BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

✓ WE COULD USE ONE QUERY INSIDE ANOTHER (VIA SUBQUERIES)

✓ WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT

WE HAVE ALREADY COME ACROSS THE SQL INSERT & CREATE TABLE STATEMENTS

RECAP

HOW DO WE CREATE TABLES?

WE HAVE ALREADY COME ACROSS THE SQL INSERT & CREATE TABLE STATEMENTS

StudentID	FirstName	LastName	Gender	Email

```
CREATE TABLE Students
(
StudentID INT NOT NULL AUTO_INCREMENT,
FirstName VARCHAR(30) NOT NULL,
LastName VARCHAR(30) NOT NULL,
Gender CHAR(1),
Email VARCHAR(30) NOT NULL,
PRIMARY KEY (StudentID)
)
```

RECAP

HOW DO WE PUT STUFF INTO TABLES?

WE HAVE ALREADY COME ACROSS THE SQL INSERT & CREATE TABLE STATEMENTS

StudentID	FirstName	LastName	Gender	Email

```
INSERT INTO TABLE Students
(FirstName,LastName,Gender,Email)
VALUES
(‘Janani’,’Ravi’,’F’,’janani@loonycorn.com’)
```

RECAP

HOW DO WE PUT STUFF INTO TABLES?

StudentID	FirstName	LastName	Gender	Email
1	Janani	Ravi	F	janani@loonycorn.com

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT

WE HAVE ALREADY COME ACROSS THE SQL
INSERT & CREATE TABLE STATEMENTS

BUT SUBQUERIES ARE FAR MORE CONVENIENT TO POPULATE A TABLE FROM EXISTING TABLES IN THE DATABASE

BUT SUBQUERIES ARE FAR MORE CONVENIENT TO POPULATE A TABLE FROM EXISTING TABLES IN THE DATABASE

WE COULD EITHER -

- CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY
- CREATE THE TABLE AND POPULATE DIRECTLY USING A SUBQUERY

- ## CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
);

INSERT INTO EmailAddresses
SELECT distinct Email,'Student' AS Category FROM Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM Faculty;
```

• CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
);
```

FIRST CREATE THE TABLE EXACTLY AS USUAL

```
INSERT INTO EmailAddresses
SELECT distinct Email,'Student' AS Category FROM Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM Faculty;
```

• CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
);
```

NEXT USE A DIFFERENT FORM OF THE INSERT STATEMENT, THAT IS FOLLOWED BY A QUERY

```
INSERT INTO EmailAddresses
SELECT distinct Email,'Student' AS Category FROM Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM Faculty;
```

• CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
);

INSERT INTO EmailAddresses
SELECT distinct Email,'Student' AS Category FROM Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM Faculty;
```

THIS INSERT STARTS OFF AS USUAL, WITH THE NAME OF THE TABLE TO INSERT INTO

- ## CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
);

INSERT INTO EmailAddresses
SELECT distinct Email,'Student' AS Category FROM Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM Faculty;
```

THIS INSERT STARTS OFF AS USUAL, WITH THE NAME OF THE TABLE TO INSERT INTO

- CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
);

INSERT INTO EmailAddresses
SELECT distinct Email,'Student' AS Category FROM Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM Faculty;
```

BUT WHAT FOLLOWS IS A QUERY

• CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
);

INSERT INTO EmailAddresses
SELECT distinct Email,'Student' AS Category FROM Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM Faculty;
```

BUT WHAT FOLLOWS IS A QUERY

ALL THAT'S NEEDED IS FOR THE QUERY TO MATCH THE TABLE IN THE NUMBER AND TYPE OF COLUMNS

BUT SUBQUERIES ARE FAR MORE CONVENIENT TO POPULATE A TABLE FROM EXISTING TABLES IN THE DATABASE

WE COULD EITHER -

- CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY
- CREATE THE TABLE AND POPULATE DIRECTLY USING A SUBQUERY

- ## CREATE THE TABLE AND POPULATE DIRECTLY USING A SUBQUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
)
AS
SELECT distinct Email,'Student' AS Category FROM
Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM
Faculty;
```

• CREATE THE TABLE AND POPULATE DIRECTLY USING A SUBQUERY

TABLE DEFINITION

QUERY

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
)
AS
SELECT distinct Email,'Student' AS Category FROM
Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM
Faculty;
```

- ## CREATE THE TABLE AND POPULATE DIRECTLY USING A SUBQUERY

TABLE DEFINITION

```
CREATE TABLE EmailAddresses
(
Email VARCHAR(30) NOT NULL,
Category VARCHAR(10) NOT NULL
)
AS
SELECT distinct Email,'Student' AS Category FROM
Students
UNION
SELECT distinct Email,'Faculty' AS Category FROM
Faculty;
```

LINKED BY 'AS'

QUERY

- ## CREATE THE TABLE AND POPULATE DIRECTLY USING A SUBQUERY

TABLE DEFINITION

```
CREATE TABLE EmailAddresses
AS
SELECT distinct Email,'Student' AS
Category FROM Students
UNION
SELECT distinct Email,'Faculty' AS
Category FROM Faculty;
```

• CREATE THE TABLE AND POPULATE DIRECTLY USING A SUBQUERY

```
CREATE TABLE EmailAddresses
AS
SELECT distinct Email 'Student' AS
Ca
UN
SE                                  AS
Ca
```

TABLE DEFINITION

WE COULD EVEN ENTIRELY SKIP THE COLUMN DEFINITIONS - BUT THEN CONSTRAINTS AND KEYS WILL BE MISSING

BUT SUBQUERIES ARE FAR MORE CONVENIENT TO POPULATE A TABLE FROM EXISTING TABLES IN THE DATABASE

WE COULD EITHER -

- CREATE THE TABLE AS USUAL, THEN INSERT DATA USING A SUBQUERY
- CREATE THE TABLE AND POPULATE DIRECTLY USING A SUBQUERY

BUT SUBQUERIES ARE FAR MORE CONVENIENT TO POPULATE A TABLE FROM EXISTING TABLES IN THE DATABASE

WE COULD ALSO USE SUBQUERIES IN UPDATE AND DELETE STATEMENTS IN SIMILAR FASHION (MORE LATER!)

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT

WE HAVE ALREADY COME ACROSS THE SQL INSERT & CREATE TABLE STATEMENTS

BUT SUBQUERIES ARE FAR MORE CONVENIENT TO POPULATE A TABLE FROM EXISTING TABLES IN THE DATABASE

BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

✓ WE COULD USE ONE QUERY INSIDE ANOTHER (VIA SUBQUERIES)

✓ WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT

BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

✓ WE COULD USE ONE QUERY INSIDE ANOTHER (VIA SUBQUERIES)

✓ WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

✓ WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT