

7/08/2025.

RANKA

DATE / /

PAGE

DBMS Lab

Name: Yashaswini

SAP ID: 590011386

* Assessment :- 0 - 10 labs

(40 M)



Attend. Performance

5M (avg) 5M

78

9.5

* lab file :- 10 marks (10 M)

* lab exam :- 50 Marks (50 M)



Mid End

50M 50M

Lab-0

Installation and Schema Design

Oracle Apex :- Interface.

Step-1 : Type Oracle Apex in google search engine.

Step-2 : Sign In the account in Oracle Apex.

Step-3 : Request a workspace.

Step-4 : → email

→ unique workspace name (YA.DBMS-25)

Step-5 : Yes / Yes

Step - 6

- Once you logged in
- SQL workshop.
- SQL commands.

Ques: Design the schema:

- 1) for (student / faculty / employee)
- 2) Identify the (job description) role (Timetable).
- 3) How they are (communicating with each other).

* Student:

Name	SAP ID	Marks	Att.	Course	Faculty name
var. char (20)	Primary key	int	char	char	var. char

* Faculty:

Name	ID	Working days	Course	No. of class
var. char (20)	var. char (20)	int	char	int

* Employee:

Name	SAP ID	Marks	Salary	ID	Working days
var. char (20)	Primary key	var. char (20)	int	var. char (20)	int

Name: Yashaswini
SAPID: 590011386

Batch: 2.

Date: 14/08/25.

Lab-1

4/8/25

RANKA

DATE / /

PAGE

Implementation of "DDL" and "DML"

Lab performance questions:

Ques 1: what is the difference between "Truncate" and "Delete"?

Truncate

- 1) Removes all rows from a table quickly.
- 2) Cannot include a WHERE clause.
- 3) It is a DDL command.
- 4) Resets Table storage space back to initial state.
- 5) Cannot be rolled back in many RDBMS.
- 6) Faster as it does not log row-by-row deletion.

Delete

- 1) Removes rows from a table one at a time based on where cond".
- 2) Can include a WHERE clause.
- 3) It is a DML command.
- 4) Does not reset storage.
- 5) Can be rolled back before commit.
- 6) Slower as each row deletion is logged.

Ques 2: what is the difference b/w "Truncate" and "DROP"?

Truncate

- 1) Remove all rows from a table but keeps the table structure for future use.

DROP

- 1) Removes the entire table, including its structure, data, index, and constraint.

- 2) It is a DDL command.
- 3) Table can still be used after truncation.
- 2) It is also a DDL command.
- 3) Table must be recreated if you need it again.

Quest: Done

Query: Done.

Quest: Done.

Quest: Done.

→ Quest: Done.

Name: Yashavan
SAP: 590011386
Date: 21-08-2025

Yashavan
21-08-2025

RANKA

DATE / /
PAGE / /

Lab-2

To understand the concepts of
constraints

Ques: Done.

* Create.

* Insert.

* Alter.

* Alter, Insert, Add.

* Alter, Add constraints.

* Create.

* Alter, Insert.

* Alter, Insert.

* Create.

* Select, From.

* Update.

* Delete.

* Alter.

* Select.

Name: Yashaswini
SAP: 590011386
Date: 28/08/2025

RANKA

DATE / /

PAGE

tab-3

use of SQL sub-query

create → 11 queries.

desc → 11 queries.

Insert Into → 29 queries.

Select → 3 queries.

Insert Into → 1 query.

Total → 55 queries.

Select → 41 queries.

where → 25 queries.

Insert into → 2 queries.

update → 2 queries.

set → 2 queries.

Delete → 1 query.

~~DELETE~~ →

or → 4 queries.

Groupby → 4 queries.

where exists → 1 query.

where not exists → 1 query.

join → 1 query.

commit → 2 queries.

And → 3 queries.

Total → 144 queries.

Name: Yashavant
SAP: 59001386
Date: 4/09/2026

RANKA
DATE / /
PAGE

Lab-4

To implement the use of
In-built functions.

Create Table

→ 2

Drop

→ 2

Insert Into → 3

values → 3

Select → 49

From → 49

Where → 3

Group by → 5

Order by → 4

* Rownum → 2

Count() → 1

Sum() → 1

Avg() → 1

Min() → 1

Max() → 1

* Length() → 1

LengthB() → 1

RPad() → 2

LPad() → 1

LTrim() → 1

RTrim() → 1

* Replace() → 1

Substr() → 6

Instr() → 2

Lower() → 1

Hex() → 1

Ascii() → 1

To_Char() → 3

To_Number() → 1

Count() → 1

Sum() → 1

Avg() → 1

Min() → 2

Max() → 2

Rownum → 2

Listagg → 4+1

Concat_WT() → 1

Length() → 2+2

ELT() → 1

Export_Set() → 1

Field() → 1

Instr() → 2+2

To_Char() → 1

Base64() → 1

Rawtohex() → 1

Substr() → 3+2+1

Lower() → 2

Lpad() → 1

Ltrim() → 1

LengthB() → 1

To_Char(Trunc()) → 1

To_Char(Trunc(mod)) → 1

To_Char(mod) → 1

Ascii(Substr()) → 1

RPad() → 2

Replace() → 1

RTrim() → 1

Soundex() → 1

DBMS_RANDOM.value() → 1

groupby → 5

RANKA

ROUND() → 1

DATE

POWER() → 2

PAGE

FROM base 6 * () → 1

ABS() → 1

ELT() → 1

ACOS() → 3+2

EXPORT SET() →

SIGN() → 1

FIELDN() → 1

SIN() → 2

FIND_IN_SET() →

SQRT() → 1

FORMAT() → 1

TAN() → 1

MAXF_SET() →

TRUNC() → 2

GROUP_CONCAT() → 4

ASIN() → 1

LISTAGS() → 2

ATAN2() → 1

LIKE → 1

ATAN() → 1

DISTINCT → 2

CEIL() → 2+1

WITHIN GROUP (order by) → 1

COS() → 2

SOUNDEX() → 1

EXP() → 1

CONNECT_BY → 1

FLOOR() → 1

PRIOR → 1

GREATEST() → 1

DBMS_RANDOM.VALUE → 1

LEAST() → 1

Total queries → 170

LN() → 1

LOG() → 2+1

MOD() → 1

Total queries → 86 queries.

Name: Yashasvani
SAP: 590011386
Date: 11-09-2025

RANKA

DATE / /

PAGE

Lab - 5

Understanding of SQL

1)

Department
Name
Code
Office - no.
Phone - no.

Student

Name

Student - no.

Security no.

Current - add.

Current - ph. no.

Permanent - add.

Permanent - ph. no.

Birth date

Sex

Class

Major - dept.

Minor - dept

Degree

offers

course

Name

Description

Semester - hours

Level

Teaches

Grade Report

student

Letter - grade

Numeric - grade

Section

Instructor

* Student :

Attributes :- Name, Address, Roll No.

Student - no. (PK)

SSN (unique)

Name

Birth - date

Sex

Class

current add.; current phone

permanent add.; permanent phone

Major Dept. (FK)

Minor Dept. (FK)

degree.

* Department :

Attributes :-

code (PK)

name (unique)

Office - no

office - phone

* Course :

Attributes :-

course - no. (PK)

course - name

description

semester - hours

level

* Grade - report :

Attributes :-

student - no. (FK)

letter - grade

numeric grade. (0,1,2,3,4).

relationships

Student - Department.

A student has one major and optional minor department.

Student (Major Dept.) → Department (1:M)

Student (Minor - Dept) → Department (0:M).

Department - course

Department → Course (1:M)

Course - Section

Course → Section (1:M)

student → Grade - report (1:M)

Section → Grade - report (1:M).

2)

Dept-Employee
Comp-ID
Name

manages

Department

Dep-ID
No. of - project
Dep-name
Location

Head Employee
Dept.
H-ID
Name

supervises

Employee
DOB
SSN
Salary
Address

Projects
Name
No. of hours

Dependencies
E-name
Sec
DOB

Depends

Name: Yashaswini

SAP: 590011386

B-2

Date: 16/10/25

① Student info \bowtie student marks

(u10) ID	name	Dept.	subject	Marks
11	x	p	Maths	20
12	y	q	DAA	22
13	(u2)	r	Discrete	18
14	w	s	OS	14

② Student info \bowtie student marks
ID = Student marks

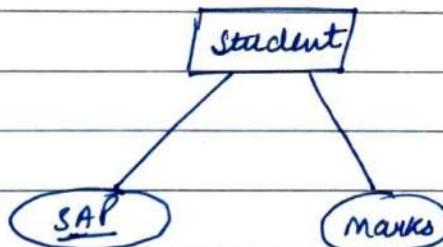
~~Name: Yashaswini~~~~SAP : 590011386~~Lab-8My SQL Views~~Ques:~~~~Create a small information~~

ER diagram of student information & write relational calculus.

o (Student ⋈I Marks)

student · ID = marks · ID.

Student	
SAP	Marks
A	20
B	50



30 queries run.

~~name: Yashaswini~~
~~SAP: 590011386~~

RANKA

DATE / /

PAGE

Lab-9

Indexing

- * create database
- * create table
- * Insert data
- * create indexes
- * Show
- * Select where
- * create index
- * select , from, where
- * select , from , index use
- * Show index
- * Drop index

11 queries.

Name: Yashaswini
SAP : 590011386

DATE	..
PAGE	..

~~29/11~~

PL / SQL Commands

- * CONSTANT
- * PI
- * Boolean flag
- * MOD
- * SORT
- * length
- * Substr
- * Greatest
- * Least
- * POWER

All done

NAME: Yashasvani

SAP ID: 590011386

DBMS LAB

INDEX

S. No.	Experiment	Date of Submission	Signature
1	Installing Oracle Apex	07/08/2025	W/ 21/8
2	To implement DDL and DML commands.	14/08/2025	W/ 24/8
3	To Implement and Understand the Concept Of Constraints	21/08/2025	W/ 21/8
4	To implement and use SQL Sub-Query	28/08/2025	W/ 28/8
5	To implement and use Inbuilt Functions	04/09/2025	W/ 4/9
6	Understanding of SQL	11/09/2025	W/ 11/9
7	To understand the use of SQL Inbuilt functions.	07/10/2025	W/ 21/10
8	To understand the use of group by and having clause and execute the SQL commands using JOIN	16/10/2025	W/ 16/10
9	MySQL Views	30/10/2025	W/ 30/10
10	Indexing	06/11/2025	W/ 6/11
11	PL/SQL Commands	23/11/2025	W/ 23/11

EXPERIMENT-3

Lab 3: To implement and use SQL Sub-Query

Objective: To understand the use of SQL Subquery.

Lab Performance Questions

Tables:

1. Student_Details
 - Columns: Student_RollNo, Stu_Name, Stu_Marks, Stu_City
2. Faculty_Details
 - Columns: Faculty_ID, Name, Dept_ID, Address
3. Department
 - Columns: Dept_ID, Faculty_ID, Dept_Name
4. Old_Employee
 - Columns: Emp_ID, Emp_Name, Emp_Salary, Address
5. New_Employee
 - Columns: Emp_ID, Emp_Name, Emp_Salary, Address
6. Employee_Details
 - Columns: Emp_ID, Emp_Name, Emp_Salary, Dept_ID
7. Student
 - Columns: Student_ID, Name, City
8. Student2
 - Columns: Student_ID, City
9. Orders
 - Columns: Order_ID, Cust_ID, Order_Date
10. Sales
 - Columns: Product_Category, Sales_Amount
11. Top_Students
 - Columns: Student_ID, Top_Marks

SQL Subquery Questions:

1. Basic **SELECT** Subquery
 - Retrieve the names and marks of students whose marks are greater than the average marks.
2. **IN Operator** Subquery
 - Fetch the names and addresses of faculties who belong to departments in either 'Noida' or 'Gurgaon.'
3. **INSERT** with Subquery
 - Insert the details of employees from the "Old_Employee" table into the "New_Employee" table, considering only those with a salary greater than 40000.
4. **UPDATE** with Subquery:
 - Increase the salary of employees by 10% for those whose department grade is 'A.'
5. **DELETE** with Subquery:
 - Delete the records of employees whose department grade is 'C.'

6. Subquery in WHERE Clause - NOT IN

- Retrieve the names of students who do not belong to the city 'Los Angeles' based on data from the "Student2" table.

7. Subquery in FROM Clause:

- Use a subquery in the FROM clause to create a derived table showing the maximum, minimum, and average number of items for each order.

8. Correlated Subquery

- Fetch the names, cities, and incomes of employees whose income is higher than the average income of employees in their respective cities.

9. EXISTS Operator

- Retrieve the name, occupation, and age of customers who have placed at least one order using the EXISTS operator.

10. ROW Subquery

- Retrieve all columns for customers whose (cust_id, occupation) matches any row of (order_id, order_date) from the "Orders" table.

11. Subquery with ALL Operator

- Find customers whose cust_id is greater than all cust_ids in the "Orders" table.

12. Nested Subqueries

- Write an SQL query using a subquery within another subquery to retrieve names and salaries of employees who earn more than the average salary of employees in the "IT" department.

13. Subquery in HAVING Clause

- Using the "Sales" table, find the total sales amount for each product category, displaying only those categories where the total sales amount is greater than the average total sales amount.

14. Subquery with GROUP BY

- Retrieve department names and the count of employees in each department, filtering out departments where the count is less than 5.

15. Multiple Subqueries in a Single Query

- Write an SQL query involving multiple subqueries to retrieve information about employees based on various conditions.

16. Subquery with BETWEEN Operators

- Find the names and marks of students who scored between 80 and 90.

17. Subquery in INSERT Statement

- Insert the details of employees from the "Old_Employee" table into the "New_Employee" table, considering only those with a salary greater than the average salary of all employees.

18. Subquery with ANY Operator

- Retrieve the names of students who scored more marks than ANY student from the "Top_Students" table.

19. Subquery in UPDATE Statement

- Update the salary of employees by 5% for those whose department is 'HR' and the salary is less than the average salary of HR department employees.

20. Subquery with NOT EXISTS

- Fetch the names and ages of customers who have NOT placed any orders based on the data from the "Orders" table.

MySQL Subquery

A subquery in MySQL is nested into another SQL query and embedded with SELECT, INSERT, UPDATE, or DELETE statements and the various operators. We can also nest the subquery with another subquery. A subquery is known as the **inner query**, and the query that contains a subquery is known as the **outer query**. The inner query executed first gives the result to the outer query, and then the main/outer query will be performed. MySQL allows us to use subquery anywhere but must be closed within parenthesis. All subquery forms and operations supported by the SQL standard will also be supported in MySQL.

The following are the rules to use subqueries:

- Subqueries should always be in **parentheses**.
- If the main query does not have multiple columns for the subquery, then a subquery can have only one column in the SELECT command.
- We can use various comparison operators with the subquery, such as >, <, =, IN, ANY, SOME, and ALL. A multiple-row operator is very useful when the subquery returns more than one row.
- We cannot use the **ORDER BY** clause in a subquery, although it can be used inside the main query.
- If we use a subquery in a **set function**, it cannot be immediately enclosed in a set function.

The following are the advantages of using subqueries:

- The subqueries make the queries in a structured form that allows us to isolate each part of a statement.
- The subqueries provide alternative ways to query the data from the table; otherwise, we need to use complex joins and unions.
- The subqueries are more readable than complex join or union statements.

SQL Subquery

The Subquery or Inner query is an SQL query placed inside another SQL query. It is embedded in the HAVING or WHERE clause of the SQL statements.

Following are the important rules which the SQL Subquery must follow:

1. The SQL subqueries can be used with the following statements along with the SQL expression operators:
 - SELECT statement,
 - UPDATE statement,
 - INSERT statement, and
 - DELETE statement.
2. The subqueries in SQL are always enclosed in the parenthesis and placed on the right side of the SQL operators.
3. We cannot use the ORDER BY clause in the subquery. But, we can use the GROUP BY clause, which performs the same function as the ORDER BY clause.
4. If the subquery returns more than one record, we must use the multiple value operators before the subquery.
5. We can use the BETWEEN operator within the subquery but not with the subquery.

Subquery with SELECT statement

In SQL, the SELECT statement uses inner or nested queries most frequently. The syntax of the subquery with the SELECT statement is described in the following block:

```
SELECT Column_Name1, Column_Name2, ...., Column_NameN  
FROM Table_Name WHERE Column_Name Comparison_Operator  
( SELECT Column_Name1, Column_Name2, ...., Column_NameN  
FROM Table_Name WHERE condition;
```

Examples of Subquery with the SELECT Statement

Example 1: This example uses the **Greater than comparison operator** with the subquery.

Let's take the table **Student_Details**, which contains **Student_RollNo.**, **Stu_Name**, **Stu_Marks**, and **Stu_City** column.

Student_RollNo.	Stu_Name	Stu_Marks	Stu_City
1001	Akhil	85	Agra
1002	Balram	78	Delhi
1003	Bheem	87	Gurgaon
1004	Chetan	95	Noida
1005	Diksha	99	Agra
1006	Raman	90	Ghaziabad
1007	Sheetal	68	Delhi

The following SQL query returns the record of those students whose marks are greater than the average total marks:

```
SELECT * FROM Student_Details WHERE Stu_Marks > ( SELECT AVG(Stu_Marks) FROM Student_Details );
```

Output:

Student_RollNo.	Stu_Name	Stu_Marks	Stu_City
1003	Bheem	87	Gurgaon
1004	Chetan	95	Noida
1005	Diksha	99	Agra
1006	Raman	90	Ghaziabad

Example 2: This example uses the **IN operator** with the subquery.

Let's take the following two tables named **Faculty_Details** and **Department** tables. The **Faculty_Details** table contains the ID, Name, Dept_ID, and address of faculties. The Department table contains the Dept_ID, Faculty_ID, and Dept_Name.

Faculty_ID	Name	Dept_ID	Address
101	Bheem	1	Gurgaon
102	Chetan	2	Noida
103	Diksha	NULL	Agra
104	Raman	4	Ghaziabad
105	Yatin	3	Noida
106	Anuj	NULL	Agra
107	Rakes	5	Gurgaon

Dept_ID	Faculty_ID	Dept_Name
1	101	BCA
2	102	B.Tech
3	105	BBA
4	104	MBA
5	107	MCA

```
SELECT * FROM Department WHERE Faculty_ID IN (
    SELECT Faculty_ID FROM Faculty WHERE City = 'Noida' OR City = 'Gurgaon' );
```

Output:

Dept_ID	Faculty_ID	Dept_Name
1	101	BCA
2	102	B.Tech
3	105	BBA
5	107	MCA

Subquery with the INSERT statement

We can also use the subqueries and nested queries with the INSERT statement in Structured Query Language. We can insert the subquery results into the table of the outer query. The syntax of the subquery with the INSERT statement is described in the following block:

INSERT INTO Table_Name SELECT * FROM Table_Name WHERE Column_Name Operator (Subquery);

Examples of Subquery with the INSERT Statement

Example1: This example inserts the record of one table into another table using subquery with **WHERE clause**.

Let's take Old_Employee and New_Employee tables. The Old_Employee and New_Employee table contain the same number of columns. But, both the tables contain different records.

Table: Old_Employee

Emp_ID	Emp_Name	Emp_Salary	Address
1001	Akhil	50000	Agra
1002	Balram	25000	Delhi
1003	Bheem	45000	Gurgaon
1004	Chetan	60000	Noida
1005	Diksha	30000	Agra
1006	Raman	50000	Ghaziabad
1007	Sheetal	35000	Delhi

Table: New_Employee

Emp_ID	Emp_Name	Emp_Salary	Address
1008	Sumit	50000	Agra
1009	Akash	55000	Delhi
1010	Devansh	65000	Gurgaon

The New_Employee contains the details of new employees. If you want to move the details of those employees whose salary is greater than 40000 from the Old_Employee table to the New_Employee table. Then for this issue, you have to type the following query in SQL:

INSERT INTO New_Employee SELECT * FROM Old_Employee WHERE Emp_Salary > 40000;

Now, you can check the details of the updated New_Employee table by using the following SELECT query:

SELECT * FROM New_Employee;

Output:

Table: New_Employee

Emp_ID	Emp_Name	Emp_Salary	Address
1008	Sumit	50000	Agra
1009	Akash	55000	Delhi
1010	Devansh	65000	Gurgaon
1001	Akhil	50000	Agra
1003	Bheem	45000	Gurgaon
1004	Chetan	60000	Noida
1006	Raman	50000	Ghaziabad

Example 2: This example describes how to use **ANY operator** with subquery in the INSERT Statement.

Here we have taken the New_Employee, old_Employee, and Department table. The data of the New_Employee table is shown in the following table:

Table: New_Employee

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1008	Sumit	50000	401

The data of the old_Employee table is shown in the below table:

Table: Old_Employee

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000	404
1002	Balram	25000	403
1003	Bheem	45000	405
1004	Chetan	60000	402
1005	Ram	65000	407
1006	Shyam	55500	NULL
1007	Shobhit	60000	NULL

The data of Department table is shown in the below table:

Dept_ID	Dept_Name	Emp_ID
401	Administration	1008
402	HR	1004
403	Testing	1002
404	Coding	1001
405	Sales	1003
406	Marketing	NULL
407	Accounting	1005

```
INSERT INTO New_Employee
SELECT * FROM Old_Employee
WHERE Emp_ID = ANY( SELECT Emp_ID FROM Department WHERE Dept_ID = 407 OR Dept_ID =
406 );
```

Now, check the details of the **New_Employee** table by using the following **SELECT** statement:

Output:

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1008	Sumit	50000	401
1005	Ram	65000	407

Subquery with the UPDATE statement

The subqueries and nested queries can be used with the UPDATE statement in Structured Query Language to update the existing table's columns. We can easily update one or more columns using a subquery with the UPDATE statement.

Syntax of Subquery with the UPDATE statement

```
UPDATE Table_Name SET Column_Name = New_value WHERE Value OPERATOR (SELECT COLU
MN_NAME FROM TABLE_NAME WHERE Condition) ;
```

Example of Subquery with the UPDATE statement

This example updates the record of one table using the IN operator with subquery in the UPDATE statement. Let's take an Employee_Details and Department table. The data of the Employee_Details table is shown in the following table:

Table: Employee_Details

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
---------------	-----------------	-------------------	----------------

1001	Akhil	50000	404
1002	Balram	25000	403
1003	Bheem	45000	405
1004	Chetan	60000	402
1005	Ram	65000	407
1006	Shyam	55500	NULL
1007	Shobhit	60000	NULL

The data of Department table is shown in the below table:

Table: Department

Dept_ID	Dept_Name	Emp_ID	Dept_Grade
401	Administration	1008	B
402	HR	1004	A
403	Testing	1002	A
404	Coding	1001	B
405	Sales	1003	A
406	Marketing	NULL	C
407	Accounting	1005	A

The following updates the salary of those employees whose Department Grade is A:

```
UPDATE Employee_Details
SET Emp_Salary = Emp_Salary + 5000
WHERE Emp_ID IN (SELECT Emp_ID FROM Department WHERE Dept_Grade = 'A');
```

The following query will show the updated data of the Employee_Details table in the output:

```
SELECT * FROM Employee_Details ;
```

Output:

Table: Employee_Details

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000	404
1002	Balram	30000	403
1003	Bheem	50000	405
1004	Chetan	65000	402
1005	Ram	70000	407
1006	Shyam	55500	NULL
1007	Shobhit	60000	NULL

Subquery with the DELETE statement

We can easily delete one or more records from the SQL table using subquery with the DELETE statement in Structured Query Language.

Syntax of Subquery with DELETE statement

```
DELETE FROM Table_Name WHERE Value OPERATOR (SELECT COLUMN_NAME FROM TABLE_NAME WHERE Condition) ;
```

Example of Subquery with DELETE statement

This example deletes the records from the table using the IN operator with subquery in the DELETE statement. Let's take an Employee_Details and Department table. The data of the Employee_Details table is shown in the following table:

Table: Employee_Details

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000	404
1002	Balram	25000	403
1003	Bheem	45000	405
1004	Chetan	60000	402
1005	Ram	65000	407
1006	Shyam	55500	NULL
1007	Shobhit	60000	NULL
1008	Ankit	48000	401

The data of Department table is shown in the below table:

Dept_ID	Dept_Name	Emp_ID	Dept_Grade
401	Administration	1008	C
402	HR	1004	A
403	Testing	1002	C
404	Coding	1001	B
405	Sales	1003	A
406	Marketing	NULL	C
407	Accounting	1005	C

The following query deletes the record of those employees from the Employee_Details whose Department Grade is C:

```
DELETE FROM Employee_Details WHERE Emp_ID IN ( SELECT Emp_ID FROM Department WHERE Dept_Grade = 'C' ) ;
```

The following query will show the updated data of the Employee_Details table in the output:

```
SELECT * FROM Employee_Details ;
```

Output:

Table: Employee_Details

Emp_ID	Emp_Name	Emp_Salary	Dept_ID
1001	Akhil	50000	404
1003	Bheem	45000	405
1004	Chetan	60000	402
1006	Shyam	55500	NULL
1007	Shobhit	60000	NULL

```
mysql> SELECT * FROM employees;
+-----+-----+-----+-----+-----+
| emp_id | emp_name | emp_age | city   | income |
+-----+-----+-----+-----+-----+
| 101 | Peter    | 32    | Newyork | 200000 |
| 102 | Mark     | 32    | California | 300000 |
| 103 | Donald   | 40    | Arizona  | 1000000 |
| 104 | Obama    | 35    | Florida  | 5000000 |
| 105 | Linklon  | 32    | Georgia  | 250000 |
| 106 | Kane     | 45    | Alaska   | 450000 |
| 107 | Adam     | 35    | California | 5000000 |
| 108 | Macculam | 40    | Florida  | 350000 |
| 109 | Brayan   | 32    | Alaska   | 400000 |
| 110 | Stephen  | 40    | Arizona  | 600000 |
| 111 | Alexander | 45    | California | 70000 |
+-----+-----+-----+-----+-----+
```

MySQL Subquery with Comparison Operator

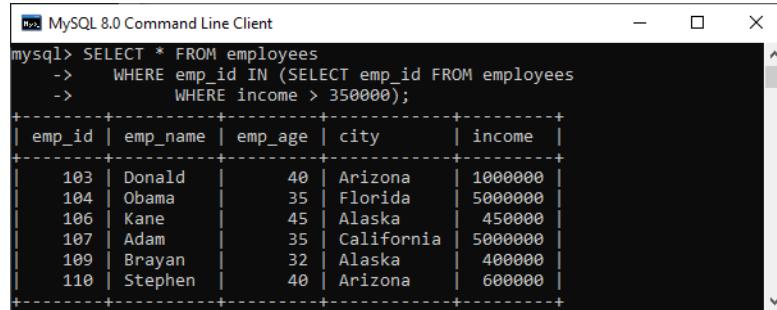
A comparison operator is an operator used to compare values and returns the result, either true or false. The following comparison operators are used in MySQL <, >, =, <>, <=>, etc. We can use the subquery before or after the comparison operators that return a single value. The returned value can be the arithmetic expression or a

column function. After that, SQL compares the subquery results with the value on the other side of the comparison operator. The example below explains it more clearly:

Following is a simple statement that returns the **employee detail whose income is more than 350000** with the help of a subquery:

```
SELECT * FROM employees
WHERE emp_id IN (SELECT emp_id FROM employees
WHERE income > 350000);
```

This query first executes the subquery that returns the **employee id whose income > 350000**. Second, the main query will return the employees all details whose employee id are in the result set returned by the subquery. After executing the statement, we will get the below output, where we can see the employee detail whose income>350000.



The screenshot shows the MySQL 8.0 Command Line Client interface. A query is being run in the command line area:

```
mysql> SELECT * FROM employees
-> WHERE emp_id IN (SELECT emp_id FROM employees
-> WHERE income > 350000);
```

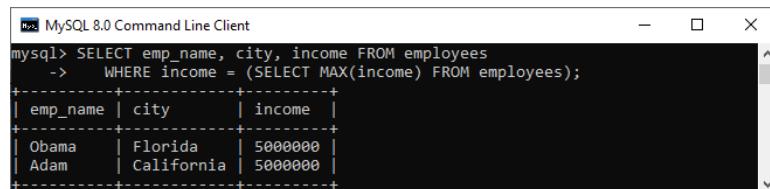
The resulting table output is:

emp_id	emp_name	emp_age	city	income
103	Donald	40	Arizona	1000000
104	Obama	35	Florida	500000
106	Kane	45	Alaska	450000
107	Adam	35	California	500000
109	Brayan	32	Alaska	400000
110	Stephen	40	Arizona	600000

Let us see an example of another comparison operator, such as equality (=) to find employee details with **maximum income** using a subquery.

```
SELECT emp_name, city, income FROM employees
WHERE income = (SELECT MAX(income) FROM employees);
```

It will give the output where we can see two employees detail who have maximum income.



The screenshot shows the MySQL 8.0 Command Line Client interface. A query is being run in the command line area:

```
mysql> SELECT emp_name, city, income FROM employees
-> WHERE income = (SELECT MAX(income) FROM employees);
```

The resulting table output is:

emp_name	city	income
Obama	Florida	500000
Adam	California	500000

MySQL Subquery with IN or NOT-IN Operator

If the subquery produces more than one value, we need to use the IN or NOT IN Operator with the WHERE clause. Suppose we have a table named "**Student**" and "**Student2**" that contains the following data:

Table: Student

Stud_ID	Name	Email	City
1	Peter	peter@javatpoint.com	Texas
2	Suzi	suzi@javatpoint.com	California
3	Joseph	joseph@javatpoint.com	Alaska
4	Andrew	andrew@javatpoint.com	Los Angeles
5	Brayan	brayan@javatpoint.com	New York

Table: Student2

Stud_ID	Name	Email	City
1	Stephen	stephen@javatpoint.com	Texas
2	Joseph	joseph@javatpoint.com	Los Angeles
3	Peter	peter@javatpoint.com	California
4	David	david@javatpoint.com	New York
5	Maddy	maddy@javatpoint.com	Los Angeles

The following subquery with NOT IN Operator returns the student detail who does not belong to Los Angeles City from both tables as follows:

```
SELECT Name, City FROM student
WHERE City NOT IN (
    SELECT City FROM student2 WHERE City='Los Angeles');
```

After execution, we can see that the result contains the student details that do not belong to Los Angeles City.

```
MySQL 8.0 Command Line Client
mysql> SELECT Name, City FROM student
-> WHERE City NOT IN (
->     SELECT City FROM student2 WHERE City='Los Angeles');
+-----+-----+
| Name | City |
+-----+-----+
| Stephen | Texas |
| Peter | California |
| David | New York |
+-----+-----+
```

MySQL Subquery in the FROM Clause

If we use a subquery in the FROM clause, MySQL will return the output from a subquery is used as a temporary table. We called this table as a derived table, inline views, or materialized subquery. The following subquery returns the maximum, minimum, and average number of items in the order table:

```
SELECT Max(items), MIN(items), FLOOR(AVG(items))
FROM
    (SELECT order_id, COUNT(order_id) AS items FROM orders
    GROUP BY order_date) AS Student_order_detail;
```

It will give the output as follows:

```
MySQL 8.0 Command Line Client
mysql> SELECT Max(items), MIN(items), FLOOR(AVG(items))
-> FROM
->     (SELECT order_id, COUNT(order_id) AS items FROM orders
->     GROUP BY order_date) AS Student_order_detail;
+-----+-----+
| Max(items) | MIN(items) | FLOOR(AVG(items)) |
+-----+-----+
| 1 | 1 | 1 |
+-----+-----+
```

MySQL Correlated Subqueries

A correlated subquery in MySQL is a subquery that depends on the outer query. It uses the data from the outer query or contains a reference to a parent query that also appears in the outer query. MySQL evaluates it once from each row in the outer query.

```
SELECT emp_name, city, income
FROM employees emp WHERE income > (
    SELECT AVG(income) FROM employees WHERE city = emp.city);
```

In the above query, we select an **employee name and city** whose income is higher than the average income of all employees in each city.

```

MySQL 8.0 Command Line Client
mysql> SELECT emp_name, city, income
-> FROM employees emp WHERE income > (
->   SELECT AVG(income) FROM employees WHERE city = emp.city);
+-----+-----+-----+
| emp_name | city      | income |
+-----+-----+-----+
| Donald   | Arizona   | 1000000 |
| Obama    | Florida   | 5000000 |
| Kane     | Alaska    | 450000  |
| Adam     | California| 5000000 |
+-----+-----+-----+

```

The subquery executes for every city of the specified table because it will change for every row. Therefore, the average income will also be changed. Then, the main query filters employee details whose income is higher than the average income from the subquery.

MySQL Subqueries with EXISTS or NOT EXISTS

The EXISTS operator is a Boolean operator that returns a true or false result. It is used with a subquery and checks the existence of data in a subquery. If a subquery returns any record at all, this Operator returns true. Otherwise, it will return false. The NOT EXISTS Operator used for negation that gives true value when the subquery does not return any row. Otherwise, it returns false. Both EXISTS and NOT EXISTS used with correlated subqueries. The following example illustrates it more clearly. Suppose we have a table **customer** and **order** that contains the data as follows:

```

MySQL 8.0 Command Line Client
mysql> SELECT * FROM customer;
+-----+-----+-----+-----+
| cust_id | name      | occupation | age   |
+-----+-----+-----+-----+
| 101    | Peter     | Engineer   | 32   |
| 102    | Joseph    | Developer  | 30   |
| 103    | John      | Leader     | 28   |
| 104    | Stephen   | Scientist  | 45   |
| 105    | Suzi      | Carpenter  | 26   |
| 106    | Bob       | Actor     | 25   |
| 107    | NULL      | NULL      | NULL |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> SELECT * FROM Orders;
+-----+-----+-----+-----+
| order_id | cust_id | prod_name | order_date |
+-----+-----+-----+-----+
| 1         | 101     | Laptop    | 2020-01-10 |
| 2         | 103     | Desktop   | 2020-02-12 |
| 3         | 106     | Iphone    | 2020-02-15 |
| 4         | 104     | Mobile    | 2020-03-05 |
| 5         | 102     | TV        | 2020-03-20 |
+-----+-----+-----+-----+

```

The below SQL statements uses EXISTS operator to find the name, occupation, and age of the customer who has placed at least one order.

```

SELECT name, occupation, age FROM customer C
WHERE EXISTS (SELECT * FROM Orders O
WHERE C.cust_id = O.cust_id);

```

This statement uses NOT EXISTS operator that returns the customer details who have not placed an order.

```

SELECT name, occupation, age FROM customer C
WHERE NOT EXISTS (SELECT * FROM Orders O
WHERE C.cust_id = O.cust_id);

```

We can see the below output to understand the above queries result.

```

MySQL 8.0 Command Line Client
mysql> SELECT name, occupation, age FROM customer C
-> WHERE EXISTS (SELECT * FROM Orders O
-> WHERE C.cust_id = O.cust_id);
+-----+-----+-----+
| name | occupation | age |
+-----+-----+-----+
| Peter | Engineer   | 32  |
| Joseph| Developer  | 30  |
| John  | Leader     | 28  |
| Stephen| Scientist | 45  |
| Bob   | Actor      | 25  |
+-----+-----+-----+
5 rows in set (0.10 sec)

mysql> SELECT name, occupation, age FROM customer C
-> WHERE NOT EXISTS (SELECT * FROM Orders O
-> WHERE C.cust_id = O.cust_id);
+-----+-----+-----+
| name | occupation | age |
+-----+-----+-----+
| Suzi | Carpenter | 26  |
| NULL | NULL       | NULL |
+-----+-----+-----+

```

MySQL ROW Subqueries

It is a subquery that returns a single row where we can get more than one column values. We can use the following operators for comparing row subqueries `=`, `>`, `<`, `>=`, `<=`, `<>`, `!=`, `<=>`. Let us see the following example:

```

SELECT * FROM customer C WHERE ROW(cust_id, occupation) =
(SELECT order_id, order_date FROM Orders O WHERE C.cust_id = O.cust_id);

```

If given row has `cust_id`, `occupation` values equal to the `order_id`, `order_date` values of any rows in the first table, the `WHERE` expression is TRUE, and each query returns those first table rows. Otherwise, the expression is FALSE, and the query produces an empty set, which can be shown in the below image:

```

MySQL 8.0 Command Line Client
mysql> SELECT * FROM customer C WHERE ROW(cust_id, occupation) =
-> (SELECT order_id, order_date FROM Orders O WHERE C.cust_id = O.cust_id);
Empty set (0.00 sec)

```

MySQL Subqueries with ALL, ANY, and SOME

We can use a subquery which is followed by the keyword ALL, ANY, or SOME after a comparison operator. The following are the syntax to use subqueries with ALL, ANY, or SOME:

```

operand comparison_operator ANY (subquery)
operand comparison_operator ALL (subquery)
operand comparison_operator SOME (subquery)

```

The ALL keyword compares values with the value returned by a subquery. Therefore, it returns TRUE if the comparison is TRUE for ALL of the values returned by a subquery. The ANY keyword returns TRUE if the comparison is TRUE for ANY of the values returned by a subquery. The ANY and SOME keywords are the same because they are the alias of each other. The following example explains it more clearly:

```

SELECT cust_id, name FROM customer WHERE
cust_id > ANY (SELECT cust_id FROM Orders);

```

```
MySQL 8.0 Command Line Client
mysql> SELECT cust_id, name FROM customer WHERE
-> cust_id > ANY (SELECT cust_id FROM Orders);
+-----+-----+
| cust_id | name   |
+-----+-----+
|    102  | Joseph |
|    103  | John   |
|    104  | Stephen|
|    105  | Suzi   |
|    106  | Bob    |
|    107  | NULL   |
+-----+-----+
```

If we use ALL in place of ANY, it will return TRUE when the comparison is TRUE for ALL values in the column returned by a subquery. For example:

```
SELECT cust_id, name FROM customer WHERE
cust_id > ALL (SELECT cust_id FROM Orders);
```

We can see the output as below:

```
MySQL 8.0 Command Line Client
mysql> SELECT cust_id, name FROM customer WHERE
-> cust_id > ALL (SELECT cust_id FROM Orders);
+-----+-----+
| cust_id | name   |
+-----+-----+
|    107  | NULL   |
+-----+-----+
```

EXPERIMENT-4

Lab 4: To implement and use Inbuilt Functions

Objective: To understand the use of SQL Inbuilt functions.

Inbuilt Functions

- 1. Aggregate Functions**
- 2. String Functions**
- 3. Mathematical Functions**
- 4. Date and Time Functions**
- 5. Conditional Functions**

1. Aggregate Functions

COUNT(): Counts the number of rows that satisfy the specified condition.

Example: `SELECT COUNT(name) FROM employee;`

SUM(): Calculates the sum of the specified column values.

Example: `SELECT SUM(working_hours) AS "Total working hours" FROM employee;`

AVG(): Calculates the average of the specified column values.

Example: `SELECT AVG(working_hours) AS "Average working hours" FROM employee;`

MIN(): Retrieves the minimum value of the specified column.

Example: `SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;`

MAX(): Retrieves the maximum value of the specified column.

Example: `SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;`

LIMIT: Limits the number of rows returned by a query.

Example: `SELECT working_date FROM employee LIMIT 1;`

ORDER BY with LIMIT: Orders the result set by a specified column and limits the number of rows returned.

Example: `SELECT working_hours FROM employee ORDER BY name DESC LIMIT 1;`

GROUP_CONCAT(): Concatenates values from multiple rows into a single string within each group.

Example: `SELECT emp_id, emp_fname, emp_lname, dept_id, GROUP_CONCAT(designation) AS designation FROM employee2 GROUP BY emp_id, emp_fname, emp_lname, dept_id;`

GROUP_CONCAT() with DISTINCT: Concatenates distinct values from multiple rows into a single string within each group.

Example: `SELECT emp_fname, dept_id, GROUP_CONCAT(DISTINCT designation) AS designation FROM employee2 GROUP BY emp_id, emp_fname, emp_lname, dept_id;`

GROUP_CONCAT() with Separator: Concatenates values using a specified separator within each group.

Example: `SELECT emp_fname, GROUP_CONCAT(DISTINCT designation SEPARATOR ';') AS designation FROM employee2 GROUP BY emp_id, emp_fname, emp_lname, dept_id;`

GROUP_CONCAT() with CONCAT_WS() and Separator: Concatenates values using a specified separator and a specified separator between values within each group.

Example: `SELECT GROUP_CONCAT(CONCAT_WS(',', emp_lname, emp_fname) SEPARATOR ';') AS employeeName FROM employee2;`

2. MySQL String Function

CONCAT_WS(): Concatenates arguments using a specified separator.

CONCAT(): Concatenates all the arguments into a string.

CHARACTER_LENGTH(): Returns the size of the specified string.

ELT(): Returns the Nth element from a list of strings.

EXPORT_SET(): Returns a string for each bit set.

FIELD(): Returns the index of a string.

FIND_IN_SET(): Returns the value of a string given its position in the argument.

FORMAT(): Formats a number with a specified number of decimal places.

FROM_BASE64(): Encodes a given string to binary format.

HEX(): Returns a specified number or string in hexadecimal format.

INSERT(): Inserts a character into a string at a specified position.

INSTR(): Returns the first occurrence of a substring in a string.

LCASE(): Returns the given string in lowercase.

LEFT(): Returns a specified number of characters from the left side of a string.

LENGTH(): Returns the length of the specified string in bytes.

LIKE(): Checks for pattern matching and returns either 1 or 0.

LOAD_FILE(): Returns the content of a file.

LOCATE(): Returns the position of a given substring in a string.

LOWER(): Returns the given string in lowercase.

LPAD(): Left-pads a string to a specified length.

LTRIM(): Removes leading spaces from a string.

MAKE_SET(): Returns values from a set for the given bit.

MID(): Extracts a substring from a string.

OCTET_LENGTH(): Returns the length of a given string.

OCT(): Returns the octal representation of a number.

ORD(): Returns the code for the leftmost character if it is multi-byte.

POSITION(): Returns the position of a given substring in a string.

QUOTE(): Returns the string passed in single quotes.

REPEAT(): Repeats a string for a specified number of times.

REPLACE(): Replaces all occurrences of a substring within a string.

REVERSE(): Reverses a string.

RIGHT(): Extracts a specified number of characters from the right side of a string.

RPAD(): Pads the specified strings from the right.

RTRIM(): Removes trailing spaces from a specified string.

SOUNDEX(): Returns the soundex string for the specified string.

3. Mathematical Functions

ABS(): Returns the absolute value of the specified number.

ACOS(): Returns the arc cosine of the given number.

SIGN(): Returns the sign of the specified number.

SIN(): Returns the sine value of the given number.

SQRT(): Returns the square root of the given number.

SUM(): Sums the values of given expressions.

TAN(): Returns the tangent of the given number.

TRUNCATE(): Truncates the given number up to a certain number of decimal places.

ASIN(): Returns the arc sine of the given number.

ATAN2(): Returns the arc tangent of the specified numbers n and m.

ATAN(): Returns the arc tangent of the specified number.

AVG(): Calculates the average value from the given expression.

CEIL() (or CEILING()): Returns the smallest value greater than or equal to the specified number.

COS(): Returns the cosine of the given number.

COT(): Returns the cotangent of the given number.

COUNT(): Gets the total count for the specified column of the table.

DEGREES(): Converts the given radian number into a degree.

DIV(): Finds the integer division by dividing the number n by the number m.

EXP(): Finds e raised to the power of the given number.

FLOOR(): Finds the greatest integer less than or equal to the specified number.

GREATEST(): Gets the largest number from the list.

LEAST(): Gets the smallest number from the list.

LN(): Gets the natural logarithm for the specified number.

LOG10(): Gets the base-10 logarithm for the specified number.

LOG(): Returns either the natural logarithm or the specified base logarithm of the given number.

LOG2(): Gets the base-2 logarithm for the specified number.

MAX(): Gets the maximum number from the given column.

MIN(): Gets the minimum number from the given column.

MOD(): Gets the remainder for the specified values.

PI(): Gets the value of pi up to 6 decimal places.

POWER(): Gets the power for the specified values.

POW(): Gets the power for the specified values.

RADIANS(): Converts the given degrees to radians.

RAND(): Generates a random number.

ROUND(): Rounds off the specified number.

Lab Performance Questions

```
CREATE TABLE employee(
    name varchar(45) NOT NULL,
    occupation varchar(35) NOT NULL,
    working_date date,
    working_hours varchar(10)
);
```

```
desc employee;
```

```
INSERT INTO employee VALUES
('Robin', 'Scientist', '2020-10-04', 12),
('Warner', 'Engineer', '2020-10-04', 10),
('Peter', 'Actor', '2020-10-04', 13),
('Marco', 'Doctor', '2020-10-04', 14),
('Brayden', 'Teacher', '2020-10-04', 12),
('Antonio', 'Business', '2020-10-04', 11);
```

```
select * from employee;
```

1. **SELECT COUNT(name) FROM employee;**
2. **SELECT SUM(working_hours) AS "Total working hours" FROM employee;**
3. **SELECT AVG(working_hours) AS "Average working hours" FROM employee;**
4. **SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;**
5. **SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;**
6. **SELECT working_date FROM employee LIMIT 1;**
7. **SELECT working_hours FROM employee ORDER BY name DESC LIMIT 1;**

```
CREATE TABLE employee2(
    emp_id varchar(45),
    emp_fname varchar(10),
    emp_lname varchar(10),
    dept_id varchar(10),
    designation varchar(25)
);
```

```
desc employee2;
```

```
INSERT INTO employee2 VALUES
(1, 'David', 'Miller', 2, 'Engineer'),
(2, 'Peter', 'Watson', 3, 'Manager'),
(3, 'Mark', 'Boucher', 1, 'Scientist'),
(2, 'Peter', 'Watson', 3, 'BDE'),
(1, 'David', 'Miller', 2, 'Developer'),
(4, 'Adam', 'Warner', 4, 'Receptionist'),
(3, 'Mark', 'Boucher', 1, 'Engineer'),
(4, 'Adam', 'Warner', 4, 'Clerk');
```

```
select * from employee2;
```

8. **SELECT emp_id, emp_fname, emp_lname, dept_id,
GROUP_CONCAT(designation) AS designation FROM employee2 GROUP BY
emp_id, emp_fname, emp_lname, dept_id;**
9. **SELECT emp_fname, dept_id, GROUP_CONCAT(DISTINCT designation) as
designation FROM employee2 group by emp_id, emp_fname, emp_lname, dept_id;**
10. **SELECT emp_fname, GROUP_CONCAT(DISTINCT designation SEPARATOR ';')
as designation FROM employee2 group by emp_id, emp_fname, emp_lname,
dept_id;**
11. **SELECT GROUP_CONCAT(CONCAT_WS(',', emp_lname, emp_fname)
SEPARATOR ';') as employeename FROM employee2;**

MySQL String Function

```
CREATE TABLE sample_table (  
    id INT,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50)  
);  
desc sample_table;
```

```
INSERT INTO sample_table (id, first_name, last_name)  
VALUES  
    (1, 'John', 'Doe'),  
    (2, 'Jane', 'Smith'),  
    (3, 'Alice', 'Johnson');
```

12. **SELECT id FROM sample_table;**
13. **SELECT first_name FROM sample_table;**
14. **SELECT last_name FROM sample_table;**
15. **SELECT CONCAT_WS('-', first_name, last_name) AS concat_ws_result FROM
sample_table;**
16. **SELECT CONCAT(first_name, ' ', last_name) AS concat_result FROM
sample_table;**
17. **SELECT CHARACTER_LENGTH(first_name) AS char_length_result FROM
sample_table;**
18. **SELECT ELT(1, first_name, last_name, 'Other') AS elt_result FROM sample_table;**
19. **SELECT EXPORT_SET(id, 1, ',', '') AS export_set_result FROM sample_table;**

- 20. SELECT FIELD(first_name, 'Alice', 'Jane', 'John') AS field_result FROM sample_table;**
- 21. SELECT FIND_IN_SET(first_name, 'John,Jane,Alice') AS find_in_set_result FROM sample_table;**
- 22. SELECT FORMAT(id, 2) AS format_result FROM sample_table;**
- 23. SELECT FROM_BASE64('SGVsbG8gd29ybGQh') AS from_base64_result FROM sample_table;**
- 24. SELECT HEX(id) AS hex_result FROM sample_table;**
- 25. SELECT INSERT(first_name, 2, 0, 'X') AS insert_result FROM sample_table;**
- 26. SELECT INSTR(first_name, 'hn') AS instr_result FROM sample_table;**
- 27. SELECT LCASE(first_name) AS lcase_result FROM sample_table;**
- 28. SELECT LEFT(first_name, 2) AS left_result FROM sample_table;**
- 29. SELECT LENGTH(first_name) AS length_result FROM sample_table;**
- 30. SELECT first_name LIKE 'J%' AS like_result FROM sample_table;**
- 31. SELECT LOAD_FILE('/path/to/file') AS load_file_result FROM sample_table;**
- 32. SELECT LOCATE('a', first_name) AS locate_result FROM sample_table;**
- 33. SELECT LOWER(first_name) AS lower_result FROM sample_table;**
- 34. SELECT LPAD(first_name, 10, '*') AS lpad_result FROM sample_table;**
- 35. SELECT LTRIM(' ' || first_name) AS ltrim_result FROM sample_table;**
- 36. SELECT MAKE_SET(2, 'Red', 'Green', 'Blue') AS make_set_result FROM sample_table;**
- 37. SELECT MID(first_name, 2, 2) AS mid_result FROM sample_table;**
- 38. SELECT OCTET_LENGTH(first_name) AS octet_length_result FROM sample_table;**
- 39. SELECT OCT(id) AS oct_result FROM sample_table;**
- 40. SELECT ORD(LEFT(first_name, 1)) AS ord_result FROM sample_table;**
- 41. SELECT POSITION('oh' IN first_name) AS position_result FROM sample_table;**
- 42. SELECT QUOTE(first_name) AS quote_result FROM sample_table;**

```
43. SELECT REPEAT(first_name, 2) AS repeat_result FROM sample_table;  
44. SELECT REPLACE(first_name, 'Jo', 'X') AS replace_result FROM sample_table;  
45. SELECT REVERSE(first_name) AS reverse_result FROM sample_table;  
46. SELECT RIGHT(last_name, 2) AS right_result FROM sample_table;  
47. SELECT RPAD(first_name, 10, '*') AS rpad_result FROM sample_table;  
48. SELECT RTRIM(first_name) AS rtrim_result FROM sample_table;  
49. SELECT SOUNDEX(first_name) AS soundex_result FROM sample_table;
```

Mathematical Functions

-- Creating a sample employee table

```
CREATE TABLE employee (  
    employee_id INT PRIMARY KEY,  
    salary DECIMAL(10, 2),  
    age INT  
);
```

-- Inserting sample data into the employee table

```
INSERT INTO employee (employee_id, salary, age)  
VALUES  
    (1, 50000.25, 30),  
    (2, 60000.75, 35),  
    (3, 75000.50, 28);
```

-- Using various functions in SQL queries on the employee table

```
50. SELECT employee_id FROM employee;  
51. SELECT ABS(salary) AS abs_salary FROM employee;  
52. SELECT ACOS(salary / 100000) AS acos_salary FROM employee;  
53. SELECT SIGN(salary) AS sign_salary FROM employee;  
54. SELECT SIN(salary / 100000) AS sin_salary FROM employee;  
55. SELECT SQRT(salary) AS sqrt_salary FROM employee;  
56. SELECT salary AS sum_salary FROM employee;  
57. SELECT TAN(salary / 100000) AS tan_salary FROM employee;  
58. SELECT TRUNCATE(salary, 2) AS truncated_salary FROM employee;  
59. SELECT ASIN(salary / 100000) AS asin_salary FROM employee;
```

60. **SELECT ATAN2(salary, age) AS atan2_salary FROM employee;**
61. **SELECT ATAN(salary / 100000) AS atan_salary FROM employee;**
62. **SELECT salary AS avg_salary FROM employee;**
63. **SELECT CEIL(salary) AS ceil_salary FROM employee;**
64. **SELECT CEILING(salary) AS ceiling_salary FROM employee;**
65. **SELECT COS(salary / 100000) AS cos_salary FROM employee;**
66. **SELECT COT(salary / 100000) AS cot_salary FROM employee;**
67. **SELECT 1 AS count_employee FROM employee;**
68. **SELECT DEGREES(salary / 100000) AS degrees_salary FROM employee;**
69. **SELECT salary DIV 2 AS div_salary FROM employee;**
70. **SELECT EXP(salary / 100000) AS exp_salary FROM employee;**
71. **SELECT FLOOR(salary) AS floor_salary FROM employee;**
72. **SELECT GREATEST(salary, 0, -50000) AS greatest_salary FROM employee;**
73. **SELECT LEAST(salary, 0, -50000) AS least_salary FROM employee;**
74. **SELECT LN(salary / 100000) AS ln_salary FROM employee;**
75. **SELECT LOG10(salary / 100000) AS log10_salary FROM employee;**
76. **SELECT LOG(salary / 100000, 2) AS log_salary FROM employee;**
77. **SELECT LOG2(salary / 100000) AS log2_salary FROM employee;**
78. **SELECT salary AS max_salary FROM employee;**
79. **SELECT salary AS min_salary FROM employee;**
80. **SELECT MOD(salary, 3) AS mod_salary FROM employee;**
81. **SELECT PI() AS pi_value FROM employee;**
82. **SELECT POWER(salary, 2) AS power_salary FROM employee;**
83. **SELECT POW(salary, 2) AS pow_salary FROM employee;**
84. **SELECT RADIANS(salary / 100000) AS radians_salary FROM employee;**

85. SELECT RAND() AS rand_value FROM employee;

86. SELECT ROUND(salary, 2) AS round_salary FROM employee;

Lab 5: Understanding of SQL

Dr. Keshav Sinha

Question 1

Title: Consider the following set of requirements for a UNIVERSITY database that is used to keep track of students' transcripts.

1. The university keeps track of each student's name, student number, Social Security number, current address and phone number, permanent address and phone number, birth date, sex, class (freshman, sophomore, ..., graduate), major department, minor department (if any), and degree program (B.A., B.S., ..., Ph.D.). Some user applications need to refer to the city, state, and ZIP Code of the student's permanent address and to the student's last name. Both Social Security number and student number have unique values for each student.
 - a. Each department is described by a name, department code, office number, office phone number, and college. Both name and code have unique values for each department.
 - b. Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of the course number is unique for each course.
 - c. Each section has an instructor, semester, year, course, and section number. The section number distinguishes sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.
 - d. A grade report has a student, section, letter grade, and numeric grade (0, 1, 2, 3, or 4).

Design an Entity-Relationship diagram for the mail order database and enter the design using a data-modeling tool such as ERWin/free tool. Specify key attributes of each entity type, and structural constraints on each relationship type. Note any unspecified requirements and make appropriate assumptions to make the specification complete.

Question 2

Title. Consider the following set of requirements for a Company database that is used to keep track of employee.

The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.

- a. A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- b. We store each employee's name, Social Security number, address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).
- c. We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee.

Design an Entity-Relationship diagram for the company database and enter the design using a data-modeling tool such as ERWin/free tool.

Question 3

Objective: To understand the concept of designing issue related to the database with creating, populating the tables. To understand the concept of data constraints that is enforced on data being stored in the table. Focus on Primary Key and the Foreign Key.

- a. **Create the tables for** Company database as per ER diagram of Exp 2.

TABLE 1: EMPLOYEE

```
[ Fname VARCHAR(15) NOT NULL,  
Minit CHAR,
```

```
Lname VARCHAR(15) NOT NULL,  
Ssn CHAR(9) NOT NULL,  
Bdate DATE,  
Address VARCHAR(30),  
Sex CHAR,  
Salary DECIMAL(10,2),  
Super_ssn CHAR(9),  
Dno INT NOT NULL,  
PRIMARY KEY (Ssn),  
FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),  
FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)  
]
```

TABLE 2: DEPARTMENT

```
[Dname VARCHAR(15) NOT NULL,  
Dnumber INT NOT NULL,  
Mgr_ssn CHAR(9) NOT NULL,  
Mgr_start_date DATE,  
PRIMARY KEY (Dnumber),  
UNIQUE (Dname),  
FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );  
]
```

TABLE 3: DEPT_LOCATIONS

```
( Dnumber INT NOT NULL,  
Dlocation VARCHAR(15) NOT NULL,  
PRIMARY KEY (Dnumber, Dlocation),  
FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

TABLE 4: PROJECT

```
( Pname VARCHAR(15) NOT NULL,  
Pnumber INT NOT NULL,  
Plocation VARCHAR(15),  
Dnum INT NOT NULL,
```

PRIMARY KEY (Pnumber),
 UNIQUE (Pname),
 FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber);

TABLE 5: WORKS_ON

(Essn CHAR(9) NOT NULL,
 Pno INT NOT NULL,
 Hours DECIMAL(3,1) NOT NULL,
 PRIMARY KEY (Essn, Pno),
 FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
 FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber));

TABLE 6: DEPENDENT

(Essn CHAR(9) NOT NULL,
 Dependent_name VARCHAR(15) NOT NULL,
 Sex CHAR,
 Bdate DATE,
 Relationship VARCHAR(8),
 PRIMARY KEY (Essn, Dependent_name),
 FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn));

b. Insert the following data into their respective tables of Company database.

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

EMPLOYEE

FNAME	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5

EMPLOYEE

FNAME	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
Franklin	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5
Alicia	Zelaya	999887777	1968-01-19	3321 Castle, Spring TX	F	25000	987654321	4
Jennifer	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4
Ramesh	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5
Joyce	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5
Ahmad	Jabbar	987987987	1969-03-29	980 Dallas, Houston TX	M	25000	987654321	4
James	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	null	1

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0

WORKS_ON

ESSN	PNO	HOURS
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	null

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	1986-04-04	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire

DEPT_LOCATIONS

DNUMBER	DLOCATION
5	Houston
5	Sugarland

Question 4

Objective: To understand the concept of data constraints that is enforced on data being stored in the table. Focus on Primary Key and the Foreign Key.

1. Create the tables described below:

Table name: CLIENT_MASTER

Description: used to store client information.

Column name	data type	Size	Constraints
CLIENTNO	Varchar	6	Primary key / first letter must start with 'C'
NAME	Varchar	20	Not Null
ADDRESS 1	Varchar	30	
ADDRESS 2	Varchar	30	
CITY	Varchar	15	
PINCODE	Integer	8	
STATE	Varchar	15	
BALDUE	Decimal	10,2	

Table Name: **PRODUCT_MASTER**

Description: used to store product information

Column name	data type	Size	Attributes
PRODUCTNO	Varchar	6	Primary Key/ first letter must start with 'P'
DESCRIPTION	Varchar	15	Not Null
PROFITPERCENT	Decimal	4,2	Not Null
UNIT MEASURE	Varchar	10	Not Null
QTYONHAND	Integer	8	Not Null

REORDERL VL	Integer	8	Not Null
SELLPRICE	Decimal	8,2	Not Null
COSTPRICE	Decimal	8,2	Not Null

Table Name: SALESMAN_MASTER

Description: used to store salesman information working for the company.

Column name	data type	Size	Attributes
SALESMANNO	Varchar	6	Primary Key/ first letter must start with 'S'
SALESMANNAME	Varchar	20	Not Null
ADDRESS 1	Varchar	30	Not Null
ADDRESS 2	Varchar	30	
CITY	Varchar	20	
PINCODE	Integer	8	
STATE	Varchar	20	
SALAMT	Real	8,2	Not Null , Cannot be 0
TGTTOGET	Decimal	6,2	Not Null , Cannot be 0
YTDSALES	Double	6,2	Not Null
REMARKS	Varchar	60	

1. Insert the following data into their respective tables:

a) Data for **CLIENT_MASTER** table:

Client no	Name	City	Pincode	State	BalDue
C00001	Ivan bayross	Mumbai	400054	Maharashtra	15000
C00002	Mamta muzumdar	Madras	780001	Tamil nadu	0
C00003	Chhaya bankar	Mumbai	400057	Maharashtra	5000
C00004	Ashwini joshi	Bangalore	560001	Karnataka	0
C00005	Hansel colaco	Mumbai	400060	Maharashtra	2000
C00006	Deepak sharma	Mangalore	560050	Karnataka	0

b) Data for **PRODUCT_MASTER** table:

Product No	Description	Profit percent	Unit measure	Quantity On hand	Recorder Level	Sell Price	Cost Price

P00001	T-Shirt	5	Piece	200	50	350	250
P0345	Shirts	6	Piece	150	50	500	350
P06734	Cotton jeans	5	Piece	100	20	600	450
P07865	Jeans	5	Piece	100	20	750	500
P07868	Trousers	2	Piece	150	50	850	550
P07885	Pull Overs	2.5	Piece	80	30	700	450
P07965	Denim jeans	4	Piece	100	40	350	250
P07975	Lycra tops	5	Piece	70	30	300	175
P08865	Skirts	5	Piece	75	30	450	300

c) Data for **SALESMAN_MASTER** table:

Salesman No	Name	Address1	Address2	City	Pin Code	State
S00001	Aman	A/14	Worli	Mumbai	400002	Maharashtra
S00002	Omkar	65	Nariman	Mumbai	400001	Maharashtra
S00003	Raj	P-7	Bandra	Mumbai	400032	Maharashtra
S00004	Ashish	A/5	Juhu	Mumbai	400044	Maharashtra

2. Exercise on retrieving records from a table.

- Find out the names of all the clients.
- Retrieve the entire contents of the Client_Master table.
- Retrieve the list of names, city and the state of all the clients.
- List the various products available from the Product_Master table.
- List all the clients who are located in Mumbai.
- Find the names of salesman who have a salary equal to Rs.3000.

3. Exercise on updating records in a table

- Change the city of ClientNo ‘C00005’ to ‘Bangalore’.
- Change the BalDue of ClientNo ‘C00001’ to Rs.1000.
- Change the cost price of ‘Trousers’ to rs.950.00.
- Change the city of the salesman to Pune.

4. Exercise on deleting records in a table

- a. Delete all salesman from the Salesman_Master whose salaries are equal to Rs.3500.
- b. Delete all products from Product_Master where the quantity on hand is equal to 100.
- c. Delete from Client_Master where the column state holds the value ‘Tamil Nadu’.

5. Exercise on altering the table structure

- a. Add a column called ‘Telephone’ of data type integer to the Client_Master table.
- b. Change the size off SellPrice column in Product_Master to 10, 2.

6. Exercise on deleting the table structure along with the data

- a. Destroy the table Client_Master along with its data.

EXPERIMENT-5

Lab 5: To implement and use Inbuilt and Control Flow Functions

Objective: To understand the use of SQL Inbuilt functions.

Inbuilt Functions

1. Date and Time Functions

- CURDATE(): Returns the current date.
- DATE_ADD(): Adds a specified time interval to a date.
- DATE_FORMAT(): Formats a date as specified.
- DATEDIFF(): Calculates the difference between two dates.
- DAY(): Extracts the day from a date.
- DAYNAME(): Returns the name of the day for a given date.
- DAYOFMONTH(): Returns the day of the month for a given date.
- DAYOFWEEK(): Returns the day of the week in numeric format for a given date.
- DAYOFYEAR(): Returns the day of the year for a given date.
- FROM_DAYS(): Converts a day number to a date.
- HOUR(): Extracts the hour from a datetime.
- LAST_DAY(): Returns the last day of the month for a given date.
- NOW(): Returns the current date and time.
- PERIOD_ADD(): Adds a specified number of months to a period.
- PERIOD_DIFF(): Calculates the difference between two periods.
- QUARTER(): Returns the quarter of the year for a given date.
- SECOND(): Extracts the second from a datetime.
- STR_TO_DATE(): Converts a string to a date using a specified format.
- SUBDATE(): Subtracts a specified time interval from a date.
- SUBTIME(): Subtracts a specified time interval from a datetime.
- SYSDATE(): Returns the system date and time.
- TIME(): Extracts the time from a datetime.
- TIME_FORMAT(): Formats a time value.
- TIME_TO_SEC(): Converts a time value to seconds.
- TIMEDIFF(): Calculates the difference between two times or datetimes.
- TIMESTAMP(): Converts an expression to a datetime.
- TO_DAYS(): Converts a date to a day number.
- WEEKDAY(): Returns the index of the day of the week for a given date.
- WEEK(): Returns the week number for a given date.
- WEEKOFYEAR(): Returns the week of the year for a given date.

2. Conditional Functions

- AND Operator: The AND operator is used to combine multiple conditions in a WHERE clause, and all conditions must be true for the row to be included in the result set.
- OR Operator: The OR operator is used to combine multiple conditions in a WHERE clause, and at least one of the conditions must be true for the row to be included in the result set.
- AND and OR Combined: You can use both AND and OR operators in a WHERE clause to create complex conditions, ensuring the proper combination of conditions for filtering rows.
- Boolean: MySQL doesn't have a native Boolean data type, but it uses 0 for FALSE and 1 for TRUE. Boolean logic is often expressed using AND, OR, and NOT operators.

- LIKE Operator: The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. It can include wildcards like `%' (matches any sequence of characters) and `_` (matches any single character).
- IN Operator: The IN operator is used in a WHERE clause to specify multiple values for a column. It checks whether a value matches any value in a list.
- ANY Operator: The ANY operator is used with a subquery and returns true if any of the subquery values meet the specified condition.
- EXISTS Operator: The EXISTS operator is used in a WHERE clause to check if a subquery returns any results. If the subquery returns at least one row, the condition is true.
- NOT Operator: The NOT operator is used to negate a condition in a WHERE clause. It reverses the result of a logical condition.
- Not Equal Operator: The `!=` or `<>` operator is used in a WHERE clause to check if two expressions are not equal.
- IS NULL Operator: The IS NULL operator is used in a WHERE clause to check if a column contains a NULL value.
- IS NOT NULL Operator: The IS NOT NULL operator is used in a WHERE clause to check if a column does not contain a NULL value.
- BETWEEN Operator: The BETWEEN operator is used in a WHERE clause to filter the result set within a range. It is inclusive, including the specified values.

Lab Performance Questions
Date and Time Functions
<pre> CREATE TABLE student (ID INT, Name VARCHAR(50), DateTime_Birth DATETIME, City VARCHAR(50)); INSERT INTO student VALUES (1, 'Mansi Shah', '2010-01-01 18:39:09', 'Pune'), (2, 'Tejal Wagh', '2010-03-04 05:13:19', 'Nasik'), (3, 'Sejal Kumari', '2010-05-01 10:31:07', 'Mumbai'), (4, 'Sonal Jain', '2010-09-09 17:17:07', 'Shimla'), (5, 'Surili Maheshwari', '2010-07-10 20:45:18', 'Surat'); 1 SELECT ID FROM student; 2 SELECT Name FROM student; 3 SELECT DateTime_Birth FROM student; 4 SELECT CURDATE() AS currentdate; 5 SELECT DATE_ADD(DateTime_Birth, INTERVAL 3 DAY) AS date_added FROM student; 6 SELECT DATE_FORMAT(DateTime_Birth, "%Y-%m-%d") AS formatted_date FROM student; 7 SELECT DATEDIFF(CURDATE(), DATE(DateTime_Birth)) AS date_diff FROM student; 8 SELECT DAY(DateTime_Birth) AS day_value FROM student; 9 SELECT DAYNAME(DateTime_Birth) AS day_name FROM student; 10 SELECT DAYOFMONTH(DateTime_Birth) AS day_of_month FROM student; 11 SELECT DAYOFWEEK(DateTime_Birth) AS day_of_week FROM student; 12 SELECT DAYOFYEAR(DateTime_Birth) AS day_of_year FROM student; 13 SELECT FROM_DAYS(737846) AS from_days_date FROM student; 14 SELECT HOUR(DateTime_Birth) AS hour_value FROM student; 15 SELECT LAST_DAY(DateTime_Birth) AS last_day_of_month FROM student; 16 SELECT NOW() AS current_datetime FROM student; </pre>

```

17 SELECT PERIOD_ADD(202201, 3) AS period_added FROM student;
18 SELECT PERIOD_DIFF(202203, 202201) AS period_difference FROM student;
19 SELECT QUARTER(DateTime_Birth) AS quarter_value FROM student;
20 SELECT SECOND(DateTime_Birth) AS second_value FROM student;
21 SELECT STR_TO_DATE('2022-05-20', '%Y-%m-%d') AS string_to_date FROM student;
22 SELECT SUBDATE(DateTime_Birth, INTERVAL 2 DAY) AS date_subtracted FROM student;
23 SELECT SUBTIME(DateTime_Birth, '03:15:00') AS time_subtracted FROM student;
24 SELECT SYSDATE() AS system_date FROM student;
25 SELECT TIME(DateTime_Birth) AS time_value FROM student;
26 SELECT TIME_FORMAT(DateTime_Birth, '%H:%i:%s') AS formatted_time FROM student;
27 SELECT TIME_TO_SEC(DateTime_Birth) AS time_to_seconds FROM student;
28 SELECT TIMEDIFF(NOW(), DateTime_Birth) AS time_difference FROM student;
29 SELECT TIMESTAMP('2022-04-10') AS timestamp_value FROM student;
30 SELECT TO_DAYS('2022-06-15') AS to_days_value FROM student;
31 SELECT WEEKDAY(DateTime_Birth) AS weekday_index FROM student;
32 SELECT WEEK(DateTime_Birth) AS week_value FROM student;
33 SELECT WEEKOFYEAR(DateTime_Birth) AS week_of_year_value FROM student;

```

Conditional Functions

```

CREATE TABLE sample_table (
    column1 INT,
    column2 VARCHAR(50),
    column3 DATE
);

INSERT INTO sample_table VALUES
(1, 'apple', '2022-01-01'),
(2, 'banana', '2022-02-15'),
(3, 'orange', '2022-03-20'),
(4, 'grape', '2022-04-10'),
(5, 'kiwi', '2022-05-05');

34 SELECT * FROM sample_table WHERE column1 > 2 AND column3 > '2022-03-01';
35 SELECT * FROM sample_table WHERE column1 = 2 OR column2 = 'orange';
36 SELECT * FROM sample_table WHERE (column1 > 2 AND column3 > '2022-03-01') OR column2 = 'banana';
37 SELECT * FROM sample_table WHERE column2 LIKE 'a%';
38 SELECT * FROM sample_table WHERE column1 IN (2, 4);
39 SELECT * FROM sample_table WHERE column1 > ANY (SELECT column1 FROM sample_table WHERE column3 > '2022-03-01');
40 SELECT * FROM sample_table WHERE EXISTS (SELECT * FROM sample_table WHERE column1 = 3);
41 SELECT * FROM sample_table WHERE NOT column1 = 2;
42 SELECT * FROM sample_table WHERE column1 <> 2;
43 SELECT * FROM sample_table WHERE column2 IS NULL;
44 SELECT * FROM sample_table WHERE column2 IS NOT NULL;
45 SELECT * FROM sample_table WHERE column1 BETWEEN 2 AND 4;

```

Control-flow-functions

- 1 Case operator
- 2 IF()
- 3 IFNULL()
- 4 NULLIF()

1 CASE Operator

The CASE statement is utilized to implement a complex conditional construct within a stored program.

- 1 **Conditional Logic:** Allows the implementation of conditional logic directly within a query. Invaluable for handling various scenarios and producing different results based on specified conditions.
- 2 **Data Transformation:** Can be used to transform data on-the-fly. Example: Categorizing or labeling data based on specific criteria.
- 3 **Dynamic Column Selection:** Enables the dynamic selection of different columns based on specific conditions.
- 4 **Contingency Plans:** Allows the setup of contingency plans in case certain conditions are not met.
- 5 **Customized Reports:** Crucial in generating customized reports where data formatting or labeling varies based on specific criteria.
- 6 **Streamlining Data Processing:** Helps in streamlining data processing, potentially avoiding the need for more complex joins or subqueries. Can lead to more efficient queries.

Syntax:

Syntax 1: Simple CASE statement

```
CASE value
    WHEN [compare_value] THEN result
    [WHEN [compare_value] THEN result ...]
    [ELSE result]
END
```

Syntax 2: Searched CASE statement

```
CASE
    WHEN [condition] THEN result
    [WHEN [condition] THEN result ...]
    [ELSE result]
END
```

Explanation

- The first syntax returns the result where `value = compare_value`.
- The second syntax returns the result for the first condition that is true.
- The list of corresponding SQL statements will execute when a search condition evaluates to true.
- The statement list in the 'ELSE' part will execute when no search condition matches.

- If there is no matching value found in the 'ELSE' part, 'NULL' will be returned.
- Each statement list can contain one or more statements, and no empty statement list is allowed.

Conditions:
<p>1. Simple CASE Statement:</p> <ul style="list-style-type: none"> - When the condition is met (CASE 1), it returns the corresponding value. - When the condition is not met (CASE 4), it returns the value in the ELSE part.
46. SELECT CASE 1 WHEN 1 THEN 'this is case one' WHEN 2 THEN 'this is case two' ELSE 'this is not in the case' END as 'how to execute case statement';
47. SELECT CASE 4 WHEN 1 THEN 'this is case one' WHEN 2 THEN 'this is case two' ELSE 'this is not in the case' END as 'how to execute case statement';
<p>2. CASE Statement with Matching Condition:</p> <ul style="list-style-type: none"> - When the condition is met (CASE 2), it returns the corresponding value.
48. SELECT CASE 2 WHEN 1 THEN 'this is case one' WHEN 2 THEN 'this is case two' ELSE 'this is not in the case' END as 'how to execute case statement';
<p>3. CASE Statement with Comparison Operators:</p> <ul style="list-style-type: none"> - Using greater than and less than operators to evaluate conditions.
49. SELECT CASE WHEN 2>3 THEN 'this is true' ELSE 'this is false' END;
50. SELECT CASE WHEN 2<3 THEN 'this is true' ELSE 'this is false' END;
<p>4. CASE Statement with No Matching Conditions:</p> <ul style="list-style-type: none"> - If none of the conditions are satisfied, it returns NULL.
51. SELECT CASE BINARY 'A' WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;
Scenario Based Questions
Question 1: Simple CASE Statement

Consider a table named `student_grades` with columns `student_id` and `grade`. Write a SQL query using the CASE statement to display the grades of students based on the following conditions:

- If the grade is 90 or above, display 'A'.
- If the grade is between 80 and 89, display 'B'.
- If the grade is between 70 and 79, display 'C'.
- If the grade is below 70, display 'F'.

Question 2: CASE Statement with Matching Condition

Assume a table named `employee_data` with columns `employee_id` and `salary`. Write a SQL query using the CASE statement to categorize employees based on their salary:

- If the salary is greater than \$100,000, categorize as 'High Income'.
- If the salary is between \$50,000 and \$100,000, categorize as 'Moderate Income'.
- If the salary is below \$50,000, categorize as 'Low Income'.

Question 3: CASE Statement with Comparison Operators

Consider a table named `product_inventory` with columns `product_id` and `quantity`. Write a SQL query using the CASE statement to determine the availability of products:

- If the quantity is greater than 50, display 'In Stock'.
- If the quantity is between 10 and 50, display 'Low Stock'.
- If the quantity is 10 or below, display 'Out of Stock'.

Question 4: CASE Statement with No Matching Conditions

Assume a table named `customer_orders` with columns `order_id` and `order_status`. Write a SQL query using the CASE statement to categorize orders based on their status:

- If the order status is 'Shipped', display 'Order Shipped'.
- If the order status is 'Processing', display 'Order Processing'.
- If the order status is 'Cancelled', display 'Order Cancelled'.
- For any other order status, display 'Unknown Status'.

Question 5: CASE Statement with Multiple Conditions

Create a table named `temperature_readings` with columns `location` and `temperature`. Write a SQL query using the CASE statement to categorize temperature readings based on the following conditions:

- If the temperature is above 30 degrees Celsius, display 'Hot'.
- If the temperature is between 20 and 30 degrees Celsius, display 'Moderate'.
- If the temperature is below 20 degrees Celsius, display 'Cool'.

Solution

52.

```
CREATE TABLE student_grades (
    student_id INT,
    grade INT
);
```

53.

```
INSERT INTO student_grades (student_id, grade) VALUES  
(1, 95),  
(2, 85),  
(3, 75),  
(4, 60);
```

54.

```
CREATE TABLE employee_data (  
    employee_id INT,  
    salary DECIMAL(10, 2)  
);
```

55.

```
INSERT INTO employee_data (employee_id, salary) VALUES  
(101, 120000),  
(102, 75000),  
(103, 45000),  
(104, 110000);
```

56.

```
CREATE TABLE product_inventory (  
    product_id INT,  
    quantity INT  
);
```

57.

```
INSERT INTO product_inventory (product_id, quantity) VALUES  
(1, 75),  
(2, 20),  
(3, 5),  
(4, 100);
```

58.

```
CREATE TABLE customer_orders (  
    order_id INT,  
    order_status VARCHAR(20)  
);
```

59.

```
INSERT INTO customer_orders (order_id, order_status) VALUES  
(1, 'Shipped'),  
(2, 'Processing'),  
(3, 'Cancelled'),  
(4, 'Pending');
```

60.

```
CREATE TABLE temperature_readings (
    location VARCHAR(50),
    temperature DECIMAL(5, 2)
);
```

61.

```
INSERT INTO temperature_readings (location, temperature) VALUES
('City A', 32.5),
('City B', 25.0),
('City C', 18.5),
('City D', 28.0);
```

-- Display grades based on conditions

62.

```
SELECT student_id, grade,
CASE
    WHEN grade >= 90 THEN 'A'
    WHEN grade BETWEEN 80 AND 89 THEN 'B'
    WHEN grade BETWEEN 70 AND 79 THEN 'C'
    ELSE 'F'
END AS grade_category
FROM student_grades;
```

-- Categorize employees based on salary

63.

```
SELECT employee_id, salary,
CASE
    WHEN salary > 100000 THEN 'High Income'
    WHEN salary BETWEEN 50000 AND 100000 THEN 'Moderate Income'
    ELSE 'Low Income'
END AS income_category
FROM employee_data;
```

-- Determine product availability based on quantity

64.

```
SELECT product_id, quantity,
CASE
    WHEN quantity > 50 THEN 'In Stock'
    WHEN quantity BETWEEN 10 AND 50 THEN 'Low Stock'
    ELSE 'Out of Stock'
END AS availability_status
FROM product_inventory;
```

-- Categorize orders based on status

65.

```

SELECT order_id, order_status,
CASE
    WHEN order_status = 'Shipped' THEN 'Order Shipped'
    WHEN order_status = 'Processing' THEN 'Order Processing'
    WHEN order_status = 'Cancelled' THEN 'Order Cancelled'
    ELSE 'Unknown Status'
END AS order_category
FROM customer_orders;

```

-- Categorize temperature readings based on conditions

66.

```

SELECT location, temperature,
CASE
    WHEN temperature > 30 THEN 'Hot'
    WHEN temperature BETWEEN 20 AND 30 THEN 'Moderate'
    ELSE 'Cool'
END AS temperature_category
FROM temperature_readings;

```

2. IF() Control Flow

The 'IF()' function in MySQL is a powerful tool for introducing conditional logic into queries, and it can be used in various scenarios to handle different conditions and produce different results. Here are some common use cases and examples:

Conditional Logic in Queries
<ul style="list-style-type: none"> The 'IF()' function allows you to implement conditional logic directly within a query. For example: <code>SELECT IF(1 > 3, 'true', 'false');</code>
Dynamic Column Selection
<ul style="list-style-type: none"> You can use 'IF()' to dynamically select different columns based on specific conditions.
Data Validation
<ul style="list-style-type: none"> 'IF()' can be used to validate data before inserting or updating a table, ensuring that only valid data is processed.
Contingency Plans
<ul style="list-style-type: none"> Setting up contingency plans in case certain conditions are not met.
Aggregate Functions with IF()
<ul style="list-style-type: none"> When used with aggregate functions like 'SUM()' or 'COUNT()', 'IF()' can selectively include or exclude certain records from the calculation.

Lab Performances

67.

```

CREATE TABLE books (
    book_id VARCHAR(10),
    book_name VARCHAR(50),
    isbn_no VARCHAR(11),
    cate_id VARCHAR(10),
    aut_id VARCHAR(10),
    pub_id VARCHAR(10),

```

```
dt_of_pub DATE,  
pub_lang VARCHAR(20),  
no_page INT,  
book_price DECIMAL(8, 2)  
);
```

68.

```
INSERT INTO books VALUES  
('BK001', 'Introduction to Electrodynamics', '0000979001', 'CA001', 'AUT001', 'P003', '2001-05-08', 'English', 201, 85.00),  
('BK002', 'Understanding of Steel Construction', '0000979002', 'CA002', 'AUT002', 'P001', '2003-07-15', 'English', 300, 105.50),  
('BK003', 'Guide to Networking', '0000979003', 'CA003', 'AUT003', 'P002', '2002-09-10', 'Hindi', 510, 200.00),  
('BK004', 'Transfer of Heat and Mass', '0000979004', 'CA002', 'AUT004', 'P004', '2004-02-16', 'English', 600, 250.00),  
('BK005', 'Conceptual Physics', '0000979005', 'CA001', 'AUT005', 'P006', '2003-07-16', NULL, 345, 145.00),  
('BK006', 'Fundamentals of Heat', '0000979006', 'CA001', 'AUT006', 'P005', '2003-08-10', 'German', 247, 112.00),  
('BK007', 'Advanced 3d Graphics', '0000979007', 'CA003', 'AUT007', 'P002', '2004-02-16', 'Hindi', 165, 56.00),  
('BK008', 'Human Anatomy', '0000979008', 'CA005', 'AUT008', 'P006', '2001-05-17', 'German', 88, 50.50),  
('BK009', 'Mental Health Nursing', '0000979009', 'CA005', 'AUT009', 'P007', '2004-02-10', 'English', 350, 145.00),  
('BK010', 'Fundamentals of Thermodynamics', '0000979010', 'CA002', 'AUT010', 'P007', '2002-10-14', 'English', 400, 225.00),  
('BK011', 'The Experimental Analysis of Cat', '0000979011', 'CA004', 'AUT011', 'P005', '2007-06-09', 'French', 225, 95.00),  
('BK012', 'The Nature of World', '0000979012', 'CA004', 'AUT005', 'P008', '2005-12-20', 'English', 350, 88.00),  
('BK013', 'Environment a Sustainable Future', '0000979013', 'CA004', 'AUT012', 'P001', '2003-10-27', 'German', 165, 100.00),  
('BK014', 'Concepts in Health', '0000979014', 'CA005', 'AUT013', 'P004', '2001-08-25', NULL, 320, 180.00),  
('BK015', 'Anatomy & Physiology', '0000979015', 'CA005', 'AUT014', 'P008', '2000-10-10', 'Hindi', 225, 135.00),  
('BK016', 'Networks and Telecommunications', '00009790_16', 'CA003', 'AUT015', 'P003', '2002-01-01', 'French', 95, 45.00);
```

69.

```
select * from books;
```

70.

```
SELECT book_name,  
IF(pub_lang="English", "English Book", "Other Language")  
AS Language  
FROM books;
```

71.

```
SELECT book_name, isbn_no,
IF((SELECT COUNT(*) FROM books WHERE pub_lang='English')>
(SELECT COUNT(*) FROM books WHERE pub_lang<>'English'),
(CONCAT("Pages: ",no_page)),(CONCAT("Price: ",book_price)))
AS "Page / Price"
FROM books;
```

72.

```
SELECT book_id, book_name,
IF(pub_lang IS NULL,'N/A',pub_lang) AS "Pub. Language"
FROM books;
```

73.

```
SELECT book_id, book_name, pub_lang
FROM books;
```

74.

```
CREATE TABLE purchase (
    invoice_no VARCHAR(10),
    invoice_dt DATE,
    ord_no VARCHAR(20),
    ord_date DATE,
    receive_dt DATE,
    book_id VARCHAR(10),
    book_name VARCHAR(50),
    pub_lang VARCHAR(20),
    cate_id VARCHAR(10),
    receive_qty INT,
    purch_price DECIMAL(8, 2),
    total_cost DECIMAL(10, 2)
);
```

75.

```
INSERT INTO purchase VALUES
('INV0001', '2008-07-15', 'ORD/08-09/0001', '2008-07-06', '2008-07-19', 'BK001', 'Introduction to
Electrodynamics', 'English', 'CA001', 15, 75.00, 1125.00),
('INV0002', '2008-08-25', 'ORD/08-09/0002', '2008-08-09', '2008-08-28', 'BK004', 'Transfer of
Heat and Mass', 'English', 'CA002', 8, 55.00, 440.00),
('INV0003', '2008-09-20', 'ORD/08-09/0003', '2008-09-15', '2008-09-23', 'BK005', 'Conceptual
Physics', NULL, 'CA001', 20, 20.00, 400.00),
('INV0004', '2007-08-30', 'ORD/07-08/0005', '2007-08-22', '2007-08-30', 'BK004', 'Transfer of
Heat and Mass', 'English', 'CA002', 15, 35.00, 525.00),
('INV0005', '2007-07-28', 'ORD/07-08/0004', '2007-06-25', '2007-07-30', 'BK001', 'Introduction to
Electrodynamics', 'English', 'CA001', 8, 25.00, 200.00),
('INV0006', '2007-09-24', 'ORD/07-08/0007', '2007-09-20', '2007-09-30', 'BK003', 'Guide to
Networking', 'Hindi', 'CA003', 20, 45.00, 900.00);
```

76.

```
select * from purchase;
```

77.

```
SELECT SUM(IF(pub_lang = 'English',1,0)) AS English,  
       SUM(IF(pub_lang <> 'English',1,0)) AS "Non English"  
FROM purchase;
```

78.

```
CREATE TABLE publishers (  
    pub_id VARCHAR(10),  
    pub_name VARCHAR(50),  
    pub_city VARCHAR(30),  
    country VARCHAR(30),  
    country_office VARCHAR(30),  
    no_of_branch INT,  
    estd DATE  
)
```

79.

```
INSERT INTO publishers VALUES  
('P001', 'Jex Max Publication', 'New York', 'USA', 'New York', 15, '1969-12-25'),  
('P002', 'BPP Publication', 'Mumbai', 'India', 'New Delhi', 10, '1985-10-01'),  
('P003', 'New Harrold Publication', 'Adelaide', 'Australia', 'Sydney', 6, '1975-09-05'),  
('P004', 'Ultra Press Inc.', 'London', 'UK', 'London', 8, '1948-07-10'),  
('P005', 'Mountain Publication', 'Houstan', 'USA', 'Sun Diego', 25, '1975-01-01'),  
('P006', 'Summer Night Publication', 'New York', 'USA', 'Atlanta', 10, '1990-12-10'),  
('P007', 'Pieterson Grp. of Publishers', 'Cambridge', 'UK', 'London', 6, '1950-07-15'),  
('P008', 'Novel Publisher Ltd.', 'New Delhi', 'India', 'Bangalore', 10, '2000-01-01');
```

80.

```
select * from publishers;
```

81.

```
SELECT COUNT(IF(country = 'USA',1,NULL)) USA,  
       COUNT(IF(country = 'UK',1,NULL)) UK,  
       COUNT(IF(country = 'India',1,NULL)) India,  
       COUNT(IF(country = 'Australia',1,NULL)) Australia  
FROM publishers;
```

Another way to achieve the similar result you can use the GROUP BY clause and the COUNT function without using the IF function, the display report is quite different.

82.

```
SELECT country, COUNT(country)
  FROM publishers
 GROUP BY country;
```

EXPERIMENT-6

Lab 6: Use of different SQL clauses and join

Objective: To understand the use of group by and having clause and execute the SQL commands using JOIN

Understanding JOINs in MySQL

- A join enables you to retrieve records from two (or more) logically related tables in a single result set.
- JOIN clauses are used to return the rows of two or more queries using two or more tables that share a meaningful relationship based on a common set of values.
- These values are usually the same column name and datatype that appear in both the participating tables being joined. These columns, or possibly a single column from each table, are called the join key or common key.
- Mostly but not all of the time, the join key is the primary key of one table and a foreign key in another table. The join can be performed as long as the data in the columns are matching.
- It can be difficult when the join involves more than two tables. It is a good practice to think of the query as a series of two table joins when the involvement of three or more tables in joins.

MySQL JOIN

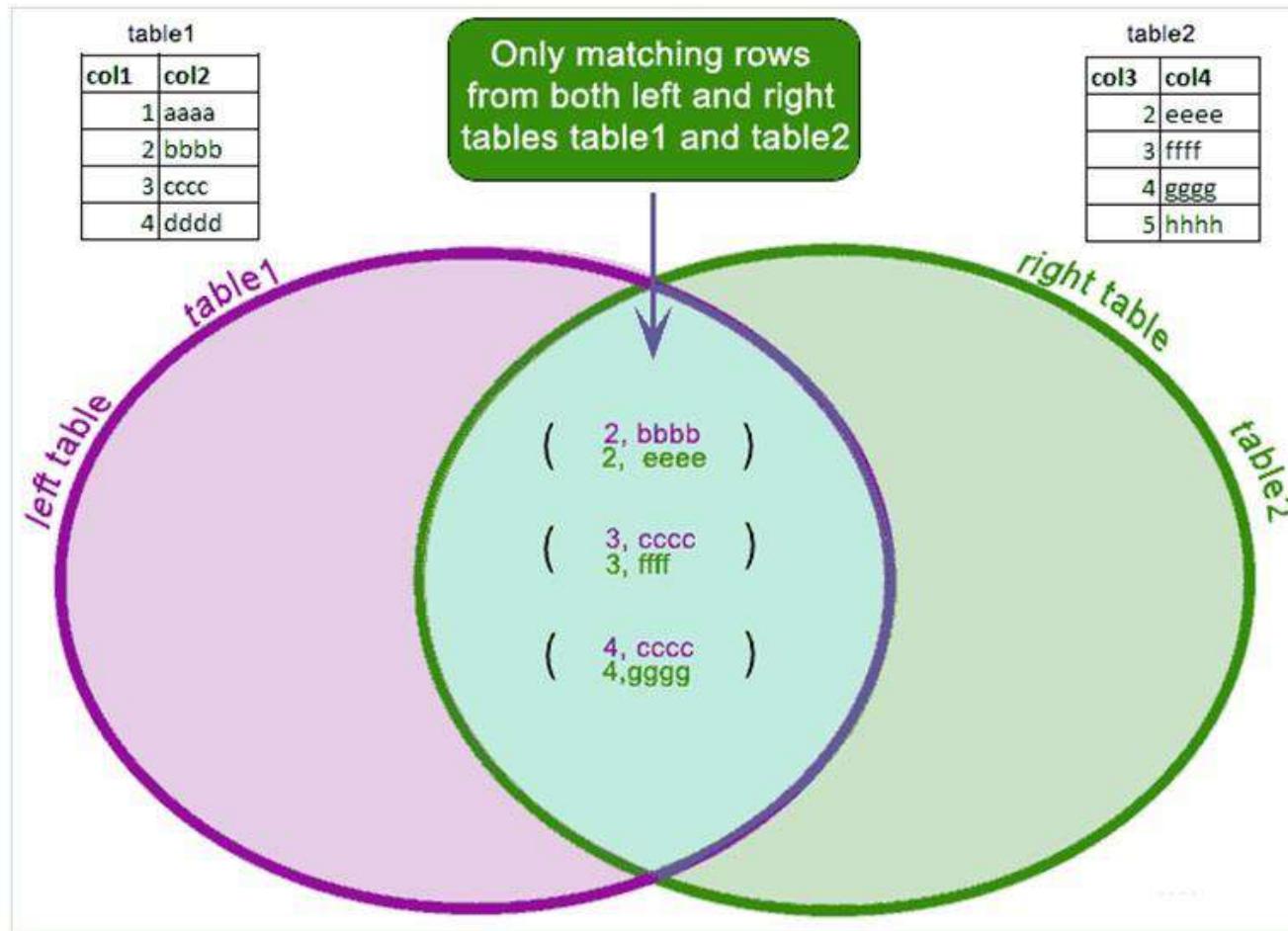
MySQL supports the following JOIN syntaxes for the table_references (A table reference is also known as a join expression.) part of SELECT statements and multiple-table UPDATE and DELETE statements :

Types of MySQL Joins

- 1. INNER JOIN**
- 2. LEFT JOIN**
- 3. RIGHT JOIN**
- 4. STRAIGHT JOIN**
- 5. CROSS JOIN**
- 6. NATURAL JOIN**

1. INNER JOIN

In MySQL the INNER JOIN selects all rows from both participating tables to appear in the result if and only if both tables meet the conditions specified in the ON clause. JOIN, CROSS JOIN, and INNER JOIN are syntactic equivalents. In standard SQL, they are not equivalent. INNER JOIN is used with an ON clause, CROSS JOIN is used otherwise.



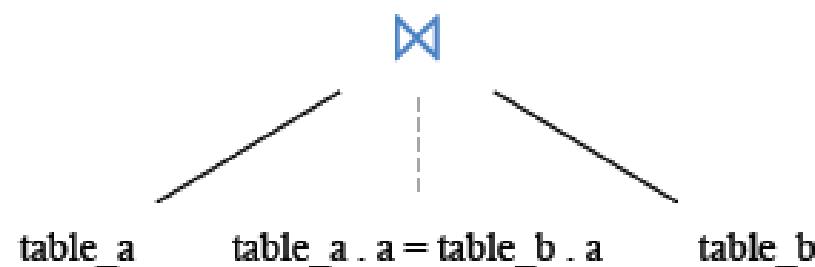
INNER JOIN

A	M	A	N
1	m	2	p
2	n	3	q
4	o	5	r

table_A table_B

```
SELECT * FROM table_A  
INNER JOIN table_B  
ON table_A.A=table_B.A;  
Output 1
```

table_a ⋈_{table_a . a = table_b . a} table_b



book_mast

book_id	book_name	isbn_no	cate_id	aut_id	pub_id	dt_of_pub	pub_lang	no_page	book_price
BK001	Introduction to Electrodynamics	0000979001	CA001	AUT001	P003	2001-05-08	English	201	85.00
BK002	Understanding of Steel Construction	0000979002	CA002	AUT002	P001	2003-07-15	English	300	105.50
BK003	Guide to Networking	0000979003	CA003	AUT003	P002	2002-09-10	Hindi	510	200.00
BK004	Transfer of Heat and Mass	0000979004	CA002	AUT004	P004	2004-02-16	English	600	250.00
BK005	Conceptual Physics	0000979005	CA001	AUT005	P006	2003-07-16	NULL	345	145.00
BK006	Fundamentals of Heat	0000979006	CA001	AUT006	P005	2003-08-10	German	247	112.00
BK007	Advanced 3d Graphics	0000979007	CA003	AUT007	P002	2004-02-16	Hindi	165	56.00
BK008	Human Anatomy	0000979008	CA005	AUT008	P006	2001-05-17	German	88	50.50
BK009	Mental Health Nursing	0000979009	CA005	AUT009	P007	2004-02-10	English	350	145.00
BK010	Fundamentals of Thermodynamics	0000979010	CA002	AUT010	P007	2002-10-14	English	400	225.00
BK011	The Experimental Analysis of Cat	0000979011	CA004	AUT011	P005	2007-06-09	French	225	95.00
BK012	The Nature of World	0000979012	CA004	AUT005	P008	2005-12-20	English	350	88.00
BK013	Environment a Sustainable Future	0000979013	CA004	AUT012	P001	2003-10-27	German	165	100.00
BK014	Concepts in Health	0000979014	CA005	AUT013	P004	2001-08-25	NULL	320	180.00
BK015	Anatomy & Physiology	0000979015	CA005	AUT014	P008	2000-10-10	Hindi	225	135.00
BK016	Networks and Telecommunications	00009790_16	CA003	AUT015	P003	2002-01-01	French	95	45.00

Category

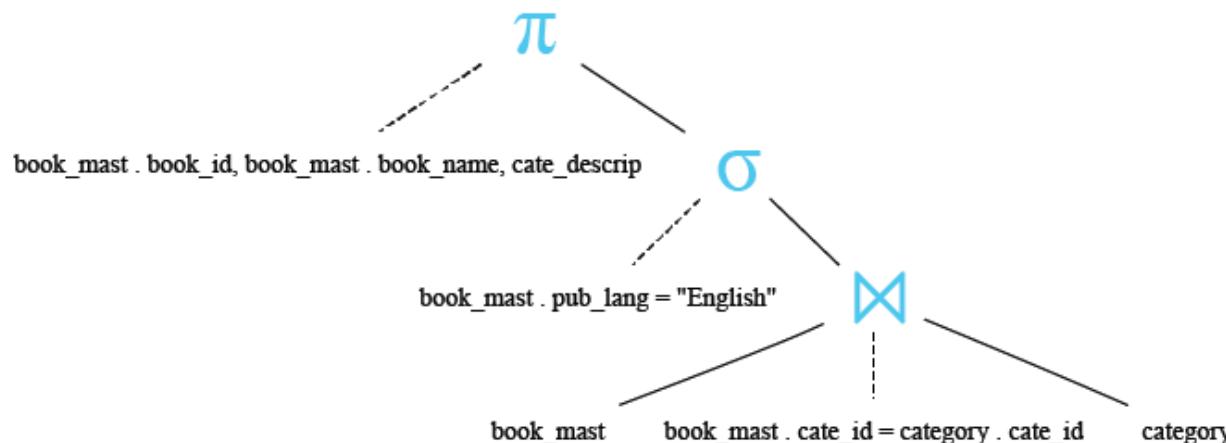
cate_id	cate_descrip
CA001	Science
CA002	Technology
CA003	Computers
CA004	Nature
CA005	Medical

```

SELECT book_mast.book_id,book_mast.book_name,cate_descrip
FROM book_mast
INNER JOIN category
ON book_mast.cate_id=category.cate_id
WHERE book_mast.pub_lang="English";

```

Output 2

$$\begin{array}{l}
\pi_{book_mast . book_id, book_mast . book_name, cate_descrip} \\
\sigma_{book_mast . pub_lang = "English"}(book_mast \bowtie book_mast . cate_id = category . cate_id^{category})
\end{array}$$


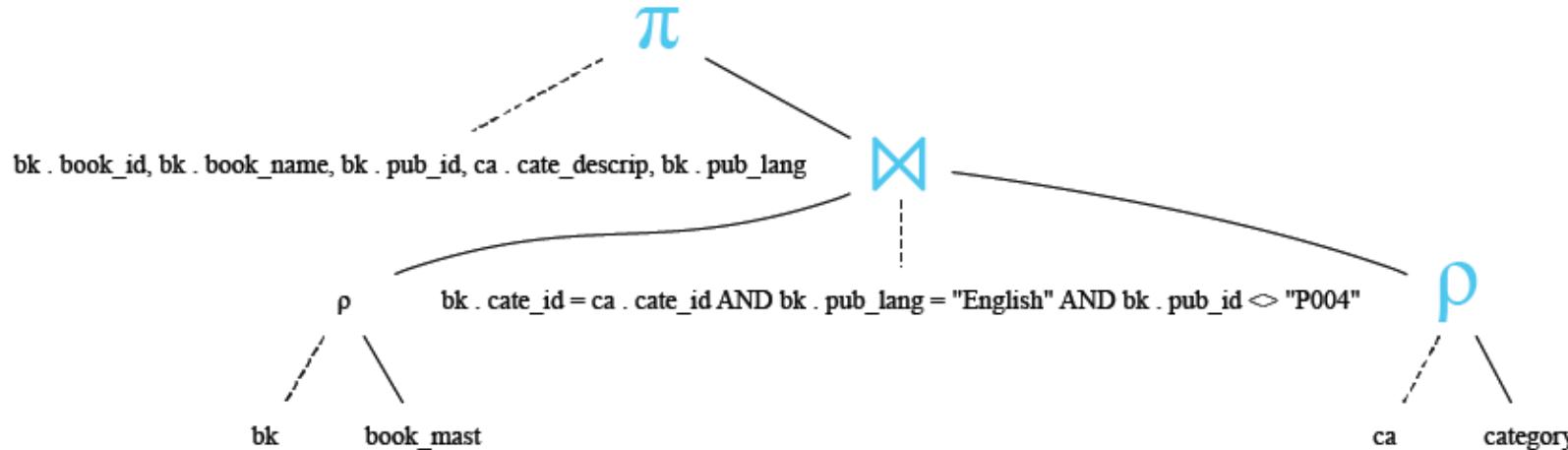
INNER JOIN with alias

```

SELECT bk.book_id,bk.book_name,bk.pub_id,ca.cate_descrip,bk.pub_lang
FROM book_mast AS bk
INNER JOIN category AS ca ON bk.cate_id=ca.cate_id AND
bk.pub_lang="English" AND bk.pub_id<>'P004';

```

Output 3

$$\pi_{bk.book_id, bk.book_name, bk.pub_id, ca.cate_descrip, bk.pub_lang} \\ (\rho_{bk \bowtie book_mast \bowtie bk.cate_id = ca.cate_id \text{ AND } bk.pub_lang = "English" \text{ AND } bk.pub_id \neq "P004"} \\ \rho_{ca \text{ category}})$$


INNER JOIN using three tables

table - doctors

docid	dname
1	A.VARMA
2	D.GOMES

table - specialize

spid	desc	docid
1	special1	1
2	special2	2

table - timeschedule

tid	tday	sit_time	docid
1	MON	17:00:00	1
2	WED	08:00:00	1
3	TUE	16:00:00	2
4	FRI	09:00:00	2

```

SELECT a.docid,a.dname,
      b.desc,c.tday,c.sit_time
FROM doctors a
INNER JOIN specialize b
ON a.docid=b.docid
INNER JOIN timeschedule c
ON a.docid=c.docid
WHERE a.docid=1 AND c.tday='WED';

```

Output 4

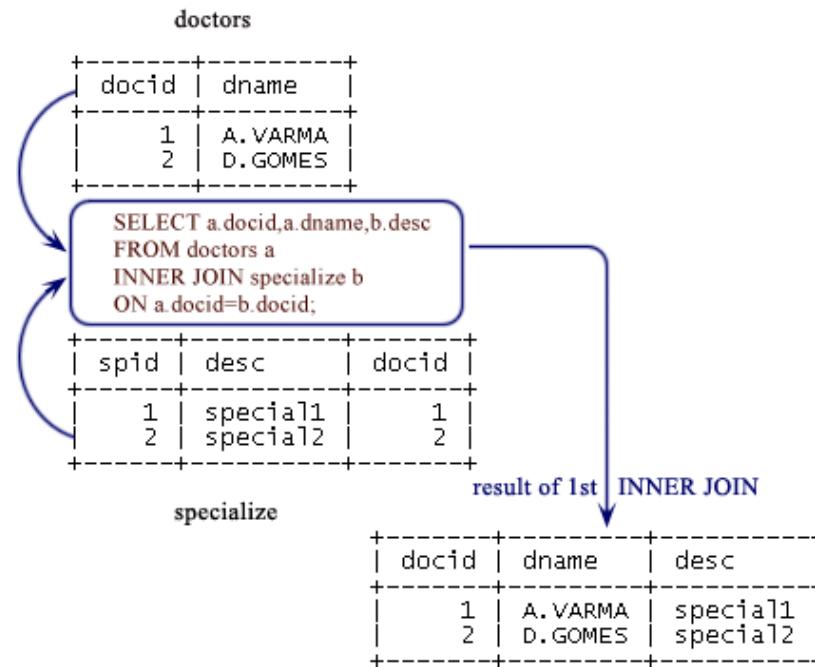
Explanation :

Step-1

```

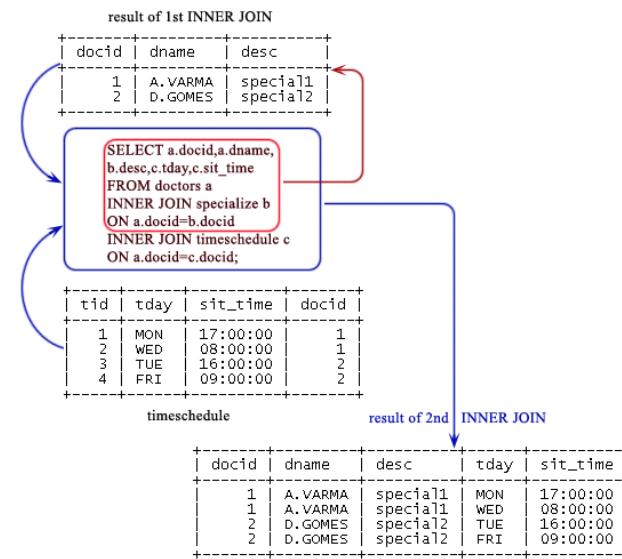
SELECT a.docid,a.dname,b.desc
FROM doctors a
INNER JOIN specialize b
ON a.docid=b.docid;

```



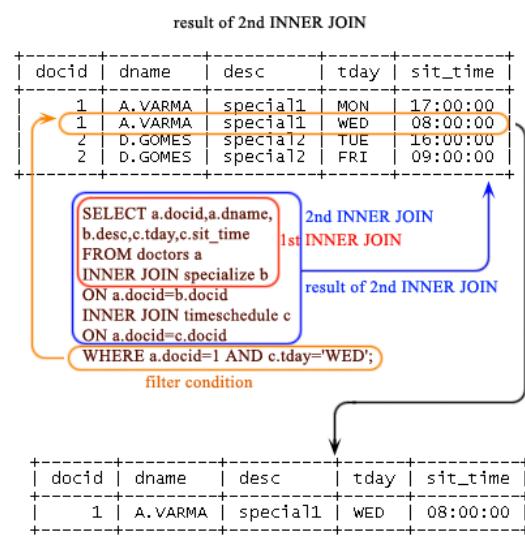
Step-2

```
SELECT a.docid,a.dname,  
b.desc,c.tday,c.sit_time  
FROM doctors a  
INNER JOIN specialize b  
ON a.docid=b.docid  
INNER JOIN timeschedule c  
ON a.docid=c.docid;
```



Step-3

```
SELECT a.docid,a.dname,  
b.desc,c.tday,c.sit_time  
FROM doctors a  
INNER JOIN specialize b  
ON a.docid=b.docid  
INNER JOIN timeschedule c  
ON a.docid=c.docid  
WHERE a.docid=1 AND c.tday='WED';
```



2. LEFT JOIN

The MySQL LEFT JOIN joins two tables and fetches rows based on a condition, which is matching in both the tables and the unmatched rows will also be available from the table written before the JOIN clause.

```
SELECT * FROM table_A  
LEFT JOIN table_B  
ON table_A.A=table_B.A;
```

Output 5

```
SELECT bk1.book_name,bk1.isbn_no,bk1.book_price,bk1.pub_lang  
FROM book_mast bk1  
LEFT JOIN book_mast bk2 ON bk1.book_price<bk2.book_price  
WHERE bk2.pub_lang='German';
```

Output 6

3. RIGHT JOIN

The RIGHT JOIN is such a join which specifies that all records be fetched from the table on the right side of the join statement, even if the table on the left has no matching record. In this case, the columns from the left table return NULL values.

```
SELECT * FROM table_A  
RIGHT JOIN table_B  
ON table_A.A=table_B.A;
```

Output 7

table111

id	aval1
1	405
2	401
3	200
4	400

table112

id	bval1	bval2
701	405	16
704	409	14
706	403	13
709	401	12

table113

id	cval1
3	17
2	12
5	15
1	16

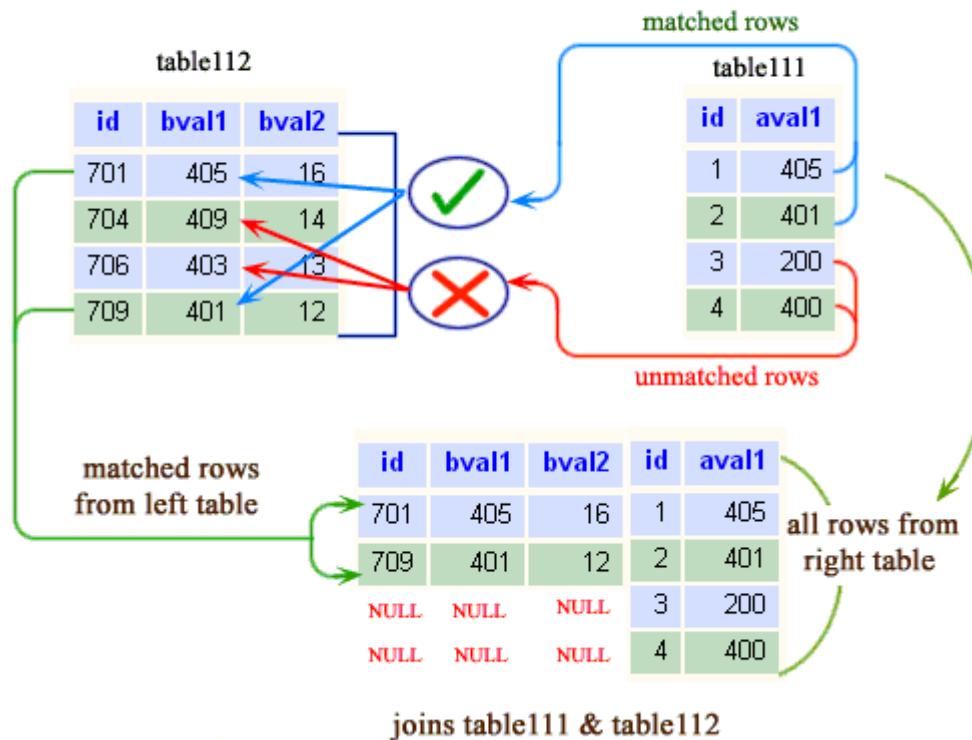
```
SELECT table112.id,table112.bval1,table112.bval2,  
table111.id,table111.aval1  
FROM table112  
RIGHT JOIN table111  
ON table112.bval1=table111.aval1;
```

Output 8

```
SELECT table112.id,table112.bval1,table112.bval2,  
table111.id,table111.aval1  
FROM table112  
RIGHT OUTER JOIN table111  
ON table112.bval1=table111.aval1;
```

Output 9

MySQL RIGHT JOIN



```
SELECT table113.id,table113.cval1,  
table111.id,table111.aval1  
FROM table113  
RIGHT OUTER JOIN table111  
USING(id);  
Output 10
```

4. STRAIGHT JOIN

A STRAIGHT_JOIN is such a join which scans and combines matching rows (if specified any condition) which are stored in associated tables otherwise it behaves like an INNER JOIN or JOIN of without any condition.

```
SELECT * FROM table_A  
STRAIGHT JOIN table_B;
```

Output 11

```
SELECT table112.id,table112.bval1,table112.bval2,  
table111.id,table111.aval1  
FROM table112  
STRAIGHT_JOIN table111;
```

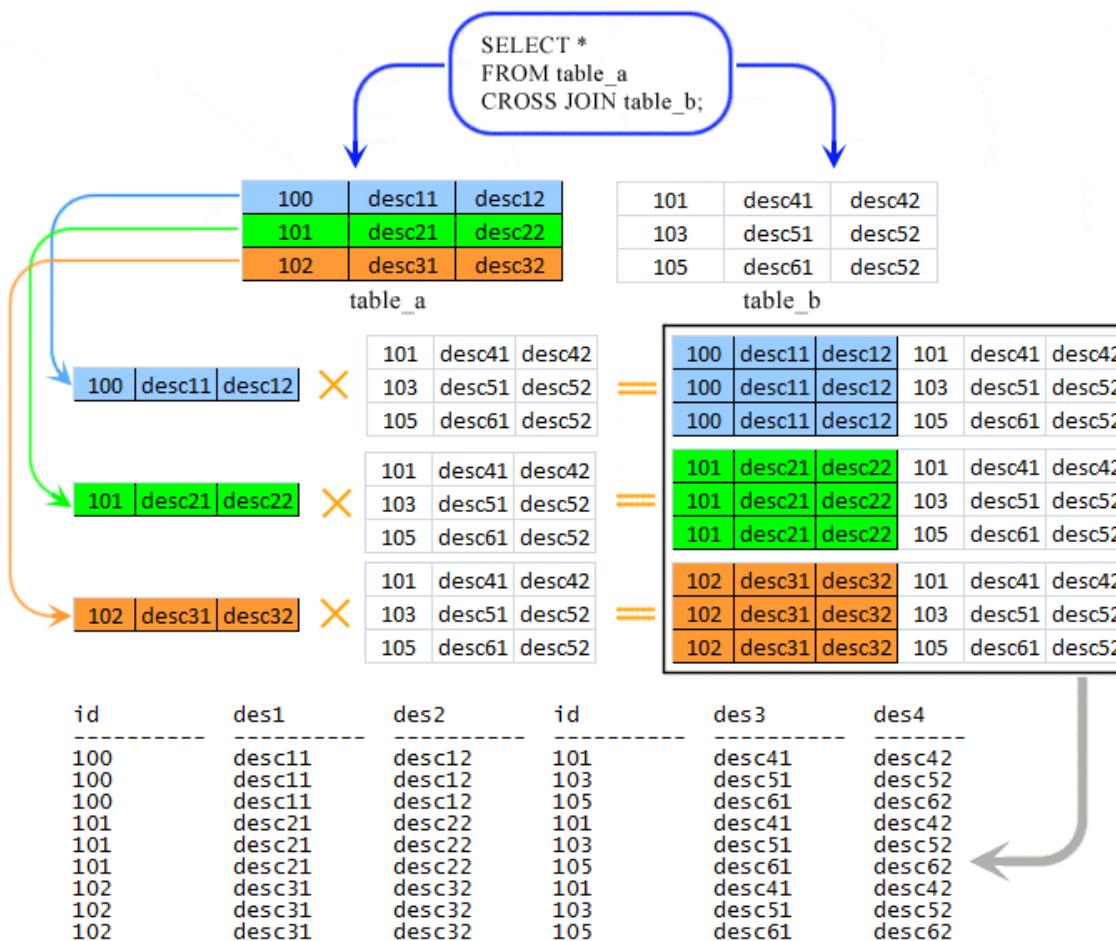
Output 12

```
SELECT table113.id,table113.cval1,  
table111.id,table111.aval1  
FROM table113  
STRAIGHT_JOIN table111  
ON table113.id=table111.id;
```

Output 13

5. CROSS JOIN

A CROSS JOIN is such a join which specifies the complete cross product of two tables. For each record in the first table, all the records in the second table are joined, creating a potentially huge result set. This command has the same effect as leaving off the join condition, and its result set is also known as a Cartesian product.



```
SELECT * FROM table_A  
CROSS JOIN table_B;
```

Output 14

```
SELECT table112.id,table112.bval1,table112.bval2,  
table111.id,table111.aval1  
FROM table112  
CROSS JOIN table111;
```

Output 15

```
SELECT *  
FROM table111  
LEFT JOIN(table112 CROSS JOIN table113)  
ON table111.id=table113.id;
```

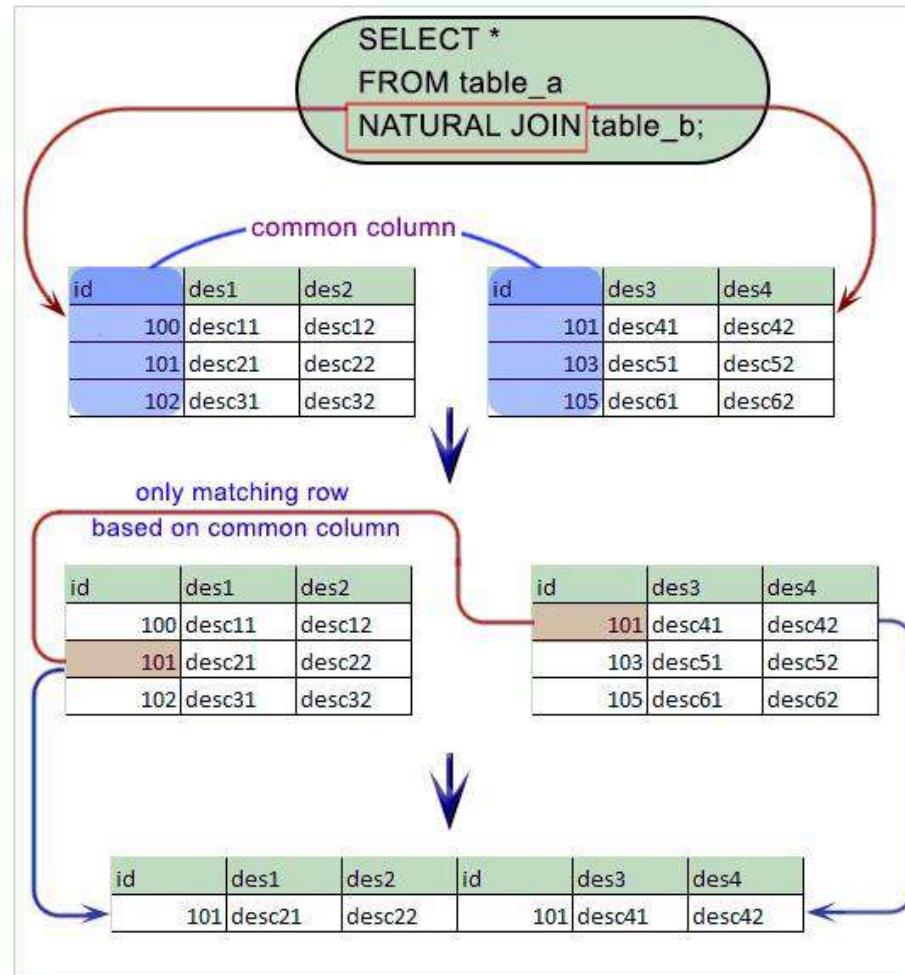
Output 16

```
SELECT table111.* ,table113.*  
FROM table111  
CROSS JOIN table113  
WHERE table111.id=table113.id;
```

Output 17

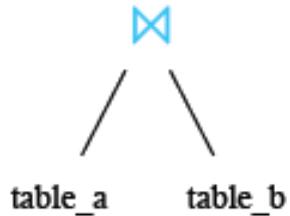
6. NATURAL JOIN

A NATURAL JOIN is such a join that performs the same task as an INNER or LEFT JOIN, in which the ON or USING clause refers to all columns that the tables to be joined have in common.



`SELECT * FROM table_A
NATURAL JOIN table_B;
Output 18`

table_a \bowtie *table_b*



NATURAL JOIN

```
SELECT id,aval1,cval1  
FROM table111  
NATURAL JOIN table113;  
Output 19
```

INNER JOIN using ON clause

```
SELECT table111.id,table111.aval1,table113.cval1  
FROM table111  
INNER JOIN table113  
ON table111.id=table113.id;  
Output 20
```

NATURAL JOIN with WHERE clause

```
SELECT id,aval1,cval1  
FROM table111  
NATURAL JOIN table113  
WHERE table111.aval1>200;  
Output 21
```

NATURAL JOIN using three tables

```
SELECT id,aval1,cval1  
FROM table111  
NATURAL JOIN table113  
natural join table114  
WHERE table111.aval1>200;
```

Output 22

Lab 7: MySQL Views

In the context of relational databases, a view is a virtual table that is based on the result of a SELECT query. A view does not store the data itself but is a saved query that can be treated as a table for querying purposes. It provides a way to simplify complex queries, encapsulate logic, and restrict access to certain columns or rows.

Some key points about views:

1. **Data Abstraction:** Views abstract the underlying complexity of the database schema by allowing users to interact with a simplified representation of the data.
2. **Security:** Views can be used to restrict access to specific columns or rows of a table, providing a layer of security. Users can be granted permission to access the view without exposing the details of the underlying tables.
3. **Simplicity and Reusability:** Views can be used to encapsulate complex SQL queries, making them easier to understand and maintain. Additionally, views can be reused in other queries or views, promoting code reusability.
4. **Dynamic Data Presentation:** Since views are dynamically generated based on the underlying tables or other views, changes in the data of the underlying tables are automatically reflected in the view.

Syntax:

```
CREATE VIEW my_view AS  
SELECT column1, column2  
FROM my_table  
WHERE column3 = 'some_condition';
```

After creating this view, you can query it as if it were a table:

Syntax:

```
SELECT * FROM my_view;
```

It's important to note that the specific syntax for creating views may vary depending on the database management system (e.g., MySQL, PostgreSQL, SQL Server) you are using.

Lab Performance Commands

Run the code and explain every Commands

1. To check the version of the database server:

SELECT VERSION();

2. To show the privileges of the current user:

SHOW PRIVILEGES;

3. To show the list of databases:

SHOW DATABASES;

4. Create View

```
CREATE TABLE user (
    userid VARCHAR(50) PRIMARY KEY,
    password VARCHAR(50),
    name VARCHAR(50)
);
```

- 5.

```
INSERT INTO user (userid, password, name)
VALUES
    ('scott123', '123@sco', 'Scott'),
    ('ferp6734', 'dloeiu@&3', 'Palash'),
    ('diana094', 'ku$j@23', 'Diana');
```

- 6.

SELECT * FROM user;

7.

```
CREATE VIEW my_v2 AS SELECT * FROM user;
```

8.

```
SELECT * FROM my_v2;
```

9.

```
CREATE VIEW my_v3 AS SELECT userid AS User_ID, password AS Password, name AS Name FROM user;
```

10.

```
SELECT * FROM my_v3;
```

11. CREATE VIEW with WHERE

```
CREATE TABLE author (
    aut_id VARCHAR(10) PRIMARY KEY,
    aut_name VARCHAR(50),
    country VARCHAR(50),
    home_city VARCHAR(50)
);
```

12.

```
INSERT INTO author (aut_id, aut_name, country, home_city)
VALUES
    ('AUT001', 'William Norton', 'UK', 'Cambridge'),
    ('AUT002', 'William Maugham', 'Canada', 'Toronto'),
    ('AUT003', 'William Anthony', 'UK', 'Leeds'),
    ('AUT004', 'S.B.Swaminathan', 'India', 'Bangalore'),
    ('AUT005', 'Thomas Morgan', 'Germany', 'Arnsberg'),
    ('AUT006', 'Thomas Merton', 'USA', 'New York'),
    ('AUT007', 'Piers Gibson', 'UK', 'London'),
```

```
('AUT008', 'Nikolai Dewey', 'USA', 'Atlanta'),  
('AUT009', 'Marquis de Ellis', 'Brazil', 'Rio De Janeiro'),  
('AUT010', 'Joseph Milton', 'USA', 'Houston'),  
('AUT011', 'John Betjeman Hunter', 'Australia', 'Sydney'),  
('AUT012', 'Evan Hayek', 'Canada', 'Vancouver'),  
('AUT013', 'E. Howard', 'Australia', 'Adelaide'),  
('AUT014', 'C. J. Wilde', 'UK', 'London'),  
('AUT015', 'Butler Andre', 'USA', 'Florida');
```

13.

```
CREATE VIEW view_author  
AS SELECT *  
FROM author  
WHERE country='USA';
```

14.

```
SELECT * FROM view_author;
```

15. CREATE VIEW with AND and OR

```
CREATE TABLE publisher (  
    pub_id VARCHAR(10) PRIMARY KEY,  
    pub_name VARCHAR(50),  
    pub_city VARCHAR(50),  
    country VARCHAR(50),  
    country_office VARCHAR(50),  
    no_of_branch INT,  
    estd DATE  
);
```

16.

```
INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office,  
no_of_branch, estd)  
VALUES  
('P001', 'Jex Max Publication', 'New York', 'USA', 'New York', 15, '1969-12-25'),
```

```
('P002', 'BPP Publication', 'Mumbai', 'India', 'New Delhi', 10, '1985-10-01'),
('P003', 'New Harrold Publication', 'Adelaide', 'Australia', 'Sydney', 6, '1975-09-05'),
('P004', 'Ultra Press Inc.', 'London', 'UK', 'London', 8, '1948-07-10'),
('P005', 'Mountain Publication', 'Houstan', 'USA', 'Sun Diego', 25, '1975-01-01'),
('P006', 'Summer Night Publication', 'New York', 'USA', 'Atlanta', 10, '1990-12-10'),
('P007', 'Pietersen Grp. of Publishers', 'Cambridge', 'UK', 'London', 6, '1950-07-15'),
('P008', 'Novel Publisher Ltd.', 'New Delhi', 'India', 'Bangalore', 10, '2000-01-01');
```

17.

```
CREATE VIEW view_publisher
AS SELECT pub_name, pub_city, country
FROM publisher
WHERE (country='USA' AND pub_city='New York')
OR (country='India' AND pub_city='Mumbai');
```

18.

```
SELECT * FROM view_publisher;
```

19. CREATE VIEW with GROUP BY

```
CREATE TABLE book_mast (
    book_id VARCHAR(10) PRIMARY KEY,
    book_name VARCHAR(100),
    isbn_no VARCHAR(20),
    cate_id VARCHAR(10),
    aut_id VARCHAR(10),
    pub_id VARCHAR(10),
    dt_of_pub DATE,
    pub_lang VARCHAR(50),
    no_page INT,
    book_price DECIMAL(10, 2)
);
```

20.

```

INSERT INTO book_mast (book_id, book_name, isbn_no, cate_id, aut_id, pub_id,
dt_of_pub, pub_lang, no_page, book_price)
VALUES
    ('BK001', 'Introduction to Electrodynamics', '0000979001', 'CA001', 'AUT001', 'P003',
    '2001-05-08', 'English', 201, 85.00),
    ('BK002', 'Understanding of Steel Construction', '0000979002', 'CA002', 'AUT002',
    'P001', '2003-07-15', 'English', 300, 105.50),
    ('BK003', 'Guide to Networking', '0000979003', 'CA003', 'AUT003', 'P002', '2002-09-
    10', 'Hindi', 510, 200.00),
    ('BK004', 'Transfer of Heat and Mass', '0000979004', 'CA002', 'AUT004', 'P004',
    '2004-02-16', 'English', 600, 250.00),
    ('BK005', 'Conceptual Physics', '0000979005', 'CA001', 'AUT005', 'P006', '2003-07-
    16', NULL, 345, 145.00),
    ('BK006', 'Fundamentals of Heat', '0000979006', 'CA001', 'AUT006', 'P005', '2003-08-
    10', 'German', 247, 112.00),
    ('BK007', 'Advanced 3d Graphics', '0000979007', 'CA003', 'AUT007', 'P002', '2004-02-
    16', 'Hindi', 165, 56.00),
    ('BK008', 'Human Anatomy', '0000979008', 'CA005', 'AUT008', 'P006', '2001-05-17',
    'German', 88, 50.50),
    ('BK009', 'Mental Health Nursing', '0000979009', 'CA005', 'AUT009', 'P007', '2004-
    02-10', 'English', 350, 145.00),
    ('BK010', 'Fundamentals of Thermodynamics', '0000979010', 'CA002', 'AUT010',
    'P007', '2002-10-14', 'English', 400, 225.00),
    ('BK011', 'The Experimental Analysis of Cat', '0000979011', 'CA004', 'AUT011',
    'P005', '2007-06-09', 'French', 225, 95.00),
    ('BK012', 'The Nature of World', '0000979012', 'CA004', 'AUT005', 'P008', '2005-12-
    20', 'English', 350, 88.00),
    ('BK013', 'Environment a Sustainable Future', '0000979013', 'CA004', 'AUT012',
    'P001', '2003-10-27', 'German', 165, 100.00),
    ('BK014', 'Concepts in Health', '0000979014', 'CA005', 'AUT013', 'P004', '2001-08-25',
    NULL, 320, 180.00),
    ('BK015', 'Anatomy & Physiology', '0000979015', 'CA005', 'AUT014', 'P008', '2000-10-
    10', 'Hindi', 225, 135.00),
    ('BK016', 'Networks and Telecommunications', '00009790_16', 'CA003', 'AUT015',
    'P003', '2002-01-01', 'French', 95, 45.00);

```

```
CREATE VIEW view_bookmast
AS SELECT pub_lang, count(*)
FROM book_mast
GROUP BY pub_lang;
```

22.

```
SELECT * FROM view_bookmast;
```

23. CREATE VIEW with ORDER BY

```
CREATE VIEW view_bookmast1
AS SELECT pub_lang, count(*)
FROM book_mast1
GROUP BY pub_lang ORDER BY pub_lang;
```

24.

```
SELECT * FROM view_bookmast1;
```

25. CREATE VIEW with BETWEEN and IN

```
CREATE VIEW view_bookmast2
AS SELECT *
FROM book_mast1
WHERE book_name BETWEEN 'A' AND 'G'
AND no_page IN(165,250,350,400,510);
```

26.

```
SELECT * FROM view_bookmast2;
```

27. CREATE VIEW with LIKE

```
CREATE VIEW view_author1
AS SELECT *
FROM author1
```

```
WHERE aut_name  
NOT LIKE 'T%' AND aut_name NOT LIKE 'W%';
```

28.

```
SELECT * FROM view_author1;
```

29. CREATE VIEW using subqueries

```
CREATE TABLE purchase (  
    invoice_no VARCHAR(10) PRIMARY KEY,  
    invoice_dt DATE,  
    ord_no VARCHAR(20),  
    ord_date DATE,  
    receive_dt DATE,  
    book_id VARCHAR(10),  
    book_name VARCHAR(100),  
    pub_lang VARCHAR(50),  
    cate_id VARCHAR(10),  
    receive_qty INT,  
    purch_price DECIMAL(10, 2),  
    total_cost DECIMAL(10, 2)  
);
```

30.

```
INSERT INTO purchase (invoice_no, invoice_dt, ord_no, ord_date, receive_dt,  
book_id, book_name, pub_lang, cate_id, receive_qty, purch_price, total_cost)  
VALUES  
    ('INV0001', '2008-07-15', 'ORD/08-09/0001', '2008-07-06', '2008-07-19', 'BK001',  
    'Introduction to Electrodynamics', 'English', 'CA001', 15, 75.00, 1125.00),  
    ('INV0002', '2008-08-25', 'ORD/08-09/0002', '2008-08-09', '2008-08-28', 'BK004',  
    'Transfer of Heat and Mass', 'English', 'CA002', 8, 55.00, 440.00),  
    ('INV0003', '2008-09-20', 'ORD/08-09/0003', '2008-09-15', '2008-09-23', 'BK005',  
    'Conceptual Physics', NULL, 'CA001', 20, 20.00, 400.00),  
    ('INV0004', '2007-08-30', 'ORD/07-08/0005', '2007-08-22', '2007-08-30', 'BK004',  
    'Transfer of Heat and Mass', 'English', 'CA002', 15, 35.00, 525.00),
```

```
('INV0005', '2007-07-28', 'ORD/07-08/0004', '2007-06-25', '2007-07-30', 'BK001',
'Introduction to Electrodynamics', 'English', 'CA001', 8, 25.00, 200.00),
('INV0006', '2007-09-24', 'ORD/07-08/0007', '2007-09-20', '2007-09-30', 'BK003',
'Guide to Networking', 'Hindi', 'CA003', 20, 45.00, 900.00);
```

31.

```
CREATE VIEW view_purchase
AS SELECT invoice_no,book_name,cate_id
FROM purchase
WHERE cate_id= (SELECT cate_id FROM book_mast WHERE no_page=201);
```

32.

```
SELECT * FROM view_purchase;
```

33. CREATE VIEW with JOIN

```
CREATE TABLE category (
    cate_id VARCHAR(10) PRIMARY KEY,
    cate_descrip VARCHAR(50)
);
```

34.

```
INSERT INTO category (cate_id, cate_descrip)
VALUES
    ('CA001', 'Science'),
    ('CA002', 'Technology'),
    ('CA003', 'Computers'),
    ('CA004', 'Nature'),
    ('CA005', 'Medical');
```

35.

```
CREATE VIEW view_purchase1
```

```
AS SELECT a.cate_id,a.cate_descrip, b.invoice_no,  
b.invoice_dt,b.book_name  
FROM category a,purchase b  
WHERE a.cate_id=b.cate_id;
```

36.

```
SELECT * FROM view_purchase1;
```

37. CREATE VIEW with UNION

```
CREATE VIEW view_bookmast3 AS  
SELECT *  
FROM book_mast  
WHERE pub_id='P001' UNION  
SELECT *  
FROM book_mast  
WHERE book_name BETWEEN 'A' AND 'G' UNION  
SELECT *  
FROM book_mast  
WHERE no_page IN(165,250,350,400,510);
```

38.

```
SELECT * FROM view_bookmast3;
```

Lab 8: Indexing

1. In a database, indexing is a technique used to optimize the retrieval of records from a table.
2. Indexes are data structures that store a small portion of the table's data, typically the values of one or more columns, along with pointers to the corresponding rows in the table.
3. These indexes are organized in a way that allows for efficient data searching, sorting, and retrieval.

Why do we use indexing in DBMS?

1. **Faster Data Retrieval:** Indexes help speed up data retrieval operations, such as SELECT queries, by providing a quick path to locate the desired rows in a table.
2. **Types of Indexes:** There are different types of indexes, including:
 - 2.1. **Primary Index:** This type of index is defined on an ordered data file, typically on a key field such as the primary key of a relation.
 - 2.2. **Secondary Index:** A secondary index may be generated from a candidate key field with a unique value in every record or a non-key field with duplicate values.
 - 2.3. **Clustering Index:** A clustering index is defined on an ordered data file, usually on a non-key field.

3. Structure-Based Indexing

- 3.1. **Dense Index:** In a dense index, there is an index record for every search key value in the database. It provides fast searching but requires more space to store index records.
 - 3.2. **Sparse Index:** Sparse indexes do not have an index record for every search key. Instead, they contain search key values and pointers to the actual data. If the desired data is not directly found via the index, the system conducts a sequential search until it finds the data.
 - 3.3. **Multilevel Index:** The index is stored alongside the actual database files on the disk. As the size of the database grows, the indices also increase. Multilevel indexing helps by breaking down the index into smaller indices, allowing the outermost level to be small enough to fit into memory.
 - 3.4. **B+ Tree:** A B+ tree is a balanced search tree commonly used for indexing. It follows a multilevel index format where leaf nodes denote actual data pointers. B+ trees ensure balanced height and support both random and sequential access.
4. **Maintenance Overhead:** While indexes can improve query performance, they also introduce overhead during data modification operations (INSERT, UPDATE, DELETE), as the indexes need to be updated to reflect changes in the underlying table.

5. Index Selection: Deciding which columns to index requires consideration of the types of queries frequently executed against the table. Indexes should be created on columns frequently used in WHERE clauses, JOIN conditions, or ORDER BY clauses.

6. Indexing Strategies: DBMSs employ various indexing strategies to organize and access data efficiently. Common strategies include B-tree, hash, and bitmap indexes, each with advantages and limitations.

7. Monitoring and Optimization: Index usage and performance should be monitored regularly. Unused indexes may consume resources unnecessarily, and poorly designed indexes can degrade performance. Periodic index optimization, such as index rebuilding or reorganization, may be necessary to maintain optimal performance.

Lab Performance Question

1. Creating a Database and Table

```
CREATE DATABASE IF NOT EXISTS ADBMS_LAB;
```

```
use ADBMS_LAB;
```

```
drop table employees;
```

```
CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    department VARCHAR(100),
    salary DECIMAL(10, 2)
);
```

```
desc employees;
```

- This script creates a database named "ADBMS_LAB" if it doesn't exist.
- It then selects the newly created database.

- Finally, it creates a table named "employees" with columns for id, name, department, and salary.

2. Inserting Data into the Table

```
INSERT INTO employees (name, department, salary) VALUES
    ('John Doe', 'Engineering', 50000.00),
    ('Jane Smith', 'HR', 45000.00),
    ('Michael Johnson', 'Engineering', 55000.00),
    ('Emily Brown', 'Sales', 48000.00);
```

```
SELECT * FROM employees;
```

- Inserts data into the "employees" table, specifying values for name, department, and salary.
- Retrieves all rows from the "employees" table.

3. Creating Indexes

```
CREATE INDEX idx_department ON employees (department);
```

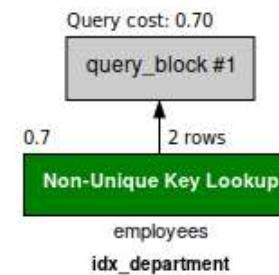
```
CREATE INDEX idx_salary ON employees (salary);
```

```
SHOW INDEX FROM employees;
```

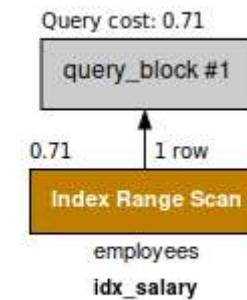
- Creates indexes on the "department" and "salary" columns of the "employees" table, which can improve query performance.

4. Querying with Conditions

```
SELECT * FROM employees WHERE department = 'Engineering';
```



```
SELECT * FROM employees WHERE salary > 50000.00;
```

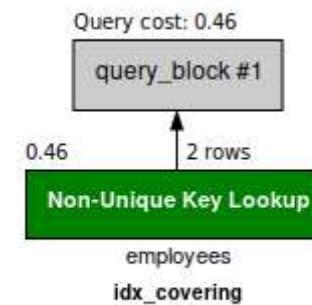


5. Creating Index

```
CREATE INDEX idx_covering ON employees (department, salary);
```

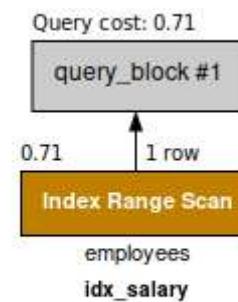
A covering index is an index that contains all the columns required by a query, allowing MySQL to retrieve the data directly from the index without accessing the actual table. This can improve query performance by avoiding the need to fetch data from the table.

```
SELECT department, salary FROM employees WHERE department = 'Engineering';
```

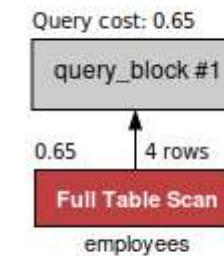


```
SELECT * FROM employees USE INDEX (idx_salary) WHERE salary > 50000.00;
```

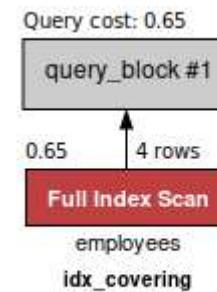
MySQL allows you to provide hints to the query optimizer to specify which index to use for a particular query. This can be useful when you want to ensure that a specific index is used, especially if the optimizer's choice is not optimal.



```
SELECT * FROM employees USE INDEX (idx_covering) WHERE salary > 50000.00;
```

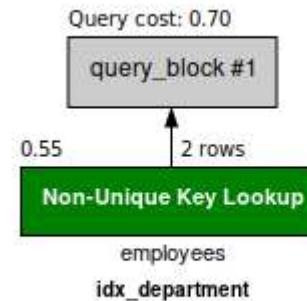


```
SELECT department, salary FROM employees USE INDEX (idx_covering) WHERE salary > 50000.00;
```



```
-- Query using index merge  
SELECT * FROM employees WHERE department = 'Engineering' AND salary > 50000.00;
```

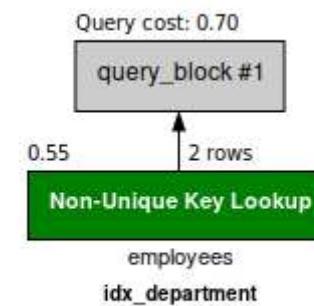
MySQL optimizer can sometimes use multiple indexes and merge their results to fulfill a query. This can be advantageous when individual indexes are selective but together cover the query conditions effectively.



```
-- Query utilizing index condition pushdown
```

```
SELECT * FROM employees WHERE salary > 50000.00 AND department = 'Engineering';
```

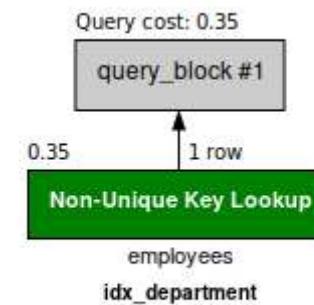
Index condition pushdown is a feature in MySQL that pushes filter conditions into the index scan, reducing the number of rows fetched from the table.



```
-- Query with forced index
```

```
SELECT * FROM employees FORCE INDEX (idx_department) WHERE department = 'Engineering';
```

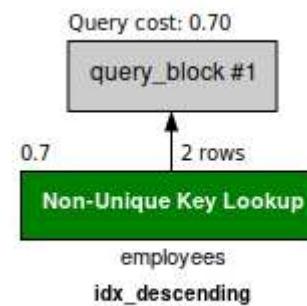
You can force MySQL to use a specific index for a query using the FORCE INDEX directive. This can be useful when you know that a particular index is optimal for a query.



```
CREATE INDEX idx_descending ON employees (department DESC);
```

To create an index in descending order, you need to explicitly specify the order using the DESC keyword.

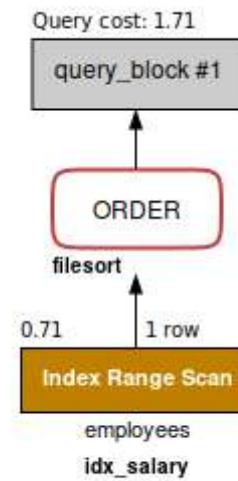
```
SELECT department, salary FROM employees USE INDEX (idx_descending) WHERE department = 'Engineering';
```



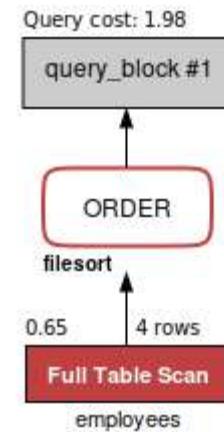
```
-- Query using an index for descending order scan
```

```
SELECT * FROM employees WHERE salary > 50000.00 ORDER BY department DESC;
```

When querying data using an index, MySQL automatically utilizes the index for both ascending and descending order scans. However, it's essential to ensure that your query conditions match the index order for optimal performance.



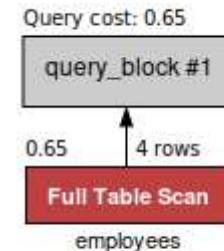
```
SELECT * FROM employees USE INDEX (idx_descending) WHERE salary > 50000.00 ORDER BY department DESC;
```



```
CREATE INDEX idx_mixed_order ON employees (department ASC, salary DESC);
```

You can also create an index with multiple columns, some in ascending order and others in descending order.

```
SELECT * FROM employees USE INDEX (idx_mixed_order) WHERE salary > 50000.00;
```



```
SHOW INDEX FROM employees;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
employees	0	PRIMARY	1	id	A	4	NULL	NULL		BTREE			YES	NULL
employees	1	idx_department	1	department	A	3	NULL	NULL	YES	BTREE			YES	NULL

employees	1	idx_salary	1	salary	A	4	NULL	NULL	YES	BTREE			YES	NULL
employees	1	idx_covering	1	department	A	3	NULL	NULL	YES	BTREE			YES	NULL
employees	1	idx_covering	2	salary	A	4	NULL	NULL	YES	BTREE			YES	NULL
employees	1	idx_descending	1	department	D	3	NULL	NULL	YES	BTREE			YES	NULL
employees	1	idx_mixed_order	1	department	A	3	NULL	NULL	YES	BTREE			YES	NULL
employees	1	idx_mixed_order	2	salary	D	4	NULL	NULL	YES	BTREE			YES	NULL

6. Drop Index

```
DROP INDEX idx_department ON employees;
DROP INDEX idx_salary ON employees;
DROP INDEX idx_covering ON employees;
DROP INDEX idx_covering ON employees;
DROP INDEX idx_descending ON employees;
DROP INDEX idx_mixed_order ON employees;
SHOW INDEX FROM employees;
```

Run every query and make the report of every Image of Indexing

7. Create the table

```
CREATE TABLE books (
    book_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255),
    author VARCHAR(255),
    publication_year INT
);
```

8. Insert The Data

```
INSERT INTO books (title, author, publication_year) VALUES
('The Adventures of Tom Sawyer', 'Mark Twain', 1876),
('The Picture of Dorian Gray', 'Oscar Wilde', 1890),
('Alice's Adventures in Wonderland', 'Lewis Carroll', 1865),
('The War of the Worlds', 'H.G. Wells', 1898),
('Dracula', 'Bram Stoker', 1897),
('Jane Eyre', 'Charlotte Brontë', 1847),
```

('Wuthering Heights', 'Emily Brontë', 1847),
('Frankenstein', 'Mary Shelley', 1818),
('The Count of Monte Cristo', 'Alexandre Dumas', 1844),
('The Adventures of Sherlock Holmes', 'Arthur Conan Doyle', 1892),
('The Strange Case of Dr. Jekyll and Mr. Hyde', 'Robert Louis Stevenson', 1886),
('The Hound of the Baskervilles', 'Arthur Conan Doyle', 1902),
('Les Misérables', 'Victor Hugo', 1862),
('Anna Karenina', 'Leo Tolstoy', 1877),
('War and Peace', 'Leo Tolstoy', 1869),
('Crime and Punishment', 'Fyodor Dostoevsky', 1866),
('Gone with the Wind', 'Margaret Mitchell', 1936),
('The Grapes of Wrath', 'John Steinbeck', 1939),
('The Old Man and the Sea', 'Ernest Hemingway', 1952),
('The Sun Also Rises', 'Ernest Hemingway', 1926),
('Of Mice and Men', 'John Steinbeck', 1937),
('One Hundred Years of Solitude', 'Gabriel García Márquez', 1967),
('Don Quixote', 'Miguel de Cervantes', 1605),
('The Scarlet Letter', 'Nathaniel Hawthorne', 1850),
('The Canterbury Tales', 'Geoffrey Chaucer', 1400),
('Heart of Darkness', 'Joseph Conrad', 1899),
('The Odyssey', 'Homer', 800),
('The Iliad', 'Homer', 800),
('Catch-22', 'Joseph Heller', 1961),
('To the Lighthouse', 'Virginia Woolf', 1927),
('Slaughterhouse-Five', 'Kurt Vonnegut', 1969),
('The Bell Jar', 'Sylvia Plath', 1963),
('The Great Expectations', 'Charles Dickens', 1861),
('The Adventures of Huckleberry Finn', 'Mark Twain', 1884),
('The Chronicles of Narnia', 'C.S. Lewis', 1950),
('Lord of the Flies', 'William Golding', 1954),
('The Outsiders', 'S.E. Hinton', 1967),
('The Road', 'Cormac McCarthy', 2006),
('Beloved', 'Toni Morrison', 1987),
('Pride and Prejudice', 'Jane Austen', 1813),
('The Hobbit', 'J.R.R. Tolkien', 1937),
('The Lord of the Rings', 'J.R.R. Tolkien', 1954),
('Harry Potter and the Philosopher's Stone', 'J.K. Rowling', 1997),
('Moby-Dick', 'Herman Melville', 1851),
('Brave New World', 'Aldous Huxley', 1932),

('The Da Vinci Code', 'Dan Brown', 2003),
('Angels & Demons', 'Dan Brown', 2000),
('The Girl with the Dragon Tattoo', 'Stieg Larsson', 2005),
('The Hunger Games', 'Suzanne Collins', 2008),
('The Catcher in the Rye', 'J.D. Salinger', 1951),
('The Martian', 'Andy Weir', 2011),
('A Game of Thrones', 'George R.R. Martin', 1996),
('The Help', 'Kathryn Stockett', 2009),
('The Girl on the Train', 'Paula Hawkins', 2015),
('Gone Girl', 'Gillian Flynn', 2012),
('The Lovely Bones', 'Alice Sebold', 2002),
('Life of Pi', 'Yann Martel', 2001),
('The Alchemist', 'Paulo Coelho', 1988),
('Twilight', 'Stephenie Meyer', 2005),
('The Secret Life of Bees', 'Sue Monk Kidd', 2001),
('The Book Thief', 'Markus Zusak', 2005),
('Water for Elephants', 'Sara Gruen', 2006),
('The Girl with the Pearl Earring', 'Tracy Chevalier', 1999),
('The Help', 'Kathryn Stockett', 2009),
('The Kite Runner', 'Khaled Hosseini', 2003),
('The Glass Castle', 'Jeannette Walls', 2005),
('Eat, Pray, Love', 'Elizabeth Gilbert', 2006),
('The Time Traveler's Wife', 'Audrey Niffenegger', 2003),
('The Goldfinch', 'Donna Tartt', 2013),
('Big Little Lies', 'Liane Moriarty', 2014),
('The Nightingale', 'Kristin Hannah', 2015),
('The Underground Railroad', 'Colson Whitehead', 2016),
('The Silent Patient', 'Alex Michaelides', 2019),
('Where the Crawdads Sing', 'Delia Owens', 2018),
('Becoming', 'Michelle Obama', 2018),
('Educated', 'Tara Westover', 2018),
('Circe', 'Madeline Miller', 2018),
('Normal People', 'Sally Rooney', 2018),
('Little Fires Everywhere', 'Celeste Ng', 2017),
('The Testaments', 'Margaret Atwood', 2019),
('The Tattooist of Auschwitz', 'Heather Morris', 2018),
('The Wife Between Us', 'Greer Hendricks', 2018),
('The Silent Patient', 'Alex Michaelides', 2019),
('Where the Crawdads Sing', 'Delia Owens', 2018),

('Becoming', 'Michelle Obama', 2018),
('Educated', 'Tara Westover', 2018),
('Circe', 'Madeline Miller', 2018),
('Normal People', 'Sally Rooney', 2018),
('Little Fires Everywhere', 'Celeste Ng', 2017),
('The Testaments', 'Margaret Atwood', 2019),
('The Tattooist of Auschwitz', 'Heather Morris', 2018),
('The Wife Between Us', 'Greer Hendricks', 2018);

LAB 10 #PL/SQL Commands

Write a PL/SQL code to accept the value of A, B & C display which is greater.

```
DECLARE
    a NUMBER := 10; -- Input value for A
    b NUMBER := 20; -- Input value for B
    c NUMBER := 15; -- Input value for C
BEGIN
    IF a > b AND a > c THEN
        DBMS_OUTPUT.PUT_LINE('A is greater');
    ELSIF b > a AND b > c THEN
        DBMS_OUTPUT.PUT_LINE('B is greater');
    ELSE
        DBMS_OUTPUT.PUT_LINE('C is greater');
    END IF;
END;
/
```

Using PL/SQL Statements create a simple loop that display message “Welcome to PL/SQL Programming” 20 times.

```
BEGIN
    FOR i IN 1..20 LOOP
        DBMS_OUTPUT.PUT_LINE('Welcome to PL/SQL Programming');
    END LOOP;
END;
/
```

Write a PL/SQL code block to find the factorial of a number.

```
DECLARE
    num NUMBER := 5; -- Input number
    factorial NUMBER := 1;
BEGIN
    FOR i IN 1..num LOOP
        factorial := factorial * i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Factorial of ' || num || ' is ' || factorial);
END;
/
```

Write a PL/SQL program to generate Fibonacci series.

```
-- Create a PL/SQL block to generate Fibonacci series
DECLARE
    -- Variables to hold Fibonacci sequence
    first_number NUMBER := 0;
    second_number NUMBER := 1;
    next_number NUMBER;
    limit_number NUMBER := 100; -- Change this limit as per requirement

    -- Cursor declaration to print Fibonacci series
    CURSOR fibonacci_cursor IS
        SELECT LEVEL AS n, first_number AS Fibonacci_Number
        FROM DUAL
```

```

CONNECT BY LEVEL <= limit_number;
BEGIN
-- Print the initial values
DBMS_OUTPUT.PUT_LINE('Fibonacci Series up to ' || limit_number || ' numbers:');
DBMS_OUTPUT.PUT_LINE('0'); -- First Fibonacci number
DBMS_OUTPUT.PUT_LINE('1'); -- Second Fibonacci number

-- Loop to generate and print Fibonacci series
FOR i IN 3..limit_number LOOP
    next_number := first_number + second_number;
    first_number := second_number;
    second_number := next_number;
    DBMS_OUTPUT.PUT_LINE(next_number);
END LOOP;
END;
/

```

Write a PL/SQL code to find the sum of first N numbers.

```

-- Create a PL/SQL block to find the sum of first N numbers
DECLARE
-- Variable to hold the sum
total_sum NUMBER := 0;

-- Variable for the number up to which sum is to be calculated
N NUMBER := 100; -- Change N to the desired number

BEGIN
-- Loop to calculate the sum of first N numbers
FOR i IN 1..N LOOP
    total_sum := total_sum + i;
END LOOP;

-- Print the sum
DBMS_OUTPUT.PUT_LINE('The sum of first ' || N || ' numbers is: ' || total_sum);
END;
/

```

WAP that accept the value of A, B & C display which is greater using Functions and Procedures.

```

DECLARE
-- Variables to hold the values of A, B, and C
A NUMBER := 10; -- Replace with your desired value for A
B NUMBER := 20; -- Replace with your desired value for B
C NUMBER := 15; -- Replace with your desired value for C

-- Function to find the maximum of two numbers
FUNCTION find_maximum(x NUMBER, y NUMBER) RETURN NUMBER IS
BEGIN
    IF x > y THEN
        RETURN x;
    ELSE
        RETURN y;
    END IF;
END find_maximum;

```

```

-- Procedure to compare three numbers and display the greatest
PROCEDURE compare_and_display_greatest(A IN NUMBER, B IN NUMBER, C IN NUMBER)
IS
    greatest NUMBER;
BEGIN
    greatest := find_maximum(find_maximum(A, B), C); -- Call the function to find the maximum of
three numbers

    -- Display the result
    IF greatest = A THEN
        DBMS_OUTPUT.PUT_LINE('A is the greatest.');
    ELSIF greatest = B THEN
        DBMS_OUTPUT.PUT_LINE('B is the greatest.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('C is the greatest.');
    END IF;
END compare_and_display_greatest;
BEGIN
    -- Call the procedure to compare and display the greatest
    compare_and_display_greatest(A, B, C);
END;
/

```

Print the statements “Welcome to PL/SQL Programming” 20 times using Functions and Procedures.

```

DECLARE
    -- Variable to store the message
    message VARCHAR2(100) := 'Welcome to PL/SQL Programming';

    -- Procedure to display message 20 times
PROCEDURE display_message IS
BEGIN
    -- Loop to display the message 20 times
    FOR i IN 1..20 LOOP
        DBMS_OUTPUT.PUT_LINE(message);
    END LOOP;
END display_message;
BEGIN
    -- Call the procedure to display the message
    display_message;
END;
/

```

Factorial of a number using function and procedure.

```

DECLARE
    -- Variable to hold the value for factorial calculation
    n NUMBER := 5; -- Replace with your desired value for factorial calculation

    -- Function to calculate factorial of a number
FUNCTION factorial(num NUMBER) RETURN NUMBER IS
    result NUMBER := 1;
BEGIN
    IF num <= 1 THEN
        RETURN result;
    END IF;
    result := result * num;
    RETURN result;
END factorial;
/

```

```

ELSE
    FOR i IN 2..num LOOP
        result := result * i;
    END LOOP;
    RETURN result;
END IF;
END factorial;

-- Procedure to calculate and display factorial
PROCEDURE calculate_and_display_factorial(n IN NUMBER) IS
    fact NUMBER;
BEGIN
    fact := factorial(n); -- Call the function to calculate factorial

    -- Display the result
    DBMS_OUTPUT.PUT_LINE('Factorial of ' || n || ' is: ' || fact);
END calculate_and_display_factorial;
BEGIN
    -- Call the procedure to calculate and display factorial
    calculate_and_display_factorial(n);
END;
/

```

Fibonacci series of a number using function and procedure.

```

DECLARE
    -- Variable to hold the number of terms in the Fibonacci series
    num_terms NUMBER := 16; -- Replace with your desired number of terms

    -- Function to calculate Fibonacci series
    FUNCTION fibonacci_series(num_terms_in NUMBER) RETURN VARCHAR2 IS
        fib_series VARCHAR2(1000) := '0, 1'; -- Initial values of Fibonacci series
        a NUMBER := 0;
        b NUMBER := 1;
        next_term NUMBER;
BEGIN
    IF num_terms_in <= 0 THEN
        RETURN 'Invalid input';
    END IF;

    IF num_terms_in = 1 THEN
        RETURN '0';
    END IF;

    IF num_terms_in = 2 THEN
        RETURN '0, 1';
    END IF;

    FOR i IN 3..num_terms_in LOOP
        next_term := a + b;
        fib_series := fib_series || ',' || next_term;
        a := b;
        b := next_term;
    END LOOP;

    RETURN fib_series;
END fibonacci_series;

```

```

-- Procedure to display Fibonacci series
PROCEDURE display_fibonacci_series(num_terms_in NUMBER) IS
    fib_series VARCHAR2(1000);
BEGIN
    fib_series := fibonacci_series(num_terms_in); -- Call the function to calculate Fibonacci series

    -- Display the result
    DBMS_OUTPUT.PUT_LINE('Fibonacci series up to ' || num_terms_in || ' terms: ' || fib_series);
END display_fibonacci_series;
BEGIN
    -- Call the procedure to display Fibonacci series
    display_fibonacci_series(num_terms);
END;
/

```

Find the sum of first N numbers using Function and Procedure.

```

DECLARE
    -- Variable to hold the number of terms
    num_terms NUMBER := 16; -- Replace with your desired number of terms

    -- Function to calculate the sum of first N numbers
    FUNCTION sum_of_first_n_numbers(n NUMBER) RETURN NUMBER IS
        total_sum NUMBER := 0;
    BEGIN
        IF n <= 0 THEN
            RETURN 0; -- Return 0 for non-positive input
        END IF;

        FOR i IN 1..n LOOP
            total_sum := total_sum + i; -- Add each number from 1 to n to total_sum
        END LOOP;

        RETURN total_sum;
    END sum_of_first_n_numbers;

    -- Procedure to display the sum of first N numbers
    PROCEDURE display_sum_of_first_n_numbers(n NUMBER) IS
        sum_result NUMBER;
    BEGIN
        sum_result := sum_of_first_n_numbers(n); -- Call the function to calculate sum
        -- Display the result
        DBMS_OUTPUT.PUT_LINE('Sum of first ' || n || ' numbers: ' || sum_result);
    END display_sum_of_first_n_numbers;
BEGIN
    -- Call the procedure to display the sum
    display_sum_of_first_n_numbers(num_terms);
END;
/

```

Lab Performance Questions

1. Write a PL/SQL code to calculate the area of a circle given its radius.
2. Create a PL/SQL program to check whether a given number is prime or not.
3. Develop a PL/SQL script to reverse a string entered by the user.
4. Design a PL/SQL code to find the largest and smallest number among three given numbers.

5. Implement a PL/SQL program to calculate the compound interest for a given principal amount, rate, and time period.



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Program Name: B. Tech CS

Semester: III

Name: Yashasvani Markanday

SAP ID: 590011386

Batch: B2

Subject: Database and Management System

3 Create all the given table.

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. In the SQL Commands pane, the command `desc Bus;` is run, and the results show the structure of the BUS table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BUS	BUS_NO	VARCHAR2	10	-	-	-	✓	-	-
	SOURCE	VARCHAR2	20	-	-	-	✓	-	-
	DESTINATION	VARCHAR2	20	-	-	-	✓	-	-
	COUCH_TYPE	VARCHAR2	20	-	-	-	✓	-	-

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. In the SQL Commands pane, the command `desc Reservation;` is run, and the results show the structure of the RESERVATION table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVATION	PREL_NO	NUMBER	-	9	0	-	✓	-	-
	JOURNEY_DATE	DATE	7	-	-	-	✓	-	-
	NO_OF_SEATS	NUMBER	-	8	0	-	✓	-	-
	ADDRESS	VARCHAR2	20	-	-	-	✓	-	-
	CONTACT_NO	NUMBER	-	9	0	-	✓	-	-
	BUS_NO	VARCHAR2	10	-	-	-	✓	-	-
	SEAT_NO	NUMBER	22	-	-	-	✓	-	-

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Rows 10 Clear Command Find Tables Schema WKSP_YYDBMS205 Save Run

```
1 desc Ticket;
```

Results Explain Describe Saved SQL History

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TICKET	TICKET_NO	NUMBER	-	9	0	-	✓	-	-
	JOURNEY_DATE	DATE	7	-	-	-	✓	-	-
	AGE	NUMBER	-	4	0	-	✓	-	-
	SEX	CHAR	10	-	-	-	✓	-	-
	SOURCEST	VARCHAR2	10	-	-	-	✓	-	-
	DESTINATION	VARCHAR2	10	-	-	-	✓	-	-
	DEP_TIME	VARCHAR2	10	-	-	-	✓	-	-
	BUS_NO	NUMBER	-	10	0	-	✓	-	-

Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Rows 10 Clear Command Find Tables Schema WKSP_YYDBMS205 Save Run

```
1 desc Passenger;
```

Results Explain Describe Saved SQL History

Object Type	Table	Object	PASSENGER						
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PASSENGER	PNR_NO	NUMBER	-	9	0	-	✓	-	-
	TICKET_NO	NUMBER	-	9	0	-	✓	-	-
	FOREIGN_KEYNAME	VARCHAR2	15	-	-	-	✓	-	-
	AGE	NUMBER	-	4	0	-	✓	-	-
	SEX	CHAR	10	-	-	-	✓	-	-
	CONTACT_NO	NUMBER	-	9	0	-	✓	-	-

Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Rows 10 Clear Command Find Tables Schema WKSP_YYDBMS205 Save Run

```
1 desc Cancellation;
```

Results Explain Describe Saved SQL History

Object Type	Table	Object	CANCELLATION						
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CANCELLATION	PNR_NO	NUMBER	-	9	0	-	✓	-	-
	JOURNEY_DATE	NUMBER	-	9	0	-	✓	-	-
	FOREIGN_KEYSEAT_NO	VARCHAR2	15	-	-	-	✓	-	-
	CONTACT_NO	NUMBER	-	9	0	-	✓	-	-

Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

4 Add a new column to the existing relation.

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user's profile 'Yashaswini Mark...' is visible on the right. The main workspace has a search bar and a schema dropdown set to 'WKSP_YYDBMS2025'. Below the search bar are buttons for 'Save' and 'Run'. The SQL command input field contains the command 'desc Bus;'. The results pane shows the description of the 'BUS' table with the following columns:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BUS	BUS_NO	VARCHAR2	10	-	-	-	✓	-	-
	SOURCE1	VARCHAR2	20	-	-	-	✓	-	-
	DESTINATION	VARCHAR2	20	-	-	-	✓	-	-
	COUCH_TYPE	VARCHAR2	20	-	-	-	✓	-	-
	BUS_FARE	NUMBER	-	10	2	-	✓	-	-

5 Change the datatype of the table from char to varchar2.

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user's profile 'Yashaswini Mark...' is visible on the right. The main workspace has a search bar and a schema dropdown set to 'WKSP_YYDBMS2025'. Below the search bar are buttons for 'Save' and 'Run'. The SQL command input field contains the command 'desc TICKET;'. The results pane shows the description of the 'TICKET' table with the following columns:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TICKET	TICKET_NO	NUMBER	-	9	0	-	✓	-	-
	JOURNEY_DATE	DATE	7	-	-	-	✓	-	-
	AGE	NUMBER	-	4	0	-	✓	-	-
	SEX	VARCHAR2	10	-	-	-	✓	-	-
	SOURCE1	VARCHAR2	10	-	-	-	✓	-	-
	DESTINATION	VARCHAR2	10	-	-	-	✓	-	-
	DEP_TIME	VARCHAR2	10	-	-	-	✓	-	-
	BUS_NO	NUMBER	-	10	0	-	✓	-	-

6 Change the name of column/field.

The screenshot shows the Oracle APEX interface with the SQL Workshop module selected. The schema is set to WKS2_YYDBMS2025. In the command editor, the command `desc Passenger;` is entered. The results pane displays the structure of the PASSENGER table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PASSENGER	PNR_NO	NUMBER	-	9	0	-	✓	-	-
PASSENGER	PASSENGER_NAME	NUMBER	-	9	0	-	✓	-	-
PASSENGER	FOREIGN_KEYNAME	VARCHAR2	15	-	-	-	✓	-	-
PASSENGER	AGE	NUMBER	-	4	0	-	✓	-	-
PASSENGER	SEX	CHAR	10	-	-	-	✓	-	-
PASSENGER	CONTACT_NO	NUMBER	-	9	0	-	✓	-	-

7 Modify the column width of all the table.

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user's profile (Yashavani Mark... yy_dilms_2025) and the schema (WKSP_YYDBMS2025). The main area displays the results of a DESCRIBE command for the 'BUS' table. The table has five columns: BUS_NO, SOURCE1, DESTINATION, COUCH_TYPE, and BUS_FARE. The 'BUS_FARE' column is highlighted with a green border. The table structure is as follows:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BUS	BUS_NO	VARCHAR2	10	-	-	-	✓	-	-
	SOURCE1	VARCHAR2	50	-	-	-	✓	-	-
	DESTINATION	VARCHAR2	20	-	-	-	✓	-	-
	COUCH_TYPE	VARCHAR2	20	-	-	-	✓	-	-
	BUS_FARE	NUMBER	-	10	2	-	✓	-	-

This screenshot shows the same Oracle SQL Workshop interface and workspace setup as the previous one. The results of the DESCRIBE command for the 'BUS' table are displayed again. The 'BUS_FARE' column is now explicitly defined with a length of 10 and a scale of 2, indicating the modification made in the previous step.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BUS	BUS_NO	VARCHAR2	10	-	-	-	✓	-	-
	SOURCE1	VARCHAR2	50	-	-	-	✓	-	-
	DESTINATION	VARCHAR2	20	-	-	-	✓	-	-
	COUCH_TYPE	VARCHAR2	20	-	-	-	✓	-	-
	BUS_FARE	NUMBER	-	10	2	-	✓	-	-

Conclusion

Through this lab, I gained practical knowledge of creating, modifying, and managing database tables using SQL commands also introduced the fundamentals of RDBMS concepts, E.F. Codd's rules, and the use of SQL for defining and manipulating data. The exercise reinforced understanding of relationships, constraints, and data types in real-world database and alter column sizes as per requirements. Differences between TRUNCATE, DELETE, and DROP were clarified for better database maintenance strategies. These exercises improved my understanding of schema design and data definition operations.



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Program Name: B. Tech CS

Semester: III

Name: Yashasvani Markanday

SAP ID: 590011386

Batch: B2

Subject: Database and Management System

1. Write an SQL script that includes DDL statements to create the following tables: - `Products` with columns: `ProductID` (integer), `ProductName` (varchar), `UnitPrice` (decimal), `Manufacturer` (varchar). - `Customers` with columns: `CustomerID` (integer), `CustomerName` (varchar), `Email` (varchar). Ensure appropriate constraints for data integrity.

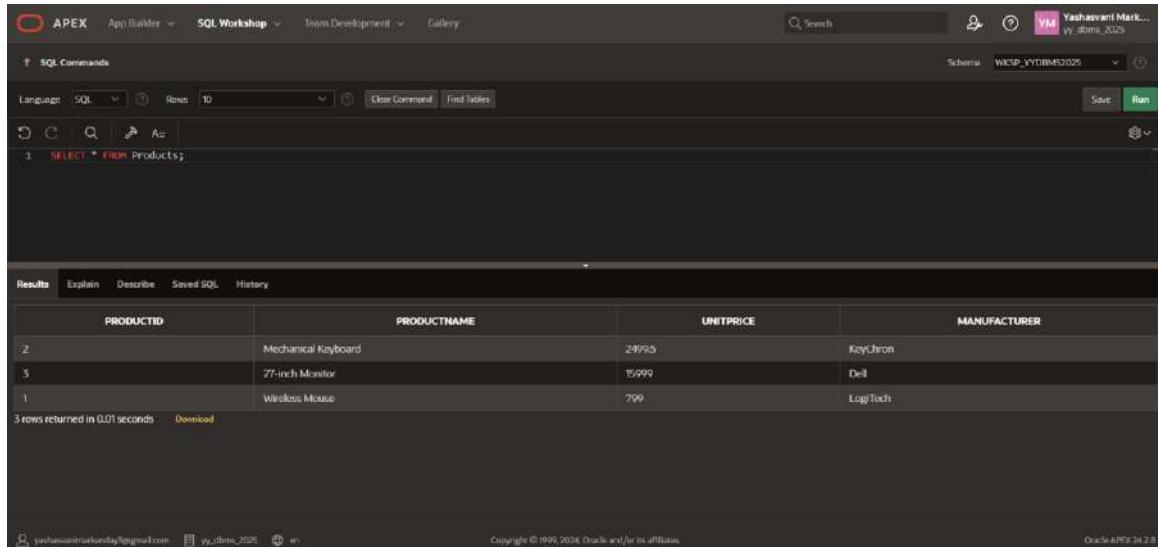
The screenshot shows the Oracle APEX SQL Workshop interface. The command entered is `desc Products;`. The results pane displays the description of the `PRODUCTS` table, which has four columns: `ProductID`, `ProductName`, `UnitPrice`, and `Manufacturer`.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PRODUCTS	ProductID	NUMBER	22	-	0	-	✓	-	-
	ProductName	VARCHAR2	100	-	-	-	✓	-	-
	UnitPrice	NUMBER	-	10	2	-	✓	-	-
	Manufacturer	VARCHAR2	100	-	-	-	✓	-	-

The screenshot shows the Oracle APEX SQL Workshop interface. The command entered is `desc Customers;`. The results pane displays the description of the `CUSTOMERS` table, which has three columns: `CustomerID`, `CustomerName`, and `Email`.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMERS	CustomerID	NUMBER	22	-	0	-	✓	-	-
	CustomerName	VARCHAR2	100	-	-	-	✓	-	-
	Email	VARCHAR2	150	-	-	-	✓	-	-

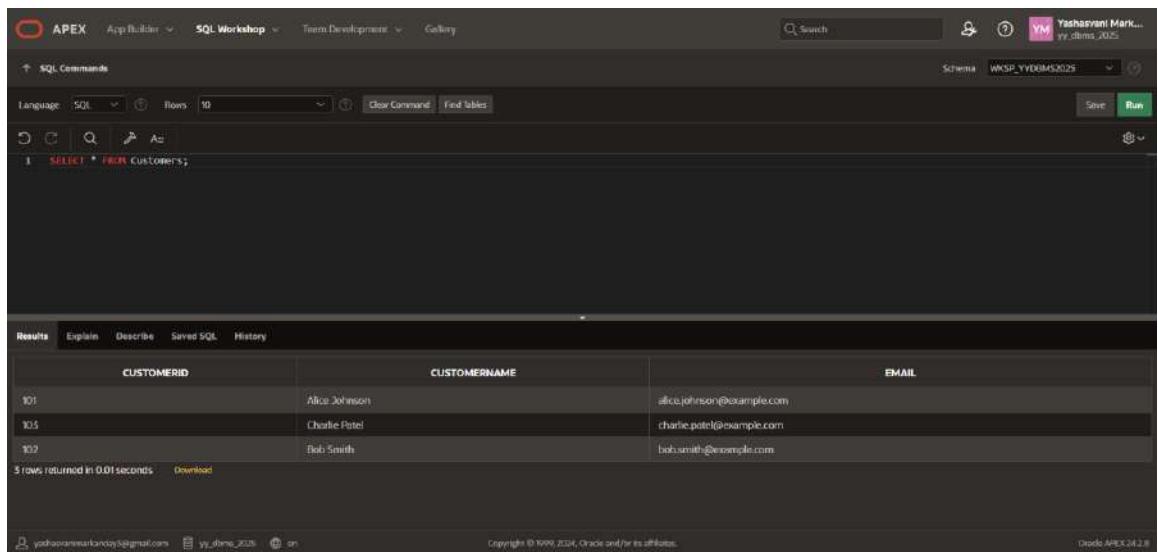
2. Insert at least three sample records into the `Products` table created in Question 1. Ensure each record has a unique product ID. Insert at least three sample records into the `Customers` table created in Question 1. Each record should have a unique customer ID.



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown shows 'WKSP_YYDBMS2025'. The main area contains a SQL command line with the query: 'SELECT * FROM Products;'. Below the command line is a results grid showing three rows of product data:

PRODUCTID	PRODUCTNAME	UNITPRICE	MANUFACTURER
2	Mechanical Keyboard	249.99	Keychron
3	27-inch Monitor	1599.99	Dell
1	Wireless Mouse	79.99	Logitech

Below the grid, it says '3 rows returned in 0.01 seconds' and there is a 'Download' link.

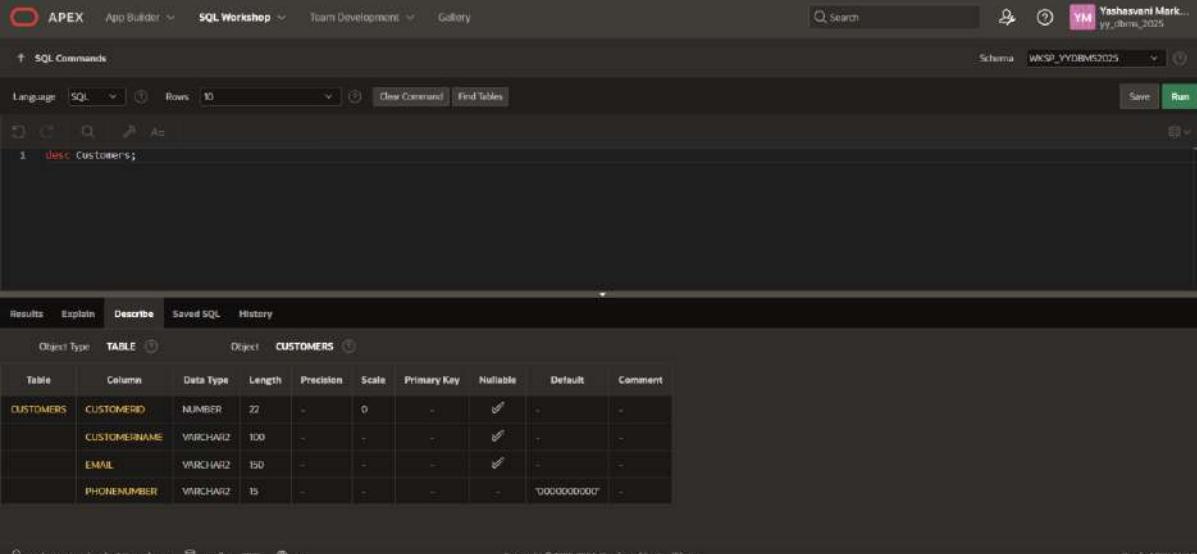


The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown shows 'WKSP_YYDBMS2025'. The main area contains a SQL command line with the query: 'SELECT * FROM Customers;'. Below the command line is a results grid showing three rows of customer data:

CUSTOMERID	CUSTOMERNAME	EMAIL
101	Alice Johnson	alice.johnson@example.com
105	Charlie Patel	charlie.patel@example.com
102	Bob Smith	bob.smith@example.com

Below the grid, it says '3 rows returned in 0.01 seconds' and there is a 'Download' link.

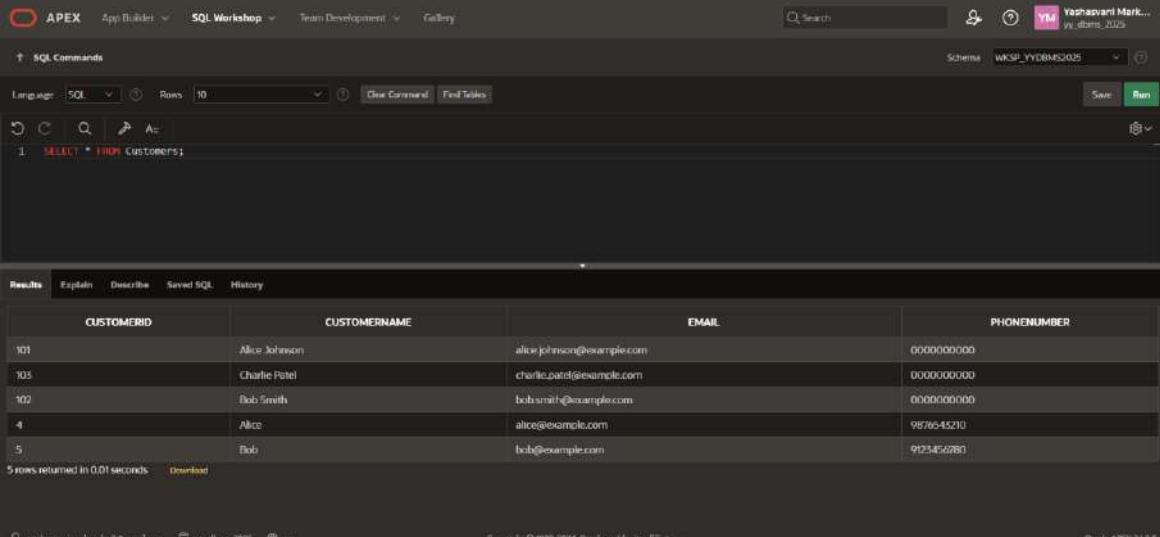
3. Add a column named `PhoneNumber` to the `Customers` table. Apply the NOT NULL constraint to ensure that this information is always provided.



The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, the command `desc CUSTOMERS;` is run, displaying the current structure of the CUSTOMERS table. The table has four columns: CUSTOMERID, CUSTOMERNAME, EMAIL, and PHONENUMBER. The PHONENUMBER column is defined as VARCHAR2(15) and is currently null. In the background, the table structure is being modified to add a new column.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMERS	CUSTOMERID	NUMBER	22	+	0	-	✓	-	-
CUSTOMERS	CUSTOMERNAME	VARCHAR2	100	-	-	-	✓	-	-
CUSTOMERS	EMAIL	VARCHAR2	150	-	-	-	✓	-	-
CUSTOMERS	PHONENUMBER	VARCHAR2	15	-	-	-	-	'0000000000'	-

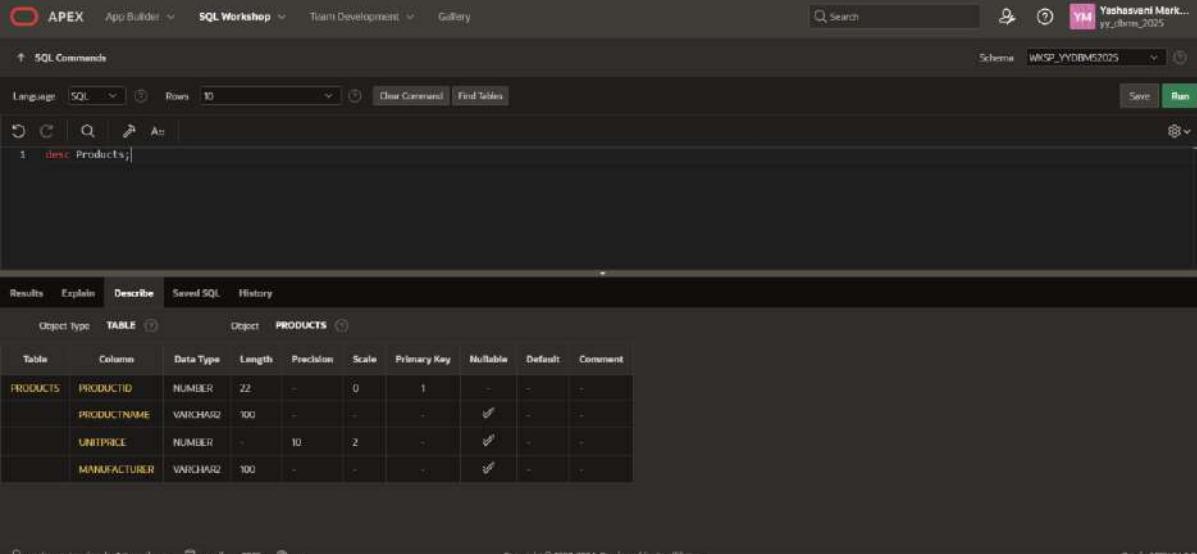
4. Implement a UNIQUE constraint on the `Email` column in the `Customers` table. Provide an example of how this constraint prevents data duplication.



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands pane contains the query `SELECT * FROM CUSTOMERS;`. The Results pane displays the data from the CUSTOMERS table, which includes five rows with unique email addresses: alice.johnson@example.com, charlie.patel@example.com, bob.smith@example.com, alice@example.com, and bob@example.com. The PHONENUMBER column is populated with placeholder values like 0000000000 and 9876543210.

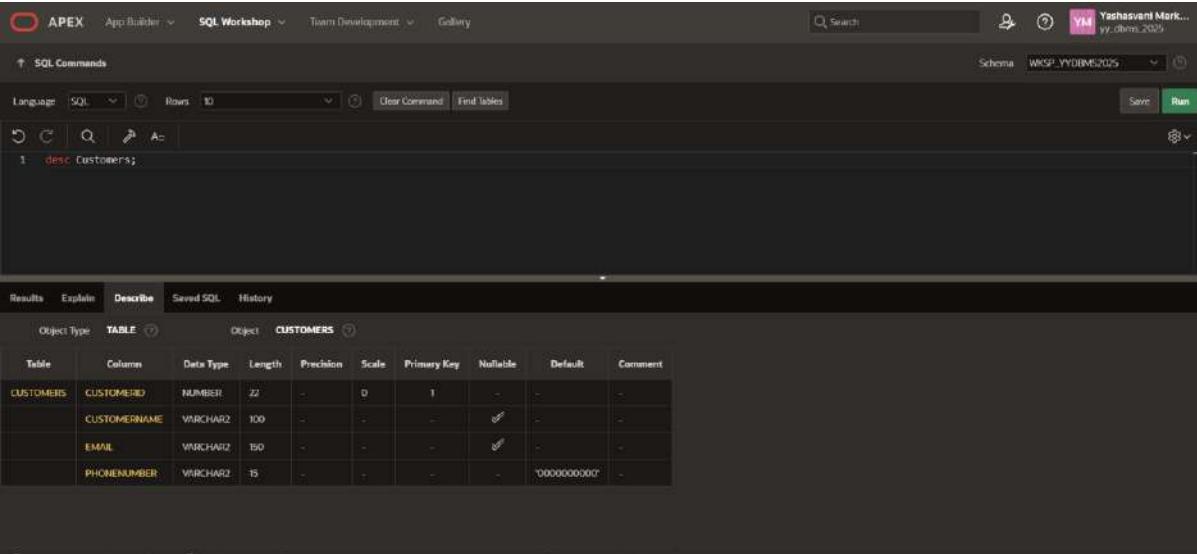
CUSTOMERID	CUSTOMERNAME	EMAIL	PHONENUMBER
101	Alice Johnson	alice.johnson@example.com	0000000000
105	Charlie Patel	charlie.patel@example.com	0000000000
102	Bob Smith	bob.smith@example.com	0000000000
4	Alice	alice@example.com	9876543210
5	Bob	bob@example.com	9876543210

5. Set the `ProductID` column as the primary key for the `Products` table. Also, define `CustomerID` as the primary key for the `Customers` table.



The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKS_P_YDBMS205. In the SQL Commands pane, the command `desc Products;` is entered. The results pane displays the description of the `PRODUCTS` table, which has four columns: `PRODUCTID`, `PRODUCTNAME`, `UNITPRICE`, and `MANUFACTURER`. The `PRODUCTID` column is identified as the Primary Key.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PRODUCTS	PRODUCTID	NUMBER	22	-	0	1	-	-	-
	PRODUCTNAME	VARCHAR2	100	-	-	-	✓	-	-
	UNITPRICE	NUMBER	-	10	2	-	✓	-	-
	MANUFACTURER	VARCHAR2	100	-	-	-	✓	-	-



The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKS_P_YDBMS205. In the SQL Commands pane, the command `desc Customers;` is entered. The results pane displays the description of the `CUSTOMERS` table, which has four columns: `CUSTOMERID`, `CUSTOMERNAME`, `EMAIL`, and `PHONENUMBER`. The `CUSTOMERID` column is identified as the Primary Key.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMERS	CUSTOMERID	NUMBER	22	-	0	1	-	-	-
	CUSTOMERNAME	VARCHAR2	100	-	-	-	✓	-	-
	EMAIL	VARCHAR2	150	-	-	-	✓	-	-
	PHONENUMBER	VARCHAR2	15	-	-	-	-	'0000000000'	-

6. Establish a relationship between the `Orders` table (containing columns `OrderID` and `CustomerID`) and the `Customers` table. Ensure referential integrity by using a FOREIGN KEY constraint.

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user profile 'Yashasvani Mark... yy_dbms_2025'. The main area has tabs for 'SQL Commands', 'Language: SQL', 'Rows: 10', 'One Command', 'Find Tables', 'Schema: WKSP_YYDBMS2025', 'Save', and 'Run'. Below these are icons for Undo, Redo, Search, and Paste. The SQL editor contains the command 'desc Orders;'. The results section has tabs for 'Results', 'Explain', 'Describe' (selected), 'Saved SQL', and 'History'. It displays the table structure for 'ORDERS' with columns: ORDERID, CUSTOMERID, and ORDERDATE. The 'Describe' tab also shows the primary key is ORDERID.

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar and schema selection are identical to the previous screenshot. The SQL editor contains the command 'SELECT * FROM Orders;'. The results section shows the output of the query:

ORDERID	CUSTOMERID	ORDERDATE
1	4	8/22/2025

Below the table, it says '1 rows returned in 0.02 seconds' and provides a 'Download' link. The bottom of the screen includes the footer information: 'yashasvanimarkanday@gmail.com yy_dbms_2025', 'Copyright © 1999-2024, Oracle and/or its affiliates.', and 'Oracle APEX 24.2.0'.

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, there is a single line of SQL code:

```
1 INSERT INTO Orders (OrderID, CustomerID, OrderDate)
2 VALUES (2, 99, TO_DATE('2025-08-22', 'YYYY-MM-DD'));
```

In the Results pane, the output is displayed in a table:

SQL	Result
1 INSERT INTO Orders (OrderID, CustomerID, OrderDate) 2 VALUES (2, 99, TO_DATE('2025-08-22', 'YYYY-MM-DD'));	DRA-02292: integrity constraint (WKSP_YYDBMS2025.SYS_C00183833853) violated - parent key not found

Below the results, it says "0.01 seconds". At the bottom, the user information is "yashwanmarkanday@gmail.com" and the schema is "WKSP_YYDBMS2025". The copyright notice is "Copyright © 1999-2024, Oracle and/or its affiliates." and the version is "Oracle APEX 24.2.0".

7. Add a column named `OrderQuantity` to the `Orders` table. Apply a CHECK constraint to limit the order quantity to a range between 1 and 100.

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, there is a single line of SQL code:

```
1 INSERT INTO Orders (OrderID, CustomerID, OrderDate, OrderQuantity)
2 VALUES (16, 4, TO_DATE('2025-08-22', 'YYYY-MM-DD'), 50);
```

In the Results pane, the output is displayed in a table:

SQL	Result
1 INSERT INTO Orders (OrderID, CustomerID, OrderDate, OrderQuantity) 2 VALUES (16, 4, TO_DATE('2025-08-22', 'YYYY-MM-DD'), 50);	1 row(s) inserted.

Below the results, it says "0.03 seconds". At the bottom, the user information is "yashwanmarkanday@gmail.com" and the schema is "WKSP_YYDBMS2025". The copyright notice is "Copyright © 1999-2024, Oracle and/or its affiliates." and the version is "Oracle APEX 24.2.0".

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command Find Tables

Run

1 `SELECT * FROM Orders;`

Results Explain Describe Saved SQL History

ORDERID	CUSTOMERID	ORDERDATE	ORDERQUANTITY
1	4	8/22/2025	-
10	4	8/22/2025	50

2 rows returned in 0.02 seconds Download

vishwasvamarkanday@gmail.com yy_dbms_2025 en Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command Find Tables

Run

1 `INSERT INTO Orders (OrderID, CustomerID, OrderDate, OrderQuantity)`

2 `VALUES (11, 4, TO_DATE('2025-08-22', 'YYYY-MM-DD'), 150);`

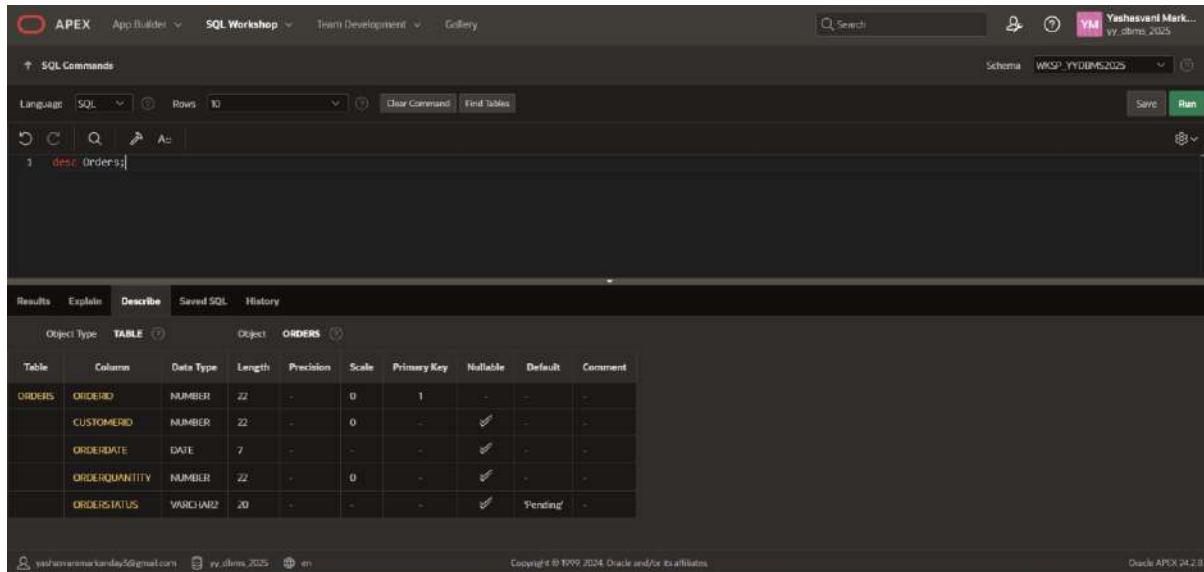
Results Explain Describe Saved SQL History

DRA-02290: check constraint (WSP_YYDBMS2015.SYS_C00183851675) violated

0.02 seconds

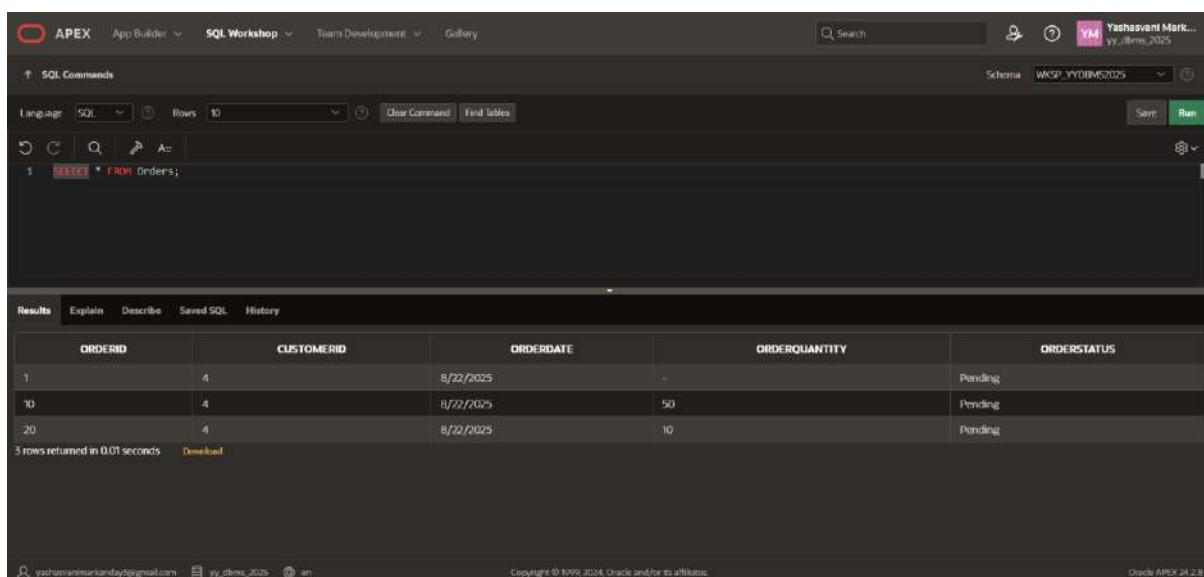
vishwasvamarkanday@gmail.com yy_dbms_2025 en Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

8. Include a column named 'OrderStatus' in the 'Orders' table. Set a DEFAULT constraint so that if the status is not specified during insertion, it defaults to 'Pending.'



The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_YYDBMS2025'. In the SQL Commands pane, the command `desc Orders;` is run, displaying the table structure:

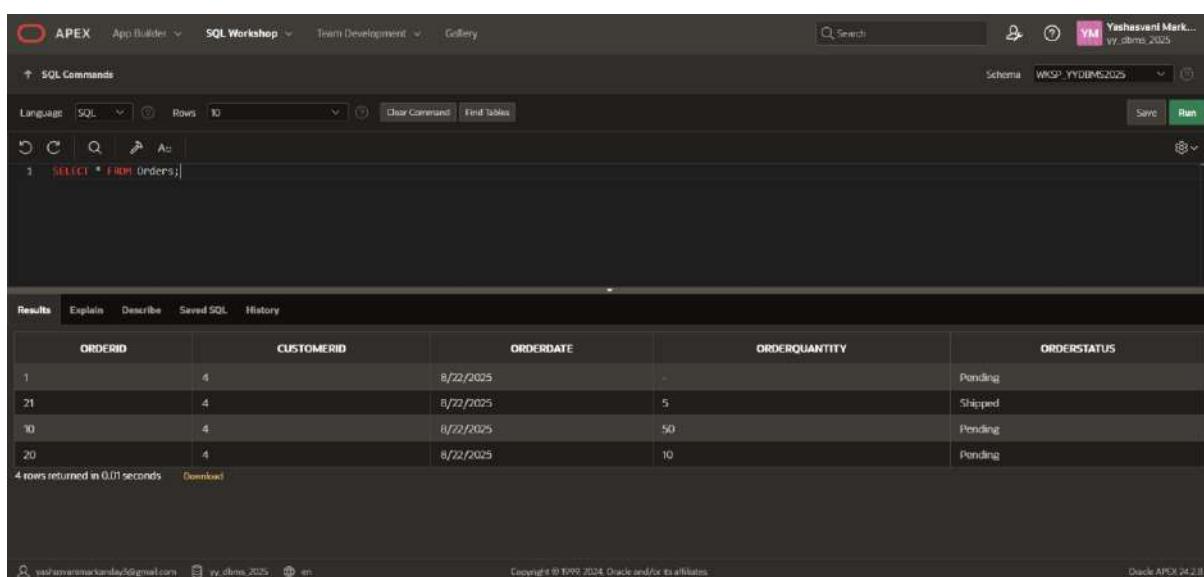
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDERS	ORDERID	NUMBER	22	-	0	1	-	-	-
	CUSTOMERID	NUMBER	22	-	0	-	✓	-	-
	ORDERDATE	DATE	7	-	-	-	✓	-	-
	ORDERQUANTITY	NUMBER	22	-	0	-	✓	-	-
	ORDERSTATUS	VARCHAR2	20	-	-	-	✓	'Pending'	-



The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_YYDBMS2025'. In the SQL Commands pane, the command `SELECT * FROM Orders;` is run, displaying the following data:

ORDERID	CUSTOMERID	ORDERDATE	ORDERQUANTITY	ORDERSTATUS
1	4	8/22/2025	-	Pending
10	4	8/22/2025	50	Pending
20	4	8/22/2025	10	Pending

3 rows returned in 0.01 seconds

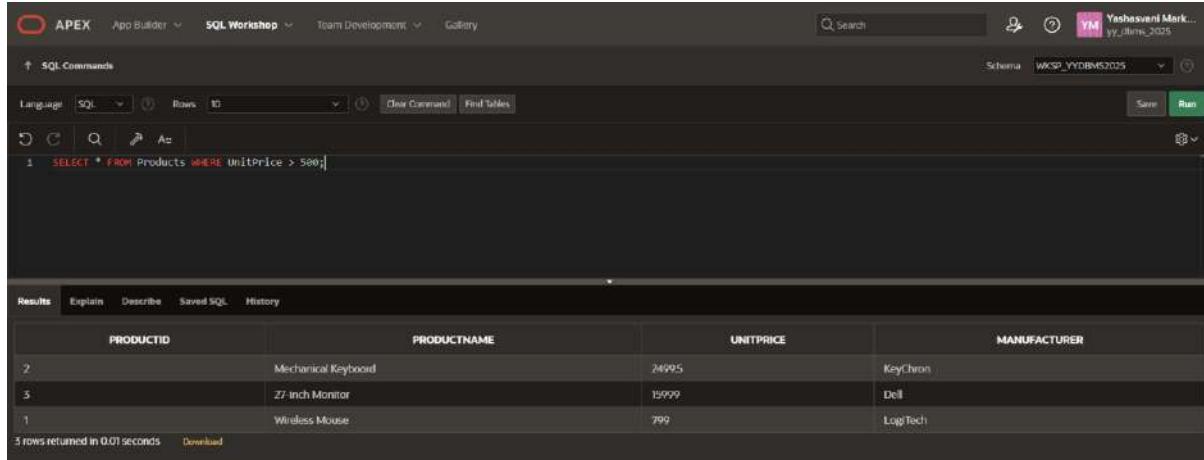


The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_YYDBMS2025'. In the SQL Commands pane, the command `SELECT * FROM Orders;` is run, displaying the following data:

ORDERID	CUSTOMERID	ORDERDATE	ORDERQUANTITY	ORDERSTATUS
1	4	8/22/2025	-	Pending
21	4	8/22/2025	5	Shipped
10	4	8/22/2025	50	Pending
20	4	8/22/2025	10	Pending

4 rows returned in 0.01 seconds

9. Create an index on the `UnitPrice` column in the `Products` table. Explain how this index can improve query performance.



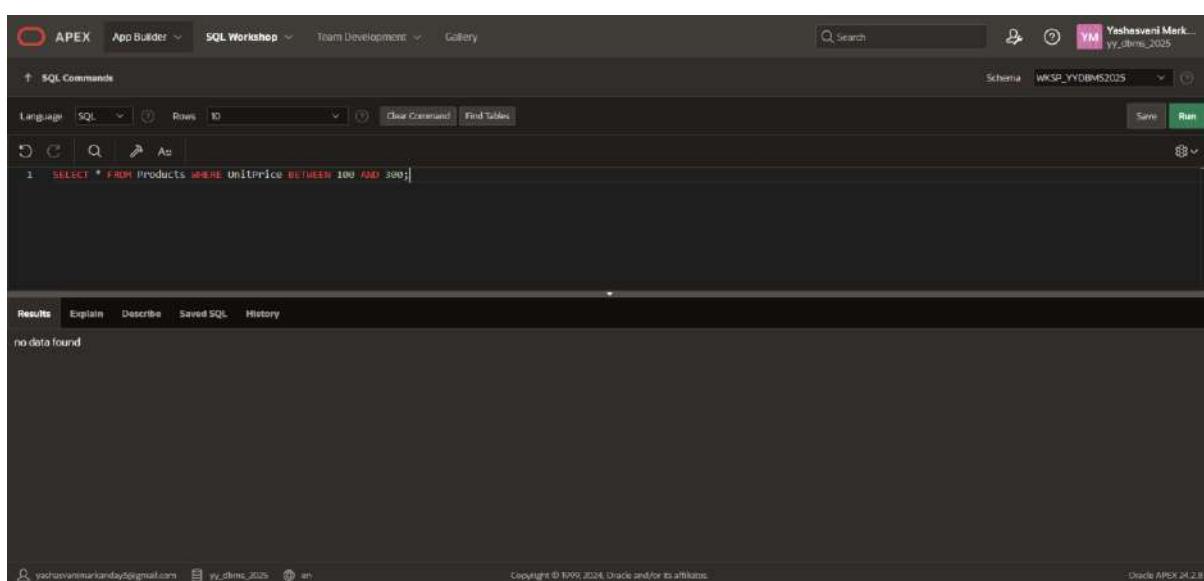
The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKS_P_YYDBMS2025. A SQL command is run:

```
1 SELECT * FROM Products WHERE UnitPrice > 500;
```

The results table displays three rows:

PRODUCTID	PRODUCTNAME	UNITPRICE	MANUFACTURER
2	Mechanical Keyboard	24995	KeyChron
3	27 Inch Monitor	15999	Dell
1	Wireless Mouse	799	LogTech

5 rows returned in 0.01 seconds. The results tab is selected.

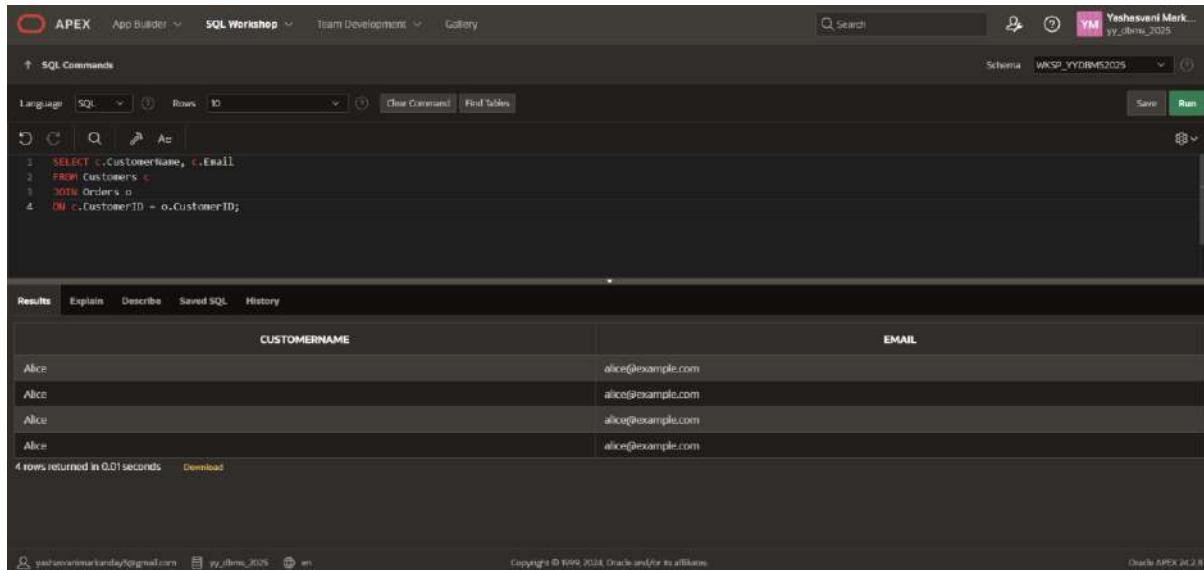


The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKS_P_YYDBMS2025. A SQL command is run:

```
1 SELECT * FROM Products WHERE UnitPrice BETWEEN 100 AND 300;
```

The results table shows no data found.

10. Write an SQL query to retrieve the names and email addresses of all customers who placed orders. Join the `Customers` and `Orders` tables to achieve this.



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following query:

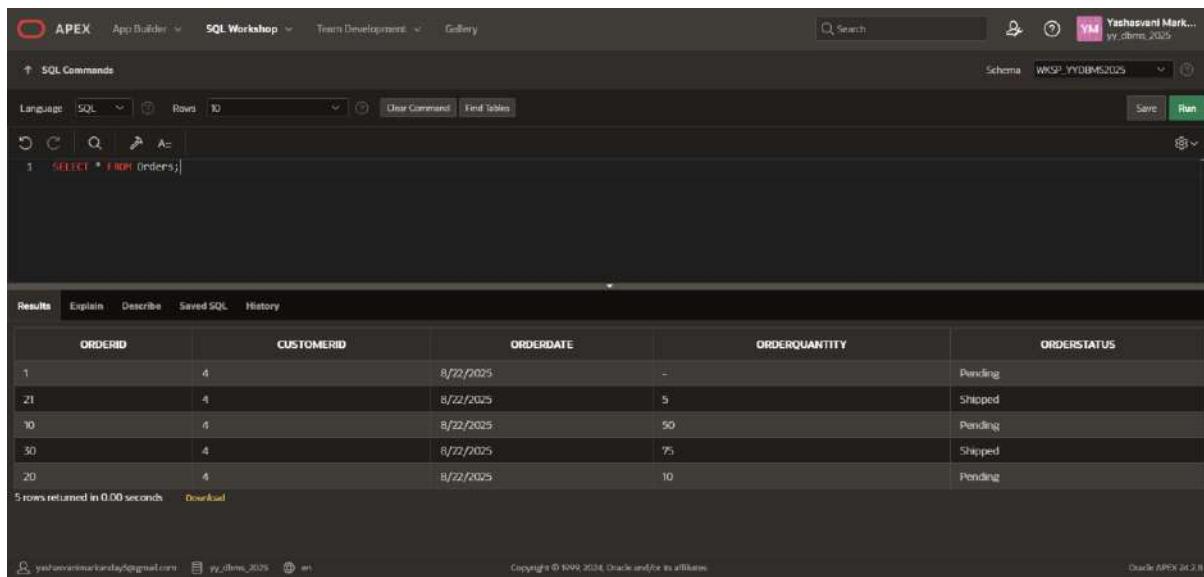
```
1 SELECT c.CustomerName, c.Email
2 FROM Customers c
3 JOIN Orders o
4 ON c.CustomerID = o.CustomerID;
```

The Results tab displays the output:

CUSTOMERNAME	EMAIL
Alice	alice@example.com

4 rows returned in 0.01 seconds

11. Modify the `OrderStatus` of all orders with a quantity greater than 50 to 'Shipped.' Provide the SQL statement for this update.



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following query:

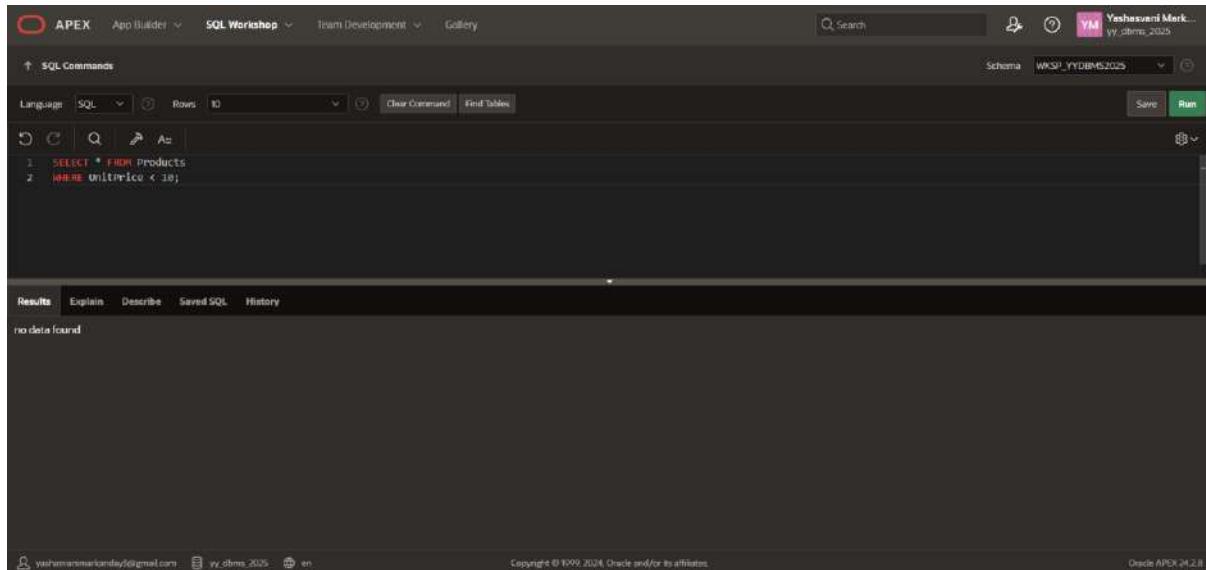
```
1 SELECT * FROM Orders;
```

The Results tab displays the output:

ORDERID	CUSTOMERID	ORDERDATE	ORDERQUANTITY	ORDERSTATUS
1	4	8/22/2025	-	Pending
21	4	8/22/2025	5	Shipped
10	4	8/22/2025	50	Pending
30	4	8/22/2025	75	Shipped
20	4	8/22/2025	10	Pending

5 rows returned in 0.00 seconds

12. Delete all records from the `Products` table where the `UnitPrice` is less than \$10. Include the SQL statement for this deletion.

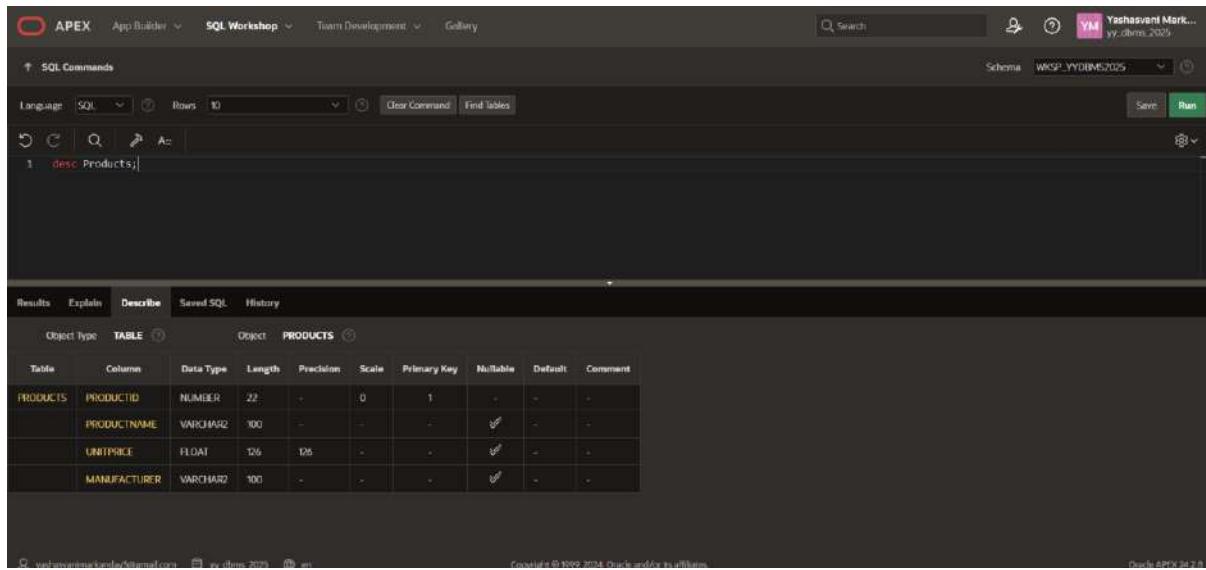


The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKS_P_YYDBMS2025. The main area displays the following SQL command:

```
1 SELECT * FROM products
2 WHERE UnitPrice < 10;
```

The results pane below shows the message "no data found".

13. The `UnitPrice` column in the `Products` table needs to be changed to a new datatype, `float`. Write an SQL statement using the ALTER TABLE statement to make this change.



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKS_P_YYDBMS2025. The main area displays the following SQL command:

```
1 desc Products;
```

The results pane below shows the table structure for the `PRODUCTS` table:

Object Type	Table	Object	PRODUCTS						
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PRODUCTS	PRODUCTID	NUMBER	22	-	0	1	-	-	-
PRODUCTS	PRODUCTNAME	VARCHAR2	100	-	-	-	Y	-	-
PRODUCTS	UNITPRICE	FLOAT	126	126	-	-	Y	-	-
PRODUCTS	MANUFACTURER	VARCHAR2	100	-	-	-	Y	-	-

14. Write an SQL query to calculate the average `UnitPrice` of all products in the `Products` table.

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows a user profile for 'Yoshavani Mark...' and the schema 'WKSP_YYDBMS2025'. The main workspace is titled 'SQL Commands' with a dropdown menu showing 'Language: SQL'. Below this is a toolbar with icons for Undo, Redo, Find, Copy, Paste, and Run. The SQL editor contains the following code:

```
1 SELECT AVG(UnitPrice) AS AveragePrice
2 FROM Products;
```

Below the code, the 'Results' tab is selected, showing the output:

AVERAGEPRICE
64.525

Text below the results indicates '1 rows returned in 0.02 seconds' and provides download options. The bottom of the screen displays copyright information for Oracle and the APEX version.



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Program Name: B. Tech CS

Semester: III

Name: Yashasvani Markanday

SAP ID: 590011386

Batch: B2

Subject: Database and Management System

Tables:

1.

The screenshot shows the Oracle APEX SQL Workshop interface. The user has run the command `desc Student_details;`. The results pane displays the structure of the `STUDENT_DETAILS` table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT_DETAILS	STUDENT_ROLLNO	NUMBER	22	-	0	1	-	-	-
	STU_NAME	VARCHAR2	20	-	-	-	-	-	-
	STU_MARKS	NUMBER	22	-	0	-	✓	-	-
	STU_CITY	VARCHAR2	50	-	-	-	✓	-	-

2.

The screenshot shows the Oracle APEX SQL Workshop interface. The user has run the command `desc Faculty_details;`. The results pane displays the structure of the `FACULTY_DETAILS` table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FACULTY_DETAILS	FACULTY_ID	NUMBER	22	-	0	1	-	-	-
	FAC_NAME	VARCHAR2	20	-	-	-	✓	-	-
	DEPT_ID	NUMBER	22	-	0	-	✓	-	-
	ADDRESS	VARCHAR2	50	-	-	-	✓	-	-

3.

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKS_P_YYDBMS2025. In the SQL Commands panel, the command `desc Department;` is entered. The results show the structure of the DEPARTMENT table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	DEPT_ID	NUMBER	22	-	-	1	-	-	-
FACULTY.ID	NUMBER	22	-	-	-	-	✓	-	-
DEPT_NAME	VARCHAR2	100	-	-	-	-	✓	-	-

4.

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKS_P_YYDBMS2025. In the SQL Commands panel, the command `desc Old_Employee;` is entered. The results show the structure of the OLD_EMPLOYEE table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
OLD_EMPLOYEE	EMP_ID	NUMBER	22	-	-	1	-	-	-
	EMP_NAME	VARCHAR2	100	-	-	-	✓	-	-
	EMP_SALARY	NUMBER	-	10	2	-	✓	-	-
	ADDRESS	VARCHAR2	200	-	-	-	✓	-	-

5.

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, the 'SQL Workshop' tab is selected. The main area contains the following SQL command:

```
1 desc New_Employee;
```

Below the command, the results section displays the structure of the 'NEW_EMPLOYEE' table:

Object Type	TABLE	Object	NEW_EMPLOYEE						
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
NEW_EMPLOYEE	EMP_ID	NUMBER	22	-	-	1	-	-	-
	EMP_NAME	VARCHAR2	100	-	-	-	✓	-	-
	EMP_SALARY	NUMBER	-	10	2	-	✓	-	-
	ADDRESS	VARCHAR2	200	-	-	-	✓	-	-

At the bottom of the interface, there is a footer with the text "Copyright © 1999-2014, Oracle and/or its affiliates." and "Oracle APEX 2d.2.0".

6.

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, the 'SQL Workshop' tab is selected. The main area contains the following SQL command:

```
1 desc Employee_Details;
```

Below the command, the results section displays the structure of the 'EMPLOYEE_DETAILS' table:

Object Type	TABLE	Object	EMPLOYEE_DETAILS						
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE_DETAILS	EMP_ID	NUMBER	22	-	-	1	-	-	-
	EMP_NAME	VARCHAR2	100	-	-	-	✓	-	-
	EMP_SALARY	NUMBER	-	10	2	-	✓	-	-
	DEPT_ID	NUMBER	22	-	-	-	✓	-	-

At the bottom of the interface, there is a footer with the text "Copyright © 1999-2014, Oracle and/or its affiliates." and "Oracle APEX 2d.2.0".

7.

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a user profile for 'Yashaswini Mark...' and the schema 'WKSP_YYDBMS2025'. The main area has tabs for 'SQL Commands' and 'Saved SQL'. The SQL command input field contains 'desc Student;'. Below the command is a table description:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT	STUDENT_ID	NUMBER	22	-	-	1	-	-	-
	NAME	VARCHAR2	100	-	-	-	✓	-	-
	CITY	VARCHAR2	100	-	-	-	✓	-	-

At the bottom, it says 'Copyright © 1999-2024, Oracle and/or its affiliates.' and 'Oracle APEX 24.2.0'.

8.

The screenshot shows the Oracle APEX SQL Workshop interface, similar to the previous one. The top navigation bar and schema information are identical. The SQL command input field now contains 'desc student2;'. Below the command is a table description:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT2	STUDENT_ID	NUMBER	22	-	-	1	-	-	-
	CITY	VARCHAR2	100	-	-	-	✓	-	-

At the bottom, it says 'Copyright © 1999-2024, Oracle and/or its affiliates.' and 'Oracle APEX 24.2.0'.

9.

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. In the SQL Commands section, the command `desc ORDERS2` is entered. The results show the description of the ORDERS2 table, which has three columns: ORDER_ID, CUST_ID, and ORDER_DATE. The ORDER_ID column is defined as NUMBER(22,0) and is the primary key. The CUST_ID column is also NUMBER(22,0). The ORDER_DATE column is defined as DATE.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDERS2	ORDER_ID	NUMBER	22	-	-	1	-	-	-
	CUST_ID	NUMBER	22	-	-	-	✓	-	-
	ORDER_DATE	DATE	7	-	-	-	✓	-	-

10.

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. In the SQL Commands section, the command `desc SALES` is entered. The results show the description of the SALES table, which has two columns: PRODUCT_CATEGORY and SALES_AMOUNT. The PRODUCT_CATEGORY column is defined as VARCHAR2(100) and is the primary key. The SALES_AMOUNT column is defined as NUMBER(12,2).

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SALES	PRODUCT_CATEGORY	VARCHAR2	100	-	-	1	✓	-	-
	SALES_AMOUNT	NUMBER	-	12	2	-	✓	-	-

11.

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The schema is set to 'WKSP_YYDBMS2025'. The main area displays the following SQL command:

```
1 desc Top_Students;
```

Below the command, the 'Describe' tab is selected, showing the structure of the 'TOP_STUDENTS' table:

Object Type	Table
Table	TOP_STUDENTS
Column	STUDENT_ID
Data Type	NUMBER
Length	22
Precision	-
Scale	-
Primary Key	1
Nullable	-
Default	-
Comment	-

Another row for 'TOP_MARKS' is partially visible below it.

SQL Subquery Questions:

1. Basic SELECT Subquery

- Retrieve the names and marks of students whose marks are greater than the average marks.

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT Stu_name, Stu_marks
2 FROM Student_details
3 WHERE Stu_marks >
4   (SELECT AVG(Stu_marks)
5    FROM Student_details
6 );
```

The results show the names and marks of students whose marks are above the average:

STU_NAME	STU_MARKS
Chetan	95
Raman	90
Bhavin	87
Diksha	93

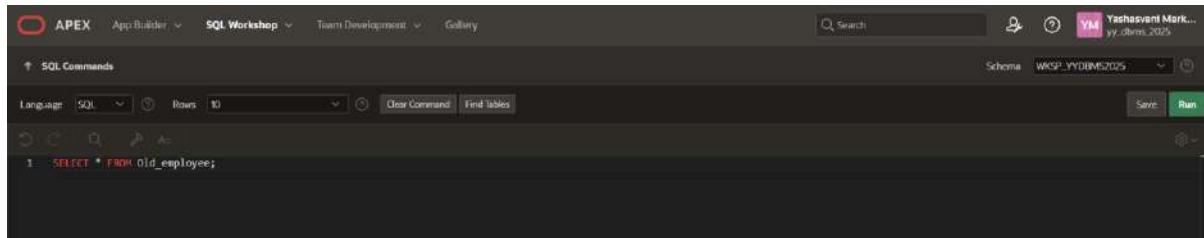
4 rows returned in 0.01 seconds.

2. IN Operator Subquery

Fetch the names and addresses of faculties who belong to departments in either 'Noida' or 'Gurgaon.'

3. INSERT with Subquery

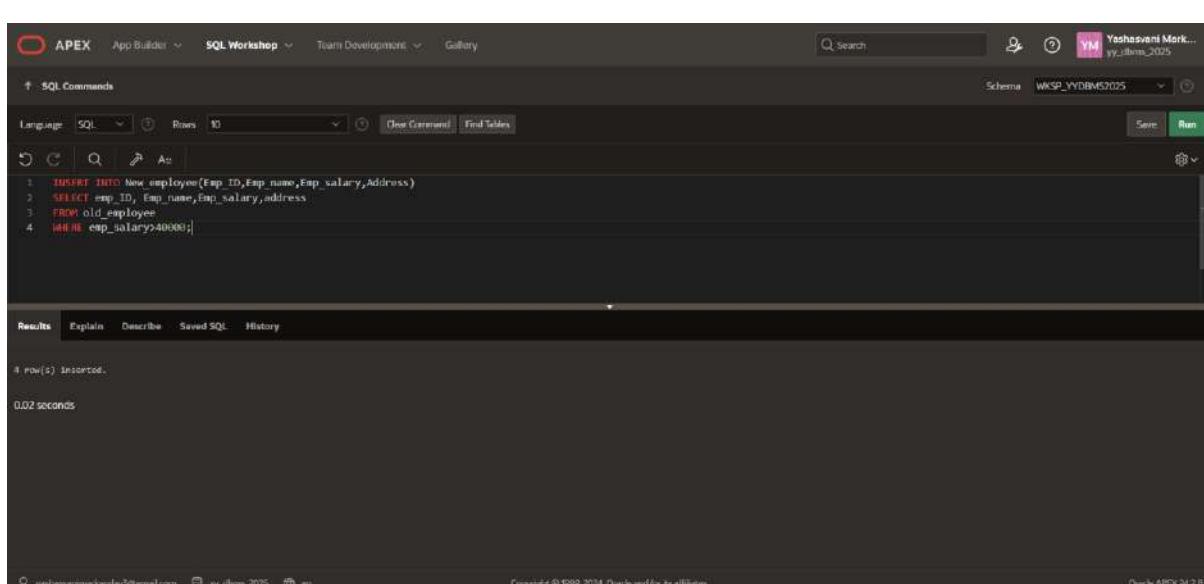
- Insert the details of employees from the "Old_Employee" table into the "New_Employee" table, considering only those with a salary greater than 40000.



The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS7025. The command bar has 'Language: SQL' selected, 'Rows: 10', and the 'Run' button is highlighted. The results pane displays a table with columns: EMP_ID, EMP_NAME, EMP_SALARY, and ADDRESS. The data includes rows for Chetan, Diksha, Akhil, Bheem, Balram, Romon, and Sheetal, all with salaries above 40000. The results are returned in 0.02 seconds.

EMP_ID	EMP_NAME	EMP_SALARY	ADDRESS
1004	Chetan	60000	Noida
1005	Diksha	30000	Agra
1001	Akhil	50000	Agra
1005	Bheem	45000	Gurgaon
1002	Balram	25000	Delhi
1006	Romon	50000	Ghaziabad
1007	Sheetal	55000	Delhi

7 rows returned in 0.02 seconds [Download](#)



The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS7025. The command bar has 'Language: SQL' selected, 'Rows: 10', and the 'Run' button is highlighted. The results pane shows the execution of an INSERT INTO statement into the New_Employee table, selecting data from the Old_Employee table where the salary is greater than 40000. The message '4 row(s) inserted.' is displayed.

```
1: INSERT INTO New_Employee(Empl_ID,Emp_name,Emp_salary,Address)
2: SELECT emp_ID, Emp_name,Emp.salary,Address
3: FROM Old_Employee
4: WHERE emp_salary>40000;
```

4 row(s) inserted.

0.02 seconds

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Rows 10 Clear Command Find Tables Schema WKSP_YYDBMS2025 Save Run

```
1 select * from new_employee;
```

Results Explain Describe Saved SQL History

EMP_ID	EMP_NAME	EMP_SALARY	ADDRESS
1008	Sumit	50000	Agra
1009	Akash	55000	Delhi
1010	Devansh	65000	Gurgaon
1001	Cheetan	60000	Noida
1001	Akhil	50000	Agra
1003	Bheem	45000	Gurgaon
1006	Raman	50000	Ghaziabad

7 rows returned in 0.01 seconds Download

yashasvari.merkanday@gmail.com yy_dbms_2025 en Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

4. UPDATE with Subquery: - Increase the salary of employees by 10% for those whose department grade is 'A.'

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Rows 10 Clear Command Find Tables Schema WKSP_YYDBMS2025 Save Run

```
1 UPDATE Employee_Details
2 SET Emp_Salary = Emp_Salary * 1.10
3 WHERE Dept_ID IN (
4   SELECT Dept_ID
5   FROM Department
6   WHERE Dept_Name = 'A'
7 );
```

Results Explain Describe Saved SQL History

0 row(s) updated.

0.05 seconds

yashasvari.merkanday@gmail.com yy_dbms_2025 en Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_YYDBMS2025'. In the SQL Commands pane, the following query is entered:

```
1 SELECT * FROM employee_details;
```

The Results pane displays the query results:

EMP_ID	EMP_NAME	EMP_SALARY	DEPT_ID
1001	Akhil	50000	404
1002	Balram	25000	405

2 rows returned in 0.00 seconds.

5. DELETE with Subquery: - Delete the records of employees whose department grade is 'C.'

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_YYDBMS2025'. In the SQL Commands pane, the following query is entered:

```
1 DELETE FROM Employee_Details
2 WHERE Dept_ID IN (
3   SELECT Dept_ID
4   FROM department
5   WHERE Dept_Name = 'C'
6 );
```

The Results pane shows the output:

0 row(s) deleted.
0.02 seconds

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_YYDBMS2025'. In the SQL Commands pane, the following query is entered:

```
1 SELECT * FROM Employee_Details;
```

The Results pane displays the query results:

EMP_ID	EMP_NAME	EMP_SALARY	DEPT_ID
1001	Akhil	50000	404
1002	Balram	25000	405

2 rows returned in 0.00 seconds.

6. Subquery in WHERE Clause - NOT IN - Retrieve the names of students who do not belong to the city 'Los Angeles' based on data from the "Student2" table.

The screenshot shows the Oracle SQL Developer interface. In the SQL tab, the following query is written:

```
1 SELECT stu_NAME
2 FROM STUDENT
3 WHERE STU_ID NOT IN (
4   SELECT S2.STU_ID
5   FROM STUDENT2 S2
6   WHERE S2.CITY = 'Los Angeles'
7 );
```

The Results tab displays the output:

STU_NAME
Balram
Akhil
Divya

Below the results, it says "3 rows returned in 0.01 seconds".

7. Subquery in FROM Clause: - Use a subquery in the FROM clause to create a derived table showing the maximum, minimum, and average number of items for each order.

The screenshot shows the Oracle APEX interface. In the SQL Workshop tab, the following CREATE TABLE statement is executed:

```
CREATE Table order_details(
  Order_ID NUMBER,
  Product_ID number,
  quantity number
);
```

The Results tab shows the message "Table created." and "0.05 seconds".

The screenshot shows the Oracle APEX interface. In the SQL Workshop tab, the following query is run:

```
SELECT
  MAX(TotalItems) AS Max_Items,
  MIN(TotalItems) AS Min_Items,
  AVG(TotalItems) AS Avg_Items
FROM (
  SELECT Order_ID, SUM(quantity) AS TotalItems
  FROM order_details
  GROUP BY Order_ID
) DERIVEDTABLE;
```

The Results tab displays the output:

MAX_ITEMS	MIN_ITEMS	AVG_ITEMS
-	-	-

Below the results, it says "1 rows returned in 0.01 seconds".

8. Correlated Subquery - Fetch the names, cities, and incomes of employees whose income is higher than the average income of employees in their respective cities.

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```

1 SELECT emp_name, emp_salary
2 FROM Employee_Details
3 WHERE emp_salary > (
4   SELECT AVG(emp_salary)
5   FROM Employee_Details
6 );

```

The results window displays the output:

EMP_NAME	EMP_SALARY
Akash	50000

1 rows returned in 0.01 seconds [Download](#)

9. EXISTS Operator - Retrieve the name, occupation, and age of customers who have placed at least one order using the EXISTS operator.

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```

1 SELECT Cust_Name, Occupation, Age
2 FROM Customers C
3 WHERE EXISTS (
4   SELECT 1
5   FROM Orders O
6   WHERE Cus_ID = Cus_ID
7 );
8

```

The results window displays the output:

CUST_NAME	OCCUPATION	AGE
Meera	Doctor	32
Rahul	Teacher	40
Arjun	Engineer	28

3 rows returned in 0.02 seconds [Download](#)

10. ROW Subquery - Retrieve all columns for customers whose (cust_id, occupation) matches any row of (order_id, order_date) from the "Orders" table.

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```

1 SELECT *
2 FROM Customers
3 WHERE (Cust_ID, Occupation) IN (
4   SELECT Cus_ID, 'Engineer'
5   FROM Orders
6   WHERE Order_Date IS NOT NULL
7 );
8

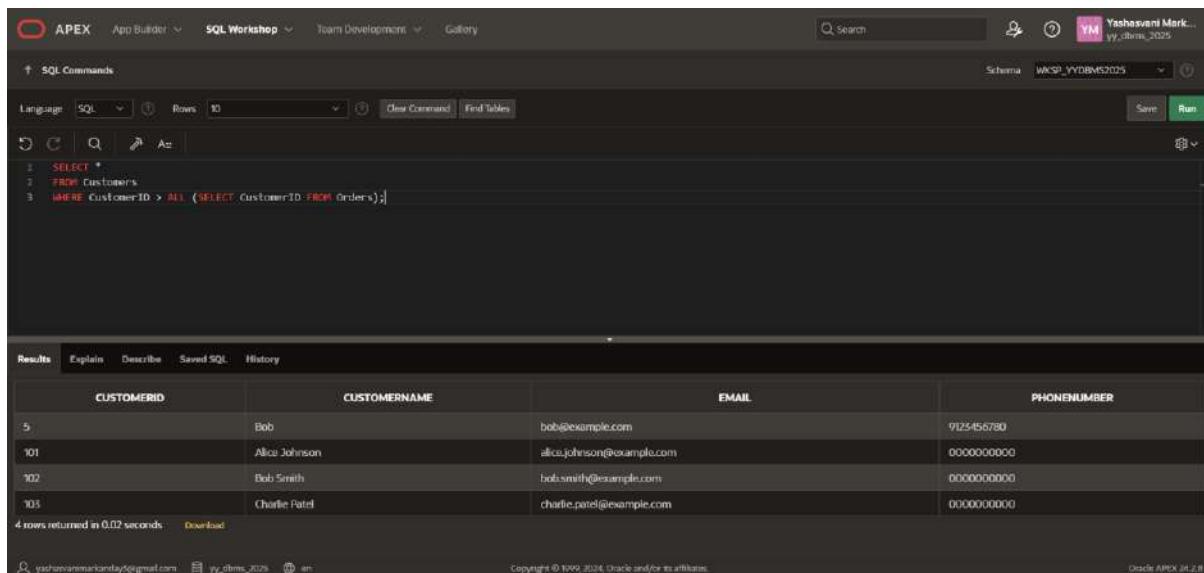
```

The results window displays the output:

CUST_ID	CUST_NAME	OCCUPATION	AGE
1	Arjun	Engineer	28

1 rows returned in 0.04 seconds [Download](#)

11. Subquery with ALL Operator - Find customers whose cust_id is greater than all cust_ids in the "Orders" table.



The screenshot shows the Oracle APEX SQL Workshop interface. The query is:

```

1 SELECT *
2 FROM Customers
3 WHERE CustomerID > ALL (SELECT CustomerID FROM Orders);

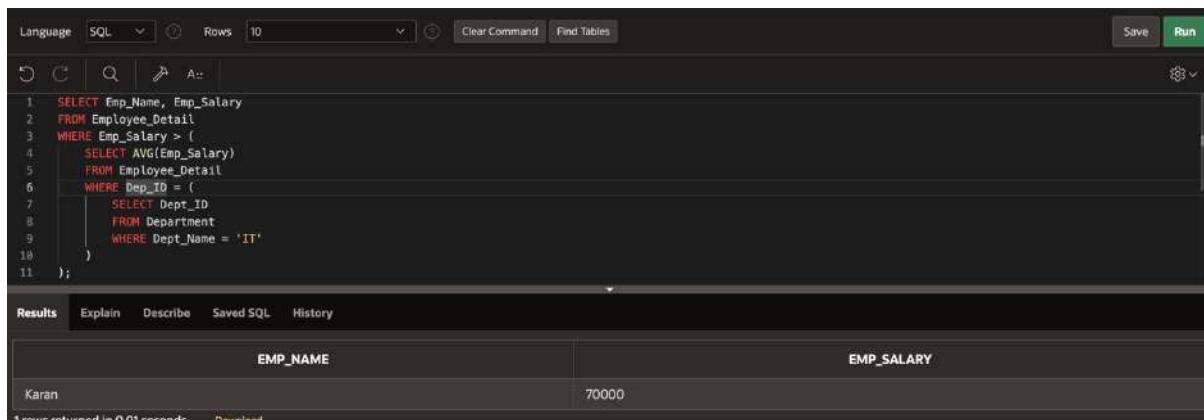
```

The results table has columns: CUSTOMERID, CUSTOMERNAME, EMAIL, and PHONENUMBER. The data is:

CUSTOMERID	CUSTOMERNAME	EMAIL	PHONENUMBER
5	Bob	bob@example.com	9125456780
101	Alicia Johnson	alice.johnson@example.com	0000000000
102	Bob Smith	bob.smith@example.com	0000000000
103	Charlie Patel	charlie.patel@example.com	0000000000

4 rows returned in 0.02 seconds.

12. Nested Subqueries - Write an SQL query using a subquery within another subquery to retrieve names and salaries of employees who earn more than the average salary of employees in the "IT" department.



The screenshot shows the Oracle APEX SQL Workshop interface. The query is:

```

1 SELECT Emp_Name, Emp_Salary
2 FROM Employee_Detail
3 WHERE Emp_Salary > (
4   SELECT AVG(Emp_Salary)
5   FROM Employee_Detail
6   WHERE Dep_ID = (
7     SELECT Dept_ID
8     FROM Department
9     WHERE Dept_Name = 'IT'
10   )
11 );

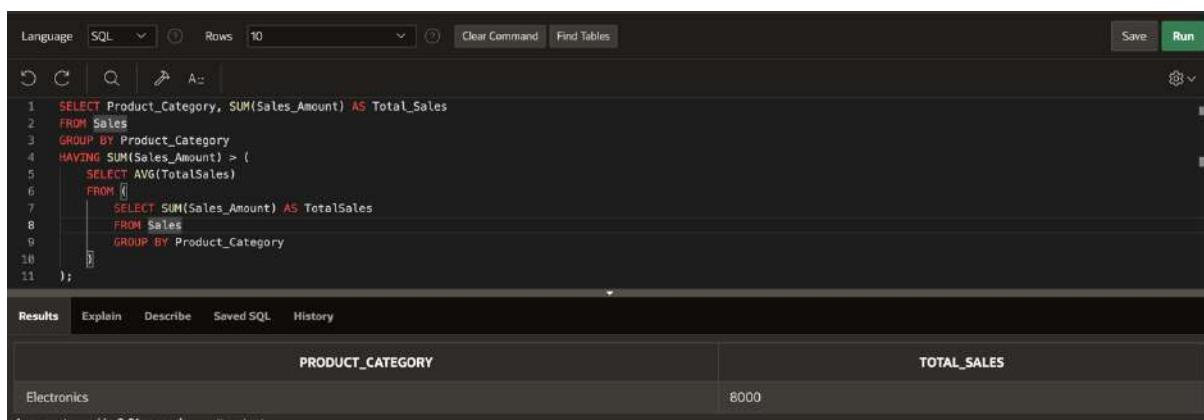
```

The results table has columns: EMP_NAME and EMP_SALARY. The data is:

EMP_NAME	EMP_SALARY
Karin	70000

1 rows returned in 0.01 seconds.

13. Subquery in HAVING Clause - Using the "Sales" table, find the total sales amount for each product category, displaying only those categories where the total sales amount is greater than the average total sales amount.



The screenshot shows the Oracle APEX SQL Workshop interface. The query is:

```

1 SELECT Product_Category, SUM(Sales_Amount) AS Total_Sales
2 FROM Sales
3 GROUP BY Product_Category
4 HAVING SUM(Sales_Amount) > (
5   SELECT AVG(TotalSales)
6   FROM (
7     SELECT SUM(Sales_Amount) AS TotalSales
8     FROM Sales
9     GROUP BY Product_Category
10   )
11 );

```

The results table has columns: PRODUCT_CATEGORY and TOTAL_SALES. The data is:

PRODUCT_CATEGORY	TOTAL_SALES
Electronics	8000

1 rows returned in 0.01 seconds.

14. Subquery with GROUP BY - Retrieve department names and the count of employees in each department, filtering out departments where the count is less than 5.

The screenshot shows a SQL editor interface with the following details:

- Language:** SQL
- Rows:** 10
- Query:**

```
1 SELECT department.dept_name, COUNT(employee.emp_id) AS emp_count
2 FROM employee
3 JOIN department ON employee.dept_id = department.dept_id
4 GROUP BY department.dept_name
5 HAVING COUNT(employee.emp_id) >= 5;
```
- Results:** A table with two columns: DEPT_NAME and EMP_COUNT. The single row returned is HR with a value of 5.
- Timing:** 1 rows returned in 0.00 seconds

15. Multiple Subqueries in a Single Query - Write an SQL query involving multiple subqueries to retrieve information about employees based on various conditions.

The screenshot shows a SQL editor interface with the following details:

- Language:** SQL
- Rows:** 10
- Query:**

```
1 SELECT EMP_NAME, EMP_SALARY, DEPT_ID
2 FROM EMPLOYEE
3 WHERE EMP_SALARY > (SELECT AVG(EMP_SALARY) FROM EMPLOYEE)
4 AND DEPT_ID IN (SELECT DEPT_ID FROM DEPARTMENT WHERE DEPT_GRADE = 'A');
```
- Results:** A table with three columns: EMP_NAME, EMP_SALARY, and DEPT_ID. The two rows returned are Charlie (60000, 1) and Eva (58000, 1).
- Timing:** 2 rows returned in 0.01 seconds

16. Subquery with BETWEEN Operators - Find the names and marks of students who scored between 80 and 90.

The screenshot shows a SQL editor interface with the following details:

- Language:** SQL
- Rows:** 10
- Query:**

```
1 SELECT Stu_Name, Stu_Marks
2 FROM Student_Details
3 WHERE Stu_Marks BETWEEN 80 AND 90;
```
- Results:** A table with two columns: STU_NAME and STU_MARKS. The single row returned is Akhil with a mark of 85.
- Timing:** 1 rows returned in 0.00 seconds

17. Subquery in INSERT Statement - Insert the details of employees from the "Old_Employee" table into the "New_Employee" table, considering only those with a salary greater than the average salary of all employees.

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```

1 INSERT INTO New_Employee (Emp_ID, Emp_Name, Emp_Salary, Address)
2 SELECT Emp_ID, Emp_Name, Emp_Salary, Address
3 FROM Old_Employee
4 WHERE Emp_Salary > (SELECT AVG(Emp_Salary) FROM Old_Employee);
5

```

Results Explain Describe Saved SQL History

2 row(s) inserted.

0.03 seconds

18. Subquery with ANY Operator - Retrieve the names of students who scored more marks than ANY student from the "Top_Students" table.

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```

1 SELECT Stu_Name, Stu_Marks
2 FROM Student_Details
3 WHERE Stu_Marks > ANY (SELECT Top_Marks FROM Top_Students);
4

```

Results Explain Describe Saved SQL History

STU_NAME	STU_MARKS
Rahul	91

1 rows returned in 0.02 seconds Download

19. Subquery in UPDATE Statement - Update the salary of employees by 5% for those whose department is 'HR' and the salary is less than the average salary of HR department employees.

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```

1 UPDATE Employee_Detail
2 SET Emp_Salary = Emp_Salary * 1.05
3 WHERE Dep_ID = (SELECT Dept_ID FROM Department WHERE Dept_Name = 'HR')
4 AND Emp_Salary < (SELECT AVG(Emp_Salary)
5 FROM Employee_Detail
6 WHERE Dep_ID = (SELECT Dept_ID FROM Department WHERE Dept_Name = 'HR')));
7

```

Results Explain Describe Saved SQL History

0 row(s) updated.

0.02 seconds

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```

1 select * from employee_Detail

```

Results Explain Describe Saved SQL History

EMP_ID	EMP_NAME	EMP_SALARY	DEP_ID	CITY
202	Anita	45000	102	-
201	Amit	55000	10	Delhi
201	Rahul	66550	101	-
202	Priya	48000	10	Delhi
203	Rohit	60000	20	Mumbai
205	Karan	70000	30	Bangalore
204	Neha	45000	20	Mumbai

20. Subquery with NOT EXISTS - Fetch the names and ages of customers who have NOT placed any orders based on the data from the "Orders" table.

The screenshot shows a SQL query editor interface. The top bar includes 'Language' set to 'SQL', 'Rows' set to '10', and buttons for 'Save' and 'Run'. Below the editor area, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table has two columns: 'CUST_NAME' and 'AGE'. A single row is displayed with the name 'Charlie' and age '28'. At the bottom left, it says '1 rows returned in 0.02 seconds' and there is a 'Download' link.

```
1 SELECT CUST_NAME, AGE
2 FROM CUSTOMERS
3 WHERE NOT EXISTS (
4     SELECT 1
5     FROM ORDERS
6     WHERE CUS_ID = CUST_ID
7 );
8
```

CUST_NAME	AGE
Charlie	28

1 rows returned in 0.02 seconds [Download](#)



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Program Name: B. Tech CS

Semester: III

Name: Yashasvani Markanday

SAP ID: 590011386

Batch: B2

Subject: Database and Management System

EXPERIMENT-4

Lab 4: To implement and use Inbuilt Functions Objective: To understand the use of SQL Inbuilt functions.

The screenshot shows the Oracle SQL Workshop interface. A SQL command is being entered in the editor:

```
1. desc employee;
```

The results pane displays the structure of the 'EMPLOYEE' table:

Name	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	EMP_NAME	VARCHAR2	5	-	-	-	-	-	-
	OCCUPATION	VARCHAR2	10	-	-	-	-	-	-
	WORKING_DATE	DATE	7	-	-	-	✓	>	-
	WORKING_HOURS	VARCHAR2	5	-	-	-	✓	>	-

The screenshot shows the Oracle SQL Workshop interface. A SQL command is being entered in the editor:

```
1. select * from employee;
```

The results pane displays the data from the 'EMPLOYEE' table:

EMP_NAME	OCCUPATION	WORKING_DATE	WORKING_HOURS
Peter	Actor	10/6/2020	11
Aishwarya	Business	10/6/2020	11
Warren	Engineer	10/6/2020	10
Marc	Doctor	10/6/2020	14
Wingking	Teacher	10/6/2020	12
India	Scientist	10/6/2020	13

1. SELECT COUNT(name) FROM employee;

A screenshot of the Oracle APEX SQL Workshop interface. The SQL command window contains the following SQL statement:

```
SELECT COUNT(emp_name) FROM employee;
```

The results section shows a single row with the column name "COUNT(EMP_NAME)" and a value of 0. The status bar at the bottom indicates "0 rows returned in 0.00 seconds".

2. SELECT SUM(working_hours) AS "Total working hours" FROM employee;

A screenshot of the Oracle APEX SQL Workshop interface. The SQL command window contains the following SQL statement:

```
SELECT SUM(working_hours) AS "Total working hours" FROM employee;
```

The results section shows a single row with the column name "Total working hours" and a value of 0. The status bar at the bottom indicates "0 rows returned in 0.01 seconds".

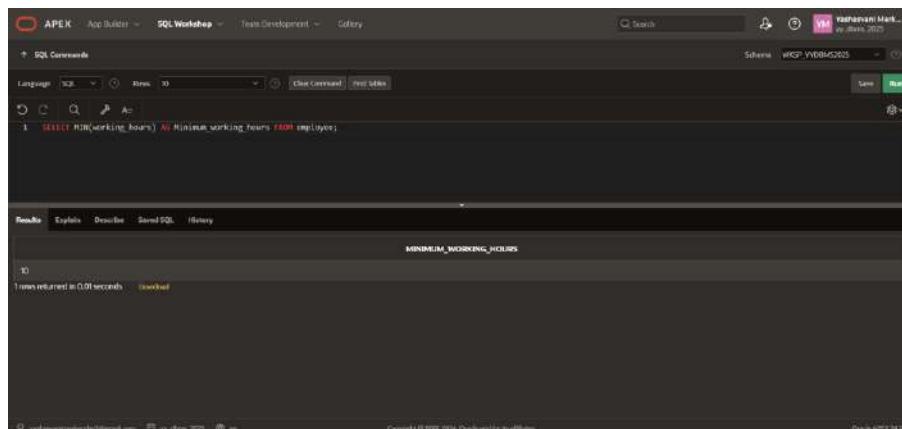
3. SELECT AVG(working_hours) AS "Average working hours" FROM employee;

A screenshot of the Oracle APEX SQL Workshop interface. The SQL command window contains the following SQL statement:

```
SELECT AVG(working_hours) AS "Average working hours" FROM employee;
```

The results section shows a single row with the column name "Average working hours" and a value of 0. The status bar at the bottom indicates "0 rows returned in 0.01 seconds".

4. SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

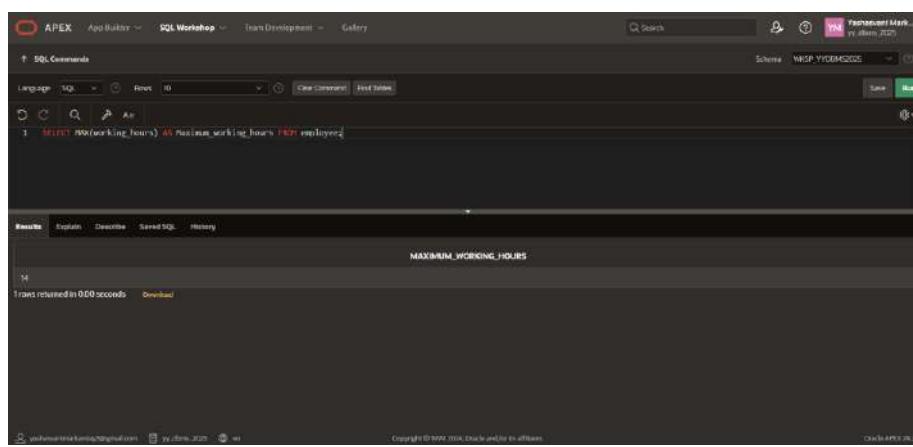
```
1 SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;
```

The results section displays the output:

MINIMUM_WORKING_HOURS
10

1 row returned in 0.01 seconds. Download

5. SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

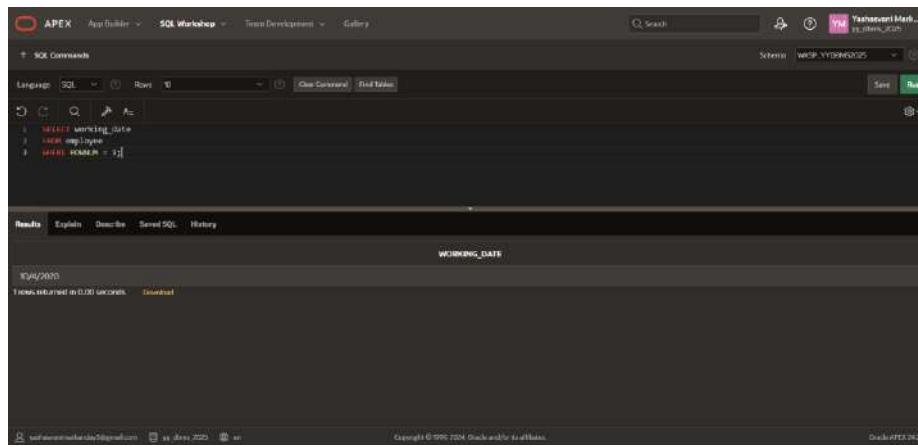
```
1 SELECT MAX(working_hours) AS maximum_working_hours FROM employee;
```

The results section displays the output:

MAXIMUM_WORKING_HOURS
14

1 row returned in 0.00 seconds. Download

6. SELECT working_date FROM employee LIMIT 1;



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

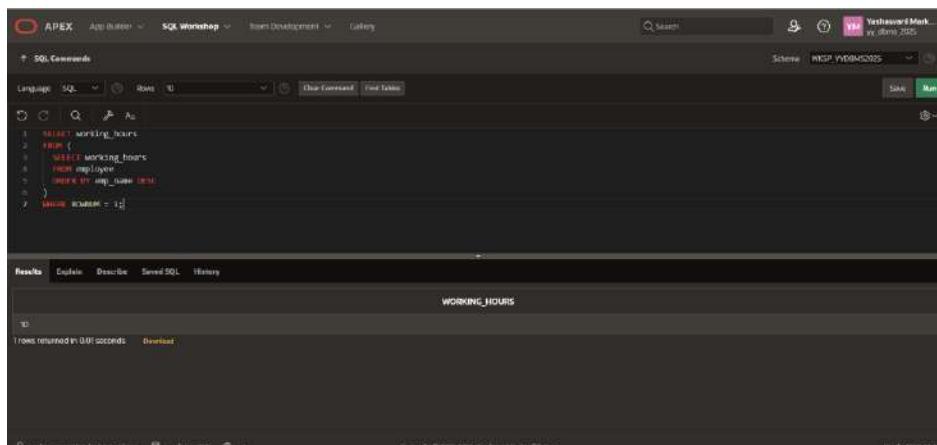
```
1 SELECT working_date
2 FROM employee
3 WHERE ROWNUM = 1;
```

The results section displays the output:

WORKING_DATE
10/10/2025

1 row returned in 0.00 seconds. Download

7. SELECT working_hours FROM employee ORDER BY name DESC LIMIT 1;



```
APEx SQL Workshop Team Development Gallery
```

SQL Commands

```
Language: SQL Rows: 10 Clear Command Find Tables
```

```
1 SELECT working_hours
2   FROM (
3     SELECT working_hours
4       FROM employee
5      ORDER BY emp_name DESC
6    )
7  WHERE ROWNUM = 1;
```

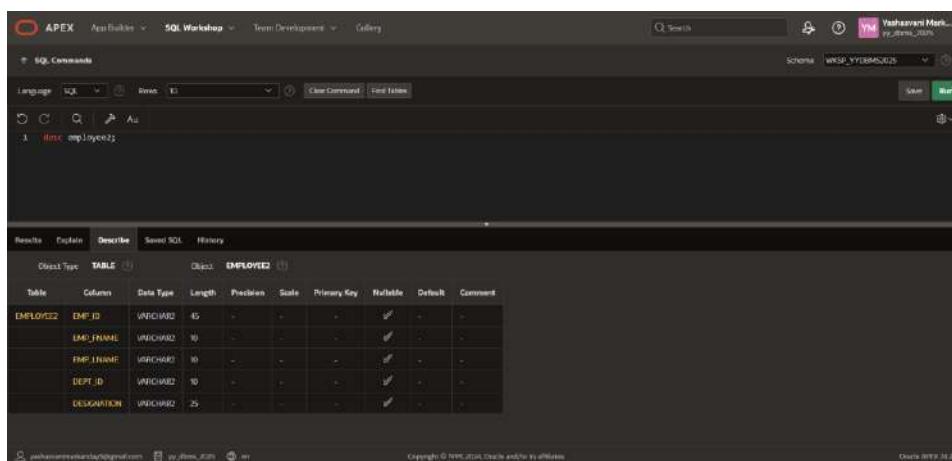
Results Explain Describe Saved SQL History

WORKING_HOURS

10

1 rows returned in 0.01 seconds Download

Copyright © 1996-2025, Oracle and/or its affiliates. Oracle Database 19c



```
APEx SQL Workshop Team Development Gallery
```

SQL Commands

```
Language: SQL Rows: 10 Clear Command Find Tables
```

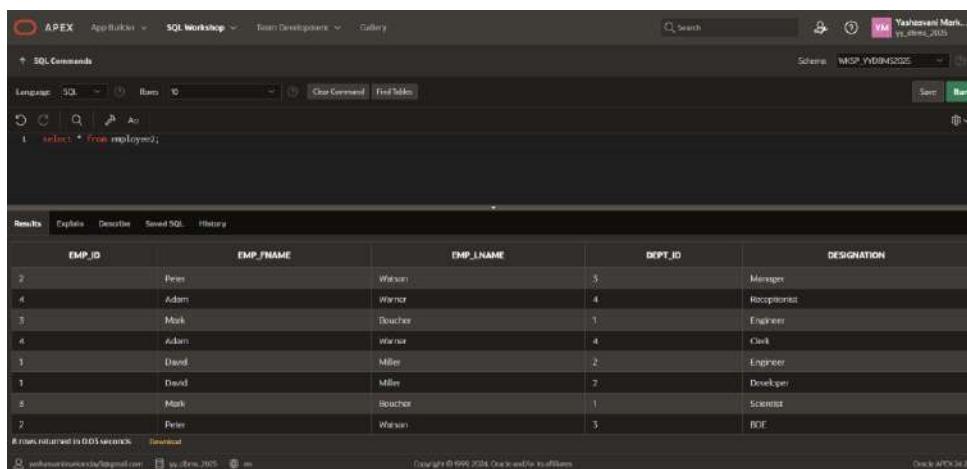
```
1 create table employee2;
```

Results Explain Describe Saved SQL History

Object Type: TABLE Object: EMPLOYEE2

Table	Columns	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE2	EMP_ID	VARCHAR2	45	-	-	-	✓	-	-
EMP_FNAME	VARCHAR2	10	-	-	-	-	✓	-	-
EMP_LNAME	VARCHAR2	10	-	-	-	-	✓	-	-
DEPT_ID	VARCHAR2	10	-	-	-	-	✓	-	-
DESIGNATION	VARCHAR2	25	-	-	-	-	✓	-	-

Copyright © 1996-2025, Oracle and/or its affiliates. Oracle Database 19c



```
APEx SQL Workshop Team Development Gallery
```

SQL Commands

```
Language: SQL Rows: 10 Clear Command Find Tables
```

```
1 select * from employee2;
```

Results Explain Describe Saved SQL History

EMP_ID	EMP_FNAME	EMP_LNAME	DEPT_ID	DESIGNATION
2	Peter	Weller	5	Manager
4	Adam	Warner	4	Receptionist
3	Mark	Boucher	3	Engineer
6	Adam	Warner	4	Civil
5	David	Miller	2	Engineer
1	David	Miller	2	Developer
8	Mark	Boucher	1	Scientist
7	Peter	Weller	5	BOE

8 rows returned in 0.03 seconds Download

Copyright © 1996-2025, Oracle and/or its affiliates. Oracle Database 19c

8. SELECT emp_id, emp_fname, emp_lname, dept_id, GROUP_CONCATdesignation) AS designation
FROM employee2 GROUP BY emp_id, emp_fname, emp_lname, dept_id;

```

1  SELECT emp_id, emp_fname, emp_lname, dept_id,
2    LISTAGG(designation, ',') WITHIN GROUP (ORDER BY designation) AS designation
3  FROM employee2
4  GROUP BY emp_id, emp_fname, emp_lname, dept_id;

```

The screenshot shows the Oracle APEX SQL Workshop interface. The query has been run and returned 4 rows. The results are displayed in a table:

EMP_ID	EMP_FNAME	EMP_LNAME	DEPT_ID	DESIGNATION
1	David	Miller	2	DeveloperEngineer
2	Peter	Webster	3	DBManager
3	Mark	Boucher	1	EngineerScientist
4	Adam	Werner	4	ClerkReceptionist

9. SELECT emp_fname, dept_id, GROUP_CONCAT(DISTINCT designation) as designation FROM employee2 group by emp_id, emp_fname, emp_lname, dept_id;

```

1  SELECT emp_fname, dept_id,
2    LISTAGG(designation, ',') WITHIN GROUP (ORDER BY designation) AS designation
3  FROM (
4    SELECT DISTINCT emp_id, emp_fname, emp_lname, dept_id, designation
5    FROM employee
6  ) t
7  GROUP BY emp_id, emp_fname, emp_lname, dept_id;

```

The screenshot shows the Oracle APEX SQL Workshop interface. The query has been run and returned 4 rows. The results are displayed in a table:

EMP_FNAME	DEPT_ID	DESIGNATION
David	2	DeveloperEngineer
Peter	3	DBManager
Mark	1	EngineerScientist
Adam	4	ClerkReceptionist

10. SELECT emp_fname, GROUP_CONCAT(DISTINCT designation SEPARATOR ';') as designation FROM employee2 group by emp_id, emp_fname, emp_lname, dept_id;

```

1  SELECT emp_fname,
2    LISTAGG(designation, ';') WITHIN GROUP (ORDER BY designation) AS designation
3  FROM (
4    SELECT DISTINCT emp_id, emp_fname, emp_lname, dept_id, designation
5    FROM employee
6  ) t
7  GROUP BY emp_id, emp_fname, emp_lname, dept_id;

```

The screenshot shows the Oracle APEX SQL Workshop interface. The query has been run and returned 4 rows. The results are displayed in a table:

EMP_FNAME	DESIGNATION
David	DeveloperEngineer
Peter	DBManager
Mark	EngineerScientist
Adam	ClerkReceptionist

11. SELECT GROUP_CONCAT(CONCAT_WS(',', emp_lname, emp_fname) SEPARATOR ';') as employeeName FROM employee2;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT LISTAGG(emp_lname || ',' || emp_fname, ',') 
2   WITHIN GROUP (ORDER BY emp_lname, emp_fname) AS employeeName
3 FROM employee2
```

The results section displays the output:

EMPLOYEENAME
oucher,Marcoucher,MarkMiller,DavidMiles,DavidWarner,AdamWarner,AdamWatson,PeterWatson,Peter

Rows returned in 0.00 seconds.

MySQL String Function

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following command:

```
1 desc sample_table;
```

The results section displays the table structure:

Object Type	TABLE	Object ID	SAMPLE_TABLE						
Table	Columns	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SAMPLE_TABLE	S_ID	NUMBER	72	-	0	-	✓	-	-
	FIRST_NAME	VARCHAR2	90	-	-	-	✓	-	-
	LAST_NAME	VARCHAR2	90	-	-	-	✓	-	-

12. SELECT id FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows the user 'Yashasvani Mark...' and the schema 'WWSPL_YYDBMS2025'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'SELECT id FROM sample_table;'. The Results tab displays the output:

S_ID
1
2
3

Below the table, it says '3 rows returned in 0.01 seconds'.

13. SELECT first_name FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows the user 'Yashasvani Mark...' and the schema 'WWSPL_YYDBMS2025'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'SELECT first_name FROM sample_table;'. The Results tab displays the output:

FIRST_NAME
John
Jane
Alice

Below the table, it says '3 rows returned in 0.01 seconds'.

14. SELECT last_name FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows the user 'Yashasvani Mark...' and the schema 'WWSPL_YYDBMS2025'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'SELECT last_name FROM sample_table;'. The Results tab displays the output:

LAST_NAME
Doe
Smith
Johnson

Below the table, it says '3 rows returned in 0.00 seconds'.

15. SELECT CONCAT_WS('-', first_name, last_name) AS concat_ws_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

```
1 SELECT first_name || '-' || last_name AS concat_ws_result
2 FROM sample_table;
```

The results section displays the output:

concat_ws_result
John Doe
Jane Smith
Alice Johnson

Below the results, it says "3 rows returned in 0.01 seconds".

16. SELECT CONCAT(first_name, ' ', last_name) AS concat_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

```
1 SELECT first_name || ' ' || last_name AS concat_result
2 FROM sample_table;
```

The results section displays the output:

concat_result
John Doe
Jane Smith
Alice Johnson

Below the results, it says "3 rows returned in 0.00 seconds".

17. SELECT CHARACTER_LENGTH(first_name) AS char_length_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

```
1 SELECT LENGTH(first_name) AS char_length_result
2 FROM sample_table;
```

The results section displays the output:

char_length_result
4
4
5

Below the results, it says "3 rows returned in 0.01 seconds".

18. SELECT ELT(1, first_name, last_name, 'Other') AS elt_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT first_name AS elt_result
2 FROM sample_table;
```

The results show the output of the ELT function:

ELT_RESULT
John
Jane
Alice

3 rows returned in 0.00 seconds

19. SELECT EXPORT_SET(id, 1, ',', '') AS export_set_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT CASE
2     WHEN s.id = 1 THEN '1'
3     ELSE ''
4     END AS export_set_result
5 FROM sample_table;
```

The results show the output of the EXPORT_SET function:

EXPORT_SET_RESULT
1

3 rows returned in 0.01 seconds

20. SELECT FIELD(first_name, 'Alice', 'Jane', 'John') AS field_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT CASE first_name
2     WHEN 'Alice' THEN 1
3     WHEN 'Jane' THEN 2
4     WHEN 'John' THEN 3
5     ELSE 0
6     END AS field_result
7 FROM sample_table;
```

The results show the output of the FIELD function:

FIELD_RESULT
3
2
1

3 rows returned in 0.00 seconds

21. SELECT FIND_IN_SET(first_name, 'John,Jane,Alice') AS find_in_set_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT CASE
2     WHEN INSTR('John,Jane,Alice', first_name) > 0
3     THEN INSTR('John,Jane,Alice', first_name)
4     ELSE 0
5 END AS find_in_set_result
6 FROM sample_table;
```

The results show three rows with values 1, 6, and 11, corresponding to the positions of 'John', 'Jane', and 'Alice' respectively.

22. SELECT FORMAT(id, 2) AS format_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT TO_CHAR(s.id, '9999.99') AS format_result
2 FROM sample_table;
```

The results show three rows with values 1.00, 2.00, and 3.00, formatted with two decimal places.

23. SELECT FROM_BASE64('SGVsbG8gd29ybGQh') AS from_base64_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT UTL_RAW.CAST_TO_VARCHAR2(
2     UTL_ENCODE.BASE64_DECODE(UTL_RAW.CAST_TO_RAW('SGVsbG8gd29ybGQh'))
3 ) AS from_base64_result
4 FROM dual;
```

The results show one row with the value 'Hello world!', which is the decoded base64 string.

24. SELECT HEX(id) AS hex_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
3: SELECT RAWTOHEX(UTL_RAW.CAST_TO_RAW(s_id)) AS hex_result
2: FROM sample_table;
```

The results are displayed in a table with one column labeled "HEX_RESULT". The values are:

HEX_RESULT
C00
C05
C04

5 rows returned in 0.01 seconds. The schema is WIS_P_YYDBMS2025.

25. SELECT INSERT(first_name, 2, 0, 'X') AS insert_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
3: SELECT SUBSTR(first_name, 1, 1) || 'X' || substr(first_name, 2)
2: AS insert_result
1: FROM sample_table;
```

The results are displayed in a table with one column labeled "INSERT_RESULT". The values are:

INSERT_RESULT
JXhn
JXane
AJXice

5 rows returned in 0.01 seconds. The schema is WIS_P_YYDBMS2025.

26. SELECT INSTR(first_name, 'hn') AS instr_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1: SELECT INSTR(first_name, 'hn') AS instr_result
2: FROM sample_table;
```

The results are displayed in a table with one column labeled "INSTR_RESULT". The values are:

INSTR_RESULT
5
0
0

3 rows returned in 0.01 seconds. The schema is WIS_P_YYDBMS2025.

27. SELECT LCASE(first_name) AS lcse_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

```
1 SELECT LOWER(first_name) AS lcse_result
2 FROM sample_table;
```

The results section displays the output:

LCSE_RESULT
john
jane
alice

5 rows returned in 0.00 seconds.

28. SELECT LEFT(first_name, 2) AS left_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

```
1 SELECT SUBSTR(first_name, 1, 2) AS left_result
2 FROM sample_table;
```

The results section displays the output:

LEFT_RESULT
jo
ja
al

3 rows returned in 0.01 seconds.

29. SELECT LENGTH(first_name) AS length_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

```
1 SELECT LENGTH(first_name) AS length_result
2 FROM sample_table;
```

The results section displays the output:

LENGTH_RESULT
4
4
5

3 rows returned in 0.01 seconds.

30. SELECT first_name LIKE 'J%' AS like_result FROM sample_table;

The screenshot shows the Oracle SQL Workshop interface. The SQL command is:

```
1 SELECT CASE WHEN first_name LIKE 'J%' THEN 1 ELSE 0 END AS like_result
2 FROM sample_table;
```

The results are displayed in a table named "LIKE_RESULT" with three rows:

LIKE_RESULT
1
1
0

5 rows returned in 0.00 seconds. The URL is https://yashasvinimarkanday@gmail.com:4848/oracle/apex/yy_dbms_2025.

31. SELECT LOAD_FILE('/path/to/file') AS load_file_result FROM sample_table;

The screenshot shows the Oracle SQL Workshop interface. The SQL command is:

```
1 SELECT LOAD_FILE is not supported in Oracle; use BFILE/DBMS_LOB or APEX file upload
2 AS load_file_result
3 FROM dual;
```

The results are displayed in a table named "LOAD_FILE_RESULT" with one row:

LOAD_FILE_RESULT
LOAD_FILE is not supported in Oracle; use BFILE/DBMS_LOB or APEX file upload

1 rows returned in 0.00 seconds. The URL is https://yashasvinimarkanday@gmail.com:4848/oracle/apex/yy_dbms_2025.

32. SELECT LOCATE('a', first_name) AS locate_result FROM sample_table;

The screenshot shows the Oracle SQL Workshop interface. The SQL command is:

```
1 SELECT INSTR(first_name, 'a') AS locate_result
2 FROM sample_table;
```

The results are displayed in a table named "LOCATE_RESULT" with three rows:

LOCATE_RESULT
0
2
0

3 rows returned in 0.00 seconds. The URL is https://yashasvinimarkanday@gmail.com:4848/oracle/apex/yy_dbms_2025.

33. SELECT LOWER(first_name) AS lower_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT LOWER(first_name) AS lower_result
2 FROM sample_table;
```

The results section displays the output:

lower_result
john
jane
Alice

Below the results, it says "3 rows returned in 0.01 seconds".

34. SELECT LPAD(first_name, 10, '*') AS lpad_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT LPAD(first_name, 10, '*') AS lpad_result
2 FROM sample_table;
```

The results section displays the output:

lpad_result
*****John
*****Jane
*****Alice

Below the results, it says "3 rows returned in 0.01 seconds".

35. SELECT LTRIM(' ' || first_name) AS ltrim_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT LTRIM(' ' || first_name) AS ltrim_result
2 FROM sample_table;
```

The results section displays the output:

ltrim_result
John
Jane
Alice

Below the results, it says "3 rows returned in 0.01 seconds".

36. SELECT MAKE_SET(2, 'Red', 'Green', 'Blue') AS make_set_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1  SELECT CASE s_id
2    WHEN 1 THEN 'Red'
3    WHEN 2 THEN 'Green'
4    WHEN 4 THEN 'Blue'
5    WHEN 3 THEN 'Red,Green'
6    WHEN 5 THEN 'Red,Blue'
7    WHEN 6 THEN 'Green,Blue'
8    WHEN 7 THEN 'Red,Green,Blue'
9    ELSE NULL END AS make_set_result
10   FROM sample_table;
```

The results section shows the output:

make_set_result
Red
Green
Red,Green

3 rows returned in 0.01 seconds. The Oracle APEX version shown is 21.2.0.

37. SELECT MID(first_name, 2, 2) AS mid_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1  SELECT SUBSTR(first_name, 2, 2) AS mid_result
2   FROM sample_table;
```

The results section shows the output:

mid_result
oh
en
il

3 rows returned in 0.01 seconds. The Oracle APEX version shown is 21.2.0.

38. SELECT OCTET_LENGTH(first_name) AS octet_length_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The query in the editor is:

```
1 SELECT LENGTH(first_name) AS octet_length_result
2 FROM sample_table;
```

The results section shows the output:

OCTET_LENGTH_RESULT
4
4
5

3 rows returned in 0.01 seconds.

39. SELECT OCT(id) AS oct_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The query in the editor is:

```
1 SELECT TO_CHAR(TRUNC(s_id/8)) ||
2      TO_CHAR(TRUNC(MOD(s_id,8)/8)) ||
3      TO_CHAR(MOD(s_id,8)) AS oct_result
4 FROM sample_table;
```

The results section shows the output:

OCT_RESULT
001
002
005

3 rows returned in 0.01 seconds.

40. SELECT ORD(LEFT(first_name, 1)) AS ord_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The query in the editor is:

```
1 SELECT ASCII(SUBSTR(first_name, 1, 1)) AS ord_result
2 FROM sample_table;
```

The results section shows the output:

ORD_RESULT
74
74
65

3 rows returned in 0.01 seconds.

41. SELECT POSITION('oh' IN first_name) AS position_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT INSTR(first_name, 'oh') AS position_result
2 FROM sample_table;
```

The results show the following output:

POSITION_RESULT
2
0
0

3 rows returned in 0.01 seconds

42. SELECT QUOTE(first_name) AS quote_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT '''' || first_name || '''' AS quote_result
2 FROM sample_table;
```

The results show the following output:

QUOTE_RESULT
'John'
'Jew'
'Alice'

3 rows returned in 0.00 seconds

43. SELECT REPEAT(first_name, 2) AS repeat_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT RPAD(first_name, LENGTH(first_name) * 2, first_name) AS repeat_result
2 FROM sample_table;
```

The results section displays the output:

REPEAT_RESULT
JohnJohn
JaneJane
AliceAlice

Below the results, it says "5 rows returned in 0.00 seconds". The URL in the address bar is <https://yashavani-mark-dbms205.csail.mit.edu:8080/apex/f?p=100:1:::1234567890:::> and the page title is "Oracle APEX 24.2.0".

44. SELECT REPLACE(first_name, 'Jo', 'X') AS replace_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT REPLACE(first_name, 'Jo', 'X') AS replace_result
2 FROM sample_table;
```

The results section displays the output:

REPLACE_RESULT
Xm
Jne
Alic

Below the results, it says "5 rows returned in 0.02 seconds". The URL in the address bar is <https://yashavani-mark-dbms205.csail.mit.edu:8080/apex/f?p=100:1:::1234567890:::> and the page title is "Oracle APEX 24.2.0".

45. SELECT REVERSE(first_name) AS reverse_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT LISTAGG(SUBSTR(first_name, LEVEL, 1))
2   BULK GROUP (ORDER BY LEVEL DESC) AS reverse_result
3 FROM sample_table
4 CONNECT BY LEVEL <= LENGTH(first_name)
5 AND PRIOR first_name = First_name
6 AND PRIOR DBMS_RANDOM.VALUE IS NOT NULL
7 GROUP BY first_name;
```

The results section displays the output:

REVERSE_RESULT
edA
ensL
nbaJ

Below the results, it says "5 rows returned in 0.01 seconds". The URL in the address bar is <https://yashavani-mark-dbms205.csail.mit.edu:8080/apex/f?p=100:1:::1234567890:::> and the page title is "Oracle APEX 24.2.0".

46. SELECT RIGHT(last_name, 2) AS right_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT SUBSTR(last_name, -2) AS right_result
2 FROM sample_table;
```

The results section shows the output:

RIGHT_RESULT
de
th
on

Below the results, it says "3 rows returned in 0.00 seconds".

47. SELECT RPAD(first_name, 10, '*') AS rpad_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT RPAD(first_name, 10, '*') AS rpad_result
2 FROM sample_table;
```

The results section shows the output:

RPAD_RESULT
John*****
Jane*****
Alex*****

Below the results, it says "3 rows returned in 0.00 seconds".

48. SELECT RTRIM(first_name) AS rtrim_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT RTRIM(first_name) AS rtrim_result
2 FROM sample_table;
```

The results section displays the output:

RTRIM_RESULT
John
Jane
Alice

Below the table, it says "3 rows returned in 0.00 seconds".

49. SELECT SOUNDEX(first_name) AS soundex_result FROM sample_table;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

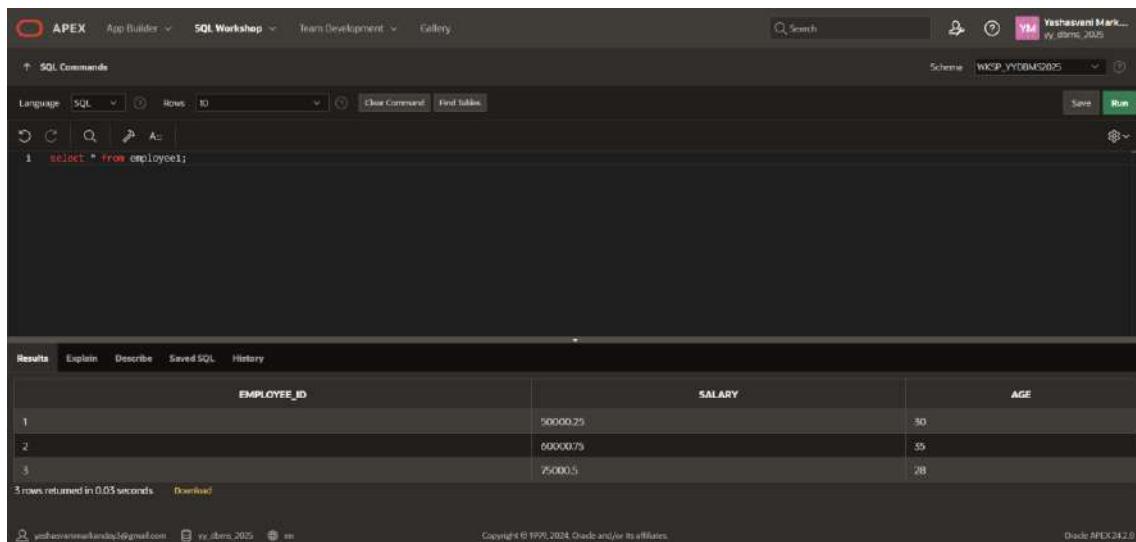
```
1 SELECT SOUNDEX(first_name) AS soundex_result
2 FROM sample_table;
```

The results section displays the output:

SOUNDEX_RESULT
J500
J500
A620

Below the table, it says "3 rows returned in 0.00 seconds".

Mathematical Functions

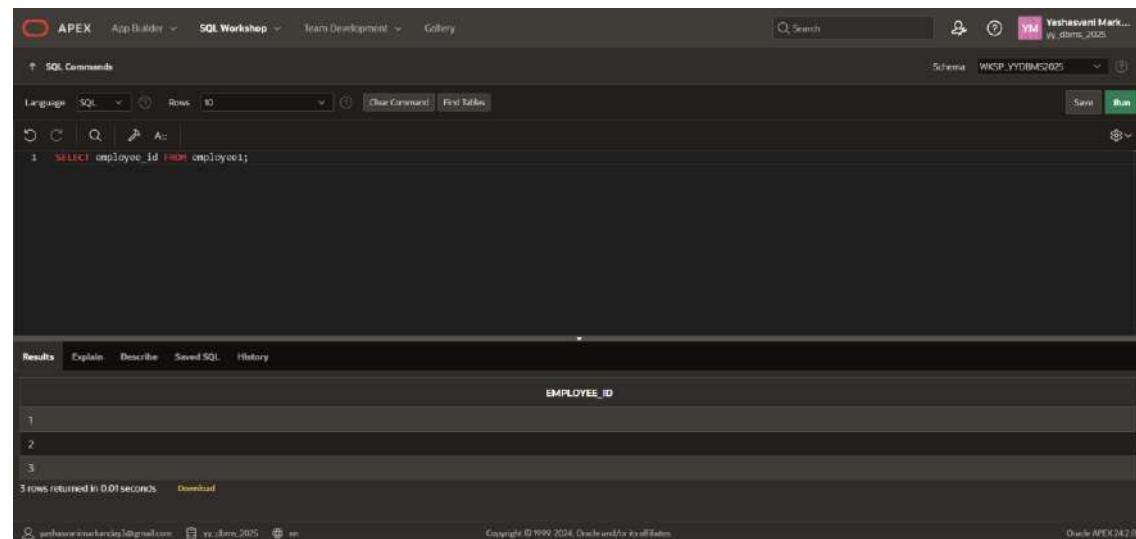


The screenshot shows the Oracle APEX SQL Workshop interface. The command window contains the SQL query: `select * from employee;`. The results window displays three rows of data from the employee table:

EMPLOYEE_ID	SALARY	AGE
1	50000.25	50
2	60000.75	35
3	75000.5	28

3 rows returned in 0.03 seconds.

50. SELECT employee_id FROM employee;

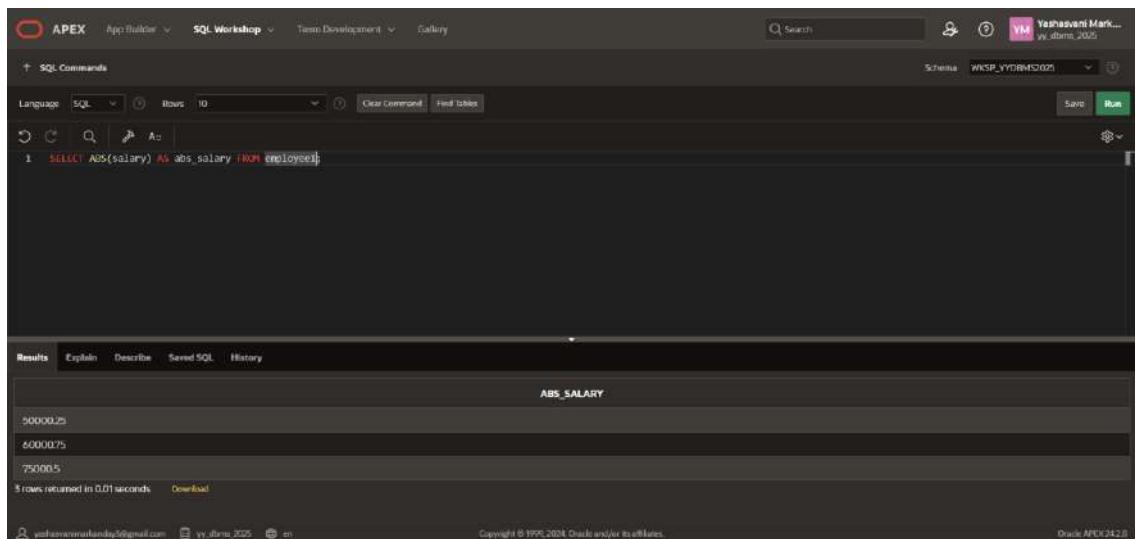


The screenshot shows the Oracle APEX SQL Workshop interface. The command window contains the SQL query: `SELECT employee_id FROM employee;`. The results window displays three rows of data from the employee table:

EMPLOYEE_ID
1
2
3

3 rows returned in 0.01 seconds.

51. SELECT ABS(salary) AS abs_salary FROM employee;



The screenshot shows the Oracle APEX SQL Workshop interface. The command window contains the SQL query: `SELECT ABS(salary) AS abs_salary FROM employee;`. The results window displays three rows of data from the employee table:

ABS_SALARY
50000.25
60000.75
75000.5

3 rows returned in 0.01 seconds.

52. SELECT ACOS(salary / 100000) AS acos_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The command window contains the following SQL code:

```
1 SELECT ACOS(salary / 100000) AS acos_salary FROM employee;
```

The results section displays the output of the query:

ACOS_SALARY
1.04279465444084635849913478026273505
.92728584296952878777342565896858825122
.7772266584713821185678591009949191

3 rows returned in 0.01 seconds. [Download](#)

53. SELECT SIGN(salary) AS sign_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The command window contains the following SQL code:

```
1 SELECT SIGN(salary) AS sign_salary FROM employee;
```

The results section displays the output of the query:

SIGN_SALARY
1
1
1

3 rows returned in 0.01 seconds. [Download](#)

54. SELECT SIN(salary / 100000) AS sin_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The command window contains the following SQL code:

```
1 SELECT SIN(salary / 100000) AS sin_salary FROM employee;
```

The results section displays the output of the query:

SIN_SALARY
.479427752591095911156661936546674919
.5646486559626052192605400577415762591
.6814418459139156095383454799827035

3 rows returned in 0.01 seconds. [Download](#)

55. SELECT SQRT(salary) AS sqrt_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Yashaswini Mark...' and the schema 'WKSP_YYDBMS2025'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'SELECT SQRT(salary) AS sqrt_salary FROM employee;'. The Results tab displays the output:

SQRT_SALARY
225.60755676627457507229545376899901589
244.95050504627919529506252572065653634
273.86297162199077658192256024125719

3 rows returned in 0.00 seconds. The bottom status bar shows the URL 'http://localhost:8080/apex/f?p=100:1:::1:::' and the copyright notice 'Copyright © 1996-2004, Oracle and/or its affiliates'.

56. SELECT salary AS sum_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Yashaswini Mark...' and the schema 'WKSP_YYDBMS2025'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'SELECT salary AS sum_salary FROM employee;'. The Results tab displays the output:

SUM_SALARY
50000.25
60000.75
75000.5

3 rows returned in 0.01 seconds. The bottom status bar shows the URL 'http://localhost:8080/apex/f?p=100:1:::1:::' and the copyright notice 'Copyright © 1996-2004, Oracle and/or its affiliates'.

57. SELECT TAN(salary / 100000) AS tan_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT TAN(salary / 100000) AS tan_salary FROM employee;
```

The results show three rows of data:

TAN_SALARY
-5.63057559e-24e-09
-6.81478957299105146525414900661565
-9.31007994795260045945793540210521

3 rows returned in 0.01 seconds. The URL is <https://ydbms2025.us-east-1.OracleCloudInfrastructure.com>.

58. SELECT TRUNCATE(salary, 2) AS truncated_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT TRUNC(salary, 2) AS truncated_salary FROM employee;
```

The results show three rows of data:

TRUNCATED_SALARY
50000.25
60000.25
70000.5

3 rows returned in 0.01 seconds. The URL is <https://ydbms2025.us-east-1.OracleCloudInfrastructure.com>.

59. SELECT ASIN(salary / 100000) AS asin_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT ASIN(salary / 100000) AS asin_salary FROM employee;
```

The results show three rows of data:

ASIN_SALARY
5.25601623520504554653034587E-01705
-6.4351041382645740275979L5747692666778
-8.8904938303558497126760779625400399

3 rows returned in 0.00 seconds. The URL is <https://ydbms2025.us-east-1.OracleCloudInfrastructure.com>.

60. SELECT ATAN2(salary, age) AS atan2_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is:

```
1 SELECT ATAN2(salary, age) AS atan2_salary FROM employee;
```

The results section displays the output:

ATAN2_SALARY
1.5701965208668052370551401975499885907
1.570215000993014566894542119547340252
1.5704229959677800240969425901090308915

Copyright © 1996-2024, Oracle and/or its affiliates.

61. SELECT ATAN(salary / 100000) AS atan_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is:

```
1 SELECT ATAN(salary / 100000) AS atan_salary FROM employee;
```

The results section displays the output:

ATAN_SALARY
.466495089988061354795564788947728025
.54042501495821952476145254009205963149
.64550450878500439431276968246446288257

Copyright © 1996-2024, Oracle and/or its affiliates.

62. SELECT salary AS avg_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is:

```
1 SELECT CEIL(salary) AS ceil_salary FROM employee;
```

The results section displays the output:

CEIL_SALARY
50001
60001
75001

Copyright © 1996-2024, Oracle and/or its affiliates.

63. SELECT CEIL(salary) AS ceil_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user profile 'Yashavant Mark...', and schema 'WKSP_YYDBMS2025'. The main area has tabs for 'SQL Commands' and 'Results'. In the 'SQL Commands' tab, the query 'SELECT CEIL(salary) AS ceiling_salary FROM employee;' is entered. The 'Results' tab displays the output:

CEILING_SALARY
50001
60001
75001

Below the results, it says '3 rows returned in 0.01 seconds'.

64. SELECT CEILING(salary) AS ceiling_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user profile 'Yashavant Mark...', and schema 'WKSP_YYDBMS2025'. The main area has tabs for 'SQL Commands' and 'Results'. In the 'SQL Commands' tab, the query 'SELECT CEIL(salary) AS ceiling_salary FROM employee;' is entered. The 'Results' tab displays the output:

CEILING_SALARY
50001
60001
75001

Below the results, it says '3 rows returned in 0.00 seconds'.

65. SELECT COS(salary / 100000) AS cos_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_YYDRMS/COS'. The SQL command entered is:

```
1 SELECT COS(salary / 100000) AS cos_salary FROM employee;
```

The results section displays the output:

COS_SALARY
.87581264332390361535789778451048645
.82553580060707530007927509125936603949
.751681460670071467298098880577427050755

3 rows returned in 0.00 seconds. The Oracle APEX version shown is 24.2.0.

66. SELECT COT(salary / 100000) AS cot_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_YYDRMS/COS'. The SQL command entered is:

```
1 SELECT COT(salary / 100000) / SIN(salary / 100000) AS cot_salary
2 FROM employee;
```

The results section displays the output:

COT_SALARY
1.82046045049977542058856945739414728
1.4646724251314888285968416910083874
1.0734853273886583778144524754638663354

3 rows returned in 0.01 seconds. The Oracle APEX version shown is 24.2.0.

67. SELECT 1 AS count_employee FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT 1 AS count_employee FROM employee;
```

The results show a single row with the value 1.

COUNT_EMPLOYEE
1

3 rows returned in 0.00 seconds

68. SELECT DEGREES(salary / 100000) AS degrees_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT (ACOS(-0.5) * 180 / ACOS(-1)) AS angle_in_degrees
2 FROM dual;
```

The results show a single row with the value 60.

ANGLE_IN_DEGREES
60

1 rows returned in 0.00 seconds

69. SELECT salary DIV 2 AS div_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT TRUNC(salary / 2) AS div_salary
2 FROM employee;
```

The results are displayed in a table named 'DIV_SALARY' with three rows:

DIV_SALARY
25000
30000
35000

3 rows returned in 0.01 seconds

70. SELECT EXP(salary / 100000) AS exp_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT EXP(salary / 100000) AS exp_salary
2 FROM employee;
```

The results are displayed in a table named 'EXP_SALARY' with three rows:

EXP_SALARY
1.7487539208417754335346974857045539
1.8221524663352759125076506789920256541
2.110106016392202825059512307076572284

3 rows returned in 0.01 seconds

71. SELECT FLOOR(salary) AS floor_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

```
1 SELECT FLOOR(salary) AS floor_salary
2 FROM employee;
```

The results section displays the output:

FLOOR_SALARY
50000
60000
75000

3 rows returned in 0.01 seconds

72. SELECT GREATEST(salary, 0, -50000) AS greatest_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is:

```
1 SELECT GREATEST(salary, 0, -50000) AS greatest_salary
2 FROM employee;
```

The results section displays the output:

GREATEST_SALARY
50000.25
60000.75
75000.5

3 rows returned in 0.01 seconds

73. SELECT LEAST(salary, 0, -50000) AS least_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT LEAST(salary, 0, -50000) AS least_salary
2 FROM employee;
```

The results show three rows of data:

LEAST_SALARY
-50000
-50000
-50000

3 rows returned in 0.01 seconds

74. SELECT LN(salary / 100000) AS ln_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is:

```
1 SELECT LN(salary / 100000) AS ln_salary
2 FROM employee;
```

The results show three rows of data:

LN_SALARY
.09514218057214526775072170416651250557
-.50693123944150521699-.089422774749428
.28467540580755658422985626751071449751

3 rows returned in 0.01 seconds

75. SELECT LOG10(salary / 100000) AS log10_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is:

```
1 SELECT LOG10(salary / 100000) AS log10_salary
2 FROM employee;
```

The results section displays the output:

LOG10_SALARY
-3010228241970003488285573253140919931
-2218455209592655050925486153756689758
-174975641321404677047770106397304591

3 rows returned in 0.01 seconds.

76. SELECT LOG(salary / 100000, 2) AS log_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is:

```
1 SELECT LOG(2, salary / 100000) AS log_salary
2 FROM employee;
```

The results section displays the output:

LOG_SALARY
.9999778547629830200729279357699957
-7369475605909016659756025426747938381
-1502788015435643046100057645702474448945

3 rows returned in 0.01 seconds.

77. SELECT LOG2(salary / 100000) AS log2_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains:

```
1 SELECT LOG2(salary / 100000) AS log2_salary
2 FROM employee;
```

The results window shows the output:

LOG2_SALARY
-0.9999927865412529830820072922957699937
-75694726059090466397660264254793.99
-41502788154396450610005784810249448945

5 rows returned in 0.01 seconds. The URL is <https://yashavani-marks-oracle-database-2025.rw.rdbms.us>.

78. SELECT salary AS max_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains:

```
1 SELECT MAX(salary) AS max_salary
2 FROM employee;]
```

The results window shows the output:

MAX_SALARY
75000.5

1 rows returned in 0.01 seconds. The URL is <https://yashavani-marks-oracle-database-2025.rw.rdbms.us>.

79. SELECT salary AS min_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains:

```
1 SELECT MIN(salary) AS min_salary
2 FROM employee;]
```

The results window shows the output:

MIN_SALARY
50000.25

1 rows returned in 0.00 seconds. The URL is <https://yashavani-marks-oracle-database-2025.rw.rdbms.us>.

80. SELECT MOD(salary, 3) AS mod_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT MOD(salary, 3) AS mod_salary
2 FROM employee;
```

The results window displays the output:

MOD_SALARY
225
75
5

Below the table, it says "3 rows returned in 0.01 seconds". The Oracle APEX version shown is 24.2.0.

81. SELECT PI() AS pi_value FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```
1 SELECT ACOS(-1) AS pi_value
2 FROM dual;
```

The results window displays the output:

PI_VALUE
3.1415926535897932384626433832795028842

Below the table, it says "1 rows returned in 0.01 seconds". The Oracle APEX version shown is 24.2.0.

82. SELECT POWER(salary, 2) AS power_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains:

```
1 SELECT POWER(salary, 2) AS power_salary
2 FROM employee;
```

The results section displays the output:

POWER_SALARY
250005000.625
500000000.5625
5625075000.25

3 rows returned in 0.00 seconds

83. SELECT POW(salary, 2) AS pow_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains:

```
1 SELECT POW(salary, 2) AS pow_salary
2 FROM employee;
```

The results section displays the output:

POW_SALARY
250005000.625
500000000.5625
5625075000.25

3 rows returned in 0.01 seconds

84. SELECT RADIANS(salary / 100000) AS radians_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains:

```
1 SELECT (salary / 100000) * (ACOS(-1) / 180) AS radians_salary
2 FROM employee;
```

The results section displays the output:

RADIANS_SALARY
.007266899730202947742857876634712275785
.016472105411659677036260415887759512927
.032900256656420075454053694803019781

3 rows returned in 0.01 seconds

85. SELECT RAND() AS rand_value FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is:

```
1 SELECT DBMS_RANDOM.VALUE AS rand_value
2 FROM dual;
```

The results section displays the output:

RAND_VALUE
.3381477647047357788264389970454574

1 rows returned in 0.01 seconds. [Download](#)

++++++

86. SELECT ROUND(salary, 2) AS round_salary FROM employee;

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is:

```
1 SELECT ROUND(salary, 2) AS round_salary
2 FROM employee;
```

The results section displays the output:

ROUND_SALARY
50000.25
60000.75
75000.5

3 rows returned in 0.00 seconds. [Download](#)

CONCLUSION

In this lab, we explored the use of SQL inbuilt functions for data manipulation and analysis. We practiced aggregate functions like COUNT, SUM, AVG, MIN, and MAX, as well as string functions such as CONCAT, LENGTH, and REPLACE. Mathematical functions including ABS, SQRT, LOG, and POWER were also implemented to understand their role in calculations. Through this experiment, we gained practical knowledge of how inbuilt functions simplify complex queries and enhance database operations.



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Program Name: B. Tech CS

Semester: III

Name: Yashasvani Markanday

SAP ID: 590011386

Batch: B2

Subject: DBMS

LAB - 5

Question 3

- Create the tables for Company database as per ER diagram of Exp 2.

TABLE 1: EMPLOYEE

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'Application', 'SQL Workshop', 'Item Development', and 'Utility'. The main area displays the creation of a table named 'EMPLOYEE'. The table structure is defined as follows:

Name	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	FNAME	VARCHAR2	40	-	-	N	Y	-	-
	OCCUPATION	PIVOT ID	30	-	-	N	Y	-	-
	WORKING_DATE	DATE	7	-	-	N	Y	-	-
	WORKING_HOURS	VARCHAR2	10	-	-	N	Y	-	-

TABLE 2: DEPARTMENT

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'Application', 'SQL Workshop', 'Item Development', and 'Utility'. The main area displays the creation of a table named 'DEPARTMENT'. The table structure is defined as follows:

Name	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	DEPT_ID	NUMBER	22	1	0	N	Y	-	-
	FACULTY_ID	NUMBER	22	-	-	N	Y	-	-
	DEPT_NAME	VARCHAR2	100	-	-	N	Y	-	-

TABLE 3: DEPT_LOCATIONS

The screenshot shows the Oracle SQL Workshop interface with the schema set to 'MGR_PROJECTS'. A table named 'DEPT_LOCATION' is selected. The table has two columns: 'DEPT_ID' (NUMBER) and 'LOCATION_ID' (NUMBER). Both columns have a length of 27 and are nullable.

TABLE 4: PROJECT

The screenshot shows the Oracle SQL Workshop interface with the schema set to 'MGR_PROJECTS'. A table named 'PROJECT' is selected. It has four columns: 'PROJ_ID' (NUMBER), 'NAME' (VARCHAR2(20)), 'LOC_ID' (NUMBER), and 'DRAH' (NUMBER). The 'NAME' column is the primary key.

TABLE 5: WORKS_ON

The screenshot shows the Oracle SQL Workshop interface with the schema set to 'MGR_PROJECTS'. A table named 'WORKS_ON' is selected. It has three columns: 'EMP_ID' (NUMBER), 'PROJ_ID' (NUMBER), and 'HOURS' (NUMBER). The 'PROJ_ID' column is the primary key.

TABLE 6: DEPENDENT

The screenshot shows the Oracle SQL Workshop interface with the schema set to 'MGR_PROJECTS'. A table named 'DEPENDENT' is selected. It has five columns: 'LINK' (CHAR), 'DEP_ID' (NUMBER), 'SDI' (CHAR), 'NAME' (CHAR), and 'RELATIONSHIP' (CHAR). The 'DEP_ID' column is the primary key.

b. Insert the following data into their respective tables of Company database.

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL Run: 10 | Rows: 10 | Clear Command | Find Table

1 SELECT * FROM DEPARTMENT;

Results

DEPT_ID	FACILITY_ID	DEPT_NAME
1	101	MCB
4	104	MBA
5	105	MCA
2	102	Btech
3	103	SE

5 rows returned in 0.05 seconds Download

Copyright © 2024 Oracle Database 22c Release 1 (22.1.0)

Oracle APEX 24.1.0

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL Run: 10 | Rows: 10 | Clear Command | Find Table

1 SELECT * FROM EMPLOYEE;

Results

EMP_NAME	OCCUPATION	WORKING_DATE	INDUSTRIAL_HOURS
Peter	Actor	10/4/2020	25
Ambrose	Business	10/4/2020	30
Wendy	Cooker	10/4/2020	35
Mario	Doctor	10/4/2020	34
Brooklynn	Teacher	10/4/2020	29
Velma	Scientist	10/4/2020	32

6 rows returned in 0.05 seconds Download

Copyright © 2024 Oracle Database 22c Release 1 (22.1.0)

Oracle APEX 24.1.0

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL Run: 10 | Rows: 10 | Clear Command | Find Table

1 SELECT * FROM PRODUCT;

Results

NAME	PHASES	LOCATION	CRM
Product	1	Before	3
ProductY	2	Separated	5
ProductZ	3	Hidden	5
ComputerMonitor	10	Stashed	4
Smartphones	20	Hidden	7
Headphones	30	Stashed	4

6 rows returned in 0.05 seconds Download

Copyright © 2024 Oracle Database 22c Release 1 (22.1.0)

Oracle APEX 24.1.0

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL Run: 10 | Rows: 10 | Clear Command | Find Table

1 SELECT * FROM SERIES_01;

Results

ESN	PMO	HOAHS
123456789	1	52.5
1234567890	2	70
5555555555	3	40
6666666666	1	20
4444444444	2	35
3333333333	2	10
5555555555	3	30
3333444444	10	10
3334444444	20	10
9999999999	50	50

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.05 seconds Download

Copyright © 2024 Oracle Database 22c Release 1 (22.1.0)

Oracle APEX 24.1.0

The screenshot shows the Oracle SQL Workshop interface with the following details:

- SQL Commands** tab is selected.
- Language**: SQL
- Results** tab is selected.
- Query**:

```
SELECT * FROM DEPENDENTS;
```
- Results** table:

DEP_ID	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
3354400001	Alice	F	4/1/1966	Daughter
3354400002	Theodore	M	6/25/1968	Son
3354400003	Jay	M	5/15/1985	Son
3354500001	Alma	M	2/18/1945	Spoouse
3354500002	Michael	M	10/10/1991	Son
3354500003	Aira	F	5/10/1998	Daughter
3354500004	Elizabeth	F	3/15/1992	Son
- Message**: 7 rows returned in 0.02 seconds

Question 4

The screenshot shows the Oracle SQL Workshop interface with the following details:

- SQL Commands** tab is selected.
- Language**: SQL
- Results** tab is selected.
- Query**:

```
CREATE TABLE CLIENT_MASTER (
    CLMNO NUMBER(6) PRIMARY KEY,
    CLNAME VARCHAR2(50) NOT NULL,
    CLTYPE CHAR(1) NOT NULL,
    ADDRESS1 VARCHAR2(50),
    ADDRESS2 VARCHAR2(50),
    CITY VARCHAR2(50),
    STATE VARCHAR2(50),
    ZIPCODE NUMBER(10,2)
);
```
- Message**: 0 rows affected.
- Time**: 0.02 seconds

The screenshot shows the Oracle SQL Workshop interface with the following details:

- SQL Commands** tab is selected.
- Language**: SQL
- Results** tab is selected.
- Query**:

```
CREATE TABLE PEOPLE_MASTER (
    PRENAME NUMBER(6) PRIMARY KEY,
    CLMNO NUMBER(6) NOT NULL,
    DEP_ID NUMBER(6) NOT NULL,
    PNAME VARCHAR2(50) NOT NULL,
    PLNAME VARCHAR2(50) NOT NULL,
    GENDERCHAR CHAR(1) NOT NULL,
    BIRTHDAY DATE NOT NULL,
    DEATHDATE DATE NOT NULL,
    RELATIONSHIP NUMBER(6) NOT NULL,
    OCCUPATION NUMBER(6) NOT NULL
);
```
- Message**: 0 rows affected.
- Time**: 0.02 seconds

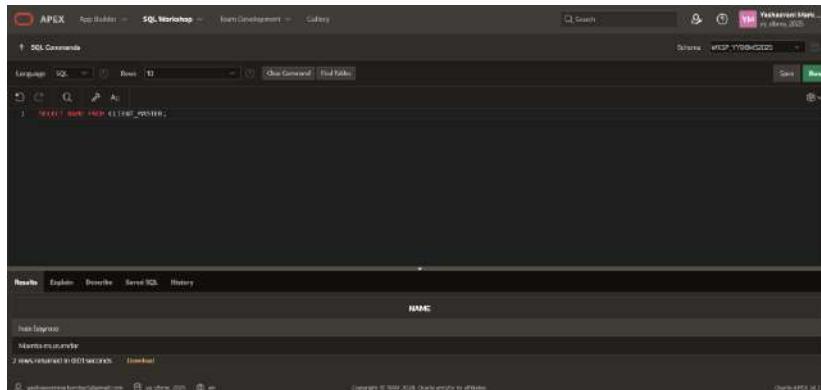
The screenshot shows the Oracle SQL Workshop interface with the following details:

- SQL Commands** tab is selected.
- Language**: SQL
- Results** tab is selected.
- Query**:

```
CREATE TABLE SALARY_MASTER (
    SAL_ID NUMBER(6) PRIMARY KEY,
    SALARYNUMBER NUMBER(10,2) NOT NULL,
    SALMONTH NUMBER(6) NOT NULL,
    AMOUNT NUMBER(10,2) NOT NULL,
    CPT NUMBER(6) NOT NULL,
    PRINCIPAL NUMBER(6) NOT NULL,
    INTEREST NUMBER(6) NOT NULL,
    SALTYPER NUMBER(6) NOT NULL,
    TOTTOTGET NUMBER(10,2) NOT NULL CHECK (TOTTOTGET > 0),
    VISAINFO NUMBER(6) NOT NULL,
    REMARKS VARCHAR2(50)
);
```
- Message**: 0 rows affected.
- Time**: 0.02 seconds

2. Exercise on retrieving records from a table.

a. Find out the names of all the clients:

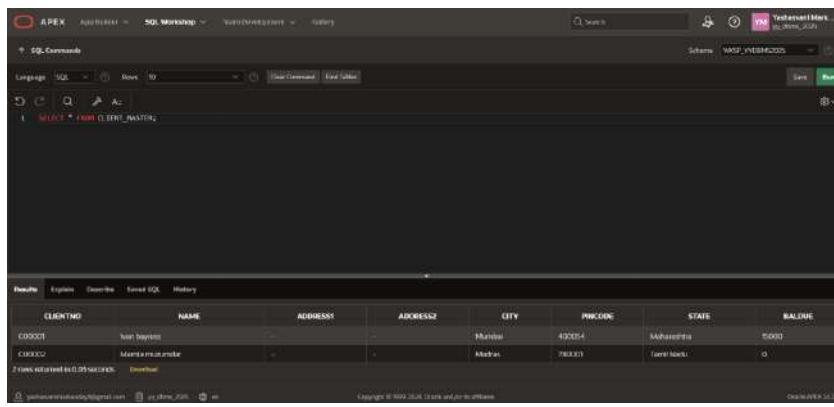


The screenshot shows the Oracle SQL Workshop interface. The SQL command window contains the query: `SELECT NAME FROM CLIENT_MASTER`. The results pane displays two rows of data:

NAME
John Doe
Maria Rodriguez

Below the results, it says "2 rows returned in 0.05 seconds".

b. Retrieve the entire contents of the Client_Master table

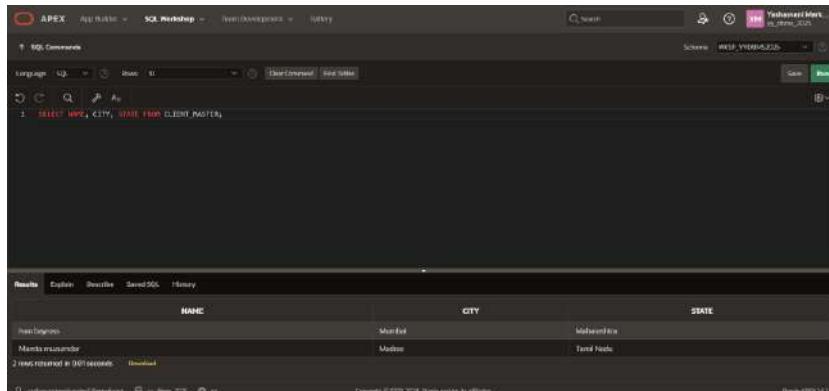


The screenshot shows the Oracle SQL Workshop interface. The SQL command window contains the query: `SELECT * FROM CLIENT_MASTER`. The results pane displays two rows of data:

CLIENTID	NAME	ADDRESS	ADDRESS2	CITY	PCODE	STATE	BALANCE
C0001	John Doe			Mumbai	400054	Maharashtra	5000
C0002	Maria Rodriguez			Madras	780011	Tamil Nadu	0

Below the results, it says "2 rows returned in 0.05 seconds".

c. Retrieve the list of names, city and the state of all the clients.



The screenshot shows the Oracle SQL Workshop interface. The SQL command window contains the query: `SELECT NAME, CITY, STATE FROM CLIENT_MASTER`. The results pane displays two rows of data:

NAME	CITY	STATE
John Doe	Mumbai	Maharashtra
Maria Rodriguez	Madras	Tamil Nadu

Below the results, it says "2 rows returned in 0.05 seconds".

d. List the various products available from the Product_Master table.

A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows 'APEX' > 'Application' > 'SQL Workshop' > 'New Development' > 'Gallery'. The main area is titled 'SQL Commands' with a schema dropdown set to 'SCOTT'. A query editor window contains the following SQL code:

```
SELECT * FROM PRODUCT_MASTER;
```

The results pane below shows a table with one row, labeled 'DESCRIPTION'.

e. List all the clients who are located in Mumbai.

A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows 'APEX' > 'Application' > 'SQL Workshop' > 'New Development' > 'Gallery'. The main area is titled 'SQL Commands' with a schema dropdown set to 'SCOTT'. A query editor window contains the following SQL code:

```
SELECT * FROM CLIENT_MASTER WHERE CITY = 'Mumbai';
```

The results pane below shows a table with one row, labeled 'CLIENTNO', 'NAME', 'ADDRESS', 'ADDRESS2', 'CITY', 'PINCODE', 'STATE', and 'BALANCE'.

f. Find the names of salesman who have a salary equal to Rs.3000.

A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows 'APEX' > 'Application' > 'SQL Workshop' > 'New Development' > 'Gallery'. The main area is titled 'SQL Commands' with a schema dropdown set to 'SCOTT'. A query editor window contains the following SQL code:

```
SELECT SALESMANNAME
FROM SALESMASTERS
WHERE SALARY = 3000;
```

The results pane below shows a table with one row, labeled 'SALESMANNAME'.

3. Exercise on updating records in a table

a. Change the city of ClientNo 'C00005' to 'Bangalore'.

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

```
1 UPDATE CLIENT_MASTER
2 SET CITY = 'Bangalore'
3 WHERE CLIENTNO = 'C00001';
```

The Results pane shows the output of the query:

```
1 rows(s) updated.
0.00 seconds.
```

b. Change the BalDue of ClientNo 'C00001' to Rs.1000.

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

```
1 UPDATE CLIENT_MASTER SET BALDUE = 1000 WHERE CLIENTNO = 'C00001';
```

The Results pane shows the output of the query:

```
1 rows(s) updated.
0.00 seconds.
```

c. Change the cost price of 'Trousers' to rs.950.00.

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

```
1 UPDATE PRODUCT_MASTER SET COSTPRICE = 950 WHERE PRODUCTNO = 'P000001004';
```

The Results pane shows the output of the query:

```
1 rows(s) updated.
0.00 seconds.
```

d. Change the city of the salesman to Pune.

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

```
1 UPDATE SALESMAN_MASTER SET CITY = 'Pune';
```

The Results pane shows the output of the query:

```
1 rows(s) updated.
0.00 seconds.
```

4. Exercise on deleting records in a table

- a. Delete all salesman from the Salesman_Master whose salaries are equal to Rs.3500.

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
DELETE FROM SALESMAN_MASTERS WHERE SALARY = 3500;
```

The results pane shows:

```
2 rows(s) deleted.  
0.02 seconds
```

- b. Delete all products from Product_Master where the quantity on hand is equal to 100.

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
DELETE FROM PRODUCT_MASTER WHERE QOH = 100;
```

The results pane shows:

```
3 row(s) deleted.  
0.02 seconds
```

- c. Delete from Client_Master where the column state holds the value 'Tamil Nadu'.

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
DELETE FROM CLIENT_MASTER WHERE STATE = 'Tamil Nadu';
```

The results pane shows:

```
1 row(s) deleted.  
0.02 seconds
```

5. Exercise on altering the table structure

- a. Add a column called 'Telephone' of data type integer to the Client_Master table.

The screenshot shows the Oracle SQL Workshop interface. The SQL editor contains the following command:

```
1 ALTER TABLE CLIENT_MASTER MODIFY Telephone NUMBER(15);
```

The results pane shows the execution details:

- Table altered.
- 0.07 seconds

At the bottom, the URL is <https://oraclesamplesql.blogspot.com>, the date is 19-Nov-2020, and the page is Grade APEX 14.2.8.

b. Change the size off SellPrice column in Product _ Master to 10, 2.

The screenshot shows the Oracle SQL Workshop interface. The SQL editor contains the following command:

```
1 ALTER TABLE PRODUCT_MASTER MODIFY SellPrice NUMBER(10,2);
```

The results pane shows the execution details:

- Table altered.
- 0.25 seconds

At the bottom, the URL is <https://oraclesamplesql.blogspot.com>, the date is 19-Nov-2020, and the page is Grade APEX 14.2.8.

6. Exercise on deleting the table structure along with the data

a. Destroy the table Client _ Master along with its data.

The screenshot shows the Oracle SQL Workshop interface. The SQL editor contains the following command:

```
1 DROP TABLE CLIENT_MASTER;
```

The results pane shows the execution details:

- Table dropped.
- 0.02 seconds

At the bottom, the URL is <https://oraclesamplesql.blogspot.com>, the date is 19-Nov-2020, and the page is Grade APEX 14.2.8.



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Program Name: B. Tech CS

Semester: III

Name: Yashasvani Markanday

SAP ID: 590011386

Batch: B2

Subject: DAA

Lab-6

```
CREATE TABLE student_p (
    id NUMBER,
    name4 VARCHAR2(50),
    dateofBirth DATE,
    city VARCHAR2(50)
);
```

Table created.
0.04 seconds

```
INSERT ALL
INTO student_p (id, name4, dateofBirth, City)
VALUES (1, 'Neeti Shah', TO_DATE('2010-01-01 12:20:00', 'YYYY-MM-DD HH24:MI:SS'), 'None')
INTO student_p (id, name4, dateofBirth, City)
VALUES (2, 'Neel Agarwal', TO_DATE('2010-01-01 12:20:00', 'YYYY-MM-DD HH24:MI:SS'), 'None')
INTO student_p (id, name4, dateofBirth, City)
VALUES (3, 'Sakshi Kumar', TO_DATE('2010-01-01 12:20:00', 'YYYY-MM-DD HH24:MI:SS'), 'None')
INTO student_p (id, name4, dateofBirth, City)
VALUES (4, 'Sonali Joshi', TO_DATE('2010-01-01 12:20:00', 'YYYY-MM-DD HH24:MI:SS'), 'Shirdi')
INTO student_p (id, name4, dateofBirth, City)
VALUES (5, 'Ayushi Mohankumar', TO_DATE('2010-01-01 12:20:00', 'YYYY-MM-DD HH24:MI:SS'), 'Purat')
SELECT * FROM dual;
```

5 rows(s) inserted.
0.04 seconds

1. SELECT ID FROM student;

The screenshot shows the Oracle SQL Developer interface. The SQL worksheet contains the following SQL statement:

```
SELECT id FROM student;
```

The results pane shows the following output:

ID
1
2
3
4
5

2 SELECT Name FROM student;

The screenshot shows the Oracle SQL Developer interface. The SQL worksheet contains the following SQL statement:

```
SELECT name FROM student;
```

The results pane shows the following output:

NAME
Maria
Teo
Sofia
Isabel

3 SELECT DateTime_Birth FROM student;

The screenshot shows the Oracle SQL Developer interface. The SQL worksheet contains the following SQL statement:

```
SELECT DateTime_Birth FROM student;
```

The results pane shows the following output:

DATETIME_BIRTH
1990-01-01 12:00:00
1990-01-01 12:00:00
1990-01-01 12:00:00
1990-01-01 12:00:00
1990-01-01 12:00:00

4 SELECT CURDATE() AS currentdate;

The screenshot shows the Oracle SQL Developer interface. The SQL worksheet contains the following SQL statement:

```
SELECT CURDATE() AS currentdate;
```

The results pane shows the following output:

CURRENTDATE
2024-01-01 12:00:00

5 SELECT DATE_ADD(DateTime_Birth, INTERVAL 3 DAY) AS date_added FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains:

```
SELECT DATE_ADD(DateTime_Birth, INTERVAL 3 DAY) AS date_added FROM student;
```

The results window shows the output:

DATE_ADDED
2010-09-07
2010-09-08
2010-09-09
2010-09-10
2010-09-11

Copyright ©2014, Oracle. All rights reserved.

6 SELECT DATE_FORMAT(DateTime_Birth, '%Y-%m-%d') AS formatted_date FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains:

```
SELECT DATE_FORMAT(DateTime_Birth, '%Y-%m-%d') AS formatted_date FROM student;
```

The results window shows the output:

FORMATTED_DATE
2010-09-07
2010-09-08
2010-09-09
2010-09-10
2010-09-11

Copyright ©2014, Oracle. All rights reserved.

7 SELECT DATEDIFF(CURDATE(), DATE(DateTime_Birth)) AS date_diff FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains:

```
SELECT DATEDIFF(CURDATE(), DATE(DateTime_Birth)) AS date_diff FROM student;
```

The results window shows the output:

DATE_DIFF
509
507
508
506
505

Copyright ©2014, Oracle. All rights reserved.

8 SELECT DAY(DateTime_Birth) AS day_value FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains:

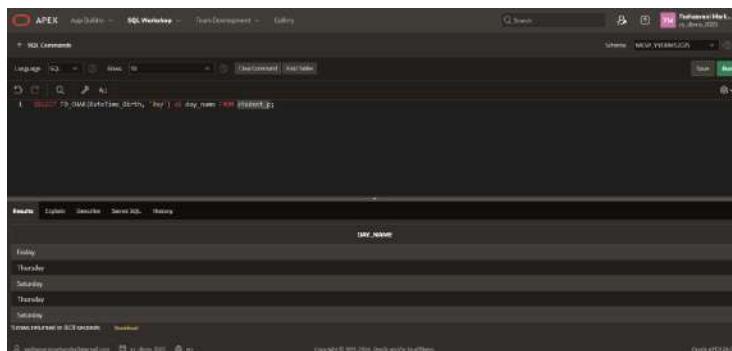
```
SELECT DAY(DateTime_Birth) AS day_value FROM student;
```

The results window shows the output:

DAY_VALUE
0
04
03
09
10

Copyright ©2014, Oracle. All rights reserved.

9 SELECT DAYNAME(DateTime_Birth) AS day_name FROM student;



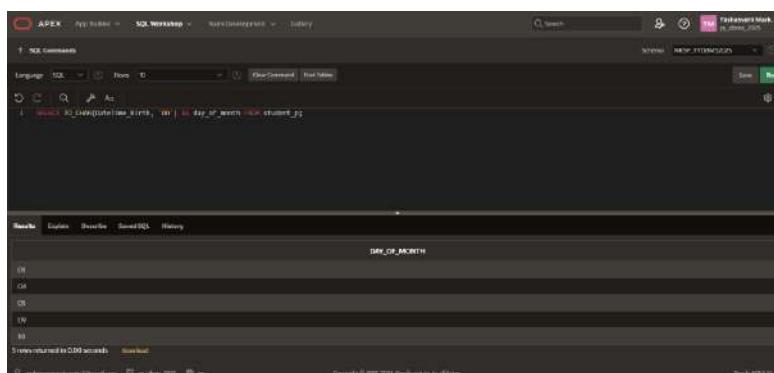
The screenshot shows the Oracle SQL Workshop interface. The query window contains the command:

```
SELECT TO_CHAR(DateTime_Birth, 'day') AS day_name FROM student;
```

The results window displays the output:

DAY_NAME
Friday
Saturday
Sunday
Sunday
Sunday

10 SELECT DAYOFMONTH(DateTime_Birth) AS day_of_month FROM student;



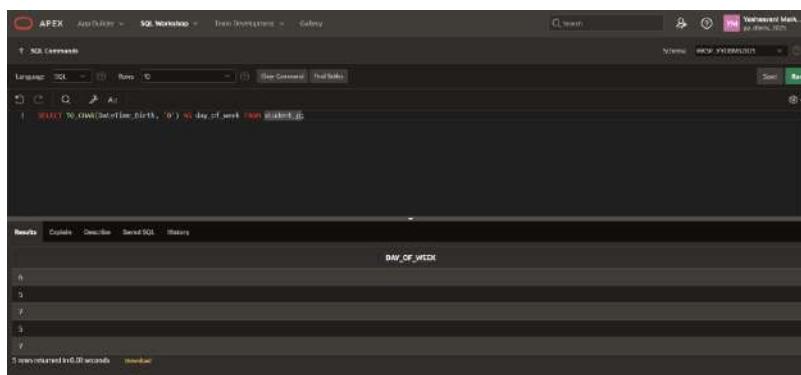
The screenshot shows the Oracle SQL Workshop interface. The query window contains the command:

```
SELECT TO_CHAR(DateTime_Birth, 'dd') AS day_of_month FROM student;
```

The results window displays the output:

DAY_OF_MONTH
21
21
21
21
21

11 SELECT DAYOFWEEK(DateTime_Birth) AS day_of_week FROM student;



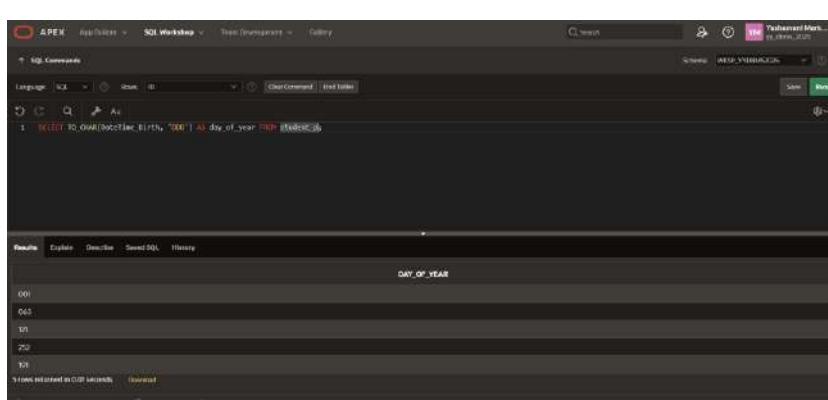
The screenshot shows the Oracle SQL Workshop interface. The query window contains the command:

```
SELECT TO_CHAR(DateTime_Birth, '01') AS day_of_week FROM student;
```

The results window displays the output:

DAY_OF_WEEK
6
5
7
3
7

12 SELECT DAYOFYEAR(DateTime_Birth) AS day_of_year FROM student;



The screenshot shows the Oracle SQL Workshop interface. The query window contains the command:

```
SELECT TO_CHAR(DateTime_Birth, 'yy') AS day_of_year FROM student;
```

The results window displays the output:

DAY_OF_YEAR
001
063
121
282
121

13 SELECT FROM_DAYS(737846) AS from_days_date FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following SQL statement:

```
SELECT TO_DATE('2019-01-01 00:00:00', 'YYYY-MM-DD HH:MI:SS') + 737846 AS from_days_date FROM dual;
```

The Results tab displays the output:

FROM_DAYS_DATE
27/05/2020

Execution time: 0.00 seconds. The status bar at the bottom indicates "Results APEX 21.1.0".

14 SELECT HOUR(DateTime_Birth) AS hour_value FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following SQL statement:

```
SELECT HOUR(DateTime_Birth) AS hour_value FROM student;
```

The Results tab displays the output:

HOUR_VALUE
04
05
06
07
08

Execution time: 0.00 seconds. The status bar at the bottom indicates "Results APEX 21.1.0".

15 SELECT LAST_DAY(DateTime_Birth) AS last_day_of_month FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following SQL statement:

```
SELECT LAST_DAY(DateTime_Birth) AS last_day_of_month FROM student;
```

The Results tab displays the output:

LAST_DAY_OF_MONTH
17/05/2020
18/05/2020
19/05/2020
20/05/2020
21/05/2020

Execution time: 0.03 seconds. The status bar at the bottom indicates "Results APEX 21.1.0".

16 SELECT NOW() AS current_datetime FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following SQL statement:

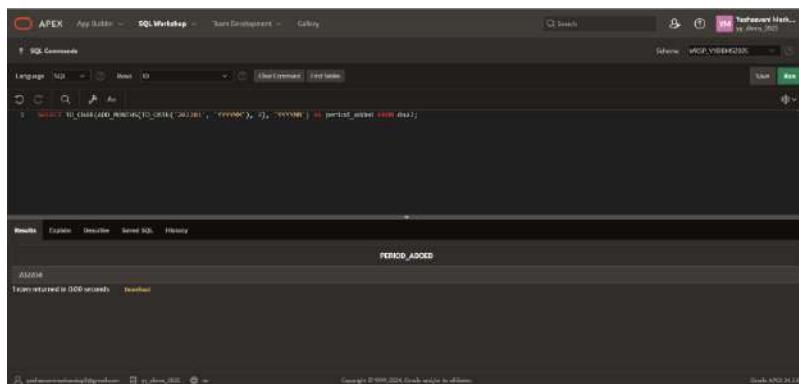
```
SELECT SYSTIMESTAMP AS current_datetime FROM dual;
```

The Results tab displays the output:

CURRENT_DATETIME
20/05/2020 10:45:45.000000000

Execution time: 0.00 seconds. The status bar at the bottom indicates "Results APEX 21.1.0".

17 SELECT PERIOD_ADD(202201, 3) AS period_added FROM student;



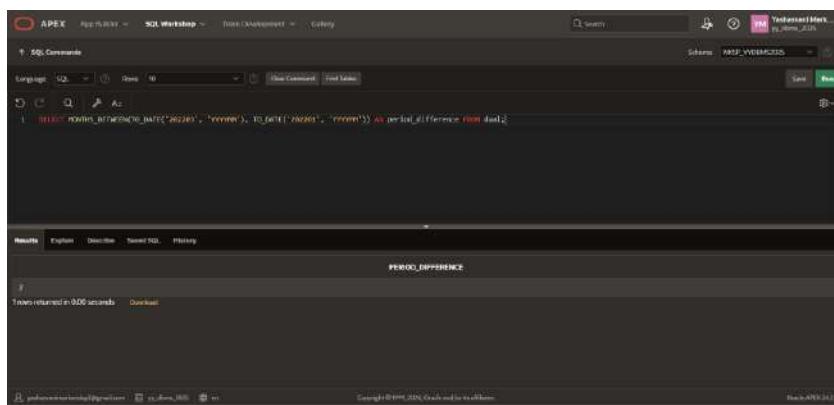
The screenshot shows the Oracle SQL Workshop interface. The SQL command is:

```
1 SELECT TO_CHAR(TO_DATE('202201','YYYYMM'), 'IYYYMM') AS period_added FROM dual;
```

The results show a single row with the value '202204'.

PERIOD_ADDED
202204

18 SELECT PERIOD_DIFF(202203, 202201) AS period_difference FROM student;



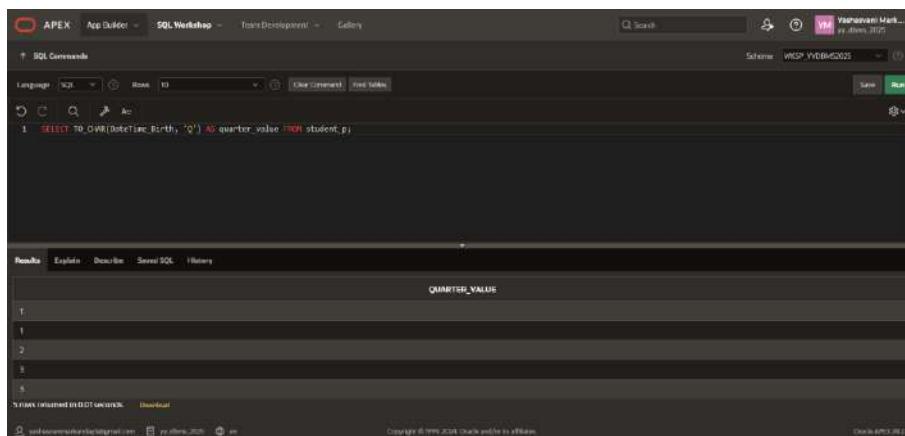
The screenshot shows the Oracle SQL Workshop interface. The SQL command is:

```
1 SELECT months_between(DATE('202203','YYYYMM'), '202201') AS period_difference FROM dual;
```

The results show a single row with the value '2'. This represents a difference of 2 months between the two specified periods.

PERIOD_DIFFERENCE
2

19 SELECT QUARTER(DateTime_Birth) AS quarter_value FROM student;



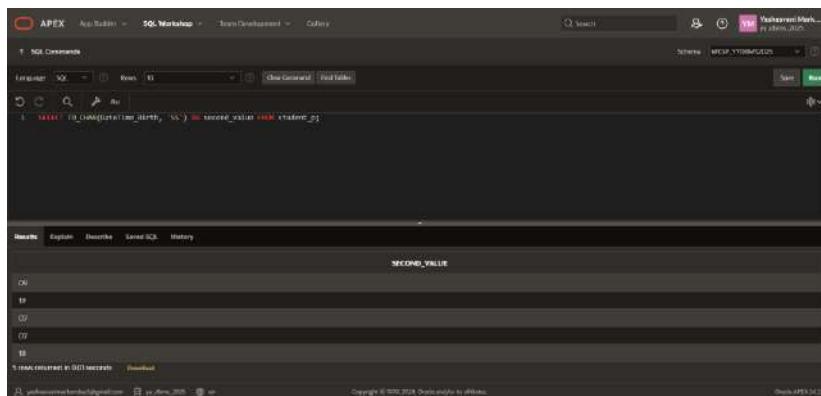
The screenshot shows the Oracle SQL Workshop interface. The SQL command is:

```
1 SELECT TO_CHAR(DateTime_Birth, 'Q') AS quarter_value FROM student_p;
```

The results show a single row with the value '1'.

QUARTER_VALUE
1

20 SELECT SECOND(DateTime_Birth) AS second_value FROM student;



The screenshot shows the Oracle SQL Workshop interface. The query window contains the following SQL statement:

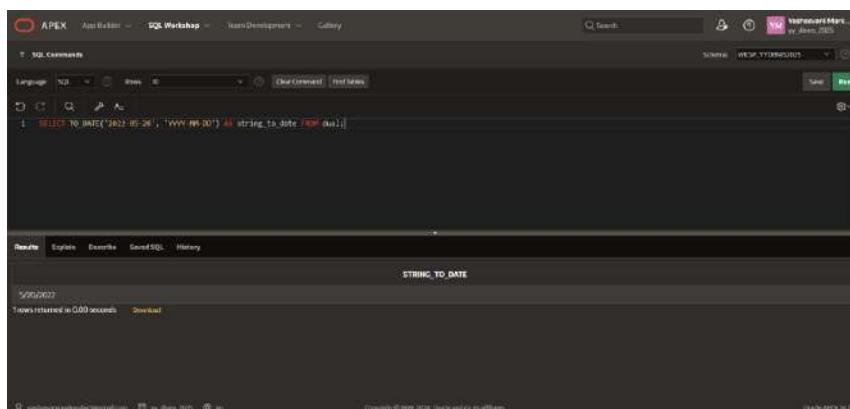
```
SELECT second(DateTime_Birth) AS second_value FROM student;
```

The results pane shows the output:

SECOND_VALUE
01
19
07
09
18

Below the results, it says "1 row returned in 0.00 seconds".

21 SELECT STR_TO_DATE('2022-05-20', '%Y-%m-%d') AS string_to_date FROM student;



The screenshot shows the Oracle SQL Workshop interface. The query window contains the following SQL statement:

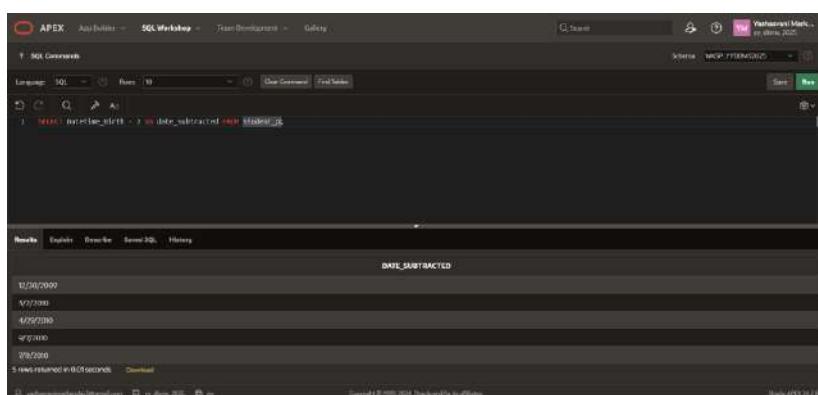
```
SELECT STR_TO_DATE('2022-05-20', '%Y-%m-%d') AS string_to_date FROM student;
```

The results pane shows the output:

STRING_TO_DATE
2022-05-20

Below the results, it says "1 row returned in 0.00 seconds".

22 SELECT SUBDATE(DateTime_Birth, INTERVAL 2 DAY) AS date_subtracted FROM student;



The screenshot shows the Oracle SQL Workshop interface. The query window contains the following SQL statement:

```
SELECT SUBDATE(DateTime_Birth, INTERVAL 2 DAY) AS date_subtracted FROM student;
```

The results pane shows the output:

DATE_SUBTRACTED
12/31/2009
5/31/2009
4/30/2009
3/31/2009
2/28/2009

Below the results, it says "5 rows returned in 0.00 seconds".

23 SELECT SUBTIME(DateTime_Birth, '03:15:00') AS time_subtracted FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains:

```
1 SELECT SUBTIME(DateTime_Birth, '03:15:00') AS time_subtracted FROM student;
```

The results window shows the output:

TIME_SUBTRACTED
17/08/2000
18/08/2000
19/08/2000
20/08/2000
21/08/2000

3 rows returned in 0.00 seconds. [Download](#)

24 SELECT SYSDATE() AS system_date FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains:

```
1 SELECT SYSDATE AS system_date FROM student;
```

The results window shows the output:

SYSTEM_DATE
27/08/2020

1 rows returned in 0.03 seconds. [Download](#)

25 SELECT TIME(DateTime_Birth) AS time_value FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains:

```
1 SELECT TIME(DateTime_Birth) AS time_value FROM student;
```

The results window shows the output:

TIME_VALUE
09:59:09
02:53:19
10:33:03
14:17:33
20:45:10

3 rows returned in 0.02 seconds. [Download](#)

26 SELECT TIME_FORMAT(DateTime_Birth, '%H:%i:%s') AS formatted_time FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following query:

```
1 SELECT TO_CHAR(DateTime_Birth, 'HH24MISS') || 'Formatted_time' FROM student_p;
```

The Results tab displays the output:

FORMATTED_TIME
085900
085930
085937
085958

5 rows returned in 0.02 seconds. [Download](#)

27 SELECT TIME_TO_SEC(DateTime_Birth) AS time_to_seconds FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following query:

```
1 SELECT EXTRACT(HOUR FROM CAST(DateTime_Birth AS TIMESTAMP)) * 3600 +
2 EXTRACT(MINUTE FROM CAST(DateTime_Birth AS TIMESTAMP)) * 60 +
3 EXTRACT(SECOND FROM CAST(DateTime_Birth AS TIMESTAMP))) AS time_to_seconds
4 FROM student_p;
```

The Results tab displays the output:

TIME_TO_SECONDS
31740
31749
31769
31777
31790

5 rows returned in 0.01 seconds. [Download](#)

28 SELECT TIMEDIFF(NOW(), DateTime_Birth) AS time_difference FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following query:

```
1 SELECT CAST(SYSDATE AS TIMESTAMP) - CAST(DateTime_Birth AS TIMESTAMP)) AS time_difference
2 FROM student_p;
```

The Results tab displays the output:

TIME_DIFFERENCE
-4000007452329000000
-40000065612486000000
-400000656107520000000
-400000656100480000000
-400000656100480000000

5 rows returned in 0.00 seconds. [Download](#)

29 SELECT TIMESTAMP('2022-04-10') AS timestamp_value FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the following SQL code:

```
1 SELECT TO_TIMESTAMP('2022-04-10', 'YYYY-MM-DD') AS timestamp_value FROM dual;
```

The results pane shows the output:

TIMESTAMP_VALUE
2022-04-10 00:00:00.000000 AM

Rows returned in 0.01 seconds. Detailed

30 SELECT TO_DAYS('2022-06-15') AS to_days_value FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the following SQL code:

```
1 SELECT TO_DATE('2022-06-15', 'YYYY-MM-DD') - TO_DATE('2001-01-01', 'YYYY-MM-DD') AS to_days_value
2 FROM dual;
```

The results pane shows the output:

TO_DAYS_VALUE
75852

Rows returned in 0.01 seconds. Detailed

31 SELECT WEEKDAY(DateTime_Birth) AS weekday_index FROM student;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the following SQL code:

```
1 SELECT TO_CHAR(DateTime_Birth, 'D') AS weekday_index FROM student;
```

The results pane shows the output:

WEEKDAY_INDEX
6
5
2
5
7

Rows returned in 0.00 seconds. Detailed

32 SELECT WEEK(DateTime_Birth) AS week_value FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains the following SQL code:

```
SELECT TO_CHAR(DateTime_Birth, 'IYYY') AS week_value FROM student_p;
```

The results section displays the output:

WEEK_VALUE
01
02
03
04
05
06
07

Below the table, it says "5 rows returned in 0.00 seconds".

33 SELECT WEEKOFYEAR(DateTime_Birth) AS week_of_year_value FROM student;

The screenshot shows the Oracle SQL Workshop interface. The query window contains the following SQL code:

```
SELECT TO_CHAR(DateTime_Birth, 'IW') AS week_of_year_value FROM student_p;
```

The results section displays the output:

WEEK_OF_YEAR_VALUE
54
01
02
03
04
05

Below the table, it says "5 rows returned in 0.00 seconds".

Conditional Functions

The first screenshot shows the execution of a query using the IFNULL function:

```
IFNULL(1,2);
```

The results show "1" and "Elapsed: 0.00 seconds".

The second screenshot shows the execution of an INSERT INTO statement:

```
INSERT INTO sample_table VALUES ('1', TO_DATE('2002-01-01', 'YYYY-MM-DD'));  
INSERT INTO sample_table VALUES ('2', 'banana', TO_DATE('2003-02-28', 'YYYY-MM-DD'));  
INSERT INTO sample_table VALUES ('3', 'orange', TO_DATE('2003-03-28', 'YYYY-MM-DD'));  
INSERT INTO sample_table VALUES ('4', 'apple', TO_DATE('2002-08-15', 'YYYY-MM-DD'));  
INSERT INTO sample_table VALUES ('5', 'kiwi', TO_DATE('2002-08-05', 'YYYY-MM-DD'))  
COMMIT = logitail;
```

The results show "5 rows inserted" and "Elapsed: 0.00 seconds".

34 SELECT * FROM sample_table WHERE column1 > 2 AND column3 > '2022-03-01';

The screenshot shows the Oracle SQL Workshop interface. The SQL command window contains the following query:

```
1 SELECT *
2 FROM sample_table
3 WHERE column1 > 2
4 AND column3 > TO_DATE('2022-03-01','YYYY-MM-DD');
```

The results window displays the following data:

COLUMN1	COLUMN2	COLUMN3
3	orange	3/20/2022
4	grape	4/10/2022
5	lime	5/5/2022

2 rows returned in 0.01 seconds. Download

35 SELECT * FROM sample_table WHERE column1 = 2 OR column2 = 'orange';

The screenshot shows the Oracle SQL Workshop interface. The SQL command window contains the following query:

```
1 SELECT *
2 FROM sample_table
3 WHERE column1 = 2
4 OR column2 = 'orange';
```

The results window displays the following data:

COLUMN1	COLUMN2	COLUMN3
2	banana	2/15/2022
3	orange	3/20/2022

2 rows returned in 0.01 seconds. Download

36 SELECT * FROM sample_table WHERE (column1 > 2 AND column3 > '2022-03-01') OR column2 = 'banana';

The screenshot shows the Oracle SQL Workshop interface. The SQL command window contains the following query:

```
1 SELECT *
2 FROM sample_table
3 WHERE (column1 > 2 AND column3 > TO_DATE('2022-03-01','YYYY-MM-DD'))
4 OR column2 = 'banana';
```

The results window displays the following data:

COLUMN1	COLUMN2	COLUMN3
2	banana	2/15/2022
3	orange	3/20/2022
4	grape	4/10/2022
5	lime	5/5/2022

4 rows returned in 0.01 seconds. Download

37 SELECT * FROM sample_table WHERE column2 LIKE 'a%';

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL statement is entered:

```
1 SELECT *
2 FROM sample_table
3 WHERE column2 LIKE 'a%';
```

In the Results pane, the output is displayed in a grid:

COLUMN1	COLUMN2	COLUMNS
1	apple	1/1/2023

Below the grid, it says "1 rows returned in 0.02 seconds".

38 SELECT * FROM sample_table WHERE column1 IN (2, 4);

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL statement is entered:

```
1 SELECT *
2 FROM sample_table_n
3 WHERE column1 IN (2, 4);
```

In the Results pane, the output is displayed in a grid:

COLUMN1	COLUMN2	COLUMNS
2	banana	3/5/2022
4	orange	4/10/2022

Below the grid, it says "2 rows returned in 0.01 seconds".

39 SELECT * FROM sample_table WHERE column1 > ANY (SELECT column1 FROM sample_table WHERE column3 > '2022-03-01');

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL statement is entered:

```
1 SELECT *
2 FROM sample_table_n
3 WHERE column1 > ANY (
4     SELECT column1
5     FROM sample_table_n
6     WHERE column3 > TO_DATE('2022-03-01', 'YYYY-MM-DD')
7 );
```

In the Results pane, the output is displayed in a grid:

COLUMN1	COLUMN2	COLUMNS
5	kiwi	5/5/2022
4	orange	4/10/2022

Below the grid, it says "2 rows returned in 0.02 seconds".

40 SELECT * FROM sample_table WHERE EXISTS (SELECT * FROM sample_table WHERE column1 = 3);

```
SELECT *
FROM sample_table_n
WHERE column1 = 1
SELECT *
FROM sample_table_n
WHERE column1 = 2
SELECT *
FROM sample_table_n
WHERE column1 = 3
;

```

Results

COLUMN1	COLUMN2	COLUMNS
1	apple	1/1/2022
2	banana	2/2/2022
3	orange	3/3/2022
4	grape	4/4/2022
5	lime	5/5/2022

5 rows returned in 0.01 seconds Download

41 SELECT * FROM sample_table WHERE NOT column1 = 2;

```
SELECT *
FROM sample_table_n
WHERE NOT (column1 = 2);

```

Results

COLUMN1	COLUMN2	COLUMNS
1	apple	1/1/2022
3	orange	3/3/2022
4	grape	4/4/2022
5	lime	5/5/2022

4 rows returned in 0.01 seconds Download

42 SELECT * FROM sample_table WHERE column1 <> 2;

```
SELECT *
FROM sample_table_n
WHERE column1 <> 2;

```

Results

COLUMN1	COLUMN2	COLUMNS
1	apple	1/1/2022
3	orange	3/3/2022
4	grape	4/4/2022
5	lime	5/5/2022

4 rows returned in 0.00 seconds Download

43 SELECT * FROM sample_table WHERE column2 IS NULL;

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL statement is entered:

```
1 SELECT *
2 FROM sample_table.n
3 WHERE column2 IS NOT NULL;
```

In the Results pane, the output is displayed in a grid format:

COLUMN	COLUMN	COLUMN
1		5/5/2022

Below the grid, it says "1 row returned in 0.00 seconds".

44 SELECT * FROM sample_table WHERE column2 IS NOT NULL;

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL statement is entered:

```
1 SELECT *
2 FROM sample_table.n
3 WHERE column2 IS NOT NULL;
```

In the Results pane, the output is displayed in a grid format:

COLUMN	COLUMN	COLUMN
1	apple	1/1/2022
2	banana	2/5/2022
3	orange	3/20/2022
4	grape	4/15/2022
5	lime	5/5/2022

Below the grid, it says "5 rows returned in 0.01 seconds".

45 SELECT * FROM sample_table WHERE column1 BETWEEN 2 AND 4;

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL statement is entered:

```
1 SELECT *
2 FROM sample_table.n
3 WHERE column1 BETWEEN 2 AND 4;
```

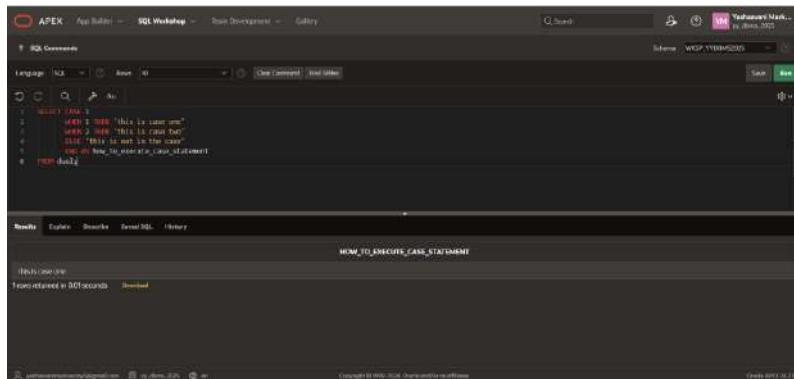
In the Results pane, the output is displayed in a grid format:

COLUMN	COLUMN	COLUMN
2	banana	2/5/2022
3	orange	3/20/2022
4	grape	4/15/2022

Below the grid, it says "3 rows returned in 0.01 seconds".

1. Simple CASE Statement: - When the condition is met (CASE 1), it returns the corresponding value. - When the condition is not met (CASE 4), it returns the value in the ELSE part.

46. SELECT CASE 1 WHEN 1 THEN 'this is case one' WHEN 2 THEN 'this is case two' ELSE 'this is not in the case' END as 'how_to_execute_case_statement';



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

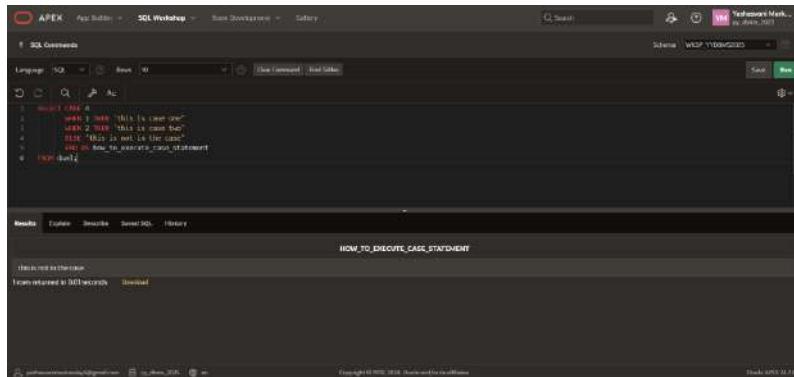
```
1 SELECT CASE
2   WHEN 1 THEN 'This is case one'
3   WHEN 2 THEN 'This is case two'
4   ELSE 'This is not in the case'
5   END AS how_to_execute_case_statement
6 FROM dual;
```

In the Results pane, the output is displayed under the heading "HOW_TO_EXECUTE_CASE_STATEMENT". The result is:

how_to_execute_case_statement
This is case one

Execution details: 1 row(s) returned in 0.00 seconds. Elapsed: 0:00:00.000000

47. SELECT CASE 4 WHEN 1 THEN 'this is case one' WHEN 2 THEN 'this is case two' ELSE 'this is not in the case' END as 'how_to_execute_case_statement';



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

```
1 SELECT CASE
2   WHEN 1 THEN 'This is case one'
3   WHEN 2 THEN 'This is case two'
4   ELSE 'This is not in the case'
5   END AS how_to_execute_case_statement
6 FROM dual;
```

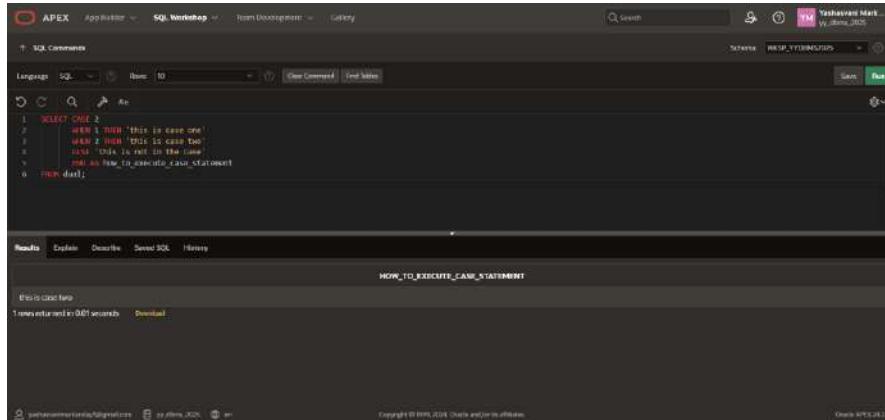
In the Results pane, the output is displayed under the heading "HOW_TO_EXECUTE_CASE_STATEMENT". The result is:

how_to_execute_case_statement
This is not in the case

Execution details: 1 row(s) returned in 0.00 seconds. Elapsed: 0:00:00.000000

2. CASE Statement with Matching Condition: - When the condition is met (CASE 2), it returns the corresponding value.

48. SELECT CASE 2 WHEN 1 THEN 'this is case one' WHEN 2 THEN 'this is case two' ELSE 'this is not in the case' END as 'how to execute case statement';



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

```
1 SELECT CASE 2
2   WHEN 1 THEN 'this is case one'
3   WHEN 2 THEN 'this is case two'
4   ELSE 'this is not in the case'
5   END AS how_to_execute_case_statement
6 FROM dual;
```

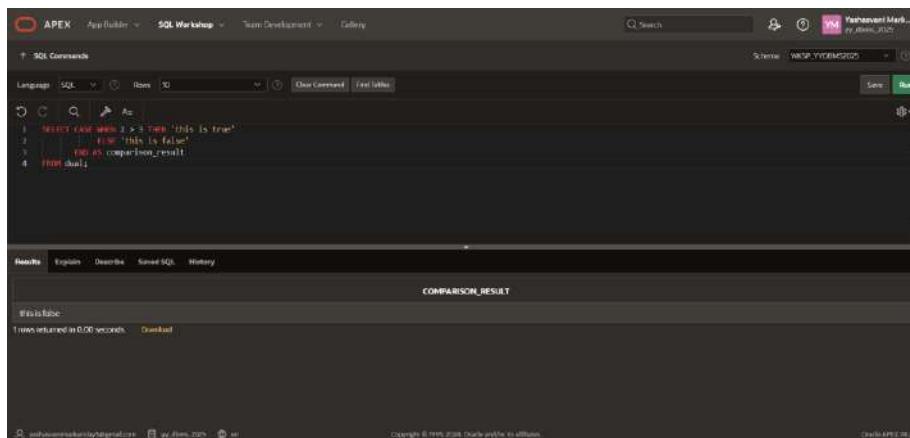
In the Results pane, the output is displayed under the heading 'HOW_TO_EXECUTE_CASE_STATEMENT':

```
this is case two
```

1 rows returned in 0.01 seconds. Download

3. CASE Statement with Comparison Operators: - Using greater than and less than operators to evaluate conditions.

49. SELECT CASE WHEN 2>3 THEN 'this is true' ELSE 'this is false' END;



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

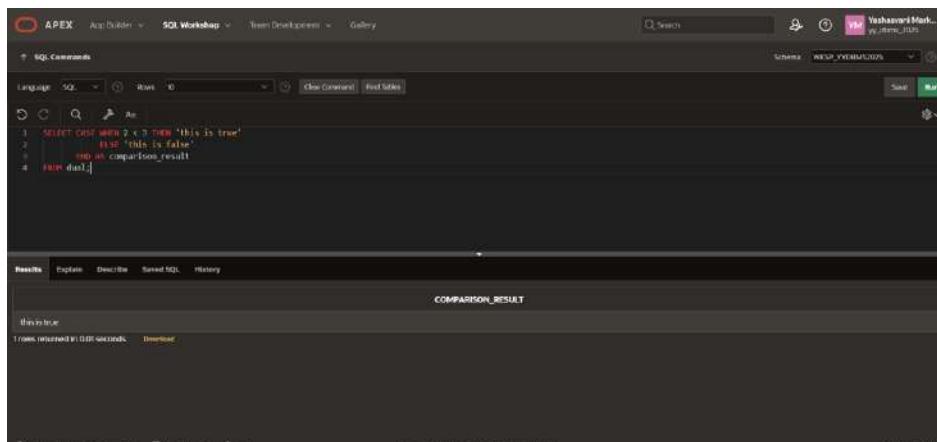
```
1 SELECT CASE WHEN 2>3 THEN 'this is true'
2   ELSE 'this is false'
3   END AS comparison_result
4 FROM dual;
```

In the Results pane, the output is displayed under the heading 'COMPARISON_RESULT':

```
this is false
```

1 rows returned in 0.00 seconds. Download

50. SELECT CASE WHEN 2<3 THEN 'this is true' ELSE 'this is false' END.



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

```
1 SELECT CASE WHEN 2<3 THEN 'this is true'
2   ELSE 'this is false'
3   END AS comparison_result
4 FROM dual;
```

In the Results pane, the output is displayed under the heading 'COMPARISON_RESULT':

```
this is true
```

1 rows returned in 0.01 seconds. Download

4. CASE Statement with No Matching Conditions: - If none of the conditions are satisfied, it returns NULL.

51. SELECT CASE BINARY 'A' WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following code:

```
1 SELECT CASE BINARY 'A' WHEN 'a' THEN 1
2          WHEN 'A' = 'b' THEN 2
3          ELSE NO_MATCHING_CASE
4      END AS no_matching_case
5 FROM dual;
```

The Results tab displays the output:

```
NO_MATCHING_CASE
```

1 row returned in 0.01 seconds.

52. CREATE TABLE student_grades (student_id INT, grade INT);

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following code:

```
1 CREATE TABLE student_grades (
2   student_id INT,
3   grade INT
4 );
```

The Results tab displays the output:

```
Table created.
0.04 seconds
```

53. INSERT INTO student_grades (student_id, grade) VALUES (1, 95), (2, 85), (3, 75), (4, 60);

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab contains the following code:

```
1 INSERT ALL
2   INTO student_grades (student_id, grade) VALUES (1, 95)
3   INTO student_grades (student_id, grade) VALUES (2, 85)
4   INTO student_grades (student_id, grade) VALUES (3, 75)
5   INTO student_grades (student_id, grade) VALUES (4, 60)
6   SELECT * FROM dual;
```

The Results tab displays the output:

```
4 rows(s) inserted.
0.01 seconds
```

54. CREATE TABLE employee_data (employee_id INT, salary DECIMAL(10, 2));

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "SQL Commands" tab is selected. The SQL editor contains the following code:

```
1 CREATE TABLE employee_data (
2     employee_id INT,
3     salary DECIMAL(10, 2)
4 );
```

The results pane shows the output of the command:

```
Table created.
```

Execution time: 0.05 seconds.

55. INSERT INTO employee_data (employee_id, salary) VALUES (101, 120000), (102, 75000), (103, 45000), (104, 110000);

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "SQL Commands" tab is selected. The SQL editor contains the following code:

```
1 BEGIN;
2   INSERT employee_data (employee_id, salary) VALUES (101, 120000);
3   INSERT employee_data (employee_id, salary) VALUES (102, 75000);
4   INSERT employee_data (employee_id, salary) VALUES (103, 45000);
5   INSERT employee_data (employee_id, salary) VALUES (104, 110000);
6 END; /* FROM Dual;
```

The results pane shows the output of the command:

```
4 rows(s) inserted.
```

Execution time: 0.05 seconds.

56. CREATE TABLE product_inventory (product_id INT, quantity INT);

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "SQL Commands" tab is selected. The SQL editor contains the following code:

```
1 CREATE TABLE product_inventory (
2     product_id INT,
3     quantity INT
4 );
```

The results pane shows the output of the command:

```
Table created.
```

Execution time: 0.03 seconds.

57. INSERT INTO product_inventory (product_id, quantity) VALUES (1, 75), (2, 20), (3, 5), (4, 100);

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, the following SQL code is entered:

```
1 INSERT ALL
2   INTO product_inventory (product_id, quantity) VALUES (1, 75)
3   INTO product_inventory (product_id, quantity) VALUES (2, 20)
4   INTO product_inventory (product_id, quantity) VALUES (3, 5)
5   INTO product_inventory (product_id, quantity) VALUES (4, 100)
6   SELECT * FROM dual;
```

The Results tab shows the output:

```
0 rows(s) inserted.
```

Execution time: 0.05 seconds

58. CREATE TABLE customer_orders (order_id INT, order_status VARCHAR(20));

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, the following SQL code is entered:

```
CREATE TABLE customer_orders (
    order_id INT,
    order_status VARCHAR(20)
);
```

The Results tab shows the output:

```
Table created.
```

Execution time: 0.02 seconds

59. INSERT INTO customer_orders (order_id, order_status) VALUES (1, 'Shipped'), (2, 'Processing'), (3, 'Cancelled'), (4, 'Pending');

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, the following SQL code is entered:

```
1 INSERT ALL
2   INTO customer_orders (order_id, order_status) VALUES (1, 'Shipped')
3   INTO customer_orders (order_id, order_status) VALUES (2, 'Processing')
4   INTO customer_orders (order_id, order_status) VALUES (3, 'Cancelled')
5   INTO customer_orders (order_id, order_status) VALUES (4, 'Pending')
6   SELECT * FROM dual;
```

The Results tab shows the output:

```
4 rows(s) inserted.
```

Execution time: 0.04 seconds

60. CREATE TABLE temperature_readings (location VARCHAR(50), temperature DECIMAL(5, 2));

```
CREATE TABLE temperature_readings (
  location VARCHAR(50),
  temperature DECIMAL(5, 2)
);
```

Results Explain Describe Saved SQL History

Table created.

0.05 seconds

61. INSERT INTO temperature_readings (location, temperature) VALUES ('City A', 32.5), ('City B', 25.0), ('City C', 18.5), ('City D', 28.0);

-- Display grades based on conditions

```
INSERT ALL
INTO temperature_readings (location, temperature) VALUES ('City A', 32.5)
INTO temperature_readings (location, temperature) VALUES ('City B', 25.0)
INTO temperature_readings (location, temperature) VALUES ('City C', 18.5)
INTO temperature_readings (location, temperature) VALUES ('City D', 28.0)
SELECT * FROM dual;
```

Results Explain Describe Saved SQL History

8 rows(s) inserted.

0.05 seconds

62. SELECT student_id, grade, CASE WHEN grade >= 90 THEN 'A' WHEN grade BETWEEN 80 AND 89 THEN 'B' WHEN grade BETWEEN 70 AND 79 THEN 'C' ELSE 'F' END AS grade_category FROM student_grades;

-- Categorize employees based on salary

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```

1 SELECT student_id, grade,
2 CASE
3 WHEN grade >= 90 THEN 'A'
4 WHEN grade BETWEEN 80 AND 89 THEN 'B'
5 WHEN grade BETWEEN 70 AND 79 THEN 'C'
6 ELSE 'F'
7 END AS grade_category
8 FROM student_grades;

```

The results grid displays the following data:

STUDENT_ID	GRADE	GRADE_CATEGORY
1	95	A
2	95	A
3	75	C
4	60	F
5	95	A
6	85	B
7	75	C
8	60	F

63. SELECT employee_id, salary, CASE WHEN salary > 100000 THEN 'High Income' WHEN salary BETWEEN 50000 AND 100000 THEN 'Moderate Income' ELSE 'Low Income' END AS income_category FROM employee_data;

-- Determine product availability based on quantity

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following code:

```

1 SELECT employee_id, salary,
2 CASE
3 WHEN salary > 100000 THEN 'High Income'
4 WHEN salary BETWEEN 50000 AND 100000 THEN 'Moderate Income'
5 ELSE 'Low Income'
6 END AS INCOME_CATEGORY
7 FROM employee_data;

```

The results grid displays the following data:

EMPLOYEE_ID	SALARY	INCOME_CATEGORY
101	120000	High Income
102	75000	Moderate Income
103	45000	Low Income
104	110000	High Income

64. SELECT product_id, quantity, CASE WHEN quantity > 50 THEN 'In Stock' WHEN quantity BETWEEN 10 AND 50 THEN 'Low Stock' ELSE 'Out of Stock' END AS availability_status FROM product_inventory;

-- Categorize orders based on status

```

1 SELECT product_id, quantity,
2       CASE
3         WHEN quantity > 50 THEN 'In Stock'
4         WHEN quantity BETWEEN 10 AND 50 THEN 'Low Stock'
5         ELSE 'Out of Stock'
6       END AS availability_status
7   FROM product_inventory;

```

The results show the following data:

PRODUCT_ID	QUANTITY	AVAILABILITY_STATUS
1	75	In Stock
2	20	Low Stock
3	5	Out of Stock
4	100	In Stock

4 rows returned in 0.03 seconds

65. `SELECT order_id, order_status, CASE WHEN order_status = 'Shipped' THEN 'Order Shipped' WHEN order_status = 'Processing' THEN 'Order Processing' WHEN order_status = 'Cancelled' THEN 'Order Cancelled' ELSE 'Unknown Status' END AS order_category FROM customer_orders;`

-- Categorize temperature readings based on conditions

```

1 SELECT order_id, order_status,
2       CASE
3         WHEN order_status = 'Shipped' THEN 'Order Shipped'
4         WHEN order_status = 'Processing' THEN 'Order Processing'
5         WHEN order_status = 'Cancelled' THEN 'Order Cancelled'
6         ELSE 'Unknown Status'
7       END AS order_category
8   FROM customer_orders;

```

The results show the following data:

ORDER_ID	ORDER_STATUS	ORDER_CATEGORY
1	Shipped	Order Shipped
2	Processing	Order Processing
3	Cancelled	Order Cancelled
4	Pending	Unknown Status

4 rows returned in 0.03 seconds

66. `SELECT location, temperature, CASE WHEN temperature > 30 THEN 'Hot' WHEN temperature BETWEEN 20 AND 30 THEN 'Moderate' ELSE 'Cool' END AS temperature_category FROM temperature_readings;`

```

1 SELECT location, temperature,
2       CASE
3         WHEN temperature > 30 THEN 'Hot'
4         WHEN temperature BETWEEN 20 AND 30 THEN 'Moderate'
5         ELSE 'Cool'
6       END AS temperature_category
7   FROM temperature_readings;

```

The results show the following data:

LOCATION	TEMPERATURE	TEMPERATURE_CATEGORY
City A	32	Hot
City B	25	Moderate
City C	18	Cool
City D	28	Moderate

4 rows returned in 0.02 seconds

67. CREATE TABLE books (book_id VARCHAR(10), book_name VARCHAR(50), isbn_no VARCHAR(11), cate_id VARCHAR(10), aut_id VARCHAR(10), pub_id VARCHAR(10), dt_of_pub DATE, pub_lang VARCHAR(20), no_page INT, book_price DECIMAL(8, 2));

```

APEX App Builder SQL Workshop Team Development Gallery
SQL Commands Rows: 10 Clear Command Find Table Save Run
CREATE TABLE books (
  book_id VARCHAR(10),
  book_name VARCHAR(50),
  isbn_no VARCHAR(11),
  cate_id VARCHAR(10),
  aut_id VARCHAR(10),
  pub_id VARCHAR(10),
  dt_of_pub DATE,
  pub_lang VARCHAR(20),
  no_page INT,
  book_price DECIMAL(8, 2)
);

```

Results Explain Describe Saved SQL History

Table created.

0.04 seconds

Copyright © 1996-2024, Oracle and/or its affiliates. Oracle APEX 24.1.0

68. INSERT INTO books VALUES ('BK001', 'Introduction to Electrodynamics', '0000979001', 'CA001', 'AUT001', 'P003', '2001-05-08', 'English', 201, 85.00), ('BK002', 'Understanding of Steel Construction', '0000979002', 'CA002', 'AUT002', 'P001', '2003-07-15', 'English', 300, 105.50), ('BK003', 'Guide to Networking', '0000979003', 'CA003', 'AUT003', 'P002', '2002-09-10', 'Hindi', 510, 200.00), ('BK004', 'Transfer of Heat and Mass', '0000979004', 'CA002', 'AUT004', 'P004', '2004-02-16', 'English', 600, 250.00), ('BK005', 'Conceptual Physics', '0000979005', 'CA001', 'AUT005', 'P006', '2003-07-16', NULL, 345, 145.00), ('BK006', 'Fundamentals of Heat', '0000979006', 'CA001', 'AUT006', 'P005', '2003-08-10', 'German', 247, 112.00), ('BK007', 'Advanced 3d Graphics', '0000979007', 'CA003', 'AUT007', 'P002', '2004-02-16', 'Hindi', 165, 56.00), ('BK008', 'Human Anatomy', '0000979008', 'CA005', 'AUT008', 'P006', '2001-05-17', 'German', 88, 50.50), ('BK009', 'Mental Health Nursing', '0000979009', 'CA005', 'AUT009', 'P007', '2004-02-10', 'English', 350, 145.00), ('BK010', 'Fundamentals of Thermodynamics', '0000979010', 'CA002', 'AUT010', 'P007', '2002-10-14', 'English', 400, 225.00), ('BK011', 'The Experimental Analysis of Cat', '0000979011', 'CA004', 'AUT011', 'P005', '2007-06-09', 'French', 225, 95.00), ('BK012', 'The Nature of World', '0000979012', 'CA004', 'AUT005', 'P008', '2005-12-20', 'English', 350, 88.00), ('BK013', 'Environment a Sustainable Future', '0000979013', 'CA004', 'AUT012', 'P001', '2003-10-27', 'German', 165, 100.00), ('BK014', 'Concepts in Health', '0000979014', 'CA005', 'AUT013', 'P004', '2001-08-25', NULL, 320, 180.00), ('BK015', 'Anatomy & Physiology', '0000979015', 'CA005', 'AUT014', 'P008', '2000-10-10', 'Hindi', 225, 135.00), ('BK016', 'Networks and Telecommunications', '00009790_16', 'CA003', 'AUT015', 'P003', '2002-01-01', 'French', 95, 45.00);

```

SQL Commands Schema WESP_YDIMENTUS
Language SQL Rows: 10 Clear Command Find Table Save Run
16 rows(s) inserted.
0.05 seconds
Copyright © 1996-2024, Oracle and/or its affiliates. Oracle APEX 24.1.0

```

```

16 rows(s) inserted.
0.05 seconds
Copyright © 1996-2024, Oracle and/or its affiliates. Oracle APEX 24.1.0

```

69. select * from books;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains the command: `SELECT * FROM books;`. The results section displays 10 rows of book data:

BOOK_ID	BOOK_NAME	ISBN_NO	CATE_ID	AUT_ID	PUBL_ID	DT_OF_PUB	PUBL_LANG	NO_PAGE	BOOK_PRICE
BK002	Understanding of Steel Construction	0000979002	CAB02	AU1002	P001	2/5/2005	English	320	305.5
BK005	Guide to Networking	0000979005	CAB05	AU1003	P002	9/10/2002	Hindi	310	205
BK004	Transfer of Heat and Mass	0000979004	CAB09	AU1004	P003	2/10/2004	English	400	250
BK006	Conceptual Physics	0000979006	CAB03	AU1005	P004	2/8/2008	-	345	145
BK008	Fundamentals of Heat	0000979008	CAB08	AU1006	P005	8/10/2003	German	247	182
BK007	Advanced 3d Graphics	0000979007	CAB09	AU1002	P002	2/10/2004	Hindi	165	56
BK009	Human Anatomy	0000979009	CAB05	AU1006	P005	5/10/2001	German	85	505
BK008	Mental Health Nursing	0000979008	CAB05	AU1009	P007	2/10/2004	English	300	145
BK010	Fundamentals of Thermodynamics	0000979010	CAB02	AU1003	P009	10/10/2009	English	400	225
BK011	The Experimental Analysis of Cat	0000979011	CAB04	AU1011	P006	5/9/2007	French	225	95

More than 10 rows available. Increase rows selector to view more rows.

70. SELECT book_name, IF(pub_lang="English", "English Book", "Other Language") AS Language FROM books;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains the command: `SELECT book_name, IF(pub_lang='English', 'English Book', 'Other Language') AS Language FROM books;`. The results section displays 10 rows of book data with the 'Language' column categorized:

BOOK_NAME	LANGUAGE
Guide to Networking	Other Language
Transfer of Heat and Mass	Other Language
Conceptual Physics	Other Language
Fundamentals of Heat	Other Language
Advanced 3d Graphics	Other Language
Human Anatomy	Other Language
Mental Health Nursing	English Book
Fundamentals of Thermodynamics	English Book
The Experimental Analysis of Cat	Other Language

More than 10 rows available. Increase rows selector to view more rows.

71. SELECT book_name, isbn_no, IF((SELECT COUNT(*) FROM books WHERE pub_lang='English') > (SELECT COUNT(*) FROM books WHERE pub_lang<>'English'), (CONCAT("Pages: ",no_page)),(CONCAT("Price: ",book_price))) AS "Page / Price" FROM books;

The screenshot shows the Oracle APEX SQL Workshop interface. The query window contains the command: `SELECT book_name, isbn_no, IF((SELECT COUNT(*) FROM books WHERE pub_lang='English') > (SELECT COUNT(*) FROM books WHERE pub_lang<>'English'), (CONCAT("Pages: ",no_page)),(CONCAT("Price: ",book_price))) AS "Page / Price" FROM books;`. The results section displays 10 rows of book data with the 'Page / Price' column calculated:

BOOK_NAME	ISBN_NO	Page / Price
Understanding of Steel Construction	0000979002	Price:305.5
Guide to Networking	0000979005	Price:200
Transfer of Heat and Mass	0000979004	Price:250
Conceptual Physics	0000979009	Price:145
Fundamentals of Heat	0000979006	Price:182
Advanced 3d Graphics	0000979007	Price:56
Human Anatomy	0000979008	Price:505
Mental Health Nursing	0000979009	Price:145

Results Explain Describe Saved SQL History

BOOK_NAME	ISBN_NO	Page / Price
Principles of Heat	000099006	Price: 12
Advanced 3d Graphics	000099007	Price: 36
Human Anatomy	000099008	Price: 50.5
Mental Health Nursing	000099009	Price: 145
Fundamentals of Thermodynamics	000099010	Price: 225
The Experimental Analysis of Cat	000099011	Price: 95

More than 10 rows available; Increase rows selector to view more rows.
10 rows returned in 0.02 seconds Download Copyright © 1996-2024 Oracle and/or its affiliates. Oracle APEX 24.1.0

72. SELECT book_id, book_name, IF(pub_lang IS NULL,'N/A',pub_lang) AS "Pub. Language" FROM books;

APEX App Builder SQL Workshop Team Development Gallery Search Scheme WNSP_YDBMS2025

SQL Commands Rows: 10 Clear Command Find Tables Save Run

```
1 SELECT book_id, book_name,
2    IF(pub_lang IS NULL, 'N/A', pub_lang)
3   FROM books;
```

Results Explain Describe Saved SQL History

BOOK_ID	BOOK_NAME	Pub. Language
BK001	Understanding of Steel Construction	English
BK002	Guide to Networking	Hindi
BK003	Transfer of Heat and Mass	English
BK004	Conceptual Physics	N/A
BK005	Fundamentals of Heat	German
BK006	Advanced 3d Graphics	Hindi
BK007	Human Anatomy	German
BK008	Mental Health Nursing	English
BK009	Fundamentals of Thermodynamics	English
BK010	The Experimental Analysis of Cat	French

More than 10 rows available; Increase rows selector to view more rows.
Copyright © 1996-2024 Oracle and/or its affiliates. Oracle APEX 24.1.0

73. SELECT book_id, book_name, pub_lang FROM books;

APEX App Builder SQL Workshop Team Development Gallery Search Scheme WNSP_YDBMS2025

SQL Commands Rows: 10 Clear Command Find Tables Save Run

```
1 SELECT book_id, book_name, pub_lang
2   FROM books;
```

Results Explain Describe Saved SQL History

BOOK_ID	BOOK_NAME	PUB_LANG
BK002	Understanding of Steel Construction	English
BK003	Guide to Networking	Hindi
BK004	Transfer of Heat and Mass	English
BK005	Conceptual Physics	-
BK006	Fundamentals of Heat	German
BK007	Advanced 3d Graphics	Hindi
BK008	Human Anatomy	German
BK009	Mental Health Nursing	English
BK010	Fundamentals of Thermodynamics	English
BK011	The Experimental Analysis of Cat	French

More than 10 rows available; Increase rows selector to view more rows.
Copyright © 1996-2024 Oracle and/or its affiliates. Oracle APEX 24.1.0

74. CREATE TABLE purchase (invoice_no VARCHAR(10), invoice_dt DATE, ord_no VARCHAR(20), ord_date DATE, receive_dt DATE, book_id VARCHAR(10), book_name VARCHAR(50), pub_lang VARCHAR(20), cate_id VARCHAR(10), receive_qty INT, purch_price DECIMAL(8, 2), total_cost DECIMAL(10, 2));

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows 'APEX' and 'SQL Workshop'. The main area is titled 'SQL Commands' with a 'Run' button. The code editor contains the following SQL script:

```
CREATE TABLE purchase (
    invno NUMBER(10),
    purchase_dt DATE,
    book_name VARCHAR(20),
    pub_name VARCHAR(20),
    pub_lang VARCHAR(20),
    upc NUMBER(10),
    quantity NUMBER(2),
    unit_price NUMBER(10),
    total_qty NUMBER(10),
    total_amt NUMBER(10)
);
```

The results tab shows the message 'Table created.' and a duration of '0.05 seconds'. The bottom status bar includes the URL 'yashaswiniandu@gmail.com', session ID 'yy_dbsn_2025', and copyright information 'Copyright © 1996, 2004, Oracle. All rights reserved.'

75. INSERT INTO purchase VALUES ('INV0001', '2008-07-15', 'ORD/08-09/0001', '2008-07-06', '2008-07-19', 'BK001', 'Introduction to Electrodynamics', 'English', 'CA001', 15, 75.00, 1125.00), ('INV0002', '2008-08-25', 'ORD/08-09/0002', '2008-08-09', '2008-08-28', 'BK004', 'Transfer of Heat and Mass', 'English', 'CA002', 8, 55.00, 440.00), ('INV0003', '2008-09-20', 'ORD/08-09/0003', '2008-09-15', '2008-09-23', 'BK005', 'Conceptual Physics', NULL, 'CA001', 20, 20.00, 400.00), ('INV0004', '2007-08-30', 'ORD/07-08/0005', '2007-08-22', '2007-08-30', 'BK004', 'Transfer of Heat and Mass', 'English', 'CA002', 15, 35.00, 525.00), ('INV0005', '2007-07-28', 'ORD/07-08/0004', '2007-06-25', '2007-07-30', 'BK001', 'Introduction to Electrodynamics', 'English', 'CA001', 8, 25.00, 200.00), ('INV0006', '2007-09-24', 'ORD/07-08/0007', '2007-09-20', '2007-09-30', 'BK003', 'Guide to Networking', 'Hindi', 'CA003', 20, 45.00, 900.00);

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows 'APEX' and 'SQL Workshop'. The main area is titled 'SQL Commands' with a 'Run' button. The code editor contains several INSERT statements:

```
INSERT INTO purchase (invno, purchase_dt, book_name, pub_name, pub_lang, upc, quantity, unit_price, total_qty, total_amt) VALUES ('INV0001', '2008-07-15', 'Introduction to Electrodynamics', 'English', 'CA001', 15, 75.00, 1125.00);
INSERT INTO purchase (invno, purchase_dt, book_name, pub_name, pub_lang, upc, quantity, unit_price, total_qty, total_amt) VALUES ('INV0002', '2008-08-25', 'Transfer of Heat and Mass', 'English', 'CA002', 8, 55.00, 440.00);
INSERT INTO purchase (invno, purchase_dt, book_name, pub_name, pub_lang, upc, quantity, unit_price, total_qty, total_amt) VALUES ('INV0003', '2008-09-20', 'Conceptual Physics', NULL, 'CA001', 20, 20.00, 400.00);
INSERT INTO purchase (invno, purchase_dt, book_name, pub_name, pub_lang, upc, quantity, unit_price, total_qty, total_amt) VALUES ('INV0004', '2007-08-30', 'Transfer of Heat and Mass', 'English', 'CA002', 15, 35.00, 525.00);
INSERT INTO purchase (invno, purchase_dt, book_name, pub_name, pub_lang, upc, quantity, unit_price, total_qty, total_amt) VALUES ('INV0005', '2007-07-28', 'Introduction to Electrodynamics', 'English', 'CA001', 8, 25.00, 200.00);
INSERT INTO purchase (invno, purchase_dt, book_name, pub_name, pub_lang, upc, quantity, unit_price, total_qty, total_amt) VALUES ('INV0006', '2007-09-24', 'Guide to Networking', 'Hindi', 'CA003', 20, 45.00, 900.00);
```

The results tab shows the message '6 rows(s) inserted.' and a duration of '0.03 seconds'. The bottom status bar includes the URL 'yashaswiniandu@gmail.com', session ID 'yy_dbsn_2025', and copyright information 'Copyright © 1996, 2004, Oracle. All rights reserved.'

76. select * from purchase;

```

APEX SQL Workshop
Schema: WISL_YDBMS205

1 select * from purchase;

```

Results

INVOICE_NO	INVOICE_DT	ORD_NO	ORD_DATE	RECEIVE_DT	BOOK_ID	BOOK_NAME	PUB_LANG	CATE_ID	RECEIVE_QTY	PURCH_PRICE	TOTAL_COST
INV0001	7/15/2008	CR01/08-09/0001	7/15/2008	7/15/2008	BK001	Introduction to Electrodynamics	English	CA001	15	25	375
INV0002	8/25/2008	CR02/08-09/0002	8/25/2008	8/25/2008	BK004	Transfer of Heat and Mass	English	CA002	8	25	200
INV0003	9/20/2008	CR03/08-09/0003	9/20/2008	9/20/2008	BK005	Concepts of Physics	-	CA001	20	20	400
INV0004	8/30/2007	CR04/07-08/0005	8/22/2007	8/30/2007	BK004	Transfer of Heat and Mass	English	CA002	10	25	250
INV0005	7/26/2007	CR05/07-08/0004	6/25/2007	7/20/2007	BK001	Introduction to Electrodynamics	English	CA001	8	25	200
INV0006	9/24/2007	CR06/07-08/0007	9/20/2007	9/20/2007	BK005	Guide to Networking	Hindi	CA005	20	45	900

6 rows returned in 0.05 seconds. Download

Copyright © 1999-2024 Oracle Corporation or its affiliates. Oracle APEX 24.2.0

77. `SELECT SUM(IF(pub_lang = 'English',1,0)) AS English, SUM(IF(pub_lang <> 'English',1,0)) AS "Non English" FROM purchase;`

```

APEX SQL Workshop
Schema: WISL_YDBMS205

1 SELECT
2   SUM(CASE WHEN pub_lang = 'English' THEN 1 ELSE 0 END) AS English,
3   SUM(CASE WHEN pub_lang <> 'English' OR pub_lang IS NULL THEN 1 ELSE 0 END) AS "Non English"
4 FROM purchase;

```

Results

ENGLISH	Non English
4	2

2 rows returned in 0.01 seconds. Download

Copyright © 1999-2024 Oracle Corporation or its affiliates. Oracle APEX 24.2.0

78. `CREATE TABLE publishers (pub_id VARCHAR(10), pub_name VARCHAR(50), pub_city VARCHAR(30), country VARCHAR(30), country_office VARCHAR(30), no_of_branch INT, estd DATE);`

```

APEX SQL Workshop
Schema: WISL_YDBMS205

1 CREATE TABLE publishers (
2   pub_id VARCHAR(10),
3   pub_name VARCHAR(50),
4   pub_city VARCHAR(30),
5   country VARCHAR(30),
6   country_office VARCHAR(30),
7   no_of_branch INT,
8   estd DATE
9 );

```

Results

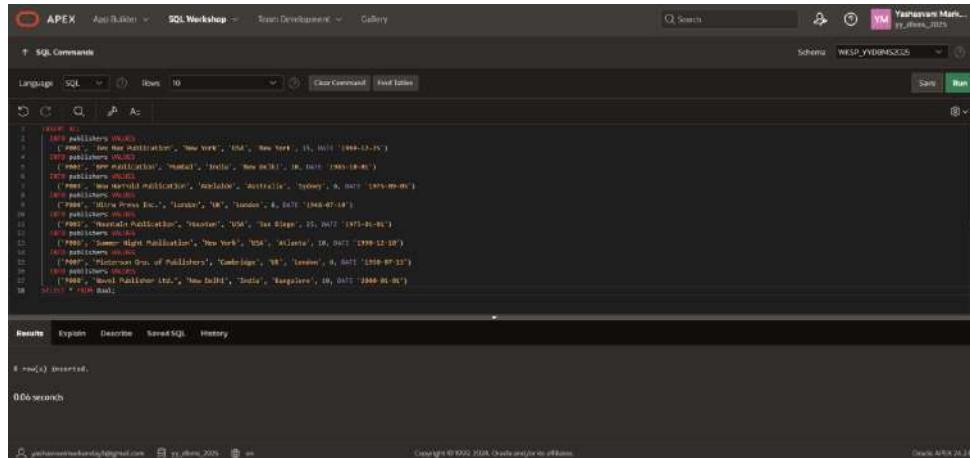
Table created.

0.04 seconds

Copyright © 1999-2024 Oracle Corporation or its affiliates. Oracle APEX 24.2.0

79. `INSERT INTO publishers VALUES ('P001', 'Jex Max Publication', 'New York', 'USA', 'New York', 15, '1969-12-25'), ('P002', 'BPP Publication', 'Mumbai', 'India', 'New Delhi', 10, '1985-10-01'), ('P003', 'New Harrold Publication', 'Adelaide', 'Australia', 'Sydney', 6, '1975-09-05'), ('P004', 'Ultra Press Inc.', 'London', 'UK', 'London', 8, '1948-07-10'), ('P005', 'Mountain Publication', 'Houstan', 'USA', 'Sun`

Diego', 25, '1975-01-01'), ('P006', 'Summer Night Publication', 'New York', 'USA', 'Atlanta', 10, '1990-12-10'), ('P007', 'Pietersen Grp. of Publishers', 'Cambridge', 'UK', 'London', 6, '1950-07-15'), ('P008', 'Novel Publisher Ltd.', 'New Delhi', 'India', 'Bangalore', 10, '2000-01-01');



```
INSERT ALL
INTO publishers VALUES
('P001', 'Fox Max Publication', 'New York', 'USA', 'New York', 15, DATE '1984-12-05')
INTO publishers VALUES
('P002', 'BHP Publication', 'Mumbai', 'India', 'New Delhi', 10, DATE '1995-08-01')
INTO publishers VALUES
('P003', 'New Harold Publication', 'Adelaide', 'Australia', 'Sydney', 6, DATE '1995-01-01')
INTO publishers VALUES
('P004', 'Ultra Print Inc.', 'London', 'UK', 'London', 8, DATE '1995-07-01')
INTO publishers VALUES
('P005', 'Mountain Publication', 'Honolulu', 'USA', 'San Diego', 25, DATE '1975-01-01')
INTO publishers VALUES
('P006', 'Summer Night Publication', 'New York', 'USA', 'Atlanta', 10, DATE '1990-12-10')
INTO publishers VALUES
('P007', 'Pietersen Grp. of Publishers', 'Cambridge', 'UK', 'London', 6, DATE '1950-07-15')
INTO publishers VALUES
('P008', 'Novel Publisher Ltd.', 'New Delhi', 'India', 'Bangalore', 10, DATE '2000-01-01')
SELECT * FROM dual;
```

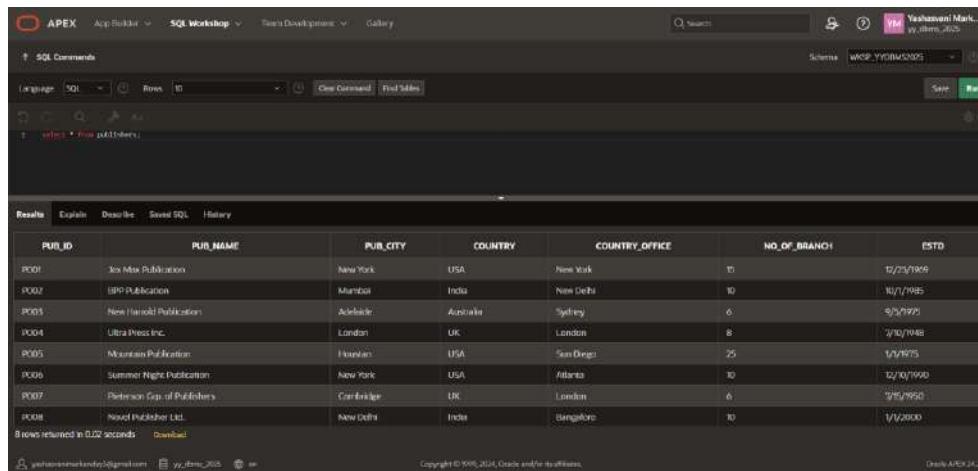
Results Explain Describe Saved SQL History

8 rows(s) inserted.

0.00 seconds

sathishmarkanday@gmail.com 2020-08-08 10:45 AM Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 21.2

80. select * from publishers;



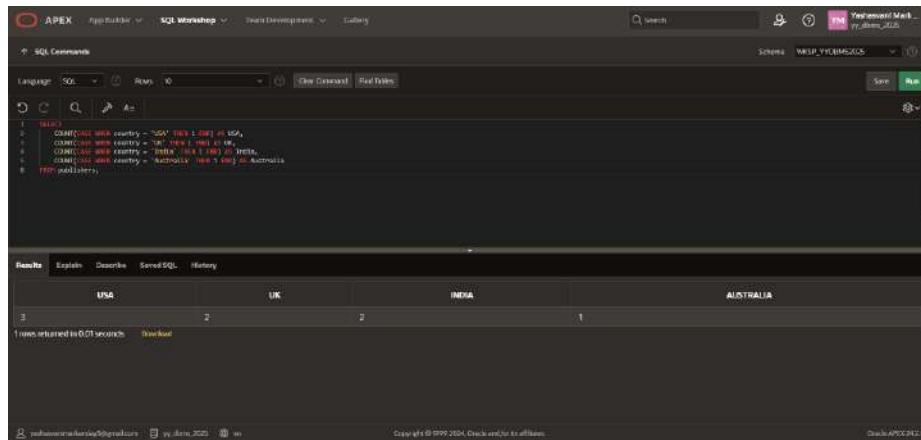
```
SELECT * FROM publishers;
```

PUB_ID	PUB_NAME	PUB_CITY	COUNTRY	COUNTRY_OFFICE	NO_OF_BRANCH	ESTD
P001	Fox Max Publication	New York	USA	New York	15	12/25/1985
P002	BHP Publication	Mumbai	India	New Delhi	10	10/1/1995
P003	New Harold Publication	Adelaide	Australia	Sydney	6	9/5/1995
P004	Ultra Print Inc.	London	UK	London	8	7/10/1998
P005	Mountain Publication	Honolulu	USA	San Diego	25	1/1/1975
P006	Summer Night Publication	New York	USA	Atlanta	10	12/25/1990
P007	Pietersen Grp. of Publishers	Cambridge	UK	London	6	7/15/1990
P008	Novel Publisher Ltd.	New Delhi	India	Bangalore	10	1/1/2000

8 rows returned in 0.02 seconds Download

sathishmarkanday@gmail.com 2020-08-08 10:45 AM Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 21.2

81. SELECT COUNT(IF(country = 'USA',1,NULL)) USA, COUNT(IF(country = 'UK',1,NULL)) UK, COUNT(IF(country = 'India',1,NULL)) India, COUNT(IF(country = 'Australia',1,NULL)) Australia FROM publishers; Another way to achieve the similar result you can use the GROUP BY clause and the COUNT function without using the IF function, the display report is quite different.



```
SELECT
    COUNT(IF(country = 'USA',1,NULL)) USA,
    COUNT(IF(country = 'UK',1,NULL)) UK,
    COUNT(IF(country = 'India',1,NULL)) INDIA,
    COUNT(IF(country = 'Australia',1,NULL)) AUSTRALIA
FROM publishers;
```

USA	UK	INDIA	AUSTRALIA
3	2	2	1

1 rows returned in 0.01 seconds Download

sathishmarkanday@gmail.com 2020-08-08 10:45 AM Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 21.2

82. SELECT country, COUNT(country) FROM publishers GROUP BY country;

The screenshot shows the AWS APACHE APACHE SQL Workshop interface. At the top, there are tabs for 'APACHE', 'API Router', 'SQL Workshop' (which is selected), 'Frontend Development', and 'CloudFormation'. On the right side, there's a user profile for 'Vedhanam M...' with a timestamp '9 Jun 2025'. The main area is titled 'SQL Commands' with a 'Results' tab selected. The results table has two columns: 'COUNTRY' and 'COUNT(COUNTRY)'. The data is as follows:

COUNTRY	COUNT(COUNTRY)
USA	3
Australia	1
UK	2
India	2

Below the table, it says '4 rows returned in 0.00 seconds'. There are also tabs for 'Explain', 'Describe', 'Saved SQL', and 'History'.



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Program Name: B. Tech CS

Semester: III

Name: Yashasvani Markanday

SAP ID: 590011386

Batch: B2

Subject: DBMS

LAB - 7

1.)

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab contains the following query:

```
1 SELECT * FROM table_A
2 INNER JOIN table_B2
3 ON table_A.A=table_B2.A;
```

The Results tab displays the output of the query:

A	NAME	B	A	DESCRIPTION
2	Mary	105	2	Pen
5	Eve	106	5	Laptop
3	Alex	104	3	Notebook

3 rows returned in 0.02 seconds

2.)

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab contains the following query:

```
1 SELECT book_mast.book_id,book_mast.book_name,cate_descrip
2 FROM book_mast
3 INNER JOIN category1
4 ON book_mast.cate_id=category1.cate_id
5 WHERE book_mast.pub_lang='English';
```

The Results tab displays the output of the query:

BOOK_ID	BOOK_NAME	CATE_DESCRIP
BK001	Introduction to Electrodynamics	science
BK002	Understanding of Steel Construction	technology
BK004	Transfer of Heat and Mass	technology
BK009	Mental Health Nursing	medical
BK010	Fundamentals of Thermodynamics	technology
BK012	The Nature of World	nature

6 rows returned in 0.02 seconds

3.)

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. The SQL Commands tab contains the following query:

```
1 SELECT bk.book_id, bk.book_name, bk.pub_id, ca.cate_descrip, bk.pub_lang
2 FROM book_mast bk
3 INNER JOIN category1 ca
4 ON bk.cate_id = ca.cate_id
5 WHERE bk.pub_lang = 'English'
6 AND bk.pub_id <> 'P004';
```

The Results tab displays the following table:

BOOK_ID	BOOK_NAME	PUB_ID	CATE_DESCRIP	PUB_LANG
BK001	Introduction to Electrodynamics	P005	science	English
BK002	Understanding of Steel Construction	P001	technology	English
BK010	Fundamentals of Thermodynamics	P007	technology	English
BK012	The Nature of World	P008	nature	English
BK009	Mental Health Nursing	P006	medical	English

5 rows returned in 0.01 seconds

4.)

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. The SQL Commands tab contains the following query:

```
1 SELECT doctors.docid, doctors.dname,
2 specialize.desc1,timeschedule.tday,timeschedule.sit_time
3 FROM doctors doctors
4 INNER JOIN specialize specialize
5 ON doctors.docid=specialize.docid
6 INNER JOIN timeschedule timeschedule
7 ON doctors.docid-timeschedule.docid
8 WHERE doctors.docid=3 AND timeschedule.tday='WED';
```

The Results tab displays the following table:

DOCID	DNAME	DESC1	TDAY	SIT_TIME
1	AVARMA	special1	WED	08:00

1 rows returned in 0.05 seconds

5.)

APEX App Builder SQL Workshop Team Development Gallery Search

SQL Commands Schema WKSP_YYDBMS2025

Language SQL Rows 10 Clear Command Find Tables Save Run

```
1 SELECT * FROM table_A
2 LEFT JOIN table_B2
3 ON table_A.A=table_B2.A;
```

Results Explain Describe Saved SQL History

A	NAME	B	A	DESCRIPTION
2	Mary	105	2	Pen
5	Eve	106	5	Laptop
3	Alex	104	3	Notebook
1	John	-	-	-
4	Sophia	-	-	-

5 rows returned in 0.03 seconds Download

yashasvanimarkanday5@gmail.com Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.8 yy_dbms_2025 en

6.)

APEX App Builder SQL Workshop Team Development Gallery Search

SQL Commands Schema WKSP_YYDBMS2025

Language SQL Rows 10 Clear Command Find Tables Save Run

```
1 SELECT bk1.book_name,bk1.isbn_no,bk1.book_price,bk1.pub_lang
2 FROM book_mast bk1
3 LEFT JOIN book_mast bk2 ON bk1.book_price<bk2.book_price
4 WHERE bk2.pub_lang='German';
```

Results Explain Describe Saved SQL History

BOOK_NAME	ISBN_NO	BOOK_PRICE	PUB_LANG
Conceptual Physics	0000978005	145	-
Anatomy & Physiology	0000978015	135	Hindi
Fundamentals of Heat	0000978006	112	German
Understanding of Steel Construction	0000978002	105.5	English
Environment a Sustainable Future	0000978015	100	German
Introduction to Electrodynamics	0000978001	85	English
Concepts in Health	0000978014	80	-
Advanced 3d Graphics	0000978007	58	Hindi
Networks and Telecommunications	0000978016	45	French
Understanding of Steel Construction	0000978002	105.5	English

yashasvanimarkanday5@gmail.com Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.8 yy_dbms_2025 en

7.)

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. The SQL command entered is:

```
1 SELECT * FROM table_A
2 RIGHT JOIN table_B2
3 ON table_A.A=table_B2.A;
```

The results show three rows returned in 0.01 seconds:

A	NAME	B	A	DESCRIPTION
2	Mary	105	2	Pen
5	Eve	106	5	Laptop
3	Alex	104	3	Notebook

Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.8.

8.)

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. The SQL command entered is:

```
1 SELECT table112.t_id,table112.bval1,table112.bval2,
2 table111.t_id,table111.avali
3 FROM table112
4 RIGHT JOIN table111
5 ON table112.bval1=table111.avali;
```

The results show four rows returned in 0.01 seconds:

T_ID	BVAL1	BVAL2	T_ID	AVAL1
701	405	16	1	405
709	401	12	2	401
-	-	-	4	400
-	-	-	3	200

Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.8.

9.)

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. The SQL command entered is:

```
1 SELECT table112.t_id,table112.bval1,table112.bval2,
2 table111.t_id,table111.avali
3 FROM table112
4 RIGHT OUTER JOIN table111
5 ON table112.bval1=table111.avali;
```

The results table has columns T_ID, BVAL1, BVAL2, T_ID, and AVAL1. The data is:

T_ID	BVAL1	BVAL2	T_ID	AVAL1
701	405	16	1	405
709	401	12	2	401
-	-	-	4	400
-	-	-	5	200

4 rows returned in 0.03 seconds

T_ID	CVAL1	AVAL1
3	17	200
2	12	401
1	16	405
4	-	400

Copyright © 1999, 2024, Oracle and/or its affiliates.

Oracle APEX 24.2.8

10.)

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. The SQL command entered is:

```
1 SELECT t_id,table113.cval1,table111.avali
2 FROM table113
3 RIGHT OUTER JOIN table111
4 USING (t_id);
```

The results table has columns T_ID, CVAL1, and AVAL1. The data is:

T_ID	CVAL1	AVAL1
3	17	200
2	12	401
1	16	405
4	-	400

4 rows returned in 0.01 seconds

T_ID	CVAL1	AVAL1
3	17	200
2	12	401
1	16	405
4	-	400

Copyright © 1999, 2024, Oracle and/or its affiliates.

Oracle APEX 24.2.8

11.)

```

SELECT * FROM table_A9
INNER JOIN table_B3
ON B.A = B.A;

```

A	B	NAME
5	105	Chair
2	106	Laptop
3	107	Notebook

3 rows returned in 0.05 seconds

As there is not straight join in oracle so inner join is equivalent to straight join only!

12.)

```

SELECT table112.t_id,table112.bval1,table112.bval2,table111.t_id,table111.avl1
FROM table112
FULL OUTER JOIN table111
ON table112.t_id = table111.t_id;

```

T_ID	BVAL1	BVAL2	T_ID	AVL1
-	-	-	1	405
-	-	-	2	401
-	-	-	3	200
-	-	-	4	400
709	401	12	-	-
706	403	13	-	-
701	405	16	-	-
704	409	14	-	-

8 rows returned in 0.01 seconds

13.)

APEX App Builder **SQL Workshop** Team Development **Gallery** **Search**

↑ SQL Commands Schema: WKSP_YYDBMS2025

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```
1 SELECT table113.t_id,
2      table113.cval1,
3      table111.t_id,
4      table111.aval1
5 FROM table113
6 INNER JOIN table111
7      ON table113.t_id = table111.t_id;
```

Results Explain Describe Saved SQL History

T_ID	CVAL1	T_ID	AVAL1
1	16	1	405
2	12	2	401
3	17	3	200

3 rows returned in 0.00 seconds [Download](#)

yashasvanimarkanday3@gmail.com Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.8

14.)

APEX App Builder **SQL Workshop** Team Development **Gallery** **Search**

↑ SQL Commands Schema: WKSP_YYDBMS2025

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```
1 SELECT * FROM table_A
2 CROSS JOIN table_B2;
```

Results Explain Describe Saved SQL History

A	NAME	B	A	DESCRIPTION
3	Alex	105	2	Pen
4	Sophia	105	2	Pen
2	Mary	105	2	Pen
5	Eve	105	2	Pen
1	John	105	2	Pen
3	Alex	106	5	Laptop
4	Sophia	106	5	Laptop
2	Mary	106	5	Laptop
5	Eve	106	5	Laptop
1	John	106	5	Laptop

More than 10 rows available. Increase rows selector to view more rows.

yashasvanimarkanday3@gmail.com Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.8

15.)

APEX SQL Workshop

```
1 SELECT table112.t_id,table112.bval1,table112.bval2,
2      table111.t_id,table111.avall
3 FROM table112
4 CROSS JOIN table111;
```

T_ID	BVAL1	BVAL2	T_ID	AVALL
701	405	16	2	401
701	405	16	3	200
701	405	16	4	400
704	409	14	1	405
704	409	14	2	401
704	409	14	3	200
704	409	14	4	400
706	403	13	1	405
706	403	13	2	401

More than 10 rows available. Increase rows selector to view more rows.

yashavanimarkanday3@gmail.com Copyright © 1999, 2024, Oracle and/or its affiliates. yy_dbms_2025 en Oracle APEX 24.2.8

16.)

APEX SQL Workshop

```
1 SELECT * FROM table111
2 LEFT JOIN(table112 CROSS JOIN table113)
3 ON table111.t_id=table113.t_id;
```

T_ID	AVALL	T_ID	BVAL1	BVAL2	T_ID	CVAL1
3	200	701	405	16	3	17
2	401	701	405	16	2	12
1	405	701	405	16	1	16
3	200	704	409	14	3	17
2	401	704	409	14	2	12
1	405	704	409	14	1	16
3	200	706	403	13	3	17
2	401	706	403	13	2	12
1	405	706	403	13	1	16
3	200	709	401	12	3	17

More than 10 rows available. Increase rows selector to view more rows.

yashavanimarkanday3@gmail.com Copyright © 1999, 2024, Oracle and/or its affiliates. yy_dbms_2025 en Oracle APEX 24.2.8

17.)

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. The SQL command window contains the following code:

```
1 SELECT table111.*  
2 FROM table111  
3 CROSS JOIN table113  
4 WHERE table111.t_id=table113.t_id;
```

The results section displays a table with four columns: T_ID, AVAL1, T_ID, and CVAL1. The data is as follows:

T_ID	AVAL1	T_ID	CVAL1
3	200	3	17
2	401	2	12
1	405	1	16

3 rows returned in 0.00 seconds

18.)

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP_YYDBMS2025. The SQL command window contains the following code:

```
1 SELECT * FROM table_A  
2 NATURAL JOIN table_B;
```

The results section displays a table with four columns: A, NAME, B, and DESCRIPTION. The data is as follows:

A	NAME	B	DESCRIPTION
2	Mary	105	Pen
5	Eve	106	Laptop
3	Alex	104	Notebook

3 rows returned in 0.03 seconds

19.)

APEX App Builder SQL Workshop Team Development Gallery Search

SQL Commands Schema WKSP_YYDBMS2025

Language SQL Rows 10 Clear Command Find Tables Save Run

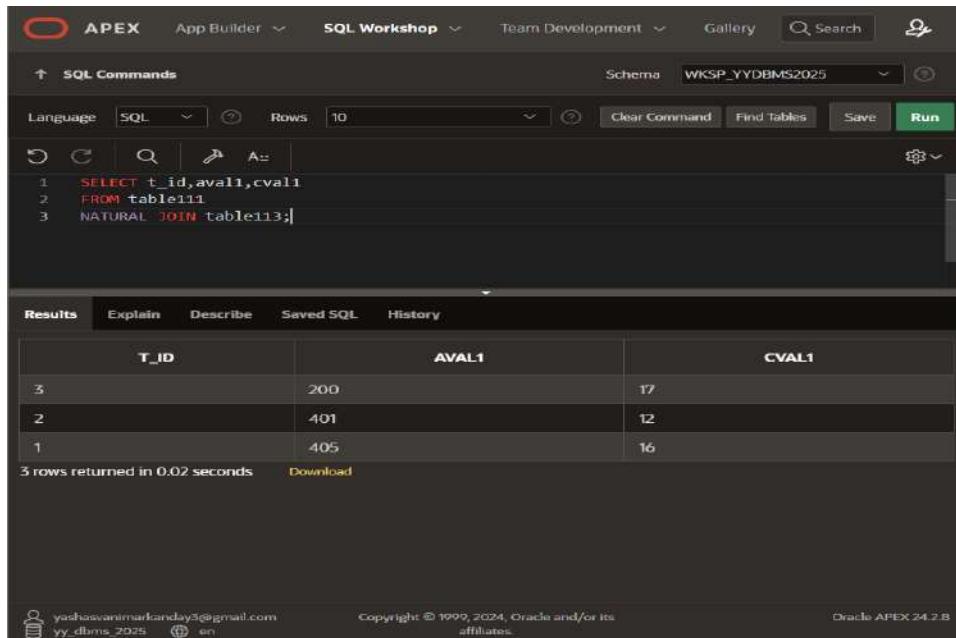
```
1 SELECT t_id,aval1,cval1
2 FROM table111
3 NATURAL JOIN table113;
```

Results Explain Describe Saved SQL History

T_ID	AVAL1	CVAL1
3	200	17
2	401	12
1	405	16

3 rows returned in 0.02 seconds Download

yashasvanimarkanday5@gmail.com Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.8



20.)

APEX App Builder SQL Workshop Team Development Gallery Search

SQL Commands Schema WKSP_YYDBMS2025

Language SQL Rows 10 Clear Command Find Tables Save Run

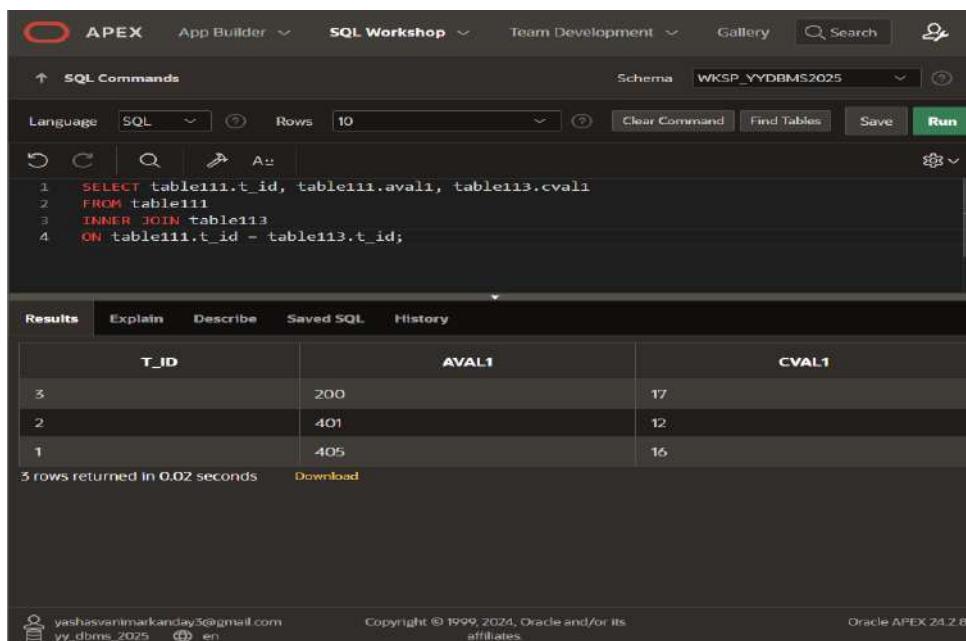
```
1 SELECT table111.t_id, table111.avali, table113.cval1
2 FROM table111
3 INNER JOIN table113
4 ON table111.t_id = table113.t_id;
```

Results Explain Describe Saved SQL History

T_ID	AVAL1	CVAL1
3	200	17
2	401	12
1	405	16

3 rows returned in 0.02 seconds Download

yashasvanimarkanday5@gmail.com Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.2.8



21).)

APEX App Builder SQL Workshop Team Development Gallery Search

↑ SQL Commands Schema WKSP_YYDBMS2025

Language SQL Rows 10 Clear Command Find Tables Save Run

```
1 SELECT t_id,aval1,cval1
2 FROM table111
3 NATURAL JOIN table113
4 WHERE table111.aval1>200;
```

Results Explain Describe Saved SQL History

T_ID	AVAL1	CVAL1
1	405	16
2	401	12

2 rows returned in 0.02 seconds Download

yashasvanimarkanday5@gmail.com Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.8 yy_dbms_2025 en

22.)

APEX App Builder SQL Workshop Team Development Gallery Search

↑ SQL Commands Schema WKSP_YYDBMS2025

Language SQL Rows 10 Clear Command Find Tables Save Run

```
1 SELECT id, aval1, cval1, dval1
2 FROM table111
3 NATURAL JOIN table113
4 NATURAL JOIN table114
5 WHERE aval1 > 200;
```

Results Explain Describe Saved SQL History

ID	AVAL1	CVAL1	DVAL1
1	405	16	500
2	405	16	300
1	401	12	500
2	401	12	300

4 rows returned in 0.02 seconds Download

yashasvanimarkanday5@gmail.com Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.8 yy_dbms_2025 en



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Program Name: B. Tech CS

Semester: III

Name: Yashasvani Markanday

SAP ID: 590011386

Batch: B2

DBMS

LAB-8

1.

The screenshot shows the Oracle SQL Workshop interface with the BANNER table selected. The table has four columns: BANNER_ID, BANNER_TYPE, BANNER_FULL, and BANNER_LEGACY. The BANNER_TYPE column contains values such as 'BANNER', 'BANNER_EEE', 'BANNER_EE', 'BANNER_EE_Designer', 'BANNER_EE_Designer_Production', 'BANNER_EE_Designer_Production_V1', and 'BANNER_EE_Designer_Production_V2'. The BANNER_FULL column contains descriptions of the banner types. The BANNER_LEGACY column contains 'COM_ID' values. The results pane shows the first few rows of the table.

2.

The screenshot shows the Oracle SQL Workshop interface with the PRIVILEGE table selected. The table has three columns: PRIVILEGE, NAME, and PROPERTY. The PRIVILEGE column lists various database privileges like ALTERSYSTEM, CREATESESSION, and UNLIMITEDTABLESPACE. The NAME column lists the names of the privileges, and the PROPERTY column indicates if they are granted or not. The results pane shows the first few rows of the table.

3.

The screenshot shows the Oracle SQL Workshop interface with the DBAUSER table selected. The table has one column: USERNAME. It lists various database users including ACCESSPRIMERA, AD00166400, AD10150008, AD111996, AD2145, AD33, AD43, AD44, AD45, and AD50. The results pane shows the first few rows of the table.

4.

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected under 'Application'. Below it, 'SQL Workshop' is chosen from the 'Development' dropdown. The main area is titled 'SQL Commands' with tabs for 'Open Command' and 'Find Table'. A search bar is at the top right. The schema dropdown shows 'SCOTT'. The results pane is empty. Below the tabs, there's a table structure for 'LOGS' with columns: ID (NUMBER), LOG_DATE (DATE), LOG_TYPE (VARCHAR2(50)), LOG_MESSAGE (VARCHAR2(100)), and LOG_SOURCE (VARCHAR2(50)). The 'LOG_TYPE' column has a primary key constraint.

5.

The screenshot shows the Oracle SQL Workshop interface. The 'SQL Commands' tab is active. The code editor contains the following SQL statement:

```
1  INSERT INTO LOGS (LOG_ID, LOG_DATE, LOG_TYPE, LOG_MESSAGE, LOG_SOURCE)
2  VALUES ('LOG001', SYSDATE, 'INFO', 'System started up.', 'Database');
3  INSERT INTO LOGS (LOG_ID, LOG_DATE, LOG_TYPE, LOG_MESSAGE, LOG_SOURCE)
4  VALUES ('LOG002', SYSDATE, 'INFO', 'User logged in.', 'Database');
5  INSERT INTO LOGS (LOG_ID, LOG_DATE, LOG_TYPE, LOG_MESSAGE, LOG_SOURCE)
6  VALUES ('LOG003', SYSDATE, 'INFO', 'User logged out.', 'Database');
```

The results pane shows the message '3 rows inserted.' and '0 errors'.

6.

The screenshot shows the Oracle SQL Workshop interface. The 'Results' tab is active. The results pane displays the data from the 'LOGS' table:

LOG_ID	LOG_DATE	LOG_TYPE	LOG_MESSAGE	LOG_SOURCE
LOG001	UNSPECIFIED	INFO	System started up.	Database
LOG002	2023-09-25 10:00:00	INFO	User logged in.	Database
LOG003	2023-09-25 10:05:00	INFO	User logged out.	Database

The message '3 row(s) returned in 0.07 seconds' is shown at the bottom.

7.

The screenshot shows the Oracle SQL Workshop interface. The 'SQL Commands' tab is active. The code editor contains the following SQL statement:

```
CREATE TABLE ITEMS (
    ITEM_ID NUMBER,
    ITEM_NAME VARCHAR2(50),
    ITEM_DESCRIPTION VARCHAR2(100),
    ITEM_PRICE NUMBER(10, 2),
    ITEM_STOCK NUMBER
);
```

The results pane shows the message 'Table created.' and '0 rows'.

8.

A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX" and "SQL Workshop". The main area has a "SQL Commands" tab selected. A query is being run: "SELECT * FROM my_v2;". The results pane shows a table with three rows:

USER_ID	PASSWORD	NAME
scott	tiger	Scott
hrpt014	hrpt014	Prash
claus020	claus020	Claus

Below the table, it says "3 rows returned in 0.03 seconds".

9.

10.

A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX" and "SQL Workshop". The main area has a "SQL Commands" tab selected. A query is being run: "SELECT * FROM my_v2;". The results pane shows a table with three rows:

USER_ID	PASSWORD	NAME
scott	tiger	Scott
hrpt014	hrpt014	Prash
claus020	claus020	Claus

Below the table, it says "3 rows returned in 0.03 seconds".

11.

A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX" and "SQL Workshop". The main area has a "SQL Commands" tab selected. A query is being run: "CREATE TABLE button (aut_id VARCHAR(20) PRIMARY KEY, aut_name VARCHAR(50), country VARCHAR(10), home_city VARCHAR(10));". The results pane shows the message "Table created." and "0.00 seconds".

12.

A screenshot of the Oracle APEX SQL Workshop interface. The title bar shows 'APEX - App Builder - SQL Workshop - Team Development - Galaxy'. The main area is titled 'SQL Commands' with a dropdown menu set to 'SQL'. Below it is a code editor containing the following SQL code:

```
1 CREATE OR REPLACE VIEW vw_author AS
2   SELECT * FROM author;
3   UNION ALL
4   SELECT * FROM author WHERE country = 'USA';
5   UNION ALL
6   SELECT * FROM author WHERE country = 'UK';
7   UNION ALL
8   SELECT * FROM author WHERE country = 'India';
9
10  SELECT * FROM dual;
```

The 'Results' tab is selected at the bottom. The output shows:

```
1 rows inserted.
0.05 seconds
```

13.

A screenshot of the Oracle APEX SQL Workshop interface. The title bar shows 'APEX - App Builder - SQL Workshop - Team Development - Galaxy'. The main area is titled 'SQL Commands' with a dropdown menu set to 'SQL'. Below it is a code editor containing the following SQL code:

```
1 CREATE OR REPLACE VIEW vw_view_author AS
2   SELECT * FROM author WHERE country='USA';
```

The 'Results' tab is selected at the bottom. The output shows:

```
View created.
0.09 seconds
```

14.

A screenshot of the Oracle APEX SQL Workshop interface. The title bar shows 'APEX - App Builder - SQL Workshop - Team Development - Galaxy'. The main area is titled 'SQL Commands' with a dropdown menu set to 'SQL'. Below it is a code editor containing the following SQL code:

```
1 SELECT * FROM vw_view_author;
```

The 'Results' tab is selected at the bottom. The output shows:

AU_ID	AU_NAME	COUNTRY	HOME_CITY
AU002	John Smith	USA	New York
AU005	Ruth Anne	USA	Honda

2 rows returned in 0.07 seconds. Inserted.

15.

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL Rows: 10 Clear Comment Test Table

```
1 CREATE UNIQ_publisher { pub_id VARCHAR(10) PRIMARY KEY, pub_name VARCHAR(50), pub_city VARCHAR(50), country VARCHAR(50), country_office VARCHAR(50), no_of_branch INT, estd DATE };
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

Copyright © 1996-2010 Oracle. All rights reserved.

16.

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL Rows: 10 Clear Comment Test Table

```
1 INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office, no_of_branch, estd)
2 VALUES ('P001', 'Pragyan Books', 'Delhi', 'India', 'New Delhi', 10, DATE '2000-12-25');
3 INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office, no_of_branch, estd)
4 VALUES ('P002', 'Pearson Education', 'London', 'UK', 'Cambridge', 20, DATE '1995-05-10');
5 INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office, no_of_branch, estd)
6 VALUES ('P003', 'Cengage Learning', 'Bengaluru', 'India', 'California', 15, DATE '2000-05-15');
7 INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office, no_of_branch, estd)
8 VALUES ('P004', 'Oxford University Press', 'Oxford', 'UK', 'London', 10, DATE '1950-06-20');
9 INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office, no_of_branch, estd)
10 VALUES ('P005', 'Springer Publications', 'Berlin', 'Germany', 'Berlin', 12, DATE '1990-01-11');
11 INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office, no_of_branch, estd)
12 VALUES ('P006', 'Wiley India Pvt Ltd', 'Mumbai', 'India', 'Mumbai', 15, DATE '2000-01-01');
13 INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office, no_of_branch, estd)
14 VALUES ('P007', 'Wiley Online', 'Toronto', 'Canada', 'Vancouver', 10, DATE '1970-07-01');
15 INSERT INTO publisher (pub_id, pub_name, pub_city, country, country_office, no_of_branch, estd)
16 VALUES ('P008', 'Wiley Publisher Ltd.', 'New Delhi', 'India', 'Delhi', 10, DATE '2000-01-01');
```

Results Explain Describe Saved SQL History

8 rows inserted.

0.00 seconds

Copyright © 1996-2010 Oracle. All rights reserved.

17.

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL Rows: 10 Clear Comment Test Table

```
1 CREATE VIEW view_publisher AS SELECT pub_name, pub_city, country FROM publisher WHERE (country='USA' AND pub_city='New York') OR (country='India' AND pub_city='Mumbai');
```

Results Explain Describe Saved SQL History

View created.

0.00 seconds

Copyright © 1996-2010 Oracle. All rights reserved.

18.

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL | Run | Save | Run | Close Command | Find Table

```
3 SELECT * FROM view_publisher;
```

Results | Explain | Describe | Saved SQL | History

PUBLISHER	PUBL_CITY	COUNTRY
ABC Max Publication	New York	USA
Paragon Books	New York	USA

From executed in 0.03 seconds. [Download](#)

Copyright © 2019, Oracle Database 19c

19.

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL | Run | Save | Run | Close Command | Find Table

```
3 CREATE TABLE book_mast_2 (book_id NUMBER(10) PRIMARY KEY, book_name VARCHAR(100), isbn_no VARCHAR(20), cat_id VARCHAR(10), aut_id VARCHAR(10), pub_id VARCHAR(10), st_of_publ NUMBER(10), pub_lang VARCHAR(20), no_pages NUMBER(10));
```

Results | Explain | Describe | Saved SQL | History

Table created.

0.25 seconds

Copyright © 2019, Oracle Database 19c

20.

APEX Application - SQL Workshop - Team Development - Gallery

SQL Commands

Language: SQL | Run | Save | Run | Close Command | Find Table

```
3 INSERT INTO book_mast_2 (book_id, book_name, isbn_no, cat_id, aut_id, pub_id, st_of_publ, pub_lang, no_pages, book_price)
VALUES ('978-1-2345-6789-1', 'Computer Organization and Architecture', '978-1-2345-6789-1', 'C001', 'A001', 'P001', 10, 'ENGLISH', 500, 45.99)
4 INSERT INTO book_mast_2 (book_id, book_name, isbn_no, cat_id, aut_id, pub_id, st_of_publ, pub_lang, no_pages, book_price)
VALUES ('978-1-2322-2222-7', 'C002', 'A002', 'P002', 10, 'ENGLISH', 600, 52.50)
5 INSERT INTO book_mast_2 (book_id, book_name, isbn_no, cat_id, aut_id, pub_id, st_of_publ, pub_lang, no_pages, book_price)
VALUES ('978-1-3335-3335-3', 'C003', 'A003', 'P003', 10, 'ENGLISH', 450, 40.00)
6 INSERT INTO book_mast_2 (book_id, book_name, isbn_no, cat_id, aut_id, pub_id, st_of_publ, pub_lang, no_pages, book_price)
VALUES ('978-1-4444-4444-4', 'C004', 'A004', 'P004', 10, 'ENGLISH', 550, 48.75)
7 INSERT INTO book_mast_2 (book_id, book_name, isbn_no, cat_id, aut_id, pub_id, st_of_publ, pub_lang, no_pages, book_price)
VALUES ('978-3-593-59355-5', 'C005', 'A005', 'P005', 10, 'ENGLISH', 350, 42.35)
8 INSERT INTO book_mast_2 (book_id, book_name, isbn_no, cat_id, aut_id, pub_id, st_of_publ, pub_lang, no_pages, book_price)
VALUES ('978-1-7777-7777-7', 'C006', 'A006', 'P006', 10, 'ENGLISH', 720, 50.00)
9 INSERT INTO book_mast_2 (book_id, book_name, isbn_no, cat_id, aut_id, pub_id, st_of_publ, pub_lang, no_pages, book_price)
VALUES ('978-1-888-8888-8', 'C007', 'A007', 'P007', 10, 'ENGLISH', 650, 60.75)
```

Results | Explain | Describe | Saved SQL | History

15 rows(s) inserted.

0.00 seconds

Copyright © 2019, Oracle Database 19c

21.

21.

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a user profile for 'Yashavani Mark...' and the schema 'WSP_YDBMS2025'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 CREATE VIEW view_bookcast AS
2 SELECT pub_lang, COUNT(*) AS TOTAL_books
3 FROM book_mst_2
4 GROUP BY pub_lang;
```

The results section shows a single row with the value 'View created.' and a duration of '0.05 seconds'. The URL at the bottom is [https://yashavanimarks2025.oracleapex.com/pls/apex/f?p=100:1:1063425:::.](https://yashavanimarks2025.oracleapex.com/pls/apex/f?p=100:1:1063425:::)

22.

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a user profile for 'Yashavani Mark...' and the schema 'WSP_YDBMS2025'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 SELECT * FROM view_bookcast;
```

The results section displays a table with two columns: 'PUB_LANG' and 'TOTAL_BOOKS'. The data is:

PUB_LANG	TOTAL_BOOKS
English	5

1 rows returned in 0.01 seconds. The URL at the bottom is [https://yashavanimarks2025.oracleapex.com/pls/apex/f?p=100:1:1063425:::.](https://yashavanimarks2025.oracleapex.com/pls/apex/f?p=100:1:1063425:::)

23.

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a user profile for 'Yashavani Mark...' and the schema 'WSP_YDBMS2025'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 CREATE VIEW view_bookcast AS
2 SELECT pub_lang, COUNT(*) AS total_books
3 FROM book_mst_2
4 GROUP BY pub_lang
5 ORDER BY pub_lang;
```

The results section shows a single row with the value 'View created.' and a duration of '0.02 seconds'. The URL at the bottom is [https://yashavanimarks2025.oracleapex.com/pls/apex/f?p=100:1:1063425:::.](https://yashavanimarks2025.oracleapex.com/pls/apex/f?p=100:1:1063425:::)

24.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command First Table Save Run

```
1 SELECT * FROM view_booksmt1;
```

Results Explain Describe Saved SQL History

PUB_LANG	TOTAL_BOOKS
English	6
French	2
German	3
Hindi	3
-	2

5 rows returned in 0.05 seconds Download

Copyright © 1999-2024 Oracle and/or its affiliates. Oracle APEX 24.1.0

25.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command First Table Save Run

```
1 CREATE VIEW view_booksmt2 AS
2 SELECT *
3 FROM book_mast
4 WHERE book_name BETWEEN 'A' AND 'G'
5 AND no_page IN (165, 250, 350, 400, 510);
```

Results Explain Describe Saved SQL History

View created.

0.05 seconds

Copyright © 1999-2024 Oracle and/or its affiliates. Oracle APEX 24.1.0

26.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command First Table Save Run

```
1 SELECT * FROM view_booksmt2;
```

Results Explain Describe Saved SQL History

BOOK_ID	BOOK_NAME	ISBN_NO	CATE_ID	AUT_ID	PUB_ID	DT_OF_PUB	PUB_LANG	NO_PAGE	BOOK_PRICE
BK005	Environment & Sustainable Future	0000978015	CA004	AU005	PO008	17/09/2005	German	355	100

1 rows returned in 0.02 seconds Download

Copyright © 1999-2024 Oracle and/or its affiliates. Oracle APEX 24.1.0

27.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

```
1 CREATE VIEW view_authors AS
2 SELECT *
3 FROM authors
4 WHERE aut_name NOT LIKE '%TX'
5 OR aut_name NOT LIKE '%W%'
```

Results Explain Describe Saved SQL History

View created.

0.03 seconds

Yashavant Mark... YY_Books_2025 Schema: WSP_YYBOOKS2025

28.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

```
1 SELECT * FROM view_authors;
```

Results Explain Describe Saved SQL History

AUT_ID	AUT_NAME	COUNTRY	HOME_CITY
AUT002	John Smith	USA	New York
AUT005	Butter Andis	USA	Florida

7 rows returned in 0.01 seconds Download

Yashavant Mark... YY_Books_2025 Schema: WSP_YYBOOKS2025

29.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

```
1 CREATE TABLE purchase (
2 invoice_no NUMBER(15) PRIMARY KEY,
3 invoice_dt DATE,
4 ord_no NUMBER(10),
5 ord_date DATE,
6 receive_dt DATE,
7 book_id NUMBER(10),
8 book_name VARCHAR2(100),
9 pub_lang VARCHAR2(10),
10 catg VARCHAR2(10),
11 receive_qty NUMBER(7),
12 purch_price NUMBER(9,2),
13 total_cost NUMBER(10,2)
14 );
```

Results Explain Describe Saved SQL History

Table created.

0.06 seconds

Yashavant Mark... YY_Books_2025 Schema: WSP_YYBOOKS2025

30.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```

1. INSERT ALL
2. INTO purchase VALUES ('INV0001', DATE '2008-07-15', '080001', DATE '2008-07-20', '08001', 'Introduction to Electrodynamics', 'English', 'CA001', 10, 400.00, 4000.00)
3. INTO purchase VALUES ('INV0002', DATE '2007-09-14', '080002', DATE '2007-09-18', '08002', 'Fundamentals of Thermodynamics', 'English', 'CA002', 10, 300.00, 3000.00)
4. INTO purchase VALUES ('INV0003', DATE '2008-08-10', '080003', DATE '2008-08-15', '08003', 'Principles of Electronics', 'English', 'CA003', 12, 250.00, 3000.00)
5. INTO purchase VALUES ('INV0004', DATE '2008-08-11', '080004', DATE '2008-08-15', '08004', 'Concepts Networks', 'English', 'CA004', 15, 500.00, 5000.00)
6. INTO purchase VALUES ('INV0005', DATE '2011-01-25', '080005', DATE '2011-01-27', '08005', 'Digital Logic Design', 'English', 'CA005', 5, 300.00, 1500.00)
7. INTO purchase VALUES ('INV0006', DATE '2007-09-14', '080006', DATE '2007-09-18', '08006', 'Operating Systems Concepts', 'English', 'CA006', 20, 600.00, 6000.00)
8. SELECT * FROM dual;

```

Results Explain Describe Saved SQL History

6 rows inserted.

0.05 seconds

Copyright © 1992-2018, Oracle and/or its affiliates. Oracle APEX 21.1.0

31.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```

1. CREATE VIEW view_purchase AS SELECT invoice_no, book_name, cate_id FROM purchase WHERE cate_id = (SELECT cate_id FROM book_list WHERE no_page=201);

```

Results Explain Describe Saved SQL History

View created.

0.05 seconds

Copyright © 1992-2018, Oracle and/or its affiliates. Oracle APEX 21.1.0

32.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```

1. SELECT * FROM view_purchase;

```

Results Explain Describe Saved SQL History

INVOICE_NO	BOOK_NAME	CATE_ID
INV0001	Introduction to Electrodynamics	CA001
INV0003	Conceptual Physics	CA001
INV0005	Introduction to Electrodynamics	CA001
INV0001	Introduction to Electrodynamics	CA001

4 rows returned in 0.02 seconds Download

Copyright © 1992-2018, Oracle and/or its affiliates. Oracle APEX 21.1.0

33.

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Item Development', 'Gallery', 'Search', and a user icon for 'Yashasvani Mark...'. The schema dropdown is set to 'WKSP_YYDBMS2025'. The main area shows a SQL command window with the following code:

```
1 CREATE TABLE category (cate_id VARCHAR(10) PRIMARY KEY, cate_descrip VARCHAR(50));
```

The results tab shows the output:

```
Table created.
```

Execution time: 0.06 seconds.

34.

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Item Development', 'Gallery', 'Search', and a user icon for 'Yashasvani Mark...'. The schema dropdown is set to 'WKSP_YYDBMS2025'. The main area shows a SQL command window with the following code:

```
1 INSERT ALL
2   INTO category (cate_id, cate_descrip) VALUES ('CATE01', 'Science')
3   INTO category (cate_id, cate_descrip) VALUES ('CATE02', 'Technology')
4   INTO category (cate_id, cate_descrip) VALUES ('CATE03', 'Computers')
5   INTO category (cate_id, cate_descrip) VALUES ('CATE04', 'Nature')
6   INTO category (cate_id, cate_descrip) VALUES ('CATE05', 'Medical')
7   SELECT * FROM dual;
```

The results tab shows the output:

```
5 rows(s) inserted.
```

Execution time: 0.05 seconds.

35.

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Item Development', 'Gallery', 'Search', and a user icon for 'Yashasvani Mark...'. The schema dropdown is set to 'WKSP_YYDBMS2025'. The main area shows a SQL command window with the following code:

```
1 CREATE VIEW view_purchasedset AS SELECT a.cate_id,a.cate_descrip, b.invoice_no, b.invoice_dt,b.book_name FROM category a, purchase b WHERE a.cate_id=b.cate_id;
```

The results tab shows the output:

```
View created.
```

Execution time: 0.05 seconds.

36.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Scheme WSP_YDBMS205

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```
1 SELECT * FROM view_purchased;
```

Results Explain Describe Saved SQL History

CATE_ID	CATE_DESCRP	INVOICE_NO	INVOICE_DT	BOOK_NAME
CA001	Science	INV0001	7/15/2008	Introduction to Electrodynamics
CA002	Technology	INV0002	8/25/2008	Transfer of Heat and Mass
CA001	Science	INV0003	9/20/2008	Conceptual Physics
CA002	Technology	INV0004	8/30/2007	Transfer of Heat and Mass
CA001	Science	INV0005	7/28/2007	Introduction to Electrodynamics
CA003	Computers	INV0006	9/24/2007	Guide to Networking
CA001	Science	INV0007	7/15/2008	Introduction to Electrodynamics
CA002	Technology	INV0008	9/24/2007	Fundamentals of Thermodynamics
CA003	Computers	INV0009	5/5/2009	Principles of Electronics
CA004	Nature	INV0004	6/15/2010	Computer Networks

More than 10 rows available. Increase rows selector to view more rows.

yashavantmark...@gmail.com yw_ddms_205 Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

37.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Scheme WSP_YDBMS205

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```
1 SELECT VIEW_NAME
2 FROM USER_VIEWS
3 WHERE VIEW_NAME = 'VIEW_BOOKLIST3';
```

Results Explain Describe Saved SQL History

VIEW_NAME
VIEW_BOOKLIST3

1 row returned in 0.05 seconds. Download

yashavantmark...@gmail.com yw_ddms_205 Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

38.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Scheme WSP_YDBMS205

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```
1 SELECT * FROM view_booklist3;
```

Results Explain Describe Saved SQL History

BOOK_ID	BOOK_NAME	ISBN_NO	CATE_ID	AUT_ID	PUB_ID	DT_OF_PUB	PUBL_LANG	NO_PAGE	BOOK_PRICE
BK002	Understanding of Steel Construction	0000978002	CA002	AUT002	P001	2/5/2003	English	300	105.5
BK004	Transfer of Heat and Mass	0000978004	CA002	AUT004	P004	6/15/2001	English	500	195
BK005	Conceptual Physics	0000978005	CA001	AUT005	P005	2/16/2003	-	345	145
BK006	Fundamentals of Heat	0000978006	CA001	AUT006	P004	8/9/2003	German	247	112
BK007	Advanced 3d Graphics	0000978007	CA003	AUT007	P005	5/7/2001	Hindi	756	58
BK009	Mental Health Nursing	0000978009	CA005	AUT009	P006	3/5/2002	English	400	225
BK010	Fundamentals of Thermodynamics	0000978010	CA002	AUT010	P007	12/16/2002	English	528	265
BK012	The Nature of World	0000978012	CA004	AUT012	P008	12/22/2001	English	250	195
BK015	Environment a Sustainable Future	0000978015	CA004	AUT015	P006	12/20/2005	German	165	100
BK014	Concepts in Health	0000978014	CA005	AUT014	P009	-	-	258	80

More than 10 rows available. Increase rows selector to view more rows.

yashavantmark...@gmail.com yw_ddms_205 Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

OUTPUTS:

1.

```

1 -- yashasvani
2 • CREATE DATABASE IF NOT EXISTS ADMS_LAB;
3 • USE ADMS_LAB;
4 • DROP TABLE IF EXISTS employees;
5 • CREATE TABLE employees (
6     id INT AUTO_INCREMENT PRIMARY KEY,
7     name VARCHAR(100),
8     department VARCHAR(100),
9     salary DECIMAL(10, 2)
10 );
11 • DESC employees;

```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(100)	YES		NULL	
department	varchar(100)	YES		NULL	
salary	decimal(10,2)	YES		NULL	

2.

```

12 -- yashasvani
13 • INSERT INTO employees (name, department, salary) VALUES
14     ('John Doe', 'Engineering', 50000.00),
15     ('Jane Smith', 'HR', 45000.00),
16     ('Michael Johnson', 'Engineering', 55000.00),
17     ('Emily Brown', 'Sales', 48000.00);
18
19 • SELECT * FROM employees;
20

```

id	name	department	salary
1	John Doe	Engineering	50000.00
2	Jane Smith	HR	45000.00
3	Michael Johnson	Engineering	55000.00
4	Emily Brown	Sales	48000.00
NULL	NULL	NULL	NULL

3.

```

20 -- yashasvani
21 • CREATE INDEX idx_department ON employees (department);
22 • CREATE INDEX idx_salary ON employees (salary);
23 • SHOW INDEX FROM employees;
24

```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
employees	0	PRIMARY	1	id	A	4	NULL	NULL	YES	BTREE		YES	NULL	
employees	1	idx_department	1	department	A	3	NULL	YES	YES	BTREE		YES	NULL	
employees	1	idx_salary	1	salary	A	4	NULL	YES	YES	BTREE		YES	NULL	

4.

```

25 -- yashasvani
26 • SELECT * FROM employees WHERE department = 'Engineering';
27 • SELECT * FROM employees WHERE salary > 50000.00;

```

id	name	department	salary
3	Michael Johnson	Engineering	55000.00
NULL	NULL	NULL	NULL

5.

```
29 -- yashasvani
30 • CREATE INDEX idx_covering ON employees (department, salary);
31 • SELECT department, salary FROM employees WHERE department = 'Engineering';
32 • SELECT * FROM employees USE INDEX (idx_salary) WHERE salary > 50000.00;
33 • SELECT * FROM employees USE INDEX (idx_covering) WHERE salary > 50000.00;
34 • SELECT department, salary FROM employees USE INDEX (idx_covering) WHERE salary > 50000.00;
35 • SELECT * FROM employees WHERE department = 'Engineering' AND salary > 50000.00;
36 • SELECT * FROM employees WHERE salary > 50000.00 AND department = 'Engineering';
37 • SELECT * FROM employees FORCE INDEX (idx_department) WHERE department = 'Engineering';
38 • CREATE INDEX idx_descending ON employees (department DESC);

38 • CREATE INDEX idx_descending ON employees (department DESC);
39 • SELECT department, salary FROM employees USE INDEX (idx_descending) WHERE department = 'Engineering';
40 • SELECT * FROM employees WHERE salary > 50000.00 ORDER BY department DESC;
41 • SELECT * FROM employees USE INDEX (idx_descending) WHERE salary > 50000.00 ORDER BY department DESC;
42 • CREATE INDEX idx_mixed_order ON employees (department ASC, salary DESC);
43 • SELECT * FROM employees USE INDEX (idx_mixed_order) WHERE salary > 50000.00;
44 • SHOW INDEX FROM employees;
```

Result Grid			
Edit: Export/Import: Wrap Cell Content:			
id	name	department	salary
3	Michael Johnson	Engineering	55000.00

6.

```
46 -- yashasvani
47 • DROP INDEX idx_department ON employees;
48 • DROP INDEX idx_salary ON employees;
49 • DROP INDEX idx_covering ON employees;
50 • DROP INDEX idx_descending ON employees;
51 • DROP INDEX idx_mixed_order ON employees;
52 • SHOW INDEX FROM employees;
```

Result Grid	
department	salary
Engineering	50000.00
Engineering	55000.00

7.

```
-- yashasvani
CREATE TABLE books_n(
book_id INT AUTO_INCREMENT PRIMARY KEY,
title VARCHAR(255),
author VARCHAR(255),
publication_year int
);
```

```
28 14:27:41 CREATE TABLE books_n(book_id INT AUTO_INCREMENT PRIMARY KEY, title VARCHAR(255), author VA... 0 row(s) affected
```

8.

```
-- yashasvani
INSERT INTO books (title, author, publication_year) VALUES
('The Adventures of Tom Sawyer', 'Mark Twain', 1876),
('The Picture of Dorian Gray', 'Oscar Wilde', 1890),
('Alice\'s Adventures in Wonderland', 'Lewis Carroll', 1865),
('The War of the Worlds', 'H.G. Wells', 1898),
('Dracula', 'Bram Stoker', 1897),
('Jane Eyre', 'Charlotte Brontë', 1847),
('Wuthering Heights', 'Emily Brontë', 1847),
('Frankenstein', 'Mary Shelley', 1818),
('The Count of Monte Cristo', 'Alexandre Dumas', 1844),
('The Adventures of Sherlock Holmes', 'Arthur Conan Doyle', 1892),
('The Strange Case of Dr. Jekyll and Mr. Hyde', 'Robert Louis Stevenson', 1886),
('The Hound of the Baskervilles', 'Arthur Conan Doyle', 1902),
('Les Misérables', 'Victor Hugo', 1862),
('Anna Karenina', 'Leo Tolstoy', 1877),
('War and Peace', 'Leo Tolstoy', 1869),
('Crime and Punishment', 'Fyodor Dostoevsky', 1866),
('Gone with the Wind', 'Margaret Mitchell', 1936),
('The Grapes of Wrath', 'John Steinbeck', 1939),
('The Old Man and the Sea', 'Ernest Hemingway', 1952),
('The Sun Also Rises', 'Ernest Hemingway', 1926),
('Of Mice and Men', 'John Steinbeck', 1937),
('One Hundred Years of Solitude', 'Gabriel García Márquez', 1967),

('Don Quixote', 'Miguel de Cervantes', 1605),
('The Scarlet Letter', 'Nathaniel Hawthorne', 1850),
('The Canterbury Tales', 'Geoffrey Chaucer', 1400),
('Heart of Darkness', 'Joseph Conrad', 1899),
('The Odyssey', 'Homer', 800),
('The Iliad', 'Homer', 800),
('Catch-22', 'Joseph Heller', 1961),
('To the Lighthouse', 'Virginia Woolf', 1927),
('Slaughterhouse-Five', 'Kurt Vonnegut', 1969),
('The Bell Jar', 'Sylvia Plath', 1963),
('The Great Expectations', 'Charles Dickens', 1861),
('The Adventures of Huckleberry Finn', 'Mark Twain', 1884),
('The Chronicles of Narnia', 'C.S. Lewis', 1950),
('Lord of the Flies', 'William Golding', 1954),
('The Outsiders', 'S.E. Hinton', 1967),
('The Road', 'Cormac McCarthy', 2006),
('Beloved', 'Toni Morrison', 1987),
('Pride and Prejudice', 'Jane Austen', 1813),
('The Hobbit', 'J.R.R. Tolkien', 1937),
('The Lord of the Rings', 'J.R.R. Tolkien', 1954),
('Harry Potter and the Philosopher\'s Stone', 'J.K. Rowling', 1997),
('Moby-Dick', 'Herman Melville', 1851),
('Brave New World', 'Aldous Huxley', 1932),
('The Da Vinci Code', 'Dan Brown', 2003),
```

('Angels & Demons', 'Dan Brown', 2000),
('The Girl with the Dragon Tattoo', 'Stieg Larsson', 2005),
('The Hunger Games', 'Suzanne Collins', 2008),
('The Catcher in the Rye', 'J.D. Salinger', 1951),
('The Martian', 'Andy Weir', 2011),
('A Game of Thrones', 'George R.R. Martin', 1996),
('The Help', 'Kathryn Stockett', 2009),
('The Girl on the Train', 'Paula Hawkins', 2015),
('Gone Girl', 'Gillian Flynn', 2012),
('The Lovely Bones', 'Alice Sebold', 2002),
('Life of Pi', 'Yann Martel', 2001),
('The Alchemist', 'Paulo Coelho', 1988),
('Twilight', 'Stephenie Meyer', 2005),
('The Secret Life of Bees', 'Sue Monk Kidd', 2001),
('The Book Thief', 'Markus Zusak', 2005),
('Water for Elephants', 'Sara Gruen', 2006),
('The Girl with the Pearl Earring', 'Tracy Chevalier', 1999),
('The Help', 'Kathryn Stockett', 2009),
('The Kite Runner', 'Khaled Hosseini', 2003),
('The Glass Castle', 'Jeannette Walls', 2005),
('Eat, Pray, Love', 'Elizabeth Gilbert', 2006),
('The Time Traveler's Wife', 'Audrey Niffenegger', 2003),
('The Goldfinch', 'Donna Tartt', 2013),
('Big Little Lies', 'Liane Moriarty', 2014),

('The Nightingale', 'Kristin Hannah', 2015),
('The Underground Railroad', 'Colson Whitehead', 2016),
('The Silent Patient', 'Alex Michaelides', 2019),
('Where the Crawdads Sing', 'Delia Owens', 2018),
('Becoming', 'Michelle Obama', 2018),
('Educated', 'Tara Westover', 2018),
('Circe', 'Madeline Miller', 2018),
('Normal People', 'Sally Rooney', 2018),
('Little Fires Everywhere', 'Celeste Ng', 2017),
('The Testaments', 'Margaret Atwood', 2019),
('The Tattooist of Auschwitz', 'Heather Morris', 2018),
('The Wife Between Us', 'Greer Hendricks', 2018);

✓ 29 14:34:45 INSERT INTO books (title, author, publication_year) VALUES ('The Adventures of Tom Sawyer', 'Mark Twain', ... 82 row(s) affected Records: 82 Duplicates: 0 Warnings: 0

Name: Yashasvani Markanday

SAPID: 590011386

DBMS LAB – 10 #PL/SQL Commands

Write a PL/SQL code to accept the value of A, B & C display which is greater

The screenshot shows the Oracle APEX SQL Workshop interface. The code in the editor is:

```
1 DECLARE
2   a NUMBER := 10; -- Input value for A
3   b NUMBER := 20; -- Input value for B
4   c NUMBER := 15; -- Input value for C
5 BEGIN
6   IF a > b AND a > c THEN
7     DBMS_OUTPUT.PUT_LINE('A is greater');
8   ELSEIF b > a AND b > c THEN
9     DBMS_OUTPUT.PUT_LINE('B is greater');
10  ELSE
11    DBMS_OUTPUT.PUT_LINE('C is greater');
12  END IF;
13 END;
```

The results pane shows the output: "B is greater".

Using PL/SQL Statements create a simple loop that display message “Welcome to PL/SQL Programming” 20 times.

The screenshot shows the Oracle APEX SQL Workshop interface. The code in the editor is:

```
1 BEGIN
2   FOR i IN 1..20 LOOP
3     DBMS_OUTPUT.PUT_LINE("Welcome to PL/SQL Programming");
4   END LOOP;
5 END;
```

The results pane shows the output: "Welcome to PL/SQL Programming" repeated 20 times.

Write a PL/SQL code block to find the factorial of a number.

```
1  DECLARE
2      num NUMBER := 5; -- Input number
3      factorial NUMBER := 1;
4  BEGIN
5      FOR i IN 1..num LOOP
6          factorial := factorial * i;
7      END LOOP;
8      DBMS_OUTPUT.PUT_LINE('Factorial of ' || num || ' is: ' || factorial);
9  END;
```

Results Explain Describe Saved SQL History

Factorial of 5 is 120
Statement preprocessed.
0.01seconds

Copyright © 1999-2020, Oracle and/or its affiliates. Oracle APEX 24.2.0

Write a PL/SQL program to generate Fibonacci series.

```
1  DECLARE
2      -- Variables to hold Fibonacci sequence
3      first_number NUMBER := 0;
4      second_number NUMBER := 1;
5      next_number NUMBER;
6      limit_number NUMBER := 100; |
7      CURSOR Fibonacci_cursor IS
8          SELECT LEVEL AS n, first_number AS Fibonacci_Number
9      FROM DUAL
10     CONNECT BY LEVEL <= limit_number;
11  BEGIN
12      -- Print the initial values
```

```
13  DBMS_OUTPUT.PUT_LINE('Fibonacci Series up to ' || limit_number || ' numbers:');
14  DBMS_OUTPUT.PUT_LINE('0'); -- First Fibonacci number
15  DBMS_OUTPUT.PUT_LINE('1'); -- Second Fibonacci number
16
17  -- Loop to generate and print Fibonacci series
18  FOR i IN 3..limit_number LOOP
19      next_number := first_number + second_number;
20      first_number := second_number;
21      second_number := next_number;
22      DBMS_OUTPUT.PUT_LINE(next_number);
23  END LOOP;
24  END;
```

Fibonacci Series up to 100 numbers:
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6771
10951
17944

Copyright © 1999-2020, Oracle and/or its affiliates. Oracle APEX 24.2.0

```

ID:5086
PK:693
E21393
108410
337811
534220
320989
324659
2178369
3524576
3267007
9227705
1000000
36515817
39888149
62645986
1113501355
163508241
26792428
323406437
7010001733
1134081170
10161111801
2971215973
408705798
777328409
12081628925

Copyright © 1996-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

ID:5086
PK:693
3295108999
53150591473
1626737272
13930386445
3232053117
36342536162
19178775979
956722676843
154808675578
258473079381
408705798
655745019843
1061024957733
13167688777565
2277789885798
3084327833851
7229384813155
1139082806866
15911049209135
308461523319329
808454011879264
3865353388933
4280454489957
211163502978059
161674625986267

Copyright © 1996-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

ID:5086
PK:693
927797979864277
3295108999
53150591473
1626737272
2361122830862985
32980882571187980
36395798721411598
90194951954759497
138398100979888
22085694931122385
42219534927406673
67010361761861258
130008778166209121
17299794160421409
208000213417851548
4240000000000000000
754011986740546429
1220616643121870758
1974074219405223167
1154041405498899565
33880998585858338873
83880998585858338877
1358812754796778648
310922995814355109820

Statement processed.

Copyright © 1996-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

```

Write a PL/SQL code to find the sum of first N numbers.

```

APEX App Builder SQL Workshop Team Development Gallery
+ SQL Commands Schema WKSP_YDIMS2025 Run
Language SQL Rows 10 Clear Command Find Table Save Run
A:
1. DECLARE
2. total_sum NUMBER := 0;
3.
4. -- Variables for the number up to which sum is to be calculated
5. N NUMBER := 100; -- Change N to the desired number
6.
7. BEGIN
8.   FOR i IN 1..N LOOP
9.     total_sum := total_sum + i;
10.  END LOOP;
11.
12. DBMS_OUTPUT.PUT_LINE('The sum of first ' || N || ' numbers is: ' || total_sum);
13. END;

```

Results	Explain	Describe	Saved SQL	History
<pre>The sum of first 100 numbers is: 5050 Statement processed. 0.01 seconds</pre>				

Copyright © 1996-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

WAP that accept the value of A, B & C display which is greater using Functions and Procedures.

```

1  DECLARE
2    -- Variables to hold the values of A, B, and C
3    A NUMBER := 10; -- Replace with your desired value for A
4    B NUMBER := 20; -- Replace with your desired value for B
5    C NUMBER := 15; -- Replace with your desired value for C
6
7    -- Function to find the maximum of two numbers
8    FUNCTION Find_maximum(x NUMBER, y NUMBER) RETURN NUMBER IS
9    BEGIN
10      IF x > y THEN
11        RETURN x;
12      ELSE
13        RETURN y;
14      END IF;
15    END find_maximum;

```

```

1  -- Procedure to compare three numbers and display the greatest
2  PROCEDURE compare_and_display_greatest(A NUMBER, B NUMBER, C NUMBER)
3  IS
4    greatest NUMBER;
5
6    BEGIN
7      greatest := find_maximum(find_maximum(A, B), C); -- Call the function to find the maximum of
8      -- three numbers
9
10     -- Display the result
11     DBMS_OUTPUT.PUT_LINE('The greatest is ' || greatest);
12     DBMS_OUTPUT.PUT_LINE('Call the procedure to compare and display the greatest');
13     DBMS_OUTPUT.PUT_LINE('compare_and_display_greatest(10, 20, 15)');
14
15   END;
16
17  -- Call the procedure to compare and display the greatest
18  compare_and_display_greatest(10, 20, 15);
19
20

```

Results Explain Describe Saved SQL History

8 is the greatest.

Statement processed.

0.00 seconds

Print the statements “Welcome to PL/SQL Programming” 20 times using Functions and Procedures.

```

1  BEGIN
2    -- We decide to store the message
3    :MESSAGE_USING_PLSQL := 'Welcome To PL/SQL Programming';
4    :MESSAGE_USING_PLSQL := RTRIM(:MESSAGE_USING_PLSQL);
5
6    PROCEDURE display_message IS
7    BEGIN
8      -- We decide to store the message in :MESSAGE
9      :MESSAGE := RTRIM(:MESSAGE_USING_PLSQL);
10     DBMS_OUTPUT.PUT_LINE(:MESSAGE);
11     DBMS_OUTPUT.PUT_LINE(:MESSAGE);
12     DBMS_OUTPUT.PUT_LINE(:MESSAGE);
13
14    END;
15
16    -- Call the procedure to display the message
17    display_message();
18
19  END;

```

Results Explain Describe Saved SQL History

Welcome to PL/SQL Programming
Welcome to PL/SQL Programming

Factorial of a number using function and procedure.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema: WKS_P_YYDBMS205

```

1  DECLARE
2      -- Variable to hold the value for factorial calculation
3      n NUMBER := 5; -- Replace with your desired value for factorial calculation
4
5      -- Function to calculate Factorial of a value
6      FUNCTION factorial(n NUMBER) RETURN NUMBER IS
7          result NUMBER := 1;
8      BEGIN
9          IF n <= 1 THEN
10              RETURN result;
11          ELSE
12              FOR i IN 2..n LOOP
13                  result := result * i;
14              END LOOP;
15          RETURN result;
16      END;
17      END factorial;
18
19  -- Procedure to calculate and display factorial
20  PROCEDURE calculate_and_display_factorial(n IN NUMBER) IS
21      fact NUMBER;
22  BEGIN
23      fact := factorial(n); -- Call the function to calculate factorial
24
25      -- Display the result
26      DBMS_OUTPUT.PUT_LINE('Factorial of ' || n || ' is: ' || fact);
27      END calculate_and_display_factorial;
28  END;
29  -- Call the procedure to calculate and display factorial
30  calculate_and_display_factorial(5);
31
32

```

Results Explain Describe Saved SQL History

Factorial of 5 is: 120
Statement processed.
0.01 seconds

Copyright © 2000-2025, Oracle and/or its affiliates. Oracle APEX 24.2.1

Fibonacci series of a number using function and procedure.

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema: WKS_P_YYDBMS205

```

1  DECLARE
2      -- Variable to hold the number of terms in the Fibonacci series
3      num_terms NUMBER := 10; -- Replace with your desired number of terms
4
5      -- Function to calculate Fibonacci series
6      FUNCTION fibonacci_series(num_terms IN NUMBER) RETURN VARCHAR2 IS
7          fib_series VARCHAR2(1000) := '0, 1'; -- Initial values of Fibonacci series
8          a NUMBER := 0;
9          b NUMBER := 1;
10         next_term NUMBER;
11     BEGIN
12         IF num_terms_in <= 0 THEN
13             RETURN 'Invalid input';
14         END IF;
15
16         IF num_terms_in = 1 THEN
17             RETURN '0';
18         END IF;
19
20         FOR i IN 2..num_terms_in - 1 LOOP
21             next_term := a + b;
22             fib_series := fib_series || ',' || next_term;
23             a := b;
24             b := next_term;
25         END LOOP;
26
27         RETURN fib_series;
28     END fibonacci_series;
29
30  -- Procedure to calculate Fibonacci series
31  PROCEDURE calculate_fibonacci_series(num_terms IN NUMBER) IS
32      fib_series VARCHAR2(1000);
33  BEGIN
34      fib_series := fibonacci_series(num_terms);
35      DBMS_OUTPUT.PUT_LINE(fib_series);
36  END calculate_fibonacci_series;
37
38

```

Results Explain Describe Saved SQL History

Fibonacci series up to 10 terms: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610
Statement processed.
0.01 seconds

Copyright © 2000-2025, Oracle and/or its affiliates. Oracle APEX 24.2.1

Find the sum of first N numbers using Function and Procedure.

The screenshot shows the Oracle APEX SQL Workshop interface. The code in the editor is:

```
1 declare
2   n number := 10; -- variable to hold the number of terms
3   sum number := 0; -- variable to hold the sum of terms
4 begin
5   for i in 1..n loop
6     sum := sum + i;
7   end loop;
8   return sum;
9 end;
```

The results pane shows the output:

```
Sum of first 10 numbers: 55
```

Statement processed.
0.01 seconds

Lab Performance Questions

1. Write a PL/SQL code to calculate the area of a circle given its radius.

The screenshot shows the Oracle APEX SQL Workshop interface. The code in the editor is:

```
1 declare
2   radius number := 7; -- Change radius as needed
3   area number;
4   pi constant number := 3.14159;
5 begin
6   area := pi * radius * radius;
7   dbms_output.put_line('Area of Circle = ' || area);
8 end;
```

The results pane shows the output:

```
Area of Circle = 384.7751
```

Statement processed.
0.01 seconds

2. Create a PL/SQL program to check whether a given number is prime or not.

```

1  DECLARE
2      num NUMBER := 29; -- Change number as needed
3      i NUMBER;
4      flag BOOLEAN := TRUE;
5  BEGIN
6      IF num <= 1 THEN
7          flag := FALSE;
8      ELSE
9          FOR i IN 2..TRUNC(SQRT(num)) LOOP
10              IF MOD(num, i) = 0 THEN
11                  flag := FALSE;
12                  EXIT;
13              END IF;
14          END LOOP;
15      END IF;
16
17      IF flag THEN
18          DBMS_OUTPUT.PUT_LINE(num || ' is a Prime Number');
19      ELSE
20          DBMS_OUTPUT.PUT_LINE(num || ' is NOT a Prime Number');
21      END IF;
22  END;

```

Results tab shows the output: 29 is a Prime Number.

Statement processed.

0.01 seconds

Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

3. Develop a PL/SQL script to reverse a string entered by the user.

```

1  DECLARE
2      input_str VARCHAR2(100) := 'PLSQL PROGRAM'; -- Change string as needed
3      reversed_str VARCHAR2(500) := '';
4  BEGIN
5      FOR i IN REVERSE 1..LENGTH(input_str) LOOP
6          reversed_str := reversed_str || SUBSTR(input_str, i, 1);
7      END LOOP;
8
9      DBMS_OUTPUT.PUT_LINE('Original String: ' || input_str);
10     DBMS_OUTPUT.PUT_LINE('Reversed String: ' || reversed_str);
11  END;

```

Results tab shows the output:

Original String: PLSQL PROGRAM
Reversed String: MARGORPL SQLP

Statement processed.

0.00 seconds

Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

4. Design a PL/SQL code to find the largest and smallest number among three given numbers.

```

1  DECLARE
2      A NUMBER := 40;
3      B NUMBER := 15;
4      C NUMBER := 25;
5
6      largest NUMBER;
7      smallest NUMBER;
8  BEGIN
9      -- Largest
10     largest := GREATEST(A, B, C);
11
12     -- Smallest
13     smallest := LEAST(A, B, C);
14
15     DBMS_OUTPUT.PUT_LINE('Largest Number = ' || largest);
16     DBMS_OUTPUT.PUT_LINE('Smallest Number = ' || smallest);
17  END;

```

Results tab shows the output:

Largest Number = 40
Smallest Number = 15

Statement processed.

Copyright © 1999-2024, Oracle and/or its affiliates. Oracle APEX 24.2.0

5. Implement a PL/SQL program to calculate the compound interest for a given principal amount, rate, and time period

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Yesheswari Merk...' and the schema 'WSP_YYDEMS2025'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. The code editor contains the following PL/SQL block:

```
1  DECLARE
2      principal NUMBER := 10000;    -- P
3      rate     NUMBER := 5;        -- R (in %)
4      time_year NUMBER := 3;      -- T (in years)
5      amount   NUMBER;
6      ci       NUMBER;
7
8  BEGIN
9      amount := principal * POWER((1 + rate/100), time_year);
10     ci := amount - principal;
11     DBMS_OUTPUT.PUT_LINE('Compound Interest = ' || ci);
12     DBMS_OUTPUT.PUT_LINE('Total Amount = ' || amount);
13 END;
```

Below the code editor, the 'Results' tab is selected, showing the output of the executed query:

```
Compound Interest = 1576.25
Total Amount = 11576.25
Statement processed.

0.00 seconds
```

The bottom status bar displays the user's email ('yesheswari.merk@gmail.com'), session ID ('yy_dems_2025'), and the Oracle APEX version ('Oracle APEX 24.2.0').