Live from New York…
It's a Saturday Night Live Database!

Designed by Jenna Ficula

Table of Contents:

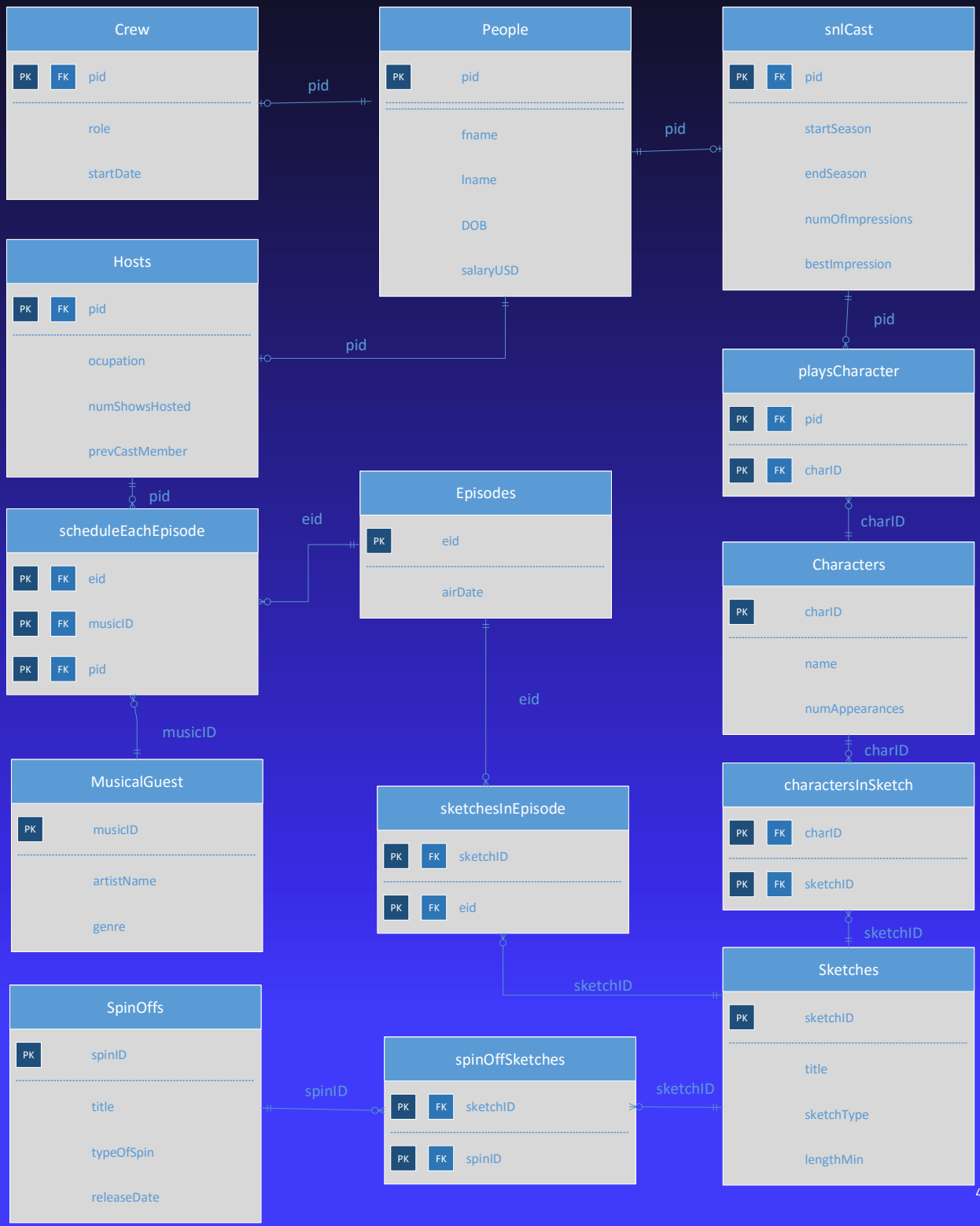
Executive Summary..... 3
Entity Relationship Diagram..... 4
Tables..... 5
Views.....15
Reports and Interesting Queries.....17
Stored Procedures.....21
Triggers.....24
Security.....25
Implementation Notes.....26
Known Problems & Future Enhancements.....27

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 and 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 Saturday Night Live employees and enthusiasts with information to organize the many elements of the show.

The database will allow for a greater understanding of how to schedule the show to provide maximum entertainment value to fans. Included in the database are statistics and analysis to better understand a show that has a profound cultural, historical, and political influence. The goal is to provide a functional, normalized database to benefit anyone who enjoys a good laugh.

ER Diagram


Crew
PK FK pid
role
startDate
People
PK pid
fname
lname
DOB
salaryUSD
snlCast
PK FK pid
startSeason
endSeason
numOfImpressions
bestImpression
pid
pid
pid
Hosts
PK FK pid
ocupation
numShowsHosted
prevCastMember
pid
playsCharacter
PK FK pid
PK FK charID
charID
Episodes
PK eid
airDate
eid
scheduleEachEpisode
PK FK eid
PK FK musicID
PK FK pid
Characters
PK charID
name
numAppearances
charID
musicID
eid
MusicalGuest
PK musicID
artistName
genre
sketchesInEpisode
PK FK sketchID
PK FK eid
charactersInSketch
PK FK charID
PK FK sketchID
sketchID
sketchID
Sketches
PK sketchID
title
sketchType
lengthMin
SpinOffs
PK spinID
title
typeOfSpin
releaseDate
spinID
spinOffSketches
PK FK sketchID
PK FK spinID
sketchID

Tables

People: this table contains all of the people and their attributes such as first name, last name, date of birth, and salary.

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

);
```

Functional Dependencies:
People (pid) →

fname, lname, DOB, salaryUSD

	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: The crew table contains the person id, the role of the crew member and the date their start date. The crew does not include cast.

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

Functional Dependencies:
Crew (pid) → role, startDate

	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

Hosts: The host table includes the person id, the occupation of the host, and the number of times the person has hosted, and if they have been a cast member.

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

Functional Dependencies:
Hosts (pid) → occupation, numShowsHosted, prevCastMember

	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: This table displays the musical guest id, name, the genre of music.

```
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

Functional Dependencies:
musicalGuest (musicID) → artistName, genre

---

scheduleEachEpisode: This table connects the episode id to the host and musical guest of the show to represent the schedule for a single episode.

```
CREATE TABLE scheduleEachEpisode(
        eid             decimal(4,2) not null references Episodes(EID),
        musicID         integer not null references musicalGuest(musicID),
        pid             integer not null references People(pid),
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

Functional Dependencies:
scheduleEachEpisode
(eid, musicID, pid) →

snlCast: This table represents the cast members and their attributes including start and end seasons, number of and best impressions.

```
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)
);
```

Functional Dependencies:
Cast (pid) → startSeason , endSeason, numOfImpressions, bestImpression

	pid integer	startseason integer	endseason integer	numofimpressions integer	bestimpression text
1	1	1	4	11	Joe Cocker
2	2	31	37	24	Paula Deen
3	3	22	27	24	Alex Trebek
4	4	16	23	27	Meat Loaf
5	5	31	38	82	Al Pacino
6	6	1	4	25	Julia Child
7	7	1	4	20	Barbra Walters
8	8	16	20	21	Bruce Springsteen
9	9	2	5	22	Walter Cronkite
10	10	14	20	39	Mick Jagger
11	11	24	29	71	Adam Sandler
12	12	10	11	8	Ed Grimley

Characters: This table holds the characters for the show including their id number, name, and number of appearances

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

Functional Dependencies:
Characters (charID) → name, numAppearances

	charid integer	name text	numappearances integer
1	1	Gilly	6
2	2	Roseanne Roseannadanna	17
3	3	Stefon	18
4	4	Samurai Futaba	17
5	5	Matt Foley	8
6	6	Opraman	10
7	7	Alex Trebek	7
8	8	Nick the Lounge Singer	9
9	9	Wayne Campbell	21
10	10	MacGruber	6
11	11	Ronnie the Mechanic	1
12	12	Joliet Jake Blues	5
13	13	Elwood Blues	5
14	14	Police Officer	1

---

playsCharacter: This table connects the cast member with the characters they play.

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

Functional Dependencies:
playsCharacter (pid, charID) →

	pid integer	charid integer
1	2	1
2	7	2
3	1	4
4	4	5
5	8	6
6	3	7
7	9	8
8	10	9
9	5	10
10	4	11
11	1	12
12	6	13
13	13	14

Sketches: This table contains the sketches and their attributes such as sketch title, type, and length.

```
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)
);
```

Functional Dependencies:
Episodes (sketchID) → title, sketchType, lengthMin

	sketchid integer	title text	sketchtype text	lengthmin numeric(3,2)
1	1	Samurai Hotel	skit	3.31
2	2	Celebrity Jeapordy	skit	6.21
3	3	Wanes World	skit	3.27
4	4	Coneheads	skit	5.12
5	5	Blues Brothers	skit	6.17
6	6	MacGruber	skit	6.53

charactersInSketch: This table connects the characters with the sketches they are in.

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

	charid integer	sketchid integer
1	4	1
2	7	2
3	9	3
4	11	4
5	12	5
6	13	5
7	10	6
8	14	1

Functional Dependencies:
Episodes (charID, sketchID) →

Episodes: This table includes the id of the episodes of the show and their airdate.

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

Functional Dependencies:
Episodes (eid) → airDate

	eid numeric(4,2)	airdate date
1	22.08	1996-12-17
2	1.07	1975-12-13
3	14.13	1989-02-18
4	39.10	2013-12-21
5	42.07	2013-12-21

---

sketchesInEpisode: This table connects the sketch and the corresponding episode the sketch is in.

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

Functional Dependencies:
Episodes (eid, sketchID) →

	sketchid integer	eid numeric(4,2)
1	2	22.08
2	1	1.07
3	3	14.13

SpinOffs: This table includes the spin offs of the show and their release date.

```
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)
);
```

	spinid integer	title text	typeofspin text	releasedate integer
1	1	Superstar	movie	1999
2	2	Waynes World	movie	1992
3	3	Coneheads	movie	1993
4	4	MacGruber	movie	2010
5	5	A Night at the Roxbury	movie	1998
6	6	Blues Brothers	movie	1980

Functional Dependencies:
SpinOffs (spinID) → title, typeOfSpin, relseaseDate

spinOffSketches: This table connects the spin off to the sketches to their corresponding sketch that the spin off is based off of.

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

	sketchid integer	spinid integer
1	3	2
2	4	3
3	5	6
4	6	4

Functional Dependencies:
spinOffSketches (sketchID, spinID) →

14

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;
```

	Episode numeric(4,2)	airdate date	Host First Name text	Host Last Name text	Musical Guest text
1	1.07	1975-12-13	Richard	Pryor	Gil Scott-Heron
2	14.13	1989-02-18	Leslie	Nielson	Cowboy Junkies
3	14.13	1989-02-18	Jimmy	Fallon	Justin Timberlake
4	22.08	1996-12-17	Martin	Short	No Doubt
5	39.10	2013-12-21	Kristen	Wiig	The xx

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;
```

	fname text	lname text	name text	title text
1	Dan	Aykroyd	Elwood Blues	Blues Brothers
2	John	Belushi	Samurai Futaba	Samurai Hotel
3	John	Belushi	Joliet Jake Blues	Blues Brothers
4	Chris	Farley	Ronnie the Mechanic	Coneheads
5	Will	Ferell	Alex Trebek	Celebrity Jeapordy
6	Bill	Hader	MacGruber	MacGruber
7	Mike	Meyers	Wayne Campbell	Wanes World

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;
```

	title text	coalesce text	releasedate integer	coalesce text	coalesce numeric	coalesce text	coalesce text
1	Waynes World	Wanes World	1992	skit	3.27	Mike	Meyers
2	MacGruber	MacGruber	2010	skit	6.53	Bill	Hader
3	Coneheads	Coneheads	1993	skit	5.12	Chris	Farley
4	Blues Brothers	Blues Brothers	1980	skit	6.17	John	Belushi
5	Blues Brothers	Blues Brothers	1980	skit	6.17	Dan	Aykroyd
6	Superstar	None	1999	None	0	None	None
7	A Night at the Roxbury	None	1998	None	0	None	None

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

```
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;
```

	numshowshosted integer	fname text	lname text	occupation text	startseason integer	numofimpressions integer	bestimpression text
1	1	Dan	Aykroyd	Actor	1	25	Julia Child
2	1	Kristen	Wiig	Actor	31	24	Paula Deen
3	1	Mike	Meyers	Actor	14	39	Mick Jagger
4	2	Jimmy	Fallon	Actor	24	71	Adam Sandler
5	3	Martin	Short	Comedian	10	8	Ed Grimley
6	3	Will	Ferell	Actor	22	24	Alex Trebek
7	7	Bill	Murray	Actor	2	22	Walter Cronkite

3. Query shows the characters where the cast member who plays them makes the least money. It also displays season the cast member started in because the earlier cast members had lower salaries.

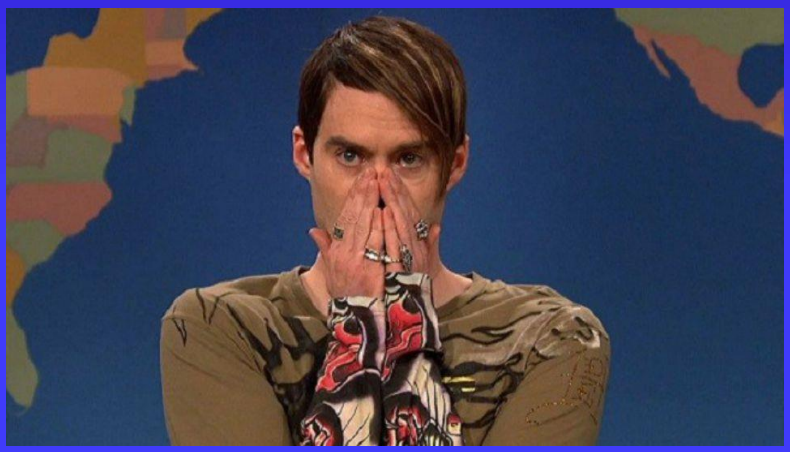
```
SELECT p.fname, p.lname, c.name, sc.startSeason
FROM characters c
    INNER JOIN playsCharacter pc        ON c.charID = pc.charID
    INNER JOIN snlCast sc               ON sc.pid = pc.pid
    INNER JOIN people p                 ON p.pid = sc.pid
WHERE (p.salaryUSD < 5000);
```

	fname text	lname text	name text	startseason integer
1	John	Belushi	Joliet Jake Blues	1
2	John	Belushi	Samurai Futaba	1
3	Dan	Aykroyd	Elwood Blues	1
4	Gilda	Radner	Roseanne Roseannadanna	1
5	Bill	Murray	Nick the Lounge Singer	2
6	Mike	Meyers	Wayne Campbell	14

4. Query shows the names of cast members who can perform more than 25 impressions.

```
SELECT p.fname, p.lname, SUM(sc.numOfImpressions) AS
"Number of Impressions"
FROM snlCast sc
        INNER JOIN people p          ON p.pid = sc.pid
GROUP BY p.pid
HAVING SUM(numOfImpressions) > 25
ORDER BY p.fname;
```

	fname text	lname text	Number of Impressions bigint
1	Bill	Hader	82
2	Chris	Farley	27
3	Jimmy	Fallon	71
4	Mike	Meyers	39



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 $$
DECLARE
        episodeInput decimal(4,2) := $1;
        resultset REFCURSOR := $2;
BEGIN
 open resultset for
        SELECT s.title AS "Sketches"
        FROM sketches s
        INNER JOIN sketchesInEpisode se
        ON s.sketchID = se.sketchID
        INNER JOIN episodes e
        ON e.eid = se.eid
                WHERE episodeInput = se.eid;
 return resultset;
end;
$$
language plpgsql;

SELECT skitsPerEpisode('01.70', 'results');
FETCH ALL FROM results;
```

	Sketches text
1	Samurai Hotel

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

```
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;
```

	Number of Characters Played bigint
1	1

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;
```

	fname text	lname text	startseason integer	endseason integer	numofseasons integer
1	Bill	Hader	31	38	7
2	Bill	Murray	2	5	3

Triggers

Updates crew with pid that has been inserted into people table with a salaryUSD < 4000 USD

```
CREATE OR REPLACE FUNCTION addtoSketches()
RETURNS TRIGGER AS
$$
BEGIN
  IF NEW.salaryUSD < 4000 THEN
    INSERT INTO Crew(pid)
    VALUES (NEW.pid);
  END IF;
  RETURN NEW;
END;
$$
language plpgsql;

-- Update trigger
CREATE TRIGGER addtoCharacters
AFTER INSERT ON people
FOR EACH ROW
EXECUTE PROCEDURE addtoSketches();

-- Test them.
INSERT INTO people(pid, fname, lname, DOB, salaryUSD)
    VALUES (100, 'John', 'Smith', '1954-12-11', 1);

Select * from crew;
```

Before Trigger:

	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

After Trigger:

	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
5	100		

Security

Administrator: The database administrator has full access to the database and can insert, update, and delete anything in the database for upkeep.

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

User: The user is anyone who would like to view the database, but has select privileges as to not alter the database. These users could be anyone from writers of the show to just SNL enthusiasts.

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

Implementation Notes

- Implementation went fairly well, although difficult at first to decide which sketches, season, and episodes should be connected and included.

- I choose to include less data to make the database manageable, but I wish I had a chance to include more.
- Retrospectively, I could have stuck to one season or one cast, although I thought a cross section of seasons, characters, and casts would make for more interesting queries and analysis.

- I had trouble with implementing triggers since pid is a commonly used primary and foreign key, it was difficult ensuring that it was not accidentally inserted into the wrong tables.

Known Problems & Future Enhancements

- Known problems include a lack of data including all of the casts, skits, seasons, and characters to expand the database.
- There could also be more connections drawn between cast members themselves other than just the skits they perform in together.
- Another improvement would to include the division of sketches themselves. For example, the "Weekend Update" section could include tables such as the rotating "anchors" of weekend update and the different "guests" that appear on the satirical news segment. This would make for interesting additional tables and more detailed show programming.
- I would also like to include the projects SNL cast members work on after they leave the show. For example, many cast members costar in movies and television shows together when they leave. (Grown Ups, Cadyshack , 30 Rock)