

INTRODUCTION

A Database Management System (DBMS) is a software application that interacts with end-users, other applications, and the database itself to capture and analyze data. It provides a structured way to store, manage, and retrieve data efficiently.

Key Features of a DBMS:

- Data Definition Language (DDL): Defines the structure of a database, including tables, columns, data types, and relationships between them.
- Data Manipulation Language (DML): Allows users to insert, update, delete, and retrieve data from the database.
- Data Query Language (DQL): Enables users to query the database to extract specific information.
- Data Control Language (DCL): Grants and revokes access privileges to different users.

Advantages of Using a DBMS:

- Data Integrity: Ensures data accuracy and consistency through validation rules and constraints.
- Data Security: Protects sensitive data from unauthorized access through user authentication and authorization mechanisms.
- Data Independence: Separates data from the applications that use it, allowing for flexible changes without affecting the other.
- Efficient Data Access: Provides optimized techniques for storing and retrieving data, improving performance.
- Data Sharing: Enables multiple users to access and modify the same data concurrently.
- Backup and Recovery: Offers mechanisms to create backups and restore data in case of failures or disasters.

Disadvantages of Using a DBMS:

- Complexity: Requires specialized knowledge and skills to design, implement, and manage databases.
- Cost: Can be expensive, especially for large-scale databases, due to software licensing, hardware infrastructure, and maintenance costs.
- Performance Overhead: Can introduce overhead in terms of processing queries and updating data, especially for complex operations.

SQL (Structured Query Language)

SQL is a powerful language used to interact with relational databases. It allows you to:

- Define Data: Create, modify, and delete database structures like tables, columns, and indexes.
- Manipulate Data: Insert, update, delete, and retrieve data from tables.
- Query Data: Retrieve specific information from the database using various query techniques.
- Control Data Access: Grant and revoke permissions to different users.

SQL Command			
DDL	DML	DCL	TCL
<ul style="list-style-type: none">➤ Create➤ Drop➤ Alter➤ Truncate➤ Rename	<ul style="list-style-type: none">➤ Insert➤ Update➤ Delete➤ Select	<ul style="list-style-type: none">➤ Grant➤ Revoke	<ul style="list-style-type: none">➤ Commit➤ Rollback➤ Save point

SQL QUERIES:

Table Creation: DDL Commands

```
1 • CREATE TABLE dept(  
2     deptno INT,  
3     dname VARCHAR(14),  
4     loc VARCHAR(13),  
5     constraint pk_dept primary key (deptno)  
6 );
```

```

7 • CREATE TABLE emp(
8     empno INT,
9     ename VARCHAR(10),
10    job VARCHAR(9),
11    mgr INT,
12    hiredate DATE,
13    sal DECIMAL(7,2),
14    comm DECIMAL(7,2),
15    deptno INT,
16    constraint pk_emp primary key (empno),
17    constraint fk_deptno foreign key (deptno) references dept (deptno)
18 );

```

Data Insertion: DML Commands

```

19 • INSERT INTO dept(deptno, dname, loc) VALUES
20    (10, 'ACCOUNTING', 'NEW YORK'),
21    (20, 'RESEARCH', 'DALLAS'),
22    (30, 'SALES', 'CHICAGO'),
23    (40, 'OPERATIONS', 'BOSTON');






```

```

25 • INSERT INTO emp VALUES
26    (7839, 'KANE', 'PRESIDENT', null, '1981-11-17', 5000, null, 10),
27    (7698, 'STEFAN', 'MANAGER', 7839, '1981-05-01', 2850, null, 30),
28    (7782, 'CLARK', 'MANAGER', 7839, '1981-06-09', 2450, null, 10),
29    (7566, 'JONES', 'MANAGER', 7839, '1981-04-02', 2975, null, 20),
30    (7788, 'SCOTT', 'ANALYST', 7566, '1981-04-19', 3000, null, 20),
31    (7902, 'DAMON', 'ANALYST', 7566, '1981-03-12', 3000, null, 20),
32    (7369, 'SMITH', 'CLERK', 7902, '1980-12-17', 800, null, 20),
33    (7499, 'ALLEN', 'SALESMAN', 7698, '1981-02-20', 1600, 300, 30),
34    (7521, 'SCOTT', 'SALESMAN', 7698, '1981-02-22', 1250, 500, 30),
35    (7654, 'MARTIN', 'SALESMAN', 7698, '1981-09-28', 1250, 1400, 30),
36    (7844, 'TRAVER', 'SALESMAN', 7698, '1981-09-08', 1500, 0, 30),
37    (7876, 'ADAMS', 'CLERK', 7788, '1987-05-23', 1100, null, 20),
38    (7900, 'JAMES', 'CLERK', 7698, '1981-03-12', 950, null, 30),
39    (7934, 'MILLER', 'CLERK', 7782, '1982-01-23', 1300, null, 10)

```






EMP TABLE

Result Grid								
Filter Rows: <input type="text"/>								
Edit:   								
Export/Import:  								
	empno	ename	job	mgr	hiredate	sal	comm	deptno
▶	7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
	7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
	7521	SCOTT	SALESMAN	7698	1981-02-22	1250.00	500.00	30
	7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
	7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
	7698	STEFAN	MANAGER	7839	1981-05-01	2850.00	NULL	30
	7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10
	7788	SCOTT	ANALYST	7566	1981-04-19	3000.00	NULL	20
	7839	KANE	PRESIDENT	NULL	1981-11-17	5000.00	NULL	10
	7844	TRAVER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
	7876	ADAMS	CLERK	7788	1987-05-23	1100.00	NULL	20
	7900	JAMES	CLERK	7698	1981-03-12	950.00	NULL	30
	7902	DAMON	ANALYST	7566	1981-03-12	3000.00	NULL	20
	7934	MILLER	CLERK	7782	1982-01-23	1300.00	NULL	10
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Ques 1. Display all the employees who are managers and clerks:

```
41 ❌ SELECT * FROM emp WHERE job IN ('MANAGER', 'CLERK');
```

```
42
```

Result Grid								
Filter Rows: <input type="text"/>								
Edit:   								
Export/Import:  								
	empno	ename	job	mgr	hiredate	sal	comm	deptno
	7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
	7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
	7698	STEFAN	MANAGER	7839	1981-05-01	2850.00	NULL	30
	7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10
	7876	ADAMS	CLERK	7788	1987-05-23	1100.00	NULL	20
	7900	JAMES	CLERK	7698	1981-03-12	950.00	NULL	30
	7934	MILLER	CLERK	7782	1982-01-23	1300.00	NULL	10
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Ques 2. Display the details of the employees who work in dept 20 as well as those in dept 30:

```
43 • SELECT * FROM emp
44 WHERE deptno IN (20, 30);
```

Result Grid

Filter Rows:

Edit:

Export/Import:

empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	SCOTT	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7698	STEFAN	MANAGER	7839	1981-05-01	2850.00	NULL	30
7788	SCOTT	ANALYST	7566	1981-04-19	3000.00	NULL	20
7844	TRAVIS	SALESMAN	7698	1981-06-09	1500.00	0.00	30

Ques 3. Display the employees hired in year 1981

```
46 • SELECT * FROM emp
47 WHERE YEAR(hiredate) = 1981;
48
```

Result Grid

Filter Rows:

Edit:

Export/Import:

empno	ename	job	mgr	hiredate	sal	comm	deptno
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	SCOTT	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7698	STEFAN	MANAGER	7839	1981-05-01	2850.00	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10
7788	SCOTT	ANALYST	7566	1981-04-19	3000.00	NULL	20

Ques 4. Display the name of the employee having minimum salary

```
49 • SELECT ename FROM emp
50 WHERE sal = (SELECT MIN(sal) FROM emp);
```

Result Grid	Filter Rows:	Export:
ename		
SMITH		

Ques 5. Display names of employees whose name starts with 'D' and ends with 'N'

```
52 • SELECT ename FROM emp
53 WHERE ename LIKE 'D%N';
```

Result Grid	Filter Rows:
ename	
DAMON	

Ques 6. Display the details of all the employees who are hired in the year 1981 and 1982

```
55 • SELECT * FROM emp
56 WHERE YEAR(hiredate) IN (1981, 1982);
57
```

Result Grid								
empno	ename	job	mgr	hiredate	sal	comm	deptno	
7788	SCOTT	ANALYST	7566	1981-04-19	3000.00	NULL	20	
7839	KANE	PRESIDENT	NULL	1981-11-17	5000.00	NULL	10	
7844	TRAVER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	
7900	JAMES	CLERK	7698	1981-03-12	950.00	NULL	30	
7902	DAMON	ANALYST	7566	1981-03-12	3000.00	NULL	20	
7934	MILLER	CLERK	7782	1982-01-23	1300.00	NULL	10	

Ques 7. Display all the employees who are clerks and have a salary less than 1500

```
58 • SELECT * FROM emp
59 WHERE job = 'CLERK' AND sal < 1500;
```

Result Grid								
empno	ename	job	mgr	hiredate	sal	comm	deptno	
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20	
7876	ADAMS	CLERK	7788	1987-05-23	1100.00	NULL	20	
7900	JAMES	CLERK	7698	1981-03-12	950.00	NULL	30	
7934	MILLER	CLERK	7782	1982-01-23	1300.00	NULL	10	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

Ques 8. Display those employees who are managers and salesman having salary more than 1500

```
61 • SELECT * FROM emp
62 WHERE job IN ('MANAGER', 'SALESMAN') AND sal > 1500;
```

Result Grid								
empno	ename	job	mgr	hiredate	sal	comm	deptno	
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20	
7698	STEFAN	MANAGER	7839	1981-05-01	2850.00	NULL	30	
7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

Ques 12. Display average salary of each jobtype and display those jobtype whose average salary is less then 2000

```
73 • SELECT job, AVG(sal) AS avg_salary
74 FROM emp
75 GROUP BY job
76 HAVING AVG(sal) < 2000;
```

Result Grid		Filter Rows:	Export:
job	avg_salary		
CLERK	1037.500000		
SALESMAN	1400.000000		

Ques 13. Display the total salary pf all employees who are working in department 10 and 20

```
80 • SELECT SUM(sal) AS total_salary
81 FROM emp
82 WHERE deptno IN (10, 20);
83
```

Result Grid		Filter Rows:	Export:
total_salary			
19625.00			

Ques 14. Find the no. of employees working in each department no.

```
84 • SELECT deptno, COUNT(*) AS num_employees
```

Result Grid		Filter Rows:	Export:
deptno	num_employees		
10	3		
20	5		
30	6		

Ques 15. Find the average salary of employees working in department 10 and 20

```
87 • SELECT AVG(sal) AS avg_salary
88 FROM emp
89 WHERE deptno IN (10, 20);
```

Result Grid		Filter Rows:	Export:
	avg_salary		
▶	2453.125000		

Ques 16. Find the no. of jobtypes working in each deptno

```
91 • SELECT deptno, job
92 FROM emp
93 GROUP BY deptno, job;
```

Result Grid		Filter Rows:	Export:
	deptno	job	
▶	20	CLERK	
	30	SALESMAN	
	20	MANAGER	
	30	MANAGER	
	10	MANAGER	
	20	ANALYST	
	10	PRESIDENT	
	30	CLERK	
	10	CLERK	



Ques 17. Find the total salary of each jobtype and display those jobtype whose total salary is greater than 7000

```
95 • SELECT job, SUM(sal) AS total_salary
96 FROM emp
97 GROUP BY job
98 HAVING SUM(sal) > 7000;
```

Result Grid		Filter Rows:	Export:
	job	total_salary	
▶	MANAGER	8275.00	

Ques 18. Find the no. of days of employment for the employees 7499 and 7566

```
107 • SELECT ename, empno, DATEDIFF(CURDATE(), hiredate) AS days_of_employment
108 FROM emp
109 WHERE empno IN (7499, 7566);
```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

ename	empno	days_of_employment
ALLEN	7499	15968
JONES	7566	15927

Ques 19. Find 90 days probation date add of joining date of employees

```
100 • SELECT ename, hiredate, DATE_ADD(hiredate, INTERVAL 90 DAY) AS probation_end_date
101 FROM emp;
```

<

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

ename	hiredate	probation_end_date
JONES	1981-04-02	1981-07-01
MARTIN	1981-09-28	1981-12-27
STEFAN	1981-05-01	1981-07-30
CLARK	1981-06-09	1981-09-07
SCOTT	1981-04-19	1981-07-18
KANE	1981-11-17	1982-02-15
TRAVER	1981-09-08	1981-12-07
ADAMS	1987-05-23	1987-08-21
JAMES	1981-03-12	1981-06-10
DAMON	1981-03-12	1981-06-10
MILLER	1982-01-23	1982-04-23

Ques 20. Find 6 months of probation date after the joining date of employees who are hired in 1981

```
103 • SELECT ename, hiredate, DATE_ADD(hiredate, INTERVAL 6 MONTH) AS probation_end_date
104 FROM emp
105 WHERE YEAR(hiredate) = 1981;
```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

ename	hiredate	probation_end_date
ALLEN	1981-02-20	1981-08-20
SCOTT	1981-02-22	1981-08-22
JONES	1981-04-02	1981-10-02
MARTIN	1981-09-28	1982-03-28
STEFAN	1981-05-01	1981-11-01
CLARK	1981-06-09	1981-12-09
SCOTT	1981-04-19	1981-10-19
KANE	1981-11-17	1982-05-17
TRAVER	1981-09-08	1982-03-08
JAMES	1981-03-12	1981-09-12
DAMON	1981-03-12	1981-09-12

Ques 21. Display the hiredate in the format '14th of March, 1981' for the employee who is having the highest salary

```

111 • SELECT ename, DATE_FORMAT(hiredate, '%D of %M, %Y') AS formatted_hiredate
112 FROM emp
113 WHERE sal = (SELECT MAX(sal) FROM emp);

```

< [Progress Bar]

Result Grid | [Grid Icon] | [Filter Rows:] | Export: [Export Icon] | Wrap Cell Content: [Wrap Icon]

	ename	formatted_hiredate
▶	KANE	17th of November, 1981

Ques 22. Display the date in the format '1st April of 1981, 05:35:15' for all the employees.

```

115 • SELECT ename, DATE_FORMAT(hiredate, '%D %M of %Y, %H:%i:%s') AS formatted_hiredate
116 FROM emp;

```

< [Progress Bar]

Result Grid | [Grid Icon] | [Filter Rows:] | Export: [Export Icon] | Wrap Cell Content: [Wrap Icon]

	ename	formatted_hiredate
▶	SMITH	17th December of 1980, 00:00:00
	ALLEN	20th February of 1981, 00:00:00
	SCOTT	22nd February of 1981, 00:00:00
	JONES	2nd April of 1981, 00:00:00
	MARTIN	28th September of 1981, 00:00:00
	STEFAN	1st May of 1981, 00:00:00
	CLARK	9th June of 1981, 00:00:00
	SCOTT	19th April of 1981, 00:00:00
	KANE	17th November of 1981, 00:00:00
	TRAVER	8th September of 1981, 00:00:00
	ADAMS	23rd May of 1987, 00:00:00

Ques 23. Decrease the salary by 50% for employee having the highest salary

```

124 • SELECT ename, sal, sal * 0.5 AS 'Salary 50%'
125 FROM emp
126 WHERE sal = (SELECT MAX(sal) FROM emp);
127

```

< [Progress Bar]

Result Grid | [Grid Icon] | [Filter Rows:] | Export: [Export Icon] | Wrap Cell Content: [Wrap Icon]

	ename	sal	Salary 50%
▶	KANE	5000.00	2500.000

Ques 24. Give 10% salary hike to all employees who were hired in the year 1981

```
118 • select ename, sal, sal*0.1+sal as "Salary With 10% Hike" from emp;
```

ename	sal	Salary With 10% Hike
SMITH	800.00	880.000
ALLEN	1600.00	1760.000
SCOTT	1250.00	1375.000
JONES	2975.00	3272.500
MARTIN	1250.00	1375.000
STEFAN	2850.00	3135.000
CLARK	2450.00	2695.000
SCOTT	3000.00	3300.000
KANE	5000.00	5500.000
TRAVER	1500.00	1650.000
ADAMS	1100.00	1210.000
JAMES	950.00	1045.000
DAMON	3000.00	3300.000

EMP179 TABLE

Ques 25. Create a duplicate table of emp with name as emp179

```
128 • CREATE TABLE emp179 AS
129 SELECT * FROM emp;
130 • select* from emp179;
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	SCOTT	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30

Ques 26. Add a constraint CHK on salary column to ensure that all salary entered is more than 700

```
135 • ALTER TABLE emp179
136 ADD CONSTRAINT chk_salary CHECK (sal > 700);
```

Ques 27. Disable the constraint CHK

```
ALTER TABLE emp179
```

```
DISABLE CONSTRAINT chk_salary; .
```

Ques 28. Drop the constraint CHK from emp1 table

```
138 • ALTER TABLE emp179
```

```
139 DROP CONSTRAINT chk_salary;
```

```
--
```

Ques 29. Drop the column address from emp1 table

```
141 • ALTER TABLE emp179
```

```
142 DROP COLUMN address;
```

BORROWER AND DEPOSITOR COMMAND

```
184 • SELECT* FROM DEPOSITOR;
```

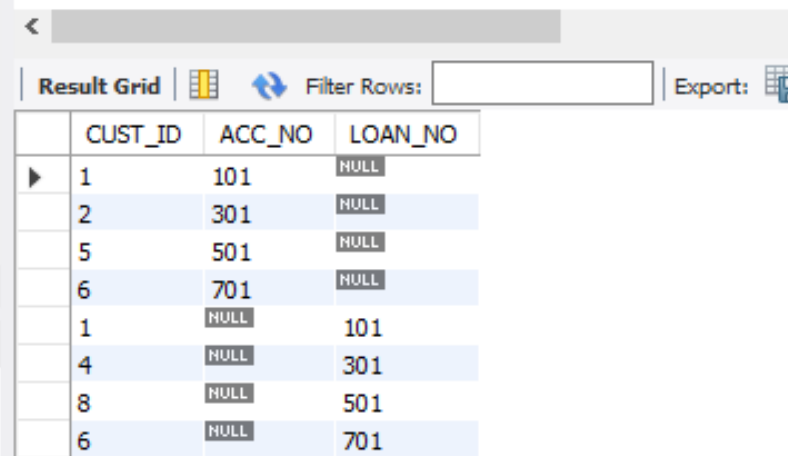
<		
Result Grid		
Filter Rows:		
	CUST_ID	ACC_NO
▶	1	101
	2	301
	5	501
	6	701

```
186 • SELECT* FROM BORROWER;
```

<		
Result Grid		
Filter Rows:		
	CUST_ID	LOAN_NO
▶	1	101
	4	301
	8	501
	6	701

Ques 30. Display customers having account in bank taken loan or both

```
188 • SELECT CUST_ID, ACC_NO, NULL AS LOAN_NO
189 FROM DEPOSITOR
190 UNION
191 SELECT CUST_ID, NULL AS ACC_NO, LOAN_NO
192 FROM BORROWER;
```

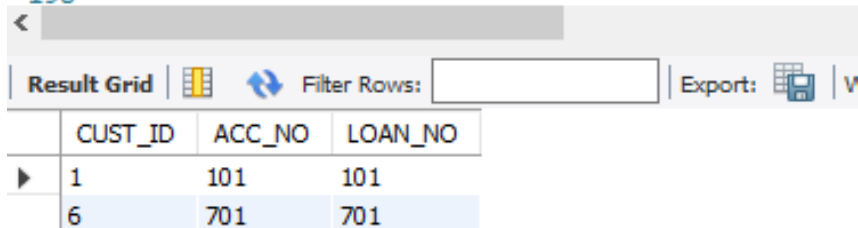


The screenshot shows a database query result grid. At the top, there is a toolbar with a 'Result Grid' button, a 'Filter Rows' input field, and an 'Export' button. Below the toolbar is a table with four columns: CUST_ID, ACC_NO, and LOAN_NO. The table contains 10 rows of data, which are the union of the DEPOSITOR and BORROWER tables. The first 6 rows correspond to the DEPOSITOR table, and the next 4 rows correspond to the BORROWER table. The LOAN_NO column is NULL for the first 6 rows, and the ACC_NO column is NULL for the last 4 rows.

	CUST_ID	ACC_NO	LOAN_NO
▶	1	101	NULL
	2	301	NULL
	5	501	NULL
	6	701	NULL
	1	NULL	101
	4	NULL	301
	8	NULL	501
	6	NULL	701

Ques 31. Display customers having account as well as taken loan

```
194 • SELECT D.CUST_ID, D.ACC_NO, B.LOAN_NO
195 FROM DEPOSITOR D
196 INNER JOIN BORROWER B
197 ON D.CUST_ID = B.CUST_ID;
198
```



The screenshot shows a database query result grid. At the top, there is a toolbar with a 'Result Grid' button, a 'Filter Rows' input field, and an 'Export' button. Below the toolbar is a table with four columns: CUST_ID, ACC_NO, and LOAN_NO. The table contains 2 rows of data, which are the inner join of the DEPOSITOR and BORROWER tables. The first row corresponds to CUST_ID 1, and the second row corresponds to CUST_ID 6. Both rows have non-NULL values for ACC_NO and LOAN_NO.

	CUST_ID	ACC_NO	LOAN_NO
▶	1	101	101
	6	701	701

Ques 32. Display the customer having only account in bank and no loan

```

199 • SELECT D.CUST_ID, D.ACC_NO
200 FROM DEPOSITOR D
201 LEFT JOIN BORROWER B
202 ON D.CUST_ID = B.CUST_ID
203 WHERE B.CUST_ID IS NULL;
204

```

Result Grid | Filter Rows: | Export:

	CUST_ID	ACC_NO
▶	2	301
	5	501

Ques 33. Display the customer having only loan no. account

```

205 • SELECT B.CUST_ID, B.LOAN_NO
206 FROM BORROWER B
207 LEFT JOIN DEPOSITOR D
208 ON B.CUST_ID = D.CUST_ID
209 WHERE D.CUST_ID IS NULL;
210
211

```

Result Grid | Filter Rows: | Ex

	CUST_ID	LOAN_NO
▶	4	301
	8	501

JOIN FUNCTION

Ques 34. Display empno, ename, job, dname, location for all the employee JOHNS AND SMITH

```

143 • SELECT emp.empno, emp.ename, emp.job, dept.dname, dept.loc
144 FROM emp
145 JOIN dept ON emp.deptno = dept.deptno
146 WHERE emp.ename IN ('JOHNS', 'SMITH');
147

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	empno	ename	job	dname	loc
▶	7369	SMITH	CLERK	RESEARCH	DALLAS

Ques 35. Display ename, salary, location of the people who hired first

```
148 • SELECT emp.ename, emp.sal, dept.loc
149 FROM emp
150 JOIN dept ON emp.deptno = dept.deptno
151 WHERE emp.hiredate = (SELECT MIN(hiredate) FROM emp);
152
```

Result Grid			
Filter Rows:		Export:	Wrap Cell Content:
▶	ename	sal	loc
▶	SMITH	800.00	DALLAS

Ques 36. Display empno, job, salary, department name for all the employees working in department 10 and 20

```
153 • SELECT emp.empno, emp.job, emp.sal, dept.dname
154 FROM emp
155 JOIN dept ON emp.deptno = dept.deptno
156 WHERE emp.deptno IN (10, 20);
157
```



Result Grid				
Filter Rows:		Export:	Wrap Cell Content:	
▶	empno	job	sal	dname
▶	7782	MANAGER	2450.00	ACCOUNTING
	7839	PRESIDENT	5000.00	ACCOUNTING
	7934	CLERK	1300.00	ACCOUNTING
	7369	CLERK	800.00	RESEARCH
	7566	MANAGER	2975.00	RESEARCH
	7788	ANALYST	3000.00	RESEARCH
	7876	CLERK	1100.00	RESEARCH
	7902	ANALYST	3000.00	RESEARCH

Ques 37. Display empno, ename, job, deptno, dname, location for all the employees who were hired in year 1981 and 1982.

```

159 • SELECT emp.empno, emp.ename, emp.job, emp.deptno, dept.dname, dept.loc
160 FROM emp
161 JOIN dept ON emp.deptno = dept.deptno
162 WHERE YEAR(emp.hiredate) IN (1981, 1982);
163

```

Result Grid						
Filter Rows: <input type="text"/>						
Export:  Wrap Cell Content: 						
empno	ename	job	deptno	dname	loc	
7934	MILLER	CLERK	10	ACCOUNTING	NEW YORK	
7566	JONES	MANAGER	20	RESEARCH	DALLAS	
7788	SCOTT	ANALYST	20	RESEARCH	DALLAS	
7902	DAMON	ANALYST	20	RESEARCH	DALLAS	
7499	ALLEN	SALESMAN	30	SALES	CHICAGO	
7521	SCOTT	SALESMAN	30	SALES	CHICAGO	
7654	MARTIN	SALESMAN	30	SALES	CHICAGO	
7698	STEFAN	MANAGER	30	SALES	CHICAGO	
7844	TRAVER	SALESMAN	30	SALES	CHICAGO	
7900	JAMES	CLERK	30	SALES	CHICAGO	

Decode SQL Queries

Ques 38. Change the dept no in the emp table to dept name values.

Worksheet

Query Builder

1 SELECT ENAME, DECODE (DEPTNO, 10, 'ACCOUNTING', 20,

2 'RESEARCH', 30, 'SALES'

3 , 40, 'OPERATIONS', DEPTNO) AS DNAME FROM EMP;

Script Output x

Query Result x

SQL | All Rows Fetched: 14 in 0.002 seconds

	ENAME	DNAME
1	KING	ACCOUNTING
2	BLAKE	SALES
3	CLARK	ACCOUNTING
4	JONES	RESEARCH
5	SCOTT	RESEARCH
6	FORD	RESEARCH
7	SMITH	RESEARCH
8	ALLEN	SALES
9	WARD	SALES
10	MARTIN	SALES
11	TURNER	SALES
12	ADAMS	RESEARCH
13	JAMES	SALES
14	MILLER	ACCOUNTING

Ques 39. Change the salaries of the employees as per the following

- If sal=1200 then increase by 5%
- If sal=1300 then increase by 10%
- If sal=1400 then increase by 15%
- If sal=1500 then increase by 20%
- If sal=1600 then increase by 25%
- If sal=1700 then increase by 30%
- If sal=1800 then increase by 35%
- If sal=1900 then increase by 40%
- If sal=2000 then increase by 45%
- Give 20% increase to unmatched

```
1 UPDATE EMP
2 SET SAL = SAL * DECODE (SAL,
3     1200, 1.05, -- Increase by 5%
4     1300, 1.10, -- Increase by 10%
5     1400, 1.15, -- Increase by 15%
6     1500, 1.20, -- Increase by 20%
7     1600, 1.25, -- Increase by 25%
8     1700, 1.30, -- Increase by 30%
9     1800, 1.35, -- Increase by 35%
10    1900, 1.40, -- Increase by 40%
11    2000, 1.45, -- Increase by 45%
12    1.20); -- Increase by 20% for unmatched
```

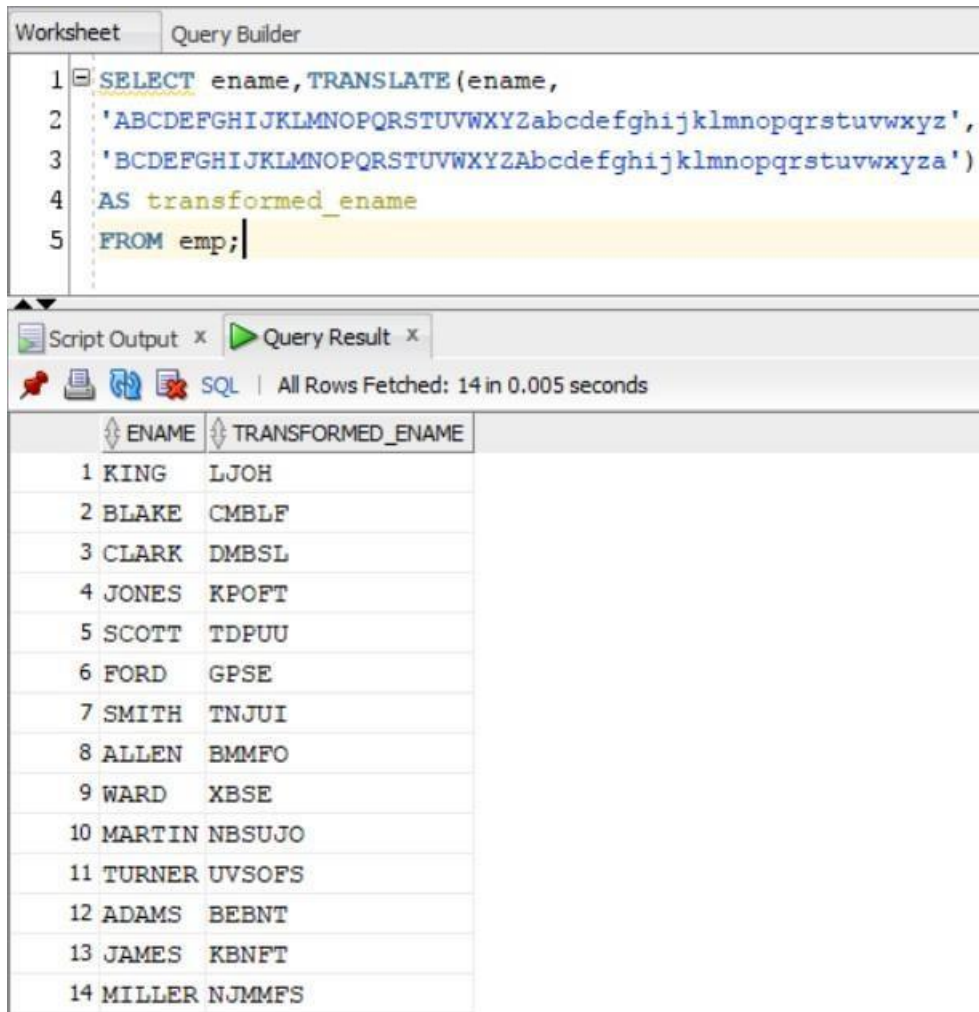
Script Output x Query Result x

Task completed in 0.026 seconds

14 rows updated.

Translate SQL Queries

Ques 40. In the emp table change the names of all the employees so that every single character in the should be replay with next character. (Translate)



The screenshot displays the SQL Developer interface. The top pane shows a query in the Query Builder:

```
1 SELECT ename, TRANSLATE(ename,  
2 'ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz',  
3 'BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz')  
4 AS transformed_ename  
5 FROM emp;
```

The bottom pane shows the Query Result with 14 rows fetched in 0.005 seconds. The result table has two columns: ENAME and TRANSFORMED_ENAME.

	ENAME	TRANSFORMED_ENAME
1	KING	LJOH
2	BLAKE	CMBLF
3	CLARK	DMBSL
4	JONES	KPOFT
5	SCOTT	TDPUU
6	FORD	GPSE
7	SMITH	TNJUI
8	ALLEN	BMMFO
9	WARD	XBSE
10	MARTIN	NBSUJO
11	TURNER	UVSOFs
12	ADAMS	BEBNT
13	JAMES	KBNFT
14	MILLER	NJMMFS

String Functions SQL Queries

Ques 41. Display all the names of the employees whose length of name is more than 5.

Worksheet		Query Builder
1		SELECT ENAME
2		FROM EMP
3		WHERE LENGTH(ENAME) > 5;

Script Output x		Query Result x
SQL All Rows Fetched: 3 in		
ENAME		
1	MARTIN	
2	TURNER	
3	MILLER	

Q49. Find the 2nd occurrence of 'A' in the job column of emp.

Worksheet		Query Builder
1		SELECT job,
2		INSTR(job, 'A', 1, 2) AS second_occurrence_of_A
3		FROM emp
4		WHERE INSTR(job, 'A', 1, 2) > 0;

Script Output x		Query Result x
SQL All Rows Fetched: 9 in 0.003 seconds		
JOB	SECOND_OCCURRENCE_OF_A	
1 MANAGER	4	
2 MANAGER	4	
3 MANAGER	4	
4 ANALYST	3	
5 ANALYST	3	
6 SALESMAN	7	
7 SALESMAN	7	
8 SALESMAN	7	
9 SALESMAN	7	

Ques 42. Display the name of the employees padded with * on the right-hand side with total length of the name is 25.

Worksheet

Query Builder

1

2

3

SELECT ename,

RPAD(ename, 25, '*') AS padded_name

FROM emp;

Script Output x

Query Result x

SQL | All Rows Fetched: 14 in 0.003 seconds

ENAME

PADDED_NAME

1 KING KING*****

2 BLAKE BLAKE*****

3 CLARK CLARK*****

4 JONES JONES*****

5 SCOTT SCOTT*****

6 FORD FORD*****

7 SMITH SMITH*****

8 ALLEN ALLEN*****

9 WARD WARD*****

10 MARTIN MARTIN*****

11 TURNER TURNER*****

12 ADAMS ADAMS*****

13 JAMES JAMES*****

14 MILLER MILLER*****

Ques 43. Trim the 1st character from all the name of the employees

Worksheet

Query Builder

1

`SELECT` `ename,`

2





`SUBSTR(ename, 2) AS trimmed_name`

3

`FROM emp;`

Script Output x

Query Result x

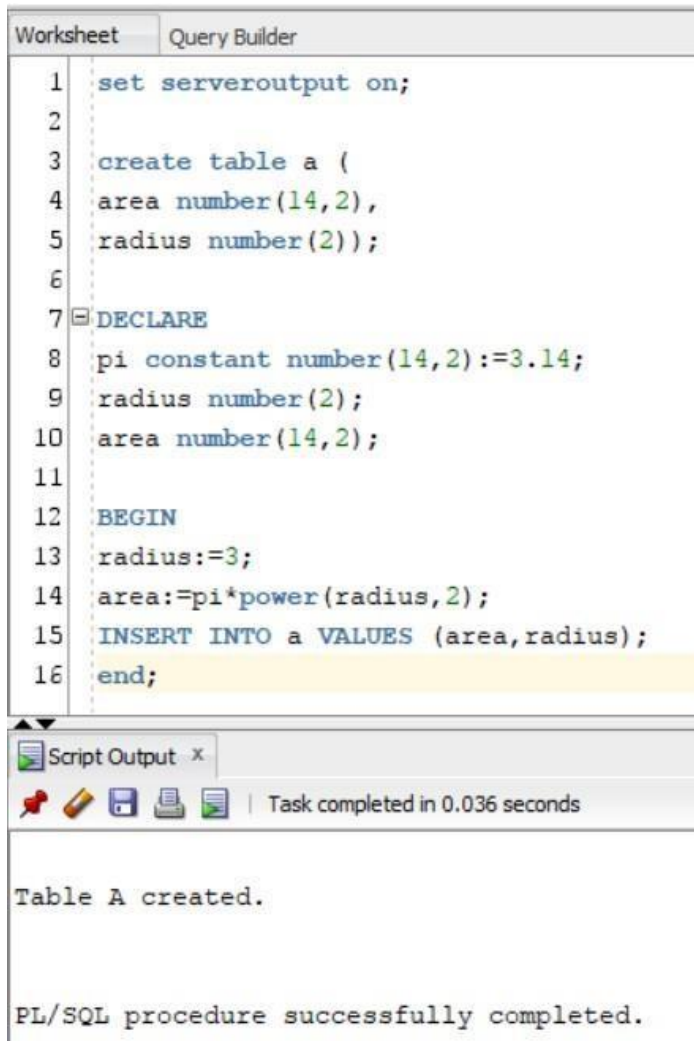


SQL | All Rows Fetched: 14 in 0.002 seconds

	ENAME	TRIMMED_NAME
1	KING	ING
2	BLAKE	LAKE
3	CLARK	LARK
4	JONES	ONES
5	SCOTT	COTT
6	FORD	ORD
7	SMITH	MITH
8	ALLEN	LLEN
9	WARD	ARD
10	MARTIN	ARTIN
11	TURNER	URNER
12	ADAMS	DAMS
13	JAMES	AMES
14	MILLER	ILLER

PL/SQL

Ques 44. Write a PL/SQL Code to calculate area of a circle.



The screenshot displays a PL/SQL development environment. At the top, there are two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, showing a script editor with the following PL/SQL code:

```
1  set serveroutput on;
2
3  create table a (
4  area number(14,2),
5  radius number(2));
6
7  DECLARE
8  pi constant number(14,2) := 3.14;
9  radius number(2);
10 area number(14,2);
11
12 BEGIN
13 radius:=3;
14 area:=pi*power(radius,2);
15 INSERT INTO a VALUES (area,radius);
16 end;
```

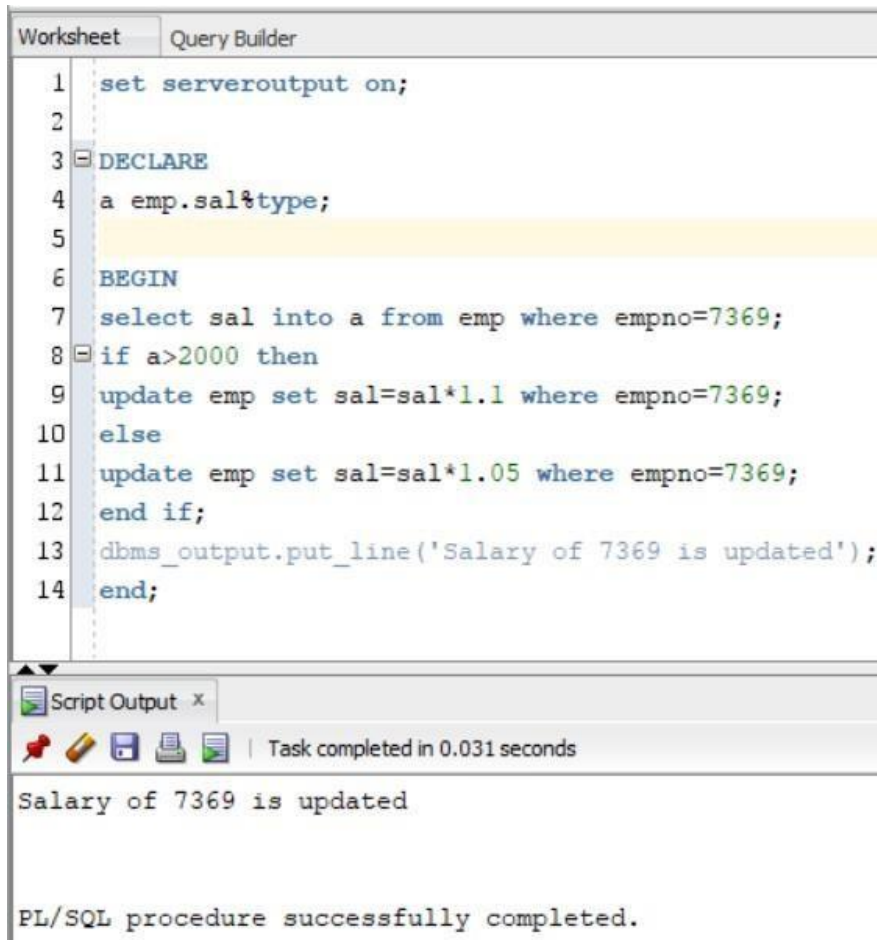
Below the script editor, there is a 'Script Output' window. It shows the results of the script execution:

```
Table A created.

PL/SQL procedure successfully completed.
```

The 'Script Output' window also includes a status bar at the top that says 'Task completed in 0.036 seconds'.

Ques 45. Increase the salary of the emp 7369 by 10% if his salary is greater or equal to 2000 otherwise increase his salary by 5%.



The screenshot displays the Oracle SQL Developer interface. The top pane, titled 'Query Builder', contains a PL/SQL script with the following lines:

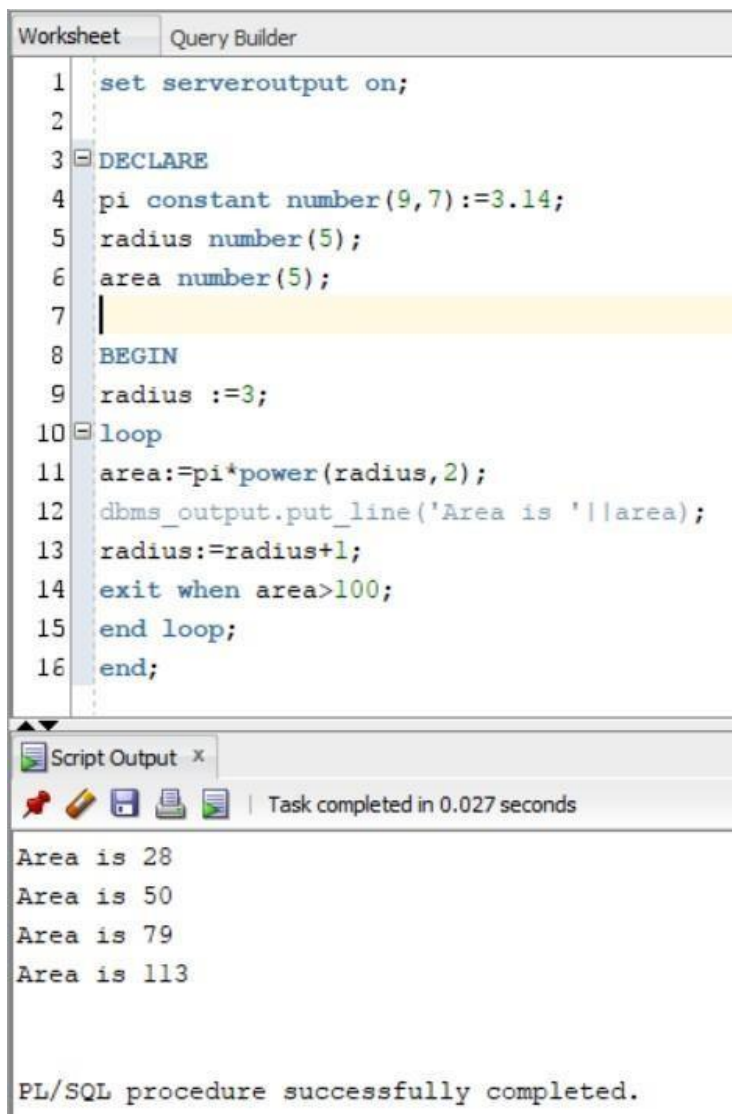
```
1  set serveroutput on;
2
3  DECLARE
4  a emp.sal%type;
5
6  BEGIN
7  select sal into a from emp where empno=7369;
8  if a>2000 then
9  update emp set sal=sal*1.1 where empno=7369;
10 else
11 update emp set sal=sal*1.05 where empno=7369;
12 end if;
13 dbms_output.put_line('Salary of 7369 is updated');
14 end;
```

The bottom pane, titled 'Script Output', shows the results of the script execution. It includes a status bar indicating 'Task completed in 0.031 seconds' and the following output messages:

```
Salary of 7369 is updated

PL/SQL procedure successfully completed.
```

Ques 46. Write PL/SQL Loop to calculate area of a circle



The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Worksheet' and 'Query Builder', contains a PL/SQL script. The script starts with 'set serveroutput on;', followed by a 'DECLARE' section where 'pi' is a constant (3.14), 'radius' is a number (5), and 'area' is a number (5). The 'BEGIN' section sets 'radius := 3;' and enters a 'loop'. Inside the loop, 'area' is calculated as 'pi * power(radius, 2)', the result is printed using 'dbms_output.put_line', 'radius' is incremented by 1, and the loop exits when 'area > 100'. The script ends with 'end loop;' and 'end;'. The bottom pane, titled 'Script Output', shows the execution results: 'Area is 28', 'Area is 50', 'Area is 79', and 'Area is 113'. Below these, it states 'PL/SQL procedure successfully completed.' and 'Task completed in 0.027 seconds'.

```
1 set serveroutput on;
2
3 DECLARE
4 pi constant number(9,7):=3.14;
5 radius number(5);
6 area number(5);
7
8 BEGIN
9 radius :=3;
10 loop
11 area:=pi*power(radius,2);
12 dbms_output.put_line('Area is '||area);
13 radius:=radius+1;
14 exit when area>100;
15 end loop;
16 end;
```

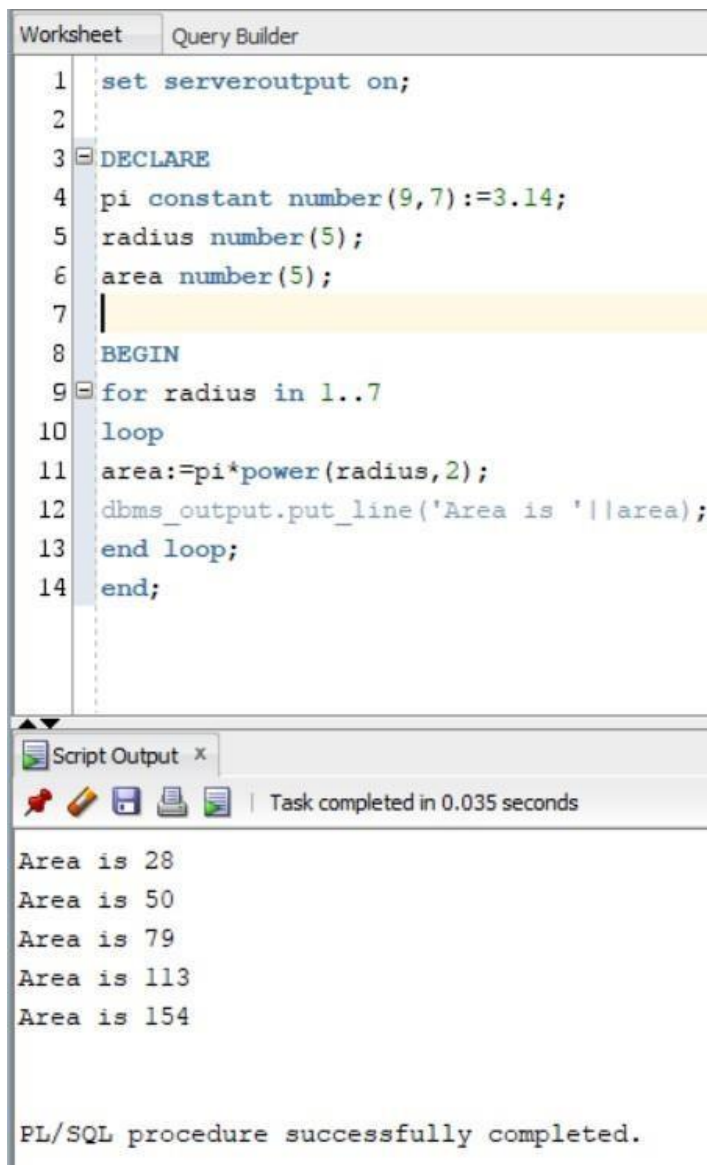
Script Output x

Task completed in 0.027 seconds

Area is 28
Area is 50
Area is 79
Area is 113

PL/SQL procedure successfully completed.

Ques 47. Write PL/SQL using for loop to calculate area of a circle



The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Worksheet', contains a PL/SQL script with line numbers 1 through 14. The script sets server output on, declares a constant pi and a variable radius, and uses a for loop to calculate the area of a circle for radii from 1 to 7. The bottom pane, titled 'Script Output', shows the execution results, including the calculated areas and a confirmation message.

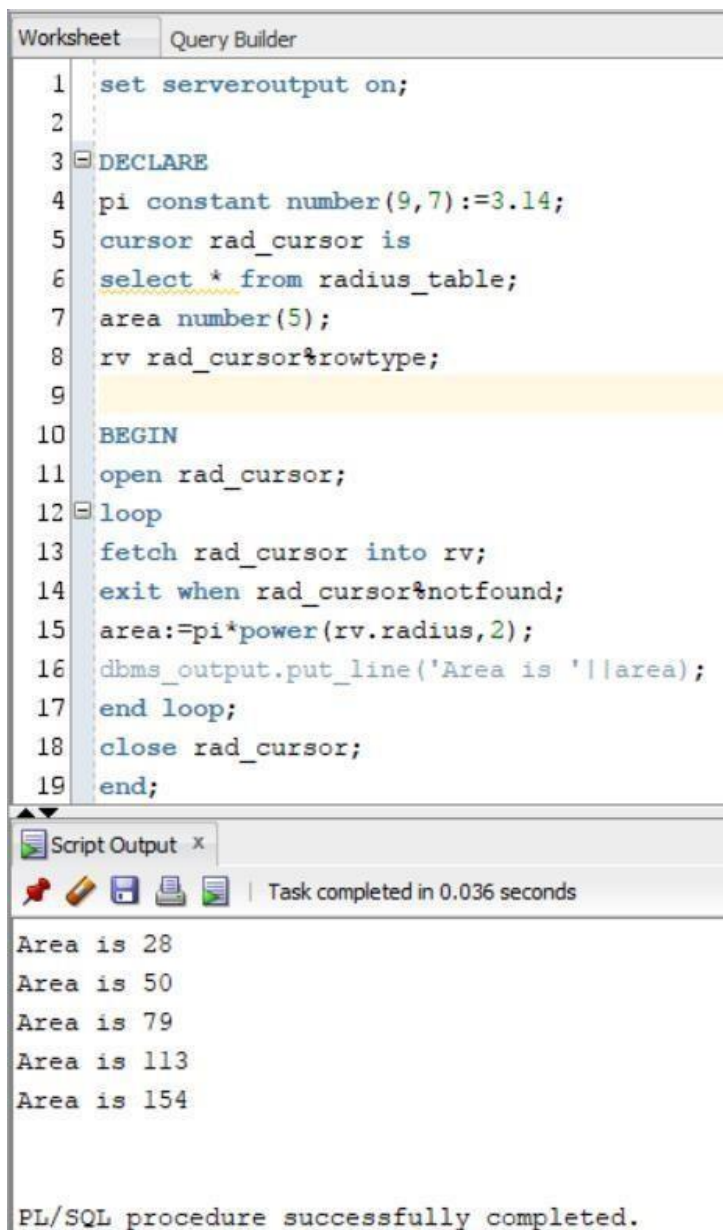
```
1 set serveroutput on;
2
3 DECLARE
4 pi constant number(9,7):=3.14;
5 radius number(5);
6 area number(5);
7
8 BEGIN
9 for radius in 1..7
10 loop
11 area:=pi*power(radius,2);
12 dbms_output.put_line('Area is '||area);
13 end loop;
14 end;
```

Script Output x
Task completed in 0.035 seconds

Area is 28
Area is 50
Area is 79
Area is 113
Area is 154

PL/SQL procedure successfully completed.

Ques 48. Write PL/SQL code to calculate area of circle using cursor.



The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Worksheet', contains a PL/SQL script. The script sets server output on, declares a constant pi, a cursor named rad_cursor, and a variable rv. It then opens the cursor, loops through the radius_table, calculates the area for each radius, and prints the result. The bottom pane, titled 'Script Output', shows the execution results, including the calculated areas for five different radii and a confirmation message.

```
1  set serveroutput on;
2
3  DECLARE
4  pi constant number(9,7):=3.14;
5  cursor rad_cursor is
6  select * from radius_table;
7  area number(5);
8  rv rad_cursor%rowtype;
9
10 BEGIN
11 open rad_cursor;
12 loop
13 fetch rad_cursor into rv;
14 exit when rad_cursor%notfound;
15 area:=pi*power(rv.radius,2);
16 dbms_output.put_line('Area is '||area);
17 end loop;
18 close rad_cursor;
19 end;
```

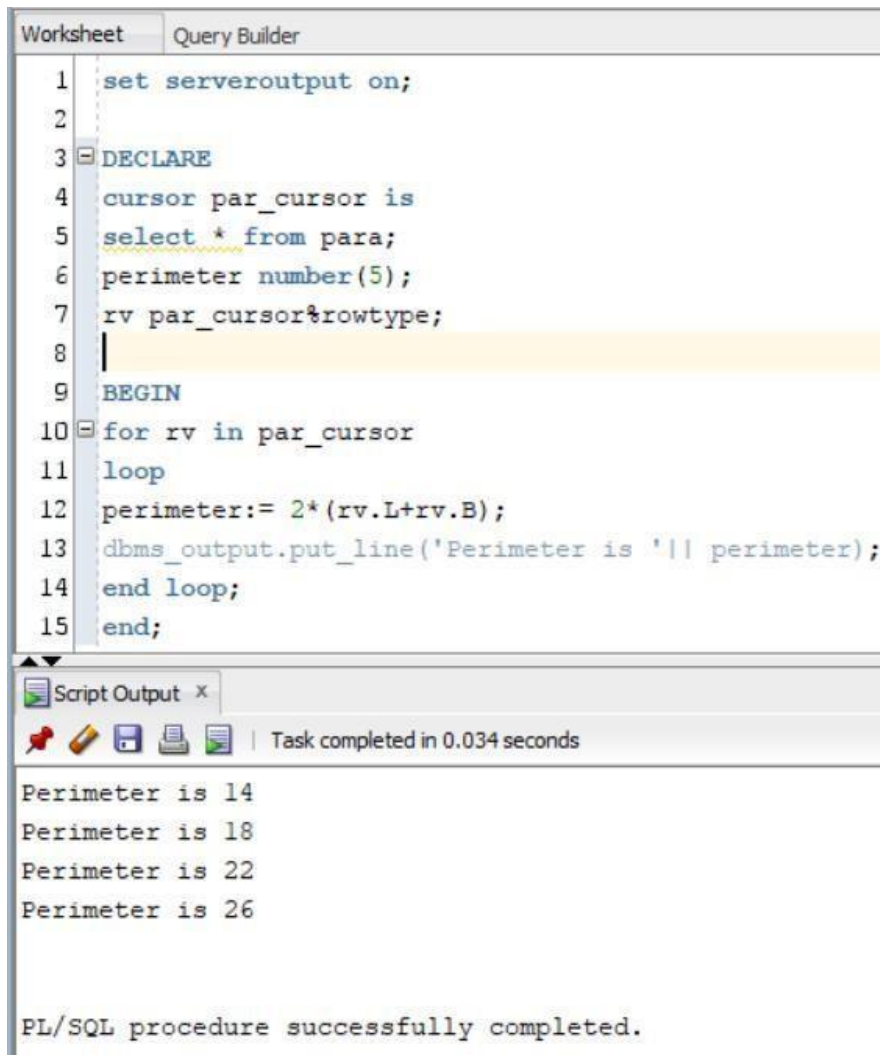
Script Output x

Task completed in 0.036 seconds

Area is 28
Area is 50
Area is 79
Area is 113
Area is 154

PL/SQL procedure successfully completed.

Ques 49. Calculate the perimeter of a rectangle, values of length and breadth are given in the table Para(L,B).



```
Worksheet  Query Builder
1  set serveroutput on;
2
3  DECLARE
4  cursor par_cursor is
5  select * from para;
6  perimeter number(5);
7  rv par_cursor%rowtype;
8
9  BEGIN
10 for rv in par_cursor
11 loop
12 perimeter:= 2*(rv.L+rv.B);
13 dbms_output.put_line('Perimeter is '|| perimeter);
14 end loop;
15 end;
```

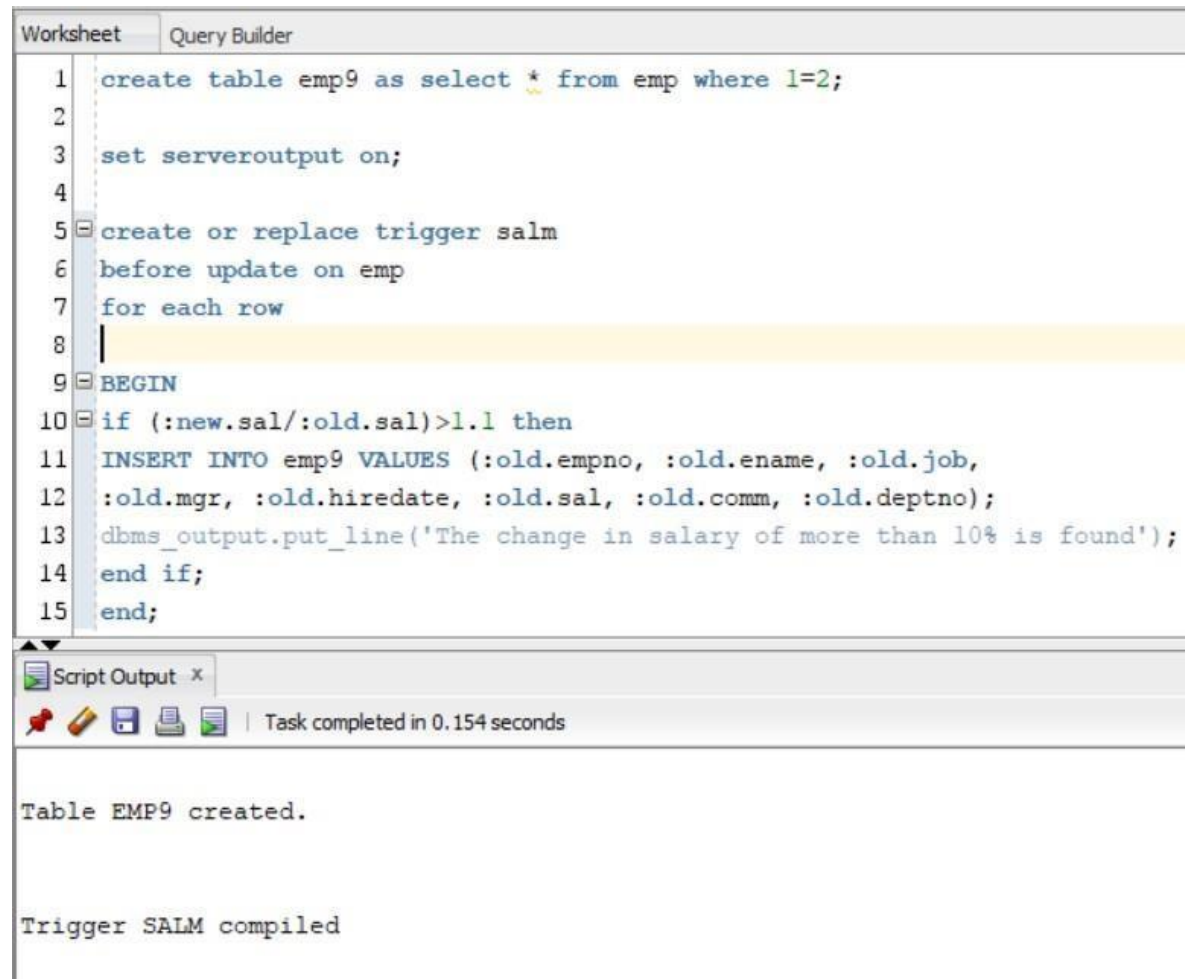
Script Output x

Task completed in 0.034 seconds

Perimeter is 14
Perimeter is 18
Perimeter is 22
Perimeter is 26

PL/SQL procedure successfully completed.

Ques 50. In the emp table monitor the salary of the emp through trigger, if change is more than 10% in the salary than that record of the employee should be shifted to emp9 table, which is having same structure as emp table.



```
Worksheet | Query Builder
1 create table emp9 as select * from emp where 1=2;
2
3 set serveroutput on;
4
5 create or replace trigger salm
6 before update on emp
7 for each row
8
9 BEGIN
10 if (:new.sal/:old.sal)>1.1 then
11 INSERT INTO emp9 VALUES (:old.empno, :old.ename, :old.job,
12 :old.mgr, :old.hiredate, :old.sal, :old.comm, :old.deptno);
13 dbms_output.put_line('The change in salary of more than 10% is found');
14 end if;
15 end;
```

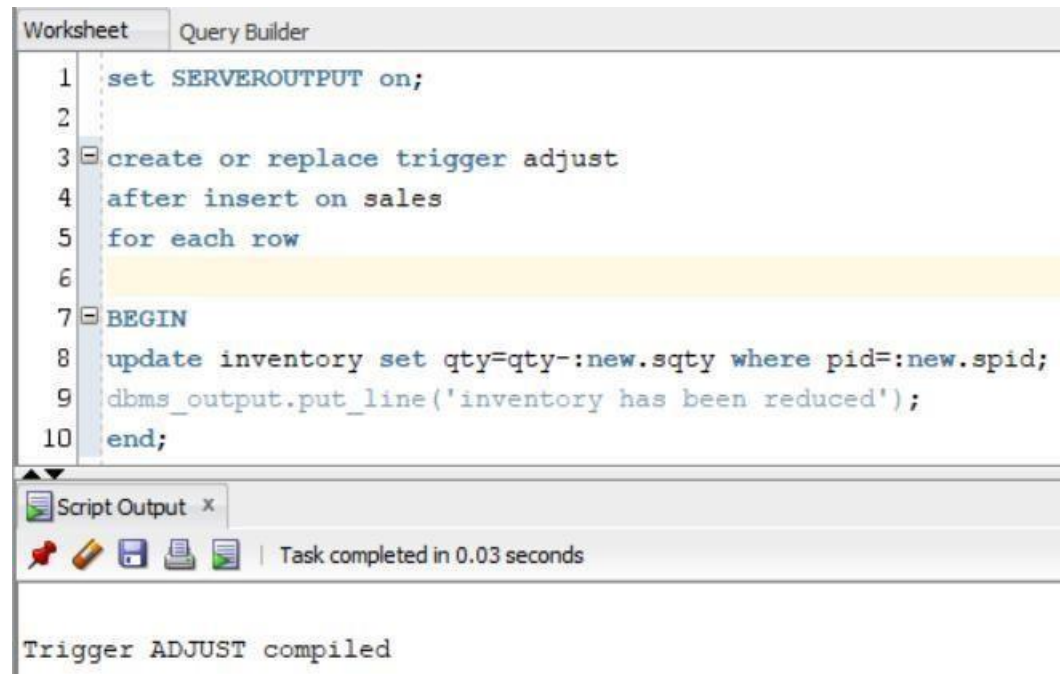
Script Output x

Task completed in 0.154 seconds

Table EMP9 created.

Trigger SALM compiled

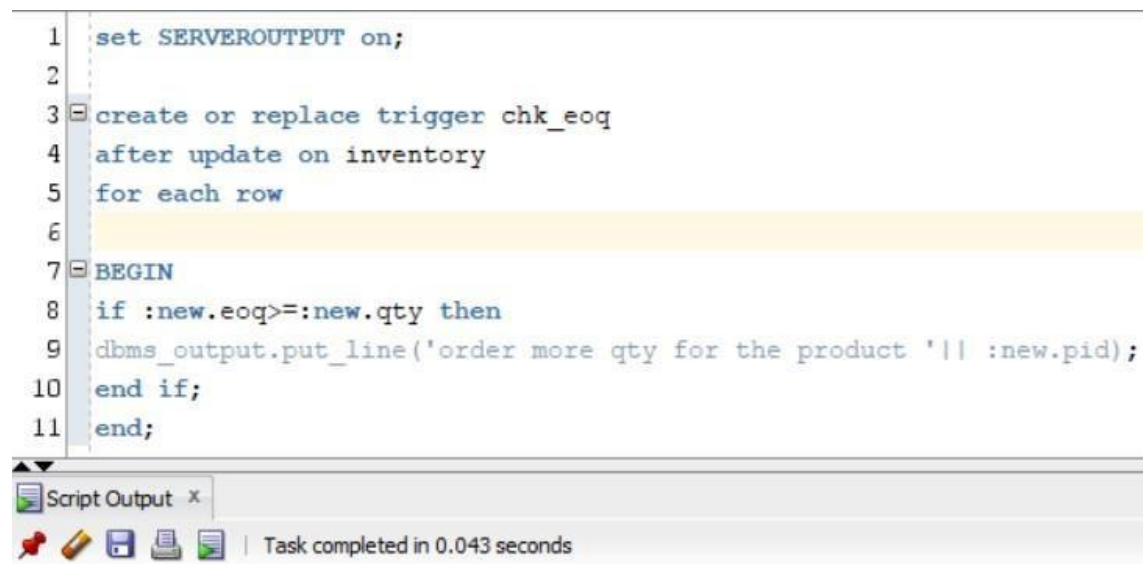
Ques 51. Create a trigger to update inventory after sales is done. Also create a trigger to check if inventory is lower than economic order quantity.



The screenshot shows the SQL Developer Query Builder interface. The 'Worksheet' tab is active, displaying a SQL script. The script starts with 'set SERVEROUTPUT on;', followed by 'create or replace trigger adjust after insert on sales for each row'. The trigger body begins with 'BEGIN', then 'update inventory set qty=qty-:new.sqty where pid=:new.spid;', followed by 'dbms_output.put_line('inventory has been reduced');', and ends with 'end;'. Below the script, the 'Script Output' window shows 'Task completed in 0.03 seconds' and the message 'Trigger ADJUST compiled'.

```
1 set SERVEROUTPUT on;
2
3 create or replace trigger adjust
4 after insert on sales
5 for each row
6
7 BEGIN
8 update inventory set qty=qty-:new.sqty where pid=:new.spid;
9 dbms_output.put_line('inventory has been reduced');
10 end;
```

Script Output x
Task completed in 0.03 seconds
Trigger ADJUST compiled

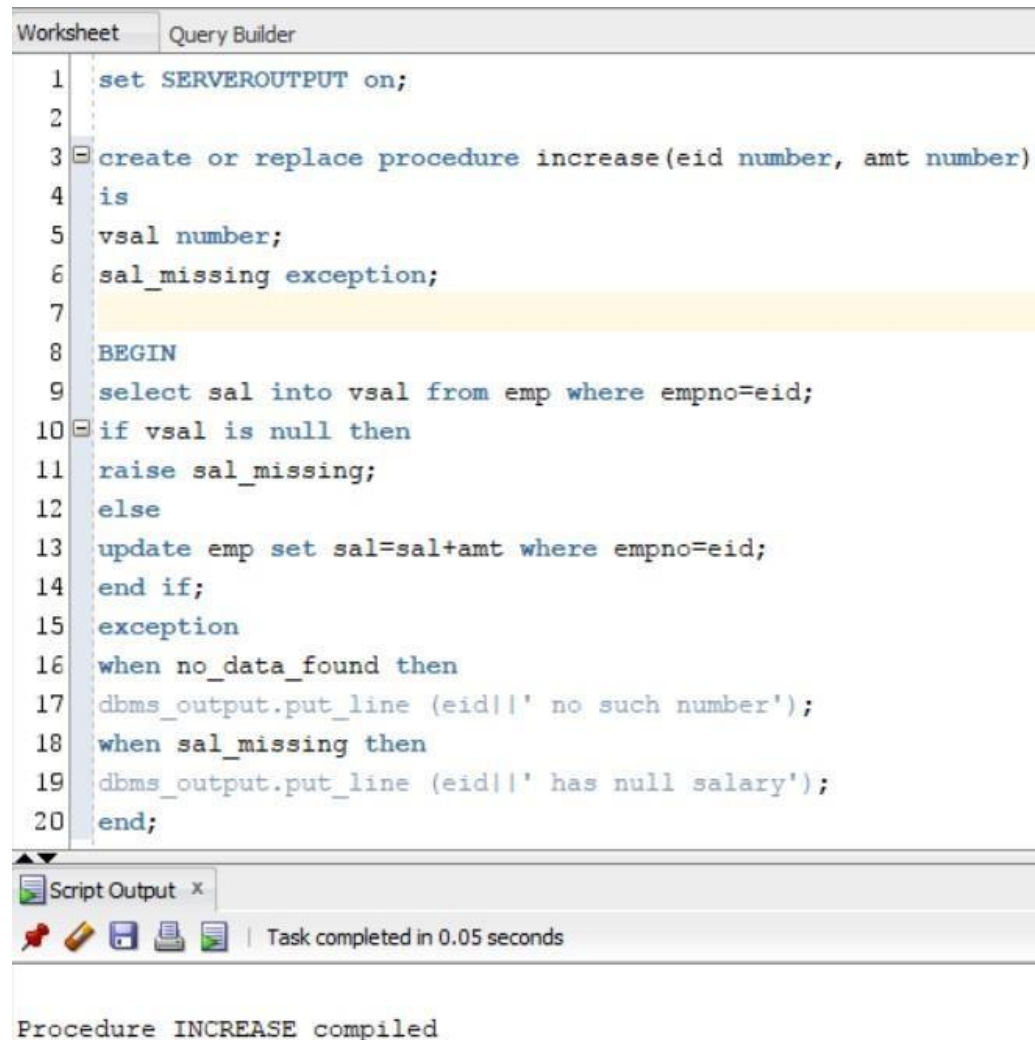


The screenshot shows the SQL Developer Query Builder interface. The 'Worksheet' tab is active, displaying a SQL script. The script starts with 'set SERVEROUTPUT on;', followed by 'create or replace trigger chk_eoq after update on inventory for each row'. The trigger body begins with 'BEGIN', then 'if :new.eoq>=:new.qty then', followed by 'dbms_output.put_line('order more qty for the product '|| :new.pid);', then 'end if;', and ends with 'end;'. Below the script, the 'Script Output' window shows 'Task completed in 0.043 seconds' and the message 'Trigger CHK_EOQ compiled'.

```
1 set SERVEROUTPUT on;
2
3 create or replace trigger chk_eoq
4 after update on inventory
5 for each row
6
7 BEGIN
8 if :new.eoq>=:new.qty then
9 dbms_output.put_line('order more qty for the product '|| :new.pid);
10 end if;
11 end;
```

Script Output x
Task completed in 0.043 seconds
Trigger CHK_EOQ compiled

Ques 52. Create a procedure to increase salary of the employee by the given amount.



```
1  set SERVEROUTPUT on;
2
3  create or replace procedure increase(eid number, amt number)
4  is
5  vsal number;
6  sal_missing exception;
7
8  BEGIN
9  select sal into vsal from emp where empno=eid;
10 if vsal is null then
11 raise sal_missing;
12 else
13 update emp set sal=sal+amt where empno=eid;
14 end if;
15 exception
16 when no_data_found then
17 dbms_output.put_line (eid||' no such number');
18 when sal_missing then
19 dbms_output.put_line (eid||' has null salary');
20 end;
```

Script Output x

Task completed in 0.05 seconds

Procedure INCREASE compiled

Ques 53. Create a function which will accept empno as a parameter and the salary of the empno will be increased as per the following.

- If salary is less than 1500 then increment should be 5% of the salary.
- If salary is between 1500-2000 then increment should be 10% of the salary, otherwise increment should be 15% of the salary.

The screenshot displays the SQL Developer interface. The top pane, titled 'Query Builder', contains the PL/SQL code for creating a function named 'hike'. The code defines a function that takes an employee ID ('eid') as a parameter and returns a number. It declares a variable 'inc' to hold the increment and 'vsal' to hold the current salary. The function logic uses an if-elsif-else structure to calculate the increment based on the salary: 5% for salaries below 1500, 10% for salaries between 1500 and 2000, and 15% for salaries above 2000. The bottom pane shows the execution of a script. The first part of the script declares a variable 'sal_inc' and calls the 'hike' function with the employee ID 7369. The output shows 'Function HIKE compiled' and 'Increment is 52.92'. The second part of the script is a simple 'end;' statement, and the final output is 'PL/SQL procedure successfully completed.'

```
1 set SERVEROUTPUT on;
2
3 create or replace function hike(eid number)
4 return number is
5 inc emp.sal%type;
6 vsal emp.sal%type;
7
8 BEGIN
9 select sal into vsal from emp where empno=eid;
10 if vsal<=1500 then
11 inc:=vsal*0.05;
12 elsif vsal>1500 and vsal <=2000 then
13 inc:=0.1*vsal;
14 else
15 inc:=0.15*vsal;
16 end if;
17 return(inc);
18 end hike;
```

Script Output x

Task completed in 0.029 seconds

Function HIKE compiled

```
3 DECLARE
4 sal_inc number(7,2);
5
6 BEGIN
7 sal_inc:= hike(7369);
8 dbms_output.put_line('Increment is '||sal_inc);
9 end;
```

Script Output x

Task completed in 0.024 seconds

Increment is 52.92

PL/SQL procedure successfully completed.