|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **STUDENT SIGN** |
| **MOST. SAYMA KHATUN** | **22-47035-1** | **sayma** |

**Class Test 01**

**Hogwarts** is a school of witchcraft and wizardry. To ensure proper management of their data the renowned school has decided to maintain a database system. Out of many bidders your company was hired to accomplish the task. Your job is to create a relational database for Hogwarts from the requirements specified below:

RDBMS- Oracle 10g

Language-SQL

Log in as User System and create a ***user*** Dumbledore who has ***password*** Phoenix. Dumbledore is granted ***unlimited tablespace***. He is also granted the permission to ***create*** tables. After logging in with his username and password Dumbledore creates ***two tables*** i.e. Student and House. ***Student*** table has five columns containing information about students ***Identification Number, Name, CGPA, Blood Status and House Number***. ***House*** table has three columns containing information about ***House Number, House Name and House Points***. Here S\_Id, H\_Id are the ***primary key columns*** of Student and House table respectively. Student table also has a ***foreign key*** column H\_No. Constraint should be applied in such a way that CGPA cannot be greater than 4.00 and House name cannot be NULL. The two tables along with their inserted data are given below:

**Table: Student Table: House**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S\_Id | S\_Name | S\_CGPA | S\_BloodStatus | H\_No |
| 2 | Harry | 3.45 | Halfblood | 11 |
| 7 | Ron | 3.01 | Pureblood | 11 |
| 12 | Hannah |  | Pureblood | 22 |
| 17 | Cedric | 3.78 | Pureblood | 22 |
| 22 | Cho | 3.55 | Muggleborn | 33 |
| 27 | Luna | 2.89 |  | 33 |
| 32 | Draco | 3.88 | Pureblood | 44 |
| 37 | Goyle | 2.10 | Pureblood | 44 |

|  |  |  |
| --- | --- | --- |
| H\_Id | H\_Name | H\_Points |
| 11 | Gryffindor | 892 |
| 22 | Hufflepuf | 785 |
| 33 | Ravenclaw | 789 |
| 44 | Slytherin | 850 |

After creating the tables and inserting data based on provided requirements write Queries (Write down the question and also the answer. Give screenshot of the result of the query.You can add more Answer Box if required) according to the following specification:

-using **ARITHMETIC** operator

-using **CONCATENATION** operator

-using **COLUMN ALIAS**

-using **LIKE** operator

-using **IS NULL** operator

-using **ORDER BY** clause

-using **SUBSTR** function

-using **NVL** function

-using **MAX** function

-using **SUM** function

-using **GROUP BY** clause

-using **HAVING** claus

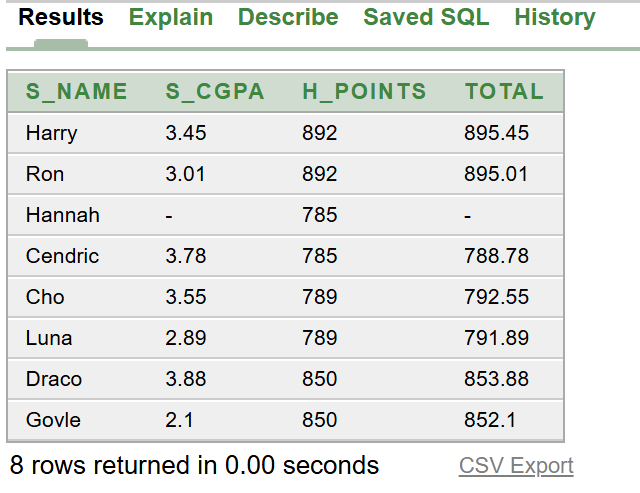
Answer:

**1.ARITHMETIC:**

Question: Calculate the total CGPA and House Points for each student. Display the student name, CGPA, House Points, and the total.

Query: SELECT S\_Name, S\_CGPA, H\_Points, (S\_CGPA + H\_Points) AS Total FROM Student

JOIN House ON Student.H\_No = House.H\_Id;

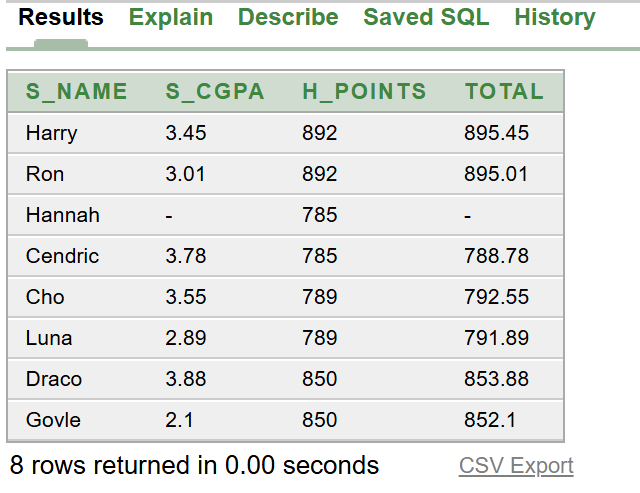


Answer:

1. **CONCATENATION:**

Question: Concatenate the student’s name and blood status in the format 'Name – BloodStatus’.

Query: SELECT S\_Name || ' - ' || S\_BloodStatus AS Student\_Info FROM Student;



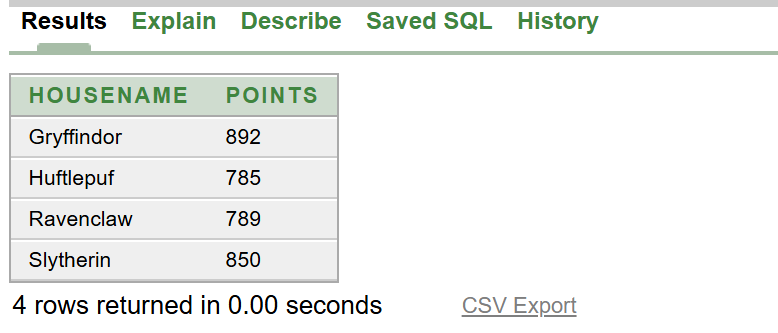
Answer:

1. **COLUMN ALIAS**

Question: Display the House Name with an alias 'HouseName' and House Points with an alias 'Points'

Query: SELECT H\_Name AS HouseName, H\_Points AS Points

FROM House;



Answer:

1. **LIKE**

Question: Retrieve all students whose names start with 'C'.

Query: SELECT S\_Name

FROM Student

WHERE S\_Name LIKE 'C%';

A screenshot of a computer

AI-generated content may be incorrect.

Answer:

1. **NULL**

Question: Retrieve all students who do not have a specified blood status.

Query: SELECT S\_Name

FROM Student

WHERE S\_BloodStatus IS NULL;

A screenshot of a computer

AI-generated content may be incorrect.

Answer:

1. **ORDER BY** clause

Question: Retrieve all students ordered by their CGPA in descending order.

Query: SELECT S\_Name, S\_CGPA

FROM Student

ORDER BY S\_CGPA DESC;

A screenshot of a data

AI-generated content may be incorrect.

Answer:

**7. SUBSTR** function

Question: Retrieve the first 3 characters of the student names.

Query: SELECT SUBSTR(S\_Name, 1, 3) AS Name\_Part

FROM Student;

A screenshot of a computer

AI-generated content may be incorrect.

Answer:

**8. NVL** function

Question: Retrieve the student names and their blood status. If the blood status is NULL, replace it with 'Unknown'.

Query: SELECT S\_Name, NVL(S\_BloodStatus, 'Unknown') AS Blood\_Status

FROM Student;

A screenshot of a computer

AI-generated content may be incorrect.

Answer:

**9. MAX** function

Question: Retrieve the highest CGPA among the students.

Query: SELECT MAX(S\_CGPA) AS Highest\_CGPA

FROM Student;

A screenshot of a computer

AI-generated content may be incorrect.

Answer:

**10. SUM** function

Question: Retrieve the total house points for each house.

Query: SELECT H.H\_Name, SUM(H\_Points) AS Total\_Points

FROM House H

GROUP BY H.H\_Name;

A screenshot of a computer

AI-generated content may be incorrect.

Answer:

**12. HAVING** clause

Question: Retrieve houses with an average CGPA greater than 3.5.

Query: SELECT H.H\_Name, AVG(S.S\_CGPA) AS Average\_CGPA

FROM Student S

JOIN House H ON S.H\_No = H.H\_Id

GROUP BY H.H\_Name

HAVING AVG(S.S\_CGPA) > 3.5;

A screenshot of a computer

AI-generated content may be incorrect.

Answer:

**11. GROUP BY** clause

Question : Retrieve the average CGPA for each house.

Query: SELECT H.H\_Name, AVG(S.S\_CGPA) AS Average\_CGPA

FROM Student S

JOIN House H ON S.H\_No = H.H\_Id

GROUP BY H.H\_Name;

A screenshot of a computer

AI-generated content may be incorrect.