IBM NAAN MUDHALVAN

CLOUD APPLICATION DEVELOPMENT

PROJECT : Data Warehousing with IBM Cloud Db2Warehouse

Phase 4: Development Part 2

In this part you will continue building your project.

Continue building the data warehouse by implementing ETL processes and enabling data exploration.

Implement ETL processes to extract, transform, and load data into the data warehouse.

Enable data architects to explore and analyze data within Db2 Warehouse using SQL queries and analysis techniques.

In Phase 4 of project, need to focus on implementing ETL (Extract, Transform, Load) processes to populate your data warehouse, which appears to be IBM Db2 Warehouse, and enable data architects to explore and analyze data using SQL queries and analysis techniques.

ETL

ETL, which stands for extract, transform and load, is a data integration process that combines data from multiple data sources into a single, consistent data store that is loaded into a data warehouse or other target system.

As the databases grew in popularity in the 1970s, ETL was introduced as a process for integrating and loading data for computation and analysis, eventually becoming the primary method to process data for data warehousing projects.

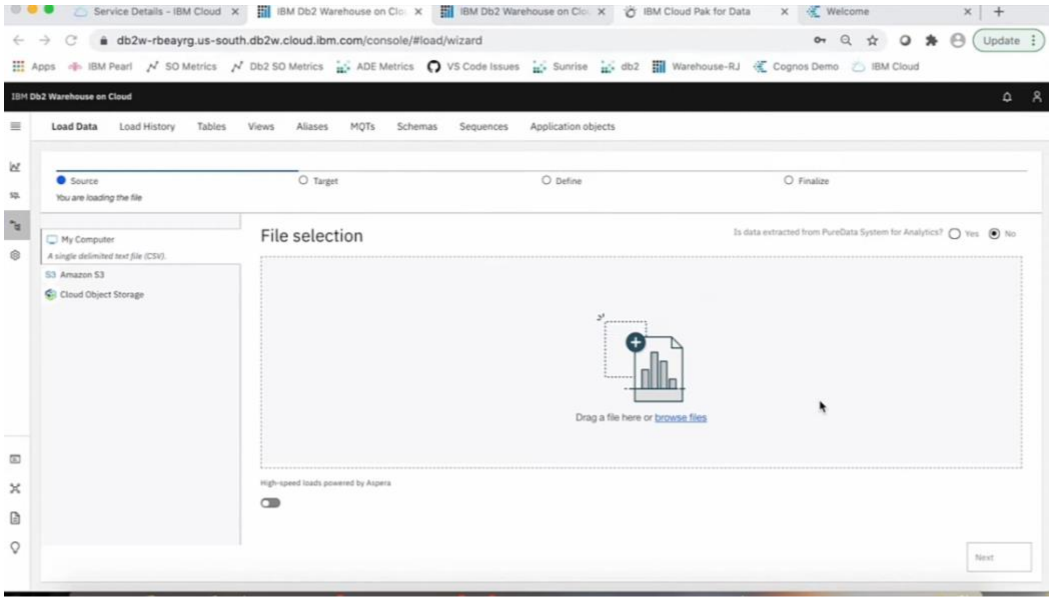
ETL provides the foundation for data analytics and machine learning workstreams. Through a series of business rules, ETL cleanses and organizes data in a way which addresses specific business intelligence needs, like monthly reporting, but it can also tackle more advanced analytics, which can improve back-end processes or end user experiences. ETL is often used by an organization to:

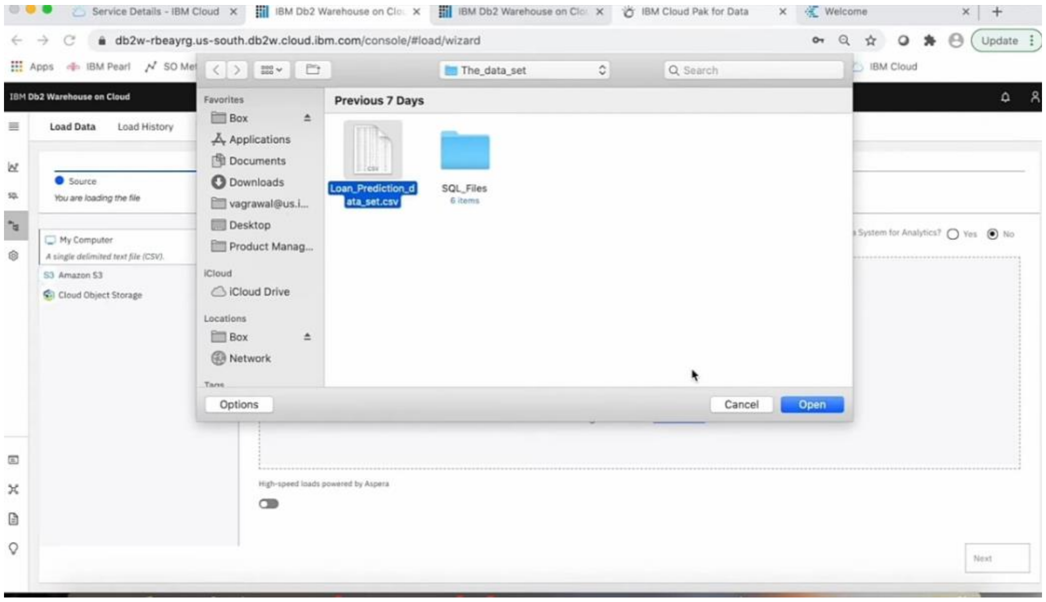
- Extract data from legacy systems
- Cleanse the data to improve data quality and establish consistency
- Load data into a target database

Here are some key steps you might consider:

1. **Data Extraction:**

   Extract data from various source systems. This could involve databases, files, APIs, or other data repositories.


db2w-rbeayrg.us-south.db2w.cloud.ibm.com/console/#load/wizard
IBM Db2 Warehouse on Cloud
Load Data
Load History
Tables
Views
Aliases
MQTs
Schemas
Sequences
Application objects
Source
You are loading the file
Target
Define
Finalize
My Computer
A single delimited text file (CSV).
Amazon S3
Cloud Object Storage
File selection
Is data extracted from PureData System for Analytics? Yes No
Drag a file here or browse files
High-speed loads powered by Aspera
Next

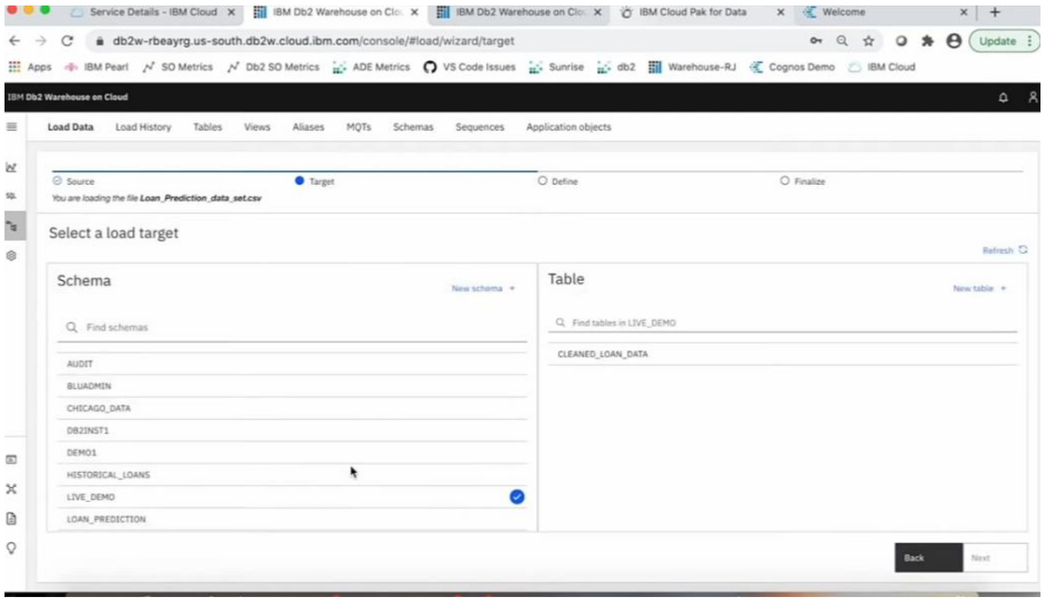


2. Data Transformation:

Clean and preprocess the extracted data. This includes tasks like data cleansing, data enrichment, and data normalization.

3.Data Loading:

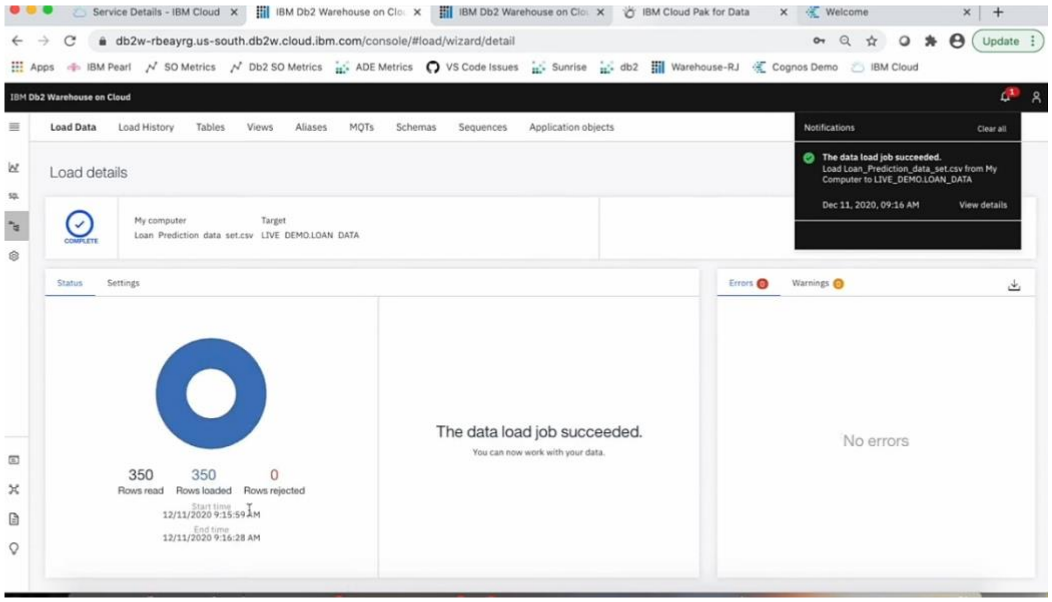
Load the transformed data into your IBM Db2 Warehouse. Depending on your needs, you might use bulk loading, real-time streaming, or other loading methods.



You are loading the file Loan_Prediction_data_set.csv

Schema
AUDIT
BLUADMIN
CHICAGO_DATA
DB2INST1
DEMO1
HISTORICAL_LOANS
LIVE_DEMO ✓
LOAN_PREDICTION

Table
CLEANED_LOAN_DATA



The data load job succeeded.
Load Loan_Prediction_data_set.csv from My Computer to LIVE_DEMO.LOAN_DATA
Dec 11, 2020, 09:16 AM

My computer: Loan_Prediction_data_set.csv
Target: LIVE_DEMO.LOAN_DATA

350 Rows read | 350 Rows loaded | 0 Rows rejected

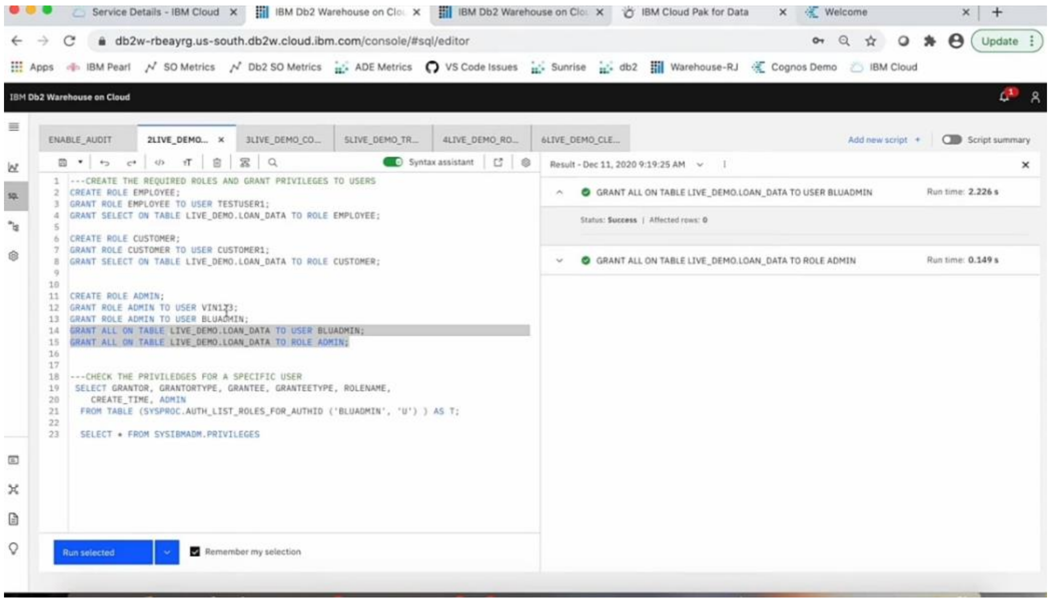
Start time 12/11/2020 9:15:59 AM
End time 12/11/2020 9:16:28 AM

The data load job succeeded.
You can now work with your data.

No errors

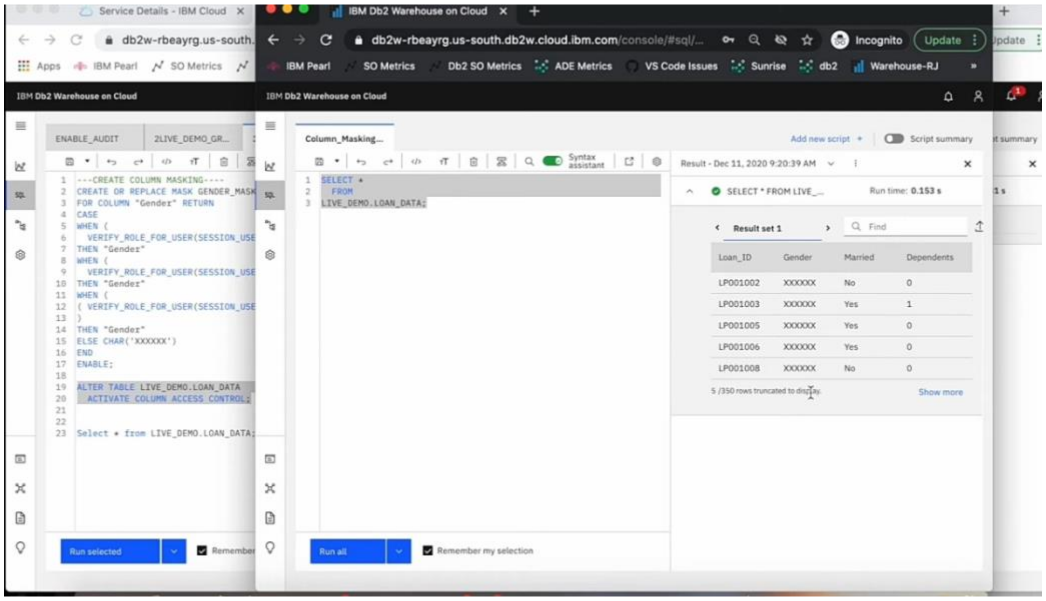
4. **Data Exploration:**

Provide data architects with tools and access to the data in Db2 Warehouse. SQL is a powerful tool for querying and analyzing data.



```
---CREATE THE REQUIRED ROLES AND GRANT PRIVILEGES TO USERS
CREATE ROLE EMPLOYEE;
GRANT ROLE EMPLOYEE TO USER TESTUSER1;
GRANT SELECT ON TABLE LIVE_DEMO.LOAN_DATA TO ROLE EMPLOYEE;

CREATE ROLE CUSTOMER;
GRANT ROLE CUSTOMER TO USER CUSTOMER1;
GRANT SELECT ON TABLE LIVE_DEMO.LOAN_DATA TO ROLE CUSTOMER;


CREATE ROLE ADMIN;
GRANT ROLE ADMIN TO USER VIN123;
GRANT ROLE ADMIN TO USER BLUADMIN;
GRANT ALL ON TABLE LIVE_DEMO.LOAN_DATA TO USER BLUADMIN;
GRANT ALL ON TABLE LIVE_DEMO.LOAN_DATA TO ROLE ADMIN;


---CHECK THE PRIVILEDGES FOR A SPECIFIC USER
 SELECT GRANTOR, GRANTORTYPE, GRANTEE, GRANTEETYPE, ROLENAME,
    CREATE_TIME, ADMIN
  FROM TABLE (SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID ('BLUADMIN', 'U') ) AS T;

  SELECT * FROM SYSIBMADM.PRIVILEGES
```

```
Result - Dec 11, 2020 9:19:25 AM
GRANT ALL ON TABLE LIVE_DEMO.LOAN_DATA TO USER BLUADMIN    Run time: 2.226 s
Status: Success | Affected rows: 0
GRANT ALL ON TABLE LIVE_DEMO.LOAN_DATA TO ROLE ADMIN       Run time: 0.149 s
```



```
---CREATE COLUMN MASKING----
CREATE OR REPLACE MASK GENDER_MASK
FOR COLUMN "Gender" RETURN
CASE
WHEN (
  VERIFY_ROLE_FOR_USER(SESSION_USE
THEN "Gender"
WHEN (
  VERIFY_ROLE_FOR_USER(SESSION_USE
THEN "Gender"
WHEN (
( VERIFY_ROLE_FOR_USER(SESSION_USE
)
THEN "Gender"
ELSE CHAR('XXXXXX')
END
ENABLE;

ALTER TABLE LIVE_DEMO.LOAN_DATA
  ACTIVATE COLUMN ACCESS CONTROL;


Select * from LIVE_DEMO.LOAN_DATA;
```

```
SELECT *
  FROM
LIVE_DEMO.LOAN_DATA;
```

Result - Dec 11, 2020 9:20:39 AM

SELECT * FROM LIVE_... Run time: 0.153 s

Result set 1

Loan_ID	Gender	Married	Dependents
LP001002	XXXXXX	No	0
LP001003	XXXXXX	Yes	1
LP001005	XXXXXX	Yes	0
LP001006	XXXXXX	Yes	0
LP001008	XXXXXX	No	0

5 /350 rows truncated to display. Show more

5. SQL Query Optimization:

Ensure that your SQL queries are optimized for performance. This might involve creating indexes, tuning queries, and monitoring query performance.

6. Data Analysis Techniques:

Support data architects in applying various data analysis techniques such as data mining, machine learning, and statistical analysis to derive insights from the data.

7. Data Governance:

Implement data governance practices to ensure data quality, security, and compliance with relevant regulations.

8. Documentation:

Document the ETL processes, data models, and analysis techniques for future reference and knowledge sharing.

How ETL works

The easiest way to understand how ETL works is to understand what happens in each step of the process.

Extract

During data extraction, raw data is copied or exported from source locations to a staging area. Data management teams can extract data from a variety of data sources, which can be structured or unstructured. Those sources include but are not limited to:

- SQL or NoSQL servers
- CRM and ERP systems
- Flat files
- Email
- Web pages

Transform

In the staging area, the raw data undergoes data processing. Here, the data is transformed and consolidated for its intended analytical use case. This phase can involve the following tasks:

- Filtering, cleansing, de-duplicating, validating, and authenticating the data.

- Performing calculations, translations, or summarizations based on the raw data. This can include changing row and column headers for consistency, converting currencies or other units of measurement, editing text strings, and more.
- Conducting audits to ensure data quality and compliance
- Removing, encrypting, or protecting data governed by industry or governmental regulators
- Formatting the data into tables or joined tables to match the schema of the target data warehouse.

Load

In this last step, the transformed data is moved from the staging area into a target data warehouse. Typically, this involves an initial loading of all data, followed by periodic loading of incremental data changes and, less often, full refreshes to erase and replace data in the warehouse.
For most organizations that use ETL, the process is automated, well-defined, continuous and batch-driven. Typically, ETL takes place during off-hours when traffic on the source systems and the data warehouse is at its lowest.

ETL and other data integration methods

ETL and ELT are just two data integration methods, and there are other approaches that are also used to facilitate data integration workflows. Some of these include:

- **Change Data Capture (CDC)** identifies and captures only the source data that has changed and moves that data to the target system. CDC can be used to reduce the resources required during the ETL "extract" step; it can also be used independently to move data that has been transformed into a data lake or other repository in real time.

- **Data replication** copies changes in data sources in real time or in batches to a central database. Data replication is often listed as a data integration method. In fact, it is most often used to create backups for disaster recovery.

- **Data virtualization** uses a software abstraction layer to create a unified, integrated, fully usable *view* of data—without physically copying, transforming or loading the source data to a target system. Data virtualization functionality enables an organization to create virtual data warehouses, data lakes and data marts from the same source data for data storage without the expense and complexity of building and managing separate platforms for each. While data virtualization can be used alongside ETL, it is increasingly seen as an alternative to ETL and to other physical data integration methods.

- **Stream Data Integration (SDI)** is just what it sounds like—it continuously consumes data streams in real time, transforms them, and loads them to a target system for analysis. The key word here is *continuously*. Instead of integrating snapshots of data extracted from sources at a given time, SDI integrates data constantly as it becomes available. SDI enables a data store for powering analytics, machine learning and real-time applications for improving customer experience, fraud detection and more.

Create data model from SQL

In order to demonstrate how you can use the strategy defined in this document to build data model we will be using below mentioned queries.

```
/*
 Query to find all events and occurrences in one year
*/
SELECT eve.eventname,count(*) as total_occurences
FROM Event eve
inner join
Date_cal dat
on
eve.dateid = dat.dateid
where dat.caldate between '2022-01-01' and '2022-12-31'
group by eve.eventname
;/*
Query to find all events on 30th Oct 2022 in Bangalore city
*/
SELECT eve.eventname,
eve.starttime,
ven.venuename,
ven.venuecity,
ven.venuecitypop,
ven.venuestate,
dat.day,
dat.holiday,
dat.caldate
FROM Event eve
```

```
inner join
Venue ven
on
eve.venueid = ven.venueid
inner join
Date_cal dat
on
eve.dateid = dat.dateid
where ven.venuecity = 'Bangalore'
and dat.caldate = '2022-10-30'
;/*
Query to find the total number of tickets sold, price paid, and commission paid for each event and then
determine profit for each event.
*/
Select  sales_event.eventname , sales_event.tot_Qtysold, sales_event.tot_Pricepaid -
sales_event.tot_commission as tot_profit
from(
select
sum(sal.Qtysold) as tot_Qtysold,
sum(sal.Pricepaid) as tot_Pricepaid,
sum(sal.commission) as tot_commission,
eve.eventname
FROM
Sales sal
inner join
Event eve
on
sal.eventid = eve.eventid
group by eve.eventname
)sales_event
order by 2 desc;/*
```

Query to find least popular events in history

*/SELECT cat.catname,eve.eventname,sum(sal.Qtysold) as tot_tickets_sold

FROM Category cat

inner join

Event eve on

cat.catid = eve.catid

inner join

Sales sal on sal.eventid = eve.eventid

GROUP BY cat.catname,eve.eventname

ORDER BY 3 asc;

SQL into Data Model – The Solution

The solution uses python packages with custom code to parse SQL and then generate **DBML output**. We will use the DBML output generated into the **web app to visualise it** and save as data model.

Step 1:

Install the **sqlparse package**

pip install sqlparse

Step 2: Install the sql-metadata package

pip install sql-metadata

Step 3: Run the below python command

import sqlparse

from sql_metadata import Parser

strsql = """

/*

 Query to find all events and occurrences in one year

*/

```
SELECT eve.eventname,count(*) as total_occurences

FROM Event eve

inner join

Date_cal dat

on

eve.dateid = dat.dateid

where dat.caldate between '2022-01-01' and '2022-12-31'

group by eve.eventname;


/*Query to find all events on 30th Oct 2022 in Bangalore city*/

SELECT
eve.eventname,eve.starttime,ven.venuename,ven.venuecity,ven.venuecitypop,ven.venuestate,dat.day,dat.
holiday,dat.caldate

FROM Event eve

inner join  Venue ven on

eve.venueid = ven.venueid inner join Date_cal dat

on

eve.dateid = dat.dateid

where ven.venuecity = 'Bangalore'

and dat.caldate = '2022-10-30';

/*

Query to find the total number of tickets sold, price paid, and commission paid for each event and then
determine profit for each event.

*/

Select  sales_event.eventname , sales_event.tot_Qtysold, sales_event.tot_Pricepaid -
sales_event.tot_commission as tot_profit

from

(

select

sum(sal.Qtysold) as tot_Qtysold,

sum(sal.Pricepaid) as tot_Pricepaid,
sum(sal.commission) as tot_commission,
eve.eventname
FROM
Sales sal
inner join
Event eve
on
sal.eventid = eve.eventid
group by eve.eventname
)sales_event
order by 2 desc;
/*
Query to find least popular events in history
*/
SELECT
cat.catname,
eve.eventname,
sum(sal.Qtysold) as tot_tickets_sold
FROM Category cat
inner join
Event eve on
cat.catid = eve.catid
inner join
Sales sal
on
sal.eventid = eve.eventid
GROUP BY cat.catname,eve.eventname
ORDER BY 3 asc;
"""
statements = sqlparse.split(strsql)
```

```
all_columns = list()
all_tables = list()
all_joins = ""

for stmt in statements:
    parser = Parser(stmt)
    for key,value in parser.columns_dict.items():
        if value is not None:
            all_columns.append(value)
    for j in parser.tables:
        all_tables.append(j.split())
    i=0
    join_cond=""
    if 'join' in parser.columns_dict:
        for j in parser.columns_dict['join']:
            if i%2 == 0:
                join_cond = join_cond + "" + "Ref: "+j
            else:
                join_cond = join_cond + " - " + j +"|"
            i = i + 1
        all_joins = all_joins + join_cond
all_joins_list = all_joins.split("|")
all_joins_list = list(set(all_joins_list))
all_joins = '|'.join(all_joins_list)
all_columns = sum(all_columns, [])
all_columns = list(set(all_columns))
all_tables = sum(all_tables, [])
all_tables = list(set(all_tables))
i=0
table_def=""
```

```
for j in all_tables:
    table_struct=""
    table_struct = "Table "+j+" {\nrowid bigint"
    for i in all_columns:
        if i.startswith(j+'.') :
            table_struct = table_struct +"\n"+ i.split('.')[-1] + " string "
   table_def = table_def + table_struct + "\n}\n|"
for i in table_def[:-1].split("|"):
    print (i)
for i in all_joins[1:].split("|"):
   print (i)
```

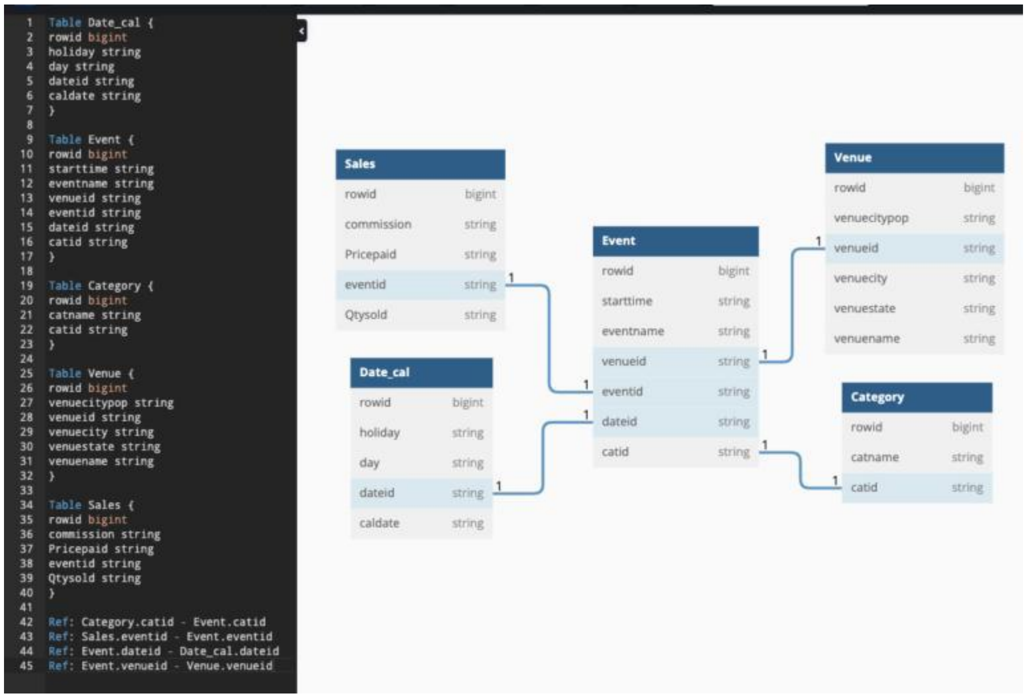
-Change the SQL as per your requirement in the Python code.

Step 4: Copy the DBML output generated

```
Table Date_cal {
rowid bigint
holiday string
day string
dateid string
caldate string
}
Table Event {
rowid bigint
starttime string
eventname string
venueid string
eventid string
dateid string
catid string
}
```

```
Table Category {
rowid bigint
catname string
catid string
}

Table Venue {
rowid bigint
venuecitypop string
venueid string
venuecity string
venuestate string
venuename string
}
Table Sales {
rowid bigint
commission string
Pricepaid string
eventid string
Qtysold string
}
Ref: Category.catid - Event.catid
Ref: Sales.eventid - Event.eventid
Ref: Event.dateid - Date_cal.dateid
Ref: Event.venueid - Venue.venueid
```

Step 5: Paste the DBML output



```
Table Date_cal {
rowid bigint
holiday string
day string
dateid string
caldate string
}

Table Event {
rowid bigint
starttime string
eventname string
venueid string
eventid string
dateid string
catid string
}

Table Category {
rowid bigint
catname string
catid string
}

Table Venue {
rowid bigint
venuecitypop string
venueid string
venuecity string
venuestate string
venuename string
}

Table Sales {
rowid bigint
commission string
Pricepaid string
eventid string
Qtysold string
}

Ref: Category.catid - Event.catid
Ref: Sales.eventid - Event.eventid
Ref: Event.dateid - Date_cal.dateid
Ref: Event.venueid - Venue.venueid
```