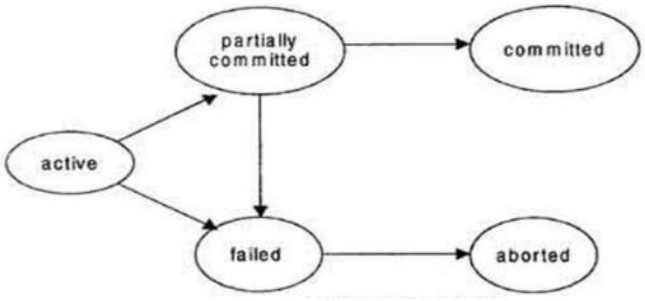
**SUMMER – 2022 EXAMINATION**
**MODEL ANSWER**

**Subject: Database Management** **Subject Code:** **22416**

**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 anyequivalent 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.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

| Q. No | Sub Q.N. | Answer | Marking Scheme |
|---|---|---|---|
| **1.** | **a) Ans.** | **Attempt any FIVE of the following:**<br>**Define database. Give any two suitable example of database.**<br>**Definition:** Database is defined as collection of related data.<br>**Example:** Banking systems , computerized medical records , online shopping system , library management system are the few examples of the database. | **10**<br>**2M**<br>***1M for definition***<br>***1M for example*** |
| | **b) Ans.** | **List any four aggregate functions.**<br>An aggregate function in SQL returns one value after calculating multiple values of a column. Various types of SQL aggregate functions are:<br>• Count()<br>• Sum() | **2M**<br>***½ M each, any four functions*** |

| | | | |
|---|---|---|---|
| | | • Avg()<br>• Min()<br>• Max() | |
| | **c) Ans.** | **Define view**<br>A view is a logical extract of a physical relation i.e. it is derived from any base relation.<br>OR<br>View: Views are virtual relations mainly used for security purpose, and can be provided on request by a particular user. | **2M**<br>***2M for definition*** |
| | **d) Ans.** | **Enlist types of exceptions in PL/SQL**<br>1) **Predefined Exception/System Defined 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.<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: Raise_application_error(Exception_Number,Error_Message); | **2M**<br>***1M for each exception.*** |
| | **e) Ans.** | **Draw diagram of transaction**<br>  | **2M**<br>***2M for correct diagram*** |
| | **f) Ans.** | **Enlist types of database failures**<br>There are many types of failures that can affect databaseprocessing. Following are the types of failure: | **2M**<br>***½ M each, any four types*** |

| | | | |
|---|---|---|---|
| | | • Hardware failures<br>• Software failures<br>• System crashes<br>• Media Failures<br>• Network Failures<br>• Transaction Failures<br>• Logical Error<br>• System error<br>• Application software error | |
| | **g) Ans.** | **Define Synonyms**.<br>A synonym is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects. We generally use synonyms when we are granting access to an object from another schema and we don't want the users to have to worry about knowing which schema owns the object. | **2M**<br>***2M for correct definition*** |
| **2.** | **a) Ans.** | **Attempt any <u>THREE</u> of the following:**<br>**Differentiate between Drop and Truncate command** | **12**<br>**4M**<br>***1M for each valid point, any four points*** |

| **DROP** | **TRUNCATE** |
|---|---|
| It is a DDL statement which deals with the structure of the table along with data | It is DDL statement which deals only with the data from the table |
| It deletes entire table at once from the disk | It deletes all records from the table at once. |
| Column structure of table does not remain on the disk | Empty column structure of the table remains on the disk |
| Syntax:<br>Drop<tablename>; | Syntax:<br>Truncate table <tablename>; |
| Example:<br>Drop employee; | Example:<br>Truncate table employee; |

| | | | |
|---|---|---|---|
| | **b) Ans.** | **Explain any two types of joins**<br>SQL Join types are as follows:<br>**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.<br>**Syntax:**<br>SELECT tablename.column1_name,tablename.column2_name FROM table_name1,table_name2 where table_name1.column_name=table_name2.column_name;<br>**Example:**<br>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;<br><br>**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.<br>**Syntax:**<br>SELECT a.column_name, b.column_name... FROM table1 a, table1 b WHERE a.common_filed = b.common_field;<br>**Example:**<br>Select x.stud_name, y.stud_name from stud_infox,stud_info y Where x.leader= y.stud_id;<br><br>**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.<br>**Syntax:**<br>Select column1name,column2name from table1name any_alias1 ,table2name any_alias2 on any_alias1.columnname(+) = any_alias2.columnname;<br>OR<br>Select column1name,column2name from table1name left outer join table2name on table1name.columnname= table2name.columnname;<br>**Example:**<br>select last_name,department_name from employeese,departments d on e.department_id(+) = d.department_id; | **4M**<br>***2M for each join, any two joins can be considered*** |

| | | | |
|---|---|---|---|
| | | OR<br><br>select last_name,department_name from employees left outer join departments on employees.department_id = departments.department_id;<br><br>**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.<br>**Syntax:**<br>Select column1name,column2name from table1name any_alias1 ,table2name any_alias2 on any_alias1.columnname =any_alias2.columnname (+);<br>OR<br>Select column1name,column2name from table1name any_alias1 right outer join table2name any_alias2 on any_alias1.columnname =any_alias2.columnname;<br>**Example:**<br>Select last_name,department_name from employeese,departments d on e.department_id = d.department_id(+);<br>OR<br>Select last_name,department_name from employees e right outer join departments d on e.department_id = d.department_id;<br><br>**5) NON EQUI JOIN:** Non equi joins is used to return result from two or more tables where exact join is not possible.<br>**Syntax:**<br>Select aliasname.column1name, aliasname.column2name from tablename alias where ;<br>**Example:**we have 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)** | **Perform the following operations on table student**<br>**i) Create view Stud_view having marks greater than 80.**<br>**ii) Permanently delete Stud_view** | **4M** |

| | | | |
|---|---|---|---|
| | **Ans.** | i) CREATE VIEW Stud-view AS SELECT * FROM student WHERE marks >80;<br><br>(ii)DROP VIEW Stud-view | ***2M for each correct operation*** |
| | **d)**<br><br><br><br><br><br>**Ans.** | **Consider following schema**<br>**Employee (empid, ename, address, designation, salary)**<br>**Perform following operations on this schema**<br>**i) Add column city varchar (15)**<br>**ii) Change ename from 'Vijay' to 'Sachin'**<br>**iii) Display employees having salary more than 50000**<br>**iv) Delete record having ename as 'Sanjay'**<br>i) ALTER TABLE EmployeeADD city varchar(15);<br><br>(ii) Update Employee set ename='Sachin' where ename='Vijay';<br><br>(iii) select * from Employee where salary > 50000;<br><br>(iv)DELETE FROM Employee WHERE ename = 'Sanjay'; | **4M**<br><br><br><br><br><br>***1M for each correct syntax*** |
| **3.** | **a)**<br><br><br><br><br>**Ans.** | **Attempt any <u>THREE</u> of the following:**<br>**Create sequence with following specification**<br>**i) Name:- empid-seq**<br>**ii) Starting value:- 101**<br>**iii) Maximum value:- 1000**<br>**iv) Incremented by 1**<br>Create sequence empid-seq<br>start with 101,<br>increment by 1,<br>max value 1000; | **12**<br>**4M**<br><br><br><br><br>***4M for correct SQL statements*** |
| | **b)**<br>**Ans.** | **Describe ACID properties of transaction**<br>Transaction Properties is commonly known as ACID properties in order to ensure Accuracy, Consistency, Isolation & data integrity.<br>**1. Atomicity**: This property states that a transaction must be treated as an atomic unit i.e., single unit. Its operation is executed or none. There must be no state in database where a transaction is left partially | **4M**<br>***1M for each correct property*** |

| | | | |
|---|---|---|---|
| | | committed state should define either before the execution or after the executed/aborted/failure of transaction.<br><br>**2. Consistency**: The database must remain in a consistent state after any transaction state. No transaction should have any adverse effect on the data residing in the database. The database was in the consistent state and it should remain consistent after the execution of transaction as well.<br><br>**3. Isolation:** In database system, more than one transaction has been executed simultaneously and in parallel. The property of isolation states that all the transaction will affect and carried out or executed as if it is the only transaction in the system. No transaction will affect existence of any other transaction.<br><br>**4. Durability**: In database system, the database should durable enough to hold its latest updates, even if the system fails before the data could return all the disk. Then the data will be updated once the system spring back to the system. | |
| | **c) Ans.** | **Explain range searching operators with suitable example.**<br>The range searching operators are: **BETWEEN & NOT BETWEEN**<br>The SQL BETWEEN condition allowsto easily test if an expression is within a range of values (inclusive). The values can be text, date, or numbers. It can be used in a SELECT, INSERT, UPDATE, or DELETE statement.<br>Syntax:<br>SELECT column_name(s)<br>FROM table_name<br>WHERE column_name BETWEEN value1 AND value2;<br>**For example:**<br>**List all the Employee details who is having salary between 4000 and 6000.**<br>SELECT * from employee where salary between 4000 and 6000;<br>**Using NOT operator with BETWEEN**<br>**Find all the Employee details whose salary is not in the range of 4000 and 5000.**<br>Select * from where salary not between4000 and 6000; | **4M**<br>***2M for explanation***<br>***2M for example*** |

| | | | |
|---|---|---|---|
| | **d)**<br><br><br>**Ans.** | **Write SQL statements to create following indices on employee table Employee(empid, ename, address, designation, salary)**<br>**i) Create composite-index emp_addr with attributes address, city**<br>**ii) Create unique index emp_ung with attribute empid.**<br>(i)CREATE INDEX emp-addr ON Employee(address, city);<br>(ii) CREATE UNIQUE INDEX emp-ung ON Employee(empid); | **4M**<br><br><br>***2M for each correct SQL statements*** |
| **4.** | **a)**<br>**Ans.** | **Attempt any <u>THREE</u> of the following:**<br>**Draw and explain PL/SQL block structure.**<br>PL/SQL Block structure is as given below:<br><br>**DECLARE**<br>Declaration of variables<br>**BEGIN**<br>PL/SQL Statements<br>**Exceptions**<br>Error handling statements<br>**END**<br><br>Pl/SQL is a block structured language divided into a logical block.<br>The blockstructure follows divides & conquer approach to solve the problem step-wise,every block consists of 3 parts:<br>**1) Declaration Section**: This section starts with keyword 'declare'.It is an optional section & defines allvariables, cursors, sub-programs & other elements to be used in the program.<br>**2) Executable commands**: This section is enclosed between the keyword BEGIN & END. It is mandatorysection. It consists of executable PL/SQL Statements of the program.It shouldhave at least one executable line of code which may be just a NULL command toindicate that nothing should be executed.<br>**3)Exception Handling:**This section starts with keyword 'Exception'. It is again optional & containsexception that handles error in the programs. | **12**<br>**4M**<br><br><br>***2M for diagram***<br><br>***2M for explanation*** |

**SUMMER – 2022 EXAMINATION**
**MODEL ANSWER**

**Subject: Database Management** **Subject Code: 22416**

| | | | |
|---|---|---|---|
| | **b) Ans.** | **Define index. Explain types of indices with proper example.**<br>An index is a schema object.<br>• It is used by the oracle server to speed up the retrieval of rows by using pointer.<br>• Indexes are logically and physically independent of the table they index.<br>• These are used to maintain automatically by the oracle server.<br>• The user_indexes data dictionary view contains the name of the index and itsuniqueness.<br>**Types Of Indexes**<br>1. Simple Index<br>2. Composite Index<br>3.Unique Index - a) Simple Unique Index b) Composite Unique Index<br><br>➢**Simple Index: Based On a single column.**<br>Syntaax: CREATE INDEX <IndexName><br>ON <Table> (column_name);<br>Example: CREATE INDEX Emp_IndexON Employee (ename);<br><br>➢**Composite Index: Based on more than one column.**<br>Syntax: CREATE INDEX <IndexName><br>ON <Table> (column_names);<br>Example: CREATE INDEX Emp_IndexON Employee (ename, eid);<br><br>➢**Unique Index: A unique index does not allow any duplicate values to beinserted into the table.**<br>**1. Simple Unique Index:**<br>Syntax: CREATE UNIQUE INDEX <IndexName><br>ON <Table> (column_name);<br>Example: CREATE UNIQUE INDEX Emp_Index<br>ON Employee (ename);<br><br>**2. Composite Unique Index:**<br>Syntax: CREATE UNIQUE INDEX <IndexName><br>ON <Table> (column_names);<br>Example: CREATE UNIQUE INDEX Emp_Index<br>ON Employee (ename,eid); | **4M**<br>***1M for definition, 3M for each indices with example*** |

**SUMMER – 2022 EXAMINATION**
**MODEL ANSWER**

**Subject: Database Management** **Subject Code: 22416**

| | | | |
|---|---|---|---|
| | **c)** | **Consider schema**<br>**Employee (empid, ename, address, designation, salary)**<br>**Write SQL statements for following**<br>**i) List maximum and minimum salary**<br>**ii) Find ename of an employees who belongs to "Mumbai"**<br>**iii) Find total salary of all managers**<br>**iv)Find empid of all employees where name end with 'i'** | **4M** |
| | **Ans.** | **i) Select max(salary),min(salary) from Employee order by salary desc;**<br>(ii) Findename of an employees who belongs to 'Mumbai'.**(1mark for any correct SQL statement)**<br>Ans (ii) Select ename from employee where address like 'Mumbai';<br>**OR** Select ename from employee where address = 'Mumbai';<br>**OR** Select ename from employee where address IN ('Mumbai);<br>(iii) Find total salary of all managers.**(1mark for any correct SQL statement)**<br>Ans c(iii) Select sum(salary) from employee group by designation having designation= 'Manager';<br>**OR** Select sum(salary) from employee group by designation having designation Like 'Manager';<br>**OR** Select sum(salary) from employee where designation ='Manager';<br>**OR** Select sum(salary) from employee where designation like 'Manager';<br>(iv) Find empid of a1l employees where name ends with 'i'.**(1 mark for correct SQL statement)**<br>Ans c(iv) Select empid, ename from employee where ename like '%i'; | ***1M for correct SQL Statement*** |
| | **d)** | **Explain lock based concurrency control algorithm** | **4M** |
| | **Ans.** | The Lock based concurrency control is also called as Two-Phase protocol/concurrent control method.<br>Locking is an operation which secures: permission to read, OR permission to write a data item. Two phase locking is a process used to gain ownership of shared resources without creating the possibility of deadlock.<br>It consists of phases- | ***4M for correct explanation*** |

| | | | |
|---|---|---|---|
| | | **1.Growing Phase:** Transaction may obtain locks but may not be releasing any locks(No. oflocks can only increase)<br>**2.Shrinking phase:** A transaction may release lock but may not obtain new locks.<br>**This locking protocol divides execution phase of a transaction into 3 parts.**<br>1.In 1st part, when transaction is divided into parts. In case of Execution, it seekspermission for lock it requires.<br>2. The 2nd is when transaction acquires all locks .<br>3.As soon as the transaction releases its 1st lock, then the third phase starts. In this phase,transaction cannot demand any new lock. It only releases the acquired lock. | |
| | **e)**<br><br><br>**Ans.** | **Write PL/SQL code using user defined exception for following scenario.**<br>**If salary of employee is greater than 20000 after giving raise by 20% then raise exception stating "Salary too high"**<br>DECLARE<br>E_SALEMP.EMPsalary%TYPE;<br>E_NAME EMP.ENAME%TYPE;<br>EX_HIGH_SAL EXCEPTION;<br>BEGIN<br>UPDATE Emp<br>SET EMPsalary=EMPsalary+EMPsalary*0.2;<br>Select EMPsalary into E_SAL from emp where EMPsalary>20000;<br>IF E_SAL>20000 THEN<br>RAISE EX_INVALID_ID;<br>ELSE<br>SELECT ENAME INTO E_NAME FROM EMP<br>WHEREEMPsalary=E_SAL;<br>DBMS_OUTPUT.PUT_LINE(E_NAME\|\|'ARE THEEMPLOYEES whose salary are updated');<br>END IF;<br>EXCEPTION<br>WHEN EX_HIGH_SAL THEN<br>DBMS_OUTPUT.PUT_LINE('Salary is too high');<br>WHEN NO_DATA_FOUND THEN<br>DBMS_OUTPUT.PUT_LINE('DATA NOT FOUND');<br>END; | **4M**<br><br><br>***4M for appropriate logic*** |

| | | | |
|---|---|---|---|
| **5.** | | **Attempt any TWO of the following** | **12** |
| | **a)** | **Create employee table with following Integrity constraints.**<br>**employee (empid, ename, phone, dob, addr, designation salary, deptno)**<br>**(i) empid as primary key**<br>**(ii) Phone as unique**<br>**(iii) deptno as not null**<br>**Also create dept table as dept (deptno, dname, totalemp) wheredeptno as primary key and totalemp with check constraint as totalemp> 10**<br>**(query with correct integrity constraints applied : 3M each)**<br>***(Note :Datatypes specification syntaxes may differ according to DBM Environment, so any other relevant syntax to apply integrity constraints can be considered)*** | **6M**<br>**3M for each**<br>***Query with correct integrity constraints applied*** |
| | **Ans.** | create table employee<br>(empidint primary key,<br>enamevarchar(20),<br>phone int unique ,<br>dob date,<br>addrvarchar(50),<br>designation varchar(20),<br>salary int,<br>deptnoint not null)<br><br>create table dept<br>(deptnoint primary key,<br>dnamevarchar(30),<br>totalempintcheck(totalemp>10)) | |
| | **b)** | **Sailor (Sid, sname, rating, age)**<br>**Boat (Bid, bname, color)**<br>**Reserve (sid, Bid, rdate)**<br>**Consider above schemas and write SQL statement for following.**<br>**i) Display average age of sailor**<br>**ii) Display name of boat reserved on date 12.12.2018**<br>**iii) List details of boats having some color as "interlake"**<br>**iv) Apply equijoin on sailor and reserve.**<br>**v) Display information of all employees having rating less than 5 and greater than 8.** | **6M** |

| | | | |
|---|---|---|---|
| | **Ans.** | **vi) List all sailors having name consist 5 letters only**<br>**i) Display average age of sailor.**<br>Select avg(age) from Sailor;<br><br>**ii) Display name of boat reserved on date 12.12.2018**<br>Select bname from Boat where Boat.Bid=Reserve.Bid where rdate='12.12.2018';<br><br>**iii) List details of boats having some color as "interlake".**<br>Select * from Boat where color ='interlake';<br><br>**iv) Apply equijoin on sailor and reserve.**<br>SELECT Sailor.Sid,<br>Sailor.sname,Sailor.rating,Sailor.age,Reserve.Bid,<br>Reserve.rdate<br>FROM Sailor,ReserveWHERE Sailor.Sid = Sailor.Sid;<br>**Or**<br>SELECT Sailor.Sid, Sailor.sname,<br>Sailor.rating,Sailor.age,Reserve.Bid, Reserve.rdate<br>FROM Sailor JOIN Reserve WHERE Sailor.Sid = Sailor.Sid;<br><br>**(v) Display information of all employees having rating less than 5 and greater than 8.**<br>Select * from Sailor where rating <5 or rating >8<br><br>**(vi) List all sailors having name consist of 5 letters only.**<br>Select * from Sailor where sname like '_____';<br>(underscore used 5 times in the pattern) | ***1M for each Query with correct syntax and logic*** |
| | **c)**<br><br>**Ans** | **Write PL/SQL program to display odd numbers between I to 50.**<br>DECLARE<br>I number;<br>BEGIN<br>I:=1;<br>Loop<br>Dbms_output.put_line(I);<br>I:=I+1;<br>Exit when I>50;<br>End loop;<br>END; | **6M**<br><br>***3M for Correct logic***<br>***3M for correct syntax*** |

**SUMMER – 2022 EXAMINATION**
**MODEL ANSWER**

**Subject: Database Management** **Subject Code:** **22416**

| | | | |
|---|---|---|---|
| **6.** | | **Attempt any TWO of the following:** | **12** |
| | **a)** | **Write SQL statements for following.**<br>**(i) Create user 'admin' with passward '123**<br>**(ii) Give user admin full access to employee table.**<br>**(iii)Remove delete privileges from admin.** | **6M** |
| | **Ans.** | **(i) Create user 'admin' with passward '123**<br>CREATE USER 'admin' IDENTIFIED BY '123';<br><br>**(ii) Give user admin full access to employee table.**<br>GRANT ALL ON employee TO admin;<br><br>**(iii)Remove delete privileges from admin.**<br>REVOKE DELETE ON employee from admin; | ***2M for each, query with correct syntax and logic*** |
| | **b)** | **Create curser emp-copy to select all records from employee table and copy them into employee2 table.** | **6M** |
| | **Ans.** | **(Assuming fields for employee table as empid,empname, deptno, designation)**<br>DECLARE<br>empid1 employee.empid%TYPE;<br>empname1employee.empname%TYPE;<br>deptno1employee.deptno%TYPE;<br>designation1 employee.designation%TYPE;<br><br>CURSOR emp_copy IS<br>SELECT empid,empname, deptno,designation FROM employee;<br>BEGIN<br>OPEN emp_copy;<br>LOOP<br>FETCH emp_copy INTO empid1,empname1,deptno1,designation1;<br>EXIT WHEN emp_copy%NOTFOUND;<br>INSERT INTO employee2<br>VALUES (empid1,empname1,deptno1, designation1);<br>END LOOP;<br>CLOSE emp_copy;<br>COMMIT;<br>END; | ***3M for correct syntax***<br>***3M for correct logic*** |

| | | | |
|---|---|---|---|
| | **c)** | **Write SQL statement for following:**<br>**(i) Create view on deposit (Accno, cname, amount) where amount is greater than 5000.**<br>**(ii) Create Synonym empdup for employee.**<br>**(iii) Drop Synonym created on employee table.** | **6M** |
| | **Ans.** | **(i) Create view on deposit (Accno, cname, amount) where amount is greater than 5000.**<br>Create view v1 as select * from deposit where amount >5000;<br><br>**(ii) Create Synonym empdup for employee.**<br>Create Synonym empdupFor employee;<br><br>**(iii) Drop Synonym created on employee table.**<br>Drop synonym empdup; | ***2M for each correct syntax and logic of Query*** |