# Raj Kumar Goel Institute of Technology, Ghaziabad


# LABORATORY MANUAL


| | | | |
|---|---|---|---|
| **Faculty Name** | : Neha | **Department** | : CSE |
| **Course Name** | : DBMS Lab | **Course Code** | : KCS-501 |
| **Year/Sem** | : $3^{rd}/5^{th}$ | **NBA Code** | : C306 |
| **Email ID** | : nehasfcs@rkgit.edu.in | **Academic Year** | : 2022-23 |


## Department of Computer Science & Engineering

## VISION OF THE INSTITUTE

To continually develop excellent professionals capable of providing sustainable solutions to challenging problems in their fields and prove responsible global citizens.

## MISSION OF THE INSTITUTE

We wish to serve the nation by becoming a reputed deemed university for providing value based professional education.

## VISION OF THE DEPARTMENT

To produce employable, self disciplined and competent IT professional capable of providing sustainable solution to problems in the field of Information Technology.

## MISSION OF THE DEPARTMENT

To provide quality knowledge in field of Information Technology and make the students capable to serve the industry and society by their innovative ideas and skills and lead to the contribution towards the overall progress and development over globe.

## PROGRAM EDUCATIONAL OUTCOMES (PEOs)

**PEO 1: Learning:** Our graduates to be competent with sound knowledge in field of Computer Science & Engineering.

**PEO 2: Employable:** To develop the ability among students to synthesize data and technical concepts for application to software product design for successful careers that meet the needs of Indian and multinational companies.

**PEO 3: Innovative:** To develop research oriented analytical ability among students to prepare them for making technical contribution to the society.

**PEO 4: Entrepreneur / Contribution:** To develop excellent leadership quality among students which they can use at different levels according to their experience and contribute for progress and development in the society.

## PROGRAM OUTCOMES (POs)

**Engineering Graduates will be able to:**

**PO1: Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2: 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.

**PO3: 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.

**PO4: 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.

**PO5: Modern tool usage**: Create, select, and apply appropriate techniques, resources, a n d modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

**PO6: 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.

**PO7: Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8: Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9: Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10: 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.

**PO11: 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 multidisciplinary environments. team, to manage projects and in

**PO12: Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## PROGRAM SPECIFIC OUTCOMES (PSOs)

**PSO1:** The ability to use standard practices and suitable programming environment to develop software solutions.

**PSO2:** The ability to employ latest computer languages and platforms in creating innovative career opportunities

## COURSE OUTCOMES (COs)

| | |
|---|---|
| **C306.1** | Understand and apply oracle 11 g products for creating tables, views, indexes, sequences and other database objects. |
| **C306.2** | Design and implement a database schema for company data base, banking data base, library information system, payroll processing system, student information system. |
| **C306.3** | Write and execute simple and complex queries using DDL, DML, DCL and TCL. |
| **C306.4** | Write and execute PL/SQL blocks, procedure functions, packages and triggers, cursors. |
| **C306.5** | Enforce entity integrity, referential integrity, key constraints, and domain constraints on database. |

## CO-PO MAPPING

| | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C306.1** | 3 | 2 | 2 | 1 | | 1 | | | | | | 1 |
| **C306.2** | 2 | 2 | 2 | 1 | | 1 | | | | | | 1 |
| **C306.3** | 2 | 1 | 1 | 1 | | 1 | | | | | | 1 |
| **C306.4** | 2 | 1 | 1 | 1 | | 1 | | | | | | 1 |
| **C306.5** | 2 | 2 | 2 | 1 | | 2 | | | | | 1 | 1 |
| **C307** | **2.2** | **1.6** | **1.6** | **1** | | **1.2** | | | | | **1** | **1** |

## CO-PSO MAPPING

| | PSO1 | PSO2 |
|---|---|---|
| **C306.1** | 2 | |
| **C306.2** | 2 | |
| **C306.3** | 2 | |
| **C306.4** | 2 | |
| **C306.5** | 2 | |
| **C306** | 2 | |

# LIST OF EXPERIMENTS

| Expt. No. | Title of experiment | Corresponding CO |
|---|---|---|
| **1.** | Creating tables and writing Queries in SQL. | C 306.1 |
| **2.** | To implement various DML Operations on table | C 306.1 |
| **3.** | To Implement the restrictions/constraints on the table | C 306.2 |
| **4.** | To Alter the structure of the table | C 306.2 |
| **5.** | To implement the concept of Joins | C 306.2 |
| **6.** | To implement the concept of grouping of Data | C 306.2 |
| **7.** | To implement the concept of Sub Queries | C 306.2 |
| **8.** | To implement the concept of Indexes and views. | C 306.2 |
| **9.** | To implement the basics of PL/SQL. | C 306.2 |
| **10.** | To implement the concept of Cursor and Trigger | C 306.4 |

| | Content Beyond Syllabus | |
|---|---|---|
| 1. | **Write a program to implement REPORTS.** | C 306.4 |
| **2.** | **Write a program to implement FORMS.** | C 306.5 |

# INTRODUCTION

A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.

The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

Computer scientists may classify database-management systems according to the database models that they support. Relational databases became dominant in the 1980s. These model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data. In the 2000s, non-relational databases became popular, referred to as NoSQL because they use different query languages.

A database has broad searching functionality. For example, a sales department could quickly search for and find all sales personnel who had achieved a certain amount of sales over a particular time period.

A database can update records in bulk – even millions or more records. This would be useful, for example, if you wanted to add new columns or apply a data patch of some sort.

If the database is relational, which most databases are, it can cross-reference records in different tables. This means that you can create relationships between tables. For instance, if you linked a Customers table with an Orders table, you could find all purchase orders from the Orders table that a single customer from the Customers table ever processed, or further refine it to return only those orders processed in a particular time period – or almost any type of combination you could imagine.

A database can perform complex aggregate calculations across multiple tables. For example, you could list expenses across multiple retail outlets, including all possible sub-totals, and then a final total.

A database can enforce consistency and data integrity, which means that it can avoid duplication and ensure data accuracy through its design and a series of constraints.

# PREFACE

Structure Query Language (SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.On the surface, a database might seem much like a spread sheet; it has data arranged in columns and rows. But that is where the similarity ends because a database is far more powerful.

Dr. Amit Singhal
Ms. Neha
Ms. Anjali Yadav
Ms. Ahimsha

# DO'S AND DONT'S

## DO's

1. Conform to the academic discipline of the department.
2. Enter your credentials in the laboratory attendance register.
3. Read and understand how to carry out an activity thoroughly before coming to the laboratory.
4. Ensure the uniqueness with respect to the methodology adopted for carrying out the experiments.
5. Shut down the machine once you are done using it.

## DONT'S

1. Eatables are not allowed in the laboratory.
2. Usage of mobile phones is strictly prohibited.
3. Do not open the system unit casing.
4. Do not remove anything from the computer laboratory without permission.
5. Do not touch, connect or disconnect any plug or cable without your faculty/laboratory Technicians permission.

# GENERAL SAFETY INSTRUCTIONS

1. Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.

2. Report fire or accidents to your faculty /laboratory technician immediately.

3. Report any broken plugs or exposed electrical wires to your faculty/laboratory technician immediately.

4. Do not plug in external devices without scanning them for computer viruses.

# GUIDELINES FOR LABORTORY RECORD PREPARATION

While preparing the lab records, the student is required to adhere to the following guidelines:

## Contents to be included in Lab Records:

1. Cover page
2. Vision
3. Mission
4. PEOs
5. POs
6. PSOs
7. COs
8. CO-PO-PSO mapping
9. Index
10. Experiments
    - Aim
    - Source code
    - Input-Output

A separate copy needs to be maintained for pre-lab written work.

The student is required to make the Lab File as per the format given on the next two pages.

# Raj Kumar Goel Institute of Technology, Ghaziabad


# DBMS Lab (KCS 551)

| | |
|---|---|
| **Name** | |
| **Roll No.** | |
| **Section- Batch** | |

# INDEX

| Experiment No. | Experiment Name | Date of Conduction | Date of Submission | Faculty Signature |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# GUIDELINES FOR ASSESSMENT

Students are provided with the details of the experiment (Aim, pre-experimental questions, procedure etc.) to be conducted in next lab and are expected to come prepared for each lab class.

Faculty ensures that students have completed the required pre-experiment questions and they complete the in-lab programming assignment(s) before the end of class. Given that the lab programs are meant to be formative in nature, students can ask faculty for help before and during the lab class.

Students‟ performance will be assessed in each lab based on the following Lab Assessment Components:

**Assessment Criteria-1:** Performance (Max. marks = 5)

**Assessment Criteria-2:** VIVA (Max. marks = 5)

**Assessment Criteria-3:** Record (Max. marks = 5)

In each lab class, students will be awarded marks out of 5 under each component head, making it total out of 15 marks.

# Experiment No.1

# Theory and Concept

Objective: Creating tables and writing Queries in SQL.

**Theory & Concepts:**

**Introduction about SQL-**

SQL (Structured Questionry Language) is a nonprocedural language, you specify what you want, not how to get it. A block structured format of English key words is used in this Questionry language. It has the following components.

**DDL (Data Definition Language)-**

The SQL DDL provides command for defining relation schemas, deleting relations and modifying relation schema.

**DML (DATA Manipulation Language)-**

It includes commands to insert tuples into, delete tuples from and modify tuples in the database.

**View definition-**

The SQL DDL includes commands for defining views.
Transaction Control- SQL includes for specifying the beginning and ending of transactions.

**Embedded SQL and Dynamic SQL-**

Embedded and Dynamic SQL define how SQL statements can be embedded with in general purpose programming languages, such as C, C++, JAVA, COBOL, Pascal and Fortran.

**Integrity-**

The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must specify. Updates that violate integrity constraints are allowed.

**Authorization-**

The SQL DDL includes commands for specifying access rights to relations and views.

**Data Definition Language-**

The SQL DDL allows specification of not only a set of relations but also information about each relation, including-

- Schema for each relation
- The domain of values associated with each attribute.
- The integrity constraints.
- The set of indices to be maintained for each relation.
- The security and authorization information for each relation.
- The physical storage structure of each relation on disk.

**Domain types in SQL-**

**The SQL standard supports a variety of built in domain types, including-**

- Char (n)- A fixed length character length string with user specified length .

- Varchar (n)- A variable character length string with user specified maximum length n.
- Int- An integer.
- Small integer- A small integer.
- Numeric (p, d)-A Fixed point number with user defined precision.
- Real, double precision- Floating point and double precision floating point numbers with machine dependent precision.
- Float (n)- A floating point number, with precision of at least n digits.
- Date- A calendar date containing a (four digit) year, month and day of the month.
- Time- The time of day, in hours, minutes and seconds Eg. Time '09:30:00'.
- Number- Number is used to store numbers (fixed or floating point).

**DDL statement for creating a table-**

**Syntax-**

Create table tablename
(columnname datatype(size), columnname datatype(size));

Creating a table from a table-

**Syntax-**

CREATE TABLE TABLENAME
[(columnname, columnname, ……)]
AS SELECT columnname, columnname……..FROM tablename;

**Insertion of data into tables-**

**Syntax-**

INSERT INTO tablename
[(columnname, columnname, ……)]
Values(expression, expression);

**Inserting data into a table from another table:**

**Syntax-**

INSERT INTO tablename
SELECT columnname, columnname, …….
FROM tablename;

**Insertion of selected data into a table from another table:**

**Syntax-**

INSERT INTO tablename
SELECT columnname, columnname……..
FROM tablename
WHERE columnname= expression;

**Retrieving of data from the tables-**

**Syntax-**

SELECT * FROM tablename;

**The retrieving of specific columns from a table-**

**Syntax-**

SELECT columnname, columnname, ….
FROM tablename;

**Elimination of duplicates from the select statement-**

**Syntax-**

SELECT DISTINCT columnname, columnname
FROM tablename;

**Selecting a data set from table data-**

**Syntax-**

SELECT columnname, columnname
FROM tablename
WHERE searchcondition;

**Q1. Create the following tables:**

**i) client_master**

| columnname | datatype | size |
|---|---|---|
| client_no | varchar2 | 6 |
| name | varchar2 | 20 |
| address1 | varchar2 | 30 |
| address2 | varchar2 | 30 |
| city | varchar2 | 15 |
| state | varchar2 | 15 |
| pincode | number | 6 |
| bal_due | number | 10,2 |

**ii) Product_master**

| Columnname | datatype | size |
|---|---|---|
| Product_no | varchar2 | |
| Description | varchar2 | |
| Profit_percent | number | |
| Unit_measure | varchar2 | |
| Qty_on_hand | number | |
| Reoder_lvl | number | |
| Sell_price | number | |
| Cost_price | number | |

**Q2- Insert the following data into their respective tables:**

| Clientno | Name | city | pincode | state | bal.due |
|---|---|---|---|---|---|
| 0001 | Ivan | Bombay | 400054 | Maharashtra | 15000 |
| 0002 | Vandana | Madras | 780001 | Tamilnadu | 0 |
| 0003 | Pramada | Bombay | 400057 | Maharashtra | 5000 |
| 0004 | Basu | Bombay | 400056 | Maharashtra | 0 |
| 0005 | Ravi | Delhi | 100001 | | 2000 |
| 0006 | Rukmini | Bombay | 400050 | Maharashtra | 0 |

**Data for Product Master:**

| Product No. | Desciption | Profit % Percent | Unit measured | Qty on hand | Reorder lvl | Sell price | Cost price |
|---|---|---|---|---|---|---|---|
| P00001 | 1.44floppies | 5 | piece | 100 | 20 | 525 | 500 |
| P03453 | Monitors | 6 | piece | 10 | 3 | 12000 | 11200 |
| P06734 | Mouse | 5 | piece | 20 | 5 | 1050 | 500 |
| P07865 | 1.22 floppies | 5 | piece | 100 | 20 | 525 | 500 |
| P07868 | Keyboards | 2 | piece | 10 | 3 | 3150 | 3050 |
| P07885 | CD Drive | 2.5 | piece | 10 | 3 | 5250 | 5100 |
| P07965 | 540 HDD | 4 | piece | 10 | 3 | 8400 | 8000 |
| P07975 | 1.44 Drive | 5 | piece | 10 | 3 | 1050 | 1000 |
| P08865 | 1.22 Drive | 5 | piece | 2 | 3 | 1050 | 1000 |

**Q3:- On the basis of above two tables answer the following Queries:**

i) Find out the names of all the clients.
ii) Retrieve the list of names and cities of all the clients.
iii) List the various products available from the product_master table.
iv) List all the clients who are located in Bombay.
v) Display the information for client no 0001 and 0002.
vi) Find the products with description as ‘1.44 drive’ and ‘1.22 Drive’.
vii) Find all the products whose sell price is greater than 5000.
viii) Find the list of all clients who stay in city ‘Bombay’ or city ‘Delhi’ or ‘Madras’.
ix) Find the product whose selling price is greater than 2000 and less than or equal to 5000.
x) List the name, city and state of clients not in the state of ‘Maharashtra’.

# Experiment No.2

# Theory and Concept

**Objective:- To implement various DML Operations on table.**
DML ( Data Manipulation Language) Data manipulation is

- The retrieval of information stored in the database.
- The insertion of new information into the database.
- The deletion of information from the database.
- The modification of information stored by the appropriate data model. There are basically two types.

(i) **Procedural DML**:- require a user to specify what data are needed and how to get those data.

(ii) **Non Procedural DML** : require a user to specify what data are needed without specifying how to get those data.

**Updating the content of a table:**
In creation situation we may wish to change a value in table without changing all values in the tuple . For this purpose the update statement can be used.

Update table name
Set columnname = experision, columnname =expression……
Where columnname = expression;

**Deletion Operation:-**
A delete reQuestionst is expressed in much the same way as Questionry. We can delete whole tuple ( rows) we can delete values on only particulars attributes.

**Deletion of all rows**

**Syntax:**
Delete from tablename :

**Deletion of specified number of rows**
**Syntax:**

Delete from table name
Where search condition ;

**Computation in expression lists used to select data**

| | | | |
|---|---|---|---|
| + | Addition | - | Subtraction |
| * | multiplication | ** | exponentiation |
| / | Division | () | Enclosed operation |

Renaming columns used with Expression Lists: - The default output column names can be renamed by the user if required

**Syntax:**

Select column name result_columnname,
Columnname result_columnname,
From table name;

**Logical Operators:**
The logical operators that can be used in SQL sentenced are

| | |
|---|---|
| AND | all of must be included |
| OR | any of may be included |
| NOT | none of could be included |

**Range Searching:** Between operation is used for range searching.

**Pattern Searching:**
The most commonly used operation on string is pattern matching using the operation 'like' we describe patterns by using two special characters.

- Percent (%) ; the % character matches any substring we consider the following examples.
- 'Perry %' matches any string beginning with perry
- '% idge % matches any string containing' idge as substring.
- ' - - - ' matches any string exactly three characters.
- ' - - - % matches any string of at least of three characters.

**Oracle functions:**
Functions are used to manipulate data items and return result. function follow the format of function _name (argument1, argument2 ..) .An arrangement is user defined variable or constant. The structure of function is such that it accepts zero or more arguments.
Examples:
Avg return average value of n

**Syntax:**
Avg ([distinct/all]n)
Min return minimum value of expr.

**Syntax:**
MIN((distict/all )expr)
Count Returns the no of rows where expr is not null
**Syntax:**
Count ([distinct/all)expr]
Count (*) Returns the no rows in the table, including duplicates and those with nulls.
Max Return max value of expr

**Syntax:**

Max ([distinct/all]expr)
Sum Returns sum of values of n

**Syntax:**
Sum ([distinct/all]n)
**Sorting of data in table**
**Syntax:**
Select columnname, columnname
From table
Order by columnname;

**Question.1 Using the table client master and product master answer the following Queries.**

i. Change the selling price of '1.44 floppy drive to Rs.1150.00
ii. Delete the record with client 0001 from the client master table.
iii. Change the city of client_no'0005' to Bombay.
iv. Change the bal_due of client_no '0001, to 1000.
v. Find the products whose selling price is more than 1500 and also find the new selling price as original selling price *15.
vi. Find out the clients who stay in a city whose second letter is a.
vii. Find out the name of all clients having 'a' as the second letter in their names.
viii. List the products in sorted order of their description.
ix. Count the total number of orders
x. Calculate the average price of all the products.
xi. Calculate the minimum price of products.
xii. Determine the maximum and minimum prices . Rename the tittle as 'max_price' and min_price respectively.
xiii. Count the number of products having price greater than or equal to 1500.

# Experiment No.3

# Theory and Concept

Objective:- To Implement the restrictions/constraints on the table.

**Data constraints:** Besides the cell name, cell length and cell data type there are other parameters i.e. other data constrains that can be passed to the DBA at check creation time. The constraints can either be placed at column level or at the table level.

i. **Column Level Constraints:** If the constraints are defined along with the column definition, it is called a column level constraint.
ii. **Table Level Constraints:** If the data constraint attached to a specify cell in a table reference the contents of another cell in the table then the user will have to use table level constraints.

**Null Value Concepts:-** while creating tables if a row locks a data value for particular column that value is said to be null . Column of any data types may contain null values unless the column was defined as not null when the table was created

**Syntax:**

**Create table tablename**
**(**columnname data type (size) not null ……)

**Primary Key:** primary key is one or more columns is a table used to uniquickly identity each row in the table. Primary key values must not be null and must be uniQuestion across the column. A multicolumn primary key is called composite primary key.

**Syntax: primary key as a column constraint**
Create table tablename
(columnname datatype (size) primary key,….)
**Primary key as a table constraint**
Create table tablename
(columnname datatype (size), columnname datatype( size)…
Primary key (columnname,columnname));

**UniQuestion key concept:-**A uniQuestion is similar to a primary key except that the purpose of a uniQuestion key is to ensure that information in the column for each record is uniQuestion as with telephone or devices license numbers. A table may have many uniQuestion keys.

**Syntax: UniQuestion as a column constraint**.
Create table table name
(columnname datatype (size) uniQuestion);
**UniQuestion as table constraint:**
Create table tablename
(columnname datatype (size),columnname datatype (size)…uniQuestion (columnname,columnname));

**Default value concept: At** the line of cell creation a default value can be assigned to it. When the user is loading a record with values and leaves this cell empty, the DBA wil automatically

load this cell with the default value specified. The data type of the default value should match the data type of the column

**Syntax:**

Create table tablename
(columnname datatype (size) default value,….);

**Foreign Key Concept :** Foreign key represents relationship between tables. A foreign key is column whose values are derived from the primary key of the same of some other table . the existence of foreign key implies that the table with foreign key is related to the primary key table from which the foreign key is derived .A foreign key must have corresponding primary key value in the primary key table to have meaning.
Foreign key as a column constraint

**Syntax :**

Create table table name
(columnname datatype (size) references another table name);

**Foreign key as a table constraint:**

**Syntax :**
Create table name
(columnname datatype (size)….
primary key (columnname);
foreign key (columnname)references table name);

**Check Integrity Constraints:** Use the check constraints when you need to enforce intergrity rules that can be evaluated based on a logical expression following are a few examples of appropriate check constraints.

- A check constraints name column of the coient_master so that the name is entered in upper case.
- A check constraint on the client_no column of the client _master so that no client_no value starts with 'c'

**Syntax:**

Create table tablename
(columnname datatype (size) CONSTRAINT constraintname)
Check (expression));

**Question.2 Create the following tables:**

i. **Sales_master**

| **Columnname** | **Datatype** | **Size** | **Attributes** |
|---|---|---|---|
| Salesman_no | varchar2 | 6 | Primary key/first letter must start with 's' |
| Sal_name | varchar2 | 20 | Not null |
| Address | varchar2 | | Not null |
| City | varchar2 | 20 | |
| State | varchar2 | 20 | |

| | | | |
|---|---|---|---|
| Pincode | Number | 6 | |
| Sal_amt | Number | 8,2 | Not null, cannot be 0 |
| Tgt_to_get | Number | 6,2 | Not null, cannot be 0 |
| Ytd_sales | Number | 6,2 | Not null, cannot be 0 |
| Remarks | Varchar2 | 30 | |

**ii. Sales_order**

| **Columnname** | **Datatype** | **Size** | **Attributes** |
|---|---|---|---|
| S_order_no | varchar2 | 6 | Primary/first letter must be 0 |
| S_order_date | Date | 6 | Primary key reference clientno of client_master table |
| Client_no | Varchar2 | 25 | |
| Dely_add | Varchar2 | 6 | |
| Salesman_no | Varchar2 | 6 | Foreign key references salesman_no of salesman_master table |
| Dely_type | Char | 1 | Delivery part(p)/full(f),default f |
| Billed_yn | Char | 1 | |
| Dely_date | Date | | Can not be lessthan s_order_date |
| Order_status | Varchar2 | 10 | Values (‘in process’;’fulfilled’;back order’;’canceled |

**I. Sales_order_details**

| **Column** | **Datatype** | **Size** | **Attributes** |
|---|---|---|---|
| S_order_no | Varchar2 | 6 | Primary key/foreign key references s_order_no of sales_order |
| Product_no | Varchar2 | 6 | Primary key/foreign key references product_no of product_master |
| Qty_order | Number | 8 | |
| Qty_disp | Number | 8 | |
| Product_rate | Number | 10,2 | |

Insert the following data into their respective tables using insert statement:

**Data for sales_man master table**

| **Salesman_no** | **Salesman name** | **Address** | **City** | **Pin code** | **State** | **Salamt** | **Tgt_to_get** | **Ytd Sales** | **Remark** |
|---|---|---|---|---|---|---|---|---|---|
| 500001 | Kiran | A/14 worli | Bombay | 400002 | Mah | 3000 | 100 | 50 | Good |
| 500002 | Manish | 65,nariman | Bombay | 400001 | Mah | 3000 | 200 | 100 | Good |
| 500003 | Ravi | P-7 Bandra | Bombay | 400032 | Mah | 3000 | 200 | 100 | Good |
| 500004 | Ashish | A/5 Juhu | Bombay | 400044 | Mah | 3500 | 200 | 150 | Good |

**Data for salesorder table:**

| S_orderno | S_orderdate | Client no | Dely type | Bill yn | Salesman no | Delay date | Orderstatus |
|---|---|---|---|---|---|---|---|
| 019001 | 12-jan-96 | 0001 | F | N | 50001 | 20-jan-96 | Ip |
| 019002 | 25-jan-96 | 0002 | P | N | 50002 | 27-jan-96 | C |
| 016865 | 18-feb-96 | 0003 | F | Y | 500003 | 20-feb-96 | F |
| 019003 | 03-apr-96 | 0001 | F | Y | 500001 | 07-apr-96 | F |
| 046866 | 20-may-96 | 0004 | P | N | 500002 | 22-may-96 | C |
| 010008 | 24-may-96 | 0005 | F | N | 500004 | 26-may-96 | Ip |

(iii)

**Data for sales_order_details table:**

| S_order no | Product no | Qty ordered | Qty disp | Product_rate |
|---|---|---|---|---|
| 019001 | P00001 | 4 | 4 | 525 |
| 019001 | P07965 | 2 | 1 | 8400 |
| 019001 | P07885 | 2 | 1 | 5250 |
| 019002 | P00001 | 10 | 0 | 525 |
| 046865 | P07868 | 3 | 3 | 3150 |
| 046865 | P07885 | 10 | 10 | 5250 |
| 019003 | P00001 | 4 | 4 | 1050 |
| 019003 | P03453 | 2 | 2 | 1050 |
| 046866 | P06734 | 1 | 1 | 12000 |
| 046866 | P07965 | 1 | 0 | 8400 |
| 010008 | P07975 | 1 | 0 | 1050 |
| 010008 | P00001 | 10 | 5 | 525 |

# Experiment No.4

# Theory and Concept

## Objective:- To Alter the structure of the table

**Modifying the Structure of Tables**-

Alter table command is used to changing the structure of a table. Using the alter table clause you cannot perform the following tasks:

(i) change the name of table
(ii) change the name of column
(iii) drop a column
(iv) decrease the size of a table if table data exists.

The following tasks you can perform through alter table command.

(i) **Adding new columns**:

Syntax
ALTER TABLE tablename
ADD (newcolumnname newdatatype (size));

**(ii) Modifying existing table**

Syntax:
ALTER TABLE tablename
MODIFY (newcolumnname newdatatype (size));

***NOTE*:** Oracle not allow constraints defined using the alter table, if the data in the table, violates such constraints.

**Removing/Deleting Tables**- Following command is used for removing or deleting a table.

Syntax:
DROP TABLE tabename:

Defining Integrity constraints in the ALTER TABLE command-

You can also define integrity constraints using the constraint clause in the ALTER TABLE command. The following examples show the definitions of several integrity constraints.

(1) **Add PRIMARY KEY**-

Syntax:
ALTER TABLE tablename
ADD PRIMARY KEY(columnname);

(2) **Add FOREIGN KEY**-

Syntax:

ALTER TABLE tablename
ADD CONSTRAINT constraintname
FOREIGN KEY(columnname) REFERENCES tablename;

Droping integrity constraints in the ALTER TABLE command:

You can drop an integrity constraint if the rule that if enforces is no longer true or if the constraint is no longer needed. Drop the constraint using the ALTER TABLE command with the DROP clause. The following examples illustrate the droping of integrity constraints.

(1) **DROP the PRIMARY KEY**-
Syntax:
ALTER TABLE tablename
DROP PRIMARY KEY

(2) **DROP FOREIGN KEY**-
Syntax:
ALTER TABLE tablename
DROP CONSTRAINT constraintname;

Question 1. Create the following tables:

**Challan_Header**

| Column name | data type | size | Attributes |
|---|---|---|---|
| Challan_no | varchar2 | 6 | Primary key |
| s_order_no | varchar2 | 6 | Foreign key references s_order_no of sales_order table |
| challan_date | date | | not null |
| billed_yn | char | 1 | values ('Y','N'). Default 'N' |

**Table Name : Challan_Details**

| Column name | data type | size | Attributes |
|---|---|---|---|
| Challan_no | varchar2 | 6 | Primary key/Foreign key references Product_no of product_master |
| Qty_disp | number | 4,2 | not null |

Q2. Insert the following values into the challan header and challan_details tables:

(i)

| Challan No | S_order No | Challan Date | Billed |
|---|---|---|---|
| CH9001 | 019001 | 12-DEC-95 | Y |
| CH865 | 046865 | 12-NOV-95 | Y |
| CH3965 | 010008 | 12-OCT-95 | Y |

Data for challan_details table

| Challan No | Product No | Qty Disp |
|---|---|---|
| CH9001 | P00001 | 4 |
| CH9001 | P07965 | 1 |
| CH9001 | P07885 | 1 |
| CH6865 | P07868 | 3 |
| CH6865 | P03453 | 4 |

| | | |
|---|---|---|
| CH6865 | P00001 | 10 |
| CH3965 | P00001 | 5 |
| CH3965 | P07975 | 2 |

**Objective –** Answer the following Queries

Q1. Make the primary key to client_no in client_master.
Q2. Add a new column phone_no in the client_master table.
Q3. Add the not null constraint in the product_master table with the columns description, profit percent , sell price and cost price.
Q4. Change the size of client_no field in the client_master table.
Q5. Select product_no, description where profit percent is between 20 and 30 both inclusive.

# Experiment No.5

# Theory & Concept

## Objective:- To implement the concept of Joins

**Joint Multiple Table (Equi Join):** Some times we require to treat more than one table as though manipulate data from all the tables as though the tables were not separate object but one single entity. To achieve this we have to join tables.Tables are joined on column that have dame data type and data with in tables.

The tables that have to be joined are specified in the FROM clause and the joining attributes in the WHERE clause.

**Algorithm for JOIN in SQL:**

1. Cartesian product of tables (specified in the FROM clause)
2. Selection of rows that match (predicate in the WHERE clause)
3. Project column specified in the SELECT clause.

**1. Cartesian product:-**

Consider two table student and course

```
Select B.*,P.*
FROM student B, course P;
```

**2. INNER JOIN:**

Cartesian product followed by selection

```
Select B.*,P.*
FROM student B, Course P
WHERE B.course # P.course # ;
```

**3. LEFT OUTER JOIN:**

LEFT OUTER JOIN = Cartesian product + selection but include rows from the left table which are unmatched pat nulls in the values of attributes belonging to th e second table

Exam:

```
Select  B.*,P*
FROM student B left join course p
ON B.course # P.course #;
```

**4. RIGHT OUTER JOIN:**

RIGHT OUTER JOIN = Cartesian product + selection but include rows from right table which are unmatched

```
Exam:
Select B.*,P.*
From student B RIGHT JOIN course P
B.course# = P course # ;
```

**5. FULL OUTER JOIN**

```
Exam
Select B.*,P.*
From student B  FULL JOIN course P
On B.course # = P course # ;
```

**A. Consider the following schema for a Library Database:**

**BOOK (*Book_id, Title, Publisher_Name, Pub_Year*)**

**BOOK_AUTHORS (Book_id, Author_*Name*)**

**PUBLISHER (*Name, Address, Phone*)**

**BOOK_COPIES (*Book_id, Branch_id, No-of_Copies*)**

**BOOK_LENDING (*Book_id, Branch_id, Card_No, Date_Out, Due_Date*)**

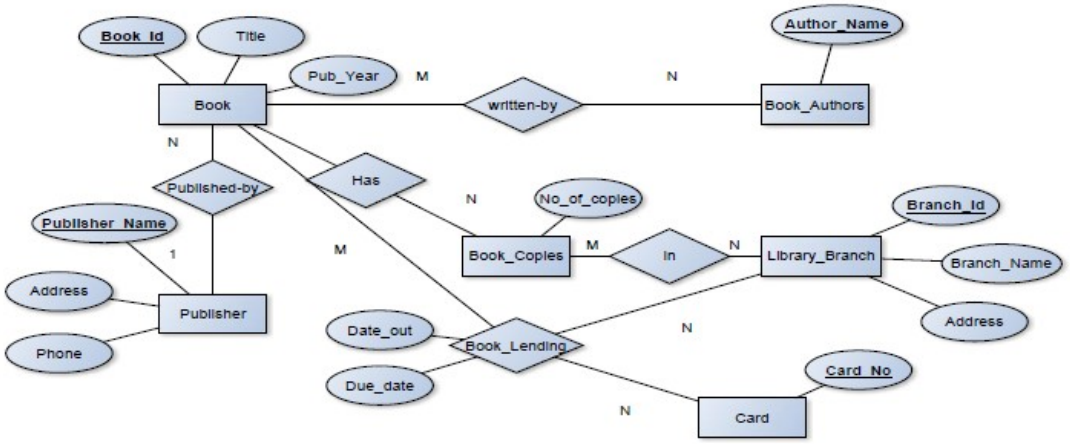
**LIBRARY_BRANCH (*Branch_id, Branch_Name, Address*)**

**Write SQL queries to :**

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.
3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. 4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
5. Create a view of all books and its number of copies that are currently available in the Library.

# EXPERIMENT NO. 6

**Entity-Relationship Diagram**




**Schema Diagram**

*Book*

| *Book_id* | *Title* | *Pub_Year* | *Publisher_Name* |
|---|---|---|---|

*Book_Authors*

| *Book_id* | *Author_name* |
|---|---|

*Publisher*

| *Name* | *Phone_no* | *Address* |
|---|---|---|

*Book_Copies*

| *Book_id* | *Branch_id* | *No_of_Copies* |
|---|---|---|

*Book_Lending*

| *Book_id* | *Branch_id* | *Card_no* | *Date_out* | *Due_date* |
|---|---|---|---|---|

*Library_Branch*

| *Branch_id* | *Address* | *Branch_name* |
|---|---|---|

**Table Creation**

```
CREATE TABLE PUBLISHER
(NAME VARCHAR2 (20) PRIMARY KEY,
PHONE INTEGER,
ADDRESS VARCHAR2 (20));

CREATE TABLE BOOK
(BOOK_ID INTEGER PRIMARY KEY,
TITLE VARCHAR2 (20),
PUB_YEAR VARCHAR2 (20),
PUBLISHER_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);

CREATE TABLE BOOK_AUTHORS
(AUTHOR_NAME VARCHAR2 (20),
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, AUTHOR_NAME));

CREATE TABLE LIBRARY_BRANCH
(BRANCH_ID INTEGER PRIMARY KEY,
BRANCH_NAME VARCHAR2 (50),
ADDRESS VARCHAR2 (50));

CREATE TABLE BOOK_COPIES
(NO_OF_COPIES INTEGER,
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, BRANCH_ID));

CREATE TABLE CARD
(CARD_NO INTEGER PRIMARY KEY);

CREATE TABLE BOOK_LENDING
(DATE_OUT DATE,
DUE_DATE DATE,
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE CASCADE,
CARD_NO REFERENCES CARD (CARD_NO) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, BRANCH_ID, CARD_NO));
```

**Insertion of Values to Tables**

```
INSERT INTO PUBLISHER VALUES (_MCGRAW-HILL‘, 9989076587, _BANGALORE‘);
INSERT INTO PUBLISHER VALUES (_PEARSON‘, 9889076565, _NEWDELHI‘);
INSERT INTO PUBLISHER VALUES (_RANDOM HOUSE‘, 7455679345, _HYDRABAD‘);
INSERT INTO PUBLISHER VALUES (_HACHETTE LIVRE‘, 8970862340, _CHENAI‘);
INSERT INTO PUBLISHER VALUES (_GRUPO PLANETA‘, 7756120238, _BANGALORE‘);

INSERT INTO BOOK VALUES (1,‘DBMS‘,‘JAN-2017‘, _MCGRAW-HILL‘);
INSERT INTO BOOK VALUES (2,‘ADBMS‘,‘JUN-2016‘, _MCGRAW-HILL‘);
INSERT INTO BOOK VALUES (3,‘CN‘,‘SEP-2016‘, _PEARSON‘);
INSERT INTO BOOK VALUES (4,‘CG‘,‘SEP-2015‘, _GRUPO PLANETA‘);
INSERT INTO BOOK VALUES (5,‘OS‘,‘MAY-2016‘, _PEARSON‘);

INSERT INTO BOOK_AUTHORS VALUES (‘NAVATHE‘, 1);
INSERT INTO BOOK_AUTHORS VALUES (‘NAVATHE‘, 2);
INSERT INTO BOOK_AUTHORS VALUES (‘TANENBAUM‘, 3);
```

```
INSERT INTO BOOK_AUTHORS VALUES ('EDWARD ANGEL', 4);
INSERT INTO BOOK_AUTHORS VALUES ('GALVIN', 5);

INSERT INTO LIBRARY_BRANCH VALUES (10,'RR NAGAR','BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (11,'RNSIT','BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (12,'RAJAJI NAGAR', 'BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (13,'NITTE','MANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (14,'MANIPAL','UDUPI');

INSERT INTO BOOK_COPIES VALUES (10, 1, 10);
INSERT INTO BOOK_COPIES VALUES (5, 1, 11);
INSERT INTO BOOK_COPIES VALUES (2, 2, 12);
INSERT INTO BOOK_COPIES VALUES (5, 2, 13);
INSERT INTO BOOK_COPIES VALUES (7, 3, 14);
INSERT INTO BOOK_COPIES VALUES (1, 5, 10);
INSERT INTO BOOK_COPIES VALUES (3, 4, 11);

INSERT INTO CARD VALUES (100);
INSERT INTO CARD VALUES (101);
INSERT INTO CARD VALUES (102);
INSERT INTO CARD VALUES (103);
INSERT INTO CARD VALUES (104);

INSERT INTO BOOK_LENDING VALUES ('01-JAN-17','01-JUN-17', 1, 10, 101);
INSERT INTO BOOK_LENDING VALUES ('11-JAN-17','11-MAR-17', 3, 14, 101);
INSERT INTO BOOK_LENDING VALUES ('21-FEB-17','21-APR-17', 2, 13, 101);
INSERT INTO BOOK_LENDING VALUES ('15-MAR-17','15-JUL-17', 4, 11, 101);
INSERT INTO BOOK_LENDING VALUES (_12-APR-17','12-MAY-17', 1, 11, 104);
```

SELECT * FROM PUBLISHER;

```
SQL> select * from publisher;

NAME                      PHONE ADDRESS
-------------------- ---------- --------------------
MCGRAW-HILL          9989076587 BANGALORE
PEARSON              9889076565 NEWDELHI
RANDOM HOUSE         7455679345 HYDRABAD
HACHETTE LIVRE       8970862340 CHENAI
GRUPO PLANETA        7756120238 BANGALORE
```

SELECT * FROM BOOK;

```
SQL> SELECT * FROM BOOK;

   BOOK_ID TITLE                PUB_YEAR             PUBLISHER_NAME
---------- -------------------- -------------------- --------------------
         1 DBMS                 JAN-2017             MCGRAW-HILL
         2 ADBMS                JUN-2016             MCGRAW-HILL
         3 CN                   SEP-2016             PEARSON
         4 CG                   SEP-2015             GRUPO PLANETA
         5 OS                   MAY-2016             PEARSON
```

SELECT * FROM BOOK_AUTHORS;

```
SQL> SELECT * FROM BOOK_AUTHORS;

AUTHOR_NAME            BOOK_ID
-------------------- ----------
NAVATHE                       1
NAVATHE                       2
TANENBAUM                     3
EDWARD ANGEL                  4
GALVIN                        5
```

SELECT * FROM LIBRARY_BRANCH;

```
SQL> SELECT * FROM LIBRARY_BRANCH;

BRANCH_ID BRANCH_NAME                                        ADDRESS
--------- -------------------------------------------------- --------------------------------------------------
       10 AR NAGAR                                          BANGALORE
       11 ANSIT                                             BANGALORE
       12 RAJAJI NAGAR                                      BANGALORE
       13 NITTE                                             MANGALORE
       14 MANIPAL                                           UDUPI
```

```
SQL> SELECT * FROM BOOK_COPIES;

NO_OF_COPIES    BOOK_ID  BRANCH_ID
------------ ---------- ----------
          10          1         10
           5          1         11
           2          2         12
           5          2         13
           7          3         14
           1          5         10
           3          4         11
```

SELECT * FROM CARD;

```
SQL> SELECT * FROM CARD;

   CARD_NO
----------
       100
       101
       102
       103
       104
```

SELECT * FROM BOOK_LENDING;

```
SQL> select * from book_lending;

DATE_OUT  DUE_DATE     BOOK_ID  BRANCH_ID    CARD_NO
--------- --------- ---------- ---------- ----------
01-JAN-17 01-JUN-17          1         10        101
11-JAN-17 11-MAR-17          3         14        101
21-FEB-17 21-APR-17          2         13        101
15-MAR-17 15-JUL-17          4         11        101
12-APR-17 12-MAY-17          1         11        104
```

**Queries:**

**1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.**

```
SELECT B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME, A.AUTHOR_NAME, C.NO_OF_COPIES,
L.BRANCH_ID
FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C, LIBRARY_BRANCH L
WHERE B.BOOK_ID=A.BOOK_ID
AND B.BOOK_ID=C.BOOK_ID  AND L.BRANCH_ID=C.BRANCH_ID;
```

```
   BOOK_ID TITLE                PUBLISHER_NAME       AUTHOR_NAME          NO_OF_COPIES  BRANCH_ID
---------- -------------------- -------------------- -------------------- ------------ ----------
         1 DBMS                 MCGRAW-HILL          NAVATHE                        10         10
         1 DBMS                 MCGRAW-HILL          NAVATHE                         5         11
         2 ADBMS                MCGRAW-HILL          NAVATHE                         2         12
         2 ADBMS                MCGRAW-HILL          NAVATHE                         5         13
         3 CN                   PEARSON              TANENBAUM                       7         14
         5 OS                   PEARSON              GALVIN                          1         10
         4 CG                   GRUPO PLANETA        EDWARD ANGEL                    3         11
```

**2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.**

```
SELECT CARD_NO  FROM BOOK_LENDING
WHERE DATE_OUT BETWEEN '01-JAN-2017' AND '01-JUL-2017'
GROUP BY CARD_NO  HAVING COUNT (*)>3;
```

```
   CARD_NO
----------
       101
```

**3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.**

```
DELETE FROM BOOK
WHERE BOOK_ID=3;
```

```
SQL> SELECT * FROM BOOK;

   BOOK_ID TITLE                PUB_YEAR             PUBLISHER_NAME
---------- -------------------- -------------------- ---------------
         1 DBMS                 JAN-2017             MCGRAW-HILL
         2 ADBMS                JUN-2016             MCGRAW-HILL
         4 CG                   SEP-2015             GRUPO PLANETA
         5 OS                   MAY-2016             PEARSON
```

**4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.**

```
CREATE VIEW V_PUBLICATION AS SELECT PUB_YEAR  FROM BOOK;
```

```
PUB_YEAR
------------
JAN-2017
JUN-2016
SEP-2016
SEP-2015
MAY-2016
```

**5. Create a view of all books and its number of copies that are currently available in the Library.**

```
CREATE VIEW V_BOOKS AS  SELECT B.BOOK_ID, B.TITLE, C.NO_OF_COPIES
FROM BOOK B, BOOK_COPIES C, LIBRARY_BRANCH L
WHERE B.BOOK_ID=C.BOOK_ID  AND C.BRANCH_ID=L.BRANCH_ID;
```

```
  BOOK_ID TITLE                NO_OF_COPIES
--------- -------------------- ------------
        1 DBMS                           10
        1 DBMS                            5
        2 ADBMS                           2
        2 ADBMS                           5
        3 CN                              7
        5 OS                              1
        4 CG                              3
```

**OBJECTIVE: A**nswer the following Queries:

**QUERIES -**

1. Find out the product which has been sold to 'Ivan Sayross.'
2. Find out the product and their quantities that will have do delivered.
3. Find the product_no and description of moving products.
4. Find out the names of clients who have purchased 'CD DRIVE'
5. List the product_no and s_order_no of customers haaving qty ordered less than 5 from the order details table for the product "1.44 floppies".
6. Find the products and their quantities for the orders placed by 'Vandan Saitwal ' and "Ivan Bayross".
7. Find the products and their quantities for the orders placed by client_no " C00001" and "C00002"
8. Find the order No,, Client No and salesman No. where a client has been received by more than one salesman.
9. Display the s_order_date in the format "dd-mm-yy" e.g. "12- feb-96"
10. Find the date , 15 days after date.

# Experiment No.7

# Theory & Concept

**Objective:- To implement the concept of grouping of Data.**

**Grouping Data From Tables:**

There are circumstances where we would like to apply the aggregate function not only to a single set of tuples, but also to a group of sets of tuples, we specify this wish in SQL using the group by clause. The attribute or attributes given in the group by clause are used to form group. Tuples with the same value on all attributes in the group by clause are placed in one group.

**Syntax:**

```
SELECT columnname, columnname
FROM  tablename
GROUP BY columnname;
```

At times it is useful to state a condition that applies to groups rather than to tuples. For example we might be interested in only those branches where the average account balance is more than 1200. This condition does not apply to a single tuple, rather it applies to each group constructed by the GROUP BY clause. To express such Questionry, we use the having clause of SQL. SQL applies predicates in the having may be used.

**Syntax:**

```
SELECT columnname, columnname
FROM  tablename
GROUP BY columnname;
HAVING searchcondition;
```

**Objective- Answer the following Queries:**

**Q1.- Print the description and total quantity sold for each product.**

**Q2.- Find the value of each product sold.**

**Q3.- Calculate the average quantity sold for each client that has a maximum order value of 15000.**

**Q4.- Find out the products which has been sold to Ivan.**

**Q5.- Find the names of clients who have 'CD Drive'.**

**Q6.- Find the products and their quantities for the orders placed by 'Vandana' and 'Ivan'.**

**Q7.- Select product_no, total qty_ordered for each product.**

**Q8.- Select product_no, product description and qty ordered for each product.**

**Q9.- Display the order number and day on which clients placed their order.**

**Q10.- Display the month and Date when the order must be delivered.**

# Experiment NO.8

# Theory & Concept

**Objective:- To implement the concept of Sub Queries.**

**Sub Queries:- A Sub Queries** is a form of an SQL statement that appears inside another SQL statement. It also termed as nested Questionry. The statement containing a Sub Queries called a parent statement. The rows returned bu the Sub Queries are use by the following statement.
It can be used by the following commands:

1. To insert records in the target table.
2. To create tables and insert records in this table.
3. To update records in the target table.
4. To create view.
5. To provide values for the condition in the WHERE , HAVING IN , SELECT,UPDATE, and DELETE statements.

Example: Creating client master table from oldclient_master, table

Create table client_master AS SELECT * FROM oldclient_master;

**Using the Union, Intersect and Minus Clause:**

**Union Clause:**
The user can put together multiple Queries and combine their output using the union clause . The union clause merges the output of two or more Queries into a single set of rows and column. The final output of union clause will be

Output: = Records only in Questionry one + records only in Questionry two + A single set of records with is common in the both Queries.

Syntax: SELECT columnname, columname FROM tablename 1
UNION
SELECT columnname, columnname From tablename2;

**Intersect Clause:** The use can put together multiple Queries and their output using the interest clause. The final output of the interest clause will be :

Output =A single set of records which are common in both Queries

Syntax: SELECT columnname, columnname
FROM tablename 1
INTERSECT
SELECT columnname, columnname
FROM tablename 2;

MINUS CLAUSE:- The user can put together multiple Queries and combine their output = records only in Questionry one

Syntax: SELECT columnname, columnname
FROM tablename MINUS
SELECT columnname, columnname
FROM tablename ;

**Objective:** Answer the following Queries:

QUERIES:

1. Find the product_no and description of non- moving products.
2. Find the customer name, address, city and pincode for the client who has placed order no "019001"
3. Find the client names who have placed order before the month of may 96.
4. Find out if product "1.44 Drive" is ordered by only client and print the client_no name to whom it was sold.
5. find the names of client who have placed orders worth Rs.10000 or more.
6. Select the orders placed by 'Rahul Desai"
7. Select the names of persons who are in Mr. Pradeep's department and who have also worked on an inventory control system.
8. Select all the clients and the salesman in the city of Bombay.
9. Select salesman name in "Bombay" who has atleast one client located at "Bombay"
10. Select the product_no, description, qty_on-hand,cost_price of non_moving items in the product_master table.

# Experiment. 9

# Theory and Concept

**Objective:- To implement the concept of Indexes and views.**

**Indexes-** An index is an ordered list of content of a column or group of columns in a table. An index created on the single column of the table is called simple index. When multiple table columns are included in the index it is called composite index.

Creating an Index for a table:-

**Syntax (Simple)**

```
CREATE INDEX index_name
ON tablename(column name);
```

Composite Index:-
```
CREATE INDEX index_name
ON tablename(columnname,columnname);
```

Creating an UniQuestion Index:-
```
CREATE UNIQUESTION INDEX indexfilename
ON tablename(columnname);
```

Dropping Indexes:- An index can be dropped by using DROP INDEX

**SYNTAX:-**

```
DROP INDEX indexfilename;
```

Views:-

Logical data is how we want to see the current data in our database. Physical data is how this data is actually placed in our database.

Views are masks placed upon tables. This allows the programmer to develop a method via which we can display predetermined data to users according to our desire.

Views may be created fore the following reasons:

1. The DBA stores the views as a definition only. Hence there is no duplication of data.
2. Simplifies Queries.
3. Can be Questionried as a base table itself.
4. Provides data security.
5. Avoids data redundancy.

**Creation of Views:- Syntax:-**
```
CREATE VIEW viewname AS
SELECT columnname,columnname
FROM tablename
WHERE columnname=expression_list;
```

**Renaming the columns of a view:- Syntax:-**
```
CREATE VIEW viewname AS
SELECT newcolumnname….
FROM tablename
WHERE columnname=expression_list;
```

**Selecting a data set from a view- Syntax:-**
```
SELECT columnname, columnname
FROM viewname
WHERE search condition;
```

**Destroying a view-**

**Syntax:-**

DROP VIEW viewname;

**Objective: Answer the following Questions**

Q1. Create an index on the table client_master, field client_no.

Q2. Create an index on the sales_order, field s_order_no.

Q3. Create an composite index on the sales_order_details table for the columns s_order_no and product_no.

Q4. Create an composite index ch_index on challan_header table for the columns challan no and s_order_no.

Q5. Create an uniQuestion index on the table salesman_master, field salesman_no.

Q6. Drop index ch_index on table challan_header.

Q7. Create view on salesman_master whose sal_amt is less than 3500.

Q8. Create a view client_view on client_master and rename the columns as name, add1, add2, city, pcode, state respectively.

Q9. Select the client names from client_view who lives in city 'Bombay'.

Q10. Drop the view client_view.

# Experiment No.10

# Theory and Concept

## Objective:- To implement the basics of PL/SQL.

**Introduction –** PL/SQL bridges the gap between database technology and procedural programming languages. It can be thought of as a development tool that extends the facilities of Oracles SQL database language. Via PL/SQL you can insert, delete, update and retrieve table data as well as use procedural techniQuestions such as writing loops or branching to another block of code.

## PL/SQL Block structure-

DECLARE
Declarations of memory variables used later
BEGIN
SQL executable statements for manipulating table data.
EXCEPTIONS
SQL and/or PL.SQL code to handle errors.
END;

**Displaying user Messages on the screen –** Any programming tool requires a method through which messages can be displayed to the user.

**dbms_output** is a package that includes a number of procedure and functions that accumulate information in a buffer so that it can be retrieved later. These functions can also be used to display message to the user.

**put_line**: put a piece of information in the buffer followed by a end of line marker. It can also be used to display message to the user.
Setting the server output on:

## SET SERVER OUTPUT ON:

Example: Write the following code in the PL/SQL block to display message to user
DBMS_OUTPUT.PUT_LINE('Display user message');

## Conditional control in PL/SQL-

Syntax:

```
IF <condition> THEN
<Action>
ELSEIF<condition>
<Action>
ELSE
<Action>
ENDIF;
```

## The WHILE LOOP:

Syntax:

```
WHILE <condition>
LOOP
```

<Action>
END LOOP;

## The FOR LOOP statement:

Syntax:

FOR variable IN [REVERSE] start—end
LOOP
<Action>
END LOOP;

**The GOTO statement**: The goto statement allows you to change the flow of control within a PL/SQL Block.

Q1. WAP in PL/SQL for addition of two numbers.

Q2. WAP in PL/SQL for addition of 1 to 100 numbers.

Q3. WAP in PL/SQL to check the given number is even or odd.

Q4. WAP in PL/SQL to inverse a number, eg. Number 5639 when inverted must be display output 9365.

Q5. WAP in PL/SQL for changing the price of product 'P00001' to 4000 if the price is less than 4000 in product_master table. The change is recorded in the old_price_table along with product_no and the date on which the price was changed last.

# Experiment No.11

# Theory and Concept

**Objective:- To implement the concept of Cursor and Trigger.**

**Cursor–** We have seen how oracle executes an SQL statement. Oracle DBA uses a work area for its internal processing. This work area is private to SQL's operation and is called a **cursor.**
The data that is stored in the cursor is called the **Active Data set.** The size of the cursor in memory is the size required to hold the number of rows in the Active Data Set.

**Explicit Cursor-** You can explicitly declare a cursor to process the rows individually. A cursor declared by the user is called **Explicit Cursor.** For Queries that return more than one row, You must declare a cursor explicitly.
The data that is stored in the cursor is called the **Active Data set.** The size of the cursor in memory is the size required to hold the number of rows in the Active

**Why use an Explicit Cursor-** Cursor can be used when the user wants to process data one row at a time.

**Explicit Cursor Management-** The steps involved in declaring a cursor and manipulating data in the active data set are:-

- Declare a cursor that specifies the SQL select statement that you want to process.
- Open the Cursor.
- Fetch the data from the cursor one row at a time.
- Close the cursor.

**Explicit Cursor Attributes-** Oracle provides certain attributes/ cursor variables to control the execution of the cursor. Whenever any cursor(explicit or implicit) is opened and used Oracle creates a set of four system variables via which Oracle keeps track of the 'Current' status of the cursor. You

- Declare a cursor that specifies the SQL select statement that you want to process.
- Open the Cursor.
- Fetch the data from the cursor one row at a time.
- Close the cursor.

## How to Declare the Cursor:-

The General Syntex to create any perticular cursor is as follows:-
**Cursor <Cursorname> is Sql Statement;**

## How to Open the Cursor:-

The General Syntex to Open any perticular cursor is as follows:-
**Open Cursorname;**

## Fetching a record From the Cursor:-

The fetch statement retrieves the rows from the active set to the variables one at a time. Each time a fetch is executed. The focus of the DBA cursor advances to the next row in the Active set.
One can make use of any loop structute(Loop-End Loop along with While,For) to fetch the records from the cursor into variable one row at a time.

The General Syntex to Fetch the records from the cursor is as follows:-
**Fetch cursorname into variable1,variable2,_____**

## Closing a Cursor:-

The General Syntex to Close the cursor is as follows:-
**Close <cursorname>;**

## Database Triggers:-

Database triggers are procedures that are stored in the database and are implicitly executed(fired) when the contents of a table are changed.

## Use of Database Triggers:-

Database triggers support Oracle to provide a highly customized database management system. Some of the uses to which the database triggers can be put to customize management information in Oracle are as follows:-

- A Trigger can permit DML statements against a table anly if they are issued, during regular bussiness hours or on predetermined weekdays.
- A trigger can also be used to keep an audit trail of a table along with the operation performed and the time on which the operation was performed.
- It can be used to prevent invalid transactions.
- Enforce complex security authorizations.

## How to apply DataBase Triggers:-

A trigger has three basic parts:-

1. A triggering event or ststement.
2. A triger restriction
3. A trigger action.

## Types of Triggers:-

Using the various options , four types of triggers can be created:-

1. **Before Statement Trigger:-** Before executing the triggering statement, the trigger action is executed.
2. **Before Row Trigger:-** Before modifying the each row affected by the triggering statement and before appropriate integrity constraints, the trigger is executed if the trigger restriction either evaluated to TRUE or was not included.'
3. **After Ststement Trigger:-** After executing the triggering statement and applying any deferred integrity canstraints, the trigger action is executed.
4. **After row Trigger:-** After modifying each row affected by the triggering statement and possibly applying appropriate integrity constraints, the trigger action is executed for the current row if the trigger restriction either evaluates to TRUE or was not included.

## Syntex For Creating Trigger:-

The syntex for Creating the Trigger is as follows:-

Create or replace Trigger<Triggername> {Before,After} {Delete, Insert, Update } On <Tablename>
For Each row when Condition
Declare
<Variable declarations>;
<Constant Declarations>;
Begin
<PL/SQL> Subprogram Body;

Exception
Exception Pl/SQL block;
End;

## How to Delete a Trigger:-

The syntex for Deleting the Trigger is as follows:-
Drop Trigger <Triggername>;

**Table Name:0- Employee**

| Column Name | Data Type | Size | Attributes |
|---|---|---|---|
| Emp_Code | Varchar2 | 10 | Primary Key |
| Ename | Varchar2 | 20 | The name of the Candidate |
| Deptno | Number | 5 | The Department No |
| Job | Vrachar2 | 20 | The Employee Job Detail |
| Salary | Number | 8,2 | The Current salary of Em |

**Table name:- Emp_raise**

| Column Name | Data Type | Size | Attributes |
|---|---|---|---|
| Emp_Code | Varchar2 | 10 | Primary Key |
| Raise_Date | date | | The Date On which the raise was given |
| Raise_Amt | Number | 8,2 | The raise given to the employee |

The HRD manager has decided to raise the salary for all the employees in the Dept No 20 by 5%. Whenever any such raise is given to the employees, a record for the same is maintained in the emp_raise table. It includes the EmpNo, the date when the raise was given and the actual raise. Write a PL/SQL block to update the salary of each employee and insert a record in the emp_raise table.

Q2:- Two Tables are there

**Table Name:- Client_Master**

| Column Name | Data Type | Size | Attributes |
|---|---|---|---|
| Client_No | Varchar2 | 6 | Primary Key/First letter must start with 'C' |
| Name | Varchar2 | 20 | Not Null |
| City | Varchar2 | 10 | |
| State | Vrachar2 | 10 | |
| Bal_Due | Number | 10,2 | |

**Table name:- auditclient**

| Column Name | Data Type | Size | Attributes |
|---|---|---|---|
| Client_No | Varchar2 | 10 | Primary Key |
| Name | Varchar2 | 20 | |
| Bal_Due | Number | 10,2 | |
| Operation | Varchar2 | 8 | |
| Odate | Date | | |

Create a transparent audit system for a table Client_master. The system must keep track of the records that are being deleted or modified and when they have been deleted or modified.

# Experiment No.12

Design and implementation of payroll processing system

# Experiment No.13

Mini project