

LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE  
OF COMPUTER APPLICATION, MUMBAI

# Data Mining and Business Intelligence Lab Manual

DR. RASHMITA PRADHAN,  
DR. SADHANA OJHA,  
ASST. PROF. KHYATI MANVAR

Faculty, Master of Computer Application (M.C.A.)

Late Bhausaheb Hiray S.S. Trust's Institute of Computer Application





# **DATA MINING AND BUSINESS INTELLIGENCE LAB MANUAL**



# DATA MINING AND BUSINESS INTELLIGENCE LAB MANUAL

LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE OF COMPUTER  
APPLICATION BANDRA, EAST MUMBAI-51

**Dr. Rashmita Pradhan**

**Co- Author: Dr. Sadhana Ojha**

**Co-Author: Assistant Professor Khyati Manvar**

---

*Faculty, Master of Computer Application (M.C.A.)*

*Late Bhausaheb Hiray S.S. Trust's Institute of Computer Application*



First Edition, 2023

Copyright © Late Bhausaheb Hiray S.S. Trust's Institute Of Computer Application, Bandra (E), Mumbai-51, 2023

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the author, except in the case of brief quotations embodied in critical reviews and certain other non-commercial uses permitted by copyright law. For permission requests, write to the publisher at the address below.

This book can be exported from India only by the publishers or by the authorized suppliers. Infringement of this condition of sale will lead to Civil and Criminal prosecution.

Paperback ISBN: 978-81-19221-54-7

eBook ISBN: 978-81-19221-50-9

WebPDF ISBN: 978-81-19221-48-6

Note: Due care and diligence has been taken while editing and printing the book; neither the author nor the publishers of the book hold any responsibility for any mistake that may have inadvertently crept in.

The publishers shall not be liable for any direct, consequential, or incidental damages arising out of the use of the book. In case of binding mistakes, misprints, missing pages, etc., the publishers' entire liability, and your exclusive remedy, is replacement of the book within one month of purchase by similar edition/reprint of the book.

Printed and bound in India by

16Leaves  
2/579, Singaravelan Street  
Chinna Neelankarai  
Chennai – 600 041, India

info@16leaves.com  
www.16Leaves.com  
Call: 91-9940638999

# Contents

|   |           |
|---|-----------|
| <b>1 OLAP with Oracle</b>   | <b>1</b>  |
| A) Analytical Functions   | 1         |
| B) Rollup and Cubes   | 7         |
| C) Windowing Functions  | 14        |
| <hr/>   |           |
| <b>2 Preparing Reports, Dashboards, Balanced score card and Analysis of Reports using Tableau</b> | <b>21</b> |
| <hr/>   |           |
| <b>3 Implementation of Data Mining Algorithms using WEKA</b>                                      | <b>47</b> |
| A) Introduction to Weka Tool  | 47        |
| B) Implementation of different Preprocessing Techniques   | 50        |
| C) Implementation of Association Techniques   | 55        |
| D) Implementation of Classification Techniques  | 58        |
| E) Implementation of Clustering Techniques  | 64        |
| <hr/>   |           |
| <b>4 Implementation of ETL(Extract Transform and Load) Processing using Pentaho Tool</b>          | <b>73</b> |
| A) Introduction to Pentaho Tool   | 73        |
| B) Creating New Transformation  | 74        |
| I) Configuration of SQL Input & Output  | 74        |
| II) Configuration of MS Excel Input & Output  | 81        |
| III) Transform [Sorting Row and Adding Sequence]  | 85        |
| IV) Flow [Filter Rows]  | 89        |





# Chapter 1 OLAP with Oracle

## Practical No 1

AIM: OLAP with Oracle.

### Description:

- A) ANALYTICAL FUNCTIONS
- B) ROLLUP AND CUBES
- C) WINDOWING FUNCTIONS

## A) Analytical Functions

It computes on aggregate value based on a group of rows.

### Analytical Functions / Clauses are:-

- a) RANK():- Rank function calculate the rank of a value in a group of values.
- b) ROW\_NUMBER():- It is used to give unique number to rows.
- c) DENSE\_RANK():- It computes the rank of a row in an ordered group of rows.
- d) NTILE():- It divides an ordered data set into number of buckets indicated by expression.
- e) FIRST:- It can be used to return the first value FROM an ordered sequence.
- f) LAST:- It can be used to return the last value FROM an ordered sequence.
- g) LEAD ():- It returns offset(incrementally increased) value of an argument column.
- h) LAG ():- This function is used to access data FROM a previous row.

### EXAMPLES:-

**Create following table.**

#### **1. EMPLOYEES(eid,ename,salary,dname)**

```
SQL> CREATE TABLE EMPLOYEES (  
eno NUMERIC(3), ename VARCHAR(10), esal NUMERIC(6),  
dname varchar(10)  
);
```

**Table created.**

**SQL>**INSERT INTO EMPLOYEES VALUES(200,'Vinay',20000,'IT');

» **1 row created.**

» **SQL>**INSERT INTO EMPLOYEES VALUES(201,'Anand',30000,'IT');

» **1 row created.**

» **SQL>**INSERT INTO EMPLOYEES VALUES(202,'Anil',25000,'Marketing');

» **1 row created.**

» **SQL>**INSERT INTO EMPLOYEES

» VALUES(203,'Yogita',20000,'Marketing');

» **1 row created.**

» **SQL>**INSERT INTO EMPLOYEES

» VALUES(204,'Priyanka',20000,'Management');

» **1 row created.**

» **SQL>**INSERT INTO EMPLOYEES

» VALUES(205,'Likhita',25000,'Management');

» **1 row created.**

» **SQL>**INSERT INTO EMPLOYEES

» VALUES(206,'Ankit',30000,'Management');

» **1 row created.**

» **SQL>**INSERT INTO EMPLOYEES

» VALUES(207,'Aniket',35000,'Accountant');

» **1 row created.**

» **SQL>** SELECT \* FROM EMPLOYEES;

| ENO | ENAME    | ESAL  | DNAME      |
|-----|----------|-------|------------|
| 200 | Vinay    | 20000 | IT         |
| 201 | Anand    | 30000 | IT         |
| 202 | Anil     | 25000 | Marketing  |
| 203 | Yogita   | 20000 | Marketing  |
| 204 | Priyanka | 20000 | Management |
| 205 | Likhita  | 25000 | Management |
| 206 | Ankit    | 30000 | Management |
| 207 | Aniket   | 35000 | Accountant |

**Perform the following queries on above table.**

**1) Display employees information FROM Employee table and give unique number to each row.**

```
SQL> SELECT eno,ename,esal,dname,ROW_NUMBER() OVER(ORDER BY ENO)"ID"
FROM Employees;
```

| ENO | ENAME    | ESAL  | DNAME      | ID |
|-----|----------|-------|------------|----|
| 200 | Vinay    | 20000 | IT         | 1  |
| 201 | Anand    | 30000 | IT         | 2  |
| 202 | Anil     | 25000 | Marketing  | 3  |
| 203 | Yogita   | 20000 | Marketing  | 4  |
| 204 | Priyanka | 20000 | Management | 5  |
| 205 | Likhita  | 25000 | Management | 6  |
| 206 | Ankit    | 30000 | Management | 7  |
| 207 | Aniket   | 35000 | Accountant | 8  |

**2) Display employees information and give unique number to each row in ascending order of salary.**

```
SQL> SELECT eno,ename,esal,dname,ROW_NUMBER() OVER(ORDER BY ESAL)"ID"
FROM Employees;
```

| ENO | ENAME    | ESAL  | DNAME      | ID |
|-----|----------|-------|------------|----|
| 200 | Vinay    | 20000 | IT         | 1  |
| 203 | Yogita   | 20000 | Marketing  | 2  |
| 204 | Priyanka | 20000 | Management | 3  |
| 205 | Likhita  | 25000 | Management | 4  |
| 202 | Anil     | 25000 | Marketing  | 5  |
| 206 | Ankit    | 30000 | Management | 6  |

| ENO | ENAME  | ESAL  | DNAME      | ID |
|-----|--------|-------|------------|----|
| 201 | Anand  | 30000 | IT         | 7  |
| 207 | Aniket | 35000 | Accountant | 8  |

3) Assign the ranks to employees in ascending order of salary.

```
SQL> SELECT eno,ename,esal,dname,RANK() OVER(ORDER BY ESAL)"Rank"
FROM Employees;
```

| ENO | ENAME    | ESAL  | DNAME      | Rank |
|-----|----------|-------|------------|------|
| 200 | Vinay    | 20000 | IT         | 1    |
| 203 | Yogita   | 20000 | Marketing  | 1    |
| 204 | Priyanka | 20000 | Management | 1    |
| 205 | Likhita  | 25000 | Management | 4    |
| 202 | Anil     | 25000 | Marketing  | 4    |
| 206 | Ankit    | 30000 | Management | 6    |
| 201 | Anand    | 30000 | IT         | 6    |
| 207 | Aniket   | 35000 | Accountant | 8    |

4) Assign the ranks to employees in descending order of salary.

```
SQL> SELECT eno,ename,esal,dname,RANK() OVER(ORDER BY ESAL DESC)"Rank"
FROM Employees;
```

| ENO | ENAME    | ESAL  | DNAME      | Rank |
|-----|----------|-------|------------|------|
| 207 | Aniket   | 35000 | Accountant | 1    |
| 206 | Ankit    | 30000 | Management | 2    |
| 201 | Anand    | 30000 | IT         | 2    |
| 205 | Likhita  | 25000 | Management | 4    |
| 202 | Anil     | 25000 | Marketing  | 4    |
| 204 | Priyanka | 20000 | Management | 6    |
| 200 | Vinay    | 20000 | IT         | 6    |
| 203 | Yogita   | 20000 | Marketing  | 6    |

5) Assign the ranks to employees in ascending order of salary using dense rank and point out the difference and display record in descending order of salary.

```
SQL> SELECT eno,ename,esal,dname,DENSE_RANK() OVER(ORDER BY ESAL)"Dense Rank"
FROM Employees ORDER BY ESAL DESC;
```

| ENO | ENAME    | ESAL  | DNAME      | Dense Rank |
|-----|----------|-------|------------|------------|
| 207 | Aniket   | 35000 | Accountant | 4          |
| 201 | Anand    | 30000 | IT         | 3          |
| 206 | Ankit    | 30000 | Management | 3          |
| 205 | Likhita  | 25000 | Management | 2          |
| 202 | Anil     | 25000 | Marketing  | 2          |
| 203 | Yogita   | 20000 | Marketing  | 1          |
| 200 | Vinay    | 20000 | IT         | 1          |
| 204 | Priyanka | 20000 | Management | 1          |

6) Write a Query for finding highest and lowest salary of each department.

```
SQL> SELECT deptid,MIN(salary) KEEP(DENSE_RANK FIRST ORDER BY Salary)"Lowest Sal", MAX(salary)
KEEP(DENSE_RANK LAST ORDER BY Salary)"Highest Sal"
```

```
FROM Employees GROUP BY deptid;
```

| DEPTID | Lowest Sal | Highest Sal |
|--------|------------|-------------|
| 105    | 25000      | 50000       |
| 106    | 60000      | 60000       |
| 107    | 20000      | 23000       |
| 114    | 45000      | 45000       |
| 115    | 65000      | 65000       |
| 117    | 40000      | 40000       |

7) Write a Query to find information of employee who were joined first in each department.

```
SQL> SELECT deptid,MIN(DOJ) KEEP(DENSE_RANK FIRST ORDER BY Salary)"Hired First"
```

```
FROM Employees GROUP BY deptid;
```

| DEPTID | Hired Fir |
|--------|-----------|
| 105    | 30-MAR-17 |
| 106    | 26-OCT-16 |
| 107    | 17-DEC-18 |
| 114    | 14-AUG-17 |
| 115    | 14-NOV-17 |
| 117    | 02-SEP-18 |

**8) Write a Query which returns a salary FROM previous row name it as Sal\_Prev. Calculate the difference between salary of current row and previous row.**

```
SQL> SELECT empid,fname,salary,LAG(Salary,1,0) OVER(ORDER BY Salary)"SAL_PREV", Salary-LAG(Salary,1,0)
OVER(ORDER BY Salary)"SAL_DIFF"
```

FROM Employees;

| EMPID | FNAME    | SALARY | SAL_PREV | SAL_DIFF |
|-------|----------|--------|----------|----------|
| 11    | Ross     | 20000  | 0        | 20000    |
| 14    | Joey     | 23000  | 20000    | 3000     |
| 6     | Poonam   | 25000  | 23000    | 2000     |
| 17    | Phoebe   | 40000  | 25000    | 15000    |
| 15    | Chandler | 45000  | 40000    | 5000     |
| 7     | Nisha    | 50000  | 45000    | 5000     |
| 9     | Monika   | 60000  | 50000    | 10000    |
| 10    | Rachel   | 60000  | 60000    | 0        |
| 19    | Tushar   | 65000  | 60000    | 5000     |

**9) Write a Query which returns a salary FROM next row name it as Sal\_Next. Calculate the difference between salary of current row and following row.**

```
SQL> SELECT empid,fname,salary,LEAD(Salary,1,0) OVER(ORDER BY Salary)"SAL_NEXT", Salary-LEAD(Salary,1,0)
OVER(ORDER BY Salary)"SAL_DIFF"
```

FROM Employees;

| EMPID | FNAME    | SALARY | SAL_NEXT | SAL_DIFF |
|-------|----------|--------|----------|----------|
| 11    | Ross     | 20000  | 23000    | -3000    |
| 14    | Joey     | 23000  | 25000    | -2000    |
| 6     | Poonam   | 25000  | 40000    | -15000   |
| 17    | Phoebe   | 40000  | 45000    | -5000    |
| 15    | Chandler | 45000  | 50000    | -5000    |
| 7     | Nisha    | 50000  | 60000    | -10000   |
| 9     | Monika   | 60000  | 60000    | 0        |
| 10    | Rachel   | 60000  | 65000    | -5000    |
| 19    | Tushar   | 65000  | 0        | 65000    |

## B) ROLLUP AND CUBES ◀◀◀

### Description:

**ROLLUP** is an extension of GROUP BY Clause. This option allows us to include extra rows that represent the subtotal and grand-total which are known as super aggregate row.

**CUBE** returns row containing a subtotal for all combination of columns.

### ROLLUP EXAMPLES:-

#### Create following table.

```
CREATE TABLE sales(  
time number(4), region varchar2(15), dept varchar2(20), profit number(10,2)  
);  
  
INSERT INTO sales values(1996,'central','pen-sales',75000);  
INSERT INTO sales values(1996,'central','book_sales',74000);  
INSERT INTO sales values(1996,'east','pen_sales',89000);  
INSERT INTO sales values(1997,'central','pen_sales',82000);  
INSERT INTO sales values(1997,'central','pen-sales',75000);  
INSERT INTO sales values(1997,'east','book_sales',74000);  
INSERT INTO sales values(1997,'east','pen_sales',89000);  
INSERT INTO sales values(1997,'west','book_sales',115000);
```

#### 1) Find total profit department wise.

```
SQL> SELECT dept,SUM(profit)"Profit" FROM sales  
GROUP BY dept;
```

---

| DEPT       | Profit |
|------------|--------|
| pen_sales  | 260000 |
| book_sales | 263000 |
| pen-sales  | 150000 |

---

## 2) Find total profit department wise along with Grand Total.

```
SQL> SELECT dept,SUM(profit)"profit" FROM sales
```

```
GROUP BY ROLLUP(dept);
```

---

| <b>DEPT</b>       | <b>profit</b> |
|-------------------|---------------|
| <b>book_sales</b> | <b>263000</b> |
| <b>pen-sales</b>  | <b>150000</b> |
| <b>pen_sales</b>  | <b>260000</b> |
|                   | <b>673000</b> |

---

## 3) Find the total profit time and region wise.

```
SQL> SELECT time,region,SUM(profit)"profit" FROM sales
```

```
GROUP BY time,region;
```

---

| <b>TIME</b> | <b>REGION</b>  | <b>profit</b> |
|-------------|----------------|---------------|
| <b>1997</b> | <b>central</b> | <b>157000</b> |
| <b>1997</b> | <b>west</b>    | <b>115000</b> |
| <b>1997</b> | <b>east</b>    | <b>163000</b> |
| <b>1996</b> | <b>central</b> | <b>149000</b> |
| <b>1996</b> | <b>east</b>    | <b>89000</b>  |

---

## 4) Find the total profit time and region wise along with Time wise total and Grand Total.

```
SQL> SELECT time,region,SUM(profit)"profit" FROM sales
```

```
GROUP BY ROLLUP(time,region);
```

---

| <b>TIME</b> | <b>REGION</b>  | <b>profit</b> |
|-------------|----------------|---------------|
| <b>1996</b> | <b>east</b>    | <b>89000</b>  |
| <b>1996</b> | <b>central</b> | <b>149000</b> |
| <b>1996</b> |                | <b>238000</b> |
| <b>1997</b> | <b>east</b>    | <b>163000</b> |
| <b>1997</b> | <b>west</b>    | <b>115000</b> |
| <b>1997</b> | <b>central</b> | <b>157000</b> |
| <b>1997</b> |                | <b>435000</b> |
|             |                | <b>673000</b> |

---