IT資格道場

CompTIA Data+ 学習用テキスト

予備知識ゼロから合格ラインへ —— 読んで理解する1冊目

無料サンプル(第1章まで)

CompTIA Data+ (DA0-002) Exam Objectives Version 2.0 準拠
2026年11月版

SAMPLE

ご利用にあたって

- このテキストは「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルも著作物です。転載・改変しての再配布・二次販売はできません。
- AI モデルの学習データとしてのご利用(学習目的の入力・データセット化を含む)はできません。
- 製品版(学習用テキスト・頻出度別ノート・暗記用ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

SAMPLE

目次

この教材の狙いと Data+ の性格

第1章 Data Concepts and Environments [1.1/1.2/1.3/1.4/1.5]

- この章の配点と問われ方
- データベースの型とファイル拡張子 [1.1]
- データの構造 [1.1]
- データ型 [1.1]
- データソースの種類 [1.2]
- データ置き場: レイク・レイクハウス・マート・サイロ・ウェアハウス [1.2]
- クラウドとオンプレミス [1.3]
- ストレージの種類 [1.3]
- コンテナ化 [1.3]
- 分析の道具 [1.4]
- AI の概念 [1.5]

サンプルはここまでです

SAMPLE

3 冊の使い方は、ダウンロード欄の案内を参照してください。

SAMPLE

この教材の狙いと Data+ の性格

このテキストの目的は、データ分析の入門資格 **CompTIA Data+** の合格です。試験コードは DA0-002 です。

項目	内容
問題数	最大 90 問
時間	90 分
合格点	675（100〜900）
前提資格	なし
配点	第 1 章 20%・第 2 章 22%・第 3 章 24%・第 4 章 20%・第 5 章 14%
本テキストの章	第 1〜5 章

章は公式の出題範囲の 5 分野に対応します。

読み方の目印は次の 3 つ。

- 出題の形：択一・複数選択・対応付け・並べ替え・パフォーマンスベース問題（PBQ）
- 章の冒頭：その分野の配点（重い分野から固めると点が伸びる）
- 見出しの末尾：公式の出題範囲の細目の番号（例：2.1）

SAMPLE

第1章 Data Concepts and Environments [1.1/1.2/1.3/1.4/1.5]

この章の配点と問われ方

この章は DA0-002 の配点の 20% です。範囲は 5 つの細目です。データの種類と型（1.1）、データの入手元と置き場（1.2）、クラウドとストレージとコンテナ（1.3）、分析の道具（1.4）、AI の用語（1.5）。

問われ方は「用語を見て、場面に当てはめる」形が中心です。たとえば「列に NULL が混じる売上データの平均を取る」「履歴を残したい次元表の更新方式を選ぶ」「画像と動画の置き場を選ぶ」。定義を暗記するだけでなく、隣の用語との違いを 1 文で言えると解けます。

データベースの型とファイル拡張子 [1.1]

リレーショナルと非リレーショナル [1.1]

リレーショナルデータベース（Relational）は、データを表（行と列）で持ちます。行が 1 件の記録で、列が属性です。表どうしは「キー」でつなぎます。AWS は、主キー・外部キー・NOT NULL・UNIQUE・DEFAULT・CHECK の制約でデータの整合性を守ると説明しています[1]。外部キーの制約は、列の値を、別の表のある行にある値に限ります（注文表の顧客 ID は、顧客表にある ID だけ）[2]。問い合わせには SQL を使います。

非リレーショナル（Non-relational、NoSQL とも呼ぶ）は、表以外の形でデータを持つデータベースの総称です。AWS は、特定のデータモデル向けに作られ、柔軟なスキーマで持つと説明し、キーバリュー型・ドキュメント型・グラフ型などを挙げています[3]。MongoDB は、データを BSON ドキュメント（フィールド

SAMPLE

と値の組）で持ち、コード上のオブジェクトの形をそのまま保存できると書いています[4]。

選ぶ基準は 3 つあります。

- 列が決まっていて、表をまたいだ集計や結合が多い: リレーショナル
- 項目が記録ごとに違う、入れ子が深い、構造が変わりやすい: 非リレーショナル（ドキュメント型）
- 関係そのものをたどる（人と人のつながりなど）: 非リレーショナル（グラフ型）

間違えやすいのは「非リレーショナル＝構造が無い」という読みです。ドキュメント型にもフィールドの形があります。決まった形を前もって強制しないだけです（形が無いわけではありません）。

ファイル拡張子 [1.1]

試験が挙げる 6 つの拡張子を、中身の作りで区別します。

拡張子	中身	構造の分類	読み方の要点
.csv	カンマ区切りのテキスト	構造化	1 行 1 レコード。先頭の見出し行は任意
.xlsx	Excel のブック	構造化（表）が多いが、書式・数式・複数シートも持つ	ZIP に束ねた XML
.json	キーと値の入れ子のテキスト	半構造化	配列とオブジェクトで入れ子にできる
.txt	書式を持たない文字だけのテキスト	非構造化	列の意味はファイルの外で決める

SAMPLE

拡張子	中身	構造の分類	読み方の要点
.jpg	JPEG 形式の画像	非構造化	画素のデータ。列や項目は無い
.dat	汎用の拡張子	中身次第	拡張子だけでは形式が決まらない

.csv の仕様は RFC 4180 にあります。1 行が 1 レコードで、行は改行（CRLF）で区切り、先頭に見出し行を置いてもよく、項目は二重引用符で囲んでも囲まなくてもかまいません。改行・二重引用符・カンマを含む項目は二重引用符で囲む決まりです[5]。実際に動かすと次のとおりです。

```
import csv, io
raw = 'id,name,note\r\n1,"Smith, J","said ""hi"""\r\n2,Lee,\r\n'
print(list(csv.reader(io.StringIO(raw, newline=""))))
# [['id','name','note'], ['1','Smith, J','said "hi"'], ['2','Lee','']]
```

"Smith, J" のカンマは引用符の中なので区切りになりません。`""` は引用符 1 個を表します。末尾の空の項目は欠損です。値の型の情報は csv に入っていないので、読み込む側が推定します。郵便番号 01234 を整数として読むと 1234 になり、先頭の 0 が消えます。pandas では引数 `dtype={"zip": str}` で文字列に固定できます（01234 のまま残ります）。

.xlsx は、Microsoft が OOXML（Office Open XML）と呼ぶ形式で、ZIP の中に XML を束ねています。拡張子を .zip に変えると中の XML が見えます。ワークシートごとに別の XML ができます[6]。csv と違い、1 つのファイルに複数のシート、数式、書式を持てます。

SAMPLE

.json は RFC 8259 の形式です。構造文字は 6 つ（`[ ] { } : ,`）で、値は文字列・数値・オブジェクト・配列と、リテラルの true・false・null です。リテラルは小文字で書く決まりで、True や NULL は誤りです[7]。.txt は書式を持たないので、取り込む側が区切りを決めます。.jpg は JPEG（ISO/IEC 10918 と ITU-T T.81）で圧縮した写真向けの画像形式で、国際標準です[8]。.dat は「データ」という意味しか持たない拡張子です。同じ .dat でも、テキストのこともバイナリのこともあります。問題で .dat が出たら、「このファイルは構造化と断定できない」と読みます。

取り違えやすい組は次のとおりです。

- .csv: 表の形だが、型の情報を持たない
- .xlsx: 複数のシートと数式を持てる（.csv は 1 つの表）
- .json: 項目名つきで入れ子を持てる（集計の前に表へ展開する）
- .txt と .jpg: どちらも非構造化

データの構造 [1.1]

構造化・半構造化・非構造化 [1.1]

構造化データ（Structured）は、決まった形（スキーマ）に従います。AWS は、構造化データは厳密な形式、つまり事前に決めたデータモデルかスキーマに従い、非構造化データはスキーマに収まらないと説明しています[9]。表、リレーショナルデータベースがこれです。

半構造化データ（Semi-structured）は、形の決まりはゆるいが、項目名などの手がかりを持つデータです。JSON が代表で、記録ごとに項目が違っても読めます。AWS は、動画の保存庫に日付・場所・話題の構造化タグが付く例を挙げ、構造化と非構造化の混じりを半構造化と呼んでいます。

SAMPLE

非構造化データ（Unstructured）は、決まった属性を持たず、検索や整理が難しいデータです（画像・音声・動画・自由記述の文章）。分析するには前処理が要ります。

3 つの区別は「列で引けるか」で付けます。列で引ければ構造化、項目名で辿れれば半構造化、どちらも無ければ非構造化です。

表の種類とスキーマ [1.1]

表（Table）の中に、分析の設計で使う 2 つの種類があります。Microsoft の資料の星形設計の説明に沿って見ます[10]。

ファクト表（Fact table）は、出来事や観測を持ちます。売上の注文、在庫の残高、為替、気温などです。持つ列は 2 種類で、次元表を指す次元キーの列と、数値の測定値（measure）の列です。行数が多く、時間とともに増え続けます。ファクト表の粒度（granularity）は、次元キーの値の組で決まります（月と商品のキーなら、月と商品の粒度）。

ディメンション表（Dimensional table、次元表）は、モデル化する対象（商品・人・場所・日付など）の一覧です[10]。一意のキー列と、絞り込みや集計の単位になる列を持ちます。行数は比較的少なめで、日付の次元表が最もよく見る表です。

2 つの表の関係は「1 対多」で、「1」の側がいつも次元表、「多」の側がファクト表です[10]。この形がスター スキーマで、ファクト表を中心に次元表が星形に並びます。

次元の属性が変わった時の扱いが、緩やかに変化するディメンション（Slowly changing dimension、SCD）です。Microsoft の資料は、星形の設計理論に 2 つの代表型、Type 1 と Type 2 があると説明しています[10]。

SAMPLE

- Type 1: 常に最新の値だけを持つ。元のデータが変わると次元表の値を上書きする。履歴は残らない。顧客のメールアドレスや電話番号のような補助の値に向く。
- Type 2: 次元のメンバーの版（履歴）を持つ。元の仕組みが版を持たない時は、データウェアハウスへの読み込み処理が変化を見つけて版を足す。版ごとの一意の識別に代理キー（surrogate key）を使い、有効期間の開始日・終了日と、現在の版を示すフラグ列を付けることが多い。

代理キーは、星形のために表へ足す一意の識別子で、元のデータには無い列です。業務上のキー（社員番号など）は、版が複数あると一意でなくなります。それで代理キーを足します。

営業担当 E01 が 2019-06-27 に東地域から西地域へ異動しました。地域別の売上を「注文時点の地域」で集計したい場合は Type 2 が要ります（小さな表で動かした結果が次です）。

```
-- dim_sales: sales_key, emp_id, region, start_date, end_date,
is_current
-- (1,'E01','East','2018-01-01','2019-06-26',0)
-- (2,'E01','West','2019-06-27','9999-12-31',1)
-- fact_sales: 注文 1 (2019-03-01, E01, 100), 注文 2 (2019-07-15, E01,
200), 注文 4 (2020-01-10, E01, 300)
SELECT d.region, SUM(f.amount)
FROM fact_sales f JOIN dim_sales d
  ON d.emp_id = f.emp_id AND f.order_date BETWEEN d.start_date AND
d.end_date
GROUP BY d.region;
-- East 100 / West 500
```

SAMPLE

注文日が有効期間に入る版へ結びつくので、過去の売上は東に残ります。同じ表を Type 1 で持つと、E01 の全件 600 が西に付きます。「現在の版だけ」（is_current = 1）へ結ぶ書き方でも同じ誤りになります。履歴を残して集計するなら、結合は「キーの一致」に加えて「日付が期間に入る」条件まで要ります。値が頻繁に変わる属性（株価など）は、次元表に持たず、ファクト表の測定値に持つのが定石です[10]。

ブリッジ表（Bridge table）は、多対多の関係を持つ表です。Microsoft の資料は、次元どうしの多対多を、測定列を持たない事実のないファクト表（factless fact）で持つと説明しています[10]。これがブリッジ表です。営業担当が複数の地域を担当できる場合、営業担当キーと地域キーの 2 列の表を作ります。

```
-- bridge_sales_region: sales_key, region_key  (3,1) (3,2) (2,2)
-- 営業担当 3 は東と西の両方、 営業担当 2 は西だけ
```

スキーマ（Schema）は、表や列の名前、型、関係といったデータの構造の取り決めです。AWS は、スキーマを、データの形式・関係・構造を決める事前の設計図と説明しています[11]。「スキーマを先に決めてから入れる」のがウェアハウス型、「入れてから読む時に決める」のがレイク型の考え方で、後の節で再び出ます。

入れ子の構造 [1.1]

JSON は、オブジェクト（`{}` で囲む名前と値の集まり）と配列（`[]` で囲む値の並び）を入れ子にできます。値の中にオブジェクトや配列を置けるので、構造が何段にも深くなります（Nested structures）。

SAMPLE

```
import json
d = json.loads('{"order_id": 1001, "customer": {"name": "佐藤", "tags":
["vip","new"]},'
               ' "items": [{"sku":"A1","qty":2},{"sku":"B2","qty":1}],
"note": null, "paid": true}')
d["customer"]["tags"][0]   # 'vip'  オブジェクト → 配列 → 先頭
d["items"][1]["qty"]       # 1      配列 → 2 番目 → オブジェクト →
qty
d["note"], d["paid"]       # None, True (null と true)
```

pandas の `json_normalize` に `record_path="items"` と `meta=["order_id"]` を渡します。items の 2 件が 1 件ずつの行になり、order_id が各行に付きます（入れ子の展開）。表に展開する作業は、第 2 章の「展開（Exploding）」と「解析（Parsing）」で再び出ます。

構造の分類で取り違えやすい組は次のとおりです。

- ファクト表と次元表: 測定値を持つ側がファクト、絞り込みの軸を持つ側が次元
- Type 1 と Type 2: 上書きなら Type 1、版を足すなら Type 2
- 次元表とブリッジ表: 多対多をつなぐ表がブリッジ
- 半構造化と非構造化: 項目名で辿れるかどうか

データ型 [1.1]

列には型（Data types）を決めます。型は、入れられる値の種類、使う記憶の大きさ、できる計算を決める約束です（名前と細かい仕様は製品で少し違いま

SAMPLE

す）。以下の値は、PostgreSQL の現行版の文書と Microsoft の資料で確かめた物です。

文字列型: char・varchar・nvarchar [1.1]

char(n) は固定長の文字列です。PostgreSQL は、char(n) を指定の幅 n まで空白で埋めて格納し、比較では末尾の空白を無視すると説明しています[12]。n を省いた char は char(1) です。varchar(n) は可変長で、n は上限です。PostgreSQL の n は文字数で、バイト数ではありません。上限より長い文字列を入れると誤りになります[12]。

nvarchar と nchar の n は、SQL Server では「バイトのペアの数」で、1〜4000 です。nchar が固定長、nvarchar が可変長で、どちらも Unicode の文字を持ちます。記憶の大きさは、nchar が n の 2 倍のバイト、nvarchar が n の 2 倍のバイトに 2 バイトを足した値です。補助文字を扱える照合順序なら UTF-16 で、そうでなければ UCS-2 です。nvarchar(max) の上限は 2^31-1 文字（2 GB）です[13]。SQL Server の char・varchar では n がバイト数で 1〜8000 です[14]。「n は文字数かバイト数か」は製品で違う、と覚えます。

選び方は次のとおりです。

- 長さが決まっているコード（都道府県コード 2 桁、国コード 2 文字）: char
- 長さがばらつく氏名・住所: varchar
- 日本語・多言語の文字を確実に扱いたい列: nvarchar

氏名のような列に char(50) を使うと、末尾が空白で埋まります。比較で末尾の空白を無視するかどうかは製品で違うので、他の製品や他の型の列と比べる時に、食い違いの元になります。

SAMPLE

欠損値: NULL [1.1]

NULL は「値が無い・分からない」を表す特別な値で、0 でも空文字でもありません。小さな表で動かして確かめました。

```
-- t: (1,80,'a') (2,NULL,'b') (3,0,'') (4,60,NULL)   列は id, score, memo
SELECT COUNT(*), COUNT(score), COUNT(memo) FROM t;   -- 4, 3, 3
SELECT AVG(score), SUM(score) FROM t;                -- 46.67, 140
SELECT AVG(COALESCE(score,0)) FROM t;                -- 35.0
SELECT id FROM t WHERE score = NULL;                 -- 0 件
SELECT id FROM t WHERE score IS NULL;                -- 2
SELECT id FROM t WHERE score <> 80;                  -- 3, 4 (id 2 は入らない)
SELECT id FROM t WHERE memo = '';                    -- 3 (NULL の id 4 は入らない)
```

読み取れる規則は 4 つです。COUNT(列) は NULL を数えず、COUNT(*) は行を数えます。AVG と SUM は NULL を除いて計算するので、score の平均は (80+0+60)/3 で 46.67 です。NULL を 0 として扱うと平均は 35.0 に下がります。どちらが正しいかは業務の意味で決めます（未回答なら除外、0 点なら 0）。NULL との比較は `=` ではなく `IS NULL` で書きます。`<>` の結果にも NULL の行は入りません。

空間型 [1.1]

空間型（Spatial）は、位置と形（点・道路・国境などの図形）を持つ型です。SQL Server は 2 つの型を持ちます。geometry は平面のユークリッド座標、

SAMPLE

geography は地球を丸いものとして扱う座標です[15]。地図に載せる店舗の位置や配送エリアの範囲が使い道です。

真偽型と数値型 [1.1]

真偽型（Boolean）は true・false を持ちます。PostgreSQL では 1 バイトで、不明の状態は NULL で表します[16]。

整数（Integer）は小数部の無い数です。PostgreSQL の integer は 4 バイトで、範囲は -2147483648 〜 +2147483647 です。smallint は 2 バイトで -32768 〜 +32767 です。bigint は 8 バイトで -9223372036854775808 〜 +9223372036854775807 です[17]。件数のように増え続ける番号が 21 億を超えるなら bigint にします。

小数は 2 系統あります。decimal（numeric）は、利用者が精度を決める正確な型で、金額のように厳密さが要る量に向きます。PostgreSQL は、整数部 131072 桁、小数部 16383 桁まで持てると書いています[17]。float は近似で持つ浮動小数点です。real は 4 バイトで有効 6 桁、double precision は 8 バイトで有効 15 桁で、どちらも「不正確（inexact）」と明記されています[17]。

```
from decimal import Decimal
0.1 + 0.2                                       # 0.30000000000000004
0.1 + 0.2 == 0.3                                # False
Decimal("0.1") + Decimal("0.2") == Decimal("0.3")   # True
```

float で金額を足し合わせると、ごく小さな誤差が積もり、合計の突合で 1 円ずれることがあります。金額・税額・残高は decimal、計測値（気温・重さ）や統計量は float、という使い分けです（float どうしの `=` の比較も避けます）。

SAMPLE

日時型 [1.1]

日時（Datetime）の代表が timestamp（日付と時刻の組）です。PostgreSQL では、timestamp と書くと timestamp without time zone と同じ型になります。timestamptz は timestamp with time zone の略です[18]。どちらも 8 バイトです。with time zone の型は、入力に付いた時間帯を UTC に直して内部に UTC で持ち、元の時間帯は保存しません[18]。

```
from datetime import datetime, timezone, timedelta
t = datetime(2026, 11, 20, 9, 30, tzinfo=timezone(timedelta(hours=9)))
t.isoformat()                                  # 2026-11-20T09:30:00+09:00
t.astimezone(timezone.utc).isoformat()     # 2026-11-20T00:30:00+00:00
```

複数の国から集まるログや注文は、時間帯付きで持つか、UTC にそろえて持ちます。時間帯なしの timestamp に東京の 9:30 とロンドンの 9:30 を混ぜると、同じ時刻に見えます。

大きなデータ: BLOB・CLOB [1.1]

BLOB（Binary Large Object）は、画像・音声・動画・PDF のようなバイナリの大きなデータを 1 つの値として持つ型です。CLOB（Character Large Object）は、長い文章のような文字の大きなデータを持つ型です。Oracle は、LOB は音声・動画・文書などを持ち、半構造化・非構造化データに向くと説明しています[19]。PostgreSQL の bytea はバイナリ文字列（バイト列）の型で、値 0 のバイトなど、文字列型が受け付けないバイトも持てます[20]。SQL Server の varchar(max) は上限が 2^31-1 バイト（2 GB）です[14]。varbinary(max) も 2^31-1 バイトです[21]。

SAMPLE

BLOB と CLOB の違いは中身の種類で、バイトの列か、文字の列かです。画像などの本体はオブジェクトストレージ（1.3 で扱う）に置き、表には場所の文字列だけを持つ設計も多く使われます。

GUID と UUID [1.1]

GUID（Globally Unique Identifier）と UUID（Universally Unique Identifier）は、同じ物の別名です。RFC 9562 は、UUID を、128 ビットで、中央の登録機関が無くても一意になる識別子と説明し、GUID とも呼ぶと書いています[22]。大きさは 16 オクテットで、16 進数を 8-4-4-4-12 の形に並べた 36 文字で書きます。バージョン 4 は乱数から作り、バージョン 7 は時刻の並びを持ちます。SQL Server の uniqueidentifier は 16 バイトの GUID で、NEWID または NEWSEQUENTIALID で作ります[23]。

連番（1, 2, 3...）と違い、複数の拠点や仕組みが別々に番号を作っても重なりません。複数のシステムのデータを統合する時の主キーに向きます。代わりに、整数より大きく、人が読んだり打ったりするには向きません。

型で取り違えやすい組は次のとおりです。

- char と varchar: 固定長か可変長か
- varchar と nvarchar: 多言語の文字（Unicode）を持てるのは nvarchar
- decimal と float: 正確か、近似か
- timestamp と timestamptz: 時間帯の扱い
- NULL と 0・空文字: それぞれ別の値
- BLOB と CLOB: バイトの列か、文字の列か
- GUID と連番: 重ならないが、長くて読みにくい

SAMPLE

データソースの種類 [1.2]

データソース（Data sources）は、分析に使うデータの入手元です。試験が挙げる入手元は 5 つで、データベース、API、Web サイトのデータ、ファイル、ログです。

データベース・API・Web サイトのデータ [1.2]

データベース（Databases）は、取引や業務の記録を持つ仕組みです。注文システムの注文表や、顧客管理の顧客表がこれです。分析側は SQL で必要な行と列を取り出します（SELECT 文）。業務の仕組みが書き込み中なので、重い問い合わせは業務を遅くすることがあります。そのため、分析用に写しを持つ置き場（次の項）を使います。

API（Application Programming Interface）は、ソフトウェアどうしがデータや機能をやり取りする窓口です。AWS は、API を、2 つのソフトウェアが決まった規則で互いに話すための仕組みと説明しています[24]。分析では、天気・為替・SNS・社内システムなどの外部のデータを、API の呼び出しで取ります。取り込みは、決まった間隔で API を呼ぶ形にします。

Web サイトのデータ（Website data）は、サイトの利用の記録や、サイトの上に載った情報です。Google アナリティクスは、各ページに小さな JavaScript の計測コードを足します。訪問者がページでどう動いたかを仮名の情報で集め、レポートにします[25]。どのページが見られたか、購入確認のページまで届いたかが分かります。

ファイルとログ [1.2]

ファイル（Files）は、csv・xlsx・json・txt など、1.1 で見た形式のファイルです。部門が手で作った表や、他社から受け取るデータはファイルで届くことが多く、列の並びや型が回ごとに変わりやすい入手元です。

SAMPLE

ログ（Logs）は、システムや機器が自動で書く出来事の記録です。NIST は、ログを「組織のシステムとネットワークで起きた出来事の記録」と定義しています[26]。Web サーバーのアクセス記録、アプリの操作の履歴、機器の稼働記録などで、1 行が 1 件の出来事です。項目に分ける前の行は文字の並びなので、半構造化または非構造化として扱います（取り込みの前に項目へ分ける作業が要ります）。

入手元は、更新の速さと信頼性で選び、基本の対応は次の 4 つです。

- 業務の確定した数字: データベース
- 外部の最新値: API
- 行動の記録: サイトの計測かログ
- 手元の資料: ファイル

データ置き場: レイク・レイクハウス・マート・サイロ・ウェアハウス [1.2]

データの置き場（Data repositories）は 5 つあります。違いは「何を入れるか」「いつ形を決めるか」「誰が使うか」です。

置き場	入れる物	形（スキーマ）を決める時	使い手・目的
データウェアハウス（Data warehouses）	取引の仕組みなどから集めた構造化データ	入れる前に設計する	分析・レポート・BI
データマート（Data marts）	1 部門に絞った、ウェアハウスの小さな一部	入れる前に設計する	部門の分析

SAMPLE

置き場	入れる物	形(スキーマ)を決める時	使い手・目的
データレイク(Data lakes)	構造化・半構造化・非構造化をそのまま	入れた後、読む時に決める	幅広い分析・機械学習
データレイクハウス(Data lakehouses)	レイクの生データと、ウェアハウスの構造・管理	両方	1 か所で BI も AI も
データサイロ(Data silos)	1 部門・1 システムに閉じて孤立したデータ	部門ごとにばらばら	置き場の種類ではなく問題の状態

データウェアハウスについて、AWS は、分析して意思決定に使う情報の中央の置き場で、取引の仕組みなどから定期的にデータが流れ込むと説明しています[27]。データは表と列で整理され、スキーマは入れる前に設計します。構造化データしか入れられません[28]。

データマートは、AWS が、組織の 1 つの事業部門に特化した情報を持つ保存の仕組みで、倉庫が持つデータの小さな選んだ部分を持つと説明しています[29]。マーケティング部門向けに、SNS の評判と顧客の記録だけを持つ、といった作りです。部門が自分のデータをすぐ分析できます。

データレイクは、AWS の説明では、構造化も非構造化も、あらゆる規模で入れられる中央の置き場です。データを「そのまま」置き、先に構造を決めなくてよい点が特徴です[30]。ウェブサーバーのログ、クリック記録、SNS、センサーのデータなど、形のばらばらな物が入ります。「先に入れて、後で処理する」のがウェアハウスとの違いです。

SAMPLE

データレイクハウスは、AWS の説明では、ウェアハウスとレイクを合わせた統一の構造です。生のデータをさまざまな形式で 1 か所に持ちながら、構造化・管理・レポートの機能を提供します[31]。レイクだけだと管理が弱く、ウェアハウスだけだと生の非構造化データを扱えない弱点を、両方の長所で補う設計です。

データサイロは、IBM が、部門・システム・場所に閉じ込められたデータの孤立した集まりで、システムとチームの間の共有を妨げると説明しています[32]。置き場の設計の名前ではなく、避けたい状態です（部門ごとに別の顧客表を持ち、顧客の数が部門で合わなくなるのが典型）。

経営会議用の月次売上レポートが、部門ごとに数字が違うと言われた場合、原因はデータサイロです。対策は、共通のウェアハウスに集めて定義をそろえることです。画像・ログ・売上表を機械学習のために 1 か所へそのまま集めるなら、レイクです（集めたあとで BI の部門別の画面も作るなら、レイクハウス）。営業部だけが使う分析なら、ウェアハウスから営業の分だけを切り出したマートが軽く済みます。

置き場の選び方を、言葉で対応させると次のとおりです。

- 部門に絞った分析: マート
- 生のまま何でも入れる: レイク
- レイクとウェアハウスの両方の性格: レイクハウス
- 部門ごとに孤立した状態: サイロ（避けたい状態）

SAMPLE

クラウドとオンプレミス [1.3]

クラウドの事業者 [1.3]

試験が挙げる事業者は、AWS（Amazon Web Services）、Azure（Microsoft）、Google（Google Cloud）の 3 つです。

S3 は、AWS が、スケーラビリティ・可用性・安全性・性能に優れたオブジェクトストレージと説明するサービスです。データレイクなどの用途が挙がっています[33]。Azure Blob Storage は、Microsoft のクラウド向けオブジェクトストレージです。大量の非構造化データ（テキストやバイナリ）向けです[34]。Cloud Storage は、Google の非構造化データの保存の仕組みです[35]。Redshift は AWS のクラウドのデータウェアハウスです[36]。BigQuery は Google の完全マネージドのデータ基盤です[37]。Power BI は Microsoft の BI、Looker は Google Cloud の BI です（1.4 で扱う）。試験では、サービス名から「どの事業者か」と「何の種類か」を言えるかが問われます。

種類	AWS	Azure	Google Cloud
オブジェクトストレージ	Amazon S3	Azure Blob Storage	Cloud Storage
データウェアハウス・分析基盤	Amazon Redshift	Microsoft Fabric の Data Warehouse	BigQuery

プライベート・パブリック・ハイブリッド [1.3]

NIST の SP 800-145 は、配置のモデルを次のように定義しています[38]。

- プライベート（Private）クラウド: 1 つの組織（複数の事業部を含む）が専用で使う基盤。所有・運用は組織自身でも第三者でもよく、場所も施設の

SAMPLE

中でも外でもよい。

- パブリック（Public）クラウド: 一般の人が使えるよう提供される基盤。事業者の施設にある。AWS・Azure・Google Cloud がこれ。
- ハイブリッド（Hybrid）クラウド: 2 つ以上の別のクラウド（プライベート・パブリックなど）を、別のまま、標準または独自の技術で結んだ物。

オンプレミス（On-premises）は、自社の施設にサーバーを置いて運用する形で、機器の購入と保守は自社が持ちます。

選び方は、データの置き場所の制約から決めます。

- 法令や社内規定で、他の組織と共有する基盤に置けないデータ: オンプレミスかプライベート
- 増減が大きい分析の負荷: パブリック
- 機密データは社内、分析の計算はクラウド: ハイブリッド

ストレージの種類 [1.3]

ストレージ（Storage types）は、データの持ち方で 3 種類、置き場所と共有の形でさらに 2 種類に分けます。

オブジェクト（Object）ストレージは、データを非構造の「オブジェクト」として持ち、各オブジェクトに一意の識別子と、中身を説明するメタデータを付けます。AWS は、写真なら撮影者・解像度・形式・作成時刻をメタデータに持つ例を挙げています[39]。階層の無い置き場に大量に置けるので、画像・動画・ログ・バックアップ、データレイクに向きます。

ファイル（File）ストレージは、フォルダとファイルの階層で持ちます。AWS は、ファイルストレージにはファイルシステム（NFS や SMB）の層が 1 つ加わると説明しています[39]。複数の利用者が同じファイルを共有する用途に向きます。

SAMPLE

ブロック（Block）ストレージは、データを同じ大きさのブロックに分けて持ち、ブロックごとに識別子を付けて読み書きします。ファイルの種類や所有者などの上位のメタデータは持ちません。データへのより直接の道を持つので速く、データベース、ブートディスク、コンテナの保存に向きます[39]。AWS の EBS は、EC2 に取り付け、手元のハードディスクのように使えるブロックストレージです[40]。

ローカル（Local）ストレージは、1 台のコンピューターに直接つながった保存先です（内蔵ディスク、直付けの SSD）。共有（Shared）ストレージは、複数の利用者や機器が同じデータへ同時に届く保存先です。AWS は、NAS（Network-attached storage）を、ファイル専用の保存機器と説明しています。ネットワーク越しにデータを常に使える状態にし、共同作業を助けます[41]。AWS の EFS は、AWS の計算サービスから使う共有ファイルストレージです[42]。

種類	単位	向く用途	見分けの言葉
オブジェクト	オブジェクト（識別子＋メタデータ）	画像・動画・ログ・レイク・バックアップ	S3、Blob、大量の非構造化
ファイル	ファイル（階層）	部署の共有フォルダ、共同編集	NFS、SMB、NAS
ブロック	同サイズのブロック	データベース、OS のディスク	速い、直接、EBS
ローカル	1 台に直付け	1 台の作業用	共有できない
共有	複数から同時	共同作業	複数の利用者・機器

大量の顧客の問い合わせ録音（音声）を、安く、容量を気にせず溜めて、のちに分析するなら、オブジェクトです。営業部の全員が同じ見積もりのひな形ファイルを編集するなら、ファイル（共有）です。取引データベースの速い読み書き

SAMPLE

には、ブロックです。軸が違うので、「ブロックでありローカル」「ファイルであり共有」のように、複数の言葉が同じ保存先に当てはまることもあります。

コンテナ化 [1.3]

コンテナ化（Containerization）は、アプリのコードと、動くのに要るファイルやライブラリを 1 つに束ね、どの基盤でも同じように動かす配備の方法です[43]。分析では、「自分の PC では動くのに、サーバーでは動かない」を防ぐために使います。Python や R のバージョンとパッケージを束ねた入れ物を、別の担当者やクラウドで同じに動かせます。

Docker の文書は、コンテナを、隔離して動く普通のプロセスと説明しています。コンテナは自分のファイルシステムと、自分から見えるプロセスとネットワークを持ち、CPU やメモリの上限も設定できます[44]。仮想マシン（VM）は、丸ごとの OS を起動します。コンテナは同じ機械の OS の核（カーネル）を共有し、OS を起動しないので、約 1 秒で始まります[44]。1 台の機械に多数を載せられるのも同じ理由です。

違いを 1 文にすると、VM は「OS ごと隔離」、コンテナは「アプリごと隔離して OS の核は共有」です。AWS は、コンテナごとに隔離された空間で動くので、1 つが故障しても他へ影響しないと説明しています[43]。

分析の道具 [1.4]

試験が挙げる道具は、コーディング環境、BI ソフト、パッケージとライブラリ、プログラミング言語、DBMS の管理ツールの 5 分類です。名前から分類と用途を言えることが基本で、同じ分類の中の隣の道具との違いが難所になります。

SAMPLE

コーディング環境: IDE・テキストエディター・ノートブック [1.4]

IDE（Integrated Development Environment、統合開発環境）は、コードの編集に加え、実行・デバッグ・変数や履歴の確認などを 1 つの画面にまとめた道具です。試験は RStudio と Visual Studio（VS）Code を IDE に挙げています[45]。Posit は、RStudio を、コンソール・直接実行できる色付きのエディター・作図・履歴・デバッグ・作業環境の管理を持つ IDE と説明しています。R と Python の両方で使え、デスクトップでもブラウザー経由でも動かせます[46]。Microsoft は、VS Code を、ソースコードエディターの手軽さに、開発者向けの機能を合わせた物と説明しています。コード補完（IntelliSense）やデバッグがその例です[47]。

テキストエディター（Text editor）は、文字を書き換える軽い道具で、csv や設定ファイルを開いて直すのに使います。コードの色付けがあっても、実行やデバッグは持たないか、少ないのが IDE との違いです。

ノートブック（Notebooks）は、コード・実行結果・文章・グラフを 1 つの文書に並べる環境で、代表が Jupyter Notebook です[48]。Jupyter の文書は、ブラウザー上で、文章（Markdown）、数式、コードを書いて動かし、結果を同じ場所に残すことを挙げています[48]。コードは「カーネル」が実行し、セル単位で動かせるので、データの確認、試行、説明つきの報告に向きます[48]。カーネルはブラウザーのウィンドウを閉じても動き続け、同じノートブックを開き直すと、同じカーネルにつながります[48]。Jupyter の文書は、コードの開発・記録・実行から結果の伝達までの、計算の過程全体を残す用途を挙げています[48]。

コードを何枚も書いて部品化し、動作確認を重ねるのは IDE です。設定ファイルをちょっと直すのはテキストエディター、データを見ながら試して、結果と説明を 1 枚にまとめて共有するのはノートブックです。

SAMPLE

BI ソフト: Tableau・Power BI・Looker [1.4]

BI（Business Intelligence）ソフトは、データをつないで、グラフや画面にし、組織の中で共有するための道具です。

Tableau は、ドラッグ操作を問い合わせに変える技術（VizQL）を持つ視覚的な分析の製品です。Tableau Desktop（作る）、Tableau Server・Tableau Cloud（共有する）などの製品群があります[49]。Power BI は、Microsoft の事業分析の基盤で、データをつなぎ、可視化し、組織内で共有する道具を一体で提供します。Microsoft Fabric の中核の 1 つでもあります[50]。Looker は、Google Cloud の BI で、LookML という言語でデータモデルを定義します。LookML は、SQL データベースの上の次元・集計・計算・関係を記述し、Looker はそのモデルから SQL を組み立てます[51]。計算の定義が 1 か所（セマンティックレイヤー）にあるので、誰が使っても同じ数字になります[52]。

3 つを分けるのは「作り方と置き場」です。Power BI は Microsoft の環境、Looker はモデル（LookML）を先に書く形、Tableau は画面の操作で分析を組み立てる形と覚えます。

パッケージとライブラリ: Anaconda・pandas・tidyverse [1.4]

パッケージ（ライブラリ）は、他の人が作った機能の部品で、自分でコードを書かずに読み込んで使います。

pandas は、Python のデータ分析・操作の道具で、オープンソースです。表の形のデータ（DataFrame）で、読み込み・絞り込み・集計・結合を行います[53]。tidyverse は、R で使うデータサイエンス向けのパッケージ群で、全部が同じ設計の考え方・文法・データ構造を共有します[54]。Anaconda は、Python のパッケージと環境を管理する conda に、定番のデータサイエンス・機械学習の

SAMPLE

パッケージ数百を加えた配布版です。Miniconda も conda を入れるインストーラーで、2 つの違いは最初に入る物です[55]。

取り違えやすいのは次の 3 つです。

- pandas: Python
- tidyverse: R
- Anaconda: Python の配布版（pandas などを最初から含む）

プログラミング言語: SAS・Python・R・Scala [1.4]

Python は、インタープリター型・オブジェクト指向の高水準言語で、高水準のデータ構造を内蔵しています[56]。データ分析・機械学習・自動化に広く使い、pandas などの部品が豊富です。

R は、統計計算とグラフのための言語・環境で、S 言語の実装の 1 つです。統計の検定やモデル、グラフの技法を持ち、拡張しやすい作りです[57]。研究や統計の分野で強く、tidyverse が加わります。

SAS は、SAS Institute の分析ソフトです。Base SAS は、データの読み込み・整形・分析・報告に使う第 4 世代言語（4GL）です[58]。

Scala は、オブジェクト指向と関数型を合わせた多目的の言語で、静的な型を持ちます。抽象化の使い方が安全で筋が通っているかを、コンパイル時に型で確かめます。JVM・JavaScript・Native の動作環境で動きます[59]。大規模データの処理基盤の Apache Spark は、Scala・Java・Python・R の API を持ちます[60]。

選び方は次のとおりです。

- 統計と作図が主: R

SAMPLE

- 汎用の分析と機械学習と自動化: Python
- 既存の業務で SAS が使われている: SAS
- 大規模な分散処理のコードを書く: Scala

DBMS の管理ツール [1.4]

DBMS（Database Management System）は、データベースを動かす基盤のソフトです。試験が挙げる 6 つは、その DBMS につないで、問い合わせ（SQL など）や管理を画面で行う道具です。

道具	つなげる相手	備考
SQL Server Management Studio（SSMS）	SQL Server と、その仲間（Azure SQL など）	Microsoft の管理環境[61]
MySQL Workbench	MySQL	GitHub のリポジトリは、MySQL Workbench 8 を EOL（開発終了）の版と表示し、新しい Workbench は別の場所と案内する[62]
MongoDB Compass	MongoDB	問い合わせ・集計・分析を視覚的に行う画面[63]
DBeaver	MySQL・MariaDB・PostgreSQL・SQLite など複数	Community は無料のオープンソース[64]
Toad	Oracle（Toad for Oracle）	Quest Software の製品。Oracle の SQL 開発と管理に使う[65]
Azure Data Studio	SQL Server など	2026-02-28 に提供終了。VS Code の MSSQL 拡張へ移行を案内[66]

SAMPLE

SSMS は、SQL Server の基盤全体を管理・設定・運用・開発する統合環境で、グラフィカルな道具と多くのスクリプトエディターを 1 つにしています[61]。DBeaver のように複数の DBMS に 1 つの画面でつなげる道具があり、SSMS や MongoDB Compass のように特定の DBMS 専用の道具もあります。

MongoDB に入ったドキュメントを、コードを書かずに絞り込み、集計して確認するなら Compass です。PostgreSQL と MySQL の両方に 1 つの画面から SQL を流すなら DBeaver です。SQL Server の運用（権限・バックアップ・ジョブ）を担当するなら SSMS です。Azure Data Studio を使っている現場へは、Microsoft が VS Code の MSSQL 拡張を勧めています[66]。

道具で取り違えやすい組は次のとおりです。

- IDE とテキストエディター: 実行・デバッグを持つか
- ノートブックと IDE: 結果と説明を 1 枚に残すか
- Anaconda と pandas: 配布版と部品
- Azure Data Studio: 提供終了済み（2026-02-28）

AI の概念 [1.5]

AI の用語は、包含の関係で並びます。AI の中に機械学習があり、その中にディープラーニングがあります。ディープラーニングで作った大規模なモデルが基盤モデルで、言語に特化した物が LLM です（そのモデルで文章や画像を作る使い方が生成 AI）。

NIST は AI を、人が決めた目的に対し、予測・推奨・判断を出して、現実か仮想の環境に影響する機械ベースの仕組みと定義しています[67]。

SAMPLE

生成 AI と LLM [1.5]

生成 AI（Generative AI）は、NIST が引用する大統領令 14110 の定義で、AI モデルの種類です。入力データの構造と特徴をまねて、新しい合成の内容（画像・動画・音声・文章など）を作ります[68]。

LLM（Large Language Model、大規模言語モデル）は、IBM の説明では、非常に大量のデータで学習したモデルです。自然言語などの内容を理解し生成して、幅広い作業ができます。学習には、書籍・記事・Web サイト・コードなどの数十億から数兆語を使い、トランスフォーマーという構造の上に作ります[69]。

生成 AI は、確かに見える誤りを出すことがあります。NIST は、これをコンファビュレーション（confabulation）と呼びます。自信を持って述べられた誤りや虚偽の内容の生成で、日常の言い方ではハルシネーション（幻覚）や捏造です[68]。分析の現場では、生成 AI が出した数字や SQL を、元のデータで検算してから使います。

基盤モデル・ディープラーニング・NLP [1.5]

基盤モデル（Foundation models）は、IBM の説明では、広大なデータで学習したモデルです。幅広い一般の作業をこなし、より専門のアプリを作る土台になります[70]。NIST の文書は、生成 AI が一般に生成の基盤モデルを指すと書いています。大統領令 14110 の定義を引き、デュアルユース基盤モデルを、広いデータで学習し、自己教師あり学習を使い、数百億以上のパラメーターを持つモデルと説明しています。幅広い場面に適用できる点も条件です[68]。1 つの基盤モデルを多くの用途の土台にできるのが特徴です。

ディープラーニング（Deep learning）は、多層のニューラルネットワークで学習する方法です。IBM は、層をなす多数の「ニューロン」が数学的な演算を行い、隣の層の間の結びつきの強さを機械学習で調整すると説明しています[71]。

SAMPLE

従来の機械学習が 1 つの数式で線を引くのに対し、ディープラーニングは調整できる小さな線を多数組み合わせます。

NLP（Natural Language Processing、自然言語処理）は、コンピューターが人の言葉を理解し、やり取りできるようにする、計算機科学の分野です。IBM は、用途として、チャットボット、文書の分類・重要情報の抽出・要約、機械翻訳を挙げています[72]。NLP は分野の名前で、LLM は言葉を扱う道具の 1 つです。

3 つの違いは、基盤モデルが「土台になるモデル」、ディープラーニングが「学習の方法」、NLP が「言葉を扱う分野」である点です。

RPA と自動レポート [1.5]

RPA（Robotic Process Automation）は、人が画面で行う繰り返しの作業を、ソフトウェアのロボットが代わりに行う仕組みです。IBM は、データの抽出、フォームへの入力、ファイルの移動のような繰り返しの事務作業を、スクリプトで人の操作をまねて実行すると説明しています。API と画面操作を組み合わせて、無関係なソフト間の処理を自動で進めます[73]。Microsoft の Power Automate では、デスクトップフローが RPA の機能を広げ、繰り返しの画面操作を自動化します[74]。

自動レポート（Automated reporting）は、RPA の使い道の 1 つです。毎週同じ手順で行うデータの取得・集計・資料の作成と配布を、ロボットに任せます。画面の操作と手順が決まっていて、判断が要らない繰り返しに向きます（月次の集計表の作成など）。

RPA と AI の違いは、RPA が決めた手順をなぞるだけで学習も判断もしないのに対し、AI がデータから予測や判断を出す点です。毎月、同じ画面から同じ数字を写す作業を無人にするなら RPA です。問い合わせの文面から内容を分類するなら、NLP を使った AI です。

SAMPLE

AI の用語で取り違えやすい組は次のとおりです。

- 基盤モデル・LLM・生成 AI: 土台のモデル、言語のモデル、作る使い方
- RPA と AI: 手順をなぞる、データから判断する
- ディープラーニングと NLP: 学習の方法、言葉を扱う分野

SAMPLE

サンプルはここまでです

続き(第2章〜巻末)は、Data+ 問題集のご購入で、頻出度別ノート・暗記用ノートと合わせて3点すべてダウンロードできます。

Footnotes

1. https://aws.amazon.com/relational-database/ ↩
2. PostgreSQL 「 5.5 Constraints 」 の 5.5.5 Foreign Keys 。 https://www.postgresql.org/docs/current/ddl-constraints.html ↩
3. https://aws.amazon.com/nosql/ ↩
4. https://www.mongodb.com/docs/manual/core/document/ ↩
5. https://www.rfc-editor.org/rfc/rfc4180.txt ↩
6. https://learn.microsoft.com/en-us/office/open-xml/spreadsheet/structure-of-a-spreadsheetml-document ↩
7. https://www.rfc-editor.org/rfc/rfc8259.txt ↩
8. https://jpeg.org/jpeg/ ↩
9. https://aws.amazon.com/compare/the-difference-between-structured-data-and-unstructured-data/ ↩
10. https://learn.microsoft.com/en-us/power-bi/guidance/star-schema ↩ ↩2 ↩3 ↩4 ↩5 ↩6
11. https://aws.amazon.com/what-is/data-lakehouse/ ↩

SAMPLE

12. https://www.postgresql.org/docs/current/datatype-character.html ↩ ↩2

13. https://learn.microsoft.com/en-us/sql/t-sql/data-types/nchar-and-nvarchar-transact-sql ↩

14. https://learn.microsoft.com/en-us/sql/t-sql/data-types/char-and-varchar-transact-sql ↩ ↩2

15. https://learn.microsoft.com/en-us/sql/relational-databases/spatial/spatial-data-sql-server ↩

16. https://www.postgresql.org/docs/current/datatype-boolean.html ↩

17. https://www.postgresql.org/docs/current/datatype-numeric.html ↩ ↩2 ↩3

18. https://www.postgresql.org/docs/current/datatype-datetime.html ↩ ↩2

19. https://docs.oracle.com/en/database/oracle/oracle-database/23/adlob/introduction-to-large-objects.html ↩

20. https://www.postgresql.org/docs/current/datatype-binary.html ↩

21. https://learn.microsoft.com/en-us/sql/t-sql/data-types/binary-and-varbinary-transact-sql ↩

22. https://www.rfc-editor.org/rfc/rfc9562.txt ↩

23. https://learn.microsoft.com/en-us/sql/t-sql/data-types/uniqueidentifier-transact-sql ↩

SAMPLE

24. https://aws.amazon.com/what-is/api/ ↩

25. https://support.google.com/analytics/answer/12159447 ↩

26. https://csrc.nist.gov/glossary/term/log ↩

27. https://aws.amazon.com/what-is/data-warehouse/ ↩

28. https://aws.amazon.com/compare/the-difference-between-a-data-warehouse-data-lake-and-data-mart/ ↩

29. https://aws.amazon.com/what-is/data-mart/ ↩

30. https://aws.amazon.com/what-is/data-lake/ ↩

31. https://aws.amazon.com/what-is/data-lakehouse/ ↩

32. https://www.ibm.com/think/topics/data-silos ↩

33. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html ↩

34. https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction ↩

35. https://cloud.google.com/storage/docs/introduction ↩

36. https://aws.amazon.com/redshift/ ↩

37. https://cloud.google.com/bigquery/docs/introduction ↩

38. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf ↩

39. https://aws.amazon.com/what-is/block-storage/ ↩ ↩2 ↩3

SAMPLE

40. https://docs.aws.amazon.com/ebs/latest/userguide/what-is-ebs.html ↩
41. https://aws.amazon.com/what-is/nas/ ↩
42. https://aws.amazon.com/efs/ ↩
43. https://aws.amazon.com/what-is/containerization/ ↩ ↩2
44. https://docs.docker.com/get-started/docker-concepts/the-basics/what-is-a-container/ ↩ ↩2
45. https://www.comptia.org/en-us/certifications/data/ ↩
46. https://docs.posit.co/ide/user/ ↩
47. https://code.visualstudio.com/docs/editor/whyvscode ↩
48. https://jupyter-notebook.readthedocs.io/en/latest/notebook.html ↩ ↩2 ↩3 ↩4 ↩5
49. https://www.tableau.com/why-tableau/what-is-tableau ↩
50. https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview ↩
51. https://cloud.google.com/looker/docs/what-is-lookml ↩
52. https://cloud.google.com/looker ↩
53. https://pandas.pydata.org/ ↩
54. https://www.tidyverse.org/ ↩
55. https://docs.anaconda.com/anaconda/ ↩

SAMPLE

56. https://www.python.org/doc/essays/blurb/ ↩

57. https://www.r-project.org/about.html ↩

58. https://www.sas.com/en_us/software/base-sas.html ↩

59. https://docs.scala-lang.org/tour/tour-of-scala.html https://www.scala-lang.org/ ↩

60. https://spark.apache.org/docs/latest/ ↩

61. https://learn.microsoft.com/en-us/ssms/sql-server-management-studio-ssms ↩ ↩2

62. https://github.com/mysql/mysql-workbench ↩

63. https://www.mongodb.com/docs/compass/ ↩

64. https://dbeaver.io/ ↩

65. https://www.quest.com/products/toad-for-oracle/ ↩

66. https://learn.microsoft.com/en-us/azure-data-studio/whats-happening-azure-data-studio ↩ ↩2

67. https://csrc.nist.gov/glossary/term/artificial_intelligence ↩

68. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf ↩ ↩2 ↩3

69. https://www.ibm.com/think/topics/large-language-models ↩

70. https://www.ibm.com/think/topics/foundation-models ↩

71. https://www.ibm.com/think/topics/deep-learning ↩

SAMPLE

72. https://www.ibm.com/think/topics/natural-language-processing ↩

73. https://www.ibm.com/think/topics/rpa ↩

74. https://learn.microsoft.com/en-us/power-automate/desktop-flows/introduction ↩

SAMPLE

CompTIA Data+ 学習用テキスト(無料サンプル)

版

2026年11月版(CompTIA Data+ (DA0-002) Exam Objectives Version 2.0 準拠)

発行

IT資格道場(https://www.shikaku-doujou.com)

お問い合わせ

info@shikaku-doujou.com

CompTIA、Data+、および関連するロゴ・認定名は、CompTIA(The Computing Technology Industry Association, Inc.)の商標または登録商標です。記載されているその他の会社名・製品名・サービス名は、各社の商標または登録商標です。

本教材は公式シラバスの内容を基に作成したもので、CompTIAとの提携・承認関係はありません。

本文・図表・設問はすべてオリジナルの著作物です。無断での転載・複製・再配布・二次販売はできません。お気づきの点は info@shikaku-doujou.com までご連絡ください。

SAMPLE