLE EMPLOYEE (
Emp_no INT PRIMARY KEY,
E_name VARCHAR(50),
E_address VARCHAR(100),
E_ph_no VARCHAR(15),
Dept_no INT,
Dept_name VARCHAR(50),
Job_id VARCHAR(10),
Salary DECIMAL(10, 2)
);
```

Insert at least 5 rows into the EMPLOYEE table:

```
INSERT INTO EMPLOYEE (Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name,
Job_id, Salary)

VALUES

(10, 'Alice', 'Mumbai', '1234567890', 101, 'HR', 'HR01', 50000),

(11, 'Bob', 'Delhi', '9876543210', 102, 'Finance', 'FN02', 60000),

(12, 'Charlie', 'Nagpur', '7890123456', 103, 'IT', 'IT03', 70000),

(13, 'Diana', 'Pune', '8901234567', 101, 'HR', 'HR04', 55000),

(14, 'Ethan', 'Chennai', '9012345678', 104, 'Operations', 'OP05', 65000);
```

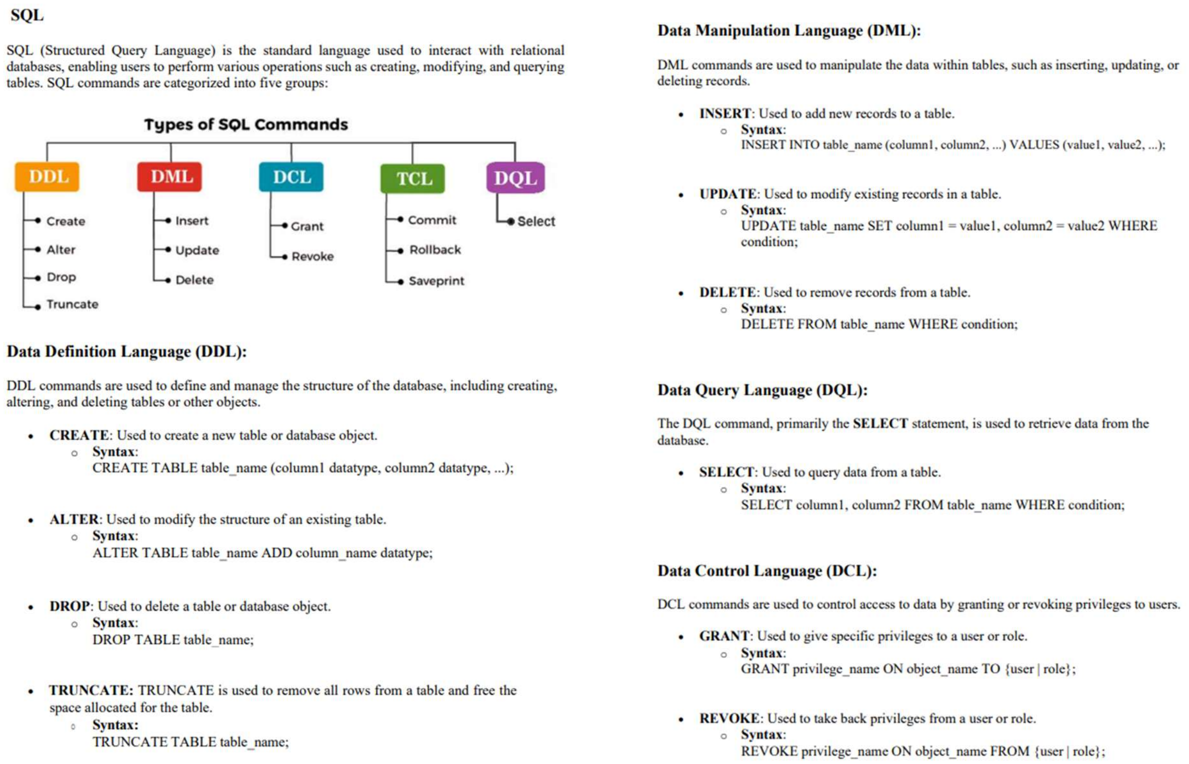
b. Display All Information of the EMPLOYEE Table

```
SELECT * FROM EMPLOYEE;
```

c. Update the City of `Emp_no` 12

```
UPDATE EMPLOYEE
SET E_address = 'Nagpur'
WHERE Emp_no = 12;
```

d. List of Different SQL Commands and Their Structures



SQL

SQL (Structured Query Language) is the standard language used to interact with relational databases, enabling users to perform various operations such as creating, modifying, and querying tables. SQL commands are categorized into five groups:

Types of SQL Commands
DDL
Create
Alter
Drop
Truncate
DML
Insert
Update
Delete
DCL
Grant
Revoke
TCL
Commit
Rollback
Saveprint
DQL
Select

Data Definition Language (DDL):

DDL commands are used to define and manage the structure of the database, including creating, altering, and deleting tables or other objects.

- **CREATE**: Used to create a new table or database object.
  - **Syntax**:
    CREATE TABLE table_name (column1 datatype, column2 datatype, ...);
- **ALTER**: Used to modify the structure of an existing table.
  - **Syntax**:
    ALTER TABLE table_name ADD column_name datatype;
- **DROP**: Used to delete a table or database object.
  - **Syntax**:
    DROP TABLE table_name;
- **TRUNCATE:** TRUNCATE is used to remove all rows from a table and free the space allocated for the table.
  - **Syntax:**
    TRUNCATE TABLE table_name;

Data Manipulation Language (DML):

DML commands are used to manipulate the data within tables, such as inserting, updating, or deleting records.

- **INSERT**: Used to add new records to a table.
  - **Syntax**:
    INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);
- **UPDATE**: Used to modify existing records in a table.
  - **Syntax**:
    UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;
- **DELETE**: Used to remove records from a table.
  - **Syntax**:
    DELETE FROM table_name WHERE condition;

Data Query Language (DQL):

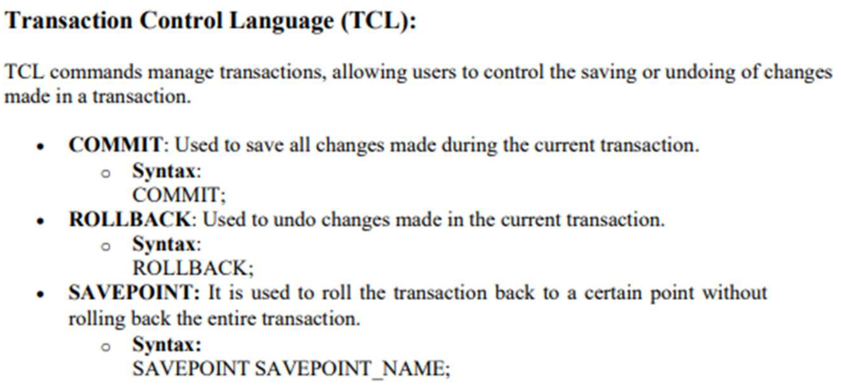
The DQL command, primarily the **SELECT** statement, is used to retrieve data from the database.

- **SELECT**: Used to query data from a table.
  - **Syntax**:
    SELECT column1, column2 FROM table_name WHERE condition;

Data Control Language (DCL):

DCL commands are used to control access to data by granting or revoking privileges to users.

- **GRANT**: Used to give specific privileges to a user or role.
  - **Syntax**:
    GRANT privilege_name ON object_name TO {user | role};
- **REVOKE**: Used to take back privileges from a user or role.
  - **Syntax**:
    REVOKE privilege_name ON object_name FROM {user | role};

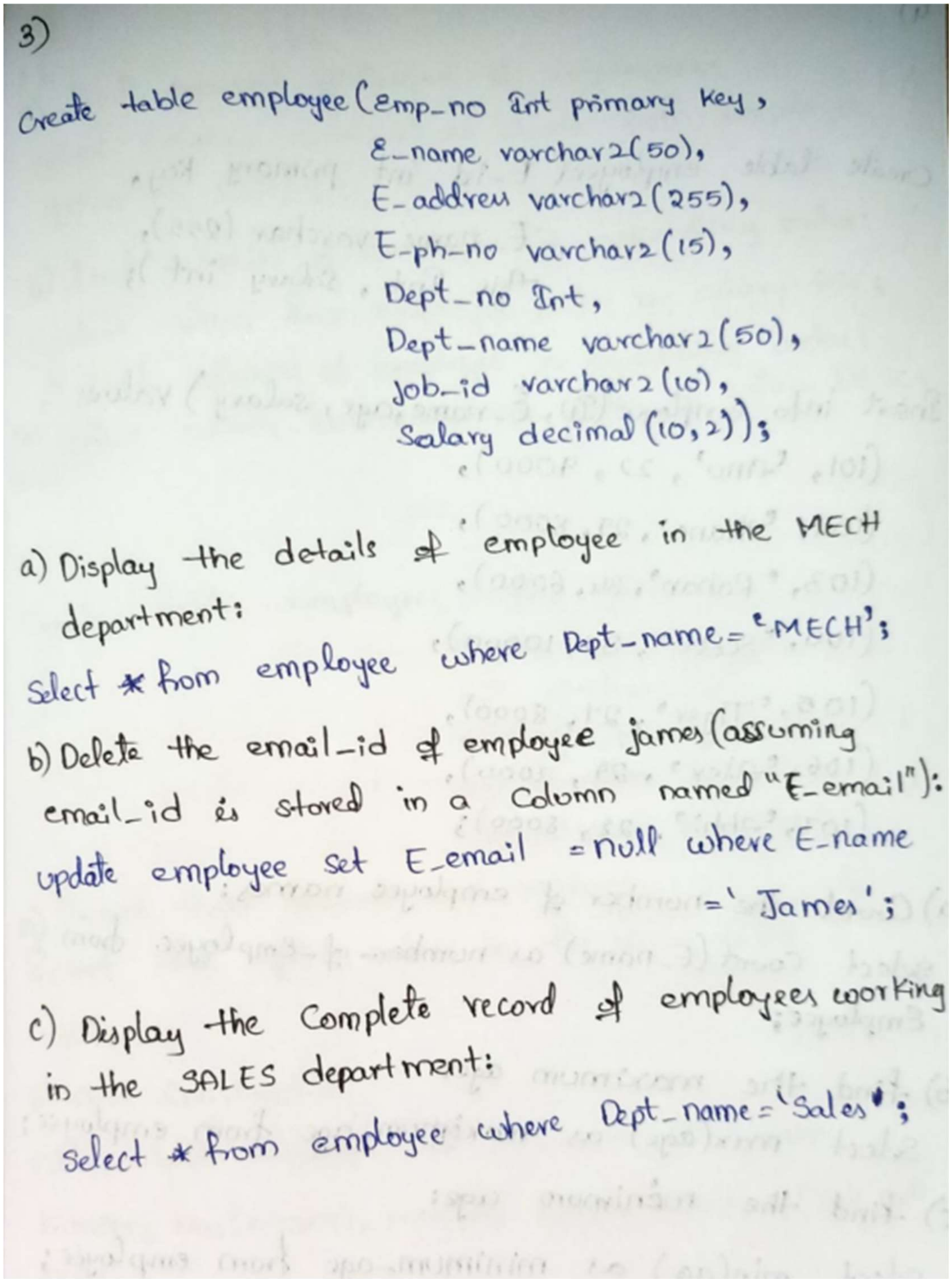


Transaction Control Language (TCL):

TCL commands manage transactions, allowing users to control the saving or undoing of changes made in a transaction.

- **COMMIT**: Used to save all changes made during the current transaction.
  - **Syntax**:
    COMMIT;
- **ROLLBACK**: Used to undo changes made in the current transaction.
  - **Syntax**:
    ROLLBACK;
- **SAVEPOINT:** It is used to roll the transaction back to a certain point without rolling back the entire transaction.
  - **Syntax:**
    SAVEPOINT SAVEPOINT_NAME;

3. Create a table EMPLOYEE with following schema: (Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name, Job_id, Salary)
Write SQL queries for following question:
a. Display the details of Employee who works in department MECH.
b. Delete the email_id of employee James.
c. Display the complete record of employees working in SALES Department.
d.State the difference between primay key,unique key,not null with an example



3)

```
Create table employee(emp_no int primary key,
                      E_name varchar2(50),
                      E_address varchar2(255),
                      E_ph_no varchar2(15),
                      Dept_no int,
                      Dept_name varchar2(50),
                      job_id varchar2(10),
                      Salary decimal (10,2));
```

a) Display the details of employee in the MECH department:

```
Select * from employee where Dept_name='MECH';
```

b) Delete the email_id of employee james (assuming email_id is stored in a column named "E_email"):

```
update employee set E_email = null where E_name
                                  = 'James';
```

c) Display the Complete record of employees working in the SALES department:

```
Select * from employee where Dept_name='Sales';
```

d.) Differences Between Primary Key, Unique Key, and Not Null

Constraint	Primary Key	Unique Key	Not Null
Definition	Ensures that a column (or combination of columns) uniquely identifies each row in the table.	Ensures that the values in a column or combination of columns are unique across all rows.	Ensures that a column cannot have `NULL` values.
Uniqueness	Enforces uniqueness and implicitly ensures no `NULL` values are allowed.	Enforces uniqueness but allows `NULL` values.	Does not enforce uniqueness; only ensures that `NULL` values are not permitted.
Number of Constraints per Table	Only one primary key is allowed per table.	Multiple unique keys can be defined in a table.	Can be applied to multiple columns.
Index Creation	Automatically creates a clustered index (if supported by the database).	Creates a non-clustered index for the column(s).	No automatic index is created.
Use Case	Used for uniquely identifying a row (e.g., `Emp_no`).	Used when a column needs unique values but isn't part of the primary identifier (e.g., `E_ph_no`).	Used when a column must always have a valid value (e.g., `E_name`).

Examples

1. Primary Key

- A table can have only one primary key, combining both uniqueness and non-null constraints.

```
CREATE TABLE EMPLOYEE (
    Emp_no INT PRIMARY KEY, -- Ensures unique and non-null employee numbers
    E_name VARCHAR(50),
    E_ph_no VARCHAR(15)
);
```

2. Unique Key

- Can be used to ensure uniqueness of values in a column that is not the primary identifier.

```
CREATE TABLE EMPLOYEE (
    Emp_no INT PRIMARY KEY,
    E_name VARCHAR(50),
    E_ph_no VARCHAR(15) UNIQUE);
```

3. Not Null

- Ensures a column always contains a value.

```
CREATE TABLE EMPLOYEE (
    Emp_no INT PRIMARY KEY,
    E_name VARCHAR(50) NOT NULL,
    E_ph_no VARCHAR(15)
);
```

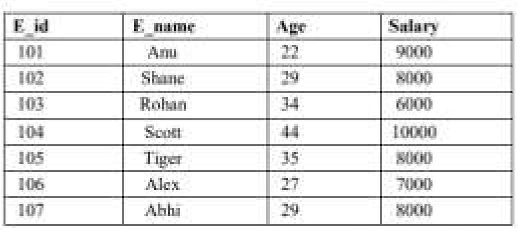
Combined Example

```
CREATE TABLE EMPLOYEE (
    Emp_no INT PRIMARY KEY,
    E_name VARCHAR(50) NOT NULL,
    E_ph_no VARCHAR(15) UNIQUE,
    Dept_no INT NOT NULL
);
```

In this example:

- `Emp_no` uniquely identifies each employee and cannot be `NULL`.
- `E_name` ensures that every employee has a name.
- `E_ph_no` ensures phone numbers are unique but can be `NULL` (e.g., employees without a phone number).

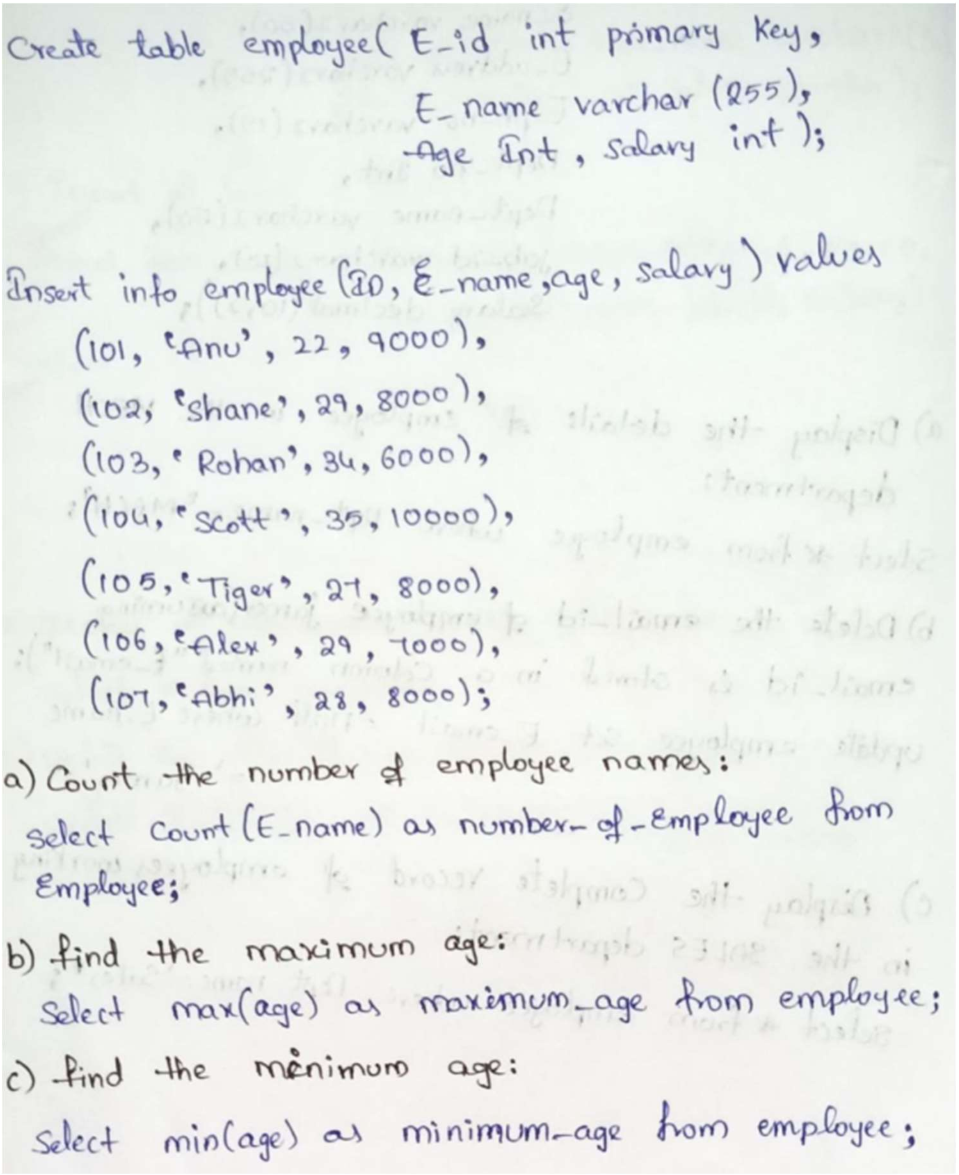
4. Create a table EMPLOYEE with following schema:



E_id	E_name	Age	Salary
101	Anu	22	9000
102	Shane	29	8000
103	Rohan	34	6000
104	Scott	44	10000
105	Tiger	35	8000
106	Alex	27	7000
107	Abhi	29	8000

Write SQL queries for following question:

a. Count number of employee names from employee table.
b. Find the Maximum age from employee table.
c. Find the Minimum age from employee table.
d. List out the types of notations used in ER-Diagram



```
Create table employee( E-id int primary key,
                       E_name varchar (255),
                       Age Int , Salary int );

Insert info employee (ID, E-name ,age, salary ) values
    (101, 'Anu', 22, 9000),
    (102; 'shane', 29, 8000),
    (103, ' Rohan', 34, 6000),
    (104, 'Scott', 35, 10000),
    (105, 'Tiger', 27, 8000),
    (106, 'Alex', 29, 7000),
    (107, 'Abhi', 28, 8000);
```

a) Count the number of employee names:

```
Select Count (E_name) as number-of-Employee from
Employee;
```

b) find the maximum age:

```
Select max(age) as maximum_age from employee;
```

c) find the minimum age:

```
Select min(age) as minimum-age from employee;
```

d.) Types of Notations Used in ER Diagrams

Entity-Relationship (ER) diagrams use specific notations to visually represent the relationships between entities in a database. The common notations are:

---

1. Entity Notations

- **Entity**: Represents a real-world object or concept with attributes.
    - **Notation**: A rectangle.
    - **Types**:
        - **Strong Entity**: Represented by a single rectangle.
        - **Weak Entity**: Represented by a double rectangle, dependent on a strong entity.

---

2. Attribute Notations

- **Attributes**: Describe the properties of an entity or relationship.
    - **Notation**: Oval shapes.
    - **Types**:
        - **Simple Attribute**: Represented by a single oval (e.g., Name).
        - **Composite Attribute**: Represented by ovals connected to sub-attributes (e.g., Full Name divided into First Name, Last Name).
        - **Derived Attribute**: Represented by a dashed oval (e.g., Age derived from Date of Birth).
        - **Multivalued Attribute**: Represented by a double oval (e.g., Phone Numbers).

---

3. Relationship Notations

- **Relationship**: Represents associations between entities.
    - **Notation**: A diamond shape.
    - **Types**:
        - **1:1 (One-to-One)**: A single instance of one entity is related to a single instance of another.
        - **1:N (One-to-Many)**: A single instance of one entity is related to multiple instances of another.
        - **M:N (Many-to-Many)**: Multiple instances of one entity are related to multiple instances of another.

---

4. Key Notations

- **Primary Key**: Represents a unique identifier for an entity.
    - **Notation**: Underlined attribute name.
- **Foreign Key**: Represents an attribute that is a primary key in another entity.
    - **Notation**: Connected by a relationship line to the parent entity.

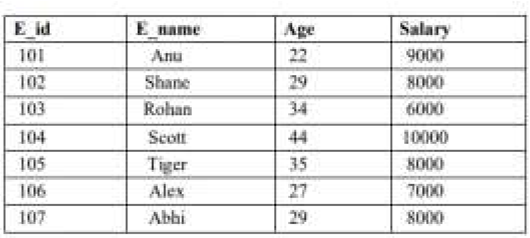
---

5. Weak Entity and Identifying Relationship

- **Weak Entity**: Cannot exist independently without a related strong entity.
    - **Notation**: Double rectangle.
- **Identifying Relationship**: Links a weak entity to its strong entity.
    - **Notation**: Double diamond.

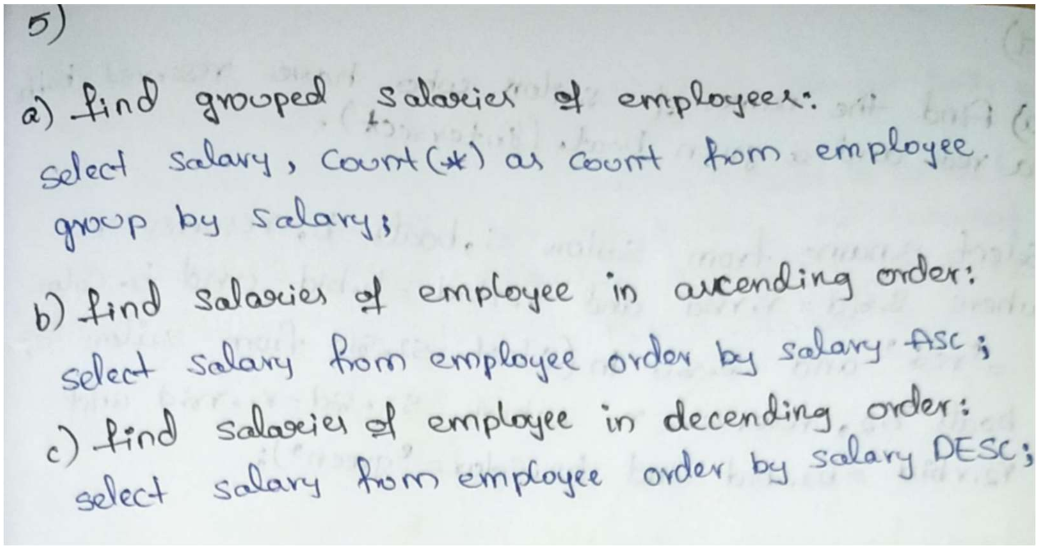
---

5. Create a table EMPLOYEE with following schema:



E_id	E_name	Age	Salary
101	Anu	22	9000
102	Shane	29	8000
103	Rohan	34	6000
104	Scott	44	10000
105	Tiger	35	8000
106	Alex	27	7000
107	Abhi	29	8000

a. Find grouped salaries of employees. (group by clause)
b. Find salaries of employee in Ascending Order. (order by clause)
c. Find salaries of employee in Descending Order.
d. Explain the following terms with example i)Entity ii)Attribute iii)Relationship iv)weak entity v)Strong Entity.



5)

a) find grouped salaries of employees:

```
select salary, Count(*) as count from employee
group by salary;
```

b) find salaries of employee in acending order:

```
select salary from employee order by salary ASC;
```

c) find salaries of employee in decending order:

```
select salary from employee order by salary DESC;
```

d.)i) Entity

An entity is a real-world object or concept that can be identified in a database.

- **Example**: A "Student" in a university database.
- **Notation**: Represented by a rectangle in an ER diagram.

ii) Attribute

An attribute is a property or characteristic of an entity that provides more information about it.

- **Example**: For a "Student" entity, attributes could be `Name`, `Roll No`, and `Date of Birth`.
- **Notation**: Represented by ovals in an ER diagram.

iii) Relationship

A relationship represents an association between two or more entities.

- **Example**: A "Student" *enrolls* in a "Course" (relationship: "Enrolls").
- **Notation**: Represented by a diamond shape in an ER diagram.

iv) Weak Entity

A weak entity cannot exist without being associated with a strong entity. It depends on the strong entity for its identification.

- **Example**: A "Dependent" entity in an insurance database depends on the "Employee" entity.
- **Notation**: Represented by a double rectangle in an ER diagram.

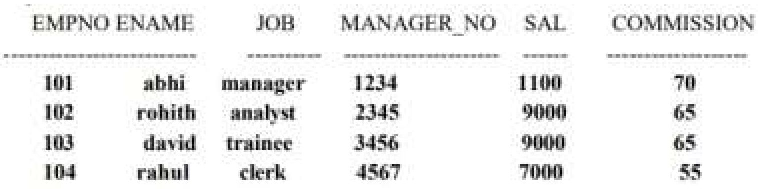
---

v) Strong Entity

A strong entity can exist independently and is not dependent on any other entity for its identification.

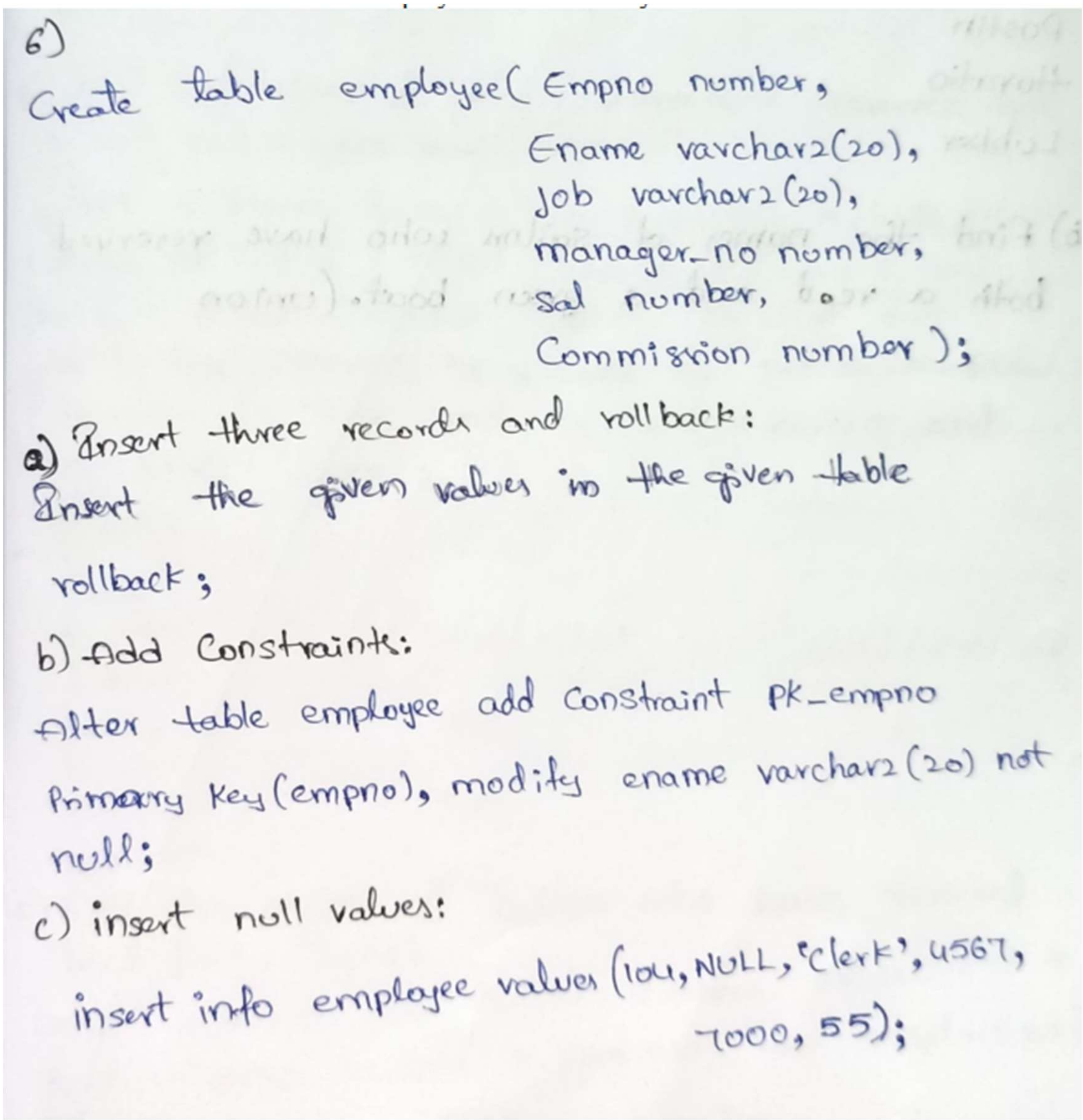
- **Example**: An "Employee" in a company database.
- **Notation**: Represented by a single rectangle in an ER diagram.

6. Create a table EMPLOYEE with following schema:



EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
101	abhi	manager	1234	1100	70
102	rohith	analyst	2345	9000	65
103	david	trainee	3456	9000	65
104	rahul	clerk	4567	7000	55

a. Insert the any three records in the employee table. Check the result.
b. Add primary key constraint and not null constraint to the employee table.
c. Insert null values to the employee table and verify the result.
d. Explain the following terms with examples i) Derived attribute ii) Composite attribute iii)Strong Entity



6)

```
Create table employee( Empno number,
                       Ename varchar2(20),
                       job varchar2(20),
                       manager_no number,
                       sal number,
                       Commission number );
```

a) Insert three records and rollback:

Insert the given values in the given table

```
rollback;
```

b) Add Constraints:

```
Alter table employee add constraint pk_empno
Primary key(empno), modify ename varchar2(20) not
null;
```

c) insert null values:

```
insert info employee values (104, NULL, 'clerk', 4567,
                             7000, 55);
```

d.) i) Derived Attribute

A derived attribute is an attribute whose value can be calculated or derived from other attributes in the database. It does not need to be stored separately.

- **Example**:
  - `Age` can be derived from the `Date of Birth`.
  - If `Date of Birth` is stored as `1990-01-01`, the current `Age` is calculated based on today's date.
- **Notation**: Represented by a dashed oval in an ER diagram.

---

ii) Composite Attribute

A composite attribute is an attribute that can be divided into smaller sub-parts, each representing more detailed information.

- **Example**:
  - `Full Name` can be divided into `First Name` and `Last Name`.
  - Similarly, `Address` can be divided into `Street`, `City`, `State`, and `Zip Code`.
- **Notation**: Represented by an oval with lines connecting to its sub-attributes in an ER diagram.

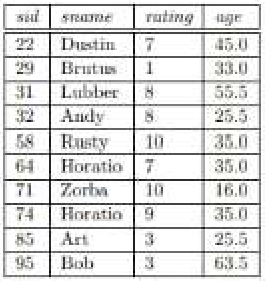
---

iii) Strong Entity

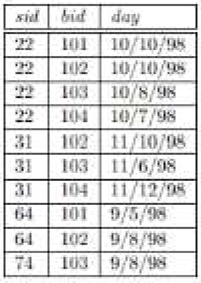
A strong entity is an entity that can exist independently and does not depend on any other entity for its identification.

- **Example**:
  - An "Employee" entity with attributes like `Emp_no`, `Name`, and `Department`. Each employee can be uniquely identified by the `Emp_no` without requiring any other entity.
- **Notation**: Represented by a single rectangle in an ER diagram.

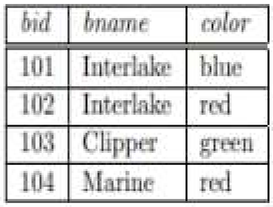
7. Create a table sailor, reserves, boats:



sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5



sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98



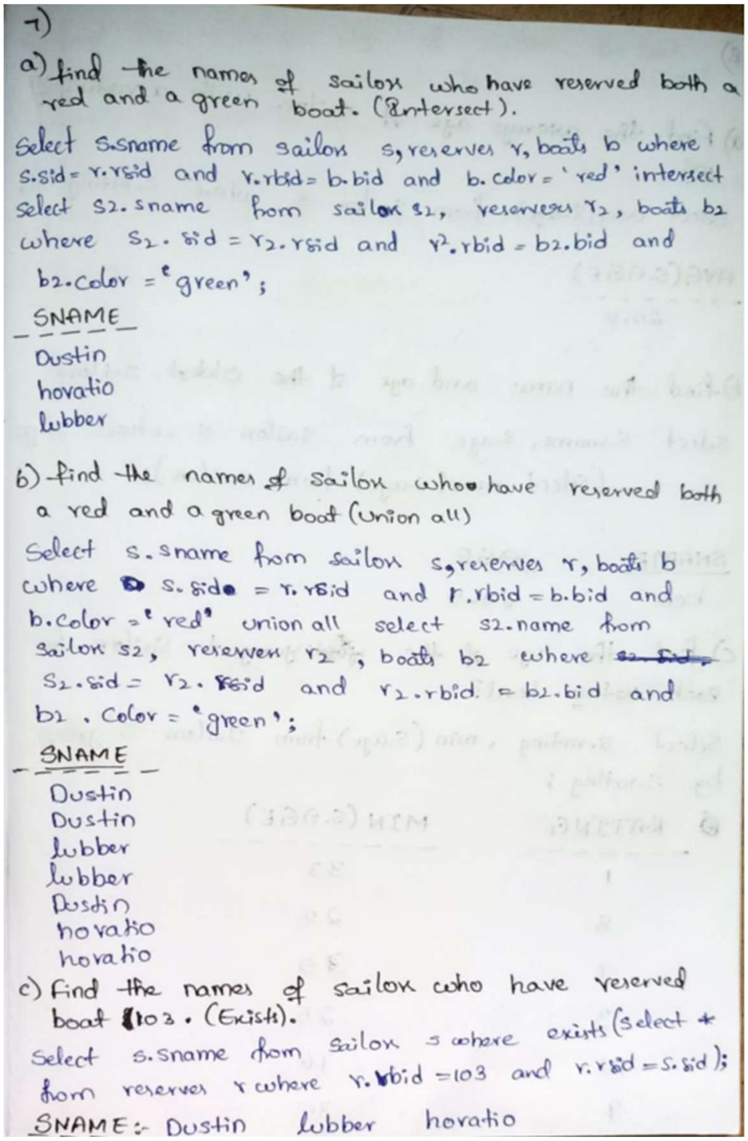
bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

a. Find the names of sailors who have reserved both a red and a green boat. (Intersect)

b. Find the names of sailors who have reserved both a red and a green boat. (union all)

c. Find the names of sailors who have reserved boat 103. (Exists)

d. Explain about different Aggregate functions with an example?



7)

a) find the names of sailors who have reserved both a red and a green boat. (Intersect).

```
Select s.sname from sailors s, reserves r, boats b where
s.sid = r.rsid and r.rbid = b.bid and b.color = 'red' intersect
select s2.sname from sailors s2, reserves r2, boats b2
where s2.sid = r2.rsid and r2.rbid = b2.bid and
b2.color = 'green';
```

SNAME

Dustin
horatio
lubber

b) find the names of sailors who have reserved both a red and a green boat (union all)

```
Select s.sname from sailors s, reserves r, boats b
where s.sid = r.rsid and r.rbid = b.bid and
b.color = 'red' union all select s2.name from
sailors s2, reserves r2, boats b2 where ~~s2.sid~~
s2.sid = r2.rsid and r2.rbid = b2.bid and
b2 . color = 'green';
```

SNAME

Dustin
Dustin
lubber
lubber
Dustin
horatio
horatio

c) Find the names of sailors who have reserved boat 103. (Exists).

```
Select s.sname from Sailors s where exists (select *
from reserves r where r.rbid = 103 and r.rsid = s.sid);
```

SNAME :- Dustin lubber horatio

Aggregate Functions in SQL

Aggregate functions perform calculations on a set of values and return a single summarized value. They are commonly used with the `GROUP BY` clause.

---

1. `SUM()`

Calculates the total sum of a numeric column.

- **Example**:
- `SELECT SUM(Salary) AS TotalSalary FROM EMPLOYEE;`

  Finds the total salary of all employees.

---

2. `AVG()`

Calculates the average value of a numeric column.

- **Example**:
- `SELECT AVG(Salary) AS AverageSalary FROM EMPLOYEE;`

  Finds the average salary of employees.

---

3. `COUNT()`

Counts the number of rows or non-NULL values in a column.

- **Example**:
- `SELECT COUNT(Emp_no) AS TotalEmployees FROM EMPLOYEE;`

  Counts the total number of employees.

---

4. `MAX()`

Returns the maximum value in a column.

- **Example**:
- `SELECT MAX(Salary) AS HighestSalary FROM EMPLOYEE;`

  Finds the highest salary among employees.

5. `MIN()`

Returns the minimum value in a column.

- **Example**:
- `SELECT MIN(Salary) AS LowestSalary FROM EMPLOYEE;`

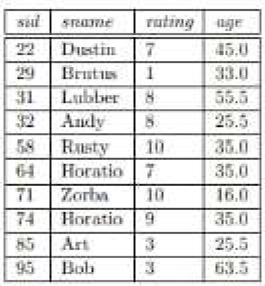
  Finds the lowest salary among employees.

Summary Query Example

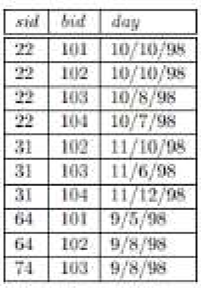
```
SELECT

    SUM(Salary) AS TotalSalary,

    AVG(Salary) AS AverageSalary,

    MAX(Salary) AS HighestSalary,

    MIN(Salary) AS LowestSalary,

    COUNT(Emp_no) AS TotalEmployees

FROM EMPLOYEE;
```

This query returns multiple aggregate results in a single output.

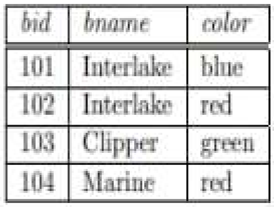
8. Create a table sailor, reserves, boats:



sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

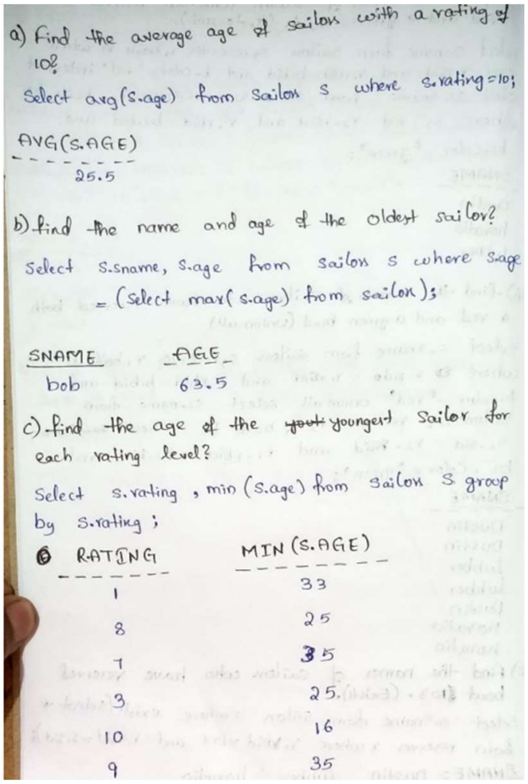


sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98



bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

a. Find the average age of sailors with a rating of 10?
b. Find the name and age of the oldest sailor?
c. Find the age of the youngest sailor for each rating level?
d. Find the average age of sailors for each rating level that has at least two sailors? (group by and Having)
e. What is Normalization? Explain about 1NF and 2NF with an example?



a) Find the average age of sailors with a rating of 10?

```
Select avg(s.age) from sailors s where s.rating=10;
```

```
AVG(S.AGE)
----------
   25.5
```

b) find the name and age of the oldest sailor?

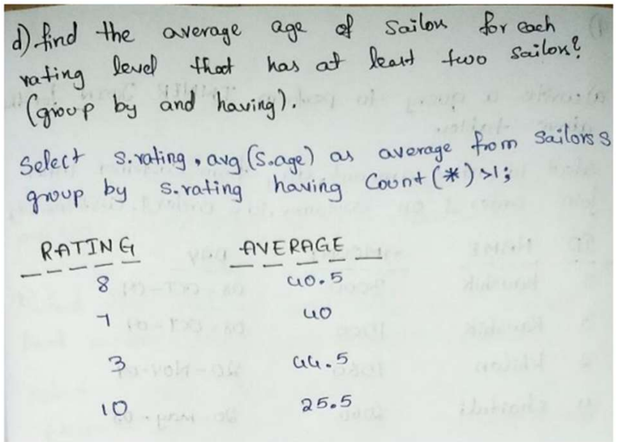
```
Select s.sname, s.age from sailors s where s.age
    = (Select max(s.age) from sailors);
```

```
SNAME     AGE
bob       63.5
```

c) find the age of the ~~yout~~ youngest Sailor for each rating level?

```
Select s.rating , min (s.age) from sailors S group
by s.rating ;
```

```
RATING          MIN(S.AGE)
--------        ----------
   1               33
   8               25
   7               35
   3               25.
   10              16
   9               35
```



d) find the average age of Sailors for each rating level that has at least two sailors? (group by and having).

```
Select s.rating, avg (s.age) as average from Sailors S
group by s.rating having Count(*)>1;
```

```
RATING          AVERAGE
--------        --------
   8              40.5
   7              40
   3              44.5
   10             25.5
```

Normalization

Normalization is the process of organizing a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller ones and defining relationships between them, ensuring data is stored efficiently and consistently.

---

1. First Normal Form (1NF)

A table is in 1NF if:

1. Each column contains atomic (indivisible) values.
2. Each row is unique, with no duplicate rows.

- **Example (Before 1NF)**:

Emp_ID	Emp_Name	Phone_Numbers
101	Alice	123456, 789012
102	Bob	345678, 901234

- *The `Phone_Numbers` column contains multiple values, which violates 1NF.*
- **After 1NF**:

Emp_ID	Emp_Name	Phone_Number
101	Alice	123456
101	Alice	789012
102	Bob	345678
102	Bob	901234

---

2. Second Normal Form (2NF)

A table is in 2NF if:

1. It is in 1NF.
2. All non-key attributes are fully functionally dependent on the primary key (no partial dependency).

- **Example (Before 2NF)**:

Emp_ID	**Dept_ID**	**Emp_Name**	**Dept_Name**
101	D01	Alice	HR
102	D02	Bob	IT

- *Here,* `Dept_Name` *depends only on* `Dept_ID`*, not the whole primary key (*`Emp_ID, Dept_ID`*).*
- **After 2NF**:
  Employee Table

Emp_ID	**Emp_Name**	**Dept_ID**
101	Alice	D01
102	Bob	D02

- **Department Table**

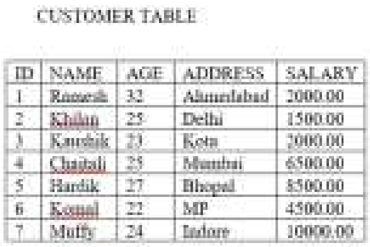
Dept_ID	**Dept_Name**
D01	HR
D02	IT

By splitting the table, `Dept_Name` is now dependent on `Dept_ID` in a separate table, eliminating partial dependency.

Benefits of Normalization

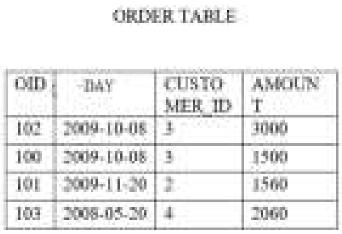
- Eliminates redundancy.
- Ensures data integrity.
- Improves query performance.

9. Create a table customer and order table:



CUSTOMER TABLE

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00



ORDER TABLE

OID	DAY	CUSTOMER_ID	AMOUNT
102	2009-10-08	3	3000
100	2009-10-08	3	1500
101	2009-11-20	2	1560
103	2008-05-20	4	2060

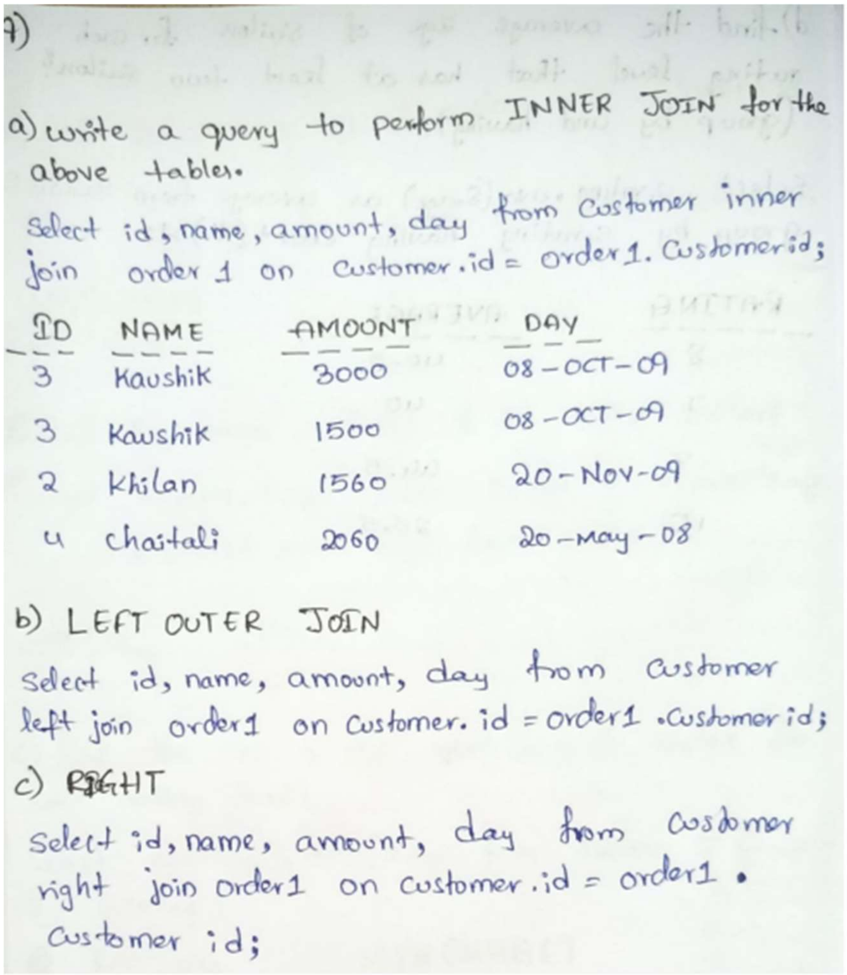
a. Write a query to perform INNER JOIN for the above tables.

b. Write a query to perform LEFT OUTER JOIN for the above tables.

c. Write a query to perform RIGHT OUTER JOIN for the above tables.

d. Write a query to perform FULL OUTER JOIN for the above tables.

e. Explain the concept of Triggers and its events with an example?



9)

a) write a query to perform INNER JOIN for the above tables.

Select id, name, amount, day from Customer inner join order 1 on Customer.id = order1.Customerid;

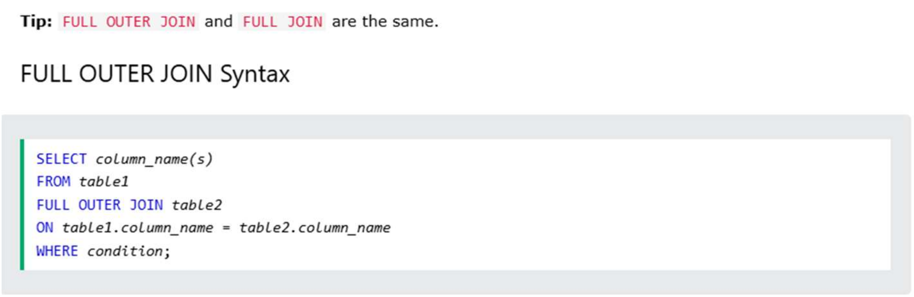
ID	NAME	AMOONT	DAY
3	Kaushik	3000	08 – OCT – 09
3	Kaushik	1500	08 – OCT – 09
2	Khilan	1560	20 – Nov – 09
4	chaitali	2060	20 – May – 08

b) LEFT OUTER JOIN

Select id, name, amount, day from Customer left join order1 on Customer.id = order1 .Customer id;

c) RIGHT

Select id, name, amount, day from Cusdomer right join Order1 on customer.id = order1 . Customer id;



Tip: FULL OUTER JOIN and FULL JOIN are the same.

FULL OUTER JOIN Syntax

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

Triggers in SQL

A **trigger** is a database object that automatically executes when a specific event occurs in a table, such as INSERT, UPDATE, or DELETE.

Types of Events

1. **INSERT**: Trigger executes when a new row is added.
2. **UPDATE**: Trigger executes when a row is updated.
3. **DELETE**: Trigger executes when a row is deleted.

Example

Track updates to the EMPLOYEE table in an AUDIT_LOG table.

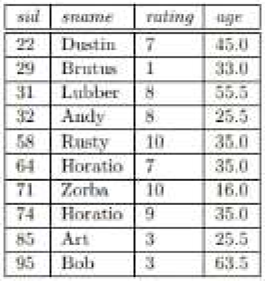
Trigger Code:

```
CREATE TRIGGER after_update_employee

AFTER UPDATE

ON EMPLOYEE

FOR EACH ROW

BEGIN

    INSERT INTO AUDIT_LOG (Emp_ID, Action, Change_Date)

    VALUES (OLD.Emp_ID, 'UPDATED', NOW());

END;
```

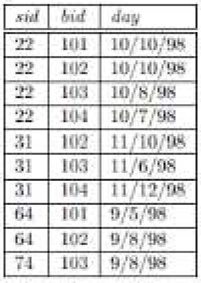
Explanation:

- Trigger activates **AFTER UPDATE** on EMPLOYEE.
- Logs the change into AUDIT_LOG with the employee ID, action (UPDATED), and current date.

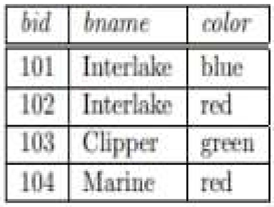
10. Create a table sailor, reserves, boats:



sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5



sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98



bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

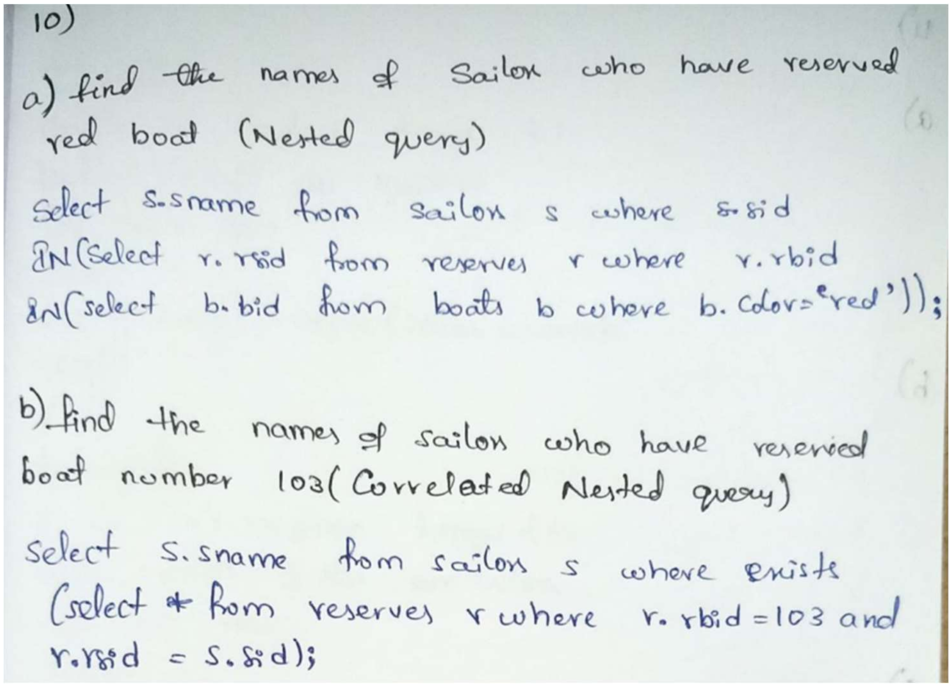
a. Find the names of sailors who have reserved red boat. (Nested Query)

b. Find the names of sailors who have reserved boat number 103. (correlated Nested Query)

c. Find the name and age of the oldest sailor?

d. Write the differences between count () and count (*) with examples?

e. Write the differences between Drop and Delete with examples?



10)

a) find the names of Sailors who have reserved red boat (Nested query)

```
Select s.sname from sailors s where s.sid
IN(Select r.rsid from reserves r where r.rbid
IN(select b.bid from boats b where b.color='red'));
```

b) find the names of sailors who have reserved boat number 103(Correlated Nested query)

```
Select S.sname from sailors s where exists
(select * from reserves r where r.rbid=103 and
r.rsid = s.sid);
```

c. in prev question

d. Differences between `COUNT()` and `COUNT(*)`

1. **`COUNT()`**:
   - Counts the number of **non-NULL** values in a specified column.
   - **Example**:
   - `SELECT COUNT(Salary) FROM EMPLOYEE;`

   This counts the number of non-NULL salary values in the `EMPLOYEE` table.

2. **`COUNT(*)`**:
   - Counts the total number of rows in a table, including rows with NULL values.
   - **Example**:
   -
     ```
     SELECT COUNT(*) FROM EMPLOYEE;
     ```

     This counts all rows in the `EMPLOYEE` table, regardless of NULL values.

Key Difference:

- `COUNT()` excludes NULL values, while `COUNT(*)` includes all rows, even those with NULL values in any column.

---

e. Differences between `DROP` and `DELETE`

1. **`DROP`**:
   - Removes a table or database completely from the database system.
   - **Example**:
   -
     ```
     DROP TABLE EMPLOYEE;
     ```

     This permanently removes the `EMPLOYEE` table from the database.

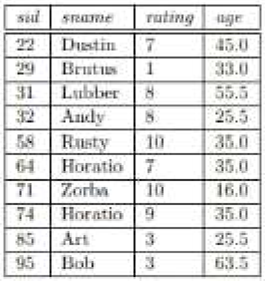
2. **`DELETE`**:
   - Removes data (rows) from a table but keeps the table structure intact.
   - **Example**:
   -
     ```
     DELETE FROM EMPLOYEE WHERE Emp_ID = 101;
     ```

     This removes the row where `Emp_ID = 101` from the `EMPLOYEE` table.

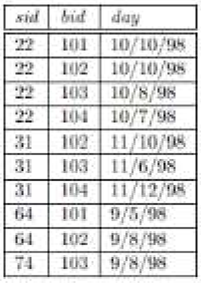
Key Differences:

- `DROP` removes the entire table (or database) and its structure, while `DELETE` removes only specific rows, leaving the table structure intact.
- `DELETE` can be rolled back if used within a transaction, while `DROP` is permanent and cannot be undone.

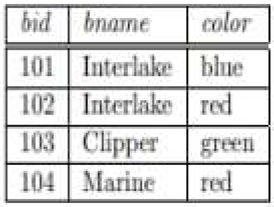
11. Create a table sailor, reserves, boats:



sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

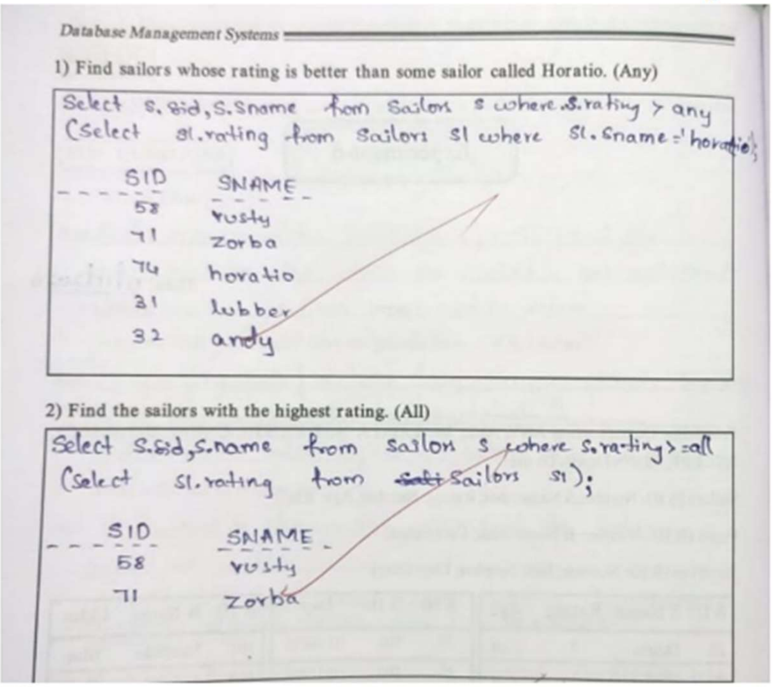


sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98



bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

a. Find sailors whose rating is better than some sailor called Horatio. (Any)
b. Find the sailors with the highest rating. (All)
c. Find the names of sailors who have reserved a red and a green boat. (Union)
d. Write about delete, truncate,drop commands with syntax?
e. Compare between primary key and unique key



Database Management Systems

1) Find sailors whose rating is better than some sailor called Horatio. (Any)

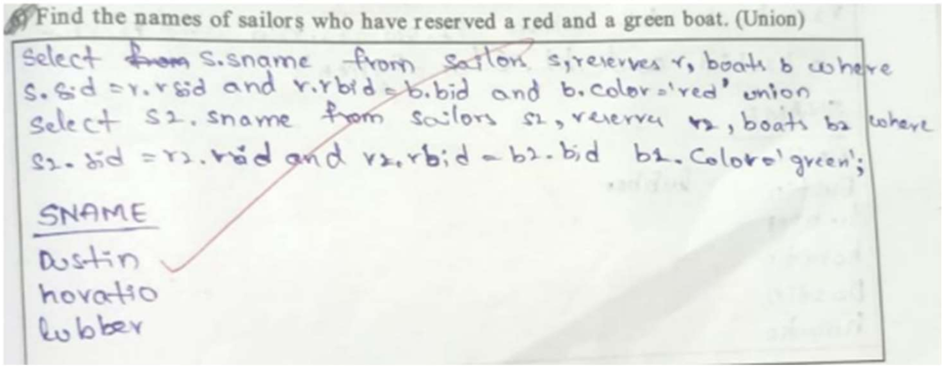
```
Select s.sid,s.sname from Sailors s where s.rating > any
(Select s1.rating from Sailors s1 where s1.sname='horatio';
```

```
    SID      SNAME
-------------------
    58       rusty
    71       Zorba
    74       horatio
    31       lubber
    32       andy
```

2) Find the sailors with the highest rating. (All)

```
Select s.sid,s.name from sailors s where s.rating>=all
(Select s1.rating from ~~salt~~ Sailors s1);
```

```
--- SID      SNAME
    58       rusty
    71       zorba
```



Find the names of sailors who have reserved a red and a green boat. (Union)

```
Select ~~from~~ s.sname from sailors s,reserves r, boats b where
s.sid =r.rsid and r.rbid=b.bid and b.color='red' union
select s2.sname from sailors s2, reserves r2, boats b2 where
s2.sid =r2.rsid and r2.rbid=b2.bid b2.Color='green';
```

```
SNAME
Dustin
horatio
lubber
```

d. DELETE, TRUNCATE, and DROP Commands

1. **`DELETE`**
   - Removes **specific rows** from a table based on a condition.
   - The table structure remains intact.
   - Can be rolled back if used inside a transaction.

   Syntax:

   ```
   DELETE FROM table_name WHERE condition;
   ```

   - **Example**:
   - `DELETE FROM EMPLOYEE WHERE Emp_ID = 101;`
2. **`TRUNCATE`**
   - Removes **all rows** from a table but keeps the table structure.
   - Cannot be rolled back in most databases.
   - Faster than `DELETE` for large tables as it does not log individual row deletions.

   Syntax:

   ```
   TRUNCATE TABLE table_name;
   ```

   - **Example**:
   - `TRUNCATE TABLE EMPLOYEE;`
3. **`DROP`**
   - Removes the entire table (or database) along with its structure and data.
   - Cannot be rolled back.

   Syntax:

   ```
   DROP TABLE table_name;
   ```

   - **Example**:
   - `DROP TABLE EMPLOYEE;`

e. Differences Between Primary Key and Unique Key

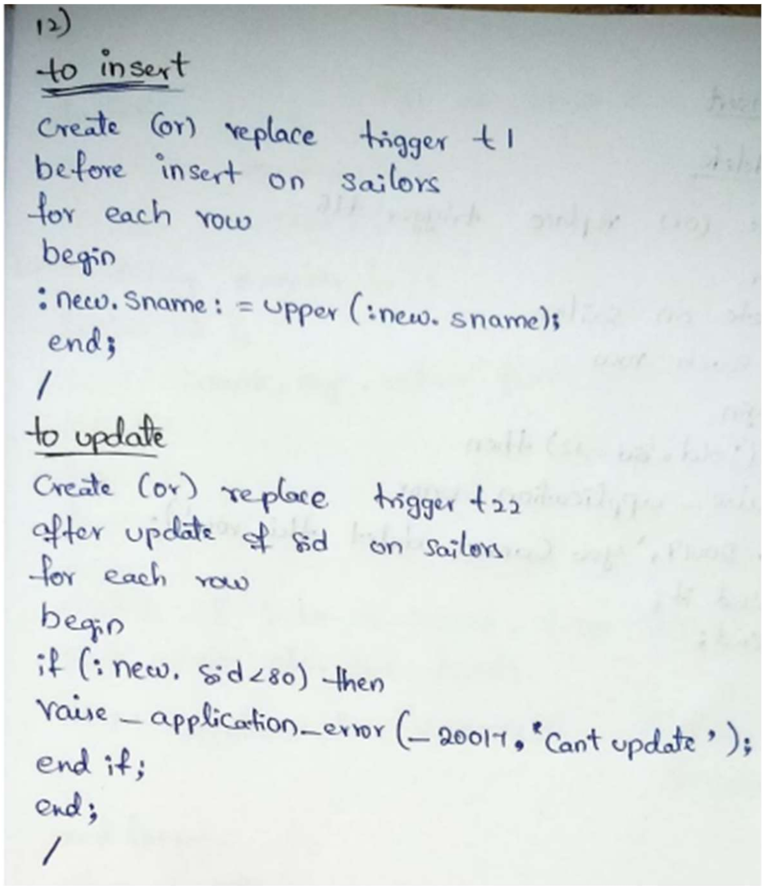
Feature	Primary Key	Unique Key
Uniqueness	Ensures that all values in the column are unique.	Ensures uniqueness but allows one NULL value.
Null Values	Cannot contain `NULL` values.	Can contain one `NULL` value.
Purpose	Uniquely identifies each record in a table.	Ensures uniqueness but is not used to identify records.
Index	Automatically creates a unique index.	Creates a unique index as well.
Number of Keys	Only one primary key can be defined per table.	Multiple unique keys can be defined in a table.
Example	`Emp_ID INT PRIMARY KEY`	`Email VARCHAR(100) UNIQUE`

Example:

```
CREATE TABLE EMPLOYEE (

  Emp_ID INT PRIMARY KEY,

  Email VARCHAR(100) UNIQUE

);
```

- `Emp_ID` is the primary key and must be unique and non-null.
- `Email` is unique, but it can allow a single `NULL` value.

12. a. Write a PL/SQL code for creation of Trigger to insert data into a table.
    b. Write a PL/SQL code for creation of trigger to update data into a table
    c. Write a PL/SQL code for creation of trigger to delete data from a table
    d. Explain the following terms with examples i) Derived attribute ii) Composite attribute iii)Strong Entity
    (d in 6th answer)



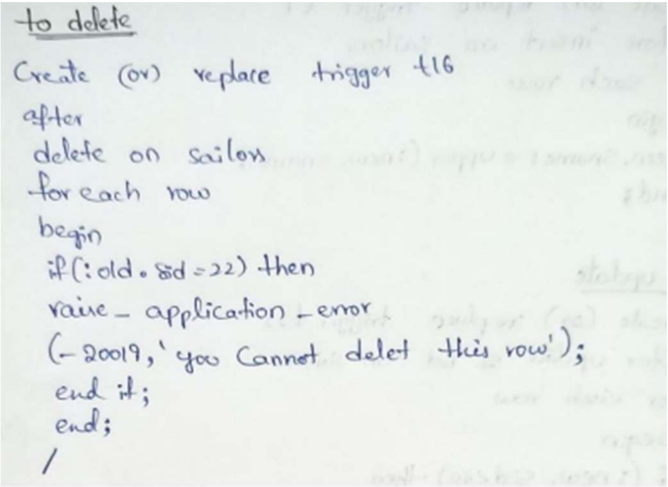
12)

to insert

```
Create (or) replace trigger t1
before insert on sailors
for each row
begin
:new.Sname : = upper (:new. sname);
end;
/
```

to update

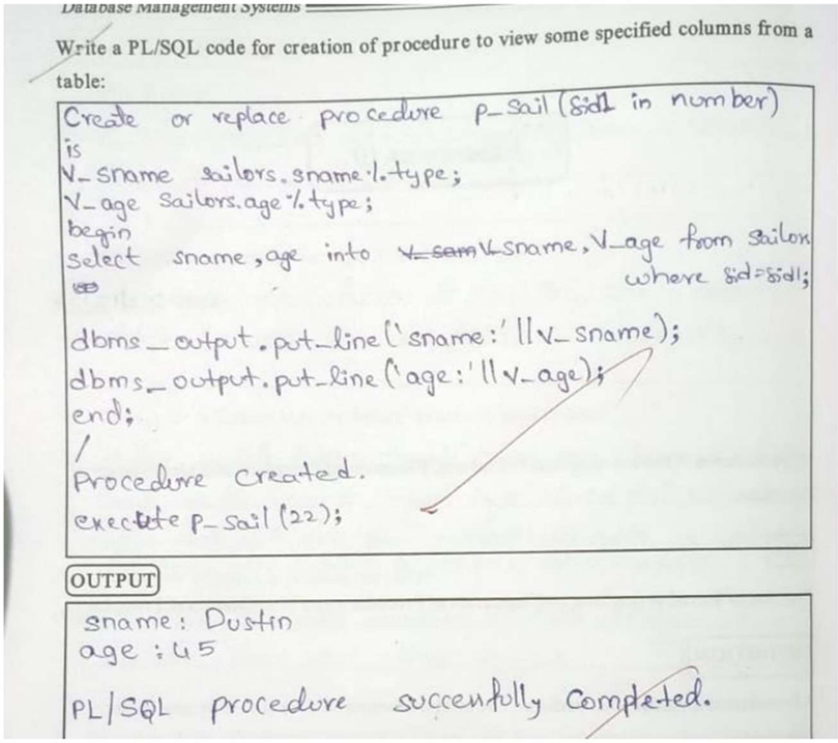
```
Create (or) replace trigger t22
after update of sid on sailors
for each row
begin
if (:new. sid<80) then
raise_application_error (-20017,'Cant update');
end if;
end;
/
```



to delete

```
Create (or) replace trigger t16
after
delete on sailors
for each row
begin
if(:old.sid=22) then
raise_ application _error
(-20019,'you Cannot delet this row');
end if;
end;
/
```

13. a. Write a PL/SQL code for creation of procedure to view some specified columns from a table.
    b. Write a PL/SQL code for modification of a procedure on specified columns from a table.
    c. Write the differences between primary key and unique key with examples?(11 e)
    d. List out the types of Notations used in ER-Diagram (4 d)



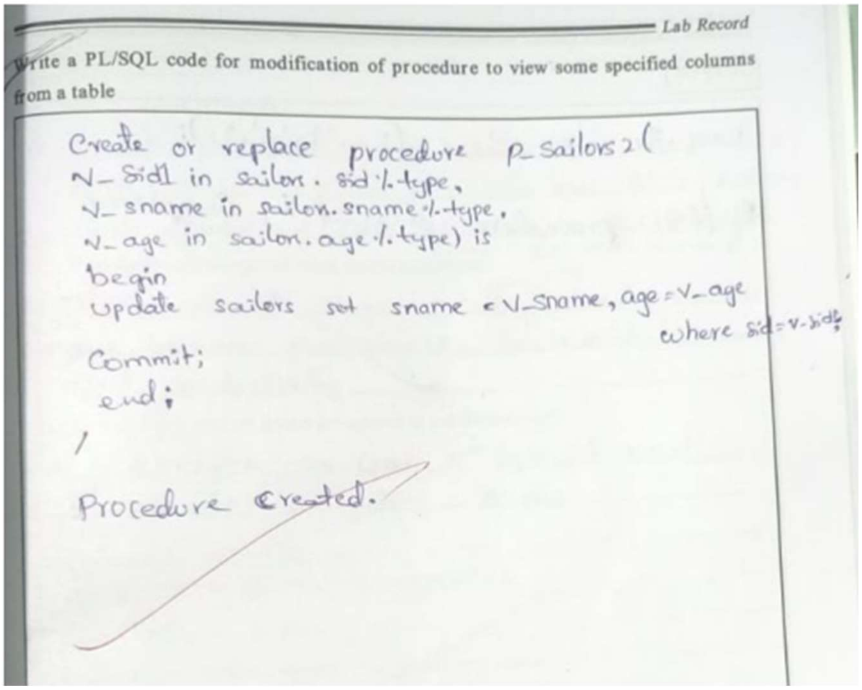
Database Management Systems

Write a PL/SQL code for creation of procedure to view some specified columns from a table:

```
Create or replace procedure p_Sail (sid1 in number)
is
V_sname sailors.sname%type;
V_age Sailors.age%type;
begin
select sname,age into ~~v_sam~~ V_sname, V_age from sailors
                                    where sid=sid1;

dbms_output.put_line('sname:'||v_sname);
dbms_output.put_line('age:'||v_age);
end;
/
Procedure created.
execute p_sail(22);
```

OUTPUT

```
sname: Dustin
age : 45

PL/SQL procedure succesfully Completed.
```

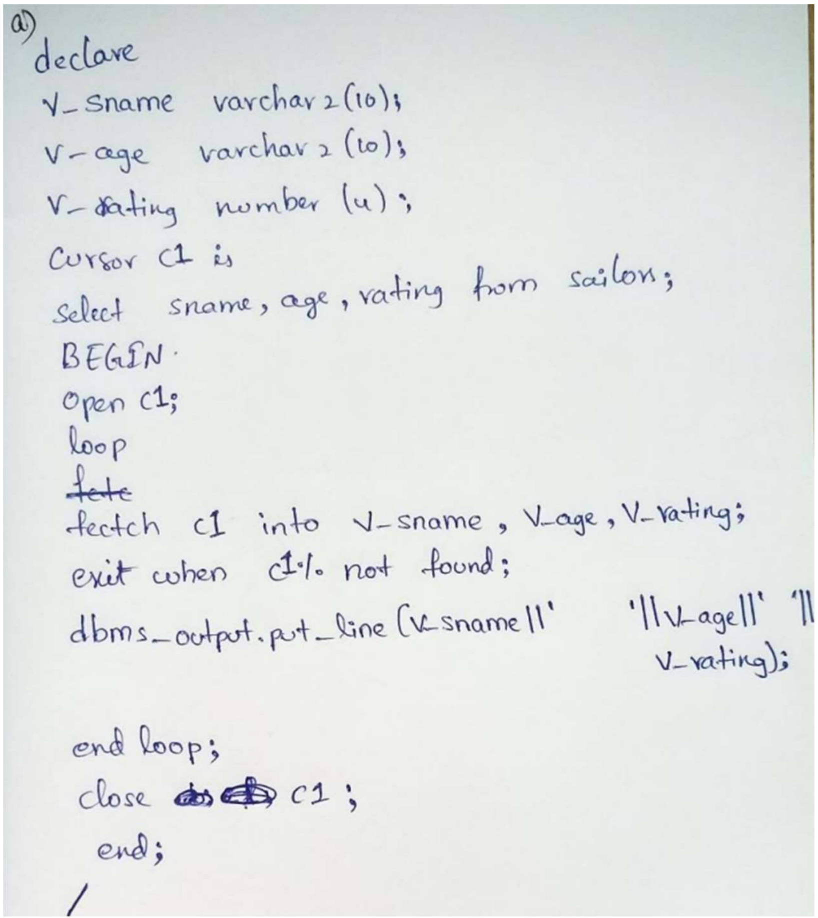


Lab Record

Write a PL/SQL code for modification of procedure to view some specified columns from a table

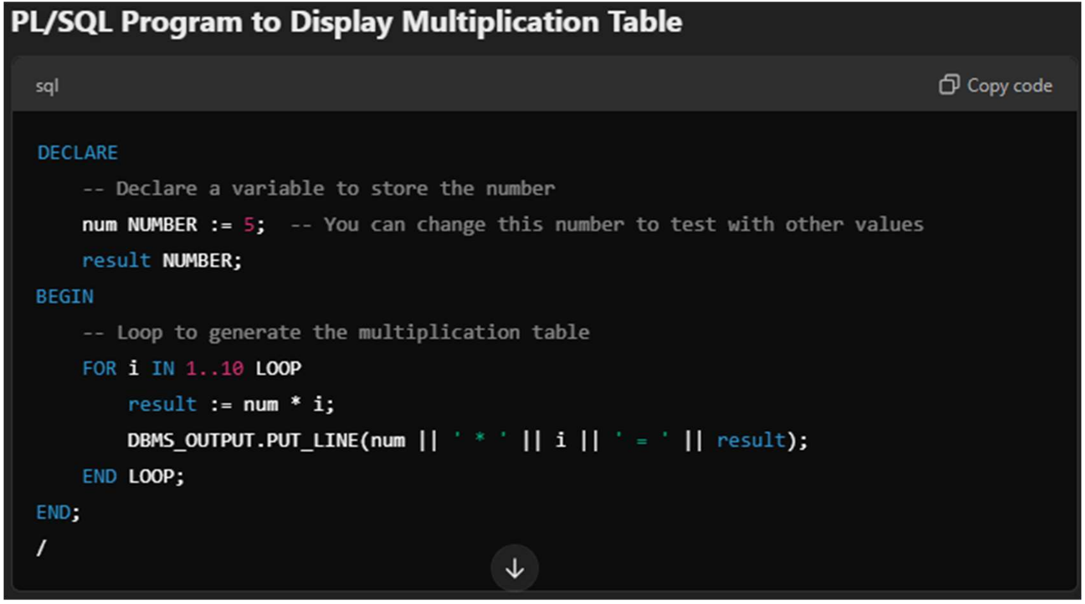
```
Create or replace procedure p_sailors2(
v_sid1 in sailors.sid%type,
v_sname in sailors.sname%type,
v_age in sailors.age%type) is
begin
update sailors set sname = v_sname, age = v_age
                                    where sid=v_sid1;
Commit;
end;
/

Procedure Created.
```

14. a.Write a PL/SQL program that uses cursor operation on any data base.
    b. Write a PL/SQL program for displaying multiplication of any number
    c. Write a PL/SQL code for creation of trigger to delete data from a table(12 c)
    d. Explain the following terms with examples i)Derived attribute ii)Composite attributeiii)Weak Entity(6 d)



a)

```
declare
V_sname   varchar2(10);
V-age   varchar2 (10);
V_rating  number (4);
Cursor C1 is
Select  sname, age, rating from  sailors;
BEGIN.
Open C1;
loop
~~fete~~
fectch c1 into V_sname , V_age, V_rating;
exit when c1% not found;
dbms_output.put_line (V_sname||'     '||V_age||' '||
                                          V_rating);

end loop;
close  c1 ;
 end;
/
```

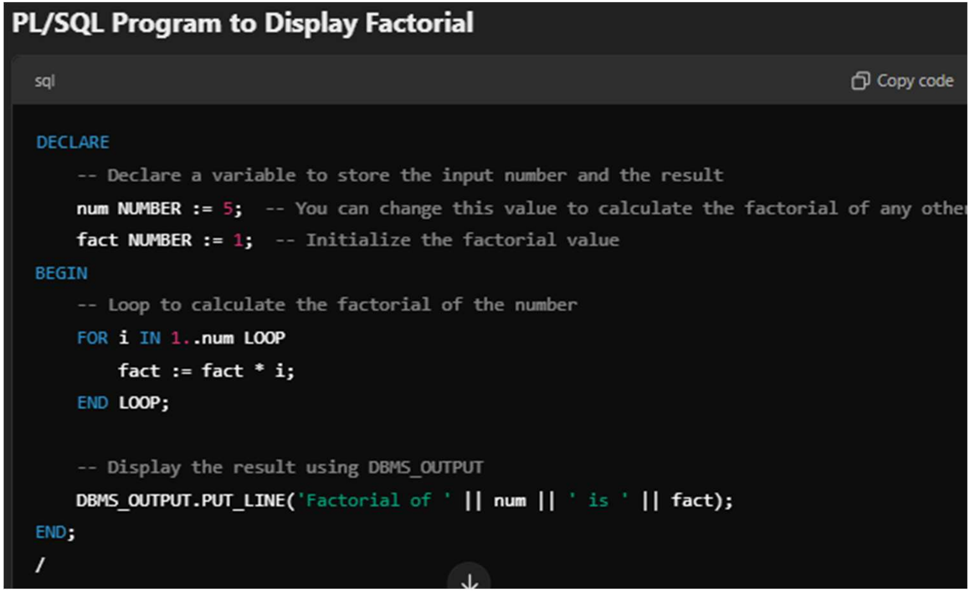


PL/SQL Program to Display Multiplication Table

```sql
DECLARE
    -- Declare a variable to store the number
    num NUMBER := 5;  -- You can change this number to test with other values
    result NUMBER;
BEGIN
    -- Loop to generate the multiplication table
    FOR i IN 1..10 LOOP
        result := num * i;
        DBMS_OUTPUT.PUT_LINE(num || ' * ' || i || ' = ' || result);
    END LOOP;
END;
/
```

15. a. Write a PL/SQL code for modification of procedure to view some specified columns from a table.(13 b)
    b. Write a PL/SQL program for displaying multiplication of any number(14 b)
    c. Write a PL/SQL code for creation of Trigger to update data into a table.(12 b)
    d. Explain the following terms with examples i) Multivalued attribute ii) Composite attribute iii)Strong Entity(6 d)

16. a) Write a PL/SQL code for creation of procedure to view some specified columns from a table.(13 a)
    b) Write a PL/SQL program for displaying factorial of any number.

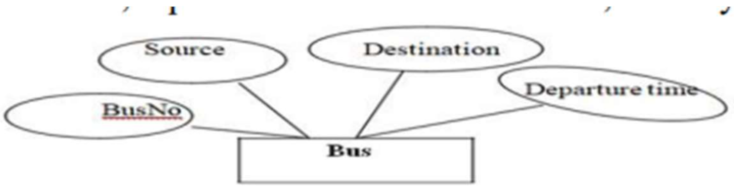


PL/SQL Program to Display Factorial

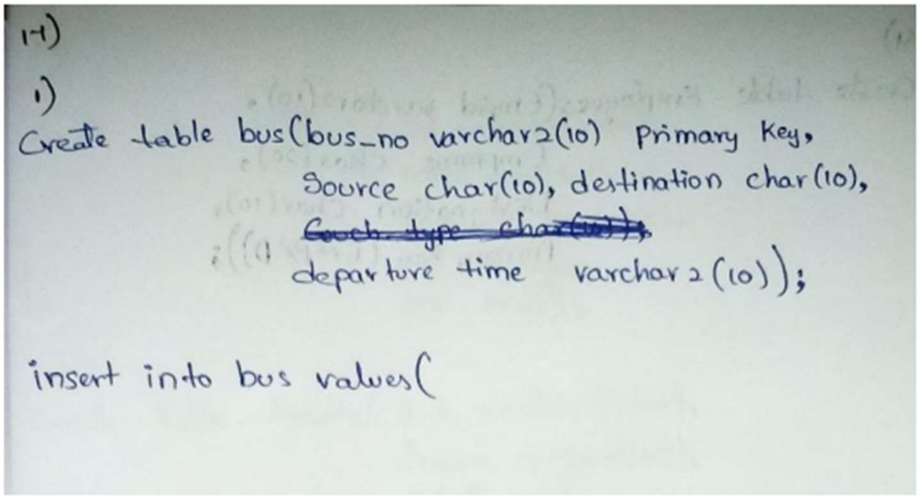
```sql
DECLARE
    -- Declare a variable to store the input number and the result
    num NUMBER := 5;  -- You can change this value to calculate the factorial of any othe
    fact NUMBER := 1;  -- Initialize the factorial value
BEGIN
    -- Loop to calculate the factorial of the number
    FOR i IN 1..num LOOP
        fact := fact * i;
    END LOOP;

    -- Display the result using DBMS_OUTPUT
    DBMS_OUTPUT.PUT_LINE('Factorial of ' || num || ' is ' || fact);
END;
/
```

    c. Write a PL/SQL code for creation of Trigger to insert data into a table.(12 c)
    d. Explain the following terms with examples i) Derived attribute ii) Composite attribute iii)Strong Entity(6d or start)

17. Converting ER Model to Relational Model (Represent entities and relationships in Tabular form, represent attributes as columns, identifying keys)



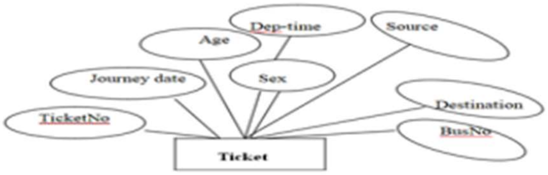
1.



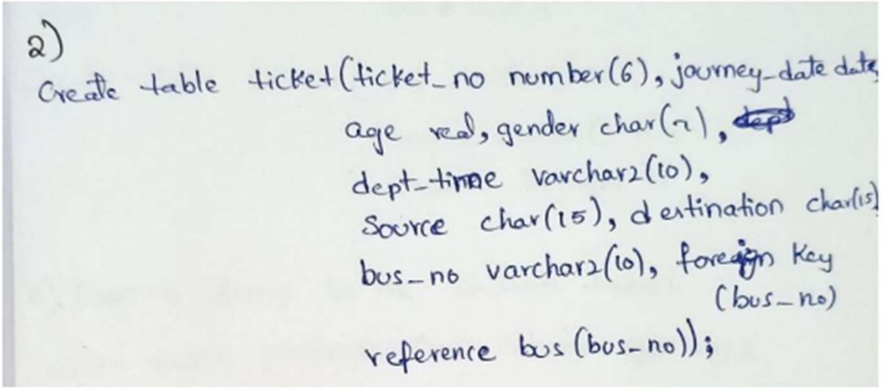
1-1)

1)

```
Create table bus(bus_no varchar2(10) Primary Key,
            Source char(10), destination char(10),
            ~~Coach type char(10)),~~
            departure time varchar2(10));

insert into bus values(
```

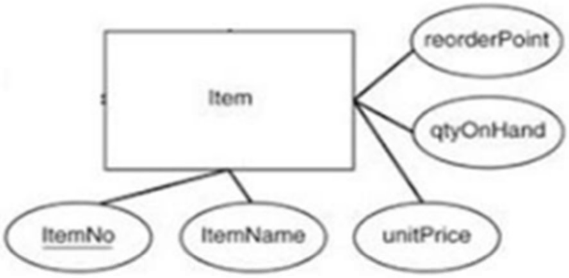


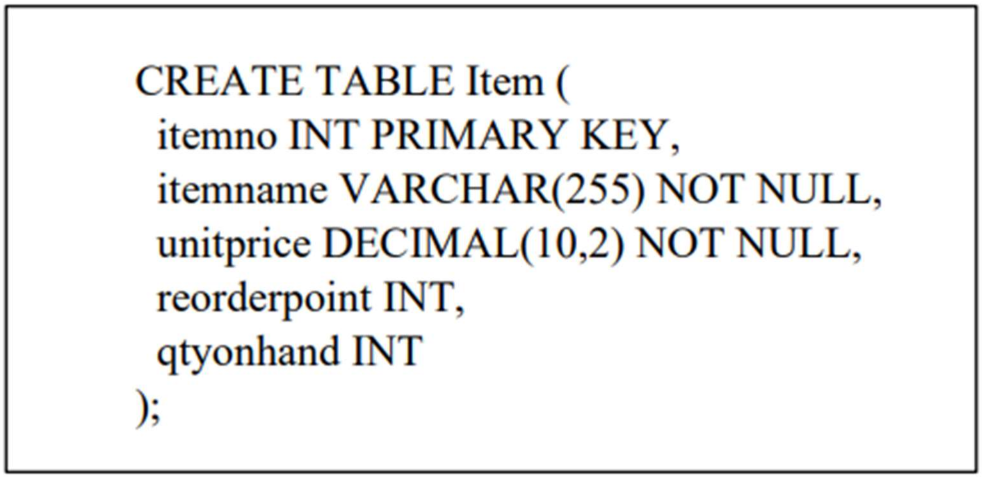
2.



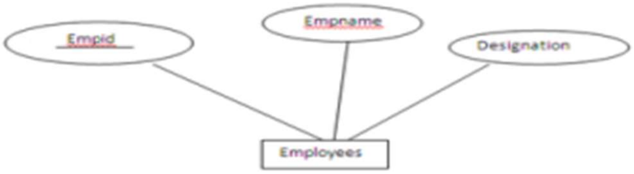
2)

```
Create table ticket(ticket_no number(6), journey_date date
            age real, gender char(7), ~~dept~~
            dept_time varchar2(10),
            Source char(15), destination char(15)
            bus_no varchar2(10), foreign Key
                                        (bus_no)
            reference bus(bus_no));
```

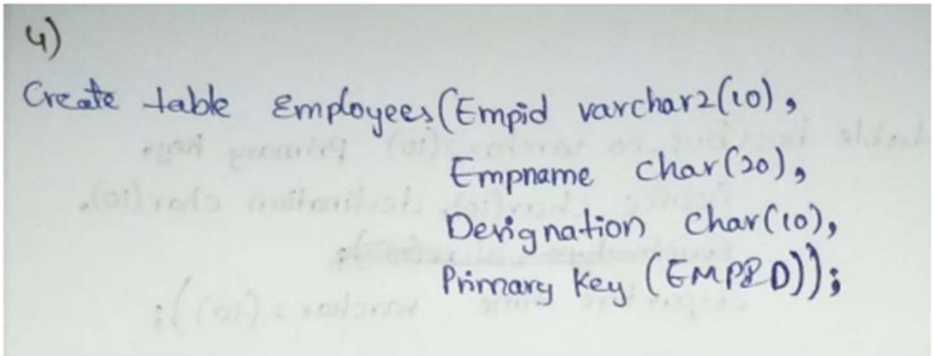




```
CREATE TABLE Item (
  itemno INT PRIMARY KEY,
  itemname VARCHAR(255) NOT NULL,
  unitprice DECIMAL(10,2) NOT NULL,
  reorderpoint INT,
  qtyonhand INT
);
```



4.



4)

```
Create table Employees(Empid varchar2(10),
                       Empname char(20),
                       Designation char(10),
                       Primary key (EMPID));
```

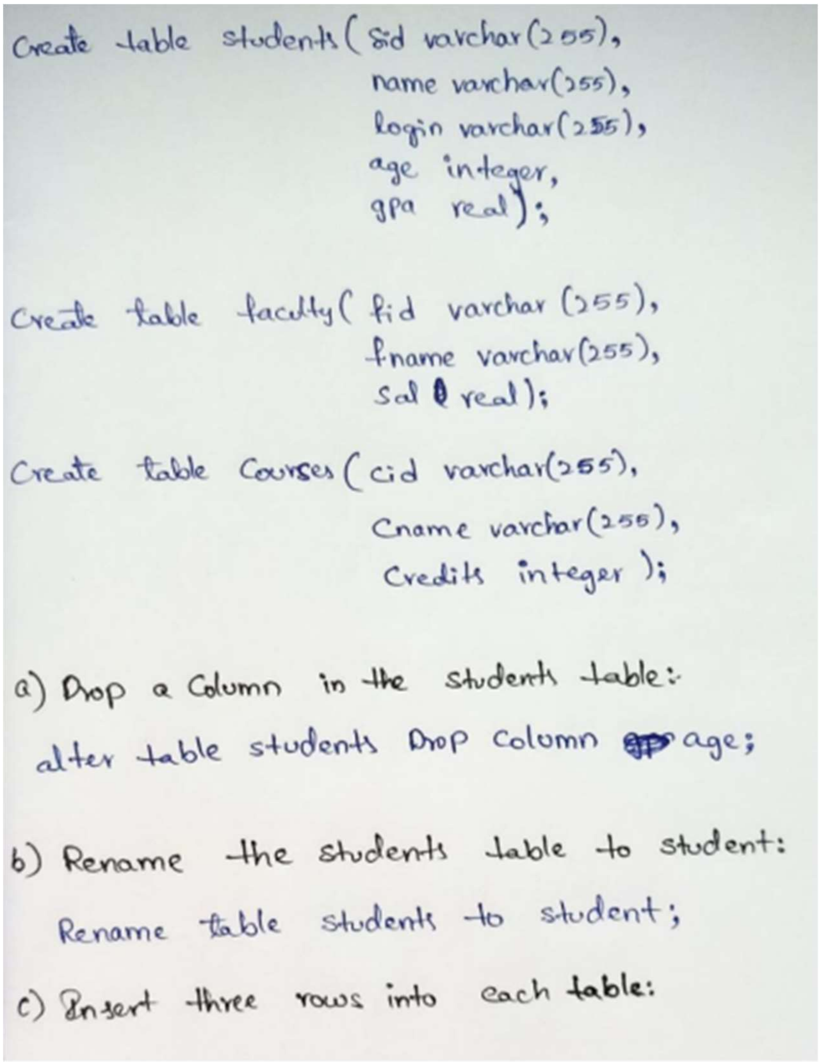
a. Write the differences between count () and count (*) with examples?(10d)

b. Write the differences between Drop and Delete with examples?(10 e)

18. Create tables for following schemas

Students(<u>*sid*</u>*: string*, *name:* string, *login:* string, *age:* integer, *gpa:* real)
Faculty(<u>*fid*</u>*:* string, *fname:* string, *sal:* real)
Courses(<u>*cid*</u>*:* string, *cname:* string, *credits:* integer)

a. write a sql query to drop a column in students table.
b. Write a query to rename table students to STUDENT
c. Write a query to insert three rows in each table.
d. Write about delete, truncate,drop commands with syntax?(11 d)
e. Differrence between primary key and unique key(3d)



```
Create table students( sid varchar(255),
                       name varchar(255),
                       login varchar(255),
                       age integer,
                       gpa real);

Create table faculty( fid varchar (255),
                      fname varchar(255),
                      sal real);

Create table Courses( cid varchar(255),
                      Cname varchar(255),
                      Credits integer );
```

a) Drop a Column in the students table:

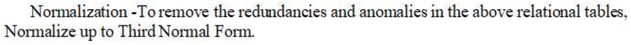
```
alter table students Drop column age;
```

b) Rename the students table to student:

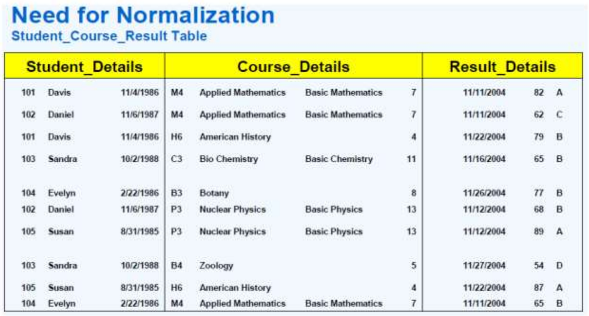
```
Rename table students to student;
```

c) Insert three rows into each table:

19.



Normalization -To remove the redundancies and anomalies in the above relational tables, Normalize up to Third Normal Form.



Need for Normalization

Student_Course_Result Table

Student_Details			Course_Details				Result_Details		
101	Davis	11/4/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	82	A
102	Daniel	11/6/1987	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	62	C
101	Davis	11/4/1986	H6	American History		4	11/22/2004	79	B
103	Sandra	10/2/1988	C3	Bio Chemistry	Basic Chemistry	11	11/16/2004	65	B
104	Evelyn	2/22/1986	B3	Botany		8	11/26/2004	77	B
102	Daniel	11/6/1987	P3	Nuclear Physics	Basic Physics	13	11/12/2004	68	B
105	Susan	8/31/1985	P3	Nuclear Physics	Basic Physics	13	11/12/2004	89	A
103	Sandra	10/2/1988	B4	Zoology		5	11/27/2004	54	D
105	Susan	8/31/1985	H6	American History		4	11/22/2004	87	A
104	Evelyn	2/22/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	65	B

1NF (First Normal Form):

- The original "Student_Course_Result" table is already in 1NF, as it has no repeating groups.

2NF (Second Normal Form):

- To achieve 2NF, we need to ensure that all non-key attributes are fully dependent on the primary key.

- The primary key in the original table is a composite key of StudentID and CourseID.

- All the attributes in the "Student_Details" and "Course_Details" sections are dependent on just the StudentID or CourseID, not the full composite key.

- Therefore, we can split the original table into three tables:

  - Student table with StudentID as the primary key

  - Course table with CourseID as the primary key

  - Result table with a composite primary key of StudentID and CourseID

3NF (Third Normal Form):

- To reach 3NF, we need to ensure there are no transitive dependencies.
- In the normalized tables, there are no transitive dependencies, as each non-key attribute is directly dependent on the primary key of its respective table.

The final normalized tables are:

Student Table:

- StudentID (PK)
- StudentName

Course Table:

- CourseID (PK)
- CourseName
- CourseCategory

Result Table:

- StudentID (FK referencing Student table)
- CourseID (FK referencing Course table)
- Grade

This 3NF design eliminates data redundancy and anomalies from the original table.

a)Difference between primary key , unique key, Not Null with an example?(3 d)

questions 20,21 use previous knowledge.

20. Create a table called EMP with the following structure.

Name	Type
EMPNO	NUMBER(6)
ENAME	VARCHAR2(20)
JOB	VARCHAR2(10)
DEPTNO	NUMBER(3)
SAL	NUMBER(7,2)

Create Dept table with the following structure.

Name	Type
DEPTNO	NUMBER(2)
DNAME	VARCHAR2(10)
LOC	VARCHAR2(10)

i) Insert into a single record in dept table

ii)Display specify columns in emp table

iii) Delete only the data working as Lecturer

iv) List the records in emp table by salary in ascending order.

v) Update the emp table to set salary of all employees to RS.14000 who are working as Manager.

vi) Write the differences between count() and count(*) with examples

21. Create a table called EMP with the following structure.

Name	Type
EMPNO	NUMBER(6)
ENAME	VARCHAR2(20)
JOB	VARCHAR2(10)
DEPTNO	NUMBER(3)
SAL	NUMBER(7,2)

Create Dept table with the following structure.

Name	Type
DEPTNO	NUMBER(2)
DNAME	VARCHAR2(10)
LOC	VARCHAR2(10)

i) Add a column experience to the emp table.

ii) Modify the column width of the job field of emp table.

iii) create the emp1 table with ename and empno, add constraints to check the empno value while entering (i.e) empno > 100

iv) Drop any column in the emp table.

v) Rename any column in dept table.

vi) List out different SQL commands? Write their basic structure

22. **Concept design with E-R Model** Consider the following information about a university database:

- Professors have an SSN, a name, an age, a rank, and a research specialty.
- Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budget.
- Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.).
- Each project is managed by one professor (known as the project's principal investigator).
- Each project is worked on by one or more professors (known as the project's co-investigators).
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate students (known as the project's research assistants).
- When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.
- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professors work in one or more departments, and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working on their degree.
- Each graduate student has another, more senior graduate student (known as a student advisor) who advises him or her on what courses to take.

ANSWER:

To model the university database, I will create an Entity-Relationship (E-R) diagram:

Entities:
- Professor
- Project
- Graduate Student
- Department
- Student Advisor

Relationships:
- Professor manages Project (1:N)
- Professor works on Project (M:N)
- Graduate Student works on Project (M:N)
- Graduate Student has Student Advisor (1:1)
- Professor works in Department (M:N)
- Graduate Student has Major Department (1:1)
- Department has Chairman Professor (1:1)

Key Attributes:
- Professor: SSN, Name, Age, Rank, Research Specialty
- Project: Project Number, Sponsor Name, Start Date, End Date, Budget
- Graduate Student: SSN, Name, Age, Degree Program
- Department: Department Number, Department Name, Main Office
- Student Advisor: SSN (referencing Graduate Student)

Cardinality Constraints:

- 1 Project is managed by 1 Professor
- 1+ Professors can work on 1 Project
- 1+ Graduate Students can work on 1 Project
- 1 Graduate Student has 1 Student Advisor
- 1+ Professors can work in 1 Department
- 1 Graduate Student has 1 Major Department
- 1 Professor can be the Chairman of 1 Department

This E-R model captures the key entities, relationships, and constraints described in the problem statement. It provides a solid foundation for designing the relational database schema.

i).Write about delete, truncate,drop commands with syntax
ii)Differrence between primary key and unique key(both answers in prev questions)

---

23. a) Analyze the problem carefully and come up with the attributes of given entities and relationships. Identify the primary keys for all the entities. Identify the other keys like candidate keys, partial keys, if any.

Entities: 1. BUS 2. Ticket3. Passenger
Relationships: 1. Reservation 2. Cancellation

Entities:

1. BUS
   - BusID (Primary Key)
   - BusNumber
   - Capacity
   - RouteNumber

2. Ticket
   - TicketID (Primary Key)
   - PassengerID (Foreign Key referencing Passenger)
   - BusID (Foreign Key referencing BUS)
   - DateOfTravel
   - SeatNumber
   - Price

3. Passenger
   - PassengerID (Primary Key)
   - FirstName
   - LastName
   - PhoneNumber
   - Email

Relationships:

1. Reservation
   - This is the relationship between Passenger and Ticket entities.
   - One Passenger can have multiple Tickets.
   - One Ticket belongs to one Passenger.

2. Cancellation

- This is the relationship between Passenger and Ticket entities.
- A Passenger can cancel one or more of their Tickets.
- A Ticket can be cancelled by one Passenger.

Key Identifiers:

- BUS entity: BusID is the primary key.
- Ticket entity: TicketID is the primary key. PassengerID and BusID are foreign keys.
- Passenger entity: PassengerID is the primary key.

b) Write a PL/SQL program that uses all cursor operation on reserves data base.

```
DECLARE
  CURSOR reserves_cursor IS
    SELECT sid, bid, day
    FROM reserves;

  reserves_rec reserves_cursor%ROWTYPE;

  cursor_found BOOLEAN;

BEGIN
  -- Open the cursor
  OPEN reserves_cursor;

  -- Fetch and process data
  cursor_found := TRUE;
  WHILE cursor_found LOOP
    FETCH reserves_cursor INTO reserves_rec;

    -- Check cursor status
    IF reserves_cursor%FOUND THEN
      -- Perform operations on the fetched data
      DBMS_OUTPUT.PUT_LINE('Student ID: ' || reserves_rec.sid);
      DBMS_OUTPUT.PUT_LINE('Bid: ' || reserves_rec.bid);
      DBMS_OUTPUT.PUT_LINE('Day: ' || reserves_rec.day);
    ELSE
      cursor_found := FALSE;
    END IF;
  END LOOP;

  -- Close the cursor
  CLOSE reserves_cursor;
END;
/
```

.

24. Create an Employee table with the following data, insert 10 records & display?

Tablename: EMPLOYEE123		
Attributes	Domain	Constraint type
Empid	Varchar2(10)	Primary key
Name	Varchar2(15)	UNIQUE
Job	Varchar2(10)	
address	Varchar2(35)	
Salary	Number(10,2)	
DOJ	Date	

a. Insert 5 records into employee table
b. Perform update and delete operation on employee table
c. Perform Aggregate functions with Syntax and examples
Use prev questions to solve above 3 questions
d. Display the names of employees starting with ‘P’ letter.

```
SELECT name
FROM employees
WHERE name LIKE 'P%';
```

e. Write a PL/SQL program that uses all cursor operation on reserves data base.(23 b)

25. Create a MERCHANT table with the following data and perform the below Operations?(use prev questions to solve)

Tablename: **MERCHANT100**		
Attributes	**Datatype**	**Constraint type**
Mer_id	Varchar2(10)	
Name	Varchar2(15)	Unique
age	Integer	
budget	Number(12,2)	Check >=100000

a. Add a New column “Address” with data type “Varchar2 (15)” to the existing table.
b. Modify the size of “Address” to “varchar2 (35)”.
c. Drop a column “age” from the table.
d. Add “Primary key” constraint for the “Mer-Id” Attribute.
e. Insert 5 records & display them.
f. List out different SQL commands? Write their basic structure

25. MERCHANT100 Table Operations

Let's go through each of the operations step by step, starting with the table creation and then performing the requested operations.

Step 1: Create the MERCHANT100 Table

```
CREATE TABLE MERCHANT100 (

    Mer_id VARCHAR2(10),

    Name VARCHAR2(15) UNIQUE,

    age INTEGER,

    budget NUMBER(12,2) CHECK (budget >= 100000)

);
```

Step 2: Operations

a. Add a New Column “Address” with data type "Varchar2(15)" to the Existing Table

```
ALTER TABLE MERCHANT100

ADD Address VARCHAR2(15);
```

This adds the "Address" column with a data type of `VARCHAR2(15)` to the existing `MERCHANT100` table.

b. Modify the Size of the “Address” Column to "Varchar2(35)"

```
ALTER TABLE MERCHANT100

MODIFY Address VARCHAR2(35);
```

This changes the size of the `Address` column from `VARCHAR2(15)` to `VARCHAR2(35)`.

c. Drop the Column “age” from the Table

```
ALTER TABLE MERCHANT100

DROP COLUMN age;
```

This removes the `age` column from the `MERCHANT100` table.

d. Add “Primary Key” Constraint for the “Mer_id” Attribute

```
ALTER TABLE MERCHANT100

ADD CONSTRAINT pk_mer_id PRIMARY KEY (Mer_id);
```

This adds a `PRIMARY KEY` constraint on the `Mer_id` column, making it unique and not null.

e. Insert 5 Records and Display Them

```
-- Inserting 5 records into the MERCHANT100 table

INSERT INTO MERCHANT100 (Mer_id, Name, budget, Address)

VALUES ('M001', 'Merchant A', 150000, 'Address A');


INSERT INTO MERCHANT100 (Mer_id, Name, budget, Address)

VALUES ('M002', 'Merchant B', 250000, 'Address B');


INSERT INTO MERCHANT100 (Mer_id, Name, budget, Address)

VALUES ('M003', 'Merchant C', 500000, 'Address C');


INSERT INTO MERCHANT100 (Mer_id, Name, budget, Address)

VALUES ('M004', 'Merchant D', 120000, 'Address D');


INSERT INTO MERCHANT100 (Mer_id, Name, budget, Address)

VALUES ('M005', 'Merchant E', 200000, 'Address E');


-- Displaying the records

SELECT * FROM MERCHANT100;
```

This will insert 5 records into the `MERCHANT100` table and then display all the records using `SELECT *`.