Sri Sathya Sai College for Women, Bhopal
(An Autonomous College Affiliated to Barkatullah University Bhopal)
Department of Higher Education, Govt. of M.P.
Under Graduate Syllabus (Annual Pattern)
As recommended by Central Board of Studies and approved by the Governor of M. P.
wef 2022-2023
(Session 2023-24)
(NEP-2020)

Class	**B.C.A.**
Year	**II Year**
Subject	**Computer Applications**
Course Title	**Database Management Systems Using PL/SQL**
Course Type	**Core Course (Major II)**
Credit Value	**4**
Max. Mark	**30+70 (Minimum Marks 35)**

Course Outcome: After the completion of this course, a student shall be able to:

- Explain the features of database management systems and relational database.
- Design conceptual models of a database using ER modelling for real life applications and construct queries in relational algebra.
- Create and populate a RDBMS for a real-life application, with constraints and keys, using SQL.
- Retrieve any type of information from a database by formulating complex queries in SQL.
- Analyse the existing design of a database schema and apply concepts of normalization to design an optimal database.

Particular

Unit I	**Introduction to DBMS:** Why database? Characteristics of data in database, DBMS. What are database advantages of DBMS? **Database Architecture and Modelling:** Conceptual, physical and logical database models, Role of DBA, Database design. **Entity Relationship (ER) Model:** Components of ER-model, ER modeling symbols, Relationships. **Enhanced Entity Relationship (EER) Model:** An Introduction, Superclass and subclass entity types, Specialization, Generalization, Attribute inheritance, Categorization & Aggregation. **Keywords:** DBMS, DBA, Entity Relationship (ER), EER, Superclass, Subclass, Specialization, Generalization, Categorization & Aggregation.
Unit II	**The Relational Data Model:** **Fundamental Concepts:** Relations, Null Values, Keys, Foreign Keys, Integrity Constraints – Entity Integrity & Relational Integrity. **Normalization Process:** First Normal Form, Functional Dependencies, Second Normal Form, Third Normal Form, Boyce-Codd Normal Form (BCNF), Fourth Normal Form; Other Normal Forms – Fifth Normal Form & Domain/Key Normal Form. **Transforming a Conceptual Model to a Relational Model:** Transforming Objects Sets and Attributes, Transforming Models without External keys, Transforming Specialization and Generalization, Object Sets, Transforming Relationships: One-One Relationships, One-Many Relationships, Many-Many Relationships; Transforming Aggregated Object Sets, Transforming Recursive Relationships.

[illegible] [illegible] [illegible] Akumar [illegible]

	Keywords: Keys, Normalization, BCNF, Aggregated Object Sets, Recursive Relationship.
Unit III	**Relational database implementation:** **(a) Relational Algebra and Calculus** Relational Algebra: Union, Intersection, Difference, Product, Select, Project, Join-Natural, Theta & Outer Join, Divide, Assignment. Relational Calculus: Target list & Qualifying Statement, The Existential Quantifier, The Universal Quantifier. **Keywords:** JOIN, Target list, Existential Quantifier, Universal Quantifier.
Unit IV	**Relational database implementation (continued):** **(b) Relational Implementation with SQL** Relational Implementation: An Overview. Schema and Table Definition: Schema definition, Data types & domains, Defining Tables, Column Definition. Data Manipulation: Simple Queries (SELECT, FROM, WHERE), Multiple-Table Queries, Subqueries, Correlated Subqueries, EXISTS and NOT EXISTS operators, Built-In Functions (SUM, AVG, COUNT, MAX and MIN), GROUP BY and HAVING clause, Built-In Functions with Subqueries. Relational Algebra Operations: UNION, INTERSECT, EXCEPT, JOIN Database Change Operations: INSERT, UPDATE, DELETE, Using SQL with Data Processing Languages; View Definition, Restrictions on View Queries and Updates. **Keywords:** Schema, SELECT, Data Manipulation, Database Change Operation, View.
Unit V	**Physical Database Systems** Introduction, Physical Access of the Database. Physical Storage Media: Secondary Storage, Physical Storage Blocks. Disk Performance Factors: Access Motion Time, Head Activation Time, Rotational Delay, Data Transfer Rate, Data Transfer Time. Data Storage Formats on Disk: Track Format, Record Format- Fixed-Length Records & Variable-Length Records, Input/Output Management. File Organizing and Addressing Methods: Sequential File Organization, Indexed Sequential File Organization, Direct File Organization, Hashing: Static Hash Functions and Dynamic Hash Functions. **Keywords**: Disk Performance Factors, Sequential File Organization, Indexed Sequential File Organization, Direct File Organisation, Hashing.

Suggestion Books:

- Gary W. Hansen & James V. Hansen, "Database Management and Design", 2nd Ed., 2007, Prentice Hall of India Pvt. Ltd.
- Instructional Software Research & Development (ISRD) Group, Lucknow "Introduction to Database Management Systems", 2006, Ace Series, Tata McGraw Hill Publishing Company Limited, New Delhi.
- Ramez Elmasri, Shamkant B. Navathe, "Fundamentals of Database Systems", 7th Edition, 2016, Pearson

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

Reference Books:

- Raghu Ramakrishnan & Johannes Gehrke, "Database Management Systems", 3rd Edition, 2014, McGraw Hill Education
- C.J. Date, "An Introduction to Database Systerm", 8th Edition, 2003, Pearson
- Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", 6th Edition, 2010, Tata McGraw Hill

Suggestive digital platform web links

- https://en.wikipedia.org/wiko/Relational_model
- http://en.wikipedia.org/wiki/Relational_algebra

Suggested equivalent online courses

- NPTEL Course: Introduction to Database Systems or Database Design

Scheme of Marks:

Maximum Marks: 100		
Continuous Comprehensive Evaluation (CCE): 30 marks, Term End Exam Theory: 70 marks		
Internal Assessment: Continuous Comprehensive Evaluation (CCE):	Class Test Assignment/ Presentation	**30**
External Assessment: University Exam Section Time:03.00 Hours	**Section (A)** Very Short questions **Section (B)** Short questions **Section (C)** Long questions	**70**
		Total 100

[illegible] [illegible] Akumar [illegible]

[illegible] [illegible]

Sri Sathya Sai College for Women, Bhopal
(An Autonomous College Affiliated to Barkatullah University Bhopal)
Department of Higher Education, Govt. of M.P.
Under Graduate Syllabus (Annual Pattern)
As recommended by Central Board of Studies and approved by the Governor of M. P.
wef 2022-2023
(Session 2023-24)
(NEP-2020)

Class	**B.C.A.**
Year	**II Year**
Subject	**Computer Applications**
Course Title	**DBMS Using PL/SQL Lab**
Course Type	**Core Course (Major II)**
Credit Value	**2**
Max. Mark	**30+70 (Minimum Marks 35)**

Course Outcome: This lab is based on the theory course of DBMS. This lab course involves the development of the practical skills in DBMS using MS-Access/Visual-FoxPro/SQL-Server/etc. This course is an attempt to upgrade and enhance students' theoretical skills and provide the hands-on experience.

After the completion of this course, a student shall be able:

- To create Databases & views.
- Execute simple & advance SQL queries.
- Use DBMS tools in the areas of database applications.

Particular

Practicum details:

Students are required to practice the concepts learnt in the theory by designing and querying a database for a chosen organization (Like: College, Library, Transport, etc). The teacher may devise appropriate weekly lab assignments to help students practice the designing, querying a database in the context of example database. Some suggestive list of experiments with their aim, problem definition, theory is given below:

Experiment-1

Aim: To draw ER Model and Relational Model for a given database. Show ER to Relational Model reduction.

Resources used: MS-Access/Visual-FoxPro/SQL-Server/etc

Problem Definition: List the data requirements for the database of the company which keeps track of the company employee, department and projects. The database designers provide the following description:

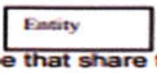
1. The company is organized into departments. Each department has unique name, unique number, and particular employee to manage the department. We keep track of the start date and the employee begins managing the department. The department has several locations.
2. The department controls a number of projects each of which has a unique name, unique number and a single location.

[illegible] [illegible] [illegible] [illegible] [illegible]

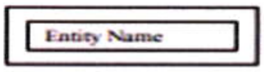
3. We store each employee names social security number, address, salary, sex and dob. An employee is assigned one department but may work on several projects which are not necessarily controlled by the same department. We keep track of the department of each employee works on each project and for insurance purpose. We keep each dependent's first name, sex, dob and relation.

Theory: The ER data model was developed to facilitate the database design by allowing specification of an enterprise schema that represents the overall logical structure of the database. The ER model data model is one of the several data models. The semantic aspect of the model lies in its representation of the meaning of the data. The ER model is very useful many database design tools drawn on concepts from the ER model. The ER model employs 3 basic notations: entity set, relationship set and attributes.

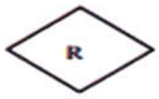
Symbols Used in ER Notation

1. 

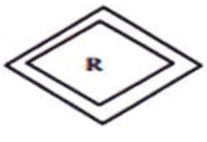
Entity set: An entity is a set of entities of the same type that share the properties or attributes.

2. 

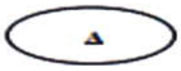
Weak entity set: An entity set may not have sufficient attributes to form a primary key. Such an entity set is termed as weak entity set.

3. 

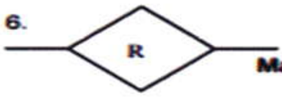
Realtionship Set: A relationship is an association among several entities. A relationship set is a set of relationship of the same type.

4. 

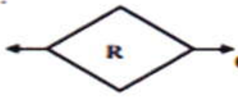
Identification relationship set for weak entity set : The relationship associating the weak entity set with the identifying entity set is called the identifying relationship.

5. 

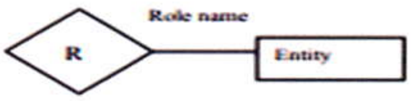
Primary key : The primary key is used to denote a candidate key that is chosen by the database designers as the principal means of identifying entities within an entity set.

6. 

Many to many relationship

7. 

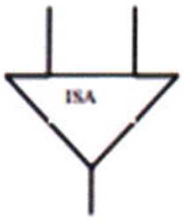
One to One relationship

8. 

Role Indicator

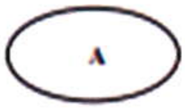
[illegible]

9.



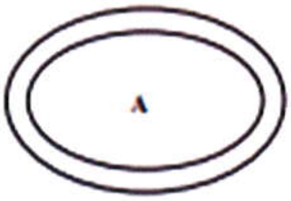
Total Generalization

10.



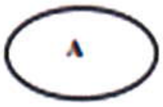
Attribute

11.



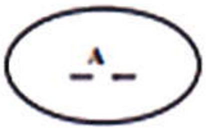
Multi valued Attribute

12.



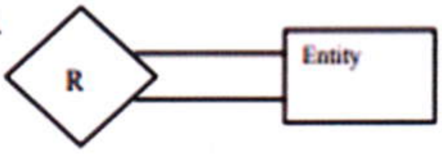
Derived Attribute

13.



Discriminating Attribute of weak entity set : The discrimination of weak entity set is a set of attributes that allows the distinction to be made.

14.

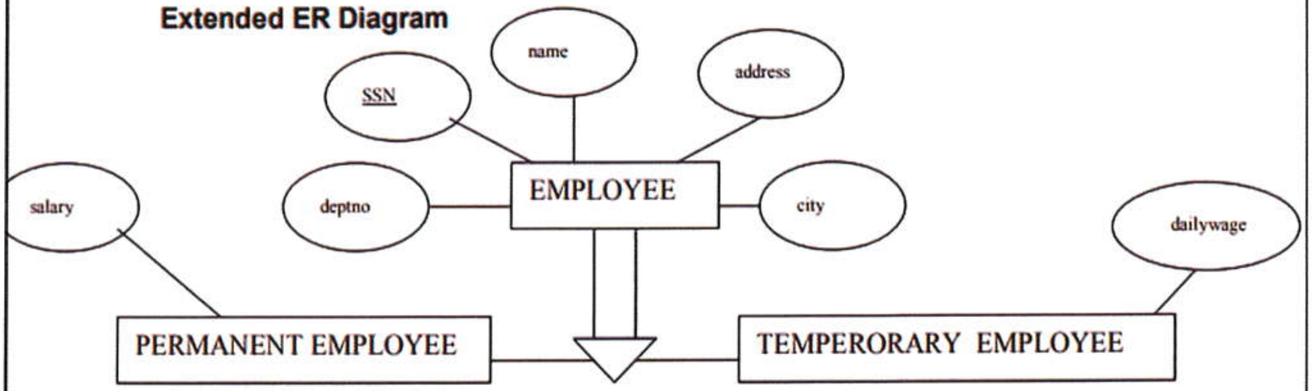


Total Participation of entity set in relationship: The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.

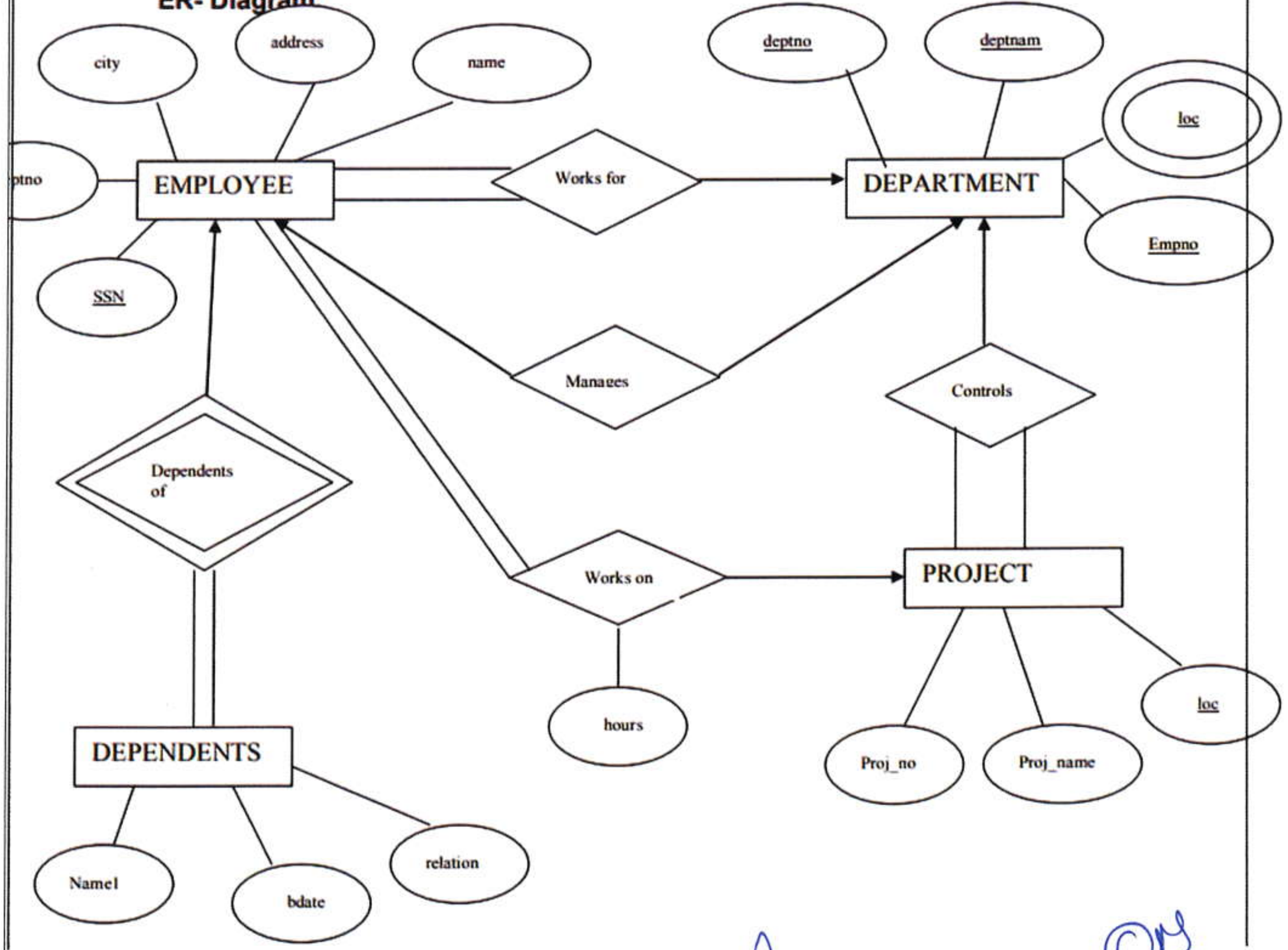
[illegible] [illegible]

[illegible] [illegible] Akumar [illegible]

Extended ER Diagram



ER- Diagram:



[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

Relational Model:

Employee

fname	SSN	address	salary	city	deptno

Department

deptno	deptname	mgr_SSN

Department_Location

deptno	deptloc

Project

projno	projname	location	deptno

Works on

SSN	hours	projname

Dependents

name1	relation	bdate	SSN

Employee

SSN
fname address salary city deptno

Department

deptno
deptname mgr_SSN

Department Location

deptno
deptloc

Project

projno
projname location deptno

Works on

SSN hours projno

Dependents

name1 relation bdate SSN

Conclusion: We have drawn ER model and Relational Model for the same.

[illegible] [illegible] [illegible] Akumar [illegible]

Experiment-2

Aim: Implementation Database

1. Creation of Database with proper constraints (Pk, Fk,.....etc)
2. Insert into database using different types of insert statements
3. Display

Resources used: MS-Access/Visual-FoxPro/SQL-Server/etc

Theory: The set of relations in a database must be specifies to the system by means of a data definition language (DDL). The SQL DDL allows specification of not only a set of relations but also specific information about the relation including:

1. The schema for each relation
2. The domain of values associated with each attribute
3. The integrity constraints
4. The set of indices to be maintained for each relation
5. The security and authorization information for each relation
6. The physical storage structure of each relation on disk

Create Table

create table tab (A_1D_1, A_2D_2, ..., A_nD_n, <integrity constraint-1>, ...<integrity constraint-k>)

where tab is the name of the relation each A_i is the name of the attribute in the schema of relation tab and D_i is the domain type of the values in the domain of attribute A_i. There are a number of different allowable integrity constraints. We specify here only the primary key for the relation.

Insert

A newly created relation is empty initially. We can use the insert command to load data into the relation.

insert into <table name> values (A_1, A_2, ..., A_n)

The values are specified in the order in which the corresponding attributes are listed in the relation schema.

Display

To display the table after creation and insertion we use the following syntax:

*select * from <table name>*

Select clause is used to list the attributes desired in the result of a query. It corresponds to the projection operation of the relational algebra. From clause lists the relations to be scanned in the evaluation of the expression. The asterisk symbol ("*") is used to denote "all attributes".

Conclusion

Thus, we have successfully created the database of company and inserted values in the database.

[illegible] Akumar [illegible] [illegible] [illegible] [illegible]

Experiment-3

Aim: Data Definition (schema) Modification

1. Alter table: add column, remove column, add constraint, remove constraint
2. Drop table
3. Show schema of any table
4. Applying different constraints check, not null, etc.

Resources used: MS-Access/Visual-FoxPro/SQL-Server/etc

Theory: The various command, clauses, functions used for the modification of database are as follows:

(1) Alter table: Alter table command is used to add attribute to an existing relation. All the tuples are assigned to null as the values for the new attribute. The form of the alter table command is

Alter table r add A D

Where, *r* is the name of an existing relation. *A* is the name of the attribute to be added and *D* is the domain of the added attribute. We can drop attribute from a relation by the command:

Alter table r drop A

(2) Update: In certain situation we may wish to change a value in a tuple without changing all values in the tuple. For this purpose, the update statement can be used, as we could for insert and delete. We can choose the tuple be updated by using a query.

```
eg, update EMPLOYEE
    set age=20
    where SSN=514065
```

The preceding update statement is applied only to tuple where SSN=514065. If we want same changes in all tuples, then we write

```
Update EMPLOYEE
set age=20
```

(3) Drop Table: To remove a relation from an SQL database we use the drop table command. The drop table command deletes all information about the dropped relation from the database

drop table r

The relation r and to delete all tuples from r, the following command is used.

delete from r

(4) Adding and Removing Columns: To add a column to an existing relation, we use

```
alter table r
add A D
```

```
eg. alter table EMPLOYEE
    add age int
```

[illegible] Akumar [illegible]

To remove a column from an existing relation we use

Alter table r
drop column A

Eg. alter table EMPLOYEE
drop column age

(5) Not Null: The not null specification prohibits the insertion of a null value. For a attribute any database modification that would cause null to be inserted in an attribute declared to be not null generates an error diagnostic. If an attribute is declared as the primary key then it cannot take a null value.

Eg, alter table EMPLOYEE
alter column salary int NOT NULL

(6) Check: The heck clause in SQL can be applied to relation declarations as well to domain declarations when applied to a relation declaration, the clause check(p) specified a predicate p that must be specified by every tuple in a relation. A common use of the check clause is to ensure that the attribute value satisfy specified condition.

Eg, alter table EMPLOYEE
add constraint em_age
check (age>19)

Conclusion: Thus, we have executed all the queries required for the modification of database.

Experiment-4

Aim: Simple SQL queries (Single table retrieval)

1. Make use of different operators (relational, logical etc.)
2. Selection of rows and columns, renaming columns, use of distinct keyword
3. String handling (%, etc.)
4. Update statement, case update
5. Delete, cascade delete (if possible)

Resources used: MS-Access/Visual-FoxPro/SQL-Server/etc

Theory:

1. Select clause: Select clause is used to list the attributes desired in the result of a query. It corresponds to the projection operation of the relational algebra:

*Eg. select *from EMPLOYEE*
-all attributes
select fname, SSN from EMPLOYEE
-only fname and SSN

2. from clause: From clause lists the relations to be scanned in the evaluation of the expansion.

3.where clause: The where clause corresponds to the selection predicate of the relational algebra. It consists of a predicate involving attribute of the relations that appear in the from clause.

(i) ***and:*** *and* clause is used when we want a result and all the conditions are satisfied in the where clause.

True and unknown = true
False and unknown = unknown
Unknown or unknown = unknown

[illegible] [illegible] Akumar [illegible] [illegible] [illegible] *(ii)* ***as*** *(Rename operator):* SQL provides a mechanism for renaming both relations and attributes. It uses the as clause taking the form

old_name as new_name

(iii) ***distinct:*** If we want to eliminate duplicates, we use the keyword distinct in the aggregation expression.

```
eg. select distinct salary
    from EMPLOYEE
```

(iv) ***String operations:*** The most commonly used operations on strings are pattern matching using the operation like we describe the patterns by using the two special characters % and _.

%: The % character matches any substring
_: The character matches any character

eg, 'Perry%' matches any string beginning with "Perry".
'%idge%' matches any string containing "idge" as substring
"_ _ _" matches any string of exactly three characters
"_ _ _ %" matches any string of at least three characters

(v) ***Update and Case Update:*** In certain situations, we may wish to change a value in a tuple without changing all the values in the tuple. For this purpose, the update statement can be used.

```
eg. update EMPLOYEE
    set age=20
    where SSN=514065
```

SQL provides a case construct which we can use to perform both the update with a single update statement avoiding the problem with the order of updates.

```
eg. update account
    set balance =case
    when balance<=1000
    then balance*1.05
    else balance*1.06
    end
```

(vi) ***delete:*** To delete a tuple from relation r, we use the following command

delete from r

where, r is the name of the relation

Conclusion: Thus, we have executed simple queries in SQL.

Experiment-5

Aim: Advanced SQL Queries-1

1. Group by, having clause, aggregate function
2. Set operations like union, union all and use of order by clause
3. Nested queries: in, not_in, exists, not exists and any, all

Resources used: MS-Access/Visual-FoxPro/SQL-Server/etc

[illegible] [illegible] Akumar [illegible] [illegible] [illegible]

Theory:

*1. **Group by clause:*** These are circumstances where we would like to apply the aggregate functions to a single set of tuples but also to a group of sets of tuples, we would like to specify this wish in SQL using the group by clause. The attributes or attributes given by the group by clause are used to form groups. Tuples with the same value on all attributes in the group by clause placed in one group:

eg.

```
select dept_no, avg(sal) as avg_sal
from EMPLOYEE
group by dept_no
```

*2. **Having clause:*** A having clause is like a where clause but only applies only to groups as a whole whereas the where clause applies to the individual rows. A query can contain both where clause and a having clause. In that case

a. The where clause is applied first to the individual rows in the tables or table structures objects in the diagram pane. Only the rows that meet the conditions in the where clause are grouped.

b. The having clause is then applied to the rows in the result set that are produced by grouping. Only the groups that meet the having conditions appear in the query output.

eg.

```
select dept_no from EMPLOYEE
group_by dept_no
having avg (salary) >=all
(select avg (salary)
   from EMPLOYEE
   group by dept_no)
```

*3. **Aggregate functions:*** Aggregate functions such as SUM, AVG, count, count (*), MAX and MIN generate summary values in query result sets. An aggregate functions (with the exception of count (*) processes all the selected values in a single column to produce a single result value:

eg.

```
select dept_no, count (*)
from EMPLOYEE
group by dept_no
```

eg.

```
select max(salary) as maximum
from EMPLOYEE
```

eg.

```
select sum(salary) as total_salary
from EMPLOYEE
```

eg.

```
select min(salary) as minsal
from EMPLOYEE
```

*4. **Union and Union Operators:*** Combines the result of two or more queries into a single result set consisting of all the rows belonging to all queries in the union. This is different from using joins that combine columns from two tables. Two basic rules for combining the result sets of two queries with union are:

A. The number and the order of the columns must be identical in all queries.

B. The data types must be compatible:

```
select max(salary) as maximum
from EMPLOYEE
union
select min(salary)
from EMPLOYEE
```

[illegible] Akumar [illegible]

union

Specifies that multiple result two or more queries into a single result set consisting of all the rows belonging to all queries into single result set consisting of all the rows belonging to all queries in the union. This is different from using joins that combine columns from two tables. Two basic rules are followed.

5. ***Order by clause:*** SQL allows the user to order the tuples in the result set of the query of a query by the values of one or more attributes using the order by clause. The default order is in the increasing order of values. We can specify the keyword DES if we want values in descending order.
6. ***Exists and not exists:*** Subqueries introduced with exists and not queries can be used for two seet theory operations: Intersection and Difference. The intersection of two sets contains all elements that belong to both of the original sets. The difference contains elements that belong to only first of the two sets.

   eg.
   ```
   select *from DEPARTMENT
   where exists (select * from PROJECT
   where DEPARTMENT.dept_no=PROJECT.dept_no)
   ```
7. ***IN and NOT IN:*** SQL allows testing tuples for membership in a relation. The "IN" connective tests for set membership where the set is a collection of values produced by select clause. The "NOT IN" connective tests for the absence of set membership. The IN and NOT IN connectives can also be used on enumerated sets.

   eg.
   ```
   select proj_name from PROJECT
   where dept_no not in (select dept_no from DEPARTMENT
   where dept_name="chemistry")
   ```
   eg.
   ```
   select fname from EMPLOYEE
   where SSN in (select mgr_SSN from DEPARTMENT)
   ```

Conclusion: Thus, we have studied and executed all the queries mentioned using various clauses.

Experiment-6

Aim: Advanced SQL Queries -2.

(1) Join (Inner & Outer)
(2) Exists & Union

Resources used: MS-Access/Visual-FoxPro/SQL-Server/etc

Theory:

JOINS: SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables.

Type of JOIN:

- **Equi Joins:**

This operation allows to connect, with a relation of equality, the tables which have at least a common attribute. One must have *n-1* conditions of join, *n* being the number of tables which intervene in the query.

If no condition of join is specified, the corresponding query will realize the Cartesian product of the implied tables.

Syntax:
```
SELECT TABLE1.col1, TABLE1.col2...
TABLE2.col1, TABLE2.col2...
```

[illegible] [illegible] [illegible] Akumar [illegible]

```
FROM table_name1, table_name2
WHERE table_name1.col1 = table_name2.col2
```

TYPE OF Equi-Joins:

An equi-join is further classified into two categories:

(a) Inner Join
(b) Outer Join

(a) Inner Join:

The INNER JOIN keyword return rows when there is at least one match in both tables.

Syntax:

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

(b) Outer Joins:

The outer join is returning all the rows returned by simple join or equijoin as well as those rows from one table that do not match any row from the other table, the symbol (+) represents outer join. the outer table operator can appear only on side of the expression.

Type of Outer Joins:

- **Left OUTER JOIN:** Return all rows from the left table, even if there are no matches in the right table.
  Syntax:
  ```
  SELECT TABLE1.column.....
  TABLE2.column.....
  FROM table_name1, table_name2
  WHERE table_name1.column(+) = table_name2.column;
  ```
- **Right OUTER JOIN:** Return all rows from the right table, even if there are no matches in the left table.
  Syntax:
  ```
  SELECT TABLE1.column.....
  TABLE2.column.....
  FROM table_name1,table_name2
  WHERE table_name1.column = table_name2.column(+);
  ```

EXISTS

EXISTS uses a subquery as a condition, where the condition is True if the subquery returns any rows, and False if the subquery does not return any rows.

Syntax:

```
SELECT columns
FROM tables
WHERE EXISTS (subquery);
```

UNION

There are occasions where you might want to see the results of multiple queries together, combining their output; use UNION.

The SQL UNION operator combines two or more SELECT statements.

Syntax:

```
SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2
```

Notice that SQL requires that the Select list (of columns) must match, column-by-column, in data type This concept is useful in situations where a primary key is related to a foreign key, but the foreign key value for some primary keys is NULL. For example, in one table, the primary key is a salesperson, and in another table is customers, with

[illegible]

their salesperson listed in the same row. However, if a salesperson has no customers, that person's name won't appear in the customer table.

Conclusion: Thus, we have studied and executed all the queries mentioned using various clauses.

Experiment-7

Aim: Implementation of views.

1. *Creation of views*
2. *Usage of views*
3. *Creation of views using views*
4. *Drop view*

Resources used: MS-Access/Visual-FoxPro/SQL-Server/etc

Theory:

Views: Any relation that is not part of any logical model but is made visible to the user as a virtual relation is called as a view. It is possible to support a large number of views on the top of any given set of actual database relation. Views help in 2 ways:

1. For security purpose
2. Create a personalized collection of relation that is better user's intuition than is logical model

Creation of Views:

1. Views is defined using 'create view' command
2. To define a view we must give the view a better name and must state the query that computes the view.

   Syntax:

   create view<view name> as <query expression>

   Where query expression is any legal query expression.
3. Once we have defined a view, we can use the view name to refer to the virtual relation that the view generation.
4. Attribute name of the view can be specified explicitly as:

   Create view $V(VA_1, VA_2, \ldots\ldots VA_n)$ *as select* $(A_1, A_2, \ldots\ldots, A_n)$ *from* R_1 *where(p)*

   where, p: predicate
   R_1: relation
   A_1-A_n: attribute of view
   V: view name

Creation of views using VIEW:

Since, view relations may appear in any place that a relation name may appear, except for restrictions on the use of views in update expressions. Thus, one view may be used in the expression defining another view. For eg. Let Emp_work_info is a view with attribute F_name, SSN, Project_no, Work_hrs. Then creation of other view can be done as:

create view new_view
select f_name, work_hrs
from emp_work_info

Updating of views

Although views are useful for the queries, they present a serious problem. If we express updates insertion or deletion on view as the modification done to the database in terms of the views must be translated to a modification to actual relations in the logical methods of database,

Drop view

A view creates earlier can be dropped using 'Drop View' command

[illegible] [illegible] [illegible] [illegible] [illegible]

Syntax:

Drop view 'r'

where, r: View Name.

It deletes all the information about view from the database.

Conclusion: Thus, we have concluded the project by studying and implementing the concept of views in SQL

Suggestion Books:

- Dr Rajeev Chopra, "Database Management System (DBMS) A Practical Approach", 2010, S Chand
- Jitendra Patel, "DBMS Lab Manual" Kindle Edition, 2012

Suggestive digital platform web links

- https://gfgc.kar.nic.in/raibag/FileHandler/270-101d616b-255a-4add-8d9b-dd2e22fec7c1.pdf
- https://pesitsouth.pes.edu/pdf/2019/July/CS/LM_DBMS%20LAB.pdf

Scheme of Marks:

Maximum Marks: 100		
Internal Assessment :	Class Interaction / Quiz Attendance Assignments (Charts / Model Seminar / Rural Service / Technology Dissemination / Report of Excursion / Lab Visits / Survey / Industrial visit)	**30**
External Assessment:	Viva Voce on Practical Practical Record File Table Work / Experiments	**70**
		Total 100

[illegible] [illegible] [illegible] Akumar [illegible] [illegible]