




GETAJOB CONSULTANTS, LLC

Database designed by: Carolena
Realmuto



EXECUTIVE SUMMARY

GETAJOB Consultants, LLC is looking to improve their work efficiency, job placement accuracy, and overall satisfactory rating in order to gain a competitive advantage over other employment agencies in the Big Apple.

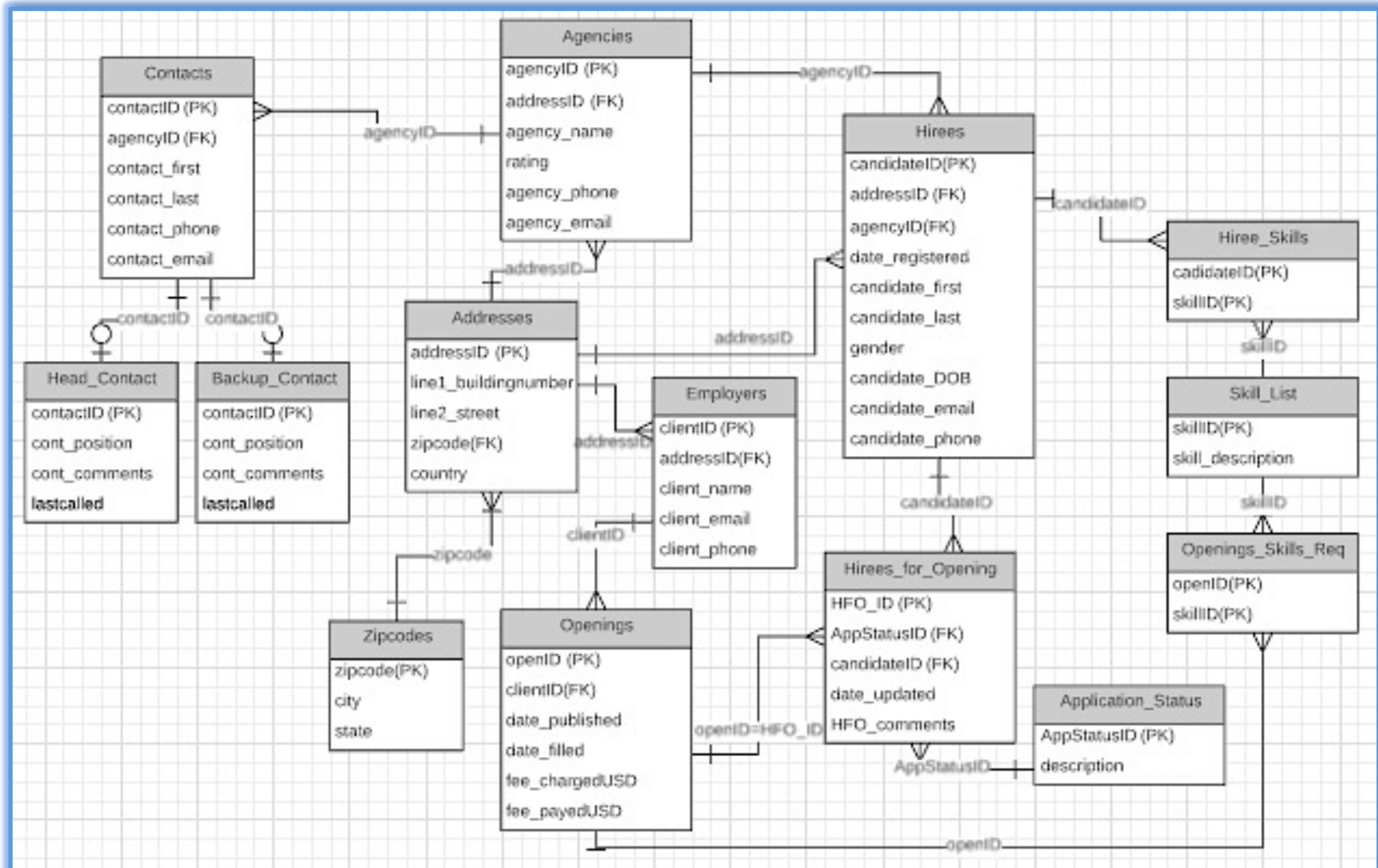
They hired me, a database developer, to create a database that can help them achieve their goals. I have designed one that gives the company an overall glance at all the agencies in the area, their customers, and openings, so it can quickly and successfully match their candidates to a job.

GETAJOB Consultants, LLC only works with Wall Street Banks in the area and their candidates are looking for an opportunity within the financial industry.

TABLE OF CONTENTS

ENTITY-RELATIONSHIP DIAGRAM.....	4
TABLES	5
VIEWS	18
STORED PROCEDURES	22
TRIGGERS.....	24
REPORTS	25
SECURITY	26
IMPLEMENTATION NOTES	27
KNOWN PROBLEMS & FUTURE ENHANCEMENTS	28

ENTITY-RELATIONSHIP DIAGRAM



TABLES

Agencies – this table lists all the available NYC Financial employment agencies and their information

- Functional dependencies: agencyID → addressID, agency_name, rating, agency_phone, agency_email

```
CREATE TABLE Agencies (  
  agencyID char(3) NOT NULL,  
  agency_name TEXT NOT NULL,  
  addressID char(3) NOT NULL references Addresses(addressID),  
  agency_phone TEXT NOT NULL,  
  agency_email TEXT,  
  rating TEXT,  
  PRIMARY KEY (agencyID)  
);
```

agencyid character(3)	agency_name text	addressid character(3)	agency_phone text	agency_email text	rating text
a01	GETAJOB Consultants	013	(212)-123-1231	contact@GETAJOB.com	A
a02	Creative Financial Staffing	010	(212)-444-3342	contact@CFS.com	AA
a03	Accelerate Financial Staffing	011	(212)-886-1363	hiring@AFS.com	BB
a04	Wall Street Services	012	(212)-299-1239	contact@wallstreetservices.com	A
a05	Minion Consulting	003	(131)-123-1837	minion@3NFConsulting.com	AA
a06	Forum Employment	003	(231)-822-8189	ask@forumemployment.com	B

Contacts- this table lists all of the recruitment agency's contacts and their contact information

- Functional dependencies: contactID → agencyID, contact_first, contact_last, contact_phone, contact_email

```
CREATE TABLE Contacts (  
    contactID char(3) NOT NULL,  
    agencyID char(3) NOT NULL references Agencies(agencyID),  
    contact_first TEXT NOT NULL,  
    contact_last TEXT NOT NULL,  
    contact_phone TEXT NOT NULL,  
    contact_email TEXT,  
    PRIMARY KEY(contactID)  
);
```

contactid character(3)	agencyid character(3)	contact_first text	contact_last text	contact_phone text	contact_email text
001	a04	Sean	Sullivan	(516)-763-1231	seansullivan@wallstreetservices.com
002	a01	Erica	Rose	(631)-624-8098	e.falco@GETAJOB.com
003	a02	Lauren	Suran	(516)-112-1314	lauransuran@CFS.com
004	a02	Ronald	Cavaliere	(516)-241-2342	ronaldcavaliere@CFS.com
005	a01	Erin	Murtha	(631)-953-2941	e.murtha@GETAJOB.com
006	a03	Brian	Haughey	(231)-524-1475	brianh@AFS.com
007	a03	John	Finnigan	(231)-813-4812	johnf@AFS.com

Head_Contact – this table is a subtype of the “Contacts” table above and is the primary contact of the agency for employers.

- Functional dependencies: contactID → cont_position, cont_comments, lastcalled

```
CREATE TABLE Head_Contact (  
    contactID char(3) NOT NULL references Contacts(contactID),  
    cont_position TEXT NOT NULL CHECK (cont_position='FT' or cont_position='PT'),  
    cont_comments TEXT,  
    lastcalled DATE NOT NULL,  
    PRIMARY KEY(contactID)  
);
```

contactid character(3)	cont_position text	cont_comments text	lastcalled date
001	FT	available from 9-5 weekdays	2016-05-24
002	FT	available from 10-3 weekdays	2017-01-29
003	FT	avaibale from 8-6 daily	2017-04-08
006	PT	available MWF	2016-12-17

Backup_Contact- this table is a subtype of the “Contacts” table and is the secondary contact of the agency for employers in case the head contact is not available.

- Functional dependencies: contactID → cont_position, cont_comments, lastcalled

```
CREATE TABLE Backup_Contact (  
    contactID char(3) NOT NULL references Contacts(contactID),  
    cont_position TEXT NOT NULL CHECK (cont_position='FT' or cont_position='PT'),  
    cont_comments TEXT,  
    lastcalled DATE NOT NULL,  
    PRIMARY KEY(contactID)  
);
```

contactid character(3)	cont_position text	cont_comments text	lastcalled date
004	FT	available 8-5 daily	2017-02-13
005	FT	available 9-4 daily	2017-01-13
007	PT	available only TF	2016-11-23

Addresses- this table lists all of the addresses of the Agencies, Hirees, and Employers

- Functional dependencies: addressID → line1_buildingnumber, line2_street, zipcode, country

```
CREATE TABLE Addresses (  
    addressID char(3) NOT NULL,  
    line1_buildingnumber TEXT NOT NULL,  
    line2_street TEXT NOT NULL,  
    zipcode TEXT NOT NULL references  
    Zipcodes(zipcode),  
    country TEXT NOT NULL,  
    PRIMARY KEY(addressID)  
);
```

addressid character(3)	line1_buildingnumber text	line2_street text	zipcode text	country text
001	3399	North Road	12601	USA
002	18	Concord Road	11050	USA
003	95	Wall Street	10005	USA
004	60	Wall Street	10005	USA
005	270	Park Ave	10017	USA
006	390	Greenwich Street	10013	USA
007	200	West Street	10282	USA
008	299	Park Ave	10171	USA
009	1585	Broadway	10036	USA
010	488	Madison Ave	10022	USA
011	5	Pennsylvania Plaza	10001	USA
012	11	Broadway	10004	USA
013	4	New York Plaza	10004	USA
014	19	Daffedil Drive	07920	USA
015	121	Sugar Lane	12524	USA
016	007	Bond Street	00007	USA

Zipcodes- this table keeps track of zipcodes and their appropriate cities and states. It also fixes the issue of cities having multiple zipcodes.

- Functional dependencies: zipcode → city, state

```
CREATE TABLE Zipcodes (  
  zipcode TEXT NOT NULL,  
  city TEXT NOT NULL,  
  state char(2) NOT NULL,  
  PRIMARY KEY(zipcode)  
);
```

zipcode text	city text	state character(2)
12601	Poughkeepsie	NY
11050	Port Washington	NY
10005	New York	NY
10017	New York	NY
10013	New York	NY
10282	New York	NY
10171	New York	NY
10036	New York	NY
10022	New York	NY
10001	New York	NY
10004	New York	NY
07920	Bridgewater	NJ
12524	Fishkill	NY
00007	Stamford	CT

Employers- table of all current and potential employers (aka clients) of the recruitment agency's contacts and whom the agencies work with.

- Functional dependencies: clientID → addressID, client_name, client_email, client_phone

```
CREATE TABLE Employers (  
  clientID char(3) NOT NULL,  
  addressID char(3) NOT NULL references Addresses(addressID),  
  client_name TEXT,  
  client_email TEXT,  
  client_phone TEXT,  
  PRIMARY KEY(clientID)  
);
```

clientid character(3)	addressid character(3)	client_name text	client_email text	client_phone text
001	004	Deutsche Bank	HR@deutschebank.com	(231)-158-1730
002	005	JP Morgan Chase	hire@jpm.com	(918)-985-2000
003	006	Citigroup	HR@citigroup.com	(919)-718-8190
004	007	Goldman Sachs	goldmanhire@GS.com	(231)-912-1939
005	008	UBS	contact@UBS.com	(918)-429-8000
006	009	Morgan Stanley	questions@morganstanley.com	(231)-131-1827
007	004	Blackrock	hiring@blackrock.com	(212)-810-5300

Openings- table of all the current job openings as published by the employers.

- Functional dependencies: openID→clientID, job_description, date_published, date_filled, fee_charged, fee_payed

```
CREATE TABLE Openings (  
  openID char(3) NOT NULL,  
  clientID char(3) NOT NULL references Employers(clientID),  
  job_description char(200) NOT NULL,  
  date_published DATE,  
  date_filled DATE,  
  fee_chargedUSD INT,  
  fee_payedUSD INT, PRIMARY KEY(openID) );
```

openid character(3)	clientid character(3)	job_description character(200)	date_published date	date_filled date	fee_chargedusd integer	fee_payedusd integer
001	001	Financial Analyst- Asset Management	2017-01-01		2500	2000
002	003	SVP - Legal Finance Management	2016-08-23	2016-12-10	2500	2500
003	002	Analyst- CIB Reporting	2016-11-16		1800	1760
004	002	Managing Director- Corp Finance OTC	2017-02-28	2017-03-18	1950	1950
005	004	Executive Director- Wealth Budgeting	2016-09-16		1550	1300
006	005	Admin - Client Banking	2017-02-23	2017-04-08	1000	1000
007	006	Analyst-Account Control	2017-01-17		890	0
008	001	Analyst-Internal Auditing	2016-10-31	2017-02-05	720	720

Skill_List- organizes different skill sets to be assigned to candidates in the hiree_skills table which were matched to the skillset in the Openings_Skills_Req table.

- Functional dependencies: skillID→skill_description

```
CREATE TABLE Skill_List (  
  skillID char(3) NOT NULL,  
  skill_description char(200),  
  PRIMARY KEY(skillID)  
);
```

skillid character(3)	skill_description character(200)
001	Asset Management
002	Auditing
003	Secretarial expertise
004	Budget reporting
005	Management expertise
006	Advanced knowledge in Excel
007	Accounting knowledge
008	Degree in Business

Openings_Skills_Req- this table pairs the opening published to a given skillset.

- Functional dependencies: [openID, skillID] →

```
CREATE TABLE Openings_Skills_Req (  
  openID char(3) NOT NULL references Openings(openID),  
  skillID char(3) NOT NULL references Skill_List(skillID),  
  PRIMARY KEY(openID, skillID)  
);
```

openid character(3)	skillid character(3)
001	001
001	006
001	008
002	006
002	004
003	004
003	008
004	005
004	006
004	008
005	005
005	004
005	008
006	003
007	007
007	006
008	007
008	002
008	006

Hiree_Skills- this table is an associative entity (similar to Openings_Skills_Req), which links the candidate to a skill.

- Functional dependencies: [candidateID, skillID] →

```
CREATE TABLE Hiree_Skills (  
  candidateID char(4) NOT NULL references Hirees(candidateID),  
  skillID char(3) NOT NULL references Skill_List(skillID),  
  PRIMARY KEY (candidateID,skillID)  
);
```

candidateid character(4)	skillid character(3)
1000	005
1000	006
1001	008
1001	001
1002	007
1002	004
1003	002
1003	003
1004	007
1004	008
1004	005

Hirees- a table of all of the current, past, and potential candidates using the agency to look for jobs.

- Functional dependencies: candidateID → addressID, agencyID, date_registered, candidate_first, candidate_last, gender, candidate_DOB, candidate_email, candidate_phone

```
CREATE TABLE Hirees (
    candidateID char(4) NOT NULL,
    addressID char(3) NOT NULL references Addresses(addressID),
    agencyID char(3) NOT NULL references Agencies(agencyID),
    date_registered DATE,
    candidate_first char(20),
    candidate_last char(20),
    gender TEXT CHECK (gender='M' or gender='F' or gender='O'),
    candidate_DOB DATE,
    candidate_email TEXT,
    candidate_phone TEXT,
    PRIMARY KEY(candidateID) );
```

candidateid character(4)	addressid character(3)	agencyid character(3)	date_registered date	candidate_first character(20)	candidate_last character(20)	gender text	candidate_dob date	candidate_email text	candidate_phone text
1000	001	a01	2016-02-19	Alan	Labouseur	M	1980-05-06	alanlabouseur@marist.edu	(845)-575-3000
1001	002	a01	2016-05-20	Carolena	Realmuto	F	1995-08-23	carolenarealmuto1@marist.edu	(516)-512-1010
1002	014	a01	2016-12-21	Kimberly	Woodward	F	1996-01-18	kimw@gmail.com	(741)-913-1929
1003	015	a01	2017-03-28	Sam	Smith	M	1980-11-12	samsmith@gmail.com	(918)-313-7192
1004	016	a01	2017-04-03	James	Bond	M	1979-06-29	jamesbond007@gmail.com	(007)-007-0007
1005	003	a01	2016-02-19	Tedd	Codd	M	1935-06-18	TeddCodd@3NFconsulting.com	(845)-575-3000
1006	003	a01	2017-04-24	Lisa	Conroy	F	1961-08-08	LisaConroy@teleworm.us	(256)-610-3127

Hirees_for_Openings- this table is an associative entity that connects Openings to Hirees and also shows the status of the recruiting process.

- Functional dependencies: HFO_ID → AppStatusID, candidateID, date_updated, HFO_comments

```
CREATE TABLE Hirees_for_Openings (  
  HFO_ID char(3) NOT NULL,  
  AppStatusID char(2) NOT NULL references Application_Status(AppStatusID),  
  date_updated DATE,  
  HFO_comments char(200),  
  PRIMARY KEY(HFO_ID)  
);
```

hfo_id character(3)	candidateid character(4)	appstatusid character(2)	date_updated date	hfo_comments character(200)
003	1002	03	2017-04-08	Balance not completely paid- email next week
006	1003	01	2017-03-12	interview date set for April 29th
001	1001	04	2017-02-28	Offer rejected- taking job at JP Morgan due to poor Deutsche bank performance

Application_Status- a table that lists the different types of application statuses that are used within the database.

- Functional dependencies: AppStatusID → description

```
CREATE TABLE Application_Status (  
    AppStatusID char(2) NOT NULL,  
    description char(250),  
    PRIMARY KEY(AppStatusID) );
```

appstatusid character(2)	description character(250)
01	interviewing
02	Offer given
03	Offer accepted
04	Offer rejected

VIEWS

ACCOUNT_BALANCES: This view is useful to GETAJOB account managers since it displays the ClientID, the employer name, the fee charged for GETAJOB's services and the amount they paid. The difference of the last two columns is the amount outstanding that still needs to be paid.

```
CREATE OR REPLACE VIEW Account_Balances AS  
  
SELECT Employers.clientID, Employers.client_name,  
    Openings.openID, Openings.fee_chargedUSD,  
    Openings.fee_payedUSD  
  
FROM Openings INNER JOIN Employers ON  
    Openings.ClientID = Employers.clientID
```

clientid character(3)	client_name text	fee_chargedusd integer	fee_payedusd integer
001	Deutsche Bank	2500	2000
002	JP Morgan Chase	1800	1760
004	Goldman Sachs	1550	1300
006	Morgan Stanley	890	0

```
and Openings.fee_chargedUSD > Openings.fee_payedUSD;
```

Hiree_Skills_Description: This view shows the candidates, their IDs, and skill description given a specific skillID (entered in query). This is useful when GETAJOB employees want to see candidates with the same skills and choose between them for job opening positions.

```
CREATE OR REPLACE VIEW Hiree_Skills_Description AS
```

```
SELECT hirees.candidateID,
```

```
    hirees.candidate_first,
```

```
    hirees.candidate_last,
```

```
    skill_list.skill_description
```

```
FROM Hiree_skills inner join Hirees ON hirees.candidateID = hiree_skills.candidateID
```

```
    inner join skill_list ON skill_list.skillid = hiree_skills.skillID
```

```
WHERE hiree_skills.skillID = '005';
```

candidateid character(4)	candidate_first character(20)	candidate_last character(20)	skill_description character(200)
1000	Alan	Labouseur	Management expertise
1004	James	Bond	Management expertise

Zipcode_Matches: This view is a combination of three smaller views (Employer_Zipcode, Agency_Zipcode, Hiree_Zipcode) and displays any employer, agency, or hiree that have matching zipcodes. The purpose of this view is to see which entities are in the same area for better placement and business decisions.

1) Employer_Zipcode: Pulls all employer names and zipcodes.

```
CREATE OR REPLACE VIEW Employer_Zipcode AS  
SELECT employers.client_name, addresses.zipcode  
FROM employers inner join addresses ON employers.addressID = addresses.addressID;
```

2) Agency_Zipcode: Pulls all Agency names and zipcodes.

```
CREATE OR REPLACE VIEW Agency_Zipcode AS  
SELECT agencies.agency_name, addresses.zipcode  
FROM agencies inner join addresses on agencies.addressID = addresses.addressID;
```

3) Hiree_Zipcode: Pulls all hirees names and zipcodes.

```
CREATE OR REPLACE VIEW Hiree_Zipcode AS  
SELECT CONCAT(hirees.candidate_first,hirees.candidate_last) AS FullName, addresses.zipcode  
FROM hirees inner join addresses on hirees.addressID = addresses.addressID;
```

***COMBO VIEW - Zipcode_Matches:** Uses the three above views to pull only the entity names that have matching zipcodes.

```
CREATE OR REPLACE VIEW Zipcode_Matches AS  
SELECT a.zipcode, client_name, agency_name, fullname  
FROM Employer_Zipcode e inner join Agency_Zipcode a on e.zipcode = a.zipcode  
inner join Hiree_Zipcode h on e.zipcode = h.zipcode;
```

zipcode text	client_name text	agency_name text	fullname text
10005	Deutsche Bank	Minion Consulting	Tedd Codd

Count_Zipcode: This view was simply created for the report 'count_by_zipcode' below.

```
CREATE OR REPLACE VIEW Count_Zipcode AS
```

```
select distinct zipcode, fullname
```

```
from Zipcode_Matches
```

```
union
```

```
select distinct zipcode, agency_name
```

```
from Zipcode_Matches
```

```
union
```

```
select distinct zipcode, client_name
```

```
from Zipcode_Matches;
```

zipcode text	fullname text
10005	Tedd Codd
10005	Blackrock
10005	Deutsche Bank
10005	Forum Employment
10005	Lisa Conroy
10005	Minion Consulting

STORED PROCEDURES

get_entities_by_zipcode: this stored procedure allows the user to input a zipcode, or part of a zipcode, and it returns all the entities with that zipcode (or similar).

```
CREATE OR REPLACE FUNCTION get_entities_by_zipcode(TEXT, REFCURSOR) returns REFCURSOR as
```

```
$$
```

```
declare
```

```

zip TEXT                := $1;

resultset REFCURSOR     := $2;

begin

  open resultset for

    SELECT zipcode, FullName

    FROM Hiree_Zipcode

    WHERE zipcode LIKE zip;

return resultset;

end;

$$

language plpgsql;

SELECT get_entities_by_zipcode('12%', 'results');

Fetch all from results;

```

zipcode text	fullname text	
12601	Alan	Labouseur
12524	Sam	Smith

TRIGGERS

balance_check: this trigger connected to the 'Openings' table fires when a number in the fee_payedUSD column is updated or inserted AND is higher than the number in the fee_chargedUSD column. This is to ensure that employees cannot note a payment higher than what they charge, or overcharging, clients for their service.

CREATE OR REPLACE FUNCTION balance_check() RETURNS trigger as

\$\$

begin

IF new.fee_payedUSD > fee_chargedusd from Openings

WHERE openID = '007'

THEN RAISE NOTICE 'fee payed cannot be higher than fee charged';

RETURN NULL;

END IF;

end;

\$\$

language plpgsql;

NOTICE: fee payed cannot be higher than fee charged

CREATE TRIGGER balance_check AFTER INSERT OR UPDATE ON OPENINGS

FOR EACH ROW EXECUTE PROCEDURE balance_check();

REPORTS

count_by_zipcode: this report allows the user to input a zipcode, and it returns the number of matches it has. This can be used in conjunction with the stored procedure 'get_entities_by_zipcode' to get the remaining information once a GETAJOB employee sees there is a match.

CREATE OR REPLACE FUNCTION count_by_zipcode(TEXT, REFCURSOR) returns REFCURSOR as

\$\$

declare

zip TEXT := \$1;

resultset REFCURSOR := \$2;

begin

open resultset for

SELECT COUNT(zipcode)

FROM Count_Zipcode

WHERE zipcode = zip;

return resultset;

end;

\$\$

language plpgsql;

select count_by_zipcode('10005','results');

Fetch all from results;

count bigint
6

SECURITY

The purpose of this section is to grant, define, and identify database privileges to the following roles that will be using the database for everyday processes.

Administrators are granted access to all capabilities on all tables:

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

Staff members can select, insert, and update all tables:

```
CREATE ROLE Staff;  
  
GRANT SELECT, INSERT, UPDATE ON ALL TABLES  
  
IN SCHEMA PUBLIC  
  
TO Staff;
```

Clients can select on granted tables and they cannot insert or delete anything within the database to ensure referential integrity and information confidentiality.

```
CREATE ROLE Clients;
```

```
GRANT SELECT ON Hirees IN SCHEMA PUBLIC TO Clients;
```

```
GRANT SELECT ON Hiree_Skills TO Clients;
```

```
GRANT SELECT ON Employers TO Clients;
```

IMPLEMENTATION NOTES

- The 'Hirees' table only includes candidates within Agency 001 (GETAJOB Consultants) since having candidate information within different agencies is a violation of confidentiality laws.
- When a finance company gives the agency a job opening, it supplies GETAJOB Consultants with skills that the job entails, that's how agency staff pairs the job opening with skills from the skill list.

KNOWN PROBLEMS & FUTURE ENHANCEMENTS

- Within the SQL CREATE statements, use the SERIAL data type for primary keys so users do not have to manually write new ones.
- Due to the scope of the project, if one was to enhance this database for realistic use, they can add more attributes such as 'Payment Type' for Openings when fees are payed, Social Security Numbers for Hirees, add more complicated skills and job descriptions, and so on and so forth.
- In order for the trigger to work, one must pass the OpenID number into the stored procedure as well as the update/insert statement. In the future, one should be able to initially pass in the parameter and fee payed and not change the stored procedure.