SQL

1. SQL Intro and Basics

1.1. Terms

-*Field:* column, is designed to maintain specific information about every record in the table
-*Record:* row, is each individual entry that exists in a table
-*Cell:* A *value* is stored at the intersection of each row and column, sometimes called a cell
-*Primary Key:* uniquely identify each row in the table; e.g. vendor_id is the primary key of the Vendors table. A primary key can also consist of two or more columns, in which case it's called a composite primary key
-*Index:* e.g. if you frequently need to sort the Vendors rows by zip code, you can set up an index for that column. Like a key, an index can include one or more columns.
-*Foreign key:* one or more columns in a table that refer to a primary key in another table. E.g., vendor_id column is the foreign key in the Invoices table and is used to create the relationship between the Vendors table and the Invoices table. (one-to-many relationship)
-*Referential integrity:* MYSQL will not allow you to add a row in a table (will display an error). This helps to maintain the integrity of the data that's stored in the database.
-*DML:* data manipulation language
-*DDL:* data definition language
-*DBA:* database administrator

1.2. How to read a database diagram?

-An entity-relationship (ER) diagram or enhanced entity-relationship (EER) diagram can be used to show how the tables in a database are defined and related.

2. SQL Syntax

- SQL is a freeform language. That means that you can include line breaks, spaces, and indentation without affecting the way the database interprets the code.
-NOT case sensitive
-Semicolon (;) after SQL statements
-*Block Comment*: is typically coded at the beginning of a group of statements and is used to document the entire gourp. Type /* at the start of the block and */ at the end.
-*Single-line comment*: is typically used to document a single statement or line of code.

3. SQL Data types

-The data type that's assigned to a column determines the type of information that can be stored in the column
Char, varchar: a string of letters, symbols and numbers
Others are the same as Python: String, Boolean, date, numeric, array, tuple (), list [], etc.

4. SQL Statements

-DML: data manipulation language, let you work with the data
-DDL: data definition language, let you work with the objects
MySQL programmers typically work with the DML statements, while database administrators (DBAs) use the DDL statements

5. SQL Select & SQL Select Distinct

```
Select column1, column2, … from table_name;
Select * from table_name;
Select invoice_id, invoice_total, credit_total + payment_total AS total_credits from invoices;
Select distinct column1, column2, … from table_name;
Select count(distinct column1) from table_name;
Select invoice_number as "Invoice Number", invoice_date as Date, invoice_total as Total from invoices
```

6. SQL Where

Select * from table_name where Country = "Mexico"; #text field (note: we should use "=" instead of "is")
Select * from table_name where CustomerID = 1; #numeric field

Operators in The Where Clause

=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not Equal. (Note: in some versions of SQL may be written as !=)
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column
NOT	The NOT operator displays a record if the condition(s) is NOT TRUE

a complete example

Select * from table1 where Country="USA" ordered by country, population desc, limit 6
-If you include ORDER BY clause, the rows in the result set are sorted in the specified sequence. Otherwise, the rows are returned in the same sequence as they appear in the base table.
-If you include LIMIT clause, the result set that's retrieved is limited to a specified number of rows. If you omit this clause, all rows that match are returned.
-Use SELECT * only when you need to retrieve all of the columns from a table. Otherwise, list the names of the columns you need.
-order by country, population desc mean that firstly order by country alphabetically, if the country is the same, then by population from large to small

7. SQL WHERE IN/WHEERE NOT IN

SELECT * FROM table1 WHERE vendor_state NOT IN ('CA', 'NV', 'OR')

8. SQL WHERE BETWEEN AND

-to test whether an expression falls within a range of values
-the two expressions used in the Between phrase for the range of values are inclusive

8. SQL And, Or, Not

Select * from Customers where not country = 'Germany';

9. SQL Concat

Select vendor_city, vendor_state, Concat(vendor_city, vendor_state) from vendors
#how to format string data using literal values
Select vendor_name, concat (vendor_city, ', ', vendor_state, ' ', vendor_zip_code) as address from vendors
#how to include apostrophes in literal values
Select concat (vendor_name, ''s address: ') as vendor, concat (vendor_city, ', ', vendor_state, ' ', vendor_zip_code) as address from vendors

Vendor Address
US Postal Service's Address: Madison, WI 53707

10. SQL Order By

11. SQL Insert Into

insert into table_name (column1, column2, column3, …)
values (value1, value2, value3, …);

#adding values for all the columns of the table
insert into table_name
values (value1, value2, value3, …);

12. SQL Null Values
-A field with a NULL value is a field with no value; a NULL value is different from a zero value or a field that contains spaces; a field with a NULL value is one that has been left blank during record creation

select column_names
from table_name
where column_name is (not) null

13. SQL Update
update table_name
set column1 = value1, column2 = value2, …
where condition;

eg
update Customers
Set ContactName = 'Alfred Schmidt', City='Frankfurt'
where CustomerID=1;

14. SQL Delete Statement
Delete From table_name where condition;

15. SQL Select Top
- The Select Top clause is used to specify the number of records to return

#for MySQL
Select column_name(s)
From table_name
Where condition
Limit number;

#select the first three records from the "Customers" table
Select * From Customers
Limit 3;

16. SQL Min and Max

select min(column_name) / max (column_name)
from table_name
where condition;

select min(price) as smallprice
from products;
SmallestPrice
2.5

17. SQL Count(), Avg() and Sum()
18. SQL Like Operator
#starting with "a"
Select * from Customers where CustomerName like 'a%';
#ending with "a"
'%a'

#have/contain “or” in any position
‘%or%’
#have “r” in the second position
‘_r%’
#starts with “a” and are at least 3 characters in length
‘a__%’
#starts with “a” and ends with “o”
‘a%o’

19. SQL Wildcard Characters
-A wailcard character is used to substitute one or more characters in a string
-use together with Like Operator

#select all customers with a city starting with “b”, “s”, or “p”
select * from Customers where city like ‘[bsp]%’;
#starting with “a”, “b”, or “c”
[a-c]%
#not starting with “b”, “s”, or “p”
[!bsp]%

20. SQL In Operator
select * from Customers where Country (not) In (‘Germany’, ‘France’, ‘UK’)

#select all customers that are from the same countries as the suppliers
select * from customers where country in (select country from suppliers);

21. SQL Between Operator
select column_names(s) from table_name where column_name between value1 and value2;
select * from products where price (not) between 10 and 20;

22. SQL Aliases
-SQL aliases are used to give a table, or a column in a table, a temporary name (change name as you like☺)
-Aliases are often used to make column names more readable
-an alias only exists for the duration of the query

Select CustomersID as ID, CustomerName as Customer from Customers;
Select CustomerName, Address + ‘, ‘ + PostalCode + ‘ ‘ + City + ‘, ‘ + Country As Address from Customers;

23. SQL Joins

select ____(variables) from ____(dataset 1) inner/left/etc. join (join type) ____(dataset 2) on ____ = ___(the connection)

#three table join
Select Orders.Order ID(those are all nicknames), Customers.CustomerName, Shippers.ShipperName from ((Orders Inner Join Customers On Orders.CustomerID = Customers.CustomerID) Inner Join Shippers On Orders.ShipperID = Shippers.ShipperID);

inner join	returns records that have matching values in both tables
left(outer) join	returns all records from the left table, and the matched records from the right table
right(outer) join	returns all records from the right table, and the matched records from the left table
full(outer) join	returns all records when there is a match in either left or right table

Self join: a self join is a regular join, but the table is joined with itself.

Select A.CustomerName As CustomerName1, B.CustomerName AS CustomerName2, A.City From Customers A, Customers B
Where A.CustomerID <> B.CustomerID
And A.City=B.City
Order By A.City;

24. SQL Union Operator

The UNION operator is used to combine the result-set of two or more Select statements.
-each select statement with union must have the same number of columns
-the columns must also have similar data types
-the columns in each select statement must also be in the same order

#union means combine without duplicate; union all will select duplicate values
Select column_name(s) from table1 union(all) select column_name(s) from table2

25. SQL Group By

group by vs order by

order by alters the order in which items are returned
group by will aggregate records by the specified columns which allow you to perform aggregation function on non-grouped columns

example:
table

ID	Name
1	Peter
2	John
3	Greg
4	Peter

select * from table order by name

ID	Name
3	Greg
2	John
1	Peter
4	Peter

select count(ID), name from table group by name

ID	Name
1	Greg
1	John
2	Peter

select name from table group by name having count(ID)>1
Peter

26. SQL Having
-The having clause was added to SQL because the WHERE keyword could not be used with aggregate functions

27. SQL Exists
-The exists operator is used to test for the existence of any record in a subquery
-The exists operator returns true if the subquery returns one or more records

```
select suppliername from suppliers where exists (select productname from produtcs where products.supplierID = supplier.supplierID and price<20);
```

28. SQL Any and All Operators

#list the ProductName if it finds ANY records in the OrderDetails table has Quantity equal to 10 (this will return TRUE because the Quantity column has some values of 10)

```
select ProductName from Products where ProductID = ANY (select ProductID from OrderDetails where Quantity = 10);
```

#list the ProductName if ALL the records in the OrderDetails table has Quantity equal to 10, note: this will return False because the Quantity column has many different values (not only the value of 10).

```
select ProductName from Products Where ProductID = ALL (select ProductID from OrderDetails where Quantity = 10);
```

29. SQL Select Into

-copy all columns into a new table

```
select * into newtable [in externaldb] from oldtable where condition;
```

30. SQL Insert Into Select

-copies data from one table and inserts it into another table

-requires that the data types in source and target tables matches

```
insert into table2 (column1, column2, column3, …)
select column1, column2, column3, …
from table1
where condition;
```

31. SQL CASE Statement

```
select OrderID, Quantity,
CASE
    WHEN Quantity > 30 Then 'The quantity is greater than 30'
    WHEN Quantity = 30 Then 'The quantity is 30"
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;
```

32. SQL NULL Functions

```
select ProductName, UnitPrice * (UnitesInStock + UnitsOnOrder) from Products;
```

in the example above, if any of the "UnitsOnOrder" values are NULL, the result will be NULL

solutions

#if

```
select ProductName, UnitPrice * (UnitsInStock + IFNULL (UnitsOnOrder, 0)) from Products;
```

#IFNULL() Function

```
select ProductName, UnitPrice * (UnitsInStock + COALESCE (UnitsOnOrder, 0)) from Products;
```

From Hackerrank Practice

https://www.hackerrank.com/challenges/weather-observation-station-3/problem

- **Select Statement**

1. Select * from city where countrycode = 'USA' and population > 100000;

From city: from xxx table
Counrtycode, popluation belong to category
USA is a real content

Select *(means all attributes) /xxx from *table* where *xxx* = 'xxx' and xxx > 10000;

2. Query a list of CITY and STATE from the STATION table.
Select CITY, STATE from STATION order by CITY, STATE;

3. Query a list of CITY names from STATION for cities that have an **even** ID number. Print the results in any order, but **exclude duplicates** from the answer.

Even: MOD(xx, 2) =0
Not even, maybe? MOD(xx, 2) ! =0

Exclude duplicates: select **distinct** xxx from xxx
Select distinct CITY from STATION where MOD (ID, 2) = 0 order by CITY;

•• Count Statement
4. Find the difference between the total number of CITY entries in the table and the number of distinct City entries in the table.

Select Count(City) – Count(Distinct City) from STATION;

5. Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths. If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

select city, length (city) from (SELECT City FROM Station ORDER BY LENGTH(City), city) where rownum=1;
select city, length (city) from (SELECT City FROM Station ORDER BY LENGTH(City) desc, city) where rownum=1;

6.1) Query the list of CITY names starting with vowels (i.e. a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

Select distinct City from Station where City like 'a%' or City like 'e%' or City like 'i%' or City like 'o%' or city like 'u%';

Note: A% means starting
%A means ending

6.2) Query the list of CITY names ending with vowels (i.e. a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

Select distinct city from Station where city like '%a' or city like '%e' or city like '%i' or city like '%o' or city like '%u';

6.3) Query …. which have vowels as both their first and last characters.
Select city from station where city regexp '^[aeiou]' and city regexp '[aeiou]$';

6.4) Query … which do not start with vowels. No duplicates.
SELECT DISTINCT city FROM station WHERE (LOWER(city) NOT LIKE ('a%') and LOWER(city) NOT LIKE ('e%') and LOWER(city) NOT LIKE ('i%') and LOWER(city) NOT LIKE ('o%') and LOWER(city) NOT LIKE ('u%')) order by city;

6.5) Query … which do not end with vowels. No duplicates.
Select distinct city from station where city not like ‘%a” and city not like ‘%e’ and city not like ‘%i’ and city not like ‘%o’ and city not like ‘%u’ order by city;

6.6) Query … which either do not start with vowels or do not end with vowels. No duplicates.
select distinct (CITY) from STATION where not regexp_like(lower(CITY),'^[aeiou].*[aeiou]$') order by CITY;

7. Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID.

Select Name from STUDENTS where Marks > 75 order by Right (Name, 3), ID;

8. Query the average population for all cities in CITY, rounded down to the nearest integer.
Select round(avg(population)) from CITY;