IBM Data Engineering Capstone Project

Project Overview

As part of the Capstone project, you will assume the role of an Associate Data Engineer who has recently joined an e-commerce organization. You will be presented with a business challenge that requires building a data platform for retailer data analytics.

In this Capstone project, you will:
Design a data platform that uses MySQL as an OLTP database and MongoDB as a NoSQL database.
Design and implement a data warehouse and generate reports from the data.
Design a reporting dashboard that reflects the key metrics of the business.
Extract data from OLTP and NoSQL databases, transform it and load it into the data warehouse, and then create an ETL pipeline.
And finally, create a Spark connection to the data warehouse, and then deploy a machine learning model.

In Module 1, you will design the OLTP database for an E-Commerce website, populate the OLTP Database with the data provided and automate the export of the daily incremental data into the data warehouse.

In Module 2, you will set up a NoSQL database to store the catalogue data for an E-Commerce website, load the E-Commerce catalogue data into the NoSQL database, and query the E-Commerce catalogue data in the NoSQL database.

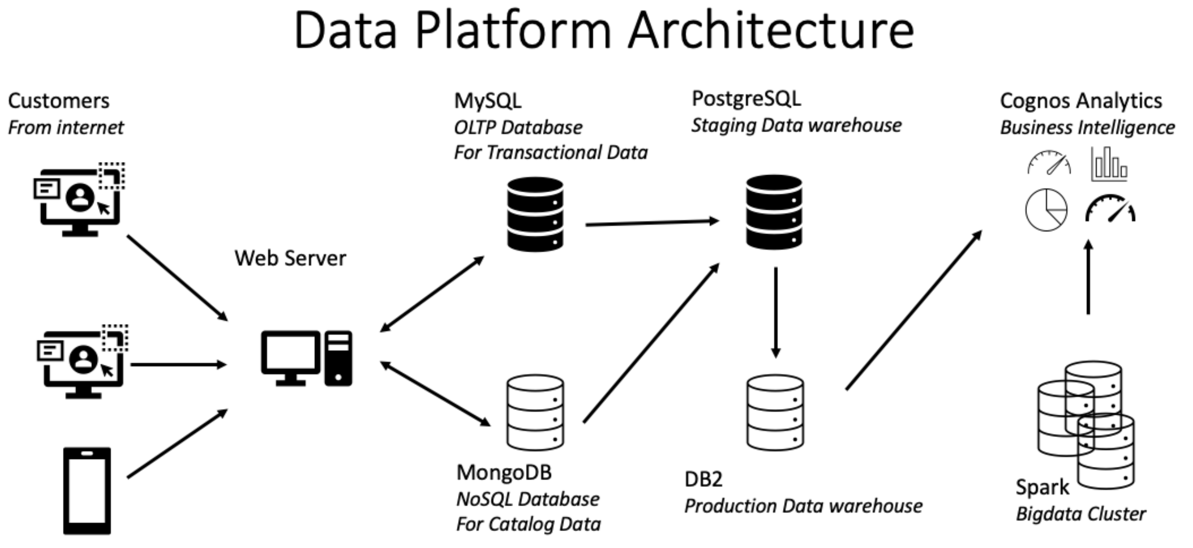
In Module 3, you will design the schema for a data warehouse based on the schema of the OLTP and NoSQL databases. You’ll then create the schema and load the data into fact and dimension tables, automate the daily incremental data insertion into the data warehouse, and create Cubes and Rollups to make the reporting easier.

In Module 4, you will create a Cognos data source that points to a data warehouse table, create a bar chart of Quarterly sales of cell phones, create a pie chart of sales of electronic goods by category, and create a line chart of total sales per month for the year 2020.

In Module 5, you will extract data from OLTP, NoSQL, and MongoDB databases into CSV format. You will then transform the OLTP data to suit the data warehouse schema and then load the transformed data into the data warehouse. Finally, you will verify that the data is loaded properly.

In the sixth and final module, you will use your skills in Big Data Analytics to create a Spark connection to the data warehouse, and then deploy a machine learning model on SparkML for making sales projections.

Data Platform Architecture



This is the data platform architecture of an ecommerce company named SoftCart. SoftCart uses a hybrid architecture, with some of its databases on premises and some on cloud.

Tools and Technologies:

- OLTP database - MySQL
- NoSQL database - MongoDB
- Production Data warehouse – DB2 on Cloud
- Staging - Data warehouse – PostgreSQL
- Big data platform - Hadoop
- Big data analytics platform – Spark
- Business Intelligence Dashboard - IBM Cognos Analytics
- Data Pipelines - Apache Airflow

Process:

SoftCart's online presence is primarily through its website, which customers access using a variety of devices like laptops, mobiles and tablets. All the catalogue data of the products is stored in the MongoDB NoSQL server. All the transactional data like inventory and sales are stored in the MySQL database server. SoftCart's webserver is driven entirely by these two databases. Data is periodically extracted from these two databases and put into the staging data warehouse running on PostgreSQL. Production data warehouse is on the cloud instance of IBM DB2 server. BI teams connect to the IBM DB2 for operational dashboard creation. IBM Cognos Analytics is used to create dashboards. SoftCart uses Hadoop cluster as its big data platform where all the data collected for analytics purposes. Spark is used to analyse the data on the Hadoop cluster. To move data between OLTP, NoSQL and the data warehouse, ETL pipelines are used and these run on Apache Airflow.

Module 1: with link to OLTP database exercise

OLTP database requirements and design

OLTP database:

OLTP database is generally used to handle everyday business transactions of an organization like a bank or a super market chain. OLTP databases can be write heavy or may have a balanced read/write load.

OLTP database requirements:

An OLTP database is expected to handle a huge number of transactions per second. Each transaction usually involves accessing (read/write) a small portion of the database, in other words the payload per transaction is small.
The time taken to execute a transaction usually called latency needs to be very less.

OLTP database design:

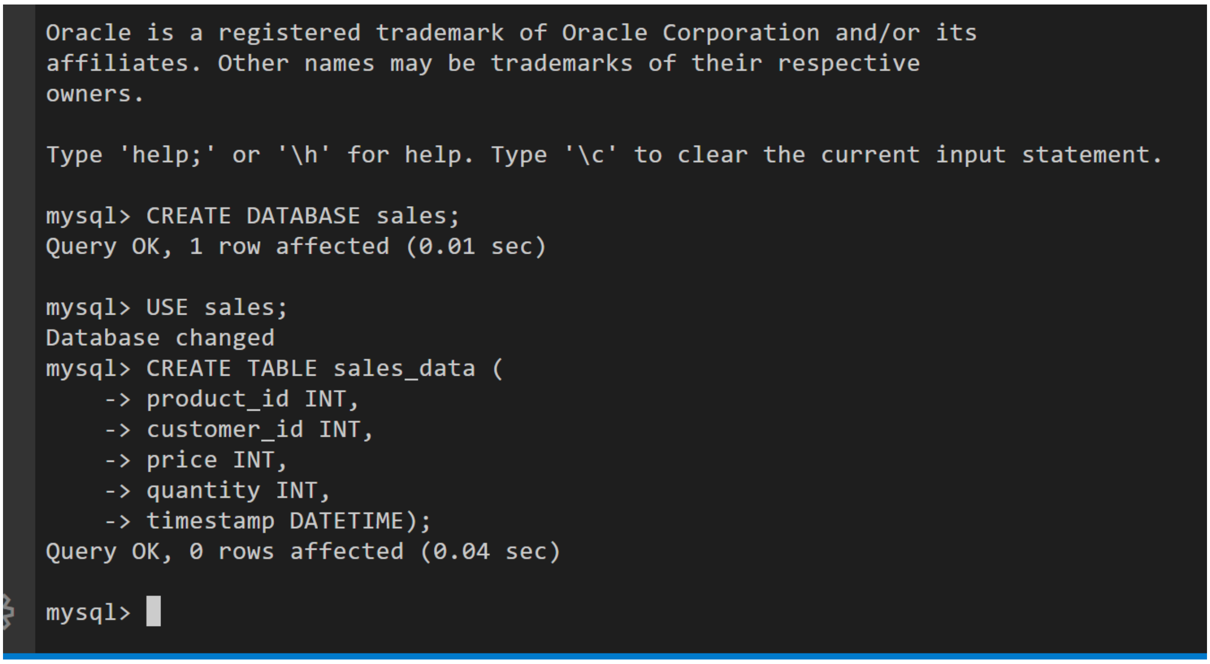
The schema of an OLTP database is highly normalized so as to achieve a very low latency. To further improve the latency an OLTP database stores only the recent data like the last few week's data. They are usually run on storage that is very fast like SSD.

Scenario

You are a data engineer at an e-commerce company. Your company needs you to design a data platform that uses MySQL as an OLTP database. You will be using MySQL to store the OLTP data.

Design the OLTP Database

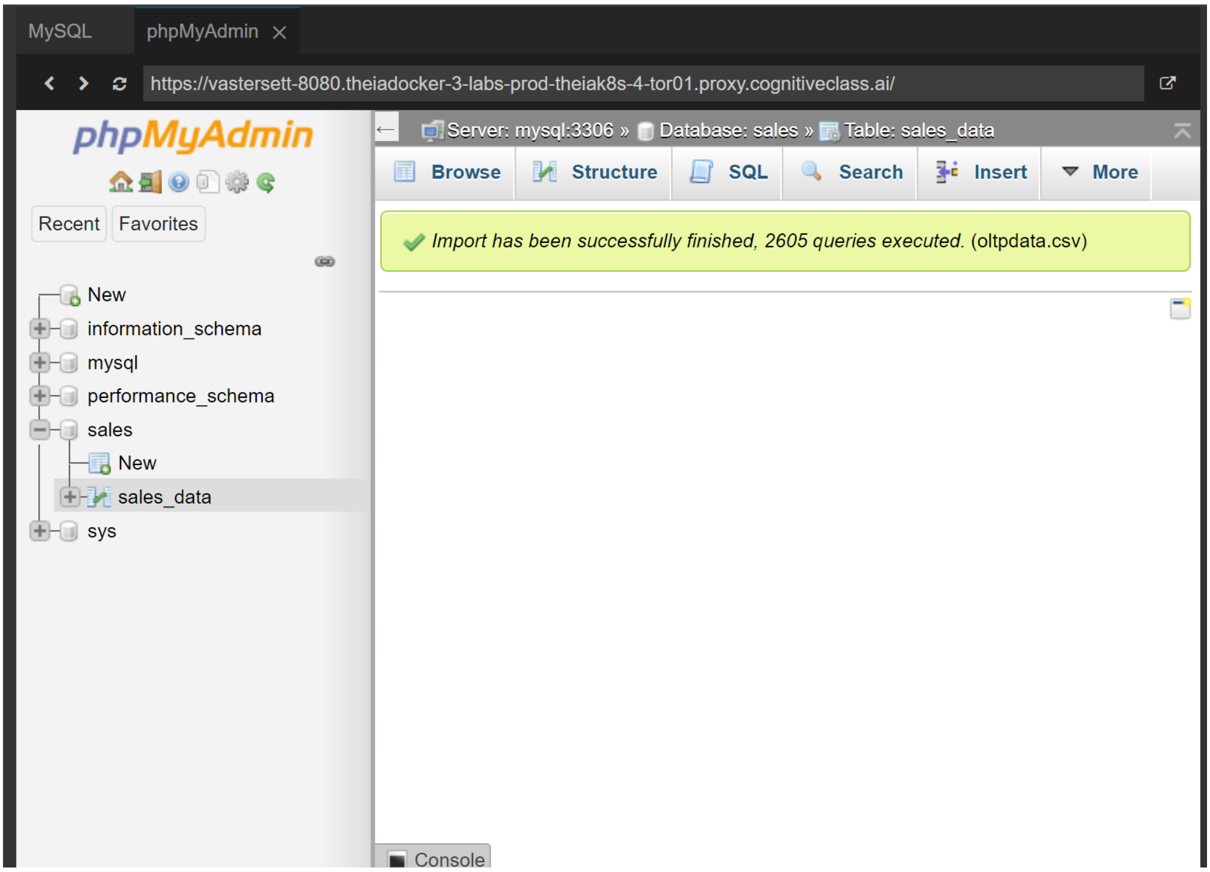
Creating sales database and sales_data table in sales database.



```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE sales;
Query OK, 1 row affected (0.01 sec)

mysql> USE sales;
Database changed
mysql> CREATE TABLE sales_data (
    -> product_id INT,
    -> customer_id INT,
    -> price INT,
    -> quantity INT,
    -> timestamp DATETIME);
Query OK, 0 rows affected (0.04 sec)

mysql> 
```

Load the given data(oltpdata.csv) into sales_data table

Importing data from oltpdata.csv file into sales_data table using phpMyAdmin GUI.

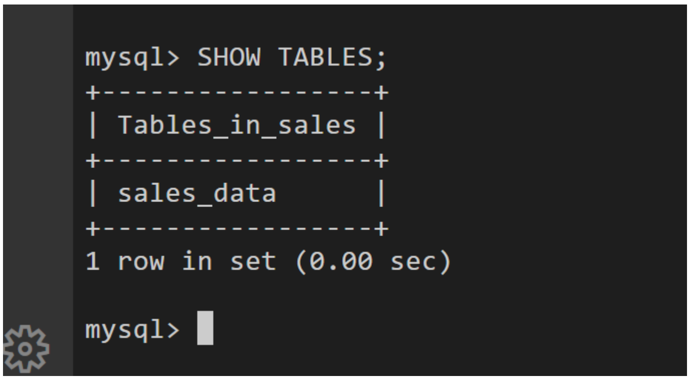


https://vastersett-8080.theiadocker-3-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

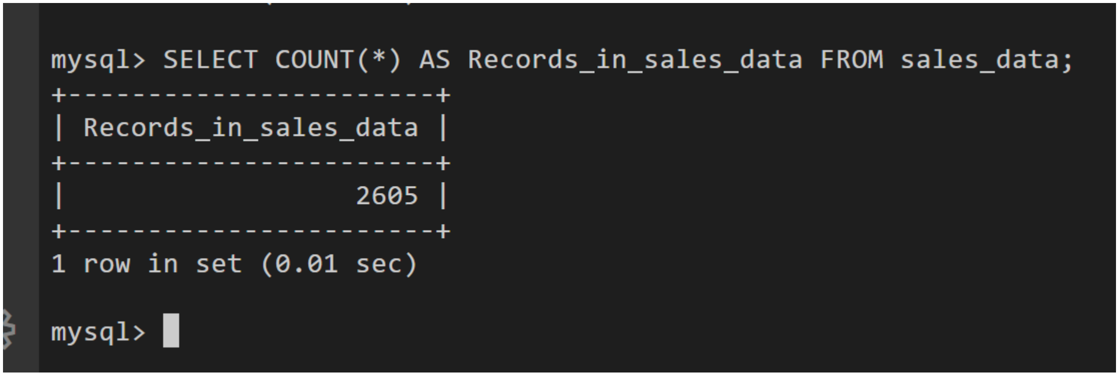
Server: mysql:3306 » Database: sales » Table: sales_data

✓ *Import has been successfully finished, 2605 queries executed.* (oltpdata.csv)

List the tables in the sales database and query the count of records in the sales_data table to make sure the data are imported correctly and successfully.



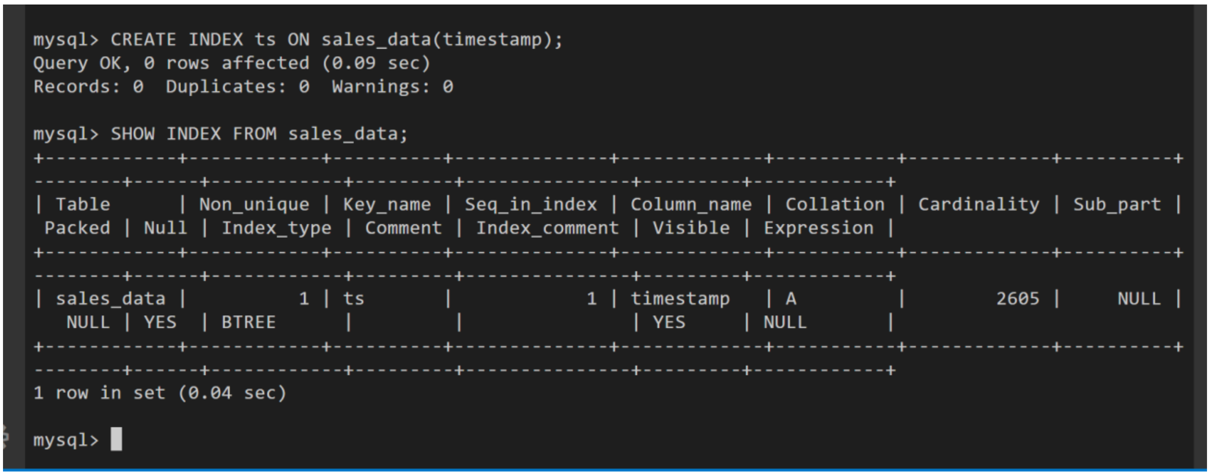
```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_sales |
+-----------------+
| sales_data      |
+-----------------+
1 row in set (0.00 sec)

mysql> 
```



```
mysql> SELECT COUNT(*) AS Records_in_sales_data FROM sales_data;
+-----------------------+
| Records_in_sales_data |
+-----------------------+
|                  2605 |
+-----------------------+
1 row in set (0.01 sec)

mysql> 
```

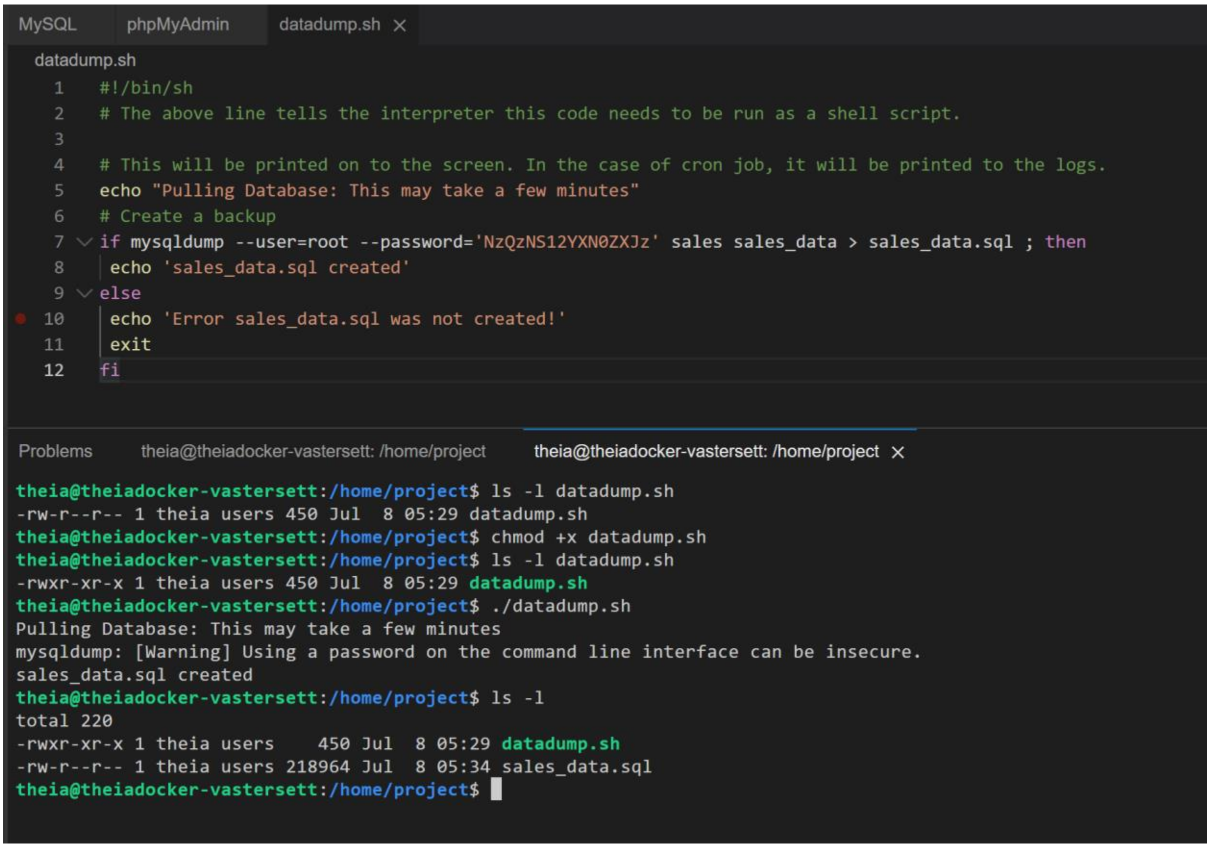
Set up Admin tasks

Creating index on the timestamp field to speed up queries.



```
mysql> CREATE INDEX ts ON sales_data(timestamp);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM sales_data;
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+
--------+------+------------+---------+---------------+---------+------------+
| Table      | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part |
 Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+
--------+------+------------+---------+---------------+---------+------------+
| sales_data |          1 | ts       |            1 | timestamp   | A         |        2605 |     NULL |
   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+------------+------------+----------+--------------+-------------+-----------+-------------+----------+
--------+------+------------+---------+---------------+---------+------------+
1 row in set (0.04 sec)

mysql> █
```

Writing a bash script(datadump.sh) that exports all the rows in the sales_data table to a file named sales_data.sql to automate the export of the daily incremental data into the data warehouse.


MySQL
phpMyAdmin
datadump.sh ×
datadump.sh
Problems
theia@theiadocker-vastersett: /home/project
theia@theiadocker-vastersett: /home/project ×

```
#!/bin/sh
# The above line tells the interpreter this code needs to be run as a shell script.

# This will be printed on to the screen. In the case of cron job, it will be printed to the logs.
echo "Pulling Database: This may take a few minutes"
# Create a backup
if mysqldump --user=root --password='NzQzNS12YXN0ZXJz' sales sales_data > sales_data.sql ; then
 echo 'sales_data.sql created'
else
 echo 'Error sales_data.sql was not created!'
 exit
fi
```

```
theia@theiadocker-vastersett:/home/project$ ls -l datadump.sh
-rw-r--r-- 1 theia users 450 Jul  8 05:29 datadump.sh
theia@theiadocker-vastersett:/home/project$ chmod +x datadump.sh
theia@theiadocker-vastersett:/home/project$ ls -l datadump.sh
-rwxr-xr-x 1 theia users 450 Jul  8 05:29 datadump.sh
theia@theiadocker-vastersett:/home/project$ ./datadump.sh
Pulling Database: This may take a few minutes
mysqldump: [Warning] Using a password on the command line interface can be insecure.
sales_data.sql created
theia@theiadocker-vastersett:/home/project$ ls -l
total 220
-rwxr-xr-x 1 theia users    450 Jul  8 05:29 datadump.sh
-rw-r--r-- 1 theia users 218964 Jul  8 05:34 sales_data.sql
theia@theiadocker-vastersett:/home/project$ █
```

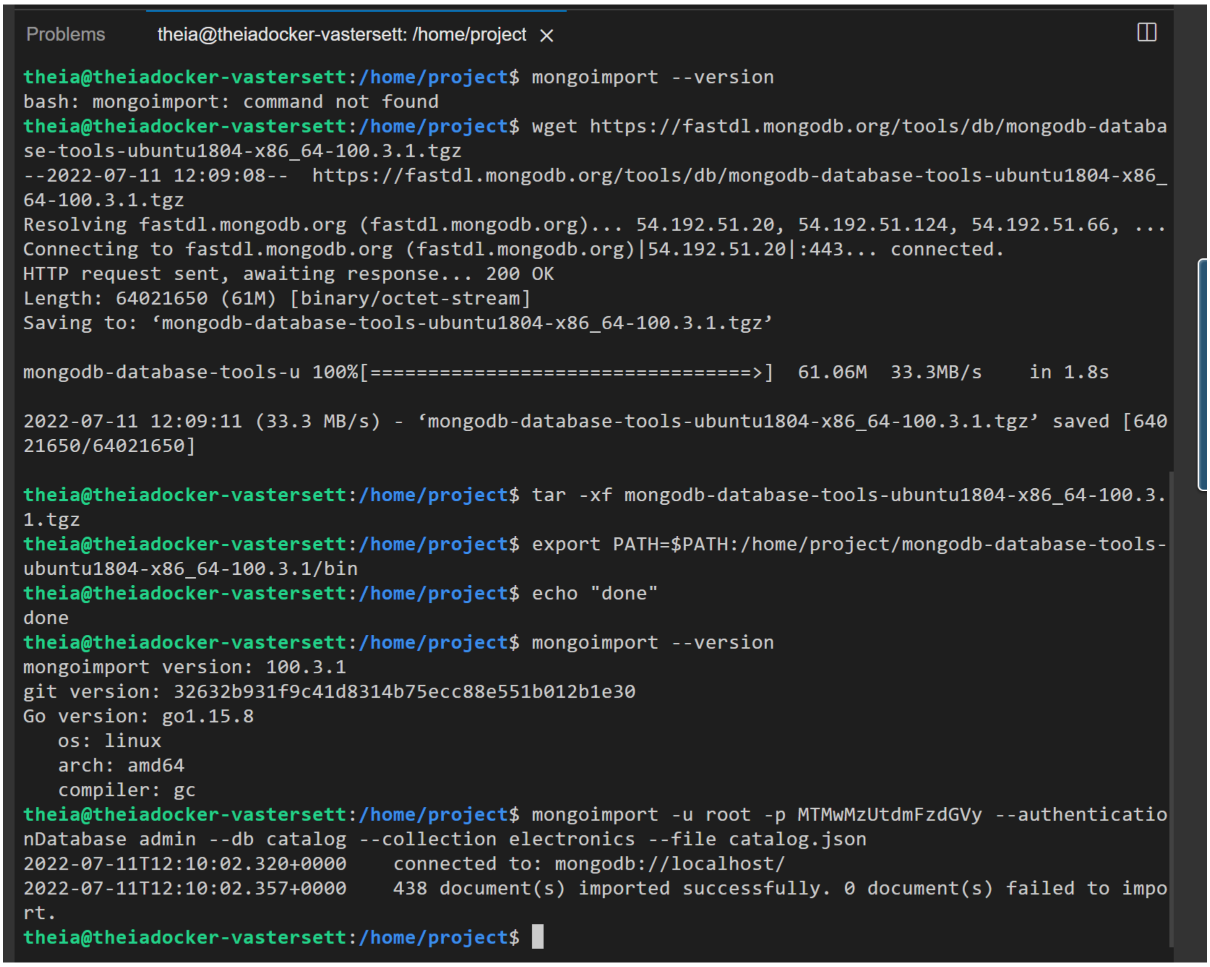
Module 2:

Scenario

You are a data engineer at an e-commerce company. Your company needs you to design a data platform that uses MongoDB as a NoSQL database. You will be using MongoDB to store the e-commerce catalogue data.

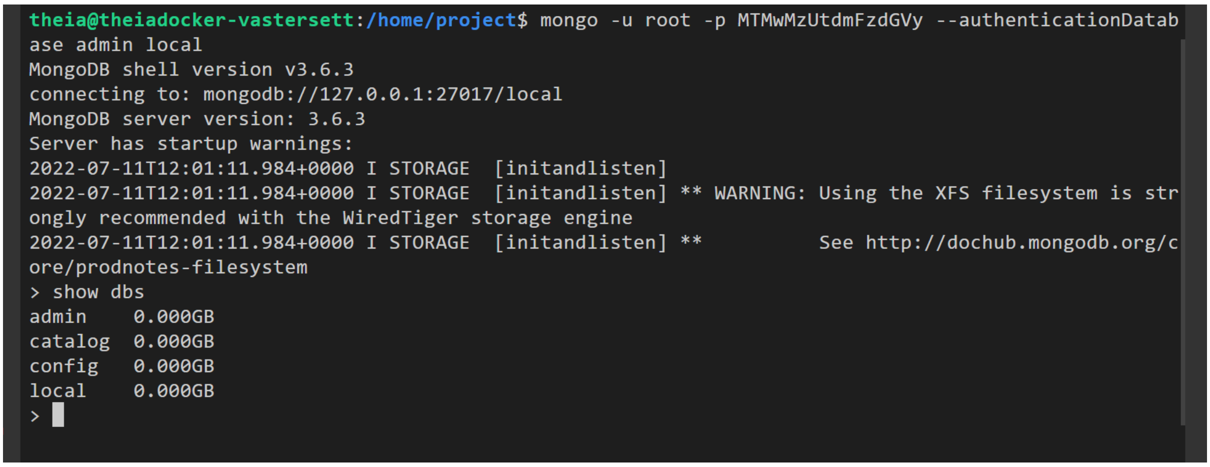
Importing data into MongoDB database

Installing mongoimport and mongoexport. Importing given data(catalog.json) into a database named ‘catalog’ and a collection named ‘electronics’ on mongodb server.

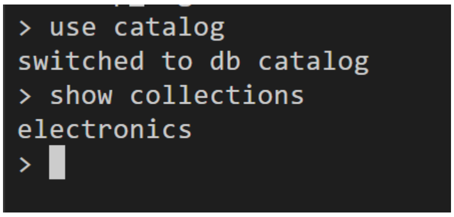


```
theia@theiadocker-vastersett:/home/project$ mongoimport --version
bash: mongoimport: command not found
theia@theiadocker-vastersett:/home/project$ wget https://fastdl.mongodb.org/tools/db/mongodb-databa
se-tools-ubuntu1804-x86_64-100.3.1.tgz
--2022-07-11 12:09:08--  https://fastdl.mongodb.org/tools/db/mongodb-database-tools-ubuntu1804-x86_
64-100.3.1.tgz
Resolving fastdl.mongodb.org (fastdl.mongodb.org)... 54.192.51.20, 54.192.51.124, 54.192.51.66, ...
Connecting to fastdl.mongodb.org (fastdl.mongodb.org)|54.192.51.20|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 64021650 (61M) [binary/octet-stream]
Saving to: ‘mongodb-database-tools-ubuntu1804-x86_64-100.3.1.tgz’

mongodb-database-tools-u 100%[===============================>]  61.06M  33.3MB/s    in 1.8s

2022-07-11 12:09:11 (33.3 MB/s) - ‘mongodb-database-tools-ubuntu1804-x86_64-100.3.1.tgz’ saved [640
21650/64021650]

theia@theiadocker-vastersett:/home/project$ tar -xf mongodb-database-tools-ubuntu1804-x86_64-100.3.
1.tgz
theia@theiadocker-vastersett:/home/project$ export PATH=$PATH:/home/project/mongodb-database-tools-
ubuntu1804-x86_64-100.3.1/bin
theia@theiadocker-vastersett:/home/project$ echo "done"
done
theia@theiadocker-vastersett:/home/project$ mongoimport --version
mongoimport version: 100.3.1
git version: 32632b931f9c41d8314b75ecc88e551b012b1e30
Go version: go1.15.8
   os: linux
   arch: amd64
   compiler: gc
theia@theiadocker-vastersett:/home/project$ mongoimport -u root -p MTMwMzUtdmFzdGVy --authenticatio
nDatabase admin --db catalog --collection electronics --file catalog.json
2022-07-11T12:10:02.320+0000    connected to: mongodb://localhost/
2022-07-11T12:10:02.357+0000    438 document(s) imported successfully. 0 document(s) failed to impo
rt.
theia@theiadocker-vastersett:/home/project$ 
```

List out all databases and collections on mongodb server to check if catalog database and electronics collection are created successfully or not.

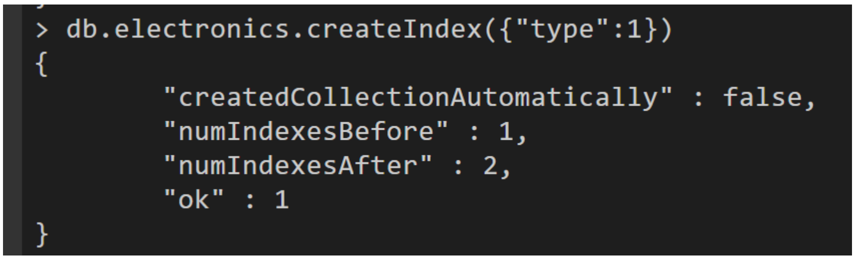


```
theia@theiadocker-vastersett:/home/project$ mongo -u root -p MTMwMzUtdmFzdGVy --authenticationDatab
ase admin local
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/local
MongoDB server version: 3.6.3
Server has startup warnings:
2022-07-11T12:01:11.984+0000 I STORAGE  [initandlisten]
2022-07-11T12:01:11.984+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is str
ongly recommended with the WiredTiger storage engine
2022-07-11T12:01:11.984+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/c
ore/prodnotes-filesystem
> show dbs
admin    0.000GB
catalog  0.000GB
config   0.000GB
local    0.000GB
>
```



```
> use catalog
switched to db catalog
> show collections
electronics
>
```

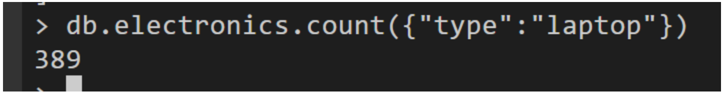
Creating index on the field ‘type’ for faster queries.



```
> db.electronics.createIndex({"type":1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

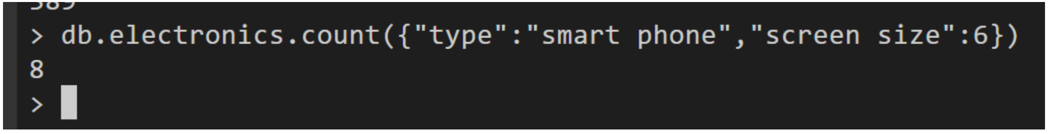
Trying out a few queries to check the imported data:

Finding the count of laptops.



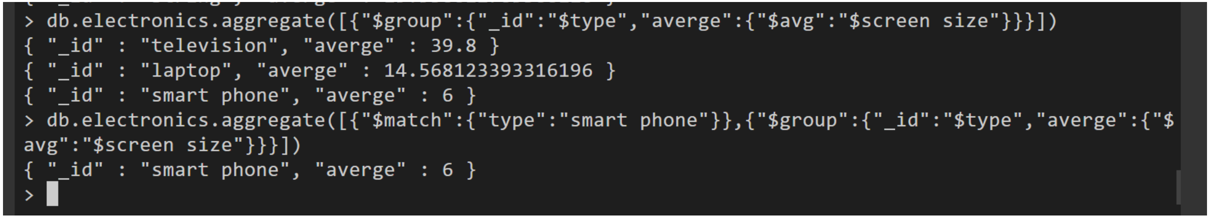
```
> db.electronics.count({"type":"laptop"})
389
```

Finding the number of smart phones with screen size of 6 inches.



```
> db.electronics.count({"type":"smart phone","screen size":6})
8
>
```

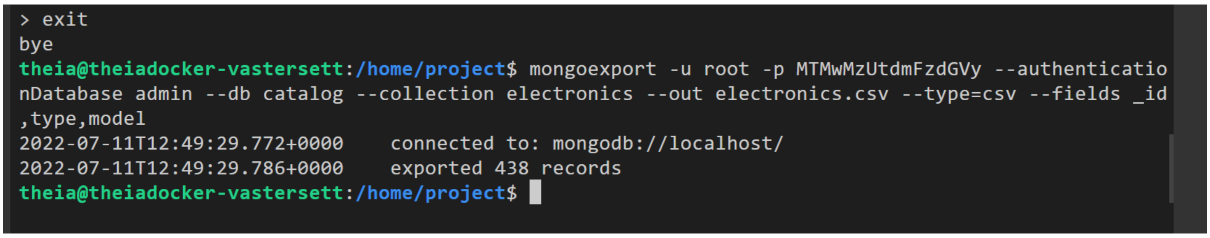
Finding the average screen size of smart phones.



```
> db.electronics.aggregate([{"$group":{"_id":"$type","averge":{"$avg":"$screen size"}}}])
{ "_id" : "television", "averge" : 39.8 }
{ "_id" : "laptop", "averge" : 14.568123393316196 }
{ "_id" : "smart phone", "averge" : 6 }
> db.electronics.aggregate([{"$match":{"type":"smart phone"}},{"$group":{"_id":"$type","averge":{"$
avg":"$screen size"}}}])
{ "_id" : "smart phone", "averge" : 6 }
>
```

Export data from MongoDB database as csv file

Exporting the fields _id, “type”, “model”, from the ‘electronics’ collection into a file named electronics.csv



```
> exit
bye
theia@theiadocker-vastersett:/home/project$ mongoexport -u root -p MTMwMzUtdmFzdGVy --authenticatio
nDatabase admin --db catalog --collection electronics --out electronics.csv --type=csv --fields _id
,type,model
2022-07-11T12:49:29.772+0000    connected to: mongodb://localhost/
2022-07-11T12:49:29.786+0000    exported 438 records
theia@theiadocker-vastersett:/home/project$ 
```

Module 3.1:

Scenario

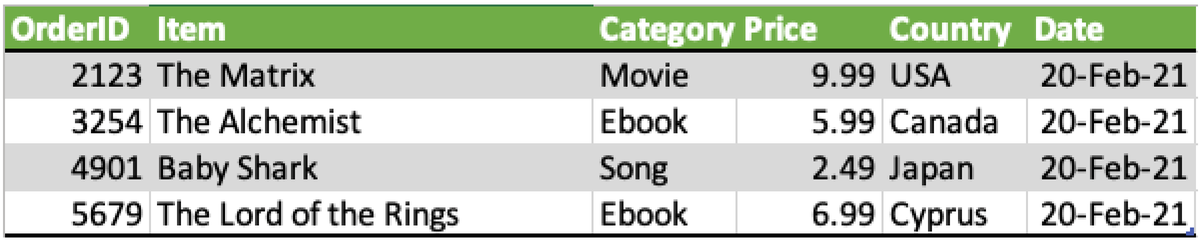
You are a data engineer hired by an ecommerce company named SoftCart.com. The company retails download only items like E-Books, Movies, Songs etc. The company has international presence and customers from all over the world. The company would like to create a data warehouse so that it can create reports like:

- total sales per year per country
- total sales per month per category
- total sales per quarter per country
- total sales per category per country

You will use your data warehousing skills to design and implement a data warehouse for the company.

Designing a Data Warehouse

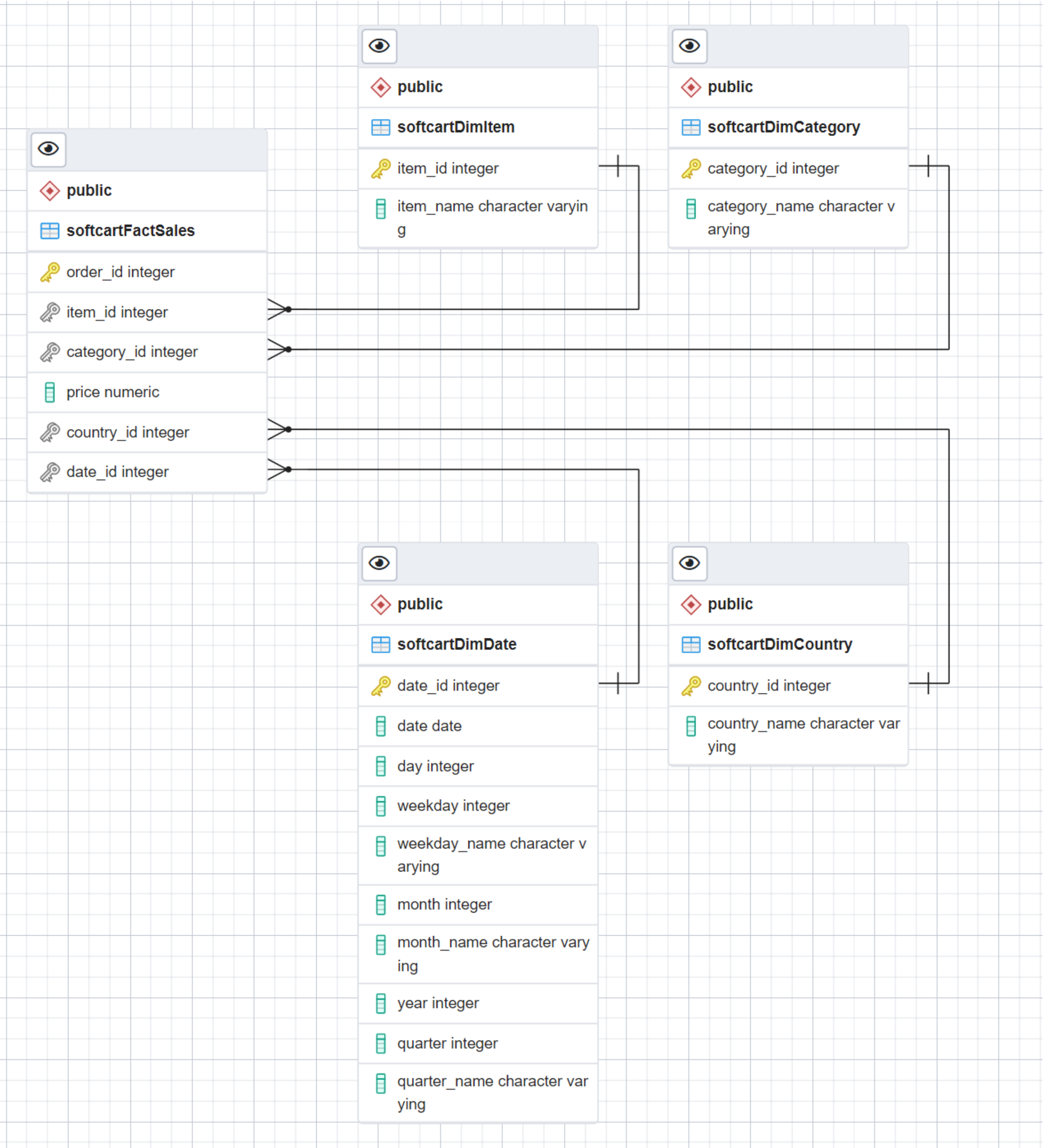
The ecommerce company has provided you the sample data.



OrderID	Item	Category	Price	Country	Date
2123	The Matrix	Movie	9.99	USA	20-Feb-21
3254	The Alchemist	Ebook	5.99	Canada	20-Feb-21
4901	Baby Shark	Song	2.49	Japan	20-Feb-21
5679	The Lord of the Rings	Ebook	6.99	Cyprus	20-Feb-21

You will start your project by designing a Star Schema for the warehouse by identifying the columns for the various dimension and fact tables in the schema. Name your database as softcart.

Creating tables and relationships amongst created tables, using the ERD design Tool of pgAdmin.



public
softcartFactSales
order_id integer
item_id integer
category_id integer
price numeric
country_id integer
date_id integer

public
softcartDimItem
item_id integer
item_name character varying

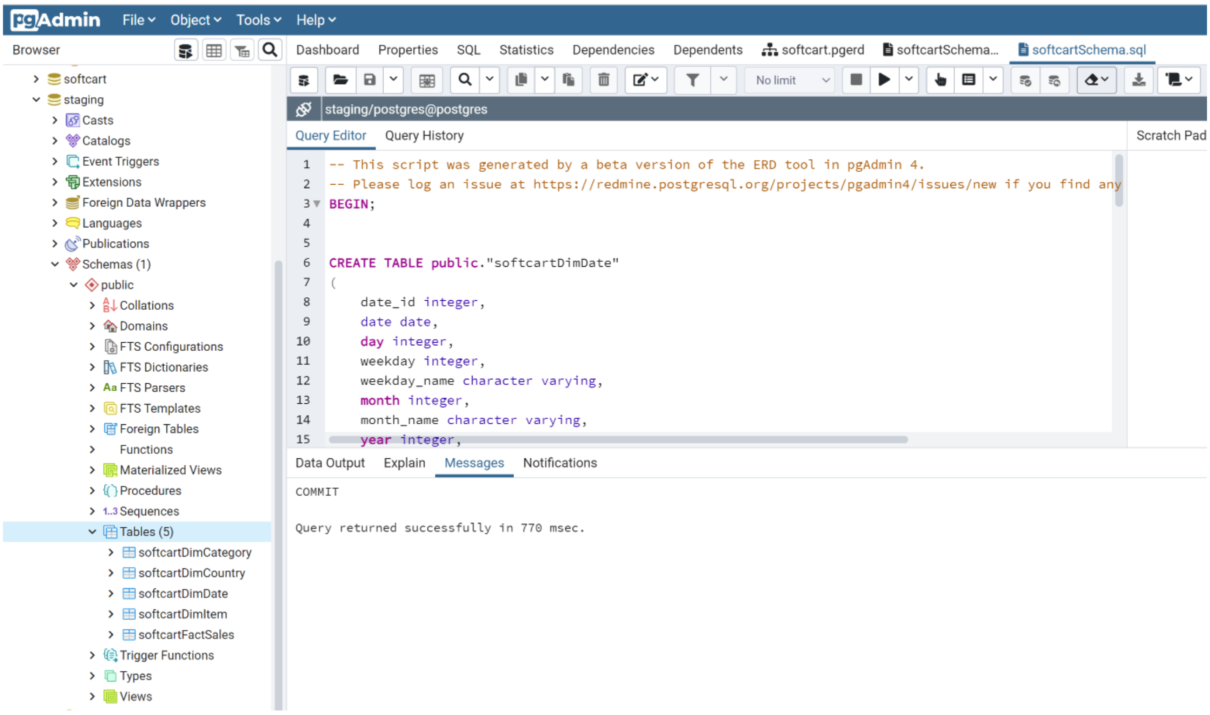
public
softcartDimCategory
category_id integer
category_name character varying

public
softcartDimDate
date_id integer
date date
day integer
weekday integer
weekday_name character varying
month integer
month_name character varying
year integer
quarter integer
quarter_name character varying

public
softcartDimCountry
country_id integer
country_name character varying

Create the schema

After creating tables and relationships using ERD design tool, generate the schema sql using the tool. Then use the created schema sql to create the schema in a database named staging.



```
-- This script was generated by a beta version of the ERD tool in pgAdmin 4.
-- Please log an issue at https://redmine.postgresql.org/projects/pgadmin4/issues/new if you find any
BEGIN;


CREATE TABLE public."softcartDimDate"
(
    date_id integer,
    date date,
    day integer,
    weekday integer,
    weekday_name character varying,
    month integer,
    month_name character varying,
    year integer,
```

```
COMMIT

Query returned successfully in 770 msec.
```

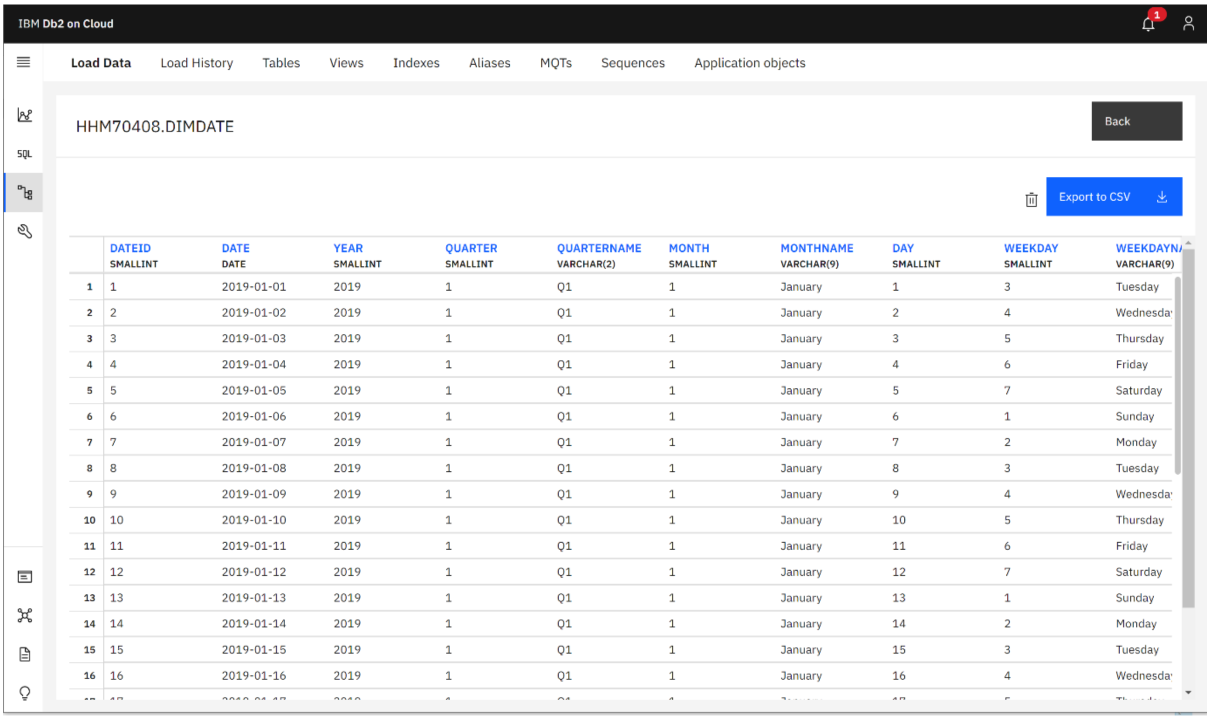
Module 3.2:

Scenario

You are a data engineer hired by an ecommerce company named SoftCart.com. The company retails download only items like E-Books, Movies, Songs etc. The company has international presence and customers from all over the world. You have designed the schema for the data warehouse in the previous assignment. Data engineering is a team game. Your senior data engineer reviewed your design. Your schema design was improvised to suit the production needs of the company. In this assignment you will generate reports out of the data in the data warehouse.

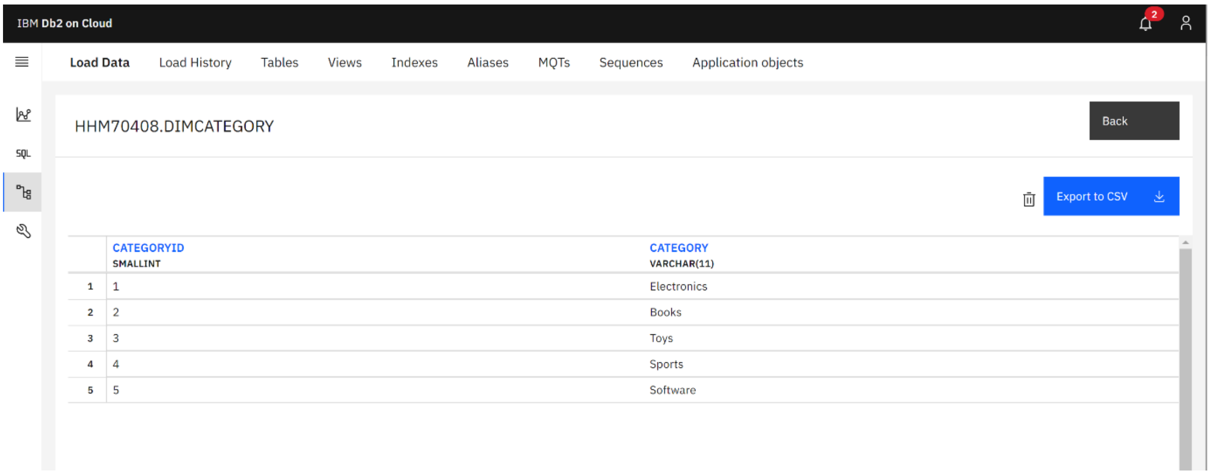
Load data into the Data Warehouse (IBM DB2)

Loading the data provided by the company in csv format into the tables (DimDate, DimCategory, DimCountry, FactSales).



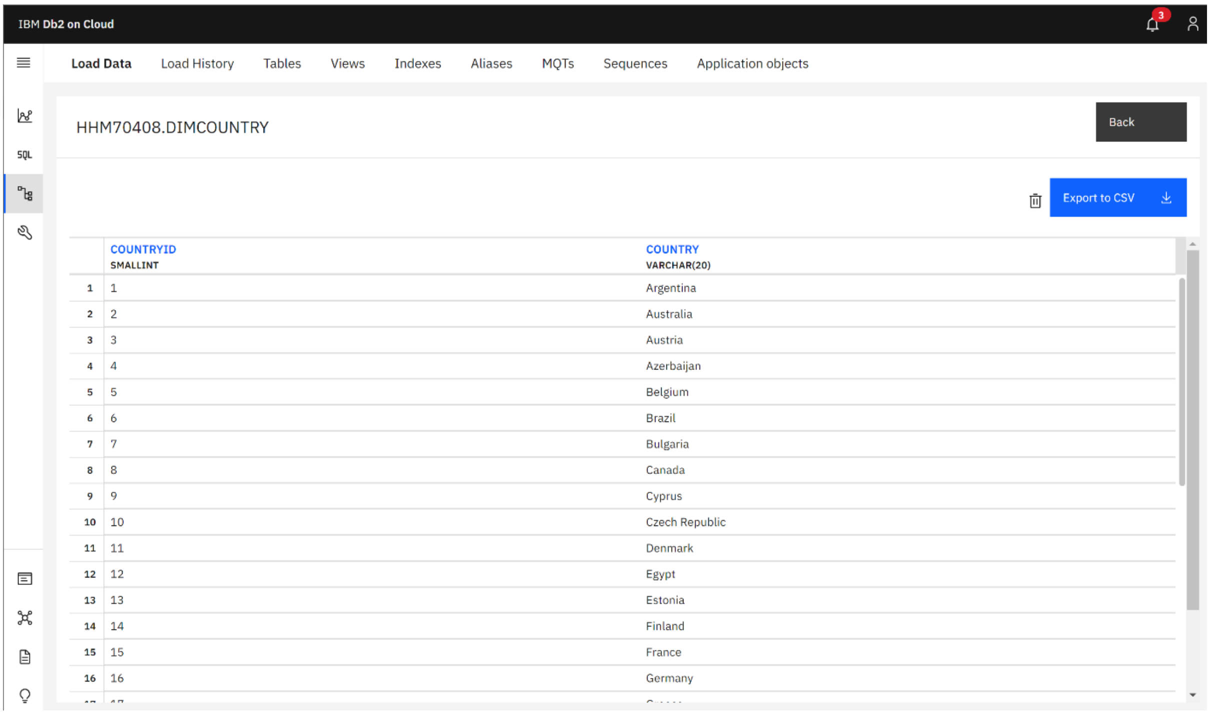
HHM70408.DIMDATE

	DATEID	DATE	YEAR	QUARTER	QUARTERNAME	MONTH	MONTHNAME	DAY	WEEKDAY	WEEKDAYNA
	SMALLINT	DATE	SMALLINT	SMALLINT	VARCHAR(2)	SMALLINT	VARCHAR(9)	SMALLINT	SMALLINT	VARCHAR(9)
1	1	2019-01-01	2019	1	Q1	1	January	1	3	Tuesday
2	2	2019-01-02	2019	1	Q1	1	January	2	4	Wednesda
3	3	2019-01-03	2019	1	Q1	1	January	3	5	Thursday
4	4	2019-01-04	2019	1	Q1	1	January	4	6	Friday
5	5	2019-01-05	2019	1	Q1	1	January	5	7	Saturday
6	6	2019-01-06	2019	1	Q1	1	January	6	1	Sunday
7	7	2019-01-07	2019	1	Q1	1	January	7	2	Monday
8	8	2019-01-08	2019	1	Q1	1	January	8	3	Tuesday
9	9	2019-01-09	2019	1	Q1	1	January	9	4	Wednesda
10	10	2019-01-10	2019	1	Q1	1	January	10	5	Thursday
11	11	2019-01-11	2019	1	Q1	1	January	11	6	Friday
12	12	2019-01-12	2019	1	Q1	1	January	12	7	Saturday
13	13	2019-01-13	2019	1	Q1	1	January	13	1	Sunday
14	14	2019-01-14	2019	1	Q1	1	January	14	2	Monday
15	15	2019-01-15	2019	1	Q1	1	January	15	3	Tuesday
16	16	2019-01-16	2019	1	Q1	1	January	16	4	Wednesda



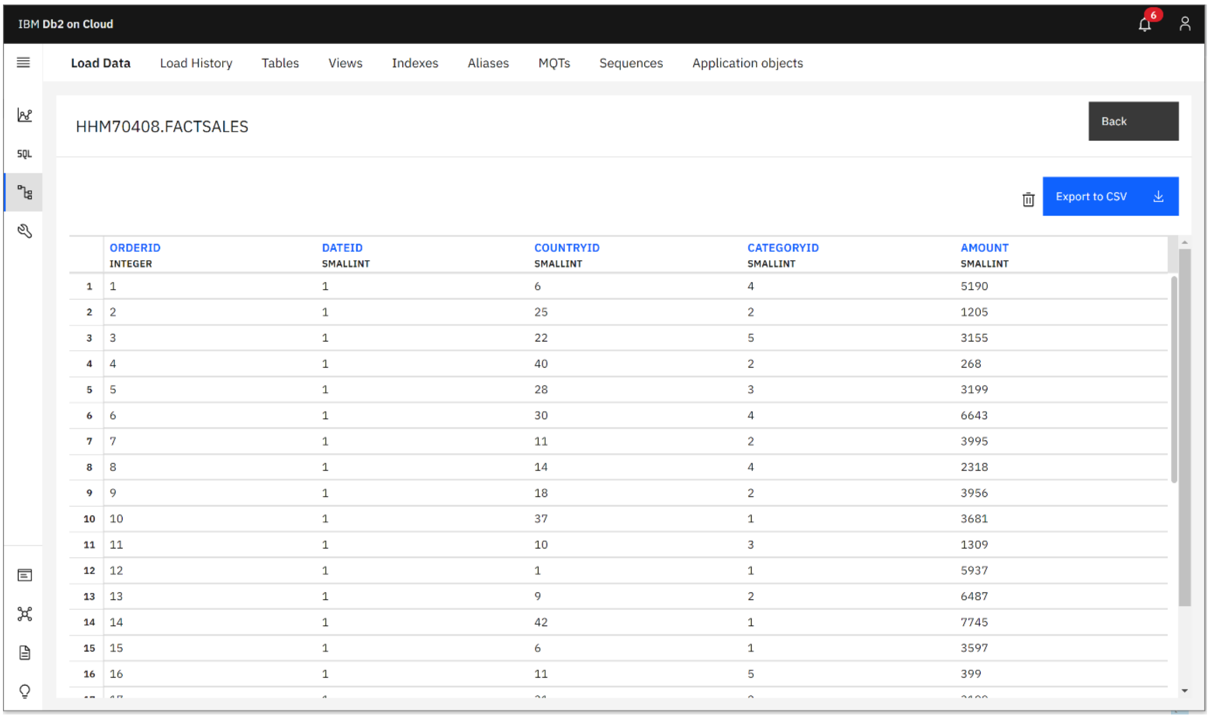
HHM70408.DIMCATEGORY

	CATEGORYID	CATEGORY
	SMALLINT	VARCHAR(11)
1	1	Electronics
2	2	Books
3	3	Toys
4	4	Sports
5	5	Software



HHM70408.DIMCOUNTRY

	COUNTRYID SMALLINT	COUNTRY VARCHAR(20)
1	1	Argentina
2	2	Australia
3	3	Austria
4	4	Azerbaijan
5	5	Belgium
6	6	Brazil
7	7	Bulgaria
8	8	Canada
9	9	Cyprus
10	10	Czech Republic
11	11	Denmark
12	12	Egypt
13	13	Estonia
14	14	Finland
15	15	France
16	16	Germany



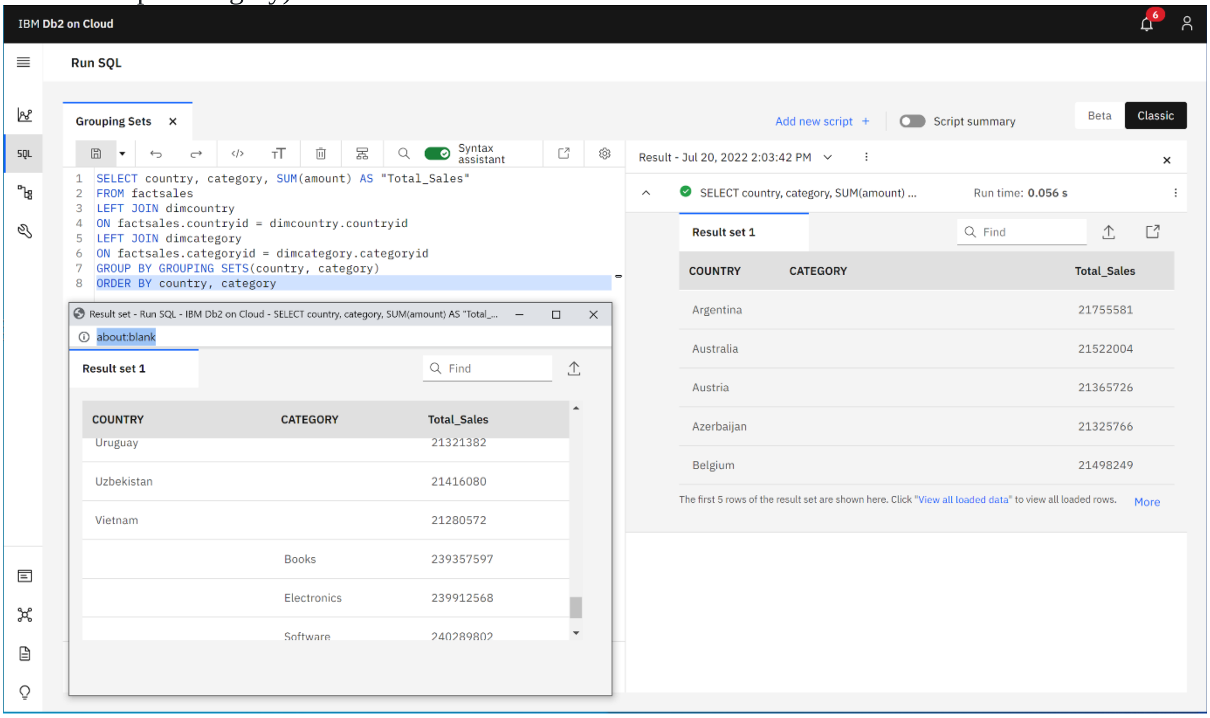
HHM70408.FACTSALES

	ORDERID INTEGER	DATEID SMALLINT	COUNTRYID SMALLINT	CATEGORYID SMALLINT	AMOUNT SMALLINT
1	1	1	6	4	5190
2	2	1	25	2	1205
3	3	1	22	5	3155
4	4	1	40	2	268
5	5	1	28	3	3199
6	6	1	30	4	6643
7	7	1	11	2	3995
8	8	1	14	4	2318
9	9	1	18	2	3956
10	10	1	37	1	3681
11	11	1	10	3	1309
12	12	1	1	1	5937
13	13	1	9	2	6487
14	14	1	42	1	7745
15	15	1	6	1	3597
16	16	1	11	5	399

Queries for data analytics

Creating and running queries to check if the created data warehouse can create reports.

Grouping sets query using the columns country, category, totalsales. (Total sales per country and Total sales per category)



```
SELECT country, category, SUM(amount) AS "Total_Sales"
FROM factsales
LEFT JOIN dimcountry
ON factsales.countryid = dimcountry.countryid
LEFT JOIN dimcategory
ON factsales.categoryid = dimcategory.categoryid
GROUP BY GROUPING SETS(country, category)
ORDER BY country, category
```

Result set - Run SQL - IBM Db2 on Cloud - SELECT country, category, SUM(amount) AS "Total_...

about:blank

Result set 1

COUNTRY	CATEGORY	Total_Sales
Uruguay		21321382
Uzbekistan		21416080
Vietnam		21280572
	Books	239357597
	Electronics	239912568
	Software	240289802

Result - Jul 20, 2022 2:03:42 PM

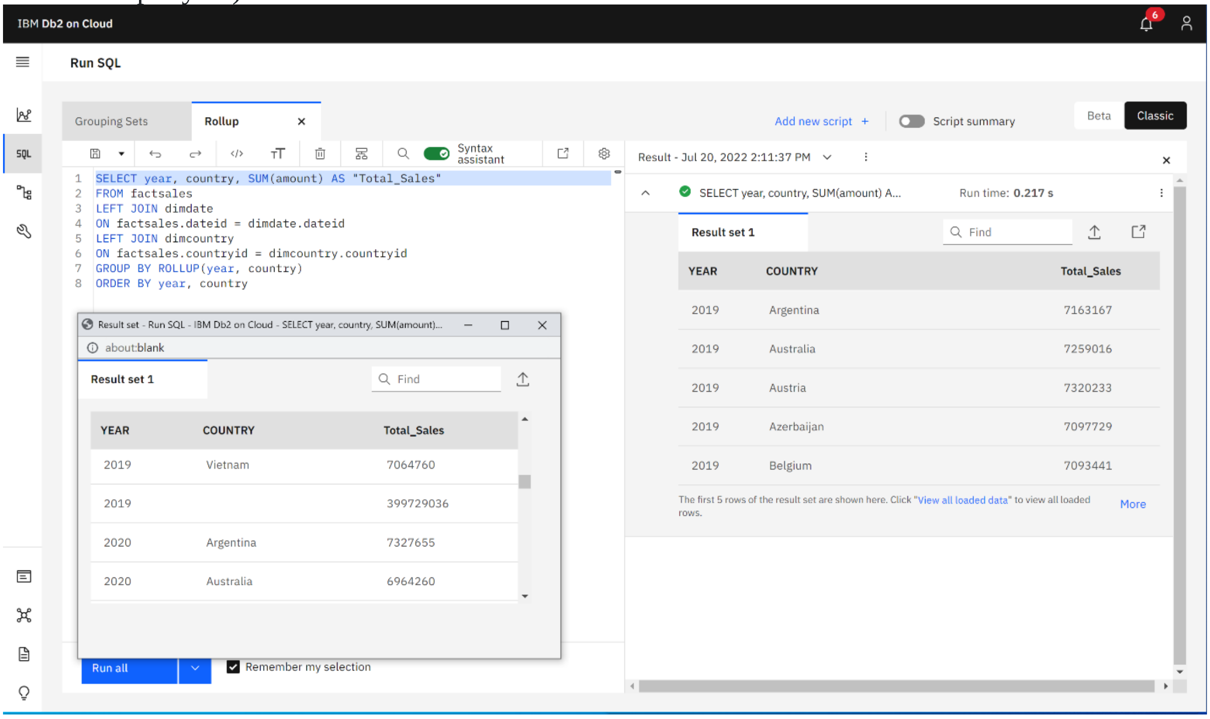
SELECT country, category, SUM(amount) ... Run time: 0.056 s

Result set 1

COUNTRY	CATEGORY	Total_Sales
Argentina		21755581
Australia		21522004
Austria		21365726
Azerbaijan		21325766
Belgium		21498249

The first 5 rows of the result set are shown here. Click "View all loaded data" to view all loaded rows. More

Rollup query using the columns year, country, and totalsales. (Total sales per year per country and Total sales per year)



```
SELECT year, country, SUM(amount) AS "Total_Sales"
FROM factsales
LEFT JOIN dimdate
ON factsales.dateid = dimdate.dateid
LEFT JOIN dimcountry
ON factsales.countryid = dimcountry.countryid
GROUP BY ROLLUP(year, country)
ORDER BY year, country
```

Result set - Run SQL - IBM Db2 on Cloud - SELECT year, country, SUM(amount)...

about:blank

Result set 1

YEAR	COUNTRY	Total_Sales
2019	Vietnam	7064760
2019		399729036
2020	Argentina	7327655
2020	Australia	6964260

Result - Jul 20, 2022 2:11:37 PM

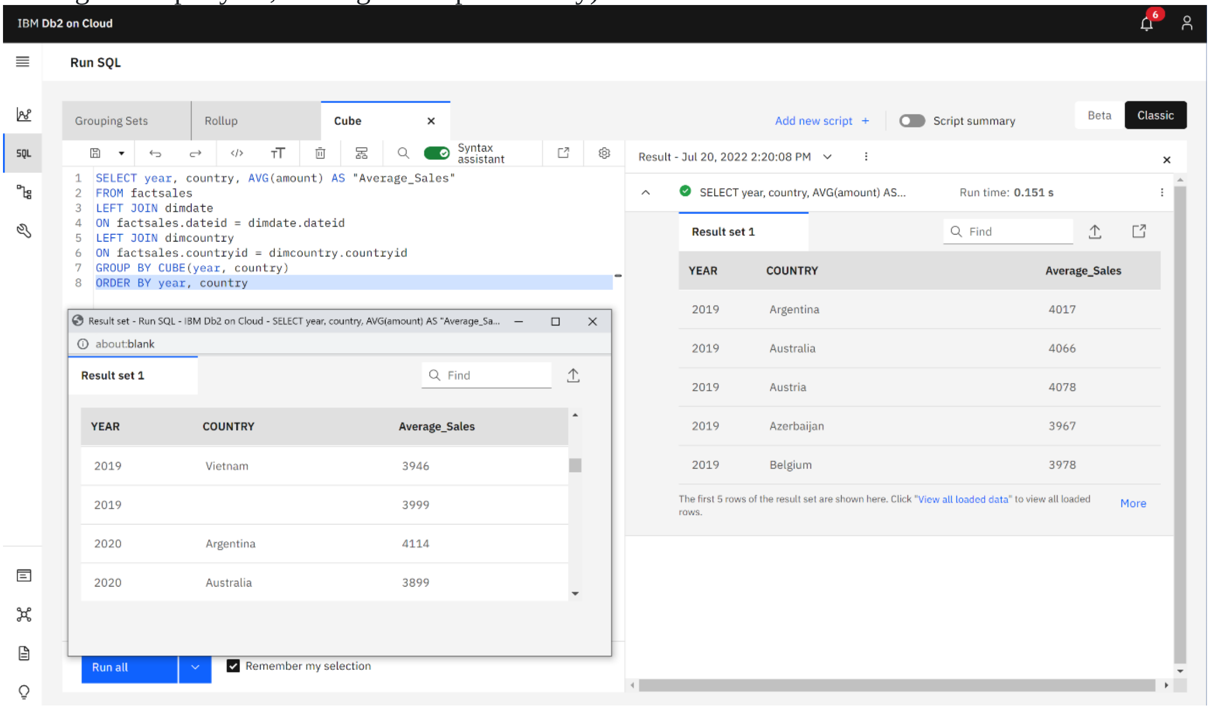
SELECT year, country, SUM(amount) A... Run time: 0.217 s

Result set 1

YEAR	COUNTRY	Total_Sales
2019	Argentina	7163167
2019	Australia	7259016
2019	Austria	7320233
2019	Azerbaijan	7097729
2019	Belgium	7093441

The first 5 rows of the result set are shown here. Click "View all loaded data" to view all loaded rows. More

Cube query using the columns year, country, and average sales. (Average sales per year per country, Average sales per year, Average sales per country)



```
SELECT year, country, AVG(amount) AS "Average_Sales"
FROM factsales
LEFT JOIN dimdate
ON factsales.dateid = dimdate.dateid
LEFT JOIN dimcountry
ON factsales.countryid = dimcountry.countryid
GROUP BY CUBE(year, country)
ORDER BY year, country
```

Result set - Run SQL - IBM Db2 on Cloud - SELECT year, country, AVG(amount) AS "Average_Sa...

about:blank

Result set 1

YEAR	COUNTRY	Average_Sales
2019	Vietnam	3946
2019		3999
2020	Argentina	4114
2020	Australia	3899

Result - Jul 20, 2022 2:20:08 PM

SELECT year, country, AVG(amount) AS... Run time: 0.151 s

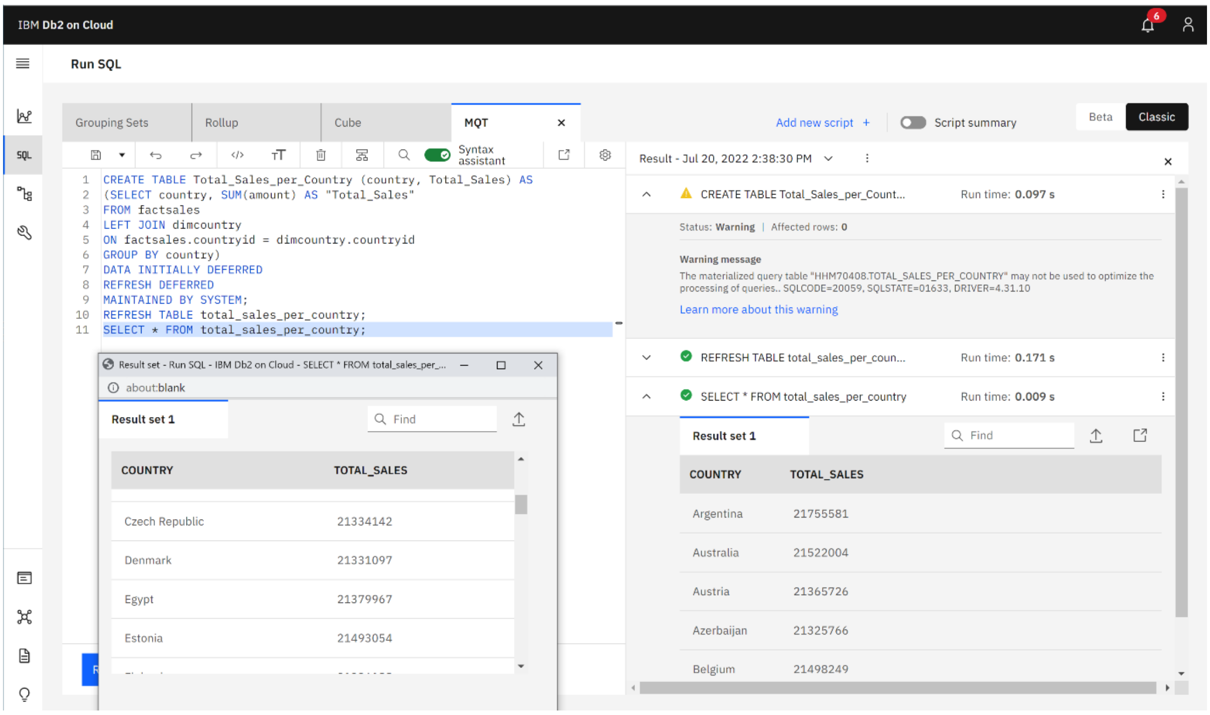
Result set 1

YEAR	COUNTRY	Average_Sales
2019	Argentina	4017
2019	Australia	4066
2019	Austria	4078
2019	Azerbaijan	3967
2019	Belgium	3978

The first 5 rows of the result set are shown here. Click "View all loaded data" to view all loaded rows.

Creating a materialized query table (MQT) named total_sales_per_country that has the columns country and total_sales to improve the performance of complex queries that operate on very large amounts of data.

Db2 uses a materialized query table to precompute the results of data that is derived from one or more tables. When you submit a query, Db2 can use the results that are stored in a materialized query table rather than compute the results from the underlying source tables on which the materialized query table is defined.



```
CREATE TABLE Total_Sales_per_Country (country, Total_Sales) AS
(SELECT country, SUM(amount) AS "Total_Sales"
FROM factsales
LEFT JOIN dimcountry
ON factsales.countryid = dimcountry.countryid
GROUP BY country)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY SYSTEM;
REFRESH TABLE total_sales_per_country;
SELECT * FROM total_sales_per_country;
```

Result set - Run SQL - IBM Db2 on Cloud - SELECT * FROM total_sales_per_...

about:blank

Result set 1

COUNTRY	TOTAL_SALES
Czech Republic	21334142
Denmark	21331097
Egypt	21379967
Estonia	21493054

Result - Jul 20, 2022 2:38:30 PM

CREATE TABLE Total_Sales_per_Count... Run time: 0.097 s

Status: Warning | Affected rows: 0

Warning message

The materialized query table "HHM70408.TOTAL_SALES_PER_COUNTRY" may not be used to optimize the processing of queries.. SQLCODE=20059, SQLSTATE=01633, DRIVER=4.31.10

Learn more about this warning

REFRESH TABLE total_sales_per_coun... Run time: 0.171 s

SELECT * FROM total_sales_per_country Run time: 0.009 s

Result set 1

COUNTRY	TOTAL_SALES
Argentina	21755581
Australia	21522004
Austria	21365726
Azerbaijan	21325766
Belgium	21498249

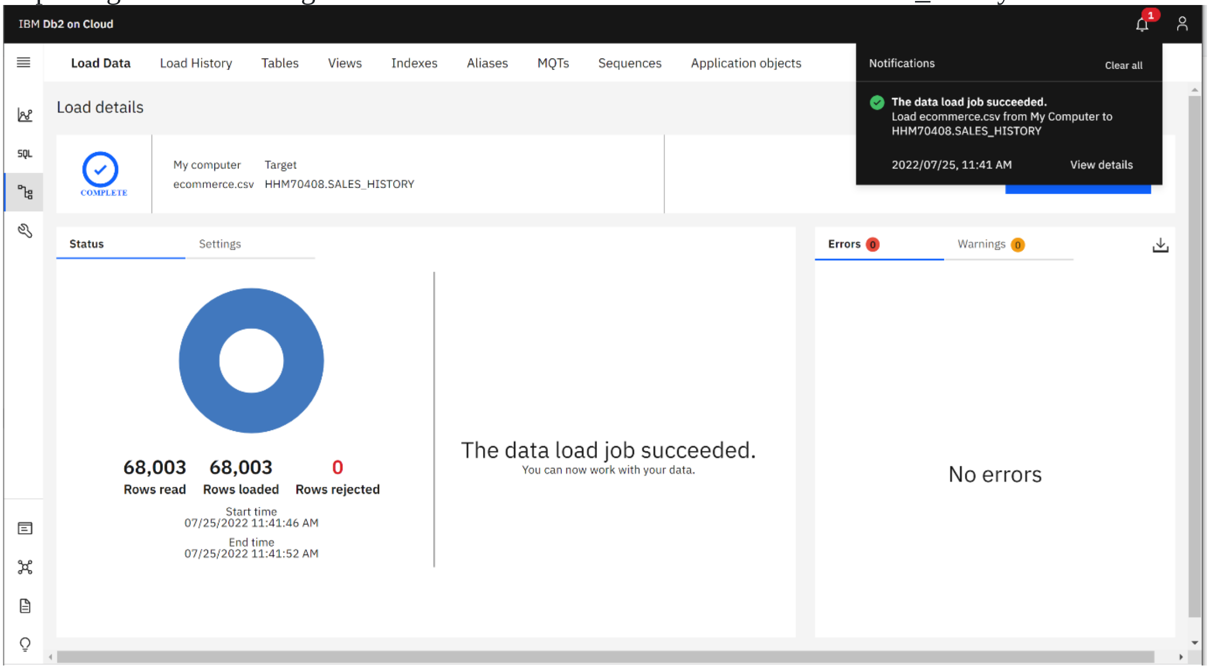
Module 4:

Scenario

You are a data engineer at an e-commerce company. Your company has finished setting up a data warehouse. Now you are assigned the responsibility to design a reporting dashboard that reflects the key metrics of the business.

Load data into the data warehouse (IBM DB2)

Importing the data in the given file “ecommerce.csv” into a table named sales_history.



IBM Db2 on Cloud

Load Data | Load History | Tables | Views | Indexes | Aliases | MQTs | Sequences | Application objects

Notifications — Clear all

The data load job succeeded.
Load ecommerce.csv from My Computer to HHM70408.SALES_HISTORY
2022/07/25, 11:41 AM — View details

Load details

COMPLETE

My computer	Target
ecommerce.csv	HHM70408.SALES_HISTORY

Status | Settings

68,003	68,003	0
Rows read	Rows loaded	Rows rejected

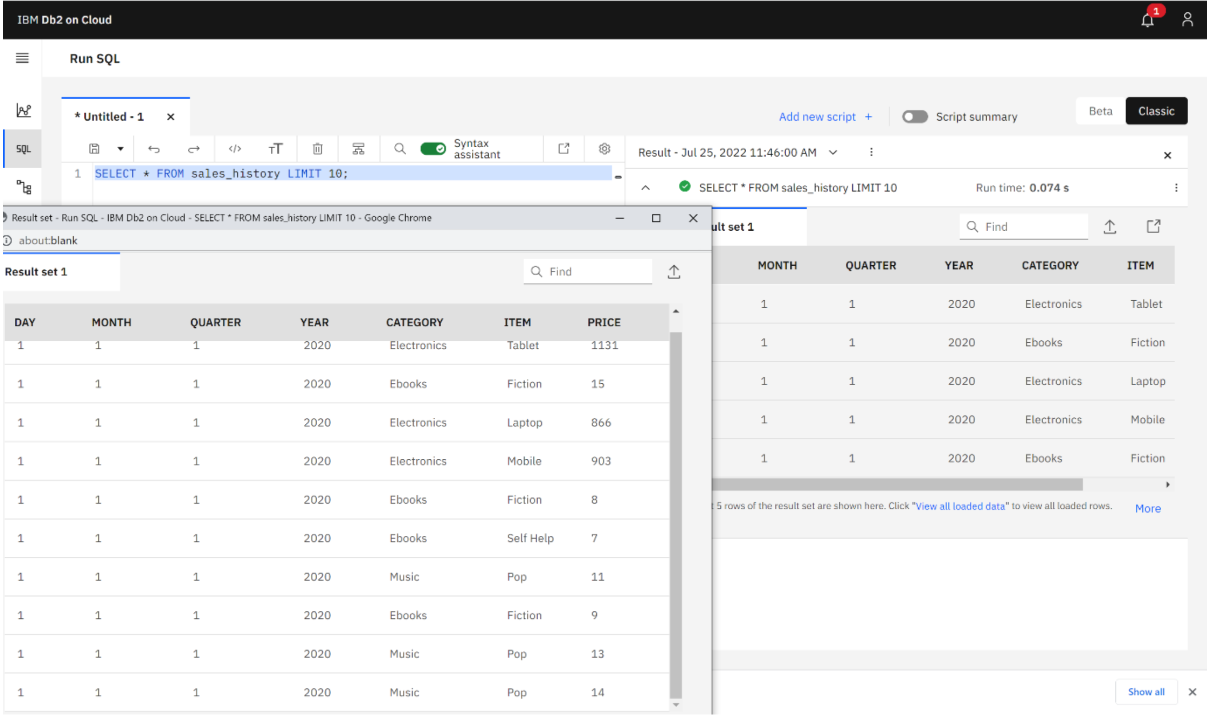
Start time
07/25/2022 11:41:46 AM
End time
07/25/2022 11:41:52 AM

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

Errors 0 | Warnings 0

No errors

Listing the first 10 rows in the sales_history table to check the loaded data.



IBM Db2 on Cloud

Run SQL

* Untitled - 1

```sql
SELECT * FROM sales_history LIMIT 10;
```

Add new script + | Script summary | Beta | Classic

Result - Jul 25, 2022 11:46:00 AM

SELECT * FROM sales_history LIMIT 10 — Run time: 0.074 s

Result set - Run SQL - IBM Db2 on Cloud - SELECT * FROM sales_history LIMIT 10 - Google Chrome
about:blank

Result set 1

DAY	MONTH	QUARTER	YEAR	CATEGORY	ITEM	PRICE
1	1	1	2020	Electronics	Tablet	1131
1	1	1	2020	Ebooks	Fiction	15
1	1	1	2020	Electronics	Laptop	866
1	1	1	2020	Electronics	Mobile	903
1	1	1	2020	Ebooks	Fiction	8
1	1	1	2020	Ebooks	Self Help	7
1	1	1	2020	Music	Pop	11
1	1	1	2020	Ebooks	Fiction	9
1	1	1	2020	Music	Pop	13
1	1	1	2020	Music	Pop	14

ult set 1

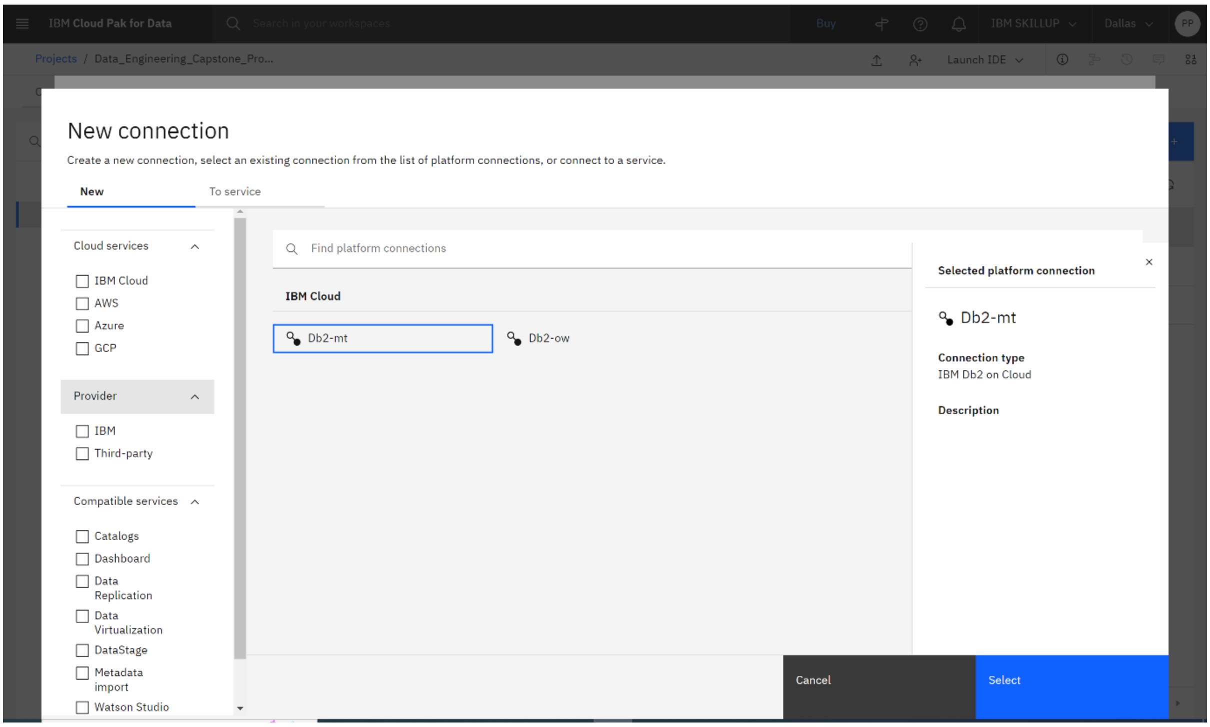
MONTH	QUARTER	YEAR	CATEGORY	ITEM
1	1	2020	Electronics	Tablet
1	1	2020	Ebooks	Fiction
1	1	2020	Electronics	Laptop
1	1	2020	Electronics	Mobile
1	1	2020	Ebooks	Fiction

t 5 rows of the result set are shown here. Click "View all loaded data" to view all loaded rows. More

Show all

Create data source in IBM Cognos

Creating connection between sales_history table in IBM DB2 and Watson Studio with added Cognos Dashboard Embedded (CDE) service to use it as a data source.



New connection

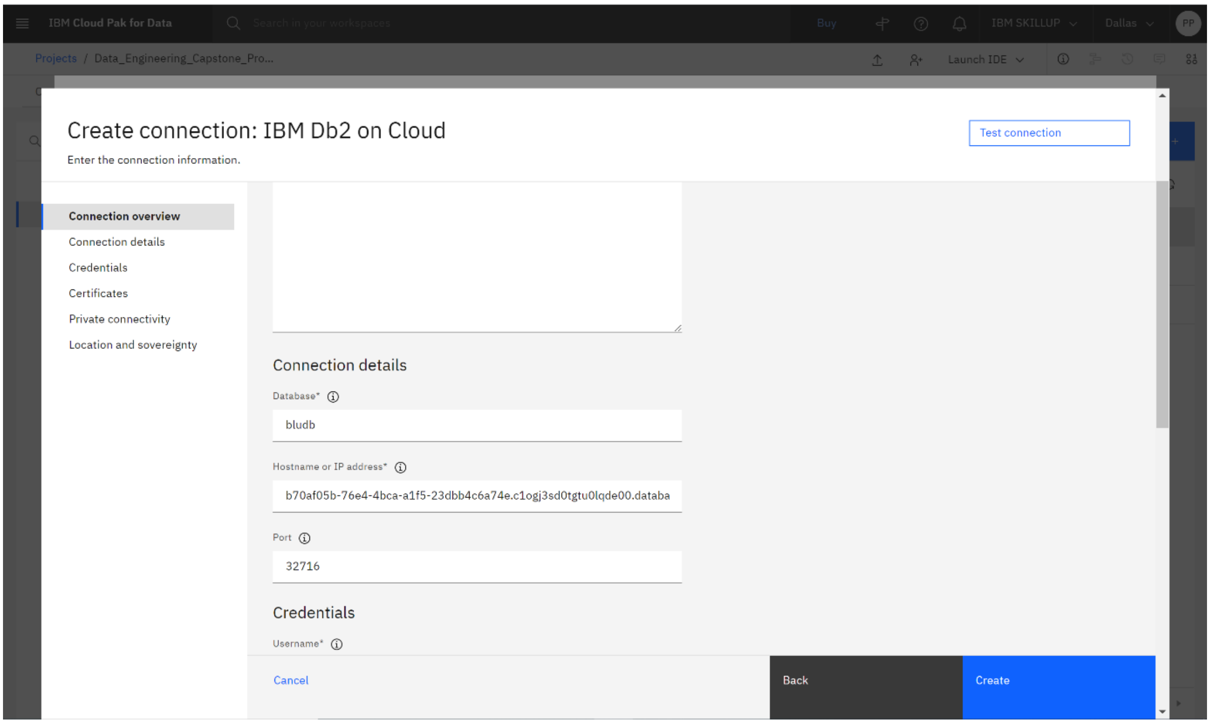
Create a new connection, select an existing connection from the list of platform connections, or connect to a service.

Selected platform connection

Db2-mt

Connection type
IBM Db2 on Cloud

Description



Create connection: IBM Db2 on Cloud

Enter the connection information.

Connection details

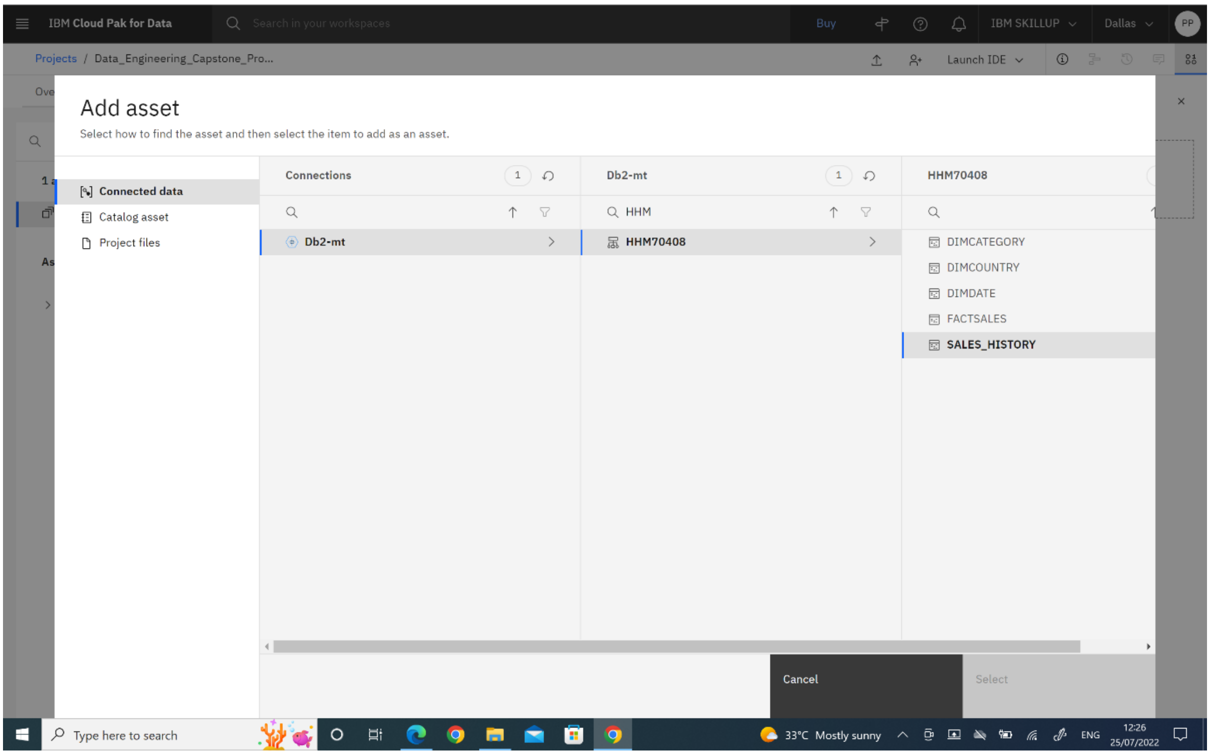
Database*: bludb

Hostname or IP address*: b70af05b-76e4-4bca-a1f5-23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databa

Port: 32716

Credentials

Username*

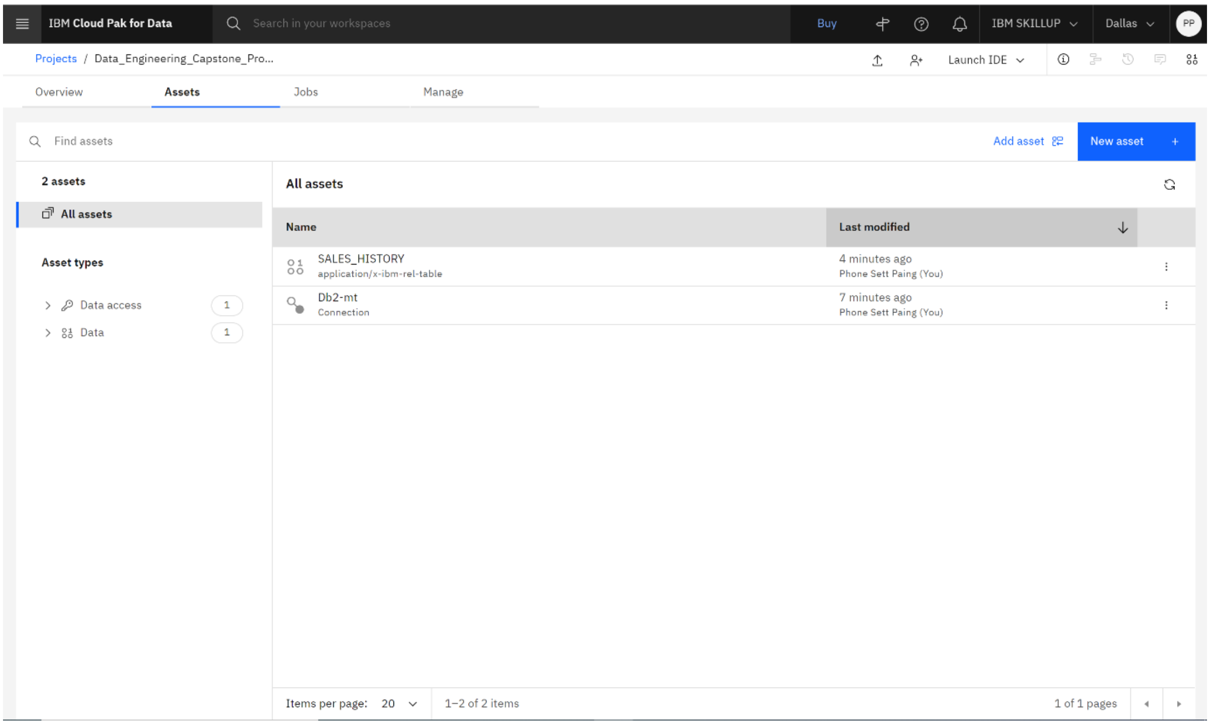


Add asset

Select how to find the asset and then select the item to add as an asset.

- Connected data
- Catalog asset
- Project files

Connections 1	Db2-mt 1	HHM70408
Db2-mt	HHM	
	HHM70408	DIMCATEGORY
		DIMCOUNTRY
		DIMDATE
		FACTSALES
		SALES_HISTORY



2 assets

All assets

Asset types

Data access	1
Data	1

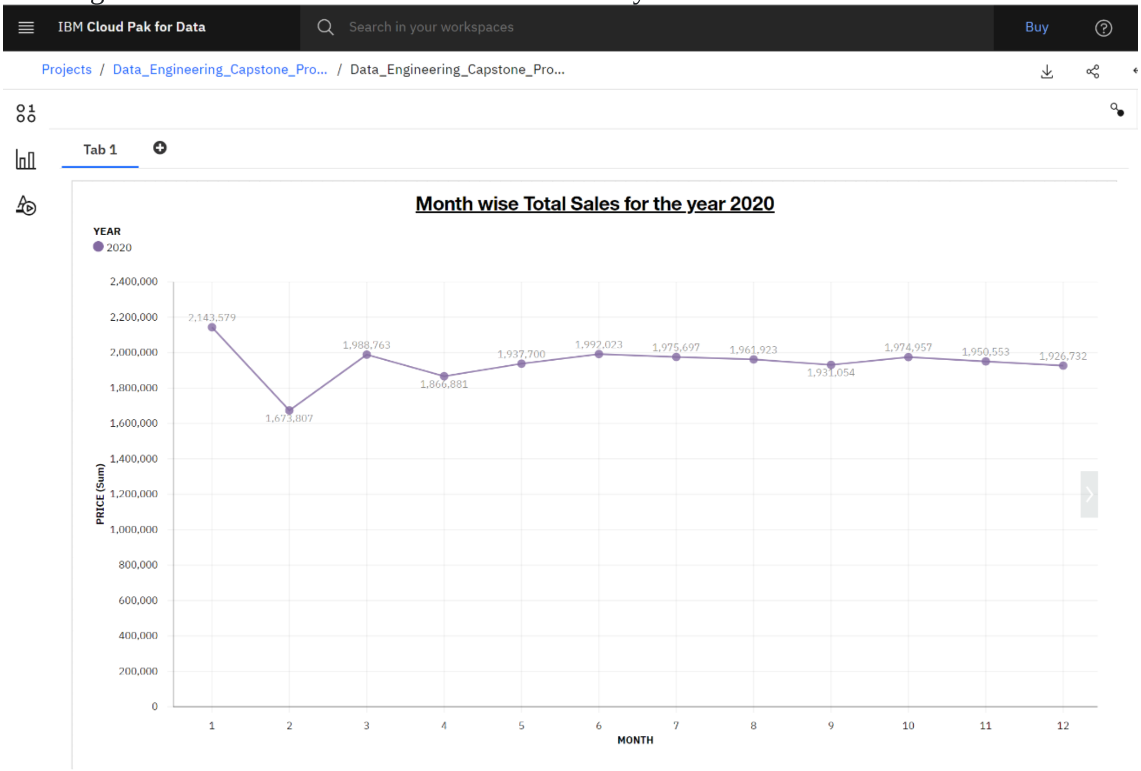
All assets

Name	Last modified
SALES_HISTORY application/x-ibm-rel-table	4 minutes ago Phone Sett Paing (You)
Db2-mt Connection	7 minutes ago Phone Sett Paing (You)

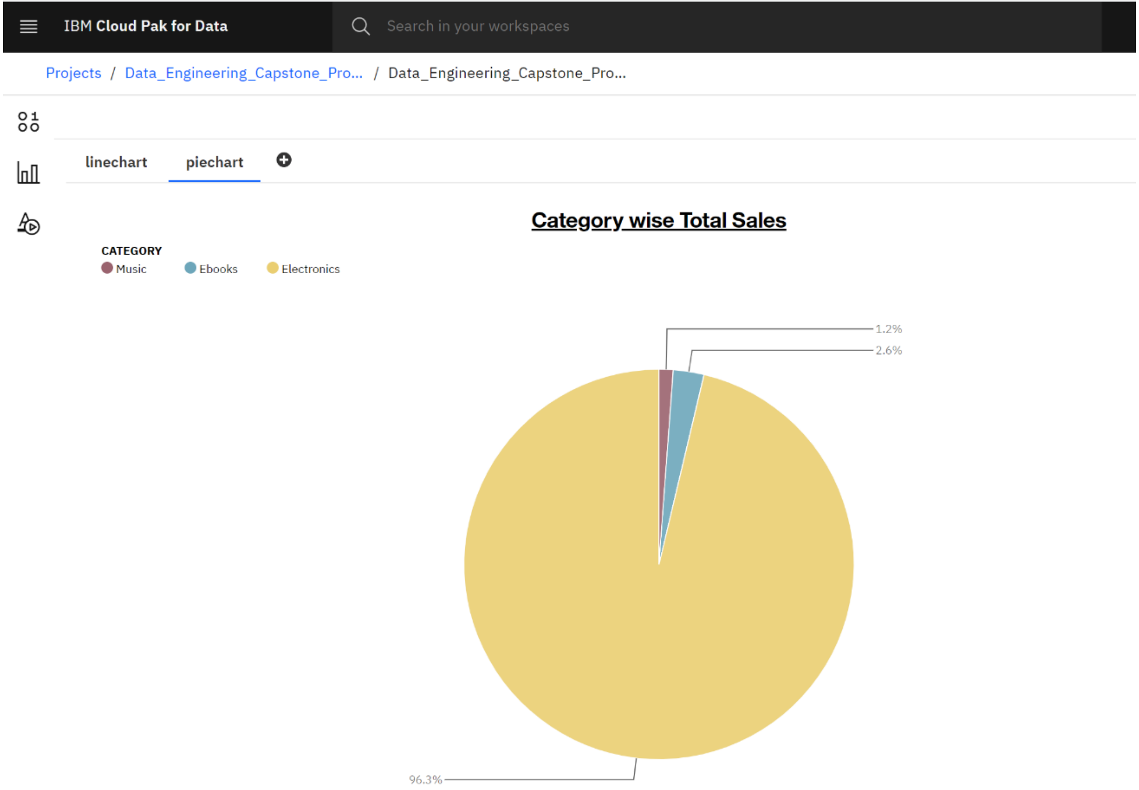
Items per page: 20 1–2 of 2 items 1 of 1 pages

Create dashboard using IBM Cognos

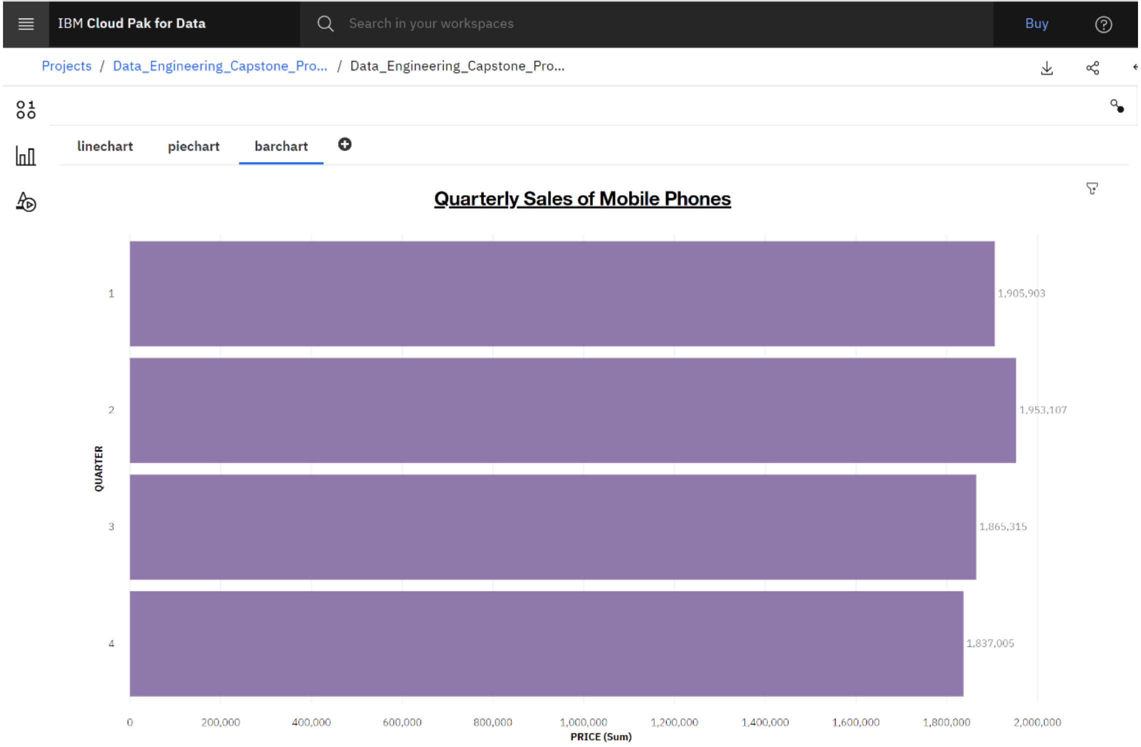
Creating a line chart of month wise total sales for the year 2020.



Creating a pie chart of category wise total sales.


IBM Cloud Pak for Data
Search in your workspaces
Projects / Data_Engineering_Capstone_Pro... / Data_Engineering_Capstone_Pro...
linechart
piechart
Category wise Total Sales
CATEGORY
● Music
● Ebooks
● Electronics
1.2%
2.6%
96.3%

Creating a bar chart of Quarterly sales of mobile phones.



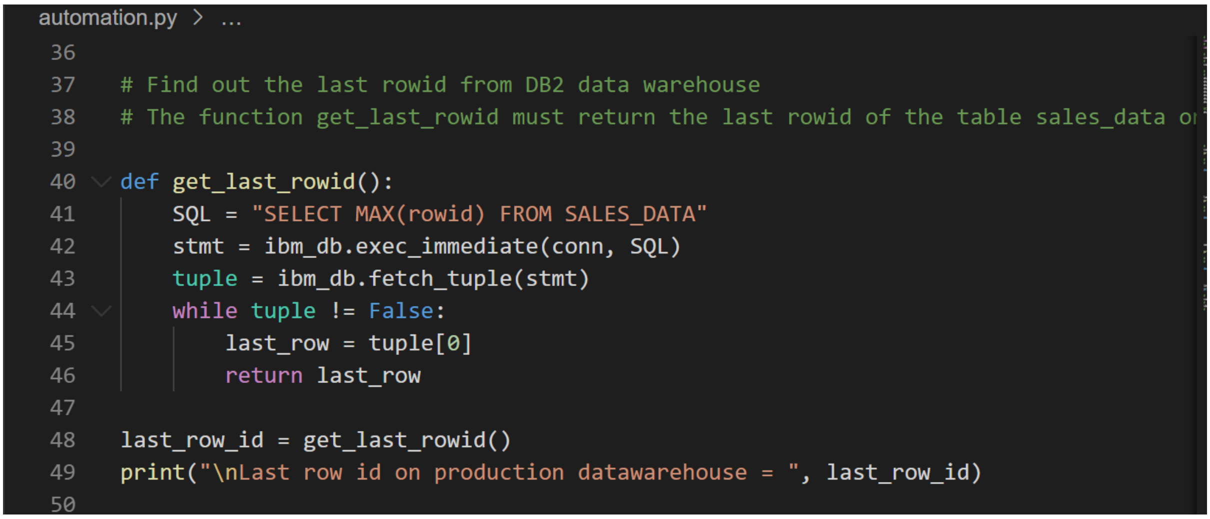
Module 5.1:

Scenario

You are a data engineer at an e-commerce company. You need to keep data synchronized between different databases/data warehouses as a part of your daily routine. One task that is routinely performed is the sync up of staging data warehouse and production data warehouse. Automating this sync up will save you a lot of time and standardize your process. You will be given a set of python scripts to start with. You will use/modify them to perform the incremental data load from MySQL server which acts as a staging warehouse to the IBM DB2 which is a production data warehouse. This script will be scheduled by the data engineers to sync up the data between the staging and production data warehouse.

Automate loading of incremental data into the data warehouse

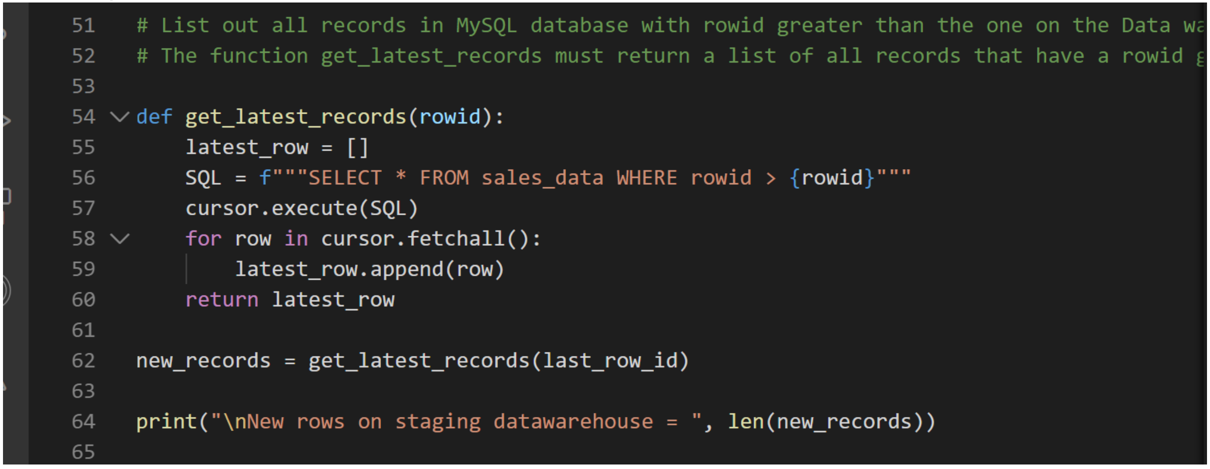
Creating the function get_last_rowid() that connect to the DB2 data warehouse and return the last rowid of the table sales_data.



automation.py > ...

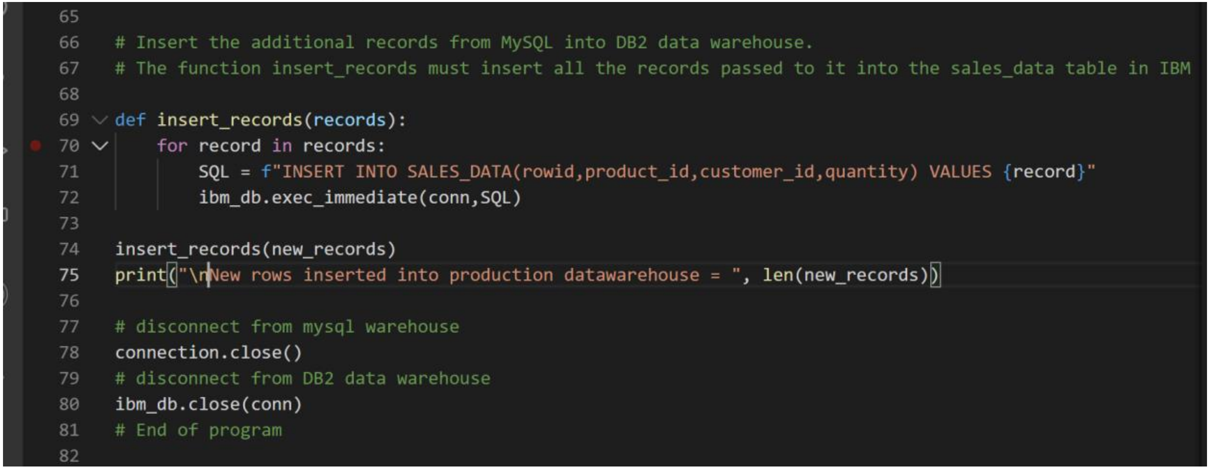
```python

# Find out the last rowid from DB2 data warehouse
# The function get_last_rowid must return the last rowid of the table sales_data or

def get_last_rowid():
    SQL = "SELECT MAX(rowid) FROM SALES_DATA"
    stmt = ibm_db.exec_immediate(conn, SQL)
    tuple = ibm_db.fetch_tuple(stmt)
    while tuple != False:
        last_row = tuple[0]
        return last_row

last_row_id = get_last_rowid()
print("\nLast row id on production datawarehouse = ", last_row_id)

```

Creating the function get_latest_records() that connect to MySQL database and return all records later than the given last_rowid from DB2 data warehouse.



```python
# List out all records in MySQL database with rowid greater than the one on the Data wa
# The function get_latest_records must return a list of all records that have a rowid g

def get_latest_records(rowid):
    latest_row = []
    SQL = f"""SELECT * FROM sales_data WHERE rowid > {rowid}"""
    cursor.execute(SQL)
    for row in cursor.fetchall():
        latest_row.append(row)
    return latest_row

new_records = get_latest_records(last_row_id)

print("\nNew rows on staging datawarehouse = ", len(new_records))

```

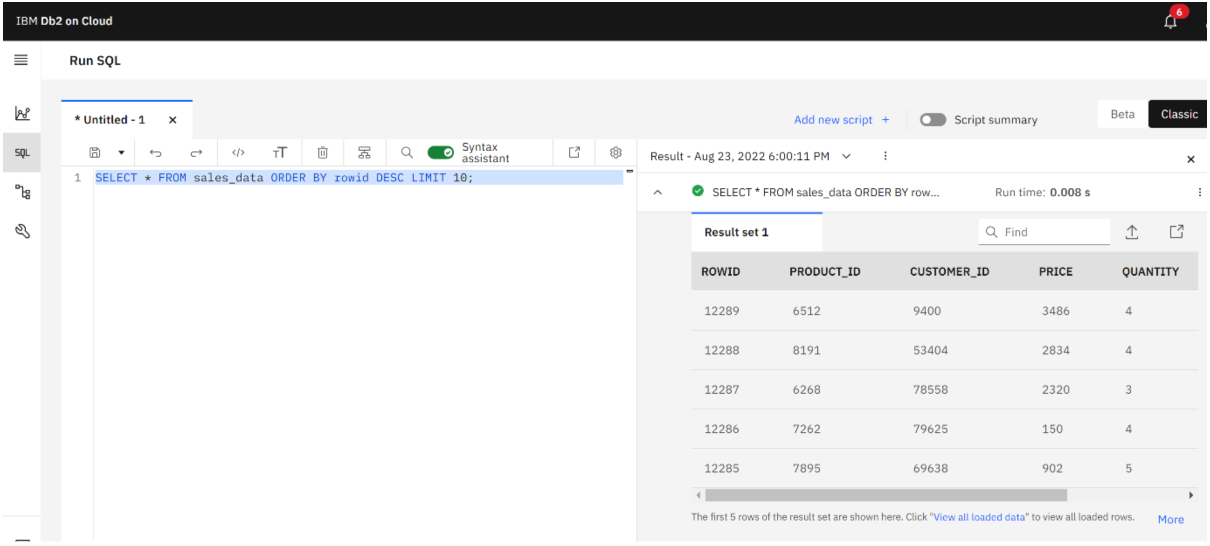
Creating the function insert_records() that connect to the DB2 data warehouse and insert all the given records (latest records from MySQL database).



```python

# Insert the additional records from MySQL into DB2 data warehouse.
# The function insert_records must insert all the records passed to it into the sales_data table in IBM

def insert_records(records):
    for record in records:
        SQL = f"INSERT INTO SALES_DATA(rowid,product_id,customer_id,quantity) VALUES {record}"
        ibm_db.exec_immediate(conn,SQL)

insert_records(new_records)
print("\nNew rows inserted into production datawarehouse = ", len(new_records))

# disconnect from mysql warehouse
connection.close()
# disconnect from DB2 data warehouse
ibm_db.close(conn)
# End of program

```

Testing the data synchronization

Data in DB2 data warehouse before synchronization



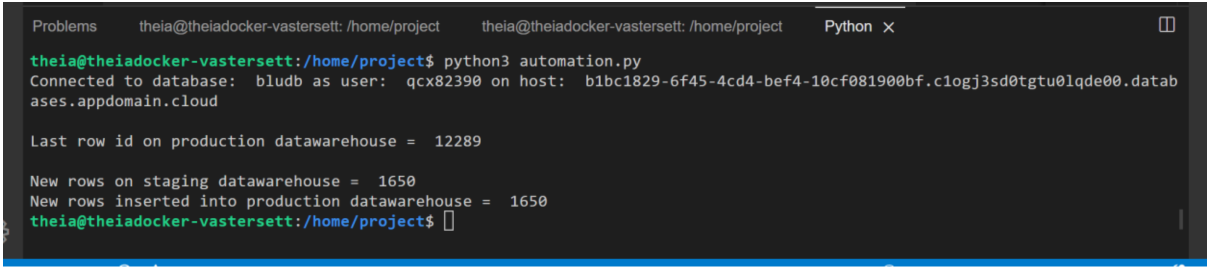
```
SELECT * FROM sales_data ORDER BY rowid DESC LIMIT 10;
```

SELECT * FROM sales_data ORDER BY row... Run time: 0.008 s

ROWID	PRODUCT_ID	CUSTOMER_ID	PRICE	QUANTITY
12289	6512	9400	3486	4
12288	8191	53404	2834	4
12287	6268	78558	2320	3
12286	7262	79625	150	4
12285	7895	69638	902	5

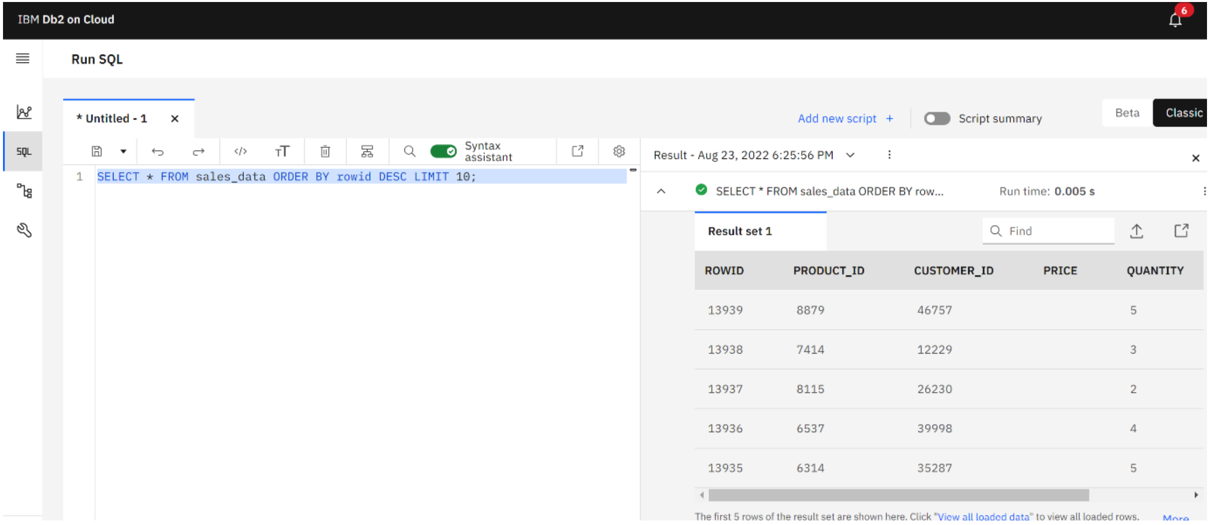
The first 5 rows of the result set are shown here. Click "View all loaded data" to view all loaded rows.

Running automation.py for data synchronization


Problems
theia@theiadocker-vastersett: /home/project
theia@theiadocker-vastersett: /home/project
Python

```
theia@theiadocker-vastersett:/home/project$ python3 automation.py
Connected to database:  bludb as user:  qcx82390 on host:  b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.datab
ases.appdomain.cloud

Last row id on production datawarehouse =  12289

New rows on staging datawarehouse =  1650
New rows inserted into production datawarehouse =  1650
theia@theiadocker-vastersett:/home/project$ 
```

Data in DB2 data warehouse after synchronization



```
SELECT * FROM sales_data ORDER BY rowid DESC LIMIT 10;
```

SELECT * FROM sales_data ORDER BY row... Run time: 0.005 s

ROWID	PRODUCT_ID	CUSTOMER_ID	PRICE	QUANTITY
13939	8879	46757		5
13938	7414	12229		3
13937	8115	26230		2
13936	6537	39998		4
13935	6314	35287		5

The first 5 rows of the result set are shown here. Click "View all loaded data" to view all loaded rows.

Data have been successfully synchronized.

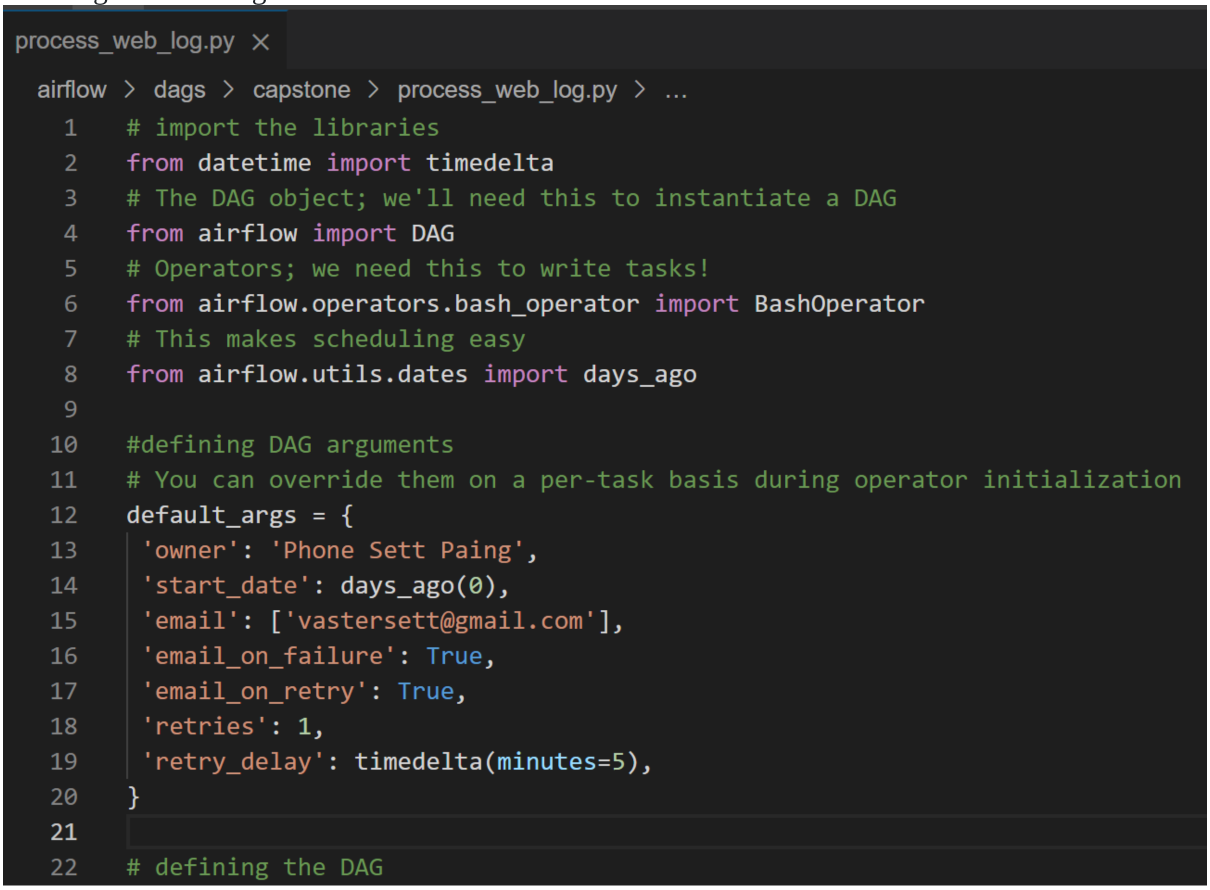
Module 5.2:

Scenario

Write a pipeline that analyses the web server log file, extracts the required fields, transforms and load.

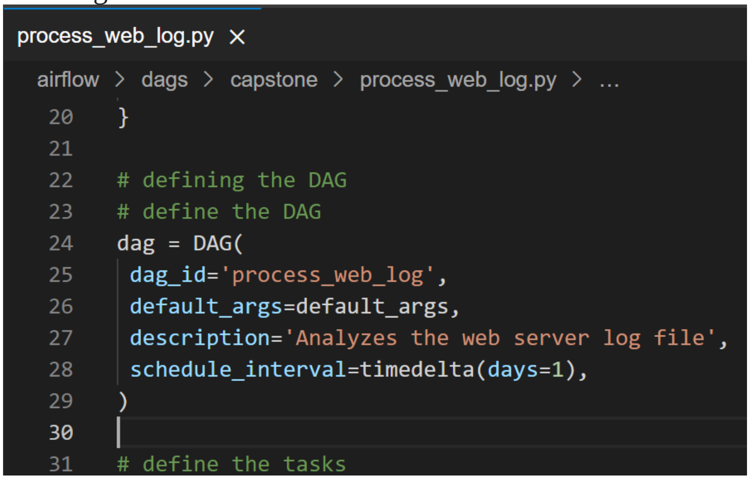
Creating Data Pipelines using Apache Airflow

Defining the DAG arguments.



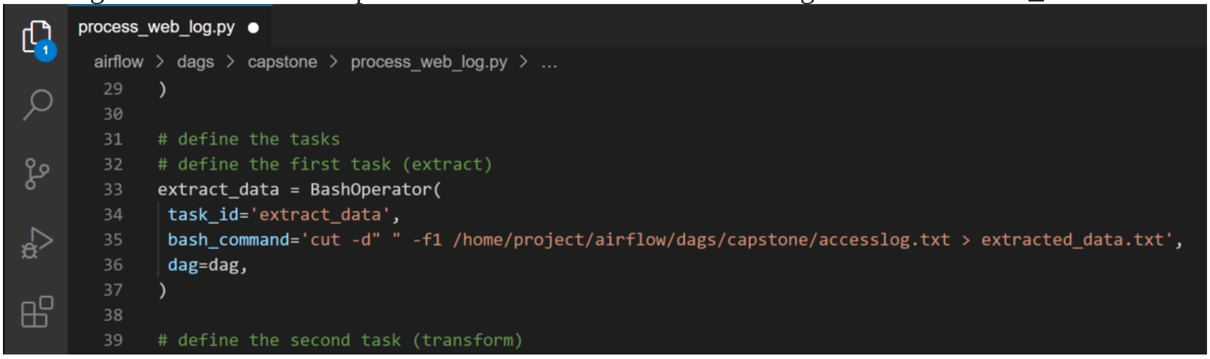
```python
# import the libraries
from datetime import timedelta
# The DAG object; we'll need this to instantiate a DAG
from airflow import DAG
# Operators; we need this to write tasks!
from airflow.operators.bash_operator import BashOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago

#defining DAG arguments
# You can override them on a per-task basis during operator initialization
default_args = {
 'owner': 'Phone Sett Paing',
 'start_date': days_ago(0),
 'email': ['vastersett@gmail.com'],
 'email_on_failure': True,
 'email_on_retry': True,
 'retries': 1,
 'retry_delay': timedelta(minutes=5),
}

# defining the DAG
```

Defining the DAG.



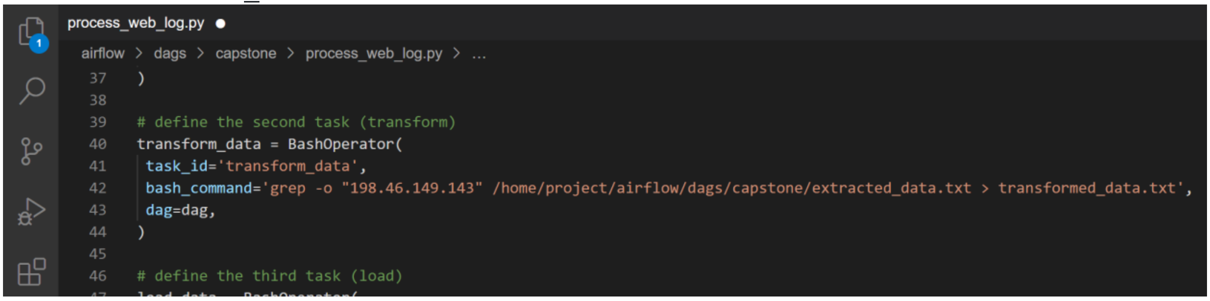
```python
}

# defining the DAG
# define the DAG
dag = DAG(
 dag_id='process_web_log',
 default_args=default_args,
 description='Analyzes the web server log file',
 schedule_interval=timedelta(days=1),
)

# define the tasks
```

Creating a task to extract the ip address field from the web server log file into extracted_data.txt file.



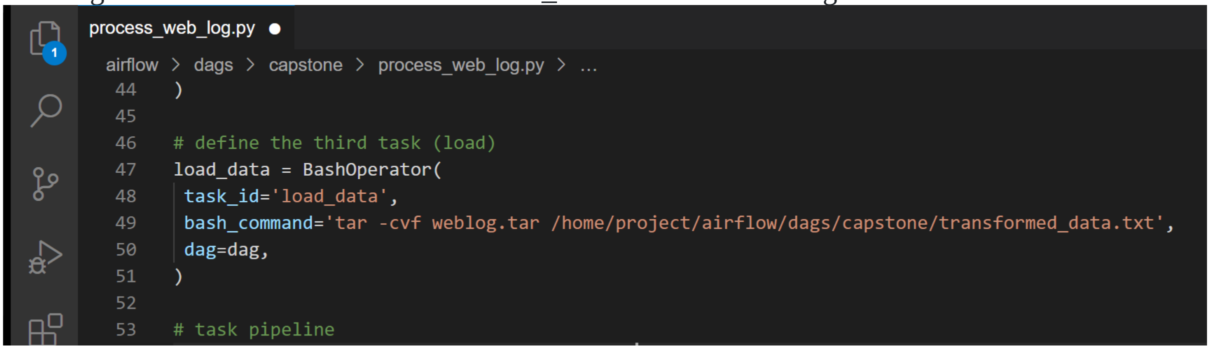
```
)

# define the tasks
# define the first task (extract)
extract_data = BashOperator(
    task_id='extract_data',
    bash_command='cut -d" " -f1 /home/project/airflow/dags/capstone/accesslog.txt > extracted_data.txt',
    dag=dag,
)

# define the second task (transform)
```

Creating a task to filter out all the occurrences of ip address “198.46.149.143” from extracted_data.txt file into transformed_data.txt file.



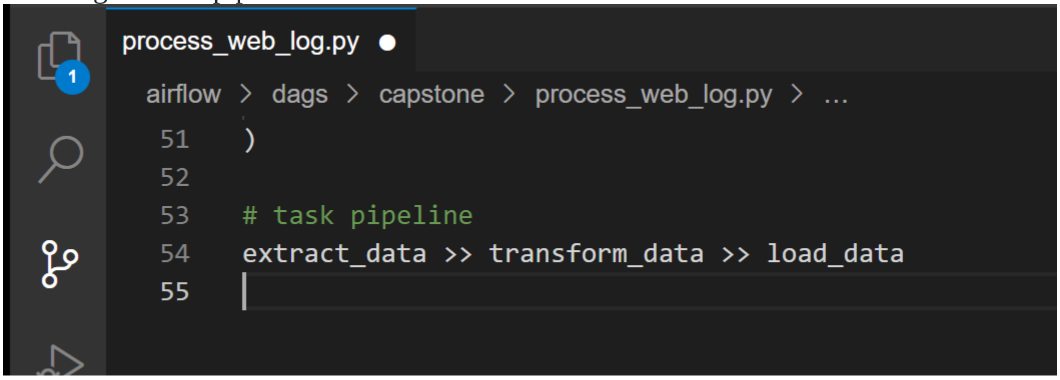
```
)

# define the second task (transform)
transform_data = BashOperator(
    task_id='transform_data',
    bash_command='grep -o "198.46.149.143" /home/project/airflow/dags/capstone/extracted_data.txt > transformed_data.txt',
    dag=dag,
)

# define the third task (load)
load_data = BashOperator(
```

Creating a task to archive the file transformed_data.txt file into weblog.tar file.



```
)

# define the third task (load)
load_data = BashOperator(
    task_id='load_data',
    bash_command='tar -cvf weblog.tar /home/project/airflow/dags/capstone/transformed_data.txt',
    dag=dag,
)

# task pipeline
```

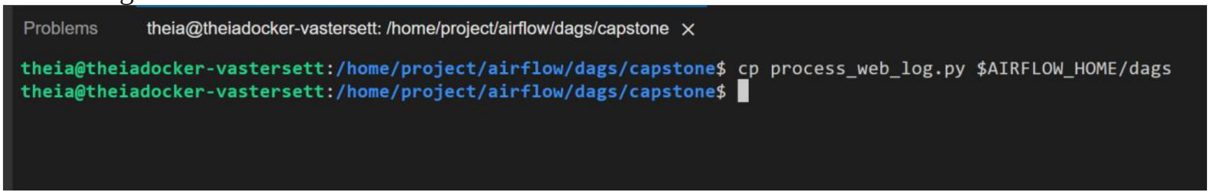
Defining the task pipeline.



```
)

# task pipeline
extract_data >> transform_data >> load_data

```

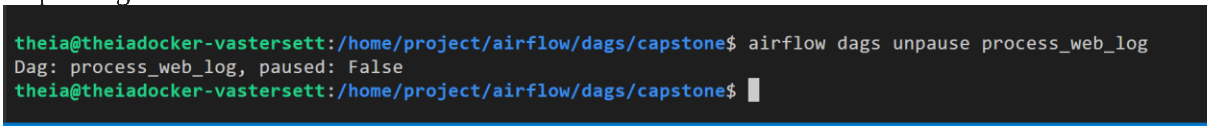
Getting the DAG operational

Submitting the DAG


Problems
theia@theiadocker-vastersett: /home/project/airflow/dags/capstone ✕

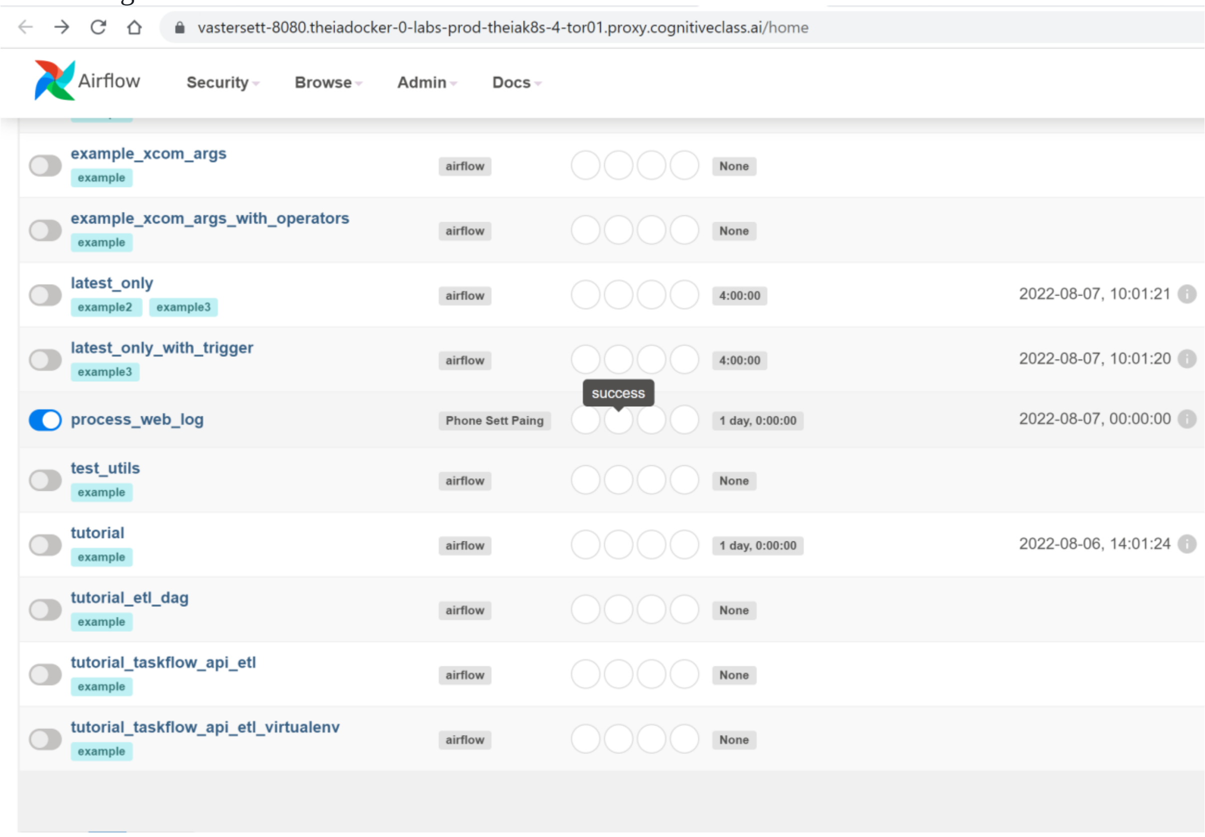
```
theia@theiadocker-vastersett:/home/project/airflow/dags/capstone$ cp process_web_log.py $AIRFLOW_HOME/dags
theia@theiadocker-vastersett:/home/project/airflow/dags/capstone$ 
```

Unpausing the DAG



```
theia@theiadocker-vastersett:/home/project/airflow/dags/capstone$ airflow dags unpause process_web_log
Dag: process_web_log, paused: False
theia@theiadocker-vastersett:/home/project/airflow/dags/capstone$ 
```

Monitoring the DAG


vastersett-8080.theiadocker-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/home
Airflow
Security
Browse
Admin
Docs
success

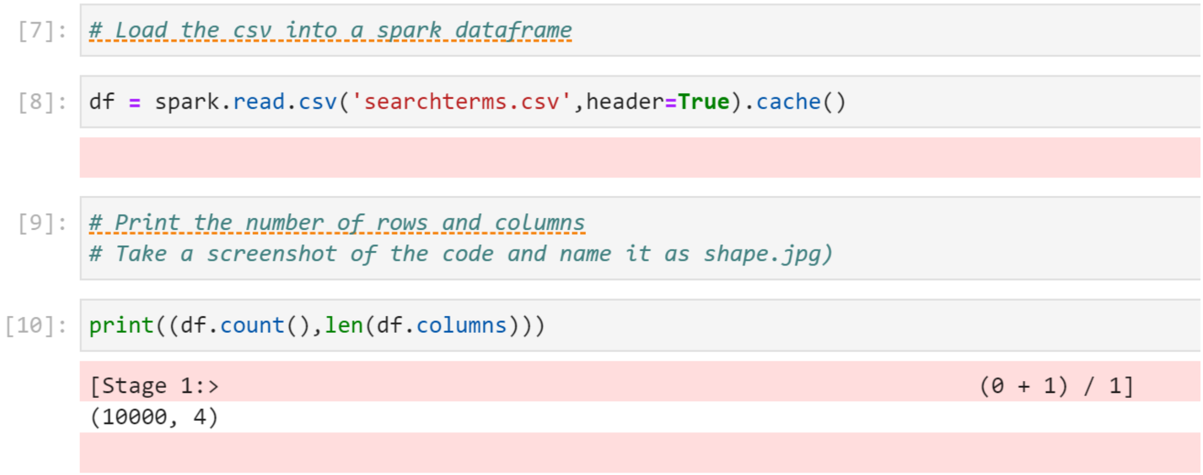
○	example_xcom_args example	airflow	○○○○	None	
○	example_xcom_args_with_operators example	airflow	○○○○	None	
○	latest_only example2 example3	airflow	○○○○	4:00:00	2022-08-07, 10:01:21
○	latest_only_with_trigger example3	airflow	○○○○	4:00:00	2022-08-07, 10:01:20
●	process_web_log	Phone Sett Paing	○○○○	1 day, 0:00:00	2022-08-07, 00:00:00
○	test_utils example	airflow	○○○○	None	
○	tutorial example	airflow	○○○○	1 day, 0:00:00	2022-08-06, 14:01:24
○	tutorial_etl_dag example	airflow	○○○○	None	
○	tutorial_taskflow_api_etl example	airflow	○○○○	None	
○	tutorial_taskflow_api_etl_virtualenv example	airflow	○○○○	None	

Module 6:

Analyse search terms on the e-commerce web server

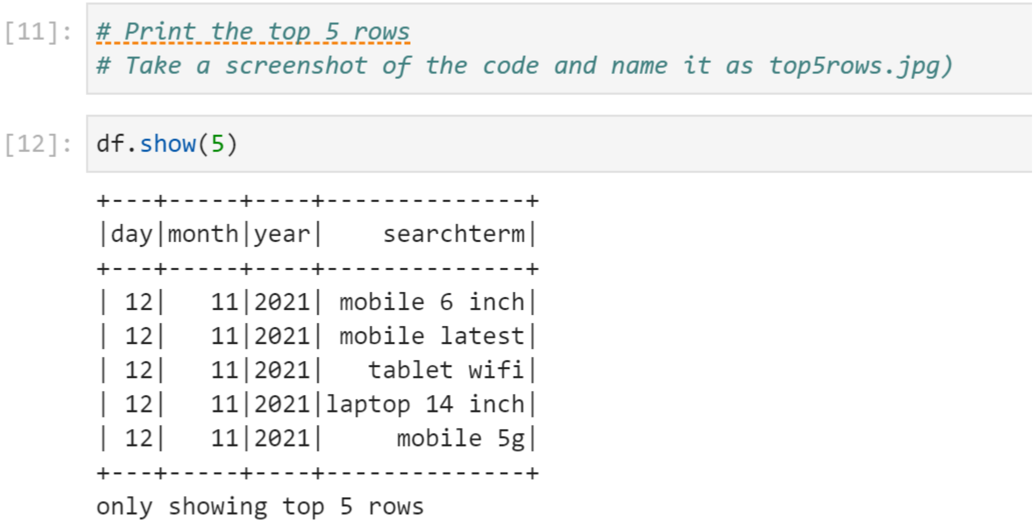
Download the search term data set for the e-commerce web server and analyse the data set, load the sales forecast model and predict the sales for the year 2023.

Printing the number of rows and columns in the data set.



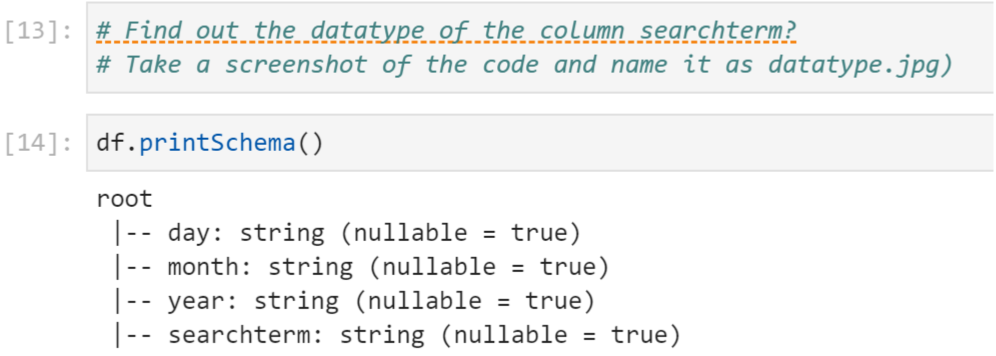
```
[7]:  # Load the csv into a spark dataframe

[8]:  df = spark.read.csv('searchterms.csv',header=True).cache()

[9]:  # Print the number of rows and columns
      # Take a screenshot of the code and name it as shape.jpg)

[10]: print((df.count(),len(df.columns)))

      [Stage 1:>                                                          (0 + 1) / 1]
      (10000, 4)
```

Printing the top 5 rows in the data set.



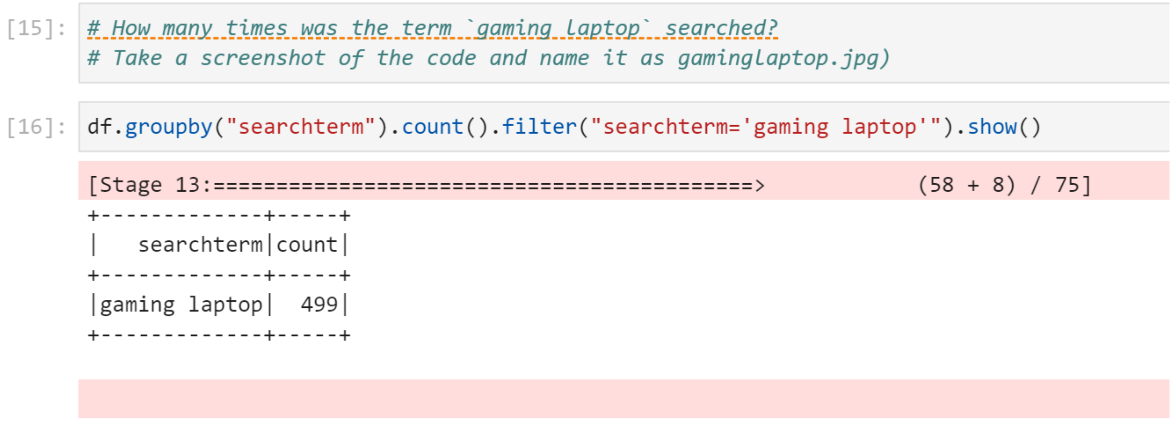
```
[11]: # Print the top 5 rows
      # Take a screenshot of the code and name it as top5rows.jpg)

[12]: df.show(5)

      +---+-----+----+--------------+
      |day|month|year|    searchterm|
      +---+-----+----+--------------+
      | 12|   11|2021| mobile 6 inch|
      | 12|   11|2021| mobile latest|
      | 12|   11|2021|   tablet wifi|
      | 12|   11|2021|laptop 14 inch|
      | 12|   11|2021|     mobile 5g|
      +---+-----+----+--------------+
      only showing top 5 rows
```

Finding the data type of the column “searchterm”.



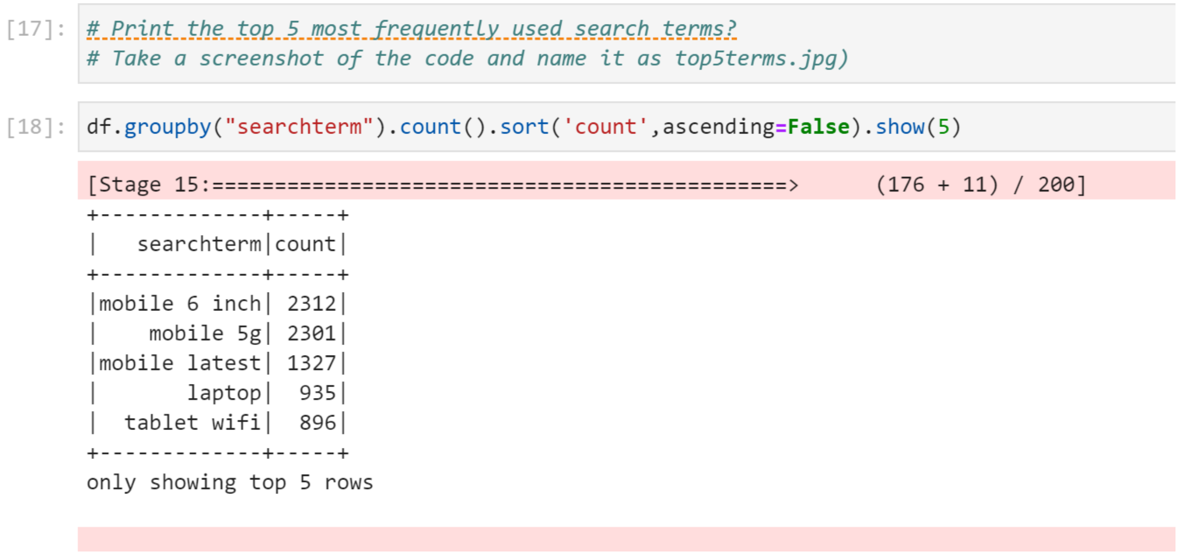
```
[13]: # Find out the datatype of the column searchterm?
      # Take a screenshot of the code and name it as datatype.jpg)

[14]: df.printSchema()

      root
       |-- day: string (nullable = true)
       |-- month: string (nullable = true)
       |-- year: string (nullable = true)
       |-- searchterm: string (nullable = true)
```

Finding the number of times, the term “gaming laptop” was searched.



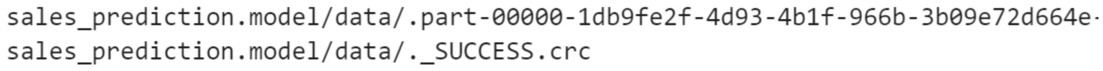
```
[15]: # How many times was the term `gaming laptop` searched?
      # Take a screenshot of the code and name it as gaminglaptop.jpg)

[16]: df.groupby("searchterm").count().filter("searchterm='gaming laptop'").show()

      [Stage 13:==========================================>             (58 + 8) / 75]
      +-------------+-----+
      |   searchterm|count|
      +-------------+-----+
      |gaming laptop|  499|
      +-------------+-----+
```

Printing the top 5 frequently used search terms.

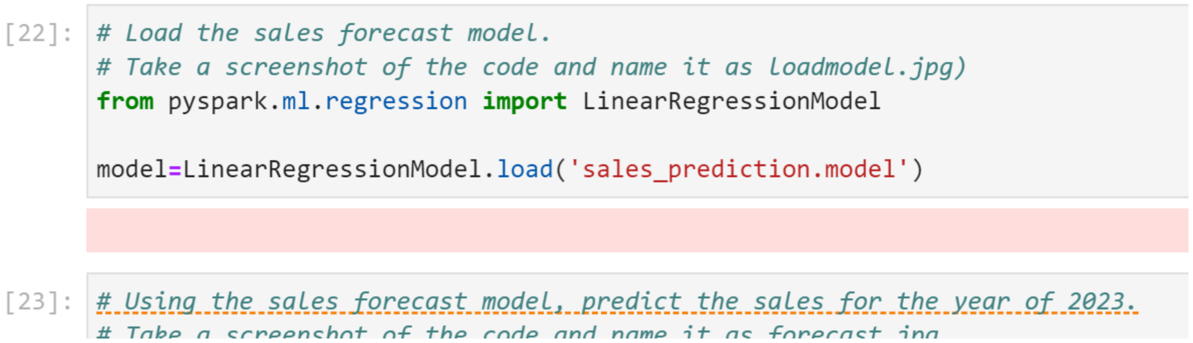


```
[17]: # Print the top 5 most frequently used search terms?
      # Take a screenshot of the code and name it as top5terms.jpg)

[18]: df.groupby("searchterm").count().sort('count',ascending=False).show(5)

      [Stage 15:============================================>      (176 + 11) / 200]
      +-------------+-----+
      |   searchterm|count|
      +-------------+-----+
      |mobile 6 inch| 2312|
      |    mobile 5g| 2301|
      |mobile latest| 1327|
      |       laptop|  935|
      |  tablet wifi|  896|
      +-------------+-----+
      only showing top 5 rows
```

Loading the sales forecast model.

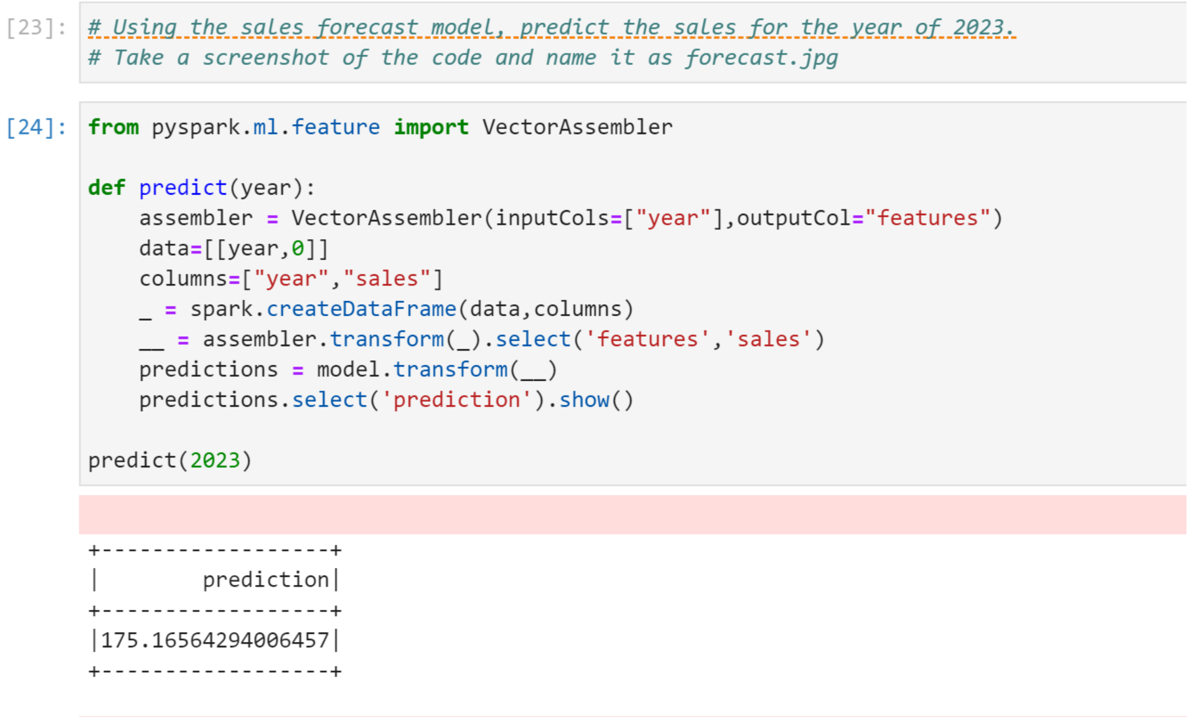


```
sales_prediction.model/data/.part-00000-1db9fe2f-4d93-4b1f-966b-3b09e72d664e-
sales_prediction.model/data/._SUCCESS.crc
```



```
[22]: # Load the sales forecast model.
      # Take a screenshot of the code and name it as loadmodel.jpg)
      from pyspark.ml.regression import LinearRegressionModel

      model=LinearRegressionModel.load('sales_prediction.model')

[23]: # Using the sales forecast model, predict the sales for the year of 2023.
      # Take a screenshot of the code and name it as forecast.jpg
```

Predicting the sales for the year 2023, using the sales forecast model.



```
[23]: # Using the sales forecast model, predict the sales for the year of 2023.
      # Take a screenshot of the code and name it as forecast.jpg
```

```
[24]: from pyspark.ml.feature import VectorAssembler

      def predict(year):
          assembler = VectorAssembler(inputCols=["year"],outputCol="features")
          data=[[year,0]]
          columns=["year","sales"]
          _ = spark.createDataFrame(data,columns)
          __ = assembler.transform(_).select('features','sales')
          predictions = model.transform(__)
          predictions.select('prediction').show()

      predict(2023)
```

```
+------------------+
|        prediction|
+------------------+
|175.16564294006457|
+------------------+
```