<u>Cloud Data Warehouse</u>

A cloud Data Warehouse is a Service that Collects, Organizes and Often Stores Data that is Used by Organizations for Different Activities including Data Analytics and Monitoring.

- ➢ Amazon Redshift.
- ➢ Google Big Query.
- ➢ IBM DB2 Warehouse.
- ➢ Microsoft Azure SQL Data Warehouse.
- ➢ Oracle Autonomous Data Warehouse.
- ➢ SAP Data Warehouse Cloud.
- ➢ SnowFlake.

<u>SNOWFLAKE</u>

Key Capabilities

- ✓ Data Warehouse that runs entirely on Cloud Infrastructure Such as Amazon AWS and Microsoft Azure Cloud.
- ✓ It Cannot Be Run on a Private Cloud, or on Hosted Infrastructures.
- ✓ It is NOT Relational Database, So It Doesn't Enforce Primary / Foreign Key Constraints.
- ✓ It Offers SnowFlake SQL Commands like DDL / DML, SQL Functions, User Defined Functions / Stored Procedures (JavaScript).
- ✓ The SnowFlake SQL is Rick Enough to meet most of The Data Warehousing Requirements.
- ✓ It Supports Select Statements, Joins, Subqueries, Insert, Update, Delete, Merge Statements, Views and Materialized and ACID transactions.
- ✓ It also Supports Common Analytical Aggregations Such as Cube, Rollup, Grouping Sets, Windowing Functions, Connect by and Hierarchical Queries.

Integration Support

- ✓ It allows Connection from Data Integration Tools Such as IBM Data Stage, Informatica, Talend, Pentaho etc.
- ✓ Self Service BI Tools Such as Power BI, Tableau, Qlikview, Spotfire etc.
- ✓ Big Data Tools Such as Apache Kafka, Apache Spark, Data Bricks etc.
- ✓ JDBC / ODBC Drivers to Connect from Your Application.
- ✓ It Offers Native Language Connectors Such as Python, Go, NodeJS etc.
- ✓ For Adhoc Querying, you can use Snowflake Web Interface, Snowflake CLI, DBeaver etc.

Unique Offering by SnowFlake

- ✓ Scalability (Storage & Compute)
- ✓ User Experience Performance Without Indexing, Tuning, Partitioning and Physical Storage Design Considerations. You don't have Multiple Knobs to Tune the Database Performance.
- ✓ Tunable Pay per Use.
- ✓ Snowflake Does Real Separation of Storage & Compute. There are 2 Notions Such as Database and Virtual Data Warehouse.

Database

- ✓ Database is Storage Layer (DDL). It allows you to Create and Define a Database, Scheme, Table and then Add Table to the Schema. Once You have Tables, you can Start Loading Data into those Tables. This is the Typical Structure That We Follow in Any Data Warehouse.

- ✓ End User -> Amazon EC2 (pay Per Use for Compute and also Supports Auto Start and Shutdown) -> Amazon S3 (Pay Per Use for Storage).
- ✓ The Data in the Tables Are Stored in Amazon S3 and It Consumes Storage Only Cost.
- ✓ There is No Compute Cost Attached to the Database Unless You Are Executing DDL / DML Queries.
- ✓ What It Means, You Will Pay for the Compute Cost While You Are Running DDL Statements to Create a Database, Schema, Table or Other Structural Objects. If All These Activities Take 30 Mins, You Will pay for 30 Mins of Computing Cost.
- ✓ Once your Table Structure is in Place, You Would Want to Load Some Data into Your Tables.
- ✓ If your Data Load Job is Running for an Hour in a Day, You Pay for Compute Cost for an Hour. Rest of the Time, you will be Paying Only for Storage Cost.
- ✓ If you are Running Queries for 3 Hours/Day, You Will Be Paying to Compute Cost for Those 3 Hours.

Virtual Data Warehouse

- ✓ It is Compute Cluster. They are Named After VM but Instead of machines, Snowflake Termed as Warehouse. In Short, We Cal them as Virtual Warehouse (VW).
- ✓ You can Create VW of Various Sizes Depending Upon Your Requirements.
    - X-Small - 1 Node | Small - 2 Nodes | Medium - 4 Nodes | large - 8 Nodes | X-Large - 16 Nodes
- ✓ These Nodes are Amazon EC2 Instances, But They Are Internal to VW but You Don't Want to Interact with them.
- ✓ Creating a VW Does Not Have Any Cost Associated with it. It's Just Metadata Creation. You can Have More than 1 VW Configurations.
- ✓ For Example, You Might Want to Create a Single Node VW for Executing DDLS or Performing Some Other Low Resource Consuming Activities. At Same Time, You Can Also Define 4 Nodes Which You Want to Use Only for Data Loading Activity. You can Create 1 More 32 Node VM Which You Wish to Apply for a High Performing Computation Job That Triggers Every Hour and Completes in less than 5 Mins.
- ✓ You Can Create Compute Resource Definition and Start it Only When You Want to Do Some Computation and Shut it Down When the Computation Activity is Over. In this Way, You Pay the Compute Cost Only When Your VW is Running.
- ✓ Best Part is You Also Know Which job us Costing You How Much Money? A Job That Runs for 5 Mins on 32 Node VW and Triggers 24 Times a Day is Charging You for 2 Hours of 32 Node VW.
- ✓ You can Also Use the Same VW to Execute Multiple Concurrent Queries.
- ✓ As the Queries Are Submitted to VW, The Warehouse Allocates Resource to Each Query and Begin Running Them. If Sufficient Resource Are Not Available to Execute All Queries, Additional Queries will be Queued Until the Necessary Resources Become Available.
- ✓ You have All Flexibility to Plan the Compute Workload and Reuse Your VWs.
- ✓ Snowflake Offers a Unique Idea of Multi-Cluster VW for Auto Scaling.
- ✓ For Example, You Created a Multi-Cluster VW and Started a Query on the VW. Snowflake Would Start Your Query Normally in the Same VW. However, If You Start Submitting More Queries on Same VW, at Some Point, they will Consume All Resources and Additional Queries will be Queued Until Existing Ones Are Complete.
- ✓ A Multi-Cluster Virtual Warehouse Will Detect Above Scenario and Automatically Launch a New VW to Execute the Queued SQL.
- ✓ SnowFlake Allows You to Configure Automatic Scale Up by Starting Additional VWs and Scale Down by Shutting Down VWs Depending Upon the Workload. This Feature is Powerful If You have Strict Performance SLAs.

SnowFlake Architecture

Layer 1 -> Service Layer Manages Meta Data, Security, Optimizer, Transactions, Concurrency.
Layer 2 -> Compute Layer -> Virtual Warehouse.
Layer 3 -> Storage Layer -> Database in Amazon S3.

- ✓ Layer 1 is What SnowFlake to Act as a Database and Comes Mostly at a Nominal Fixed price.
- ✓ Your Main Cost is Associated with the Storage and Compute Requirements.
- ✓ Separation of Storage and Compute is Available with Many Other Products Such as Amazon EMR (Elastic Map Reduce) and Data Bricks. You also have Rich Enough SQL Offered by Spark SQL.
- ✓ Snowflakes Provides Better User Experience and Makes It Simple to Implement.

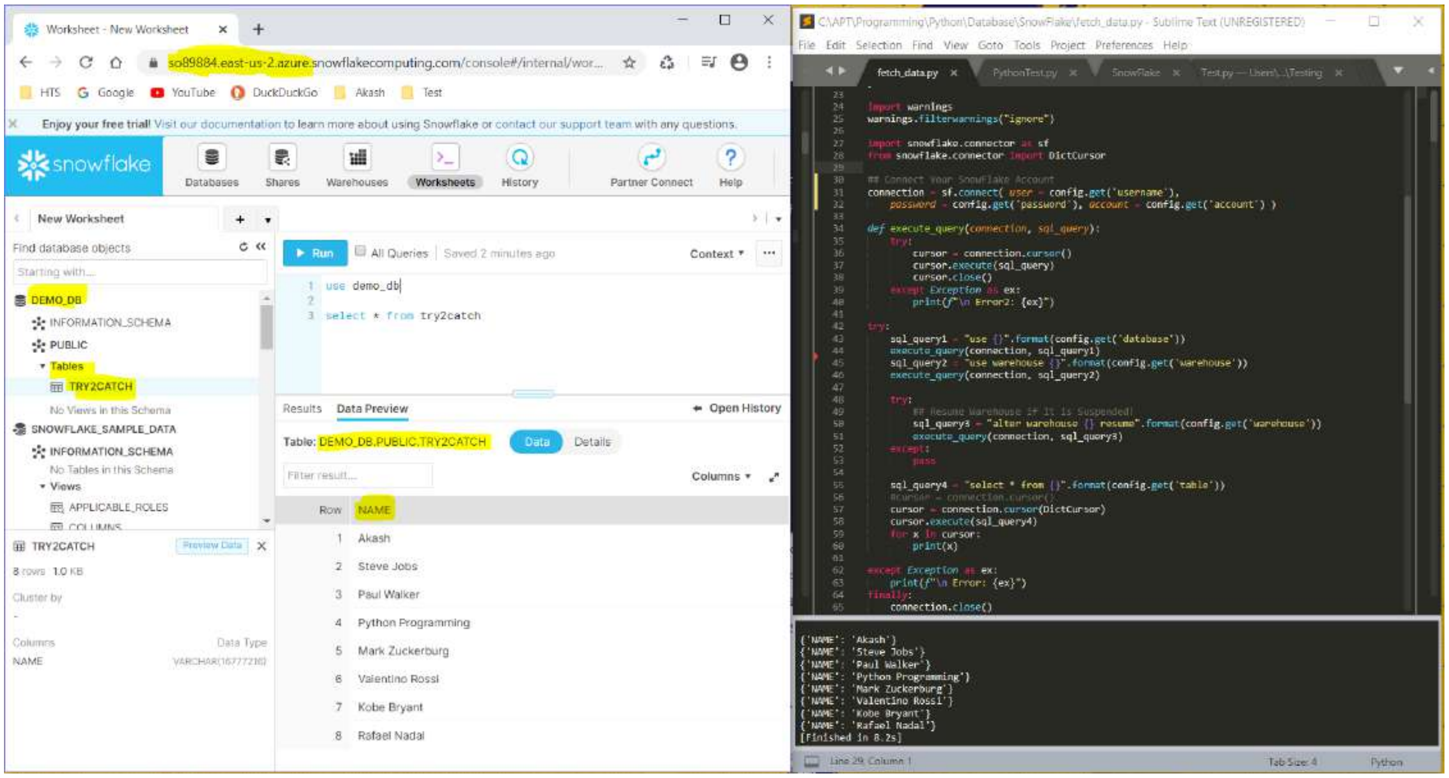
- ✓ SnowFlake Decided to use Amazon S3 as Its Database Storage.
- ✓ Challenges with S3 are I/O Latency, CPU Overhead (HDFS on Local Storage Works a Lot Better than Amazon S3) and Object Storage (Overwrite Only).
- ✓ Features of S3 are High Availability, Durability and API to Read Part of Data from S3 Fil Without Loading an Entire File is Kind of Random Access Capability.

- ✓ Let’s Assume That You Have a Large Table
- ✓ SnowFlake Breaks the Super Large Table into Multiple Smaller Partitions, call it as Micro Partitions Which is Not More than 500 MB in Size.
- ✓ Each Table Partition between 50 to 500 MB of Uncompressed Data.
- ✓ Reorganize the Data in Each Partition to Make is Columnar (Column Values in the Partition Rre Stored Together). Then Compress Each Column. Now its UNIQUE.
- ✓ You are Not Compressing the Entire Partition, Only Column Values Individually.
- ✓ Finally, They Add a Header to Each Micro Partition. The Header Contains Offset and length of Each Column Stored Within the Micro Partition. They also Store Some Other Metadata Information in the Header, BUT the Column Offsets are the Most Critical Information for US to Understand the Read Pattern.
- ✓ Now, these Micro Partitions Are Stored in S3 as Immutable Files and Lot of Additional Metadata and Statistics About These Micro Partitions is Maintained in the SnowFlake Metadata Layer.
- ✓ That’s why they Call the Cloud Service Layer of the SnowFlake as Brain of the SnowFlake.
- ✓ When you Fire an SQL Query, SnowFlake Would Know Which Tables and What Columns Do You Want to Read. Which Partition Files in S3 Belongs to Desired Table. So they know, Which Files to Read but Instead of Reading Complete File, They First Read the Header and then read Only Those Columns Which You Wanted to Read.
- ✓ This is Possible Because S3 APIs Allow Them to Read Part of File Based on Offset and Length.

- ✓ SnowFlake Stores Metadata and Statistics About All the Data Stored in a Micro-Partition. So They Know the Range of Values and Number of Distinct Values for Each of the Columns in the Micro-Partitions.
- ✓ So When You Apply a Filter in Where Clause, They Know Which Micro-Partitions Would have the Desired Date. This Allows the First Level of Partition Pruning and Targets Only Those Micro-Partitions Where the Desired Data is Stored.
- ✓ Next Step, They First Read the Micro-Partition Header and then Only Read the Desired Columns. This Allows the Second Level of Column Pruning and Minimize the Overall I/O of the SQL.
- ✓ If you have Time Series Data and Your Queries Are Filtering on a Timeframe. SnowFlake Partitioning Strategy Works like a Charm.

SnowFlake | Python Fetch Data from SnowFlake Personal Account

fetch_data.py



Worksheet - New Worksheet
so89884.east-us-2.azure.snowflakecomputing.com/console#/internal/wor...
HTS Google YouTube DuckDuckGo Akash Test
Enjoy your free trial! Visit our documentation to learn more about using Snowflake or contact our support team with any questions.
snowflake
Databases Shares Warehouses Worksheets History Partner Connect Help
New Worksheet
Find database objects
Starting with...
DEMO_DB
INFORMATION_SCHEMA
PUBLIC
Tables
TRY2CATCH
No Views in this Schema
SNOWFLAKE_SAMPLE_DATA
INFORMATION_SCHEMA
No Tables in this Schema
Views
APPLICABLE_ROLES
COLUMNS
TRY2CATCH Preview Data
8 rows 1.0 KB
Cluster by
Columns Data Type
NAME VARCHAR(16777216)
Run All Queries Saved 2 minutes ago Context
Results Data Preview Open History
Table: DEMO_DB.PUBLIC.TRY2CATCH Data Details
Filter result...
Columns
C:\APT\Programming\Python\Database\SnowFlake\fetch_data.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
fetch_data.py PythonTest.py SnowFlake Test.py — Users\...\Testing
Line 29, Column 1 Tab Size: 4 Python

```sql
use demo_db

select * from try2catch
```

Row	NAME
1	Akash
2	Steve Jobs
3	Paul Walker
4	Python Programming
5	Mark Zuckerburg
6	Valentino Rossi
7	Kobe Bryant
8	Rafael Nadal

```python

import warnings
warnings.filterwarnings("ignore")

import snowflake.connector as sf
from snowflake.connector import DictCursor

## Connect Your SnowFlake Account
connection = sf.connect( user = config.get('username'),
    password = config.get('password'), account = config.get('account') )

def execute_query(connection, sql_query):
    try:
        cursor = connection.cursor()
        cursor.execute(sql_query)
        cursor.close()
    except Exception as ex:
        print(f"\n Error2: {ex}")

try:
    sql_query1 = "use {}".format(config.get('database'))
    execute_query(connection, sql_query1)
    sql_query2 = "use warehouse {}".format(config.get('warehouse'))
    execute_query(connection, sql_query2)

    try:
        ## Resume Warehouse if It is Suspended!
        sql_query3 = "alter warehouse {} resume".format(config.get('warehouse'))
        execute_query(connection, sql_query3)
    except:
        pass

    sql_query4 = "select * from {}".format(config.get('table'))
    #cursor = connection.cursor()
    cursor = connection.cursor(DictCursor)
    cursor.execute(sql_query4)
    for x in cursor:
        print(x)

except Exception as ex:
    print(f"\n Error: {ex}")
finally:
    connection.close()
```

```
{'NAME': 'Akash'}
{'NAME': 'Steve Jobs'}
{'NAME': 'Paul Walker'}
{'NAME': 'Python Programming'}
{'NAME': 'Mark Zuckerburg'}
{'NAME': 'Valentino Rossi'}
{'NAME': 'Kobe Bryant'}
{'NAME': 'Rafael Nadal'}
[Finished in 8.2s]
```

SnowFlake Portal

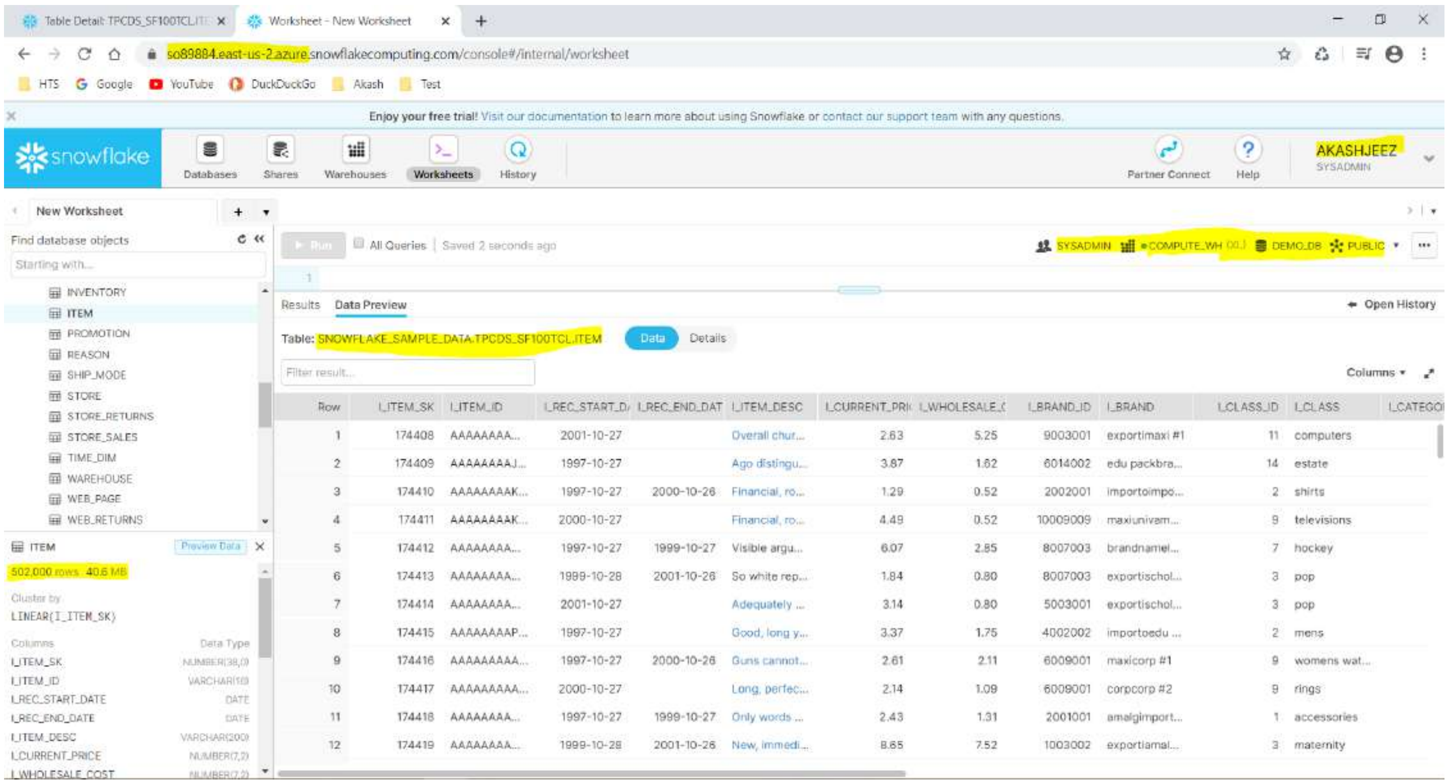


Table Detail: TPCDS_SF100TCL.IT
Worksheet - New Worksheet
so89884.east-us-2.azure.snowflakecomputing.com/console#/internal/worksheet
HTS Google YouTube DuckDuckGo Akash Test
Enjoy your free trial! Visit our documentation to learn more about using Snowflake or contact our support team with any questions.
snowflake
Databases Shares Warehouses Worksheets History Partner Connect Help
AKASHJEEZ
SYSADMIN
New Worksheet
Find database objects
Starting with...
INVENTORY
ITEM
PROMOTION
REASON
SHIP_MODE
STORE
STORE_RETURNS
STORE_SALES
TIME_DIM
WAREHOUSE
WEB_PAGE
WEB_RETURNS
Run All Queries Saved 2 seconds ago
SYSADMIN COMPUTE_WH (XS) DEMO_DB PUBLIC
Results Data Preview Open History
Table: SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.ITEM Data Details
Filter result...
Columns

ITEM

502,000 rows 40.6 MB

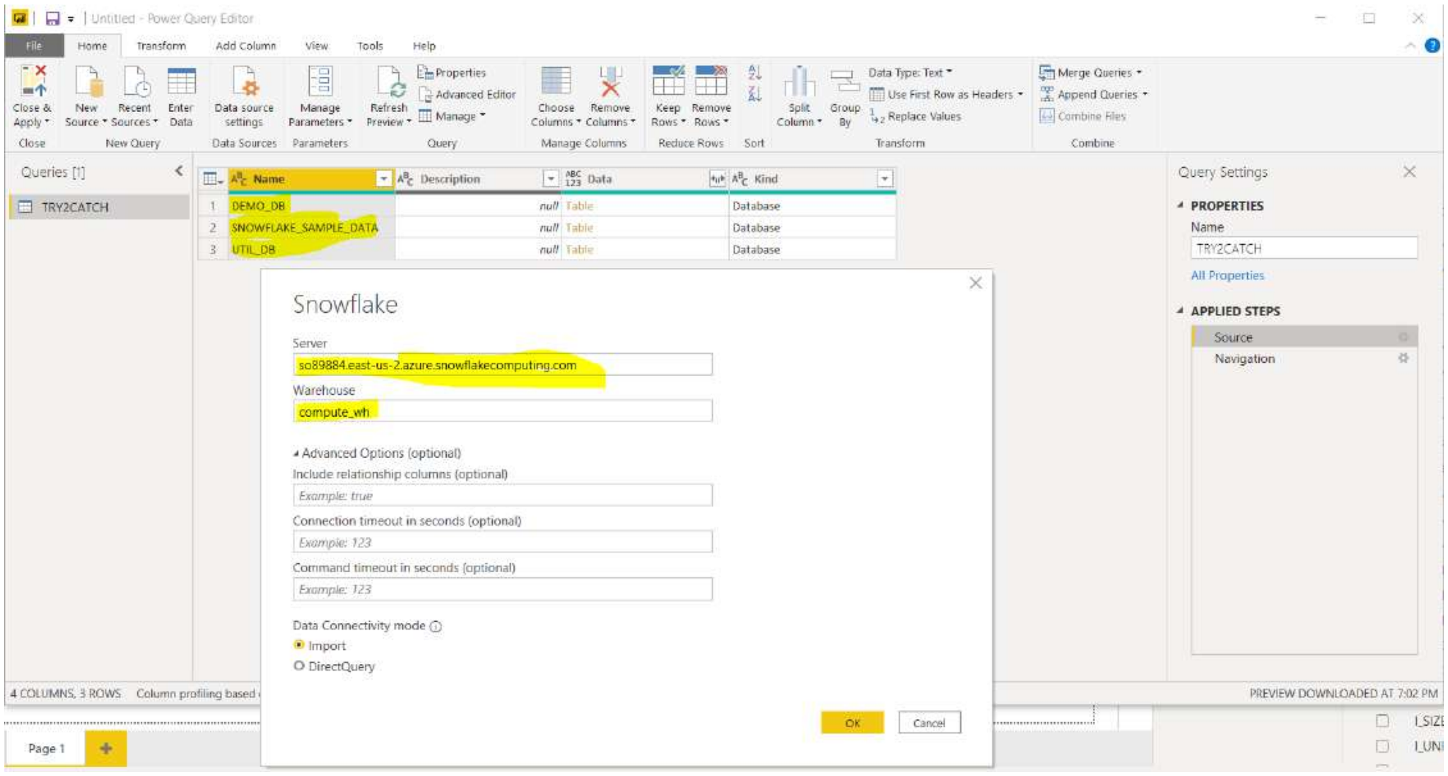
Cluster by

LINEAR(I_ITEM_SK)

Columns	Data Type
I_ITEM_SK	NUMBER(38,0)
I_ITEM_ID	VARCHAR(16)
I_REC_START_DATE	DATE
I_REC_END_DATE	DATE
I_ITEM_DESC	VARCHAR(200)
I_CURRENT_PRICE	NUMBER(7,2)
I_WHOLESALE_COST	NUMBER(7,2)

Row	I_ITEM_SK	I_ITEM_ID	I_REC_START_D/	I_REC_END_DAT	I_ITEM_DESC	I_CURRENT_PRI	I_WHOLESALE_(	I_BRAND_ID	I_BRAND	I_CLASS_ID	I_CLASS	I_CATEGO
1	174408	AAAAAAAA...	2001-10-27		Overall chur...	2.63	5.25	9003001	exportimaxi #1	11	computers	
2	174409	AAAAAAAAJ...	1997-10-27		Ago distingu...	3.87	1.62	6014002	edu packbra...	14	estate	
3	174410	AAAAAAAAK...	1997-10-27	2000-10-26	Financial, ro...	1.29	0.52	2002001	importoimpo...	2	shirts	
4	174411	AAAAAAAAK...	2000-10-27		Financial, ro...	4.49	0.52	10009009	maxiunivam...	9	televisions	
5	174412	AAAAAAAA...	1997-10-27	1999-10-27	Visible argu...	6.07	2.85	8007003	brandnamel...	7	hockey	
6	174413	AAAAAAAA...	1999-10-28	2001-10-26	So white rep...	1.84	0.80	8007003	exportischol...	3	pop	
7	174414	AAAAAAAA...	2001-10-27		Adequately ...	3.14	0.80	5003001	exportischol...	3	pop	
8	174415	AAAAAAAAP...	1997-10-27		Good, long y...	3.37	1.75	4002002	importoedu ...	2	mens	
9	174416	AAAAAAAAA...	1997-10-27	2000-10-26	Guns cannot...	2.61	2.11	6009001	maxicorp #1	9	womens wat...	
10	174417	AAAAAAAAA...	2000-10-27		Long, perfec...	2.14	1.09	6009001	corpcorp #2	9	rings	
11	174418	AAAAAAAA...	1997-10-27	1999-10-27	Only words ...	2.43	1.31	2001001	amalgimport...	1	accessories	
12	174419	AAAAAAAA...	1999-10-28	2001-10-26	New, immedi...	8.65	7.52	1003002	exportiamal...	3	maternity	

Configure SnowFlake Account in PowerBI Tool.



	Name	Description	Data	Kind
1	DEMO_DB	null	Table	Database
2	SNOWFLAKE_SAMPLE_DATA	null	Table	Database
3	UTIL_DB	null	Table	Database

Snowflake

Server: so89884.east-us-2.azure.snowflakecomputing.com

Warehouse: compute_wh

Advanced Options (optional)

Include relationship columns (optional): Example: true

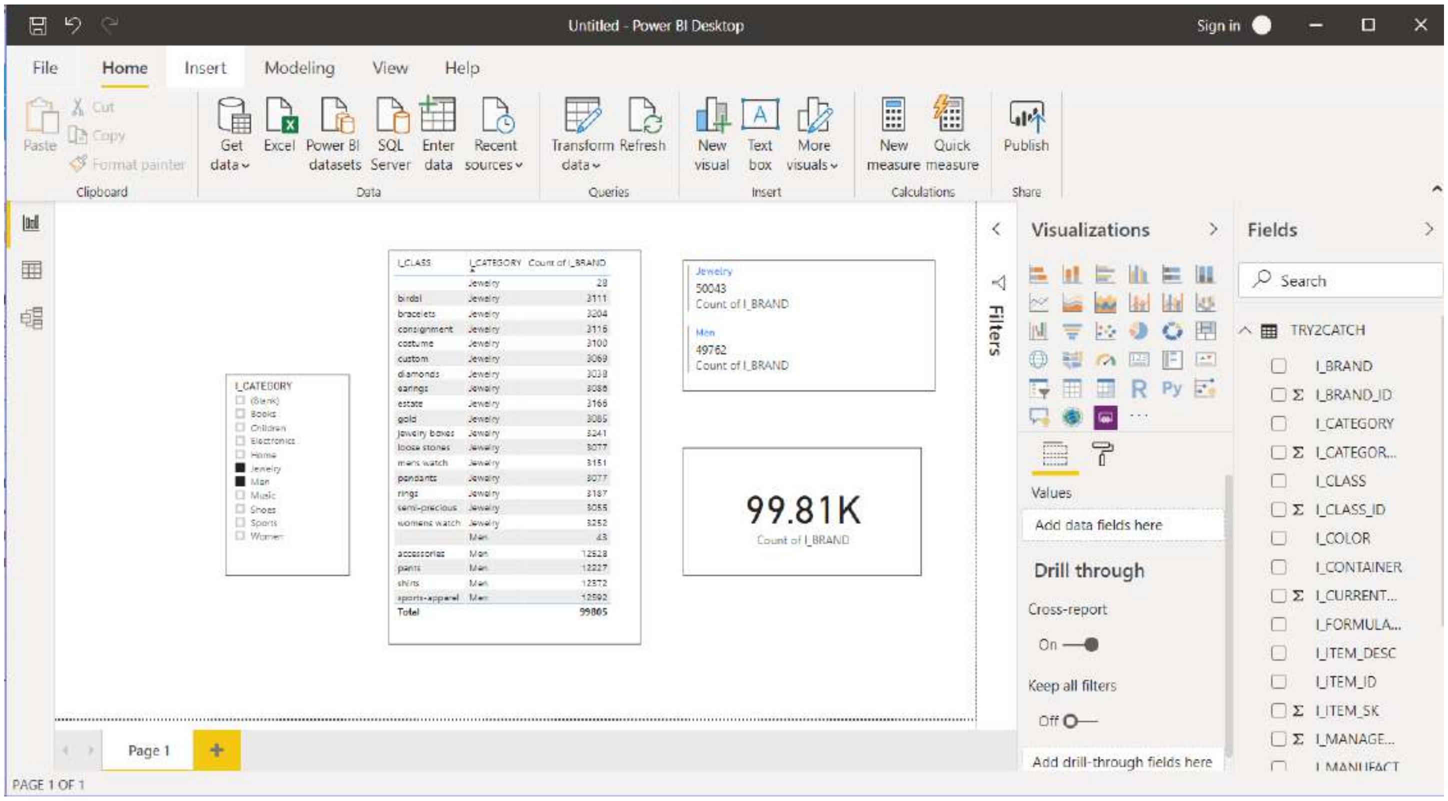
Connection timeout in seconds (optional): Example: 123

Command timeout in seconds (optional): Example: 123

Data Connectivity mode: [✓ Import] DirectQuery

Query Settings — Name: TRY2CATCH — Applied Steps: Source, Navigation

Visualize SnowFlake Data in PowerBI Tool.



I_CLASS	I_CATEGORY	Count of I_BRAND
	Jewelry	28
birdal	Jewelry	3111
bracelets	Jewelry	3204
consignment	Jewelry	3116
costume	Jewelry	3100
custom	Jewelry	3069
diamonds	Jewelry	3038
earings	Jewelry	3086
estate	Jewelry	3166
gold	Jewelry	3085
jewelry boxes	Jewelry	3241
loose stones	Jewelry	3077
mens watch	Jewelry	3151
pendants	Jewelry	3077
rings	Jewelry	3187
semi-precious	Jewelry	3055
womens watch	Jewelry	3252
	Men	43
accessories	Men	12528
pants	Men	12227
shirts	Men	12372
sports-apparel	Men	12592
Total		99805

Jewelry 50043 Count of I_BRAND

Men 49762 Count of I_BRAND

99.81K Count of I_BRAND