Charotar University of Science and Technology [CHARUSAT]
Faculty of Technology and Engineering
Chandubhai S. Patel Institute of Technology
Devang Patel Institute of Advance Technology & Research (DEPSTAR)
Subject: Database Management System

Practical List [A.Y. 2024-25]

Practical-1

Aim: Global Trust Bank is expanding its operations and requires a robust database management system to efficiently manage its employees, job profiles, customers' accounts, and loan information. The bank has laid out specific requirements and constraints to ensure data integrity, uniqueness, and completeness.

Requirements

- **Employee Management**:
  - **Job Profiles**: Maintain records of different job profiles.
  - **Employees**: Store detailed information about employees, including their association with job profiles.
- **Customer Management**:
  - **Accounts**: Maintain separate records for customers' bank accounts.
  - **Loans**: Maintain separate records for customers' loan details

Design and implement the schema as per the given information.

Constraints –

- Not Null Constraints: Critical fields must not be null to ensure data completeness.
- Unique Constraints: Certain fields must have unique values to avoid duplicates (e.g., Account Number).
- Check Constraints: Enforce domain integrity by limiting the values that can be placed in a column.

Tasks:-

- Create Table Job (job_id, job_title, min_sal, max_sal)

COLUMN NAME	DATA TYPE
job_id	Varchar2(15)
job_title	Varchar2(30)
min_sal	Number(7,2)
max_sal	Number(7,2)

- Create table Employee (emp_no, emp_name, emp_sal, emp_comm, dept_no)

COLUMN NAME	DATA TYPE
emp_no	Number(10)
emp_name	Varchar2(30)
emp_sal	Number(8,2)
emp_comm	Number(8,2)

dept_no	Number(8,2)

- Create table deposit (a_no, name,bname, amount,a_date).

COLUMN NAME	DATA TYPE
a_no	Varchar2(15)
cname	Varchar2(15)
bname	Varchar2(15)
amount	Number(10,2)
a_date	Date

- Create table borrow (loan no,cname,bname,amount).

COLUMN NAME	DATA TYPE
loanno	Varchar2(15)
cname	Varchar2(15)
bname	Varchar2(15)
amount	Number(10,2)

- Insert the following values in the table Employee.

emp_no	emp_name	emp_sal	emp_comm	dept _no
101	Smith	800	455	20
102	Snehal	1600	0	25
103	Adama	1100	425	20
104	Aman	3000		15
105	Anita	5000	50,000	10
106	Anamika	2975		30

- Insert the following values in the table JOB.

job_id	job_title	min_sal	max_sal
IT_PROG	Programmer	4000	10000
MK_MGR	Marketing manager	9000	15000

FI_MGR	Finance manager	8200	12000
FI_ACC	Account	4200	9000
LEC	Lecturer	6000	17000
COMP_OP	Computer Operator	1500	13000

- Insert the following values in the table deposit.

a_no	**cname**	**bname**	**amount**	**a_date**
101	Anil	andheri	7000	01-jan-06
102	sunil	virar	5000	15-jul-06
103	jay	villeparle	6500	12-mar-06
104	vijay	andheri	8000	17-sep-06
105	keyur	dadar	7500	19-nov-06
106	mayur	borivali	5500	21-dec-06

- Insert the following values in the table borrow.

loanno	**cname**	**bname**	**amount**
201	ANIL	VRCE	1000.00
206	MEHUL	AJNI	5000.00
311	SUNIL	DHARAMPETH	3000.00
321	MADHURI	ANDHERI	2000.00
375	PRMOD	VIRAR	8000.00
481	KRANTI	NEHRU PLACE	3000.00

- Describe the table Job, employee, deposit, borrow.

Practical-2

Aim: Global Trust Bank is expanding its operations and requires a robust database management system to efficiently manage its employees, job profiles, customers' accounts, and loan information. The bank has laid out specific requirements and constraints to ensure data integrity, uniqueness, and completeness. Perform Data Definition Language (DDL) commands and change the existing schema as per the given information.

Constraints –

- Not Null Constraints: Ensure critical fields are not null.
- Unique Constraints: Ensure data integrity by limiting column values.

- Check Constraints: Ensure columns like Account Number have unique values.

Tasks:-

1) Create a table supplier from an employee with all the columns and verify.

Test case: Verify suppler table consists of all columns of employee table.

2) Create table sup1 from an employee with the first two columns and verify.

Test case: Verify sup1 consists of first two columns of employee table.

3) Create table sup2 from employee with no data and verify.

Test case: Verify sup2 table contains no data.

4) Insert the data into sup2 from employee whose name is ‘Anita’ and verify.

Test case: Verify the details of employee whose name is ‘Anita’ is inserted in sup2 table.

5) Rename the table sup2 and verify.

Test case: Verify the sup2 table name is changed.

6) Destroy table sup1 with all the data and verify.

Test case: Verify table sup1 is destroyed or not.

7) Add one column phone to an employee with size of column is Varchar2(10) and verify.

Test case: Verify phone column is added in employee as per the column size.

8) Modify column phone and change type to char(10) and verify.

Test case: Verify phone column type is changed to char(10).

9) Delete employee_name column from sup2 and verify;

Test case: Verify employee_name column is deleted from sup2.

10) Rename the column salary to new_sal in sup2 and verify.

Test case: Verify the column name of salary is changed to new_sal in sup2 table.

Practical-3

Aim: Global Trust Bank is expanding its operations and requires a robust database management system to efficiently manage its employees, job profiles, customers' accounts, and loan information. The bank has laid out specific requirements and constraints to ensure data integrity, uniqueness, and completeness. Perform Data Definition Language (DDL) commands and change the existing schema as per given information.

Constraints –

- Not Null Constraints: Ensure critical fields are not null.
- Unique Constraints: Ensure data integrity by limiting column values.
- Check Constraints: Ensure columns have unique values where required.

Test Cases-

1) Retrieve all data from **employee, jobs and deposit.**

Test Case: Verify all rows are retrieved from each table.

2) Display job title and maximum salary of all jobs.

Test Case: Verify the job title and maximum salary for each job.

3) Write a query to find out all the branches.

Test Case: Verify the list of all unique branches.

4) Display all the account no. into which rupees are between dates **01-01-06 and 25-07-06**.

Test Case: Verify the account numbers with deposits in the specified date range.

5) Display names of all customers whose account is deposited after **09-oct-06**

   Test Case: Verify the names of customers with accounts deposited after the specified date.

6) Display name and salary of employee whose department no is 20. Give alias name to name of employee.

   Test Case: Verify the names and salaries of employees in department 20 with the alias.

7) Display employee no, name and department details of those employee whose department lies **in(10,20).**

   Test Case: Verify the details of employees in departments 10 and 20.

8) Display employee no, name and department details of those employee whose department **not in(15,30) except 25 .**

   Test Case: Verify the details of employees excluding departments 15, 30, and 25.

9) Display employee no, name and department details of those employee whose department no is **between 15 and 25.**

   Test Case: Verify the details of employees in departments between 15 and 25.

10) Display name of all employee whose emp_comm contains the **non-null** values.

    Test Case: Verify the names of employees with non-null emp_comm values.

11) Combine two columns min_sal and max_sal and display it one column using common alias name.
    Test Case: Verify the combined minimum and maximum salary displayed as a single column.
12) Insert the data into sup2 from employee.
    Test Case: Verify the data from Employee is inserted into Supplier2.

13) Delete all the rows from sup1 as sup.

    Test Case: Verify all rows are deleted from Sup1.

14) Delete the detail of supplier whose emp_no is 103.

Test Case: Verify the row with emp_no 103 is deleted from Supplier.

15) Update the name of employee to **'Aman'** name whose emp_name is 'Anita'.

Test Case: Verify the name 'Anita' is updated to 'Aman'.

16) Update the value of employee name whose employee number is 103.
Test Case: Verify the name is updated for the employee with EmployeeID 103.

17) Find out the maximum and minimum salary form job table.
Test Case: Verify the maximum and minimum salary from the JobProfile table.

18) Find out the average salary of employee.
Test Case: Verify the average salary of employees.

19) Count the total no as well as distinct rows in dept_no column with a condition of salary greater than 1000 of employee.
Test Case: Verify the total and distinct count of departments with a salary greater than 1000.

20) Display the detail of all employees in ascending order, descending order of their name and no.
Test Case: Verify the employee details sorted by name in ascending order and by number in descending order.

21) Display the dept_no in ascending order and accordingly display emp_comm in descending order.
Test Case: Verify the dept_no in ascending order and emp_comm in descending order.

22) Update the value of emp_comm to 500 where dept_no is 20.
Test Case: Verify the emp_comm is updated to 500 for department 20.

23) Display the emp_comm in ascending order with null value first and accordingly sort employee salary in descending order.
Test Case: Verify emp_comm in ascending order with null values first and salary in descending order.

24) Display the emp_comm in ascending order with null value last and accordingly sort emp_no in descending order.

Test Case: Verify emp_comm in ascending order with null values last and emp_no in descending order.

Practical-4

Aim: - You are a database administrator for a multinational bank. The bank requires insights and maintenance of its employee and customer databases to ensure data consistency and retrieval of relevant information for various operations. Your tasks involve applying constraints and writing SQL queries to retrieve specific data based on given conditions. Below are the tasks to be performed:

The bank maintains the following schemas:

Employee Schema

- Emp_ID (Primary Key)
- Emp_Name (Not Null)
- Emp_Salary (Not Null, Check: Greater than zero)
- Job_ID (Unique)
- Other attributes you can add as needed.

Customer Schema

- Cust_ID (Primary Key)
- Cust_Name (Not Null)
- Branch (Not Null)
- Other attributes you can add as needed**.**

Constraints –

- Not Null Constraints: Ensure critical fields are not null.
- Unique Constraints: Ensure data integrity by limiting column values.
- Check Constraints: Ensure columns have unique values where required.

Tasks: –

The HR team wants to identify specific employees for a new project. Write SQL queries to:

1. Retrieve the details of employees whose names:
    - Start with the letter 'A'.
    - Have 'a' as the third character.
    - Test Case: Verify that Aman and Adama’s details are displayed.

2. Display the names, employee numbers, and salaries of employees whose names:
    - Are exactly 5 characters long.
    - Start with 'Ani'.
    - Test Case: Verify that Anita's details are displayed.

To analyze naming patterns in the employee database, perform the following:

1. Retrieve the details of employees whose second character is either 'M' or 'N'.
    - Test Case: Verify that Aman, Anamika, and Anita's details are displayed.
2. Retrieve the details of employees whose second character is 'n', and their names are exactly 5 characters long.
    - Exclude null values for employee details.
    - Test Case: Verify that Anita's details are displayed.
3. Identify employees whose details are partially incomplete (contain null values) and whose third character in their names is 'a'.
    - Test Case: Verify that Aman's details are displayed.

The marketing department plans a special campaign for customers in specific branches. Write a query to:

- Retrieve the names of all customers whose branch is either 'Andheri', 'Dadar', or 'Virar'.
- **Test Case**: Verify that Anil, Sunil, and Keyur's names are displayed.

To optimize the workforce and their roles, retrieve the following:

1. Job names whose job IDs begin with 'FI_'.
    - **Test Case**: Verify that FI_MGR and FI_ACC are displayed.
2. Titles of jobs that end with '_MGR' and have a maximum salary greater than ₹12,000.
    - **Test Case**: Verify that Marketing Manager is displayed.

Practical-5

Aim: As a database administrator for a global bank, you are responsible for managing and analyzing employee and customer data stored in the bank's database. Your tasks involve using SQL functions to

manipulate and retrieve critical information efficiently. These operations ensure seamless data communication and compliance with bank regulations.

Constraints

- **Not Null Constraints**: Critical fields like names and salaries must not be null.
- **Unique Constraints**: Ensure integrity of fields like Job_ID.
- **Check Constraints**: Validate positive salary values.

The bank maintains the following schemas:

1. **JobProfile Table**: Stores details of employees and their job roles.
    - Emp_ID (Primary Key)
    - Emp_Name (Not Null)
    - Emp_Salary (Not Null, Check: Greater than zero)
    - Job_ID (Unique)
    - Department
2. **Customer Table**: Stores customer details.
    - Cust_ID (Primary Key)
    - Cust_Name (Not Null)

The HR department wants to analyze employee salaries for better planning. Write SQL queries to:

1. Calculate the average salary of employees (with and without duplicates).
    - **Test Case**: Verify the average salary and distinct average salary.
2. Retrieve the minimum salary from the JobProfile table.
    - **Test Case**: Verify the minimum salary.
3. Count the total number of employees and distinct departments.
    - **Test Case**: Verify the total employee count and distinct department count.
4. Retrieve the maximum salary from the JobProfile table.
    - **Test Case**: Verify the maximum salary.
5. Calculate the total and distinct sum of all salaries.
    - **Test Case**: Verify the total and distinct sum of salaries.

The finance team needs specific salary calculations for tax and benefits. Write SQL queries to:

1. Calculate the absolute difference between each employee's salary and ₹1,000.
    - **Test Case**: Verify the absolute difference for each salary.

2. Compute the square of each employee's salary.
    - **Test Case**: Verify the squared salary values.
3. Round salaries to two decimal places.
    - **Test Case**: Verify salaries rounded to two decimal places.
4. Find the square root of salaries.
    - **Test Case**: Verify the square root of each salary.

To ensure uniformity in names across systems, perform the following:

1. Convert all employee first names to lowercase, uppercase, and initial caps.
    - **Test Cases**: Verify first names in all three formats.
2. Extract the first three characters of employee first names.
    - **Test Case**: Verify extracted substrings.
3. Find the length of each employee's first name.
    - **Test Case**: Verify the length of first names.
4. Remove leading 'A' and trailing 'a' from employee first names.
    - **Test Cases**: Verify names after applying LTRIM and RTRIM.
5. Pad employee first names with '*' on the left and right, ensuring a total length of 10.
    - **Test Cases**: Verify left-padded and right-padded names.

The data migration team requires conversions between different data types:

1. Convert a string representation of a salary to a numeric format.
    - **Test Case**: Verify the numeric conversion.
2. Format a numeric salary value into a string with specific formatting.
    - **Test Case**: Verify the formatted string.

To assist in employee scheduling, perform the following:

1. Calculate the date after adding 6 months to the current date.
    - **Test Case**: Verify the calculated date.
2. Retrieve the last day of the current month.
    - **Test Case**: Verify the last day.
3. Calculate the number of months between two dates.
    - **Test Case**: Verify the months between the dates.
4. Find the next Monday from the current date.
    - **Test Case**: Verify the next Monday's date.

To identify overlaps and differences between employees and customers, write SQL queries to:

1. Retrieve the union of first names from employees and customers.
    - **Test Case**: Verify the union result.
2. Retrieve the union of first names (including duplicates).
    - **Test Case**: Verify the union all result.
3. Find the intersection of first names from employees and customers.
    - **Test Case**: Verify the intersecting names.
4. Identify first names present in the employees table but not in customers.
    - **Test Case**: Verify the difference result.

Practical-6

AIM: You are a database administrator for a manufacturing and consulting company. The company maintains two primary tables: Product and Employee Company (emp_company). You are tasked with solving business queries related to order quantities, employee salaries, and company analysis using SQL grouping and aggregate functions. To manipulate and retrieve meaningful insights using grouping and aggregate functions in SQL while adhering to database constraints and integrity rules.

- Not Null Constraints: Critical fields such as product numbers, employee names, and salaries must not contain null values.
- Unique Constraints: Ensure the integrity of unique fields like Product_no and ENAME.
- Check Constraints: Validate that quantities and salaries have valid positive values

1. **Product Table**: Tracks order details for various products.
   - Detorder_no (Primary Key)
   - Product_no (Not Null, Unique)
   - Qty_order (Not Null, Check: Greater than zero)
2. **emp_company Table**: Tracks employees, their companies, and salaries.
   - ENAME (Not Null, Unique)
   - CNAME (Not Null)
   - SALARY (Not Null, Check: Greater than zero)

The logistics department has provided the following product order details to be inserted into the Product table:

Detorder_no	Product_no	Qty_order
O19001	P00001	10
O19001	P00002	3
O19002	P00001	4
O19003	P00004	2
O19004	P00003	6
O19005	P00005	2
O19006	P00004	7

1. **Insert Values**:
   - Insert the above data into the Product table.
   - **Test Case**: Verify that the values are inserted correctly by displaying all rows.
2. **Total Quantity per Product**:
   - Retrieve the product numbers and total quantities ordered for each product.
   - **Test Case**: Verify that the sum of quantities ordered for each product number is calculated correctly.
3. **Filter Specific Products**:
   - Retrieve the product numbers and total quantities ordered for products P00001 and P00004.

- **Test Case**: Verify that the sum of quantities ordered for product numbers P00001 and P00004 is calculated correctly.

The HR department has provided the following employee data to be inserted into the emp_company table:

ENAME	CNAME	SALARY
Anil	ACC	1500
Shankar	TATA	2000
Jay	WIPRO	1800
Sunil	WIPRO	1700
Vijay	TATA	5000
Prakash	TATA	3000
Ajay	ACC	8000
Abhay	ACC	1800

1. **Insert Values**:
   - Insert the above data into the emp_company table.
   - **Test Case**: Verify that the values are inserted correctly by displaying all rows.
2. **Maximum Salary per Company**:
   - List the company names and the maximum salary in each company.
   - **Test Case**: Verify that the maximum salary for each company is calculated correctly.
3. **Average Salary per Company**:
   - Calculate the average salary for each company.
   - **Test Case**: Verify that the average salary for each company is calculated correctly.
4. **Filter Companies by Average Salary**:
   - List the names of companies with an average salary greater than ₹1,500.
   - **Test Case**: Verify that the companies with an average salary greater than ₹1,500 are listed correctly.
5. **Exclude a Specific Company**:
   - Calculate the average salary for each company except ACC.
   - **Test Case**: Verify that the average salary for each company, excluding ACC, is calculated correctly.

Practical-7

AIM - As a database administrator for a university, you are tasked with designing and implementing a database schema in the MS Access tool. This database should efficiently manage relationships between departments, courses, students, and their academic performance. To ensure data integrity and consistency, you will define master-slave relationships with appropriate integrity constraints.

Charotar University of Science and Technology [CHARUSAT]
Faculty of Technology and Engineering
Chandubhai S. Patel Institute of Technology
Devang Patel Institute of Advance Technology & Research (DEPSTAR)
Subject: Database Management System

Practical List [A.Y. 2024-25]

The university database consists of the following entities:

1. **Department (Master Table)**:
    - Stores information about university departments.
    - Attributes: Dept_ID (Primary Key), Dept_Name (Not Null, Unique).
2. **Course (Slave Table)**:
    - Stores courses offered by departments.
    - Attributes: Course_ID (Primary Key), Course_Name (Not Null), Dept_ID (Foreign Key referencing Department).
3. **Student (Slave Table)**:
    - Stores details of enrolled students.
    - Attributes: Student_ID (Primary Key), Student_Name (Not Null), Dept_ID (Foreign Key referencing Department).
4. **Enrollment (Slave Table)**:
    - Tracks student enrollments in courses.
    - Attributes: Enrollment_ID (Primary Key), Student_ID (Foreign Key referencing Student), Course_ID (Foreign Key referencing Course), Grade.

Tasks

1. **Create Master Table (Department)**:
    - Design a Department table with Dept_ID as the Primary Key.
    - Enforce the following constraints:
        - Dept_Name must be unique.
        - No null values in Dept_Name.
    - **Test Case**: Verify that duplicate Dept_Name values cannot be inserted.
2. **Create Slave Table (Course)**:
    - Design a Course table with Course_ID as the Primary Key.

- Establish a relationship with the Department table using the Dept_ID foreign key.
- Enforce referential integrity with the following rules:
  - Cascade updates: If a Dept_ID is updated in the Department table, the corresponding Dept_ID in the Course table should update automatically.
  - Restrict deletions: Prevent deleting a department if courses are linked to it.
- **Test Case**: Verify that deleting a department linked to courses is restricted.

3. **Create Slave Table (Student)**:
   - Design a Student table with Student_ID as the Primary Key.
   - Establish a relationship with the Department table using the Dept_ID foreign key.
   - Enforce referential integrity to ensure that each student belongs to a valid department.
   - **Test Case**: Verify that a student cannot be added with a Dept_ID that does not exist in the Department table.
4. **Create Slave Table (Enrollment)**:
   - Design an Enrollment table with Enrollment_ID as the Primary Key.
   - Establish relationships:
     - Student_ID as a foreign key referencing the Student table.
     - Course_ID as a foreign key referencing the Course table.
   - Enforce referential integrity for cascading updates and restricting deletions:
     - Cascade updates: If a Student_ID or Course_ID is updated in their respective tables, the changes should reflect in the Enrollment table.
     - Restrict deletions: Prevent deleting a student or course if enrollment records exist.
   - **Test Case**: Verify that deleting a student or course linked to enrollments is restricted.
5. **Data Validation Rules**:
   - Ensure that grades in the Enrollment table only accept valid values (A, B, C, D, F).
   - **Test Case**: Verify that invalid grades (e.g., E or empty values) cannot be inserted into the Grade field.

6. **Data Entry**:
    - Populate the tables with sample data for departments, courses, students, and enrollments.
    - Example Data:
        - **Department Table**:

Dept_ID	Dept_Name
D001	Computer Engg
D002	Electronics
D003	Civil Engg

        - **Course Table**:

Course_ID	Course_Name	Dept_ID
C001	Data Structures	D001
C002	Circuit Theory	D002

        - **Student Table**:

Student_ID	Student_Name	Dept_ID
S001	Alice	D001
S002	Bob	D003

        - **Enrollment Table**:

Enrollment_ID	Student_ID	Course_ID	Grade
E001	S001	C001	A

    - **Test Case**: Verify that the inserted sample data adheres to all constraints and relationships.

Practical-8

Charotar University of Science and Technology [CHARUSAT]
Faculty of Technology and Engineering
Chandubhai S. Patel Institute of Technology
Devang Patel Institute of Advance Technology & Research (DEPSTAR)
Subject: Database Management System

Practical List [A.Y. 2024-25]

AIM - You are tasked with designing a database for managing a sales and client management system. The database should support the creation, updating, and querying of data related to salespeople, customers, orders, and clients while ensuring data integrity through constraints. (Refer the attached excel sheet for Database)

1. Managing Salespeople Information: The organization needs a table to store data about its salespeople. Each salesperson must have a unique ID, a name, and optionally, a city and commission rate.

Tasks:

1. Create the Salespeople Table:
    - Attributes:
        - Snum: Salesperson ID (Primary Key, number).
        - Sname: Salesperson Name (Not Null).
        - City: City (nullable).
        - Comm: Commission Rate (nullable).

Constraints:

- Primary Key Constraint: Ensure Snum uniquely identifies each record.
- Not Null Constraint: Ensure Sname is not null.

2. Test Cases:
    - Insert valid records to ensure successful insertion.
    - Try inserting a record with duplicate Snum and ensure the system rejects it.
    - Attempt to insert a record with a null Sname and ensure it fails.
    - Insert a record with null City and Comm and ensure it is allowed.
    - Retrieve all records from the Salespeople table to verify data integrity.

2. Managing Customer Information: The system must store customer details, including their relationship with salespeople. Customers have unique IDs, names, and optionally a city. They also have a rating with a default value.

Tasks:

1. Create the Customer Table:
    - Attributes:
        - Cnum: Customer ID (Primary Key, number).
        - Cname: Customer Name (Not Null).
        - City: City (nullable).
        - Rating: Customer rating (Default: 10).
        - Snum: Salesperson ID (Foreign Key references Salespeople).

Constraints:

- Primary Key Constraint: Ensure Cnum uniquely identifies each record.
- Foreign Key Constraint: Ensure Snum references Salespeople(Snum).
- Default Constraint: Ensure Rating defaults to 10 when not provided.

2. Test Cases:
    - Insert valid records to verify proper insertion.

- Insert records with missing Rating to ensure the default value is applied.
- Attempt to insert a record with null Cname and ensure it fails.
- Try inserting a record with a non-existent Snum and ensure it fails.
- Insert records with null City and verify successful insertion.
- Retrieve all records from the Customer table to verify the data.

3. Managing Orders: The company tracks orders placed by customers, which are managed by salespeople. Each order has an ID, amount, date, and links to customers and salespeople.

Tasks:

1. Create the Order Table:
   - Attributes:
     - Order_no: Order Number (Primary Key, number).
     - Amount: Order Amount (nullable).
     - Odate: Order Date (nullable).
     - Cnum: Customer ID (Foreign Key references Customer).
     - Snum: Salesperson ID (Foreign Key references Salespeople).

Constraints:

- Primary Key Constraint: Ensure Order_no uniquely identifies each record.
- Foreign Key Constraint: Ensure Cnum references Customer(Cnum).
- Foreign Key Constraint: Ensure Snum references Salespeople(Snum).

2. Test Cases:
   - Insert valid records to verify proper insertion.
   - Attempt to insert records with non-existent Cnum or Snum and ensure the system rejects them.
   - Insert records with null Amount or Odate and ensure they are allowed.
   - Retrieve all records from the Order table to verify data integrity.

4. Managing Sales Orders: Sales orders track additional details such as delivery type and order status, with constraints to ensure consistency.

Tasks:

1. Create the Sales_order Table:
   - Attributes:
     - Order_no: Order Number (Primary Key, first letter must start with ‘O’).
     - Order_date: Date of the Order.
     - Client_no: Client ID (Foreign Key references Client_master).
     - Dely_addr: Delivery Address.
     - Salesman_no: Salesperson ID (Foreign Key references Salesman_master).
     - Dely_type: Delivery Type (Default: ‘F’).
     - Order_status: Order Status (Allowed Values: In Process, Fulfilled, Backorder, Cancelled).

Constraints:

- Primary Key, Foreign Key, Check, and Default constraints.

2. Test Cases:
    - Verify that all constraints, including the primary key, check constraints, and default values, are applied correctly.

5. Managing Salesman Details: The organization needs to maintain records of salesmen, including their target and year-to-date sales, with strict constraints for validation.

Tasks:

1. Create the Salesman_master Table:
    - Attributes:
        - Salesman_no: Salesman Number (Primary Key, must start with 'S').
        - Salesman_name: Salesman Name (Not Null).
        - Address, City, Pincode, State: Additional details.
        - Sal_amt: Salary (Not Null, cannot be 0).
        - Ytd_sales: Year-to-date Sales (Not Null, cannot be 0).
        - Tgt_sales: Target Sales (Not Null).

Constraints:

- Primary Key, Not Null, and Check constraints.

2. Test Cases:
    - Verify that records with invalid Sal_amt or Ytd_sales are rejected.
    - Ensure all constraints are correctly applied.

6. Managing Client Details: Clients are key stakeholders in the sales process. Their data must be consistent and adhere to specific constraints.

Tasks:

1. Create the Client_master Table:
    - Attributes:
        - Client_no: Client Number (Primary Key, must start with 'C').
        - Name: Client Name (Not Null).
        - Address, City, State, Pincode: Client details.
        - Bal_due: Balance Due.

Constraints:

- Primary Key, Not Null, and Check constraints.

2. Test Cases:
    - Verify that all constraints, including the primary key, not null, and check constraints, are correctly applied.
    - Ensure invalid records are rejected.

Practical-9

AIM - You are a database administrator for a company managing sales, customers, and orders. The system must provide robust querying capabilities to retrieve, analyze, and manipulate data based on specific business requirements. Your task is to use SQL JOIN commands and constraints to answer real-world queries about salespeople, customers, and orders. (Refer the attached excel sheet for Database)

1. **Salespeople**: Contains information about sales staff, including their ID, name, city, and commission rate.

2. **Customer**: Stores details of customers, their ratings, and the salesperson managing them.

3. **Order**: Tracks order details, including the order amount, date, and links to customer and salesperson records.

Task:

1. The sales manager wants to identify all customers being handled by salespeople named Peel or Motika.
2. The finance team wants to exclude invalid orders where the amount is 0 or missing.
3. The management wants to analyze the highest order values processed by salespeople for orders exceeding $3000.
4. The HR department wants to find relationships between salespeople and customers in the same city.
5. The operations team needs a list of orders along with the names of customers who placed them.
6. Identify customers managed by salespeople earning more than 12% commission.
7. Identify customers with a rating identical to Hoffman's.
8. The analytics team wants to find the number of customers with ratings exceeding the average rating of customers from San Jose.
9. Identify salespeople whose total orders exceed the largest single order amount in the system.
10. Create a categorized listing of customers based on their ratings. Ratings >= 400 should be marked as 'High Rating'; otherwise, 'Low Rating'.

Practical-10

AIM - You are a database developer for a financial application that requires frequent mathematical calculations. One critical operation is multiplying two numbers provided by users and returning the result. To achieve this, you must create a robust PL/SQL procedure that handles various scenarios, including positive and negative numbers, zero values, and floating-point numbers. Create a PL/SQL

procedure that calculates the product of two numbers, ensuring error handling for invalid inputs and consistent results.

Task:

1. A user wants to calculate the product of two positive numbers, e.g., 10 and 5, to determine total costs.
2. A user wants to verify that multiplying any number with zero results in zero, e.g., 15 and 0.
3. A user is calculating the product of two negative numbers, e.g., -4 and -6, which should result in a positive product.
4. A user is calculating the product of a positive and a negative number, e.g., 7 and -3, to validate the outcome as negative.
5. A user wants to calculate the product of two decimal numbers, e.g., 2.5 and 4.2, for precision-sensitive operations.

Practical-11

Aim: Your organization maintains an employee database that includes details like names and ages. To ensure data integrity and audit compliance, any record deleted due to specific conditions must first be archived. You are tasked with implementing a system where records of employees aged 21 are deleted from the main table but saved in an archive table before deletion.

Task:

1. The HR department wants to remove all records of employees aged 21 from the Employee table as part of a policy update. Use a PL/SQL block to delete records from the main table based on the condition.
2. Implement a trigger that automatically archives the original record in a separate table before deletion.

Practical-12

Aim: Your task is to design and implement database functionalities for managing product information in an e-commerce system. The system should allow users to create, retrieve, update, and delete product data, compute discounts on products using a function, and maintain an audit trail for product updates using a trigger.

Task:

1. Create a stored procedure to insert a new product into the Product table.
2. Create a stored procedure to retrieve product details by product_id.
3. Create a stored procedure to update the stock of a product.
4. Create a stored procedure to delete a product by product_id.
5. Create a function to calculate the discounted price of a product.

6. Implement a trigger to maintain an audit trail of updates made to the Product table. Before any update, the trigger should insert a record into an AuditTrail table, storing the old product details along with the update timestamp.

Test Cases

Task 1: CRUD Operations

1. **Add Product:** Add a product with name "Laptop", category "Electronics", price 50000, and stock 10.
   Expected Result: Product is added successfully.
2. **Retrieve Product Details:** Retrieve details of the product with product_id = 1.
   Expected Result: Product details are displayed correctly.
3. **Update Stock:** Update the stock of product_id = 1 to 15.
   Expected Result: Stock is updated successfully.
4. **Delete Product:** Delete the product with product_id = 1.
   Expected Result: Product is removed from the table.

Task 2: Discount Calculation

1. **Input:** Price = 1000, Discount = 10%.
   Expected Result: Discounted Price = 900.

Task 3: Trigger Validation

1. **Update Product Price or Stock:** Update the price of product_id = 2 to 55000.
   Expected Result: Old price and stock are recorded in AuditTrail with a timestamp.

Practical-13

Aim: export data from a SQL database to other commonly used file formats such as Excel, CSV, and Text.

Note:

- Access to a SQL database (MySQL, PostgreSQL, Oracle, or SQL Server).
- Software tools like SQL Workbench, pgAdmin, or MySQL Workbench.

Task:

1. Export Data to CSV Format
2. Export Data to Excel Format
3. Export Data to Text Format
4. Export Data Using SQL Server Tools
5. Write a stored procedure to automate exporting data.