UNIT-IV (Relational Database Language , PL/SQL)

Section-A (2 Marks)

Important questions

1. Define (i) Primary Key (ii) Foreign Key (iii) unique key.

(i)primary key:A primary key can consist of one or more columns on a table. Primary key constraints define a column or series of columns that uniquely identify a given row in a table. Any column that is defined as a primary key column is automatically set with a NOT NULL status.

(ii) foreignkey:A foreign key constraint is used to enforce a relationship between two tables. A foreign key is a column (or a group of columns) whose values are derived from the Primary key or unique key of some other table.

(iii)unique key: Unique key will not allow duplicate values. A table can have more than one Unique key. A unique constraint defines a column, or series of columns, that must be unique in value. Unique key may contain null values.

2.What is SQL?

SQL stands for Structured Query Language ,SQL is a declarative (non-procedural) language. SQL is (usually) not case-sensitive, but we'll write SQL keywords in upper case for emphasis. Some database systems require a semicolon at the end of each SQL statement.

3.What is procedure in PL/SQL?

PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms −

- **Functions** − These subprograms return a single value; mainly used to compute and return a value.
- **Procedures** − These subprograms do not return a value directly; mainly used to perform an action.

4.What is an Exception? Mention major types of Exceptions.

An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using **EXCEPTION** block in the program and an appropriate action is taken against the error condition. There are two types of exceptions −

- System-defined exceptions
- User-defined exceptions

5.Mention different categories of SQL statements.

Data Definition Language (DDL)
Data control language(DCL)
Data manipulation language(DML)
Transaction control language(TCL)

6.Define (i) Tuple (ii) Query (iii) Domain.

(i)Tuple-a single row of a table,which contains a single record for the relation.

(ii)Query: Query is a request for data or information from a database table or combination of tables .eg:sql.

(iii) Domain : A **domain** is essentially a data type with optional constraints (restrictions on the allowed set of values). The user who defines a **domain** becomes its owner.

7.What is Cursor?

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

8.What is Trigger?

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events −

- A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
- A **database definition (DDL)** statement (CREATE, ALTER, or DROP).

- A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

9.List out different types of Join Operations.

There are different types of joins available in SQL:

INNER join
OUTER join (LEFT,RIGHT,FULL)
CROSS join

10. What is group by Clause? Explain the difference between Group by and Order by Clause.

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) togroup the result-set by one or more columns.

The **ORDER BY clause** is used to **sort** the data in **ascending** or descending**order**, based on one or more columns. Some databases **sort** the query results in an**ascending order** by default.

11.What is PL/SQL? Mention any two advantages of PL/SQL.

The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database. PL/SQL is a completely portable, high-performance transaction-processing language.

Advantages of PL/SQL

- PL/SQL supports both static and dynamic SQL. Static SQL supports DML operations and transaction control from PL/SQL block. In Dynamic SQL, SQL allows embedding DDL statements in PL/SQL blocks.
- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.
- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.
- Applications written in PL/SQL are fully portable.
- PL/SQL provides high security level.

12.Write the structure of PL/SQL.

The basic structure of a PL/SQL block −

```
DECLARE
<declarations section>
BEGIN
<executable command(s)>
EXCEPTION
<exception handling>
END;
```

Every PL/SQL statement ends with a semicolon (;).

Declarations

This section starts with the keyword **DECLARE**. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.

Executable Commands

This section is enclosed between the keywords **BEGIN** and **END** and it is a mandatory section. It consists of the executable PL/SQL statements of the program.

Exception Handling

This section starts with the keyword **EXCEPTION**. This optional section contains **exception(s)** that handle errors in the program.

13.Explain referential integrity.

Referential integrity (RI) is a relational **database** concept, which states that table relationships must always be consistent. In other words, any foreign key field must agree with the primary key that is referenced by the foreign key.

Section-B (5 Marks and 10 Marks)

1. Explain numeric and aggregate functions in SQL.

Numerical functions:

Oracle allows arithmetic operators to be used while viewing records from a table orwhile performing data manipulation operations such as insert, updated and delete.

These are:
+ Addition
- Subtraction
/ Division
* Multiplication

() Enclosed Operations

Consider the below employee table(gkemp)

```
SQL> select * from gkemp;

     EMPID ENAME                          ESAL
---------- -------------------- ----------
       101 Nisarga                       8500
       102 Varsha G Kalyan              15000
       103 Eenchara                      5000
       104 Mohith G Kalyan              12500
       105 Kavitha                      18000


SQL> select esal,esal+2500 from gkemp;

      ESAL  ESAL+2500
---------- ----------
      8500      11000
     15000      17500
      5000       7500
     12500      15000
     18000      20500

SQL> select esal,esal-500 from gkemp;

      ESAL   ESAL-500
---------- ----------
      8500       8000
     15000      14500
      5000       4500
     12500      12000
     18000      17500

SQL> select esal, esal/100 from gkemp;

      ESAL   ESAL/100
---------- ----------
      8500         85
     15000        150
      5000         50
     12500        125
     18000        180

SQL> select esal, esal*10 from gkemp;

      ESAL    ESAL*10
---------- ----------
      8500      85000
     15000     150000
      5000      50000
     12500     125000
     18000     180000
```

```
SQL>  select esal,(10+esal)/100 from gkemp;

      ESAL (10+ESAL)/100
---------- -------------
      8500          85.1
     15000         150.1
      5000          50.1
     12500         125.1
     18000         180.1
```

SQL Aggregate / Group Functions

Group functions return results based on groups of rows, rather than on singlerows. returns the number of rows in the query.SQL aggregate functions return asingle value, calculated from values in a column.

Useful aggregate functions:
a) COUNT() - Returns the number of rows
b) AVG() - Returns the average value
c) MAX() - Returns the largest value
d) MIN() - Returns the smallest value
e) SUM() - Returns the total sum
Consider the below employee table(gkemp)

```
SQL> select * from gkemp;

     EMPID ENAME                          ESAL
---------- -------------------- ----------
       101 Nisarga                       8500
       102 Varsha G Kalyan              15000
       103 Eenchara                      5000
       104 Mohith G Kalyan              12500
       105 Kavitha                      18000
```

a) COUNT()
The COUNT() function counts number of values present in the column excluding Null values.

```
SQL> select count(empid) as totalemployee from gkemp;

TOTALEMPLOYEE
-------------
            5
```

b) AVG()
The AVG() function returns the average value of a column specified.

```
SQL> select avg(esal) as averagesalary from gkemp;

AVERAGESALARY
-------------
        11800
```

c) MAX()
The MAX() function returns the highest value of a particular column.

```
SQL> select max(esal) as maximumsalary from gkemp;

MAXIMUMSALARY
-------------
        18000
```

d) MIN()
The MIN() function returns the smallest value of a particular column.

```
SQL> select min(esal) as minimumsalary from gkemp;

MINIMUMSALARY
-------------
         5000
```

e) SUM()
The SUM() function returns the sum of column values.

```
SQL> select sum(esal) as totalsalary from gkemp;

TOTALSALARY
-----------
      59000
```

2.Explain the following:

(i) IF statement.
(ii) While loop
(iii) FOR Loop

(i) IF statement

The **IF statement** associates a condition with a sequence of statements enclosed by the keywords **THEN** and **END IF**. If the condition is **TRUE**, the statements get executed, and if the condition is **FALSE** or **NULL**, then the **IF** statement does nothing.

Syntax

Syntax for **IF-THEN** statement is −

```
IF condition THEN
  S;
```

```
END IF;
```

Where *condition* is a Boolean or relational condition and S is a simple or compound statement. Following is an example of the IF-THEN statement −

```
IF (a <=20) THEN

c:= c+1;

END IF;
```

WHILE Loop

A **WHILE LOOP** statement in PL/SQL programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

```
WHILE condition LOOP
sequence_of_statements
END LOOP;
```

Example

```
DECLARE

a number(2):=10;

BEGIN

  WHILE a <20 LOOP

dbms_output.put_line('value of a: '|| a);

a:= a +1;

END LOOP;

END;

/
```

FOR LOOP

A **FOR LOOP** is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax

```
FOR counter IN initial_value ..final_value LOOP
sequence_of_statements;
END LOOP;
```

Following is the flow of control in a **For Loop** −

- The initial step is executed first, and only once. This step allows you to declare and initialize any loop control variables.
- Next, the condition, i.e., *initial_value* ..*final_value* is evaluated. If it is TRUE, the body of the loop is executed. If it is FALSE, the body of the loop does not execute and the flow of control jumps to the next statement just after the for loop.
- After the body of the for loop executes, the value of the counter variable is increased or decreased.
- The condition is now evaluated again. If it is TRUE, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes FALSE, the FOR-LOOP terminates.

Example

```
DECLARE
a number(2);
BEGIN
  FOR a in10..20 LOOP
dbms_output.put_line('value of a: '|| a);
END LOOP;
END;
/
```

3.Explain all categories of Database Languages.

Data Definition Language (DDL)

It is a set of SQL commands used to create, modify and delete database structure but not data. It also define indexes (keys), specify links between tables, and impose constraints between tables. DDL commands are auto COMMIT.

The most important DDL statements in SQL are:

- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- TRUNCATE TABLE- deletes all records from a table
- DROP TABLE - deletes a table

Data Manipulation Language (DML)

It is the area of SQL that allows changing data within the database. The query and update commands form the DML part of SQL:

- INSERT - inserts new data into a database
- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database

Data Control Language (DCL)
It is the component of SQL statement that control access to data and to the database. Occasionally DCL statements are grouped with DML Statements.

- COMMIT –Save work done.
- SAVEPOINT – Identify a point in a transaction to which you can later rollback. ROLLBACK – Restore database to original since the last COMMIT.
- GRANT – gives user's access privileges to database.
- REVOKE – withdraw access privileges given with GRANT command.

4.Explain types of Database Triggers in detail.

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events −

- A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
- A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
- A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Creating Triggers

The syntax for creating a trigger is −

```
CREATE [OR REPLACE ] TRIGGER trigger_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR]| UPDATE [OR]| DELETE}

[OF col_name]

ON table_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

Declaration-statements
```

```
BEGIN
Executable-statements
EXCEPTION
Exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name − Creates or replaces an existing trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
- [OF col_name] − This specifies the column name that will be updated.
- [ON table_name] − This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

5.Explain types of Database Cursors in detail.

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

There are two types of cursors −

- Implicit cursors
- Explicit cursors

Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

The following table provides the description of the most used attributes in cursors −

S.No	Attribute & Description
1	**%FOUND** Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
2	**%NOTFOUND** The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
3	**%ISOPEN** Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	**%ROWCOUNT** Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is −

```
CURSOR cursor_name IS select_statement;
```

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example −

```
CURSOR c_customers IS
   SELECT id, name, address FROM customers;
```

6.Explain various types of Join Operations in detail with Examples.

The SQL **Joins** clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each. Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal symbol.

SQL Join Types:

There are different types of joins available in SQL:

INNER
OUTER(LEFT,RIGHT,FULL)
CROSS

Consider the below tables for Join Operations examples

```
SQL> create table gkproduct(product_id number(3) primary key,
  2  product_name varchar(15),
  3  supplier_name varchar(15),
  4  price number(5));

Table created.

SQL> create table gkorder(order_id number(4) primary key,
  2  product_id number(3) references gkproduct(product_id),
  3  total_units number(3),
  4  customer_name varchar(15));

Table created.
```

```
SQL> insert into gkproduct values(100,'Camera','Nikon',30000);

1 row created.

SQL> insert into gkproduct values(101,'Television','Onida',15000);

1 row created.

SQL> insert into gkproduct values(102,'Refrigerator','videocon',18000);

1 row created.

SQL> insert into gkproduct values(103,'Ipod','Apple',16000);

1 row created.

SQL> insert into gkproduct values(104,'Mobile','Samsung',8000);

1 row created.

SQL> insert into gkorder values(5100,104,30,'Infosys');

1 row created.

SQL> insert into gkorder values(5101,102,15,'GKMV');

1 row created.

SQL> insert into gkorder values(5102,103,25,'Wipro');

1 row created.

SQL>  insert into gkorder values(5103,101,10,'TCS');

1 row created.

SQL> select * from gkproduct;

PRODUCT_ID PRODUCT_NAME    SUPPLIER_NAME        PRICE
---------- --------------- --------------- ----------
       100 Camera          Nikon                30000
       101 Television      Onida                15000
       102 Refrigerator    videocon             18000
       103 Ipod            Apple                16000
       104 Mobile          Samsung               8000

SQL> select * from gkorder;

  ORDER_ID PRODUCT_ID TOTAL_UNITS CUSTOMER_NAME
---------- ---------- ----------- ---------------
      5100        104          30 Infosys
      5101        102          15 GKMV
      5102        103          25 Wipro
      5103        101          10 TCS
```

INNER Join

Inner join are also known as Equi Joins. They are the most common joins used in SQL. They are known as equi joins because it uses the equal sign as the comparison operator (=). The INNER join returns all rows from both tables where there is a match.

Consider the above tables (gkproduct and gkorder),

For example: If you want to display the product information for each order the query will be as given below

```
SQL> select gko.order_id,gkp.product_name,gkp.price, gkp.supplier_name,gko.total_units
  2  from gkproduct gkp, gkorder gko
  3  where gko.product_id=gkp.product_id;

  ORDER_ID PRODUCT_NAME         PRICE SUPPLIER_NAME   TOTAL_UNITS
---------- --------------- ---------- --------------- -----------
      5103 Television           15000 Onida                    10
      5101 Refrigerator         18000 videocon                 15
      5102 Ipod                 16000 Apple                    25
      5100 Mobile                8000 Samsung                  30
```

OUTER Join

OUTER join condition returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables. The sql outer join operator in Oracle is (+) and is used on one side of the join condition only.

For example: If you want to display all the product data along with order items data, with null values displayed for order items if a product has no order item, the sql query for outer join would be as shown below(ie First Query).

```
SQL> select gkp.product_id, gkp.product_name,gko.order_id,gko.total_units
  2  from gkproduct gkp, gkorder gko
  3  where gko.product_id(+)=gkp.product_id;

PRODUCT_ID PRODUCT_NAME      ORDER_ID TOTAL_UNITS
---------- --------------- ---------- -----------
       104 Mobile                5100          30
       102 Refrigerator          5101          15
       103 Ipod                  5102          25
       101 Television            5103          10
       100 Camera

SQL> select gkp.product_id, gkp.product_name,gko.order_id,gko.total_units
  2  from gkproduct gkp, gkorder gko
  3  where gkp.product_id(+)=gko.product_id;

PRODUCT_ID PRODUCT_NAME      ORDER_ID TOTAL_UNITS
---------- --------------- ---------- -----------
       101 Television            5103          10
       102 Refrigerator          5101          15
       103 Ipod                  5102          25
       104 Mobile                5100          30
```

NOTE: If the (+) operator is used in the left side of the join condition it is equivalent to left outer join. If used on the right side of the join condition it is equivalent to right outer join.

OUTER JOIN :

Outer Join retrieves Either, the matched rows from one table and all rows in the other table Or, all rows in all tables (it doesn't matter whether or not there is a match).

There are three kinds of Outer Join :

LEFT OUTER JOIN or LEFT JOIN

This join returns all the rows from the left table in conjunction with the matching rows from the right table. If there are no columns matching in the right table, it returns NULL values.

RIGHT OUTER JOIN or RIGHT JOIN

This join returns all the rows from the right table in conjunction with the matching rows from the left table. If there are no columns matching in the left table, it returns NULL values.

FULL OUTER JOIN or FULL JOIN

This join combines left outer join and right outer join. It returns row from either table when the conditions are met and returns null value when there is no match.

In other words, OUTER JOIN is based on the fact that : ONLY the matching entries in ONE OF the tables of the tables(FULL) SHOULD be listed.

CROSS Join

It is the Cartesian product of the two tables involved. It will return a table with consists of records which combines each row from the first table with each row of the second table.
The result of a CROSS JOIN will not make sense in most of the situations.
Moreover, we won't need this at all (or needs the least, to be precise).

```
SQL> select * from gkproduct
  2  CROSS JOIN
  3  gkorder;

PRODUCT_ID PRODUCT_NAME    SUPPLIER_NAME        PRICE   ORDER_ID PRODUCT_ID TOTAL_UNITS CUSTOMER_NAME
---------- --------------- --------------- ---------- ---------- ---------- ----------- ------------
       100 Camera          Nikon                30000       5100        104          30 Infosys
       101 Television      Onida                15000       5100        104          30 Infosys
       102 Refrigerator    videocon             18000       5100        104          30 Infosys
       103 Ipod            Apple                16000       5100        104          30 Infosys
       104 Mobile          Samsung               8000       5100        104          30 Infosys
       100 Camera          Nikon                30000       5101        102          15 GKMV
       101 Television      Onida                15000       5101        102          15 GKMV
       102 Refrigerator    videocon             18000       5101        102          15 GKMV
       103 Ipod            Apple                16000       5101        102          15 GKMV
       104 Mobile          Samsung               8000       5101        102          15 GKMV
       100 Camera          Nikon                30000       5102        103          25 Wipro

PRODUCT_ID PRODUCT_NAME    SUPPLIER_NAME        PRICE   ORDER_ID PRODUCT_ID TOTAL_UNITS CUSTOMER_NAME
---------- --------------- --------------- ---------- ---------- ---------- ----------- ------------
       101 Television      Onida                15000       5102        103          25 Wipro
       102 Refrigerator    videocon             18000       5102        103          25 Wipro
       103 Ipod            Apple                16000       5102        103          25 Wipro
       104 Mobile          Samsung               8000       5102        103          25 Wipro
       100 Camera          Nikon                30000       5103        101          10 TCS
       101 Television      Onida                15000       5103        101          10 TCS
       102 Refrigerator    videocon             18000       5103        101          10 TCS
       103 Ipod            Apple                16000       5103        101          10 TCS
       104 Mobile          Samsung               8000       5103        101          10 TCS

20 rows selected.
```

INNER JOIN: returns rows when there is a match in both tables.
LEFT JOIN: returns all rows from the left table, even if there are no matches in the right table.
RIGHT JOIN: returns all rows from the right table, even if there are no matches in the left table.
FULL JOIN: returns rows when there is a match in one of the tables.
SELF JOIN: is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
CARTESIAN JOIN: returns the Cartesian product of the sets of records from the two or more joined tables.