

ORACLE LAB

BCA-DS-552

Manav Rachna International Institute of Research and Studies

School of Computer Applications

Department of Computer Applications

Submitted By	
Student Name	Aahan Chhabra
Roll No	22/FC/BCA(CS)/001
Programme	Bachelor of Computer Applications
Semester	5 th Semester
Section	D
Department	Computer Applications
Batch	2022-25
Submitted To	
Faculty Name	Mrs. Stuti Tandon

EXERCISE 1

AIM: Create the following table.

Customer

<u>Column name</u>	<u>Data type</u>	<u>Size</u>	<u>Constraint</u>
SID	Varchar2	4	Primary Key
First_Name	Char	20	
Last_name	Char	20	

Orders

<u>Column name</u>	<u>Data type</u>	<u>Size</u>	<u>Constraint</u>
Order_ID	Varchar2	4	Primary Key
Order_date	Char	20	
Customer_SID	Varchar2	20	Foreign Key
Amount	Number		Check > 20000

Output:

```
SQL Worksheet

1 v CREATE TABLE Customer
2   (
3     SID VARCHAR2(4) PRIMARY KEY,
4     First_Name CHAR(20),
5     Last_name CHAR(20)
6   );
7
8 v CREATE TABLE Orders
9   (
10    Order_ID VARCHAR2(4) PRIMARY KEY,
11    Order_date CHAR(20),
12    Customer_SID VARCHAR2(4),
13    Amount NUMBER CHECK (Amount > 20000),
14    FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)
15  );
16

Table created.

Table created.
```

EXERCISE 2

AIM: Insert 5 records for each table.

Output:

SQL Worksheet

```
1 INSERT INTO Customer VALUES ('C001', 'John', 'Doe');
2 INSERT INTO Customer VALUES ('C002', 'Jane', 'Smith');
3 INSERT INTO Customer VALUES ('C003', 'Alex', 'James');
4 INSERT INTO Customer VALUES ('C004', 'Chris', 'Evans');
5 INSERT INTO Customer VALUES ('C005', 'Emma', 'Watson');
6
7 INSERT INTO Orders VALUES ('0001', '2024-08-01', 'C001', 25000);
8 INSERT INTO Orders VALUES ('0002', '2024-08-02', 'C002', 22000);
9 INSERT INTO Orders VALUES ('0003', '2024-08-03', 'C003', 21000);
10 INSERT INTO Orders VALUES ('0004', '2024-08-04', 'C004', 30000);
11 INSERT INTO Orders VALUES ('0005', '2024-08-05', 'C005', 31000);
12
```

SQL Worksheet

SID	FIRST_NAME	LAST_NAME
C001	John	Doe
C002	Jane	Smith
C003	Alex	James
C004	Chris	Evans
C005	Emma	Watson

Download CSV

5 rows selected.

ORDER_ID	ORDER_DATE	CUSTOMER_SID	AMOUNT
0001	2024-08-01	C001	25000
0002	2024-08-02	C002	22000
0003	2024-08-03	C003	21000
0004	2024-08-04	C004	30000
0005	2024-08-05	C005	31000

Download CSV

EXERCISE 3

AIM: Customer SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.

Output:

```
CREATE TABLE Orders
(
  Order_ID VARCHAR2(4) PRIMARY KEY,
  Order_date CHAR(20),
  Customer_SID VARCHAR2(4),
  Amount NUMBER CHECK (Amount > 20000),
  FOREIGN KEY (Customer_SID) REFERENCES Customer(SID)
);
```

EXERCISE 4

AIM: List the details of the customers along with the amount.

Output:

SQL Worksheet

```
1 v SELECT SID, First_Name, Last_name, Amount
2 FROM Customer
3 JOIN Orders ON Customer.SID = Orders.Customer_SID;
4
```

SID	FIRST_NAME	LAST_NAME	AMOUNT
C001	John	Doe	25000
C002	Jane	Smith	22000
C003	Alex	James	21000
C004	Chris	Evans	30000
C005	Emma	Watson	31000

[Download CSV](#)

EXERCISE 5

AIM: List the customers whose names end with “s”.

Output:

```
SQL Worksheet

1 select * from Customer where trim(last_name) like '%s';
2
```

SID	FIRST_NAME	LAST_NAME
C003	Alex	James
C004	Chris	Evans

Download CSV

2 rows selected.

EXERCISE 6

AIM: List the orders where amount is between 21000 and 30000

Output:

```
SQL Worksheet

1 select * from Orders where Amount between 21000 and 30000;
2
```

ORDER_ID	ORDER_DATE	CUSTOMER_SID	AMOUNT
0001	2024-08-01	C001	25000
0002	2024-08-02	C002	22000
0003	2024-08-03	C003	21000
0004	2024-08-04	C004	30000

Download CSV

4 rows selected.

EXERCISE 7

AIM: List the orders where amount is increased by 500 and replace with name “new amount”.

Output:

```
SQL Worksheet

1  update Orders set Amount = Amount + 500;
2
3  select Order_ID, Amount as "New Amount" from Orders;
4
```

5 row(s) updated.

ORDER_ID	New Amount
0001	25500
0002	22500
0003	21500
0004	30500
0005	31500

Download CSV

EXERCISE 8

AIM: Display the order_id and total amount of orders.

Output:

```
SQL Worksheet

1 v SELECT Order_ID, SUM(Amount) AS Total_Amount
2 FROM Orders
3 GROUP BY Order_ID;
4
```

ORDER_ID	TOTAL_AMOUNT
0001	25500
0002	22500
0003	21500
0004	30500
0005	31500

[Download CSV](#)

5 rows selected.

EXERCISE 9

AIM: Calculate the total amount of orders that has more than 15000.

Output:

```
SQL Worksheet

1 select sum(Amount) as Total_Amount from Orders where Amount > 15000;
2
```

TOTAL_AMOUNT
131500
Download CSV

EXERCISE 10

AIM: Display all the string functions used in SQL.

Output:

```
SELECT
  LOWER('ORACLE') AS "Lowercase",    -- Converts string to lowercase
  UPPER('oracle') AS "Uppercase",    -- Converts string to uppercase
  SUBSTR('ORACLE', 2, 3) AS "Substring", -- Extracts substring
  LENGTH('ORACLE') AS "Length",      -- Returns length of string
  INSTR('ORACLE', 'A') AS "Position", -- Returns position of a character
  LPAD('123', 5, '0') AS "Left Padding", -- Pads a string on the left
  RPAD('123', 5, '0') AS "Right Padding", -- Pads a string on the right
  TRIM('O' FROM 'ORACLE') AS "Trimmed" -- Trims a specified character
FROM DUAL;
```

EXERCISE 11

AIM: Create the following tables.

Student

<u>Column_name</u>	<u>Data type</u>	<u>Size</u>	<u>Constraint</u>
RollNo	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

Student1

<u>Column_name</u>	<u>Data type</u>	<u>Size</u>	<u>Constraint</u>
R_No	Varchar2	20	Primary Key
Name	Char	20	
Class	Varchar2	20	
Marks	Number	6,2	

Output:

```
SQL Worksheet

1 v create table Student
2 (
3     RollNo varchar(20) primary key,
4     Name char(20),
5     Class varchar(20),
6     Marks number(6,2)
7 );
8
9 v create table Student1
10 (
11     R_No varchar(20) primary key,
12     Name char(20),
13     Class varchar(20),
14     Marks number(6,2)
15 );

Table created.

Table created.
```

1. Select with Arithmetic Operations

SELECT StudentID, FirstName, Age, Age + 1 AS NextYearAge FROM Students;

StudentID	FirstName	Age	NextYearAge
1	John	20	21
2	Jane	22	23
3	Michael	21	22
4	Emily	19	20
5	Anna	23	24

5. Primary and Foreign Key Relationships

1. Create Table with Foreign Key

```
CREATE TABLE Advisors (  
    AdvisorID INT PRIMARY KEY,  
    AdvisorName VARCHAR(100),  
    StudentID INT,  
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID)  
);
```

Advisors

AdvisorID	AdvisorName	StudentID
empty		

2. Insert Data into Table with Foreign Key

INSERT INTO Advisors (AdvisorID, AdvisorName, StudentID) VALUES (1, 'Dr. Smith', 2);

Advisors

AdvisorID	AdvisorName	StudentID
1	Dr. Smith	2

6. Join Operations

1. Inner Join

```
SELECT Students.FirstName, Students.LastName, Courses.CourseName  
FROM Students  
JOIN Enrollments ON Students.StudentID = Enrollments.StudentID  
JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```

FirstName	LastName	CourseName
John	Doe	Introduction to Programming
John	Doe	Calculus I
Jane	Smith	General Physics
Michael	Johnson	Introduction to Programming
Emily	Davis	Organic Chemistry

2. Left Join

```
SELECT Students.FirstName, Students.LastName, Enrollments.Grade  
FROM Students  
LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;
```

FirstName	LastName	Grade
John	Doe	A
John	Doe	B
Jane	Smith	A
Michael	Johnson	C
Emily	Davis	B
Anna	Taylor	

3. Right Join

```
SELECT Students.FirstName, Students.LastName, Enrollments.Grade
FROM Students
```

```
RIGHT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;
```

Output: Displays all enrollments and the respective student details, if available.

1. Step 4: All Queries

1. Create Students Table

```
CREATE TABLE Students (
  StudentID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Age INT,
  Major VARCHAR(50)
);
```

Students				
StudentID	FirstName	LastName	Age	Major
empty				

2. Create Courses Table

```
CREATE TABLE Courses (
  CourseID INT PRIMARY KEY,
  CourseName VARCHAR(100),
  Credits INT
);
```

Courses		
CourseID	CourseName	Credits
empty		

3. Create Enrollments Table

```
CREATE TABLE Enrollments (
  EnrollmentID INT PRIMARY KEY,
  StudentID INT,
  CourseID INT,
  Grade CHAR(1),
  FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
  FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

Enrollments

EnrollmentID	StudentID	CourseID	Grade
empty			

4. Insert Data into Students Table

INSERT INTO Students (StudentID, FirstName, LastName, Age, Major) VALUES (1, 'John', 'Doe', 20, 'Computer Science');

Students

StudentID	FirstName	LastName	Age	Major
1	John	Doe	20	Computer Science

5. Insert Data into Courses Table

INSERT INTO Courses (CourseID, CourseName, Credits) VALUES (101, 'Introduction to Programming', 4);

Courses

CourseID	CourseName	Credits
101	Introduction to Programming	4

6. Insert Data into Enrollments Table

INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (1, 1, 101, 'A');

Enrollments

EnrollmentID	StudentID	CourseID	Grade
1	1	101	A

7. Select All Students

SELECT * FROM Students;

StudentID	FirstName	LastName	Age	Major
1	John	Doe	20	Computer Science

8. Select all Courses

SELECT * FROM Courses;

CourseID	CourseName	Credits
101	Introduction to Programming	4

9. Select All Enrollments

SELECT * FROM Enrollments;

EnrollmentID	StudentID	CourseID	Grade
1	1	101	A

10. Select Students Older than 19

SELECT * FROM Students WHERE Age > 19;

StudentID	FirstName	LastName	Age	Major
1	John	Doe	20	Computer Science

11. Update Student Major

UPDATE Students SET Major = 'Software Engineering' WHERE StudentID = 1;

Students				
StudentID	FirstName	LastName	Age	Major
1	John	Doe	20	Software Engineering

12. Delete a Student Record

DELETE FROM Students WHERE StudentID = 4;

Output: Emily Davis' record deleted.

13. Create Departments Table

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

Departments	
DepartmentID	DepartmentName
empty	

14. Add Column to Students Table

ALTER TABLE Students ADD Email VARCHAR(100);

Students					
StudentID	FirstName	LastName	Age	Major	Email
1	John	Doe	20	Software Engineering	

15. Drop Departments Table

DROP TABLE Departments;

Output: Departments table dropped.

16. Insert New Student with Email

```
INSERT INTO Students (StudentID, FirstName, LastName, Age, Major, Email) VALUES (5, 'Anna', 'Taylor', 23, 'Biology', 'anna.taylor@example.com');
```

Students					
StudentID	FirstName	LastName	Age	Major	Email
1	John	Doe	20	Software Engineering	
5	Anna	Taylor	23	Biology	anna.taylor@example.com

17. Select Students with Age Arithmetic Operation

SELECT StudentID, FirstName, Age, Age + 1 AS NextYearAge FROM Students;

StudentID	FirstName	Age	NextYearAge
1	John	20	21
5	Anna	23	24

18. Create Advisors Table with Foreign Key

```
CREATE TABLE Advisors (
  AdvisorID INT PRIMARY KEY,
  AdvisorName VARCHAR(100),
  StudentID INT,
  FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
);
```

Advisors

AdvisorID	AdvisorName	StudentID
empty		

19. Insert Data into Advisors Table

```
INSERT INTO Advisors (AdvisorID, AdvisorName, StudentID) VALUES (1, 'Dr. Smith', 2);
```

20. Inner Join Students and Enrollments

```
SELECT Students.FirstName, Students.LastName, Courses.CourseName
FROM Students
JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```

FirstName	LastName	CourseName
John	Doe	Introduction to Programming

21. Left Join Students and Enrollments

```
SELECT Students.FirstName, Students.LastName, Enrollments.Grade
FROM Students
LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;
```

FirstName	LastName	Grade
John	Doe	A
Anna	Taylor	

22. Right Join Students and Enrollments

```
SELECT Students.FirstName, Students.LastName, Enrollments.Grade
FROM Students
```


RIGHT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;
Output: Displays all enrollments and the respective student details, if available.

23. Count Number of Students

```
SELECT COUNT(*) AS NumberOfStudents FROM Students;
```

NumberOfStudents
2

24. Select Distinct Majors

```
SELECT AVG(Age) AS AverageAge FROM Students;
```

Major
Software Engineering
Biology

25. Select Average Age of Students

```
SELECT AVG(Age) AS AverageAge FROM Students;
```

AverageAge
21.5

26. Select Sum of Credits

```
SELECT SUM(Credits) AS TotalCredits FROM Courses;
```

TotalCredits
4

27. Select Students Grouped by Major

```
SELECT Major, COUNT(*) AS NumberOfStudents FROM Students GROUP BY Major;
```

Major	NumberOfStudents
Biology	1
Software Engineering	1

28. Select Students with a Specific Major

```
SELECT * FROM Students WHERE Major = 'Biology';
```

StudentID	FirstName	LastName	Age	Major	Email
5	Anna	Taylor	23	Biology	anna.taylor@example.com

29. Select Students with Age Between 20 and 22

```
SELECT * FROM Students WHERE Age BETWEEN 20 AND 22;
```

StudentID	FirstName	LastName	Age	Major	Email
1	John	Doe	20	Software Engineering	

30. Select Students with Names Starting with 'J'

```
SELECT * FROM Students WHERE FirstName LIKE 'J%';
```

StudentID	FirstName	LastName	Age	Major	Email
1	John	Doe	20	Software Engineering	

31. Select Students in Ascending Order of Age

SELECT * FROM Students ORDER BY Age ASC;

StudentID	FirstName	LastName	Age	Major	Email
1	John	Doe	20	Software Engineering	
5	Anna	Taylor	23	Biology	anna.taylor@example.com

32. Select Students in Descending Order of Last Name

SELECT * FROM Students ORDER BY LastName DESC;

StudentID	FirstName	LastName	Age	Major	Email
5	Anna	Taylor	23	Biology	anna.taylor@example.com
1	John	Doe	20	Software Engineering	

33. Select Top 3 Oldest Students

SELECT * FROM Students ORDER BY Age DESC LIMIT 3;

StudentID	FirstName	LastName	Age	Major	Email
5	Anna	Taylor	23	Biology	anna.taylor@example.com
1	John	Doe	20	Software Engineering	

34. Update Course Credits

UPDATE Courses SET Credits = 5 WHERE CourseID = 101;

Courses

CourseID	CourseName	Credits
101	Introduction to Programming	5

35. Delete a Course Record

DELETE FROM Courses WHERE CourseID = 104;

Output: Deletes course with ID 104.

36. Select Students Enrolled in a Specific Course

SELECT Students.FirstName, Students.LastName

FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

WHERE Enrollments.CourseID = 101;

FirstName	LastName
John	Doe

37. Select Courses with More Than 3 Credits

```
SELECT * FROM Courses WHERE Credits > 3;
```

CourseID	CourseName	Credits
101	Introduction to Programming	5

38. Select Students with Null Email

```
SELECT * FROM Students WHERE Email IS NULL;
```

Output					
StudentID	FirstName	LastName	Age	Major	Email
1	John	Doe	20	Software Engineering	

39. Select Students with Non-Null Email

```
SELECT * FROM Students WHERE Email IS NOT NULL;
```

StudentID	FirstName	LastName	Age	Major	Email
5	Anna	Taylor	23	Biology	anna.taylor@example.com

40. Select Students Who Have Taken Multiple Courses

```
SELECT Students.FirstName, Students.LastName
FROM Students
JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
GROUP BY Students.StudentID, Students.FirstName, Students.LastName
HAVING COUNT(Enrollments.CourseID) > 1;
```

SQL query successfully executed. However, the result set is empty.

41. Select Enrollments with Grades A or B

```
SELECT * FROM Enrollments WHERE Grade IN ('A', 'B');
```

EnrollmentID	StudentID	CourseID	Grade
1	1	101	A

42. Select Students with Age Not Between 18 and 22

```
SELECT * FROM Students WHERE Age NOT BETWEEN 18 AND 22;
```

StudentID	FirstName	LastName	Age	Major	Email
5	Anna	Taylor	23	Biology	anna.taylor@example.com

43. Select Enrollments in Ascending Order of Grade

```
SELECT * FROM Enrollments ORDER BY Grade ASC;
```

EnrollmentID	StudentID	CourseID	Grade
1	1	101	A

44. Select Course Names and Credits

```
SELECT CourseName, Credits FROM Courses;
```

CourseName	Credits
Introduction to Programming	5

45. Select Students Enrolled in Specific Course with Grade A

```
SELECT Students.FirstName, Students.LastName
FROM Students
JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
WHERE Enrollments.CourseID = 101 AND Enrollments.Grade = 'A';
```

FirstName	LastName
John	Doe

46. Select Students Grouped by Major and Age

```
SELECT Major, Age, COUNT(*) AS NumberOfStudents
FROM Students
GROUP BY Major, Age;
```

Major	Age	NumberOfStudents
Biology	23	1
Software Engineering	20	1

47. Select Students and Their Enrollments

```
SELECT Students.FirstName, Students.LastName, Enrollments.CourseID
FROM Students
JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;
```

FirstName	LastName	CourseID
John	Doe	101

48. Select Course Names with Student Count

```
SELECT Courses.CourseName, COUNT(Enrollments.StudentID) AS StudentCount
FROM Courses
JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID
GROUP BY Courses.CourseName;
```

CourseName	StudentCount
Introduction to Programming	1

49. Select Advisors and Their Students

```
SELECT Advisors.AdvisorName, Students.FirstName, Students.LastName
FROM Advisors
```

JOIN Students ON Advisors.StudentID = Students.StudentID;

SQL query successfully executed. However, the result set is empty.

50. Select Students Who Haven't Taken Any Courses

```
SELECT Students.FirstName, Students.LastName  
FROM Students  
LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID  
WHERE Enrollments.StudentID IS NULL;
```

FirstName	LastName
Anna	Taylor