# Live from New York...
# It's a Saturday Night Live Database

Designed by Jenna Ficula

# Table of Contents:

# Executive Summary

This project depicts the design and functionality of a database created for Saturday Night Live. Since its premiere in 1975, there have been 42 seasons of SNL with upwards of 800 episodes and thousands of skits. Therefore, this database has been compiled contains only certain instances of the show which could scale to a larger implementation the database. The general public are the assumed users, but more specifically, the staff and members of the show.

The goal of this database is to provide snl staff and enthusists with information to manage the many elements such as skits, characters, hosts, musical guests that go into scheduling a show that has such a long run and impact on television history.

# ER Diagram

# Tables

**People**

```
CREATE TABLE People(
        pid             integer not null,
        fname           text not null,
        lname           text not null,
        DOB             date,
        salaryUSD       integer,
        primary key(pid)

);
```

| | pid integer | fname text | lname text | dob date | salaryusd integer |
|---|---|---|---|---|---|
| 1 | 1 | John | Belushi | 1949-01-24 | 4000 |
| 2 | 2 | Kristen | Wiig | 1973-08-22 | 12500 |
| 3 | 3 | Will | Ferell | 1967-07-16 | 17500 |
| 4 | 4 | Chris | Farley | 1964-02-15 | 5000 |
| 5 | 5 | Bill | Hader | 1978-06-07 | 12500 |
| 6 | 6 | Dan | Aykroyd | 1952-07-01 | 4000 |
| 7 | 7 | Gilda | Radner | 1946-06-28 | 4000 |
| 8 | 8 | Adam | Sandler | 1966-09-09 | 5000 |
| 9 | 9 | Bill | Murray | 1950-09-21 | 4000 |
| 10 | 10 | Mike | Meyers | 1963-05-25 | 4000 |
| 11 | 11 | Jimmy | Fallon | 1963-05-25 | 10000 |
| 12 | 15 | Tom | Richards | 1972-09-14 | 1000 |
| 13 | 16 | Doug | Abeles | 1963-05-25 | 10000 |
| 14 | 17 | Lorne | Michaels | 1944-11-17 | 350000000 |
| 15 | 18 | James | Downey | 1956-08-29 | 5000 |
| 16 | 12 | Martin | Short | 1950-03-26 | |
| 17 | 13 | Richard | Pryor | 1940-12-01 | |
| 18 | 14 | Leslie | Nielson | 1926-02-11 | |

**Crew**

```
CREATE TABLE Crew(
        pid             integer not null references People(pid),
        role            text not null,
        startDate       date,
primary key(pid)

);
```

**Hosts**

```
CREATE TABLE Hosts(
        pid             integer not null references People(pid),
        occupation      text not null,
        numShowsHosted  integer not null,
        prevCastMemb    boolean,
primary key(pid)

);
```

| | pid integer | role text | startdate date |
|---|---|---|---|
| 1 | 15 | Camera Man | 1988-06-09 |
| 2 | 17 | Creator | 1975-08-02 |
| 3 | 18 | Writer | 1998-12-09 |
| 4 | 16 | Director | 1980-02-11 |

| | pid integer | occupation text | numshowshosted integer | prevcastmemb boolean |
|---|---|---|---|---|
| 1 | 12 | Comedian | 3 | t |
| 2 | 13 | Comedian | 1 | f |
| 3 | 14 | Actor | 1 | f |
| 4 | 11 | Actor | 2 | t |
| 5 | 2 | Actor | 1 | t |
| 6 | 9 | Actor | 7 | t |
| 7 | 10 | Actor | 1 | t |
| 8 | 3 | Actor | 3 | t |
| 9 | 6 | Actor | 1 | t |

**MusicalGuest**

```
CREATE TABLE musicalGuest(
        musicID                 integer not null,
        artistName              text not null,
        genre                   text,
primary key(musicID)
);
```

| | musicid integer | artistname text | genre text |
|---|---|---|---|
| 1 | 1 | No Doubt | Pop Punk |
| 2 | 2 | Gil Scott-Heron | soul |
| 3 | 3 | Cowboy Junkies | Rock |
| 4 | 4 | Justin Timberlake | Pop |
| 5 | 5 | The xx | R & B |

**scheduleEachEpisode**

```
CREATE TABLE scheduleEachEpisode(
        eid             decimal(4,2) not null references Ep
        musicID                 integer not null references
musicalGuest(musicID),
        pid             integer not null references People(
primary key(eid, musicID, pid)
);
```

| | eid numeric(4,2) | musicid integer | pid integer |
|---|---|---|---|
| 1 | 22.08 | 1 | 12 |
| 2 | 1.07 | 2 | 13 |
| 3 | 14.13 | 3 | 14 |
| 4 | 14.13 | 4 | 11 |
| 5 | 39.10 | 5 | 2 |

**snlCast**

```
CREATE TABLE snlCast(
        pid                     integer not null references People(pid),
        startSeason             integer not null,
        endSeason               integer not null,
        numOfImpressions            integer,
        bestImpression      text
--foreign key(pid)
);
```

**Characters**

```
CREATE TABLE Characters(
        charID          integer not null references Characters(charID),
        name            text not null,
        numAppearances                  integer,
primary key(charID)
);
```

**playsCharacter**

```
CREATE TABLE playsCharacter(
        pid             integer not null references People(pid),
        charID          integer not null references Characters(charID),
primary key(pid, charID)
);
```

**Sketches**

```
CREATE TABLE Sketches(
        sketchID        integer not null,
        title           text not null,
        sketchType      text not null check(sketchType ='coldOpen' or sketchType
='commercial' or sketchType ='weekendUpdate' or sketchType ='skit'),
        lengthMin       decimal (3,2) not null,
primary key(sketchID)
);
```

**charactersInSketch**

```
CREATE TABLE charactersInSketch(
        charID          integer not null references Characters(charID),
        sketchID        integer not null references Sketches(sketchID),
        primary key(charID, sketchID)
);
```

**Episodes**

```
CREATE TABLE Episodes(
        eid             decimal(4,2) not null,
        airDate         date not null,
primary key(eid)
);
```

**sketchesInEpisode**

```
CREATE TABLE sketchesInEpisode(
        sketchID        integer not null references Sketches(sketchID),
        eid             decimal(4,2) not null references Episodes(EID),
primary key(sketchID,eid)
);
```

**SpinOffs**

```
CREATE TABLE SpinOffs(
        spinID          integer not null,
        title           text not null,
        typeOfSpin      text not null check(typeOfSpin='movie' or typeOfSpin
='tvShow'),
        releaseDate     integer not null,
primary key(spinID)
);
```

**spinOffSketches**

```
CREATE TABLE spinOffSketches(
        sketchID        integer not null references Sketches(sketchID),
        spinID          integer not null references SpinOffs(spinID),
primary key(sketchID, spinID)
);
```

# Views

**episodeGuide**: Lists the number, celebrity host, musical guest and air date of each episode.

```
CREATE OR REPLACE VIEW episodeGuide as
SELECT e.eid AS "Episode",
        e.airDate,
       p.fname AS "Host First Name",
       p.lname AS "Host Last Name",
       mg.artistName AS "Musical Guest"
FROM people p, MusicalGuest mg, scheduleEachEpisode shed, episodes e
WHERE shed.musicID = mg.musicID AND shed.pid = p.pid AND shed.eid = e.eid
ORDER BY e.eid ASC;

select * from episodeGuide;
```

## **characterCast**: Lists the cast member, the characters they play, and the corresponding sketch the character is in.

```
CREATE OR REPLACE VIEW characterCast as
SELECT sc.pid, c.name, c.charID
        FROM characters c
        INNER JOIN playsCharacter pc        ON c.charID  = pc.charID
        INNER JOIN snlCast sc               ON pc.pid    = sc.pid;
CREATE OR REPLACE VIEW peopleCast as
SELECT p.pid, p.fname, p.lname
        FROM people p
        INNER JOIN snlCast sc               ON p.pid     = sc.pid;
CREATE OR REPLACE VIEW characterSketch as
SELECT s.title, cS.charID
        FROM Sketches s
        INNER JOIN charactersInSketch cS    ON cS.sketchID = s.sketchID
        INNER JOIN characters c             ON cS.charID = c.charID;
select * FROM characterSketch;
Select pc.fname, pc.lname, cc.name, cS.title
from peopleCast pc
INNER JOIN characterCast cc                 ON pc.pid    = cc.pid
INNER JOIN characterSketch cS               ON cS.charID             = cc.charID
ORDER BY lname ASC;
```

# Reports and Interesting Queries

1. Query to return the Spin Off movies created from SNL sketches including the title, release date, and name of the cast members in the movie.

```
SELECT so.title, s.title, so.releaseDate, s.sketchType, s.lengthMin,
p.fname, p.lname
FROM SpinOffs so
LEFT OUTER JOIN spinOffSketches ss          ON ss.spinID = so.spinID
LEFT OUTER JOIN Sketches s                  ON s.sketchID       =
ss.sketchID
LEFT OUTER JOIN charactersInSketch cs       ON cs.sketchID      =
s.sketchID
LEFT OUTER JOIN characters c                ON c.charID  = cs.charID
LEFT OUTER JOIN playsCharacter pc           ON pc.charID        =
c.charID
LEFT OUTER JOIN people p                    ON p.pid       = pc.pid;
```

2. Query to return the show hosts who were previous cast members as well as the season the cast member started, the number of impressions, best impressions, ordered by number of shows hosted

```
--shows hosts who are also previous cast members
SELECT h.numShowsHosted, p.fname, p.lname,
h.occupation,startSeason, numOfImpressions, bestImpression
FROM  hosts h
INNER JOIN people p              ON p.pid = h.pid
INNER JOIN snlCast sc            ON p.pid = sc.pid
WHERE prevCastMemb = true
ORDER BY numShowsHosted ASC;
```

Stored Procedures

skitsPerEpisode: Takes an episode number as and argument and returns the titles of the sketches for the given episode.

```
CREATE OR REPLACE FUNCTION skitsPerEpisode(decimal(4,2), REFCURSOR)
RETURNS refcursor as $$
DECLARE
        episodeInput decimal(4,2) := $1;
        resultset REFCURSOR := $2;
BEGIN
 open resultset for
        SELECT s.title AS "Sketches"
        FROM sketches s
        INNER JOIN sketchesInEpisode se
        ON s.sketchID = se.sketchID
        INNER JOIN episodes e
        ON e.eid = se.eid
                WHERE episodeInput = se.eid;
 return resultset;
end;
$$
```

# numbOfCharacters: Takes a cast member name as an argument and returns the number of characters they play.

```
-- Stored Procedure to see how many characters each cast member plays
CREATE OR REPLACE FUNCTION numbOfCharacters(text, REFCURSOR)
RETURNS refcursor as $$
DECLARE
                castMembInput text := $1;
                resultset REFCURSOR := $2;
BEGIN
 open resultset for
                SELECT  count(pc.pid) AS "Number of Characters Played"
                FROM playsCharacter pc
                INNER JOIN people p
                ON p.pid = pc.pid
                                WHERE castMembInput = p.fname;
 return resultset;
end;
$$
language plpgsql;

SELECT numbOfCharacters('Adam', 'results');
FETCH ALL FROM results;
```

# seasonsInCast: Takes cast member name as a function and returns the number of seasons they were on SNL

```
CREATE OR REPLACE FUNCTION seasonsInCast(text, REFCURSOR)
RETURNS refcursor as $$
DECLARE
                castInput text := $1;
                resultset REFCURSOR := $2;
BEGIN
 open resultset for
                SELECT  p.fname, p.lname, sc.startSeason, sc.endSeason,
                (sc.endSeason - sc.startSeason) AS numOfSeasons
                FROM people p INNER JOIN snlCast sc on p.pid = sc.pid
                                WHERE castInput = p.fname;
 return resultset;
end;
$$
language plpgsql;

SELECT seasonsInCast('Bill', 'results');
FETCH ALL FROM results;
```

# Triggers

```
create or replace function quitShow() returns trigger as
$$
begin
if new.endSeason is not null
and (select endSeason
from snlCast
where pid = new.pid) is null
then
update snlCast
set endSeason = new.endSeason
where pid = new.pid;
end if;
return new;
end;
$$
language plpgsql;
```

```
create trigger quitShow
after update on snlCast
for each row
```

# Security

```
--security
CREATE ROLE admin;
GRANT ALL ON TABLES
IN SCHEMA PUBLIC
TO admin;

CREATE ROLE user
GRANT SELECT
ON ALL TABLES IN SCHEMA PUBLIC
TO user;
```

# Implementation Notes

# Known Problems & Future Enhancements