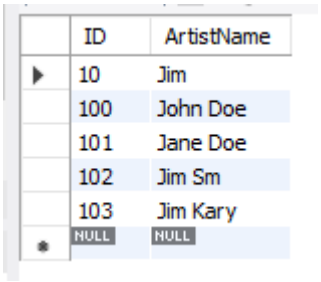
Week 14 Quiz Explanation

Q1. What will be the result after executing the below query:

```
CREATE TABLE artists (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    ArtistName VARCHAR(255) NOT NULL
);

INSERT INTO artists (ID, ArtistName) VALUES (100, 'John Doe');
INSERT INTO artists (ArtistName) VALUES ('Jane Doe');
INSERT INTO artists (ID, ArtistName) VALUES (NULL, 'Jim Sm');
INSERT INTO artists (ID, ArtistName) VALUES (10, 'Jim ');
INSERT INTO artists (ID, ArtistName) VALUES (NULL, 'Jim Kary');

SELECT max(ID) from artists;
```

Option 1- 11
Option 2- 3
Option 3- 103
Option 4- 104

Artists TABLE:



ID	ArtistName
10	Jim
100	John Doe
101	Jane Doe
102	Jim Sm
103	Jim Kary
NULL	NULL

Here's a breakdown of the inserted values:

1. `INSERT INTO artists (ID, ArtistName) VALUES (100, 'John Doe');`
   In this case, you explicitly set the "ID" to 100. The auto-increment attribute will continue from there.

2. `INSERT INTO artists (ArtistName) VALUES ('Jane Doe');`
   You didn't specify an "ID," so the auto-increment mechanism will assign the next available unique value. It will be 101

3. `INSERT INTO artists (ID, ArtistName) VALUES (NULL, 'Jim Sm');`
   When you insert a `NULL` value in an auto-increment primary key column, it will take the next available auto-increment value. 102

4. `INSERT INTO artists (ID, ArtistName) VALUES (10, 'Jim ');`
   Here, you explicitly set the "ID" to 10.

5. `INSERT INTO artists (ID, ArtistName) VALUES (NULL, 'Jim Kary');`
   Again, inserting a `NULL` value might result in the next available auto-increment value. Here is the catch, you might think that next value would be 11, but this is not the case.

As ID column is Primary key and on auto-increment mode, auto-increment will try to give the next available value and this is done by reordering the rows in sorted order, and then whichever row comes last autoincrement is done on that row value.

In this case, ID = 102 would come last, so the next value for ID would be 103

Now, let's discuss the result of `SELECT max(ID) from artists;`. The query is asking for the maximum "ID" value in the "artists" table.

Hence, in our case, the correct answer is "Option 3: 103" because 103 is the maximum "ID" value in the table.

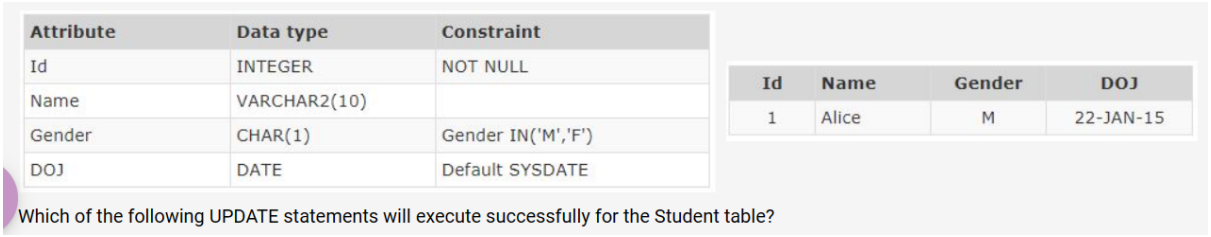
Q2. No of records in the albums table after the below query is executed (given that query of Q1 is already executed):

```
CREATE TABLE albums (
   ID INT PRIMARY KEY AUTO_INCREMENT,
   Title VARCHAR(255) NOT NULL,
   ArtistID INT NOT NULL,
   ReleaseDate DATE NOT NULL,
   Label VARCHAR(255) NOT NULL,
   Price DECIMAL(10, 2) NOT NULL CHECK (Price > 0),
);

INSERT INTO albums (Title, ArtistID, ReleaseDate, Label, Price)
VALUES ('Greatest Hits', 10, '2023-01-01', 'Universal Music', 10.00);

INSERT INTO albums (ID, Title, ArtistID, ReleaseDate, Label, Price)
VALUES (100, 'Greatest Hits', 1, '2023-01-02', 'CampusX Music', 12.00);

INSERT INTO albums (ID, Title, ArtistID, ReleaseDate, Label, Price)
VALUES (NULL, 'Greatest Hits', 101, '2023-01-01', 'Universal Music', 10.00);

INSERT INTO albums (Title, ArtistID, ReleaseDate, Label, Price)
VALUES ('Greatest Hits', 10, '2023-01-01', 'Universal Music', -10.00);
```

Option 1- 1
Option 2- 3
Option 3- 4
Option 4- 2

Correct- 1
Explanation: Second insert query will raise an error as **ArtistID = 1** is not present in the **artists** table. So after that, no query will be executed. Only one insert query to the albums table is executed.

Q3. Consider the following table structure and content for the 'Student' table:



Attribute	Data type	Constraint
Id	INTEGER	NOT NULL
Name	VARCHAR2(10)	
Gender	CHAR(1)	Gender IN('M','F')
DOJ	DATE	Default SYSDATE

Id	Name	Gender	DOJ
1	Alice	M	22-JAN-15

Which of the following UPDATE statements will execute successfully for the Student table?

Which of the following UPDATE statements will execute successfully for the Student table?

Option 1: UPDATE Student SET Gender = 'Male' WHERE Id = 1
Option 2: UPDATE Student SET Gender = 'X' WHERE Id = 1
Option 3: UPDATE Student SET Id = NULL WHERE Name = 'Alice'
Option 4: UPDATE Student SET Gender = 'F', DOJ = NULL WHERE Name = 'Alice'

Q4. Consider the 'Student' table as:

Attribute	Data type	Constraint
Id	INTEGER	NOT NULL
Name	VARCHAR2(10)	
Gender	CHAR(1)	
DOJ	DATE	DEFAULT SYSDATE

Id	Name	Gender	DOJ
1	Alice	F	22-JAN-15
2	Alan	M	20-JAN-15
3	Harry	M	01-FEB-15
4	Tina	F	20-JAN-15

Which of the following UPDATE statements will Update 2 rows for the Student table?

Option 1: UPDATE Student SET DOJ = SYSDATE WHERE Gender = 'M'
Option 2: UPDATE Student SET DOJ = SYSDATE WHERE Id = 1
Option 3: UPDATE Student SET Id = NULL WHERE Gender = 'M'
Option 4: UPDATE Student SET Name = 'Heena' WHERE Gender = 'F' AND Id = 4

Explanation :
Option 2 : Will only update one row.
Option 3 : Id column can't have value NULL, because of NOT NULL constraint.
Option 4 : Will only update one row. Last row

Q5: Which of the following columns in a table cannot be updated?
Option 1: DATE type columns in the table

Option 2: Columns that allow NULL values in the table
Option 3: A primary key column that also serves as a foreign key reference in another table
Option 4: All of the above

Explanation :
Option 3: A primary key column that also serves as a foreign key reference in another table

Primary key columns serve as unique identifiers for records in a table, and they are typically not updated because changing the primary key would affect the integrity of the data and potentially break relationships with other tables where this key is used as a foreign key. Therefore, primary key columns are not typically updated.

Options 1 and 2 are not necessarily true in all cases. You can update columns of DATE type, and you can also update columns that allow NULL values if the table permits it.

Q6: Customer table consists of a column Balance. The table contains a single row with a value of **20000.56**. What will be the values of ROUND, ROUND1, and FLOOR when the below query is executed?

SELECT ROUND(Balance) "ROUND", ROUND(Balance,1) "ROUND1", FLOOR(Balance) "FLOOR" FROM Customer;

Option 1: 20001 20000.56 20001
Option 2: 20000 20000.6 20000
Option 3: 20001 20000.56 20000
Option 4: 20001 20000.6 20000

Option 4: 20001 20000.6 20001

Here's the explanation:
- `ROUND(Balance)` rounds the value 20000.56 to the nearest whole number, which is 20001.
- `ROUND(Balance, 1)` rounds the value to one decimal place, so it becomes 20000.6.
- `FLOOR(Balance)` returns the largest integer less than or equal to 20000.56, which is 20000.

Q7: Consider the following record of the customer table:

LASTNAME
Desouza

What will be the output of the following query?
SELECT LENGTH(SUBSTR(LASTNAME, 3,7)) FROM CUSTOMER;

Option 1: 7
Option 2: 5
Option 3: 8
Option 4: 3

This SQL query will calculate the length of the substring of the string "DESOUZA." It takes a substring of "DESOUZA" starting from the 3rd character (index 3) and takes the next 7 characters. “SOUZA” - Length 5

Q8: Consider the following record of the customer table:

ACCDT
8-JAN-13

Which of the following query displays the data in the following format:

DAY	MON
08	01

Option 1: SELECT TO_CHAR(Accdt,'dy')"DAY", TO_CHAR(Accdt,'mon')"MON" FROM Customer
Option 2: SELECT TO_DATE(Accdt,'dy')"DAY", TO_CHAR(Accdt,'mon')"MON" FROM Customer
Option 3: SELECT TO_CHAR(Accdt,'dd')"DAY", TO_CHAR(Accdt,'mm')"MON" FROM Customer
Option 4: SELECT TO_DATE(Accdt,'dd')"DAY", TO_DATE(Accdt,'mm')"MON" FROM Customer

Explanation: In Option 3 query, 'dd' is used to extract the day of the month as a two-digit number (e.g., '08' for January 8th), and 'mm' is used to extract the month as a two-digit number (e.g., '01' for January).

Q9: Which of the following aggregate functions can be applied on both number and character-typed columns? [Choose any **THREE**]

Option 1: MIN
Option 2: MAX
Option 3: AVG
Option 4: COUNT
Option 5: SUM

Q10: Table T exists with single column A having 4 rows with values – 100, 200, NULL, 300. What is the output of the following query?

SELECT MAX(A) FROM t WHERE a>1000;

Option 1: 300
Option 2: NULL
Option 3: No rows
Option 4: Error is thrown

Explanation: No rows have a value greater than 1000, so NULL will be the output.

Q11: Which of the following date function can be used to get the timestamp from a database?

Option 1: SYSDATE
Option 2: SYSTEMDATE
Option 3: SYSTIMESTAMP
Option 4: SYSTEMTIMESTAMP

Explanation: SYSTIMESTAMP

The SYSTIMESTAMP function is used to retrieve the system timestamp from a database. This function returns the current date and time including fractional seconds and time zone information. It's commonly used to obtain the current timestamp for various database operations

Q12: Which of the following statement(s) is/are FALSE about order by clause? [Choose any **TWO**]

Option 1: Order by can be used with DATE attributes
Option 2: Order by default sort is ascending order

Option 3: Where clause can be used after the order by clause
Option 4: Order by only works on the number column

Q13: Consider the following “Quotation” table. How many rows would be displayed when the following query is executed?

SELECT ItemCode, COUNT(*) FROM Quotation GROUP BY SupplierId, ItemCode;

QuotationId	SupplierId	ItemCode	QuotedPrice
Q1001	S1001	I1012	1500
Q1002	S1002	I1012	1400
Q1003	S1003	I1013	1450
Q1004	S1001	I1012	600
Q1005	S1004	I1013	625

Option 1: 2
Option 2: 4
Option 3: Error is thrown
Option 4: 5

Explanation :
4 Groups will be formed : (S1001, I1012), (S1002, I1012), (S1003, I1013) and (S1004, I1013)

Q14: Given the employee table as input, choose the query that displays department-wise average salaries sorted in descending order.

ID	ENAME	DEPT	SALARY
1	James Potter	FSI	75000
2	Ethan McCarty	ETA	90000
3	Emily Rayner	ETA	25000
4	Jack Abraham	ETA	30000

Option 1: SELECT AVG(Salary) AvgSal, Dept FROM Employee GROUP BY Dept ORDER BY 1

Option 2: SELECT AVG(Salary) AvgSal, Dept FROM Employee ORDER BY AVG(Salary) GROUP BY Dept
Option 3: SELECT AVG(Salary) AvgSal, Dept FROM Employee GROUP BY Dept ORDER BY AvgSal DESC
Option 4: SELECT Dept, AVG(Salary) AvgSal FROM Employee GROUP BY 1 ORDER BY 2 DESC

Explanations for each of the options:

1. Option 1: This query calculates the average salary (`AVG(Salary)`) for each department (`Dept`) and then attempts to order the results by the first column (`ORDER BY 1`). However, it does not specify a descending order (`DESC`), so the default sorting order will be ascending.

2. Option 2: In this query, it attempts to order the result set by the calculated average salary (`AVG(Salary)`) before the grouping (`GROUP BY Dept`). This is not valid because the aggregation function is used in the `ORDER BY` clause before the grouping operation. This query would not produce error.

3. Option 3: This is the correct query. It first calculates the average salary for each department and then orders the results in descending order by the alias `AvgSal` using `ORDER BY AvgSal DESC`.

In summary, Option 3 is the correct choice for calculating department-wise average salaries and ordering the results in descending order by the `AvgSal` alias.

Q15: Which of the following statement(s) is/are FALSE about UNION? [Choose any **TWO**]

Option 1: Column Names in all queries in a UNION must match position wise
Option 2: UNION can be used with an UPDATE statement
Option 3: Data Types in all queries in a UNION must match position wise
Option 4: The number of columns must match the query