وثيقة التسليم - Olist مشروع مستودع بيانات

المقرر: Data Engineering Assignment

مجموعة البيانات: Olist E-Commerce (Kaggle)

الطالب: وليد عبدالقادر العباسي

المدرب: الدكتور المهندس أحمد السيد

التاريخ: 30/4/2026

جدول المحتويات

1. نظرة عامة على المشروع
2. الهندسة المعمارية وقرارات التصميم
3. نموذج البيانات
4. خط أنابيب البيانات ETL
5. تحسين الأداء
6. طبقة التقارير
7. التحقق وضمان الجودة
8. مشاكل جودة البيانات التي تمت معالجتها
9. الافتراضات الأساسية والمفاضلات
10. قائمة المخرجات المسلمة
11. كيفية التشغيل
12. الخلاصة

---

1. نظرة عامة على المشروع

ما تم بناؤه

مستودع بيانات متكامل وجاهز للإنتاج لمنصة Olist البرازيلية للتجارة الإلكترونية. يقوم الحل باستيراد البيانات التشغيلية الخام من 11 جدول في قاعدة بيانات SQLite (حوالي 1.5 مليون صف) وتحويلها عبر **هندسة Medallion (→ فضية → برونزية)** إلى **Star Schema بطريقة Kimball** محسّنة للاستعلامات التحليلية عالية الأداء. تحتوي على **11 بعداً** و **5 جداول حقائق**.

المقاييس الرئيسية

المقياس	القيمة
النظام المصدر	SQLite (11 جدول، 1,559,764 صف)
النظام الهدف	PostgreSQL (olist_olap)
الهندسة المعمارية	Kimball Star Schema + نمط Medallion
جداول الأبعاد	11 جدول (165,666 صف)

المقياس	القيمة
جداول الحقائق	5 (717,241 صف) جداول
مراحل ETL	8 (آلية بالكامل) مراحل
عدد ملفات Python	13 ملف
إجمالي أسطر الكود	~1,800 سطر
فهارس الأداء	20 فهرس B-tree
التحقق الهيكلي	29 الكل ناجح - فحصاً
مقارنة المصدر بالهدف	37 الكل ناجح - فحصاً

---

2. الهندسة المعمارية وقرارات التصميم

2.1 لماذا Kimball Star Schema؟

البديل	لماذا تم رفضه
Inmon (3NF)	معقدة متعددة الجداول لكل تقرير. استعلامات أبطأ. أصعب على مستخدمي الأعمال JOINs يتطلب
Wide Flat Table	تكرار هائل للبيانات. مستحيل الصيانة. غير مرن للأسئلة الجديدة
Data Vault	Over-engineering لهذا الحجم من البيانات. تنفيذ معقد بدون فائدة على هذا النطاق

الاختيار: Kimball Star Schema للأسباب التالية:

- أقل = استعلامات أسرع JOINs الأبعاد غير المطابقة تعني
- مفهوم بديهي لمستخدمي الأعمال (الحقائق = أحداث، الأبعاد = سياق)
- دعم أصلي في جميع أدوات BI (Power BI, Tableau, Looker)
- مناسب لحجم البيانات (~1.5M صف)

2.2 لماذا جداول حقائق متعددة؟

تعارض في الدقة (Grain): لو دمجنا كل شيء في جدول حقائق واحد، سنواجه:

العملية التجارية	الدقة الطبيعية	عدد الصفوف	لو تم الدمج
المبيعات	order_id + order_item_id	112,650	التقييمات ستتكرر لكل منتج
المدفوعات	order_id + payment_sequential	103,886	الطلبات متعددة الدفعات ستكرر المبيعات
التقييمات	review_id	99,224	الطلبات متعددة المنتجات ستحصل على تقييمات مكررة
اكتساب البائعين	mql_id	8,000	لكل أعمدة المبيعات NULL قيم
أحداث الطلب	order_id + event_type	393,481	المبيعات ستتضاعف 4 مرات

الاختيار: 5 جداول حقائق منفصلة - كل منها بدقته الصحيحة. لا يوجد عد مزدوج. تحليلات نظيفة.

2.3 لماذا هندسة Medallion؟

الطبقة	الغرض	الفائدة
البرونزية	نسخ خام 1:1 مع `_loaded_at`	أرشيف تاريخي. لو كان منطق التنظيف فيه أخطاء، نعيد المعالجة من البرونزية بدون إعادة الاستخراج من المصدر
الفضية	منظفة، مدمجة، مرقمة	Star Schema يمكن للمحللين الاستعلام مباشرة للعمل الاستكشافي دون لمس
الذهبية	جاهز Star Schema للأعمال	محسّن للتقارير. مفاتيح بديلة. فهارس أداء

2.4 جميع قرارات التصميم في لمحة

#	القرار	المبرر
1	Kimball Star Schema	استعلامات سريعة، مناسبة لأدوات BI
2	جداول حقائق متعددة (5)	الدقة الصحيحة لكل عملية تجارية
3	هندسة Medallion	تتبع البيانات، إمكانية إعادة المعالجة
4	فصل Dim_Location	مصدر واحد للحقيقة الجغرافية
5	Dim_Lead + seller_key اختياري	تحليل كامل لمسار التحويل (من lead إلى بائع)
6	Fact_Order_Events (Factless Fact)	استعلامات بسيطة لدورة حياة الطلب
7	SCD Type 1	المصدر يفتقر لتاريخ التغييرات
8	عدم وجود product_key في Fact_Reviews	احترام دقة التقييم (لكل طلب، وليس لكل منتج)
9	فهرس أداء 20	أسرع بـ 10-100 مرة JOINs
10	مفاتيح بديلة رقمية	تخزين أصغر بـ 8 مرات، مقارنات أسرع
11	فحص مقارنة 37	تحقق آلي من المصدر إلى الهدف
12	بعد Dim_Time	تحليل أنماط الشراء خلال اليوم

---

3. نموذج البيانات

3.1 نظرة عامة على Star Schema

```
                    ┌─────────────┐
                    │  dim_date   │
                    └─────────────┘
                          │
                          │ (11 علاقة)
        ┌─────────────────┼─────────────────┐
        │                 │                 │
        ▼                 ▼                 ▼
┌─────────────┐   ┌─────────────┐   ┌─────────────┐
│ dim_customer │   │  dim_seller  │   │  dim_product  │
```

```
└──────────────┘     └──────────────┘     └──────────────┘
        │                    │                    │
        │                    │                    │
        │    ┌───────────────┼────────────────┐   │
        │    │               │                │   │
┌───────▼────▼──┐  ┌─────────▼──────┐  ┌──────▼────▼──┐
│  fact_sales   │  │ fact_payments  │  │  fact_reviews │
│ (112,650 صف)  │  │ (103,886 صف)   │  │ (99,224 صف)   │
└───────────────┘  └────────────────┘  └───────────────┘

┌──────────────────┐      ┌────────────────────┐
│ fact_seller_acq  │      │  fact_order_events │
│ (8,000 صف)       │      │  (393,481 صف)      │
└──────────────────┘      └────────────────────┘
```

3.2 دقة جداول الحقائق (أهم عنصر في التصميم)

جدول الحقائق	الدقة (Grain)	ما يمثله الصف الواحد	عدد الصفوف
fact_sales	`order_id + order_item_id`	منتج واحد في طلب واحد	112,650
fact_payments	`order_id + payment_sequential`	معاملة دفع واحدة	103,886
fact_reviews	`review_id`	تقييم عميل واحد	99,224
fact_seller_acquisition	`mql_id`	عميل محتمل مؤهل واحد	8,000
fact_order_events	`order_id + event_type_key`	حدث واحد في دورة حياة طلب	393,481

3.3 جداول الأبعاد

البعد	عدد الصفوف	المفتاح	المصدر
dim_date	1,096	date_key	تم توليده (2016-2018)
dim_time	1,440	time_key	تم توليده (0-1439 دقيقة)
dim_customer	99,441	customer_key	silver.customers
dim_seller	3,095	seller_key	silver.sellers
dim_product	32,951	product_key	silver.products + الترجمة
dim_location	19,615	location_key	silver.geolocation (مُجمع)
dim_lead	8,000	lead_key	silver.leads
dim_payment_type	5	payment_type_key	قيم فريدة
dim_order_status	8	order_status_key	قيم فريدة
dim_event_type	5	event_type_key	ثابت (شراء، اعتماد، إلخ)

البعد	عدد الصفوف	المفتاح	المصدر
dim_lead_source	10	lead_source_key	قيم فريدة

3.4 مصفوفة العلاقات

البعد	fact_sales	fact_payments	fact_reviews	fact_seller_acq	fact_order_events
dim_date	✅ ×5	✅	✅ ×2	✅ ×2	✅
dim_time	✅	-	-	-	-
dim_customer	✅	✅	✅	-	✅
dim_seller	✅	-	-	✅ (اختياري)	✅
dim_product	✅	-	-	-	-
dim_location	✅ ×2	-	-	-	-
dim_order_status	✅	-	-	-	-
dim_payment_type	-	✅	-	-	-
dim_lead	-	-	-	✅	-
dim_lead_source	-	-	-	✅	-
dim_event_type	-	-	-	-	✅

المجموع: 22 علاقة. الرسم الكامل متوفر في `images/star_schema_diagram.png`.

---

4. خط أنابيب البيانات ETL

4.1 نظرة عامة على خط الأنابيب

```
main.py (أمر واحد)
  │
  ├── المرحلة 1: الاستخراج (src/extract/migrate.py)
  │   SQLite ──────────────────────────▶ olist_oltp (PostgreSQL)
  │   جدول، 1,559,764 صف تم ترحيلها 11
  │
  ├── المرحلة 2: البرونزية (src/transform/bronze/build_bronze.py)
  │   olist_oltp ──────────────────────▶ olist_olap.bronze
  │   الزمني _loaded_at نسخة طبق الأصل + طابع
  │
  ├── المرحلة 3: الفضية (src/transform/silver/build_silver.py)
  │   البرونزية ──────────────────────▶ olist_olap.silver
  │   منظفة، مدمجة، تم تجميع الترميز الجغرافي
  │
  ├── المرحلة 4: بناء الأبعاد (src/transform/gold/build_dimensions.py)
  │   11 ▶──────────────────────── الفضية DataFrame للأبعاد
  │   تم تعيين المفاتيح البديلة
  │
```

```
    ├── المرحلة 5: بناء الحقائق (src/transform/gold/build_facts.py)
    │   5 ▶──────────────── للحقائق DataFrame الفضية + الأبعاد
    │   تم استبدال المفاتيح الطبيعية بالمفاتيح البديلة
    │
    ├── المرحلة 6: التحميل (src/load/load_to_gold.py)
    │   DataFrames ──────────────────────▶ olist_olap.dimensions + olist_olap.facts
    │   تم إنشاء 20 فهرس أداء
    │
    ├── المرحلة 7: التحقق (src/validate.py)
    │   فحصاً هيكلياً (عدد الصفوف، المفاتيح الفارغة، التكامل المرجعي) 29
    │
    └── المرحلة 8: المقارنة (src/reconciliation/run_reconciliation.py)
        فحص مقارنة بين المصدر والهدف 37
```

4.2 مجموعة التقنيات المستخدمة

المكون	التقنية	الغرض
اللغة	Python 3.8+	تنفيذ كامل لخط الأنابيب
معالجة البيانات	Pandas	تحويلات DataFrame
الاتصال بالمصدر	SQLite3	قراءة olist.sqlite
الاتصال بالهدف	SQLAlchemy + psycopg2	عمليات PostgreSQL
الإعدادات	python-dotenv	متغيرات البيئة من .env

4.3 خصائص خط الأنابيب

الخاصية	كيف تحققت
قابلية التكرار	قبل الإنشاء `DROP IF EXISTS CASCADE` جميع الجداول تستخدم. أمر واحد: `python main.py`
Idempotent	في كل مكان `if_exists='replace'`. تشغيل الخط الأنابيب عدة مرات ينتج نفس النتيجة
معالجة الفشل	tracebacks يلتقط ويعرض جميع الأخطاء مع main.py في `try/except`
قابلية التوسع	للبيانات الأكبر. المنطق يبقى دون تغيير PySpark/Dask بـ Pandas يمكن استبدال
نمطي	مستقل. تعديل مرحلة لا يؤثر على الأخرى Python كل مرحلة ملف

4.4 هيكل الكود

```
code/
├── main.py                          # المنسق الرئيسي
├── requirements.txt                 # الاعتماديات
├── .env.example                     # نموذج بيانات الاعتماد
└── src/
    ├── config.py                    # الإعدادات والاتصالات
    ├── extract/migrate.py           # 1 المرحلة
    ├── transform/
    │   ├── bronze/build_bronze.py   # 2 المرحلة
```

```
    │   ├── silver/build_silver.py     # 3 المرحلة
    │   └── gold/
    │       ├── build_dimensions.py    # 4 المرحلة
    │       └── build_facts.py         # 5 المرحلة
    ├── load/load_to_gold.py           # 6 المرحلة
    ├── validate.py                    # 7 المرحلة
    └── reconciliation/
        ├── reconciliation_queries.py  # تعريفات الاستعلامات
        └── run_reconciliation.py     # 8 المرحلة
```

---

5. تحسين الأداء

5.1 استراتيجية الفهرسة

تم إنشاؤها على جميع أعمدة المفاتيح الخارجية في جداول الحقائق **B-tree فهرس 20**.

جدول الحقائق	الأعمدة المفهرسة
fact_sales	customer_key, seller_key, product_key, purchase_date_key, delivered_customer_date_key, order_status_key, customer_location_key, seller_location_key (8 فهارس)
fact_payments	customer_key, payment_date_key, payment_type_key (3 فهارس)
fact_reviews	customer_key, review_creation_date_key (2 فهارس)
fact_seller_acquisition	lead_key, seller_key, lead_source_key (3 فهارس)
fact_order_events	customer_key, seller_key, event_date_key, event_type_key (4 فهارس)

تأثير الأداء:

نوع الاستعلام	بدون فهرس	مع فهرس
JOIN fact_sales ↔ dim_customer	مسح كامل للجدول (112K صف)	بحث بالفهرس (~3 قراءات صفحة)
تصفية بنطاق تاريخي	مسح كامل للجدول	مسح نطاق بالفهرس
JOIN مع GROUP BY	Hash join على مسح كامل	Nested loop مع فهرس

5.2 المفاتيح البديلة

جميع الأبعاد تستخدم **مفاتيح بديلة** **INTEGER** بدلاً من المفاتيح الطبيعية VARCHAR(32).

المقياس	المفاتيح الطبيعية (VARCHAR)	المفاتيح البديلة (INT)
التخزين لكل FK	32 bytes	4 bytes
حجم الفهرس	أكبر	أصغر بـ 8 مرات
سرعة JOIN	أبطأ (مقارنة نصوص)	أسرع (مقارنة أعداد صحيحة)

مثال: fact_sales يحتوي 112,650 صف × 5 FKs. المفاتيح البديلة توفر ~15 MB من التخزين لأعمدة FK وحدها.

5.3 تحسينات إضافية

التقنية	الحالة	الفائدة
Materialized Views	جاهزة للتنفيذ	تجميعات يومية محسوبة مسبقاً
Role-Playing Dimensions	مطبقة (dim_date ×11)	جدول واحد يخدم 11 علاقة
Degenerate Dimensions	مطبقة (order_id)	عبر الحقائق بدون بعد إضافي JOINs

---

6. طبقة التقارير

للاطلاع على `queries/sample_queries.sql` الإجابة على جميع أسئلة الأعمال المطلوبة وأكثر. راجع Star Schema يتيح SQL الكامل.

6.1 الأسئلة المطلوبة وطريقة الإجابة عليها

سؤال العمل	الاستعلام	الجداول الرئيسية المستخدمة
كيف تتجه المبيعات عبر الزمن؟	الإيرادات الشهرية مع مقارنة سنوية	fact_sales + dim_date
من هم العملاء الأكثر قيمة؟	أفضل 10 حسب القيمة التراكمية	fact_sales + dim_customer
ما الذي يؤثر على أداء التسليم؟	التأخير حسب الولاية والمنطقة	fact_sales + dim_location
ما هي المنتجات/الفئات التي تدر الإيرادات؟	إيرادات الفئة مع درجات التقييم	fact_sales + dim_product + fact_reviews

6.2 استعلامات تحليلية إضافية (فوق المطلوب)

الاستعلام	القيمة التجارية
سلوك الدفع (الأقساط مقابل قيمة الطلب)	فهم كيف تؤثر خيارات الدفع على الإنفاق
مسار دورة حياة الطلب (من الشراء للتسليم)	تحديد أين تتسرب الطلبات
مسار اكتساب البائعين حسب القناة التسويقية	تحسين الإنفاق التسويقي
معدل تكرار شراء العملاء	قياس الولاء والاحتفاظ
المبيعات اليومية حسب أيام الأسبوع مقابل العطل	تخطيط التوظيف والحملات
تأثير تأخير التسليم على درجات التقييم	قياس كيف تؤثر اللوجستيات على الرضا
بطاقة أداء أفضل البائعين	تحديد أفضل الشركاء
الإيرادات حسب المنطقة البرازيلية	تخطيط التوسع الجغرافي

---

7. التحقق وضمان الجودة

7.1 التحقق الهيكلي (29 فحصاً)

الوصف	العدد	الفئة
جداول الحقائق تطابق أعداد المصدر الفضي	5	مقارنة عدد الصفوف
المفاتيح الأساسية فارغة (باستثناء seller_key للعملاء غير المحولين)	18	الكشف عن المفاتيح الخارجية الفارغة
جميع FKs لها مفاتيح أساسية مطابقة في الأبعاد	6	التكامل المرجعي

7.2 اختبار المقارنة (37 فحصاً)

يقارن المقاييس المجمعة بين المصدر (olist_olap) والهدف (olist_oltp):

ما تتحقق منه	الفحوصات	الفئة
لا يوجد فقدان للبيانات أثناء ETL	6	A - عدد الصفوف
دقة الإيرادات، الشحن، المدفوعات	5	B - المقاييس المالية
الحالات، أنواع الدفع، درجات التقييم، الفئات، العملاء المحتملين	7	C-G - التوزيعات
الأحداث، الجغرافيا، التواريخ، المنتجات، الأقساط، التسليم	19	H-N - متقدم

7.3 نتائج التحقق

```
============================================================
  تقرير التحقق - 29 فحصاً
============================================================
  ✓ جميع الفحوصات ناجحة
============================================================

============================================================
  ملخص المقارنة - 37 فحصاً
============================================================
  المجموع: 37 | ناجح: 37 | فاشل: 0
  الحالة: ✓ المصدر والهدف متطابقان
============================================================

╔═══════════════════════════════════════════════════════════╗
╚═══════════════════ تم إكمال خط الأنابيب بنجاح ═══════════╝
```

---

8. مشاكل جودة البيانات التي تمت معالجتها

#	المشكلة	جدول المصدر	كيف عالجناها
1	إحداثيات بصيغة علمية (`-2.354e+01`)	geolocation	تحويلها إلى صيغة عشرية صحيحة (`-23.54`)
2	أسماء مدن غير متناسقة (`sao paulo` مقابل `são paulo`)	geolocation	توحيد: lowercase، إزالة علامات التشكيل، تقليم المسافات

#	المشكلة	جدول المصدر	كيف عالجناها
3	1,000,163 صف مكرر في geolocation	geolocation	تجميع حسب zip_code_prefix ← 19,615 صف فريد
4	أسماء فئات إنجليزية مفقودة	products	الربط مع جدول الترجمة؛ ملء الفارغ بالاسم البرتغالي الأصلي
5	اسم عمود به خطأ إملائي (`product_name_lenght`)	products	تصحيح الاسم إلى `product_name_length`
6	تواريخ مخزنة كنصوص	orders, reviews	تحويلها إلى DATETIME/TIMESTAMP صحيح
7	قيم NULL في leads_closed (average_stock، إلخ)	leads_closed	له معنى: العميل لم) NULL الحفاظ على يتحول بعد)
8	صفوف دفع متعددة لكل طلب	order_payments	الحفاظ عليها كصفوف منفصلة (الدقة الصحيحة)

9. الافتراضات الأساسية والمفاضلات

9.1 الافتراضات الأساسية

الافتراض	السبب
كل order_item 1 = يمثل كمية	هيكل البيانات المصدرية: صف واحد لكل منتج فريد في كل طلب
تاريخ الدفع ≈ تاريخ الشراء	يفتقر إلى طابع زمني صريح للدفع `order_payments` جدول
الرموز البريدية للترميز الجغرافي تغطي معظم العملاء	من 99,441 عميل (0.3%) لديهم رموز بريدية غير موجودة في 302 جدول geolocation
SCD Type 1 كافٍ	البيانات المصدرية لا توفر تاريخ تغييرات العناوين

9.2 المفاضلات

المفاضلة	ما كسبناه	ما ضحينا به
جداول حقائق متعددة مقابل جدول واحد	دقة صحيحة، لا عد مزدوج	استعلامات عبر الحقائق تتطلب JOINs
Star Schema 3 مقابلNF	استعلامات سريعة، بسيطة لأدوات BI	بعض تكرار البيانات في الأبعاد
جدول Fact_Order_Events	استعلامات مسار بسيطة، أداء سريع	صف Kتخزين إضافي 393
الطبقة البرونزية	إمكانية إعادة المعالجة، سجل التدقيق	تخزين إضافي، خط أنابيب أطول
20 فهرساً	أداء استعلامات أسرع 10-100 مرة	تخزين إضافي ~10%

10. قائمة المخرجات المسلمة

#	المخرج المطلوب	ما سلمناه	الحالة
1	**نموذج البيانات (رسم + شرح)**	`images/star_schema_diagram.png` + `docs/star_schema_diagram.md` + `docs/data_dictionary.md`	✅
2	**تنفيذ ETL/ELT**	`code/` - 13 ملف Python، 8 مراحل	✅
3	**المخطط الهدف (DWH)**	`docs/data_dictionary.md` - 16 جدول موثق بالكامل	✅
4	**استعلامات تحليلية نموذجية**	`queries/sample_queries.sql` - 12 استعلام أعمال	✅
5	**توثيق القرارات**	`docs/decisions.md` - 12 سجل قرار معماري	✅

مخرجات إضافية (فوق المطلوب)

العنصر	الموقع	الغرض
إثبات تنفيذ الخط الأنابيب	`pipeline_output.txt`	يثبت نجاح جميع الفحوصات الـ 66
استعلامات المقارنة	`queries/reconciliation_queries.sql`	استعلام تحقق من المصدر مقابل 18 الهدف
رسم الهندسة المعمارية	`images/architecture_diagram.png`	تصور كامل لتدفق البيانات
رسم طبقات Medallion	`images/medallion_layers.png`	مراحل التحويل الثلاث
تعليمات الإعداد	`code/README.md`	دليل تشغيل خطوة بخطوة

---

11. كيفية التشغيل

المتطلبات الأساسية

- Python 3.8+
- PostgreSQL 12+

بداية سريعة

```
# 1. الانتقال إلى مجلد الكود
cd code

# 2. تثبيت الاعتماديات
pip install -r requirements.txt

# 3. إعداد بيانات اعتماد قاعدة البيانات
cp .env.example .env
# تعديل .env بمعلومات PostgreSQL كلمة ، المستخدم ، المنفذ ، المضيف) الخاصة بك
المرور)
```

```
# 4. وضع olist.sqlite في مجلد data/
# تحميل من: https://www.kaggle.com/datasets/terencicp/e-commerce-dataset-by-olist-as-an-sqlite-database

# 5. تشغيل خط الأنابيب الكامل
python main.py
```

الخرج المتوقع

- المراحل 1-8 تنفذ بالتسلسل
- فحصاً هيكلياً ناجحاً 29/29
- فحص مقارنة ناجحاً 37/37
- الرسالة النهائية: **PIPELINE COMPLETED SUCCESSFULLY**

هيكل المشروع بعد الإعداد

```
Olist_DWH_Project/
├── README.md                    # توثيق كامل للمشروع
├── code/                        # جميع الأكواد + code/README.md
├── docs/                        # 5 ملفات توثيق مفصلة
├── images/                      # 3 رسوم بيانية احترافية
├── queries/                     # 12 استعلام مقارنة + 18 استعلام تحليلي
└── pipeline_output.txt          # سجل تنفيذ ناجح
```

---

12. الخلاصة

ما تم تسليمه

للتجارة الإلكترونية يقوم بـ Olist **حل مستودع بيانات كامل وجاهز للإنتاج** لمنصة:

1. **استيراد** البيانات التشغيلية الخام من SQLite (11 جدول، 1.5M صف)
2. **تحويلها** عبر هندسة Medallion (برونزية → فضية → ذهبية)
3. **نمذجتها** كـ Kimball Star Schema (11 جداول حقائق + 5 بعداً)
4. **تحسينها** بـ 20 فهرس أداء ومفاتيح بديلة رقمية
5. **التحقق منها** بـ 66 فحصاً آلياً (29 هيكلي + 37 مقارنة)
6. **توثيق** كل قرار وافتراض ومفاضلة

نقاط القوة الرئيسية

نقطة القوة	الدليل
نمذجة بيانات صحيحة	5 حقائق بدقات مختلفة تمنع العد المزدوج
خط أنابيب نظيف	نمطي، 8 مراحل محددة جيداً، Idempotent، أمر واحد
جاهز للإنتاج	معالجة أخطاء، تسجيل، فهارس، مفاتيح بديلة

الدليل	نقطة القوة
ADR رسوم بيانية، 12 3 ،docs ملفات 5 ،README	**موثق بشكل شامل**

التوافق مع معايير التقييم

كيف تميزنا	المعيار
قراراً معمارياً موثقاً مع البدائل والمبررات 12	**التفكير**
دقة صحيحة لكل جدول حقائق، 22 علاقة نظيفة، لا عد مزدوج	**النمذجة**
قابل للتكرار ،Idempotent ،خط أنابيب نمطي من 8 مراحل	**الهندسة**
فهرساً، مفاتيح بديلة، أبعاد متعددة الأدوار 20	**الأداء**
توثيق واضح على كل المستويات (تعليقات الكود ← docs ← README)	**التواصل**

نهاية وثيقة التسليم