DATABASE MANAGEMENT SYSTEMS

LAB MANUAL

Year	**:**	**2018 - 2019**
Course Code	**:**	ACS104
Regulations	**:**	**IARE - R16**
Semester	**:**	**III**
Branch	**:**	**INFORMATION TECHNOLOGY**

Prepared By

Ms. K Laxmi Narayanamma

Assistant Professor

Mr.N Bhaswanth

Assistant Professor

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad.

COMPUTER SCIENCE AND ENGINEERING

1. PROGRAM OUTCOMES:

B.TECH - PROGRAM OUTCOMES (POS)	
PO-1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO-2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO-3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations .
PO-4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO-5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations .
PO-6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice**.**
PO-7	**Environment and sustainability:** Understand the impact of the professional engineering solution sin societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development .
PO-8	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice **(Ethics)**.
PO-9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO-10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions .
PO-11	**Project management and finance** :Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments .
PO-12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in

B.TECH - PROGRAM OUTCOMES (POS)	
	independent and life-long learning in the broadest context of technological change**.**

2. PROGRAM SPECIFIC OUTCOMES:

PROGRAM SPECIFIC OUTCOMES (PSO's)	
PSO-1	**Professional Skills:** The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient design of computer-based systems of varying complexity.
PSO-2	**Software Engineering Practices:** The ability to apply standard practices and strategies in software service management using open-ended programming environments with agility to deliver a quality service for business success.
PSO-3	**Successful Career and Entrepreneurship:** The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies.

3. ATTAINMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES

S.No	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
WEEK-1	Creation of Tables	PO3,PO5	PSO1,PSO2
WEEK-2	Queries using DDL and DML.	PO1,PO3,PO5	PSO1,PSO3
WEEK-3	Queries using aggregate functions.	PO1,PO2,PO5	PSO2,PSO3
WEEK-4	Programs On Pl/Sql	PO3,PO4,PO5,PO6	PSO2,PSO3
WEEK-5	Procedures And Functions	PO3,PO4,PO5	PSO2,PSO3
WEEK-6	Triggers	PO3,PO4,PO5	PSO2,PSO3
WEEK-7	Procedures	PO3,PO4,PO5	PSO2,PSO3
WEEK-8	Cursors	PO1,PO2,PO3	PSO1,PSO2,PSO3
WEEK-9	Case Study: Book Publishing Company	PO2,PO3,PO4,PO5	PSO1,PSO2,PSO3
WEEK-10	Case Study: General Hospital	PO2,PO3,PO4,PO5	PSO1,PSO2,PSO3
WEEK-11	Case Study: Car Rental Company	PO2,PO3,PO4,PO5	PSO1,PSO2,PSO3
WEEK-12	Case Study: Student Progress Monitoring System	PO2,PO3,PO4,PO5	PSO1,PSO2,PSO3

4. MAPPING COURSE OBJECTIVES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

Course Objectives	Program Outcomes												Program Specific Outcomes		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO1 1	PO1 2	PSO1	PSO2	PSO3
I	√		√		√								√	√	
II	√	√			√								√		√
III	√	√			√									√	√
IV	√		√		√	√								√	√
V			√	√	√								√	√	√
VI	√	√	√										√	√	√
VII√	√	√	√										√	√	√

5. SYLLABUS:

DATABASE MANAGEMENT SYSTEMS LABORATORY								
IV Semester: IT								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
		L	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
ACS104	**Core**	-	-	3	2	30	70	100
Contact Classes: Nil	**Tutorial Classes: Nil**	**Practical Classes: 36**				**Total Classes: 36**		

OBJECTIVES:

The course should enable the students to:

I. Implement the basic knowledge of SQL queries and relational algebra.
II. Construct database models for different database applications.
III. Apply normalization techniques for refining of databases.
IV. Practice various triggers, procedures, and cursors using PL/SQL.

LIST OF EXPERIMENTS

WEEK-1 **CREATION OF TABLES**

1. Create a table called Employee with the following structure.

Name	Type
Empno	Number
Ename	Varchar2(20)
Job	Varchar2(20)
Mgr	Number
Sal	Number

a. Add a column commission with domain to the Employee table.
b. Insert any five records into the table.
c. Update the column details of job
d. Rename the column of Employ table using alter command.
e. Delete the employee whose empno is 19.

2. Create department table with the following structure.

Name	Type
Deptno	Number
Deptname	Varchar2(20)
location	Varchar2(20)

a. Add column designation to the department table.
b. Insert values into the table.
c. List the records of emp table grouped by deptno.
d. Update the record where deptno is 9.
e. Delete any column data from the table.

3. Create a table called Customer table

Name	Type
Cust name	Varchar2(20)
Cust street	Varchar2(20)
Cust city	Varchar2(20)

a. Insert records into the table.
b. Add salary column to the table.
c. Alter the table column domain.
d. Drop salary column of the customer table.
e. Delete the rows of customer table whose cust_city is 'hyd'.

4. Create a table called branch table.

Name	Type
Branch name	Varchar2(20)
Branch city	Varchar2(20)
asserts	Number

a. Increase the size of data type for asserts to the branch.
b. Add and drop a column to the branch table.
c. Insert values to the table.
d. Update the branch name column
e. Delete any two columns from the table

5. Create a table called sailor table

Name	Type
Sid	Number
Sname	Varchar2(20)
rating	Varchar2(20)

a. Add column age to the sailor table.
b. Insert values into the sailor table.
c. Delete the row with rating >8.
d. Update the column details of sailor.
e. Insert null values into the table.

6. Create a table called reserves table

Name	Type
Boat id	Integer
sid	Integer
day	Integer

a. Insert values into the reserves table.
b. Add column time to the reserves table.
c. Alter the column day data type to date.
d. Drop the column time in the table.
e. Delete the row of the table with some condition.

WEEK -2 QUERIES USING DDL AND DML

1. a. Create a user and grant all permissions to the user.
   b. Insert the any three records in the employee table and use rollback. Check the result.
   c. Add primary key constraint and not null constraint to the employee table.
   d. Insert null values to the employee table and verify the result.
2. a. Create a user and grant all permissions to the user.
   b. Insert values in the department table and use commit.
   c. Add constraints like unique and not null to the department table.
   d. Insert repeated values and null values into the table.
3. a. Create a user and grant all permissions to the user.
   b. Insert values into the table and use commit.

c. Delete any three records in the department table and use rollback.
d. Add constraint primary key and foreign key to the table.

4. a. Create a user and grant all permissions to the user.
b. Insert records in the sailor table and use commit.
c. Add save point after insertion of records and verify save point.
d. Add constraints not null and primary key to the sailor table.

5. a. Create a user and grant all permissions to the user.
b. Use revoke command to remove user permissions.
c. Change password of the user created.
d. Add constraint foreign key and not null.

6. a. Create a user and grant all permissions to the user.
b. Update the table reserves and use savepoint and rollback.
c. Add constraint primary key , foreign key and not null to the reserves table
d. Delete constraint not null to the table column.

WEEK -3 QUERIES USING AGGREGATE FUNCTIONS

1. a. By using the group by clause, display the enames who belongs to deptno 10 along with average salary.
b. Display lowest paid employee details under each department.
c. Display number of employees working in each department and their department number.
d. Using built in functions, display number of employees working in each department and their department name from dept table. Insert deptname to dept table and insert deptname for each row, do the required thing specified above.
e. List all employees which start with either B or C.
f. Display only these ename of employees where the maximum salary is greater than or equal to 5000.

2. a. Calculate the average salary for each different job.
b. Show the average salary of each job excluding manager.
c. Show the average salary for all departments employing more than three people.
d. Display employees who earn more than the lowest salary in department 30
e. Show that value returned by sign (n) function.
f. How many days between day of birth to current date.

3. a. Show that two substring as single string.
b. List all employee names, salary and 15% rise in salary.
c. Display lowest paid emp details under each manager
d. Display the average monthly salary bill for each deptno.
e. Show the average salary for all departments employing more than two people.
f. By using the group by clause, display the eid who belongs to deptno 05 along with average salary.

4. a. Count the number of employees in department 20
b. Find the minimum salary earned by clerk.
c. Find minimum, maximum, average salary of all employees.
d. List the minimum and maximum salaries for each job type.
e. List the employee names in descending order.
f. List the employee id, names in ascending order by empid.

5. a. Find the sids ,names of sailors who have reserved all boats called “INTERLAKE
Find the age of youngest sailor who is eligible to vote for each rating level with at least two such sailors.
b. Find the sname , bid and reservation date for each reservation.
c. Find the ages of sailors whose name begin and end with B and has at least 3 characters.
d. List in alphabetic order all sailors who have reserved red boat.
e. Find the age of youngest sailor for each rating level.

6. a. List the Vendors who have delivered products within 6 months from order date.
b. Display the Vendor details who have supplied both Assembled and Sub parts.
c. Display the Sub parts by grouping the Vendor type (Local or Non Local).
d. Display the Vendor details in ascending order.
e. Display the Sub part which costs more than any of the Assembled parts.
f. Display the second maximum cost Assembled part.

WEEK - 4 PROGRAMS ON PL/SQL

1. a. Write a PL/SQL program to swap two numbers.
   b. Write a PL/SQL program to find the largest of three numbers.
2. a. Write a PL/SQL program to find the total and average of 6 subjects and display the grade.
   b. Write a PL/SQL program to find the sum of digits in a given number.
3. a. Write a PL/SQL program to display the number in reverse order.
   b. Write a PL / SQL program to check whether the given number is prime or not.
4. a. Write a PL/SQL program to find the factorial of a given number.
   b. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns radius and area.
5. a. Write a PL/SQL program to accept a string and remove the vowels from the string. (When 'hello' passed to the program it should display 'Hll' removing e and o from the world Hello).
   b. Write a PL/SQL program to accept a number and a divisor. Make sure the divisor is less than or equal to 10. Else display an error message. Otherwise Display the remainder in words.

WEEK -5 PROCEDURES AND FUNCTIONS

1. Write a function to accept employee number as parameter and return Basic +HRA together as single column.
2. Accept year as parameter and write a Function to return the total net salary spent for a given year.
3. Create a function to find the factorial of a given number and hence find NCR.
4. Write a PL/SQL block o pint prime Fibonacci series using local functions.
5. Create a procedure to find the lucky number of a given birth date.
6. Create function to the reverse of given number.

WEEK-6 TRIGGERS

1. Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

   CUSTOMERS table:

ID	NAME	AGE	ADDRESS	SALARY
1	Alive	24	Khammam	2000
2	Bob	27	Kadappa	3000
3	Catri	25	Guntur	4000
4	Dena	28	Hyderabad	5000
5	Eeshwar	27	Kurnool	6000
6	Farooq	28	Nellur	7000

2. Creation of insert trigger, delete trigger, update trigger practice triggers using the passenger database. Passenger(Passport_ id INTEGER PRIMARY KEY, Name VARCHAR (50) Not NULL, Age Integer Not NULL, Sex Char, Address VARCHAR (50) Not NULL);
   a. Write a Insert Trigger to check the Passport_id is exactly six digits or not.
   b. Write a trigger on passenger to display messages '1 Record is inserted', '1 record is deleted', '1 record is updated' when insertion, deletion and updation are done on passenger respectively.
3. Insert row in employee table using Triggers. Every trigger is created with name any trigger have same name must be replaced by new name. These triggers can raised before insert, update or delete rows on data base. The main difference between a trigger and a stored procedure is that the former is attached to a table and is only fired when an INSERT, UPDATE or DELETE occurs.
4. Convert employee name into uppercase whenever an employee record is inserted or updated. Trigger

to fire before the insert or update.

5. Trigger before deleting a record from emp table. Trigger will insert the row to be deleted into table called delete _emp and also record user who has deleted the record and date and time of delete.
6. Create a transparent audit system for a table CUST_MSTR. The system must keep track of the records that are being deleted or updated.

WEEK-7	PROCEDURES

1. Create the procedure for palindrome of given number.
2. Create the procedure for GCD: Program should load two registers with two Numbers and then apply the logic for GCD of two numbers. GCD of two numbers is performed by dividing the greater number by the smaller number till the remainder is zero. If it is zero, the divisor is the GCD if not the remainder and the divisors of the previous division are the new set of two numbers. The process is repeated by dividing greater of the two numbers by the smaller number till the remainder is zero and GCD is found.
3. Write the PL/SQL programs to create the procedure for factorial of given number.
4. Write the PL/SQL programs to create the procedure to find sum of N natural number.
5. Write the PL/SQL programs to create the procedure to find Fibonacci series.
6. Write the PL/SQL programs to create the procedure to check the given number is perfect or not.

WEEK-8	CURSORS

1. Write a PL/SQL block that will display the name, dept no, salary of fist highest paid employees**.**
2. Update the balance stock in the item master table each time a transaction takes place in the item transaction table. The change in item master table depends on the item id is already present in the item master then update operation is performed to decrease the balance stock by the quantity specified in the item transaction in case the item id is not present in the item master table then the record is inserted in the item master table.
3. Write a PL/SQL block that will display the employee details along with salary using cursors.
4. To write a Cursor to display the list of employees who are working as a Managers or Analyst.
5. To write a Cursor to find employee with given job and deptno.
6. Write a PL/SQL block using implicit cursor that will display message, the salaries of all the employees in the ‘employee’ table are updated. If none of the employee’s salary are updated we get a message 'None of the salaries were updated'. Else we get a message like for example, 'Salaries for 1000 employees are updated' if there are 1000 rows in ‘employee’ table.

WEEK-9	CASE STUDY: BOOK PUBLISHING COMPANY

A publishing company produces scientific books on various subjects. The books are written by authors who specialize in one particular subject. The company employs editors who, not necessarily being specialists in a particular area, each take sole responsibility for editing one or more publications.

A publication covers essentially one of the specialist subjects and is normally written by a single author. When writing a particular book, each author works with on editor, but may submit another work for publication to be supervised by other editors. To improve their competitiveness, the company tries to employ a variety of authors, more than one author being a specialist in a particular subject for the above case study, do the following:

1. Analyze the data required.
2. Normalize the attributes.

Create the logical data model using E-R diagrams.

WEEK -10	CASE STUDY GENERAL HOSPITAL

A General Hospital consists of a number of specialized wards (such as Maternity, Pediatric, Oncology, etc). Each ward hosts a number of patients, who were admitted on the recommendation of their own GP and confirmed by a consultant employed by the Hospital. On admission, the personal details of every patient are recorded. A separate register is to be held to store the information of the tests undertaken and the results of a prescribed treatment. A number of tests may be conducted for each patient. Each patient is assigned to one leading consultant but may be examined by another doctor, if required. Doctors are specialists in some branch of medicine and may be leading consultants for a number of patients, not necessarily from the same ward. For the above case study, do the following.

1. Analyze the data required.

2. Normalize the attributes.
   Create the logical data model using E-R diagrams.

WEEK -11	CASE STUDY: CAR RENTAL COMPANY

A database is to be designed for a car rental company. The information required includes a description of cars, subcontractors (i.e. garages), company expenditures, company revenues and customers. Cars are to be described by such data as: make, model, year of production, engine size, fuel type, number of passengers, registration number, purchase price, purchase date, rent price and insurance details. It is the company policy not to keep any car for a period exceeding one year. All major repairs and maintenance are done by subcontractors (i.e. franchised garages), with whom CRC has long-term agreements. Therefore the data about garages to be kept in the database includes garage names, addresses, range of services and the like. Some garages require payments immediately after a repair has been made; with others CRC has made arrangements for credit facilities. Company expenditures are to be registered for all outgoings connected with purchases, repairs, maintenance, insurance etc. Similarly the cash inflow coming from all sources: Car hire, car sales, insurance claims must be kept of file. CRC maintains a reasonably stable client base. For this privileged category of customers special credit card facilities are provided. These customers may also book in advance a particular car. These reservations can be made for any period of time up to one month. Casual customers must pay a deposit for an estimated time of rental, unless they wish to pay by credit card. All major credit cards are accepted. Personal details such as name, address, telephone number, driving license, number about each customer are kept in the database. For the above case study, do the following:

1. Analyze the data required.
2. Normalize the attributes.
   Create the logical data model using E-R diagrams.

WEEK-12	CASE STUDY: STUDENT PROGRESS MONITORING SYSTEM

A database is to be designed for a college to monitor students' progress throughout their course of study. The students are reading for a degree (such as BA, BA (Hons) M.Sc., etc) within the framework of the modular system. The college provides a number of modules, each being characterized by its code, title, credit value, module leader, teaching staff and the department they come from. A module is coordinated by a module leader who shares teaching duties with one or more lecturers. A lecturer may teach (and be a module leader for) more than one module. Students are free to choose any module they wish but the following rules must be observed: Some modules require pre- requisites modules and some degree programmes have compulsory modules. The database is also to contain some information about students including their numbers, names, addresses, degrees they read for, and their past performance i.e. modules taken and examination results. For the above case study, do the following:

1. Analyze the data required.
2. Normalize the attributes.
3. Create the logical data model i.e., ER diagrams.
4. Comprehend the data given in the case study by creating respective tables with primary keys and foreign keys wherever required.
5. Insert values into the tables created (Be vigilant about Master- Slave tables).
6. Display the Students who have taken M.Sc course.
7. Display the Module code and Number of Modules taught by each Lecturer.
8. Retrieve the Lecturer names who are not Module Leaders.
9. Display the Department name which offers ‘English’ module.
10. Retrieve the Prerequisite Courses offered by every Department (with Department names).
11. Present the Lecturer ID and Name who teaches ‘Mathematics’.
12. Discover the number of years a Module is taught.
13. List out all the Faculties who work for ‘Statistics’ Department.
14. List out the number of Modules taught by each Module Leader.
15. List out the number of Modules taught by a particular Lecturer.
16. Create a view which contains the fields of both Department and Module tables. (Hint- The fields like Module code, title, credit, Department code and its name).
17. Update the credits of all the prerequisite courses to 5. Delete the Module ‘History’ from the Module table.

Reference Books:

1. Ramez Elmasri, Shamkant, B. Navathe, "Database Systems", Pearson Education, 6th Edition, 2013.
2. Peter Rob, Carles Coronel, "Database System Concepts", Cengage Learning, 7th Edition, 2008.
3. M L Gillenson, "Introduction to Database Management", Wiley Student Edition, 2012.

SOFTWARE AND HARDWARE REQUIREMENTS FOR A BATCH OF 36 STUDENTS:

HARDWARE:

Desktop Computer Systems: 36 nos

SOFTWARE:

Application Software: Oracle RDBMS

6. INDEX

S.No	List of Experiments	Page No
1	INTRODUCTION TO ORACLE	1
2	CREATION OF TABLES	5
3	QUERIES USING DDL AND DML	12
4	QUERIES USING AGGREGATE FUNCTIONS	17
5	PROGRAMS ON PL/SQL	21
6	PROCEDURES AND FUNCTIONS	25
7	TRIGGERS	28
8	PROCEDURES	33
9	CURSORS	34
10	CASE STUDY: BOOK PUBLISHING COMPANY	35
11	CASE STUDY: GENERAL HOSPITAL	36
12	CASE STUDY: CAR RENTAL COMPANY	37
13	CASE STUDY: STUDENT PROGRESS MONITORING SYSTEM	38
14	VIVA QUESTIONS AND ANSWERS	39

INTRODUCTION TO ORACLE

1. DDL: Data Definition Language (DDL) statements are used to define the database structure or schema.

DDL Commands: Create, Alter, Drop, Rename, Truncate

CREATE - to create objects in the database

ALTER - alters the structure of the database

DROP - delete objects from the database

TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed

RENAME - rename an object

2. DML: Data Manipulation Language (DML) statements are used for managing data within schema objects and to manipulate data of a database objects.

DML Commands: Insert, Update, Delete, Select

INSERT - insert data into a table

UPDATE - updates existing data within a table

DELETE - deletes all records from a table, the space for the records remain

SELECT - retrieve data from the a database

3. DCL: Data Control Language (DCL) statements are used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it. To control the data of a database.

DCL Commands: Grant, Revoke

GRANT - gives user's access privileges to database

REVOKE -withdraw access privileges given with the GRANT command

4. TCL: Transaction Control (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

TCL Commands: Commit, Rollback, Save point

COMMIT - save work done

SAVEPOINT - identify a point in a transaction to which you can later roll back

ROLLBACK - restore database to original since the last COMMIT

Syntax with examples

1. DDL (Data Definition Language) Commands: CREATE, ALTER and DROP.

CREATE: This command useful for creating creating table.

Syntax:

create table [table name] (column1 datatype[size], column 2 datatype[size],… column n datatype[size]);

Ex:

SQL >create table student (s_rollno number(10) primary key,s_name varchar2(10), gender varchar2(5),dob date,addr1 varchar2(10),addr2 varchar2(10),city varchar2(10), percentage number(4));

SQL> DESC STUDENT;

Name	Null?	Type
S_ROLLNO	NOT NULL	NUMBER(10)
S_NAME		VARCHAR2(10)
GENDER		VARCHAR2(5)
DOB		DATE
ADDR1		VARCHAR2(10)
ADDR2		VARCHAR2(10)
CITY		VARCHAR2(10)
PERCENTAGE		NUMBER(4)

SQL > select s_rollno,s_name from student;

no rows selected.

Create table by using Constraints:

Constraints are two types:

1. Table Level Constraints.
2. Column Level Constraints.

1. NOT NULL:

a) ***Not null constraint at column level.***

Syntax:

<col><datatype>(size)not null

SQL > create table emp(e_id varchar(5) NOT NULL,e_name varchar(10), e_design varchar(10),dept varchar(10),mgr varchar(10),salary number(10));

2. UNIQUE :

Unique constraint at column level.

Syntax: <col><datatype>(size)unique

Ex:-

SQL > create table depositor(customer_name varchar(10),acc_no number(15) UNIQUE, brach_name varchar(10));

Unique constraint at table level:

Syntax:

Create table tablename(col=format,col=format,unique(<col1>,<col2>));

Ex:-

SQL > create table depositor1(customer_name varchar(10),acc_no number(15), brach_name varchar(10),UNIQUE(acc_no));

3. PRIMARY KEY:

Primary key constraint at column level

Syntax:

<col><datatype>(size)primary key;

Ex:-

SQL> create table customer(customer_id number (5) PRIMARY KEY, customer_name varchar(10),customer_street varchar(10),brach_name varchar(10));

Primary key constraint at table level.

Syntax:

Create table tablename(col=format,col=format primary key(col1>,<col2>);

Ex:-

SQL > create table customer1(customer_id number (5),customer_name varchar(10),customer_street varchar(10),brach_name varchar(10),PRIMARY KEY(customer_id));

4. CHECK:

Check constraint constraint at column level.

Syntax: <col><datatype>(size) check(<logical expression>)

Ex:-create table loan(loan_no varchar(10),customer_name varchar(10), balance number (10) CHECK(balance>1000));

Check constraint constraint at table level.

Syntax: check(<logical expression>)

Ex:- create table loan1(loan_no varchar(10),customer_name varchar(10), balance number (10), CHECK(balance>1000));

5. FOREIGN KEY:

Foreign key constraint at column level.

Syntax:

Column_name Datatype(size) REFERENCES parent_table_name (parent_column_name)

Ex:- CREATE TABLE books (book_id NUMBER(3), book_title VARCHAR2(30), book_price

NUMBER(3), book_author_id NUMBER(3) REFERENCES author(author_id));

Foreign key constraint at table level

Syntax:

CONSTRAINT constraint_name FOREIGN KEY(child_table_column) REFERENCES Parent_table_name(parent_table_column)

Ex:-CREATE TABLE books (book_id NUMBER(3) CONSTRAINT bok_bi_pk PRIMARY KEY, book_title VARCHAR2(30), book_price NUMBER(3), book_author_id NUMBER(3),CONSTRAINT bok_ai_fk FOREIGN KEY (book_author_id) REFERENCES author(author_id));

WEEK-1
CREATION OF TABLES

1) Create a table called Employee with the following structure.

Name	Type
Empno	Number
Ename	Varchar2(10)
Job	Varchar2(10)
Mgr	Number
Sal	Number

a. Add a column commission with domain to the Employee table.
b. Insert any five records into the table.
c. Update the column details of job
d. Rename the column of Employ table using alter command.
e. Delete the employee whose Empno is 105.

SOLUTION:

```
SQL> create table employee(empno number,ename varchar2(10),job varchar2(10),mgr
number,sal number);
Table created.
SQL> desc employee;
 Name                      Null?    Type
 ------------------------- -------- ----------------------------
 EMPNO                              NUMBER
 ENAME                              VARCHAR2(10)
 JOB                                VARCHAR2(10)
 MGR                                NUMBER
 SAL                                NUMBER
```

a. Add a column commission with domain to the Employee table.

```
SQL> alter table employee add(commission number);
Table altered.
SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NUMBER
 ENAME                                     VARCHAR2(10)
 JOB                                       VARCHAR2(10)
 MGR                                       NUMBER
 SAL                                       NUMBER
 COMMISSION                                NUMBER
```

b. Insert any five records into the table.

```
SQL> insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission');
Enter value for empno: 101
Enter value for ename: abhi
Enter value for job: manager
```

```
Enter value for mgr: 1234
Enter value for sal: 10000
Enter value for commission: 70
old   1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission')
new   1: insert into employee values(101,'abhi','manager',1234,10000,'70')
1 row created.

SQL> /
Enter value for empno: 102
Enter value for ename: rohith
Enter value for job: analyst
Enter value for mgr: 2345
Enter value for sal: 9000
Enter value for commission: 65
old   1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission')
new   1: insert into employee values(102,'rohith','analyst',2345,9000,'65')
1 row created.
SQL> /
Enter value for empno: 103
Enter value for ename: david
Enter value for job: analyst
Enter value for mgr: 3456
Enter value for sal: 9000
Enter value for commission: 65
old   1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission')
new   1: insert into employee values(103,'david','analyst',3456,9000,'65')
1 row created.

SQL> /
Enter value for empno: 104
Enter value for ename: rahul
Enter value for job: clerk
Enter value for mgr: 4567
Enter value for sal: 7000
Enter value for commission: 55
old   1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission')
new   1: insert into employee values(104,'rahul','clerk',4567,7000,'55')
1 row created.

SQL> /
Enter value for empno: 105
Enter value for ename: pramod
Enter value for job: salesman
Enter value for mgr: 5678
Enter value for sal: 5000
Enter value for commission: 50
old   1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission')
new   1: insert into employee values(105,'pramod','salesman',5678,5000,'50')
1 row created.

SQL> select * from employee;
```

```
    EMPNO ENAME      JOB              MGR       SAL COMMISSION
---------- ---------- ---------- ---------- ---------- ----------
       101 abhi       manager          1234      10000         70
       102 rohith     analyst          2345       9000         65
       103 david      analyst          3456       9000         65
       104 rahul      clerk            4567       7000         55
       105 pramod     salesman         5678       5000         50
```

c. Update the column details of job

```
SQL> update employee set job='trainee'  where empno=103;
1 row updated.

SQL> select * from employee;
    EMPNO ENAME      JOB              MGR       SAL COMMISSION
---------- ---------- ---------- ---------- ---------- ----------
       101 abhi       manager          1234      10000         70
       102 rohith     analyst          2345       9000         65
       103 david      trainee          3456       9000         65
       104 rahul      clerk            4567       7000         55
       105 pramod     salesman         5678       5000         50
```

d. Rename the column of Employ table using alter command.

```
SQL> alter table employee rename column mgr to manager_no;

Table altered.

SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(10)
 MANAGER_NO                                         NUMBER
 SAL                                                NUMBER
 COMMISSION                                         NUMBER
```

e. Delete the employee whose Empno is 105.

```
SQL> delete employee where empno=105;
1 row deleted.

SQL> select * from employee;
    EMPNO ENAME      JOB        MANAGER_NO       SAL COMMISSION
---------- ---------- ---------- ---------- ---------- ----------
       101 abhi       manager          1234      10000         70
       102 rohith     analyst          2345       9000         65
       103 david      trainee          3456       9000         65
       104 rahul      clerk            4567       7000         55
```

2) Create department table with the following structure.

Name	Type
Deptno	Number
Deptname	Varchar2(10)
location	Varchar2(10)

a. Add column designation to the department table.
b. Insert values into the table.
c. List the records of dept table grouped by deptno.
d. Update the record where deptno is 9.
e. Delete any column data from the table.

SOLUTION:

```
SQL> create table department(deptno number,deptname varchar2(10),location varchar2(10));
Table created.

SQL> desc department;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER
 DEPTNAME                                           VARCHAR2(10)
 LOCATION                                           VARCHAR2(10)
```

a. Add column designation to the department table.

```
SQL> alter table department add(designation varchar2(10));
Table altered.

SQL> desc department;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER
 DEPTNAME                                           VARCHAR2(10)
 LOCATION                                           VARCHAR2(10)
 DESIGNATION                                        VARCHAR2(10)
```

b. Insert values into the table.

```
SQL> insert into department values(&deptno,'&deptname','&location','&designation');
Enter value for deptno: 9
Enter value for deptname: accounting
Enter value for location: hyderabad
Enter value for designation: manager
old   1: insert into department values(&deptno,'&deptname','&location','&designation')
new   1: insert into department values(9,'accounting','hyderabad','manager')

1 row created.

SQL> /
```

```
Enter value for deptno: 10
Enter value for deptname: research
Enter value for location: chennai
Enter value for designation: professor
old   1: insert into department values(&deptno,'&deptname','&location','&designation')
new   1: insert into department values(10,'research','chennai','professor')

1 row created.
SQL> /
Enter value for deptno: 11
Enter value for deptname: sales
Enter value for location: banglore
Enter value for designation: salesman
old   1: insert into department values(&deptno,'&deptname','&location','&designation')
new   1: insert into department values(11,'sales','banglore','salesman')
1 row created.
SQL> /
Enter value for deptno: 12
Enter value for deptname: operations
Enter value for location: mumbai
Enter value for designation: operator
old   1: insert into department values(&deptno,'&deptname','&location','&designation')
new   1: insert into department values(12,'operations','mumbai','operator')
1 row created.

SQL> insert into department values(&deptno,'&deptname','&location','&designation');
Enter value for deptno: 9
Enter value for deptname: accounting
Enter value for location: chennai
Enter value for designation: manager
old   1: insert into department values(&deptno,'&deptname','&location','&designation')
new   1: insert into department values(9,'accounting','chennai','manager')
1 row created.

SQL> select * from department ;

    DEPTNO     DEPTNAME   LOCATION   DESIGNATION
---------- ---------- ---------- ----------
         9 accounting   hyderabad  manager
        10 research     chennai    professor
        11 sales        banglore   salesman
        12  operations  mumbai     operator
         9 accounting   chennai    manager
```

c. List the records of dept table grouped by deptno.

```
SQL> select deptno,deptname from department group by deptno,deptname;

    DEPTNO DEPTNAME
---------- ----------
         9 accounting
```

12 operations
10 research
11 sales

d. Update the record where deptno is 9.

SQL> update department set designation='accountant' where deptno=9;

2 rows updated.
SQL> select * from department;

DEPTNO	DEPTNAME	LOCATION	DESIGNATION
9	accounting	hyderabad	accountant
10	research	chennai	professor
11	sales	banglore	salesman
12	operations	mumbai	operator
9	accounting	chennai	accountant

e. Delete any column data from the table.

SQL> alter table department drop(designation);
Table altered.
SQL> select * from department;

DEPTNO	DEPTNAME	LOCATION
9	accounting	hyderabad
10	research	chennai
11	sales	banglore
12	operations	mumbai
9	accounting	Chennai

LAB ASSIGNMENT:

1. Create a table called Customer table

Name	**Type**
Cust name	Varchar2(20)
Cust street	Varchar2(20)
Cust city	Varchar2(20)

a. Insert records into the table.
b. Add salary column to the table.
c. Alter the table column domain.
d. Drop salary column of the customer table.
e. Delete the rows of customer table whose cust_city is ‘hyd’.

2. Create a table called branch table.

Name	**Type**
Branch name	Varchar2(20)
Branch city	Varchar2(20)

asserts	Number

a. Increase the size of data type for asserts to the branch.
b. Add and drop a column to the branch table.
c. Insert values to the table.
d. Update the branch name column
e. Delete any two columns from the table

3. Create a table called sailor table

Name	**Type**
Sid	Number
Sname	Varchar2(20)
rating	Varchar2(20)

a. Add column age to the sailor table.
b. Insert values into the sailor table.
c. Delete the row with rating >8.
d. Update the column details of sailor.
e. Insert null values into the table.

4. Create a table called reserves table

Name	**Type**
Boat id	Integer
sid	Integer
day	Integer

a. Insert values into the reserves table.
b. Add column time to the reserves table.
c. Alter the column day data type to date.
d. Drop the column time in the table.
e. Delete the row of the table with some condition.

WEEK -2
QUERIES USING DDL AND DML

1. **a. Create a user and grant all permissions to the user.**
   b. Insert the any three records in the employee table and use rollback. Check the result.
   c. Add primary key constraint and not null constraint to the employee table.
   d. Insert null values to the employee table and verify the result.

SOLUTION:

a) create a user and grant all permissions to the user.

CONNECT <USER-NAME>/<PASSWORD>@<DATABASE NAME>;

--Create user query

CREATE USER <USER NAME> IDENTIFIED BY <PASSWORD>;

--Provide roles

GRANT CONNECT,RESOURCE,DBA TO <USER NAME>;

--Assigning privileges

GRANT CREATE SESSION GRANT ANY PRIVILEGE TO <USER NAME>;
GRANT UNLIMITED TABLESPACE TO <USER NAME>;

--Provide access to tables.

GRANT SELECT, UPDATE, INSERT, DELETE ON <TABLE NAME> TO <USER NAME>;

b) Insert the any three records in the employee table and use rollback. Check the result.

SQL> SELECT * FROM EMPLOYEE;

EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
101	abhi	manager	1234	1100	70
102	rohith	analyst	2345	9000	65
103	david	trainee	3456	9000	65
104	rahul	clerk	4567	7000	55

SQL> insert into employee values(&empno,'&ename','&job',&manager_no,&sal,&commission);
Enter value for empno: 105
Enter value for ename: aravind
Enter value for job: salesman
Enter value for manager_no: 5678
Enter value for sal: 5000
Enter value for commission: 50
old 1: insert into employee values(&empno,'&ename','&job',&manager_no,&sal,&commission)
new 1: insert into employee values(105,'aravind','salesman',5678,5000,50)

1 row created.

SQL> rollback;
Rollback complete.

SQL> SELECT * FROM EMPLOYEE;

EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
101	abhi	manager	1234	1100	70
102	rohith	analyst	2345	9000	65
103	david	trainee	3456	9000	65
104	rahul	clerk	4567	7000	55

c) Add primary key constraint and not null constraint to the employee table.

SQL> alter table employee modify(empno number primary key, ename varchar2(10) not null);
Table altered.
SQL> desc employee;

Name	Null?	Type
EMPNO	NOT NULL	NUMBER
ENAME	NOT NULL	VARCHAR2(10)
JOB		VARCHAR2(10)
MANAGER_NO		NUMBER
SAL		NUMBER
COMMISSION		NUMBER

d) Insert null values to the employee table and verify the result.

SQL> desc employee;

Name	Null?	Type
EMPNO	NOT NULL	NUMBER
ENAME	NOT NULL	VARCHAR2(10)
JOB	NOT NULL	VARCHAR2(10)
MANAGER_NO		NUMBER
SAL	NOT NULL	NUMBER
COMMISSION		NUMBER

SQL> insert into employee values(&empno,'&ename','&job',&manager_no,&sal,&commission);
Enter value for empno: 105
Enter value for ename: mohith
Enter value for job: salesman
Enter value for manager_no: 5678
Enter value for sal: null
Enter value for commission: 50
old 1: insert into employee values(&empno,'&ename','&job',&manager_no,&sal,&commission)
new 1: insert into employee values(105,'mohith','salesman',5678,null,50)
insert into employee values(105,'mohith','salesman',5678,null,50)
*

2. a. create a user and grant all permissions to the user.
b. Insert values in the department table and use commit.
c. Add constraints like unique and not null to the department table.
d. Insert repeated values and null values into the table.

SOLUTION:

a) create a user and grant all permissions to the user.

CONNECT <USER-NAME>/<PASSWORD>@<DATABASE NAME>;

--Create user query

CREATE USER <USER NAME> IDENTIFIED BY <PASSWORD>;

--Provide roles

GRANT CONNECT,RESOURCE,DBA TO <USER NAME>;

--Assigning privileges

GRANT CREATE SESSION GRANT ANY PRIVILEGE TO <USER NAME>;
GRANT UNLIMITED TABLESPACE TO <USER NAME>;

--Provide access to tables.

GRANT SELECT, UPDATE, INSERT, DELETE ON <TABLE NAME> TO <USER NAME>;

b) Insert values in the department table and use commit.

```
SQL> insert into department values(&deptno,'&deptname','&location');
Enter value for deptno: 13
Enter value for deptname: sales
Enter value for location: delhi
old   1: insert into department values(&deptno,'&deptname','&location')
new   1: insert into department values(13,'sales','delhi')
1 row created.

SQL> commit;
Commit complete.

SQL> select * from department;
   DEPTNO   DEPTNAME   LOCATION
----------  ----------  ----------
        9   accounting  hyderabad
       10   research    chennai
       11   sales       banglore
       12   operations  mumbai
        9   accounting  chennai
       13   sales       delhi
```

6 rows selected.

c) Add constraints like unique and not null to the department table.

SQL> alter table department modify(deptno number unique);

Table altered.

SQL> alter table department modify(location varchar2(10) not null);

Table altered.

SQL> DESC DEPARTMENT;

Name	Null?	Type
DEPTNO		NUMBER
DEPTNAME		VARCHAR2(10)
LOCATION	NOT NULL	VARCHAR2(10)

d) Insert repeated values and null values into the table.

```
SQL> insert into department values(&deptno,'&deptname','&location');
Enter value for deptno: 10
Enter value for deptname: research
Enter value for location:
old   1: insert into department values(&deptno,'&deptname','&location')
new   1: insert into department values(10,'research','')
insert into department values(10,'research','')
SQL> insert into department values(&deptno,'&deptname','&location');
Enter value for deptno: 10
Enter value for deptname: research
Enter value for location: hyderabad
old   1: insert into department values(&deptno,'&deptname','&location')
new   1: insert into department values(10,'research','hyderabad')
insert into department values(10,'research','hyderabad')
```

LAB ASSIGNMENT:

1 a. create a user and grant all permissions to the user.
   b. Insert values into the table and use commit.
   c. Delete any three records in the department table and use rollback.
   d. Add constraint primary key and foreign key to the table.

2 a. create a user and grant all permissions to the user.
   b. Insert records in the sailor table and use commit.
   c. Add save point after insertion of records and verify save point.
   d. Add constraints not null and primary key to the sailor table.

3 a. create a user and grant all permissions to the user.
   b. Use revoke command to remove user permissions.
   c. Change password of the user created.
   d. Add constraint foreign key and not null.

4 a. create a user and grant all permissions to the user.
b. Update the table reserves and use savepoint and rollback.
c. Add constraint primary key, foreign key and not null to the reserves table
d. Delete constraint not null to the table column.

WEEK -3
QUERIES USING AGGREGATE FUNCTIONS

AIM :- Queries using aggregate functions(COUNT,AVG,MIN,MAX,SUM),Group by,Order by,Having.

E_id	E_name	Age	Salary
101	Anu	22	9000
102	Shane	29	8000
103	Rohan	34	6000
104	Scott	44	10000
105	Tiger	35	8000
106	Alex	27	7000
107	Abhi	29	8000

(i) Create Employee table containing all Records.

```
SQL> create table emp(eid number,ename varchar2(10),age number,salary number);
        Table created.
SQL> desc emp;
 Name                 Null?              Type
 -------------------- --------           ----------------------------
 EID                                     NUMBER
 ENAME                                   VARCHAR2(10)
 AGE                                     NUMBER
 SALARY                                  NUMBER
```

(ii)Count number of employee names from employee table.

```
SQL> select count(ename) from emp;
        COUNT(ENAME)
        ------------------------
                7
```

(iii)Find the Maximum age from employee table.

```
SQL> select max(age) from emp;
        MAX(AGE)
        -----------------
            44
```

(iv)Find the Minimum age from employee table.

```
SQL>  select min(age) from emp;
        MIN(AGE)
        ----------------
            22
```

(v)Display the Sum of age employee table.

```
SQL> select sum(age) from emp;
        SUM(AGE)
        ----------------
            220
```

(vi)Display the Average of age from Employee table.

```
SQL>  select avg(age) from emp;
        AVG(AGE)
        ----------
        31.4285714
```

(vii)Create a View for age in employee table.

```
SQL> create or replace view A as select age from emp where age<30;
        View created.
```

(viii)Display views

```
SQL> select * from A;
        AGE
      ----------
        22
        29
        27
        29
```

(ix)Find grouped salaries of employees.(group by clause)

```
SQL> select salary from emp group by salary;
      SALARY
      ----------
        9000
       10000
        8000
        6000
        7000
```

(x).Find salaries of employee in Ascending Order.(order by clause)

```
SQL> select ename,salary from emp order by salary;
      ENAME      SALARY
      ---------- ----------
       rohan       6000
        alex       7000
       shane       8000
        abhi       8000
        tiger      8000
        anu        9000
        scott     10000

          7 rows selected.
```

(xi) Find salaries of employee in Descending Order.

```
SQL> select ename,salary from emp order by salary desc;
      ENAME      SALARY
      ---------- ----------
        scott      10000
        anu         9000
```

shane	8000
abhi	8000
tiger	8000
alex	7000
rohan	6000

7 rows selected.

(xii)Having Clause.

SQL> select ename,salary from emp where age<29 group by ename,salary having salary<10000;

ENAME	SALARY
alex	7000
anu	9000

LAB ASSIGNMENT:

Case Study 1:

a) By using the group by clause, display the enames who belongs to deptno 10 along with average salary.
b) Display lowest paid employee details under each department.
c) Display number of employees working in each department and their department number.
d) Using built in functions, display number of employees working in each department and their department name from dept table. Insert deptname to dept table and insert deptname for each row, do the required thing specified above.
e) List all employees which start with either B or C.
f) Display only these ename of employees where the maximum salary is greater than or equal to 5000.

Case Study 2:

a) Calculate the average salary for each different job.
b) Show the average salary of each job excluding manager.
c) Show the average salary for all departments employing more than three people.
d) Display employees who earn more than the lowest salary in department 30
e) Show that value returned by sign (n) function.
f) How many days between day of birth to current date.

Case Study 3:

a) Show that two substring as single string.
b) List all employee names, salary and 15% rise in salary.
c) Display lowest paid emp details under each manager
d) Display the average monthly salary bill for each deptno.
e) Show the average salary for all departments employing more than two people.
f) By using the group by clause, display the eid who belongs to deptno 05 along with average salary.

Case Study 4:

a) Count the number of employees in department 20
b) Find the minimum salary earned by clerk.
c) Find minimum, maximum, average salary of all employees.
d) List the minimum and maximum salaries for each job type.

e) List the employee names in descending order.
f) List the employee id, names in ascending order by empid.

Case Study 5:

a) Find the sids ,names of sailors who have reserved all boats called "INTERLAKE
b) Find the age of youngest sailor who is eligible to vote for each rating level with at least two such sailors.
c) Find the sname , bid and reservation date for each reservation.
d) Find the ages of sailors whose name begin and end with B and has at least 3 characters.
e) List in alphabetic order all sailors who have reserved red boat.
f) Find the age of youngest sailor for each rating level.

Case Study 6:

a) List the Vendors who have delivered products within 6 months from order date.
b) Display the Vendor details who have supplied both Assembled and Sub parts.
c) Display the Sub parts by grouping the Vendor type (Local or Non Local).
d) Display the Vendor details in ascending order.
e) Display the Sub part which costs more than any of the Assembled parts.
f) Display the second maximum cost Assembled part.

WEEK – 4
PROGRAMS ON PL/SQL

1 a) Write a PL/SQL block to find the maximum number from given three numbers.

```
declare
a number;
b number;
c number;
begin
  a:=&a;
  b:=&b;
  c:=&c;
  if  (a>b and a>c) then
          dbms_output.put_line('a is maximum ' || a);
  elsif (b>a and b>c) then
          dbms_output.put_line('b is maximum ' || b);
  else
          dbms_output.put_line('c is maximum ' || c);
  end if;
end;
  /
```

1b) write a PL/SQL program for swapping 2 numbers.

```
declare
a number(3);
b number(3);
begin
      a:=&a;
      b:=&b;
      dbms_output.put_line('Before swapping a= '||a||' and b= '||b);
      a:=a+b;
      b:=a-b;
      a:=a-b;
      dbms_output.put_line('After swapping a= '||a||' and b= '||b);
      end;
  /
```

2 a) Write a PL/SQL program to find the total and average of 4 subjects and display the grade

```
declare
 java number(10);
 dbms number(10);
 co number(10);
 mfcs number(10);
 total number(10);
 avgs number(10);
 per number(10);
 begin
      dbms_output.put_line('ENTER THE MARKS');
```

```
        java:=&java;
        dbms:=&dbms;
        co:=&co;
        mfcs:=&mfcsl;
        total:=(java+dbms+co+mfcs);
        per:=(total/600)*100;
        if java<40 or dbms<40 or co<40 or mfcs<40 then
            dbms_output.put_line('FAIL');
        if per>75 then
            dbms_output.put_line('GRADE A');
        elsif per>65 and per<75 then
            dbms_output.put_line('GRADE B');
        elsif per>55 and per<65 then
            dbms_output.put_line('GRADE C');
        else
            dbms_output.put_line('INVALID INPUT');
        end if;
            dbms_output.put_line('PERCENTAGE IS '||per);
end;
/
```

2 b) Write a program to accept a number and find the sum of the digits

```
declare
n number(5):=&n;
s number:=0;
r number(2):=0;
begin
  while n !=0
  loop
  r:=mod(n,10);
  s:=s+r;
  n:=trunc(n/10);
  end loop;
dbms_output.put_line('sum of digits of given number is '||s);
end;
/
```

3 a) PL/SQL Program to accept a number from user and print number in reverse order.

```
declare
 num1 number(5);
 num2 number(5);
 rev number(5);
begin
 num1:=&num1;
 rev:=0;
 while num1>0
 loop
  num2:=num1 mod 10;
  rev:=num2+(rev*10);
  num1:=floor(num1/10);
 end loop;
```

```
  dbms_output.put_line('Reverse number is: '||rev);
end;
/
```

3b) Write a PL / SQL program to check whether the given number is prime or not.

```
declare
    num number;
    i number:=1;
    c number:=0;
begin
     num:=&num;
    for i in 1..num
    loop
      if((mod(num,i))=0)
       then
          c:=c+1;
      end if;
   end loop;
  if(c>2)
  then
     dbms_output.put_line(num||' not a prime');
  else
    dbms_output.put_line(num||' is prime');
  end if;
end;
/
```

4 a) Write a PL/SQL program to find the factorial of a given number.

```
declare
     i number(4):=1;
     n number(4):=&n;
     f number(4):=1;
begin
  for i in 1..n
  loop
    f:=f*i;
   end loop;
   Dbms_output.put_line('the factorial of '||n||' is:'||f);
end;
/
```

4 b) calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in table areas. Consisting of two columns radius and area

```
Declare
pi constant number(4,2) := 3.14;
radius number(5);
area number(14,2);
Begin
radius := 3;
While radius <=7
Loop
        area := pi* power(radius,2);
        Insert into areas values (radius, area);
```

```
            radius:= radius+1;
        end loop;
        end;
        /
```

5a) Write a PL/SQL program to accept a string and remove the vowels from the string. (When 'hello' passed to the program it should display 'Hll' removing e and o from the world Hello).

```
set serveroutput on
set verify off
accept vstring prompt "Please enter your string: ";
declare
  vnewstring varchar2(100);
begin
  vnewstring := regexp_replace('&vstring', '[aeiouAEIOU]','');
  dbms_output.put_line('The new string is: ' || vnewstring);
end;
/
```

5 b) Write a PL/SQL program to accept a number and a divisor. Make sure the divisor is less than or equal to 10. Else display an error message. Otherwise Display the remainder.

```
select remainder(37,5) "remainder" from dual ;
```

WEEK -5
PROCEDURES AND FUNCTIONS

1) calculate the net salary and year salary if da is 30% of basic, hra is 10% of basic and pf is 7% if basic salary is less than 8000, pf is 10% if basic sal between 8000 to 160000.

```
declare
   ename varchar2(15);
   basic number;
   da number;
   hra number;
   pf number;
   netsalary number;
   yearsalary number;
begin
   ename:='&ename';
   basic:=&basic;
   da:=basic * (30/100);
   hra:=basic * (10/100);
      if (basic < 8000)
   then
        pf:=basic * (8/100);
      elsif (basic >= 8000 and basic <= 16000)
   then
      pf:=basic * (10/100);

end if;
      netsalary:=basic + da + hra - pf;
      yearsalary := netsalary*12;

dbms_output.put_line('Employee name : ' || ename);
dbms_output.put_line('Providend Fund : ' || pf);
dbms_output.put_line('Net salary : ' || netsalary);
dbms_output.put_line('Year salary : '|| yearsalary);
end;
/
```

2) Create a function to find the factorial of a given number and hence find NCR.

```
SQL> create or replace function fact(n number)
return number is
a number:=n;
f number:=1;
i number;
begin
for i in 1..n
loop
    f:=f*a;
    a:=a-1;
end loop;
return f;
end;
/
```

```
SQL> create or replace function ncr(n number ,r number)
 return number is
 n1 number:=fact(n);
 r1 number:=fact(r);
 nr1 number:=fact(n-r);
 result number;
 begin
 result:=(n1)/(r1*nr1);
 return result;
 end;
 /
```

3) Print Fibonacci series using local functions.

```
sql>create or replace function fib (n positive) return integer is
begin
if (n = 1) or (n = 2) then -- terminating condition
return 1;
else
return fib(n - 1) + fib(n - 2); -- recursive call
end if;
end fib;
/
```

-- Test Fibonacci Series:

```
SQL>SELECT fib(1), fib(2), fib(3), fib(4), fib(5) FROM dual;
```

4) write a pl/sql function accept date of birth as "dd-mm-yyyy" and sum all digits till you get single digit number to show as he lucky number.

```
SQL> set serverout on
SQL> declare
      l_input varchar2(20) := '31/01/1978';
      l_output int;
    begin
      loop
        dbms_output.put_line('-------------------');
        dbms_output.put_line('l_input='||l_input);
        l_output := 0;
        for i in 1 .. length(l_input)
       loop
         if substr(l_input,i,1) between '0' and '9' then
            l_output := l_output + to_number(substr(l_input,i,1));
         end if;
       end loop;
       dbms_output.put_line('l_output='||l_output);
       exit when l_output < 10;
       l_input := to_char(l_output);
     end loop;
     dbms_output.put_line('-------------------');
     dbms_output.put_line('Lucky='||l_output);
   end;
   /
```

```
-----------------
l_input=31/01/1978
l_output=30
-----------------
l_input=30
l_output=3
-----------------
Lucky=3

PL/SQL procedure successfully completed.
```

WEEK-6

TRIGGERS

1. Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

CUSTOMERS table:

ID	NAME	AGE	ADDRESS	SALARY
1	Alive	24	Khammam	2000
2	Bob	27	Kadappa	3000
3	Catri	25	Guntur	4000
4	Dena	28	Hyderabad	5000
5	Eeshwar	27	Kurnool	6000
6	Farooq	28	Nellur	7000

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary  - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

Trigger created.

Here following two points are important and should be noted carefully:

OLD and NEW references are not available for table level triggers, rather you can use them for record level triggers.

If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.

Above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using DELETE operation on the table.

Let us perform some DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table:

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'Kriti', 22, 'HP', 7500.00 );
```

When a record is created in CUSTOMERS table, above create trigger **display_salary_changes** will be fired and it will display the following result:

Old salary:
New salary: 7500
Salary difference:

2) Convert employee name into uppercase whenever an employee record is inserted or updated. Trigger to fire before the insert or update.

```
SQL> create table Employee(
 2   ID              VARCHAR2(4 BYTE) NOT NULL,
 3   First_Name       VARCHAR2(10 BYTE),
 4   Last_Name        VARCHAR2(10 BYTE),
 5   Start_Date       DATE,
 6   End_Date         DATE,
 7   Salary           NUMBER(8,2),
 8   City             VARCHAR2(10 BYTE),
 9   Description      VARCHAR2(15 BYTE)
10  )
11  /

Table created.

SQL> CREATE OR REPLACE TRIGGER employee_insert_update
  2   BEFORE INSERT OR UPDATE ON employee
  3   FOR EACH ROW
  4 DECLARE
  5   dup_flag  INTEGER;
  6 BEGIN
  7     --Force all employee names to uppercase.
  8 :NEW.first_name := UPPER(:NEW.first_name);
  9 END;
 10  /

Trigger created.

SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date,  End_Date,  Salary,  City,
Description)
  2            values ('01','Jason',    'Martin',  to_date('19960725','YYYYMMDD'),
to_date('20060725','YYYYMMDD'), 1234.56, 'Toronto',  'Programmer')
  3  /

1 row created.

SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date,  End_Date, Salary,  City,
Description)
  2            values('02','Alison',   'Mathews', to_date('19760321','YYYYMMDD'),
to_date('19860221','YYYYMMDD'), 6661.78, 'Vancouver','Tester')
  3  /
```

```
1 row created.

SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date, End_Date, Salary,  City,
Description)
  2             values('03','James',    'Smith',   to_date('19781212','YYYYMMDD'),
to_date('19900315','YYYYMMDD'), 6544.78, 'Vancouver','Tester')
  3  /

1 row created.

SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date, End_Date, Salary,  City,
Description)
  2             values('04','Celia',    'Rice',    to_date('19821024','YYYYMMDD'),
to_date('19990421','YYYYMMDD'), 2344.78, 'Vancouver','Manager')
  3  /

1 row created.

SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date, End_Date,  Salary,  City,
Description)
  2             values('05','Robert',   'Black',   to_date('19840115','YYYYMMDD'),
to_date('19980808','YYYYMMDD'), 2334.78, 'Vancouver','Tester')
  3  /

1 row created.

SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date,  End_Date, Salary, City,
Description)
  2             values('06','Linda',    'Green',   to_date('19870730','YYYYMMDD'),
to_date('19960104','YYYYMMDD'), 4322.78,'New York',  'Tester')
  3  /

1 row created.

SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date, End_Date,  Salary, City,
Description)
  2             values('07','David',    'Larry',   to_date('19901231','YYYYMMDD'),
to_date('19980212','YYYYMMDD'), 7897.78,'New York',  'Manager')
  3  /

1 row created.

SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date, End_Date, Salary, City,
Description)
  2             values('08','James',    'Cat',     to_date('19960917','YYYYMMDD'),
to_date('20020415','YYYYMMDD'), 1232.78,'Vancouver', 'Tester')
  3  /

1 row created.

SQL> select * from Employee
  2  /
```

```
ID   FIRST_NAME LAST_NAME  START_DAT END_DATE      SALARY CITY       DESCRIPTION
---- ---------- ---------- --------- --------- ---------- ---------- ------------------------------------------------------------
01   JASON      Martin              25-JUL-96       25-JUL-06    1234.56    Toronto    Programmer
02   ALISON     Mathews             21-MAR-76       21-FEB-86    6661.78    Vancouver  Tester
03   JAMES      Smith               12-DEC-78        15-MAR-90 6544.78    Vancouver  Tester
04   CELIA      Rice                24-OCT-82       21-APR-99    2344.78    Vancouver  Manager
05   ROBERT     Black               15-JAN-84       08-AUG-98    2334.78    Vancouver  Tester
06   LINDA      Green               30-JUL-87        04-JAN-96    4322.78    New York   Tester
07   DAVID      Larry               31-DEC-90       12-FEB-98    7897.78     New York   Manager
08   JAMES      Cat                 17-SEP-96       15-APR-02    1232.78    Vancouver  Tester

8 rows selected.

SQL> drop table Employee
  2 /

Table dropped.
```

3) Trigger before deleting a record from emp table. Trigger will insert the row to be deleted into another table and also record the user who has deleted the record.

```
SQL>     CREATE OR REPLACE TRIGGER employee_before_delete
  2     BEFORE DELETE
  3        ON employee
  4        FOR EACH ROW
  5     DECLARE
  6       v_username varchar2(10);
  7     BEGIN
  8       -- Find username of person performing the DELETE on the table
  9        SELECT user INTO v_username
 10          FROM dual;
 11      -- Insert record into audit table
 12          INSERT INTO employee_audit (id, salary, delete_date,deleted_by )
 13                       VALUES (:old.id,:old.salary, sysdate, v_username );
 14     END;
 15     /

Trigger created.

SQL> delete from employee;

8 rows deleted.

SQL> select * from employee_audit;

ID      SALARY DELETE_DA DELETED_BY
---- ---------- --------- ----------
01      1234.56 09-SEP-06 JAVA2S
02      6661.78 09-SEP-06 JAVA2S
03      6544.78 09-SEP-06 JAVA2S
04      2344.78 09-SEP-06 JAVA2S
```

05 2334.78 09-SEP-06 JAVA2S
06 4322.78 09-SEP-06 JAVA2S
07 7897.78 09-SEP-06 JAVA2S
08 1232.78 09-SEP-06 JAVA2S

8 rows selected.

SQL> drop table employee_audit;

Table dropped.

LAB ASSIGNMENT:

1. Creation of insert trigger, delete trigger, update trigger practice triggers using the passenger database.
   Passenger(Passport_ id INTEGER PRIMARY KEY, Name VARCHAR (50) Not NULL, Age Integer Not NULL, Sex Char, Address VARCHAR (50) Not NULL);
   Write a Insert Trigger to check the Passport_id is exactly six digits or not.
   Write a trigger on passenger to display messages '1 Record is inserted', '1 record is deleted', '1 record is updated' when insertion, deletion and updation are done on passenger respectively.

2. Insert row in employee table using Triggers. Every trigger is created with name any trigger have same name must be replaced by new name. These triggers can raised before insert, update or delete rows on data base. The main difference between a trigger and a stored procedure is that the former is attached to a table and is only fired when an INSERT, UPDATE or DELETE occurs.

3. Create a transparent audit system for a table CUST_MSTR. The system must keep track of the records that are being deleted or updated.

WEEK-7
PROCEDURES

1) Create a procedure to reverse a string.

```
CREATE OR REPLACE PROCEDURE ReverseOf(input IN varchar2(50)) IS
DECLARE
    reverse varchar2(50);
BEGIN
    FOR i in reverse 1..length(input) LOOP
        reverse := reverse||''||substr(input, i, 1);
    END LOOP;
    dbms_output.put_line(reverse);
END;
/
```

Lab Assignment:

1. Create the procedure for palindrome of given number.
2. Create the procedure for GCD: Program should load two registers with two Numbers and then apply the logic for GCD of two numbers. GCD of two numbers is performed by dividing the greater number by the smaller number till the remainder is zero. If it is zero, the divisor is the GCD if not the remainder and the divisors of the previous division are the new set of two numbers. The process is repeated by dividing greater of the two numbers by the smaller number till the remainder is zero and GCD is found.
3. Write the PL/SQL programs to create the procedure for factorial of given number.
4. Write the PL/SQL programs to create the procedure to find sum of N natural number.
5. Write the PL/SQL programs to create the procedure to find Fibonacci series.
6. Write the PL/SQL programs to create the procedure to check the given number is perfect or not.

WEEK-8
CURSORS

DEFINITION OF A CURSOR

1. Cursor can be created to store the values from table temporally.
2. In execution these values fetch from cursor for access the data base
    - Create cursor fetch the values from the table
    - Declare the variables
    - Open the cursor
    - Fetch the values from the cursor
    - Close the cursor

CURSOR EXAMPLE:

```
declare
cursor xx is select empno,ename,sal from emp26;
a_empno emp26.empno%type;
a_ename emp26.ename%type;
a_sal emp26.sal%type;
begin
open xx;
loop
fetch xx into a_empno,a_ename,a_sal;
exit when xx% not found;
dbms_output.put_line(a_empno||' '||a_ename||' '||a_sal);
end loop;
close xx;
end;

SQL> /
```

LAB ASSIGNMENT:

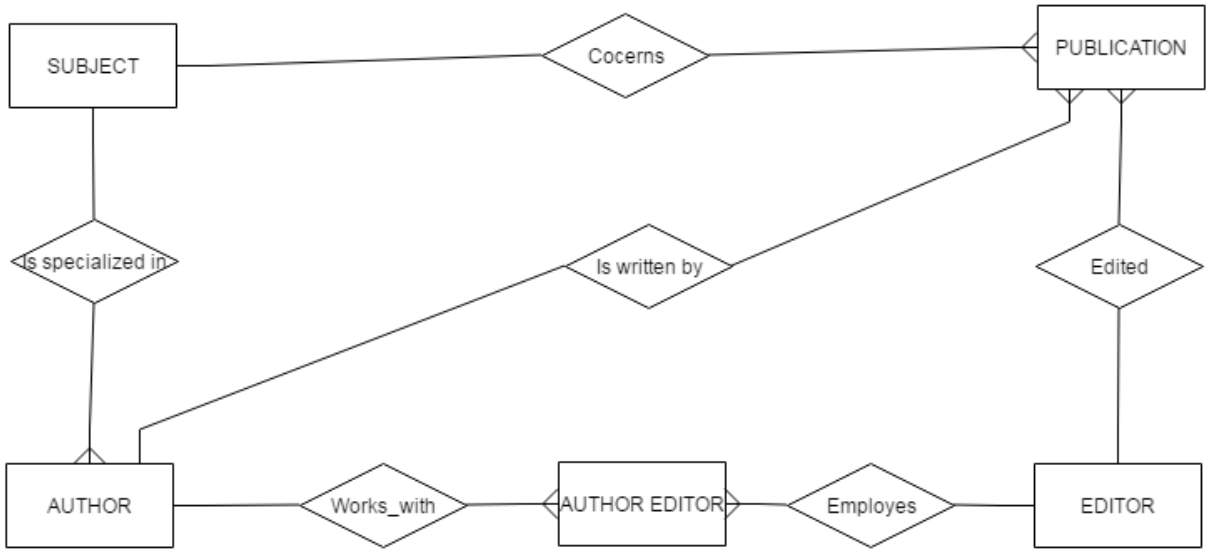
1. Write a PL/SQL block that will display the name, dept no, salary of fist highest paid employees**.**
2. Update the balance stock in the item master table each time a transaction takes place in the item transaction table. The change in item master table depends on the item id is already present in the item master then update operation is performed to decrease the balance stock by the quantity specified in the item transaction in case the item id is not present in the item master table then the record is inserted in the item master table.
3. Write a PL/SQL block that will display the employee details along with salary using cursors.
4. To write a Cursor to display the list of employees who are working as a Managers or Analyst.
5. To write a Cursor to find employee with given job and deptno.
6. Write a PL/SQL block using implicit cursor that will display message, the salaries of all the employees in the ‘employee’ table are updated. If none of the employee’s salary are updated we get a message 'None of the salaries were updated'. Else we get a message like for example, 'Salaries for 1000 employees are updated' if there are 1000 rows in ‘employee’ table.

WEEK-9

CASE STUDY: BOOK PUBLISHING COMPANY

AIM: A publishing company produces scientific books on various subjects. The books are written by authors who specialize in one particular subject. The company employs editors who, not necessarily being specialists in a particular area, each take sole responsibility for editing one or more publications.

A publication covers essentially one of the specialist subjects and is normally written by a single author. When writing a particular book, each author works with on editor, but may submit another work for publication to be supervised by other editors. To improve their competitiveness, the company tries to employ a variety of authors, more than one author being a specialist in a particular subject.



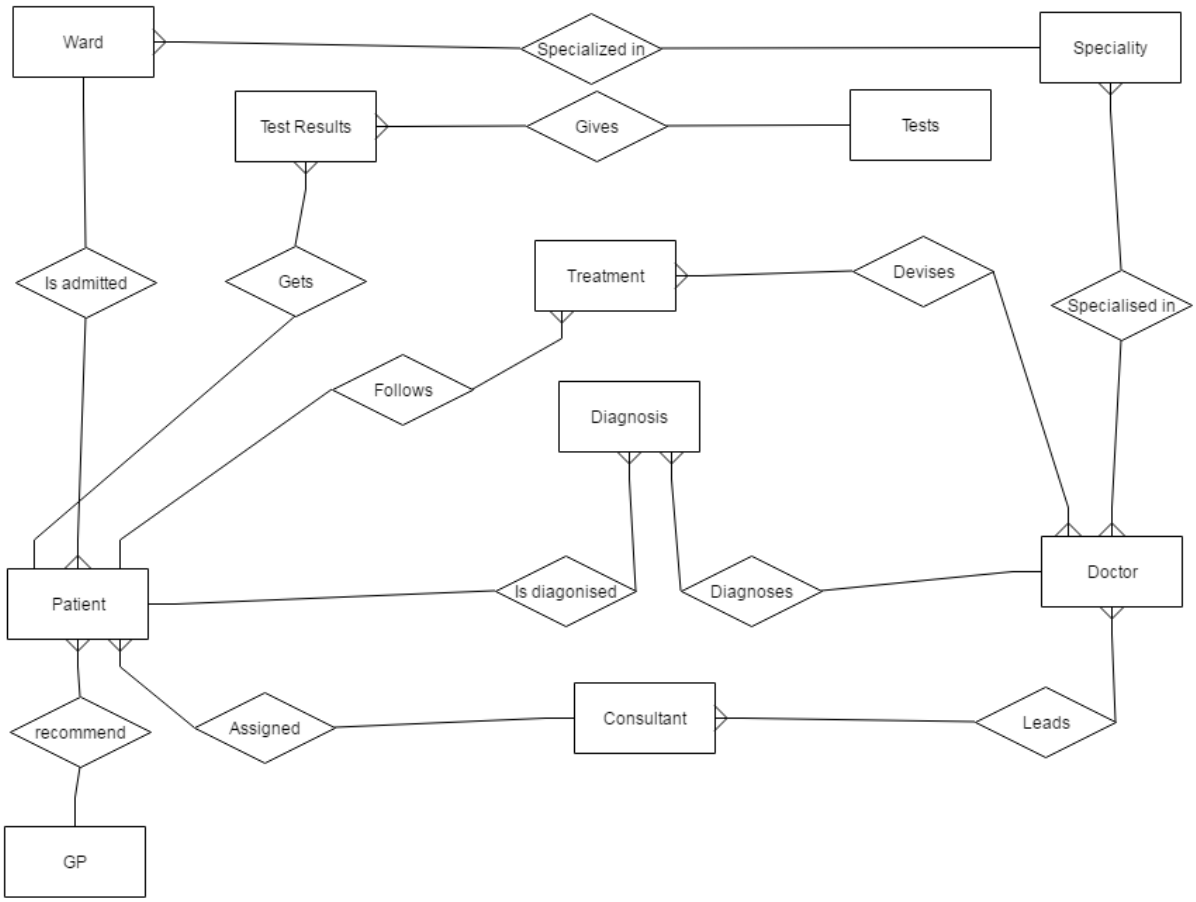
LAB ASSIGNMENT:

1. Analyze the data required.
2. Normalize the attributes.
3. Create the logical data model using E-R diagrams

WEEK -10
CASE STUDY: GENERAL HOSPITAL

AIM: A General Hospital consists of a number of specialized wards (such as Maternity, Paediatry, Oncology, etc). Each ward hosts a number of patients, who were admitted on the recommendation of their own GP and confirmed by a consultant employed by the Hospital. On admission, the personal details of every patient are recorded.

A separate register is to be held to store the information of the tests undertaken and the results of a prescribed treatment. A number of tests may be conducted for each patient. Each patient is assigned to one leading consultant but may be examined by another doctor, if required. Doctors are specialists in some branch of medicine and may be leading consultants for a number of patients, not necessarily from the same ward.



LAB ASSIGNMENT:

1. Analyze the data required.
2. Normalize the attributes.
3. Create the logical data model using E-R diagrams

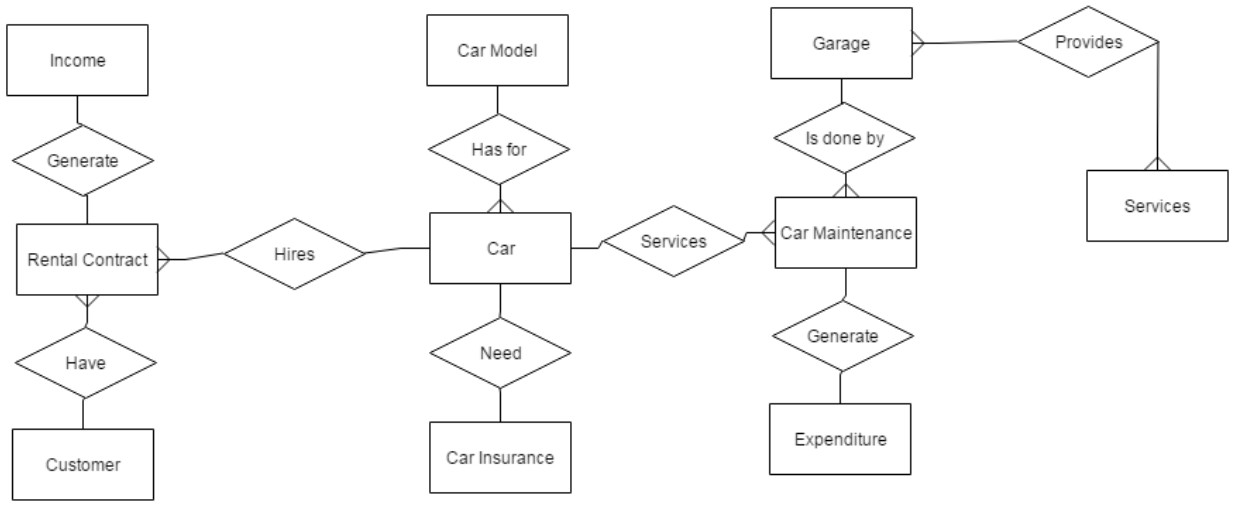
WEEK -11

CASE STUDY: CAR RENTAL COMPANY

AIM: A database is to be designed for a Car Rental Co. (CRC). The information required includes a description of cars, subcontractors (i.e. garages), company expenditures, company revenues and customers. Cars are to be described by such data as: make, model, year of production, engine size, and fuel type, number of passengers, registration number, purchase price, purchase date, rent price and insurance details. It is the company policy not to keep any car for a period exceeding one year.

All major repairs and maintenance are done by subcontractors (i.e. franchised garages), with whom CRC has long-term agreements. Therefore the data about garages to be kept in the database includes garage names, addresses, range of services and the like. Some garages require payments immediately after a repair has been made; with others CRC has made arrangements for credit facilities. Company expenditures are to be registered for all outgoings connected with purchases, repairs, maintenance, insurance etc.

Similarly the cash inflow coming from all sources - car hire, car sales, insurance claims - must be kept of file.CRC maintains a reasonably stable client base. For this privileged category of customers special credit card facilities are provided. These customers may also book in advance a particular car. These reservations can be made for any period of time up to one month. Casual customers must pay a deposit for an estimated time of rental, unless they wish to pay by credit card. All major credit cards are accepted. Personal details (such as name, address, telephone number, driving license, number) about each customer are kept in the database.



LAB ASSIGNMENT:

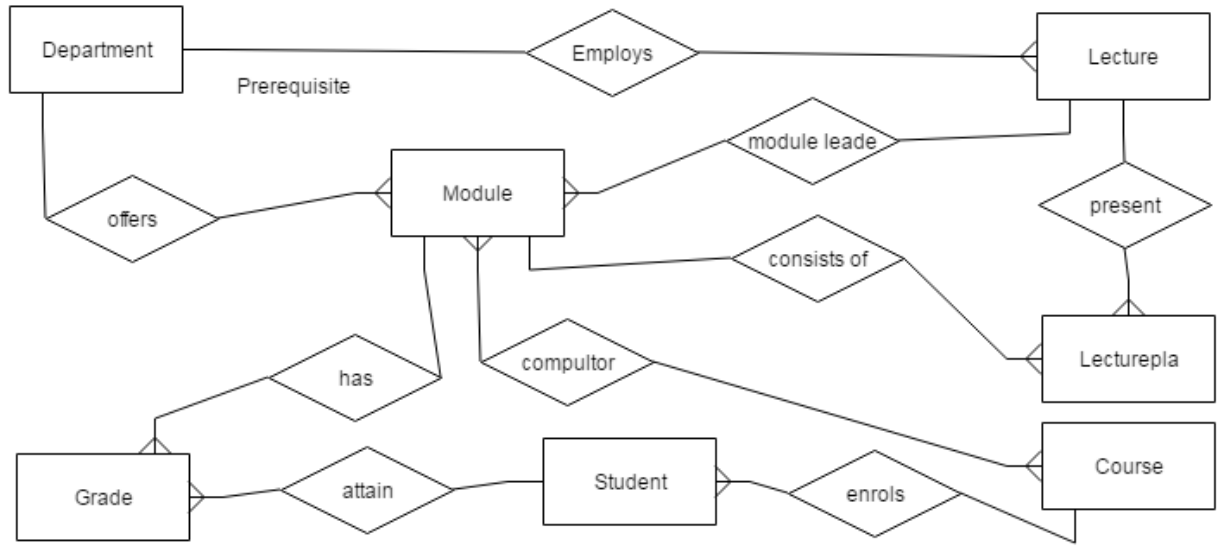
1. Analyze the data required.
2. Normalize the attributes.
3. Create the logical data model using E-R diagrams

WEEK-12

CASE STUDY: STUDENT PROGRESS MONITORING SYSTEM

AIM: A database is to be designed for a college to monitor students' progress throughout their course of study. The students are reading for a degree (such as BA, BA(Hons) MSc, etc) within the framework of the modular system. The college provides a number of module, each being characterised by its code, title, credit value, module leader, teaching staff and the department they come from. A module is co- ordinated by a module leader who shares teaching duties with one or more lecturers.

A lecturer may teach (and be a module leader for) more than one module. Students are free to choose any module they wish but the following rules must be observed: some modules require pre-requisites modules and some degree programmes have compulsory modules. The database is also to contain some information about students including their numbers, names, addresses, degrees they read for, and their past performance (i.e. modules taken and examination results).



LAB ASSIGNMENT:

1. Analyze the data required.
2. Normalize the attributes.
3. Create the logical data model using E-R diagrams

VIVA QUESTIONS WITH ANSWERS

UNIT-1

1. What is database?
A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

2. What is DBMS?
It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

3. What is a Database system?
The database and DBMS software together is called as Database system.

4. Advantages of DBMS?
- Redundancy is controlled.
- Unauthorized access is restricted.
- Providing multiple user interfaces.
- Enforcing integrity constraints.
- Providing backup and recovery.

5. Disadvantage in File Processing System?
- Data redundancy & inconsistency.
- Difficult in accessing data.
- Data isolation.
- Data integrity.
- Concurrent access is not possible.
- Security Problems.

6. Describe the three levels of data abstraction?
Three levels of abstraction:
Physical level: The lowest level of abstraction describes how data are stored.
Logical level: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
View level: The highest level of abstraction describes only part of entire database.

7. Define the "integrity rules"
There are two Integrity rules.
Entity Integrity: States that Primary key cannot have NULL value
Referential Integrity: States that Foreign Key can be either a NULL value or should be Primary Key value of other relation.

8. What is extension and intension?
Extension: It is the number of tuples present in a table at any instance. This is time dependent.
Intension: It is a constant value that gives the name, structure of table and the constraints laid on it.

9. What is Data Independence?
Data independence means that "The application is independent of the storage structure and access strategy of data". In other words, the ability to modify the schema definition in one level should not affect the schema definition in the next higher level.
Two types of Data Independence:
Physical Data Independence: Modification in physical level should not affect the logical level.
Logical Data Independence: Modification in logical level should affect the view level.

10. What is a view? How it is related to data independence?
A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary. Growth and restructuring of base tables is not reflected in views. Thus the
View can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

11. What is Data Model?
A collection of conceptual tools for describing data, data relationships data semantics and constraints.

12. What is E-R model?
This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

13. What is Object Oriented model?
This model is based on collection of objects. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

14. What is an Entity?
It is a 'thing' in the real world with an independent existence.

15. What is an Entity type?
It is a collection (set) of entities that have same attributes.

16. What is an Entity set?
It is a collection of all entities of particular entity type in the database.

17. What is an Extension of entity type?
The collections of entities of a particular entity type are grouped together into an entity set.

18. What is Weak Entity set?
An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

19. What is an attribute?
It is a particular property, which describes the entity.

20. What is a Relation?
A relation is defined as a set of tuples.

21. What is degree of a Relation?
It is the number of attribute of its relation schema.

22. What is Relationship?
It is an association among two or more entities.

23. What is Relationship set?
The collection (or set) of similar relationships.

24. What is Relationship type?
Relationship type defines a set of associations or a relationship set among a given set of entity types.

25. What is degree of Relationship type?

It is the number of entity type participating.

UNIT-2

1. What is DDL (Data Definition Language)?
A data base schema is specifies by a set of definitions expressed by a special language called DDL.

2. What is VDL (View Definition Language)?
It specifies user views and their mappings to the conceptual schema.

3. What is DML (Data Manipulation Language)?
This language that enable user to access or manipulate data as organized by appropriate data model.

4. What is DML Compiler?
It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

5. What is Query evaluation engine?
It executes low-level instruction generated by compiler.

6. What is DDL Interpreter?
It interprets DDL statements and records them in tables containing metadata.

7. What is a query?
A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

8. What do you mean by Correlated sub query?
A correlated sub query can be easily identified if it contains any references to the parent sub query columns in its WHERE clause. Columns from the sub query cannot be referenced anywhere else in the parent query.

9. Are the resulting relations of PRODUCT and JOIN operation the same?
No.
PRODUCT: Concatenation of every row in one relation with every row in another.
JOIN: Concatenation of rows from one relation and related rows from another.

10. What is database Trigger?
A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can be defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written in PL/SQL.

11. What are stored-procedures? What are the advantages of using them?
Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result tothe client. Stored procedures are used to reduce network traffic.

12. Define super key and give example to illustrate the super key?
Set of one or more attributes taken collectively, allowing to identify uniquely an entity in the entity set.Eg1. {SSN} and {SSN, Cust_name} of customer table are super keys.Eg2. {Branch name} and {Branch name, Branch city} of Branch table re super keys.

13. Define candidate key and give example to illustrate the candidate key?

Super keys with no proper subset are called the candidate keys. Otherwise it is called minimal super key. Candidate key is nothing but the primary key used in SQL. Eg1. {SSN} is the candidate keyfor the super keys {SSN} and {SSN, Cust_name} of customer table.Eg2. {Branch name} is the candidate key for the super keys {Branch name} and {Branch name, Branch city} of Branch table.

14. What is Primary key?

A key chosen to act as the means by which to identify tuples in a relation.

15. What is foreign key?

A foreign key of relation R is a set of its attributes intended to be used (by each tuple in R) for identifying/referring to a tuples in some relation S. (R is called the referencing relation and S the referenced relation.) For this to make sense, the set of attributes of R forming the foreign key should "correspond to" some superkey of S. Indeed, by definition we require this superkey to be the primary key of S.

14. What is a 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.

UNIT-3

1. What is normalization?

It is a process of analyzing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

- Minimizing redundancy
- Minimizing insertion, deletion and update anomalies.

2. What is Functional Dependency?

A Functional dependency is denoted by X-> Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y].

3. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

4. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency X-> Y is fully functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

5. What is 2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

6. What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X A either of the following is true
X is a Super-key of R.
A is a prime attribute of R.
In other words, if every non prime attribute is non-transitively dependent on primary key.

7. What is BCNF (Boyce-Codd Normal Form)?

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD X A, X must be a candidate key.

8. What is 4NF?

A relation schema R is said to be in 4NF if for every multivalued dependency X Y that holds over R, one of following is true X is subset or equal to (or) XY = R. X is a super key.

9. What is 5NF?
A Relation schema R is said to be 5NF if for every join dependency {R1, R2... Rn} that holds R, one the following is true
i) Ri = R for some i.
ii) The join dependency is implied by the set of FD, over R in which the left side is key of R.

10.What is dependency preservation?
Dependency Preservation Property enables us to enforce a constraint on the original relation from corresponding instances in the smaller relations.

11. What is Lossless join property?
Lossless join property enables us to find any instance of the original relation from corresponding instances in the smaller relations

12. What are Multivalued dependencies?
A multivalued dependency (MVD) X ->->Y specified on R, where X, and Y are both subsets of R and Z = (R – (X Y)) specifies the following restrictions on r(R)
t3[X]=t4[X]=t1[X]=t2[X]
t3[Y] = t1[Y] and t4[Y] = t2[Y]
t3[Z] = t2[Z] and t4[Z] = t1 [Z]

UNIT-4

1. What is a transaction?
A transaction is a logical unit of database processing that includes one or more database access operations (e.g., insertion, deletion, modification, or retrieval operations).

2. List the ACID properties?
a) Atomicity b) Consistency c) Isolation d) Durability

3. What is Atomicity?
A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.

4. What is Consistency?
A transaction is consistency preserving if its complete execution take(s) the database from one consistent state to another.

5. What is Isolation?
A transaction should appear as though it is being executed in isolation from other transactions. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.

6. What is Durability?
The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

7. When two operations in a Schedule Rollbacks?
Two operations in a schedule are said to conflict if they satisfy all three of the following
Conditions:
1. They belong to different transactions;
2. They access the same item X; and
3. At least one of the operations is a write_item(X).

8. Define recoverable schedule.
Recoverable schedule is the one where for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, the commit operation of Ti appears before the commit operation of Tj.

9. What is a checkpoint and when does it occur?
A Checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

10. What is blind write?
If a transaction writes a data item without reading the data is called blind write. This sometimes causes inconsistency.

11. Define serial schedule?
A schedule, S is serial if for every transaction T participating in the schedule and all the operations of T is executed consecutively in the schedule; otherwise the schedule is called Non-serial schedule.

12. What is the use of locking?
It is used to prevent concurrent transactions from interfering with one another and enforcing an additional condition that guarantees serializability.

UNIT-5

1. What is indexing?
Indexing is a technique for determining how quickly specific data can be found.

2. What is dense index?
If there is an index entry for every data record.

3. What is sparse index?
If there is an index entry for subset of data records.

4. What is primary index?
If there is a key, ordering field then it is primary index.

5. What is clustering index?
If there is non-key, ordering field then it is clustering index.

6. What is secondary index?
If there is non-ordering field then it is secondary index

7. What is multilevel index?
It is a tree built by indexing the indexes.

8. What is a file?
A ***file*** is a set of records stored as a unit on disk.

9. What is a B+ tree?
An organizational structure for information storage and retrieval in the form of a tree in which all terminal nodes are the same distance from the base, and all non-terminal nodes have between *n* and 2 *n* subtrees or pointers (where *n* is an integer).

10. What is Linear hashing?
Linear hashing allows for the expansion of the hash table one slot at a time.