Table of Contents

Part A: Table Creation

Create relational database schema for the proposed application area from Phase I:

- *Actual SQL create table commands are required.*
- *A graphical diagram of the relations is required.*

User Table

```
CREATE TABLE `users` (
  `Username` varchar(255) NOT NULL,
  `Email` varchar(255) DEFAULT NULL,
  `FirstName` varchar(45) DEFAULT NULL,
  `LastName` varchar(45) DEFAULT NULL,
  `Password` varchar(255) NOT NULL,
  `Role` int NOT NULL,
  `Status` int NOT NULL,
  PRIMARY KEY (`Username`)
);
```

Stocks Table

```
CREATE TABLE `stocks` (
  `Ticker` varchar(10) NOT NULL,
  `Name` varchar(255) NOT NULL,
  PRIMARY KEY (`Ticker`)
);
```

Comments Table

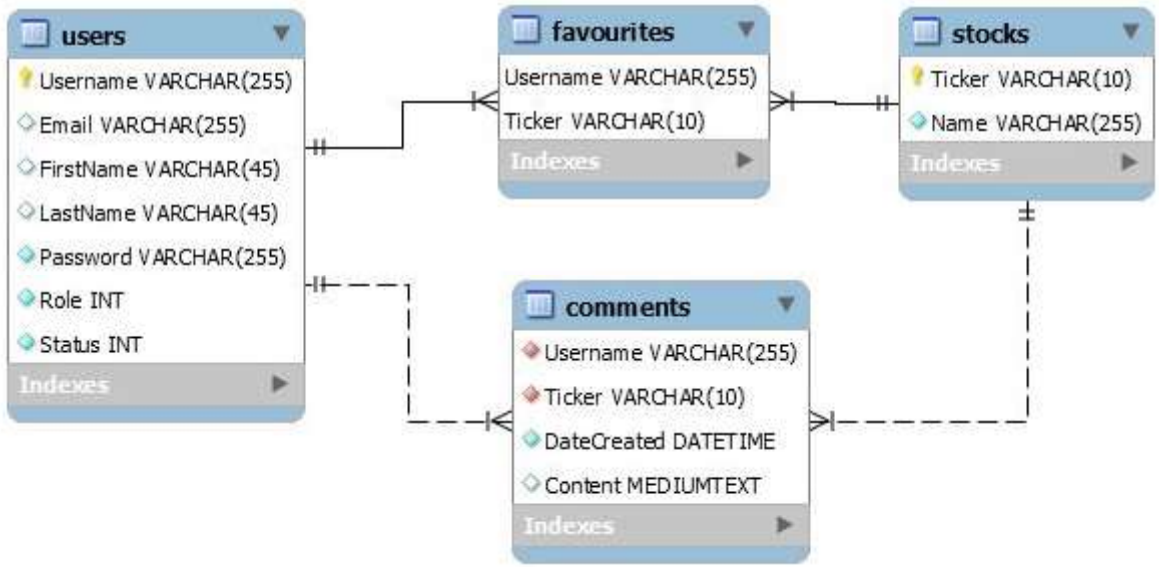
```
CREATE TABLE `comments` (
  `Username` VARCHAR(255) NOT NULL,
  `Ticker` VARCHAR(10) NOT NULL,
  `DateCreated` DATETIME NOT NULL,
  `Content` MEDIUMTEXT NULL,
  INDEX `USER_FKEY_idx` (`Username` ASC) VISIBLE,
  INDEX `STOCK_FKEY_idx` (`Ticker` ASC) VISIBLE,
  CONSTRAINT `USER_FKEY`
    FOREIGN KEY (`Username`)
    REFERENCES `monte_carlo`.`users` (`Username`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `STOCK_FKEY`
```

```
    FOREIGN KEY (`Ticker`)
    REFERENCES `monte_carlo`.`stocks` (`Ticker`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION);
```

Favourites Table

```
CREATE TABLE `favourites` (
  `Username` varchar(255) NOT NULL,
  `Ticker` varchar(10) NOT NULL,
  PRIMARY KEY (`Username`,`Ticker`),
  KEY `STOCK_FKEY_idx` (`Ticker`),
  CONSTRAINT `FAV_STOCK_FKEY` FOREIGN KEY (`Ticker`) REFERENCES `stocks`
(`Ticker`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `FAV_USER_FKEY` FOREIGN KEY (`Username`) REFERENCES `users`
(`Username`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Relational Schema Diagram



Part B: Sample Data

Populate your database with sample data. Each relation should contain at least 6 tuples. Make sure that the populated data is suitable for the type of queries in Part C.

User Table

```
INSERT INTO users (Username, Email, FirstName, LastName, Password, Role,
Status) VALUES
('A', NULL,  'Graem', 'Sheppard', 'password', 0, 0),
('S', NULL, 'Scott', 'Garland', 'password', 0, 0),
('T', NULL, 'Tyler', 'Wunderlich', 'password', 1, 1),
('F', NULL, 'Fazal', 'Rahman', 'password', 1, 1),
('khalid', NULL, 'Khalid', 'Hafeez', 'password', 2, 1),
('joseph', NULL, 'Joseph', 'Robertson', 'password', 2, 1);
```

Stocks Table

```
INSERT INTO stocks (Ticker, Name)
VALUES
('AAPL', 'Apple Inc.'),
('GOOGL', 'Alphabet Inc.'),
('AMZN', 'Amazon.com, Inc'),
('TSLA', 'Tesla, Inc.'),
('UBER', 'Uber Technologies, Inc.'),
('ZM', 'Zoom Video Communications, Inc'),
('FB', 'Facebook, Inc'),
('SNAP', 'Snap Inc.');
```

Comments Table

```
INSERT INTO comments (Username, Ticker, DateCreated, Content)
VALUES
('khalid', 'AAPL', '2020-01-01', '2 the moon'),
('A', 'AAPL', '2020-02-02', 'to the moon'),
('S', 'SNAP', '2020-03-03', 'lost all my life savings smh'),
('F', 'ZM', '2020-04-04', 'good stock'),
('T', 'FB', '2020-05-05', 'aaaaaaaaaaaaa'),
('A', 'ZM', '2020-06-06', 'zoom'),
('joseph', 'SNAP', '2020-07-07', 'wow good buy!'),
('khalid', 'AAPL', '2020-05-05', 'to the moon!');
```

Favourites Table

```
INSERT INTO favourites (Username, Ticker)
VALUES
('A', 'AAPL'),
('S', 'FB'),
('A', 'GOOGL'),
('T', 'TSLA'),
('F', 'SNAP'),
('khalid', 'AAPL'),
('T', 'ZM'),
('F', 'TSLA'),
('A', 'ZM');
```

Part C: Views

Create (write English description and SQL syntax) of 10 views that a user of the database system would find useful. From these 10 views, the first 5 are common for all groups, and you can create your own views for the remaining ones.

- *View 1: Computes a join of at least three tables*
- *View 2: Uses nested queries with the ANY or ALL operator and uses a GROUP BY clause*
- *View 3: A correlated nested query*
- *View 4: Uses a FULL JOIN*
- *View 5: Uses nested queries with any of the set operations UNION,EXCEPT, or INTERSECT*

View 1

Description: This table shows all user comments on their favourite stocks.

```
CREATE VIEW comments_on_favourites AS
SELECT U.Username, C.Content, C.DateCreated, C.Ticker
FROM users AS U
JOIN comments AS C
JOIN favourites AS F
ON U.Username = F.Username
      AND U.Username = C.Username
      AND C.Ticker = F.Ticker
```

View 2

Description: This table shows all comments with inappropriate content.

```
CREATE VIEW detect_inappropriate_comment AS
SELECT Username
FROM users
WHERE  Username = ANY
(SELECT comments.Username FROM comments WHERE comments.Content LIKE '%bad_word%')
GROUPBY Username
```

View 3

Description: This view shows all users who have made a comment.

```
CREATE VIEW users_with_comments
SELECT U.Username
FROM users AS U
WHERE U.Username = ANY (
      SELECT C.Username
      FROM comments AS C
)
```

View 4

Description: This table shows all usernames and the comments they made.

```
CREATE VIEW user_comments AS
SELECT U.Username, C.Content
FROM U
FULL JOIN C ON U.Username = C.Username
ORDER BY C.Content;
```

View 5

Description: This view displays the users who have more than 1 comment and more than 0 favourites.

```
CREATE VIEW users_who_have_more_than_one_comment_and_more_than_zero_favourites
(SELECT U.Username
FROM users AS U
WHERE U.Username = ANY (
 SELECT C.Username
    FROM comments AS C
    GROUP BY C.Username
    HAVING Count(*) > 1
))
UNION
(
SELECT U.Username
FROM users AS U
WHERE U.Username = ANY (
 SELECT F.Username
    FROM favourites AS F
    GROUP BY F.Username
    HAVING Count(*) > 0
))
```

View 6

Description: This table gets all user info except for the password.

```
CREATE VIEW users_info AS
SELECT U.Username, U.Email, U.FirstName, U.LastName, U.Role, U.Status
FROM users AS U
```

View 7

Description: This table shows the amount of times a certain stock was favourited.

```
CREATE VIEW num_times_favourited
SELECT F.ticker, Count(*)
FROM favourites AS F
GROUP BY F.ticker
```

View 8

Description: This table shows all emails of users with a registered email address.

```
CREATE VIEW all_registered_emails
SELECT U.Username, U.Email
FROM users AS U
WHERE U.Email IS NOT NULL
```

View 9

Description: This table shows all currently suspended users.

```
CREATE VIEW all_suspended_users
SELECT U.Username, U.Email
FROM users AS U
WHERE U.Status = 2
```

View 10

Description: This table shows all online users sorted by their role.

```
CREATE VIEW online_by_role AS
SELECT Count(Username),
FROM users AS U
WHERE U.Status = 1
GROUP BY U.Role
```

Part D: ER Schema Diagram

Create an ER schema diagram for your project database.