SQL MasterClass (II)

Barcelona Data Institute

Iago Novoa
Head of BI at letgo

1. Quick Review
2. How to create a Database
3. Homework
4. How to Export and Import data in Redshift
5. QA

SQL MasterClass

Barcelona Data Institute

1. Quick Review
2. How to create a Database
3. Homework
4. How to Export and Import data in Redshift
5. QA

SQL MasterClass

Barcelona Data Institute

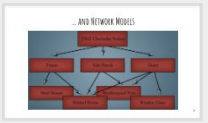
1

2

3

4

5



6

7

8

9

10

11

12


13

14


15

16

17

18

19

20


21

22

23

24

25

26


27

28

29


30

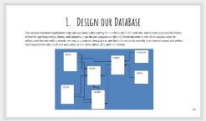
31


32

33

34

35



36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

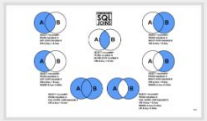
54

55

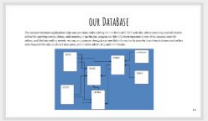
56

57

58



59



60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

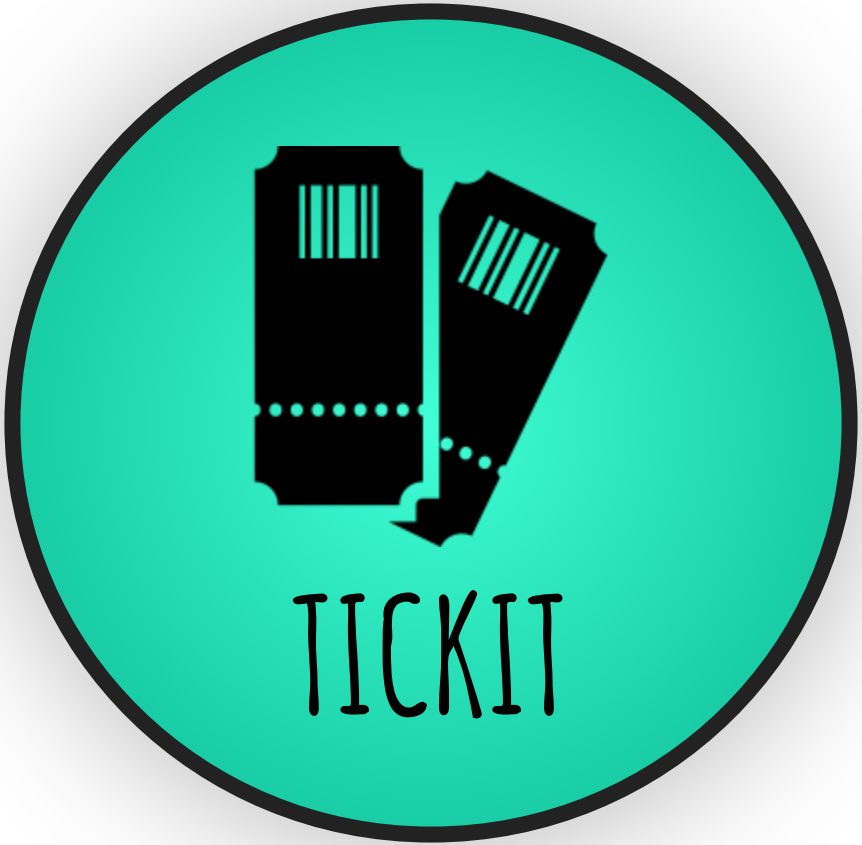
80


I don't understand at all
I need to go over this again
I think I got it, but am not completely comfortable
I got it
I can explain it to someone else

1. Quick Review
2. **How to create a Database**
3. Homework
4. How to Export and Import data in Redshift
5. QA

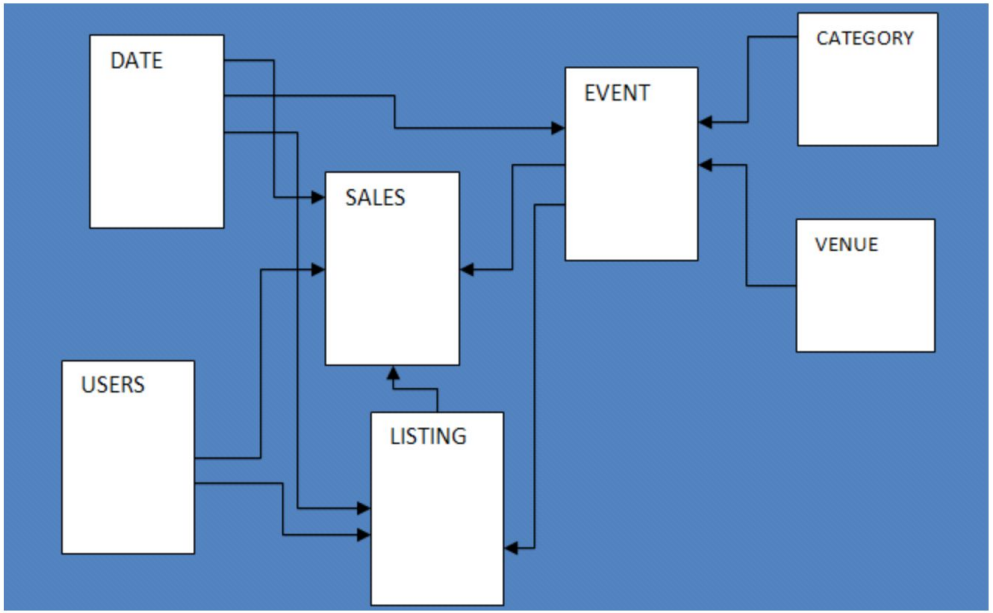
SQL MasterClass

Barcelona Data Institute


TICKIT

1. Design our DataBase

This sample database application helps analysts track sales activity for the fictional TICKIT web site, where users buy and sell tickets online for sporting events, shows, and concerts. In particular, analysts can identify ticket movement over time, success rates for sellers, and the best-selling events, venues, and seasons. Analysts can use this information to provide incentives to buyers and sellers who frequent the site, to attract new users, and to drive advertising and promotions.



2. Define Tables

CATEGORY Table

Column Name	Data Type	Description
CATID	SMALLINT	Primary key, a unique ID value for each row. Each row represents a specific type of event for which tickets are bought and sold.
CATGROUP	VARCHAR(10)	Descriptive name for a group of events, such as **Shows** and **Sports**.
CATNAME	VARCHAR(10)	Short descriptive name for a type of event within a group, such as **Opera** and **Musicals**.
CATDESC	VARCHAR(30)	Longer descriptive name for the type of event, such as **Musical theatre**.

EVENT Table

Column Name	Data Type	Description
EVENTID	INTEGER	Primary key, a unique ID value for each row. Each row represents a separate event that takes place at a specific venue at a specific time.
VENUEID	SMALLINT	Foreign-key reference to the VENUE table.
CATID	SMALLINT	Foreign-key reference to the CATEGORY table.
DATEID	SMALLINT	Foreign-key reference to the DATE table.
EVENTNAME	VARCHAR(200)	Name of the event, such as **Hamlet** or **La Traviata**.
STARTTIME	TIMESTAMP	Full date and start time for the event, such as **2008-10-10 19:30:00**.

SALES Table

Column Name	Data Type	Description
SALESID	INTEGER	Primary key, a unique ID value for each row. Each row represents a sale of one or more tickets for a specific event, as offered in a specific listing.
LISTID	INTEGER	Foreign-key reference to the LISTING table.
SELLERID	INTEGER	Foreign-key reference to the USERS table (the user who sold the tickets).
BUYERID	INTEGER	Foreign-key reference to the USERS table (the user who bought the tickets).
EVENTID	INTEGER	Foreign-key reference to the EVENT table.
DATEID	SMALLINT	Foreign-key reference to the DATE table.
QTYSOLD	SMALLINT	The number of tickets that were sold, from **1** to **8**. (A maximum of 8 tickets can be sold in a single transaction.)
PRICEPAID	DECIMAL(8,2)	The total price paid for the tickets, such as **75.00** or **488.00**. The individual price of a ticket is PRICEPAID/QTYSOLD.
COMMISSION	DECIMAL(8,2)	The 15% commission that the business collects from the sale, such as **11.25** or **73.20**. The seller receives 85% of the PRICEPAID value.
SALETIME	TIMESTAMP	The full date and time when the sale was completed, such as **2008-05-24 06:21:47**.

VENUE Table

Column Name	Data Type	Description
VENUEID	SMALLINT	Primary key, a unique ID value for each row. Each row represents a specific venue where events take place.
VENUENAME	VARCHAR(100)	Exact name of the venue, such as **Cleveland Browns Stadium**.
VENUECITY	VARCHAR(30)	City name, such as **Cleveland**.
VENUESTATE	CHAR(2)	Two-letter state or province abbreviation (United States and Canada), such as **OH**.
VENUESEATS	INTEGER	Maximum number of seats available at the venue, if known, such as **73200**. For demonstration purposes, this column contains some null values and zeroes.

DATE Table

Column Name	Data Type	Description
DATEID	SMALLINT	Primary key, a unique ID value for each row. Each row represents a day in the calendar year.
CALDATE	DATE	Calendar date, such as **2008-06-24**.
DAY	CHAR(3)	Day of week (short form), such as **SA**.
WEEK	SMALLINT	Week number, such as **26**.
MONTH	CHAR(5)	Month name (short form), such as **JUN**.
QTR	CHAR(5)	Quarter number (**1** through **4**).
YEAR	SMALLINT	Four-digit year (**2008**).
HOLIDAY	BOOLEAN	Flag that denotes whether the day is a public holiday (U.S.).

LISTING Table

Column Name	Data Type	Description
LISTID	INTEGER	Primary key, a unique ID value for each row. Each row represents a listing of a batch of tickets for a specific event.
SELLERID	INTEGER	Foreign-key reference to the USERS table, identifying the user who is selling the tickets.
EVENTID	INTEGER	Foreign-key reference to the EVENT table.
DATEID	SMALLINT	Foreign-key reference to the DATE table.
NUMTICKETS	SMALLINT	The number of tickets available for sale, such as **2** or **20**.
PRICEPERTICKET	DECIMAL(8,2)	The fixed price of an individual ticket, such as **27.00** or **206.00**.
TOTALPRICE	DECIMAL(8,2)	The total price for this listing (NUMTICKETS*PRICEPERTICKET).
LISTTIME	TIMESTAMP	The full date and time when the listing was posted, such as **2008-03-18 07:19:35**.

USERS Table

Column Name	Data Type	Description
USERID	INTEGER	Primary key, a unique ID value for each row. Each row represents a registered user (a buyer or seller or both) who has listed or bought tickets for at least one event.
USERNAME	CHAR(8)	An 8-character alphanumeric username, such as **PGL08LJI**.
FIRSTNAME	VARCHAR(30)	The user's first name, such as **Victor**.
LASTNAME	VARCHAR(30)	The user's last name, such as **Hernandez**.
CITY	VARCHAR(30)	The user's home city, such as **Naperville**.
STATE	CHAR(2)	The user's home state, such as **GA**.
EMAIL	VARCHAR(100)	The user's email address; this column contains random Latin values, such as **turpis@accumsanlaoreet.org**.
PHONE	CHAR(14)	The user's 14-character phone number, such as **(818) 765-4255**.
LIKESPORTS, ...	BOOLEAN	A series of 10 different columns that identify the user's likes and dislikes with **true** and **false** values.

3. Create Tables

```
create table users(
        userid integer not null distkey sortkey,
        username char(8),
        firstname varchar(30),
        lastname varchar(30),
        city varchar(30),
        state char(2),
        email varchar(100),
        phone char(14),
        likesports boolean,
        liketheatre boolean,
        likeconcerts boolean,
        likejazz boolean,
        likeclassical boolean,
        likeopera boolean,
        likerock boolean,
        likevegas boolean,
        likebroadway boolean,
        likemusicals boolean);

create table venue(
        venueid smallint not null distkey sortkey,
        venuename varchar(100),
        venuecity varchar(30),
        venuestate char(2),
```

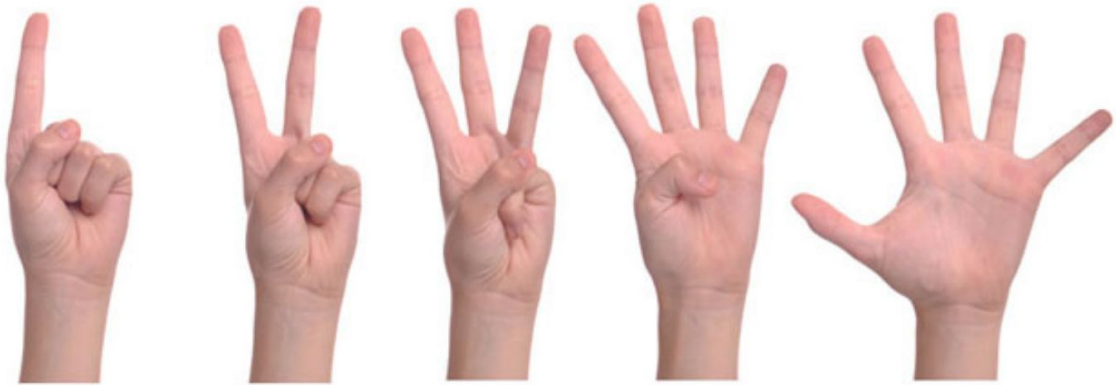
4. Load data

```
copy users from 's3://awssampledbuswest2/tickit/allusers_pipe.txt'
credentials 'aws_iam_role=<iam-role-arn>'
delimiter '|' region 'us-west-2';

copy venue from 's3://awssampledbuswest2/tickit/venue_pipe.txt'
credentials 'aws_iam_role=<iam-role-arn>'
delimiter '|' region 'us-west-2';

copy category from 's3://awssampledbuswest2/tickit/category_pipe.txt'
credentials 'aws_iam_role=<iam-role-arn>'
delimiter '|' region 'us-west-2';

copy date from 's3://awssampledbuswest2/tickit/date2008_pipe.txt'
credentials 'aws_iam_role=<iam-role-arn>'
delimiter '|' region 'us-west-2';

copy event from 's3://awssampledbuswest2/tickit/allevents_pipe.txt'
credentials 'aws_iam_role=<iam-role-arn>'
delimiter '|' timeformat 'YYYY-MM-DD HH:MI:SS' region 'us-west-2';

copy listing from 's3://awssampledbuswest2/tickit/listings_pipe.txt'
credentials 'aws_iam_role=<iam-role-arn>'
delimiter '|' region 'us-west-2';

copy sales from 's3://awssampledbuswest2/tickit/sales_tab.txt'
credentials 'aws_iam_role=<iam-role-arn>'
delimiter '\t' timeformat 'MM/DD/YYYY HH:MI:SS' region 'us-west-2';
```

Now we are ready to start querying...

select * from sql_masterclass.sales;

select * from sql_masterclass.sales limit 10;


I don't understand at all
I need to go over this again
I think I got it, but am not completely comfortable
I got it
I can explain it to someone else

1. Quick Review
2. How to create a Database
3. **Homework**
4. How to Export and Import data in Redshift
5. QA

SQL MasterClass

Barcelona Data Institute

Homework 1

Task:

- Get first name, last name and initials (first letter of the first name and first letter of last name) from 10 users
- Save the query as homework1.sql in GitHub in sql_masterclass folder

Tips:

- Use String functions

	firstname	lastname	initials
1	Lars	Ratliff	LR
2	Colton	Roy	CR
3	Bruce	Beck	BB
4	Henry	Cochran	HC
5	Cody	Moss	CM
6	Ralph	Bird	RB
7	Wing	Jennings	WJ
8	Guy	Cochran	GC
9	Emerald	Chan	EC
10	Gloria	Rodriguez	GR

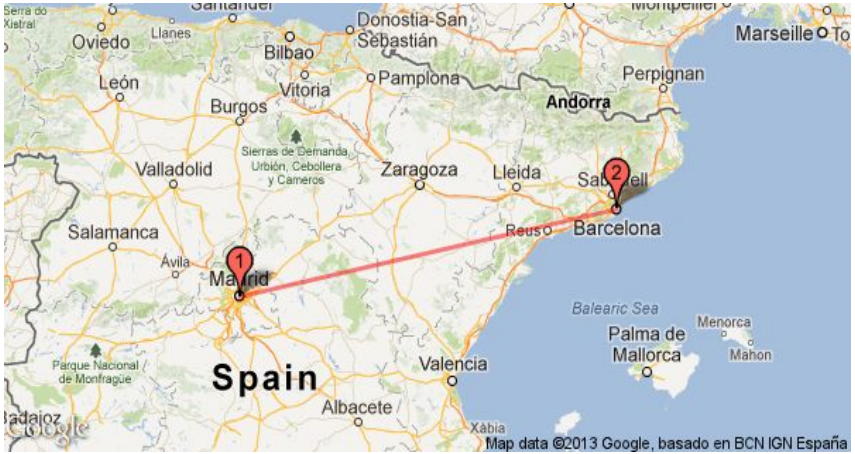
Solution Homework 1

```sql
select
    firstname,
    lastname,
    left(firstname, 1) || left(lastname, 1) as initials
from
    sql_masterclass.users
limit 10;
```

Homework 2

Task:

- Calculate distance in km between Madrid (40.4168 N, 3.7038 W) and Barcelona (41.3851 N, 2.1734 E)
- Save the query as homework2.sql in GitHub in sql_masterclass folder



Tips:

- Look for a math formula to calculate distance between 2 points

Solution Homework 2

```
select
    ACOS(SIN(PI()*40.4168/180.0)*SIN(PI()*41.3851/180.0)+
    COS(PI()*40.4168/180.0)*COS(PI()*41.3851/180.0)*
    COS(PI()*(-2.1734)/180.0-PI()*3.7038/180.0))*6371 as distance;
```

Homework 3

Task:

- Get total number of unsold tickets per venue in Ohio state
- Save the query as homework3.sql in GitHub in sql_masterclass folder

Tip:

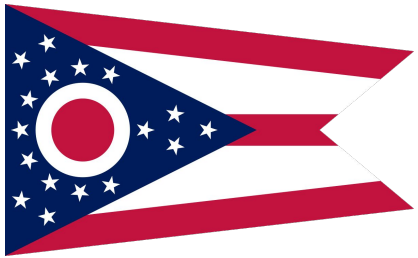
- Use subqueries to avoid duplication of rows

```
select
    eventname
from
    sql_masterclass.event
where
    venueid IN (select distinct venueid from sql_masterclass.venue where venuestate='OH');
```

Result

select eventname from sql_masterclass.ev | Enter a SQL expression to filter results (use Ctrl+Space)

	eventname
1	Smashing Pumpkins
2	Gogol Bordello
3	Carrie Underwood
4	Gavin DeGraw
5	Billy Joel



Solution Homework 3

```
select
  v.venuename,
  sum(l.numtickets) - sum(s.qtysold) as unsold_tickets
from
  sql_masterclass.venue v
inner join
  sql_masterclass.event e on v.venueid=e.venueid
inner join
  (select eventid, sum(numtickets) as numtickets from sql_masterclass.listing group by 1) l on l.eventid=e.eventid
left join
  (select eventid, sum(qtysold) as qtysold from sql_masterclass.sales group by 1) s on s.eventid=e.eventid
where
  v.venuestate='OH'
group by 1;
```

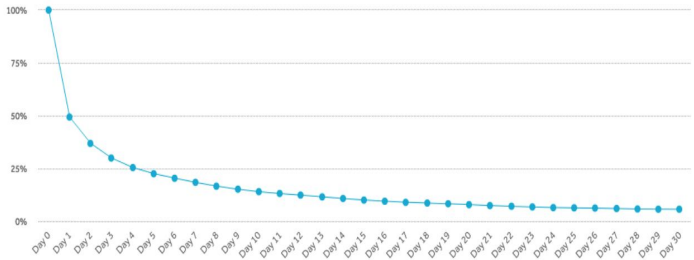
Solution Homework 3

```
with tickets_per_event as
    (select eventid, sum(numtickets) as numtickets from sql_masterclass.listing group by 1),
sold_per_event as
    (select eventid, sum(qtysold) as qtysold from sql_masterclass.sales group by 1)
select
  v.venuename,
  sum(l.numtickets) - sum(s.qtysold) as unsold_tickets
from
  sql_masterclass.venue v
inner join
  sql_masterclass.event e on v.venueid=e.venueid
inner join
  tickets_per_event l on l.eventid=e.eventid
left join
  sold_per_event s on s.eventid=e.eventid
where
  v.venuestate='OH'
group by 1;
```

Homework 4

Task:

- Calculate 30 days Daily Retention for those who transact for the first time in the first half of 2017 in tangarana.payments_transactions
- Transaction Retention measures how often users come back to transact after the first transaction
- Save the query as homework4.sql in GitHub in sql_masterclass folder



Tips:

- Get first users that transact for the first time in the first half of 2017
- Out of them, calculate the volume of those users that transact on day0, day2, day3, ... as shown on screenshot on the right
- Use the function datediff to calculate the difference between dates

datediff	count
0	14,925
1	23
2	11
3	24
4	14
5	15
6	13
7	15
8	20
9	13
10	14

Solution Homework 4

```
with transactions as
(select
  user_id,
  transaction_timestamp,
  first_value(transaction_timestamp)
    over (partition by user_id order by transaction_timestamp rows between unbounded preceding and unbounded following) as first_transaction_timestamp
from
  tangarana.payment_transactions
where
  transaction_type='payment')
select
  datediff(day, first_transaction_timestamp, transaction_timestamp) as datediff,
  count(distinct user_id)
from
  transactions
where
  datediff(day, first_transaction_timestamp, transaction_timestamp)<=30 and
  first_transaction_timestamp>='2017-01-01' and
  first_transaction_timestamp<'2017-07-01'
group by 1;
```

Homework 5

Task:

- We have data from visits and purchases from an ecommerce website in the following format:
  - "sql_masterclass.transactions"
  - "sql_masterclass.visits"
- Assign a visit to every transaction
- Save the query as homework5.sql in GitHub in sql_masterclass folder

Tips:

- Use lead or lag window functions to add a calculated column in one of the tables
- Learn how to use the function NVL in Redshift

```
select * from sql_masterclass.transactions limit 5;
```

Result

select * from sql_masterclass.transac | Enter a SQL expression to

	transaction_id	user_id	transaction_time
1	8,788,759	672,040	2016-10-03 13:52:45
2	8,708,891	649,612	2016-10-03 17:29:26
3	8,722,370	652,175	2016-10-04 04:34:10
4	8,621,989	985,347	2016-10-04 10:43:12
5	8,712,478	1,289,767	2016-10-05 09:35:57

```
select * from sql_masterclass.visits limit 5;
```

Result

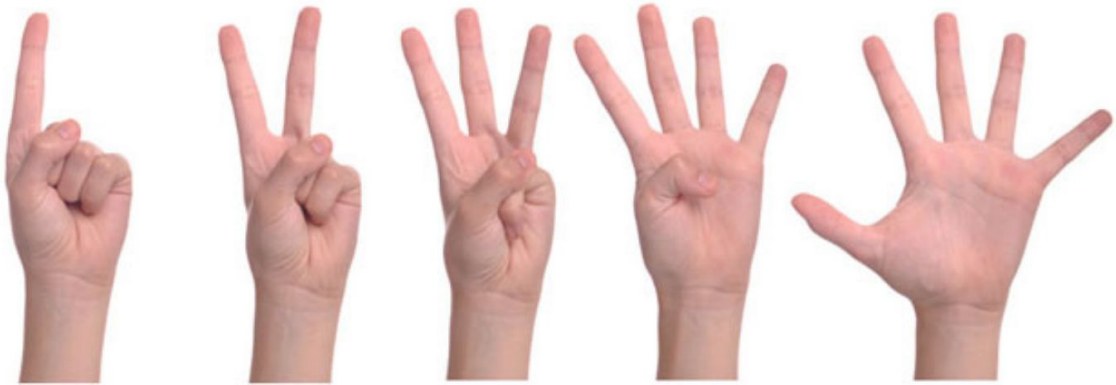
select * from sql_masterclass.visits lir | Enter a SQL expres

	visit_id	user_id	visit_time
1	9,994	672,040	2016-10-03 13:28:45
2	432	1,289,767	2016-10-05 09:11:57
3	5,184	665,882	2016-10-10 20:20:17
4	8,053	1,064,435	2016-10-14 09:38:14
5	9,020	666,534	2016-10-17 11:23:00

	user_id	transaction_id	visit_id	transaction_time	visit_time
1	652,910	8,727,907	7,949	2016-10-06 01:33:50	2016-10-06 01:09:50
2	657,037	8,564,577	6,402	2016-10-08 12:59:28	2016-10-08 12:35:28
3	657,037	8,767,214	7,357	2016-12-24 15:01:03	2016-12-02 22:37:03
4	665,882	8,531,769	5,184	2016-10-10 20:44:17	2016-10-10 20:20:17
5	666,302	8,506,137	6,263	2016-10-05 18:48:17	2016-10-05 18:24:17
6	666,534	8,700,906	9,020	2016-10-17 11:47:00	2016-10-17 11:23:00
7	669,008	8,563,949	1,996	2016-10-17 00:33:24	2016-10-15 09:49:24
8	671,735	8,520,433	6,435	2016-12-09 05:06:32	2016-12-07 14:22:32
9	671,998	8,710,930	7,796	2016-10-23 08:26:36	2016-10-23 08:02:36
10	672,640	8,481,493	2,959	2016-10-21 03:18:29	2016-10-19 12:34:29

Solution Homework 5

```
with visits_enriched as
  (select
    visit_id,
    user_id,
    visit_time,
    nvl(lead(visit_time) over (partition by user_id order by visit_time), '2099-01-01') as next_visit_time
   from
     sql_masterclass.visits)
select
  t.user_id,
  t.transaction_id,
  v.visit_id,
  t.transaction_time,
  v.visit_time
from
  sql_masterclass.transactions t
left join
  visits_enriched v on t.user_id=v.user_id and t.transaction_time>=v.visit_time and t.transaction_time<v.next_visit_time;
```


I don't understand at all
I need to go over this again
I think I got it, but am not completely comfortable
I got it
I can explain it to someone else

1. Quick Review
2. How to create a Database
3. Homework
4. **How to Export and Import data in Redshift**
5. QA

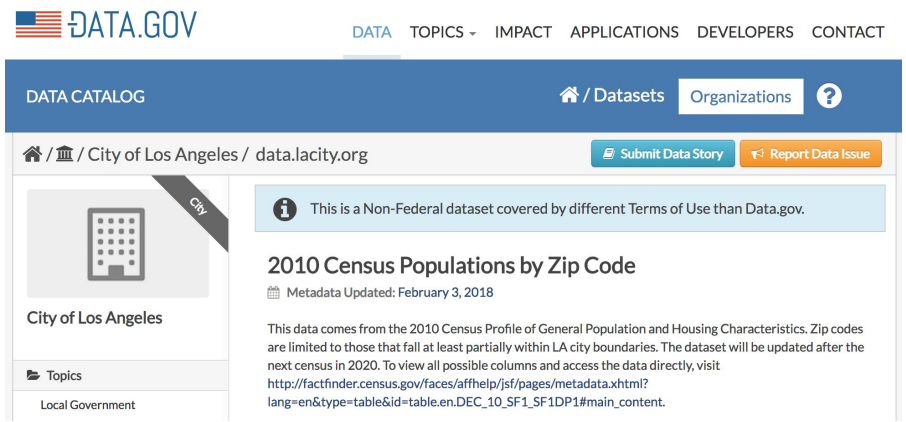
SQL MasterClass

Barcelona Data Institute

Loading Data in Redshift

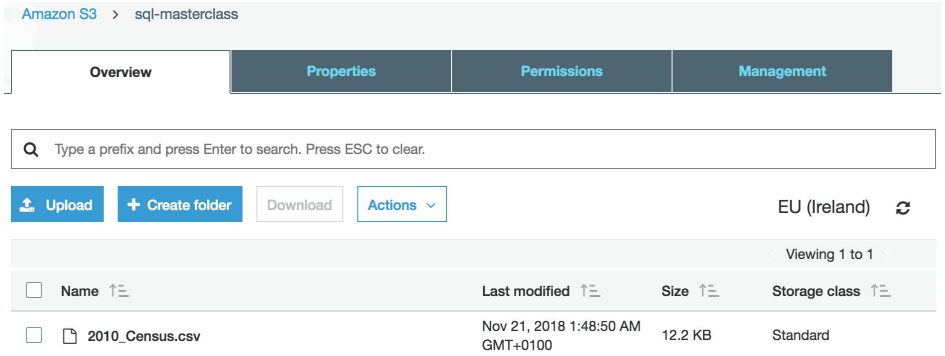
1. Get the data you want to load
2. Upload data file into AWS S3
3. Create table in RedshifT to store that data
4. Load data using copy command

https://catalog.data.gov/dataset/2010-census-populations-by-zip-code



Loading Data in Redshift (II)

1. Get the data you want to load
2. **Upload data file into AWS S3**
3. Create table in RedshifT to store that data
4. Load data using COPY command



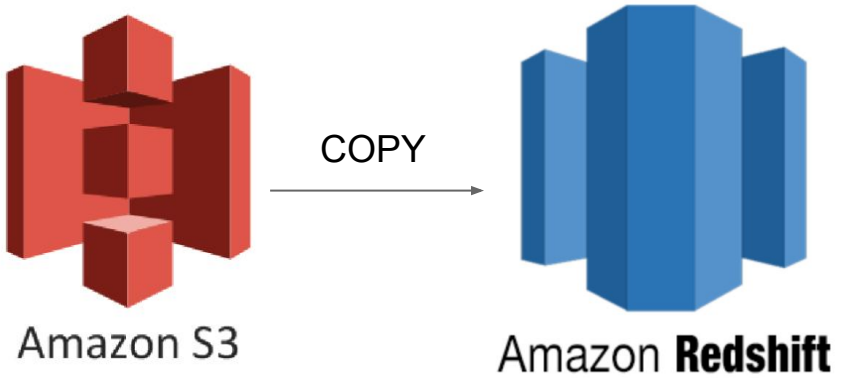
Loading Data in Redshift (III)

1. Get the data you want to load
2. Upload data file into AWS S3
3. **Create table in Redshift to store that data**
4. Load data using COPY command


```
create table sql_masterclass.census(
    zip_code integer not null distkey sortkey,
    total_population integer,
    median_age float,
    total_males integer,
    total_females integer,
    total_households integer,
    average_household_size float);
```

Loading Data in Redshift (IV)

1. Get the data you want to load
2. Upload data file into AWS S3
3. Create table in RedshifT to store that data
4. Load data using copy command

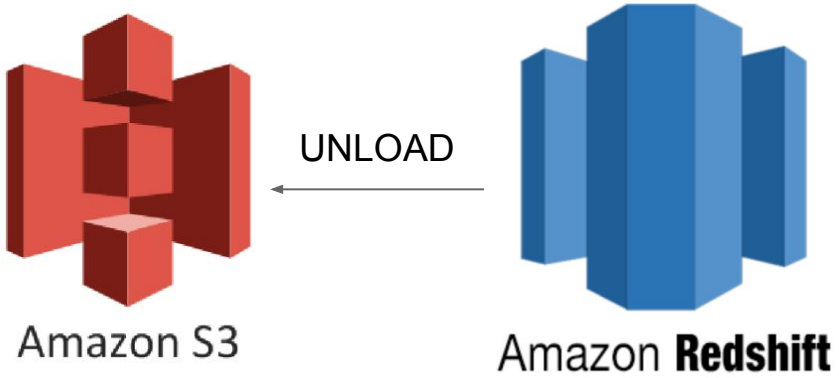



```
-- copy with IAM role

    copy sql_masterclass.census from 's3://data.public.bdatainstitute.com/sql_masterclass/2010_Census_Populations_by_Zip_Code.csv'
    iam_role 'arn:aws:iam::XXX'
    delimiter ',' IGNOREHEADER 1;

-- copy with Access Keys

    copy sql_masterclass.census from 's3://data.public.bdatainstitute.com/sql_masterclass/2010_Census_Populations_by_Zip_Code.csv'
    credentials 'aws_access_key_id=XXX;aws_secret_access_key=XXX'
    delimiter ',' IGNOREHEADER 1;
```

Unloading Data from Redshift

1. Save query results into S3 by using unload command

```
-- unload with IAM role

unload ('select
    sum(total_population) as total_population,
    sum(total_males) as total_males,
    sum(total_females) as total_females
from
    sql_masterclass.census')
to 's3://data.public.bdatainstitute.com/sql_masterclass/20181130_1_unload_census'
iam_role 'arn:aws:iam::XXX'
HEADER
DELIMITER as ','
PARALLEL OFF;

-- unload with Access Keys

unload ('select
    sum(total_population) as total_population,
    sum(total_males) as total_males,
    sum(total_females) as total_females
from
    sql_masterclass.census')
to 's3://data.public.bdatainstitute.com/sql_masterclass/20181130_2_unload_census'
credentials 'aws_access_key_id=XXX;aws_secret_access_key=XXX'
HEADER
DELIMITER as ','
PARALLEL OFF;
```



Loading Data ...

Load spend data from https://s3-eu-west-1.amazonaws.com/sql-masterclass/20181127_daily_spend.csv into "sql_masterclass.daily_spend"

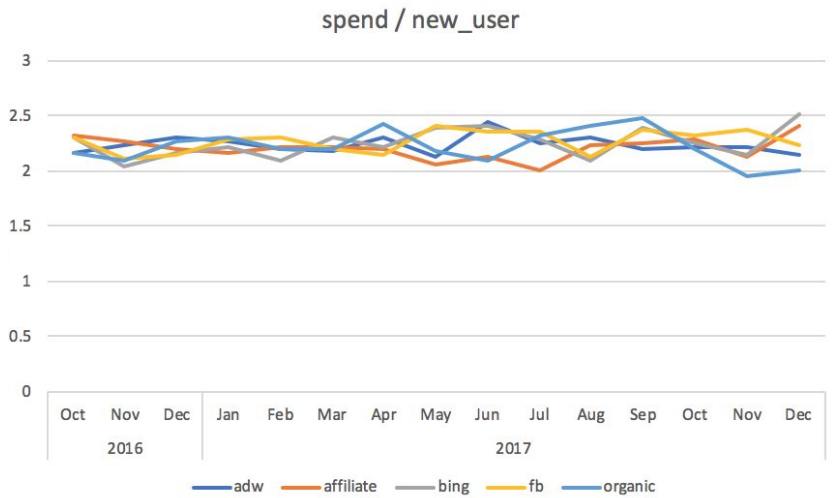
```
drop table if exists sql_masterclass.daily_spend;

create table sql_masterclass.daily_spend(
    date TIMESTAMP not null distkey sortkey,
    utm_medium varchar(100),
    utm_source varchar(100),
    spend float);

select * from sql_masterclass.daily_spend limit 10;

copy sql_masterclass.daily_spend from 's3://data.public.bdatainstitute.com/sql_masterclass/20181127_daily_spend.csv'
iam_role 'arn:aws:iam::XXX'
delimiter ',' IGNOREHEADER 1;
```

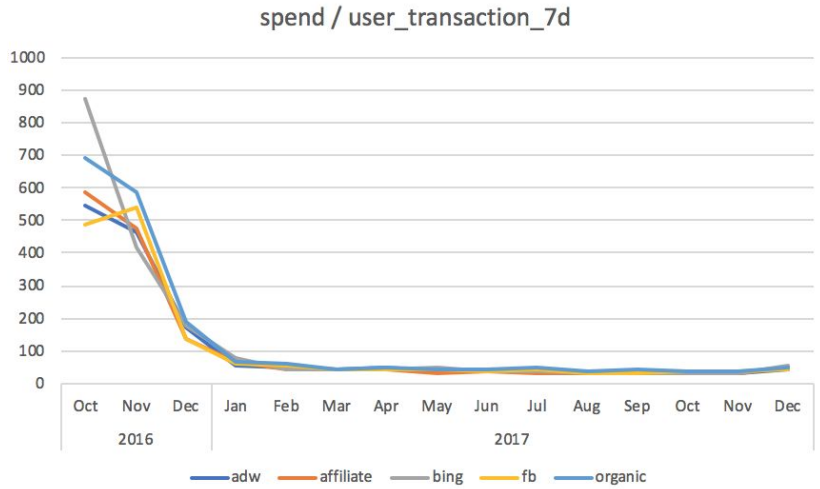
Last Exercise

calculate monthly spend/new_user and spend/user_with_transaction_in_7_days medium and source

Final Exercise

Using tables "sql_masterclass.daily_spend", "tangarana.users" and "tangarana.payment_transactions" calculate daily spend/new_user and spend/user_with_transaction_in_7_days by medium and source

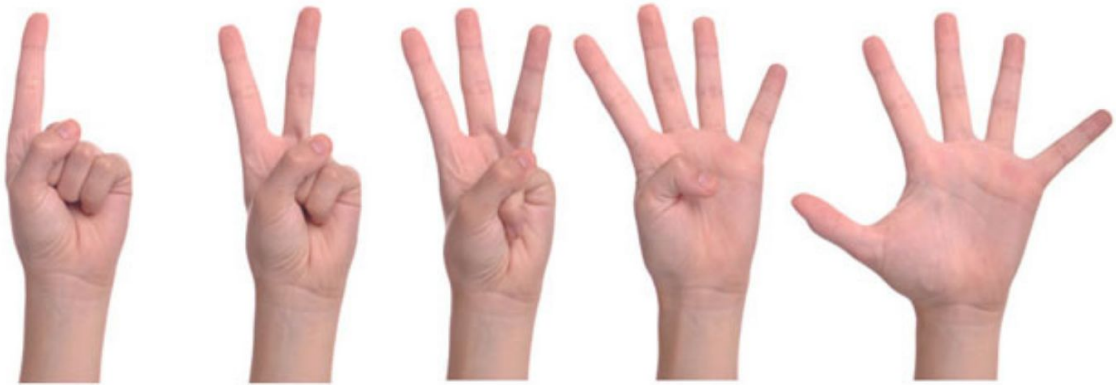




Final Exercise (II)

Using tables "sql_masterclass.daily_spend", "tangarana.users" and "tangarana.payment_transactions" calculate monthly spend/new_user and spend/user_with_transaction_in_7_days by medium and source

```
with new_users_and_users_transaction_7d as
(select
    date(u.user_acquisition_timestamp) as date,
    u.utm_medium,
    u.utm_source,
    count(distinct u.user_id) as new_users,
    count(distinct CASE WHEN datediff (days,u.user_acquisition_timestamp,p.transaction_timestamp) <= 6 THEN p.user_id END) as users_transaction_7d
from
    tangarana.users u
left join
    tangarana.payment_transactions p on p.user_id=u.user_id
group by 1, 2, 3)
select
    s.date,
    s.utm_medium,
    s.utm_source,
    s.spend,
    u.new_users,
    users_transaction_7d
from
    sql_masterclass.daily_spend s
left join
    new_users_and_users_transaction_7d u
on
    s.date=u.date and s.utm_medium=u.utm_medium and s.utm_source=u.utm_source;
```


I don't understand at all
I need to go over this again
I think I got it, but am not completely comfortable
I got it
I can explain it to someone else

1. Quick Review
2. How to create a Database
3. Homework
4. How to Export and Import data in Redshift
5. QA

SQL MasterClass

Barcelona Data Institute


YOUR
FEEDBACK
MATTERS!

Deepen Learning

- With your super pen and in one post-it, write (2 min):
  - Your 2 main learnings/takeaways
  - Your 2 main challenges for the week
- Find a partner that is not your table mate
- Share and help/support each other (3 min)
- Find a new partner, repeat (3 min)
  - Look at what is different, after the first share

Data Analyst Progress

12%

THANK YOU

SQL MasterClass

Barcelona Data Institute