Week 4 Quiz

1. What are the benefits of Orchestration over simple Cron scheduling?
    - With Orchestration, you can exclusively use the user interface (UI) to configure, define, and trigger your DAG.
    - Orchestration allows you to set up dependencies between data pipeline tasks, monitor tasks, receive alerts, and create fallback plans, compared to scheduling tasks with Cron.
    - Orchestration reduces the operational overhead compared to scheduling data pipeline tasks with Cron.
    - Orchestration simplifies the task of scheduling by eliminating the need for programming or scripting that is needed for scheduling data pipeline tasks with Cron.
2. Some of the orchestration engines require that you define your data pipeline as a Directed Acyclic Graph (DAG). What is a DAG?
    - A DAG is a graph used to visualize your data pipeline, where each task is a node and dependencies are illustrated as edges, which flow in one direction and can point to a previous task.
    - A DAG is a graph used to visualize your data pipeline, where each task is a node and dependencies are illustrated as edges indicating that data flows only in one direction and doesn't form any cycles.
    - A DAG is a graph used to visualize your data pipeline, where each node represents data and the tasks that need to be applied to the data are illustrated as edges.
3. One of the Airflow's core components is the scheduler. What is the role of the scheduler?
    - The scheduler is responsible for directly executing the tasks and updating their status in the UI.
    - The scheduler monitors all your DAGS and their tasks, and triggers the tasks based on the schedule or when their dependencies are complete.
    - The scheduler stores the Python scripts that define your DAGs and executes them on a schedule.
    - The scheduler is where you can monitor, manually trigger, and troubleshoot the behavior of your overall DAGs and the tasks within each DAG.

4. Which of the following tasks can you do in the Airflow UI? Select all that apply.

   - Update the Python script that contains the DAG definition.
   - Create a global variable.

     You can manually create global variables in the Airflow UI and use them in your code using the method `Variable.get()`.
   - Check the logs of any task to understand any execution errors in your DAG.

     This is one of the Airflow UI features that helps you understand the specific reasons behind task failures by offering detailed error messages that guide you in troubleshooting and resolving issues with your DAGS.
   - Manually trigger a DAG.

5. When you create a DAG instance in Airflow, which parameter do you need to specify if you don't want the scheduler to start the DAG runs for any missed interval when you unpause the DAG?

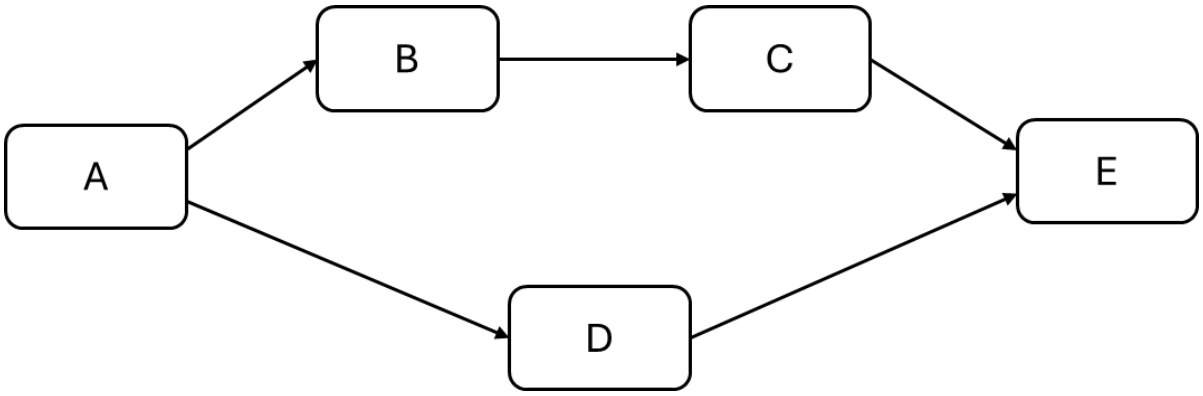
   - start_date
   - catchup
   - description
   - tags

   If this parameter is set to true, when you unpause the DAG, the scheduler will kick off a DAG run for any missed interval when the DAG was paused. If you don't want this to happen, then you need to set catchup to False.

6. XCom is an Airflow feature for sharing data between tasks. According to this week's videos, for which of the following use cases should XCom be used? Select all that apply.

   - To pass information about dates between tasks.
   - To pass a single value accuracy metric computed in an upstream task to a downstream task.
   - To pass metadata between tasks.
   - To pass large DataFrames between tasks.

7. Consider the following DAG.



Which of the following statements correctly represents the DAG structure?

- A>> [B>> C, D] >> E
- A>> D >> [B, C] >> E
- A>> [B, C] >> D >> E

8. Which of the following statements is true about TaskFlow API?
   - TaskFlow API is a new programming paradigm that relies on the explicit use of Airflow operators.
   - TaskFlow API is an old programming paradigm for defining tasks and their dependencies in a given data pipeline.
   - TaskFlow API is a programming paradigm that allows you to write your DAGs in an easier and more concise way using decorators.
   - TaskFlow API represents a REST API that allows external applications to interact with your DAG.
9. According to this week's lecture materials, which of the following is/are best practices for writing Airflow DAGS? Select all that apply.
   - Use variables to store values that are used more than once or that may need to be updated
   - Include top-level code before the DAG definition in the DAG file.
   - Group related operations into a single task.

10. Consider the following code:

```
task_1 = PythonOperator(task_id="extract", python_callable=extract_data)
task_2 = PythonOperator(task_id="transform", python_callable=transform_data)
task_3 = PythonOperator(task_id="load", python_callable=load_data)

#define dependencies
None >> None >> None
```

Which of the following statements correctly defines the dependencies?

- "extract">>"transform" >> "load"
- extract_data >> transform_data >> load_data
- task_1>> task_2 >> task_3