Department of Information Technology

NAME: Ayush Vinod Upadhyay
ROLL No: I025
Sap ID: 60003220131
BRANCH: Information Technology
Batch: I1-1
SUBJECT: DBMS

Experiment No: 3

Accessing & Modifying Data in Oracle

The Fundamentals of a SELECT Statement

A *SELECT* statement in SQL enables you to retrieve existing data from a Oracle database.

The full syntax of the *SELECT* statement is complex, but the main clauses can be summarized as follows:

- SELECT *select_list*
- [INTO *new_table_name*]
- FROM *table_list*
- [WHERE *search_conditions*]
- [GROUP BY *group_by_list*]
- [HAVING *search_conditions*]
- [ORDER BY *order_list* [ASC | DESC]]

Using Keywords in the Select List
The select list can also contain keywords that control the final format of the result set.
The DISTINCT Keyword
The DISTINCT keyword eliminates duplicate rows from a result set.

```
SELECT DISTINCT ShipCity, ShipRegion
FROM Orders
ORDER BY ShipCity
```

Department of Information Technology

The TOP n Rows

The *FETCH FIRST n Rows only* keyword specifies that the first *n* rows of the result set are to be returned. If ORDER BY is specified, the rows are selected after the result set is ordered. The *n* placeholder is the number of rows to return

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s)
FETCH FIRST number ROWS ONLY;
```

The AS Keyword

You can improve the readability of a *SELECT* statement by giving a table an alias (also known as a correlation name or range variable). A table alias can be assigned either with or without the AS keyword: ■ *table_name* AS *table_alias*
■ *table_name table_alias*

In the following example, the alias p is assigned to the Publishers table:

```
SELECT p.pub_id, p.pub_name
FROM publishers AS p
```

Types of Information in the Select List

A select list can include many types of information, such as a simple expression or a scalar subquery. The following example shows many of the items that you can include in a select list:

```
SELECT FirstName + ' ' + LastName AS "Employee Name",
HomePhone,
Region
FROM Employees
```

The INTO Clause

The INTO clause enables you to specify that the result set will be used to create a new table with the name defined in the clause. A *SELECT...INTO* statement can be used to combine data from several tables or views into one table. You can also use it to create a new table containing data selected from a linked server.

```
SELECT FirstName, LastName
INTO EmployeeNames FROM Employees
```

The result set that is generated by the statement creates the EmployeeNames table. The new table will contain the FirstName column and the LastName column, and those columns will contain the values from the Employees table.

The WHERE, GROUP BY, and HAVING Clauses

The WHERE and HAVING clauses in a *SELECT* statement control the rows from the source tables that are used to build the result set. The WHERE and HAVING clauses are filters. They specify a series of search conditions, and only those rows that meet the terms of the search conditions are used to build the result set. Those rows that meet the search conditions are said to be qualified to participate in the result set.

```
SELECT CustomerID, CompanyName
FROM Customers
WHERE Region = 'WA'
```

The HAVING clause is typically used in conjunction with the GROUP BY clause, although it can be specified without GROUP BY. The HAVING clause specifies more filters that are applied after the WHERE clause performs its filtering.

The GROUP BY Clause

The GROUP BY keywords are followed by a list of columns, known as the grouping columns. The GROUP BY clause restricts the rows of the result set. There is only one row for each distinct value in the grouping column or columns. Each result set row contains summary data related to the specific value of its grouping columns.
Typically, the HAVING clause is used with the GROUP BY clause, although HAVING can be specified separately.

Processing the WHERE, GROUP BY, and HAVING Clauses

Understanding the correct sequence in which the WHERE, GROUP BY, and HAVING clauses are applied helps in coding efficient queries:

- The WHERE clause is used to filter the rows that result from the operations specified in the FROM clause.
- The GROUP BY clause is used to group the output of the WHERE clause.
- The HAVING clause is used to filter rows from the grouped result.

The ORDER BY Clause

The ORDER BY clause sorts a query result by one or more columns (up to 8060 bytes). A sort can be *ascending* (ASC) or *descending* (DESC). If neither is specified, ASC is

assumed. If more than one column is named in the ORDER BY clause, sorts are nested. The following statement sorts the rows in the Titles table, first by publisher (in descending order), then by type (in ascending order within each publisher), and finally by price (also ascending, because DESC is not specified):

```
SELECT Pub_id, Type, Title_id, Price
FROM Titles
ORDER BY Pub_id DESC, Type, Price
```

To retrieve all data from the Titles table

```
SELECT * FROM Titles
```

To retrieve data from specific columns in the Titles table

```
SELECT Title_id, Title, Price, Ytd_sales
FROM Titles
```

To specify the condition that the result set must meet

```
SELECT Title_id, Title, Price, Ytd_sales
FROM Titles
WHERE Price > 10
```

To specify the order in which the result set appears

```
SELECT Title_id, Title, Price, Ytd_sales
FROM Titles
WHERE Price > 10
ORDER BY Price DESC, Title
```

To group data in a result set

```
SELECT Type, AVG(Price) AS AvgPrice
FROM Titles
WHERE Price > 10
GROUP BY Type
ORDER BY AvgPrice DESC
```

To create a table for the result set

```
SELECT Type, AVG(Price) AS AvgPrice
INTO TypeAvgPrice
FROM Titles
WHERE Price > 10
GROUP BY Type
ORDER BY AvgPrice DESC
```

Using a SELECT Subquery to Add Data

You can use a SELECT subquery in the *INSERT* statement to add values to a table from one or more other tables or views. A subquery enables you to add more than one row at a time.

A SELECT subquery in an *INSERT* statement is used to add subsets of existing data to a table, whereas a VALUES clause is used in an *INSERT* statement to add new data to a table.

```
INSERT INTO NewBooks (BookTitle, BookType)
SELECT Title, Type
FROM Titles
WHERE Type = 'mod_cook'
```

This *INSERT* statement uses the output of the SELECT subquery to provide the data that will be inserted into the NewBooks table.

Modifying the structure of table:

Adding new column:

Syntax:

ALTER TABLE < table name >
ADD (<new column name><data type> (<size>) ;
<New column name><data type> (<size>…);

Example:

Add the field Telephone No a number data type field that can hold a number up to 8 Digits in length.

ALTER TABLE BANK,
ADD(Telephone number (8));

Dropping a column from a table:

Syntax:

ALTER TABLE <table name> DROP
COLUMN <column name>; **Example:**

Alter table Bank DROP column Telephone No;

Note that using ALTER TABLE you cannot change:

Department of Information Technology

1. the name of the table
2. the name of the column
3. decrease the size of a column if table data exists.

a. Find out the customers who stay in an area ‘SA’, or area ‘BI’ or area ‘CH’.

FROM CUST WHERE AREA='SA' OR AREA='BI'OR AREA='CH';

CUST_ID	LNAME	FNAME	AREA	PHONE_NO
A01	Border	Allan	SA	723622
A03	Kumar	Ravi	BI	545621
A04	Rai	Sunita	CH	983724

b. List the movies in sorted order of their titles.

SELECT * FROM MOVIE ORDER BY Title ASC;

MOVIE_NO	TITLE	TYPE	STAR	PRICE
1	Carry On doctor	Comedy	Leslie Phollips	175
6	Coma	Suspense	Michael Douglas	100
7	Dracula	Horror	Gary Oldman	150.25
4	Home Alone	Comedy	Macaulay Culkin	150
3	Pretty Woman	Romance	Richard Gere	150.55
8	Quick Change	Comedy	Bill Murray	190
2	The Firm	Thriller	Tom Cruise	200
5	The Fugitive	Thriller	Harison Ford	200

c. Calculate the total price of all the movies.

SELECT SUM(Price) AMOUNT FROM MOVIE;

AMOUNT
1315.8

d. Determine the maximum and minimum movie prices. Rename the columns headings as MAXIMUM and MINIMUM while displaying the output.

SELECT MAX(Price) MAXIMUM ,MIN(Price) MINIMUM FROM MOVIE;

MAXIMUM	MINIMUM
200	100

e. Find the number of movies of each type.

SELECT Type,COUNT(*) FROM MOVIE GROUP BY Type;

TYPE	COUNT(*)
Comedy	3
Thriller	2
Romance	1
Horror	1
Suspense	1

f. Print the type and average price of each type.

SELECT Type,AVG(Price) FROM MOVIE GROUP BY Type;

TYPE	AVG(PRICE)
Comedy	171.66666666666666666666666666666666666667
Thriller	200
Romance	150.55
Horror	150.25
Suspense	100

g. Calculate the average price for each type that has average price > 150.

SELECT Type,AVG(Price) FROM MOVIE GROUP BY Type HAVING AVG(PRICE)>150;

TYPE	AVG(PRICE)
Comedy	171.66666666666666666666666666666666666667
Thriller	200
Romance	150.55
Horror	150.25

h. Calculate the average price of all movies where type is ‘comedy’ or ‘thriller and price>=150

i. Print the names of all customers whose customer id is between A01 and A05

SELECT Fname,Lname FROM CUST WHERE Cust_id>'A01' AND Cust_id<'A05';

FNAME	LNAME
Tina	Shields
Ravi	Kumar
Sunita	Rai

j. List the various movie types available from the movie table.

SELECT Type FROM MOVIE GROUP BY Type;

TYPE
Comedy
Thriller
Romance
Horror
Suspense

k. Find the movies whose price is greater than 150 and less than or equal to 200.

SELECT Title FROM MOVIE WHERE Price>150 AND Price <=200;

TITLE
Carry On doctor
The Firm
Pretty Woman
The Fugitive
Dracula
Quick Change

l. Retrieve the top 5 customers.

SELECT * FROM CUST WHERE ROWNUM<=5;

CUST_ID	LNAME	FNAME	AREA	PHONE_NO	AGE
A01	Border	Allan	SA	723622	-
A02	Shields	Tina	Mo	123784	-
A03	Kumar	Ravi	BI	545621	-
A04	Rai	Sunita	CH	983724	-
A05	-	Sachi	DR	253489	-

m. Retrieve the top 5 customers in the alphabetical order of first name.

SELECT *FROM CUST where ROWNUM<=5 ORDER BY Fname;

CUST_ID	LNAME	FNAME	AREA	PHONE_NO	AGE
A01	Border	Allan	SA	723622	-
A03	Kumar	Ravi	BI	545621	-
A05	-	Sachi	DR	253489	-
A04	Rai	Sunita	CH	983724	-
A02	Shields	Tina	Mo	123784	-

n. Alter the customer table to add the age of every customer.

ALTER TABLE CUST
ADD AGE NUMBER;
SELECT * FROM CUST;

CUST_ID	LNAME	FNAME	AREA	PHONE_NO	AGE
A01	Border	Allan	SA	723622	-
A02	Shields	Tina	Mo	123784	-
A03	Kumar	Ravi	BI	545621	-
A04	Rai	Sunita	CH	983724	-
A05	-	Sachi	DR	253489	-
A06	Smith	James	WA	634672	-

o. Create a table 'NewCustomer'.Insert the last names and first names of all the customers into this table using select subquery.

CREATE TABLE NEWCUST
(FNAME VARCHAR(20),LNAME VARCHAR(20));
INSERT INTO NEWCUST(FNAME ,LNAME)
(SELECT Fname,Lname FROM CUST);
SELECT * FROM NEWCUST;

FNAME	LNAME
Allan	Border
Tina	Shields
Ravi	Kumar
Sunita	Rai
Sachi	-
James	Smith

p. Print the information of invoice table in the following format for all records :

The Invoice No. Of Customer Id. {cust_id} is {inv_no} and Movie No. Is {movie_no}.

SELECT 'THE INVOICE NO. OF THE CUSTOMERS ID.'||Cust_id||' is '||Inv_no||'and
Movie no. is '||Movie_no AS PRINT FROM INVOICE

PRINT
THE INVOICE NO. OF THE CUSTOMERS ID.A01 is I01and Movie no. is 4
THE INVOICE NO. OF THE CUSTOMERS ID.A02 is I02and Movie no. is 3
THE INVOICE NO. OF THE CUSTOMERS ID.A02 is I03and Movie no. is 1
THE INVOICE NO. OF THE CUSTOMERS ID.A03 is I04and Movie no. is 6
THE INVOICE NO. OF THE CUSTOMERS ID.A04 is I05and Movie no. is 7
THE INVOICE NO. OF THE CUSTOMERS ID.A06 is I06and Movie no. is 2
THE INVOICE NO. OF THE CUSTOMERS ID.A05 is I07and Movie no. is 9
THE INVOICE NO. OF THE CUSTOMERS ID.A01 is I08and Movie no. is 9
THE INVOICE NO. OF THE CUSTOMERS ID.A03 is I09and Movie no. is 5
THE INVOICE NO. OF THE CUSTOMERS ID.A06 is I10and Movie no. is 8