

Experiment No.08

PART A

(PART A: TO BE REFERRED BY STUDENTS)

A.1 Aim: To implement group by, aggregate functions, joins, nested subqueries for solving queries.

Task :

Task1:

Step 1: Create database mythemepark_yourrollno

Step 2: Run sqlsript “mythemepark.sql” – Creates all tables

Step 3: Write Relational schema of themepark

Step3: Run sqlsript “mythemeparkdata.sql” – Insertion of data

Step4: Execute following sql statements

Select * from THEMEPARK;

Select * from EMPLOYEE;

Select * from TICKET;

Select * from ATTRACTION;

Select * from HOURS;

Select * from SALES;

Select * from SALES_LINE;

Task2:

1	Display ticket type and park code, ticket price where ticket price is more than 15 for child category. Sort the result in descending order of ticket number.
2	Display all parks whose name ends with Land.
3	Display the attractions whose attractive name is Null.
4	List all rows for which EMP_NUM is not 106.

5	Find out how many unique theme parks are in the ATTRACTION table.
6	<p>Enter the following two queries and examine their outputs. Can you explain why the number of rows returned is different?</p> <pre>SELECT COUNT(*)FROM ATTRACTION;</pre> <pre>SELECT COUNT(ATTRACT_NAME)FROM ATTRACTION;</pre>
7	Display the minimum and maximum ticket price of each park.
8	Join the THEMEPARK and TICKET tables with common PARK_CODE (without the join keyword; use cartesian product).
9	Perform a natural join of the SALES and SALES_LINE tables and return only selected attributes: TRANSACTION_NO, SALE_DATE, LINE_NO, LINE_QTY, LINE_PRICE.
10	Perform a join of the SALES and SALES_LINE tables using the ON clause on TRANSACTION_NO.
11	List the park code, park name, and attraction name for all attractions and include those theme parks with no currently listed attractions (use left outer join with ON clause).
12	Find the price of all tickets with a price less than or equal to the average ticket price. (use nested subqueries)
13	Write a query that displays the first name and last name of all employees who earn more than the average hourly rate. Do not display duplicate rows. (use nested subqueries)
14	Display all employees who work in a Theme Park that has the word 'Fairy' in its name. (use nested subqueries)
15	List all PARK_CODES where the total quantity of tickets sold is greater than the average quantity sold. . (use nested subqueries)
16	List all PARK_CODES where the total quantity of tickets sold is greater than the average quantity sold. (use nested subqueries)
17	List the difference between each tickets' price and the average ticket price. (use nested subqueries)

A.2 Prerequisite:

DML commands of SQL

A.3 Outcome:

After successful completion of this experiment students will be able to

1. Apply knowledge of relational algebra and structured query language to retrieve and manage data in relational databases.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Portal or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Portal access available)

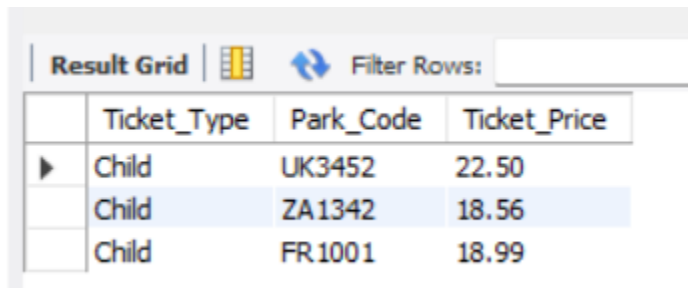
Roll No. I066	Name: Srihari Thyagarajan
Program: B Tech Artificial Intelligence	Division: I
Batch: B3	Date of Experiment: 21/09/2022
Date of Submission:	Grade:

B.1 Commands and Output:

Query 1:

```
SELECT Ticket_Type, Park_Code, Ticket_Price from TICKET WHERE (Ticket_Price > 15 and Ticket_Type = "Child")
```

```
ORDER BY Ticket_NO DESC;
```



	Ticket_Type	Park_Code	Ticket_Price
▶	Child	UK3452	22.50
	Child	ZA1342	18.56
	Child	FR1001	18.99

Query 2:

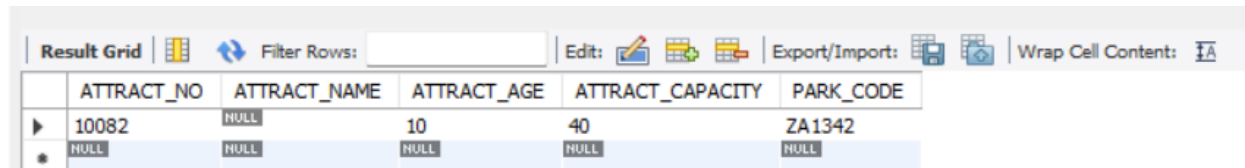
```
SELECT Park_Name from THEMEPARK WHERE Park_Name Like "%Land";
```



Park_Name
FairyLand
MiniLand
PleasureLand

Query 3:

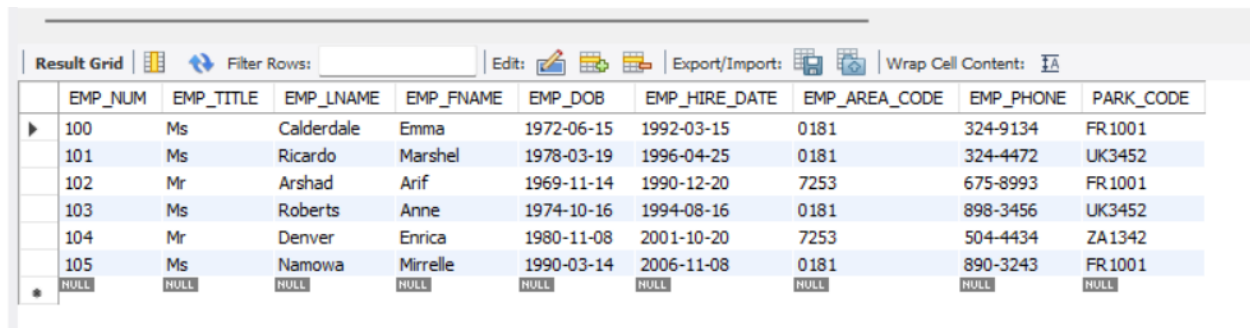
SELECT * from Attraction WHERE Attract_Name IS NULL;



	ATTRACT_NO	ATTRACT_NAME	ATTRACT_AGE	ATTRACT_CAPACITY	PARK_CODE
▶	10082	NULL	10	40	ZA1342
*	NULL	NULL	NULL	NULL	NULL

Query 4:

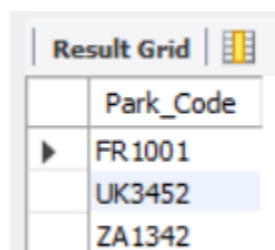
SELECT * from Employee WHERE EMP_NUM != 106;



	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	Marshel	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	Namowa	Mirrelle	1990-03-14	2006-11-08	0181	890-3243	FR1001
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 5:

SELECT DISTINCT(Park_Code) from Attraction;

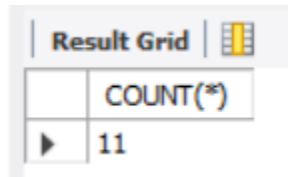


	Park_Code
▶	FR1001
	UK3452
	ZA1342

Query 6:

1. SELECT COUNT(*)FROM ATTRACTION;
2. SELECT COUNT(ATTRACT_NAME)FROM ATTRACTION;

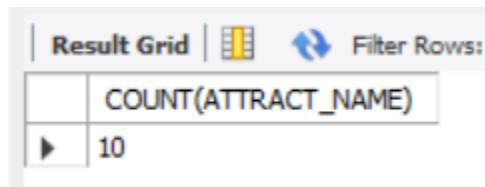
1.



A screenshot of a database interface showing a 'Result Grid'. The grid has two columns. The first column is empty. The second column contains the text 'COUNT(*)' in the header row and the value '11' in the data row. There is a small icon to the right of the header row.

	COUNT(*)
▶	11

2.



A screenshot of a database interface showing a 'Result Grid'. The grid has two columns. The first column is empty. The second column contains the text 'COUNT(ATTRACT_NAME)' in the header row and the value '10' in the data row. There is a small icon to the right of the header row. Above the grid, there is a 'Filter Rows:' button with a double-headed arrow icon.

	COUNT(ATTRACT_NAME)
▶	10

Both the outputs are different.

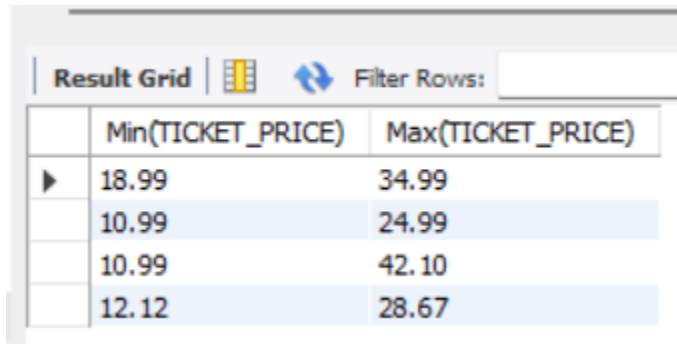
Reason -

In Count(*), the SQL Query counts the total number of rows, which includes rows which have null values as well.

In Count(Attract_Name), the SQL Query counts the total number of rows which have a value. Rows which have NULL Values aren't counted.

Query 7:

```
SELECT Min(TICKET_PRICE), Max(TICKET_PRICE) FROM TICKET  
GROUP BY(PARK_Code);
```

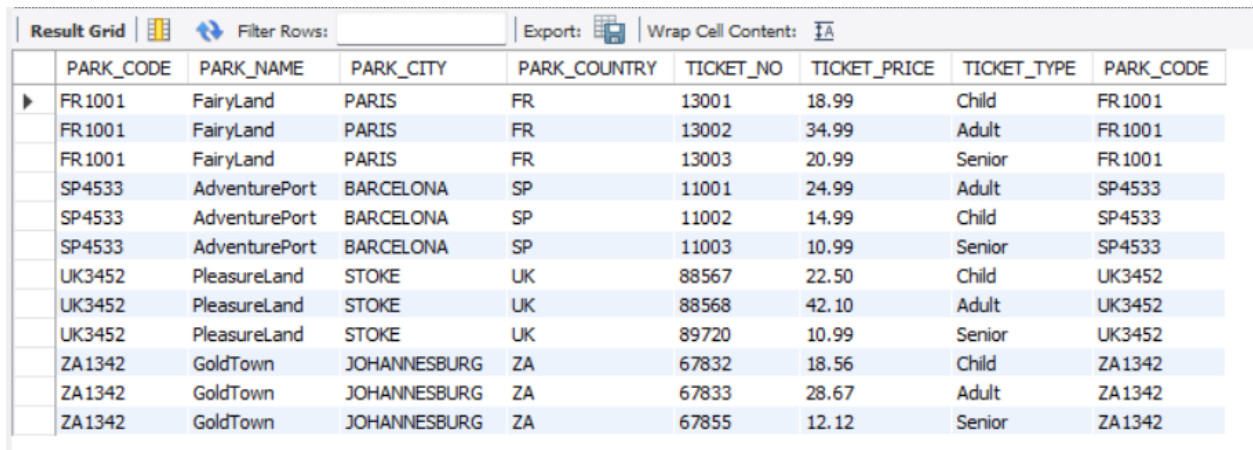


The screenshot shows a 'Result Grid' window with a toolbar containing icons for grid view, refresh, and a 'Filter Rows' input field. The grid displays the results of a query that calculates the minimum and maximum ticket prices for each park. The columns are 'Min(TICKET_PRICE)' and 'Max(TICKET_PRICE)'. There are four rows of data, each with a blue selection bar on the left.

	Min(TICKET_PRICE)	Max(TICKET_PRICE)
▶	18.99	34.99
	10.99	24.99
	10.99	42.10
	12.12	28.67

Query 8:

```
SELECT * FROM THEMEPARK as Th, TICKET as T WHERE Th.Park_code = T.Park_code;
```







The screenshot shows a 'Result Grid' window with a toolbar containing icons for grid view, refresh, 'Filter Rows', 'Export', and 'Wrap Cell Content'. The grid displays the results of a query that joins the 'THEMEPARK' table (aliased as Th) with the 'TICKET' table (aliased as T) on the 'Park_code' field. The columns are 'PARK_CODE', 'PARK_NAME', 'PARK_CITY', 'PARK_COUNTRY', 'TICKET_NO', 'TICKET_PRICE', 'TICKET_TYPE', and 'PARK_CODE'. There are 15 rows of data, each with a blue selection bar on the left.

	PARK_CODE	PARK_NAME	PARK_CITY	PARK_COUNTRY	TICKET_NO	TICKET_PRICE	TICKET_TYPE	PARK_CODE
▶	FR1001	FairyLand	PARIS	FR	13001	18.99	Child	FR1001
	FR1001	FairyLand	PARIS	FR	13002	34.99	Adult	FR1001
	FR1001	FairyLand	PARIS	FR	13003	20.99	Senior	FR1001
	SP4533	AdventurePort	BARCELONA	SP	11001	24.99	Adult	SP4533
	SP4533	AdventurePort	BARCELONA	SP	11002	14.99	Child	SP4533
	SP4533	AdventurePort	BARCELONA	SP	11003	10.99	Senior	SP4533
	UK3452	PleasureLand	STOKE	UK	88567	22.50	Child	UK3452
	UK3452	PleasureLand	STOKE	UK	88568	42.10	Adult	UK3452
	UK3452	PleasureLand	STOKE	UK	89720	10.99	Senior	UK3452
	ZA1342	GoldTown	JOHANNESBURG	ZA	67832	18.56	Child	ZA1342
	ZA1342	GoldTown	JOHANNESBURG	ZA	67833	28.67	Adult	ZA1342
	ZA1342	GoldTown	JOHANNESBURG	ZA	67855	12.12	Senior	ZA1342

Query 9:

SELECT TRANSACTION_NO, SALE_DATE, LINE_NO, LINE_QTY, LINE_PRICE FROM
Sales NATURAL JOIN SALES_LINE;

Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 					
	TRANSACTION_NO	SALE_DATE	LINE_NO	LINE_QTY	LINE_PRICE
▶	12781	2007-05-18	1	2	69.98
	12781	2007-05-18	2	1	14.99
	12782	2007-05-18	1	2	69.98
	12783	2007-05-18	1	2	41.98
	12784	2007-05-18	2	1	14.99
	12785	2007-05-18	1	1	14.99
	12785	2007-05-18	2	1	34.99
	12785	2007-05-18	3	4	139.96
	34534	2007-05-18	1	4	168.40
	34534	2007-05-18	2	1	22.50
	34534	2007-05-18	3	2	21.98
	34535	2007-05-18	1	2	84.20
	34536	2007-05-18	1	2	21.98
	34537	2007-05-18	1	2	84.20
	34537	2007-05-18	2	1	22.50
	34538	2007-05-18	1	2	21.98

	34539	2007-05-18	1	2	21.98
	34539	2007-05-18	2	2	84.20
	34540	2007-05-18	1	4	168.40
	34540	2007-05-18	2	1	22.50
	34540	2007-05-18	3	2	21.98
	34541	2007-05-18	1	2	84.20
	67589	2007-05-18	1	2	57.34
	67589	2007-05-18	2	2	37.12
	67590	2007-05-18	1	2	57.34
	67590	2007-05-18	2	2	37.12
	67591	2007-05-18	1	1	18.56
	67591	2007-05-18	2	1	12.12
	67592	2007-05-18	1	4	114.68
	67593	2007-05-18	1	2	57.34
	67593	2007-05-18	2	2	37.12

Query 10:

SELECT * FROM Sales JOIN SALES_LINE on Sales.TRANSACTION_NO =
Sales_Line.TRANSACTION_NO;

Result Grid								
Filter Rows:				Export:		Wrap Cell Content:		
	TRANSACTION_NO	PARK_CODE	SALE_DATE	TRANSACTION_NO	LINE_NO	TICKET_NO	LINE_QTY	LINE_PRICE
	12781	FR 1001	2007-05-18	12781	1	13002	2	69.98
	12781	FR 1001	2007-05-18	12781	2	13001	1	14.99
	12782	FR 1001	2007-05-18	12782	1	13002	2	69.98
	12783	FR 1001	2007-05-18	12783	1	13003	2	41.98
	12784	FR 1001	2007-05-18	12784	2	13001	1	14.99
	12785	FR 1001	2007-05-18	12785	1	13001	1	14.99
	12785	FR 1001	2007-05-18	12785	2	13002	1	34.99
	12785	FR 1001	2007-05-18	12785	3	13002	4	139.96
	34534	UK3452	2007-05-18	34534	1	88568	4	168.40
	34534	UK3452	2007-05-18	34534	2	88567	1	22.50
	34534	UK3452	2007-05-18	34534	3	89720	2	21.98
	34535	UK3452	2007-05-18	34535	1	88568	2	84.20
	34536	UK3452	2007-05-18	34536	1	89720	2	21.98
	34537	UK3452	2007-05-18	34537	1	88568	2	84.20
	34537	UK3452	2007-05-18	34537	2	88567	1	22.50
	34538	UK3452	2007-05-18	34538	1	89720	2	21.98
	34539	UK3452	2007-05-18	34539	1	89720	2	21.98
	34539	UK3452	2007-05-18	34539	2	88568	2	84.20
	34540	UK3452	2007-05-18	34540	1	88568	4	168.40
	34540	UK3452	2007-05-18	34540	2	88567	1	22.50
	34540	UK3452	2007-05-18	34540	3	89720	2	21.98
	34541	UK3452	2007-05-18	34541	1	88568	2	84.20
	67589	ZA1342	2007-05-18	67589	1	67833	2	57.34
	67589	ZA1342	2007-05-18	67589	2	67832	2	37.12
	67590	ZA1342	2007-05-18	67590	1	67833	2	57.34
	67590	ZA1342	2007-05-18	2007-05-18	2	67832	2	37.12
	67591	ZA1342	2007-05-18	67591	1	67832	1	18.56
	67591	ZA1342	2007-05-18	67591	2	67855	1	12.12
	67592	ZA1342	2007-05-18	67592	1	67833	4	114.68
	67593	ZA1342	2007-05-18	67593	1	67833	2	57.34
	67593	ZA1342	2007-05-18	67593	2	67832	2	37.12

Query 11:

SELECT t.park_code, park_name, attract_name from THEMEPARK t LEFT OUTER JOIN
Attraction a

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
park_code	park_name	attract_name	
FR1001	FairyLand	ThunderCoaster	
FR1001	FairyLand	SpinningTeacups	
FR1001	FairyLand	FlightToStars	
FR1001	FairyLand	Ant-Trap	
FR1001	FairyLand	Carnival	
NL1202	Efling	NULL	
SP4533	AdventurePort	NULL	
SW2323	Labyrinthe	NULL	
UK2622	MiniLand	NULL	
UK3452	PleasureLand	3D-Lego_Show	
UK3452	PleasureLand	BlackHole2	
UK3452	PleasureLand	Pirates	
UK3452	PleasureLand	UnderSeaWord	
ZA1342	GoldTown	NULL	
ZA1342	GoldTown	GoldRush	

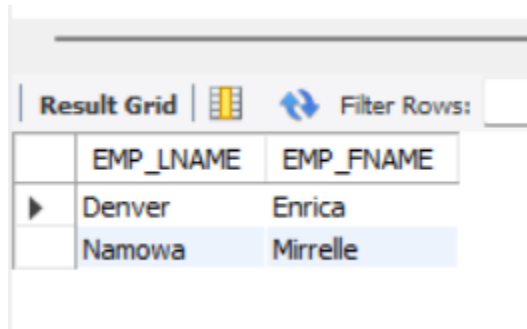
Query 12:

SELECT * FROM TICKET WHERE TICKET_PRICE <= (SELECT AVG(Ticket_Price) from
Ticket);

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
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	
89720	10.99	Senior	UK3452	
NULL	NULL	NULL	NULL	

Query 13:

```
SELECT distinct e.EMP_LNAME, e.EMP_FNAME FROM Employee e, Hours h WHERE  
e.EMP_NUM = h.EMP_NUM and h.hour_rate >= (SELECT AVG(HOUR_RATE) from Hours);
```

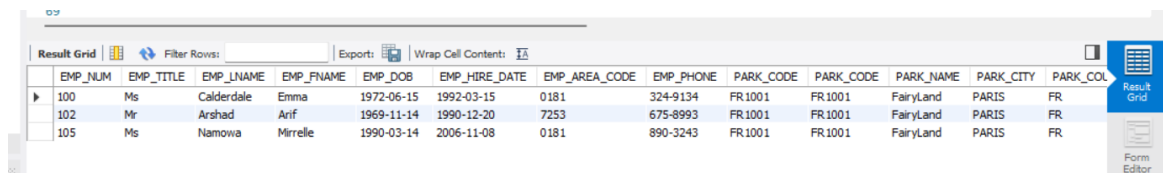


The screenshot shows a 'Result Grid' interface with a 'Filter Rows' button. The grid contains two rows of data with columns 'EMP_LNAME' and 'EMP_FNAME'.

	EMP_LNAME	EMP_FNAME
▶	Denver	Enrica
	Namowa	Mirrelle

Query 14:

```
SELECT * FROM Employee e, Themepark t WHERE e.Park_Code = t.Park_Code and  
t.park_name = (SELECT Park_name from THEMEPARK WHERE Park_name LIKE  
"%Fairy%");
```



The screenshot shows a 'Result Grid' interface with a 'Filter Rows' button and an 'Exports' button. The grid contains three rows of data with columns: EMP_NUM, EMP_TITLE, EMP_LNAME, EMP_FNAME, EMP_DOB, EMP_HIRE_DATE, EMP_AREA_CODE, EMP_PHONE, PARK_CODE, PARK_CODE, PARK_NAME, PARK_CITY, and PARK_COL.

	EMP_NUM	EMP_TITLE	EMP_LNAME	EMP_FNAME	EMP_DOB	EMP_HIRE_DATE	EMP_AREA_CODE	EMP_PHONE	PARK_CODE	PARK_CODE	PARK_NAME	PARK_CITY	PARK_COL
▶	100	Ms	Calderdale	Emma	1972-06-15	1992-03-15	0181	324-9134	FR1001	FR1001	FairyLand	PARIS	FR
	102	Mr	Arshad	Arif	1969-11-14	1990-12-20	7253	675-8993	FR1001	FR1001	FairyLand	PARIS	FR
	105	Ms	Namowa	Mirrelle	1990-03-14	2006-11-08	0181	890-3243	FR1001	FR1001	FairyLand	PARIS	FR

Query 15:

```
SELECT park_code from sales s, sales_line s1 WHERE s.TRANSACTION_NO =  
s1.TRANSACTION_NO and s1.line_qty >= (SELECT avg(line_qty) from Sales_line);
```

Result Grid	
	park_code
▶	FR1001
	FR1001
	FR1001
	FR1001
	UK3452
	UK3452
	UK3452
	UK3452
	UK3452
	UK3452
	UK3452
	UK3452
	UK3452
	UK3452
	UK3452
	ZA1342

	ZA1342
	ZA1342
	ZA1342
	ZA1342
	ZA1342
	ZA1342

Query 17:

```
SELECT TICKET_PRICE - (SELECT AVG(TICKET_PRICE) FROM TICKET) FROM  
TICKET;
```

	TICKET_PRICE - (SELECT AVG(TICKET_PRICE) FROM TICKET)
▶	3.250000
	-6.750000
	-10.750000
	-2.750000
	13.250000
	-0.750000
	-3.180000
	6.930000
	-9.620000
	0.760000
	20.360000
	-10.750000

B.2 Curiosity Questions:

1. Solve below queries for given tables:

Distributor (Dno, Dname, Daddress, Dphone)

Item (Itemno, Itemname, Colour, Weight)

Dist_Item(Dno, Itemno, Qty)

- a. Find distributor who has never supplied any item (using sub query).
- b. Count total number of items of each colour.
- c. Count number of items supplied by each distributor.

Solutions -

- a.

```
SELECT Dname
FROM Distributor
WHERE DNO NOT IN (SELECT DNO FROM Dist_Item WHERE QTY = 0)
```
- b.

```
SELECT COUNT(distinct(ItemNo))
FROM Item
GROUP BY colour;
```
- c.

```
SELECT COUNT(distinct(Itemno)) FROM Dist_Item as d1, Distributor as d
WHERE d1.Dno = d.DNo
GROUP BY colour;
```

B.3 Conclusion:

From the experiment, I learnt the following:

Implementation of relational algebra and structured query language to retrieve and manage data in relational databases.