MSCDA5540: Assignment 1

Submitted by: A00431152 (Bhagya Shree)

Tools Used:

- MySQL Server and Workbench 8.0
- Operating System: Windows 10 Pro

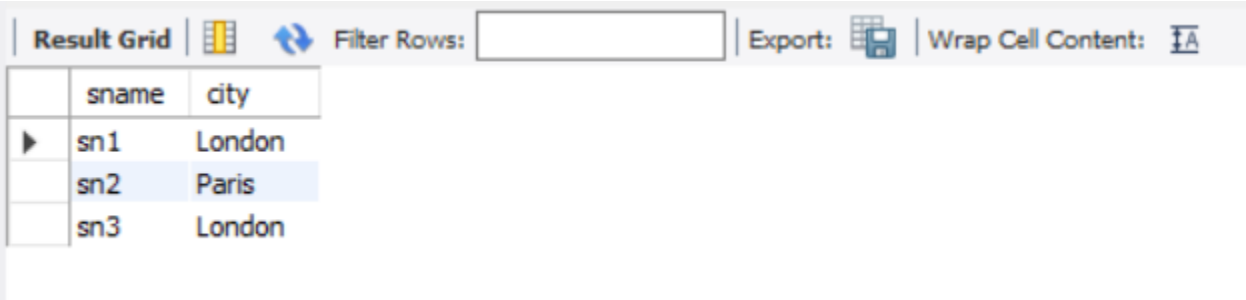
QUESTION1

Query :

(a) Get the names and locations of the suppliers who have shipped part with pno = 3.

```
SELECT
   sname, city
FROM
   S,
   SP
WHERE
   S.sno = SP.sno AND SP.pno = 3 ;
```

Output:



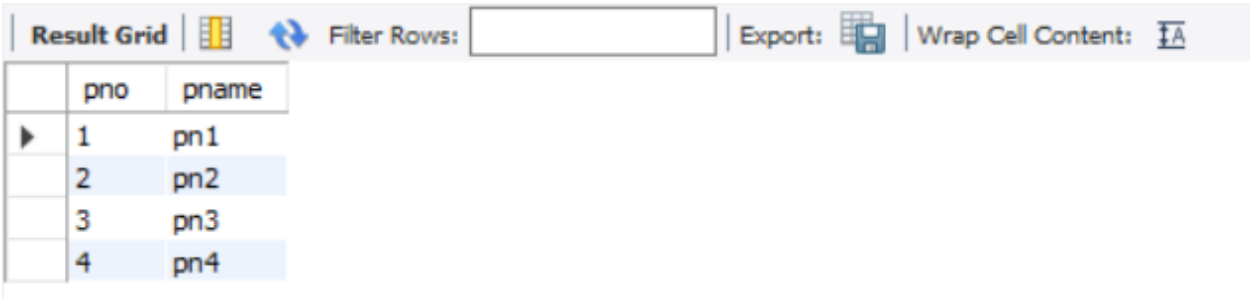
sname	city
sn1	London
sn2	Paris
sn3	London

Query

(b) Get the part numbers and names of parts that have been shipped by suppliers located in Paris with status at least 20.

```
SELECT
   P.pno, pname
FROM
   P,
   S,
   SP
WHERE
   P.pno = SP.pno AND S.sno = SP.sno
      AND S.city = 'Paris'
      AND S.status >= 20;
```

Output:



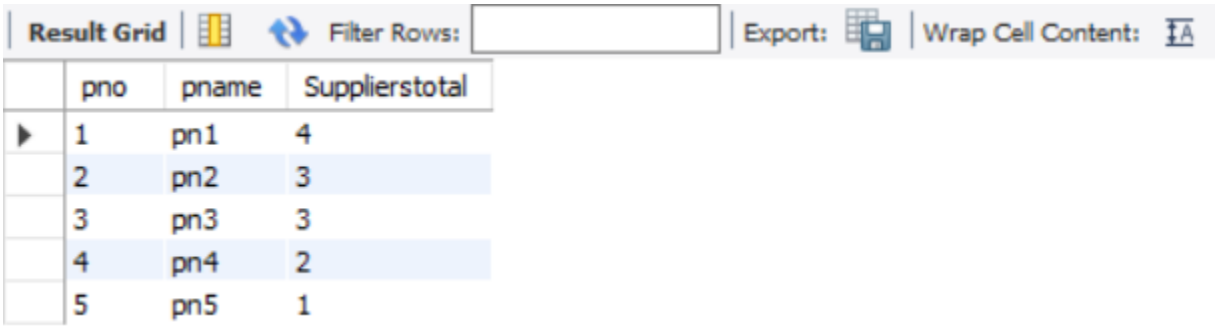
pno	pname
1	pn1
2	pn2
3	pn3
4	pn4

Query :3

(c) For each part, show the part number, name, and the number of suppliers who have supplied the part.

```
SELECT
   P.pno, pname, COUNT(*) AS Supplierstotal
FROM
   P,
   SP
WHERE
   P.pno = SP.pno
GROUP BY SP.pno;
```

Output:



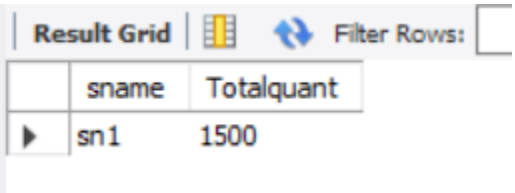
pno	pname	Supplierstotal
1	pn1	4
2	pn2	3
3	pn3	3
4	pn4	2
5	pn5	1

Query :

(d) For each London supplier who has shipped at least 1000 parts, show the name of the supplier and the total number of parts he/she has shipped.

```
SELECT
   sname, SUM(SP.qty) AS TotalQuantity
FROM
   S,
   SP
WHERE
   S.sno = SP.sno AND S.city = 'London'
GROUP BY SP.sno
HAVING TotalQuantity >= 1000;ELECT sname,sum(SP.qty) AS TotalQuantity FROM S,SP WHERE S.sno = SP.sno AND S.city = "London"
 GROUP BY SP.sno HAVING TotalQuantity >= 1000;
```

Output:



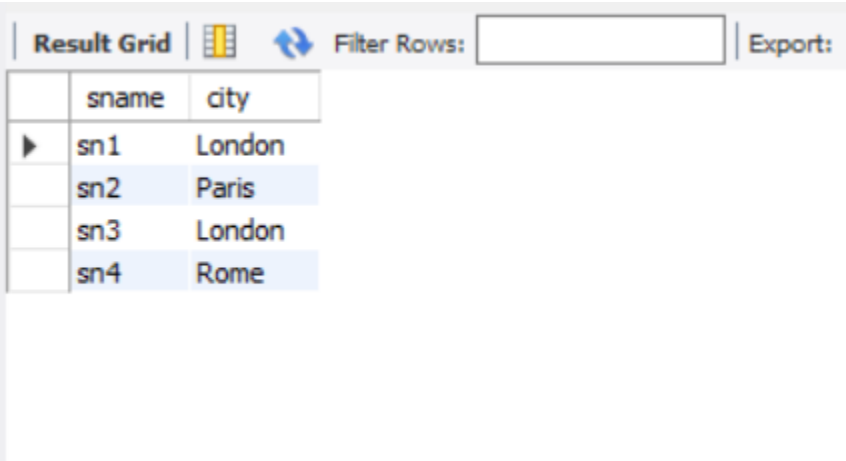
Query :

(e) Get the names and cities of the suppliers who have supplied all parts that weigh less than 4 grams.

```
SELECT DISTINCT
   s.sname, s.city
FROM
   S,
   P,
   SP
WHERE
   P.pno = SP.pno AND S.sno = SP.sno
      AND P.weight < 4;
```

Note : distinct keyword ,to get the distinct name and city

Output:



QUESTION 2

(a)Get the names of courses in the CS department

T ← σ(Department=“CS”)(Course)
Result ← π(Course_name) (T);

(b) Get the names of students who got an 'A' in CS3380.

/*Join Section and Grade report based on section id and then find the rows based on Grade =’A’) */

s1 ← (σ Grade = ‘A’ ((Section) section_identifier = section_identifier ⋈ (Grade_report))

/* extract rows from result set s1 where course number is CS3380*/
s2 ← (σ course_number = ‘CS3380’ (S1))

/* project student name after joining result set s2 and Student table*/
R← π Name ((Student) student_number = student_number ⋈ (s2))

(c) Get the instructors who have taught CS1310 and CS3380.

π Instructor (σ course_number = ‘CS1310’ AND course_number = ‘CS3380’ (SECTION))

(d) Get the instructors who have taught all courses of the CS department.

/* project course_number in result set s1 from course table for CS as dept. */

$s1 \leftarrow \pi$ course_number (σ Department = 'CS' (Course))

/* Get Instructor and course_number after joining ,section to result set s1 based on course_num */

$s2 \leftarrow \pi$ Instructor ,course_number ((section) course_number =course_number $\bowtie$ (s1))

/* Get the instructor who taught all courses by dividing resultsets2 by s1 */

$R \leftarrow \pi$ Instructor ((s2) $\div$ (s1))

QUESTION 3

Query :

(a) **Goal:** Summarizing Data in Groups
Table: customer
Query: Write a query that displays the following statistics for each country:
- Total number of customers
- Total number of male customers
- Total number of female customers
- Percent of all customers that are male (Percent Male).

Display the result by value of Percent Male so that the country with the lowest value is listed first, with the remaining countries following in ascending order.

```
SELECT
   Country,
   SUM(Gender = 'M') AS Male,
   SUM(Gender = 'F') AS Female,
   SUM(Gender = 'M') * 100 / COUNT(*) AS percentMale
FROM
   customers
GROUP BY country
ORDER BY percentMale;
```

Output:

Result Grid | Filter Rows: | Export:

Country	Male	Female	percentMale
ZA	1	3	25.0000
CA	7	8	46.6667
US	15	13	53.5714
AU	5	3	62.5000
DE	7	3	70.0000
TR	7	0	100.0000
IL	5	0	100.0000

Query :

(b) **Goal:** Summarizing Data in Groups

Table: product_dim, order_fact
Query: Create a result by combining two tables.
- Include columns Product_ID, Product_Name from product_dim table.
- Include a column with the label Total Sold. Use a summary function to create this column, which displays the quantity sold for each product.
- Specifies the tables product_dim, with the alias p and order_fact with the alias o.
- Join the tables by matching the values of the appropriate columns in each table.
- Groups the results by Product_ID from product_dim table and Product_Name.
- Orders the rows so that products with the highest number sold appear at the top of the report and then by Product_Name.

Note: DO NOT use nested queries

```
SELECT
   p.product_id, p.product_name, SUM(o.Quantity) AS 'totalsold'
FROM
   product_dim p
      JOIN
   order_fact o ON p.product_id = o.product_id
GROUP BY p.product_id , product_name
ORDER BY totalsold DESC , p.product_name;
```

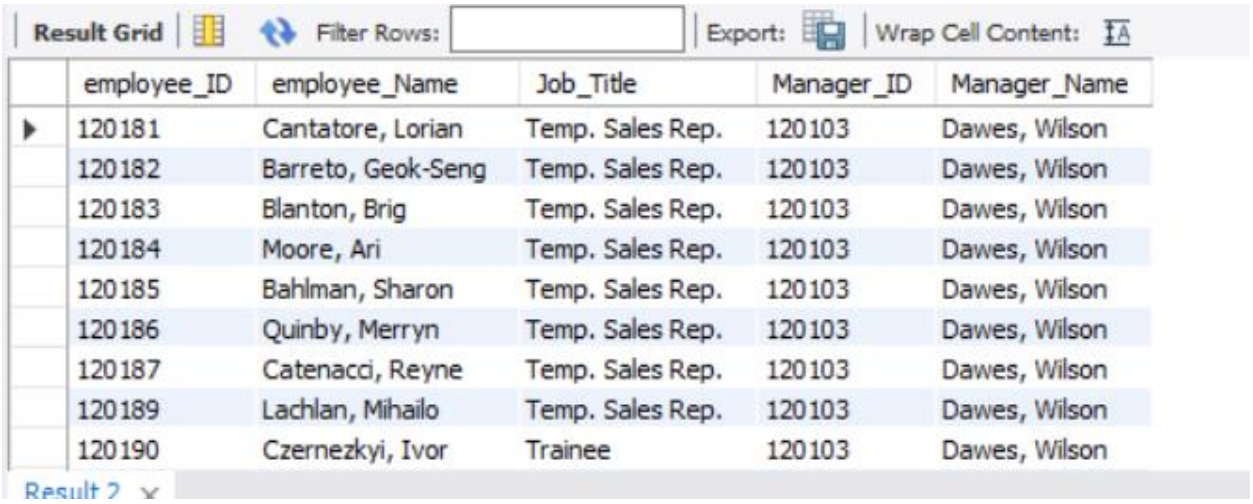
Output:

product_id	product_name	totalsold
2.30101E+11	Backpack Flag, 6,5x9 Cm.	153
2.30101E+11	Basic 10, Left , Yellow/Black	153
2.30101E+11	Collapsible Water Can	153
2.30101E+11	Comfort Shelter	153
2.30101E+11	Cup Picnic Mug 25 Cl.	153
2.30101E+11	Dome Tent Monodome Alu	153
2.30101E+11	Expedition 10,Small,Left,Blue Ribbon	153
2.30101E+11	Expedition 10,Small,Right,Blue Ribbon	153
2.30101E+11	Expedition 20,Large,Right,Forestgreen	153

Query :

(c) **Goal:** Create a result with a self-join.
Table: employee_addresses, staff
Query: Display result of all trainees and workers at company. For each trainee or temporary worker, the report should include the employee ID, name and job title, and manager ID and name. The report should be ordered by Employee_ID.

```
SELECT distinct
   e.employee_ID,
   e.employee_Name,
   s.Job_Title,
   s.Manager_ID,
   e1.employee_Name AS Manager_Name
FROM
   employee_addresses e,
   employee_addresses e1,
   staff s
WHERE
   e.employee_ID = s.employee_ID
      AND e1.employee_ID = s.Manager_ID
      AND s.Job_Title IN ('Trainee' , 'Temp. Sales Rep.')
ORDER BY e.employee_ID;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_ID	employee_Name	Job_Title	Manager_ID	Manager_Name
120181	Cantatore, Lorian	Temp. Sales Rep.	120103	Dawes, Wilson
120182	Barreto, Geok-Seng	Temp. Sales Rep.	120103	Dawes, Wilson
120183	Blanton, Brig	Temp. Sales Rep.	120103	Dawes, Wilson
120184	Moore, Ari	Temp. Sales Rep.	120103	Dawes, Wilson
120185	Bahlman, Sharon	Temp. Sales Rep.	120103	Dawes, Wilson
120186	Quinby, Merryn	Temp. Sales Rep.	120103	Dawes, Wilson
120187	Catenacci, Reyne	Temp. Sales Rep.	120103	Dawes, Wilson
120189	Lachlan, Mihailo	Temp. Sales Rep.	120103	Dawes, Wilson
120190	Czernezkyi, Ivor	Trainee	120103	Dawes, Wilson

Result 2

Query :

(d) **Goal:** LEAD and LAG functions
Table: employee_payroll
Query: 1) Calculate the difference between the salary of the current row and the previous row. 2) Calculate the difference between the salary of current row and the following row.

```
SELECT  Employee_ID, salary as "Salary",
    LAG(salary, 1,0) OVER (ORDER BY Employee_ID) AS "Salary_prev"  ,
    LEAD(salary, 1,0) OVER (ORDER BY Employee_ID) AS "Salary_Next" ,
    Salary -LAG(Salary,1,0) over (order by Employee_ID) as "Lag_Difference",
    Salary -LEAD(Salary,1,0) over (order by Employee_ID) as "LEAD_Difference"
FROM employee_payroll;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Employee_ID	Salary	Salary_prev	Salary_Next	Lag_Difference	LEAD_Difference
120101	163040	0	108255	163040	54785
120102	108255	163040	87975	-54785	20280
120103	87975	108255	46230	-20280	41745
120104	46230	87975	27110	-41745	19120
120105	27110	46230	26960	-19120	150
120106	26960	27110	30475	-150	-3515
120107	30475	26960	27660	3515	2815
120108	27660	30475	26495	-2815	1165
120109	26495	27660	28615	-1165	-2120

REFERENCES

- https://www.w3schools.com/sql
- Lecture Slides on Relational Algebra
- Lecture Slides on Joins
- Lecture Slides on Sql Query