# Database Exercises – SQL

|  |  |
| --- | --- |
| **Exercise** | SQL.1 |
| **Script** | **GameDBScript01** |
| **Purpose** | Try out some relatively simple SQL queries on a single table. |
| **Description** | The script creates a table **Weapon** in the **GameDB** database, popu­lated with 15 rows. |
| **Steps** | 1. If you do not already have a local database named **GameDB**, you should create such a database first. 2. Run the script on the **GameDB** database. 3. Open a **SQL Query** window. Write and execute SQL queries that return the following data from the **Weapon** table (expected results from execu­ting the queries can be found below):    1. Full details for all weapons.    2. Name, type and maximum damage for all weapons.    3. Full details for weapons above item level 20.    4. Full details for weapons with minimal damage below 25.    5. Full details for weapons with item level between 10 and 25.    6. Full details for maces.    7. Full details for maces and guns.    8. Name and maximum damage for all epic weapons.    9. Name and rarity for all weapons with a jewel socket.    10. Name and maximum damage for all rare weapons that are not two-handed weapons.    11. Name and type for weapons with a name starting with “S”.    12. Name and type for weapons with a name at most 10 characters long. |
| **Expected**  **Results** | |  |  | | --- | --- | | **a.** | 15 rows with full details | | **b.** | 15 rows with three columns in each row. | | **c.** | 6 rows with full details | | **d.** | 5 rows with full details | | **e.** | 7 rows with full details | | **f.** | 5 rows with full details | | **g.** | 9 rows with full details | | **h.** | |  |  | | --- | --- | | Redeemer | 70 | | Lucille | 350 | | | **i.** | 7 rows with two columns in each row. | | **j.** | Four rows with two columns in each row. | | **k.** | Three rows with two columns in each row. | | **l.** | Three rows with two columns in each row. | |

|  |  |
| --- | --- |
| **Exercise** | SQL.2 |
| **Script** | **GameDBScript01** |
| **Purpose** | Try out some SQL queries involving calculation, possibly using some of the built-in aggregate functions. |
| **Description** | The script creates a table **Weapon** in the **GameDB** database, popu­lated with 15 rows. |
| **Steps** | 1. If you do not already have a local database named **GameDB**, you should create such a database first. 2. Run the script on the **GameDB** database. 3. Open a **SQL Query** window. Write and execute SQL queries that return the following data from the **Weapon** table (expected results from execu­ting the queries can be found below):    1. The minimum item level for any weapon in the table. Name the returned column **MinItemLevel**.    2. The maximal number of jewel sockets for any weapon in the table. Name the returned column **MaxJewel­Sock­ets**.    3. The number of weapons in the table that have a maxi­mal damage above 100. Name the returned column **NumberOfHighDamage­Weapons**.    4. The number of weapons in the table with a rarity above Common. Name the returned column **NumberOfWeaponsThatDontSuck**.    5. The average item level for all swords.    6. The total number of jewel sockets on the weapons in the table.    7. The name, and “scaled-down” minimal and maximal damage for all weapons. The “scaled-down” damage is the original damage divided by 3. Call the columns for scaled-down damage for **NewMinDamage** and **NewMaxDamage**, respectively. |
| **Expected**  **Results** | |  |  | | --- | --- | | **a.** | 3 | | **b.** | 3 | | **c.** | 4 | | **d.** | 7 | | **e.** | 14 | | **f.** | 11 | | **g.** | All min- and max-damage values divided by 3. | |

|  |  |
| --- | --- |
| **Exercise** | SQL.3 |
| **Script** | **GameDBScript01** |
| **Purpose** | Try out some SQL queries involving ordering and grouping. |
| **Description** | The script creates a table **Weapon** in the **GameDB** database, popu­lated with 15 rows. |
| **Steps** | 1. If you do not already have a local database named **GameDB**, you should create such a database first. 2. Run the script on the **GameDB** database. 3. Open a **SQL Query** window. Write and execute SQL queries that return the following data from the **Weapon** table (expected results from execu­ting the queries can be found below):    1. Full details for all weapons, ordered by name.    2. Full details for all weapons, ordered by number of jewel sockets (ascending). In case of equal number of jewel sockets, order by item level (ascending) .    3. Name and type for all weapons, grouped by weapon type, then name.    4. Number of weapons for each weapon type. Name the column containing the count **NoOfWeapons**.    5. Average item level for weapons in each category of rarity. Name the column containing the average item level **AverageItemLevel**.    6. The number of weapons in each combination of wea­pon type and rarity (e.g. Common Guns, Epic Maces, etc.).    7. The number of weapons in each combination of wea­pon type and rarity (e.g. Common Guns, Epic Maces, etc.), but only including combinations containing at least three weapons. |
| **Expected**  **Results** | |  |  | | --- | --- | | **a.** | Full details, ordered alphabetically by name. | | **b.** | Full details, ordered numerically by jewel sockets, then item level. | | **c.** | Name and type for all weapons, ordered by type, then name. | | **d.** | |  |  | | --- | --- | | Gun | 4 | | Mace | 5 | | Sword | 6 | | | **e.** | |  |  | | --- | --- | | Common | 14 | | Epic | 26 | | Rare | 15 | | | **f.** | 8 rows with rarity, type and count | | **g.** | |  |  |  | | --- | --- | --- | | Common | Gun | 3 | | Rare | Mace | 3 | | Common | Sword | 4 | | |

|  |  |
| --- | --- |
| **Exercise** | SQL.4 |
| **Script** | **GameDBScript02** |
| **Purpose** | Try out some multi-table SQL queries. |
| **Description** | The script creates three tables in the **GameDB** database:   * **Weapon**, popu­lated with 15 rows. * **Character**, popu­lated with 6 rows. * **WeaponsOwned**, popu­lated with 16 rows. |
| **Steps** | 1. If you do not already have a local database named **GameDB**, you should create such a database first. 2. Run the script on the **GameDB** database. 3. Open a **SQL Query** window. Write and execute SQL queries that return the following data from the giventables (expected results from execu­ting the queries can be found below):    1. The name and type of the weapon that Frida uses with the right hand.    2. The name of all characters that use an Epic weapon.    3. The average item level of all weapons owned by Warriors.    4. As c), but now return the average item level for each category of rarity |
| **Expected**  **Results** | |  |  |  |  | | --- | --- | --- | --- | | **a.** | |  |  | | --- | --- | | Lucille | Mace | | | **b.** | Romanov  Frida | | **c.** | 15 | | **d.** | |  |  | | --- | --- | | Common | 13 | | Epic | 26 | | Rare | 10 | | |