CIS550

PENNDASH

Restaurant Review Website

GROUP 33
Prednya Ramesh, Aniket Ghorpade, John Harshith, Tanisha Admane

PROJECT DESCRIPTION

The goal of this group project is to develop PennDash, a user-friendly restaurant review platform focusing on the state of Pennsylvania (PA, USA). The website will enable users to explore restaurants based on various filters, such as type of cuisine, location, ratings, and enable them to bookmark and favorite their desired restaurants.

The platform will incorporate the following key functionalities :

- Detailed restaurant reviews and ratings along with their business information (e.g., Wi-Fi availability, outdoor seating, vehicle parking, etc.)
- Location mapping via Google Maps integration.
- Users can save their favorite restaurants and bookmark them.
- Secure login/logout functionality for an optimized user experience.
- Restaurant analytics from Yelp’s public dataset will be shown to the user.

PROBLEM AND GOAL

Diners in Pennsylvania lack access to platforms providing hyper-local, tailored restaurant recommendations based on preferences and amenities.

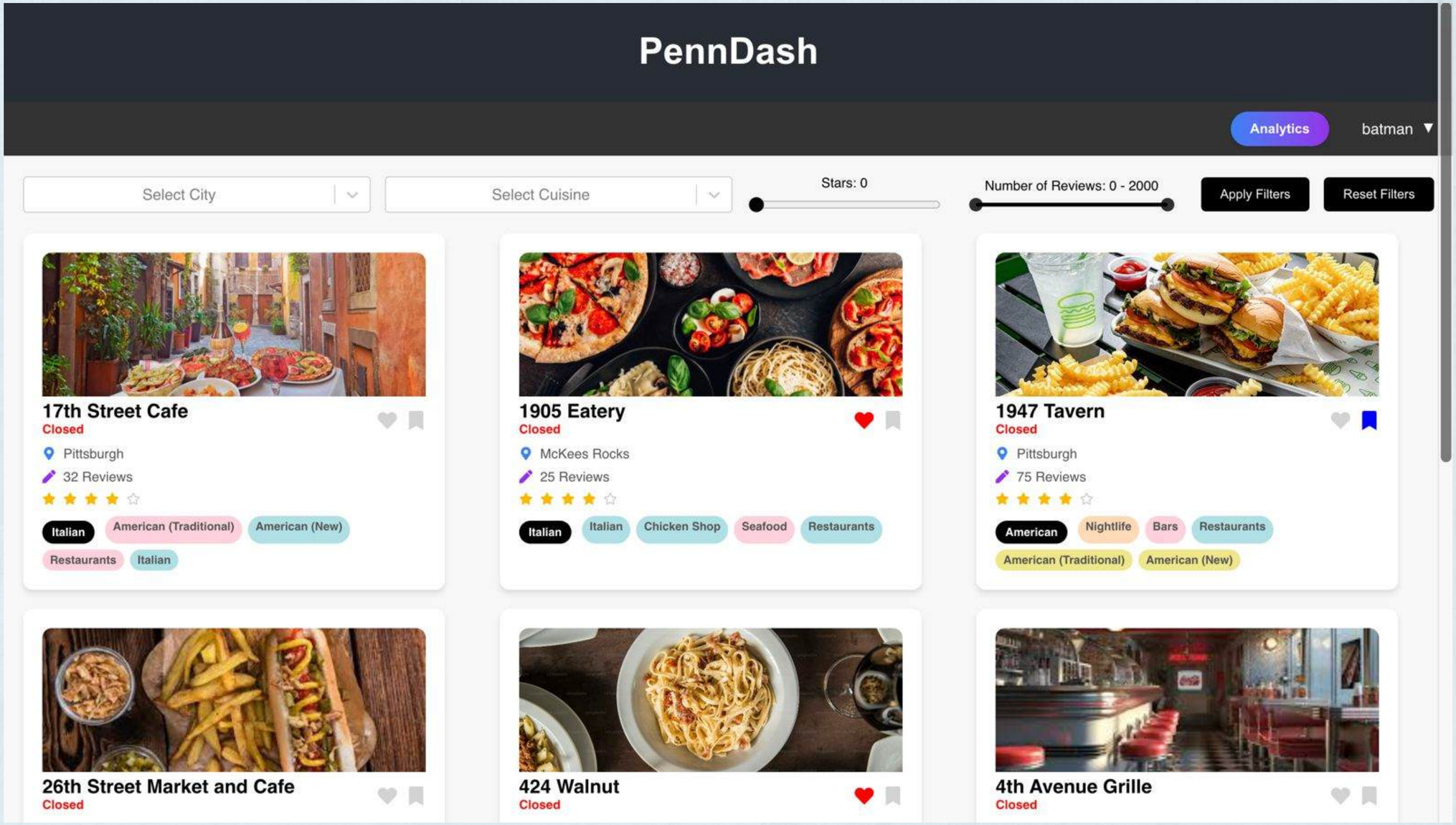
Existing platforms focus on national or global markets but fail to optimize for state-specific data, making it harder for users to find relevant dining options.

PennDash Solution:

A Pennsylvania-specific restaurant recommendation platform that,

- Combines user-friendly filters and trending insights.
- Can bookmark/favorite restaurants for future reference.
- Incorporates navigation and amenity-based reviews.

PREVIEW OF OUR WEBSITE



DATASETS

Dataset Files : *https://www.yelp.com/dataset*

yelp_academic_dataset_business.json

- **Description :** Contains business-level information (e.g., business ID, name, address, category, latitude, longitude, attributes like Wi-Fi or outdoor seating availability).
- **Size :** ~150,000 businesses across multiple regions, including Pennsylvania.
- **Attributes Summary :**
  - **Mean rating :** 3.8
  - **Average review count :** 120

yelp_academic_dataset_review.json

- **Description :** Includes detailed reviews such as review text, rating, and user ID.
- **Size :** 7 million reviews (~5 GB).
- **Attributes Summary :**
  - **Average review length :** 200 words
  - **Star rating distribution :** Normally distributed, with the mean around 3.8

yelp_academic_dataset_user.json

- **Descript. :** Details user profiles, including no. of reviews written and their friends.
- **Size :** 2 million user profiles.
- **Attributes Summary :**
  - **Average useful votes per user :** 4.2
  - **Mean friend count :** 7

yelp_academic_dataset_checkin.json

- **Descript. :** Contains the business ID and timestamps for when users checked in.
- **Size :** ~1.2 million check-ins.

yelp_academic_dataset_tip.json

- **Descript. :** Contains shorter, informal tips by users, offering concise restaurant feedback.
- **Size :** ~2 million tips.

Basic Descriptive Analysis of the Data

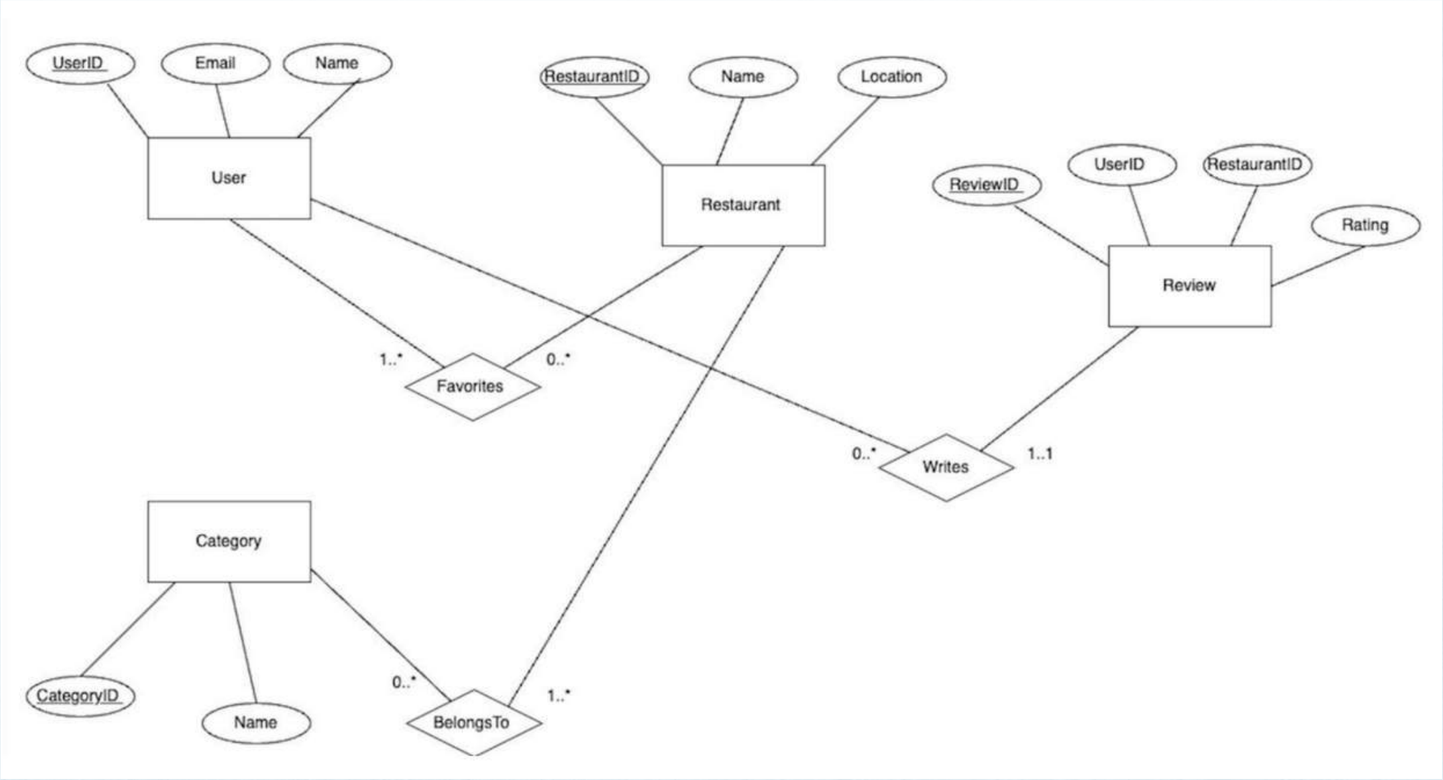
- **Restaurant Types :** Includes diverse categories such as Bakeries, Japanese, Burger joints, Chinese restaurants, Thai, Vietnamese, Sushi Bars and Cafes.

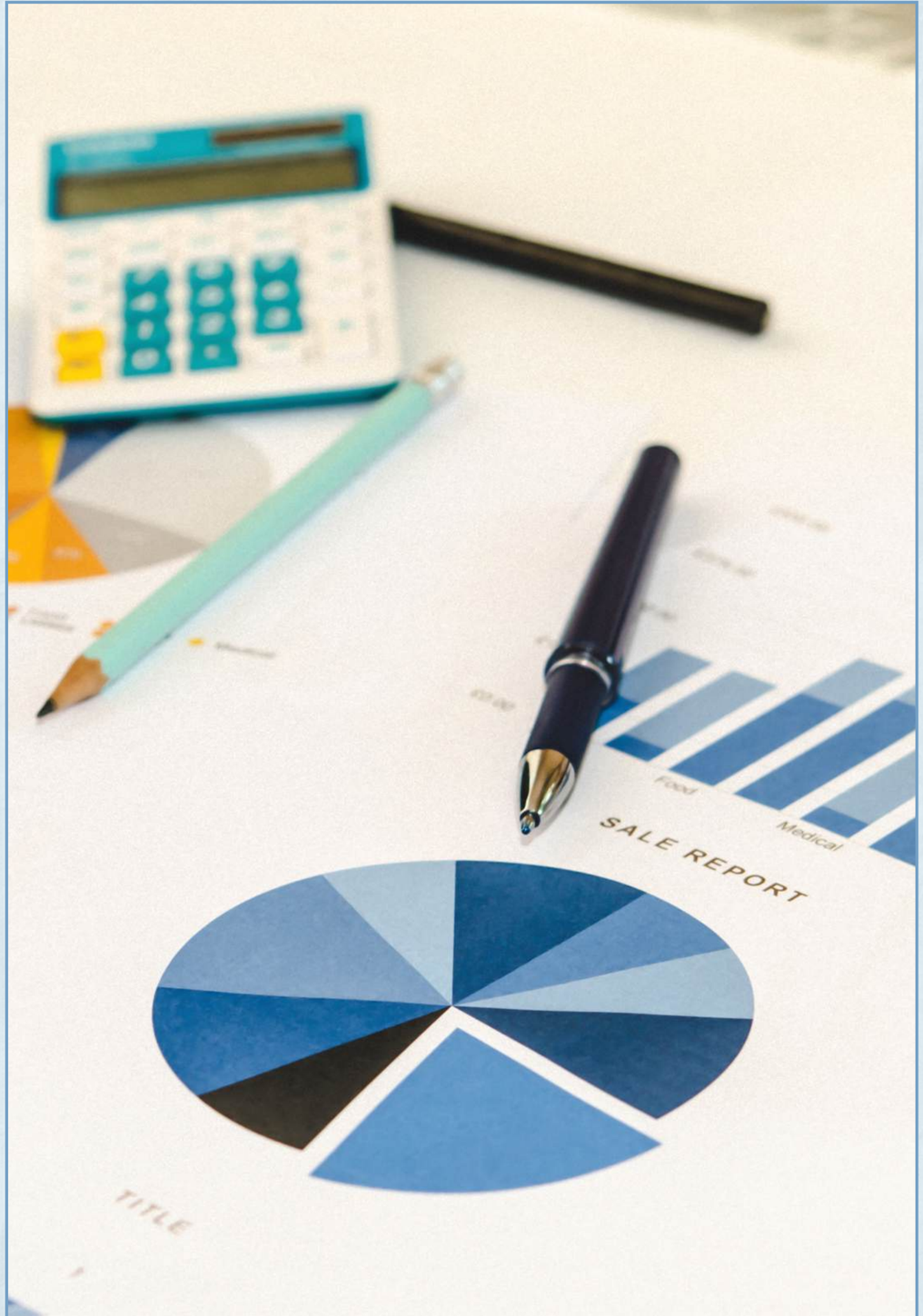
- **Rating Trends :** Distribution shows a concentration around 3 to 5 stars.

- **Regional Breakdown :** The dataset includes multiple US and Canadian states, but we will filter restaurants within Pennsylvania.

SCHEMA DESIGN

The schema will include entities such as User, Restaurant, Review, Category, Check-in, and Tip, with relationships among users, restaurants through reviews, favorites, check-in.





NORMALIZATION

Normalization :

Table Name	Action	Reason
businesses	Changed	Split location data into business_locations to remove redundancy.
business_locations	New	Isolated location-specific data for reusability and integrity.
business_attributes	Changed	Renamed columns for consistency; structure is unchanged.
business_hours	Changed	Normalized days into rows for easier filtering and querying.
checkin	Changed	Added checkin_id for unique identification and changed hour to TIME.
users, user_favorites, user_stats, tips,	Unchanged	Already normalized and structured efficiently.
user_reviews	Changed	Normalized text for better storage and querying.

3NF PROOF

1. Business Table

Functional Dependencies:

- `business_id → name, address, city, state, postal_code, latitude, longitude, stars, review_count, is_open`
- `business_id` is the primary key.
- All non-prime attributes (e.g., name, stars, address) depend only on `business_id`.
- There are no transitive dependencies.

Conclusion: The Business table is in **3NF**.

2. Users Table

Functional Dependencies:

- `user_id → name, review_count, yelping_since, useful, funny, cool, fans, average_stars, compliments`
- `user_id` is the primary key.
- All non-prime attributes depend solely on `user_id`.
- There are no transitive dependencies.

Conclusion: The Users table is in **3NF**.

3. Review Table

Functional Dependencies:

- `review_id → user_id, business_id, stars, useful, funny, cool, text, date`
- `review_id` is the primary key.
- All non-prime attributes depend solely on `review_id`.
- `user_id` and `business_id` are foreign keys, maintaining relationships without introducing transitive dependencies.

Conclusion: The Review table is in **3NF**.

4. Business_Category Table (Many-to-Many Relationship)

Functional Dependencies:

- Composite Key: `(business_id, category_id)`
- No non-prime attributes exist.
- Each row contains only foreign keys linking businesses and categories.

Conclusion: The Business_Category table is in **3NF**.

5. Business_Attributes Table

Functional Dependencies:

- `business_id → AcceptsInsurance, WiFi, RestaurantsTakeOut, GoodForKids, etc.`
- `business_id` is the primary key.
- Each non-prime attribute depends solely on `business_id`.
- No transitive dependencies exist.

Conclusion: The Business_Attributes table is in **3NF**.

3NF PROOF

6. Business_Hours Table

Functional Dependencies:

- Composite Key: `(business_id, day_of_week) → opening_time, closing_time`
- All non-prime attributes depend on the full composite key `(business_id, day_of_week)`.
- There are no transitive dependencies.

Conclusion: The Business_Hours table is in **3NF**.

7. Check-in Table

Functional Dependencies:

- `checkin_id → business_id, date`
- `checkin_id` is the primary key.
- Non-prime attributes depend solely on `checkin_id`.

Conclusion: The Check-in table is in **3NF**.

8. Tip Table

Functional Dependencies:

- `tip_id → user_id, business_id, text, date, compliment_count`
- `tip_id` is the primary key.
- All non-prime attributes depend solely on `tip_id`.
- `user_id` and `business_id` are foreign keys without introducing transitive dependencies.

Conclusion: The Tip table is in **3NF**.

9. Category Table

Functional Dependencies:

- `category_id → category_name`
- `category_id` is the primary key.
- `category_name` is atomic and depends only on `category_id`.

Conclusion: The Category table is in **3NF**.

All tables in the PennDash database schema adhere to Third Normal Form (3NF).

- **Each non-prime attribute depends only on the primary key.**
- **There are no partial dependencies or transitive dependencies.**

EXAMPLE OF A COMPLEX QUERY

Top 5 restaurants reviewed most by users whose friends also reviewed them

```
const query = `
    WITH UserFriendsReviews AS (
      SELECT r.business_id, uf.user_id, COUNT(DISTINCT r.review_id) AS
review_count_by_friends
      FROM normalized_user_reviews r
      JOIN user_stats uf ON r.user_id = uf.user_id
      GROUP BY r.business_id, uf.user_id
    ),
    PopularWithFriends AS (
      SELECT ufr.business_id, SUM(ufr.review_count_by_friends) AS
total_reviews_by_friends
      FROM UserFriendsReviews ufr
      GROUP BY ufr.business_id
      ORDER BY total_reviews_by_friends DESC
      LIMIT 5
    )
    SELECT b.name AS restaurant_name, bl.city as city,
pwf.total_reviews_by_friends
    FROM PopularWithFriends pwf
    JOIN normalized_businesses b ON pwf.business_id = b.business_id
    JOIN business_locations bl ON b.location_id = bl.location_id;
`
```

PERFORMANCE

	ORIGINAL	OPTIMIZED
COMPLEX QUERY 1	5.7sec	0.3sec
COMPLEX QUERY 2	9.3sec	0.5sec
COMPLEX QUERY 3	5.8sec	0.2sec
COMPLEX QUERY 4	5.5sec	0.1sec
COMPLEX QUERY 5	18.3sec	1.1sec

TECHNICAL CHALLENGES

Large Dataset Management	Handling millions of records efficiently required database optimization.
Query Performance	Complex queries (e.g., CTEs, joins) needed optimization for low latency.
Data Cleaning	Preprocessing Yelp data to resolve missing values and inconsistencies.
Feature Prioritization	Balancing core features with time constraints (e.g., Google Maps).

THANK YOU

Feel free to ask any questions!