WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Database Management System **Subject Code: 22319**

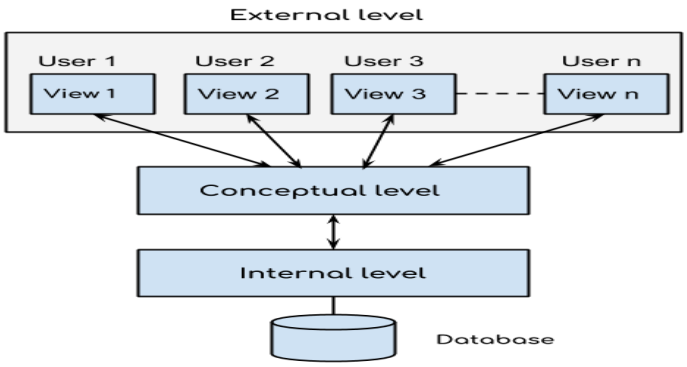
Important Instructions to examiners:

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.	**(a) Ans.**	**Attempt any FIVE of the following:** **State any two advantages of DBMS over file processing system.** **Advantages of DBMS over file processing system:** • Reduction in Data redundancy • Data consistency and integrity • Data security • Privacy • Easy access of data • Easy recovery • Flexibility	**10** **2M** ***Any two advantages 1M each***
	(b) Ans.	**Draw three level architecture of DBMS.**	**2M**

WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Database Management System **Subject Code: 22319**

			***Correct diagram 2M***
	(c) Ans.	**Define table and field.** **Table:** A table is a collection of related data held in table format. It is a set of data elements using a model of vertical columns and horizontal rows. **Field:** Each table contains field which is a data structure, used to hold the data. It can also be termed as attribute.	**2M** ***Each definition 1M***
	(d) Ans.	**Enlist DML commands.** • Insert - used to insert new row into table • Delete- used to delete a row from the table • Update – used to modify data in the table. • Select – used to view data from a table.	**2M** ***Any 2 commands 1M each***
	(e) Ans.	**Define primary key and foreign key.** **Primary key** is an attribute or set of attributes used to identify an entity from an entity set. All the values of a primary key should be unique and null values are not allowed. **Foreign key** is an attribute of an entity which is the primary key of another entity. It is used to show relation between entities. The table containing foreign key is called the child table.	**2M** ***Each definition 1M***
	(f) Ans.	**List any four string functions in SQL.** Initcap(String) – converts first character of string to upper case Upper(String) – converts the string to upper case Lower(String) – converts string to lower case Length(String) – returns the number of characters in the string Instr(String, sub) – returns the location of the substring Lpad(String,char,number) – returns the string left padded with the character specified to a total of length specified. Rpad(String,char,number) – returns the string right padded with the	**2M** ***Any four string functions ½M each***

<table>
<tr><td></td><td></td><td>character specified to a total of length specified.
Ltrim(String) -removes white space or other specified characters from the left end of the string
Rtrim(String)--removes white space or other specified characters from the right end of the string
Replace(String, char,char) – replace all occurrence of a substring by another substring
Substring(String,number) – extracts substring from the string
Translate(String,char,char) – replace all occurrence of characters by other characters</td><td></td></tr>
<tr><td></td><td>(g)
Ans.</td><td>State any two advantages of functions in PL/SQL.
Advantages of functions in PL/SQL:
• Work can be divided into smaller modules so that it can be manageable and also enhances the readability of the code.
• It promotes reusability.
• It is secure, as the code is in the database and hides the internal database details from the user.
• It improves performance against running SQL queries multiple times.</td><td>2M

Any two advantages 1M each</td></tr>
<tr><td>2.</td><td>(a)

Ans.</td><td>Attempt any THREE of the following:
Distinguish between Network and Hierarchical model. (Any four points)
<table>
<tr><th>Sr. No.</th><th>Network Model</th><th>Hierarchical model</th></tr>
<tr><td>1</td><td>Represents tree like structure with many roots</td><td>Represents tree like structure with one root</td></tr>
<tr><td>2</td><td>Reflects M:N(many to many) relations</td><td>Reflects 1:N (one-to-many)relations</td></tr>
<tr><td>3</td><td>Allows a child to have more than one parent</td><td>There can be only one parent node</td></tr>
<tr><td>4</td><td>Relationship is represented as pointers or links</td><td>Relationships between records is of parent-child type</td></tr>
<tr><td>5</td><td>This model is free from such inconsistency as there is only a single occurrence of a record set.</td><td>There are multiple occurrence of child records and therefore inconsistency</td></tr>
<tr><td>6</td><td>Searching a record is easy as there are multiple paths</td><td>Searching a record is difficult as a child can be</td></tr>
</table></td><td>12
4M

Any four points 1M each</td></tr>
</table>

			to a data element.	reached only through a parent

	(b) Ans.	**Explain any four set operators in SQL with example.** 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. **1) Union:** The Union of two or more sets contains all elements, which are present in either or both. Union works as or. The duplicates of both the tables will appear only once. *E.g*. select ename from emp1 union select ename from emp2; **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 emp1 union all select ename from emp2; **3) Intersection:** The intersection of two sets includes elements which are present in both. *E.g.* select ename from emp1 intersect select ename from emp2; **4) Minus:** The minus of two sets includes elements from set1 minus elements of set2. *E.g*. select ename from emp1 minus select ename from emp2;	**4M** ***1M each for explanation of operators with example***
	(c) Ans.	**Describe Views and write a command to create view.** A view is a virtual table based on the result set of the SQL statement. The fields in a view are fields from one or more than one table in the database. SQL functions, where, join statements can be added to a view and the data in it can be presented as if it were from one table. The database engine recreates the data, using the view's SQL statement, every time a user queries a view. A view can be updated using the create or replace view command. For deleting a view, drop query can be used. **General syntax to create a view:** create view viewname as select query. ***Eg:*** create view vw_student as select stud_id, stud_name,ssc_per from student;	**4M** ***Explanation 3M*** ***General syntax/ example 1M***
	(d) Ans.	**Explain implicit and explicit cursors.** A cursor is a temporary work area created in system memory when an SQL statement is executed. A cursor is a set of rows together with a pointer that identifies a current row. It is a database object to retrieve	**4M**

		data from result set on row at a time. **Implicit cursor:** these types of cursors are generated and used by the system during the manipulation of a DML query. An implicit cursor is also generated by the system when a single row is selected by a SELECT command. Programmers cannot control the implicit cursors. **Explicit cursor:** this type of cursor is created by the user when the select command returns more than one row, and only one row is to be processed at a time. An explicit cursor can move from one row to another in a result set. An explicit cursor uses a pointer that holds the record of a row. To create an explicit cursor the following steps are used. 1. Declare cursor: this is done in the declaration section of PL/SQL program. 2. Open: this step is done before the cursor is used to fetch the records. 3. Fetch: used to retrieve data row by row from the cursor. 4. Close: once the processing of the data is done, the cursor can be closed.	***Each explanation 2M***
3.	**(a) Ans.**	**Attempt any THREE of the following:** **State and explain 3NF with example.** **3NF:** An entity is said to be in the third normal form when, 1) It satisfies the criteria to be in the second normal form. 2) There exists no transitive functional dependency. (Transitive functional dependency can be explained with the relationship link between three tables. If table A is functionally dependent on B, and B is functionally dependent on C then C is transitively dependent on A). Let us consider the Schema given: (Supplier_no,SupplierName,Supplier_city,Order_no,Order_quantity, Order_amount,Product_code,Product name,rate) Step 1.To convert it into 2NF, We have to decompose the given table into two tables with fully functional dependencies and establishing a referential integrity constraint relationship among the two tables. Table2: Supplier Details (Supplier_no,Supplier_name,Supplier_city) Table 3:Order Details (Order_no,Order_ quantity,Order_amount,Supplier_no Product_code,	**12** **4M** ***Explanation 2M*** ***Any example 2M***

		product_name,rate) Now the above two tables are in 2NF Step 2: To convert the above tables in 3NF, we have to decomposehem in three tables satisfying the transitive dependencies property. Table 4: Supplier Details (Supplier_no,Supplier_name,Supplier_city) Table 5: Product Details: (Product_code, product_name,rate) Table 6: Order Details (or Transaction Details) ((Order_no,Supplier_no,Product_code,Order_quantity,Order_amount) Hence the above three tables are satisfying Transitive dependencies. Thus they are in 3NF.	
	(b) Ans.	**Define index. Explain it's types.** An **Index** is a schema object. It is used by the oracle server to improve the speed of retrieval of the rows from a table .Indexes are of two types based on number of columns included in the index. The **types of index** are: 1) **Simple index**: An index created on a single column of table is called as simple index ***Syntax:*** SQL>Create Index index_name on tablename(attribute); *Example:*Create index emp_index on emp(empno); 2) **Composite Index:** An index created on more than one column is called composite index. ***Syntax:*** SQL>Create Index index_name on tablename(attribute1,attribute2); *Example:* Create index emp_index on emp(empno,ename);	**4M** ***Definitio n 1M*** ***Each type 1½M***
	(c) Ans.	**Explain Exception handling with it's types.** 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 − 1) System-defined exceptions/Predefined exceptions/Built-in exceptions	**4M** ***Explana tion 2M***

		2) User-defined exception
		Predefined exceptions- PL/SQL provides predefined Exception, which are executed when any database rule is violated by a program. Example: NO_DATA_FOUND, ZERO_DIVIDE.
		Syntax for Predefined Exception Handling:
		The general syntax for exception handling is as follows.
		DECLARE
		<declarations section>
		BEGIN
		<executable command(s)>
		EXCEPTION
		<exception handling goes here >
		WHEN exception1 THEN
		exception1-handling-statements
		WHEN exception2 THEN
		exception2-handling-statements
		WHEN exception3 THEN
		exception3-handling-statements
		.....
		WHEN others THEN
		exception3-handling-statements
		END;
		User defined Exceptions:
		PL/SQL allow us to define our own exception according to the need of our program. A user defined exception must be declared and then raised explicitly.
		Syntax for User defined Exception:
		DECLARE
		exception_name EXCEPTION;
		BEGIN
		IF condition THEN
		RAISE exception_name;
		END IF;
		EXCEPTION
		WHEN exception_name THEN
		statement;
		END;
			Types 2M

	(d) Ans.	**Explain ACID properties of traction.** A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further. **ACID Properties** A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain **A**tomicity, **C**onsistency, **I**solation, and **D**urability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity. • **Atomicity**: This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction. • **Consistency:** The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well. • **Isolation**: In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction. • **Durability:** The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.	**4M** ***Explanation of each property 1M***
4.	**(a) Ans.**	**Attempt any THREE of the following:** **Explain strong and weak entity set.** **Strong entity set:** An **entity set** that has sufficient attributes to form a primary key is	**12** **4M**

		called as **Strong entity set**. *Example:* Employee is a Strong entity with attributes as empid, name, address, salary, birthdate among which empid can be considered as primary key. **Weak entity set:** The **entity set** which does not have sufficient attributes to form a primary key is called as **Weak entity set**. A **weak entity** is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key. The foreign key is typically a primary key of an entity it is related to. *Example:* Employee has "dependents" with name, birthdate, and relationship to employee and it can be related to employee with the help of empid, so "dependents" is a weak entity which depends on strong entity "Employee".	***Each entity set 2M***
	(b) Ans.	**Describe create & alter command with syntax & example.** 1) The SQL **CREATE TABLE** statement is used to create a new table. ***Syntax*** The basic syntax of the CREATE TABLE statement is as follows – *CREATE TABLE table_name* *(* *column1 datatype (size),* *column2 datatype(size),* *column3 datatype(size),* *....* *);* ***Example:*** CREATE TABLE Persons (PersonIDnumber(10), LastNamevarchar2(20), FirstNamevarchar2(20), Address varchar2(20), City varchar2(20)	**4M** ***Each command 2M***

		); 2)**The ALTER TABLE** statement is used to add, delete, or modify columns in an existing table. The ALTER TABLE statement is also used to add and drop various constraints on an existing table. i) To add Columns in a table *Syntax:*ALTER TABLE *table_name* ADD *column_name datatype*; *Example* ALTER TABLE Customers ADD Email varchar2(20); ii) To delete a column in a table ALTER TABLE *table_name* DROP COLUMN *column_name*; *Example* ALTER TABLE Customers DROP COLUMN Email; iii) To modify a column in a table Syntax: ALTER TABLE *table_name* MODIFY COLUMN *column_name datatype*; Example ALTER TABLE Customers MODIFY COLUMN *customeridnumeric(10)*; iv) To add Constraints in A table *Syntax:* ALTER TABLE *table_name* ADD constraint constraintname (*column_name)*; *Example:* ALTER TABLE Customers ADD constraint primary key(CustomerID);	

	(c) Ans.	**Define database trigger. How to create and delete 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).
		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;
		To delete a trigger:
		Syntax:
		DROP TRIGGER trigger_name.
			4M
			Definition 1M
			Create 2M
			Delete 1M

	(d) Ans.	**Explain any one control structure in PL/SQL with example.** PL/SQL has three categories of control statements: conditional selection statements, loop statements and sequential control statements. PL/SQL categories of control statements are: • **Conditional selection statements**, which run different statements for different data values.	**4M**

		The conditional selection statements are IF and CASE.
		• **Loop statements**, which run the same statements with a series of different data values.
		
		The loop statements are the basic LOOP, FOR LOOP, and WHILE LOOP.
		
		The EXIT statement transfers control to the end of a loop. The CONTINUE statement exits the current iteration of a loop and transfers control to the next iteration. Both EXIT and CONTINUE have an optional WHEN clause, where you can specify a condition.
		
		Sequential control statements, which are not crucial to PL/SQL programming.
		
		The sequential control statements are GOTO, which goes to a specified statement, and NULL, which does nothing.
		
		1)Conditional Control: IF and CASE Statements:
		The IF statement lets us execute a sequence of statements conditionally. That is, whether the sequence is executed or not depends on the value of a condition. There are three forms of IF statements: IF-THEN, IF-THEN-ELSE, and IF-THEN-ELSIF. The CASE statement is a compact way to evaluate a single condition and choose between many alternative actions.
		
		IF-THEN Statement: The simplest form of IF statement associates a condition with a sequence of statements enclosed by the keywords THEN and END IF (not ENDIF), as follows:
		IF condition THEN
		sequence_of_statements
		END IF;
		
		IF-THEN-ELSE Statement: The second form of IF statement adds the keyword ELSE followed by an alternative sequence of statements, as follows:
		IF condition THEN
		sequence_of_statements1
			Explanation of Any one control structure 4M

```
ELSE
  sequence_of_statements2
END IF;
```

IF-THEN-ELSIF Statement: The third form of IF statement uses the keyword ELSIF (not ELSEIF) to introduce additional conditions, as follows:

```
IF condition1 THEN
  sequence_of_statements1
ELSIF condition2 THEN
  sequence_of_statements2
ELSE
  sequence_of_statements3
END IF;
```

CASE Statement: Like the IF statement, the CASE statement selects one sequence of statements to execute.

```
IF grade = 'A' THEN
dbms_output.put_line('Excellent');
ELSIF grade = 'B' THEN
dbms_output.put_line('Very Good');
ELSIF grade = 'C' THEN
dbms_output.put_line('Good');
ELSIF grade = 'D' THEN
dbms_output. put_line('Fair');
ELSIF grade = 'F' THEN
dbms_output.put_line('Poor');
ELSE
dbms_output.put_line('No such grade');
END IF;

END CASE;
```

2) Iterative Control: LOOP and EXIT Statements: LOOP statements let us execute a sequence of statements multiple times. There are three forms of LOOP statements: LOOP, WHILE-LOOP, and FOR-LOOP.

LOOP: The simplest form of LOOP statement is the basic (or

infinite) loop, which encloses a sequence of statements between the keywords LOOP and END LOOP, as follows:

```
LOOP
sequence_of_statements
END LOOP;
```

WHILE-LOOP: The WHILE-LOOP statement associates a condition with a sequence of statements enclosed by the keywords LOOP and END LOOP, as follows:

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

Before each iteration of the loop, the condition is evaluated. If the condition is true, the sequence of statements is executed, then control resumes at the top of the loop. If the condition is false or null, the loop is bypassed and control passes to the next statement.

FOR-LOOP: Whereas the number of iterations through a WHILE loop is unknown until the loop completes, the number of iterations through a FOR loop is known before the loop is entered. FOR loops iterate over a specified range of integers. The range is part of an *iteration scheme*, which is enclosed by the keywords FOR and LOOP. A double dot (..) serves as the range operator. The syntax follows:

```
FOR counter IN [REVERSE] lower_bound..higher_bound LOOP
sequence_of_statements
END LOOP
```

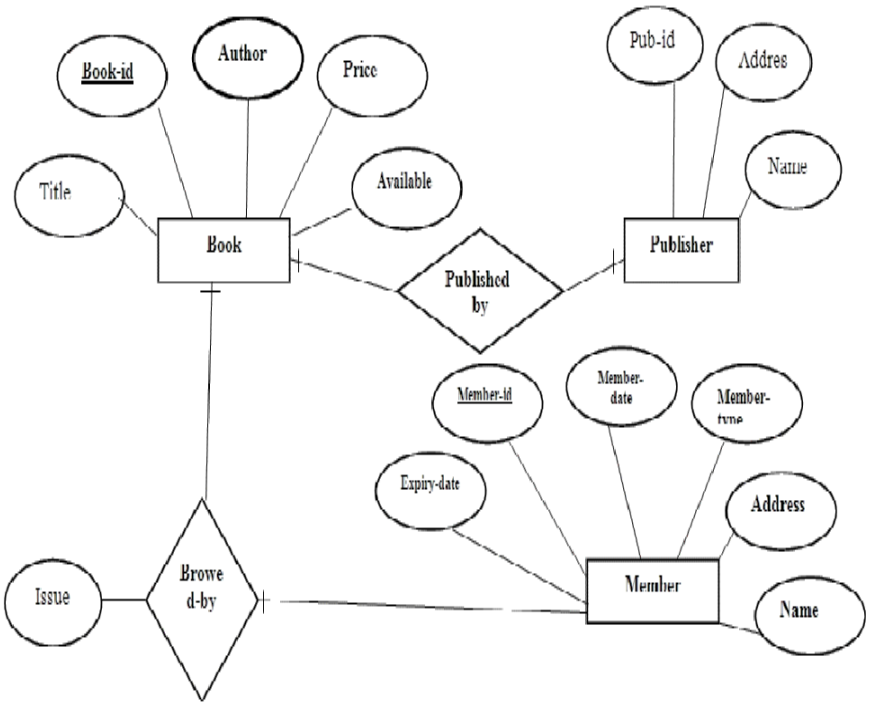
3)Sequential Control: GOTO and NULL Statements: Unlike the IF and LOOP statements, the GOTO and NULL statements are not crucial to PL/SQL programming.

GOTO Statement

The GOTO statement branches to a label unconditionally.

```
BEGIN
  ...
  GOTO insert_row;
```

		... <<insert_row>> INSERT INTO empVALUES ... END;	
	(e) Ans.	**Describe database backups with it's types.** **Regular backups** are required to protect database and ensure *its* restoration in case of failure. Various *backup types* provide different protection to our database. Backing up and restoring data is one of the most important responsibilities of IT professionals Three common types of database backups can be run on a desired system: **normal (full), incremental and differential**. **i) Normal or Full Backups:** When a normal or full backup runs on a selected drive, all the files on that drive are backed up. This, of course, includes system files, application files, user data — everything. Those files are then copied to the selected destination (backup tapes, a secondary drive or the cloud), and all the archive bits are then cleared. Normal backups are the fastest source to restore lost data because all the data on a drive is saved in one location. **ii) Incremental Backups:** A common way to deal with the long running times required for full backups is to run them only on weekends. Many businesses then run incremental backups throughout the week since they take far less time. An incremental backup will grab **only the files that have been updated since the last normal backup**. Once the incremental backup has run, that file will not be backed up again unless it changes or during the next full backup. **iii) Differential Backups:** An alternative to incremental database backups that has a less complicated restore process is a differential backup. Differential backups and recovery are similar to incremental in that these backups grab only files that have been updated since the last normal backup. However, differential backups do not clear the archive bit. So a file	**4M** ***Description 2M*** ***Types 2M***

		that is updated after a normal backup will be archived every time a differential backup is run until the next normal backup runs and clears the archive bit.	
5.	**(a)** **Ans.**	**Attempt any TWO of the following:** **Draw an ER diagram for library management system. (Use Books, Publisher & Member entities).** ***(Note: Consider any relevant diagram)*** 	**12** **6M** ***Correct entities 2M*** ***Correct symbols 2M*** ***Correct relations hips 2M***
	(b) **Ans.**	**Write a command to crate table student (rollno, Stud_name, branch, class, DOB, City, Contact_no) and write down queries for following:** **(i) Insert one row into the table** **(ii) Save the data** **(iii) Insert second row into the table** **(iv) Undo the insertion of second row** **(v) Create save point S_1.** **(vi) Insert one row into the table.**	**6M**

		SQL>Create table student(**Rollno number(5),** **Stud_name char(10,** **branch varchar(10),** **class varchar(10),** **DOB date,** **city varchar(15),** **Contact_no number(12)** **);** (i) Insert one row into the table: **SQL>Insert into student values(1,'Ram','CO','FirstYear','12-jun-2001','Pune',98576867)** (ii) Save the data: **SQL> commit;** **(OR)** **SQL> commit work;** (iii)Insert second row into the table: **SQL>Insert into student values(2,'Raj','CO','FirstYear','22-Sep-2002','Mumbai',98896863)** (iv)Undo the insertion of second row: **SQL> rollback;** **(OR)** **SQL> rollback work;** (v)Create savepoint s1: **SQL>Savepoint s1;** (vi) insert one row into the table: **SQL>Insert into student values(3,'Beena','CO','FirstYear','30-Dec-2002','Mumbai',97846455)**	***Create table 3M*** ***Each correct Query ½M each***
	(c)	**Consider following schema:** **EMP (empno, deptno, ename, salary, designation, join_date, DOB, dept_location). Write down SQL queries for following:** **(i) Display employees name & number in decreasing order of salary.**	**6M**

	Ans.	**(ii) Display employee name & employee number whose designation is Manager.** **(iii) Display age of employees with ename.** **(iv) Display total salary of all employees.** **(v) Display employee names having deptno as 20 and dept_location is Mumbai** **(vi) Display name of employee who earned lowest salary.** (i)Display employees name &number in descending order of salary: **SQL> select ename,empno from EMP order by salary desc;** (ii) Display employee name & employee number whose designation is Manager. **SQL> select ename,empno from EMP where designation='Manager';** (iii) Display age of employees with ename **SQL>select round ((sysdate - DOB) /365, 0) as "age",ename from EMP;** **OR** **select months_between(TRUNC(sysdate),DOB)/12 as "age" ,ename from EMP ;** **(**Note consider any other logic also)** (iv)Display total salary of all employees. **SQL> select sum(salary) from EMP;** (v)Display employee names having deptno as 20 and dept_location is Mumbai. **SQL> select enamefrom EMP where deptno=20 and dept_location='Mumbai';** (vi)Display name of employee who earned lowest salary **SQL> select ename from EMP where salary=(select min(salary) from EMP);**	***Each correct Query 1M***
6.	**(a)**	**Attempt any TWO of the following:** **Consider the structure for book table as Book-Master (bookid, bookname, author, no_of copies, price)** **Write down SQL queries for following:**	**12** **6M**

	Ans.	**(i) Write a command to create Book_master table.** **(ii) Get authorwise list of all books.** **(iii) Display all books whose price is between ₹ 500 & ₹ 800.** **(iv) Display all books with details whose name start with 'D'.** **(v) Display all books whose price is above ₹ 700.** **(vi) Display all books whose number of copies are less than 10.** (i)Write a command to create Book_Master table table. **SQL>Create table Book-Master(** **bookid number(5),** **bookname char(10),** **authorvarchar(20),** **no_of_copiesnumber(10),** **price number(10,2)** **);** (ii)Get authorwise list of all books. **SQL>Select sum(no_of copies) from Book_Master group by author;** (iii)Display all books whose price is between Rs.500 & Rs. 800 **SQL> Select * from Book_Master where price between 500 and 800;** **OR** **SQL> Select * from Book_Master where price >=500 and price<=800;** (iv) Display all books with details whose name start with 'D' **SQL> Select bookname from Book_Master where bookname like 'D%';** (v)Display all books whose price is above Rs. 700 **SQL>Select * from Book_Master where price >700;** (vi) Display all books whose number of copies are less than 10 **SQL>Select * from Book_Master where no_of_copies<10;**	***Each correct Query 1M***
	(b)	**Write a PL/SQL program to print n even numbers using For Loop.** ***(Note: Any other logic can be allowed)***	**6M**

	Ans.	declare
		num number;
		n number:=&n;
		begin
		for num in 1..n loop
		if(mod(num,2)=0) then
		dbms_output.put_line('Even no are :'||num);
		end if;
		end loop;
		end;
			Correct logic 3M
			
			Correct syntax 3M

	(c) **Ans.**	**Describe database privileges. Write down the procedure for granting & revoking privileges in database objects to the users.** **Database privileges:** When multiple users can access database objects, authorization can be controlled to these objects with privileges. Every object has an owner. Privileges control if a user can modify an object owned by another user. Privileges are granted or revoked either by the instance administrator, a user with the ADMIN privilege or, for privileges to a certain object, by the owner of the object. **1) System Privileges:** System privileges are privileges given to users to allow them to perform certain functions that deal with managing the database and the server e.gCreate user, Create table, Drop table etc. **2) Object Privileges:** Object privileges are privileges given to users as rights and restrictions to change contents of database object – where database objects are things like tables, stored procedures, indexes, etc. Ex. Select,insert,delete,update,execute,references etc **<u>Procdure for granting privileges</u>** **Grant:** This command is used to give permission to user to do operations on the other user's object. ***Syntax:*** **Grant**<object privileges>**on**<object name>**to**<username>[with grant option] ; ***Example:*** Grant select, update on emp to user1; **<u>Procedure for revoking privileges</u>** **Revoke:** This command is used to withdraw the privileges that has	**6M** ***Database Privileges 2M*** ***Procedure for granting privileges 2M***

		been granted to a user. ***Syntax:*** **Revoke** <object privileges>**on**<object name>**from** <username> ; ***Example*****:** Revoke select, update on emp from user1;	***Procedure for revoking privileges 2M***