SAP SD Sales Analytics Platform Solution Design Document

SAP to Azure Cloud with CDC, Medallion Architecture & AI Analytics

Project:	ERP Sales Analytics - SAP SD
Version:	2.0 (Enhanced)
Date:	October 25, 2025
Architecture:	Medallion (Bronze-Silver-Gold)
Source System:	SAP ECC/S4HANA SD Module

SAP SD Module - Source Tables

This solution extracts data from SAP Sales & Distribution (SD) module focusing on the complete order-to-cash process. Tables are categorized by Master Data (reference/dimensional) vs Transaction Data (events/facts), with specific extraction strategies for each type.

Master Data Tables

Table	Description	Key Fields	Load Strategy	Est. Rows
KNA1	Customer Master General	KUNNR	Full Weekly + CDC	~500K
KNVV	Customer Sales Data	KUNNR, VKORG, VTWEG, SPART	Full Weekly + CDC	~1M
KNB1	Customer Company Data	KUNNR, BUKRS	Full Weekly + CDC	~500K
KNVP	Customer Partner Functions	KUNNR, VKORG, VTWEG, SPART, PARVW	Full Weekly	~2M
MARA	Material General Data	MATNR	Full Weekly + CDC	~200K
MARC	Material Plant Data	MATNR, WERKS	Full Weekly + CDC	~800K
MAKT	Material Descriptions	MATNR, SPRAS	Full Weekly	~400K
MVKE	Material Sales Data	MATNR, VKORG, VTWEG	Full Weekly	~600K
T001	Company Codes	BUKRS	Full Daily	~50
TVKO	Sales Organizations	VKORG	Full Daily	~20
TVTW	Distribution Channels	VTWEG	Full Daily	~10
TSPA	Divisions	SPART	Full Daily	~15
T023	Material Groups	MATKL	Full Daily	~500
T005	Countries	LAND1	Full Weekly	~250
T171T	Product Hierarchy Text	PRODH, SPRAS	Full Weekly	~5K

Transaction Data Tables

Table	Description	Key Fields	Load Strategy	Daily Vol
VBAK	Sales Document Header	VBELN	Incremental CDC (ERDAT)	~5K
VBAP	Sales Document Item	VBELN, POSNR	Incremental CDC (ERDAT)	~25K
VBUK	Sales Document Status	VBELN	Incremental CDC (AEDAT)	~5K
VBUP	Sales Item Status	VBELN, POSNR	Incremental CDC (AEDAT)	~25K
VBEP	Sales Schedule Lines	VBELN, POSNR, ETENR	Incremental CDC (ERDAT)	~30K
LIKP	Delivery Header	VBELN	Incremental CDC (ERDAT)	~3K
LIPS	Delivery Item	VBELN, POSNR	Incremental CDC (ERDAT)	~15K
VBRK	Billing Document Header	VBELN	Incremental CDC (ERDAT)	~4K
VBRP	Billing Document Item	VBELN, POSNR	Incremental CDC (ERDAT)	~20K
VBFA	Sales Document Flow	VBELV, POSNV, VBELN, POSNN	Incremental CDC	~30K
KONV	Pricing Conditions	KNUMV, KPOSN, STUNR, ZAEHK	Incremental (Join)	~100K
VBPA	Sales Partners	VBELN, POSNR, PARVW	Incremental CDC	~40K
VTTK	Shipment Header	TKNUM	Incremental CDC (ERDAT)	~2K
VTTP	Shipment Item	TKNUM, TPNUM	Incremental CDC (ERDAT)	~8K

SAP Change Data Capture (CDC) Concepts

Change Data Capture enables efficient incremental extraction by identifying only records that have changed since the last extraction. SAP provides multiple CDC mechanisms depending on table type and S/4HANA vs ECC system version.

CDC Mechanisms by Table Type

Mechanism	SAP Tables	How It Works	Best For	Implementation
Application Timestamp	ERDAT, ERZET AEDAT, AEZET	Created/changed date/time fields in application tables	Transaction tables (VBAK, VBRK, LIKP)	WHERE ERDAT >= :watermark OR AEDAT >= :watermark
Change Documents	CDHDR CDPOS	SAP change document framework logs all changes to enabled tables	Master data (KNA1, MARA) with history	Query CDHDR for OBJECTCLAS Join CDPOS for field changes
Status Fields	GBSTK, FKSTK WBSTK, LFSTK	Overall, billing, goods movement, delivery status	Incomplete documents Status tracking	WHERE GBSTK NOT IN ('C') for open docs
Timestamp Fields	LAST_CHANGE_DATE_TIME	UTC timestamp field in S/4HANA tables	S/4HANA tables All changes	WHERE LAST_CHANGE_DATE_TIME >= :watermark
Delta Queue (ODP)	ROOSOURCE ROOSGENQT	Operational Data Provisioning extractors	Certified extractors SAP recommended	Configure ODP context Subscribe to datasource

CDC Implementation Example: VBAK Sales Orders

```
-- Example: Extract VBAK changes since last load -- Watermark stored in Azure SQL: 2024-01-15 08:00:00 SELECT VBELN, --
Sales Document Number ERDAT, -- Created Date ERZET, -- Created Time ERNAM, -- Created By AEDAT, -- Changed Date AUDAT, --
Document Date VBTYP, -- Document Category (Order, Quotation, etc.) AUART, -- Order Type VKORG, -- Sales Organization VTWEG,
-- Distribution Channel SPART, -- Division KUNNR, -- Sold-to Customer NETWR, -- Net Value WAERK, -- Currency GBSTK, --
Overall Status ABSTK, -- Rejection Status LIFSK, -- Delivery Block FAKSK -- Billing Block FROM VBAK WHERE ERDAT >=
'20240115' -- Created since last watermark OR (AEDAT >= '20240115' -- OR Changed since last watermark AND AEDAT IS NOT NULL)
ORDER BY ERDAT, ERZET; -- Result: Only documents created or modified since last run -- Typical result: 150-200 orders per
hour in high-volume systems
```

Watermark Management Strategy

Watermarks track the last successfully extracted timestamp for each table, enabling reliable incremental loads without data loss or duplication:

Table Name	Last Watermark	Watermark Type	Status	Records
VBAK	2024-01-15 14:30:00	ERDAT/AEDAT	Success	187
VBAP	2024-01-15 14:30:00	ERDAT	Success	934
VBRK	2024-01-15 14:00:00	ERDAT/AEDAT	Success	145
VBRP	2024-01-15 14:00:00	ERDAT	Success	728
LIKP	2024-01-15 14:15:00	ERDAT	Success	98
KNA1	2024-01-14 00:00:00	CDHDR.UDATE	Success	23
MARA	2024-01-14 00:00:00	CDHDR.UDATE	Success	15

Data Lake Folder Structure

The Azure Data Lake follows a hierarchical organization separating Bronze (raw), Silver (cleansed), and Gold (curated) layers. Within each layer, data is further organized by data type (master vs transactional), source system, table name, and load pattern.

Complete Folder Hierarchy

```
/datalake-prod/ ■ ■■■ bronze/ # Raw data layer (immutable) ■ ■■■ master/ # Master data (dimensions) ■ ■ ■■■ customer/ ■
■ ■ ■■■ kna1/ ■ ■ ■ ■ ■■■ full/ ■ ■ ■ ■ ■ ■■■ 2024/01/15/ ■ ■ ■ ■ ■ ■ ■■■ kna1_20240115_083000.parquet (~1.2GB, 500K
rows) ■ ■ ■ ■ ■ ■■■ 2024/01/08/ ■ ■ ■ ■ ■ ■■■ kna1_20240108_083000.parquet ■ ■ ■ ■ ■■■ incremental/ ■ ■ ■ ■ ■■■
2024/01/15/ ■ ■ ■ ■ ■■■ kna1_changes_20240115.parquet (~5MB, 234 changes) ■ ■ ■ ■■■ knvv/ ■ ■ ■ ■ ■■■
full/2024/01/15/knvv_20240115_083000.parquet (~2.5GB, 1M rows) ■ ■ ■ ■■■ knb1/ ■ ■ ■ ■■■
full/2024/01/15/knb1_20240115_083000.parquet (~1GB, 500K rows) ■ ■ ■ ■ ■ ■■■ material/ ■ ■ ■ ■■■ mara/ ■ ■ ■ ■ ■■■
full/2024/01/15/mara_20240115_083000.parquet (~450MB, 200K rows) ■ ■ ■ ■■■ marc/ ■ ■ ■ ■ ■■■
full/2024/01/15/marc_20240115_083000.parquet (~1.8GB, 800K rows) ■ ■ ■ ■■■ makt/ ■ ■ ■ ■ ■■■
full/2024/01/15/makt_20240115_083000.parquet (~80MB, 400K rows) ■ ■ ■ ■■■ mvke/ ■ ■ ■ ■■■
full/2024/01/15/mvke_20240115_083000.parquet (~600MB, 600K rows) ■ ■ ■ ■ ■ ■■■ organizational/ ■ ■ ■ ■■■ t001/ # Company
Codes ■ ■ ■ ■ ■■■ full/2024/01/15/t001_20240115_080000.parquet (~1MB, 50 rows) ■ ■ ■ ■■■ tvko/ # Sales Orgs ■ ■ ■ ■ ■■■
full/2024/01/15/tvko_20240115_080000.parquet (~500KB, 20 rows) ■ ■ ■ ■■■ tvtw/ # Distribution Channels ■ ■ ■ ■■■
full/2024/01/15/tvtw_20240115_080000.parquet (~300KB, 10 rows) ■ ■ ■ ■ ■ ■■■ product_hierarchy/ ■ ■ ■■■ t171t/ ■ ■ ■■■
full/2024/01/15/t171t_20240115_080000.parquet (~10MB, 5K rows) ■ ■ ■ ■■■ transactional/ # Transaction data (facts) ■ ■
■■■ sales_orders/ ■ ■ ■ ■■■ vbak/ # Sales Header ■ ■ ■ ■ ■■■ full/ ■ ■ ■ ■ ■ ■■■ 2024/01/01/ ■ ■ ■ ■ ■ ■■■
vbak_full_20240101.parquet (~15GB, 5M historical orders) ■ ■ ■ ■ ■■■ incremental/ ■ ■ ■ ■ ■■■ 2024/01/15/08/ ■ ■ ■ ■ ■
■■■ vbak_delta_20240115_08.parquet (~12MB, 187 orders) ■ ■ ■ ■ ■■■ 2024/01/15/14/ ■ ■ ■ ■ ■ ■■■
vbak_delta_20240115_14.parquet (~15MB, 234 orders) ■ ■ ■ ■ ■■■ 2024/01/15/20/ ■ ■ ■ ■ ■■■ vbak_delta_20240115_20.parquet
(~11MB, 172 orders) ■ ■ ■ ■ ■ ■ ■ ■■■ vbap/ # Sales Items ■ ■ ■ ■ ■■■ incremental/ ■ ■ ■ ■ ■■■ 2024/01/15/08/ ■ ■ ■ ■
■■■ vbap_delta_20240115_08.parquet (~45MB, 934 items) ■ ■ ■ ■ ■ ■ ■ ■■■ vbuk/ # Sales Header Status ■ ■ ■ ■ ■■■
incremental/2024/01/15/08/ ■ ■ ■ ■ ■■■ vbuk_delta_20240115_08.parquet (~8MB, 187 status) ■ ■ ■ ■ ■ ■ ■ ■■■ vbup/ #
Sales Item Status ■ ■ ■ ■■■ incremental/2024/01/15/08/ ■ ■ ■ ■■■ vbup_delta_20240115_08.parquet (~28MB, 934 status) ■ ■
■ ■ ■ ■■■ deliveries/ ■ ■ ■ ■■■ likp/ # Delivery Header ■ ■ ■ ■ ■■■ incremental/2024/01/15/ ■ ■ ■ ■ ■■■
likp_delta_20240115_14.parquet (~6MB, 98 deliveries) ■ ■ ■ ■■■ lips/ # Delivery Items ■ ■ ■ ■■■ incremental/2024/01/15/
■ ■ ■ ■■■ lips_delta_20240115_14.parquet (~22MB, 487 items) ■ ■ ■ ■ ■ ■■■ billing/ ■ ■ ■ ■■■ vbrk/ # Billing Header ■
■ ■ ■ ■■■ incremental/2024/01/15/ ■ ■ ■ ■ ■■■ vbrk_delta_20240115_14.parquet (~9MB, 145 invoices) ■ ■ ■ ■■■ vbrp/ #
Billing Items ■ ■ ■ ■■■ incremental/2024/01/15/ ■ ■ ■ ■■■ vbrp_delta_20240115_14.parquet (~35MB, 728 items) ■ ■ ■ ■ ■
■■■ document_flow/ ■ ■ ■ ■■■ vbfa/ ■ ■ ■ ■■■ incremental/2024/01/15/ ■ ■ ■ ■■■ vbfa_delta_20240115_14.parquet (~18MB,
1245 flows) ■ ■ ■ ■ ■ ■■■ pricing/ ■ ■ ■ ■■■ konv/ ■ ■ ■ ■■■ incremental/2024/01/15/ ■ ■ ■ ■■■
konv_delta_20240115_14.parquet (~120MB, 4500 conditions) ■ ■ ■ ■ ■ ■■■ partners/ ■ ■ ■■■ vbpa/ ■ ■ ■■■
incremental/2024/01/15/ ■ ■ ■■■ vbpa_delta_20240115_14.parquet (~25MB, 1872 partners) ■ ■ ■ ■■■ staging/ # Temporary
landing area ■ ■ ■■■ inbound/ ■ ■ ■ ■■■ vbak_20240115_143000.csv # Raw extracted CSV ■ ■ ■ ■■■ vbap_20240115_143000.csv
■ ■ ■■■ validated/ ■ ■ ■■■ vbak_validated.parquet # After format validation ■ ■ ■■■ vbap_validated.parquet ■ ■ ■ ■■■
cdc/ # Change document capture ■ ■■■ change_documents/ ■ ■ ■■■ cdhdr/ ■ ■ ■ ■■■ 2024/01/15/ ■ ■ ■ ■■■
cdhdr_20240115_080000.parquet (~5MB, KNA1/MARA changes) ■ ■ ■■■ cdpos/ ■ ■ ■■■ 2024/01/15/ ■ ■ ■■■
cdpos_20240115_080000.parquet (~15MB, field-level changes) ■ ■■■ application_logs/ ■ ■■■ 2024/01/15/ ■ ■■■
extraction_log_20240115.json ■ ■■■ silver/ # Cleansed/validated data ■ ■■■ master/ ■ ■ ■■■ dim_customer/ # Delta table
(SCD Type 2) ■ ■ ■ ■■■ _delta_log/ ■ ■ ■ ■ ■■■ 00000000000000000000.json ■ ■ ■ ■ ■■■ 00000000000000000001.json ■ ■ ■
■■■ part-00000-*.snappy.parquet ■ ■ ■ ■■■ part-00001-*.snappy.parquet ■ ■ ■ ■ ■ ■■■ dim_material/ # Delta table (SCD
Type 2) ■ ■ ■ ■■■ [Delta Lake files...] ■ ■ ■ ■ ■ ■■■ dim_organization/ ■ ■ ■■■ [Delta Lake files...] ■ ■ ■ ■■■
transactional/ ■ ■■■ fact_sales_orders/ # Delta table ■ ■ ■■■ _delta_log/ ■ ■ ■■■ order_date=2024-01-15/ ■ ■ ■■■
part-*.snappy.parquet ■ ■ ■ ■■■ fact_deliveries/ ■ ■ ■■■ delivery_date=2024-01-15/ ■ ■ ■■■ part-*.snappy.parquet ■ ■ ■
■■■ fact_billing/ ■ ■■■ billing_date=2024-01-15/ ■ ■■■ part-*.snappy.parquet ■ ■■■ gold/ # Business-ready curated data
■ ■■■ facts/ ■ ■ ■■■ fact_sales_daily/ # Pre-aggregated daily sales ■ ■ ■ ■■■ year=2024/month=01/day=15/ ■ ■ ■ ■■■
part-*.snappy.parquet (~500MB, 5K daily aggregates) ■ ■ ■ ■ ■ ■■■ fact_customer_orders/ # Customer order analytics ■ ■ ■
■■■ [Partitioned by customer_segment/date] ■ ■ ■ ■ ■ ■■■ fact_product_sales/ # Product sales metrics ■ ■ ■■■
[Partitioned by product_category/date] ■ ■ ■ ■■■ dimensions/ ■ ■ ■■■ dim_customer_scd2/ # Customer with full history ■ ■
■ ■■■ [Delta Lake SCD Type 2] ■ ■ ■ ■ ■ ■■■ dim_product_hierarchy/ # Product with hierarchy ■ ■ ■ ■■■ [Delta Lake
current view] ■ ■ ■ ■ ■ ■■■ dim_date/ # Date dimension ■ ■ ■■■ [Full calendar 2020-2030] ■ ■ ■ ■■■ metrics/ ■ ■■■
kpi_sales_performance/ # Pre-calculated KPIs ■ ■ ■■■ [Daily/Monthly/Quarterly metrics] ■ ■ ■ ■■■ kpi_customer_analytics/
# Customer metrics ■ ■■■ [CLV, Churn Score, RFM segments] ■ ■■■ metadata/ # Control & audit data ■■■ watermarks/ ■ ■■■
watermark_state.json # Current watermarks per table ■ ■■■ watermark_history/ ■ ■■■ 2024/01/ ■ ■■■
watermarks_20240115.parquet ■ ■■■ audit/ ■ ■■■ load_history/ ■ ■ ■■■ 2024/01/ ■ ■ ■■■ load_audit_20240115.parquet #
Load stats per pipeline ■ ■ ■ ■■■ data_quality/ ■ ■■■ 2024/01/ ■ ■■■ quality_checks_20240115.parquet # DQ validation
results ■ ■■■ lineage/ ■ ■■■ table_dependencies.json # Source-to-target mappings ■ ■■■ schemas/ ■■■ bronze/ ■ ■■■
vbak_schema.json ■■■ silver/ ■ ■■■ fact_sales_orders_schema.json ■■■ gold/ ■■■ fact_sales_daily_schema.json
```

Folder Structure Design Principles

- **Layer Separation:** Bronze (raw, immutable), Silver (cleansed, validated), Gold (business-ready) - clear boundaries
- **Master vs Transaction:** Explicit separation enables different processing patterns (full vs incremental, SCD vs append)
- **Date Partitioning:** Year/Month/Day/Hour hierarchy for efficient time-based queries and data lifecycle management
- **File Naming:** Consistent naming: {table}_{type}_{timestamp}.{format} enables automated discovery and processing
- **Delta Lake in Silver/Gold:** ACID transactions, time travel, schema evolution, and efficient upserts
- **Staging Area:** Temporary landing for validation before committing to Bronze ensures data quality
- **Metadata Separation:** Watermarks, audit, lineage in dedicated folder for governance and troubleshooting

Synthetic Data Generation for Testing

To enable development and testing without access to production SAP data, we generate realistic synthetic datasets that mimic production data volume, distribution, and relationships while containing no real business information.

Synthetic Data Approach

- **Volume Matching:** Generate datasets matching expected production volumes (500K customers, 5M orders, etc.)
- **Referential Integrity:** Maintain FK relationships (VBAP.VBELN → VBAK.VBELN, VBAK.KUNNR → KNA1.KUNNR)
- **Realistic Distributions:** Mimic real patterns (80/20 rule for top customers, seasonal order patterns, pricing variance)
- **Data Type Accuracy:** Correct field types, lengths, and constraints matching SAP data dictionary
- **Locale Realism:** Region-appropriate names, addresses, currencies, and business rules
- **Temporal Consistency:** Realistic date sequences (VBAK.ERDAT <= LIKP.ERDAT <= VBRK.ERDAT)

Sample Synthetic Data - VBAK (Sales Orders)

```
VBELN |ERDAT |AUART|VKORG|VTWEG|SPART|KUNNR |NETWR |WAERK|GBSTK
-----------|---------|-----|-----|-----|-----|----------|-----------|-----|----- 0000001234 |20240115 |OR |1000 |10 |00
|0000100234|12450.00 |USD |B 0000001235 |20240115 |OR |1000 |10 |00 |0000100891|8920.50 |USD |C 0000001236 |20240115 |ZOR
|1000 |10 |00 |0000102456|156780.00 |USD |A 0000001237 |20240115 |OR |2000 |10 |00 |0000205673|4567.25 |EUR |B 0000001238
|20240115 |OR |1000 |20 |00 |0000100234|23450.75 |USD |C -- Characteristics: -- ~5,000 orders per day (187 per hour during
business hours) -- Order types: OR (70%), ZOR (20%), QT (10%) -- Sales Orgs: 1000 (60%), 2000 (30%), 3000 (10%) -- Status
distribution: A-Complete (40%), B-In Process (35%), C-Blocked (25%)
```

Sample Synthetic Data - VBAP (Sales Order Items)

```
VBELN |POSNR|MATNR |ARKTX |KWMENG |VRKME|NETWR |WAERK
-----------|-----|--------------|-------------------------|--------|-----|---------|----- 0000001234
|00010|000000000012345|Widget A Standard |100.000 |EA |5000.00 |USD 0000001234 |00020|000000000023456|Widget B Premium
|50.000 |EA |7450.00 |USD 0000001235 |00010|000000000034567|Gadget C Industrial |10.000 |EA |8920.50 |USD 0000001236
|00010|000000000045678|Machine Part XL-1000 |5.000 |EA |156780.00|USD 0000001237 |00010|000000000012345|Widget A Standard
|75.000 |EA |3750.00 |EUR 0000001237 |00020|000000000056789|Service - Installation |1.000 |EA |817.25 |EUR --
Characteristics: -- Average 5 items per order (range: 1-50) -- Material distribution: 20% of materials account for 80% of
order volume -- Quantities follow log-normal distribution -- ~25,000 order items per day
```

Sample Synthetic Data - KNA1 (Customer Master)

```
KUNNR |NAME1 |LAND1|PSTLZ |ORT01 |KTOKD|ERDAT |BRSCH
----------|----------------------|-----|------|---------------|-----|---------|----- 0000100234|Acme Industrial Corp |US
|10001 |New York |0001 |20180523 |1200 0000100891|Global Tech Solutions |US |94105 |San Francisco |0001 |20190315 |7100
0000102456|Manufacturing Intl LLC|US |60601 |Chicago |0002 |20170822 |2900 0000205673|European Distributors |DE |10115
|Berlin |0001 |20200101 |5100 0000308942|Pacific Rim Trading |SG |018956|Singapore |0001 |20210614 |5200 --
Characteristics: -- 500,000 customers total -- Customer groups: 0001-Standard (70%), 0002-Key Account (20%), 0003-Export
(10%) -- Industries (BRSCH): Manufacturing (30%), Retail (25%), Services (20%), Other (25%) -- Geographic: US (60%), Europe
(25%), Asia-Pacific (15%)
```

Python Synthetic Data Generator Script

```
import pandas as pd import numpy as np from faker import Faker from datetime import datetime, timedelta import random fake =
Faker(['en_US', 'de_DE', 'en_GB']) def generate_kna1_master(num_customers=500000): """Generate KNA1 customer master data"""
customers = [] for i in range(num_customers): kunnr = f"{i:010d}" country = np.random.choice(['US', 'DE', 'GB', 'FR', 'SG'],
p=[0.6, 0.15, 0.1, 0.08, 0.07]) ktokd = np.random.choice(['0001', '0002', '0003'], p=[0.7, 0.2, 0.1]) customers.append({
'KUNNR': kunnr, 'NAME1': fake.company(), 'LAND1': country, 'PSTLZ': fake.zipcode(), 'ORT01': fake.city(), 'KTOKD': ktokd,
'ERDAT': (datetime.now() - timedelta(days=random.randint(1, 2000))).strftime('%Y%m%d'), 'BRSCH': np.random.choice(['1200',
'2900', '5100', '7100'], p=[0.3, 0.25, 0.25, 0.2]) }) return pd.DataFrame(customers) def generate_vbak_orders(num_days=30,
orders_per_day=5000, customer_df=None): """Generate VBAK sales order headers with CDC timestamps""" orders = [] start_date =
datetime.now() - timedelta(days=num_days) for day in range(num_days): current_date = start_date + timedelta(days=day)
daily_orders = orders_per_day + np.random.randint(-500, 500) # Variance for hour in range(8, 20): # Business hours
hourly_orders = int(daily_orders / 12) + np.random.randint(-20, 20) for _ in range(hourly_orders): vbeln = f"{len(orders) +
1000000:010d}" erdat = current_date.strftime('%Y%m%d') erzet = f"{hour:02d}{np.random.randint(0,
60):02d}{np.random.randint(0, 60):02d}" kunnr = customer_df.sample(1)['KUNNR'].values[0] if customer_df is not None else
f"{np.random.randint(1, 500000):010d}" auart = np.random.choice(['OR', 'ZOR', 'QT'], p=[0.7, 0.2, 0.1]) vkorg =
np.random.choice(['1000', '2000', '3000'], p=[0.6, 0.3, 0.1]) netwr = round(np.random.lognormal(8, 1.5), 2) # Log-normal
distribution gbstk = np.random.choice(['A', 'B', 'C'], p=[0.4, 0.35, 0.25]) orders.append({ 'VBELN': vbeln, 'ERDAT': erdat,
'ERZET': erzet, 'AUART': auart, 'VKORG': vkorg, 'VTWEG': '10', 'SPART': '00', 'KUNNR': kunnr, 'NETWR': netwr, 'WAERK': 'USD'
if vkorg == '1000' else 'EUR', 'GBSTK': gbstk, 'AEDAT': erdat if np.random.random() < 0.3 else None # 30% changed later })
return pd.DataFrame(orders) def generate_vbap_items(vbak_df, avg_items_per_order=5): """Generate VBAP sales order items
linked to VBAK""" items = [] for _, order in vbak_df.iterrows(): num_items = max(1,
int(np.random.poisson(avg_items_per_order))) for item_num in range(1, num_items + 1): posnr = f"{item_num * 10:05d}" matnr =
f"{np.random.randint(1, 100000):018d}" kwmeng = round(np.random.lognormal(3, 1.2), 3) netpr = round(np.random.uniform(10,
5000), 2) items.append({ 'VBELN': order['VBELN'], 'POSNR': posnr, 'MATNR': matnr, 'KWMENG': kwmeng, 'VRKME': 'EA', 'NETWR':
round(kwmeng * netpr, 2), 'WAERK': order['WAERK'], 'ERDAT': order['ERDAT'] }) return pd.DataFrame(items) # Generate
datasets print("Generating synthetic data...") customers_df = generate_kna1_master(500000) orders_df =
generate_vbak_orders(30, 5000, customers_df) items_df = generate_vbap_items(orders_df, 5) # Save to parquet
customers_df.to_parquet('/mnt/user-data/outputs/synthetic_kna1.parquet', compression='snappy')
orders_df.to_parquet('/mnt/user-data/outputs/synthetic_vbak.parquet', compression='snappy')
items_df.to_parquet('/mnt/user-data/outputs/synthetic_vbap.parquet', compression='snappy') print(f"Generated
{len(customers_df)} customers, {len(orders_df)} orders, {len(items_df)} items")
```

Architecture Diagrams

Business Use Cases & Analytics

The platform enables multiple analytics use cases leveraging the curated SAP SD data. Each use case is implemented as a Gold layer data mart with pre-computed metrics and optimized for specific business questions.

Use Case 1: Sales Performance Analytics

Business Question: What are our daily/weekly/monthly sales trends by product, customer, and region?
Data Sources: VBAK, VBAP, VBRK, VBRP, KNA1, MARA
Key Metrics: Gross sales, net sales, order count, average order value, YoY growth
Gold Layer Tables: fact_sales_daily, fact_sales_monthly, dim_customer, dim_product
Refresh Frequency: Daily (incremental)
Power BI Dashboard: Sales Executive Dashboard with drill-down by product hierarchy and geography

Use Case 2: Order-to-Cash Cycle Analysis

Business Question: How long does it take from order creation to invoice payment? Where are bottlenecks?
Data Sources: VBAK, LIKP, VBRK, VBFA (document flow), payment data
Key Metrics: Days to deliver, days to invoice, cash collection days, blocked order percentage
Gold Layer Tables: fact_order_to_cash, dim_status_reasons
Refresh Frequency: Daily
Power BI Dashboard: Operations Dashboard showing cycle time trends and exception monitoring

Use Case 3: Customer Analytics & Segmentation

Business Question: Who are our most valuable customers? Which customers are at risk of churn?
Data Sources: VBAK, VBAP, VBRK, KNA1, KNVV, CRM opportunity data
Key Metrics: Customer lifetime value (CLV), recency-frequency-monetary (RFM) scores, churn propensity
Gold Layer Tables: fact_customer_orders, dim_customer_segment, kpi_customer_analytics
Refresh Frequency: Weekly
Power BI Dashboard: Customer Intelligence Dashboard with segmentation and retention analysis

Use Case 4: Product Performance & Profitability

Business Question: Which products drive revenue and profit? Which should be discontinued?
Data Sources: VBAP, VBRP, MARA, MARC, KONV (pricing conditions), cost data
Key Metrics: Revenue by product, units sold, margin percentage, inventory turnover, pricing variance
Gold Layer Tables: fact_product_sales, dim_product_hierarchy, kpi_product_profitability
Refresh Frequency: Daily
Power BI Dashboard: Product Management Dashboard with profitability analysis and pricing insights

Use Case 5: Sales Forecasting with ML

Business Question: What will our sales be next quarter? Which products will have highest demand?
Data Sources: Historical sales (VBAK, VBAP), economic indicators, seasonality, promotions
Key Metrics: Forecasted sales volume, confidence intervals, forecast accuracy (MAPE)
Gold Layer Tables: fact_sales_history (time series), dim_date (calendar features), features_sales_ml
Refresh Frequency: Weekly (model retraining monthly)
ML Model: Prophet or ARIMA for time series, XGBoost for demand prediction
Power BI Dashboard: Sales Forecast Dashboard with actual vs forecast comparison

Use Case 6: RAG Chatbot - Conversational Analytics

Business Question: Enable business users to ask questions in natural language
Sample Questions:

- "What were total sales for customer Acme Corp last quarter?"
- "Show me top 10 products by revenue in Europe"
- "Why did sales drop 15% in the Western region last month?"
- "Which customers haven't ordered in the last 90 days?"

Data Sources: All Gold layer tables (indexed in Azure Cognitive Search)
LLM Model: GPT-4 with RAG (retrieval-augmented generation)
Security: Row-level security applied, PII redaction, Azure OpenAI private deployment
Interface: Web app with chat interface, Power BI Q&A; visual, Microsoft Teams bot

Implementation Summary

This enhanced solution design provides a complete blueprint for implementing an enterprise-grade SAP SD Sales Analytics platform on Azure. The architecture leverages: **Key Technical Components:** • SAP-native CDC mechanisms (CDHDR/CDPOS, application timestamps) for efficient incremental extraction • Medallion architecture (Bronze-Silver-Gold) with clear separation of concerns • Master vs Transaction data segregation for optimized processing patterns • Delta Lake for ACID compliance and time travel capabilities • Comprehensive folder structure in ADLS Gen2 for organized data management • Synthetic data generation for development and testing environments **Business Value:** • Real-time analytics on sales orders, deliveries, and billing • Customer segmentation and lifetime value analysis • Product performance and profitability insights • Predictive sales forecasting with machine learning • Conversational analytics through RAG-powered chatbot **Operational Excellence:** • Automated CDC-based incremental loads minimize SAP system impact • Watermark management ensures data consistency and reliability • Comprehensive monitoring and alerting for proactive issue resolution • Infrastructure-as-code for repeatable deployments across environments The 26-week implementation roadmap provides a phased approach to deliver incremental value while managing complexity and risk. The platform is designed to scale with growing data volumes and evolving business requirements.

Next Steps: 1. **SAP System Assessment:** Validate CDC mechanisms availability, RFC connectivity, authorization objects 2. **Azure Environment Setup:** Provision subscriptions, resource groups, and network infrastructure 3. **Synthetic Data Generation:** Create test datasets for development using provided scripts 4. **POC Development:** Build end-to-end pipeline for one table (VBAK) to validate architecture 5. **Stakeholder Alignment:** Present design to SAP Basis team, business users, and security team 6. **Project Kickoff:** Assemble team, finalize timeline, and begin Phase 1 implementation