Lab 5 : COLUMN/AGGREGATE/GROUP FUNCTION

FUNCTION	OUTPUT
COUNT	The number of rows containing non-null values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column

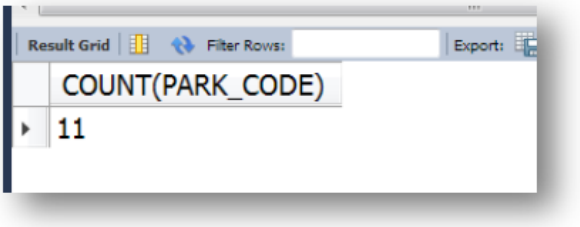
COUNT

The COUNT function is used to tally the number of non-null values of an attribute. COUNT can be used in conjunction with the DISTINCT clause

1. If you wanted to find out how many theme parks contained attractions from the ATTRACTION table.

```
SELECT COUNT(PARK_CODE)
FROM ATTRACTION;
```

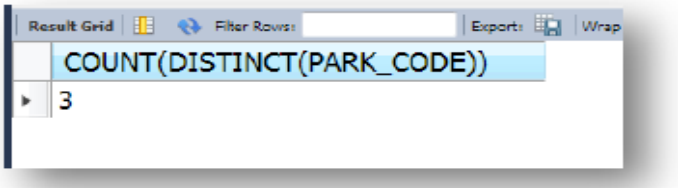
The query would return 11 rows as shown as below:



2. if you wanted to know how many different Theme parks were in the ATTRACTION table, you would modify the query.

```
SELECT COUNT(DISTINCT(PARK_CODE))
FROM ATTRACTION;
```

The query would return as below:



3. Write a query that displays the number of distinct employees in the HOURS table. You should label the column “Number of Employees”.

   SELECT COUNT (DISTINCT(EMP_NUM)) AS ‘ Number of Employees’
   FROM HOURS;

   COUNT always returns the number of non-null values in the given column. Another use for the COUNT function is to display the number of rows returned by a query, including the rows that contain rows using the syntax COUNT(*).

4. Enter the following two queries and examine their output.
   SELECT COUNT(*)
   FROM ATTRACTION;

   The query would return as below:

COUNT(*)
11

   SELECT COUNT(ATTRACT_NAME)
   FROM ATTRACTION;

   The query would return as below:

COUNT(ATTRACT_NAME)
10

MIN and MAX

Function to compute the highest and lowest value

1. What is the highest and lowest ticket price sold in all Theme parks.

   SELECT MIN(TICKET_PRICE),max(TICKET_PRICE)
   FROM TICKET;

   The query would return as below:

MIN(TICKET_PRICE)	MAX(TICKET_PRICE)
10.99	42.10

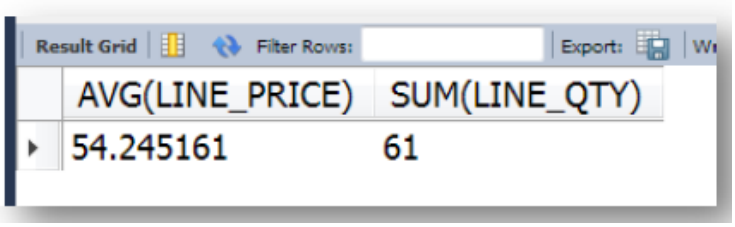
SUM and AVG

The SUM function computes the total sum for any specified attribute, using whatever condition(s) you have imposed. The AVG function calculates the arithmetic mean (average) for a specified attribute.

1. Displays the average amount spent on Theme park tickets per customer (LINE_PRICE) and the total number of tickets purchase (LINE_QTY).

```
SELECT AVG(LINE_PRICE), SUM(LINE_QTY)
FROM SALES_LINE;
```

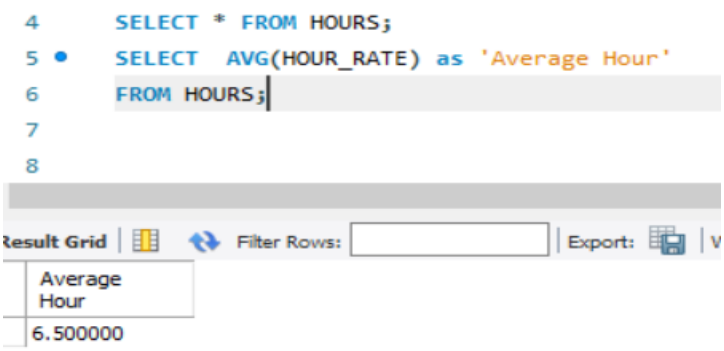
The query would return as below:



AVG(LINE_PRICE)	SUM(LINE_QTY)
54.245161	61

2. Write a query that displays the average hourly rate that has been paid to all employees. Hint use the HOURS table. Replace calculate column using 'Average Hour'.

```
SELECT  AVG(HOUR_RATE) as 'Average Hour'
FROM HOURS;
```



Average Hour
6.500000

3. Write a query that displays the average attraction age for all attractions where the PARK_CODE = 'UK3452'.

```
SELECT AVG(ATTRACT_AGE)
FROM ATTRACTION
WHERE PARK_CODE  = 'UK3452';
```

GROUP BY

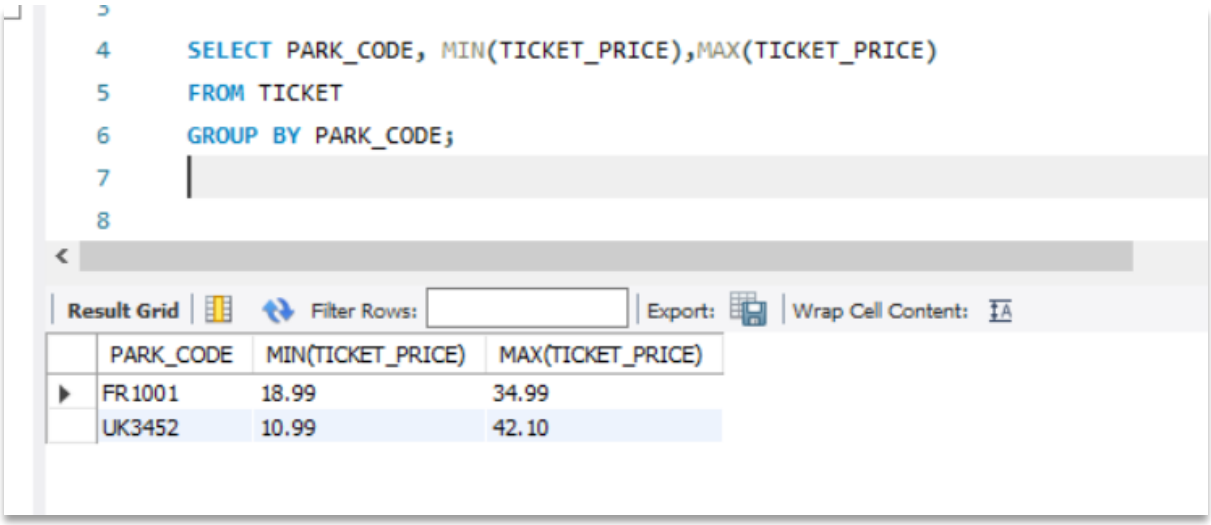
The GROUP BY clause is generally used when you have attribute columns combined with aggregate functions in the SELECT statement.

It is valid only when used in conjunction with one of the SQL aggregate functions, such as COUNT, MIN, MAX, AVG and SUM.

The GROUP BY clause appears after the WHERE statement. When using GROUP BY you should include all the attributes that are in the SELECT statement that do not use an aggregate function.

1. Displays park code , the minimum and maximum ticket price of all parks.

   ```
   SELECT PARK_CODE, MIN(TICKET_PRICE), MAX(TICKET_PRICE)
   FROM TICKET
   GROUP BY PARK_CODE;
   ```



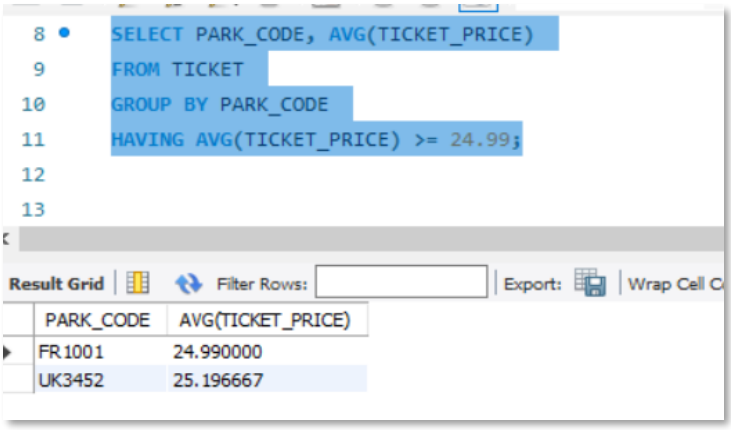
2. Enter the query above and check the results against the output. What happens if you miss out the GROUP BY clause?

HAVING

The HAVING clause is an extension to the GROUP BY clause and is applied to the output of a GROUP BY operation

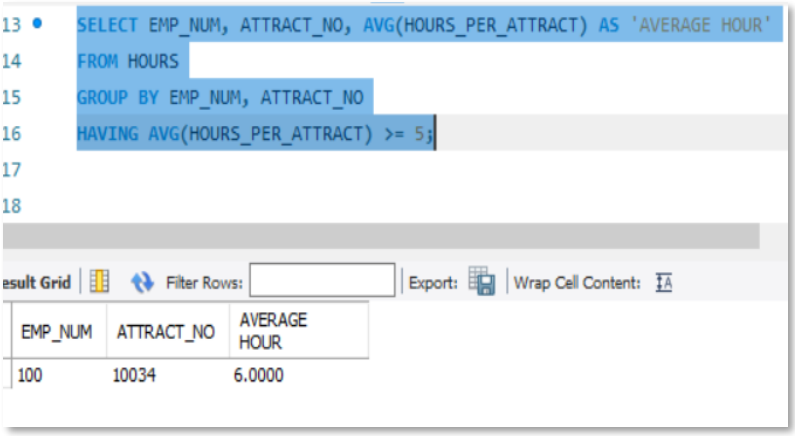
1. You wanted to list the average ticket price at each Theme Park but wanted to limit the listing to Theme Parks whose average ticket price was greater or equal to €24.99.

SELECT PARK_CODE, AVG(TICKET_PRICE)
FROM TICKET
GROUP BY PARK_CODE
HAVING AVG(TICKET_PRICE) >= 24.99;



2. Using the HOURS table, write a query to display the employee number (EMP_NUM), the attraction number (ATTRACT-NO) and the average hours worked per attraction (HOURS_PER_ATTRACT). Limiting the result to where the average hours worked per attraction is greater or equal to 5. Replace calculated column name using 'AVERAGE HOUR'

SELECT EMP_NUM, ATTRACT_NO, AVG(HOURS_PER_ATTRACT) AS 'AVERAGE HOUR'
FROM HOURS
GROUP BY EMP_NUM, ATTRACT_NO
HAVING AVG(HOURS_PER_ATTRACT) >= 5;



EXERCISE:

1. Write a query to calculate the number of unique park code that exist in the TICKET table, replace calculated column with ‘Park_Count’.

2. Display the employee numbers of all employees and the total of payment (Hour rate multiply Hour per attract) they have worked. Use ‘Total payment’ for calculated column. Sort the output according employee number descending.

3. Display the employee numbers of all employees and the total hours they have worked. Use ‘Total Hours’ for calculated column. Sort the output according employee number descending.

4. Show the attraction number and the minimum and maximum hourly rate for each attraction. Use alias MIN and MAX for calculated column.

5. Write a query to show the transaction numbers and AVERAGE line prices (use the SALES_LINE table). Display average value that are greater than €50 only.

6. Display Transaction Number, Sale date information from the SALES table. Calculate the number of sales after 1 January 2022. Sort the result in descending order of the sale date.

7. Using the TICKET table, write a query to display the park code and the average ticket price. Limiting the average ticket price greater or equal to 20. Replace calculated column name using ‘AVERAGE PRICE’.