

Database Design Project

Oracle Baseball League Store Database

Project Scenario:

You are a small consulting company specializing in database development. You have just been awarded the contract to develop a data model for a database application system for a small retail store called Oracle Baseball League (OBL).

The Oracle Baseball League store serves the entire surrounding community selling baseball kit. The OBL has two types of customer, there are individuals who purchase items like balls, cleats, gloves, shirts, screen printed t-shirts, and shorts. Additionally customers can represent a team when they purchase uniforms and equipment on behalf of the team.

Teams and individual customers are free to purchase any item from the inventory list, but teams get a discount on the list price depending on the number of players. When a customer places an order we record the order items for that order in our database.

OBL has a team of three sales representatives that officially only call on teams but have been known to handle individual customer complaints.

Section 6 Lesson 7 Exercise 1: Restricting Data Using WHERE

Limit rows using WHERE (S6L7 Objective 1)

In this exercise you will refine the data that is returned in your query by adding a WHERE clause to your SELECT statement.

Part 1: Using the WHERE Clause.

1. Using the unique customer number in the where clause display all columns for Maria Galant.

Code:

```
SELECT *  
FROM customers  
WHERE first_name = 'Maria' AND last_name = 'Galant';
```

Output:

The screenshot displays a SQL query editor with the following code:

```
SELECT start_date, TO_CHAR (start_time, 'HH24:MI:SS'), price, end_date, TO_CHAR (end_time, 'HH24:MI:SS')  
FROM price_history;  
  
INSERT INTO price_history (start_date, start_time, price, itm_number)  
VALUES(SYSDATE, SYSDATE, 99.99, 'premium_bar');  
  
SELECT start_date, TO_CHAR (start_time, 'HH24:MI:SS'), price, end_date, TO_CHAR (end_time, 'HH24:MI:SS')  
FROM price_history;  
  
SELECT *  
FROM customers  
WHERE first_name = 'Maria' AND last_name = 'Galant';
```

Below the editor, the 'Script Output' pane shows the execution results:

Task completed in 0.766 seconds

7 rows selected.

CTR_NU	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBE	CURRENT_BALANCE	SRE_TEM	LOYALT
c01986	margal87@delphiview.com	Maria	Galant	01442736589	125.65	sr03	t003

2. Display the first name, last name and customer number for all customers who have a current balance of greater than 100. Use an appropriate alias for your column headings.

Code:

```
SELECT  
first_name AS FirstName,  
last_name AS LastName,  
ctr_number AS CustomerNumber  
FROM  
customers  
WHERE  
current_balance > 100;
```

Output:

Worksheet

Query Builder

```
FROM price_history;

SELECT *
FROM customers
WHERE first_name = 'Maria' AND last_name = 'Galant';

SELECT
    first_name AS FirstName,
    last_name AS LastName,
    ctr_number AS CustomerNumber
FROM
    customers
WHERE
    current_balance > 100;
```

Script Output x Query Result x

Task completed in 0.257 seconds

31-MAY-17 16:35:30 149

7 rows selected.

CTR_NU	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBE	CURRENT_BALANCE	SRE_TEM	LOYALT
c01986	margal87@delphiview.com	Maria	Galant	01442736589	125.65	sr03	t003

FIRSTNAME	LASTNAME	CUSTOM
Robert	Thornberry	c00001
John	Doe	c00101
Maria	Galant	c01986

- Display the order id, date and time of all orders that were placed before the 28th of May 2019. Use an appropriate alias for your column headings.

Code:

```

SELECT
  id AS OrderID,
  odr_date AS OrderDate,
  odr_time AS OrderTime
FROM
  orders
WHERE
  TO_DATE(odr_date, 'DD-Mon-YYYY') < TO_DATE('28-May-2019', 'DD-Mon-YYYY');

```

Output:

```

customers
WHERE
    current_balance > 100;

SELECT
    id AS OrderID,
    odr_date AS OrderDate,
    odr_time AS OrderTime
FROM
    orders
WHERE
    TO_DATE(odr_date, 'DD-Mon-YYYY') < TO_DATE('28-May-2019', 'DD-Mon-YYYY');

```

Script Output x Query Result x

Task completed in 0.329 seconds

FIRSTNAME	LASTNAME	CUSTOM
Robert	Thornberry	c00001
John	Doe	c00101
Maria	Galant	c01986

ORDERID	ORDERDATE	ORDERTIME
or0101250	17-APR-17	17-APR-17
or0101350	24-MAY-17	24-MAY-17
or0101425	28-MAY-17	28-MAY-17
or0101681	02-JUN-17	02-JUN-17
or0101750	18-JUN-17	18-JUN-17

Part 2: Range Conditions: BETWEEN Operator

1. Display the inventory id, cost and number of units using appropriate aliases for all items that have a trade cost of between 3.00 and 15.00.

Code:

```
SELECT
id AS InventoryID,
cost AS Cost,
units AS NumberOfUnits
FROM
inventory_list
WHERE
cost BETWEEN 3.00 AND 15.00;
```

Output:

The screenshot shows a database query interface with two main panes. The top pane, titled 'Query Builder', contains the following SQL code:

```
WHERE
    TO_DATE(odr_date, 'DD-Mon-YYYY') < TO_DATE('28-May-2019', 'DD-Mon-YYYY');

SELECT
    id AS InventoryID,
    cost AS Cost,
    units AS NumberOfUnits
FROM
    inventory_list
WHERE
    cost BETWEEN 3.00 AND 15.00;
```

The bottom pane, titled 'Query Result', shows the output of the query. It contains two tables of data. The first table has columns ORDERID, ORDERDATE, and ORDERTIME. The second table has columns INVENTORYID, COST, and NUMBEROFUNITS.

ORDERID	ORDERDATE	ORDERTIME
or0101250	17-APR-17	17-APR-17
or0101350	24-MAY-17	24-MAY-17
or0101425	28-MAY-17	28-MAY-17
or0101681	02-JUN-17	02-JUN-17
or0101750	18-JUN-17	18-JUN-17

INVENTORYID	COST	NUMBEROFUNITS
11010230125	7.99	250
11010230126	5.24	87

Part 3: Membership Conditions: IN Operator

1. Display the inventory id, cost and number of units using appropriate aliases for all items that have 50, 100, 150 or 200 units in stock.

Code:

SELECT

id AS InventoryID,

cost AS Cost,

units AS NumberOfUnits

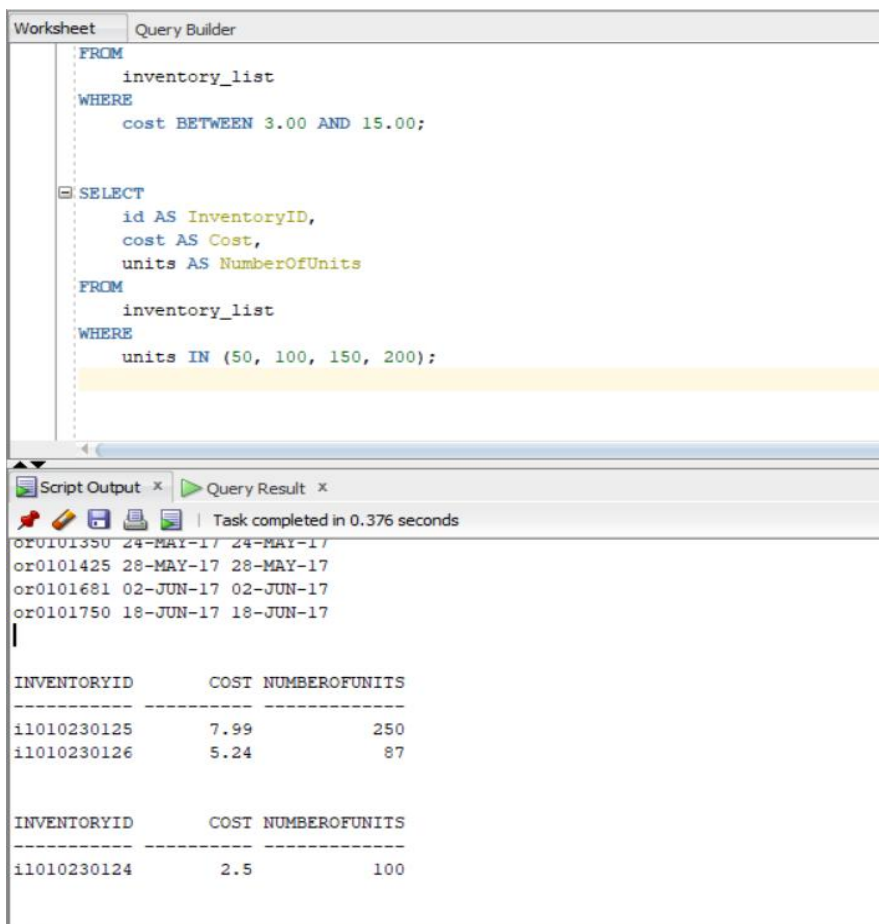
FROM

inventory_list

WHERE

units IN (50, 100, 150, 200);

Output:



The screenshot displays a database query builder interface. The top pane shows the query being constructed:

```
FROM
  inventory_list
WHERE
  cost BETWEEN 3.00 AND 15.00;

SELECT
  id AS InventoryID,
  cost AS Cost,
  units AS NumberOfUnits
FROM
  inventory_list
WHERE
  units IN (50, 100, 150, 200);
```

The bottom pane shows the query results. It includes a status bar indicating the task completed in 0.376 seconds. Below this, there is a list of inventory items with their IDs, costs, and unit counts.

INVENTORYID	COST	NUMBEROFUNITS
11010230125	7.99	250
11010230126	5.24	87
11010230124	2.5	100

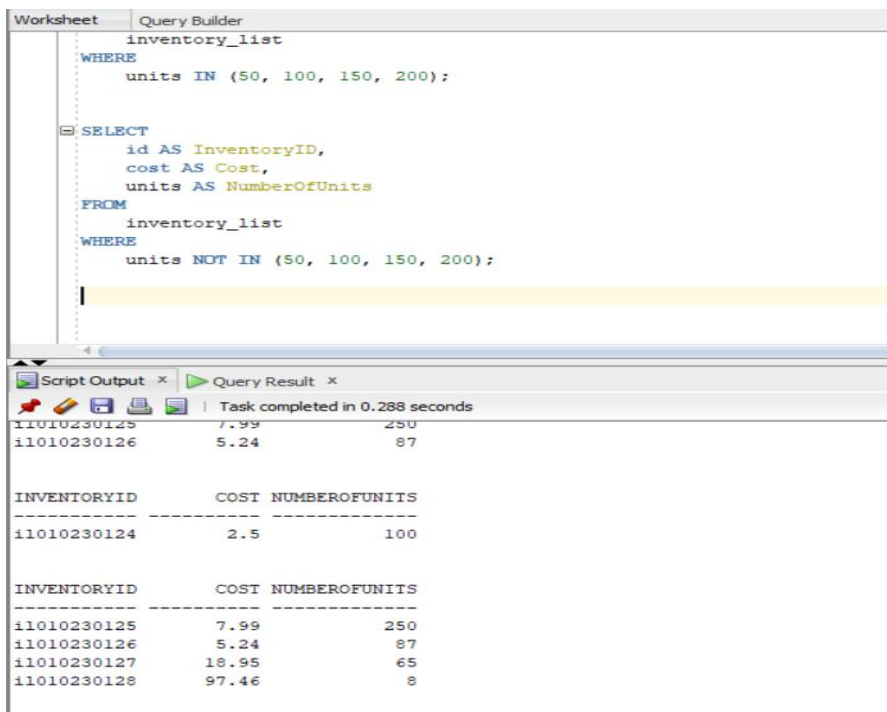
Part 4: Membership Conditions: NOT IN Operator

1. Display the inventory id, cost and number of units using appropriate aliases for all items that do not have 50, 100, 150 or 200 units in stock.

Code:

```
SELECT
id AS InventoryID,
cost AS Cost,
units AS NumberOfUnits
FROM
inventory_list
WHERE
units NOT IN (50, 100, 150, 200);
```

Output:



The screenshot shows a database query tool interface. The top pane displays the SQL query: `SELECT id AS InventoryID, cost AS Cost, units AS NumberOfUnits FROM inventory_list WHERE units NOT IN (50, 100, 150, 200);`. The bottom pane shows the query results in a table format. The table has three columns: INVENTORYID, COST, and NUMBEROFUNITS. The results are as follows:

INVENTORYID	COST	NUMBEROFUNITS
11010230125	7.99	250
11010230126	5.24	87
11010230124	2.5	100
11010230125	7.99	250
11010230126	5.24	87
11010230127	18.95	65
11010230128	97.46	8

Part 5: Pattern Matching: LIKE Operator

1. Display item number and name of all items that have a name that begins with g. Use an appropriate alias for your column headings.

Code:

SELECT

itm_number AS ItemNumber,

name AS ItemName

FROM

items

WHERE

name LIKE 'g%';

Output:

The screenshot shows an SQL Worksheet interface with two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying two SQL queries. The first query is a SELECT statement from the 'inventory_list' table, filtering for items where 'units' are not in the list (50, 100, 150, 200). The second query is a SELECT statement from the 'items' table, filtering for items where the 'name' starts with 'g' using the LIKE operator. Below the queries, there are two result windows. The 'Script Output' window shows the execution time of the first query as 0.276 seconds. The 'Query Result' window displays the results of the second query, showing a list of items with their item numbers and names.

```
FROM
inventory_list
WHERE
units NOT IN (50, 100, 150, 200);

SELECT
itm_number AS ItemNumber,
name AS ItemName
FROM
items
WHERE
name LIKE 'g%';
```

Script Output x Query Result x

Task completed in 0.276 seconds

INVENTORYID	COST	NUMBEROFUNITS
11010230124	2.5	100
11010230125	7.99	250
11010230126	5.24	87
11010230127	18.95	65
11010230128	97.46	8

ITEMNUMBER	ITEMNAME
im01101044	gloves
im01101047	game top

Part 6 : Pattern Matching: Combining Wildcard Characters with the LIKE Operator

1. Display item number and name of all items that have a name that contain a lowercase o. Use an appropriate alias for your column headings.

Code:

SELECT

itm_number AS ItemNumber,

name AS ItemName

FROM

items

WHERE

name LIKE '%o%';

Output:

The screenshot shows an SQL Worksheet interface with two queries. The first query is a SELECT statement with a WHERE clause using a wildcard pattern. The second query is a similar SELECT statement. Below the queries, the 'Query Result' tab shows the output of the second query, displaying item numbers and names.

```
SQL Worksheet History
0.296 seconds

Worksheet Query Builder

SELECT
  itm_number AS ItemNumber,
  name AS ItemName
FROM
  items
WHERE
  name LIKE 'g%';

SELECT
  itm_number AS ItemNumber,
  name AS ItemName
FROM
  items
WHERE
  name LIKE '%o%';

Script Output x Query Result x
Task completed in 0.296 seconds

ITEMNUMBER ITEMNAME
-----
im01101044 gloves
im01101047 game top

ITEMNUMBER ITEMNAME
-----
im01101044 gloves
im01101046 socks
im01101047 game top
```