Deloitte.

Test of Basic Knowledge of SQL

Using this test you can assess whether you have basic working knowledge of SQL that is prerequisite for attending Deloitte's Data Science Academy – Intermediate program. To verify accuracy of your answers, use the key with correct answers at the end of this document. In case that your success rate will be lower than 75% (i.e., you will answer correctly less than 30 questions out of 40), you should consider consulting available on-line resources that will enable you to become more deeply familiarized with SQL. For that purpose, we have provided you for each question with a link that will bring you to the website that will help you to grasp and practice given SQL concept.

There is no strict time limit for finishing the test, but we suppose that you will be able to finish it in ~20 minutes. Some of the questions within the test assume that following *Customers, Orders, Products* and *OrderDetails* tables from the *Test* database are available to you.

Customers table:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Mary A	Mary Austin	338 City Road	London	EC1V 2NX	UK
2	KJ	Karl Jung	Graefestr. 2	Berlin	10967	Germany
3	Jamie_Wats	James Watson	3116 Massachusetts Avenue	Washington	20004	USA
…	…	…	…	…	…	…

Orders table:

OrderID	CustomerID	OrderDate
10308	5	2017-01-21
10309	80	2017-01-22
10310	97	2017-01-22
…	…	…

Products table:

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Lip Gloss	1	1	2 bottles	20
2	Eyeshadow makeup	1	2	1 box	10
3	Lip Balm	2	1	1 bottle	5
…	…	…	…	…	…

OrderDetails table:

OrderDetailID	OrderID	ProductID	Quantity
1	12508	5	1
2	12508	6	10
3	12509	25	50
…	…	…	…

Q1) What SQL statement would you use to create SQL database named *Test*?

a) CREATE DBT Test;
b) CREATE Test;
c) CREATE DATABASE Test;
d) MAKE DATABASE Test;

Q2) What SQL statement would you use to remove SQL database named *Test*?

a) DROP DATABASE Test;
b) REMOVE Test;
c) DROP DBT Test;
d) REMOVE DATABASE Test;

Q3) Assign individual relationships that one can find in ERD (Entity Relationship Diagram) to their appropriate descriptions.

a)
b)
c)
d)
e)
f)
g)

1) zero or many (optional)
2) one or more (mandatory)
3) one to one
4) one and only one (mandatory)
5) one to many (mandatory)
6) many
7) zero or one (optional)

Q4) Which kind of relationship best describes relationship between *Products* (A) and *OrderDetails* (B) tables?

a) 
b) 
c) 
d) 

Q5) Fulfill missing parts in SQL statement below to create *Products* table.

```
_____________ _________ (
  ProductID INT _____________,
  ProductName VARCHAR(255),
  SupplierID INT,
  CategoryID INT,
  Unit _____________,
  Price _____________
);
```

Q6) Primary Key is a combination of two types of constraints. Find the correct one in the list below.

a) UNIQUE & DEFAULT
b) CHECK & UNIQUE
c) UNIQUE & NOT NULL
d) CHECK & AUTO INCREMENT

Q7) Which field in the *Orders* table has a role of FOREIGN KEY in relation to *Customers* table?

a) CustomerID
b) OrderID
c) OrderDate
d) CustomerName

Q8) What SQL statement would you use to put a new record into the *Orders* table?

a) INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES (200, 125, '2018-02-05');
b) PUT INTO Orders (OrderID, CustomerID, OrderDate) VALUES (200, 125, '2018-02-05');
c) INSERT VALUES (200, 125, '2018-02-05') INTO Orders (OrderID, CustomerID, OrderDate);
d) PUT VALUES (200, 125, '2018-02-05') INTO Orders (OrderID, CustomerID, OrderDate);

Q9) What SQL statement would you use to modify the existing record of *CustomerID* to 10 for *OrderID* #10308 in the *Orders* table?

a) CHANGE SET Orders CustomerID = 10 WHERE OrderID = 10308;
b) UPDATE SET Orders CustomerID = 10 WHERE OrderID = 10308;
c) CHANGE Orders SET CustomerID = 10 WHERE OrderID = 10308;
d) UPDATE Orders SET CustomerID = 10 WHERE OrderID = 10308;

Q10) What SQL statement would you use to delete order with *OrderID* #10308 from the *Orders* table?

a) DELETE FROM Orders WHERE OrderID = 10308;
b) CUT FROM Orders WHERE OrderID = 10308;
c) MOVE Orders WHERE OrderID = 10308;
d) REMOVE FROM Orders WHERE OrderID = 10308;

Q11) What SQL statement would you use to add an *Age* field to the *Customers* table?

a) UPDATE TABLE Customers ADD Age INT;
b) ALTER TABLE Customers ADD Age INT;
c) UPDATE TABLE ADD Age INT Customers;
d) ALTER TABLE ADD Age INT Customers;

Q12) What SQL statement would you use to remove *Customers* table from *Test* database?

a) REMOVE TABLE Customers;
b) REMOVE Customers;
c) DROP Customers;
d) DROP TABLE Customers;

Q13) Fulfill missing parts in SQL statement below to extract all records from *Customers* table.

_____ _____
_____ Customers;

Q14) Fulfill missing parts in SQL statement below to extract *CustomerName* and *Address* from *Customers* table.

_____ _____
_____ Customers;

Q15) Fulfill missing parts in SQL statement below to extract all distinct countries from *Customers* table.

_____ _____ Country
_____ Customers;

Q16 Fulfill missing parts in SQL statement below to extract all records from *Products* table that will include only products with price higher than 20 EUR.

_____ _____
_____ Products
_____ Price _____;

Q17) Fulfill missing parts in SQL statement below to extract all records from *Customers* table that will include only those customers who have NULL values in *Address* field.

_____ _____
_____ Customers
_____ _____;

Q18) Fulfill missing parts in SQL statement below to extract all records from *Customers* table that will include only those customers who are from Germany or UK.

_____ _____
_____ Customers
_____ Country _____;

Q19) Fulfill missing parts in SQL statement below to extract all records from *Customers* table that will include only those customers who are not from USA.

_____ _____
_____ Customers
_____ _____ Country _____;

Q20) Fulfill missing parts in SQL statement below to extract all records from *Products* table that will include only those products that are supplied by supplier with *SupplierID* #1 and that belong to *CategoryID* #2.

_____ _____
_____ Products
_____ _____ _____ _____;

Q21) Fulfill missing parts in SQL statement below to arrange records in *Products* table according to *Price* in descending order.

_____ _____
_____ Products
_____ _____;

Q22) Fulfill missing parts in SQL statement below to extract the first 50 records from *Customers* table.

_____ _____ _____
_____ Customers;

Q23) Fulfill missing parts in SQL statement below to find maximum *Price* for products listed in *Products* table.

_____ _____
_____ Products;

Q24) What statement will you use to count number of records within *Customers* table?

a) SELECT ALL FROM Customers;
b) SELECT N FROM Customers;
c) SELECT COUNT(*) FROM Customers;
d) SELECT NROW FROM Customers

Q25) Fulfill missing parts in SQL statement below to find average *Price* for products listed in *Products* table.

_____ _____
_____ Products;

Q26) Fulfill missing parts in SQL statement below to find overal number of ordered products using *Quantity* field in *OrderDetails* table.

_____ _____
_____ OrderDetails;

Q27) Fulfill missing parts in SQL statement below to find all customers listed in the *Customers* table whose name starts with letter „b".

_____ _____
_____ Customers
_____ CustomerName _____;

Q28) Fulfill missing parts in SQL statement below to find all customers listed in the *Customers* table whose name starts with letter „b" and ends with letter „o".

_____ _____
_____ Customers
_____ CustomerName _____;

Q29) Fulfill missing parts in SQL statement below to find all customers listed in the *Customers* table whose name has letter „b" in the second position.

_____ _____
_____ Customers
_____ CustomerName _____;

Q30) Fulfill missing parts in SQL statement below to find all customers listed in the *Customers* table who live in Germany, UK, and USA.

_____ _____
_____ Customers
_____ _____ ___ ("Germany", "UK", "USA");

Q31) Fulfill missing parts in SQL statement below to find all products listed in the *Products* table whose price belongs to range from 5 to 25 EUR, including the begin and end values.

_____ _____
_____ Products
_____ Price _____ ___ _____ _____;

Q32) What statement would you use to change temporarily name of the *CustomerName* field to *Customer* within *Customer* table?

a) SELECT CustomerName AS Customer FROM Customers;
b) SELECT CustomerName ALIAS Customer FROM Customers;
c) SELECT CustomerName LIKE Customer FROM Customers;
d) SELECT CustomerName TO Customer FROM Customers;

Q33) Fulfill missing parts in SQL statement below to select all orders with existing customer information.

SELECT Orders.OrderID, Customers.CustomerName
FROM _____
_____ Customers _____ _____. CustomerID = Customers. CustomerID;

Q34) Fulfill missing parts in SQL statement below to select all customers and any orders they might have.

SELECT Customers.CustomerName, Orders.OrderID
FROM _____
_____ Orders _____ _____.CustomerID = Orders.CustomerID;

Q35) Fulfill missing parts in SQL statement below to select all customers and any orders they might have.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM _____________
_____________ Customers _____  Orders.CustomerID = _____________.CustomerID;
```

Q36) Fulfill missing parts in SQL statement below to select all customers and all orders.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM _____________
_____________________ Orders _____  Customers.CustomerID = _____________.CustomerID;
```

Q37) What operator would you use to merge selects from two different tables with the same number of columns in the same order and with similar data types?

a) JOIN
b) MERGE
c) UNITE
d) UNION

Q38) Fulfill missing parts in SQL statement below to calculate overal *Quantity* for each *ProductID* and arrange the resulting list in descending order according to this new metric.

```
SELECT ____________________________ AS Overall_Quantity
FROM OrderDetails
_____________  _________________
_____________  _________________;
```

Q39) Fulfill missing parts in SQL statement below to filter products whose overal *Quantity* is higher than 100 and arrange the resulting list in descending order according to the overal *Quantity*.

```
SELECT ____________________________________ AS Overall_Quantity
FROM OrderDetails
_____________  _________________
_____________  _________________
_____________  _________________;
```

Q40) Fulfill missing parts in SQL statement below to create new field that will classify products listed in the *Products* table as "Cheap" when their *Price* will be lower than 10 EUR or as "Expensive" otherwise.

```
SELECT ProductID, Price,
________
        ________ Price ________  ________ "Cheap"
        ________ "Expensive"
________ AS Price_Level
FROM Products;
```

Test of Basic Knowledge of SQL – Key

Topic: CREATE DATABASE, Study Link: https://www.w3schools.com/sql/sql_create_db.asp

Q1) What SQL statement would you use to create SQL database named *Test*?

a) CREATE DBT Test;
b) CREATE Test;
c) CREATE DATABASE Test;
d) MAKE DATABASE Test;

Topic: DROP DATABASE, Study Link: https://www.w3schools.com/sql/sql_drop_db.asp

Q2) What SQL statement would you use to remove SQL database named *Test*?

a) DROP DATABASE Test;
b) REMOVE Test;
c) DROP DBT Test;
d) REMOVE DATABASE Test;

Topic: ENTITY RELATIONSHIP DIAGRAM, Study Link:
https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model

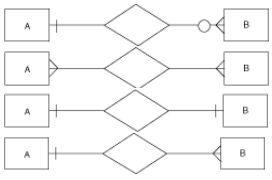
Q3) Assign individual relationships that one can find in ERD (Entity Relationship Diagram) to their appropriate descriptions.

a)
b)
c)
d)
e)
f)
g)

1) zero or many (optional) = **g)**
2) one or more (mandatory) = **d)**
3) one to one = **a)**
4) one and only one (mandatory) = **e)**
5) one to many (mandatory) = **b)**
6) many = **c)**
7) zero or one (optional) = **f)**

Topic: ENTITY RELATIONSHIP DIAGRAM, Study Link:
https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model

Q4) Which kind of relationship best describes relationship between *Products* (A) and *OrderDetails* (B) tables?

a)
b) 

c)
d)

Topic: CREATE TABLE, Study Link: https://www.w3schools.com/sql/sql_create_table.asp

Q5) Fulfill missing parts in SQL statement below to create *Products* table.

```
CREATE TABLE Products  (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(255),
  SupplierID INT,
  CategoryID INT,
  Unit VARCHAR(255),
  Price INT
);
```

Topic: Constraints – PRIMARY KEY, Study Link: https://www.w3schools.com/sql/sql_constraints.asp

Q6) Primary Key is a combination of two types of constraints. Find the correct one in the list below.

a) UNIQUE & DEFAULT
b) CHECK & UNIQUE
c) UNIQUE & NOT NULL
d) CHECK & AUTO INCREMENT

Topic: Constraints – FOREIGN KEY, Study Link: https://www.w3schools.com/sql/sql_foreignkey.asp

Q7) Which field in the *Orders* table has a role of FOREIGN KEY in relation to *Customers* table?

a) CustomerID
b) OrderID
c) OrderDate
d) CustomerName

Topic: INSERT INTO, Study Link: https://www.w3schools.com/sql/sql_insert.asp

Q8) What SQL statement would you use to put a new record into the *Orders* table?

a) INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES (200, 125, '2018-02-05');
b) PUT INTO Orders (OrderID, CustomerID, OrderDate) VALUES (200, 125, '2018-02-05');
c) INSERT VALUES (200, 125, '2018-02-05') INTO Orders (OrderID, CustomerID, OrderDate);
d) PUT VALUES (200, 125, '2018-02-05') INTO Orders (OrderID, CustomerID, OrderDate);

Topic: UPDATE, Study Link: https://www.w3schools.com/sql/sql_update.asp

Q9) What SQL statement would you use to modify the existing record of *CustomerID* to 10 for *OrderID* #10308 in the *Orders* table?

a) CHANGE SET Orders CustomerID = 10 WHERE OrderID = 10308;
b) UPDATE SET Orders CustomerID = 10 WHERE OrderID = 10308;
c) CHANGE Orders SET CustomerID = 10 WHERE OrderID = 10308;
d) UPDATE Orders SET CustomerID = 10 WHERE OrderID = 10308;

Topic: DELETE, Study Link: https://www.w3schools.com/sql/sql_delete.asp

Q10) What SQL statement would you use to delete order with *OrderID* #10308 from the *Orders* table?

a) DELETE FROM Orders WHERE OrderID = 10308;
b) CUT FROM Orders WHERE OrderID = 10308;
c) MOVE Orders WHERE OrderID = 10308;
d) REMOVE FROM Orders WHERE OrderID = 10308;

Topic: ALTER TABLE, Study Link: https://www.w3schools.com/sql/sql_alter.asp

Q11) What SQL statement would you use to add an *Age* field to the *Customers* table?

a) UPDATE TABLE Customers ADD Age INT;
b) ALTER TABLE Customers ADD Age INT;
c) UPDATE TABLE ADD Age INT Customers;
d) ALTER TABLE ADD Age INT Customers;

Topic: DROP TABLE, Study Link: https://www.w3schools.com/sql/sql_drop_table.asp

Q12) What SQL statement would you use to remove *Customers* table from *Test* database?

a) REMOVE TABLE Customers;
b) REMOVE Customers;
c) DROP Customers;
d) DROP TABLE Customers;

Topic: SELECT, Study Link: https://www.w3schools.com/sql/sql_select.asp

Q13) Fulfill missing parts in SQL statement below to extract all records from *Customers* table.

SELECT *
FROM Customers;

Q14) Fulfill missing parts in SQL statement below to extract *CustomerName* and *Address* from *Customers* table.

SELECT CustomerName, Address
FROM Customers;

Topic: SELECT DISTINCT, Study Link: https://www.w3schools.com/sql/sql_distinct.asp

Q15) Fulfill missing parts in SQL statement below to extract all distinct countries from *Customers* table.

SELECT DISTINCT Country
FROM Customers;

Topic: WHERE, Study Link: https://www.w3schools.com/sql/sql_where.asp

Q16) Fulfill missing parts in SQL statement below to extract all records from *Products* table that will include only products with price higher than 20 EUR.

SELECT *
FROM Products
WHERE Price **> 20**;

Topic: Null Values, Study Link: https://www.w3schools.com/sql/sql_null_values.asp

Q17) Fulfill missing parts in SQL statement below to extract all records from *Customers* table that will include only those customers who have NULL values in *Address* field.

SELECT *
FROM Customers
WHERE Address IS NULL;

Topic: AND, OR, NOT, Study Link: https://www.w3schools.com/sql/sql_and_or.asp

Q18) Fulfill missing parts in SQL statement below to extract all records from *Customers* table that will include only those customers who are from Germany or UK.

SELECT *
FROM Customers
WHERE Country **= "Germany" OR Country = "UK"** ;

Q19) Fulfill missing parts in SQL statement below to extract all records from *Customers* table that will include only those customers who are not from USA.

SELECT *
FROM Customers
WHERE NOT Country **= "USA"**;

Q20) Fulfill missing parts in SQL statement below to extract all records from *Products* table that will include only those products that are supplied by supplier with *SupplierID* #1 and that belong to *CategoryID* #2.

```
SELECT *
FROM Products
WHERE  SupplierID = 1   AND  CategoryID = 2;
```

Topic: ORDER BY, Study Link: https://www.w3schools.com/sql/sql_orderby.asp

Q21) Fulfill missing parts in SQL statement below to arrange records in *Products* table according to *Price* in descending order.

```
SELECT *
FROM Products
ORDER BY Price DESC;
```

Topic: SELECT TOP, Study Link: https://www.w3schools.com/sql/sql_top.asp

Q22) Fulfill missing parts in SQL statement below to extract the first 50 records from *Customers* table.

```
SELECT  TOP 50 *
FROM Customers;
```

Topic: MIN, MAX, Study Link: https://www.w3schools.com/sql/sql_min_max.asp

Q23) Fulfill missing parts in SQL statement below to find maximum *Price* for products listed in *Products* table.

```
SELECT MAX(Price)
FROM Products;
```

Topic: COUNT, AVG, SUM, Study Link: https://www.w3schools.com/sql/sql_count_avg_sum.asp

Q24) What statement will you use to count number of records within *Customers* table?

a) SELECT ALL FROM Customers;
b) SELECT N FROM Customers;
c) SELECT COUNT(*) FROM Customers;
d) SELECT NROW FROM Customers

Q25) Fulfill missing parts in SQL statement below to find average *Price* for products listed in *Products* table.

```
SELECT AVG(Price)
FROM Products;
```

Q26) Fulfill missing parts in SQL statement below to find overal number of ordered products using *Quantity* field in *OrderDetails* table.

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

Topic: LIKE, WILDCARDS Study Link: https://www.w3schools.com/sql/sql_like.asp, https://www.w3schools.com/sql/sql_wildcards.asp

Q27) Fulfill missing parts in SQL statement below to find all customers listed in the *Customers* table whose name starts with letter „b".

```
SELECT *
FROM Customers
WHERE CustomerName LIKE "b%";
```

Q28) Fulfill missing parts in SQL statement below to find all customers listed in the *Customers* table whose name starts with letter „b" and ends with letter „o".

```
SELECT *
FROM Customers
WHERE CustomerName LIKE "b%o";
```

Q29) Fulfill missing parts in SQL statement below to find all customers listed in the *Customers* table whose name has letter „b" in the second position.

```
SELECT *
FROM Customers
WHERE CustomerName LIKE "_b%";
```

Topic: IN, Study Link: https://www.w3schools.com/sql/sql_in.asp

Q30) Fulfill missing parts in SQL statement below to find all customers listed in the *Customers* table who live in Germany, UK, and USA.

```
SELECT *
FROM Customers
WHERE Country IN ("Germany", "UK", "USA");
```

Topic: BETWEEN, Study Link: https://www.w3schools.com/sql/sql_between.asp

Q31) Fulfill missing parts in SQL statement below to find all products listed in the *Products* table whose price belongs to range from 5 to 25 EUR, including the begin and end values.

```
SELECT *
FROM Products
WHERE Price BETWEEN 5 AND 20;
```

Topic: ALIASES, Study Link: https://www.w3schools.com/sql/sql_alias.asp

Q32) What statement would you use to change temporarily name of the *CustomerName* field to *Customer* within *Customer* table?

a) **SELECT CustomerName AS Customer FROM Customers;**
b) SELECT CustomerName ALIAS Customer FROM Customers;
c) SELECT CustomerName LIKE Customer FROM Customers;
d) SELECT CustomerName TO Customer FROM Customers;

Topic: INNER JOIN, Study Link: https://www.w3schools.com/sql/sql_join_inner.asp

Q33) Fulfill missing parts in SQL statement below to select all orders with existing customer information.

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers. CustomerID;
```

Topic: LEFT JOIN, Study Link: https://www.w3schools.com/sql/sql_join_left.asp

Q34) Fulfill missing parts in SQL statement below to select all customers and any orders they might have.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON  Customers.CustomerID = Orders.CustomerID;
```

Topic: RIGHT JOIN, Study Link: https://www.w3schools.com/sql/sql_join_right.asp

Q35) Fulfill missing parts in SQL statement below to select all customers and any orders they might have.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Orders
RIGHT JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Topic: FULL JOIN, Study Link: https://www.w3schools.com/sql/sql_join_full.asp

Q36) Fulfill missing parts in SQL statement below to select all customers and all orders.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

Topic: UNION, Study Link: https://www.w3schools.com/sql/sql_union.asp

Q37) What operator would you use to merge selects from two different tables with the same number of columns in the same order and with similar data types?

a) JOIN
b) MERGE
c) UNITE
d) UNION

Topic: GROUP BY, Study Link: https://www.w3schools.com/sql/sql_groupby.asp

Q38) Fulfill missing parts in SQL statement below to calculate overal *Quantity* for each *ProductID* and arrange the resulting list in descending order according to this new metric.

```
SELECT ProductID, SUM(Quantity) AS Overall_Quantity
FROM OrderDetails
GROUP BY ProductID
ORDER BY Overall_Quantity DESC;
```

Topic: HAVING, Study Link: https://www.w3schools.com/sql/sql_having.asp

Q39) Fulfill missing parts in SQL statement below to filter products whose overal *Quantity* is higher than 100 and arrange the resulting list in descending order according to the overal *Quantity*.

```
SELECT ProductID, SUM(Quantity) AS Overall_Quantity
FROM OrderDetails
GROUP BY ProductID
HAVING Overall_Quantity > 100
ORDER BY Overall_Quantity DESC;
```

Topic: CASE, Study Link: https://www.w3schools.com/sql/sql_case.asp

Q40) Fulfill missing parts in SQL statement below to create new field *Price_Level* that will classify products listed in the *Products* table as "Cheap" when their *Price* will be lower than 10 EUR or as "Expensive" otherwise.

```
SELECT ProductID, Price,
CASE
        WHEN Price < 10 THEN "Cheap"
        ELSE "Expensive"
END AS Price_Level
FROM Products;
```
