Database and Management System

Department of Computer Engineering

Khwaja Fareed University of Engineering and Information Technology
Rahim Yar Khan, Pakistan

Contents

1 Lab 1: Introduction 1

2 Lab 2: Familirization with DML 7

3 Lab 3: DRL(DATA RETRIEVAL LANGUAGE) 10

4 Lab 4: TCL,DCL and Constraints 15

5 Lab 5: selection, projection, sorting on a simple table 25

6 Lab 6: SQL Operators, 32

7 Lab 7: SQL Functions 34

8 Lab 8: Multi-table queries (JOIN OPERATIONS) 39

9 Lab 9: Nested queries 43

10 Lab 10: PL/SQL INTRODUCTION 47

11 Lab 11: PL/SQL, Loops 50

12 Lab 12: Programs using Cursors, Cursor loops and records 54

13 Lab 13:Stored procedures in PL/SQL 57

14 Lab 14:Functions in PL/SQL 60

15 Lab 15: Error handling and Triggers in PL/SQL 62

16 Lab 16: User Defined Types 65

Chapter 1

Lab 1: Introduction

Objectives

Introduction to Oracle and SQL.

Introduction

Oracle has many tools such as SQL * PLUS, Oracle Forms, Oracle Report Writer, Oracle Graphics etc.

- **SQL * PLUS**: The SQL * PLUS tool is made up of two distinct parts. These are
  - **interactive SQL**: Interactive SQL is designed for create, access and manipulate data structures like tables and indexes.
  - **PL/SQL**: PL/SQL can be used to developed programs for different applications.
- **Oracle Forms**: This tool allows you to create a data entry screen along with the suitable menu objects. Thus it is the oracle forms tool that handles data gathering and data validation in a commercial application.
- **Report Writer**: Report writer allows programmers to prepare innovative reports using data from the oracle structures like tables, views etc. It is the report writer tool that handles the reporting section of commercial application.
- **Oracle Graphics**: Some of the data can be better represented in the form of pictures. The oracle graphics tool allows programmers to prepare graphs using data from oracle structures like tables, views etc.

SQL (Structured Query Language)

Structured Query Language is a database computer language designed for managing data in relational database management systems(RDBMS), and originally based upon Relational Algebra. Its scope includes data query and update, schema creation and modification, and data access control. SQL was one of the first languages for Edgar F. Codd's relational model in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks" and became the most widely used language for relational databases.

- IBM developed SQL in mid of 1970's.
- Oracle incorporated in the year 1979.

- SQL used by IBM/DB2 and DS Database Systems.
- SQL adopted as standard language for RDBS by ASNI in 1989.

DATA TYPES

CHAR (Size) This data type is used to store character strings values of fixed length. The size in brackets determines the number of characters the cell can hold. The maximum number of character is 255 characters.

- VARCHAR (Size) / VERCHAR2 (Size): This data type is used to store variable length alphanumeric data. The maximum character can hold is 2000 character.
- NUMBER (P, S): The NUMBER data type is used to store number (fixed or floating point). Number of virtually any magnitude may be stored up to 38 digits of precision. Number as large as 9.99 * 10 124. The precision (p) determines the number of places to the right of the decimal. If scale is omitted then the default is zero. If precision is omitted, values are stored with their original precision up to the maximum of 38 digits.
- DATE: This data type is used to represent date and time. The standard format is DD-MM-YY as in 17-SEP-2009. To enter dates other than the standard format, use the appropriate functions. Date time stores date in the 24-Hours format. By default the time in a date field is 12:00:00 am, if no time portion is specified. The default date for a date field is the first day the current month.
- LONG: This data type is used to store variable length character strings containing up to 2GB. Long data can be used to store arrays of binary data in ASCII format. LONG values cannot be indexed, and the normal character functions such as SUBSTR cannot be applied.
- RAW: The RAW data type is used to store binary data, such as digitized picture or image. Data loaded into columns of these data types are stored without any further conversion. RAW data type can have a maximum length of 255 bytes. LONG RAW data type can contain up to 2GB.

INTERACTIVE SQL

Syntax : VERB(Parameter _1,Parameter _2,Parameter _3,........Parameter _n); SQL language is sub-divided into several language elements, including:

- Clauses, which are in some cases optional, constituent components of statements and queries.
- Expressions, which can produce either scalar values or tables consisting of columns and rows of data.
- Predicates which specify conditions that can be evaluated to SQL three-valued logic (3VL) Boolean truth values and which are used to limit the effects of statements and queries, or to change program flow.
- Queries which retrieve data based on specific criteria.
- Statements which may have a persistent effect on schemas and data, or which may control transactions, program flow, connections, sessions, or diagnostics.

- SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.
- Insignificant white space is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

There are five types of SQL statements. They are:

- DATA DEFINITION LANGUAGE (DDL)
- DATA MANIPULATION LANGUAGE (DML)
- DATA RETRIEVAL LANGUAGE (DRL)
- TRANSATIONAL CONTROL LANGUAGE (TCL)
- DATA CONTROL LANGUAGE (DCL)

The Data Definition Language (DDL)

The Data Definition Language (DDL) is used to

- Creating a table
- Altering table structure by adding, deleting or modifying columns
- Destroy databases and database objects.

These commands will primarily be used by database administrators during the setup and removal phases of a database project.

Create new Table

It defines each column of the table uniquely. Each column has minimum of three attributes, a name , data type and size.

Syntax

```
CREATE TABLE <table name> (<col1> <datatype>(<size>),<col2> <datatype><size>));
Ex:CREATE TABLE emp(empno number(4) primary key, ename char(10));
```

Add new column to existing table

Once table created within a database, one may wish to modify the definition of it. The ALTER command allows you to make changes to the structure of a table without deleting and recreating it.

Syntax

ALTER TABLE <tablename> add(<new col><datatype(size),<new col>datatype(size));
Ex:ALTER TABLE emp add(sal number(7,2));

Drop Column

In an existing table, a column can be deleted or dropped using the ALTER command.

Syntax

ALTER TABLE <tablename> drop column <col>;
Ex:ALTER TABLE emp drop column sal;

Modify Column

In an existing table, a column can be modified using the ALTER command.

Syntax

ALTER TABLE <tablename> modify(<col><newdatatype>(<newsize>));
Ex:ALTER TABLE emp modify(ename varchar2(15));

Renaming the table

We can rename a table using RENAME command.

Syntax

RENAME <oldtable> to <new table>;
Ex. RENAME emp to emp1;

Destroying tables

A table can be destroyed or deleted using DROP command.

Syntax

DROP TABLE <tablename>;
Ex. DROP TABLE emp1;

Summary

1: Create a table named BRANCH with following Structure

Data	Field Name	Type	Constraint
Branch Number	branchno	number(1)	Primary Key
Branch	branchname	varchar2(30)	Not Null

- Add the column 'emailid' to table BRANCH with Constraint UNIQUE.
- Modify the column named 'branch name' to Varchar2(25)
- Rename the table name to 'Branch Info'
- Delete the column named 'emailid'

2: Create a table named STUDENT with following Structure

Data	Field Name	Type	Constraint
Student Name	name	varchar2(30)	Not Null
Student Number	registerno	number(11)	Primary Key
Branch Number	branchno	number(1)	Foreign Key
Section	sec	varchar2(1)	Not Null
Joined Date	joindate	date	Not Null
Mark	mark	number(5,2)	Not Null

- Add the column 'emailid' to table STUDENT with Constraint UNIQUE.
- Modify the column named 'student name' to Varchar2(25)
- Rename the table name to 'Branch student data'
- Delete the column named 'emailid'

Practise Exercise

1: Create a table named MARKGRADE with following Structure

Data	**Field Name**	**Type**	**Constraint**
Grade	grade	varchar2(1)	Not Null
Lowest Mark	lowmark	number(5,2)	Not Null
Highest Mark	highmark	number(5,2)	Not Null

- Add the column ′student name′ to table STUDENT with Constraint varchar2(30).
- Modify the column named ′student name′ to Varchar2(25)
- Rename the table name to ′students marks grade′

2: Create a table named PROJECT with following Structure

Data	**Field Name**	**Type**	**Constraint**
Project Number	pno	number(3)	Primary Key
Project Name	pname	varchar2(60)	
Project Manager	pmgr	number(4)	Not Null
Persons	persons	number(3)	
Budjet	budjet	number(8,2)	
Project Start date	pstart	date	Not Null
Project End Date	pend	date	

Chapter 2

Lab 2: Familirization with DML

Objectives

To write queries using Insert, Select, Update, and Delete Commands in Data Manipulation Language (DML).

Introduction

Data Manipulation Language (DML) is used by computer programs or database users to retrieve, insert, delete and update data in a database. Currently, the most popular data manipulation language is that of SQL, which is used to retrieve and manipulate data in a Relational database

SELECT

An SQL SELECT statement returns a result set of records from one or more tables. It retrieves zero or more rows from one or more base tables or views in a database.

Example:
SELECT * FROM student;
* means all columns in a table The SELECT statement has many optional clauses:

INSERT

The number of columns and values must be the same. The values specified (or implied) by the INSERT statement must satisfy all the applicable constraints (such as primary keys, CHECK constraints, and NOT NULL constraints). If a syntax error occurs or if any constraints are violated, the new row is not added to the table and an error returned instead.

Syntax

- INSERT INTO table (column1, [column2, ...]) VALUES (value1, [value2, ...])
- INSERT INTO table VALUES (value1, [value2, ...])

Example

- student(name,registerno,branchno,section,joindate,mark,emailid) VALUES ('Aarthi',11306205001,1,'A','01-apr-2006',99, 'abc@gmail.com')

- INSERT INTO student VALUES ('Aarthi',11306205001,1,'A','01-apr-2006',99, 'abc@gmail.com')

UPDATE

A SQL UPDATE statement that changes the data of one or more records in a table. Either all the rows can be updated, or a subset may be chosen using a condition.

Syntax

UPDATE table_name SET column_name = value [, column_name = value ...] [WHERE condition]

Example

UPDATE student SET mark = mark + 2 WHERE registerno = 11306205001;

DELETE

An SQL DELETE statement removes one or more records from a table. A subset may be defined for deletion using a condition, otherwise all records are removed. Any rows that match the WHERE condition will be removed from the table. If the WHERE clause is omitted, all rows in the table are removed. The DELETE statement should thus be used with caution!

Syntax

DELETE FROM table_name [WHERE condition]

Example

DELETE FROM student WHERE registerno = 11306205001;

Summary

1: Insert following Values into the Table Branch.

BRANCHNO	BRANCHNAME
1	Civil
2	Computer Science
3	Electrical
4	Electronics
5	Information Technology
6	Instrumentation
7	Mechanical
8	MBA
9	MCA

2: Insert following Values into the Table Student.
3: Display the contents of Student Table.
4: Display the contents of Branch Table.

NAME	REGISTERNO	BRANCHNO	SECTION	JOINDATE	MARK
ADITYA KUMAR	11305104001	2	A	24-JUL-2005	99
NANDA KUMARAN	11305104061	2	B	30-MAY-2005	80
AASHA RANI	11308104001	2	A	02-AUG-2008	99
MANDYAM HARIKA	11308106061	4	B	04-AUG-2008	75
ABDUL SAMAD	11308205001	5	A	02-AUG-2008	100
PRIYA DHARSHINI	11308205062	5	B	18-SEP-2008	90
AISHWARYA	11308103001	1	A	09-AUG-2008	78
RAMMANOHAR	11308103030	1	A	01-AUG-2008	65
AHAMED THAHA	11308114001	7	A	25-JUL-2008	90
PRAVEEN	11308114025	7	A	28-JUL-2008	75
ABINAYA	11308105001	3	A	01-AUG-2008	90
ANISH KUMAR	11308107001	6	A	26-JUL-2008	90
KANIMOZHI	11308107027	6	A	06-AUG-2008	80
MADHU USOODHANA	11308105023	3	A	28-JUL-2008	90

Practise Exercise

1: Create and insert values in Table named EMP

empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	SMITH	CLERK	7902	1993-06-13	800.00	0.00	20
7499	ALLEN	SALESMAN	7698	1998-08-15	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1996-03-26	1250.00	500.00	30
7566	JONES	MANAGER	7839	1995-10-31	2975.00		20
7698	BLAKE	MANAGER	7839	1992-06-11	2850.00		30
7782	CLARK	MANAGER	7839	1993-05-14	2450.00		10
7788	SCOTT	ANALYST	7566	1996-03-05	3000.00		20
7839	KING	PRESIDENT		1990-06-09	5000.00	0.00	10
7844	TURNER	SALESMAN	7698	1995-06-04	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1999-06-04	1100.00		20
7900	JAMES	CLERK	7698	2000-06-23	950.00		30
7934	MILLER	CLERK	7782	2000-01-21	1300.00		10
7902	FORD	ANALYST	7566	1997-12-05	3000.00		20
7654	MARTIN	SALESMAN	7698	1998-12-05	1250.00	1400.00	30

2: Update the 'sal' of 'Scott' to 1500.00.
3: Update the emp table to set the salary of all employees to Rs 15000.00 who are working as Clerk.
4: select employee name, job from the emp table.
5: Delete only those who are working as President.

Chapter 3

Lab 3: DRL(DATA RETRIEVAL LANGUAGE)

Objectives

Introduction and practise of DRL

Introduction

WHERE

WHERE specifies which rows to retrieve.
A WHERE clause in SQL specifies that a SQL Data Manipulation Language (DML) statement should only affect rows that meet a specified criteria. WHERE clauses are not mandatory clauses of SQL DML statements, but should be used to limit the number of rows affected by a SQL DML statement or returned by a query. WHERE is an SQL reserved word.
EXAMPLE : SELECT * FROM student WHERE mark > 90

GROUP BY

GROUP BY groups rows sharing a property so that an aggregate function can be applied to each group.
A GROUP BY statement in SQL specifies that a SQL SELECT statement returns a list that is grouped by one or more columns, usually in order to apply some sort of aggregate function to certain columns.
SELECT branchno, AVG(mark) FROM student GROUP BY branchno;

HAVING

HAVING selects among the groups defined by the GROUP BY clause.
A HAVING statement in SQL specifies that a SQL SELECT statement should only return rows where aggregate values meet the specified conditions.
Example:
SELECT branchno, AVG(mark) FROM student GROUP BY branchno HAVING AVG(mark) > 80;

ORDER BY

ORDER BY specifies an order in which to return the rows.
An ORDER BY clause in SQL specifies that a SQL SELECT statement returns a result set with the rows being sorted by the values of one or more columns. ORDER BY is the only way to sort the rows in the result set. Without this clause, the relational database system may return the rows in any order.
Example:
SELECT * FROM student ORDER BY registerno;
SELECT * FROM student ORDER BY mark;

. JOIN using SELECT - FROM - ORDER BY

This query is used to display a set of fields from two relations by matching a common field in them in an ordered manner based on some fields.
Syntax SELECT a set of fields from both relations FROM relation1, relation2 WHERE relation1.field _x = relation2.field _y ORDER BY field _z; **Example** SQL>SELECT empno,ename,job,dname FROM emp,dept WHERE emp.deptno = 20 ORDER BY job;

```
EMPNO     ENAME        JOB DNAME
------      ------    ----------------
7788 SCOTT ANALYST ACCOUNTING
7902 FORD ANALYST ACCOUNTING
------
7566 JONES MANAGER OPERATIONS
7566 JONES MANAGER SALES
  20 rows selected.
```

JOIN using SELECT - FROM - GROUP BY

This query is used to display a set of fields from two relations by matching a common field in them and also group the corresponding records for each and every value of a specified key(s) while displaying.
Syntax SELECT a set of fields from both relations FROM relation1,relation2 WHERE relation1.field _x=relation2.field _y GROUP BY field _z;
Example SQL> SELECT empno,SUM(SALARY) FROM emp,dept WHERE emp.deptno =20 GROUP BY empno;

```
EMPNO   SUM (SALARY)
-------   --------
7369 3200
7566 11900
7788 12000
7876 4400
```

UNION

This query is used to display the combined rows of two different queries, which are having the same structure, without duplicate rows.

Syntax SELECT field _1,field _2,....... FROM relation1 WHERE (Condition) UNION SELECT field _1,field _2,....... FROM relation2 WHERE (Condition); **Example** SQL> SELECT * FROM STUDENT;

```
SNO  SNAME
-----------
1  kumar
2  ravi
3  ramu
```

SQL> SELECT * FROM STD;

```
SNO  SNAME
-----------
3  ramu
5  lalitha
9  devi
1  kumar
```

SQL> SELECT * FROM student UNION SELECT * FROM std;

```
SNO  SNAME
----  ------
1  kumar
2  ravi
3  ramu
5  lalitha
```

INTERSET

This query is used to display the common rows of two different queries, which are having the same structure, and to display a selected set of fields out of them. **Syntax** SELECT field _1,field _2,.. FROM relation1 WHERE (Condition) INTERSECT SELECT field _1,field _2,.. FROM relation2 WHERE(Condition); **Example** SQL> SELECT * FROM student INTERSECT SELECT * FROM std;

```
SNO SNAME
----------
1 Kumar
```

MINUS

This query is used to display all the rows in relation1,which are not having in the relation2. Syntax: SELECT field _1,field _2,......FROM relation1 WHERE(Condition) MINUS SELECT field _1,field _2,..... FROM relation _2 WHERE(Conditon); SQL> SELECT * FROM student MINUS SELECT * FROM std;

```
SNO   SNAME
----------
2 RAVI
3 RAMU
```

Summary

1- Create and insert random values in table named ’Student’ Student(name, registration No., CNIC, DOB, Class, Department No, Class incharge, monthly Fee)
Note: table must have following departments CS,IT,Engineering,Arts
2- Create and insert values in table named ’Department’ Department(Dept No, Dept Name, total students, total courses)
3- Perform following queries on above two tables.

- Show all items of ’Student’ table.
- Show all items of ’Department’ table.
- Display the details of students who are in CS Department.
- Display the name of the Department No of IT Department.
- Display the Number of students in each department.
- Display the Name and registration of students whose fees is greater than Rs.500
- Display the Maximum fees that a student pay.
- Display the minimum fees that a student pay.

Practise Exercise

1- Create and insert values in the table named 'Project'

Data	Field Name	DataType	Constraint
Project Number	pno	number(3)	Primary Key
Project Name	pname	varchar2(60)	
Project Manager	pmgr	number(4)	Not Null
Persons	persons	number(3)	
Budjet in Rupees	budjet	number(8,2)	
Project Start date	pstart	date	Not Null
Project End Date	pend	date	

Perform the following queries on above table 'Project'
Write the Queries and show the results.

- Display all items of table
- Display only Project name and Budjet of every project
- Display the project name and project manager whoes budget is lowest
- Display the project name and project manager whoes budget is highest
- Arrange the table in ascending order according to the project start date.
- Display only those manager whose persons are more than 5.
- Delete the record of those manager whoes budget is the lowest.
- Update the Datatype of Project name to varchar2(30)

Chapter 4

Lab 4: TCL,DCL and Constraints

Objectives

Introduction to TRANSATIONAL CONTROL LANGUAGE (T.C.L), DATA CONTROL LANGUAGE (D.C.L) and Constraints

TRANSATIONAL CONTROL LANGUAGE (T.C.L)

A transaction is a logical unit of work. All changes made to the database can be referred to as a transaction. Transaction changes can be mode permanent to the database only if they are committed a transaction begins with an executable SQL statement ends explicitly with either role back or commit statement.

COMMIT

This command is used to end a transaction only with the help of the commit command transaction changes can be made permanent to the database.
Syntax
SQL>COMMIT;
Example
SQL>COMMIT;

SAVE POINT

Save points are like marks to divide a very lengthy transaction to smaller once. They are used to identify a point in a transaction to which we can latter role back. Thus, save point is used in conjunction with role back.
Syntax
SQL>SAVE POINT ID;
Example
SQL>SAVE POINT xyz;

ROLE BACK

A role back command is used to undo the current transactions. We can role back the entire transaction so that all changes made by SQL statements are undo (or) role back a transaction

to a save point so that the SQL statements after the save point are role back.
Syntax
ROLE BACK(current transaction can be role back)
ROLE BACK to save point ID;
Example
SQL>ROLE BACK;
SQL>ROLE BACK TO SAVE POINT xyz;

DATA CONTROL LANGUAGE (D.C.L)

DCL provides uses with privilege commands the owner of database objects (tables), has the soul authority ollas them. The owner (data base administrators) can allow other data base uses to access the objects as per their requirement

GRANT

The GRANT command allows granting various privileges to other users and allowing them to perform operations with in their privileges For Example, if a uses is granted as 'SELECT' privilege then he/she can only view data but cannot perform any other DML operations on the data base object GRANTED privileges can also be withdrawn by the DBA at any time.
Syntax
SQL>GRANT PRIVILEGES on object-name To user-name;
Example
SQL>GRANT SELECT, UPDATE on emp To hemanth;

REVOKE

To with draw the privileges that has been GRANTED to a uses, we use the REVOKE command
Syntax
SQL>REVOKE PRIVILEGES ON object-name FROM user_name;
Example
SQL>REVOKE SELECT, UPDATE ON emp FROM ravi;
1. Creation, altering and dropping of tables and inserting rows into a table (use constraints while creating tables) examples using SELECT command.

CREATE

CREATE TABLE
This is used to create a new relation.
Syntax
CREATE TABLE relation-name
(field-1 data-type(Size),field-2 data-type(Size), .. .);
Example
SQL>CREATE TABLE Student (sno NUMBER(3) PRIMARY KEY ,sname CHAR(10),class CHAR(5);

ALTER

(a)ALTER TABLE ...ADD...: This is used to add some extra fields into existing relation.
Syntax
ALTER TABLE relation-name ADD(new field-1 data-type(size), new field-2 data-type(size),..);
Example
SQL>ALTER TABLE std ADD(Address CHAR(10));
(b)ALTER TABLE...MODIFY...: This is used to change the width as well as data type of fields of existing relations.
Syntax
ALTER TABLE relation-name MODIFY (field-1 newdata-type(Size), field-2 newdata-type(Size),....field-newdata-type(Size));
Example
SQL>ALTER TABLE student MODIFY(sname VARCHAR(10),class VARCHAR(5));

Constraints

There are 5 constraints available in ORACLE:

NOT NULL

When a column is defined as NOTNULL, then that column becomes a mandatory column. It implies that a value must be entered into the column if the record is to be accepted for storage in the table.
Syntax
CREATE TABLE Table-Name(column-name data-type(size) NOT NULL,);
Example
CREATE TABLE student (sno NUMBER(3)NOT NULL, name CHAR(10));

UNIQUE

The purpose of a unique key is to ensure that information in the column(s) is unique i.e. a value entered in column(s) defined in the unique constraint must not be repeated across the column(s). A table may have many unique keys.
Syntax
CREATE TABLE Table-Name(column-name data-type(size) UNIQUE, —);
Example
CREATE TABLE student (sno NUMBER(3) UNIQUE, name CHAR(10));

CHECK

Specifies a condition that each row in the table must satisfy. To satisfy the constraint, each row in the table must make the condition either TRUE or unknown (due to a null).
Syntax
CREATE TABLE Table-Name(column-name data-type(size) CHECK(logical expression),–);
Example

CREATE TABLE student (sno NUMBER (3), name CHAR(10),class CHAR(5),CHECK(class IN('CSE','CAD','VLSI'));

PRIMARY KEY

A field which is used to identify a record uniquely. A column or combination of columns can be created as primary key, which can be used as a reference from other tables. A table contains primary key is known as Master Table.

- It must uniquely identify each record in a table.
- It must contain unique values.
- It cannot be a null field.
- It cannot be multi port field.
- It should contain a minimum no. of fields necessary to be called unique.

Syntax
CREATE TABLE Table-Name(column-name data-type(size) PRIMARY KEY,—); **Example**
CREATE TABLE faculty (fcode NUMBER(3) PRIMARY KEY, fname CHAR(10));

FOREIGN KEY

It is a table level constraint. We cannot add this at column level. To reference any primary key column from other table this constraint can be used. The table in which the foreign key is defined is called a detail table. The table that defines the primary key and is referenced by the foreign key is called the master table.
Syntax
CREATE TABLE Table-Name(column-name data-type(size) FOREIGN KEY(column-name) REFERENCES table-name);
Example
CREATE TABLE subject (scode NUMBER (3) PRIMARY KEY,subname CHAR(10),fcode NUMBER(3),FOREIGN KEY(fcode) REFERENCE faculty);
Defining integrity constraints in the alter table command:
Syntax
ALTER TABLE Table-Name ADD PRIMARY KEY (column-name);
Example
ALTER TABLE student ADD PRIMARY KEY (sno);
(Or)
Syntax
ALTER TABLE table-name ADD CONSTRAINT constraint-name PRIMARY KEY(colname)
Example
ALTER TABLE student ADD CONSTRAINT SN PRIMARY KEY(SNO)
Dropping integrity constraints in the alter table command:
Syntax
ALTER TABLE Table-Name DROP constraint-name;
Example
ALTER TABLE student DROP PRIMARY KEY;
(or)
Syntax

ALTER TABLE student DROP CONSTRAINT constraint-name;
Example
ALTER TABLE student DROP CONSTRAINT SN;
Queries using Aggregate functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and dropping of Views.

Aggregative operators

In addition to simply retrieving data, we often want to perform some computation or summarization. SQL allows the use of arithmetic expressions. We now consider a powerful class of constructs for computing aggregate values such as MIN and SU

1. Count
COUNT following by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (*) indicates all the tuples of the column.
Syntax
COUNT (Column name)
Example
SELECT COUNT (Sal) FROM emp;
2. SUM
SUM followed by a column name returns the sum of all the values in that column.
Syntax
SUM (Column name)
Example
SELECT SUM (Sal) From emp;
3. AVG
AVG followed by a column name returns the average value of that column values.
Syntax
AVG (n1,n2..)
Example
Select AVG(10, 15, 30) FROM DUAL;
4. MAX
MAX followed by a column name returns the maximum value of that column.
Syntax
MAX (Column name)
Example
SELECT MAX (Sal) FROM emp;
SQL> select deptno,max(sal) from emp group by deptno;

```
DEPTNO   MAX(SAL)
------   --------
10        5000
20        3000
30        2850
```

SQL> select deptno,max(sal) from emp group by deptno having max(sal)<3000;

```
DEPTNO  MAX(SAL)
-----   --------
30      2850
```

5. MIN
MIN followed by column name returns the minimum value of that column.
Syntax
MIN (Column name)
Example
SELECT MIN (Sal) FROM emp;
SQL>select deptno,min(sal) from emp group by deptno having min(sal)>1000;

```
DEPTNO   MIN(SAL)
-----    --------
10        1300
```

VIEW

In SQL, a view is a virtual table based on the result-set of an SQL statement.
A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.
A view is a virtual table, which consists of a set of columns from one or more tables. It is similar to a table but it doest not store in the database. View is a query stored as an object.
Syntax
CREATE VIEW view-name AS SELECT set of fields FROM relation-name
WHERE (Condition)
Example
SQL>CREATE VIEW employee AS SELECT empno,ename,job FROM EMP WHERE job = 'clerk';
View created.

```
SQL> SELECT * FROM EMPLOYEE;
EMPNO ENAME JOB
-----------------
7369 SMITH CLERK
7876 ADAMS CLERK
7900 JAMES CLERK
7934 MILLER CLERK
```

Example
CREATE VIEW [Current Product List] AS SELECT ProductID,ProductName FROM Products WHERE Discontinued=No
DROP VIEW
This query is used to delete a view , which has been already created.
Syntax
DROP VIEW View-name;
Example
SQL> DROP VIEW EMPLOYEE;
View dropped

```
Queries using Conversion functions (to_char, to_number and to_date),
string functions (Concatenation, lpad, rpad, ltrim, rtrim, lower, upper,
initcap, length, substr and instr), date functions (Sysdate, next_day,
add_months, last_day, months_between, least, greatest, trunc, round,
to_char, to_date)
```

Conversion functions

To-char
TO-CHAR (number) converts n to a value of VARCHAR2 data type, using the optional num-

ber format fmt. The value n can be of type NUMBER, BINARY-FLOAT, or BINARY-DOUBLE.
SQL>select to-char(65,'RN')from dual;
LXV
To-number
TO-NUMBER converts expr to a value of NUMBER data type.
SQL> Select to-number('1234.64') from Dual;
1234.64
To-date
TO-DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE data type.
SQL>SELECT TO-DATE('January 15, 1989, 11:00 A.M.')FROM DUAL;
TO-DATE('
———
15-JAN-89

String functions

Concat
CONCAT returns char1 concatenated with char2. Both char1 and char2 can be any of the datatypes
SQL>SELECT CONCAT('ORACLE','CORPORATION')FROM DUAL;
ORACLECORPORATION
Lpad
LPAD returns expr1, left-padded to length n characters with the sequence of characters in expr2.
SQL>SELECT LPAD('ORACLE',15,'*')FROM DUAL;
*********ORACLE
Rpad
RPAD returns expr1, right-padded to length n characters with expr2, replicated as many times as necessary.
SQL>SELECT RPAD ('ORACLE',15,'*')FROM DUAL;
ORACLE*********
Ltrim
Returns a character expression after removing leading blanks.
SQL>SELECT LTRIM('SSMITHSS','S')FROM DUAL;
MITHSS
Rtrim
Returns a character string after truncating all trailing blanks
SQL>SELECT RTRIM('SSMITHSS','S')FROM DUAL;
SSMITH
Lower
Returns a character expression after converting uppercase character data to lowercase.
SQL>SELECT LOWER('DBMS')FROM DUAL;
dbms
Upper
Returns a character expression with lowercase character data converted to uppercase.
SQL>SELECT UPPER('dbms')FROM DUAL;
DBMS
Length

Returns the number of characters, rather than the number of bytes, of the given string expression, excluding trailing blanks.
SQL>SELECT LENGTH('DATABASE')FROM DUAL;
8
Substr
Returns part of a character, binary, text, or image expression.
SQL>SELECT SUBSTR('ABCDEFGHIJ'3,4)FROM DUAL;
CDEF
Instr
The INSTR functions search string for substring. The function returns an integer indicating the position of the character in string that is the first character of this occurrence.
SQL>SELECT INSTR('CORPORATE FLOOR','OR',3,2)FROM DUAL;
14

Date functions

Sysdate
SQL>SELECT SYSDATE FROM DUAL;
29-DEC-08
next_day
SQL>SELECT NEXT_DAY(SYSDATE,âĂŹWEDâĂŹ)FROM DUAL;
05-JAN-09
add_months
SQL>SELECT ADD_MONTHS(SYSDATE,2)FROM DUAL;
28-FEB-09
last_day
SQL>SELECT LAST_DAY(SYSDATE)FROM DUAL;
31-DEC-08
months_between
SQL>SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE)FROM EMP;
4
Least
SQL>SELECT LEAST('10-JAN-07','12-OCT-07')FROM DUAL;
10-JAN-07
Greatest
SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;
10-JAN-07
Trunc
SQL>SELECT TRUNC(SYSDATE,'DAY')FROM DUAL;
28-DEC-08
Round
SQL>SELECT ROUND(SYSDATE,'DAY')FROM DUAL;
28-DEC-08
to_char
SQL> select to_char(sysdate, "dd mm yy") from dual;
24-mar-05.
to_date
SQL> select to _date(sysdate, "dd mm yy") from dual;
24-mar-o5.

Practise Exercise

1- Display all employee names and salary whose salary is greater than minimum salary of the company and job title starts with M.
2-Issue a query to find all the employees who work in the same job as Arjun.
3- Display all the dept numbers available with the dept and emp tables.

Chapter 5

Lab 5: selection, projection, sorting on a simple table

Objectives

Simple queries: selection, projection, sorting on a simple table

Creating The Tables

1 Create table EMPLOYEE with the following attributes and then insert the following data in to EMPLOYEE table.

```
NAME        MINIT    LNAME     SSN            BDATE             ADDRESS
SEX   SAL       SUPERSSN           DNO
---------------------------------------------------------------------------------------------
John          B     Smith 123456789    09-JAN-1965      731 Fondren, Houston, TX
M      30000 333445555          5
Franklin     T   Wong 333445555    08-DEC-1955      638 Voss, Houston, TX
M      40000 888665555       5
Alicia     J    Zelaya 999887777    19-JUL-1968        3321, Castle, Spring, TX
F      25000 987654321       4
Jennifer     S Wallace 987654321    20-JUN-1941      291, Berry, Bellaire, TX
F      43000 888665555       4
Ramesh   K    Narayan 666884444    15-SEP-1962      975 Fire Oak, Humble, TX
M     38000     333445555          5
Joyce    A    English 453453453     31-JUL-1972        5631 Rice, Houston, TX
F      25000 333445555       5
Ahmad    V    Jabbar  987987987        29-MAR-1969     980 Dallas, Houston, TX
M      25000  987654321        4
James      E     Borg     888665555     10-NOV-37     450 Stone, Houston, Tx
M     55000                     1
```

Ans:

2. Create table DEPARTMENT with the following attributes and then insert the following data into DEPARTMENT table.

```
DNAME        DNUMBER        MGRSSN           MGRSTARTDATE
-----------------------------------------------------------------------------------------------
Research        5           333445555       22-MAY-1988
Administration  4           987654321          01-JAN-1995
Headquarters    1           888665555          19-JUN-1981
```

Ans:

3. Create table DEPT_LOCATIONS with the following attributes and insert the following data into DEPT_LOCATIONS table.

```
 DNUMBER DLOCATION
 ---------------------------------------------
     1 Houston
    4 Stafford
    5 Bellaire
    5 Sugarland
    5 Houston
```

Ans:

4.Create table PROJECT with the following attributes and insert the following data into PROJECT table.

```
 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
```

Ans:

5.Create table DEPENDENT with the following attributes and insert the following data into DEPENDENT table.

```
 ESSN       DEPENDENT_NAME   SEX BDATE RELATIONSHIP
---------------------------------------------------------------------------------------------------
333445555   Alice              F   05-APR-86   DUAGHTER
333445555   Theodore           M   25-OCT-83   SON
333445555   Joy                F   03-MAY-58   SPOUSE
987654321   Abner              M   28-FEB-42   SPOUSE
123456789   Michael            M   04-JAN-88   SON
123456789   Alice              F   30-DEC-88   DAUGHTER
123456789   Elizabeth          F   05-MAY-67   SPOUSE
```

Ans:

6. Create table WORKS_ON with the following attributes and then insert the following data into WORKS_ON table.

```
 ESSN            PNO    HOURS
------------------------------------------------
123456789        1         32.5
123456789        2         7.5
666884444        3         40
453453453        1         20
453453453        2         20
333445555        2         10
333445555        3         10
333445555        10        10
333445555        20        10
999887777        30        30
999887777        10        10
987987987        10        35
987987987        30        5
987654321        30        20
987654321        20        15
888665555        20
```

Ans:

Assigning Key Attributes to Created Tables

- Create table employee (fname varchar(15) NOT NULL,minit char(5),lname varchar2(15) NOT NULL,ssn varchar2(9) NOT NULL,bdate date,address varchar2(30),sex char(3),salary decimal(10,2),superssn varchar2(9),dno number(7) NOT NULL,PRIMARY KEY(ssn));

(Or)

Alter Table Employee add Primary key (ssn);

- Create table department (dname varchar2(15) NOT NULL,dnumber number(7) NOT NULL, mgrssn varchar2(9),mgrstartdate date,PRIMARY KEY(dnumber), UNIQUE(dname));

  (Or)

  Alter Table department add Primary key (dnumber);

- Create table dept_locations(dnumber number(7) NOT NULL,dlocation varchar2(15) NOT NULL, PRIMARY KEY(dnumber,dlocation));

  (Or)

  Alter Table dept_locations add Primary key (dnumber,dlocation);

- create table project(pname varchar2(15) NOT NULL,pnumber number(7) NOT NULL,plocation varchar2(15),dnum number(4) NOT NULL,PRIMARY KEY(pnumber), UNIQUE(pname));

  (Or)

  Alter Table project add Primary key (pnumber);

- create table dependent(essn varchar2(9) NOT NULL,dependent_name varchar2(15) NOT NULL,sex char(3),bdate date,relationship varchar2(12),PRIMARY KEY(essn,dependent_name));

  (Or)

  Alter Table dependent add Primary key (essn,dependent_name);

- create table works_on(essn varchar2(9) NOT NULL,pno number(3) NOT NULL,hours decimal(4,1) ,PRIMARY KEY(essn,pno));

  (Or)

  Alter Table works_on add Primary key (essn,pno);

Simple queries

1.write a query to display emp table.

2. List all employee values.

3. List empno,empname and salary.

4.List the names of all MANAGERS.

5.List all clerks in deptno. 30.

6.List all employee names whose mgr no is 7698.

7.List jobs dept 20.

8.List employee names whose salary is between 2000 and 3000.

9.List employee in the dependent 10,20.

10.list employee names which begin with S.

11.List employee names having 'A' in their names.

12.List employee who have joined in JAN.

13.List employees who have joined in the year 81.

14.List all distinct jobs.

15.List employee names in alphabetical order.

16.List employee names alphabetically department by deptno.

17.List employee numbers,name sal,DA(15

18.List employee names having an experience more than15 years.

19.List employee names whose commission is NULL.

20.List employee who do not report to anybody.

21.List maximum sal,minimum sal,average sal,

22.List the numbers of people and average slary in deptno 30.

23.List maximum sal and minimum sal in the designation SALESMAN and CLERK.

24.List the numbers of people and average salary of employee joined in 81,82 and 83.

25.List jobs that are unique to deptno 20 .

26.List employee names and their joining date in the folloeing formats

```
A.SMITH   17th DEC NINETEEEN EIGHTY
B.SMITH   SEVENTEENTH DEC NINTEEN EIGHTY
C.SMITH    week day of joining
D.SMITH     17/12/80
```

27.List employee names and their experience in yesrs.

28.Display a given date as a string in different formats.

29. List employee names with length of the names sorted on length.

30.List employee names with length of the name sorted on lengrh.

31.List employee names with length of the name sorted on lengrh.

32.List employee names and the string with out first character and last character in their name.

33.SQL>select deptno,min(sal),max(sal),avg(sal) from emp where deptno in(10,30) group by deptno;

34.SQL>select deptno,min(sal),max(sal),avg(sal) from emp group by deptno;

35.SQL>select job,min(sal),max(sal),avg(sal) from emp group by job;

36.SQL>select deptno,min(sal),max(sal),count(8) from emp group by deptno having count(*)>=2;

37.List employee names and dept names with which they are associated.

Chapter 6

Lab 6: SQL Operators,

Sql Operators (Simple-complex conditions)

1)Between,And:
SELECT FNAME, SALARY FROM EMPLOYEE WHERE SALARY BETWEEN 20000 AND 30000;
2)Like:
1. SELECT FNAME FROM EMPLOYEE WHERE LNAME LIKE'S%';
2. Write a query to retrieve all employees who were born during the 1950s
Ans:

IN:
SELECT FNAME FROM EMPLOYEE WHERE DNO IN(1,4,5);
NOT:
SELECT FNAME FROM EMPLOYEE WHERE DNO NOT IN(1,2,3);

ASC-DESC ordering combinations

1) Write a query to retrieve names and salaries of employees in the descending order of their salaries.
Ans:

2) Write a query to retrieve names and salaries of employees in the Ascending order of their names.
Ans:

F) Renaming attributes:
1) Rename Employee table to Employees?
Ans:

G) Delete command:
Delete from employees where ssn=123456789;
Ans:

H) ALTER: Used to modify or add or delete an attribute
1. To add a column for department table
Alter table department add dspl varchar2(5);
Ans:

2. To modify already exsisting column
Alter table department modify dloc varchar2(10);
Ans:

3. To delete a column
Alter table dept drop column dspl;
Ans:

I) DROP: used to delete a table.
Syntax: Drop table tablename;
Ans:

Chapter 7

Lab 7: SQL Functions

Objectives

Familirization with SQL Functions

SQL Functions

These functions are used to manipulating data items and returning the results.

- Group functions or Aggregate functions.
- Single Row or scalar function.

Group functions or Aggregate functions

These functions operated a set of values or rows

- Sum()
- Avg()
- Min()
- Max()
- Count()

Sum()
Sum() used to find out the sum of the salary of employees.
Ex:List the sum of the salary of employees
Select sum(sal) from emp;
Output

Avg()

It find out the average salaries of employees.
Ex:List the average salary of the employees
Select avg(sal) from emp;
Output

Min()
Min() used to find out the minimum salary of an employee in the given table.
Ex:list out the minimum salary of an employee in the emp table.
Select min(sal) from emp;
Output

Max()
Max() used to find out the maximum salary of an employee in the given table.
Ex:list out the maxiimum salary of an employee in the emp table.
Select max(sal) from emp;
Output

Count()
Count() used to list out the number of values in a particular table.
Ex:
1.List the numbers of jobs.
select count (job) from emp;
Output

2.List the numbers of people and average salary in deptno 30.
select count(*),avg(sal) from emp where deptno=30;
Output

Single Row or scalar function

These functions are operated a single row at a time.

Abs()

Abs() used to find the absolute value.

Select abs(10) from dual;

Output

Power()

Power():find the power.

Select power(2,3) from dual;

Output

Sqrt()

Sqrt() used to find the square root of a given value.

Select sqrt(9) from dual;

Output

Round()

Round() used to find the round of the value.

Select round(12.35,1) from dual;

Output

Truncate()

Truncate() used to find the truncate value.

Select trunc(12.35,1) from dual;

Output

Exp()
Exp() used to used to find the exponential of given number.
Select exp(3) from dual;
Output

Greatest()
Greastest() used to find out the greater value.
Select greatest(10,20,30) from dual;
Output

Least()
Least() used to find out the leastervalue.
Select least(10,20,30) from dual;
Output

Mod()
Mod() used to fina tha module of given numbers.
Select mod(3,2) from dual;
Output

Floor()
Floor() used to find the floor value.
Select floor(12.56) from dual;
Output

Sign()
Sign() used to find the sign of a number.

Select sign(-10) from dual;
Output

Select sign(10) from dual;
Output

Log()
Log() used to find logarithmic value.
Select log(3,2) from dual;
Output

In these function we are using date functions also there are listed below:

```
 Select months_between('26-jun-06','25-may-06') from dual;
MONTHS_BETWEEN('26-JUN-09','25-MAY-09')
1.03225806
Select add_months('26-jun-06',5') from dual;
ADD_MONTH
26-NOV-06
Select next_day('26-jun-09','monday') from dual;
NEXT_DAY(
29-JUN-09
Select last_day('26-jun-09') from dual;
LAST_DAY(
30-JUN-09
```

Chapter 8

Lab 8: Multi-table queries (JOIN OPERATIONS)

Objectives

Introduction of Multi-table queries (JOIN OPERATIONS)

Introduction

Here are the different types of the JOINs in SQL:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Return all records when there is a match in either left or right table

INNER JOIN

The INNER JOIN keyword selects records that have matching values in both tables.
Syntax

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

LEFT OUTER JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.
Syntax

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

RIGHT OUTER JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.
Syntax

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

FULL OUTER JOIN

The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.
Note: FULL OUTER JOIN can potentially return very large result-sets!
Syntax

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```

Practise Exercise

1. Write a query to retrieve name and address of all employees who work for the 'Research' department.

2. Write a query to retrieve employee's first and last name and first and last name of his or her immediate supervisor.

3. Write a query to retrieve list of employees and the projects they are working on, ordered by department and with in each department, ordered alphabetically by last name, first name.

4. For every project located in 'Stafford', list the project number, the controlling department number and the department manager's last name, birth date.

5. Find the sum of the salaries of all employees, the maximum salary, the minimum salary and the average salary.

6. Find the sum of the salaries of all employees, the maximum salary, the minimum salary and the average salary of all employees of the 'Research' department.

7. Count the number of employees working in the 'Research' department.

8. For each department, retrieve the department number, the number of employees in the department and their average salary.

9. For each project, retrieve the project number, Project name and the number of employees who work on that project.

10. For each project on which more than two employees work, retrieve the project number, project name and the number of employees who work on the project.

11. For each project, retrieve the project number, Project name and the number of employees from department 5 who work on the project.

Chapter 9

Lab 9: Nested queries

Objectives

Understanding of Nested queries

Introduction

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.
A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.
Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
There are a few rules that subqueries must follow:

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

Subqueries with the SELECT Statement

Subqueries are most frequently used with the SELECT statement.
The basic syntax is as follows

```
\\
SELECT column_name [, column_name ]
FROM   table1 [, table2 ]
WHERE  column_name OPERATOR
(SELECT column_name [, column_name ]
FROM table1 [, table2 ]
[WHERE])
```

Subqueries with the INSERT Statement

Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. The selected data in the subquery can be modified with any of the character, date or number functions.
The basic syntax is as follows.

```
INSERT INTO table_name [ (column1 [, column2 ]) ]
SELECT [ *|column1 [, column2 ]
FROM table1 [, table2 ]
[ WHERE VALUE OPERATOR ]
```

Subqueries with the UPDATE Statement

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.
The basic syntax is as follows.

```
UPDATE table
SET column_name = new_value
[ WHERE OPERATOR [ VALUE ]
   (SELECT COLUMN_NAME
   FROM TABLE_NAME)
   [ WHERE) ]
```

Subqueries with the DELETE Statement

The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above.
The basic syntax is as follows.

```
DELETE FROM TABLE_NAME
[ WHERE OPERATOR [ VALUE ]
   (SELECT COLUMN_NAME
   FROM TABLE_NAME)
   [ WHERE) ]
```

Practise Exercise

1. Write a nested query to retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.

2. [Using exists]

3. Write a query to show resulting salaries if every employee working on 'ProductX' project is given a 10 percent raise.

4. For each department that has more than two employees, retrieve the department number and the number of its employees who are making more than or equal to $30,000.

5. Write a nested query to retrieve the names of employees who have no dependents.

6. Write a nested query to list the names of managers who have at least one dependent.

7. Write a nested query to retrieve the names of all employees who have two or more dependents.

8. Write a nested query to retrieve the SSNs of all employees who work on project number 1, 2 or 3.

9. Write a nested query to retrieve the names of all employees whose salary is greater than the salary of all the employees in department 5.

Note: while doing nested queries consider the output as inner and apply conditions one by one with each condition satisfying in one condition.

Chapter 10

Lab 10: PL/SQL INTRODUCTION

Objectives

Introduction to PL/SQL

Introduction

PL/SQL stands for Procedural Language extension of SQL. PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL.
Oracle uses a PL/SQL engine to processes the PL/SQL statements. A PL/SQL code can be stored in the client system (client-side) or in the database (server-side).
Advantages of PL/SQL:

- Block Structures: PL SQL consists of blocks of code, which can be nested within each other. Each block forms a unit of a task or a logical module. PL/SQL Blocks can be stored in the database and reused.
- Procedural Language Capability: PL SQL consists of procedural language constructs such as conditional statements (if else statements) and loops like (FOR loops).
- Better Performance: PL SQL engine processes multiple SQL statements simultaneously as a single block, thereby reducing network traffic.
- Error Handling: PL/SQL handles errors or exceptions effectively during the execution of a PL/SQL program. Once an exception is caught, specific actions can be taken depending upon the type of the exception or it can be displayed to the user with a message.

*Syntax of PL/SQL program:
Declare
Variable declaration;
Begin
Executable statements;
end;
*Conditional Statements in PL/SQL
As the name implies, PL/SQL supports programming language features like conditional statements, iterative statements.

The programming constructs are similar to how you use in programming languages like Java and C++. In this section I will provide you syntax of how to use conditional statements in PL/SQL programming.
*IF THEN ELSE STATEMENT

1) IF condition THEN
Statement 1;
ELSE
Statement 2;
END IF;
2) IF condition 1 THEN
Statement 1;
Statement 2;
ELSIF condtion2 THEN
Statement 3;
ELSE
Statement 4;
END IF
*Example

1) Write PL/SQL code for finding specific Employee salary in given table.

```
    DECLARE
VAR_SALARY NUMBER(16);
VAR_EMPNO NUMBER(16):=7839;
BEGIN
SELECT SAL INTO VAR_SALARY FROM EMP WHERE EMPNO=VAR_EMPNO;
DBMS_OUTPUT.PUT_LINE(VAR_SALARY);
DBMS_OUTPUT.PUT_LINE('THE EMPLOYEE OF'|| ' '|| VAR_EMPNO || 'HAS SALARY'||
' ' || VAR_SALARY);
END;
/
```

Output

2) Write PL/SQL code to find Largest of three numbers.

```
DECLARE
A NUMBER;
B NUMBER;
```

```
C NUMBER;
BEGIN
A:=&A;
B:=&B;
C:=&C;
IF A=B AND B=C AND C=A THEN
DBMS_OUTPUT.PUT_LINE('ALL ARE EQUAL');
ELSE IF A>B AND A>C THEN
DBMS_OUTPUT.PUT_LINE('A IS GREATER');
ELSE IF B>C THEN
DBMS_OUTPUT.PUT_LINE('B IS GREATER');
ELSE
DBMS_OUTPUT.PUT_LINE('C IS GREATER');
END IF;
END IF;
END IF;
END;
/
```

Output

Practise Exercise

1) Write PL/SQL code to find smallest of three numbers.

Chapter 11

Lab 11: PL/SQL, Loops

Objectives

Introduction of loops in PL/SQL

Introduction

*Loops in PL/SQL
There are three types of loops in PL/SQL:

- Simple Loop
- While Loop
- For Loop

*Simple Loop A Simple Loop is used when a set of statements is to be executed at least once before the loop terminates. An EXIT condition must be specified in the loop, otherwise the loop will get into an infinite number of iterations. When the EXIT condition is satisfied the process exits from the loop.
Syntax:
LOOP
Statements;
EXIT;
or EXIT WHEN condition ;
END LOOP;

*While Loop A WHILE LOOP is used when a set of statements has to be executed as long as a condition is true. The condition is evaluated at the beginning of each iteration. The iteration continues until the condition becomes false.
Syntax:
WHILE <condition>
LOOP statements;
END LOOP;

*FOR Loop A FOR LOOP is used to execute a set of statements for a predetermined number of times. Iteration occurs between the start and end integer values given. The counter is always incremented by 1. The loop exits when the counter reaches the value of the end integer.

Syntax:
FOR counter IN val1..val2
LOOP statements;
END LOOP;

*Examples
1) Write PL/SQL code for finding Even Numbers.

```
    BEGIN
FOR I IN 1..100 LOOP
IF MOD(I,2)=0 THEN
DBMS_OUTPUT.PUT_LINE(I);
END IF;
END LOOP;
END;
/
```

Output

2) Write PL/SQL code to find given number is Prime or not.

```
DECLARE
N NUMBER;
I NUMBER;
PR NUMBER(2):=1;
BEGIN
N:=&N;
FOR I IN 2..N/2 LOOP
IF MOD(N,I)=0 THEN
PR:=0;
END IF;
END LOOP;
IF PR=1 THEN
DBMS_OUTPUT.PUT_LINE('THE GIVEN NUMBER IS PRIME'||N);
ELSE
DBMS_OUTPUT.PUT_LINE('THE GIVEN NO IS NOT PRIME'||N);
END IF;
END;
/
```

Output

3) Write PL/SQL code to accept the text and reverse the text and test whether the given character is Palandrome or not.

```
DECLARE
G VARCHAR2(20);
R VARCHAR2(20);
BEGIN
G:='&G';
DBMS_OUTPUT.PUT_LINE('THE GIVEN TEXT :'||G);
FOR I IN REVERSE 1..LENGTH(G) LOOP
R:=R||SUBSTR(G,I,1);
END LOOP;
DBMS_OUTPUT.PUT_LINE('THE REVERSED TEXT:'||R);
IF R=G THEN
DBMS_OUTPUT.PUT_LINE('THE GIVEN TEXT IS PALINDROME');
ELSE
DBMS_OUTPUT.PUT_LINE('THE GIVEN TEXT IS NOT PALINDROME');
END IF;
END;
/
```

Output

Practise Exercise

1) Write PL/SQL code to find Reverse of a given number.
2)Write PL/SQL code to generate Fibonacci series for given number.
3) 11) Write PL/SQL code to print the numbers in this form

```
1
1 2
123
```

Chapter 12

Lab 12: Programs using Cursors, Cursor loops and records

Objectives

Introduction and practise of PL/SQL programs usings Cursors, Cursor loops and records

Introduction

Write PL/SQL code to UPDATE values in created tables by using Implicit Cursors.

```
DECLARE
VAR_ROWS NUMBER(5);
BEGIN
UPDATE EMP SET SAL=SAL+100;
IF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('NONE OF THE SALARIES WERE UPDATED');
ELSE IF SQL%FOUND THEN
VAR_ROWS:=SQL%ROWCOUNT;
DBMS_OUTPUT.PUT_LINE('SALARIES FOR'||VAR_ROWS||'EMPLOYEES ARE UPDATED');
END IF ;
END IF;
END;
/
```

Output

Write PL/SQL code to display Employee details using Explicit Cursors.

```
DECLARE
CURSOR EMP_CUR IS SELECT * FROM EMP;
EMP_REC EMP%ROWTYPE;
BEGIN
OPEN EMP_CUR;
LOOP
FETCH EMP_CUR INTO EMP_REC;
EXIT WHEN EMP_CUR%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(EMP_REC.EMPNO || '  ' ||EMP_REC.ENAME || '  '
||EMP_REC.SAL);
END LOOP;
CLOSE EMP_CUR;
END;
/
```

Output

Write PL/SQL code in Cursor to display employee names and salary.

```
DECLARE
CURSOR CL IS SELECT * FROM EMP;
BEGIN
FOR I IN CL
LOOP
DBMS_OUTPUT.PUT_LINE(I.ENAME||'   '||I.SAL);
END LOOP;
END;
/
```

Output

Practise Exercise

1) Write a program to write a Cursor to display the list of employees who are Working as a Managers or Analyst.

Lab 13:Stored procedures in PL/SQL

Objectives

Creating stored procedures and functions

Introduction

Write PL/SQL code in Procedure to find Reverse number

```
CREATE OR REPLACE PROCEDURE REVNUM(NUM NUMBER)
IS
REV INTEGER(6);
NUM1 NUMBER(6);
BEGIN
NUM1:=NUM;
REV:=0;
WHILE NUM1>0
LOOP
REV:=REV*10+MOD(NUM1,10);
NUM1:=TRUNC(NUM1/10);
END LOOP;
DBMS_OUTPUT.PUT_LINE(REV);
END REVNUM;
/
```

Output

Write PL/SQL code in Procedure to find Factorial of a given number by using call procedure.

```
a)    CREATE OR REPLACE PROCEDURE FACT(A IN NUMBER,B OUT NUMBER) IS
F NUMBER(4):=1;
BEGIN
FOR I IN 1..A
LOOP
F:=F*I;
END LOOP;
B:=F;
END;
/
b)      DECLARE
X NUMBER(4):=&X;
Y NUMBER;
BEGIN
FACT(X,Y);
DBMS_OUTPUT.PUT_LINE('FACTORIAL OF' ||'   '||X||'   ' ||'IS'||'   '||Y);
END;
/
```

Output

Write a Procedure to check the given number is prime or not by using call procedure.

```
PROCEDURE DEFINITION:

CREATE or REPLACE PROCEDURE isprimepro(num in number,chec out number) IS
temp NUMBER;
BEGIN
temp:=num;
    FOR itr IN 2..(num-1)
        LOOP
        IF(mod(num,itr)=0) THEN
```

```
              chec:=1;
        END IF;
       END LOOP;
END;
/

PL/SQL BLOCK TO CALL THE PROCEDURE:

DECLARE
    chec NUMBER;
   given_number NUMBER;
BEGIN
   given_number:=&given_number;
     isprimepro(given_number,chec);
    IF chec=1  THEN
    dbms_output.put_line('number is not prime');
    ELSE
    dbms_output.put_line('number is   prime');
  END IF;
  END;
/
```

Output

Chapter 14

Lab 14:Functions in PL/SQL

Objection

Introduction of functions in PL/SQL

Introduction

Write a Function to check the given number is prime or not.

```
FUNCTION DEFINITION :

CREATE or  REPLACE FUNCTION isprime(num in number)  RETURN
number IS
temp NUMBER;
BEGIN
temp:=num;
FOR itr  IN 2..(num-1)
    LOOP
        IF (mod(temp,itr)=0)  THEN
             return 1;
        END IF;
    END LOOP;
return 0;
END;
/

PL/SQL BLOCK TO CALL THE FUNCTION:

DECLARE
given_number  NUMBER;
prime_check  NUMBER;
BEGIN
given_number:=&given_number;
prime_check:=isprime(given_number);
IF prime_check=0 THEN
dbms_output.put_line('NUMBER IS PRIME');
```

```
 ELSE
dbms_output.put_line('NUMBER IS  NOT PRIME');
  END IF;
END;
/
```

Output

Write pl./sql code in Function for Factorialnumber.

```
(a) set serveroutput on;

create or replace Function FunFact(a number) return number IS
f number(4):=1;
begin
for i in 1..a
loop
f:=f*i;
end loop;
return f;
end;
/
(b)  set serveroutput on;
Declare
n number(2):=&n;
r number(4);
Begin
r:=FunFact(n);
dbms_output.put_line('factorial of'||n||'is:'||r);
end;
/
```

Output

Chapter 15

Lab 15: Error handling and Triggers in PL/SQL

Objectives

Error handling and Trigers in PL/SQL

Introduction

Error Handling

Write a PL/SQL block to handle the following BUILT-IN EXCEPTIONS.

```
DECLARE
M NUMBER(4);
MYERROR EXCEPTION;
BEGIN
SELECT COMM INTO M FROM EMP WHERE EMPNO=7839;
IF M IS NULL THEN
RAISE MYERROR;
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('ERROR OF DATA');
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('ERROR OF TOO MANY ROWS');
WHEN MYERROR THEN
DBMS_OUTPUT.PUT_LINE('ERROR FOUND NULL');
END;
/
```

Output

Triggers

Write pl/sql code for before insert Trigger program
Steps for doing Trigger Program

- First create a table called Person(Don't insert any values into created table)
- Write code for Trigger and run the trigger program
- Insert values into the created table after sucessfully completion of trigger program and see the trigger output.

```
(a) CREATE TABLE PERSON1(ID INTEGER,NAME VARCHAR2(30),DOB DATE,PRIMARY
       KEY(ID));
(b) CREATE OR REPLACE TRIGGER PERSON_INSERT_BEFORE
      BEFORE INSERT ON PERSON1 FOR EACH ROW
      BEGIN
     DBMS_OUTPUT.PUT_LINE('BEFORE INSERT OF'||:NEW.NAME);
    END;
     /
(c) INSERT INTO PERSON1 VALUES(1,'JOHN DOE',SYSDATE);
```

Output

Write pl/sql code for After inser Trigger

```
(a)  CREATE OR REPLACE TRIGGER PERSON_INSERT_AFTER
      AFTER INSERT ON PERSON1 FOR EACH ROW
     BEGIN
     DBMS_OUTPUT.PUT_LINE('AFTER INSERT OF'||:NEW.NAME);
    END;
      /
(b) INSERT INTO PERSON1 VALUES(2,'JAME DOE',SYSDATE);
```

Output

Write pl/sql code for If Statement Trigger.

```
(a) CREATE OR REPLACE TRIGGER PERSON_BIUD
     BEFORE INSERT OR UPDATE OR DELETE ON PERSON1 FOR EACH ROW
    BEGIN
IF INSERTING THEN
DBMS_OUTPUT.PUT_LINE('INSERTING PERSON:'||:NEW.NAME);
ELSE IF UPDATING THEN
DBMS_OUTPUT.PUT_LINE('UPDATING PERSON:'||:OLD.NAME||'TO'||:NEW.NAME);
ELSE IF DELETING THEN
DBMS_OUTPUT.PUT_LINE('DELETING PERSON:'||:OLD.NAME);
END IF;
END IF;
END IF;
END;
/
(b) INSERT INTO PERSON1 VALUES(4,'CSE','2-SEP-2009');
(C) UPDATE PERSON1 SET NAME='COMPUTER'WHERE NAME='CSE';
(D) DELETE PERSON1 WHERE NAME='COMPUTER';
(E) SELECT * FROM PERSON1;
```

Output

Chapter 16

Lab 16: User Defined Types

Objectives

Creating Objects in pl/sql

Introduction

Write PL/SQL Code for creating Objects with an example.

```
(a)  create or replace   type person as object ( last_name varchar2(100),
     phone varchar2(100),   member function get_last_name return varchar2,
     member function get_phone_number return varchar2 )  not final
    /
Type created.
(b) create or replace
  type body person as
  member function get_last_name return varchar2 is
   begin
  return self.last_name;
  end;
  member function get_phone_number return varchar2 is
   begin
  return self.phone;
 end;
 end;
 /
Type body created.
(c)       declare
                  l_person person;
                  begin
                  l_person := person( 'C', 'B' );
                dbms_output.put_line( l_person.last_name );
              end;
              /
             C
         PL/SQL procedure successfully completed.
```

Output