

## Lab 4: Basic SELECT statements

The learning objectives of this lab are to

- Use arithmetic operators in SQL statements
- Select rows from a table with conditional restrictions
- Apply logical operators to have multiple conditions

### 4.1 Using arithmetic operators in SQL statements

SQL commands are often used in conjunction with arithmetic operators. As you perform mathematical operations on attributes, remember the rules of precedence. As the name suggests, the **rules of precedence** are the rules that establish the order in which computations are completed. For example, note the order of the following computational sequence:

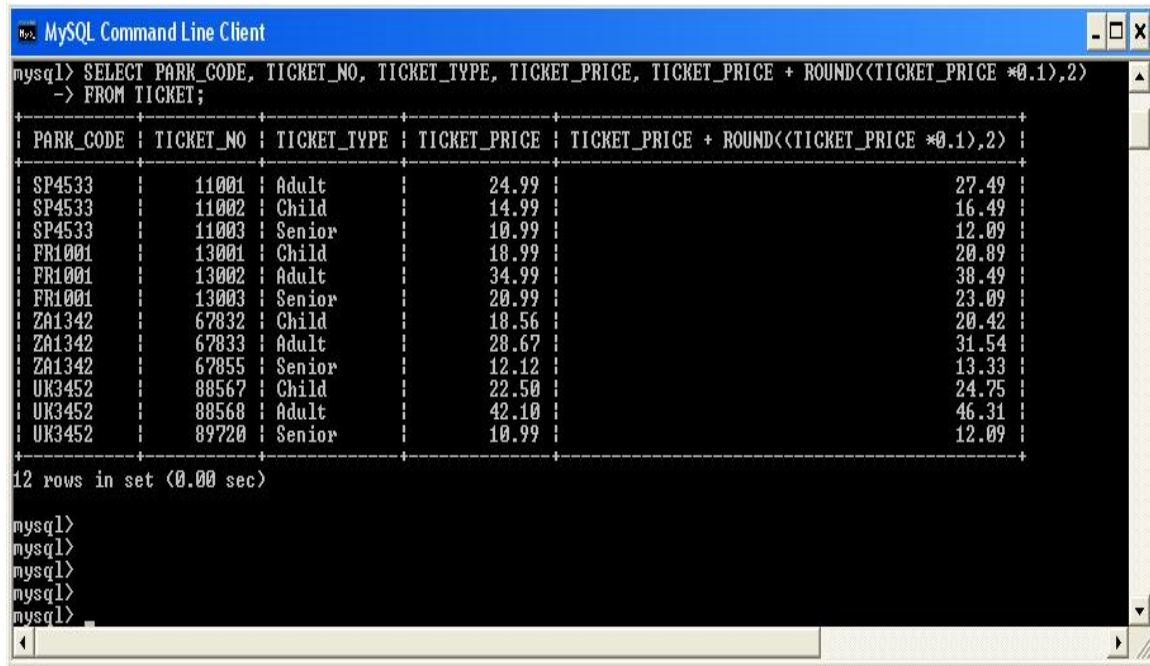
1. Perform operations within parentheses
2. Perform power operations
3. Perform multiplications and divisions
4. Perform additions and subtractions

**Task 4.1** Suppose the owners of all the theme parks wanted to compare the current ticket prices, with an increase in the price of each ticket by 10%. To generate this query type:

```
SELECT PARK_CODE, TICKET_NO, TICKET_TYPE, TICKET_PRICE,  
TICKET_PRICE + ROUND((TICKET_PRICE * 0.1), 2)
```

FROM TICKET;

The output for this query is shown in Figure 19. The ROUND function is used to ensure the result is displayed to two decimal places.



```
mysql> SELECT PARK_CODE, TICKET_NO, TICKET_TYPE, TICKET_PRICE, TICKET_PRICE + ROUND((TICKET_PRICE * 0.1), 2)
-> FROM TICKET;
```

PARK_CODE	TICKET_NO	TICKET_TYPE	TICKET_PRICE	TICKET_PRICE + ROUND((TICKET_PRICE * 0.1), 2)
SP4533	11001	Adult	24.99	27.49
SP4533	11002	Child	14.99	16.49
SP4533	11003	Senior	10.99	12.09
FR1001	13001	Child	18.99	20.89
FR1001	13002	Adult	34.99	38.49
FR1001	13003	Senior	20.99	23.09
ZA1342	67832	Child	18.56	20.42
ZA1342	67833	Adult	28.67	31.54
ZA1342	67855	Senior	12.12	13.33
UK3452	88567	Child	22.50	24.75
UK3452	88568	Adult	42.10	46.31
UK3452	89720	Senior	10.99	12.09

12 rows in set (0.00 sec)

```
mysql>
mysql>
mysql>
mysql>
mysql>
```

**Figure 19: Output showing 10% increase in ticket prices**

You will see in Figure 19 that the last column is named after the arithmetic expression in the query. To rename the column heading, a column alias needs to be used. Modify the query as follows and note that the name of the heading has changed to PRICE\_INCREASE when you execute the following query.

```
SELECT PARK_CODE, TICKET_NO, TICKET_TYPE, TICKET_PRICE,
TICKET_PRICE + ROUND((TICKET_PRICE * 0.1), 2) PRICE_INCREASE
FROM TICKET;
```

**Note**

When dealing with column names that require spaces, the optional keyword AS can be used. For example:

```
SELECT PARK_CODE, TICKET_NO, TICKET_TYPE, TICKET_PRICE,
TICKET_PRICE + ROUND((TICKET_PRICE *0.1),2) AS
“PRICE INCREASE”
FROM TICKET;
```

## 4.2 Selecting rows with conditional restrictions

Numerous conditional restrictions can be placed on the selected table contents in the WHERE clause of the SELECT statement. For example, the comparison operators shown in Table 1 can be used to restrict output.

**Table 1 Comparison Operators**

SYMBOL	MEANING
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> or !=	Not equal to
BETWEEN	Used to check if an attribute is within a range.
IN	Used to check if an attribute value matches any value within a list.

LIKE	Used to check if an attribute value matches a given string pattern.
IS NULL / IS NOT NULL	Used to check if an attribute is NULL / is not NULL.

We will now explore some of these conditional operators using examples.

## Greater than

The following example uses the “greater than” operator to display the theme park code, ticket price and ticket type of all tickets where the ticket price is greater than €20.00.

```
SELECT PARK_CODE, TICKET_TYPE, TICKET_PRICE
FROM TICKET
WHERE TICKET_PRICE > 20;
```

The output is shown in Figure 20.

```

MySQL Command Line Client
mysql> SELECT PARK_CODE, TICKET_TYPE, TICKET_PRICE
-> FROM TICKET
-> WHERE TICKET_PRICE > 20;
+-----+-----+-----+
| PARK_CODE | TICKET_TYPE | TICKET_PRICE |
+-----+-----+-----+
| SP4533    | Adult       | 24.99        |
| FR1001    | Adult       | 34.99        |
| FR1001    | Senior      | 20.99        |
| ZA1342    | Adult       | 28.67        |
| UK3452    | Child       | 22.50        |
| UK3452    | Adult       | 42.10        |
+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
  
```

**Figure 20: Tickets costing greater than €20.00**

**Task 4.2** Type in and execute the query and test out the greater than operator. Do you get the same results has shown in Figure 20?

**Task 4.3** Modify the query you have just executed to display tickets that are less than €30.00.

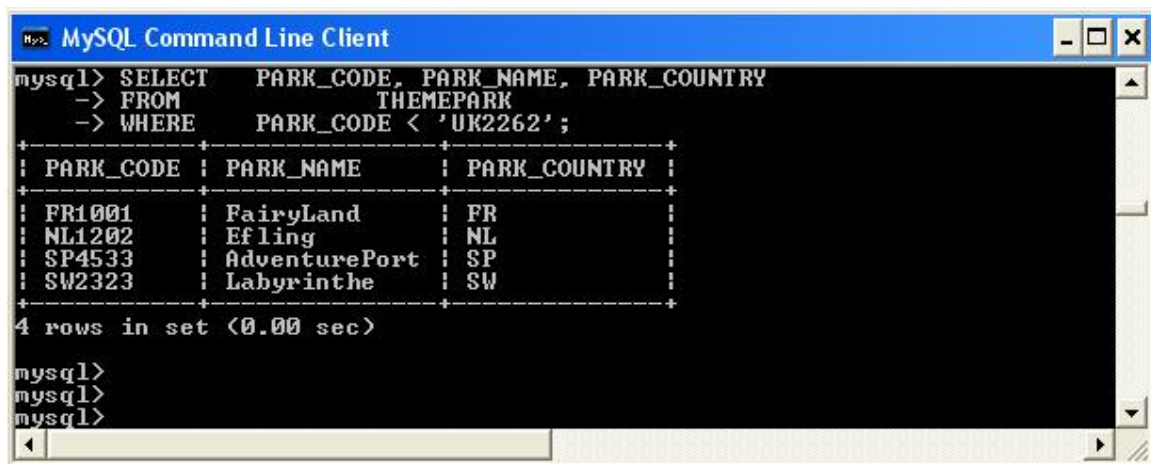
### Character comparisons

Comparison operators may even be used to place restrictions on character-based attributes.

**Task 4.4** Execute the following query which produces a list of all rows in which the PARK\_CODE is alphabetically less than UK2262. (Because the ASCII code value for the letter *B* is greater than the value of the letter *A*, it follows that *A* is less than *B*.)

Therefore, the output will be generated as shown in Figure 21.

```
SELECT    PARK_CODE, PARK_NAME, PARK_COUNTRY
FROM      THEMEPARK
WHERE     PARK_CODE < 'UK2262';
```



```
mysql> SELECT    PARK_CODE, PARK_NAME, PARK_COUNTRY
-> FROM      THEMEPARK
-> WHERE     PARK_CODE < 'UK2262';
```

PARK_CODE	PARK_NAME	PARK_COUNTRY
FR1001	FairyLand	FR
NL1202	Ef ling	NL
SP4533	AdventurePort	SP
SW2323	Labyrinthe	SW

```
4 rows in set (0.00 sec)

mysql>
mysql>
mysql>
```

**Figure 21: Example of character comparison**

### BETWEEN

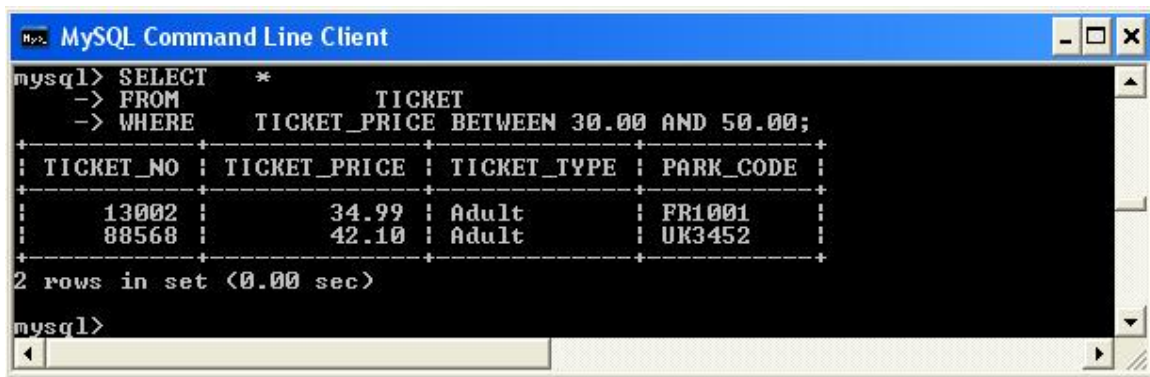
The operator BETWEEN may be used to check whether an attribute value is within a range of values. For example, if you want to see a listing for all tickets whose prices are between €30 and €50, use the following command sequence:

```
SELECT      *

FROM        TICKET

WHERE       TICKET_PRICE BETWEEN 30.00 AND 50.00;
```

Figure 22 shows the output you should see for this query.

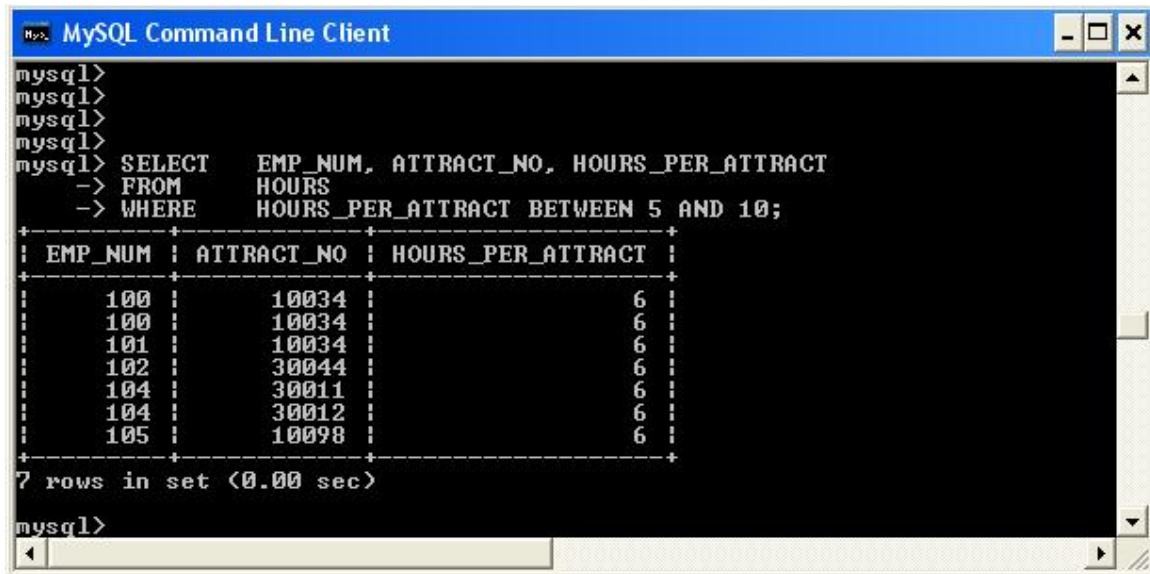


```
mysql> SELECT      *
-> FROM        TICKET
-> WHERE       TICKET_PRICE BETWEEN 30.00 AND 50.00;
+-----+-----+-----+-----+
| TICKET_NO | TICKET_PRICE | TICKET_TYPE | PARK_CODE |
+-----+-----+-----+-----+
| 13002 | 34.99 | Adult | FR1001 |
| 88568 | 42.10 | Adult | UK3452 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

**Figure 22: Displaying ticket prices BETWEEN two values.**

**Task 4.5** Write a query which displays the employee number, attraction no, the hours worked per attraction and the date worked where the hours worked per attraction is between 5 and 10. Hint you will need to select data from the HOURS table. The output for the query is shown in Figure 23.



```

mysql>
mysql>
mysql>
mysql> SELECT EMP_NUM, ATTRACT_NO, HOURS_PER_ATTRACT
        -> FROM HOURS
        -> WHERE HOURS_PER_ATTRACT BETWEEN 5 AND 10;
+-----+-----+-----+
| EMP_NUM | ATTRACT_NO | HOURS_PER_ATTRACT |
+-----+-----+-----+
| 100     | 10034      | 6                 |
| 100     | 10034      | 6                 |
| 101     | 10034      | 6                 |
| 102     | 30044      | 6                 |
| 104     | 30011      | 6                 |
| 104     | 30012      | 6                 |
| 105     | 10098      | 6                 |
+-----+-----+-----+
7 rows in set (0.00 sec)

mysql>

```

Figure 23: Output for Task 4.5

## IN

The IN operator is used to test for values which are in a list. The following query finds only the rows in the SALES\_LINE table that match up to a specific sales transaction. i.e. TRANSACTION\_NO is either 12781 or 67593.

```

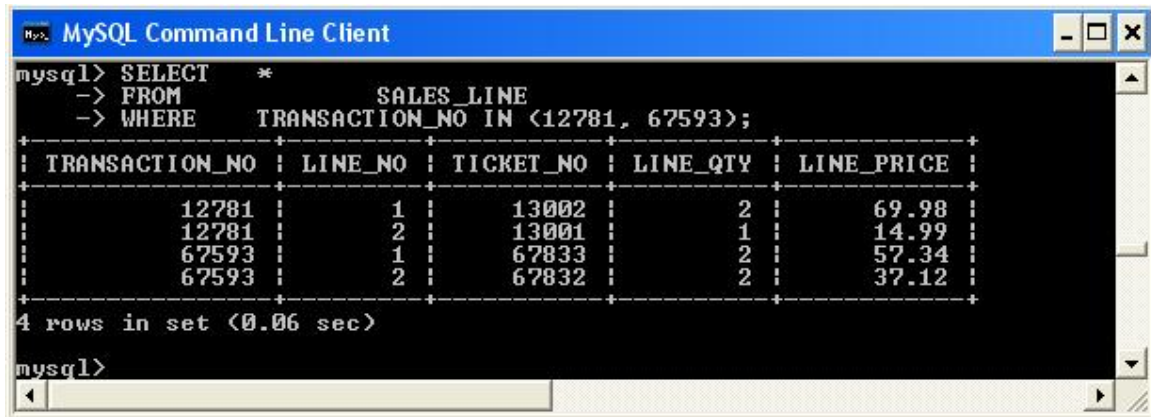
SELECT      *

FROM        SALES_LINE

WHERE       TRANSACTION_NO IN (12781, 67593);

```

The result of this query is shown in Figure 24.



```
mysql> SELECT *
-> FROM      SALES_LINE
-> WHERE      TRANSACTION_NO IN (12781, 67593);
```

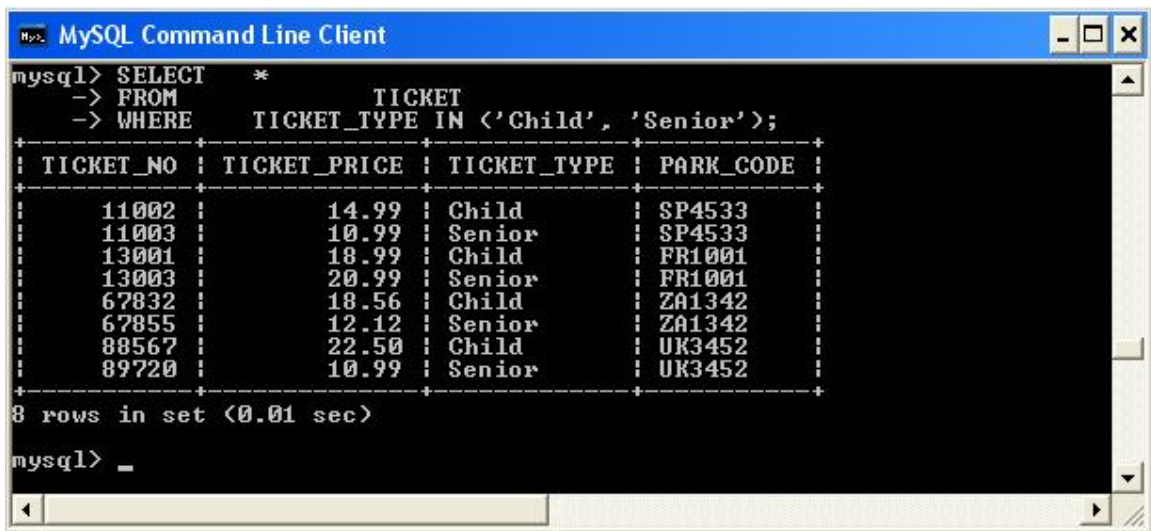
TRANSACTION_NO	LINE_NO	TICKET_NO	LINE_QTY	LINE_PRICE
12781	1	13002	2	69.98
12781	2	13001	1	14.99
67593	1	67833	2	57.34
67593	2	67832	2	37.12

```
4 rows in set (0.06 sec)

mysql>
```

Figure 24 Selecting rows using the IN command

**Task 4.6** Write a query to display all tickets that are of type Senior or Child. Hint: Use the TICKET table. The output you should see is shown in Figure 25.



```
mysql> SELECT *
-> FROM      TICKET
-> WHERE      TICKET_TYPE IN ('Child', 'Senior');
```

TICKET_NO	TICKET_PRICE	TICKET_TYPE	PARK_CODE
11002	14.99	Child	SP4533
11003	10.99	Senior	SP4533
13001	18.99	Child	FR1001
13003	20.99	Senior	FR1001
67832	18.56	Child	ZA1342
67855	12.12	Senior	ZA1342
88567	22.50	Child	UK3452
89720	10.99	Senior	UK3452

```
8 rows in set (0.01 sec)

mysql> _
```

Figure 25. Output for Task 4.6

## LIKE

The LIKE operator is used to find patterns within string attributes. Standard SQL allows you to use the percent sign (%) and underscore (\_) wildcard characters to make matches

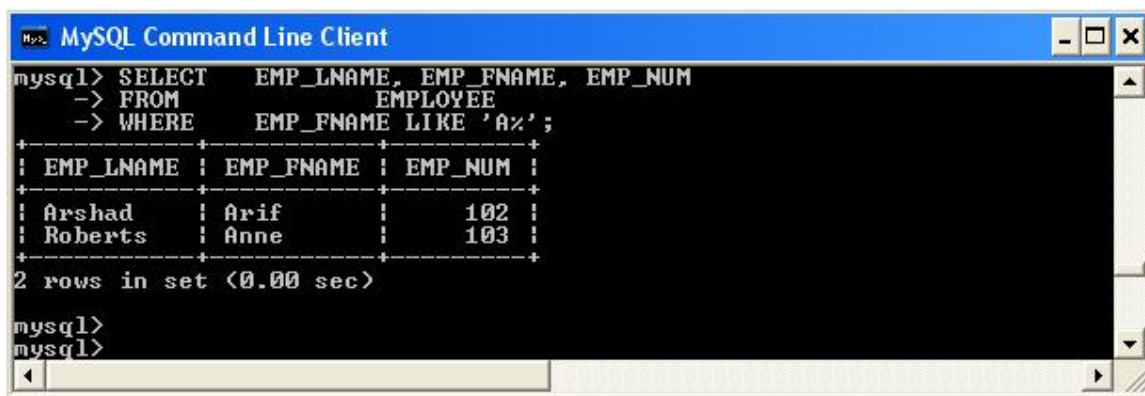


when the entire string is not known. % means any and all *following* characters are eligible while \_ means any *one* character may be substituted for the underscore.

**Task 4.7** Enter the following query which finds all EMPLOYEE rows whose first names begin with the letter A.

```
SELECT      EMP_LNAME, EMP_FNAME, EMP_NUM
FROM        EMPLOYEE
WHERE       EMP_FNAME LIKE 'A%';
```

Figure 26 shows the output you should see for this query.

A screenshot of the MySQL Command Line Client window. The title bar reads "MySQL Command Line Client". The command prompt shows the following SQL query: 

```
mysql> SELECT EMP_LNAME, EMP_FNAME, EMP_NUM
-> FROM EMPLOYEE
-> WHERE EMP_FNAME LIKE 'A%';
```

 The output is displayed in a table format with three columns: EMP\_LNAME, EMP\_FNAME, and EMP\_NUM. The data rows are: Arshad, Arif, 102 and Roberts, Anne, 103. Below the table, it says "2 rows in set (0.00 sec)". The prompt "mysql>" is shown again at the bottom.

EMP_LNAME	EMP_FNAME	EMP_NUM
Arshad	Arif	102
Roberts	Anne	103

**Figure 26 Query using the LIKE command**

**Task 4.8** Write a query which finds all Theme Parks that have a name ending in 'Land'.

The output you should see is shown in Figure 27.



```

mysql> select *
-> from themepark
-> WHERE PARK_NAME LIKE '%Land';
+-----+-----+-----+-----+
| PARK_CODE | PARK_NAME | PARK_CITY | PARK_COUNTRY |
+-----+-----+-----+-----+
| FR1001    | FairyLand | PARIS     | FR            |
| UK2622    | MiniLand  | WINDSOR   | UK            |
| UK3452    | PleasureLand | STOKE    | UK            |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>

```

Figure 27 Solution to Task 4.8

### NULL and IS NULL

IS NULL is used to check for a null attribute value. In the following example, the query lists all attractions that do not have an attraction name assigned (ATTRACT\_NAME is null). The query could be written as:

```

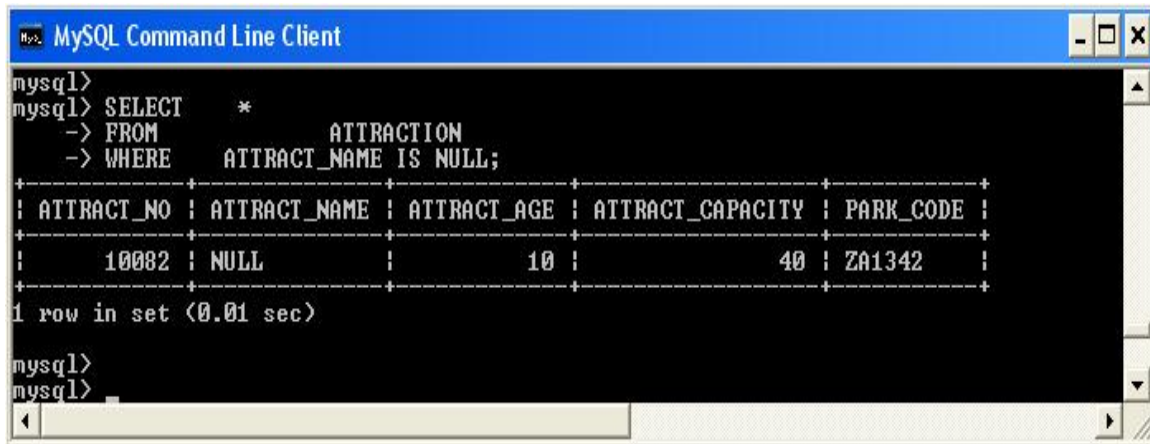
SELECT      *

FROM        ATTRACTION

WHERE       ATTRACT_NAME IS NULL;

```

The output for this query is shown in Figure 28.



```

mysql>
mysql> SELECT      *
-> FROM            ATTRACTION
-> WHERE           ATTRACT_NAME IS NULL;
+-----+-----+-----+-----+-----+
| ATTRACT_NO | ATTRACT_NAME | ATTRACT_AGE | ATTRACT_CAPACITY | PARK_CODE |
+-----+-----+-----+-----+-----+
| 10082      | NULL         | 10          | 40                | ZA1342    |
+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql>
mysql>

```

**Figure 28 Listing all Attractions with no name**

## **Logical Operators**

SQL allows you to have multiple conditions in a query through the use of logical operators: AND, OR and NOT. NOT has the highest precedence, followed by AND, and then followed by OR. However, you are strongly recommended to use parentheses to clarify the intended meaning of the query.

### **AND**

This logical AND connective is used to set up a query where there are two conditions which must be met for the query to return the required row(s). The following query displays the employee number (EMP\_NUM) and the attraction number (ATTRACT\_NUM) for which the numbers of hours worked (HOURS\_PER\_ATTRACT) by the employee is greater than 3 and the date worked (DATE\_WORKED) is after 18<sup>th</sup> May 2007.

```
SELECT    EMP_NUM, ATTRACT_NO
FROM      HOURS
WHERE     HOURS_PER_ATTRACT > 3
AND       DATE_WORKED > '18-MAY-07';
```

This query will produce the output shown in Figure 29.

```

mysql>
mysql> SELECT      EMP_NUM, ATTRACT_NO
-> FROM            HOURS
-> WHERE           HOURS_PER_ATTRACT > 3
-> AND             DATE_WORKED > '18-MAY-07';
+-----+-----+
| EMP_NUM | ATTRACT_NO |
+-----+-----+
| 100     | 10034      |
| 100     | 10034      |
| 101     | 10034      |
| 102     | 30044      |
| 104     | 30011      |
| 104     | 30012      |
| 105     | 10098      |
+-----+-----+
7 rows in set, 1 warning (0.00 sec)

mysql>
mysql>
mysql>
mysql>

```

Figure 29 Query results using the AND operator

**Task 4.9** Enter the query above and check you results with those shown in Figure 29.

**Task 4.10** Write a query which displays the details of all attractions which are suitable for children aged 10 or under and have a capacity of less than 100. You should not display any information for attractions which currently have no name. Your output should correspond to that shown in Figure 30.

```

mysql> SELECT      *
-> FROM            ATTRACTION
-> WHERE           ATTRACT_AGE <= 10
-> AND             ATTRACT_CAPACITY < 100
-> AND             ATTRACT_NAME IS NOT NULL;
+-----+-----+-----+-----+-----+
| ATTRACT_NO | ATTRACT_NAME | ATTRACT_AGE | ATTRACT_CAPACITY | PARK_CODE |
+-----+-----+-----+-----+-----+
| 10056      | SpinningTeacups | 4          | 62               | FR1001    |
| 30012      | Pirates        | 10         | 42               | UK3452    |
| 30044      | UnderSeaWord   | 4          | 80               | UK3452    |
| 98764      | GoldRush       | 5          | 80               | ZA1342    |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

Figure 30: Query results for Task 4.10

**OR**

If you wanted to list the names and countries of all Theme parks where of invoice numbers where PARK\_COUNTRY = 'FR' OR PARK\_COUNTRY = 'UK' you would write the following query.

```
SELECT PARK_NAME, PARK_COUNTRY
```

```
FROM THEMEPARK
```

```
WHERE PARK_COUNTRY = 'FR'
```

```
OR PARK_COUNTRY = 'UK';
```

The output is shown in Figure 31.

```

mysql> SELECT PARK_NAME, PARK_COUNTRY
-> FROM THEMEPARK
-> WHERE PARK_COUNTRY = 'FR'
-> OR PARK_COUNTRY = 'UK';
+-----+-----+
| PARK_NAME | PARK_COUNTRY |
+-----+-----+
| FairyLand | FR           |
| Miniland  | UK           |
| PleasureLand | UK          |
+-----+-----+
3 rows in set (0.00 sec)

mysql>
mysql>

```

**Figure 31: Query results using the OR operator;**

When using AND and OR in the same query it is advisable to use parentheses to make explicit the precedence.

**Task 4.11** Test the following query and check your output with that shown in Figure 32.

Can you work out what this query is doing?

```

SELECT      *
FROM        ATTRACTION
WHERE       (PARK_CODE LIKE 'FR%'
AND ATTRACT_CAPACITY <50) OR (ATTRACT_CAPACITY > 100);

```

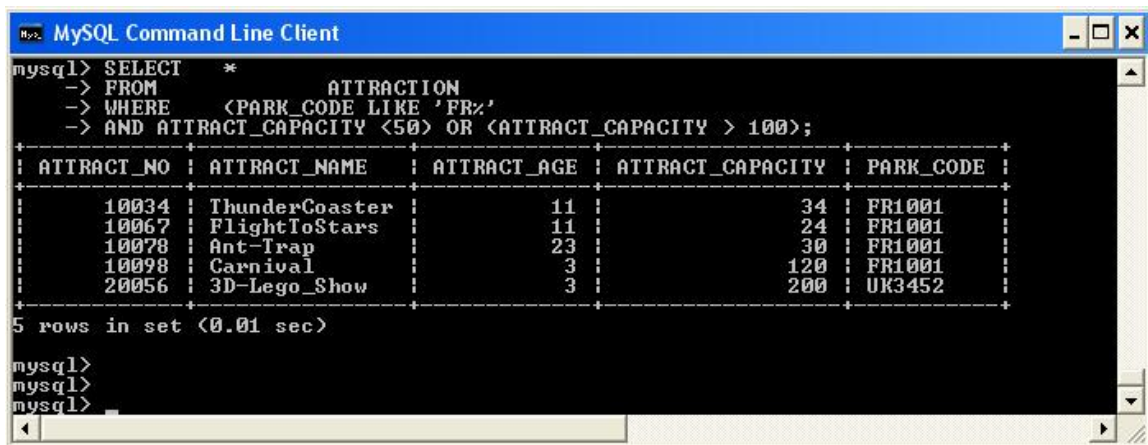


Figure 32: AND and OR example

## NOT

The logical operator **NOT** is used to negate the result of a conditional expression. If you want to see a listing of all rows for which EMP\_NUM is not 106, the query would look like:

```

SELECT      *
FROM EMPLOYEE
WHERE       NOT (EMP_NUM = 106);

```

The results of this query are shown in Figure 33. Note that the condition is enclosed in parentheses; that practice is optional, but it is highly recommended for clarity.

```

mysql>
mysql> SELECT *
  -> FROM EMPLOYEE
  -> WHERE NOT (EMP_NUM = 106);
+-----+-----+-----+-----+-----+-----+-----+-----+
| EMP_NUM | EMP_TITLE | EMP_LNAME | EMP_FNAME | EMP_DOB | EMP_HIRE_DATE | EMP_AREA_CODE | EMP_PHONE | PARK_CODE |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 100 | Ms | Calderdale | Emma | 1972-06-15 | 1992-03-15 | 0181 | 324-9134 | FR1001 |
| 101 | Ms | Ricardo | Marshal | 1978-03-19 | 1996-04-25 | 0181 | 324-4472 | UK3452 |
| 102 | Mr | Arshad | Arif | 1969-11-14 | 1990-12-20 | 7253 | 675-8993 | FR1001 |
| 103 | Ms | Roberts | Anne | 1974-10-16 | 1994-08-16 | 0181 | 898-3456 | UK3452 |
| 104 | Mr | Denver | Enrica | 1980-11-08 | 2001-10-20 | 7253 | 504-4434 | ZA1342 |
| 105 | Ms | Nanova | Mirrelle | 1990-03-14 | 2006-11-08 | 0181 | 898-3243 | FR1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
mysql>

```

**Figure 33: Listing all employees except EMP\_NUM=106**

### Exercises

**E4.1** Write a query to display all Theme Parks except those in the UK.

**E4.2** Write a query to display all the sales that occurred on the 18<sup>th</sup> May 2007.

**E4.3** Write a query to display the ticket prices between €20 AND €30.

**E4.4** Display all attractions that have a capacity of more than 60 at the Theme Park FR1001.

**E4.5** Write a query to display the hourly rate for each attraction where an employee had worked, along with the hourly rate increased by 20%. Your query should only

display the ATTRACT\_NO, HOUR\_RATE and the HOUR\_RATE with the 20% increase.