



JIO – HOTSTAR Merger Analysis

- A Data Driven Approach

Domain - Telecom & Streaming Services
Function - Strategy and Operations

by

Raghudathesh G P, Prathviraj N

Manipal School of Information Sciences

1. Problem Statement

Jio, a leading telecommunications provider in India, is planning a strategic merger with Hotstar, one of the country's premier streaming platforms, to combine JioCinema's extensive subscriber base with Hotstar's diverse content library, aiming to revolutionize digital streaming and establish the Jio-Hotstar platform as India's leading OTT service.

To support this merger, Jio's management team seeks to conduct a comprehensive analysis of both platforms' performance and user behavior from January to November 2024, examining overall platform performance, content consumption patterns, subscriber growth trends, user inactivity behavior, and subscription upgrade and downgrade patterns.

The insights gained from this study will inform data-driven decisions, optimize content strategies, and ensure the merged Jio-Hotstar platform achieves market leadership in India's competitive OTT industry.

1. Problem Statement

- The analysis should provide clear, actionable insights into the following areas:
 - **Content Library Analysis:** A detailed comparison of content types available on JioCinema and Hotstar, highlighting differences and similarities in genres, formats, and offerings.
 - **Subscriber Insights:** An examination of subscriber acquisition trends, including growth patterns and demographic variations such as age, gender, and geographic distribution.
 - **Inactivity Analysis:** Identification of user inactivity patterns, segmented by age groups, city tiers (e.g., metro, Tier 1, Tier 2), and subscription plans, to understand engagement challenges.
 - **Upgrade Patterns:** Insights into subscription upgrades, including the factors influencing users' decisions to opt for higher-tier plans, such as content preferences or promotional offers.

1. Problem Statement

- **Downgrade Patterns:** Analysis of subscription downgrades, identifying trends and reasons behind users' decisions to switch to lower-tier plans or cancel subscriptions.
- **Content Consumption Behavior:** Evaluation of content consumption patterns, including total watch time, preferred devices (e.g., mobile, TV, web), and variations across user demographics like age, gender, and location.
- These insights will enable the management team to make informed decisions, optimize content and subscription strategies, and position the merged Jio-Hotstar platform as the leading OTT service in India.

Suggested Metrics and Analyses for Initial Insights

- The proposed metrics and analyses outlined below provide a foundation for deriving initial insights into the performance and user behavior of JioCinema and Hotstar from January to November 2024.
- You are encouraged to explore additional relevant metrics and analyses to uncover deeper trends and deliver more comprehensive findings to support the strategic merger and optimize the Jio-Hotstar platform's content and subscription strategies.

Total content items	Total users	Paid users	Paid users (%)
Active users	Inactive users	Inactive Rate (%)	Active Rate (%)
Upgraded users	Upgrade Rate (%)	Downgraded users	Downgrade Rate (%)
Total watch time (hrs)	Average watch time (hrs)	Monthly users Growth Rate (%)	Upgrade / Downgrade Rate (%)



- The following questions leverage available data to analyze the performance and user behavior of JioCinema and Hotstar from January to November 2024.

1. Total Users and Growth Trends

- What is the total number of users for JioCinema and Hotstar, and how do their growth trends compare over the analysis period (January–November 2024)?

2. Content Library Comparison

- What is the total number of content items available on JioCinema versus Hotstar? How do they differ in terms of language and content type (e.g., movies, TV shows, sports)?

3. User Demographics

- What is the distribution of users by age group, city tier (e.g., Metro, Tier 1, Tier 2/3), and subscription plan for each platform?

4. Active vs. Inactive Users

- What percentage of JioCinema and Hotstar users are active versus inactive? How do these rates vary by age group and subscription plan?



1. Watch Time Analysis

- What is the average watch time for JioCinema versus Hotstar during the analysis period? How does watch time compare across city tiers and device types (e.g., mobile, TV, web)?

2. Inactivity Correlation

- How do inactivity patterns correlate with total watch time or average watch time? Are users with lower engagement more likely to become inactive?

3. Downgrade Trends

- How do downgrade trends differ between JioCinema and Hotstar? Are downgrades more prevalent on one platform compared to the other?

4. Upgrade Patterns

- What are the most common upgrade transitions (e.g., Free to Basic, Free to VIP, Free to Premium) for JioCinema and Hotstar? How do these transitions differ across platforms?

5. Paid Users Distribution

- How does the percentage of paid users (e.g., Basic and Premium for JioCinema; VIP and Premium for Hotstar) vary across platforms? Analyze the proportion of premium users in Tier 1, Tier 2, and Tier 3 cities and identify notable trends or differences.

1. Revenue Analysis

- Using the monthly subscription prices below, calculate the total revenue generated by JioCinema and Hotstar from January to November 2024:

Platform	Plan	Price (INR - ₹)
JioCinema	Basic	₹69
	Premium	₹129
Hotstar	VIP	₹159
	Premium	₹359

- The calculation should account for:**
 - Subscriber count for each plan.
 - Active duration of subscribers on their respective plans.
 - Upgrades and downgrades during the period, ensuring revenue reflects the time spent on each plan.



Secondary Analysis and Recommendations

- The following questions aim to guide strategic recommendations for the merged JioCinema-Hotstar platform:
 1. **Increasing Engagement Among Inactive Users:**
 - What strategies can the merged platform implement to boost engagement among inactive users and convert them into active users?
 2. **Brand Campaigns for Market Leadership:**
 - What types of brand campaigns should the merged platform launch to establish itself as the leading OTT platform in India?
- 1. **Pricing Strategy for Competitiveness**
 - How should the merged platform price its subscription plans to remain competitive while maintaining profitability?
- 1. **Telecom Partnerships for Subscriber Growth**
 - How can the platform leverage partnerships with telecom companies to expand its subscriber base?



Secondary Analysis and Recommendations

- The following questions aim to guide strategic recommendations for the merged JioCinema-Hotstar platform:
 1. **AI and Machine Learning for Personalization:**
 - What role can AI and machine learning play in personalizing the user experience and improving content discovery?
 2. **Brand Ambassador Selection:**
 - Who should be the brand ambassador for the merged JioCinema-Hotstar platform to effectively represent its identity and attract a diverse audience?



meta_data.txt

- **You have two datasets from separate data sources:**
 - Hotstar_db: MySQL data source (Jotstar_db.sql)
 - JioCinema: MySQL data source (Jiocinema_db.sql)
 - Import these files into Power BI via the Data Source section.




Jotstar_db.sql

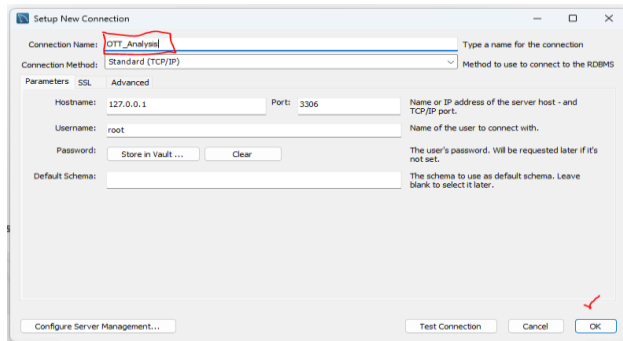


LioCinema_db.sql

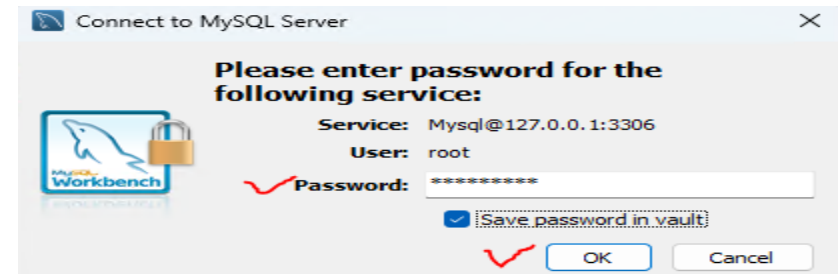


Loading Hotstar_dbDataset into Workbench

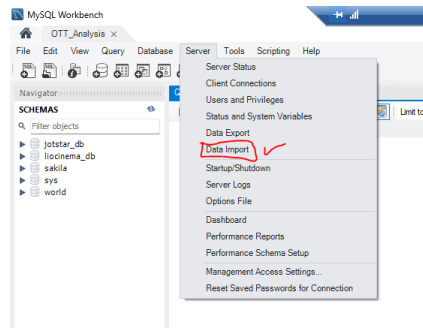
- 1. Launch MySQL Workbench 8.0 CE → 2. Create MySQL Connections  → 3.



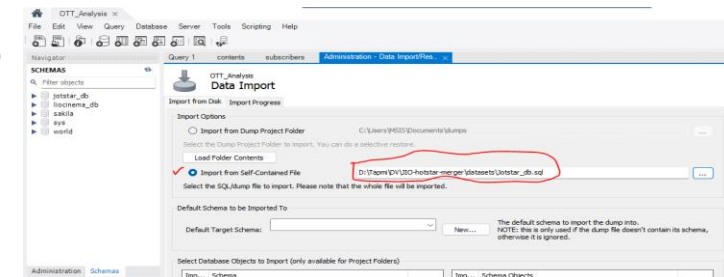
- 4. Click on  → 5. Enter UN and PW: UN: **root** PW: **root@aiml**



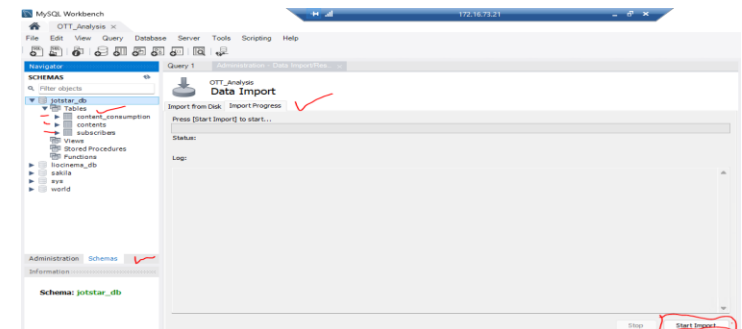
- 6. Load data set



- Select data source: jotstar_db

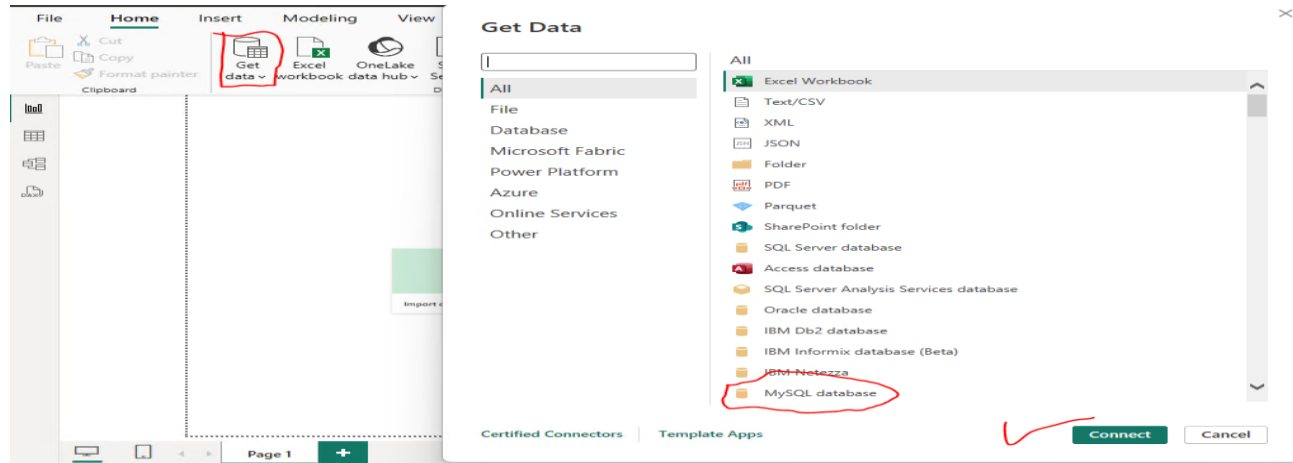


Jotstar_db.sql

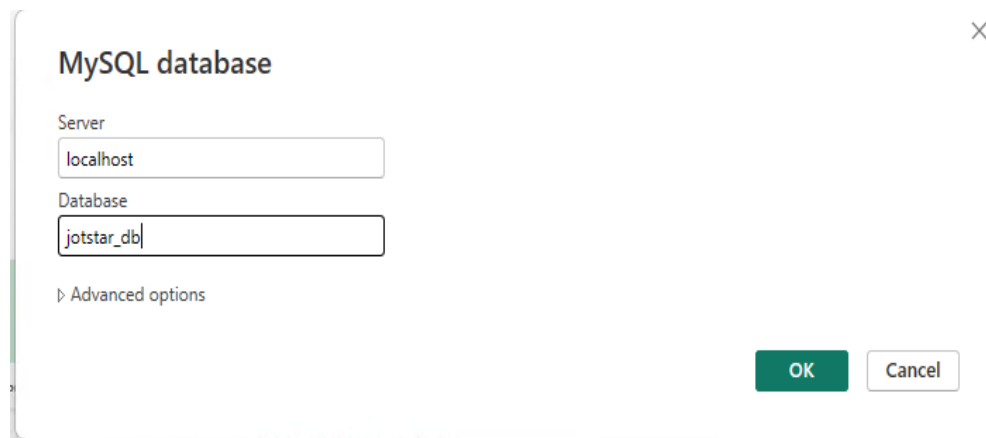


Loading Hotstar Dataset into PowerBI

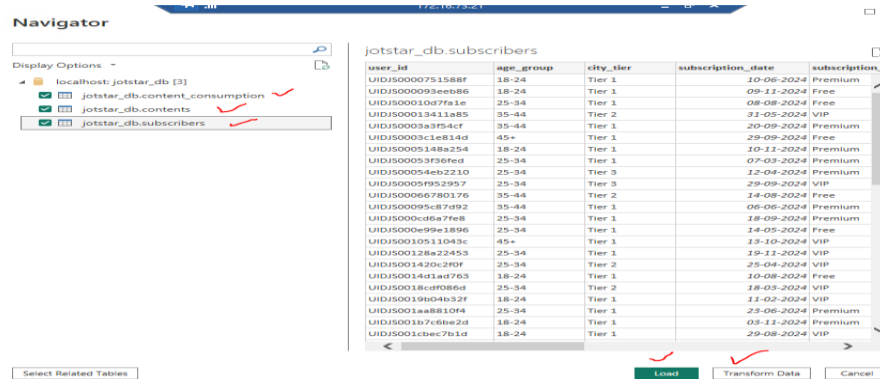
→ 1. Launch Power Bi → 2. Select getdata and do the following:



→ 3. Server: localhost/Ipaddress and database_name: jotstar_db

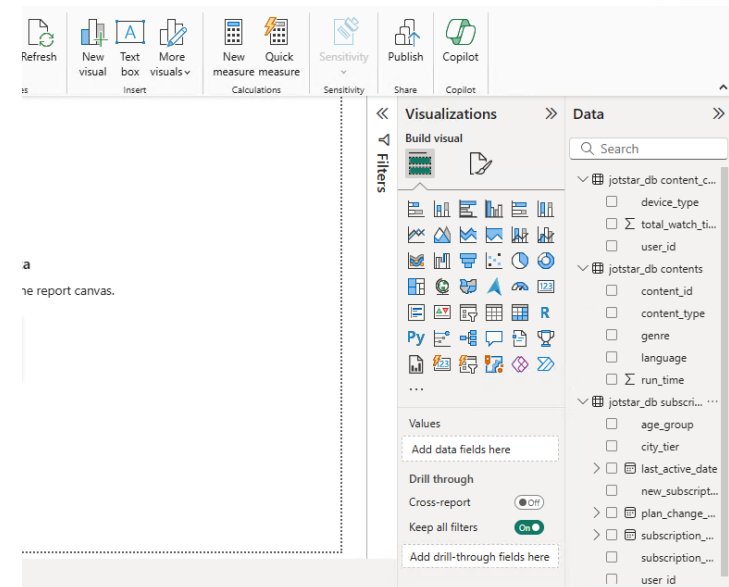
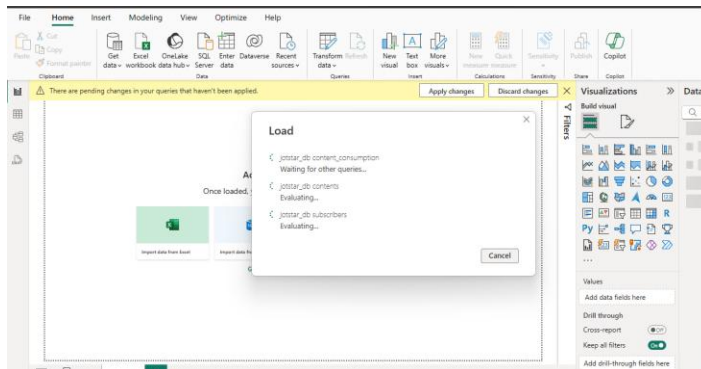


→ 4. Select the tables of interest and click load



→ 5. Wait the dataset loads

→ 6. Now dataset will be visible in right pane in data section





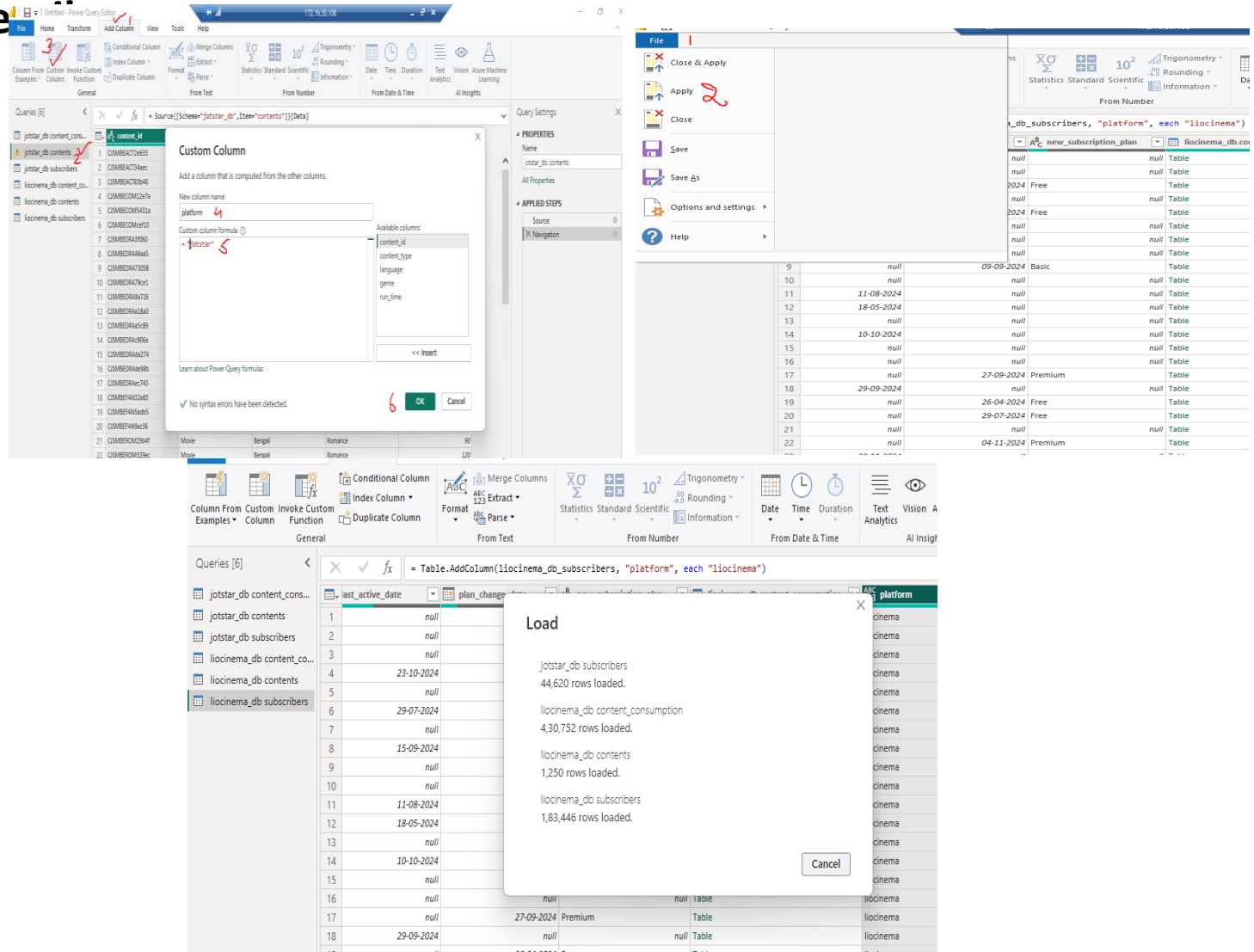
Loading Jiocinema_dbDataset into Workbench

- Repeat the procedure as discussed with Hotstar dataset.



LioCinema_db.sql

- Repeat the procedure as discussed below for all the tables one by one to add platform

The screenshot shows the Power Query Editor interface. The 'Custom Column' dialog box is open, showing the formula `= [platform]` and the available columns list. The 'Load' dialog box is also open, showing the table `jotstar_db subscribers` with 44,620 rows loaded. The main query editor shows the table `jotstar_db subscribers` with columns `last_active_date` and `platform`.

Custom Column Dialog:

- New column name: `platform`
- Custom column formula: `= [platform]`
- Available columns: `content_id`, `content_type`, `language`, `genre`, `run_time`

Load Dialog:

- Table: `jotstar_db subscribers`
- Rows loaded: 44,620

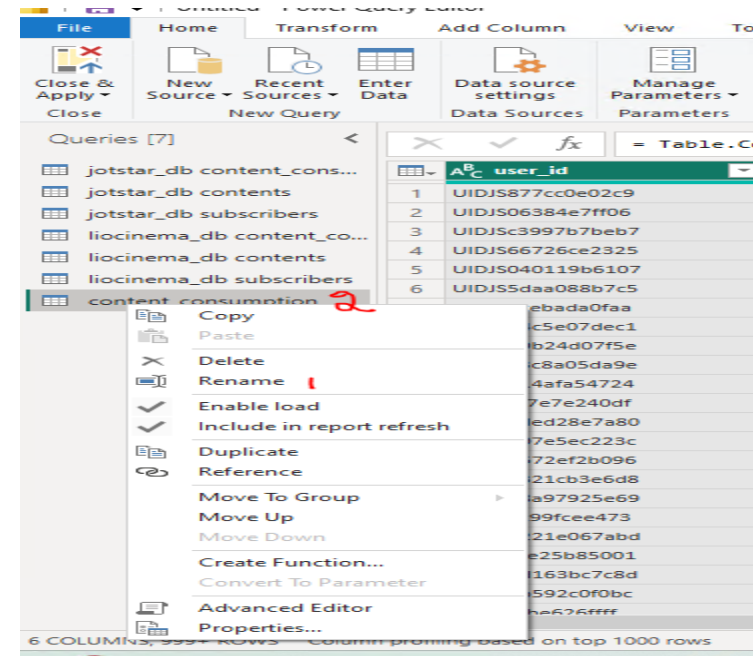
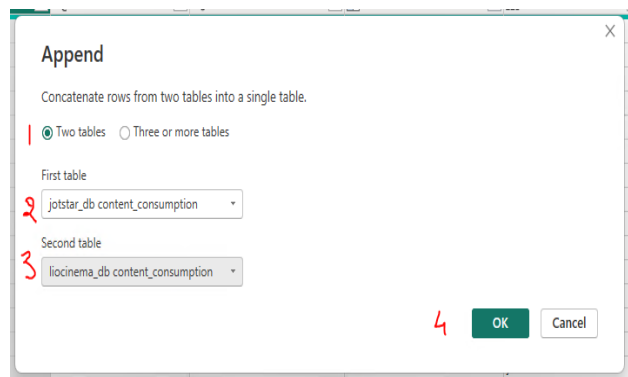
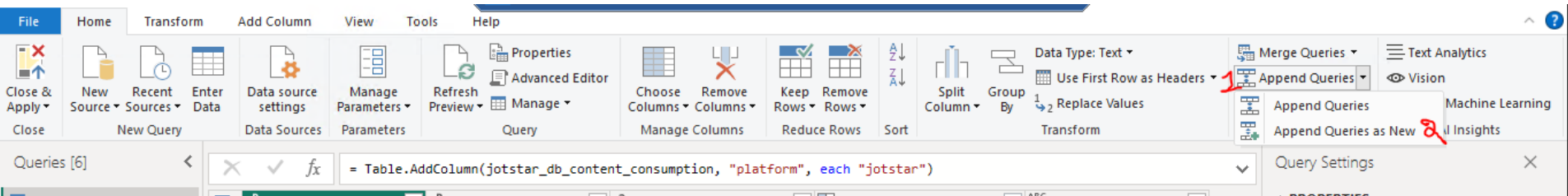
Main Query Editor:

- Table: `jotstar_db subscribers`
- Columns: `last_active_date`, `platform`

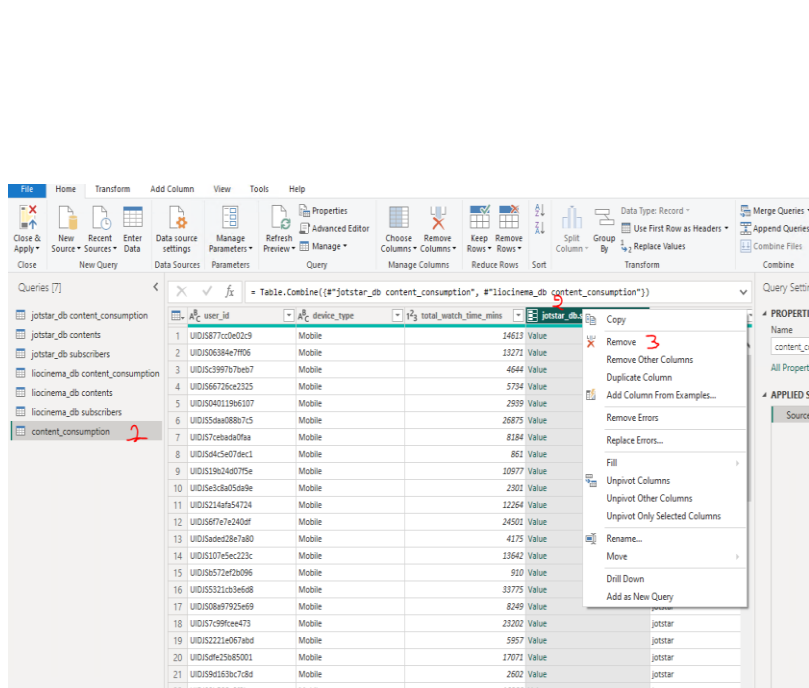


Data Transform – Step 2 – Creating a new master table

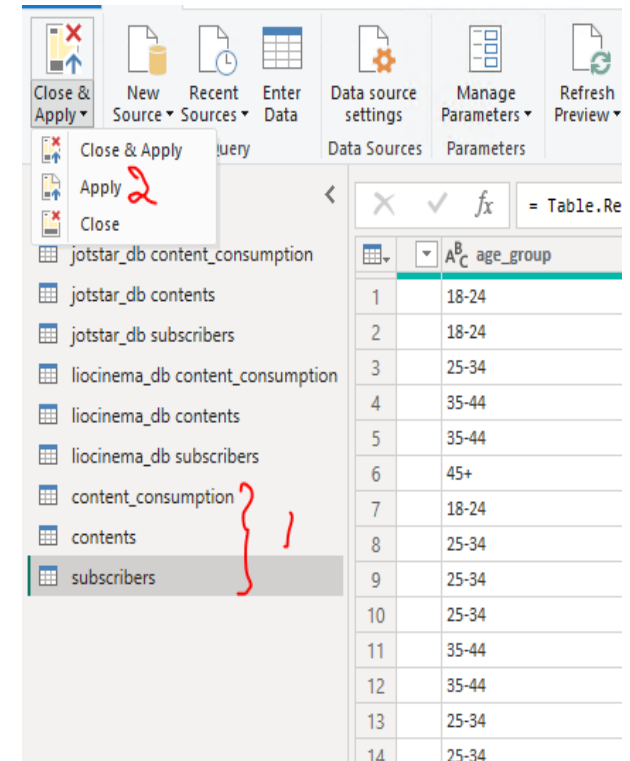
- Repeat the procedure as discussed below for matching tables one by one for both OTTs and create three additional master tables :



- Delete the subscriber info column created in master table content_consumption which is not required in processing:



The screenshot shows the Power Query Editor interface. The 'Queries' pane on the left lists several tables, with 'content_consumption' selected and marked with a red '2'. The main area displays a table with columns: 'user_id', 'device_type', and 'total_watch_time_mins'. A context menu is open over the table, showing various actions. The 'Remove' option is highlighted with a red '2'.

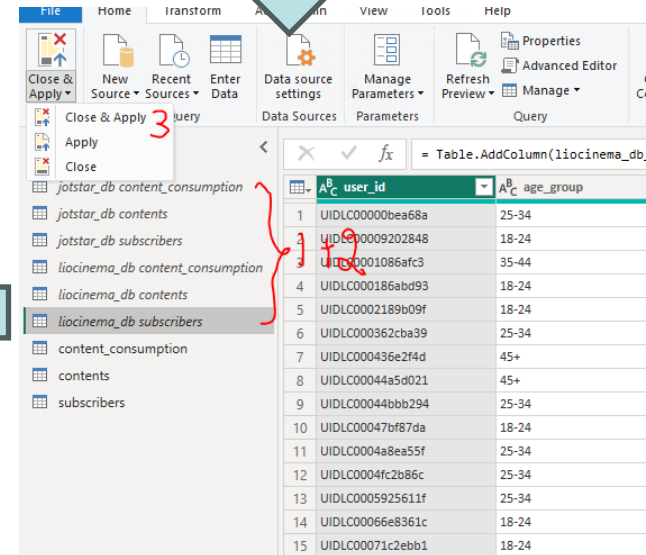
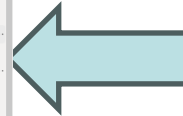
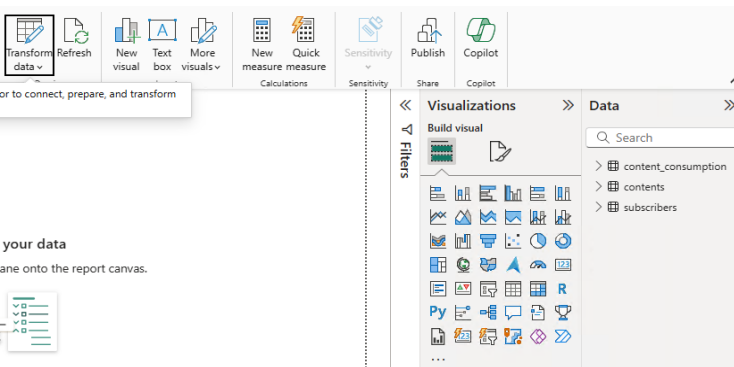
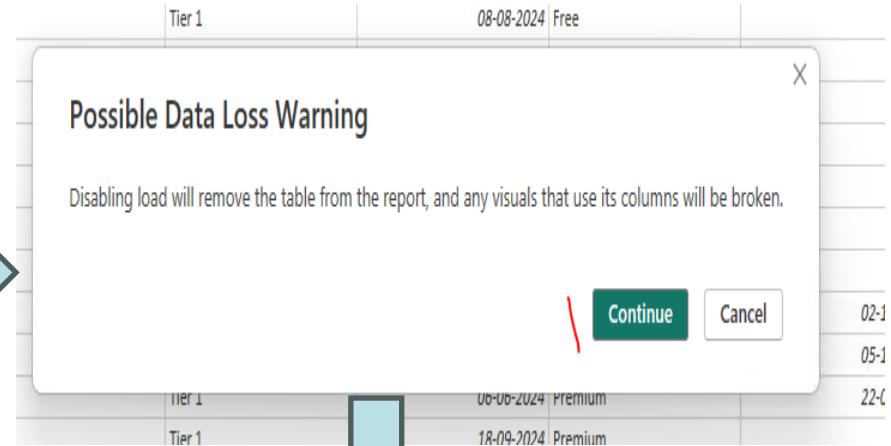
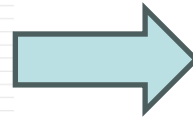
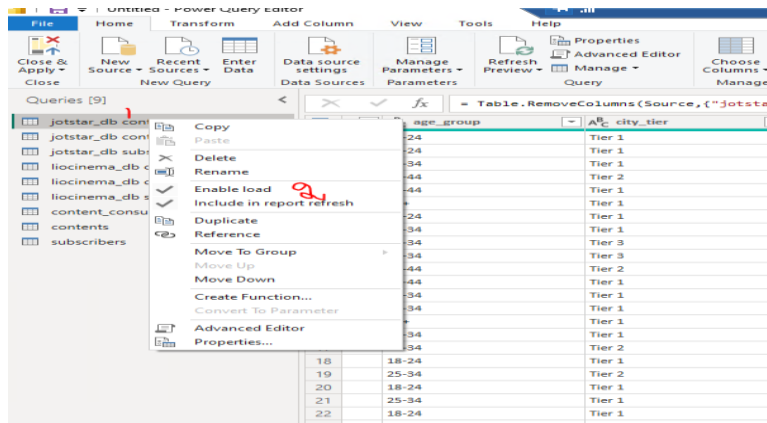


The screenshot shows the Power Query Editor interface. The 'Queries' pane on the left lists several tables, with 'content_consumption' selected and marked with a red '2'. The main area displays a table with columns: 'user_id', 'device_type', and 'total_watch_time_mins'. A context menu is open over the table, showing various actions. The 'Remove' option is highlighted with a red '2'.



Data Transform – Step 3 – Disable the redundant tables from the data load

- Repeat the shown procedure for 6 tables 3 from each ott platform:





Data Transform – Step 4 – Analyzing the column type and its properties

- Repeat the shown procedure for 6 tables 3 from each ott platform:

Queries [9]

- jotstar_db content_consumption
- jotstar_db contents
- jotstar_db subscribers
- liocinema_db content_consumption
- liocinema_db contents
- liocinema_db subscribers
- content_consumption**
- contents
- subscribers

Formula Bar: = Table.RemoveColumns(Source,{"jotstar_db.subscribers", "liocinema_db.subscribers"})

	user_id	device_type	total_watch_time_mins	platform
	Valid 100%	Valid 100%	Valid 100%	Valid 100%
	Error 0%	Error 0%	Error 0%	Error 0%
	Empty 0%	Empty 0%	Empty 0%	Empty 0%
1	UIDJS877cc0e02c9	Mobile	14613	jotstar
2	UIDJS06384e7ff06	Mobile	13271	jotstar
3	UIDJSc3997b7beb7	Mobile	4644	jotstar
4	UIDJS66726ce2325	Mobile	5734	jotstar
5	UIDJS040119b6107	Mobile	2939	jotstar
6	UIDJS5daa088b7c5	Mobile	26875	jotstar
7	UIDJS7cebada0faa	Mobile	8184	jotstar
8	UIDJSd4c5e07dec1	Mobile	861	jotstar
9	UIDJS19b24d07f5e	Mobile	10977	jotstar
10	UIDJSe3c8a05da9e	Mobile	2301	jotstar

Query Settings

PROPERTIES

Name: content_consumption

APPLIED STEPS

- Source
- Removed Columns



Data Transform – Step 4 – Analyzing the column type and change if required

- Repeat the shown procedure for 6 tables 3 from each ott platform:

Untitled - Power Query Editor

File Home Transform Add Column View Tools Help

Query Settings

Layout Data Preview Columns Parameters Advanced Dependencies

Queries [9]

- jotstar_db content_consumption
- jotstar_db contents
- jotstar_db subscribers
- liocinema_db content_consumption
- liocinema_db contents
- liocinema_db subscribers
- content_consumption
- contents
- subscribers

Table: RemoveColumns(Source, {"jotstar_db.subscribers", "liocinema_db.subscribers"})

	user_id	device_type	total_watch_time_mins	platform
	Valid 100%	Valid 100%	Valid 100%	1.2 Decimal Number
	Error 0%	Error 0%	Error 0%	\$ Fixed decimal number
	Empty 0%	Empty 0%	Empty 0%	123 Whole Number
				% Percentage
				Date/Time
				Date
				Time
				Date/Time/Timezone
				Duration
				ABC Text 3
				True/False
				Binary
				Using Locale...
1	UIDJS877cc0e02c9	Mobile	14613	jotstar
2	UIDJS06384e7ff06	Mobile	13271	jotstar
3	UIDJSc3997b7beb7	Mobile	4644	jotstar
4	UIDJS66726ce2325	Mobile	5734	jotstar
5	UIDJS040119b6107	Mobile	2939	jotstar
6	UIDJS5daa088b7c5	Mobile	26875	jotstar
7	UIDJS7cebada0faa	Mobile	8184	jotstar
8	UIDJSd4c5e07dec1	Mobile	861	jotstar
9	UIDJS19b24d07f5e	Mobile	10977	jotstar
10	UIDJSe3c8a05da9e	Mobile	2301	jotstar
11	UIDJS214afa54724	Mobile	12264	jotstar
12	UIDJS6f7e7e240df	Mobile	24501	jotstar
13	UIDJSaded28e7a80	Mobile	4175	jotstar
14	UIDJS107e5ec223c	Mobile	13642	jotstar
15	UIDJSb572ef2b096	Mobile	910	jotstar



Data Transform – Step 4 – Analyzing the column type and change if required

- Repeat the shown procedure for 4 columns in subscribers:

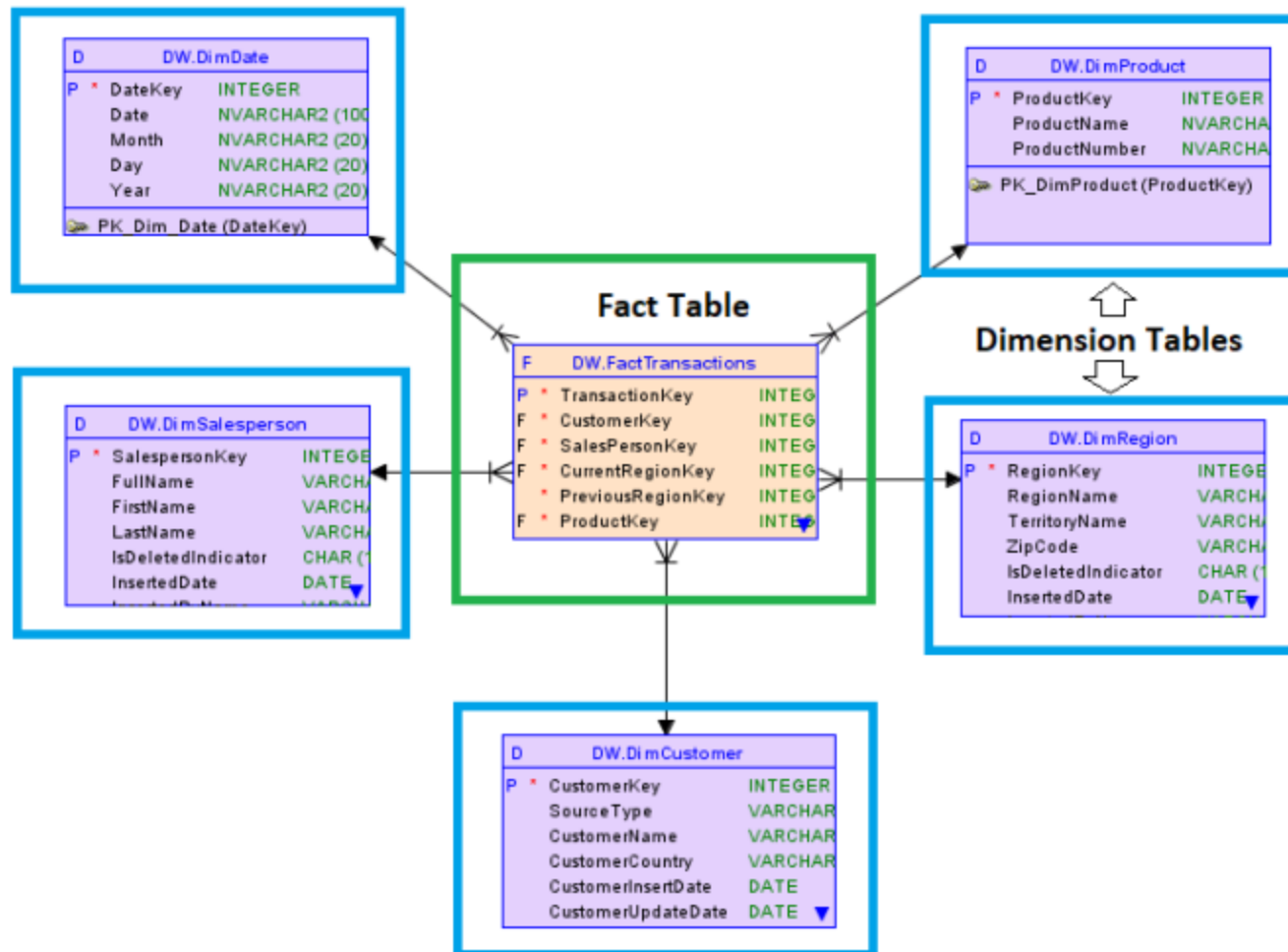
The screenshot displays the Power Query Editor interface. The 'Transform' tab is active, and the 'Data Preview' pane shows the 'subscribers' table. The 'subscription_date' column is selected, and a context menu is open, showing the 'Date' option. The 'last_active_date' column is also selected, and a context menu is open, showing the 'Date' option. The 'plan_change_date' column is also selected, and a context menu is open, showing the 'Date' option. The 'new_subscription_plan' column is also selected, and a context menu is open, showing the 'Text' option. The 'platform' column is also selected, and a context menu is open, showing the 'Text' option. The 'Query Settings' pane on the right shows the 'Properties' tab, with the 'Name' field set to 'subscribers'. The 'Applied Steps' pane shows the 'Removed Columns' step.

Table: RemoveColumns(Source,{"jotstar_db.content_consumption", "liocinema_db.content_consumption"})

	city_tier	subscription_date	subscription_plan	last_active_date	plan_change_date
1	Tier 1	18-09-2024	Premium	18-09-2024	18-09-2024
2	Tier 1	14-05-2024	Free	14-05-2024	14-05-2024
3	Tier 1	13-10-2024	VIP	13-10-2024	13-10-2024
4	Tier 1	19-11-2024	VIP	19-11-2024	19-11-2024
5	Tier 1	25-04-2024	VIP	25-04-2024	25-04-2024
6	Tier 1	10-08-2024	Free	10-08-2024	10-08-2024
7	Tier 1				
8	Tier 1				
9	Tier 1				
10	Tier 1				
11	Tier 1				
12	Tier 1				
13	Tier 1				
14	Tier 1				
15	Tier 1				
16	Tier 1				
17	Tier 1				
18	Tier 1				

Table: TransformColumnTypes(#"Removed Columns",{"subscription_date", type date}, {"last_active_date", type date}, {"plan_change_date", type date}, {"new_subscription_plan", type text}, {"platform", type text})

	subscription_plan	last_active_date	plan_change_date	new_subscription_plan	platform
1	Premium	18-09-2024	18-09-2024	Premium	jotstar
2	Free	14-05-2024	14-05-2024	Free	jotstar
3	Free	13-10-2024	13-10-2024	Free	jotstar
4	VIP	19-11-2024	19-11-2024	VIP	jotstar
5	Premium	25-04-2024	25-04-2024	Premium	jotstar
6	Free	10-08-2024	10-08-2024	Free	jotstar
7	Premium			Premium	jotstar
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					





Fact Table:

- A fact table is a database table that contains quantitative data (metrics or measures) about business processes or events, such as sales, orders, or website visits, along with foreign keys that connect to dimension tables.
- It forms the core of a Star or Snowflake Schema, enabling analysis by linking measurable data to descriptive attributes in dimension tables.

Characteristics

- **Quantitative Data:** Stores numerical data (e.g., sales amounts, quantities) used for calculations like sums, averages, or counts.
- **Foreign Keys:** Contains columns that reference primary keys in dimension tables, defining relationships (e.g., one-to-many in Star Schema).
- **Large Volume:** Typically has many rows, as it captures transactional or event-level data.
- **Sparse or Dense:** Can be sparse (few measures per row) or dense (many measures), depending on the business process.
- **Additive or Semi-Additive:** Measures are often additive (e.g., sales amount can be summed across dimensions) or semi-additive (e.g., account balances summed across some dimensions but not time).
- **Central to Schema:** Sits at the center of a Star Schema or connects to normalized dimensions in a Snowflake Schema.



Content

- **Measures:** Numerical values representing business metrics (e.g., SalesAmount, QuantitySold, Profit).
- **Foreign Keys:** Columns that link to primary keys in dimension tables (e.g., ProductID, CustomerID, DateID).
- **Metadata (Optional):** Occasionally includes non-key attributes like transaction IDs or timestamps for auditing, though these are less common.

Purpose

- To store transactional or event-based data for analytical processing, enabling aggregations, filtering, and reporting in tools like Power BI.
- To provide a foundation for business intelligence tasks, such as calculating total sales, average order value, or trends over time.
- To link quantitative data to descriptive dimensions for slicing and dicing (e.g., sales by product or region).



Size

- **Large:** Fact tables are typically the largest tables in a data warehouse, containing millions or billions of rows, as they store every instance of a business event or transaction.
- **Factors Affecting Size:**
 - Granularity: Finer granularity (e.g., per transaction) results in more rows.
 - Time Span: Longer historical data increases row count.
 - Number of Measures: More measures per row increase storage but not row count.
- In Power BI, fact tables are optimized for import or DirectQuery modes, with compression techniques reducing memory usage.

Granularity

- **Definition:** The level of detail in the fact table, determined by the combination of foreign keys and the measures they represent.
- **Examples:**
 - A sales fact table might have a granularity of one row per sale transaction (e.g., per OrderID, DateID, ProductID).
 - A website analytics fact table might have a granularity of one row per page view (e.g., per UserID, DateID, PageID).
- **Impact:** Granularity affects the size of the fact table and the types of analyses possible. Finer granularity (e.g., per transaction) allows detailed analysis but increases storage needs, while coarser granularity (e.g., daily aggregates) reduces size but limits detail.



Keys

- **Foreign Keys:** Columns that reference primary keys in dimension tables (e.g., ProductID links to the Product table's primary key). These define relationships, typically one-to-many in Star Schema or many-to-one in Snowflake Schema.
- **No Primary Key (Typically):** Fact tables often lack a primary key, as rows are not always uniquely identified by a single column. However, a composite key (combination of foreign keys) may define uniqueness at the granularity level.
- **Surrogate Keys (Optional):** In some cases, a fact table may include a unique transaction ID or surrogate key for auditing or tracking purposes.

Use Case

- **Retail:** Tracking sales transactions to analyze revenue by product, store, or time period.
- **Finance:** Recording account transactions to calculate balances or interest over time.
- **E-Commerce:** Logging website orders to measure order value, conversion rates, or customer behavior.
- **Healthcare:** Storing patient visits to analyze treatment costs or visit frequency by department.
- **Example in Power BI:** A sales dashboard showing total sales by region, product category, and month, using a fact table linked to Product, Customer, and Date dimensions.



Keys

- **Foreign Keys:** Columns that reference primary keys in dimension tables (e.g., ProductID links to the Product table's primary key). These define relationships, typically one-to-many in Star Schema or many-to-one in Snowflake Schema.
- **No Primary Key (Typically):** Fact tables often lack a primary key, as rows are not always uniquely identified by a single column. However, a composite key (combination of foreign keys) may define uniqueness at the granularity level.
- **Surrogate Keys (Optional):** In some cases, a fact table may include a unique transaction ID or surrogate key for auditing or tracking purposes.

Use Case

- **Retail:** Tracking sales transactions to analyze revenue by product, store, or time period.
- **Finance:** Recording account transactions to calculate balances or interest over time.
- **E-Commerce:** Logging website orders to measure order value, conversion rates, or customer behavior.
- **Healthcare:** Storing patient visits to analyze treatment costs or visit frequency by department.
- **Example in Power BI:** A sales dashboard showing total sales by region, product category, and month, using a fact table linked to Product, Customer, and Date dimensions.



Example Columns

- **For a Sales Fact Table in a retail scenario:**
- **Foreign Keys:**
 - ProductID (links to Product dimension)
 - CustomerID (links to Customer dimension)
 - DateID (links to Date dimension)
 - StoreID (links to Store dimension)
- **Measures:**
 - SalesAmount (e.g., \$100.50)
 - QuantitySold (e.g., 5 units)
 - Discount (e.g., \$10.00)
 - Profit (e.g., \$30.00)
- **Optional Metadata:**
 - TransactionID (e.g., T12345 for auditing)
 - Timestamp (e.g., 2025-07-29 14:30:00)



Types of Operations Performed

- **Read Operations:**
 - **SELECT Queries:** Retrieve data for reporting (e.g., `SELECT SUM(SalesAmount) FROM Sales WHERE DateID = 20250729`).
 - **Aggregations:** Calculate sums, averages, counts, or other metrics (e.g., total sales, average order value).
 - **Filtering:** Apply filters based on dimension attributes (e.g., sales for a specific product category).
- **Joins:** Join fact table with dimension tables to enrich data (e.g., join Sales with Product to get ProductName).
- **DAX Calculations in Power BI:**
 - **Measures:** Total Sales = `SUM(Sales[SalesAmount])`
 - **Calculated Columns:** Compute derived values, like Profit Margin = `Sales[Profit] / Sales[SalesAmount]`.
 - **Time Intelligence:** Year-to-Date Sales = `TOTALYTD(SUM(Sales[SalesAmount]), Date[Date])`.
- **Slicing/Dicing:** Analyze data by different dimensions (e.g., sales by region or product).
- **Write Operations (Rare):** Insert new transactional data in ETL processes, typically outside Power BI (e.g., in SQL Server).



Other Properties

- **Types of Fact Tables:**
 - **Transactional:** Records individual events (e.g., each sale). Most common, with fine granularity.
 - **Snapshot:** Captures data at a point in time (e.g., daily account balances). Semi-additive measures are common.
 - **Accumulating Snapshot:** Tracks lifecycle events of a process (e.g., order stages from placement to delivery). Includes multiple date foreign keys.
 - **Factless:** Contains only foreign keys, no measures, used to track events or relationships (e.g., student attendance by class and date).



Practical Example in Power BI

- **Scenario:** A retail company tracks sales data.
- **Fact Table:** Sales
 - Columns: SalesAmount, QuantitySold, ProductID, CustomerID, DateID, StoreID
 - Granularity: One row per sale transaction.
- **Dimension Tables:**
 - Product: ProductID (PK), ProductName, Category
 - Customer: CustomerID (PK), CustomerName, Region
 - Date: DateID (PK), Date, Month, Year
 - Store: StoreID (PK), StoreName, Location
- **Relationships:** One-to-many from each dimension (e.g., Product[ProductID] to Sales[ProductID]).
- **Power BI Usage:**
 - Create a measure: Total Sales = SUM(Sales[SalesAmount]).
 - Build a visual: Bar chart showing sales by product category.
 - Filter by Date[Year] or Customer[Region] for dynamic analysis.

Dimension Table:

- A dimension table is a database table that contains descriptive, qualitative attributes (e.g., names, categories, or dates) used to describe and categorize the facts stored in a fact table.
- It connects to fact tables via primary and foreign key relationships, providing context for metrics like sales, orders, or visits in analytical systems.

Characteristics

- **Descriptive Data:** Stores qualitative attributes (e.g., product names, customer regions) rather than numerical measures.
- **Denormalized (Star Schema):** In a Star Schema, dimension tables are typically denormalized, consolidating related attributes into a single table to reduce joins.
- **Normalized (Snowflake Schema):** In a Snowflake Schema, dimension tables are split into multiple normalized tables to eliminate redundancy, forming hierarchical structures.
- **Smaller Size:** Contains fewer rows compared to fact tables, as they represent unique entities (e.g., products, customers).
- **Primary Key:** Includes a unique identifier (primary key) for each row, used to link to fact tables.
- **Hierarchical Structure:** Often includes attributes that form hierarchies (e.g., Year → Quarter → Month in a Date dimension).



Content

- **Descriptive Attributes:** Textual or categorical data (e.g., ProductName, CustomerRegion, DateMonth).
- **Primary Key:** A unique identifier for each row (e.g., ProductID, CustomerID).
- **Hierarchical Attributes:** Columns that support drill-down analysis (e.g., Category → Subcategory in a Product dimension).
- **Metadata (Optional):** Additional fields like audit columns (e.g., CreatedDate) or flags (e.g., IsActive).

Purpose

- To provide context for fact table measures, enabling meaningful analysis (e.g., sales by product category or region).
- To support filtering, grouping, and sorting in reports and dashboards in Power BI.
- To define hierarchies for drill-down or roll-up analysis (e.g., analyzing sales by year, then quarter, then month).
- To ensure data consistency by storing unique descriptive values referenced by fact tables.



Granularity

- **Definition:** The level of detail in the dimension table, typically one row per unique entity (e.g., one row per product, customer, or date).
- **Examples:**
 - A Product dimension might have one row per unique product (granularity = product level).
 - A Date dimension might have one row per day (granularity = daily level).
- **Impact:** Granularity determines how finely data can be filtered or grouped. Coarser granularity (e.g., one row per product category) limits detail but reduces table size.

Size

- **Smaller than Fact Tables:** Dimension tables typically have far fewer rows (hundreds to thousands) compared to fact tables (millions to billions).
- **Factors Affecting Size:**
 - Number of Unique Entities: A Product dimension with 1,000 products has 1,000 rows.
 - Attribute Count: More descriptive columns increase storage but not row count.
 - Normalization (Snowflake Schema): Normalized dimension tables are smaller individually but increase the total number of tables.
- In Power BI, dimension tables are compressed in import mode, minimizing memory usage.

Keys

- **Primary Key:** A unique identifier for each row (e.g., ProductID, CustomerID). Often a surrogate key (e.g., auto-incremented integer) for performance.
- **Foreign Key Relationship:** The primary key in a dimension table is referenced by foreign keys in the fact table, typically forming a one-to-many relationship in a Star Schema.
- **Hierarchical Keys (Snowflake Schema):** In normalized dimensions, additional keys (e.g., CategoryID, SubcategoryID) link related tables.

Use Case

- **Retail:** A Product dimension to filter sales by category or brand in a Power BI dashboard.
- **Finance:** A Date dimension to analyze financial performance by quarter or year.
- **E-Commerce:** A Customer dimension to segment orders by region or customer type.
- **Healthcare:** A Department dimension to group patient visits by specialty or location.
- **Example in Power BI:** A sales report showing revenue by product category (from Product dimension) and month (from Date dimension), linked to a Sales fact table.



Example Columns

- For a retail scenario, example dimension tables and their columns:

- Product Dimension**

Product ID	Product Name	Category	Sub-category	Price	Brand
1	Laptop	Electronics	Computers	\$999.99	TechCorp

- Customer Dimension**

Customer ID	Customer Name	Region	Email	Customer Type
C001	John Doe	North America	john.doe@email.com	Premium

- Date Dimension**

DateID	Date	Month	Quarter	Year	IsWeekend
20250729	2025-07-29	July	Q3	2025	No

- Store Dimension**

StoreID	StoreName	City	State	Manager
S01	Downtown Store	New York	NY	Jane Smith



Types of Dimension Tables:

- **Conformed Dimensions:** Shared across multiple fact tables (e.g., a Date dimension used for Sales and Inventory fact tables).
- **Junk Dimensions:** Combine low-cardinality attributes (e.g., flags like IsActive, IsDiscounted) into one table to simplify the model.
- **Degenerate Dimensions:** Attributes stored in the fact table instead of a separate dimension (e.g., OrderID in a Sales fact table).
- **Role-Playing Dimensions:** A single dimension used in multiple contexts (e.g., a Date dimension for OrderDate and ShipDate in a Sales fact table).

Cardinality in Relationships:

- Typically forms **one-to-many** relationships with fact tables (dimension on the “one” side).
- In Power BI, set cardinality in **Manage Relationships** or Model View (e.g., Product[ProductID] to Sales[ProductID]).
- Use **Single** cross-filter direction for most scenarios to optimize performance and avoid ambiguity.



Practical Example in Power BI

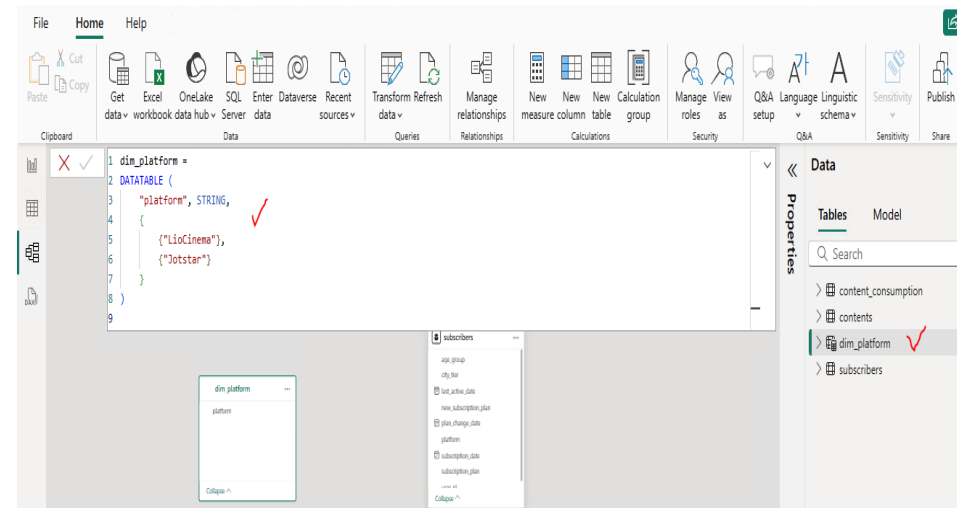
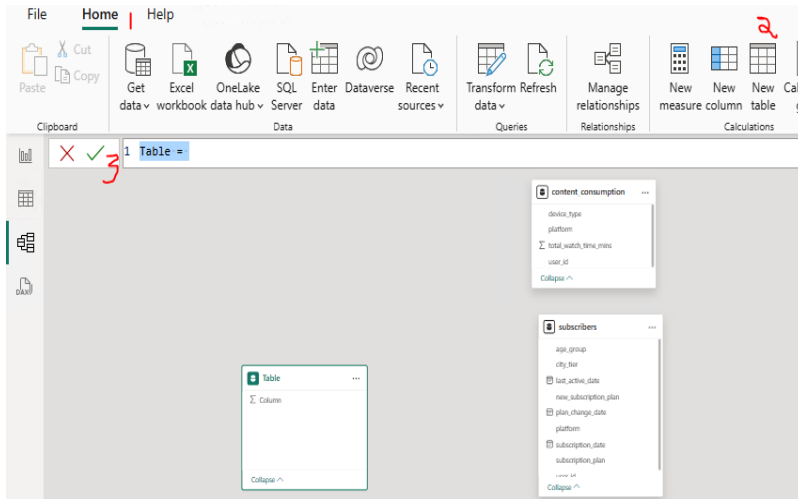
- **Scenario:** A retail company analyzes sales data.
- **Dimension Table:** Product
 - Columns: ProductID (PK), ProductName, Category, Subcategory, Price
 - Example Data:
 - ProductID: 1, ProductName: "Laptop", Category: "Electronics", Subcategory: "Computers", Price: \$999.99
 - ProductID: 2, ProductName: "Mouse", Category: "Electronics", Subcategory: "Accessories", Price: \$29.99
- **Fact Table:** Sales
 - Columns: SalesAmount, QuantitySold, ProductID, CustomerID, DateID
- **Relationships:** One-to-many from Product[ProductID] to Sales[ProductID].
- **Power BI Usage:**
 - Create a measure: Total Sales = SUM(Sales[SalesAmount]).
 - Build a visual: Bar chart showing sales by Product[Category].
 - Add a slicer for Product[Subcategory] to filter interactively.



- Different types of table: 1. fact table and 2. dimension table.
- Need for Dim table.
- Identification of dim table required.
- Dim table are derived from fact tables and they have unique entries.
- For our case we create following dim tables:
 - 1. Dim date
 - 2. Dim platform
 - 3. Dim device type
 - 4. Dim age group
 - 5. Dim Demography (city for our case)



Data Modelling – Creation of Dimension Table – dim_platform

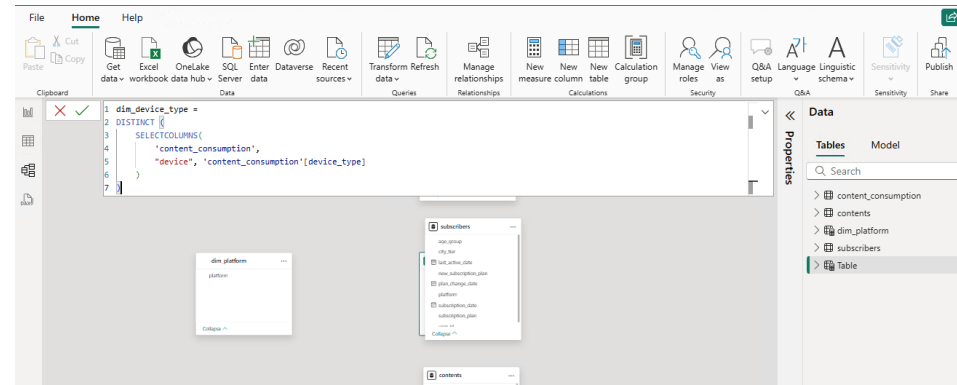
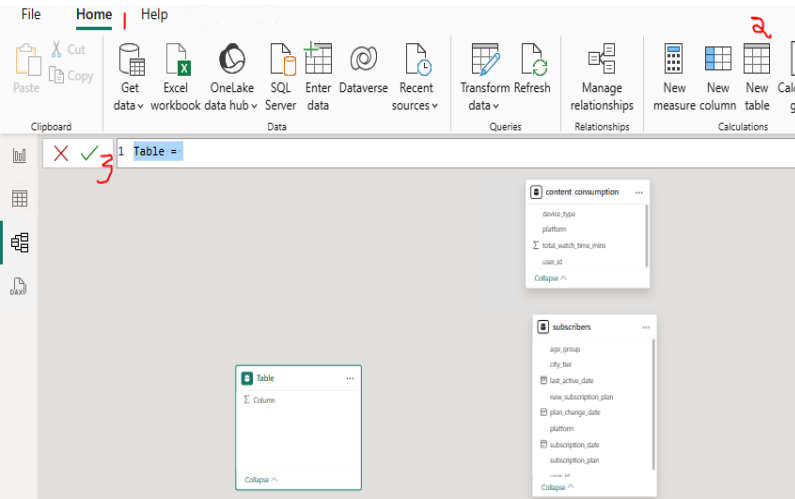


```
dim_platform =  
DATATABLE (  
    "platform", STRING,  
    {  
        {"LioCinema"},  
        {"Jotstar"}  
    }  
)
```

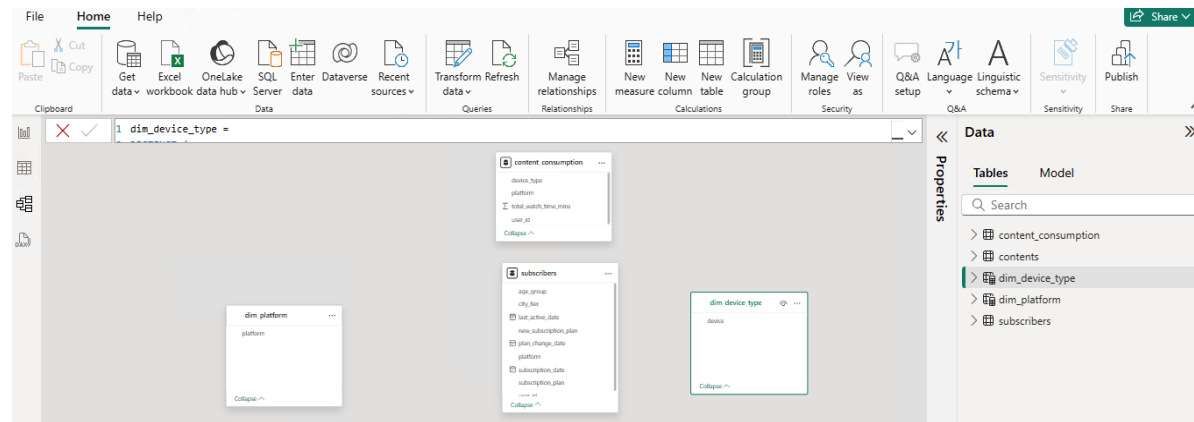
Help:
URL: <https://dax.guide/datatable/>
Search for datatable → use the syntax and refer examples



Data Modelling – Creation of Dimension Table – dim_device



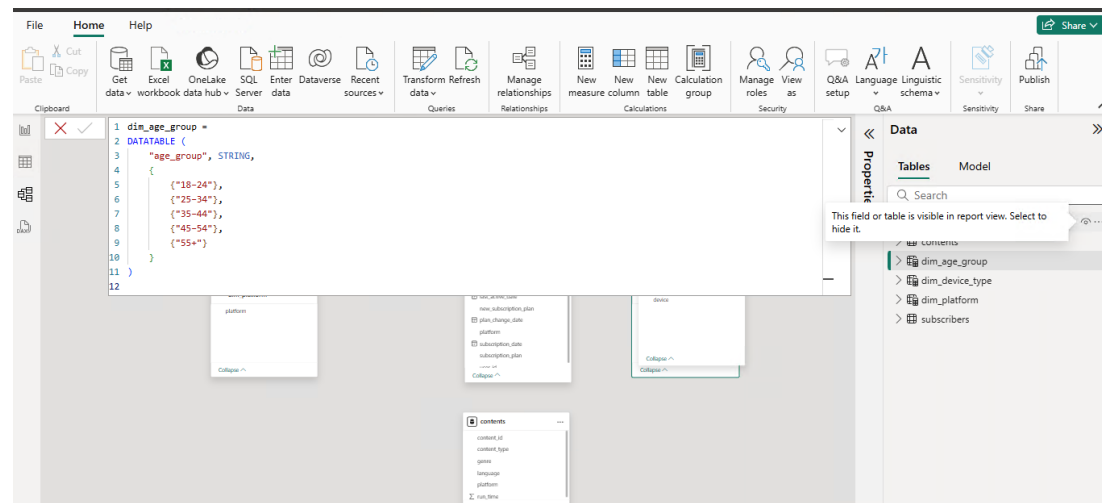
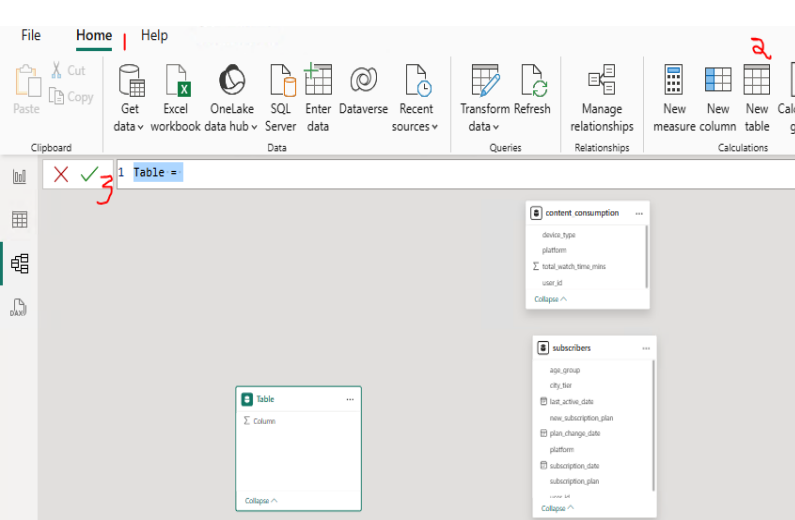
```
dim_device_type =  
DISTINCT (  
    SELECTCOLUMNS(  
        'content_consumption',  
        "device",  
        'content_consumption'[device_type]  
    )  
)
```



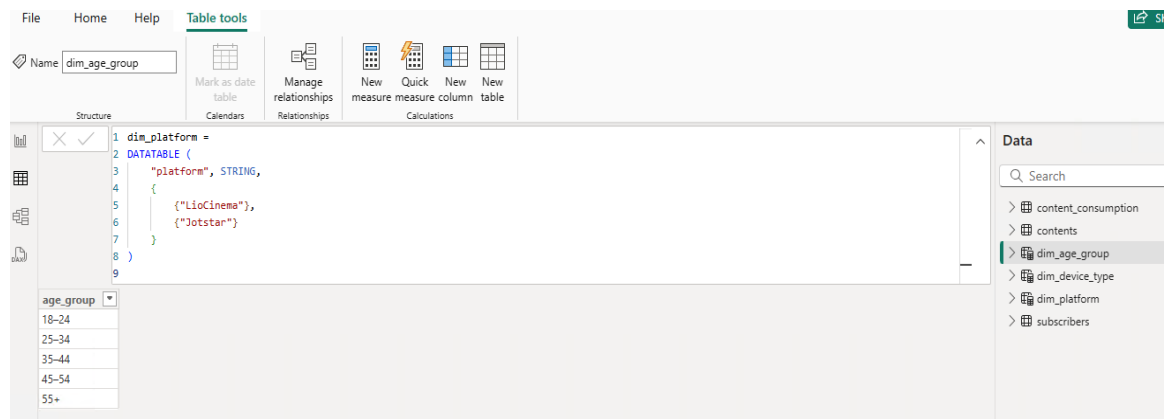
Help:
URL: <https://dax.guide/datatable/>
Search for datatable → use the syntax and refer examples



Data Modelling – Creation of Dimension Table – dim_agegroup



```
dim_age_group =  
DATATABLE (  
    "age_group", STRING,  
    {  
        {"18-24"},  
        {"25-34"},  
        {"35-44"},  
        {"45-54"},  
        {"55+"}  
    }  
)
```



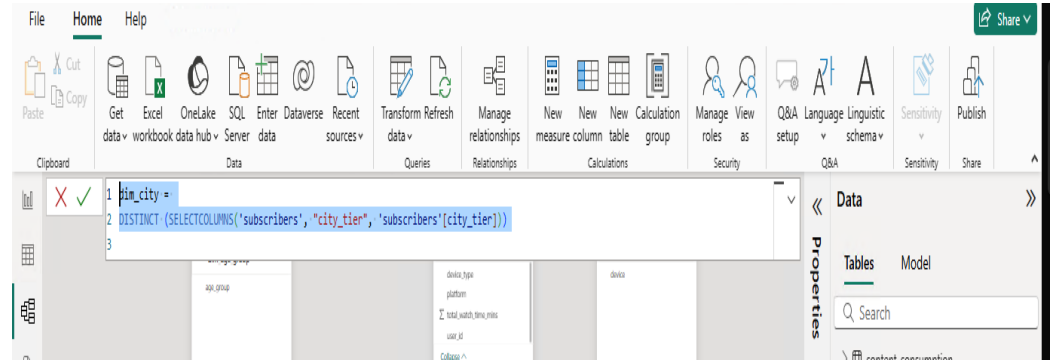
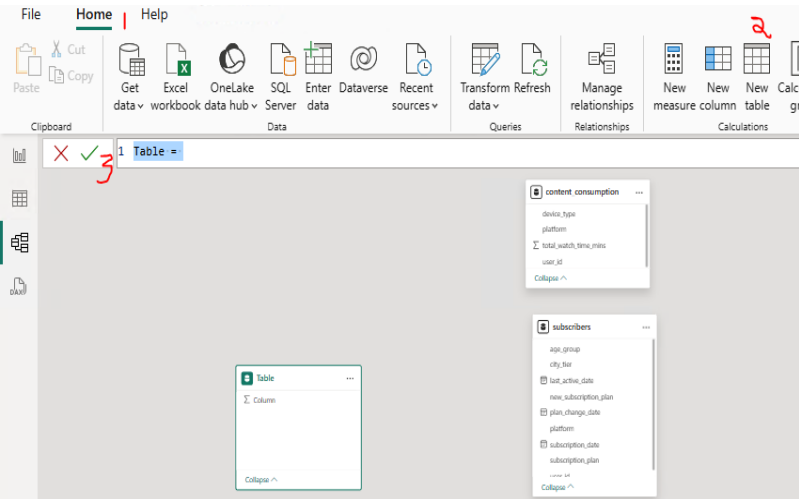
Help:

URL: <https://dax.guide/datatable/>

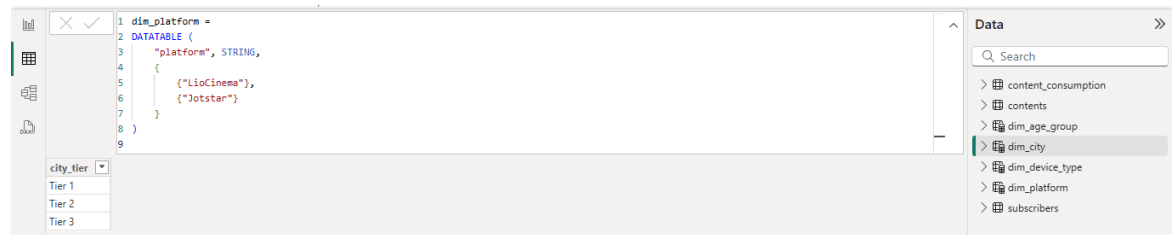
Search for datatable → use the syntax and refer examples



Data Modelling – Creation of Dimension Table – dim_city (explain as values are derived from existing table)



dim_city =
DISTINCT
(SELECTCOLUMNS('subscribers', "city_tier",
'subscribers'[city_tier]))



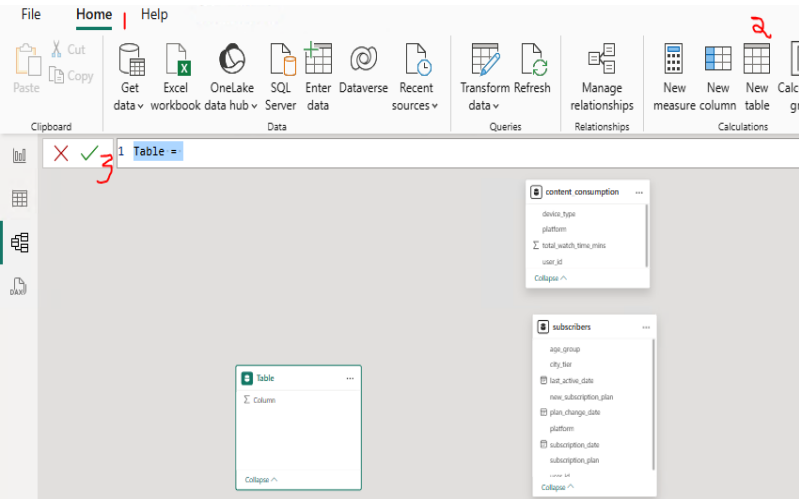
Help:

URL: <https://dax.guide/datatable/>

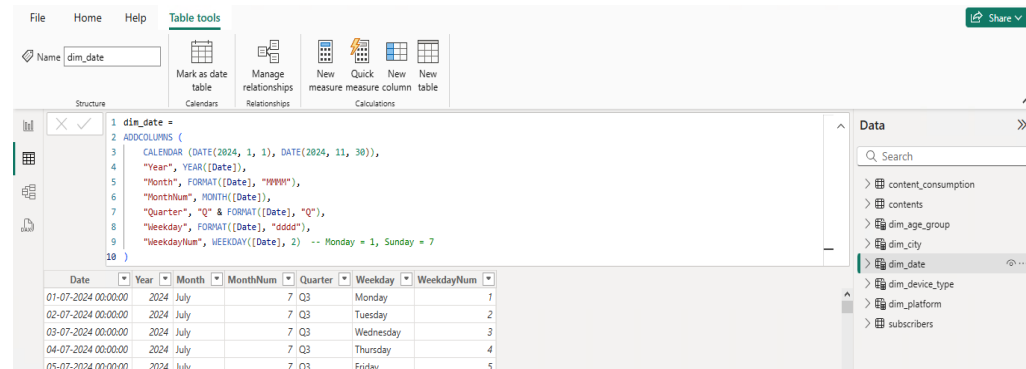
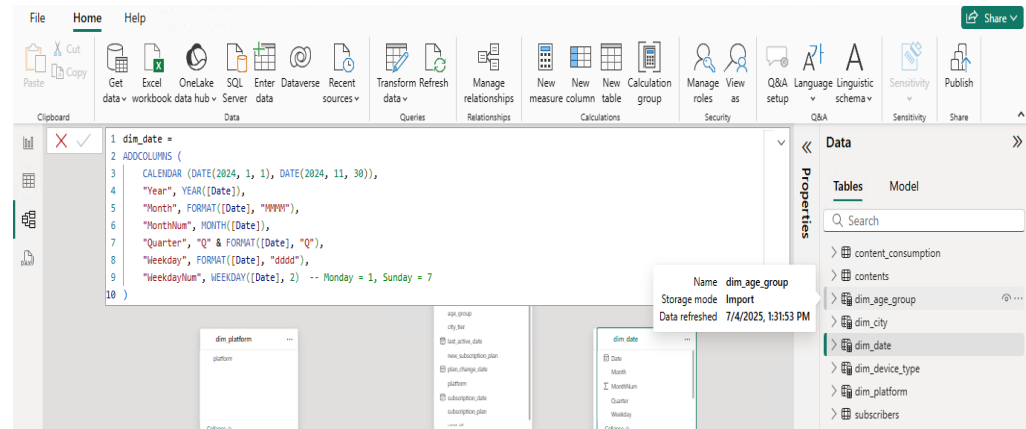
Search for datatable → use the syntax and refer examples



Data Modelling – Creation of Dimension Table – dim_date (explain as values are derived from existing table)



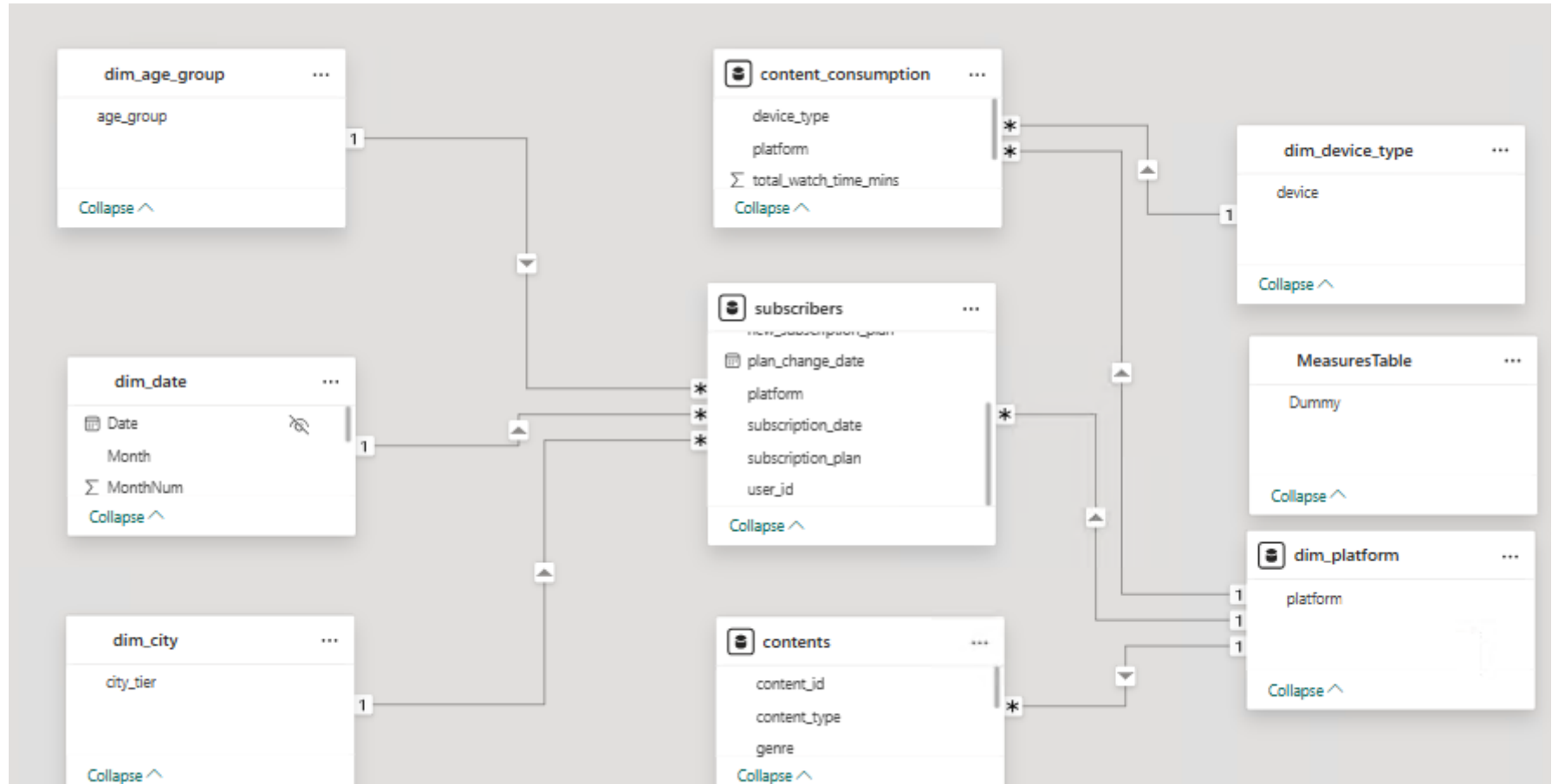
```
dim_date =  
ADDCOLUMNS (  
    CALENDAR (DATE(2024, 1, 1), DATE(2024, 11, 30)),  
    "Year", YEAR([Date]),  
    "Month", FORMAT([Date], "MMMM"),  
    "MonthNum", MONTH([Date]),  
    "Quarter", "Q" & FORMAT([Date], "Q"),  
    "Weekday", FORMAT([Date], "dddd"),  
    "WeekdayNum", WEEKDAY([Date], 2) -- Monday = 1, Sunday = 7  
)
```



Help:

URL: <https://dax.guide/datatable/>

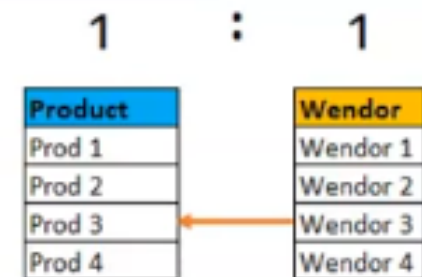
Search for datatable → use the syntax and refer examples



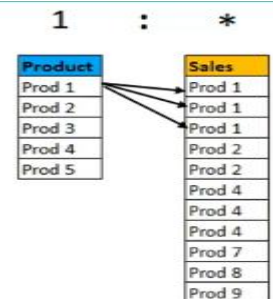


- In Power BI, cardinality refers to the type of relationship between two tables based on how their columns are related.
- Cardinality is critical when defining relationships, as it determines how Power BI handles data interactions between these tables.
- Types of Cardinality in Power BI:
 - One-to-One (1:1)
 - One-to-Many (1:*)
 - Many-to-One (*:1)
 - Many-to-Many (:)

- **Definition:** In this type of relationship, each row in the first table corresponds to exactly one row in the second table.
- **Example:** A table of employees and their corresponding user profiles (if each employee has only one profile).
- **Preferable When:** You have matching data in both tables with unique identifiers.
- **Use Case:** Two tables representing different sets of attributes for the same unique records (e.g., employee details in one table and employee access rights in another).
- **Performance:** It is generally efficient because Power BI doesn't have to handle many-to-many relationships.
- **Note:** This is less common in Power BI, as it often indicates redundant data that could be combined into a single table for simplicity.



- **Definition:** In this type, one row in the first table can relate to many rows in the second table, but each row in the second table can only relate to one row in the first table.
- **Example:** A table of customers (where each customer is unique) and a table of orders (where a customer can place multiple orders).
- **Preferable When:** This is the most common and preferred cardinality in Power BI. A dimension table (like customers, products) relates to a fact table (like orders, sales).
- **Use Case:** Sales reports where each product in a product table is related to multiple sales records in a sales table.
- **Common Scenario:** Relationships between lookup/dimension tables (like Products, Customers) and fact tables (like Sales, Transactions).
- **Performance:** Highly efficient and recommended for large datasets because it's the typical structure of star schema models.

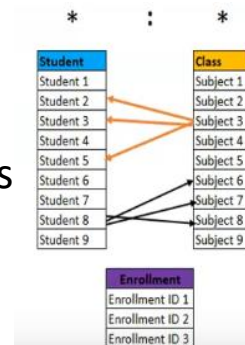




Many-to-One (*:1)

- **Definition:** Essentially the same as One-to-Many but from the perspective of the other table. Many rows in the first table relate to one row in the second table.
- **Example:** Same as the One-to-Many relationship example but reversed in direction (from the orders table to the customer table).
- **Preferable When:** Similar to One-to-Many, this is highly preferred in most use cases.
- **Use Case:** Useful when viewing the relationship from the fact table perspective.

- **Definition:** In this relationship, many rows in one table can relate to many rows in another table.
- **Example:** A table of students and a table of courses, where students can enroll in multiple courses and each course can have multiple students.
- **Not Preferable When:**
 - Many-to-Many relationships can lead to complex models and ambiguous queries
 - They are less efficient, potentially leading to slower performance.
 - Should be avoided if possible.
- **Use Case:** Sometimes unavoidable when working with certain datasets, such as many-to-many relationships between customers and orders for complex scenarios.
- **Performance:** Can cause slower report performance, particularly with large datasets.
- **Preferred Alternative:** Consider restructuring the model (e.g., using bridge tables) to eliminate direct many-to-many relationships.
- Power BI uses an intermediate table or composite models to handle this.



- **Preferred Cardinality:**

- One-to-Many (1:) or Many-to-One (:1) are typically preferred because they represent a clear, unambiguous relationship, leading to efficient performance.
- One-to-One (1:1) is acceptable but less common because most models involve some degree of aggregation.

- **Not Preferred Cardinality**

- Many-to-Many (::) relationships are not preferable due to the complexity they introduce into the model and potential performance issues. It's recommended to avoid this type of relationship whenever possible by restructuring the tables.

Create relationship

Select tables and columns that are related.

Product

ProductKey	Product	Standard Cost	Color	Subcategory	Category
215	Sport-100 Helmet, Black	\$12.0278	Black	Helmets	Accessories
216	Sport-100 Helmet, Black	\$13.8792	Black	Helmets	Accessories
217	Sport-100 Helmet, Black	\$13.0863	Black	Helmets	Accessories

Sales

ProductKey	ResellerKey	EmployeeKey	SalesTerritoryKey	SalesOrderNumber	Quantity	Unit Price
343	258	282	4	SO46960	1	\$4.70
325	258	282	4	SO46970	1	\$4.70
341	258	282	4	SO46970	1	\$4.70

Cardinality: One to many (1:*)

Cross filter direction: Single

☒ Make this relationship active

☐ Assume referential integrity

☐ Apply security filter in both directions

OK Cancel

Suggested Metrics and Analyses for Initial Insights

- The proposed metrics and analyses outlined below provide a foundation for deriving initial insights into the performance and user behavior of JioCinema and Hotstar from January to November 2024.
- You are encouraged to explore additional relevant metrics and analyses to uncover deeper trends and deliver more comprehensive findings to support the strategic merger and optimize the Jio-Hotstar platform's content and subscription strategies.

Total content items	Total users	Paid users	Paid users (%)
Active users	Inactive users	Inactive Rate (%)	Active Rate (%)
Upgraded users	Upgrade Rate (%)	Downgraded users	Downgrade Rate (%)
Total watch time (hrs)	Average watch time (hrs)	Monthly users Growth Rate (%)	Upgrade / Downgrade Rate (%)



KPI – Insights 1. Total content items

- Create a Blank Table for Measures

1. Create a Blank Table (No Data)

- Go to Modeling > New Table, and enter this DAX:
- **MeasuresTable = DATATABLE("Dummy", STRING, {{" "}})**
- This creates a one-row, one-column dummy table just to hold your measures. You can ignore the column — it's just a placeholder.

• -----

- Add Measures to the Blank Table

Total Content Items
3610

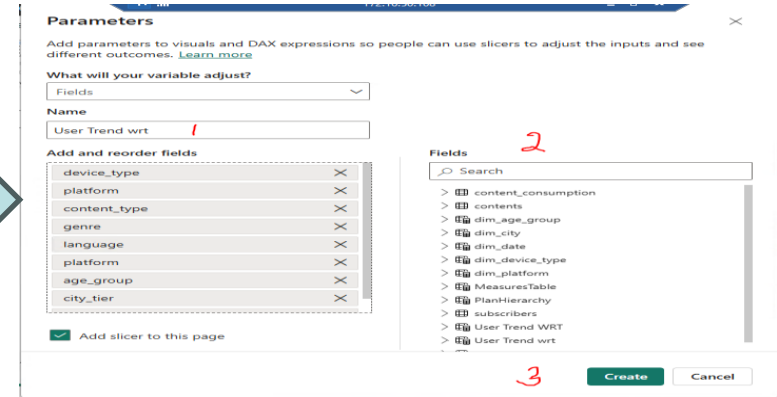
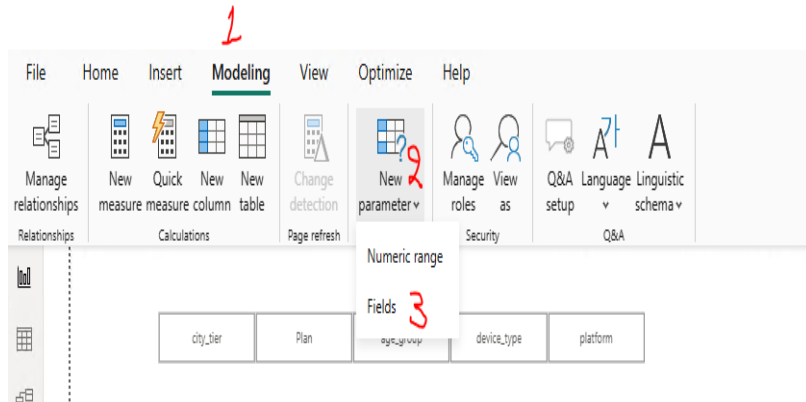
- To add measures to this table:
 - Click on the new table (MeasuresTable) in the Fields pane.
 - Go to Modeling > New Measure.
 - Enter your measure like this:
- **Total Content Items = CALCULATE(DISTINCTCOUNT(contents[content_id]))**
- This measure will now belong to the MeasuresTable, keeping it separate from your data tables.

KPI	DAX
Total content Items	Total Content Items = CALCULATE(DISTINCTCOUNT(contents[content_id]))
Total users	Total users = CALCULATE(DISTINCTCOUNT(subscribers[user_id]))
Paid users Count	Paid Users Count = CALCULATE(DISTINCTCOUNT('subscribers'[user_id]), FILTER('subscribers', COALESCE('subscribers'[new_subscription_plan], 'subscribers'[subscription_plan]) IN { "Basic", "VIP", "Premium" })))
paid user %	paid user % = [Paid Users Count]/[Total users] (Show then how to convert into %)
Active users, Inactive users	<p>Create a column in subscriber table as:</p> <pre>UserStatus = SWITCH(TRUE(), ISBLANK('subscribers'[last_active_date]), "Active", 'subscribers'[last_active_date] < TODAY() - 30, "Inactive")</pre> <p>-----create a dax-----</p> <pre>Active User Count = CALCULATE(DISTINCTCOUNT(subscribers[user_id]), 'subscribers'[UserStatus] = "Active")</pre> <p>-----</p> <pre>Inactive User Count = CALCULATE(DISTINCTCOUNT(subscribers[user_id]), 'subscribers'[UserStatus] = "Inactive")</pre>



KPI	DAX
Active rate %	Active rate % = DIVIDE([Active User Count],[Total users],0)
Inactive rate %	Inactive rate % = DIVIDE([Inactive User Count],[Total users],0)
Upgraded Users Count	<p>First create a table PlanHierarchy</p> <pre>DATATABLE ("Platform", STRING, "Plan", STRING, "Rank", INTEGER, { { "LioCinema", "Basic", 1 }, { "LioCinema", "Premium", 2 }, { "Jotstar", "VIP", 1 }, { "Jotstar", "Premium", 2 } })</pre> <p>-----</p> <p>Upgraded Users Count =</p> <pre>CALCULATE(DISTINCTCOUNT('subscribers'[user_id]), FILTER('subscribers', NOT ISBLANK('subscribers'[new_subscription_plan]) && LOOKUPVALUE(PlanHierarchy[Rank], PlanHierarchy[Platform], 'subscribers'[platform], PlanHierarchy[Plan], 'subscribers'[new_subscription_plan])) > LOOKUPVALUE(PlanHierarchy[Rank], PlanHierarchy[Platform], 'subscribers'[platform], PlanHierarchy[Plan], 'subscribers'[subscription_plan]))</pre>

User Behaviour Analysis using parameters



UserTrend Analysis

device_type	content_type	language	age_group	device
platform	genre	platform	city_tier	



Queries & Suggestions



THANK YOU...