Team 5
Big Data Management
Prof. Garg
Names: Prince Musonerwa, Jingyi Huang, Foster Mosden, Sunday Nwanyim

Part 1:

1.1 SQL queries and first 10 rows of results.

```
-- Create a cosine similarity table
CREATE TABLE IF NOT EXISTS cosine_similarity (
  user_A INT,
  user_B INT,
  similarity_score FLOAT,
  PRIMARY KEY(user_A, user_B)
);


INSERT INTO cosine_similarity(user_A, user_B, similarity_score)
WITH CustomerInventoryCounts AS (
```

```
    SELECT
        customer_id,
        inventory_id,
        COUNT(*) as cnt
    FROM rental
    GROUP BY customer_id, inventory_id
),
-- Get the common counts
PairwiseCommonCounts AS (
    SELECT
        a.customer_id AS user_A,
        b.customer_id AS user_B,
        COUNT(*) AS common_count
    FROM CustomerInventoryCounts a
    JOIN CustomerInventoryCounts b
    ON a.inventory_id = b.inventory_id
    WHERE a.customer_id < b.customer_id
    GROUP BY a.customer_id, b.customer_id
),
-- Get the counts for each customer
CustomerDistinctCounts AS (
    SELECT
        customer_id,
        COUNT(DISTINCT inventory_id) as distinct_count
    FROM rental
    GROUP BY customer_id
),

AllPairs AS (
    SELECT
        c1.customer_id AS user_A,
        c2.customer_id AS user_B
    FROM CustomerDistinctCounts c1
    CROSS JOIN CustomerDistinctCounts c2
    WHERE c1.customer_id < c2.customer_id
)
SELECT
    ap.user_A,
    ap.user_B,
    -- checks if the pairwisecommon function is not null, if yes returns
the pcc.common_ount
    COALESCE(pcc.common_count, 0) / (SQRT(cdc1.distinct_count *
cdc2.distinct_count)) AS similarity_score
```

```
FROM AllPairs ap
LEFT JOIN PairwiseCommonCounts pcc ON ap.user_A = pcc.user_A AND ap.user_B
= pcc.user_B
JOIN CustomerDistinctCounts cdc1 ON ap.user_A = cdc1.customer_id
JOIN CustomerDistinctCounts cdc2 ON ap.user_B = cdc2.customer_id;
```

Table result

```
64 •    select * from cosine_similarity limit 10;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

user_A	user_B	similarity_score
1	2	0
1	3	0
1	4	0
1	5	0
1	6	0
1	7	0
1	8	0
1	9	0
1	10	0
1	11	0

1.2 SQL queries and result table with user_id, first_name, film_id, and film title.

```
-- Write a function that takes a user_A id and return the User_B with the
highest cosine similarity

DELIMITER $$
CREATE FUNCTION get_User_B(user_A_id INT) RETURNS INT
BEGIN
    DECLARE user_b_highest_score INT;
    DECLARE sim_score FLOAT;
    SELECT user_B, similarity_score into user_b_highest_score, sim_score
      FROM cosine_similarity
      WHERE user_A = user_A_id
      AND similarity_score = (
            SELECT MAX(similarity_score)
            FROM cosine_similarity
            WHERE user_A = user_A_id
      );
```

```
    RETURN user_b_highest_score;
END $$
DELIMITER ;


-- Create a procedure that recommend a movie to user A
DELIMITER $$
CREATE PROCEDURE recommend_movies(IN user_A_id INT)
BEGIN
    DECLARE user_b_highest_score INT;
    DECLARE user_A_first_name VARCHAR(255);

    -- Fetch the first name of user_A for display
    SELECT first_name INTO user_A_first_name FROM customer WHERE
customer_id = user_A_id;

    -- Call the function get_User_B to return the user_B with the highest
similarity score
    SET user_b_highest_score = get_User_B(user_A_id);
    SELECT user_A_id as 'User A Id', user_A_first_name AS 'User A First
Name', f.film_id, f.title, r.inventory_id
      FROM rental r
    JOIN customer c ON r.customer_id = c.customer_id
      JOIN inventory i ON r.inventory_id = i.inventory_id
      JOIN film f ON i.film_id = f.film_id
      LEFT JOIN rental r1 ON r.inventory_id = r1.inventory_id AND
r1.customer_id = user_A_id
      WHERE r.customer_id = user_b_highest_score
      AND r1.inventory_id IS NULL ORDER BY f.title LIMIT 1;
END $$
DELIMITER ;

-- call the stored procedure
CALL recommend_movies(1);
```

Table result

```
118     -- call the stored procedure
119 •   CALL recommend_movies(1);
120
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	User A Id	User A First Name	film_id	title	inventory_id
▶	1	MARY	9	ALABAMA DEVIL	42

Part 2:

2.1 SQL queries to load the data and first 5 rows of each table

SQL queries to load the data

```
SET GLOBAL local_infile=1;

DROP SCHEMA IF EXISTS yelp;
CREATE SCHEMA yelp;
USE yelp;

DROP TABLE IF EXISTS yelp_business;
CREATE TABLE `yelp_business` (
    `business_id` VARCHAR(255) PRIMARY KEY NOT NULL,
    `name` TEXT,
    `neighborhood` TEXT,
    `address` TEXT,
    `city` TEXT,
    `state` TEXT,
    `postal_code` VARCHAR(255) DEFAULT NULL,
    `latitude` DECIMAL(14,12),
    `longitude` DOUBLE,
    `stars` DOUBLE,
    `review_count` INT,
    `is_open` INT,
```

```
    `categories` TEXT
);
LOAD DATA LOCAL INFILE
"C:\\Users\\d_mos\\Downloads\\Yelp\\yelp_business.csv"
INTO TABLE yelp_business
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;

DROP TABLE IF EXISTS yelp_business_attributes;
CREATE TABLE `yelp_business_attributes` (
    `business_id` VARCHAR(255) PRIMARY KEY NOT NULL,
    `AcceptsInsurance` TEXT,
    `ByAppointmentOnly` TEXT,
    `BusinessAcceptsCreditCards` TEXT,
    `BusinessParking_garage` TEXT,
    `BusinessParking_street` TEXT,
    `BusinessParking_validated` TEXT,
    `BusinessParking_lot` TEXT,
    `BusinessParking_valet` TEXT,
    `HairSpecializesIn_coloring` TEXT,
    `HairSpecializesIn_africanamerican` TEXT,
    `HairSpecializesIn_curly` TEXT,
    `HairSpecializesIn_perms` TEXT,
    `HairSpecializesIn_kids` TEXT,
    `HairSpecializesIn_extensions` TEXT,
    `HairSpecializesIn_asian` TEXT,
    `HairSpecializesIn_straightperms` TEXT,
    `RestaurantsPriceRange2` TEXT,
    `GoodForKids` TEXT,
    `WheelchairAccessible` TEXT,
    `BikeParking` TEXT,
    `Alcohol` TEXT,
    `HasTV` TEXT,
    `NoiseLevel` TEXT,
    `RestaurantsAttire` TEXT,
    `Music_dj` TEXT,
    `Music_background_music` TEXT,
    `Music_no_music` TEXT,
```

```
    `Music_karaoke` TEXT,
    `Music_live` TEXT,
    `Music_video` TEXT,
    `Music_jukebox` TEXT,
    `Ambience_romantic` TEXT,
    `Ambience_intimate` TEXT,
    `Ambience_classy` TEXT,
    `Ambience_hipster` TEXT,
    `Ambience_divey` TEXT,
    `Ambience_touristy` TEXT,
    `Ambience_trendy` TEXT,
    `Ambience_upscale` TEXT,
    `Ambience_casual` TEXT,
    `RestaurantsGoodForGroups` TEXT,
    `Caters` TEXT,
    `WiFi` TEXT,
    `RestaurantsReservations` TEXT,
    `RestaurantsTakeOut` TEXT,
    `HappyHour` TEXT,
    `GoodForDancing` TEXT,
    `RestaurantsTableService` TEXT,
    `OutdoorSeating` TEXT,
    `RestaurantsDelivery` TEXT,
    `BestNights_monday` TEXT,
    `BestNights_tuesday` TEXT,
    `BestNights_friday` TEXT,
    `BestNights_wednesday` TEXT,
    `BestNights_thursday` TEXT,
    `BestNights_sunday` TEXT,
    `BestNights_saturday` TEXT,
    `GoodForMeal_dessert` TEXT,
    `GoodForMeal_latenight` TEXT,
    `GoodForMeal_lunch` TEXT,
    `GoodForMeal_dinner` TEXT,
    `GoodForMeal_breakfast` TEXT,
    `GoodForMeal_brunch` TEXT,
    `CoatCheck` TEXT,
    `Smoking` TEXT,
    `DriveThru` TEXT,
```

```
    `DogsAllowed` TEXT,
    `BusinessAcceptsBitcoin` TEXT,
    `Open24Hours` TEXT,
    `BYOBCorkage` TEXT,
    `BYOB` TEXT,
    `Corkage` TEXT,
    `DietaryRestrictions_dairy-free` TEXT,
    `DietaryRestrictions_gluten-free` TEXT,
    `DietaryRestrictions_vegan` TEXT,
    `DietaryRestrictions_kosher` TEXT,
    `DietaryRestrictions_halal` TEXT,
    `DietaryRestrictions_soy-free` TEXT,
    `DietaryRestrictions_vegetarian` TEXT,
    `AgesAllowed` TEXT,
    `RestaurantsCounterService` TEXT,
    FOREIGN KEY (`business_id`) REFERENCES
`yelp_business`(`business_id`)
);
LOAD DATA LOCAL INFILE
"C:\\Users\\d_mos\\Downloads\\Yelp\\yelp_business_attributes.csv"
INTO TABLE yelp_business_attributes
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

DROP TABLE IF EXISTS yelp_business_hours;
CREATE TABLE `yelp_business_hours` (
    `business_id` VARCHAR(255) PRIMARY KEY NOT NULL,
    `monday` TEXT,
    `tuesday` TEXT,
    `wednesday` TEXT,
    `thursday` TEXT,
    `friday` TEXT,
    `saturday` TEXT,
    `sunday` TEXT,
    FOREIGN KEY (`business_id`) REFERENCES
`yelp_business`(`business_id`)
);
LOAD DATA LOCAL INFILE
```

```
"C:\\Users\\d_mos\\Downloads\\Yelp\\yelp_business_hours.csv"
INTO TABLE yelp_business_hours
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

DROP TABLE IF EXISTS yelp_checkin;
CREATE TABLE `yelp_checkin` (
    `business_id` VARCHAR(255) NOT NULL,
    `weekday` VARCHAR(3) NOT NULL,
    `hour` TIME NOT NULL,
    `checkins` INT,
    PRIMARY KEY (`business_id`, `weekday`, `hour`),
    FOREIGN KEY (`business_id`) REFERENCES
`yelp_business`(`business_id`)
);
LOAD DATA LOCAL INFILE
"C:\\Users\\d_mos\\Downloads\\Yelp\\yelp_checkin.csv"
INTO TABLE yelp_checkin
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

DROP TABLE IF EXISTS yelp_user;
CREATE TABLE `yelp_user` (
     `user_id` VARCHAR(255) PRIMARY KEY NOT NULL,
     `name` text,
    `review_count` int,
    `yelping_since` text,
    `friends` LONGTEXT,
    `useful` int,
    `funny` int,
    `cool` int,
    `fans` int,
    `elite` text,
    `average_stars` double,
    `compliment_hot` int,
    `compliment_more` int,
    `compliment_profile` int,
```

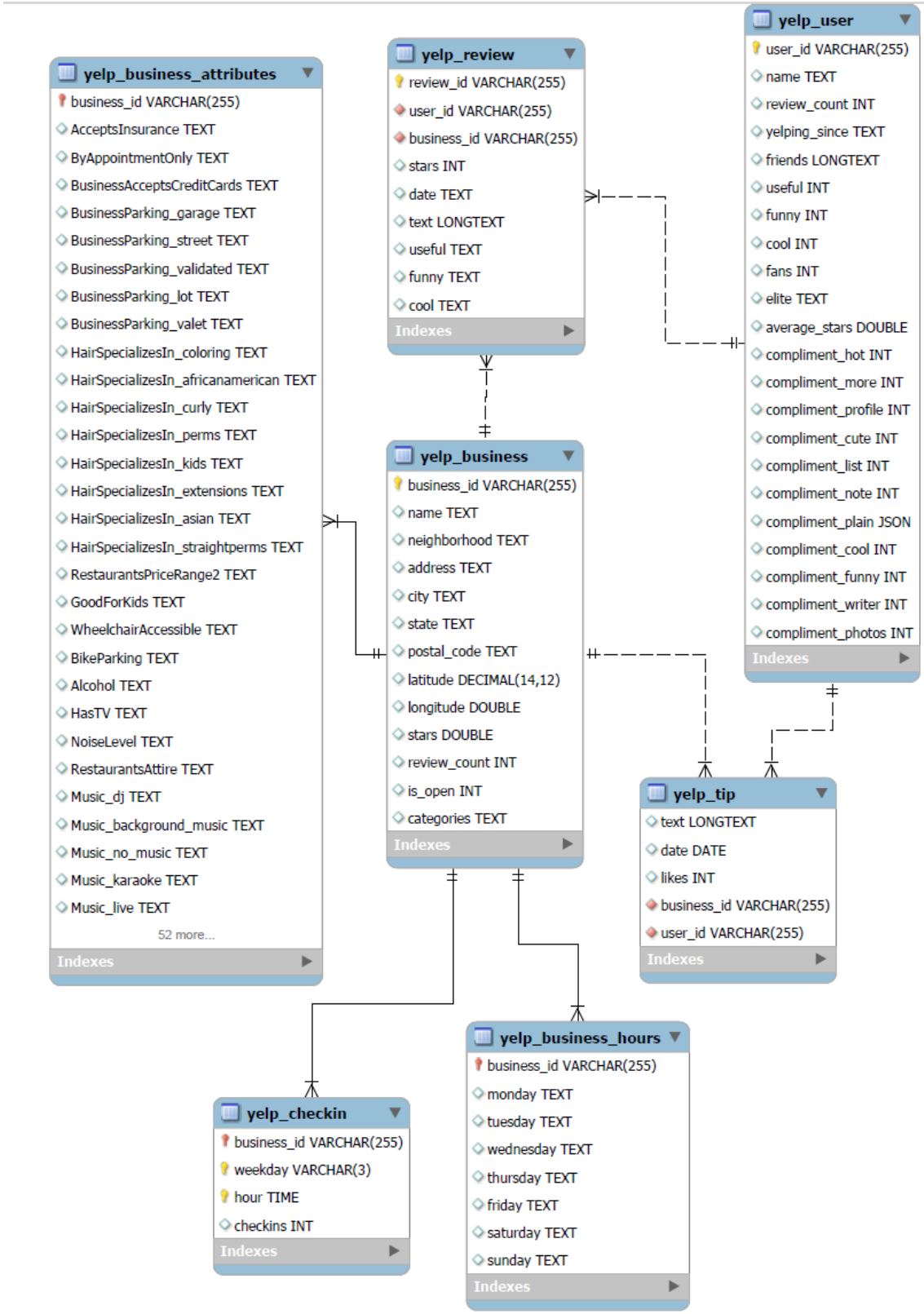
```
    `compliment_cute` int,
    `compliment_list` int,
    `compliment_note` int,
    `compliment_plain` json,
    `compliment_cool` int,
    `compliment_funny` int,
    `compliment_writer` int,
    `compliment_photos` int
);
LOAD DATA LOCAL INFILE
"C:\\Users\\d_mos\\Downloads\\Yelp\\yelp_user.csv"
INTO TABLE yelp_user
FIELDS TERMINATED BY ','
ENCLOSED BY '\"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;

SET @@global.sql_mode= '';
DROP TABLE IF EXISTS yelp_review;
CREATE TABLE `yelp_review` (
    `review_id` VARCHAR(255) PRIMARY KEY NOT NULL,
    `user_id` VARCHAR(255) NOT NULL,
    `business_id` VARCHAR(255) NOT NULL,
    `stars` INT NULL,
    `date` TEXT NULL,
    `text` LONGTEXT NULL,
    `useful` TEXT NULL,
    `funny` TEXT NULL,
    `cool` TEXT NULL,
    FOREIGN KEY (`user_id`) REFERENCES `yelp_user`(`user_id`),
    FOREIGN KEY (`business_id`) REFERENCES
`yelp_business`(`business_id`)
);
LOAD DATA LOCAL INFILE
"C:\\Users\\d_mos\\Downloads\\Yelp\\yelp_review.csv"
INTO TABLE yelp_review
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
```

```
IGNORE 1 LINES;

DROP TABLE IF EXISTS yelp_tip;
CREATE TABLE `yelp_tip` (
    `text` LONGTEXT DEFAULT NULL,
    `date` DATE DEFAULT NULL,
    `likes` INT DEFAULT NULL,
    `business_id` VARCHAR(255) NOT NULL,
    `user_id` VARCHAR(255) NOT NULL,
    FOREIGN KEY (`business_id`) REFERENCES
`yelp_business`(`business_id`),
    FOREIGN KEY (`user_id`) REFERENCES `yelp_user`(`user_id`)
);
LOAD DATA LOCAL INFILE
"C:\\Users\\d_mos\\Downloads\\Yelp\\yelp_tip.csv"
INTO TABLE yelp_tip
FIELDS TERMINATED BY ','
ENCLOSED BY '\"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

first 5 rows of each table



1.Yelp_business:

```
11 •  USE yelp;
12
13 •  SELECT * FROM yelp_business LIMIT 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

business_id	name	neighborhood	address	city	state	postal_code	latitude	longitude	stars	review_count	is_open	categories
__1uG7MLxWGFIv2fCGPiQQ	"SpinalWorks Chiropractic"		"15640 N 7th St, Ste A3"	Phoenix	AZ	85022	33.628854000000	-112.0659754	5	26	1	Physical Therapy;Chiropractors;
__3I-DDkqM9XjLH1cJl3VA	"Montallegro Barber Shop"	Villeray-Saint-Michel-Parc-Extension	"7244 Rue Hutchison"	Montreal	QC	H3N 1Z1	45.529863700000	-73.6237259	5	13	1	Hair Salons;Barbers;Beauty & Sp
__3qOwWFBUE8mdOToI7YrQ	"Custom Kings"	Southeast	""	Las Vegas	NV	88901	36.055656949200	-115.169422094	1	12	1	Screen Printing/T-Shirt Printing;U
__47_7H-yK3HChO5vyut_Q	"Instant Muffler and Autorepair"		"1295 Weston Road"	York	ON	M6M 4R2	43.689236600000	-79.4952863	1	3	1	Auto Repair;Automotive
__6jYJ6Hm-Qq8XQEGDrOGQ	"Winfield Gene DO"		"2121 S Mill Ave"	Tempe	AZ	85282	33.405593800000	-111.9394369	4	4	1	Doctors;Health & Medical

2.Yelp_business_attributes:

```
11 •  USE yelp;
12
13 •  SELECT * FROM yelp_business LIMIT 5;
14 •  SELECT * FROM yelp_business_attributes LIMIT 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

business_id	AcceptsInsurance	ByAppointmentOnly	BusinessAcceptsCreditCards	BusinessParking_garage	BusinessParking_street	BusinessParking_validated	BusinessParking_lot	BusinessParking_valet	HairSpecializesIn_coloring	HairSpecializ
__1uG7MLxWGFIv2fCGPiQQ	Na	Na	Na	True	Na	Na	Na	Na	Na	Na
__3I-DDkqM9XjLH1cJl3VA	Na	Na	False	Na	Na	Na	Na	Na	Na	Na
__3qOwWFBUE8mdOToI7YrQ	Na	Na	Na	True	Na	Na	Na	Na	Na	Na
__6jYJ6Hm-Qq8XQEGDrOGQ	Na	Na	True	Na	Na	Na	Na	Na	Na	Na
__8j8yhsmE98wNWHJNyAgw	Na	Na	Na	Na	False	False	False	False	False	Na

3.Yelp_business_hours:

```
11 •  USE yelp;
12
13 •  SELECT * FROM yelp_business LIMIT 5;
14 •  SELECT * FROM yelp_business_attributes LIMIT 5;
15 •  SELECT * FROM yelp_business_hours LIMIT 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

business_id	monday	tuesday	wednesday	thursday	friday	saturday	sunday
__1uG7MLxWGFIv2fCGPiQQ	7:0-11:0	14:0-18:0	14:0-18:0	14:0-18:0	7:0-11:0	8:0-9:0	None
__3I-DDkqM9XjLH1cJl3VA	8:0-19:0	8:0-19:0	8:0-19:0	8:0-19:0	8:0-19:0	8:0-19:0	8:0-19:0
__3qOwWFBUE8mdOToI7YrQ	None	None	None	None	None	None	None
__47_7H-yK3HChO5vyut_Q	None	None	None	None	None	None	None
__6jYJ6Hm-Qq8XQEGDrOGQ	None	None	None	None	None	None	None

4.Yelp_checkin:

```
11 •  USE yelp;
12
13 •  SELECT * FROM yelp_business LIMIT 5;
14 •  SELECT * FROM yelp_business_attributes LIMIT 5;
15 •  SELECT * FROM yelp_business_hours LIMIT 5;
16 •  SELECT * FROM yelp_checkin LIMIT 5;
```

Result Grid | Filter Rows: | Edit:

business_id	weekday	hour	checkins
__1uG7MLxWGFIv2fCGPiQQ	Fri	00:00:00	1
__1uG7MLxWGFIv2fCGPiQQ	Fri	14:00:00	1
__1uG7MLxWGFIv2fCGPiQQ	Fri	16:00:00	2
__1uG7MLxWGFIv2fCGPiQQ	Fri	17:00:00	1
__1uG7MLxWGFIv2fCGPiQQ	Mon	00:00:00	1

5.Yelp_review:

```
11 •  USE yelp;
12
13 •  SELECT * FROM yelp_business LIMIT 5;
14 •  SELECT * FROM yelp_business_attributes LIMIT 5;
15 •  SELECT * FROM yelp_business_hours LIMIT 5;
16 •  SELECT * FROM yelp_checkin LIMIT 5;
17 •  SELECT * FROM yelp_review LIMIT 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

review_id	user_id	business_id	stars	date	text	useful	funny	cool
___-Bw8LtQgezPiN9xJWaQ	lMkqkljZsQ1pmOZb_bbP8A	jCNBZnkIFv_0omLVTgNR6Q	5	2017-10-11	Don't know how I missed this place after so man...	0	0	0
__0SrSAAHBiM7XAbXsW-A	KPPOpDFYO5HBOVOdFgSDWw	9Q1ZtzTPFWG4fJiFSko5Xg	4	2011-09-21	I visited Cantina Laredo for a Sunday Brunch an...	0	0	0
__0XFGhjOU1H8Y3cVYjMA	OsFWc7PMDDACG9MMit7kGQ	oWboXKe_xk6Vcr2gBEuxuw	2	2014-05-11	Be aware. There is an extremely limited menu. ...	0	3	0
__3SR6DPz0F6gLBxxjuVw	vHF4LqmMRkhrLD5FE-8HXA	TgEKtJGC-cN9rrCKgSDx8g	5	2017-09-23	This place is amazing. The strawberry cheesec...	0	0	0
___4_AFJm_fOE-HTgPDxjw	6mn-M3f75hdynz245p-fBA	q7MorRPzU_J-iekeDKUKgw	5	2011-02-02	i really liked this place.. tequila bar!! how aweso...	3	0	3

6.yelp_tip:

```
11 •  USE yelp;
12
13 •  SELECT * FROM yelp_business LIMIT 5;
14 •  SELECT * FROM yelp_business_attributes LIMIT 5;
15 •  SELECT * FROM yelp_business_hours LIMIT 5;
16 •  SELECT * FROM yelp_checkin LIMIT 5;
17 •  SELECT * FROM yelp_review LIMIT 5;
18 •  SELECT * FROM yelp_tip LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

text	date	likes	business_id	user_id
Great breakfast large portions and friendly wait...	2015-08-12	0	jH19V2I9fIslnNhDzPmdkA	ZcLKXikTHYOnYt5VYRO5sg
Nice place. Great staff. A fixture in the townshi...	2014-06-20	0	dAa0hB2yrnHzVmsCkN4YvQ	oaYhjqBbh18ZhU0bpyzSuw
Happy hour 5-7 Monday - Friday	2016-10-12	0	dAa0hB2yrnHzVmsCkN4YvQ	ulQ8Nyj7jCUR8M83SUMoRQ
Parking is a premium, keep circling, you will eve...	2017-01-28	0	ESzO3Av0b1_TzKOiqzbQYQ	ulQ8Nyj7jCUR8M83SUMoRQ
Homemade pasta is the best in the area	2017-02-25	0	k7WRPbDd7rztjHcGGkEjlw	ulQ8Nyj7jCUR8M83SUMoRQ

7. Yelp_user:

```
11 •  USE yelp;
12
13 •  SELECT * FROM yelp_business LIMIT 5;
14 •  SELECT * FROM yelp_business_attributes LIMIT 5;
15 •  SELECT * FROM yelp_business_hours LIMIT 5;
16 •  SELECT * FROM yelp_checkin LIMIT 5;
17 •  SELECT * FROM yelp_review LIMIT 5;
18 •  SELECT * FROM yelp_tip LIMIT 5;
19 •  SELECT * FROM yelp_user LIMIT 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

user_id	name	review_count	yelping_since	friends	useful	funny	cool	fans	elite	average_stars	compliment_hot	compliment_more	compliment_profile	compliment_cute	complime
__DPmKJsBF2X6ZKgAeGqg	Charlotte	3	2014-04-07	None	0	0	1	0	None	3.33	0	0	0	0	0
__fEWlObjtPaZ-pK0eq9g	Nerissa	7	2016-05-09	Ngwaot7XkD4g75hHBY3wnQ	0	0	0	0	None	5	0	0	0	0	0
__I9ZYdYGkZ6dMYxwJEIQ	Jim	168	2011-08-17	JWrq6BEnAFoRZ4zDmlB8Yg, hOE5zGo6HVQGn...	140	45	117	9	None	3.98	1	3	1	1	0
__MTsBloHHjvybJ5DrTYw	TJ	10	2011-03-25	KN6I2UWhB-N2nXtaeLgNsg, R3ZzMoqN3FibAu8...	3	0	0	0	None	4.9	0	0	0	0	0
__QCazm0YrHLd3uNUPYMA	Mike	5	2014-07-31	None	10	3	1	0	None	5	0	0	0	0	0

2.2 SQL queries to ETL the data into new fact and dim tables, and ER model with the Star schema (use reverse engineer)

SQL queries to ETL the data into new fact and dim tables

```
USE yelp;

DROP TABLE IF EXISTS Dim_business;
CREATE TABLE Dim_Business (
```

```
    business_id VARCHAR(255) PRIMARY KEY NOT NULL,
    `name` TEXT,
    neighborhood TEXT,
    address TEXT,
    city TEXT,
    state TEXT,
    postal_code VARCHAR(255),
    latitude DOUBLE,
    longitude DOUBLE,
    stars DOUBLE,
    review_count INT,
    is_open INT,
    categories TEXT
);
INSERT INTO Dim_Business (business_id, name, neighborhood, address, city,
state, postal_code, latitude, longitude, stars, review_count, is_open,
categories)
SELECT business_id, name, neighborhood, address, city, state, postal_code,
latitude, longitude, stars, review_count, is_open, categories
FROM yelp_business;


DROP TABLE IF EXISTS Dim_Checkin;
CREATE TABLE Dim_Checkin (
    checkin_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    business_id VARCHAR(255) NOT NULL,
    `weekday` TEXT,
    hour_start VARCHAR(50) DEFAULT NULL,
    hour_end VARCHAR(50) DEFAULT NULL,
    checkins INT,
    FOREIGN KEY (business_id) REFERENCES Dim_Business(business_id)
);
INSERT INTO Dim_Checkin (business_id, `weekday`, hour_start, hour_end,
checkins)
SELECT
    ybh.business_id,
    CASE
        WHEN ybh.monday IS NOT NULL THEN 'Monday'
        WHEN ybh.tuesday IS NOT NULL THEN 'Tuesday'
        WHEN ybh.wednesday IS NOT NULL THEN 'Wednesday'
        WHEN ybh.thursday IS NOT NULL THEN 'Thursday'
        WHEN ybh.friday IS NOT NULL THEN 'Friday'
        WHEN ybh.saturday IS NOT NULL THEN 'Saturday'
```

```
        WHEN ybh.sunday IS NOT NULL THEN 'Sunday'
    END AS weekday,
    SUBSTRING_INDEX(ybh.monday, '-', 1) AS hour_start,
    COALESCE(NULLIF(SUBSTRING_INDEX(ybh.monday, '-', -1), ''), '24:00') AS
hour_end,
    COUNT(*) AS checkins
FROM
    yelp_business_hours ybh
WHERE
    ybh.monday IS NOT NULL OR
    ybh.tuesday IS NOT NULL OR
    ybh.wednesday IS NOT NULL OR
    ybh.thursday IS NOT NULL OR
    ybh.friday IS NOT NULL OR
    ybh.saturday IS NOT NULL OR
    ybh.sunday IS NOT NULL
GROUP BY
    ybh.business_id,
    weekday,
    hour_start,
    hour_end;

DROP TABLE IF EXISTS Dim_User;
CREATE TABLE Dim_User (
    user_id VARCHAR(255) PRIMARY KEY NOT NULL,
    name TEXT,
    review_count INT,
    yelping_since TEXT,
    friends LONGTEXT,
    useful INT,
    funny INT,
    cool INT,
    fans INT,
    elite TEXT,
    average_stars DOUBLE,
    compliment_hot INT,
    compliment_more INT,
    compliment_profile INT,
    compliment_cute INT,
    compliment_list INT,
    compliment_note INT,
    compliment_plain JSON,
    compliment_cool INT,
```

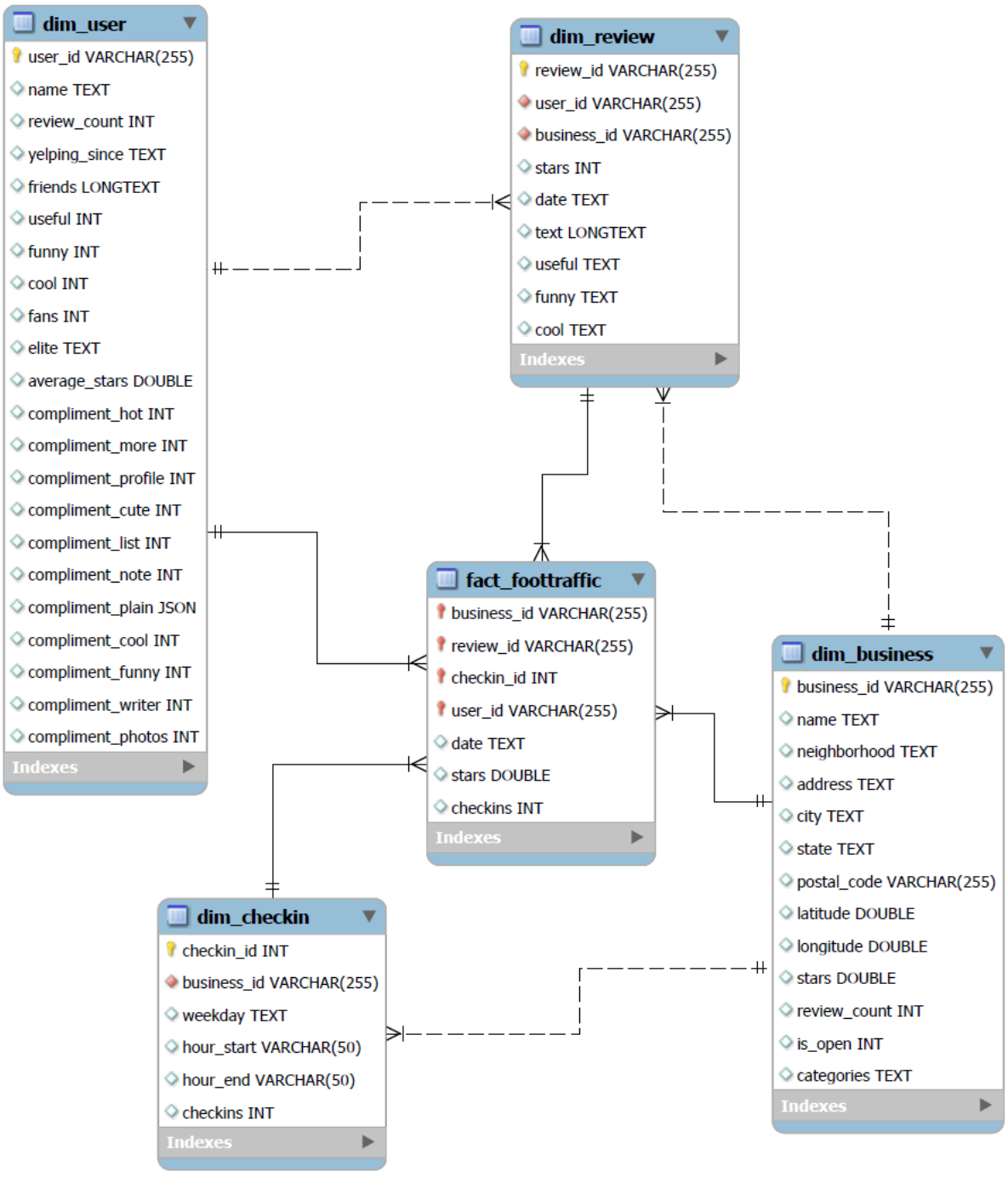
```
    compliment_funny INT,
    compliment_writer INT,
    compliment_photos INT
);
INSERT INTO Dim_User (user_id, `name`, review_count, yelping_since,
friends, useful, funny, cool, fans, elite, average_stars, compliment_hot,
compliment_more, compliment_profile, compliment_cute, compliment_list,
compliment_note, compliment_plain, compliment_cool, compliment_funny,
compliment_writer, compliment_photos)
SELECT user_id, `name`, review_count, yelping_since, friends, useful,
funny, cool, fans, elite, average_stars, compliment_hot, compliment_more,
compliment_profile, compliment_cute, compliment_list, compliment_note,
compliment_plain, compliment_cool, compliment_funny, compliment_writer,
compliment_photos
FROM yelp_user;

DROP TABLE IF EXISTS Dim_Review;
CREATE TABLE Dim_Review (
    review_id VARCHAR(255) PRIMARY KEY NOT NULL,
    user_id VARCHAR(255) NOT NULL,
    business_id VARCHAR(255) NOT NULL,
    stars INT,
    `date` TEXT,
    `text` LONGTEXT,
    useful TEXT,
    funny TEXT,
    cool TEXT,
    FOREIGN KEY (user_id) REFERENCES Dim_User(user_id),
    FOREIGN KEY (business_id) REFERENCES Dim_Business(business_id)
);
INSERT INTO Dim_Review (review_id, user_id, business_id, stars, `date`,
`text`, useful, funny, cool)
SELECT review_id, user_id, business_id, stars, `date`, `text`, useful,
funny, cool
FROM yelp_review;

DROP TABLE IF EXISTS Fact_FootTraffic;
CREATE TABLE Fact_FootTraffic (
    business_id VARCHAR(255) NOT NULL,
    review_id VARCHAR(255) NOT NULL,
    checkin_id INT NOT NULL,
    user_id VARCHAR(255) NOT NULL,
    `date` TEXT,
```

```
    stars DOUBLE,
    checkins INT,
    PRIMARY KEY (business_id, review_id, checkin_id, user_id),
    FOREIGN KEY (business_id) REFERENCES Dim_Business(business_id),
    FOREIGN KEY (review_id) REFERENCES Dim_Review(review_id),
    FOREIGN KEY (checkin_id) REFERENCES Dim_Checkin(checkin_id),
    FOREIGN KEY (user_id) REFERENCES Dim_User(user_id)
);
INSERT INTO Fact_FootTraffic (business_id, review_id, checkin_id, user_id,
`date`, stars, checkins)
SELECT b.business_id, r.review_id, ch.checkin_id, u.user_id, r.`date`,
r.stars, ch.checkins
FROM dim_business b
JOIN dim_review r ON r.business_id = b.business_id
JOIN dim_checkin ch ON ch.business_id = b.business_id
JOIN dim_user u ON u.user_id = r.user_id;
```

ER model with the Star schema (use reverse engineer)



Part 3

We would transition the **review** table to a NoSQL database. For reviews, we are dealing with large volumes of semi-structured and rapidly changing data, horizontal scalability is needed. During a network partition, allowing users to review would preserve their willingness to comment, and allowing different users to see different comments doesn't have severe negative consequences, so we value availability over consistency in the review table.

We would transition the **recommendations** table to a NoSQL database because the data can be distributed across multiple servers to handle high read loads, while occasional inconsistencies can be tolerated.

We would NOT transition the **user information** table to a NoSQL database. Security and data integrity are paramount in user authentication. Relational databases ensure strong consistency and ACID properties to prevent unauthorized access and ensure accurate user data.

We would NOT transition the **finance** table to a NoSQL database. During a network partition, if a business partner withdraws money from Yelp from two separate endpoints and the sum of withdrawals exceeds the actual savings, this would lead to a negative balance for Yelp, so consistency matters more than availability.

We would NOT transition the **employee/HR** table to a NoSQL database because it requires high partition tolerance, the table contains employees' personal and payroll information, so maintaining access to data, even during network failures or system outages is crucial.