1) Create a Sales Table and Use Aggregate Functions a) Create a Sales table with columns: SaleID, ProductID, Quantity, SaleAmount, and SaleDate. b) Insert at least 10 sales records with different products and quantities. c) Write a query to calculate the total revenue generated using the SUM function. d) Find the product with the highest sale amount using the MAX function. e) Retrieve the average sale amount per transaction using the AVG function.

Ans: create table sales_11(

sale_id number(5),

product_id number(5),

quantity number(5),

sale_amount number(5),

sale_date date

);

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)

VALUES (1, 101, 2, 40, TO_DATE('2025-04-11', 'YYYY-MM-DD'));

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)

VALUES (2, 101, 2, 40, TO_DATE('2025-04-11', 'YYYY-MM-DD'));

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)

VALUES (1, 101, 2, 40, TO_DATE('2025-04-11', 'YYYY-MM-DD'));

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)

VALUES (4, 101, 2, 40, TO_DATE('2025-04-11', 'YYYY-MM-DD'));

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)

VALUES (5, 101, 2, 40, TO_DATE('2025-04-11', 'YYYY-MM-DD'));

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)

VALUES (6, 106, 7, 50, TO_DATE('2025-04-14', 'YYYY-MM-DD'));

```
INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)
VALUES (7, 107, 8, 60, TO_DATE('2015-04-11', 'YYYY-MM-DD'));

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)
VALUES (8, 108, 9, 45, TO_DATE('2024-03-21', 'YYYY-MM-DD'));

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)
VALUES (9, 109, 10, 49, TO_DATE('2023-10-25', 'YYYY-MM-DD'));

INSERT INTO sales_11 (sale_id, product_id, quantity, sale_amount, sale_date)
VALUES (10, 110, 12, 75, TO_DATE('2022-09-17', 'YYYY-MM-DD'));

select * from sales_11

SELECT SUM(sale_amount) AS TotalRevenue
FROM sales_11;

SELECT product_id, sale_amount
FROM sales_11
WHERE sale_amount = (SELECT MAX(sale_amount) FROM sales_11);

SELECT AVG(sale_amount) AS AverageSaleAmount
FROM sales_11;
```

2) Use DDL and DML Commands a) Create a Products table with columns for ProductID, ProductName, Price, and StockQuantity using DDL commands. b) Insert five product records and display all products using a SELECT query. c) Update the price of a product with ProductID = 3 and check the changes using a SELECT statement. d) Delete a product from the table and verify whether the changes are reflected. e) Alter the table to add a new column Discount and set a default value of 5%.

ANS:

```
CREATE TABLE Products_11 (
```

```
  ProductID NUMBER(5) PRIMARY KEY,
  ProductName VARCHAR2(50),
  Price NUMBER(8),
  StockQuantity NUMBER(5)
);

insert into products_11 (ProductID, ProductName, Price, StockQuantity )
values(101, 'glocery', 250, 50);

insert into products_11 (ProductID, ProductName, Price, StockQuantity )
values(102, 'parlour', 300, 40);

insert into products_11 (ProductID, ProductName, Price, StockQuantity )
values(103, 'toys', 400, 55);

insert into products_11 (ProductID, ProductName, Price, StockQuantity )
values(104, 'hotel', 350, 35);

insert into products_11 (ProductID, ProductName, Price, StockQuantity )
values(105, 'motel', 450, 60);

select * from products_11

UPDATE Products_11
SET Price = 120
WHERE ProductID = 103;

delete from products_113
where ProductName = 'hotel';

alter table Products_11
```

add(discount number(5) DEFAULT 5);

select * from products_11

3) Create a Customer Table with Integrity Constraints a) Create a Customers table with constraints: CustomerID (PRIMARY KEY), Email (UNIQUE), Age (CHECK Age > 18). b) Insert a valid customer record and verify that the default country is assigned if not explicitly provided. c) Attempt to insert a customer with an age of 16 and observe the CHECK constraint violation. d) Try inserting two customers with the same email ID and observe the UNIQUE constraint violation. e) Retrieve all customers who are older than 25 and belong to a country other than 'India'.

ANS:

```
CREATE TABLE Customers_11 (
   CustomerID NUMBER(5) PRIMARY KEY,
   CustomerName VARCHAR2(50),
   Email VARCHAR2(100) UNIQUE,
   Age NUMBER(3) CHECK (Age > 18),
   Country VARCHAR2(50) DEFAULT 'India'
);

INSERT INTO Customers_11 (CustomerID, CustomerName, Email, Age)
VALUES (1, 'Rajesh Kumar', 'rajesh@gmail.com', 25);

SELECT * FROM Customers_11;

INSERT INTO Customers_11 (CustomerID, CustomerName, Email, Age)
VALUES (2, 'Ankit Sharma', 'ankit@gmail.com', 16);

INSERT INTO Customers_11 (CustomerID, CustomerName, Email, Age)
VALUES (3, 'Sneha Verma', 'sneha@gmail.com', 30);

INSERT INTO Customers_11 (CustomerID, CustomerName, Email, Age)
VALUES (4, 'Priya Singh', 'sneha@gmail.com', 28);
```

SELECT *

FROM Customers_11

WHERE Age > 25

AND Country != 'India';

4)Create a Table with Constraints a) Create an EmployeeDetails table with EmployeeID as the PRIMARY KEY and DepartmentID as a FOREIGN KEY referencing a Department table. b) Insert a valid employee record with an existing DepartmentID, then attempt to insert an employee with a non-existent DepartmentID and observe the constraint violation. c) Insert an employee with a duplicate EmployeeID and check how the primary key constraint prevents duplicate entries. d) Modify the Salary column to have a UNIQUE constraint and attempt to insert two employees with the same salary to test the constraint. e) Write a query to delete an employee from EmployeeDetails and ensure that the deletion does not violate any referential integrity constraints.

ANS:

```
CREATE TABLE Department (
   DepartmentID INT PRIMARY KEY,
   DepartmentName VARCHAR(100)
);

CREATE TABLE EmployeeDetails (
   EmployeeID INT PRIMARY KEY,
   EmployeeName VARCHAR(100),
   DepartmentID INT,
   Salary DECIMAL(10,2),
   FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);

INSERT INTO Department (DepartmentID, DepartmentName)
VALUES (1, 'HR');

INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (101, 'Alice', 1, 50000.00);
```

```
INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (102, 'Bob', 99, 55000.00);

INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (101, 'Charlie', 1, 60000.00)

ALTER TABLE EmployeeDetails
ADD CONSTRAINT unique_salary UNIQUE (Salary);

INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (103, 'David', 1, 65000.00);

INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (104, 'Eva', 1, 65000.00);

DELETE FROM EmployeeDetails
WHERE EmployeeID = 103;
```

5) Create an Employee Table with Various Columns a) Create a table Employee with attributes: EmployeeID (INT, PRIMARY KEY), Name (VARCHAR), Salary (DECIMAL), JoiningDate (DATE), and ActiveStatus (BOOLEAN). b) Insert five sample employee records and ensure each employee has a unique EmployeeID. c) Write a query to find all employees who joined before January 1, 2023. d) Update the salary of an employee named ‘Amit Sharma’ by 10% and display the updated record. e) Retrieve all employees who are currently active (ActiveStatus = TRUE).

ANS:

```
CREATE TABLE Employee (
   EmployeeID INT PRIMARY KEY,
   Name VARCHAR2(100),
   Salary DECIMAL(10, 2),
   JoiningDate DATE,
   ActiveStatus CHAR(1) CHECK (ActiveStatus IN ('Y', 'N'))
);
```

```
INSERT INTO Employee (EmployeeID, Name, Salary, JoiningDate, ActiveStatus) VALUES (1, 'Amit Sharma', 50000, TO_DATE('2022-05-15', 'YYYY-MM-DD'), 'Y');

INSERT INTO Employee (EmployeeID, Name, Salary, JoiningDate, ActiveStatus) VALUES (2, 'Neha Verma', 60000, TO_DATE('2023-02-10', 'YYYY-MM-DD'), 'Y');

INSERT INTO Employee (EmployeeID, Name, Salary, JoiningDate, ActiveStatus) VALUES (3, 'Rahul Mehta', 55000, TO_DATE('2021-08-20', 'YYYY-MM-DD'), 'N');

INSERT INTO Employee (EmployeeID, Name, Salary, JoiningDate, ActiveStatus) VALUES (4, 'Sneha Kapoor', 70000, TO_DATE('2020-11-05', 'YYYY-MM-DD'), 'Y');

INSERT INTO Employee (EmployeeID, Name, Salary, JoiningDate, ActiveStatus) VALUES (5, 'Vikram Singh', 45000, TO_DATE('2023-03-01', 'YYYY-MM-DD'), 'N');

SELECT *

FROM Employee

WHERE JoiningDate < TO_DATE('2023-01-01', 'YYYY-MM-DD');

UPDATE Employee

SET Salary = Salary * 1.10

WHERE Name = 'Amit Sharma';

SELECT *

FROM Employee

WHERE Name = 'Amit Sharma';

SELECT *

FROM Employee

WHERE ActiveStatus = 'Y';
```

6) aggregate Functions (on a single table: Create a Sales table with columns: SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, and SaleDate.) a) From the Sales table, calculate the total sales amount (SUM) generated in the month of February 2025. b) Find the average (AVG) billing amount from the Sales table to assess customer spending behavior. c)

Identify the minimum (MIN) quantity of products sold in any transaction using the Sales table. d) Determine the highest (MAX) discount applied on any sale using the Sales table. e) Use the COUNT function to find how many transactions were recorded in the Sales table for the product “Laptop”.

ANS

```
CREATE TABLE Sales (
   SaleID INT PRIMARY KEY,
   ProductID INT,
   ProductName VARCHAR2(100),
   Quantity INT,
   Discount DECIMAL(5,2),
   SaleAmount DECIMAL(10,2),
   SaleDate DATE
);


INSERT INTO Sales (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, SaleDate) VALUES (1, 101, 'Laptop', 2, 5.00, 1500.00, TO_DATE('2025-02-10', 'YYYY-MM-DD'));


INSERT INTO Sales (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, SaleDate) VALUES (2, 102, 'Smartphone', 1, 2.50, 800.00, TO_DATE('2025-02-15', 'YYYY-MM-DD'));


INSERT INTO Sales (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, SaleDate) VALUES (3, 103, 'Tablet', 3, 10.00, 900.00, TO_DATE('2025-01-20', 'YYYY-MM-DD'));


INSERT INTO Sales (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, SaleDate) VALUES (4, 101, 'Laptop', 1, 8.00, 750.00, TO_DATE('2025-03-05', 'YYYY-MM-DD'));


INSERT INTO Sales (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, SaleDate) VALUES (5, 104, 'Monitor', 2, 7.00, 400.00, TO_DATE('2025-02-22', 'YYYY-MM-DD'));


select * from Sales


SELECT SUM(SaleAmount) AS Total_Sales_February_2025
```

```
FROM Sales

WHERE SaleDate BETWEEN TO_DATE('2025-02-01', 'YYYY-MM-DD') AND TO_DATE('2025-02-28', 'YYYY-MM-DD');


SELECT AVG(SaleAmount) AS Average_Billing_Amount

FROM Sales;


SELECT MIN(Quantity) AS Minimum_Quantity_Sold

FROM Sales;


SELECT MAX(Discount) AS Maximum_Discount

FROM Sales;


SELECT COUNT(*) AS Laptop_Transactions

FROM Sales

WHERE ProductName = 'Laptop';
```

7) Constraints (on a single table: Employees) a) Create the Employees table with EmployeeID as PRIMARY KEY, Email as UNIQUE, and Salary with a CHECK (Salary > 10000) constraint. b) Add a NOT NULL constraint on the Name column in the Employees table and try inserting a record without the name. c) Add a DEFAULT value 'Active' to the Status column in Employees, and insert a record without specifying the status to verify the default. d) Insert a record into Employees where Salary is less than 10000 to test the CHECK constraint. e) Try inserting two employees with the same Email ID to verify the enforcement of the UNIQUE constraint.

ANS:

```
CREATE TABLE Employees (
   EmployeeID INT PRIMARY KEY,
   Name VARCHAR2(100),
   Email VARCHAR2(100) UNIQUE,
   Salary DECIMAL(10, 2) CHECK (Salary > 10000),
   Status VARCHAR2(20) DEFAULT 'Active'
);
ALTER TABLE Employees
MODIFY Name VARCHAR2(100) NOT NULL;
```

```
INSERT INTO Employees (EmployeeID, Email, Salary)
VALUES (1, 'john.doe@example.com', 12000);

INSERT INTO Employees (EmployeeID, Name, Email, Salary)
VALUES (2, 'John Doe', 'john.doe@example.com', 15000);
SELECT * FROM Employees WHERE EmployeeID = 2;

INSERT INTO Employees (EmployeeID, Name, Email, Salary)
VALUES (3, 'Jane Smith', 'jane.smith@example.com', 9000);

INSERT INTO Employees (EmployeeID, Name, Email, Salary)
VALUES (4, 'Mike Johnson', 'mike@example.com', 17000);

INSERT INTO Employees (EmployeeID, Name, Email, Salary)
VALUES (5, 'Michael J', 'mike@example.com', 18000);
```

8) DDL and DML Commands a) Use DDL commands to create a Library database and define a Books table with fields: BookID, Title, Author, Genre, and Price. b) Insert at least five sample records into the Books table using INSERT (DML) and verify them using a SELECT query. c) A new column PublicationYear needs to be added. Use ALTER TABLE to modify the existing table structure. d) Update the price of all books published before 2020 by increasing 10% using the UPDATE statement. e) Use DELETE to remove all books where the genre is 'Outdated Technology' and validate the change with a SELECT query.

ANS:

```
CREATE TABLE Book (
   BookID NUMBER PRIMARY KEY,
   Title VARCHAR2(100),
   Author VARCHAR2(100),
   Genre VARCHAR2(50),
   Price NUMBER(8,2)
);
```

```
INSERT INTO Book (BookID, Title, Author, Genre, Price) VALUES (1, 'The Silent Patient', 'Alex Michaelides', 'Thriller', 450.00);

INSERT INTO Book (BookID, Title, Author, Genre, Price) VALUES (2, 'Deep Work', 'Cal Newport', 'Productivity', 550.00);

INSERT INTO Book (BookID, Title, Author, Genre, Price) VALUES (3, 'Clean Code', 'Robert C. Martin', 'Programming', 650.00);

INSERT INTO Book (BookID, Title, Author, Genre, Price) VALUES (4, 'Outdated Tech Guide', 'John Doe', 'Outdated Technology', 300.00);

INSERT INTO Book (BookID, Title, Author, Genre, Price) VALUES (5, 'AI Superpowers', 'Kai-Fu Lee', 'Technology', 700.00);

SELECT * FROM Book;

ALTER TABLE Books

ADD PublicationYear NUMBER(4);

UPDATE Books SET PublicationYear = 2019 WHERE BookID = 1;

UPDATE Books SET PublicationYear = 2016 WHERE BookID = 2;

UPDATE Books SET PublicationYear = 2008 WHERE BookID = 3;

UPDATE Books SET PublicationYear = 2010 WHERE BookID = 4;

UPDATE Books SET PublicationYear = 2018 WHERE BookID = 5;

UPDATE Books

SET Price = Price * 1.10

WHERE PublicationYear < 2020;

DELETE FROM Books

WHERE Genre = 'Outdated Technology';

SELECT * FROM Books;
```

9) DDL and DML Commands (on a single table: Books) a) Create a table Books using DDL with fields: BookID, Title, Author, Price, and StockAvailable. b) Insert 5 book records into the Books table using the INSERT command. c) Modify the structure of Books table by adding a new column Genre using the ALTER TABLE command. d) Use the UPDATE command to increase the price of all books by RS 50 in the Books table. e) Delete all records from the Books table where StockAvailable is 0 using the DELETE command

ANS:

```
CREATE TABLE Books_1 (
   BookID NUMBER PRIMARY KEY,
   Title VARCHAR2(100),
   Author VARCHAR2(100),
   Price NUMBER(8,2),
   StockAvailable NUMBER
);
```

```
INSERT INTO Books_1 (BookID, Title, Author, Price, StockAvailable) VALUES (1, 'The Alchemist', 'Paulo Coelho', 300, 5);
```

```
INSERT INTO Books_1 (BookID, Title, Author, Price, StockAvailable) VALUES (2, 'Atomic Habits', 'James Clear', 450, 3);
```

```
INSERT INTO Books_1 (BookID, Title, Author, Price, StockAvailable) VALUES (3, 'Introduction to Algorithms', 'Thomas H. Cormen', 1200, 2);
```

```
INSERT INTO Books_1 (BookID, Title, Author, Price, StockAvailable) VALUES (4, 'The Great Gatsby', 'F. Scott Fitzgerald', 250, 0);
```

```
INSERT INTO Books_1 (BookID, Title, Author, Price, StockAvailable) VALUES (5, 'Zero to One', 'Peter Thiel', 500, 4);
```

```
ALTER TABLE Books_1
ADD Genre VARCHAR2(50)
```

```
UPDATE Books_1
SET Price = Price + 50;
```

```
DELETE FROM Books_1

WHERE StockAvailable = 0;

select * from Books_1
```

10) Analyze Sales Performance Using Aggregate Functions Create a Sales table with columns: SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, and SalesPerson.) a) Calculate the total quantity of products sold across all transactions in the Sales table. b) Find the average sale amount for transactions made in March 2025. c) Identify the product with the minimum sale quantity from the Sales table. d) Determine the maximum discount offered in February 2025. e) Count how many sales were made by each salesperson using GROUP BY SalesPerson

ANS:

```
CREATE TABLE Sales_0 (
   SaleID NUMBER PRIMARY KEY,
   ProductID NUMBER,
   ProductName VARCHAR2(100),
   Quantity NUMBER,
   Discount NUMBER(5,2),
   SaleAmount NUMBER(10,2),
   SalesPerson VARCHAR2(100),
   SaleDate DATE
);

INSERT INTO Sales_0 (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount,
SalesPerson, SaleDate)

VALUES (1, 101, 'Laptop', 5, 10, 250000, 'Amit', TO_DATE('2025-03-05', 'YYYY-MM-DD'));


INSERT INTO Sales_0 (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount,
SalesPerson, SaleDate)

VALUES (2, 102, 'Smartphone', 10, 5, 50000, 'Sneha', TO_DATE('2025-03-10', 'YYYY-MM-DD'));


INSERT INTO Sales_0 (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount,
SalesPerson, SaleDate)

VALUES (3, 103, 'Headphones', 2, 15, 3000, 'Amit', TO_DATE('2025-02-15', 'YYYY-MM-DD'));
```

```
INSERT INTO Sales_0 (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, SalesPerson, SaleDate)

VALUES (4, 104, 'Monitor', 1, 20, 15000, 'Sneha', TO_DATE('2025-02-28', 'YYYY-MM-DD'));


INSERT INTO Sales_0 (SaleID, ProductID, ProductName, Quantity, Discount, SaleAmount, SalesPerson, SaleDate)

VALUES (5, 105, 'Keyboard', 3, 0, 4500, 'Rahul', TO_DATE('2025-03-20', 'YYYY-MM-DD'));


SELECT SUM(Quantity) AS TotalQuantitySold

FROM Sales_0;


SELECT AVG(SaleAmount) AS AverageSaleMarch2025

FROM Sales_0

WHERE TO_CHAR(SaleDate, 'YYYY-MM') = '2025-03';


SELECT ProductName, Quantity

FROM Sales_0

WHERE Quantity = (SELECT MIN(Quantity) FROM Sales_0);


SELECT MAX(Discount) AS MaxDiscountFeb2025

FROM Sales_0

WHERE TO_CHAR(SaleDate, 'YYYY-MM') = '2025-02';


SELECT SalesPerson, COUNT(*) AS SalesCount

FROM Sales_0

GROUP BY SalesPerson;
```