_____

**WINTER– 18 EXAMINATION**
**Subject Name: Database Management System**  **Model Answer**  Subject Code: 22319

**Important Instructions to examiners:**

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

| Q. No. | Sub Q. N. | Answer | Marking Scheme |
|---|---|---|---|
| 1 | | **Attempt any FIVE of the following :** | **10 M** |
| | **a** | **List disadvantages of typical file processing system.** | 2 M |
| | **Ans** | Disadvantages of file processing system<br><br>1. **Data redundancy and inconsistency**<br><br>2. **Difficulty in accessing data**<br><br>3. **Data isolation**<br><br>4. **Integrity problems**<br><br>5. **Atomicity problems** | any 4 disadvantages – 1/2 Mark each disadvantage |
| | **b** | **Define i)Data Abstraction  ii)Data Redundancy** | 2 M |
| | **Ans** | **1. Data Abstraction :**<br><br>Many end users are not computer trained so it is needed to hide complex data structures from them.<br><br>**Hiding complexity of data structures from end user** through different levels is known as data abstraction.<br><br>It has 3 levels : | 1 mark Data Abstraction description  1 mark Data Redundancy description |

_____

| | | | |
|---|---|---|---|
| | | a. Physical level<br><br>b. logical level<br><br>c. view level<br><br>   **2. Data redundancy :**<br><br>   The **repetition of information** is known as redundancy .This redundancy leads to higher storage and access cost.<br><br>   It may lead to data inconsistency, that is different copies of the same data may have different values. | |
| | **c** | **Define the term:**<br><br>i)     **Candidate key**<br><br>ii)    **ii) Primary key** | 2 M |
| | **Ans** | **Candidate key:** In a relation, there may be a primary key or may not, but there may be a key **or combination of keys which uniquely identify the record**. Such a key is called as Candidate key.<br><br>OR<br><br>A candidate key is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data.<br><br>The candidate key can be simple (having only one attribute) or composite as well.<br><br>For Example, {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.<br><br>**Primary key:** A key which is selected by the designer to uniquely identify the entity is called as Primary key. A primary key cannot contain duplicate values and it can never contain null values inside it.<br><br>Example, RollNo attribute is a primary key for Relation Student. | 1 mark Candidate key<br>1 mark Primary Key |
| | **d** | **List Four DDL commands with syntax.** | 2 M |
| | **Ans** | **DDL commands**<br><br>   **1. 1.Create**<br><br>**Syntax :** create table <table_name>(Column_name1 datatype1, column_name2 Datatype2,…Column_nameN DatatypeN);<br><br>   **2. Drop** | 1/2 mark for each command and ½ mark for syntax |

| | | | |
|---|---|---|---|
| | | **Syntax**: drop table <table_name>; | |
| | | **3. Desc**<br><br>**Syntax**: describe <table_name>;<br><br>**OR**<br><br>Desc <table_name><br><br>**4. Truncate**<br><br>**Syntax**: truncate table <table_name>;<br><br>**5. Alter**<br><br>Syntax: Alter table <table_name> add Column_name Datatype (size); | |
| | e | **Define Normalization, list its types.** | 2 M |
| | Ans | **Normalization:**<br><br>Normalization can be defined as process of decomposition/division of database tables to avoid the data redundancy.<br><br>**Types of Normalization:**<br><br>1. 1NF<br>2. 2NF<br>3. 3NF<br>4. BCNF | 1 mark for Normalization definition and 1 mark for types |
| | f | **Enlist four aggregate functions.** | 2 M |
| | Ans | SUM()<br><br>AVG()<br><br>MAX()<br><br>MIN()<br><br>COUNT() | any 4 functions |
| | g | **Define Cursor. List the two types of cursor.** | 2 M |

| | | | |
|---|---|---|---|
| | **Ans** | **Cursor:** The Oracle Engine uses a work area for its internal processing in order to execute an SQL statement. This work area is private to SQL''s operations and is called a Cursor.<br><br>**OR**<br><br>A **cursor** is a temporary work area created in the system memory when a SQL statement is executed.<br><br>**Types of cursor are:**<br><br> 1) Implicit cursor<br><br> 2) Explicit cursor | 1 mark Cursor definition and 1 mark for types of cursor |
| **2** | | **Attempt any THREE of the following :** | **12 M** |
| | **a** | **Distinguish between network model and hierarchical model.** | 4 M |
| | **Ans** | (see table below) | any 4 points |

| Sr. No. | Hierarchical model | Network model |
|---|---|---|
| 1. | Hierarchical model is not more popular than network model | Network model is more popular than the hierarchical and relational model. |
| 2. | It does not uses client server architecture | It uses client –server architecture |
| 3. | One to many relationship is maintained. | One to many and many to many relationship is maintained. |
| 4. | Hierarchical model is based on tree like structure with one root. | Network model is based on tree like structure with many roots. |
| 5. | One child or many children have only one parent | Many children have many parent |
| 6. | Main application of hierarchical model is in the mainframe database system. | It is upgraded version of hierarchical model so used in network |

| | | | |
|---|---|---|---|
| | **b** | **Explain set Operators with example.** | 4 M |
| | **Ans** | Set operators combine the results of two component queries into a single result. Queries containing set operators are called as compound queries. Set operators in SQL are represented with following special keywords as: Union, Union all, intersection & minus. Consider data from two tables emp and employee as | 1 mark for explanation and 1 mark for example each |

**Emp**                                              **Employee**

| Ename |
|-------|
| a |
| b |
| c |
| d |

| Ename |
|-------|
| c |
| e |

**1) Union:** The Union of two or more sets contains all elements, which are present in either or both. Union works as or.

E.g. select ename from emp union select ename from employee;

The output considering above data is :

**Output**

| Ename |
|-------|
| a |
| b |
| c |
| d |
| e |

**2) Union all:** The Union of 2 or more sets contains all elements, which are present in both, including duplicates.

E.g. select ename from emp union all select ename from employee;

The output considering above data is:
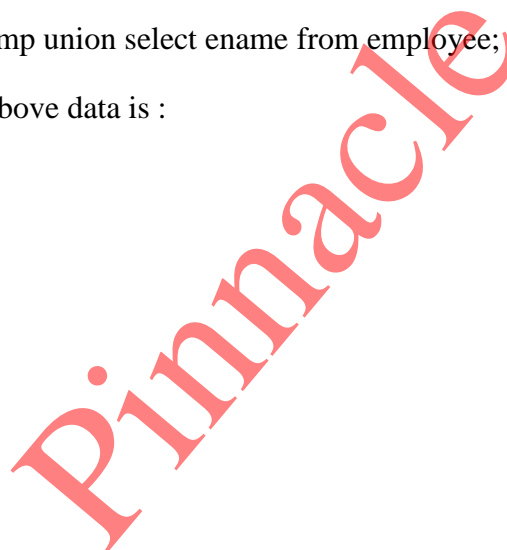
**Output**

| Ename |
|-------|
| a |
| b |
| c |
| c |
| d |
| e |

**3) Intersection:** The intersection of two sets includes elements which are present in both.  E.g. select ename from emp intersect select ename from employee;

The output considering above data is:

Output

| Ename |
|-------|
| c |

**4) Minus:**  The minus of two sets includes elements from set1 minus elements of set2.

E.g. select ename from emp minus select ename from employee;

The output considering above data is:

| Ename |
|-------|
| a |
| b |
| d |

| | c | **Explain any four String functions with example.** | 4 M |
|---|---|---|---|
| | **Ans** | **i) Lower(char)-**  Returns the input string with all letters in lower case.  Example: SQL>Select lower ('RAJESH') from dual; | 1 mark for explanation and 1 mark for example each |

Output: rajesh

**ii) Upper(char)-**

Returns the input string with all letters in upper case.

Example: SQL>Select upper ('rajesh') from dual;

Output: RAJESH

**iii) Ltrim(char,set)-**

It removes or trims from left of character string

. Example: SQL>Select Ltrim('university','univ') from dual;

Output: ersity

**iv) Rtrim(char,set)-**

It removes or trims from right of character string.

Example: SQL>Select Rtrim('university','sity') from dual;

Output: univer

**v) Length(char)-**

It returns length of character string.

Example: SQL> Select length('University') from dual; Output:10

**vi)  Concat(str1,str2,...)-**

Returns the string that result from concatenating the arguments.

Example: Select Concat('employee', 'name') from dual;

Output: employeename

**vii)  Lpad(str, len, padstr)-**

Returns the string str, left-padded with the string padstr to a length of len characters.
Example: Select lpad(ename,10.'*') from emp where empno=7782;

**viii)   Rpad(str,len,padstr)-**

Returns the string str, right-padded with the string    padstr to a length of  len characters.

Example: Select rpad(ename,10.'*') from emp where empno=7782;

| | | | |
|---|---|---|---|
| | | **viii) Substr(Char,m,n)-**<br><br>It returns a portion of char, beginning at a character m, n character long.<br><br>Example: Select substr('College',3,4) from dual;<br><br>Output: lleg | |
| | d | **Describe exception handling in brief.** | 4 M |
| | Ans | Exception Handling: Exception is nothing but an error. Exception can be raise when DBMS encounters errors or it can be raised explicitly.<br><br>When the system throws a warning or has an error it can lead to an exception. Such exception needs to be handled and can be defined internally or user defined.<br><br>Exception handling is nothing but a code block in memory that will attempt to resolve current error condition.<br><br>**Syntax:**<br><br>**DECLARE ;**<br><br>**Declaration section**<br><br>**…executable statement;**<br><br>**EXCEPTION**<br><br>**WHEN ex_name1 THEN ;**<br><br>**Error handling statements/user defined action to be carried out;**<br><br>**END;**<br><br>**Types of Exception:**<br><br>**1) Predefined Exception/system defined exception/named exception:** Are always automatically raised whenever related error occurs. The most common errors that can occur during the execution of PL/SQL. Not declared explicitly i.e. cursor already open, invalid cursor, no data found, zero divide and too many rows etc. Programs are handled by system defined Exceptions.<br><br>**2) User defined exception:** It must be declare by the user in the declaration part of the block where the exception is used. It is raised explicitly in sequence of statements using:<br><br>Raise_application_error(Exception_Number, Error_Message); | any relevant 4 points 1 mark each |
| **3** | | **Attempt any THREE of the following :** | **12 M** |
| | a | **Describe commit and rollback with syntax and example.** | 4 M |

| | Ans | **Commit:** | Description and syntax – 1 Mark example 1 Mark for each |
|---|---|---|---|
| | | The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command | |
| | | The syntax:  SQL> COMMIT; | |
| | |  Or | |
| | |    COMMIT WORK; | |
| | | Example : | |
| | | SQL>Commit; | |
| | | **Rollback:** | |
| | | The ROLLBACK command is used to undo transactions that have not already been saved to the database. | |
| | | The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued. | |
| | | The syntax for ROLLBACK is: | |
| | |  ROLLBACK TO SAVEPOINT_NAME; | |
| | | OR | |
| | |  ROLLBACK; | |
| | | OR | |
| | | ROLLBACK WORK; | |
| | | Example: | |
| | | SQL>ROLLBACK; | |
| | **b** | **Explain joins in SQL with examples.** | 4 M |
| | Ans | JOIN: | Definition: 2 marks, Any 2 types with description: 1 mark each |
| | | A SQL join is an instruction to combine data from two sets of data (i.e. two tables).  A JOIN clause is used to combine rows from two or more tables, based on a related column between them. SQL Join types are as follows: | |
| | | 1) **INNER JOIN or EQUI JOIN:** | |
| | | A join which is based on equalities is called equi join. In equi join comparison operator "=" is used to perform a Join. | |

Syntax:

SELECT tablename.column1_name,tablename.column1_name

FROM table_name1,table_name2

where table_name1.column_name=table_name2.column_name;

Example:

Select stud_info.stud_name, stud_info.branch_code, branch_details.location

From stud_info, branch_details

Where Stud_info.branch_code=branch_details.branch_code;

**2) SELF JOIN:**

The SQL 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.

Syntax:

SELECT a.column_name, b.column_name

FROM table1 a, table1 b

WHERE a.common_filed = b.common_field;

Example:

Select x.stud_name, y.stud_name

from stud_info x, stud_info y

Where x.leader= y.stud_id;

**3) LEFT OUTER JOIN:**

A left outer join retains all of the rows of the "left" table, regardless of whether there is a row that matches on the "right" table.

Syntax:

Select column1name,column2name

from table1name any_alias1 ,table2name any_alias2

on any_alias1.columnname(+) = any_alias2.columnname;

OR

Select column1name,column2name

from table1name left outer join table2name

on table1name.columnname= table2name.columnname;

Example:

select last_name, department_name

from employees e, departments d

on e.department_id(+) = d.department_id;

OR

select last_name, department_name

from employees left outer join departments

on employees.department_id = departments.department_id;


4) **RIGHT OUTER JOIN:**

A right outer join retains all of the rows of the "right" table, regardless of whether there is a row that matches on the "left" table.

Syntax:

Select column1name, column2name

from table1name any_alias1, table2name any_alias2

on any_alias1.columnname =any_alias2.columnname (+);

OR

Select column1name, column2name

from table1name  any_alias1 right outer join table2 name any_alias2

on any_alias1.columnname =any_alias2.columnname;

Example:

Select last_name,department_name from employees e, departments d on e.department_id = d.department_id(+);

OR

Select last_name, department_name

from employees e right outer join departments d

on e.department_id = d.department_id;

5) **NON EQUI JOIN:**

Non equi joins is used to return result from two or more tables where exact join is not possible.

Syntax:

Select aliasname.column1name, aliasname.column2name from tablename alias where <condition using range>;

For example:

In emp table and salgrade table. The salgrade table contains grade and their low salary and high salary. Suppose you want to find the grade of employees based on their salaries then you can use NON EQUI join.

Select e.empno, e.ename, e.sal, s.grade

from emp e, salgrade s

where e.sal between s.lowsal and s.hisal;

| | c | **Explain function in PL/SQL with example.** | 4 M |
|---|---|---|---|
| | **Ans** | Function: Function is a logically grouped set of SQL and Pl/SQL statements that perform a specific task. A function is same as a procedure except that it returns a value. A function is created using the CREATE FUNCTION statement. | Description: 2 marks, Example : 2 marks |

Syntax:

```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
< function_body >
END [function_name];
```

Where,

- *function-name* specifies the name of the function.
- [OR REPLACE] option allows the modification of an existing function.
- The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
- The function must contain a **return** statement.
- The *RETURN* clause specifies the data type you are going to return from the function.
- *function-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

Example:

CREATE OR REPLACE FUNCTION Success_cnt

RETURN number

IS    cnt number(7) := 0;

BEGIN

SELECT count(*) into cnt

FROM candidate where result='Pass';

   RETURN cnt;

END;

/

| | d | **Explain database security with its requirements in detail.** | 4 M |
|---|---|---|---|
| | **Ans** | Database security<br><br>Database security refers to the collective measures used to protect and secure a database or database management software from illegal use and malicious threats and attacks.<br><br>Requirements of Database Security:<br><br>1. For prevention of data theft such as bank account numbers, credit card information, passwords, work related documents or sheets, etc.<br><br>2.  2. To make data remain safe and confidential.<br><br>3. To provide confidentiality which ensures that only those individuals should ever | Definition 1 Mark, 3 mark for detail description |

be able to view data they are not entitled to.

4. To provide integrity which ensures that only authorized individuals should ever be able change or modify information.

5. To provide availability which ensure that the data or system itself is available for use when authorized user wants it.

6. To provide authentication which deals with the desire to ensure that an authorized individual.

7. To provide non-repudiation which deals with the ability to verify that message has been sent and received by an authorized user.

**OR**

**1. Confidentiality:** The principle of confidentiality specifies that only sender and intended recipients should be able to access the contents of a message. Confidentiality gets compromised if an unauthorized person is able to access the contents of a message

**2. Integrity:** when the contents of the message are changed after the sender sends it, but before it reaches the intended recipient, we say that the integrity of the message is lost.

**3. Authentication:** Authentication helps to establish proof of identities. The Authentication process ensures that the origin of a message is correctly identified.

4. **Availability:** The goal of availability s to ensure that the data, or the system itself, is available for use when the authorized user wants it.

| 4 | | **Attempt any THREE of the following :** | **12 M** |
|---|---|---|---|
| | **a** | **Explain the four roles of database administrator.** | 4 M |
| | **Ans** | **1. Schema Definition** The Database Administrator creates the database schema by executing DDL statements. Schema includes the logical structure of database table (Relation) like data types of attributes, length of attributes, integrity constraints etc.<br><br>**2. Storage structure and access method definition** The DBA creates appropriate storage structures and access methods by writing a set of definitions which is translated by data storage and DDL compiler.<br><br>**3. Schema and physical organization modification** DBA writes set of definitions to modify the database schema or description of physical storage organization. | 1 Mark for each role |

| | | | |
|---|---|---|---|
| | | **4. Granting authorization for data access** The DBA provides different access rights to the users according to their level. Ordinary users might have highly restricted access to data, while you go up in the hierarchy to the administrator, you will get more access rights. Integrity constraints specifications: Integrity constraints are written by DBA and they are stored in a special file which is accessed by database manager while updating data.<br><br>**5. Routine Maintenance some of the routine maintenance activities of a DBA is given below.**<br><br>(i) Taking backup of database periodically<br><br>(ii) Ensuring enough disk space is available all the time.<br><br>(iii) Monitoring jobs running on the database.<br><br>(iv) Ensure that performance is not degraded by some expensive task submitted by some users.<br><br>**6. Integrity- constraint specification:** Integrity constraints are written by DBA and they are stored in a special file, which is accessed by database manager, while updating the data. | |
| | **b** | **State and Explain 1 NF and 2 NF with example.** | 4 M |
| | **Ans** | **First Normal Form (1NF):**<br><br>A relation R is said to be in first normal form (1NF) if the domain of all attributes of R are atomic.<br><br>OR<br><br>A table is in the first normal form if it contains no repeating elements groups. Example: Supplier(sno,sname,location,pno,qty) | For 1NF - 2 Marks, For 2NF -2 Marks |

| SNO | SNAME | LOCATION | PNO | QTY |
|---|---|---|---|---|
| S1 | Abc | Mumbai | P1 | 200 |
| S2 | Pqr | Pune | P2 | 300 |
| S3 | Lmn | Delhi | P1 | 400 |

The above relation is in 1NF as all the domains are having atomic value. But it is not in 2NF.

**Second Normal Form (2NF):**

A relation is said to be in the second normal form if it is in first normal form and all the non key attributes are fully functionally dependent on the primary key.

Example:

In the above relation NAME, LOCATION depends on SNO and QTY on (SNO, PNO) so the table can be split up into two tables as Supplier(SNO,SNAME,LOCATION) and SP(SNO,PNO,QTY) and now both the tables are in second normal form.

Supplier

| SNO | SNAME | LOCATION |
|-----|-------|----------|
| S1  | Abc   | Mumbai   |
| S2  | Pqr   | Pune     |
| S3  | Lmn   | Delhi    |

Supplier_Product

| SNO | PNO | QTY |
|-----|-----|-----|
| S1  | P1  | 200 |
| S2  | P2  | 300 |
| S3  | P1  | 400 |

| | c | **Draw the block structure of PL/SQL. List advantages of PL/SQL.** | | 4 M |
|---|---|---|---|---|
| | **Ans** | Declare (Optional)<br><br>   --Use for declaring variables<br><br>Begin (Mandatory)<br><br>   --Use for writing executable code;<br><br>Exception (Optional)<br><br>   --Use to write exceptions to be catch during run time.<br><br>End; (Mandatory)<br><br>   --To terminate PL-SQL block/ code. | | For block structure - 2 Marks, For advantages -2 Marks |

| | | | |
|---|---|---|---|
| | | **Advantages of PL/SQL:**<br><br>1. PL/SQL is portable and high transaction processing language.<br><br>2. PL/SQL is in fact procedural language but it also supports object oriented programming.<br><br>3. It allows user to write as well as access the functions and procedures from outside the programs.<br><br>4. It has got built in libraries of packages. | |
| | d | **Write step by step syntax to create, open and close cursor in PL/SQL.** | 4 M |
| | Ans | A cursor holds the rows (one or more) returned by a SQL statement.<br><br>**Declaring:** This term is used to declare a cursor so that memory initialization will take place.<br><br>A cursor is declared by defining the SQL statement that returns a result set.<br><br>Example:<br><br>Declare CURSOR Winter_18 IS SELECT roll_no, std_name, percentage FROM student;<br><br>**Opening:** A Cursor is opened and populates data by executing the SQL statement defined by the cursor.<br><br>Example:<br><br>Open Winter_18;<br><br>**Closing a Cursor:** This forces cursor for releasing the allocated memory assigned/ occupied by cursor.<br><br>Example:<br><br>CLOSE Winter_18; | 2 marks, Opening: 1 mark, Closing cursor: 1 mark |
| | e | **Explain Transaction ACID properties.** | 4 M |
| | Ans | **ACID properties of transaction**<br><br>**1. Atomicity:** When one transaction takes place, many operations occur under one transaction. Atomicity means either all operations will take place property and reflect in the database or none of them will be reflected.<br><br>**2. Consistency:** Consistency keeps the database consistent. Execution of a transaction | For each property - 1 Mark |

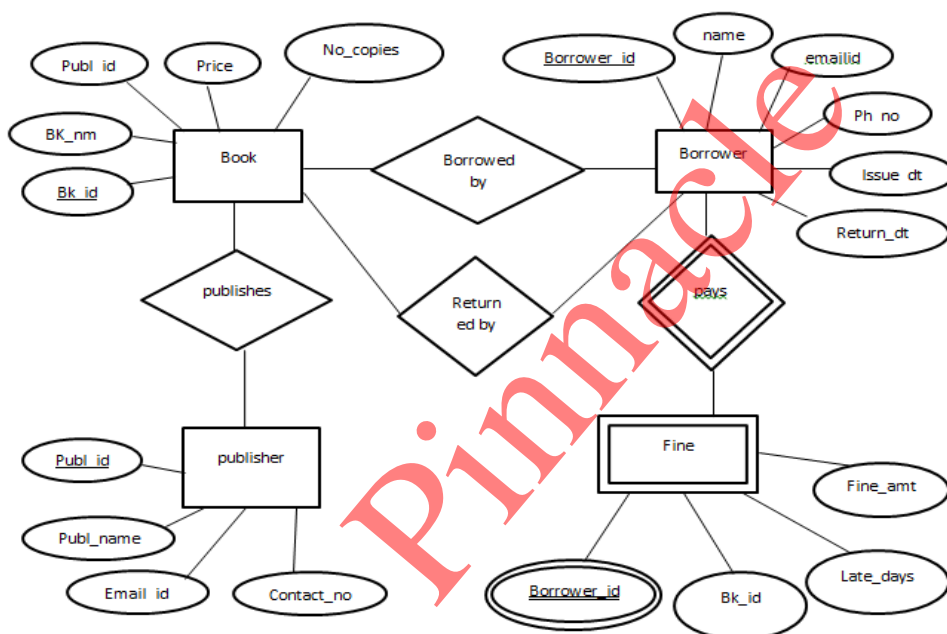| | | | |
|---|---|---|---|
| | | needs to take place in isolation. It helps in reducing complications of executing multiple transactions at a time and preserves the consistency of the database.<br><br> **3. Isolation:** It is necessary to maintain isolation for the transactions. This means one transaction should not be aware of another transaction getting executed. Also their intermediate result should be kept hidden**.**<br><br>**4. Durability:** When a transaction gets completed successfully, it is important that the changes made by the transaction should be preserved in database in spite of system failures. | |
| **5** | | **Attempt any TWO of the following :** | **12 M** |
| | **a** | **Draw an E-R diagram of library management system considering issue and return, fine calculation facility, also show primary key, weak entity and strong entity.** | 6 M |
| | **Ans** |  | Correct entities: 2M, correct symbols: 2M, Correct relationships: 2M |
| | **b** | **Consider the following database**<br>**Employee(emp_id,emp_name,emp_city,emp_addr,emp_dept,join_date)**<br><br>  i)   **Display the emp_id of employee who live in city 'Pune' or 'Nagpur'.**<br><br>  ii)  **Change the employee name 'Ayush' to 'Ayan'.**<br><br>  iii) **Display the total number of employee whose dept is 50.** | 6 M |
| | **Ans** | i) Display the emp_id of employee who live in city 'Pune' or 'Nagpur'<br><br>select emp_id<br><br>from Employee | Each query : 2M |

| | | | |
|---|---|---|---|
| | | where emp_city='Pune' or emp_city='Nagpur' | |
| | | ii) Change the employee name 'Ayush' to 'Ayan' | |
| | | update Employee | |
| | | set emp_name='Ayan' | |
| | | where emp_name='Ayush' | |
| | | iii) Display the total number of employee whose dept is 50 | |
| | | Select count(*) | |
| | | from Employee | |
| | | where emp_dept=50; | |
| | c | **Consider the following schema Depositor (ACC_no, Name, PAN, Balance). Create a view on Depositor having attributes(ACC_No,PAN) where balance is greater than 100000** | 6 M |
| | Ans | create view v1<br><br>    as<br><br> select ACC_No,PAN<br><br>from Depositor<br><br>where balance > 100000; | Correct logic 3M, Correct syntax :3M |
| 6 | | **Attempt any TWO of the following :** | **12 M** |
| | a | **Create a sequence**<br><br>**i) Sequence name is Seq_1, Start with 1, increment by 1, minimum value 1, maximum value 20.**<br><br>**ii) Use a seq_1 to insert the values into table Student( ID Number(10), Name char (20));**<br><br>**iii) Change the Seq_1 max value 20 to 50.**<br><br>**iv) Drop the sequence.** | 6 M |
| | Ans | i) create sequence Seq_1 start with 1 increment by 1 minvalue 1 maxvalue 20;<br><br>ii) insert into student values(Seq_1.nextval,'ABC'); | Query 1: 2M, Query 2: 2M, Query 3 : 1M, Query 4 : 1M |

| | | | |
|---|---|---|---|
| | | iii)    Alter sequence Seq_1 maxvalue 50;<br><br>iv)    Drop sequence Seq_1; | |
| | b | **Write a PL/SQL program which accepts the customer_ID from the user. If the enters an invalid ID then the exception invalid_id is raised using exception handling.** | 6 M |
| | Ans | DECLARE<br>  c_id numeric(10);<br>  invalid_id_Exception Exception;<br>BEGIN<br>  c_id:=&c_id;<br>  if(c_id<0) then<br>  raise invalid_id_Exception;<br>  end if;<br>EXCEPTION<br>  WHEN invalid_id_Exception THEN<br>    dbms_output.put_line('Invalid customer id');<br>END; | Correct logic : 3M, Correct syntax : 3M |
| | c | **i) create user 'Rahul'**<br><br>**ii) grant create, select,insert,update, delete, drop privilege to 'Rahul'**<br><br>**iii) Remove the select privilege from user 'Rahul'** | 6 M |
| | Ans | (i) **create user Rahul identified by rahul1234;**<br><br>(ii)<br><br>  1)  assuming table Employee for granting permissions to user 'Rahul' for select, insert, update and delete privilege)<br><br>**Grant select, insert,update,delete on employee to Rahul;**<br><br>  2)  for create and drop privilege which are system privileges not specific to any object such as table<br><br>**Grant connect, resource, DBA to Rahul;**<br><br> iii)  (assuming table Employee for revoking permissions to user 'Rahul')<br><br>**Revoke select on Employee from Rahul;** | each query : 2M |