Certification Questions Practice Set (Data Transformation)

1. With respect to external functions, what are all the factors that contribute to affecting the number of concurrent calls from Snowflake to a remote OR proxy service ? Choose any two.

   A) The size of the database.

   B) The size of the external stage.

   C) The size of each user's query.

   D) The number of warehouses.

   E) The instance of Snowflake.

   Answers : C, D

   Explanation :

   https://docs.snowflake.com/en/sql-reference/external-functions-implementation

Concurrency

Resource requirements depend upon the way that rows are distributed across calls (many parallel calls with a few rows each vs. one call with the same total number of rows). A system that supports high capacity does not necessarily support high concurrency, and vice-versa. You should estimate the peak concurrency required, as well as the largest reasonable individual workloads, and provide enough resources to handle both types of peaks.

Furthermore, the concurrency estimate should take into account that Snowflake can parallelize external function calls. A single query from a single user might cause multiple calls to the remote service in parallel. Several factors affect the number of concurrent calls from Snowflake to a proxy service or remote service, including:

- The number of concurrent users who are running queries with external functions.
- The size of each user's query.
- The amount of compute resources in the virtual warehouse (i.e. the warehouse size).
- The number of warehouses.

---

2. Mark all the correct recommendations to improve latency of External functions?

A) Place the API Gateway in a different cloud platform and region compared to the one having your Snowflake instance
B) If you wrote the remote service (rather than using an existing service), deploy that remote service in the same cloud platform and region as it is called from.
C) Filter data using Snowflake SQL and then send only filtered records to reduce latency of response

Answers : B, C

Explanation : https://docs.snowflake.com/en/sql-reference/external-functions-best-practices

Minimize latency

To minimize latency and improve performance of external function calls, Snowflake recommends doing the following when practical:

- Put the API Gateway in the same cloud platform and region as Snowflake instances that call it most frequently (or with the largest amount of data).
- If you wrote the remote service (rather than using an existing service), deploy that remote service in the same cloud platform and region as it is called from.
- Send as little data as possible. For example, if the remote service will examine inputs values and operate on only a subset of them, then it is usually more efficient to filter in SQL and send only the relevant rows to the remote service, rather than send all rows to the remote service and let it filter.

  As another example, if you are processing a column that contains large semi-structured data values, and the remote service will operate on only a small piece of each of those data values, it is usually more efficient to extract the relevant piece using Snowflake SQL and send only that piece, rather than send the entire column and have the remote service do the extraction of the small piece before processing.

---

3. Which of the following statements are TRUE about external functions:(choose any two)

A) External functions can be overloaded, 2 different functions can be of the same name but can have different signatures.

B) External functions cannot be overloaded.

C) External functions can accept parameters.

D) The return value of external functions can be of compound value like that of a JSON which is of VARIANT data type.

Answers : A, C, D

Explanation :

From the perspective of a user running a SQL statement, an external function behaves like any other UDF . External functions follow these rules:

- External functions return a value.
- External functions can accept parameters.
- An external function can appear in any clause of a SQL statement in which other types of UDF can appear. For example:

```
select my_external_function_2(column_1, column_2)
    from table_1;

select col1
    from table_1
    where my_external_function_3(col2) < 0;

create view view1 (col1) as
    select my_external_function_5(col1)
        from table9;
```

- An external function can be part of a more complex expression:

```
select upper(zipcode_to_city_external_function(zipcode))
  from address_table;
```

- The returned value can be a compound value, such as a VARIANT that contains JSON.
- External functions can be overloaded; two different functions can have the same name but different signatures (different numbers or data types of input parameters).

---

4. Which commands will start an evaluation of a DataFrame? Select two.

A) DataFrame.random_split()

B) DataFrame.collect()

C) DataFrame.select()

D) DataFrame.col()

E) DataFrame.show()

Answers : B, E

Explanation :

https://docs.snowflake.com/en/developer-guide/snowpark/python/working-with-dataframes

Performing an Action to Evaluate a DataFrame ¶

As mentioned earlier, the DataFrame is lazily evaluated, which means the SQL statement isn't sent to the server for execution until you perform an action. An action causes the DataFrame to be evaluated and sends the corresponding SQL statement to the server for execution.

The following methods perform an action:

Class	Method	Description
`DataFrame`	`collect`	Evaluates the DataFrame and returns the resulting dataset as an `list` of `Row` objects.
`DataFrame`	`count`	Evaluates the DataFrame and returns the number of rows.
`DataFrame`	`show`	Evaluates the DataFrame and prints the rows to the console. This method limits the number of rows to 10 (by default).
`DataFrameWriter`	`save_as_table`	Saves the data in the DataFrame to the specified table. Refer to Saving Data to a Table.

---

5. A company has an extensive script in Scala that transforms data by leveraging DataFrames. A Data Engineer needs to move these transformations to Snowpark. What characteristics of data transformations in Snowpark should be considered to meet this requirement? (Choose two.)

A) It is possible to join multiple tables using DataFrames.

B) Snowpark operations are executed lazily on the server.

C) User-Defined Functions (UDFs) are not pushed down to Snowflake.

D) Snowpark requires a separate cluster outside of Snowflake for computations.

Answers : A, B

---

6. What is the correct method of querying a User-Defined Table Function (UDTF) that returns two columns (col1, col2)?

1. SELECT my_udtf(col1, col2)
2. SELECT $1, $2 FROM TABLE(my_udtf())
3. SELECT TABLE(my_udtf(col1, col2))
4. SELECT $1, $2 FROM RESULT_SCAN(my_udtf())

Answers : 2

Explanation : https://docs.snowflake.com/en/developer-guide/udf/sql/udf-sql-tabular-functions

$1, $2 are regular positional arguments used to return the 1st two columns of the result of the SELECT query.

To call any UDTF in Snowflake the syntax to be followed is :

```
SELECT ...
  FROM TABLE ( udtf_name (udtf_arguments) )
```

---

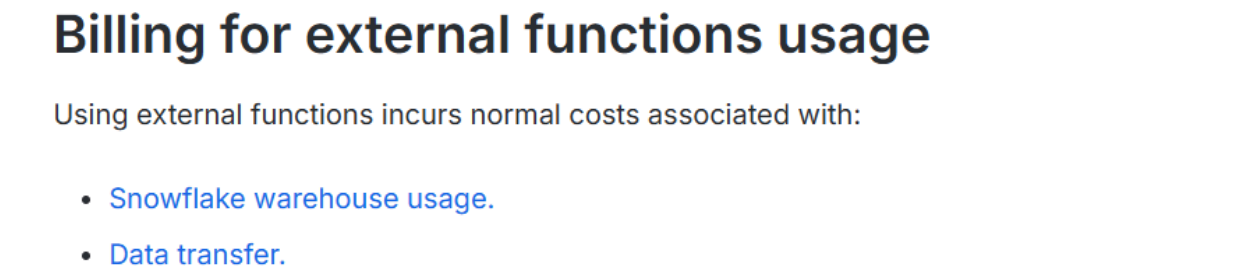
7. Which of the following is a characteristic of Snowflake external functions?

1. They must be processed through a HTTP proxy service with a GET request.
2. They must be written in JavaScript or SQL.

3. They do not require a virtual warehouse or a schema.
4. They incur costs through virtual warehouse usage and data transfer.

Answer : 4

Explanation : https://docs.snowflake.com/en/sql-reference/external-functions-introduction



---

8. When using the CURRENT_ROLE and CURRENT_USER functions with secure UDFs that will be shared with Snowflake accounts, Snowflake returns a NULL value for these functions?

A. TRUE

B. FALSE

Answer : TRUE

Explanation :

---

9. While creating even a secure UDF, Snowflake recommends using randomized identifiers (e.g., generated by UUID_STRING) instead of sequence-generated values?

A. TRUE

B. FALSE

Answer :

Explanation :

---

10. A Data Engineer is performing the following steps in sequence while working on Stream s1 created on table t1.

Step 1: Begin transaction.

Step 2: Query stream s1 on table t1.

Step 3: Update rows in table t1.

Step 4: Query stream s1.

Step 5: Commit transaction.

Step 6: Begin transaction.

Step 7: Query stream s1.

Mark the incorrect operational statements:

A. For Step 2, the stream returns the change data capture records between the current position to the Transaction 1 start time. If the stream is used in a DML statement, the stream is then locked to avoid changes by concurrent transactions.

B. For Step 4, returns the CDC data records by streams with updated rows happened in Step 3 because streams work in repeated committed mode in which statements see any changes made by previous statements executed within the same transaction, even though those changes are not yet committed.

C. For Step 5, if the stream was consumed in DML statements within the transaction, the stream position advances to the transaction start time.

D. For Step 7, results do include table changes committed by Transaction 1.

E. If Transaction 2 had begun before Transaction 1 was committed, queries to the stream would have returned a snapshot of the stream from the position of the stream to the beginning time of Transaction 2 and would not see any changes committed by Transaction 1

Answer : B

Explanation : Query will not return any CDC records from the stream because the transaction is yet not committed

---

11. Does an external function's identifier need to be unique for the schema in which the function is created?

1. True
2. False

Answer : False

Explanation : External functions are unique in terms of their signature i.e name + arguments passed

---

12. A table called MY_TABLE contains the following information:

Which command should we use to convert the array into individual rows?

1. JOIN
2. FLATTEN
3. AGGREGATE

Answer : 2

---

13. What information does an API Integration object store?

1. The cloud platform provider.
2. The type of proxy service.
3. The identifier and access credentials for a cloud platform role with sufficient privileges to use the proxy service.
4. Allowed (and optionally blocked) endpoints and resources on those proxy services.
5. All of the above.

Answer : 5

---

14. Samantha is working on a data transformation task in Snowflake and needs to use data from an external stage that resides in AWS S3. What is the most efficient way to access this data?

A. Load the data from S3 into a Snowflake table before processing.
B. Query the data directly using an external table.
C. Use the COPY INTO command to load data from S3 into a staging table.
D. Move the data to local storage and then load it into Snowflake.

Answer : B

---

15 .When using Snowpark to read data from a stage file in Snowflake, which method is used to open the file in read-binary mode?

A) SnowflakeFile.read()

B) SnowflakeFile.open()

C) SnowflakeFile.load()

D) SnowflakeFile.read_binary()

Answer : B

Explanation : In Snowpark, to read a file from a stage in binary mode, you would use the SnowflakeFile.open() method. This method allows you to open the file in various modes, including binary mode, for reading the file contents directly from the stage.

The other options, like read(), load(), or read_binary(), are not valid methods in this context for reading binary data from a stage file using Snowpark.

---

16. In Snowpark, if you want to ensure that your DataFrame operations are executed lazily until an action is called, which of the following methods should you use to trigger execution?

A) df.show()

B) df.collect()

C) df.persist()

D) df.describe()

Answer : B

---

17. Which of the following scenarios is most appropriate for using a stored procedure in Snowflake?

A) Simple transformations on a single table.

B) Complex business logic involving multiple tables and conditional logic.

C) Real-time data streaming.

D) Creating a temporary view

Answer : B

---

18. When defining a stored procedure in Snowflake using JavaScript, which function is used to execute a SQL statement within the procedure?

A) sql.execute()

B) snowflake.query()

C) snowflake.execute()

D) sql.executeQuery()

Answer : C

---

19. You've written a procedure which returns a string value that needs to be used in another function call. Markall the options below that will help correctly capture the returned value from the procedure

- A) Call the stored procedure and then call the RESULT_SCAN function and pass it the statement ID generated for the stored procedure

- B) store a result set in a temporary table or permanent table, and use that table after returning from the stored procedure call.
- C) Capture the value in a binding variable within a Snowflake scripting block in SQL

D) This isn't possible since procedures do not return any value

Answers : A, B, C

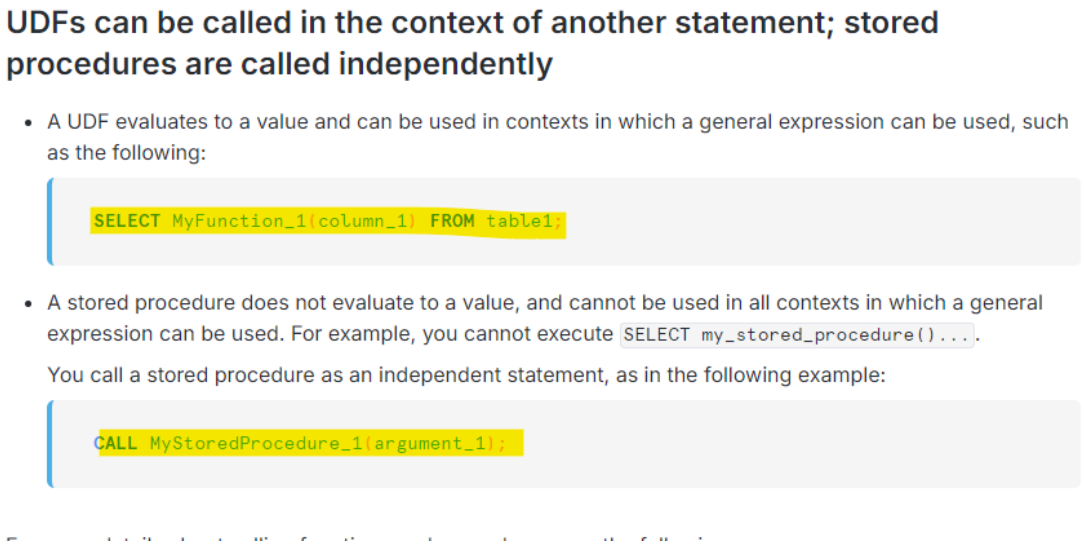
---

20. Mark all incorrect statements about functions/procedures :

- Procedures can be called in context of another statement while UDFs are called independently
- UDF return values are directly usable in SQL while stored procedure return values may not be
- Multiple UDFs may be called with one statement. However, a single stored procedure is called with one statement
- Procedures may access the database with simple queries only. However, UDFs can execute DDL and DML statements

Answers : A, D

D is surely wrong since UFs can't execute DDL and DML statements.



UDFs can be called in the context of another statement; stored procedures are called independently

- A UDF evaluates to a value and can be used in contexts in which a general expression can be used, such as the following:

```
SELECT MyFunction_1(column_1) FROM table1;
```

- A stored procedure does not evaluate to a value, and cannot be used in all contexts in which a general expression can be used. For example, you cannot execute `SELECT my_stored_procedure()...`.

  You call a stored procedure as an independent statement, as in the following example:

```
CALL MyStoredProcedure_1(argument_1);
```

So, as per above A is also wrong

---

21. How can the following relational data be transformed into semi-structured data using the LEAST amount of operational overhead?

A. Use the TO_JSON function.

B. Use the PARSE_JSON function to produce a VARIANT value.

C. Use the OBJECT_CONSTRUCT function to return a Snowflake object.

D. Use the TO_VARIANT function to convert each of the relational columns to VARIANT.

Answer : C

Explanation : B is out because PARSE_JSON consumes a JSON object.

---

22. Which methods can be used to create a DataFrame object in Snowpark? (Choose three.)

A. session.jdbc_connection()

B. session.read.json()

C. session.table()

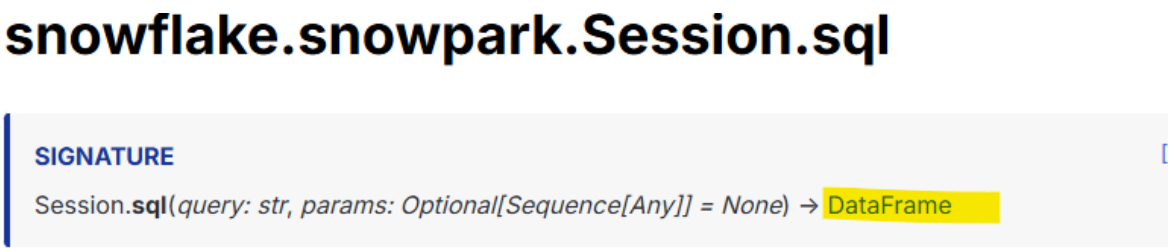
D. DataFrame.write()

E. session.builder()

F. session.sql()

Answers : B, C, F

A is for database connection. E is for building the session. B, C and F return the data type as Dataframe



`session.read.json()`: This method is used to create a DataFrame from JSON files.

`session.table()`: This method is used to create a DataFrame by referencing an existing table in Snowflake.

`session.sql()`: This method creates a DataFrame by executing an SQL query and returning the result as a DataFrame.

---

23. The JSON below is stored in a VARIANT column named V in a table named jCustRaw :


```
{
 "_id": "6282638561cf48544e2ef7e9",
 "company": "FLYBOYZ",
 "isActive": true,
 "name": "Dean Head",
 "teamMembers": [
  {
   "age": 29,
```

```
    "eyeColor": "green",
    "name": "Dominique Grimes",
    "registered": "2017-02-19T06:12:36+06:00"
  },
  {
    "age": 39,
    "eyeColor": "green",
    "name": "Pearl Dunlap",
    "registered": "2018-05-12T09:21:42+05:00"
  },
  {
    "age": 22,
    "eyeColor": "blue",
    "name": "Cardenas Warren",
    "registered": "2019-04-08T01:24:29+05:00"
  }
 ]
}
```


Which query will return one row per team member (stored in the teamMembers array) along with all of the attributes of each team member?

A.

```
select
    t2.name AS memberName,
    t2.registered::timestamp AS registeredDttm,
    t2.age AS age,
    t2.eyeColor AS eyeColor
from jCustRaw t1,
lateral flatten(v) t2;
```

B.

```
select
    t2.value:name::varchar AS memberName,
    t2.value:registered::timestamp AS registeredDttm,
    t2.value:age::number AS age,
    t2.value:eyeColor::varchar AS eyeColor
from jCustRaw t1,
lateral flatten(input => v:teamMembers) t2;
```

C.

```
select
    v:teamMembers.name::varchar AS memberName,
    v:teamMembers.registered::timestamp AS registeredDttm,
```

```
    v:teamMembers.age::number AS age,
    v:teamMembers.eyeColor::varchar AS eyeColor
from jCustRaw;
```

D.

```
select
    v:teamMembers[0].name::varchar AS memberName,
    v:teamMembers[0].registered::timestamp AS registeredDttm,
    v:teamMembers[0].age::number AS age,
    v:teamMembers[0].eyeColor::varchar AS eyeColor
from jCustRaw;
```

Answer :

Explanation :

---

24. What should a Data Engineer consider when configuring an API integration to create external functions in Snowflake? (Select TWO).

1. An API integration object can be used across different cloud platform accounts.
2. Multiple external functions can use the same API integration object, so the same HTTPS proxy service could be used.
3. The role `SYSADMIN` has granted the global `CREATE INTEGRATION` privilege to other roles for a decentralized API integration.

4. The Snowflake default roles `ACCOUNTADMIN` and `SYSADMIN` can execute `CREATE API INTEGRATION` statements.
5. Snowflake accounts can have multiple API integration objects for different cloud platform accounts.

Answers : 2,5

Explanation :

---

25. Which SQL statements are valid to execute stored procedures? (Select TWO).

1. call proc1(1), proc2(2);
2. CALL stproc1(2 * 5.14::float);
3. CALL stproc1(SELECT COUNT(*) FROM stproc_test_table1);
4. select * from (call proc1(1));
5. call proc1(1) + proc1(2);

Answer : 2,3

Explanation :

---

26. Data from an event log application, generating heavily nested JSON, is streaming into Snowflake for security analytics.

What is best practice for optimizing performance on semi-structured workloads on Snowflake?

1. Specify the schema of the load during ingestion while storing the log import date in a `VARIANT` column.
2. Put a structured view on top of the semi-structured table.
3. Use external tables and execute queries directly on cloud storage without ingesting the data.
4. Extract and store the timestamp from the log into a separate column for better pruning, while storing the JSON log in a `VARIANT` column

Answer : 4

Explanation :

An upstream application from Snowflake generates a JSON sales receipt document, containing a receipt header and multiple line items. Thousands of receipts are collected in a few hundred files daily on cloud storage. A Data Engineer would like to regularly ingest these files into Snowflake for querying.

How should the Engineer load and transform the data in Snowflake?

1. In the `INGEST` command, extract all attributes from the JSON records, flatten the JSON and store the results directly in a structured `SALES` table and `LINEITEMS` table.
2. Have the upstream application process and split the sales receipt into separate files and then load each file into a Snowflake table with a `VARIANT` column.
3. Use Snowpipe to continuously load the data in a `VARIANT` column in a table. Create views and/or another table to structure the data for efficient querying.
4. In the `INGEST` command, first flatten and load the `SALES` table and then flatten and load the `LINEITEMS` table.

Answer :

Explanation :

A Data Engineer is importing JSON data from an external stage into a table with a `VARIANT` column using the `COPY INTO` command. During testing, the Engineer discovers that the import sometimes fails with parsing errors, due to malformed JSON values. The Engineer decides to set the `VARIANT` column to `NULL` when a parsing error is encountered.

Which function should be used to meet this requirement?

1. PARSE_JSON
2. TRY_PARSE_JSON
3. VALIDATE
4. TO_JSON

Assume the below table exists:

create table foo (

name STRING,

entered_at TIMESTAMP);

Consider the following stored procedure:

```
create procedure sp1()
returns varchar
language javascript
AS
$$
snowflake.execute (
  {sqlText: "insert into foo values ('Bob', CURRENT_TIMESTAMP)"}
);

snowflake.execute (
  {sqlText: "begin transaction"}
);
snowflake.execute (
  {sqlText: "insert into foo values ('Jane', CURRENT_TIMESTAMP)"}
);
snowflake.execute (
  {sqlText: "CALL sp2()"}
);
snowflake.execute (
  {sqlText: "rollback"}
);
```

```
snowflake.execute (
  {sqlText: "insert into foo values ('Frank', CURRENT_TIMESTAMP)"}
);

return "";
$$;
create procedure sp2()
returns varchar
language javascript
AS
$$
snowflake.execute (
  {sqlText: "begin transaction"}
);

snowflake.execute (
  {sqlText: "insert into foo values ('Zach', CURRENT_TIMESTAMP)"}
);
snowflake.execute (
  {sqlText: "commit"}
);
```

return "";

$$;

Assuming auto commit is set to TRUE, the below commands are issued, in order:

TRUNCATE FOO;

INSERT INTO FOO VALUES ('Mary', CURRENT_TIMESTAMP);

CALL SP1();

SELECT name FROM FOO ORDER BY entered_at;

What names will be returned by the final SELECT statement?

1. Mary, Bob, Zach, Frank
2. Mary, Bob, Jane, Zach, Frank
3. Mary, Bob, Jane, Frank
4. Mary, Bob, Frank

Consider the below pseudocode:

CREATE PROCEDURE p1() ...

$$

INSERT INTO test_table VALUES ('1');

INSERT INTO test_table VALUES ('2');

```
$$;


CREATE PROCEDURE p2() ...
$$
  INSERT INTO test_table VALUES ('3');
  BEGIN TRANSACTION;
  INSERT INTO test_table VALUES ('4');
  CALL p1();
  COMMIT WORK;
  INSERT INTO test_table VALUES ('5');
$$;
INSERT INTO test_table VALUES ('6');
ALTER SESSION SET AUTOCOMMIT = FALSE;
BEGIN TRANSACTION;
INSERT INTO test_table VALUES ('7');
CALL p2();
INSERT INTO test_table VALUES ('8');
ROLLBACK;
```

What additional values will be added to `test_table` after this code is executed?

1. 6, 7, 3, 4, 1, 2

2. 6, 4, 1, 2
3. 1, 2, 4, 6
4. 2, 1, 4, 3, 7, 6

What will occur if a User-Defined Function (UDF) executes a `MERGE INTO` command?

1. The command would not qualify as a scalar or table function.[1]
2. The command would run as expected.
3. The command would fail because only updates or inserts can be performed.
4. Privileges would need to be applied to the function owner to access the tables.

Explanation : In Snowflake UDF can be used for only READ only kind of transactions. B & C were direct eliminations.

Which query will return a SQL NULL when it is executed?

1. select parse_json('[null]');
2. select parse_json('{ "id": null }'):a;
3. select to_char(parse_json('{ "id": null }'):id);
4. select parse_json('null');

Which statements accurately describe the relationship between JavaScript stored procedures and transactions? (Select TWO).

1. Stored procedures do not support transactions.
2. A transaction can be inside a stored procedure.
3. Only one transaction can be executed inside a stored procedure.
4. Transactions can be started in one stored procedure and finished in another stored procedure.

A stored procedure can be inside a transaction