

Chinook Data from Azure SQL to Snowflake DW using ADF

End-to-end ETL pipeline using **Azure Data Factory** to migrate Chinook music store data from **Azure SQL Database** → **Azure Blob Storage** (Parquet) → **Snowflake Data Warehouse** with dimensional modeling.

Github Link: https://github.com/vedantmane12/chinook_movie_data_AzureSQL_to_Snowflake_Data_PL

Azure User: mane.v@northeastern.edu

Subscription: Azure for Students

Resource Group: dadabi

Region: US West 2

Key Vault: kvault-damg7370

Storage Account: storagedamg7370

Database: DAMG7370FALL2025 (dada-bi.database.windows.net)

User: damg7370

Data Factory (V2): adfdemo-damg7370

▼ Linked Services

Azure SQL: ls_azuresqldb

Azure Key Vault: ls_azurekeyvault

Azure BLOB Storage: ls_azureblobstorage

Snowflake: ls_snowflake

▼ Dataset

Azure SQL: chinook_ds

Azure BLOB Storage: chinook_ds_parquet

Snowflake: chinook_ds_snowflake

▼ Pipelines

- `extract_SQLDB_PL` → Extract Azure SQL to Parquet files
- `stage_Partquet_PL` → Load Parquet to Snowflake STAGE
- `load_ALBUM_DIM_PL` → Load album dimension
- `load_ARTIST_DIM_PL` → Load artist dimension
- `load_CUSTOMER_DIM_PL` → Load customer dimension
- `load_GENRE_DIM_PL` → Load genre dimension
- `load_INVOICE_DIM_PL` → Load invoice dimension
- `load_INVOICELINE_DIM_PL` → Load invoice line dimension
- `load_SALES_FACT_PL` → Load sales fact table

▼ Dataflows

- `transform_source_parquet` → Transform Parquet files and add audit columns before loading to Snowflake STAGE
- `df_load_ALBUM_DIM` → Extract from STAGE.Album, generate surrogate keys, and load to DW.ALBUM_DIM
- `df_load_ARTIST_DIM` → Extract from STAGE.Artist, generate surrogate keys, and load to DW.ARTIST_DIM
- `df_load_CUSTOMER_DIM` → Extract from STAGE.Customer, generate surrogate keys, and load to DW.CUSTOMER_DIM

- `df_load_GENRE_DIM` → Extract from STAGE.Genre, generate surrogate keys, and load to DW.GENRE_DIM
- `df_load_INVOICE_DIM` → Extract from STAGE.Invoice, generate surrogate keys, and load to DW.INVOICE_DIM
- `df_load_INVOICELINE_DIM` → Extract from STAGE.InvoiceLine, generate surrogate keys, and load to DW.INVOICELINE_DIM

Azure SQL Database

DAMG7370FALL2025 (dada-bi/DAMG7370FALL2025) | Query editor (preview)

Query 1

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

Tables

- chinook.Album
- chinook.Artist
- chinook.Customer
- chinook.Employee
- chinook.Genre
- chinook.Invoice
- chinook.InvoiceLine
- chinook.MediaType
- chinook.Playlist
- chinook.PlaylistTrack
- chinook.Track

Key Vault

Key vaults <> **kvault-damg7370** **Key vault**

You are viewing a new version of Browse experience. Click here to access the old experience.

Name **kvault-damg7370**

Overview

- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Access policies
- Resource visualizer
- Events
- Objects
- Settings
- Monitoring
- Automation
- Help

Essentials

Resource group (move) dadabi	Vault URI https://kvault-damg7370.vault.azure.net/
Location West US 2	Sku (Pricing tier) Standard
Subscription (move) Azure for Students	Directory ID a8ec281-aaa3-4dae-ac9b-9a398b9215e7
Subscription ID e20d3eb4-d2f4-492a-8186-6bebe3fb7941	Directory Name Northeastern University
	Soft-delete Enabled
	Purge protection Disabled

Tags [\(edit\)](#)
application : damg7370fall2025 environment : demo

Get started Properties Monitoring Tools + SDKs Tutorials

Manage keys and secrets used by apps and services

Showing 1 - 1 of 1. Display [auto](#) count:

Add or remove favorites by pressing **Cmd + Shift + F**

Azure BLOB Storage Account

Home > **Storage center | Storage accounts (Blobs)** >

storagedamg7370 **Storage account**

Search **Upload** **Open in Explorer** **Delete** **Move** **Refresh** **Open in mobile** **CLI / PS** **Feedback**

Overview

- Activity log
- Tags
- Diagnose and solve problems
- Access Control (IAM)
- Data migration
- Events
- Storage browser
- Storage Mover
- Partner solutions

Properties

Resource group (move) : dadabi	Performance : Standard
Location : westus2	Replication : Locally-redundant storage (LRS)
Subscription (move) : Azure for Students	Account kind : StorageV2 (general purpose v2)
Subscription ID : e20d3eb4-d2f4-492a-8186-6bebe3fb7941	Provisioning state : Succeeded
Disk state : Available	Created : 10/10/2025, 2:04:16 PM

Blob service

Hierarchical namespace	Disabled	Require secure transfer for REST API operations	Enabled
Default access tier	Hot	Storage account key access	Enabled
Blob anonymous access	Disabled	Minimum TLS version	Version 1.2
Blob soft delete	Enabled (7 days)	Infrastructure encryption	Disabled
Container soft delete	Enabled (7 days)		
Versioning	Disabled		
Change feed	Disabled	Public network access	Enabled
NFS v3	Disabled	Public network access scope	Enable from all networks
Allow cross-tenant replication	Disabled	Private endpoint connections	0
Storage tasks assignments	None	Network routing	Microsoft network routing
		Endpoint type	Standard

Security

Networking

Add or remove favorites by pressing **Cmd + Shift + F**

[File service](#)

Azure Data Factory Setup

The screenshot shows the Microsoft Azure Data Factory Overview page for a deployment named "Microsoft.DataFactory-20251010140708". The deployment status is marked as "complete". Key details include:

- Deployment name:** Microsoft.DataFactory-20251010140708
- Subscription:** Azure for Students
- Resource group:** dadabi
- Start time:** 10/10/2025, 2:11:38 PM
- Correlation ID:** 30c1999-7ecd-4826-9c2b-58880d556576

The "Deployment details" section lists one resource: "adfdemo-damg7370" (Data factory V2) with status "OK".

On the right side, there are links for "Cost management", "Microsoft Defender for Cloud", "Free Microsoft tutorials", "Work with an expert", and "Next steps" with a "Go to resource" button.

Add or remove favorites by pressing Cmd + Shift + F

ADF Linked Services

The screenshot shows the Microsoft Azure Data Factory Linked services page for the "adfdemo-damg7370" factory. The left sidebar shows navigation options like General, Connections, and Author. The main area displays a list of linked services:

Name	Type	Related	Annotations
ls_azureblobstorage	Azure Blob Storage	0	
ls_azurekeyvault	Azure Key Vault	2	
ls_azuresqldb	Azure SQL Database	0	
ls_snowflake	Snowflake V2	0	

Datasets in ADF

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists Pipelines, Change Data Capture (p...), Datasets (with 'chinook_ds' selected), Data flows, and Power Query. The main area displays the 'chinook_ds' dataset, which is an Azure SQL Database. The 'Properties' panel on the right shows the dataset's name as 'chinook_ds' and its type as 'Azure SQL Database'. The 'Annotations' section is empty. Below the dataset details, there is a configuration pane for 'Connection', 'Schema', and 'Parameters'. Under 'Connection', the 'Linked service' dropdown is set to 'ls_azuresqldb', and the 'Test connection' status is 'Connection successful'. The 'Table' field contains the expression '@dataset().schema_name . @dataset().table_name'. A 'Preview data' button is present.

Azure Linked Service Parameters

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar navigation bar includes General, Connector upgrade advisor, Factory settings, Connections (selected), Integration runtimes, Microsoft Purview, ADF in Microsoft Fabric, Source control, Git configuration, ARM template, Author, Triggers, Global parameters, Data flow libraries, Security, Credentials, Customer managed key, Outbound rules, and Managed private endpoints. The main area shows the 'Linked services' section, listing four entries: ls_azureblobstorage (Azure Blob), ls_azurekeyvault (Azure Key Vault), ls_azuresqldb (Azure SQL), and ls_snowflake (Snowflake). The 'ls_azurekeyvault' entry is selected. The right-hand panel displays the 'Edit linked service' configuration for 'ls_azurekeyvault'. It shows the 'Account selection method' as 'Enter manually', 'Fully qualified domain name' as '@{linkedService().servername}', 'Database name' as '@{linkedService().databasename}', 'Authentication type' as 'SQL authentication', 'User name' as '@{linkedService().username}', and 'Secret name' as 'ls_azurekeyvault'. Other fields include 'AKV linked service' (set to 'ls_azurekeyvault'), 'Secret version' (set to 'Latest version'), 'Always encrypted' (unchecked), 'Encrypt' (unchecked), and a 'Test connection' button.

ADF Pipelines

Copy Azure SQL to BLOB Storage

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, Data flows, and Power Query. The main panel displays the 'extract_SQLDB_PL' pipeline details. Under the 'Activities' tab, the 'Output' section shows a table of 12 items from a recent run. The table includes columns for Activity name, Activity status, Activit..., Run start, Duration, and Integration runtime. All activities show a green checkmark indicating success.

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:04:15 AM	20s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:04:11 AM	18s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:03:56 AM	18s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:03:56 AM	21s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:03:51 AM	18s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:03:34 AM	21s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:03:32 AM	18s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:03:12 AM	20s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:03:12 AM	19s	AutoResolveIntegrationRuntime (V)
sql_2_parquet	Succeeded	Copy data	10/11/2025, 12:03:12 AM	20s	AutoResolveIntegrationRuntime (V)
ForEachSourceTable	Succeeded	ForEach	10/11/2025, 12:03:11 AM	1m 26s	

The screenshot shows the Microsoft Azure Storage Container overview for 'stagedata'. The container contains a single folder named 'chinook'. The table below lists 11 items (blobs) within the 'chinook' folder, all of which are parquet files. The columns include Name, Last modified, Access tier, Blob type, Size, and Lease state.

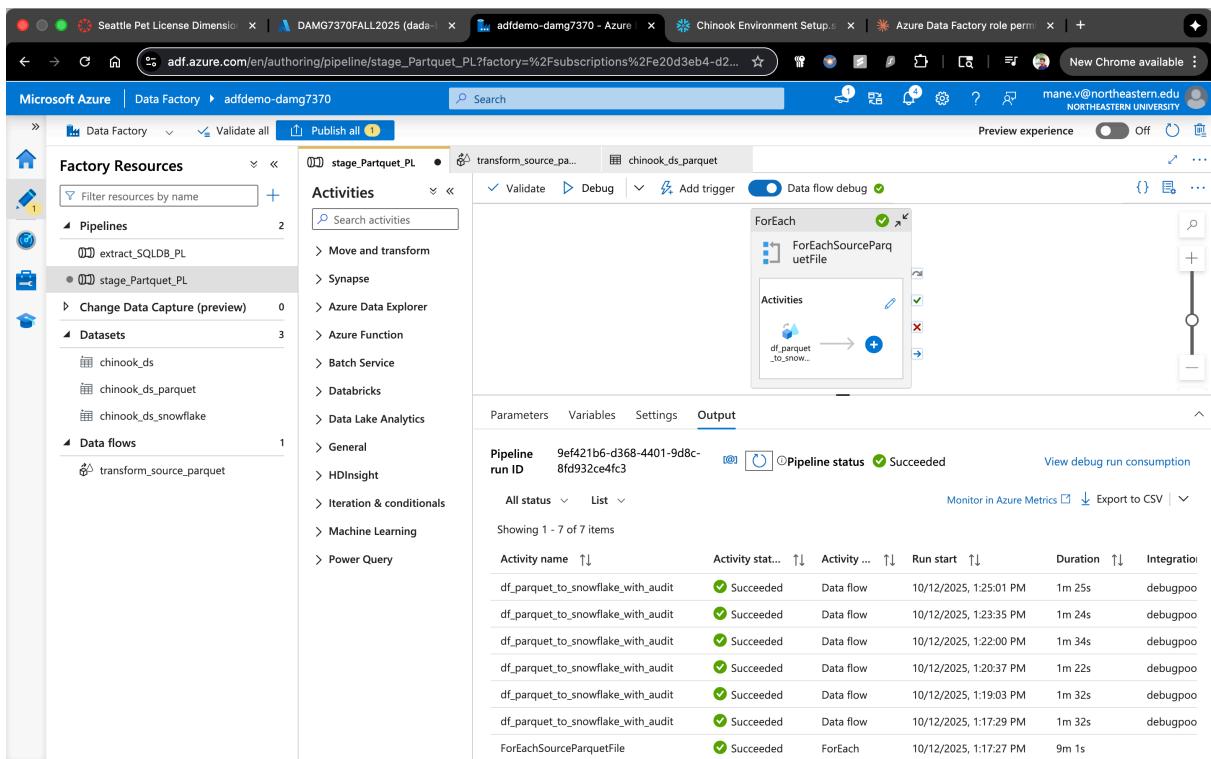
Name	Last modified	Access tier	Blob type	Size	Lease state
..					
Album.parquet	10/11/2025, 12:03:27 AM	Hot (Inferred)	Block blob	9.4 KiB	Available
Artist.parquet	10/11/2025, 12:03:29 AM	Hot (Inferred)	Block blob	6.49 KiB	Available
Customer.parquet	10/11/2025, 12:03:27 AM	Hot (Inferred)	Block blob	8.9 KiB	Available
Employee.parquet	10/11/2025, 12:03:47 AM	Hot (Inferred)	Block blob	3.73 KiB	Available
Genre.parquet	10/11/2025, 12:03:53 AM	Hot (Inferred)	Block blob	892 B	Available
Invoice.parquet	10/11/2025, 12:03:52 AM	Hot (Inferred)	Block blob	13.98 KiB	Available
InvoiceLine.parquet	10/11/2025, 12:04:08 AM	Hot (Inferred)	Block blob	22.08 KiB	Available
MediaType.parquet	10/11/2025, 12:04:11 AM	Hot (Inferred)	Block blob	647 B	Available
Playlist.parquet	10/11/2025, 12:04:14 AM	Hot (Inferred)	Block blob	788 B	Available
PlaylistTrack.parquet	10/11/2025, 12:04:26 AM	Hot (Inferred)	Block blob	20.24 KiB	Available
Track.parquet	10/11/2025, 12:04:33 AM	Hot (Inferred)	Block blob	116.12 KiB	Available

Add or remove favorites by pressing Cmd + Shift + F

Azure BLOB Storage to Snowflake Staging with Data Column Transformations

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists pipelines, datasets, and data flows. In the center, the 'Activities' section displays a 'ForEach' activity. The 'Activities' tab is selected, showing a single child activity named 'ForEachSourceParquetFile'. Below the activities, the 'General' tab is active, showing the activity's name as 'ForEachSourceParquetFile' and its state as 'Activated'. The 'Description' and 'Activity state' sections are also visible.

The screenshot shows the Microsoft Azure Data Factory data flow editor. The 'Factory Resources' sidebar is visible on the left. The main area displays a data flow with three main components: 'SourceParquet', 'AddAuditColumns', and 'SnowflakeSink'. The 'SourceParquet' component is connected to the 'AddAuditColumns' component, which is then connected to the 'SnowflakeSink' component. Below the data flow, the 'Parameters' tab is selected, showing a 'New' button.



Load DATE_DIM into SNOWFLAKE WH

```

INSERT INTO DW.DATE_DIM
WITH DATES AS (
    SELECT
        DATEADD(DAY, SEQ4(), DATE '2000-01-01') AS DATE_KEY
    FROM TABLE(GENERATOR(ROWCOUNT => 365*30))
)
SELECT
    TO_CHAR(DATE_KEY, 'YYYYMMDD') DATE_KEY,
    DATE_KEY FULL_DATE,
    DAY(DATE_KEY) DAY_NUM,
    TO_CHAR(DATE_KEY, 'DY') WEEKDAY_ABBR,
    DAYOFWEEK(DATE_KEY) WEEKDAY_NUM,
    DAYOFYEAR(DATE_KEY) DAY_OF_YEAR_NUM,
    WEEK(DATE_KEY) WEEK_OF_YEAR,
    MONTH(DATE_KEY) MONTH_NUM,
    TO_CHAR(DATE_KEY, 'Mon') MONTH_ABBR,
    QUARTER(DATE_KEY) QUARTER_NUM,
    CASE QUARTER(DATE_KEY)
        WHEN 1 THEN 'Q1'
        WHEN 2 THEN 'Q2'
        WHEN 3 THEN 'Q3'
        WHEN 4 THEN 'Q4'
    END QUARTER_NAME,
    YEAR(DATE_KEY) YEAR_NUM,
    DATE_TRUNC('MONTH', DATE_KEY) FIRST_DAY_OF_MONTH,
    LAST_DAY(DATE_KEY) LAST_DAY_OF_MONTH,
    CASE WHEN DAYOFWEEK(DATE_KEY) IN (0, 6)
        THEN 'Y'
        ELSE 'N'
    END DAY_OF_WEEK
)

```

```
END IS_WEEKEND
FROM DATES;
```

CHINOOK_DB / DW / DATE_DIM

	DATE_KEY	FULL_DATE	DAY_NUM	WEEKDAY_ABBR	WEEKDAY_NUM	DAY_OF_YEAR_NUM	WEEK_OF_YEAR
1	20000101	2000-01-01	1	Sat	6	1	52
2	20000102	2000-01-02	2	Sun	0	2	52
3	20000103	2000-01-03	3	Mon	1	3	1
4	20000104	2000-01-04	4	Tue	2	4	1
5	20000105	2000-01-05	5	Wed	3	5	1
6	20000106	2000-01-06	6	Thu	4	6	1
7	20000107	2000-01-07	7	Fri	5	7	1
8	20000108	2000-01-08	8	Sat	6	8	1
9	20000109	2000-01-09	9	Sun	0	9	1
10	20000110	2000-01-10	10	Mon	1	10	2
11	20000111	2000-01-11	11	Tue	2	11	2
12	20000112	2000-01-12	12	Wed	3	12	2
13	20000113	2000-01-13	13	Thu	4	13	2
14	20000114	2000-01-14	14	Fri	5	14	2
15	20000115	2000-01-15	15	Sat	6	15	2
16	20000116	2000-01-16	16	Sun	0	16	2
17	20000117	2000-01-17	17	Mon	1	17	3
18	20000118	2000-01-18	18	Tue	2	18	3

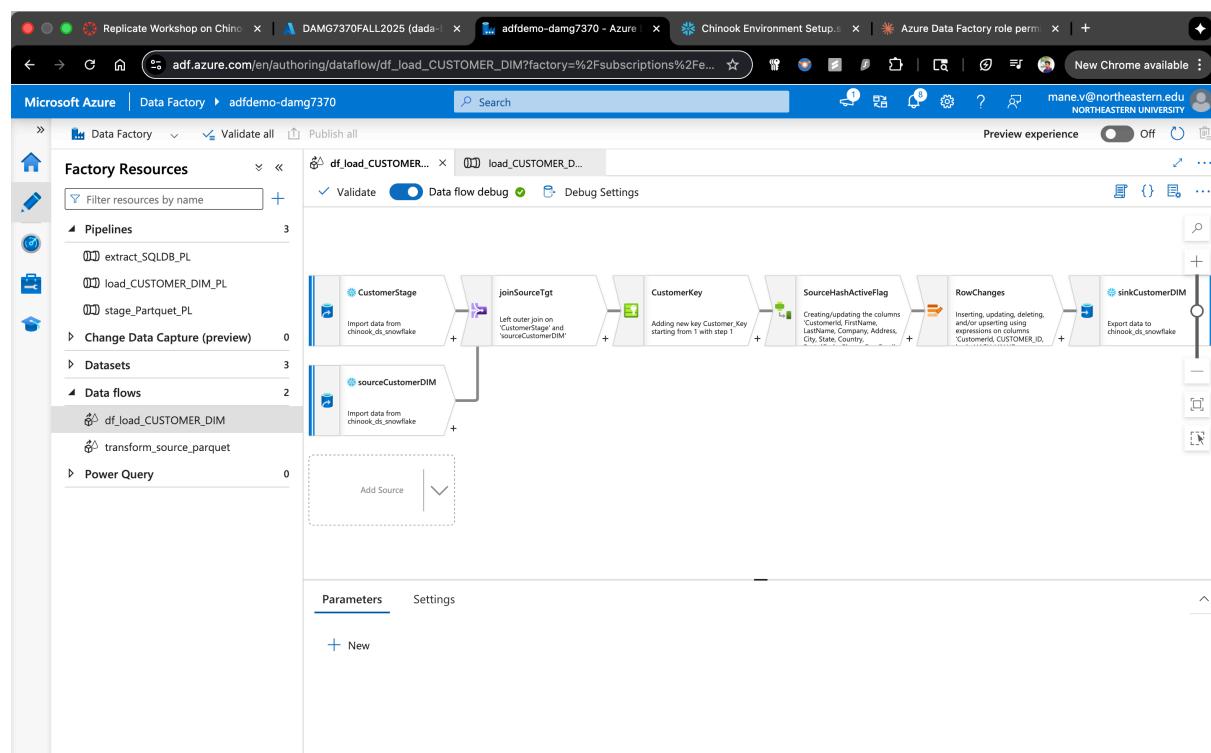
Load TIME_DIM into Snowflake WH

```
INSERT INTO DW.TIME_DIM
WITH H AS (
    SELECT SEQ4() AS Hr FROM TABLE(GENERATOR(ROWCOUNT => 24))
),
M AS (
    SELECT SEQ4() AS Mi FROM TABLE(GENERATOR(ROWCOUNT => 60))
)
SELECT
    LPAD(H.Hr, 2, '0') || LPAD(M.Mi, 2, '0') AS TIME_KEY,
    H.Hr,
    M.Mi,
    LPAD(H.Hr, 2, '0') || ':' || LPAD(M.Mi, 2, '0') AS TIME_24_HR
FROM H, M;
```

CHINOOK_DB / DW / TIME_DIM

	TIME_KEY	...	HOUR_NUM	MINUTE_NUM	TIME_24_HR
1	0		0	0	00:00
2	100		1	0	01:00
3	200		2	0	02:00
4	300		3	0	03:00
5	400		4	0	04:00
6	500		5	0	05:00
7	600		6	0	06:00
8	700		7	0	07:00
9	800		8	0	08:00
10	900		9	0	09:00
11	1000		10	0	10:00
12	1100		11	0	11:00
13	1200		12	0	12:00
14	1300		13	0	13:00
15	1400		14	0	14:00
16	1500		15	0	15:00
17	1600		16	0	16:00
18	1700		17	0	17:00

Load CUSTOMER_DIM from Snowflake STAGE



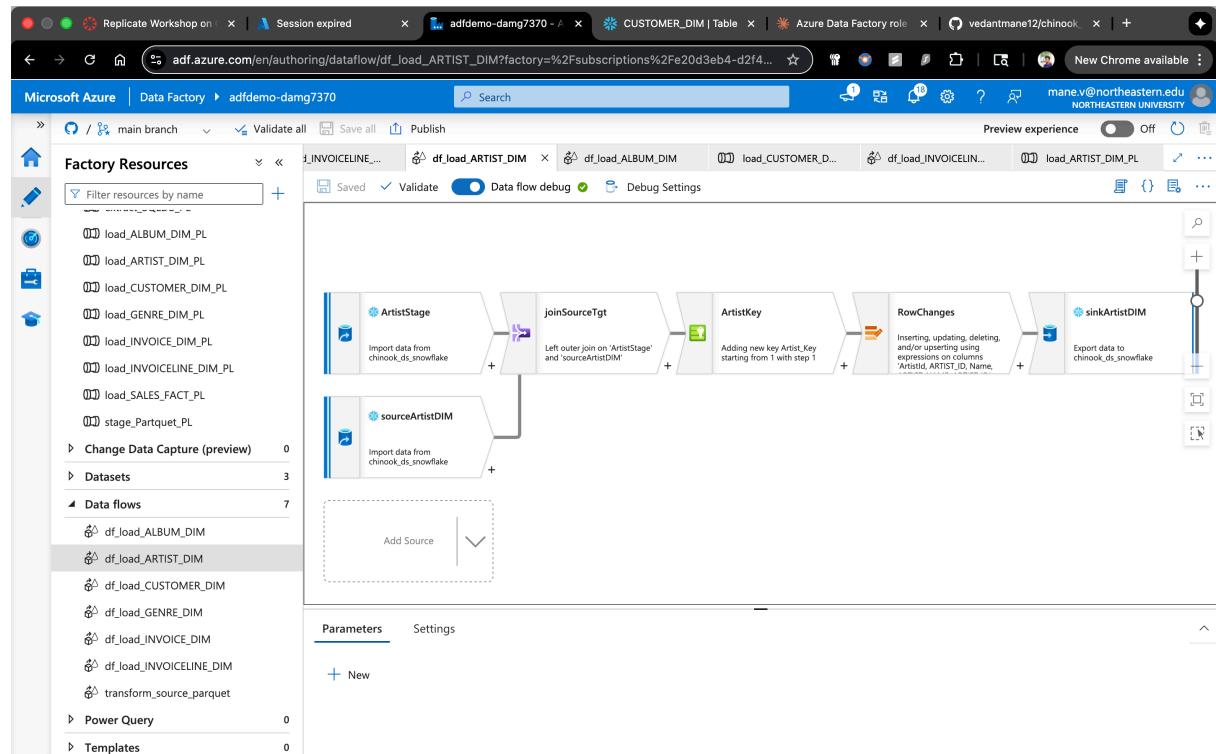
CHINOOK_DB / DW / CUSTOMER_DIM

Table Details Columns Data Preview Copy History Data Quality PREVIEW Lineage

COMPUTE_WH 59 Rows • Updated just now

	CUSTOMER_KEY	CUSTOMER_ID	FIRST_NAME	LAST_NAME	COMPANY_NAME	CITY
1	1	55	Mark	Taylor	null	Sidney
2	2	21	Kathy	Chase	null	Reno
3	3	59	Puja	Srivastava	null	Bangal
4	4	16	Frank	Harris	Google Inc.	Mounta
5	5	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	São Jo
6	6	51	Joakim	Johansson	null	Stockh
7	7	20	Dan	Miller	null	Mounta
8	8	17	Jack	Smith	Microsoft Corporation	Redmo
9	9	56	Diego	Gutiérrez	null	Buenos
10	10	43	Isabelle	Mercier	null	Dijon
11	11	42	Wyatt	Girard	null	Bordea
12	12	48	Johannes	Van der Berg	null	Amster
13	13	10	Eduardo	Martins	Woodstock Discos	São Pa
14	14	22	Heather	Leacock	null	Orland
15	15	14	Mark	Philips	Telus	Edmon
16	16	33	Ellie	Sullivan	null	Yellowk
17	17	26	Richard	Cunningham	null	Fort Wo
18	18	9	Kara	Nielsen	null	Copenf

Load ARTIST_DIM from Snowflake STAGE



CHINOOK_DB / DW / ARTIST_DIM

Table Details Columns Data Preview Copy History Data Quality PREVIEW Lineage

COMPUTE_WH 100 of 275 Rows • Updated just now

	ARTIST_KEY	ARTIST_ID	ARTIST_NAME	SOURCE_ID
1	1	75	Vinicius, Toquinho & Quarteto Em Cy	ADF_PIPELINE
2	2	189	Otto	ADF_PIPELINE
3	3	215	Academy of St. Martin in the Fields Chamber Ensemble & Sir Neville Marriner	ADF_PIPELINE
4	4	239	Sir Neville Marriner & William Bennett	ADF_PIPELINE
5	5	182	Nega Gizza	ADF_PIPELINE
6	6	55	David Coverdale	ADF_PIPELINE
7	7	176	The Flaming Lips	ADF_PIPELINE
8	8	126	Raul Seixas	ADF_PIPELINE
9	9	252	Amy Winehouse	ADF_PIPELINE
10	10	21	Various Artists	ADF_PIPELINE
11	11	59	Santana	ADF_PIPELINE
12	12	108	Mônica Mariano	ADF_PIPELINE
13	13	269	Michele Campanella	ADF_PIPELINE
14	14	218	Orchestre Révolutionnaire et Romantique & John Eliot Gardiner	ADF_PIPELINE
15	15	229	Boston Symphony Orchestra & Seiji Ozawa	ADF_PIPELINE
16	16	140	The Doors	ADF_PIPELINE
17	17	260	Adrian Laper & Doreen de Feis	ADF_PIPELINE
18	18	129	Simply Red	ADF_PIPELINE

Load ALBUM_DIM from Snowflake STAGE

Microsoft Azure | Data Factory | adfdemo-damg7370

main branch | Validate all | Save all | Publish

Factory Resources | Filter resources by name

Data flows | df_load_ALBUM_DIM

load_ALBUM_DIM... | df_load_ARTIST_DIM | df_load_ALBUM_DIM | load_CUSTOMER_DIM... | df_load_INVOICELIN... | load_ARTIST_DIM_PL

load_ALBUM_DIM_PL | load_ARTIST_DIM_PL | load_CUSTOMER_DIM_PL | load_GENRE_DIM_PL | load_INVOICE_DIM_PL | load_INVOICELINE_DIM_PL | load_INVOICE_PL | load_SALES_FACT_PL | stage_Partquet_PL

df_load_ALBUM_DIM | df_load_ARTIST_DIM | df_load_CUSTOMER_DIM | df_load_GENRE_DIM | df_load_INVOICE_DIM | df_load_INVOICELINE_DIM | transform_source_parquet

Parameters | Settings

Add Source

AlbumDIM | joinSourceTgt | AlbumKey | RowChanges | sinkAlbumDIM

sourceAlbumDIM

Database Explorer

CHINOOK_DB / DW / ALBUM_DIM

Table Details Columns Data Preview Copy History Data Quality PREVIEW Lineage

COMPUTE_WH 100 of 347 Rows • Updated just now

	ALBUM_KEY	ALBUM_ID	TITLE	ARTIST_ID
1	1	75	Angel Dust	82
2	2	189	New Adventures in Hi-Fi	124
3	3	215	The Police Greatest Hits	141
4	4	239	War	150
5	5	325	Bartok: Violin & Viola Concertos	255
6	6	334	Weill: The Seven Deadly Sins	264
7	7	182	Vs.	118
8	8	55	Chronicle, Vol. 2	76
9	9	176	Original Soundtracks 1	116
10	10	126	Unplugged Live	52
11	11	252	Un-Led-Ed	157
12	12	309	Palestrina: Missa Papae Marcelli & Allegri: Miserere	244
13	13	21	Prenda Minha	16
14	14	59	Deep Purple In Rock	58
15	15	108	Rock In Rio CD1	90
16	16	269	Temple of the Dog	204
17	17	291	Puccini: Madama Butterfly - Highlights	225
18	18	337	Szymanowski: Piano Works, Vol. 1	266

Load GENRE_DIM from Snowflake STAGE

Microsoft Azure | Data Factory | adfdemo-damg7370

main branch / df_load_ALBUM_DIM validate all Save all Publish

Preview experience Off

Factory Resources

- Pipelines: extract_SQLDB_PL, load_ALBUM_DIM_PL, load_ARTIST_DIM_PL, load_CUSTOMER_DIM_PL, load_GENRE_DIM_PL, load_INVOICE_DIM_PL, load_INVOICELINE_DIM_PL, load_SEALES_FACT_PL, stage_Partquet_PL
- Data flows: df_load_ALBUM_DIM, df_load_ARTIST_DIM, df_load_CUSTOMER_DIM, df_load_GENRE_DIM, df_load_INVOICE_DIM, df_load_INVOICELINE_DIM, transform_source_parquet
- Datasets: 3

df_load_GENRE_DIM

Parameters Settings

+ New

```

graph LR
    subgraph Source [sourceGenreDIM]
        direction TB
        S[Import data from chinook_ds_snowflake]
    end
    subgraph Sink [sinkGenreDIM]
        direction TB
        SINK[Export data to chinook_ds_snowflake]
    end
    subgraph Genres [GenreKey]
        direction TB
        G[Adding new key Genre_Key starting from 1 with step 1]
    end
    subgraph Changes [RowChanges]
        direction TB
        C[Inserting, updating, deleting, and/or upserting using expressions on columns Commit (GENRE_ID)]
    end
    subgraph Join [joinSourceTgt]
        direction TB
        J[Left outer join on 'GenreDIM' and 'sourceGenreDIM']
    end
    subgraph Import [GenreDIM]
        direction TB
        I[Import data from chinook_ds_snowflake]
    end
    S --> J
    I --> J
    J --> G
    G --> C
    C --> SINK

```

Screenshot of the Snowflake Database Explorer showing the ARTIST_DIM table in the CHINOOK_DB / DW schema.

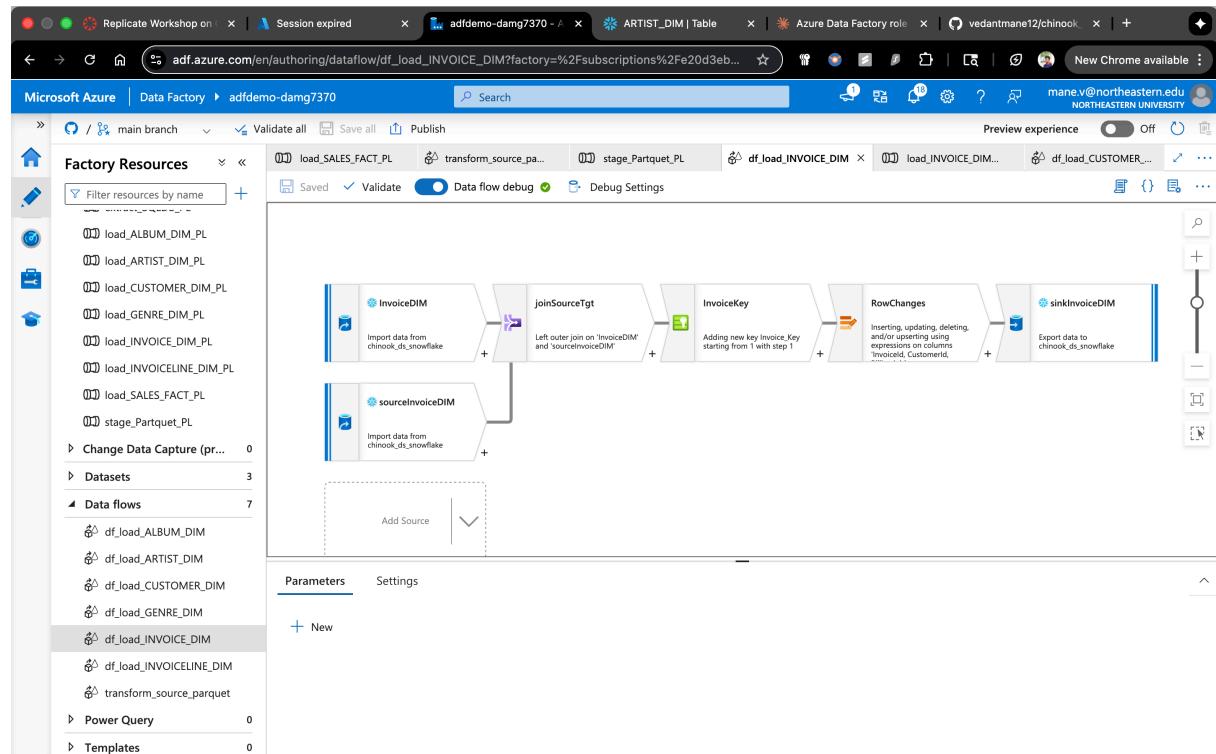
Table Details:

- Table: ARTIST_DIM
- Owner: ACCOUNTADMIN
- Last Updated: 1 day ago
- Rows: 275
- Size: 70KB

Data Preview:

	ARTIST_KEY	ARTIST_ID	ARTIST_NAME	SOURCE_ID
1	1	75	Vinicius, Toquinho & Quarteto Em Cy	ADF_PIPELINE
2	2	189	Otto	ADF_PIPELINE
3	3	215	Academy of St. Martin in the Fields Chamber Ensemble & Sir Neville Marriner	ADF_PIPELINE
4	4	239	Academy of St. Martin in the Fields, Sir Neville Marriner & William Bennett	ADF_PIPELINE
5	5	182	Nega Gizza	ADF_PIPELINE
6	6	55	David Coverdale	ADF_PIPELINE
7	7	176	The Flaming Lips	ADF_PIPELINE
8	8	126	Raul Seixas	ADF_PIPELINE
9	9	252	Amy Winehouse	ADF_PIPELINE
10	10	21	Various Artists	ADF_PIPELINE
11	11	59	Santana	ADF_PIPELINE
12	12	108	Mônica Mariano	ADF_PIPELINE
13	13	269	Michele Campanella	ADF_PIPELINE
14	14	218	Orchestre Révolutionnaire et Romantique & John Eliot Gardiner	ADF_PIPELINE
15	15	229	Boston Symphony Orchestra & Seiji Ozawa	ADF_PIPELINE
16	16	140	The Doors	ADF_PIPELINE
17	17	260	Adrian Laper & Doreen de Feis	ADF_PIPELINE
18	18	129	Simply Red	ADF_PIPELINE

Load INVOICE_DIM from Snowflake STAGE



Screenshot of the Database Explorer in the HORIZON CATALOG showing the CHINOOK_DB / DW / INVOICE_DIM table.

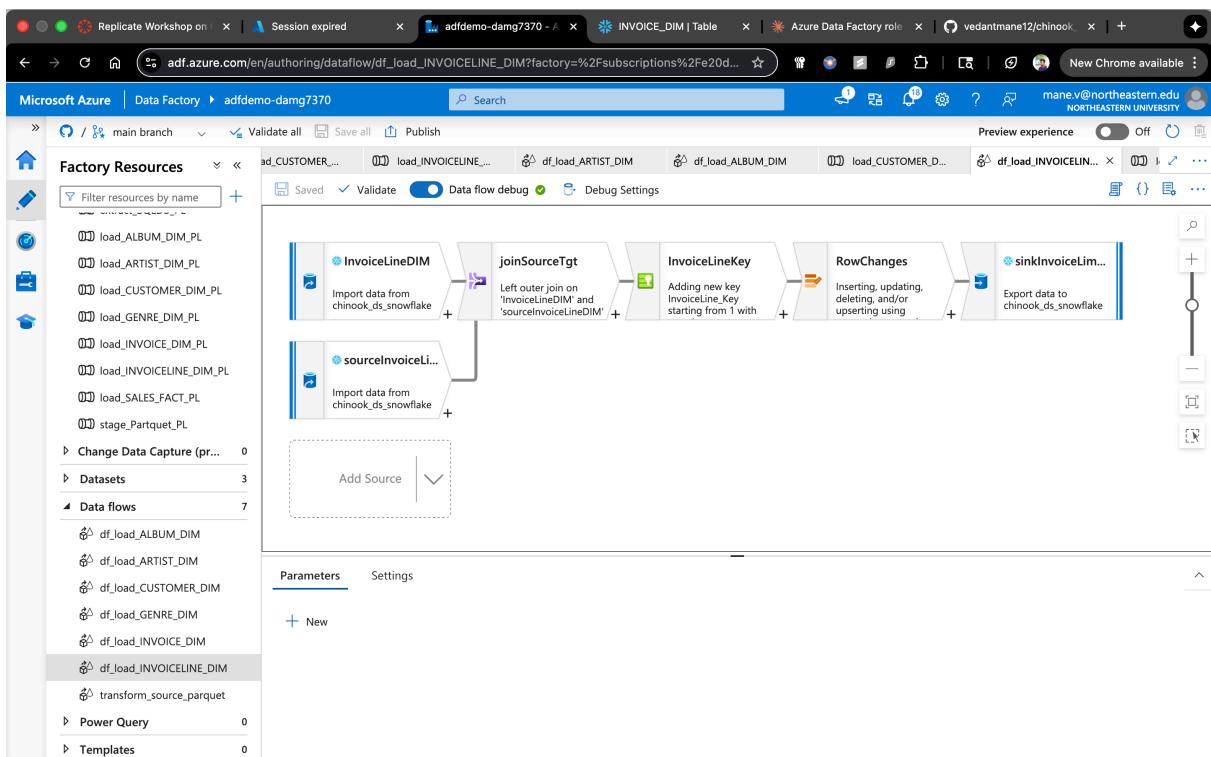
Table Details:

- Table: INVOICE_DIM
- Owner: ACCOUNTADMIN
- Created: 17 hours ago
- Rows: 412
- Size: 13.0KB

Data Preview:

	INVOICE_KEY	INVOICE_ID	CUSTOMER_ID	SALE_DATE	BILLING_ADDRESS	BILLING_CITY
1	1	75	49	2009-11-17	Ordynacka 10	Warsaw
2	2	189	23	2011-04-18	69 Salem Street	Boston
3	3	215	42	2011-07-30	9, Place Louis Barthou	Bordeaux
4	4	239	55	2011-11-21	421 Bourke Street	Sidney
5	5	325	45	2012-11-29	Erzsébet krt. 58.	Budapest
6	6	334	39	2013-01-07	4, Rue Milton	Paris
7	7	182	44	2011-03-18	Porthaninkatu 9	Helsinki
8	8	55	8	2009-08-24	Gretvystraat 63	Brussels
9	9	176	8	2011-02-15	Gretvystraat 63	Brussels
10	10	126	35	2010-07-13	Rua dos Campeões Europeus de Viena, 4350	Porto
11	11	252	11	2012-01-22	Av. Paulista, 2022	São Paulo
12	12	309	22	2012-09-26	120 S Orange Ave	Orlando
13	13	21	55	2009-04-04	421 Bourke Street	Sidney
14	14	59	17	2009-09-08	1 Microsoft Way	Redmond
15	15	108	47	2010-04-13	Via Degli Scipioni, 43	Rome
16	16	269	36	2012-03-26	Tautentienstraße 8	Berlin
17	17	291	38	2012-06-30	Barbarossastraße 19	Berlin
18	18	337	56	2013-01-28	307 Macacha Güemes	Buenos Aires

Load INVOICELINE_DIM from Snowflake STAGE



CHINOOK_DB / DW / INVOICELINE_DIM

	INVOICELINE_KEY	INVOICELINE_ID	INVOICE_ID	TRACK_ID	UNIT_PRICE	QUANTITY	SOURCE_ID
1	1	75	14	463	0.99	1	ADF_PIPELINE
2	2	189	35	1159	0.99	1	ADF_PIPELINE
3	3	215	40	1286	0.99	1	ADF_PIPELINE
4	4	239	45	1424	0.99	1	ADF_PIPELINE
5	5	325	60	1946	0.99	1	ADF_PIPELINE
6	6	334	61	2027	0.99	1	ADF_PIPELINE
7	7	642	117	416	0.99	1	ADF_PIPELINE
8	8	720	132	903	0.99	1	ADF_PIPELINE
9	9	796	146	1367	0.99	1	ADF_PIPELINE
10	10	1171	215	134	0.99	1	ADF_PIPELINE
11	11	1331	246	1116	0.99	1	ADF_PIPELINE
12	12	1543	285	2382	0.99	1	ADF_PIPELINE
13	13	1684	311	3238	1.99	1	ADF_PIPELINE
14	14	1715	317	3444	0.99	1	ADF_PIPELINE
15	15	1890	348	1012	0.99	1	ADF_PIPELINE
16	16	2092	386	2255	0.99	1	ADF_PIPELINE
17	17	2115	390	2377	0.99	1	ADF_PIPELINE
18	18	2195	404	2877	1.99	1	ADF_PIPELINE

Load SALES_FACT from Snowflake WH

Microsoft Azure | Data Factory > adfdemo-damg7370

Activities

Name *	Description	Activity state	Timeout	Retry	Retry interval (sec)	Secure output	Secure input
LoadSALESFACT		Activated	0:00:10:00	0	30		

```
WITH aggsales AS (
SELECT
    cd.CUSTOMER_KEY,
    i.INVOICE_ID AS INVOICE_ID,
    i.SALE_DATE AS SALE_DATE,
```

```

        SUM(il.QUANTITY * il.UNIT_PRICE) AS TOTAL_SALE_AMT
    FROM DW.INVOICE_DIM i
    JOIN DW.INVOICELINE_DIM il ON i.INVOICE_ID = il.INVOICE_ID
    JOIN DW.CUSTOMER_DIM cd ON cd.CUSTOMER_ID = i.CUSTOMER_ID
    GROUP BY
        i.INVOICE_ID,
        cd.CUSTOMER_KEY,
        i.SALE_DATE
    )
    SELECT
        STAGE.SALES_FACT_SEQ.nextval AS SALES_KEY,
        CUSTOMER_KEY,
        INVOICE_ID,
        -- SALE_DATE,
        dd.DATE_KEY AS DATE_DIM_KEY,
        td.TIME_KEY AS TIME_DIM_KEY,
        TOTAL_SALE_AMT,
        'ADF_PIPELINE' AS SOURCE_ID,
        CURRENT_TIMESTAMP() AS DATE_TO_WAREHOUSE
    FROM aggsales a
    JOIN DW.DATE_DIM dd ON dd.FULL_DATE = DATE(a.SALE_DATE)
    JOIN DW.TIME_DIM td ON td.TIME_24_HR = TO_CHAR(a.SALE_DATE, 'HH24:MI')
    WHERE NOT EXISTS (
        SELECT 1
        FROM DW.SALES_FACT sf
        WHERE sf.INVOICE_ID = a.INVOICE_ID
    );

```

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists various data flows and activities. In the center, the 'Activities' pane shows a 'Copy data' activity named 'LoadSALESFACT'. The 'Source' tab is selected, showing the 'chinook_ds_snowflake' dataset with 'DW' as the schema name and 'INVOICE_DIM' as the table name. The 'Query' tab displays the T-SQL code for generating the source data:

```

WITH aggsales AS (
    SELECT
        cd.CUSTOMER_KEY,
        i.INVOICE_ID AS INVOICE_ID,
        i.SALE_DATE AS SALE_DATE,
        SUM(il.QUANTITY * il.UNIT_PRICE) AS TOTAL_SALE_AMT
    FROM DW.INVOICE_DIM i
    JOIN DW.INVOICELINE_DIM il ON i.INVOICE_ID = il.INVOICE_ID
    JOIN DW.CUSTOMER_DIM cd ON cd.CUSTOMER_ID = i.CUSTOMER_ID
    GROUP BY
        i.INVOICE_ID,
        cd.CUSTOMER_KEY,
        i.SALE_DATE
)

```

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists various pipelines, datasets, and activities. The main workspace displays the configuration for the 'load_SALES_FACT_PL' pipeline. A modal window titled 'Copy data' is open, showing a single activity named 'LoadSALESFACT'. Below the modal, the 'Sink' tab is selected in the configuration pane, which shows the target dataset as 'chinook_ds_snowflake' with schema 'DW' and table 'SALES_FACT'. Other tabs include General, Source, Mapping, Settings, and User properties.

This screenshot shows the same Azure Data Factory interface after a successful pipeline run. The 'load_SALES_FACT_PL' pipeline is now highlighted in the 'Activities' list. The main workspace displays the pipeline's status as 'Succeeded'. The 'Output' tab is selected, showing details of the most recent run. The run ID is 3efa921a-03b8-40a0-8ae7-5b46e4f3891a. The pipeline status is also marked as 'Succeeded'. Below this, a table provides a detailed view of the run activities, showing one item named 'LoadSALESFACT' with a status of 'Succeeded'.

The screenshot shows the Snowflake Database Explorer interface. On the left, the navigation pane displays the HORIZON CATALOG, Databases, and the CHINOOK_DB schema. Under CHINOOK_DB, there is a DW schema which contains several dimension tables like ALBUM_DIM, ARTIST_DIM, CUSTOMER_DIM, etc., and the fact table SALES_FACT. The SALES_FACT table is currently selected. The main panel shows the table details: 412 rows, updated just now, and a preview of the data. The preview table has columns: SALES_KEY, CUSTOMER_KEY, INVOICE_ID, DATE_DIM_KEY, TIME_DIM_KEY, TOTAL_SALE_AMT, and SOURCE_ID. The data includes rows from 1 to 18, with various values for each column. A footer URL https://app.snowflake.com/mimfsey/hz22766/#/data/databases/CHINOOK_DB/schemas/DW/table/SALES_FACT/data-preview is visible.

Why Use JOIN for loading SALES_FACT?

Overview

When loading the `SALES_FACT` table, we use **INNER JOIN** for dimension lookups instead of LEFT JOIN. This design choice ensures data integrity, enforces business rules, and maintains referential integrity in our star schema.

Joins Used in SALES_FACT Load

INNER JOINs (Dimension Lookups)

```
-- Invoice to Invoice Lines
FROM DW.INVOICE_DIM i
JOIN DW.INVOICELINE_DIM il ON i.INVOICE_ID = il.INVOICE_ID

-- Invoice to Customer
JOIN DW.CUSTOMER_DIM cd ON cd.CUSTOMER_ID = i.CUSTOMER_ID

-- Sales to Date Dimension
JOIN DW.DATE_DIM dd ON dd.FULL_DATE = DATE(a.SALE_DATE)

-- Sales to Time Dimension
JOIN DW.TIME_DIM td ON td.TIME_24_HR = TO_CHAR(a.SALE_DATE, 'HH24:MI')
```

LEFT JOIN (Incremental Load Only)

```
-- Anti-join pattern to find new records
LEFT JOIN DW.SALES_FACT sf ON sf.INVOICE_ID = a.INVOICE_ID
WHERE sf.INVOICE_ID IS NULL
```

Why INNER JOIN?

1. Enforces Referential Integrity

Our `SALES_FACT` table has foreign key constraints:

```
CONSTRAINT FK_SALES_DATE FOREIGN KEY (DATE_DIM_KEY) REFERENCES DW.DATE_DIM(DATE_KEY)
CONSTRAINT FK_SALES_TIME FOREIGN KEY (TIME_DIM_KEY) REFERENCES DW.TIME_DIM(TIME_KEY)
CONSTRAINT FK_SALES_CUSTOMER FOREIGN KEY (CUSTOMER_KEY) REFERENCES DW.CUSTOMER_DIM(CUSTOMER_KEY)
```

INNER JOIN ensures:

- Every fact record has valid dimension keys
- No NULL foreign keys that would violate constraints
- Insert operations succeed without constraint violations

What happens with LEFT JOIN:

- Missing dimensions would create NULL foreign keys
- Foreign key constraint violations would cause pipeline failures
- Less clear error messages

2. Acts as Data Quality Gate

INNER JOIN automatically **excludes incomplete or invalid records**:

Issue	INNER JOIN Behavior	LEFT JOIN Behavior
Missing customer	Record excluded	NULL customer_key → Fails insert
Date outside DATE_DIM range	Record excluded	NULL date_key → Fails insert
Invoice without line items	Record excluded	NULL amounts → Bad data

Result: Only complete, valid records enter the fact table.

3. Implements Business Rules

Certain relationships are **mandatory** from a business perspective:

- **Every sale MUST have a customer** - Cannot have anonymous sales
- **Every invoice MUST have line items** - No invoice without products
- **Every sale MUST have a date/time** - Required for time-series analysis

INNER JOIN enforces these rules at the **data level**, preventing logically invalid records.

4. Maintains Star Schema Integrity

In dimensional modeling (Kimball methodology):

- Fact tables should always reference valid dimension keys
- NULL foreign keys break analytical queries
- All dimension lookups must succeed for reliable reporting

INNER JOIN guarantees:

- All aggregations work correctly
- No NULL handling needed in BI queries
- Query performance remains optimal

When We Use LEFT JOIN

Only for incremental loading (anti-join pattern):

```
LEFT JOIN DW.SALES_FACT sf ON sf.INVOICE_ID = a.INVOICE_ID  
WHERE sf.INVOICE_ID IS NULL
```

Purpose:

- Find invoices that DON'T exist in SALES_FACT yet
- Prevent duplicate loads
- This is NOT a dimension lookup - it's a duplicate-check

What Would Happen with LEFT JOIN for Dimensions?

Example: Missing Customer

```
-- Using LEFT JOIN (wrong approach)  
LEFT JOIN DW.CUSTOMER_DIM cd ON cd.CUSTOMER_ID = i.CUSTOMER_ID
```

Result:

- Invoice with missing customer has `CUSTOMER_KEY = NULL`
- Foreign key constraint violation during insert
- Pipeline fails with cryptic error
- Harder to identify which records caused the issue

Example: Missing Date

```
-- Using LEFT JOIN (wrong approach)  
LEFT JOIN DW.DATE_DIM dd ON dd.FULL_DATE = DATE(a.SALE_DATE)
```

Result:

- Sales outside DATE_DIM range get `DATE_DIM_KEY = NULL`
- Cannot analyze sales by time period
- Foreign key constraint fails
- Less obvious what the root cause is

Summary

Aspect	Why INNER JOIN
Data Integrity	Prevents NULL foreign keys and constraint violations
Data Quality	Excludes incomplete records automatically
Business Rules	Enforces mandatory relationships
Star Schema	Maintains dimensional model integrity
Error Detection	Clear failure points when dimensions are missing

The **only LEFT JOIN** in our query is for the incremental load check (anti-join pattern), which serves a different purpose: identifying new records that haven't been loaded yet.

This strategy ensures `SALES_FACT` contains only **complete, valid, and analytically useful** records for reliable reporting and analysis.

Key Takeaway

INNER JOIN = Data Quality Gate

If a record can't join to all required dimensions, it shouldn't be in the fact table. This prevents bad data from entering the warehouse and ensures downstream analytics remain reliable.