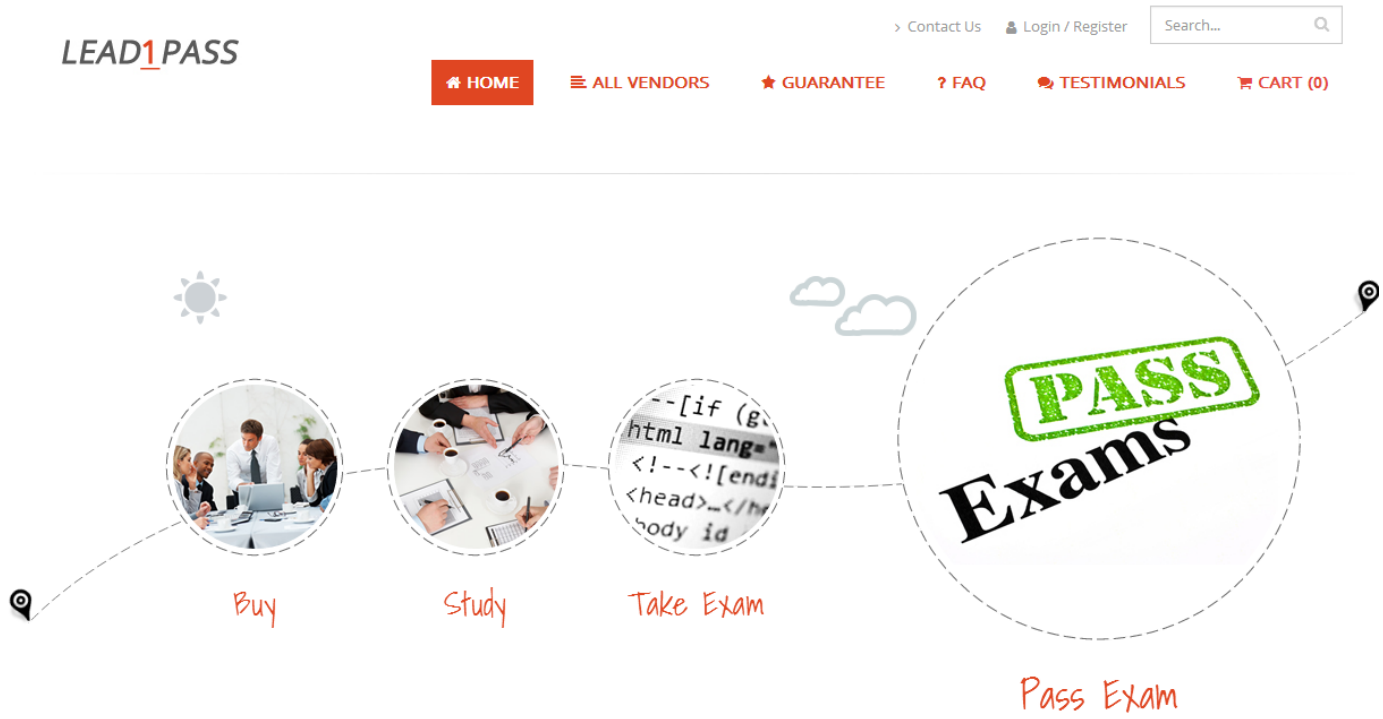
Lead1Pass



Try **PDF Demo** before you buy

Instant Download

After Payment, our system will send you the products you purchase in mailbox in a minute after payment. If not received within 2 hours, please contact us.

365 Days Free Updates

Free update is available within 365 days after your purchase. After 365 days, you will get 50% discounts for updating.

Money Back Guarantee

Full refund if you fail the corresponding exam in 60 days after purchasing. And Free get any another product.

Security & Privacy

We respect customer privacy. We use McAfee's security service to provide you with utmost security for your personal information & peace of mind.

http://www.lead1pass.com/

Latest Exam Guide & Learning Materials

Exam : **Databricks-Certified-Data-Engineer-Professional-JPN**

Title : Databricks Certified Data Engineer Professional Exam (Databricks-Certified-Data-Engineer-Professional日本語版)

Vendor : Databricks

Version : DEMO

QUESTION NO: 1
上流システムは、特定のデータバッチの日付をパラメータとしてDatabricks Jobs APIに渡すように構成されています。スケジュール対象のノートブックは、このパラメータを使用して、以下のコードでデータを読み込みます。
df = spark.read.format("parquet").load(f"/mnt/source/(date)")
上記のコード ブロックで使用されている日付 Python 変数を作成するには、どのコード ブロックを使用する必要がありますか?
A. date = spark.conf.get("date")
B. input_dict = input()
date= input_dict["date"]
C. import sys
date = sys.argv[1]
D. date = dbutils.notebooks.getParam("date")
E. dbutils.widgets.text("date", "null")
date = dbutils.widgets.get("date")
Answer: E
Explanation:
The code block that should be used to create the date Python variable used in the above code block is:
dbutils.widgets.text("date", "null") date = dbutils.widgets.get("date") This code block uses the dbutils.widgets API to create and get a text widget named "date" that can accept a string value as a parameter. The default value of the widget is "null", which means that if no parameter is passed, the date variable will be "null". However, if a parameter is passed through the Databricks Jobs API, the date variable will be assigned the value of the parameter.
For example, if the parameter is "2021-11-01", the date variable will be "2021-11-01". This way, the notebook can use the date variable to load data from the specified path.

QUESTION NO: 2
データアナリストの分散チームは、自動スケーリングが設定されたインタラクティブなクラスター上でコンピューティングリソースを共有しています。ワークスペース管理者は、コストとクエリスループットをより適切に管理するために、クラスターのスケールアップが多数の同時ユーザーによるものか、リソースを大量に消費するクエリによるものかを評価したいと考えています。
クラスターのサイズ変更イベントのタイムラインはどこで確認できますか?
A. ドライバーのログファイル
B. クラスターイベントログ
C. 実行者のログファイル
D. ワークスペース監査ログ
E. ガングリオン
Answer: B

QUESTION NO: 3
ジョブ実行履歴の保持に関して正しい記述はどれですか?

A. ジョブ実行ログをエクスポートまたは削除するまで保持されます
B. 90日間、またはカスタム実行構成を通じて実行IDが再利用されるまで保持されます。
C. 60日間保持され、その間にノートブックの実行結果をHTMLにエクスポートできます。
D. 60日間保存され、その後ログはアーカイブされます
E. 30日間保持され、その間にジョブ実行ログをDBFSまたはS3に配信できます。
Answer: C

QUESTION NO: 4
Databricksワークスペース管理者は、各データエンジニアリンググループに対して対話型クラスターを構成しました。コスト管理のため、クラスターは30分間操作が行われないと終了するように設定されています。
各ユーザーは、割り当てられたクラスターに対して、いつでもワークロードを実行できる必要があります。
ユーザーがワークスペースに追加されているが、権限が付与されていないと仮定すると、すでに構成されているクラスターを起動してアタッチするためにユーザーが必要とする最小限の権限は次のどれですか。
A. 必要なクラスターに対する「管理可能」権限
B.
ワークスペース管理者権限、クラスター作成が許可されます。必要なクラスターに対する「接続可能」権限
C. クラスタの作成が許可されます。必要なクラスタに対する「接続可能」権限
D. 必要なクラスターに対する「再起動可能」権限
E. クラスタの作成が許可されます。必要なクラスタに対する「再起動可能」権限
Answer: D
Explanation:
https://learn.microsoft.com/en-us/azure/databricks/security/auth-authz/access-control/cluster-acl
https://docs.databricks.com/en/security/auth-authz/access-control/cluster-acl.html

QUESTION NO: 5
本番環境で構造化ストリーミング
ジョブをスケジュールする場合、クエリの失敗から自動的に回復し、コストを低く抑える構成はどれですか。
A. クラスター: 新しいジョブ クラスター;
再試行回数: 無制限
最大同時実行数: 無制限
B. クラスター: 新しいジョブ クラスター;
再試行: なし;
最大同時実行数: 1
C. クラスター: 既存の多目的クラスター。
再試行回数: 無制限
最大同時実行数: 1
D. クラスター: 既存の多目的クラスター。
再試行回数: 無制限

最大同時実行数: 1
E. クラスター: 既存の多目的クラスター。
再試行: なし;
最大同時実行数: 1
Answer: D
Explanation:
The configuration that automatically recovers from query failures and keeps costs low is to use a new job cluster, set retries to unlimited, and set maximum concurrent runs to 1. This configuration has the following advantages:
A new job cluster is a cluster that is created and terminated for each job run. This means that the cluster resources are only used when the job is running, and no idle costs are incurred. This also ensures that the cluster is always in a clean state and has the latest configuration and libraries for the job.
Setting retries to unlimited means that the job will automatically restart the query in case of any failure, such as network issues, node failures, or transient errors. This improves the reliability and availability of the streaming job, and avoids data loss or inconsistency. Setting maximum concurrent runs to 1 means that only one instance of the job can run at a time. This prevents multiple queries from competing for the same resources or writing to the same output location, which can cause performance degradation or data corruption. Therefore, this configuration is the best practice for scheduling Structured Streaming jobs for production, as it ensures that the job is resilient, efficient, and consistent.

QUESTION NO: 6
データエンジニアリングチームは、Delta Lakeテーブルの値を監視するためのDatabricks SQLクエリとアラートを設定しました。recent_sensor_recordingsテーブルには、過去5分間の記録のタイムスタンプと温度に加え、識別用のsensor_idが含まれています。
アラートを作成するには、以下のクエリを使用します。

```
SELECT MEAN(temperature), MAX(temperature), MIN(temperature)
FROM recent_sensor_recordings
GROUP BY sensor_id
```

クエリは1分ごとに更新され、常に10秒以内に完了するように設定されています。アラートは、平均気温が120度を超えた場合にトリガーされるように設定されています。通知は最大1分ごとに送信されます。
このアラートが 3
分間連続して通知を生成し、その後停止する場合、どのステートメントが正しいでしょうか。
A. クエリの3回連続実行で、すべてのセンサーの合計平均温度が120を超えました。
B. recent_sensor_recordingstable はクエリを3回連続して実行しても応答しませんでした。
C. ソースクエリは3分間連続して正しく更新されなかったため、再起動されました。
D. クエリの3回連続実行で、少なくとも1つのセンサーの最大温度記録が120を超えました。
E. クエリの3回連続実行で、少なくとも1つのセンサーの平均温度記録が120を超えました。
Answer: E
Explanation:
This is the correct answer because the query is using a GROUP BY clause on the sensor_id

column, which means it will calculate the mean temperature for each sensor separately. The alert will trigger when the mean temperature for any sensor is greater than 120, which means at least one sensor had an average temperature above 120 for three consecutive minutes. The alert will stop when the mean temperature for all sensors drops below 120.

QUESTION NO: 7
ジュニア開発者が、ノートブック内のコードが開発環境で正しい結果を生成しないと訴えています。共有されたスクリーンショットを見ると、Databricks Reposでバージョン管理されたノートブックを使用しているにもかかわらず、古いロジックを含む個人用ブランチを使用していることがわかります。目的のブランチ「dev-2.3.9」は、ブランチ選択ドロップダウンに表示されていません。
どのアプローチによって、開発者はこのノートブックの現在のロジックを確認できるでしょうか?
A. Repos を使用してプル リクエストを作成し、Databricks REST API を使用して現在のブランチを dev-2.3.9 に更新します。
B. Repos を使用してリモート Git リポジトリから変更をプルし、dev-2.3.9 ブランチを選択します。
C. Repos を使用して dev-2.3.9 ブランチをチェックアウトし、現在のブランチとの競合を自動解決します。
D. すべての変更をリモート Git リポジトリのメイン ブランチにマージし、リポジトリを再度クローンします。
E. Repos を使用して現在のブランチと dev-2.3.9 ブランチをマージし、プルリクエストを作成してリモートリポジトリと同期します。
Answer: B
Explanation:
This is the correct answer because it will allow the developer to update their local repository with the latest changes from the remote repository and switch to the desired branch. Pulling changes will not affect the current branch or create any conflicts, as it will only fetch the changes and not merge them. Selecting the dev-2.3.9 branch from the dropdown will checkout that branch and display its contents in the notebook.

QUESTION NO: 8
セキュリティ チームは、Databricks シークレット モジュールを利用して外部データベースに接続できるかどうかを調査しています。
すべてのPython変数を文字列で定義してコードをテストした後、パスワードをsecretsモジュールにアップロードし、現在アクティブなユーザーに適切な権限を設定します。その後、コードを以下のように変更します(他の変数はすべて変更しません)。

```
password = dbutils.secrets.get(scope="db_creds", key="jdbc_password")

print(password)

df = (spark
  .read
  .format("jdbc")
  .option("url", connection)
  .option("dbtable", tablename)
  .option("user", username)
  .option("password", password)
  )
```

上記のコードが実行されると何が起こるかを説明している文はどれですか?
A. 外部テーブルへの接続は失敗し、文字列「redacted」が出力されます。
B.
ノートブックに対話型入力ボックスが表示されます。正しいパスワードを入力すると、接続が成功し、エンコードされたパスワードが DBFS に保存されます。
C.
ノートブックに対話型入力ボックスが表示されます。正しいパスワードを入力すると、接続が成功し、パスワードがプレーンテキストで印刷されます。
D.
外部テーブルへの接続は成功します。パスワードの文字列値がプレーンテキストで出力されます。
E. 外部テーブルへの接続は成功し、文字列「redacted」が出力されます。
Answer: E
Explanation:
This is the correct answer because the code is using the dbutils.secrets.get method to retrieve the password from the secrets module and store it in a variable. The secrets module allows users to securely store and access sensitive information such as passwords, tokens, or API keys. The connection to the external table will succeed because the password variable will contain the actual password value. However, when printing the password variable, the string "redacted" will be displayed instead of the plain text password, as a security measure to prevent exposing sensitive information in notebooks.

QUESTION NO: 9
データサイエンスチームはMLflowを使用して本番環境モデルを作成し、ログに記録しました。以下のコードは、本番環境モデルを正しくインポートして適用し、予測結果を「customer_id LONG, predictions DOUBLE, date
DATE」というスキーマを持つpredsという名前の新しいデータフレームとして出力します。

```
from pyspark.sql.functions import current_date

model = mlflow.pyfunc.spark_udf(spark, model_uri="models:/churn/prod")
df = spark.table("customers")
columns = ["account_age", "time_since_last_seen", "app_rating"]
preds = (df.select(
  "customer_id",
  model(*columns).alias("predictions"),
  current_date().alias("date")
  )
```

データサイエンスチームは、予測結果をDelta Lakeテーブルに保存し、すべての予測を時系列で比較できるようにしたいと考えています。チャーン予測は1日に最大1回まで行われます。
潜在的な計算コストを最小限に抑えながらこのタスクを実行するコード
ブロックはどれですか。

A. preds.write.mode("append").saveAsTable("churn_preds")

B. preds.write.format("delta").save("/preds/churn_preds")

C.

```
(preds.writeStream
  .outputMode("overwrite")
  .option("checkpointPath", "/_checkpoints/churn_preds")
  .start("/preds/churn_preds")
)
```

D.

```
(preds.write
  .format("delta")
  .mode("overwrite")
  .saveAsTable("churn_preds")
)
```

E.

```
(preds.writeStream
  .outputMode("append")
  .option("checkpointPath", "/_checkpoints/churn_preds")
  .table("churn_preds")
)
```

Answer: A

QUESTION NO: 10
アップストリームソースは、Parquetデータを1時間ごとのバッチとして、現在の日付を名前

とするディレクトリに書き込みます。夜間のバッチジョブは、次のコードを実行して、日付変数で示される前日のすべてのデータを取り込みます。

```
(spark.read
  .format("parquet")
  .load(f"/mnt/raw_orders/{date}")
  .dropDuplicates(["customer_id", "order_id"])
  .write
  .mode("append")
  .saveAsTable("orders")
)
```

フィールド customer_id と order_id
は、各注文を一意に識別するための複合キーとして機能すると想定します。
上流システムが、単一の注文に対して数時間間隔を置いて重複したエントリを生成することが時々あることがわかっている場合、正しい記述はどれですか。

A. 注文テーブルへの各書き込みには一意のレコードのみが含まれ、ターゲット
テーブルに重複のないレコードのみが書き込まれます。

B.
注文テーブルへの各書き込みには一意のレコードのみが含まれますが、新しく書き込まれたレコードにはターゲット
テーブルに既に存在する重複レコードが含まれている場合があります。

C.
注文テーブルへの各書き込みには一意のレコードのみが含まれます。同じキーを持つ既存のレコードがターゲット テーブルに存在する場合、これらのレコードは上書きされます。

D.
注文テーブルへの各書き込みには一意のレコードのみが含まれます。同じキーを持つ既存のレコードがターゲット テーブルに存在する場合、操作は失敗します。

E.
注文テーブルへの各書き込みでは、新規レコードと既存レコードの結合に対して重複排除が実行され、重複レコードが存在しないことが保証されます。

Answer: B

Explanation:

This is the correct answer because the code uses the dropDuplicates method to remove any duplicate records within each batch of data before writing to the orders table. However, this method does not check for duplicates across different batches or in the target table, so it is possible that newly written records may have duplicates already present in the target table. To avoid this, a better approach would be to use Delta Lake and perform an upsert operation using mergeInto.

QUESTION NO: 11

データエンジニアリングチームのジュニアメンバーが、Databricksノートブックの言語相互運用性について調査しています。以下のコードの目的は、geo_lookupテーブルに含まれるアフリカ大陸の国々で発生したすべての売上のビューを登録することです。
コードを実行する前に、現在のデータベースで SHOW TABLES

を実行すると、データベースには geo_lookup と sales の 2 つのテーブルのみが含まれていることがわかります。

```
Cmd 1
%python
countries_af = [x[0] for x in
spark.table("geo_lookup").filter("continent='AF'").select("country").collect()]

Cmd 2
%sql
CREATE VIEW sales_af AS
  SELECT *
  FROM sales
  WHERE city IN countries_af
  AND CONTINENT = "AF"
```

インタラクティブ ノートブックでこれらのコマンド セルを順番に実行した結果を正しく説明している記述はどれですか。

A. どちらのコマンドも成功します。show tablesを実行すると、countries atとsales atがビューとして登録されていることが確認できます。

B.
コマンド1は成功します。コマンド2は、アクセス可能なすべてのデータベースでcountries afという名前のテーブルまたはビューを検索します。このエンティティが存在する場合、コマンド2は成功します。

C. Cmd 1 は成功し、Cmd 2 は失敗します。countries at は PySpark DataFrame を表す Python 変数になります。

D. どちらのコマンドも失敗します。新しい変数、テーブル、ビューは作成されません。

E. Cmd 1 は成功し、Cmd 2 は失敗します。countries at は文字列のリストを含む Python 変数になります。

Answer: E

Explanation:
This is the correct answer because Cmd 1 is written in Python and uses a list comprehension to extract the country names from the geo_lookup table and store them in a Python variable named countries af. This variable will contain a list of strings, not a PySpark DataFrame or a SQL view.
Cmd 2 is written in SQL and tries to create a view named sales af by selecting from the sales table where city is in countries af. However, this command will fail because countries af is not a valid SQL entity and cannot be used in a SQL query. To fix this, a better approach would be to use spark.sql() to execute a SQL query in Python and pass the countries af variable as a parameter.

QUESTION NO: 12
天気記録の Delta テーブルは日付ごとにパーティション分割されており、以下のスキーマを持ちます。
日付 DATE、デバイスID INT、温度 FLOAT、緯度 FLOAT、経度 FLOAT
北極圏内のすべてのレコードを検索するには、以下のフィルターを使用してクエリを実行します。

緯度 > 66.3
Delta
エンジンがロードするファイルを識別する方法について説明している記述はどれですか。
A.
すべてのレコードは運用データベースにキャッシュされ、その後フィルターが適用されます。
B.
Parquetファイルのフッターをスキャンして、緯度列の最小値と最大値の統計を取得します。
C.
すべてのレコードは接続されたストレージにキャッシュされ、その後フィルターが適用されます
D. Deltaログをスキャンして、緯度列の最小値と最大値の統計を取得します。
E. Hiveメタストアは緯度列の最小および最大統計をスキャンします。
Answer: D
Explanation:
This is the correct answer because Delta Lake uses a transaction log to store metadata about each table, including min and max statistics for each column in each data file. The Delta engine can use this information to quickly identify which files to load based on a filter condition, without scanning the entire table or the file footers. This is called data skipping and it can improve query performance significantly. Verified Reference: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; [Databricks Documentation], under "Optimizations - Data Skipping" section.
In the Transaction log, Delta Lake captures statistics for each data file of the table. These statistics indicate per file:
- Total number of records
- Minimum value in each column of the first 32 columns of the table
- Maximum value in each column of the first 32 columns of the table
- Null value counts for in each column of the first 32 columns of the table When a query with a selective filter is executed against the table, the query optimizer uses these statistics to generate the query result. it leverages them to identify data files that may contain records matching the conditional filter.
For the SELECT query in the question, The transaction log is scanned for min and max statistics for the price column.

QUESTION NO: 13
データエンジニアリングチームは、顧客からの忘れ去られる(データを削除する)リクエストを処理するジョブを設定しました。削除が必要なすべてのユーザーデータは、デフォルトのテーブル設定を使用してDelta Lakeテーブルに保存されます。
チームは、毎週日曜日の午前1時に、前週のすべての削除処理をバッチジョブとして実行することにしました。このジョブの合計所要時間は1時間未満です。毎週月曜日の午前3時には、バッチジョブが組織全体のすべてのDelta
Lakeテーブルに対して一連のVACUUMコマンドを実行します。
コンプライアンス担当者は最近、Delta
Lakeのタイムトラベル機能について知りました。これにより、削除されたデータへの継続的

なアクセスが可能になるのではないかと懸念しています。
すべての削除ロジックが正しく実装されていると仮定すると、どのステートメントがこの問題に正しく対処していますか?

A. vacuum
コマンドは削除されたレコードを含むすべてのファイルを完全に削除するため、削除されたレコードにはタイムトラベルで約 24 時間アクセスできる場合があります。

B. デフォルトのデータ保持しきい値は 24
時間であるため、削除されたレコードを含むデータ ファイルは、翌日にバキューム
ジョブが実行されるまで保持されます。

C. Delta Lake タイム トラベルではテーブルの履歴全体へのフル
アクセスが提供されるため、削除されたレコードは完全な管理者権限を持つユーザーがいつでも再作成できます。

D. Delta Lake の削除ステートメントには ACID
保証があるため、削除ジョブが完了するとすぐに、削除されたレコードはすべてのストレージ システムから完全に消去されます。

E. デフォルトのデータ保持しきい値は 7 日間であるため、削除されたレコードを含むデータ ファイルは、8 日後にバキューム ジョブが実行されるまで保持されます。

Answer: E
Explanation:
https://learn.microsoft.com/en-us/azure/databricks/delta/vacuum

QUESTION NO: 14
ジュニア データ エンジニアが、次の JSON を Databricks REST API エンドポイント 2.0/jobs/create に投稿するワークロードを構成しました。

```
{
  "name": "Ingest new data",
  "existing_cluster_id": "6015-954420-peace720",
  "notebook_task": {
    "notebook_path": "/Prod/ingest.py"
  }
}
```

すべての構成と参照リソースが利用可能であると仮定した場合、このワークロードを 3 回実行した結果を説明するステートメントはどれですか。

A. 「新しいデータの取り込み」という名前の 3
つの新しいジョブがワークスペースに定義され、それぞれ 1 日 1 回実行されます。

B. 参照されたノートブックで定義されたロジックは、指定されたクラスター ID
の構成を持つ新しいクラスターで 3 回実行されます。

C. ワークスペースに「新しいデータの取り込み」という名前の 3
つの新しいジョブが定義されますが、ジョブは実行されません。

D. 「新しいデータの取り込み」という名前の新しいジョブが 1
つワークスペースに定義されますが、実行されません。

E.

参照されているノートブックで定義されたロジックは、参照されている既存の多目的クラスターで 3 回実行されます。

Answer: C

Explanation:

Databricks jobs create will create a new job with the same name each time it is run.

In order to overwrite the extsting job you need to run databricks jobs reset

QUESTION NO: 15

上流システムが変更データキャプチャ(CDC)ログを出力し、クラウドオブジェクトストレージディレクトリに書き込んでいます。ログ内の各レコードは、変更の種類(挿入、更新、削除)と、変更後の各フィールドの値を示しています。ソーステーブルには、フィールドpk_idで識別される主キーがあります。

監査目的のため、データガバナンスチームはソースシステムで有効であったすべての値の完全な記録を維持したいと考えています。分析目的のため、各レコードの最新の値のみを記録する必要があります。これらのレコードを取り込むDatabricksジョブは1時間に1回実行されますが、各レコードは1時間の間に複数回変更されている可能性があります。

これらの要件を満たすソリューションはどれですか?

A. pk_id ごとに個別の履歴テーブルを作成し、union all
を実行して履歴テーブルをフィルタリングし、最新の状態を取得して、テーブルの現在の状態を解決します。

B. merge into を使用して、各 pk_id の最新のエントリをブロンズ
テーブルに挿入、更新、または削除し、すべての変更をシステム全体に伝播します。

C.
テーブルへの順序付けられた一連の変更を反復処理し、各変更を順番に適用します。監査ログを作成するには、Delta Lake のバージョン管理機能を利用します。

D. Delta Lake の変更データ フィードを使用して、外部システムからの CDC
データを自動的に処理し、すべての変更を Lakehouse
内のすべての依存テーブルに伝播します。

E. すべてのログ情報をブロンズ テーブルに取り込み、merge into を使用して各 pk_id
の最新のエントリをシルバー
テーブルに挿入、更新、または削除し、現在のテーブル状態を再作成します。

Answer: E

Explanation:

CDF captures changes only from a Delta table and is only forward-looking once enabled. The CDC logs are writing to object storage. So you would need to ingestion those and merge into downstream tables.

QUESTION NO: 16

1時間ごとのバッチジョブは、クラウドオブジェクトストレージコンテナからデータファイルを取り込むように構成されています。各バッチは、ソースシステムが特定の時間に生成したすべてのレコードを表します。これらのレコードをレイクハウスに処理するバッチジョブは、遅れて到着するデータが欠落しないように十分な遅延が設定されます。user_idフィールドはデータの一意のキーを表し、次のスキーマを持ちます。

user_id BIGINT、username STRING、user_utc STRING、user_region STRING、last_login BIGINT、auto_pay BOOLEAN、last_updated BIGINT

新しいレコードはすべてaccount_historyというテーブルに取り込まれ、ソースと同じスキーマ内のすべてのデータの完全な記録が保持されます。システム内の次のテーブルはaccount_currentという名前で、各一意のuser_idの最新の値を表すType
1テーブルとして実装されています。
数百万のユーザー アカウントがあり、1
時間ごとに数万のレコードが処理されると仮定すると、記述されている account_current テーブルを各時間ごとのバッチ
ジョブの一部として効率的に更新するには、どの実装を使用できますか?

A.
自動ローダーを使用して、アカウント履歴ディレクトリ内の新しいファイルをサブスクライブします。Structured Streaminq トリガー 1
回限りのジョブを構成して、新しく検出されたファイルをアカウントの現在のテーブルに一括更新します。

B. ユーザー ID
でグループ化し、最終更新の最大値をフィルタリングしたアカウント履歴テーブルに対するクエリの結果を使用して、各バッチでアカウントの現在のテーブルを上書きします。

C. 最後に更新されたフィールドと最後に処理された時間、およびユーザー ID
による最後の最大入力を使用してアカウント履歴のレコードをフィルターし、各ユーザー ID の最新の値を更新または挿入するマージ ステートメントを記述します。

D. Delta Lake のバージョン履歴を使用して、アカウント履歴の最新バージョンと 1
つ前のバージョンの違いを取得し、これらのレコードを現在のアカウントに書き込みます。

E.
最後に更新されたフィールドと最後に処理された時間を使用してアカウント履歴のレコードをフィルターし、ユーザー名の重複を排除します。各ユーザー名の最新の値を更新または挿入するためのマージ ステートメントを記述します。

Answer: C
Explanation:
This is the correct answer because it efficiently updates the account current table with only the most recent value for each user id. The code filters records in account history using the last updated field and the most recent hour processed, which means it will only process the latest batch of data. It also filters by the max last login by user id, which means it will only keep the most recent record for each user id within that batch. Then, it writes a merge statement to update or insert the most recent value for each user id into account current, which means it will perform an upsert operation based on the user id column.

QUESTION NO: 17
Lakehouse内のcustomer_churn_paramsというテーブルは、機械学習チームによる顧客離脱予測に使用されています。このテーブルには、複数の上流ソースから得られた顧客情報が含まれています。現在、データエンジニアリングチームは、上流データソースから得られた最新の有効な値でこのテーブルを毎晩上書きすることで、データを更新しています。
MLチームが使用しているチャーン予測モデルは、本番環境では比較的安定しています。チームは過去24時間以内に変更されたレコードのみに基づいて予測を行うことに注力しています。
変更されたレコードの識別を簡素化するアプローチはどれでしょうか?

A. customer_churn_params

テーブルのすべての行に解約モデルを適用しますが、予測が変更されていない行を無視して予測テーブルにアップサートを実行するロジックを実装します。
B. 完全な出力モードを使用してバッチ ジョブを構造化ストリーミング
ジョブに変換します。customer_churn_params
テーブルから読み取り、解約モデルに対して増分予測を行うように構造化ストリーミング
ジョブを構成します。
C. 新しい予測を行う前に、一意の顧客を識別するキーで、以前のモデル予測と現在の
customer_churn_params
との差を計算します。以前の予測に含まれていない顧客についてのみ予測を行います。
D. 上書きロジックを変更して、呼び出しによって入力されたフィールドを含める
データが書き込まれるときに spark.sql.functions.current_timestamp()
が呼び出されます。このフィールドを使用して、特定の日付に書き込まれたレコードを識別します。
E. 現在の上書きロジックをマージ
ステートメントに置き換えて、変更されたレコードのみを変更します。変更データ
フィードによって識別された変更されたレコードについて予測を行うロジックを記述します。
Answer: E
Explanation:
The approach that would simplify the identification of the changed records is to replace the current overwrite logic with a merge statement to modify only those records that have changed, and write logic to make predictions on the changed records identified by the change data feed.
This approach leverages the Delta Lake features of merge and change data feed, which are designed to handle upserts and track row-level changes in a Delta table. By using merge, the data engineering team can avoid overwriting the entire table every night, and only update or insert the records that have changed in the source data. By using change data feed, the ML team can easily access the change events that have occurred in the customer_churn_params table, and filter them by operation type (update or insert) and timestamp. This way, they can only make predictions on the records that have changed in the past 24 hours, and avoid re-processing the unchanged records.

QUESTION NO: 18
テーブルは次のコードで登録されます。

```
CREATE TABLE recent_orders AS (
  SELECT a.user_id, a.email, b.order_id, b.order_date
  FROM
    (SELECT user_id, email
    FROM users) a
    INNER JOIN
    (SELECT user_id, order_id, order_date
    FROM orders
    WHERE order_date >= (current_date() - 7)) b
    ON a.user_id = b.user_id
)
```

users と orders はどちらも Delta Lake テーブルです。recent_orders をクエリした結果を説明するステートメントはどれですか。

A. すべてのロジックはクエリ時に実行され、クエリの終了時にソース テーブルの有効なバージョンを結合した結果が返されます。

B. テーブルが定義されるとすべてのロジックが実行され、テーブルの結合結果が DBFS に保存されます。この保存されたデータは、テーブルがクエリされたときに返されます。

C. テーブルが定義されると結果が計算され、キャッシュされます。これらのキャッシュされた結果は、新しいレコードがソース テーブルに挿入されるたびに増分更新されます。

D. すべてのロジックはクエリ時に実行され、クエリの開始時点のソース テーブルの有効なバージョンを結合した結果が返されます。

E. 各ソース テーブルのバージョンはテーブル トランザクション ログに保存され、クエリ結果はクエリごとに DBFS に保存されます。

Answer: B

Explanation:

Table is created and data of join will be stored on DBFS and it will be returned on query time.

QUESTION NO: 19

運用ワークロードは、外部の変更データキャプチャフィードからの更新を、常時稼働の構造化ストリームジョブとして Delta Lake テーブルに段階的に適用します。このテーブルのデータが最初に移行された際に、OPTIMIZE が実行され、ほとんどのデータファイルのサイズが 1 GB に変更されました。ストリーミング運用ジョブでは、自動最適化と自動圧縮の両方がオンになっていました。最近のデータファイルのレビューでは、テーブル内の各パーティションには少なくとも 1 GB のデータが含まれており、テーブル全体のサイズは 10 TB を超えているにもかかわらず、ほとんどのデータファイルは 64 MB 未満であることがわかりました。

ファイル サイズが小さくなる理由として考えられるのは次のうちどれでしょうか?

A. Databricks は、MERGE 操作の所要時間を短縮するために、ターゲット ファイル サイズを小さくするように自動調整しました。

B. テーブルで計算されたZオーダーインデックスがファイルの圧縮を妨げています。C テーブルで計算されたブルームフィラーインデックスがファイルの圧縮を妨げています。
C. Databricks は、テーブル内のデータの全体的なサイズに基づいて、ターゲット ファイルのサイズを小さくするように自動調整しました。
D. Databricks は、各パーティションのデータ量に基づいて、ターゲット ファイル サイズを小さくするように自動調整しました。
Answer: A
Explanation:
This is the correct answer because Databricks has a feature called Auto Optimize, which automatically optimizes the layout of Delta Lake tables by coalescing small files into larger ones and sorting data within each file by a specified column. However, Auto Optimize also considers the trade- off between file size and merge performance, and may choose a smaller target file size to reduce the duration of merge operations, especially for streaming workloads that frequently update existing records. Therefore, it is possible that Auto Optimize has autotuned to a smaller target file size based on the characteristics of the streaming production job.

QUESTION NO: 20
ストリーム静的結合と静的デルタ テーブルに関する正しい記述はどれですか。
A. ストリーム静的結合の各マイクロバッチでは、各マイクロバッチの時点での静的 Delta テーブルの最新バージョンが使用されます。
B. ストリーム静的結合の各マイクロバッチは、ジョブの初期化時点での静的 Delta テーブルの最新バージョンを使用します。
C. チェックポイント
ディレクトリは、結合に存在する一意のキーの状態情報を追跡するために使用されます。
D. 一貫性の問題のため、ストリーム静的結合では静的デルタ テーブルを使用できません。
E. チェックポイント ディレクトリは、静的 Delta テーブルの更新を追跡するために使用されます。
Answer: A
Explanation:
This is the correct answer because stream-static joins are supported by Structured Streaming when one of the tables is a static Delta table. A static Delta table is a Delta table that is not updated by any concurrent writes, such as appends or merges, during the execution of a streaming query. In this case, each microbatch of a stream-static join will use the most recent version of the static Delta table as of each microbatch, which means it will reflect any changes made to the static Delta table before the start of each microbatch.

QUESTION NO: 21
ジュニアデータエンジニアの1人が、DataFrame dfを用いたグループ化された集計機能を備えたストリーミングデータパイプラインの開発を依頼されました。このパイプラインでは、重複しない5分間隔ごとに平均湿度と平均気温を計算する必要があります。イベントはデバイスごとに1分ごとに記録されます。
ストリーミング DataFrame df には次のスキーマがあります。
「device_id INT、event_time TIMESTAMP、temp FLOAT、humidity FLOAT」
コードブロック:

```
df.withWatermark("event_time", "10 minutes")
    .groupBy(
        _____,
        "device_id"
    )
    .agg(
        avg("temp").alias("avg_temp"),
        avg("humidity").alias("avg_humidity")
    )
    .writeStream
    .format("delta")
    .saveAsTable("sensor_avg")
```

このタスクを完了するには、コード
ブロック内の空白を正しく埋める応答を選択してください。

A. to_interval("event_time", "5 minutes").alias("time")

B. window("event_time", "5 minutes").alias("time")

C. "event_time"

D. window("event_time", "10 minutes").alias("time")

E. lag("event_time", "10 minutes").alias("time")

Answer: B

Explanation:
This is the correct answer because the window function is used to group streaming data by time intervals. The window function takes two arguments: a time column and a window duration. The window duration specifies how long each window is, and must be a multiple of 1 second. In this case, the window duration is "5 minutes", which means each window will cover a non-overlapping five- minute interval. The window function also returns a struct column with two fields: start and end, which represent the start and end time of each window. The alias function is used to rename the struct column as "time".

QUESTION NO: 22
データアーキテクトは、2つの構造化ストリーミングジョブが単一のブロンズDeltaテーブルに同時に書き込みを行うシステムを設計しました。各ジョブはApache Kafkaソースの異なるトピックをサブスクライブしますが、同じスキーマでデータを書き込みます。ディレクトリ構造をシンプルに保つため、データエンジニアは両方のストリームで共有されるチェックポイントディレクトリをネストすることにしました。
提案されたディレクトリ構造を以下に示します。

```
./bronze
├── _checkpoint
├── _delta_log
├── year_week=2020_01
├── year_week=2020_02
└── ...
```

このチェックポイント
ディレクトリ構造が特定のシナリオに対して有効であるかどうか、またその理由を説明している記述はどれですか。
A. いいえ。Delta Lake はトランザクション ログ内のストリーミング
チェックポイントを管理します。
B. はい。両方のストリームで単一のチェックポイント ディレクトリを共有できます。
C. いいえ。Delta Lake テーブルに書き込むことができるのは 1 つのストリームだけです。
D. はい。Delta Lake は無制限の同時書き込みをサポートします。
E. いいえ。各ストリームには独自のチェックポイント ディレクトリが必要です。
Answer: E
Explanation:
This is the correct answer because checkpointing is a critical feature of Structured Streaming that provides fault tolerance and recovery in case of failures. Checkpointing stores the current state and progress of a streaming query in a reliable storage system, such as DBFS or S3. Each streaming query must have its own checkpoint directory that is unique and exclusive to that query. If two streaming queries share the same checkpoint directory, they will interfere with each other and cause unexpected errors or data loss.

QUESTION NO: 23
本番環境にデプロイされた構造化ストリーミングジョブで、ピーク時に遅延が発生しています。現在、通常の実行時には、各マイクロバッチのデータ処理時間は3秒未満です。しかし、ピーク時には各マイクロバッチの実行時間が非常に不安定になり、30秒を超えることもあります。ストリーミング書き込みのトリガー間隔は現在10秒に設定されています。
他のすべての変数を一定に保ち、レコードを 10
秒以内に処理する必要があると仮定すると、どの調整が要件を満たすでしょうか。
A. トリガー間隔を 5
秒に短縮します。バッチをより頻繁にトリガーすると、アイドル状態の実行プログラムは、前のバッチで実行時間が長いタスクが終了している間に、次のバッチの処理を開始できます。
B. トリガー間隔を 30
秒に増やします。レコードがドロップされないようにするには、トリガー間隔を各バッチの最大実行時間に近い値に設定するのが常にベスト プラクティスです。
C. チェックポイント
ディレクトリを変更せずにトリガー間隔を変更することはできません。現在のストリーム状態を維持するには、シャッフル パーティションの数を増やして並列処理を最大化します。
D. トリガー 1 回オプションを使用し、10 秒ごとにクエリを実行するように Databricks
ジョブを構成します。これにより、バックログされたすべてのレコードが各バッチで処理されるようになります。
E. トリガー間隔を 5
秒に短縮します。バッチをより頻繁にトリガーすると、レコードのバックアップや大きなバッチによるスピルの発生を防ぐことができます。
Answer: E
Explanation:
The adjustment that will meet the requirement of processing records in less than 10 seconds

is to decrease the trigger interval to 5 seconds. This is because triggering batches more frequently may prevent records from backing up and large batches from causing spill. Spill is a phenomenon where the data in memory exceeds the available capacity and has to be written to disk, which can slow down the processing and increase the execution time. By reducing the trigger interval, the streaming query can process smaller batches of data more quickly and avoid spill. This can also improve the latency and throughput of the streaming job.

QUESTION NO: 24
Delta Lake Auto Compaction について説明している記述はどれですか?
A.
書き込みが完了した後、ファイルをさらに圧縮できるかどうかを検出するための非同期ジョブが実行されます。圧縮できる場合は、デフォルトの 1 GB
に向けて最適化ジョブが実行されます。
B. ジョブ
クラスターが終了する前に、最新のジョブ中に変更されたすべてのテーブルに対して最適化が実行されます。
C. 最適化された書き込みでは、ディレクトリ
パーティションではなく論理パーティションが使用されます。パーティション境界はメタデータでのみ表されるため、書き込まれる小さなファイルの数は少なくなります。
D. データはメモリに直接コミットされるのではなく、メッセージング
バスのキューに入れられます。ジョブが完了すると、すべてのデータはメッセージング
バスから 1 つのバッチでコミットされます。
E.
書き込みが完了した後、ファイルをさらに圧縮できるかどうかを検出するための非同期ジョブが実行されます。圧縮できる場合は、デフォルトの 128 MB
に向けて最適化ジョブが実行されます。
Answer: E
Explanation:
This is the correct answer because it describes the behavior of Delta Lake Auto Compaction, which is a feature that automatically optimizes the layout of Delta Lake tables by coalescing small files into larger ones. Auto Compaction runs as an asynchronous job after a write to a table has succeeded and checks if files within a partition can be further compacted. If yes, it runs an optimize job with a default target file size of 128 MB. Auto Compaction only compacts files that have not been compacted previously.

QUESTION NO: 25
Spark Structured Streaming で使用される一般的なプログラミング
モデルの特徴を説明する記述はどれですか。
A. 構造化ストリーミングは、GPU の並列処理を活用して、高度な並列データ
スループットを実現します。
B. 構造化ストリーミングはメッセージング バスとして実装されており、Apache Kafka
から派生しています。
C. 構造化ストリーミングは、特殊なハードウェアと I/O
ストリームを使用して、データ転送のレイテンシを 1 秒未満に抑えます。

D. 構造化ストリーミングは、データ
ストリームに到着する新しいデータを、無制限のテーブルに追加される新しい行としてモデル化します。
E.
構造化ストリーミングは、キャッシュされたステージの増分状態値を保持するノードの分散ネットワークに依存します。
Answer: D
Explanation:
The key idea in Structured Streaming is to treat a live data stream as a table that is being continuously appended. This leads to a new stream processing model that is very similar to a batch processing model. You will express your streaming computation as standard batch-like query as on a static table, and Spark runs it as an incremental query on the unbounded input table. Let's understand this model in more detail.
https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html