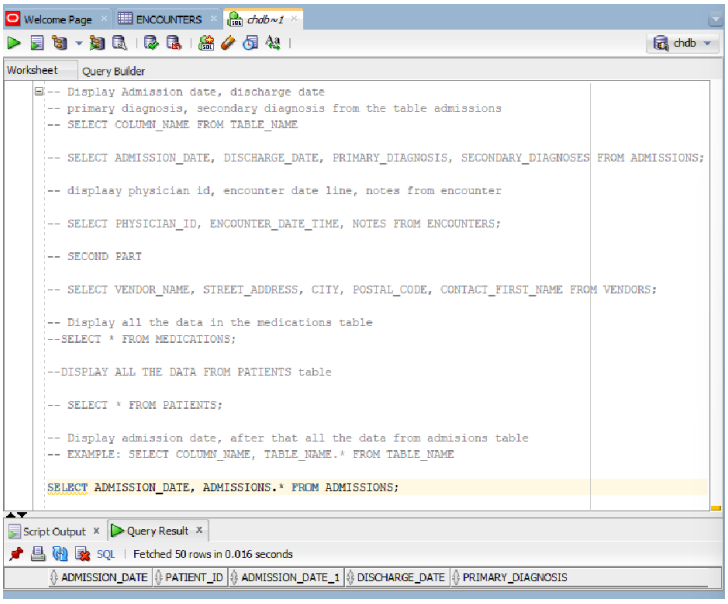
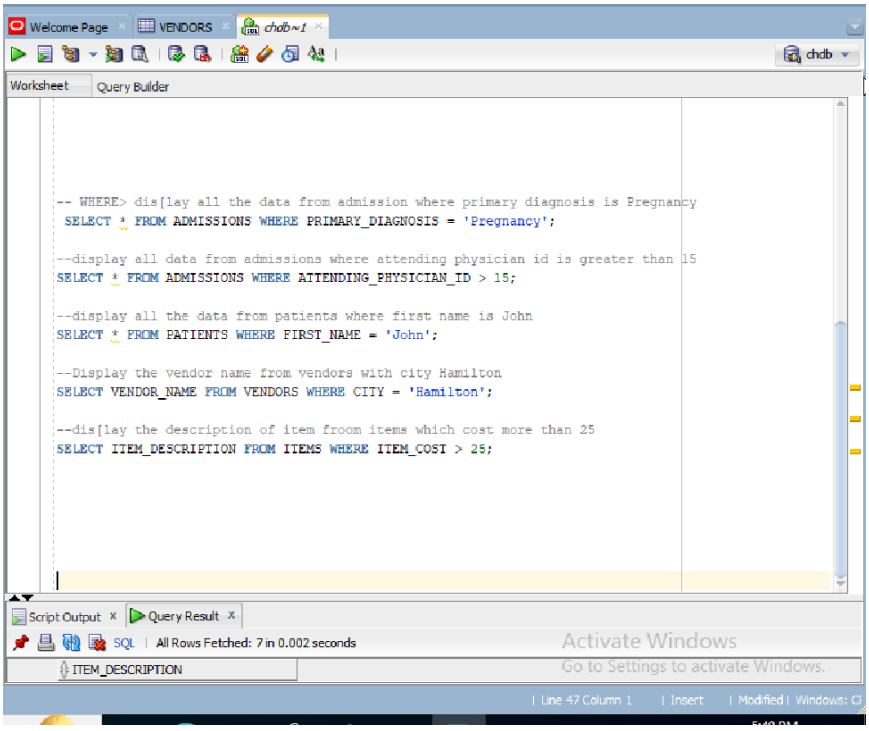
Sql developer commands

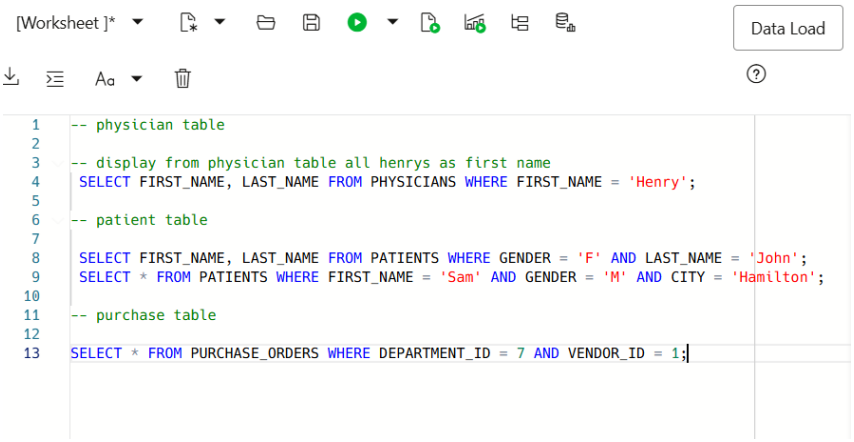
Sql SELECT




WHERE clause



## Logical AND operator




```sql
-- physician table

-- display from physician table all henrys as first name
SELECT FIRST_NAME, LAST_NAME FROM PHYSICIANS WHERE FIRST_NAME = 'Henry';

-- patient table

SELECT FIRST_NAME, LAST_NAME FROM PATIENTS WHERE GENDER = 'F' AND LAST_NAME = 'John';
SELECT * FROM PATIENTS WHERE FIRST_NAME = 'Sam' AND GENDER = 'M' AND CITY = 'Hamilton';

-- purchase table

SELECT * FROM PURCHASE_ORDERS WHERE DEPARTMENT_ID = 7 AND VENDOR_ID = 1;
```

## LOGICAL OR OPERATOR

```sql
--LOGICAL OPERATOR OR

-- SELECT * FROM PURCHASE_ORDERS WHERE DEPARTMENT_ID = 7 OR VENDOR_ID = 1;
--SELECT ADMISSION_DATE FROM ADMISSIONS WHERE PRIMARY_DIAGNOSIS = 'Diabetes' OR PRIMARY_DIA
--SELECT * FROM MEDICATIONS WHERE PACKAGE_SIZE = '58 Tablets' OR STRENGTH = '10 MG' OR SIG =


-- DIFFERENT OPERATORS = < > !=
-- SELECT * FROM UNIT_DOSE_ORDERS WHERE MEDICATION_ID > 56 AND PATIENT_ID != 2056;
```

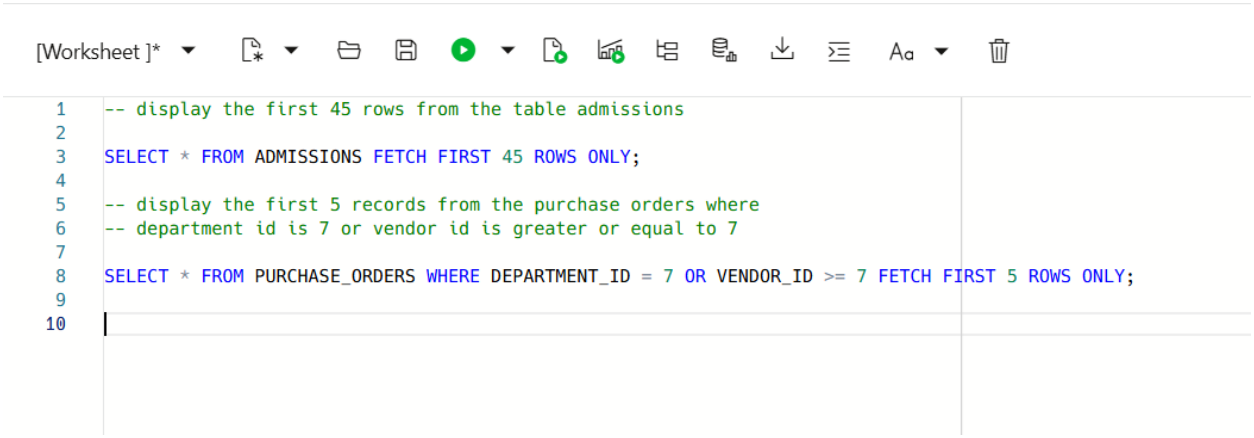
## DISTINCT (different value)

```sql
-- id different entries not repeated in a collumn
SELECT DISTINCT(MEDICATION_ID) FROM UNIT_DOSE_ORDERS;

SELECT DISTINCT(DEPARTMENT_ID) FROM PURCHASE_ORDERS WHERE VENDOR_ID < 8 AND TOTAL_AMOUNT > 100;
```

## COUNT (rows)

```sql

--  count rows with specifications

 SELECT COUNT(DEPARTMENT_ID) FROM PURCHASE_ORDERS; --count all

 SELECT DISTINCT(DEPARTMENT_ID) FROM PURCHASE_ORDERS; -- just distinct

 SELECT COUNT(DISTINCT (DEPARTMENT_ID)) FROM PURCHASE_ORDERS; --count unique

SELECT COUNT(DISTINCT (FIRST_NAME)) FROM PATIENTS;
```

FETCH




```sql
-- display the first 45 rows from the table admissions

SELECT * FROM ADMISSIONS FETCH FIRST 45 ROWS ONLY;

-- display the first 5 records from the purchase orders where
-- department id is 7 or vendor id is greater or equal to 7

SELECT * FROM PURCHASE_ORDERS WHERE DEPARTMENT_ID = 7 OR VENDOR_ID >= 7 FETCH FIRST 5 ROWS ONLY;

```

Min amount

```sql


-- Display the min total amount from the purchase orders

SELECT MIN(TOTAL_AMOUNT) FROM PURCHASE_ORDERS;

--display the min total amount of purchase orders where department id is
-- not equal to 7 and vendor id is less  than 6

SELECT MIN(TOTAL_AMOUNT) FROM PURCHASE_ORDERS WHERE NOT DEPARTMENT_ID = 7 AND VENDOR_ID < 6;
```

MAX amount

```sql


--Display the max units ordered from the purcchase orders line

SELECT MAX(UNITS_ORDERED) FROM PURCHASE_ORDER_LINES;

--Display the max cosr per item if the units recieved is greater than
-- 50 and the item id is not equal to 10 or lin num is less than 10

SELECT MAX(COST_PER_ITEM) FROM PURCHASE_ORDER_LINES WHERE (UNITS_RECEIVED > 50
AND ITEM_ID != 10 ) OR LINE_NUM < 10;


```

Avg and sum

```sql

--display the avg cost per item, max units ordered If the units recieved
-- is less than 5 and item id is greater than 1

SELECT AVG(COST_PER_ITEM), MAX(UNITS_ORDERED) FROM PURCHASE_ORDER_LINES
 WHERE UNITS_RECEIVED < 5 AND ITEM_ID > 1;

-- displa the sum of all the units ordered, minimum units canccelled,
-- avg of all units ordered from purchase order lines
-- if the units cancelled is less than 5 and cost per item is greater than
-- or equal to 13

SELECT SUM(UNITS_ORDERED), MIN(UNITS_CANCELLED), AVG(UNITS_ORDERED)
FROM PURCHASE_ORDER_LINES WHERE UNITS_CANCELLED < 5 AND COST_PER_ITEM >= 13;
```

NOT ( greater than is opposite to less than or equal to)

// >= BECOMES <

// <= BECOMES >

// < BECOMES >=

// > BECOMES <=

// = BECOMES !=

```sql

-- Display the min units received; max units cancelled from purchase orders lines
-- if units ordered is not less than 5 and cost per item
-- is not greater than or queal to 10000

SELECT MIN(UNITS_RECEIVED), MAX(UNITS_CANCELLED) FROM PURCHASE_ORDER_LINES
WHERE NOT (UNITS_ORDERED < 5 AND COST_PER_ITEM >= 10000);
```

NULL

```sql

--select min department id,  top 100 records from purchase orders
--where total amount is null and VenDor is not equal to 0

SELECT MIN(DEPARTMENT_ID) FROM PURCHASE_ORDERS WHERE (TOTAL_AMOUNT IS NULL)
AND VENDOR_ID != 0 FETCH FIRST 100 ROWS ONLY;


-- Display the medication cost where medication cost is not GREATER than 50000
--and sig is not null from med table

SELECT MEDICATION_COST FROM MEDICATIONS WHERE NOT (MEDICATION_COST > 50000
AND SIG IS NULL);
```

ALIASES

```sql

-- display the patient id as "OHIP" and dosage as "Med Dosage" from
--unit DOSE orders where SIG = 'QID' and medication id is not equal to null

SELECT PATIENT_ID AS "OHIP", DOSAGE AS "Med Dosage" FROM UNIT_DOSE_ORDERS
WHERE SIG = 'QID' AND MEDICATION_ID IS NOT NULL;

```

WILDCARDS and LIKE

```sql
-- display top 55 medications descriptions, medication cost as 'Cost', from
--medication table if the medication cost is not greater than 50 or pkg size
-- ends with 'Tablets'

SELECT MEDICATION_DESCRIPTION, MEDICATION_COST AS "Cost" FROM MEDICATIONS
WHERE NOT MEDICATION_COST > 50 OR PACKAGE_SIZE LIKE '%Tablet' FETCH  FIRST 55
ROWS ONLY;
```

RANGE

```sql

--display the minimum number of units recieved from the purchase order lines
-- if the cost per item ends with 85 cents or item id is in the range 10-19
-- //use like operator for both conditions

SELECT MIN(UNITS_RECEIVED) FROM PURCHASE_ORDER_LINES
WHERE COST_PER_ITEM LIKE '%.85' OR ITEM_ID LIKE '1_';
```

```sql
--Display all the details from the items if the description does not ends with
-- 'Bag' and item cost is greater than or equal to 8

SELECT * FROM ITEMS WHERE NOT (ITEM_DESCRIPTION LIKE '%Bag'
OR ITEM_DESCRIPTION LIKE '%Tube') AND ITEM_COST >= 8;
```

CEIL AND ROUND (n in round is the amount of decimals you want)

```sql
-- display the item cost into the neareset upper value /CEIL for the itmes
-- if the usage_ytd is not equal to 12 or qualify on hand is less than 100000

SELECT CEIL(ITEM_COST) FROM ITEMS WHERE USAGE_YTD != 12
OR QUANTITY_ON_HAND < 100000;

--display the total amount rounded to nearest integer with no deimal points
--from purchase orders if the vendor is less than 26
-- and greater than or equal to 2 or order status is active

SELECT ROUND(TOTAL_AMOUNT, 0) FROM PURCHASE_ORDERS WHERE (VENDOR_ID < 26
AND VENDOR_ID >= 2) OR ORDER_STATUS LIKE 'Active';
```

CONCAT

```sql
--Display the managers full name by combining first name and last name
--from department if the department contains 'Finance'

SELECT CONCAT(MANAGER_FIRST_NAME, MANAGER_LAST_NAME)
AS "Full Name" FROM DEPARTMENTS WHERE DEPARTMENT_NAME LIKE '%Finance';
```

FILTERING (ORDER BY)

Can sort more columns with comma And parameters for column ( the speifications)
ASC= ASCENDING ORDER (Default), DESC = descending order (highest to lowest)
FILTRATION FIRST, THEN SORTING (where first, order by second

```sql
--display all the details from purchase orders
-- by sorting based on total amount

SELECT * FROM PURCHASE_ORDERS ORDER BY TOTAL_AMOUNT ASC;

--display item description from items table by sorting based ob item cost
--and quantity on hand and descending sort based on primary vendor

SELECT ITEM_DESCRIPTION FROM ITEMS ORDER BY ITEM_COST ASC, QUANTITY_ON_HAND ASC,
PRIMARY_VENDOR_ID DESC;
```

```sql
--display the item description by sorting based on usage ytd and
-- if the item desc contains 'Biopsy' and quantity on hand
-- is greater than or equal to 5

SELECT ITEM_DESCRIPTION FROM ITEMS WHERE ITEM_DESCRIPTION
LIKE 'Biopsy%' AND QUANTITY_ON_HAND >= 2 ORDER BY USAGE_YTD;

--display the item description, item cost as cost from items if the usage ytd is
-- on the range 20 - 100 and primary vendor is not in the range 1000-1008

SELECT ITEM_DESCRIPTION, ITEM_COST AS "Cost"  FROM ITEMS
WHERE (USAGE_YTD BETWEEN 20 AND 100) AND (PRIMARY_VENDOR_ID NOT BETWEEN 1000 AND 1008);
```

FILTER BY DATE

```sql
-- Display the admission if the secondary diagnosi is Diabetes
-- or Migrane or Osteoporosis ad if the admission date is greater than
-- 10/4/2017 and sort the data based on attending physician id

SELECT * FROM ADMISSIONS WHERE SECONDARY_DIAGNOSES = 'Diabetes' OR
SECONDARY_DIAGNOSES = 'Migrane' OR SECONDARY_DIAGNOSES = 'Osteoporosis' AND
ADMISSION_DATE > TO_DATE('10-04-17', 'dd-MM-YY') ORDER BY ATTENDING_PHYSICIAN_ID ASC;
```

IN (filter rows based on whether a colum value mathches any value in specifie list of values)

```sql

SELECT * FROM ADMISSIONS WHERE SECONDARY_DIAGNOSES IN ('Diabetes', 'Migrane', 'Osteoporosis') AND
ADMISSION_DATE > TO_DATE('10-04-17', 'dd-MM-YY') ORDER BY ATTENDING_PHYSICIAN_ID ASC;
```

```sql

--Display the items if the item description is not 'DVT Stockings', 'Telementary Unit'
--'Epidural Tray' and if the itme cost is between 28 and 100 and
--sort the data based on quantity on hand

SELECT * FROM ITEMS WHERE ITEM_DESCRIPTION NOT IN('DVT Stockings', 'Telementary Units', 'Epidural Tray
AND ITEM_COST BETWEEN 28 AND 100 ORDER BY QUANTITY_ON_HAND ASC;
```

Relating the PURCHASE_ORDER_LINES table and ITEM table with ITEM_ID

```
19
20
21  --Display the min item id that has a quantity on hand greater than 15
22  SELECT MIN(ITEM_ID) FROM ITEMS WHERE QUANTITY_ON_HAND > 15;
23
24  --Display the purchase order line where item id is 3
25  SELECT * FROM PURCHASE_ORDER_LINES WHERE ITEM_ID = 3;
```

SUBQUERY

```
26
27
28  --Display the admission if the attending physician  id is the physician id
29  -- a physician in the speciality 'Internlist'
30
31  SELECT PHYSICIAN_ID FROM PHYSICIANS WHERE SPECIALTY = 'Internist';
32
33  SELECT * FROM ADMISSIONS WHERE ATTENDING_PHYSICIAN_ID = (SELECT PHYSICIAN_ID FROM PHYSICIANS
34  WHERE SPECIALTY = 'Internist');
```

```
35
36  --display all items of all the vendors from city 'Hamilton'
37  SELECT VENDOR_ID FROM VENDORS WHERE CITY = 'Hamilton';
38
39  SELECT * FROM ITEMS WHERE PRIMARY_VENDOR_ID IN (SELECT VENDOR_ID
40  FROM VENDORS WHERE CITY = 'Hamilton');
41
```

ALIASES, current date, current timestamp

```
1   -- display the first name and last name of managers from department table
2   -- along with the prefix/aliases of table name
3   -- // SELECT ALI.COLUMN1, ALI.COLUMN2 FROM TABLE_NAME ALI
4
5   SELECT dep.MANAGER_FIRST_NAME, dep.MANAGER_LAST_NAME FROM DEPARTMENTS dep;
6
7   -- Display the total amount, order status and current session date from
8   --purchase orders along with aliases of table for column names
9   -- and sort the data based on dep id
10
11  SELECT po.TOTAL_AMOUNT, po.ORDER_STATUS, CURRENT_DATE FROM PURCHASE_ORDERS po
12  ORDER BY po.DEPARTMENT_ID ASC;
```

```sql
--display the name units recieved, units cancelled, current session timestamp
--current date from purchase order lines if the units ordered are greater than 0;

SELECT pol.UNITS_RECEIVED, pol.UNITS_CANCELLED, CURRENT_TIMESTAMP, CURRENT_DATE
FROM PURCHASE_ORDER_LINES pol WHERE pol.UNITS_ORDERED > 0;
```

TO_CHAR, TO_NUMBER

```sql


--display the entered date and patient id, medication id, current timestamp,
--from unit dose orders and convert the entered date and patient id to chars
--use appropriafe aliases dor the table

SELECT TO_CHAR(udo.ENTERED_DATE), TO_CHAR(udo.PATIENT_ID),
udo.MEDICATION_ID, CURRENT_TIMESTAMP FROM UNIT_DOSE_ORDERS udo;

--Dispaly medication desc and cost from meds table along with columnn
--that has a string '0001' and convert it into a number when
--you display and sort the data based in medication cost and sig='PRN'

SELECT med.MEDICATION_DESCRIPTION, med.MEDICATION_COST, TO_NUMBER('0001')
FROM MEDICATIONS med WHERE med.SIG = 'PRN' ORDER BY MEDICATION_COST ASC;
```

SYS DATE

```sql

--Display the first and last name of patient along with sys date, curr date
--and if the city = 'hamilton' (patients table)

SELECT ptn.FIRST_NAME, ptn.LAST_NAME, SYSDATE, CURRENT_DATE
FROM PATIENTS ptn WHERE ptn.CITY = 'Hamilton';
```

UPPER, LOWER

```sql

--display the medication desc, med desc in lower, med desc upper from medications
--table  if the sig is prn sort the data based on medication cost

SELECT MEDICATION_DESCRIPTION, LOWER(MEDICATION_DESCRIPTION),
UPPER(MEDICATION_DESCRIPTION) FROM MEDICATIONS WHERE SIG = 'PRN'
ORDER BY MEDICATION_COST;
```

SUBSTRING and REPLACE

```sql

--display the item desc, substring of item desc from indez 3 and
--length of string 6 if the item cost is 5 and quantity on hand is less than 1000
--and st it based on highest to lowest usage ytd
--SUBSTRING(COLUMN_NAME, POSITION, SUBSTR_LENGTH)

SELECT ITEM_DESCRIPTION, SUBSTRING(ITEM_DESCRIPTION; 3, 6) FROM ITEMS WHERE
ITEM_COST = 5 AND QUANTITY_ON_HAND < 1000 ORDER BY USAGE_YTD DESC;

--Display the package size, replace the string "tablets" in package size with string
--"tablet" of the med desc ends with "n"
--REPLACE(STRING, SEARCH_STRING, REPLACEMENT_STRING)

SELECT PACKAGE_SIZE REPLACE("tablets", "tablet") FROM MEDICATIONS WHERE
MEDICATION_DESCRIPTION LIKE '%n';
```

SUBQUERY PRACTICE

```sql
--Display the quantity, unit cost, units recieved, cancelled from
--purchase order lines based on the purchase order id
--if the total amount is greater than 10 from purchase orders
--subquery - use purchase id as connector / pk/ fk

SELECT PURCHASE_ORDER_ID FROM PURCHASE_ORDERS WHERE TOTAL_AMOUNT > 10;

SELECT COST_PER_ITEM, UNITS_RECEIVED, UNITS_CANCELLED FROM PURCHASE_ORDER_LINES
WHERE PURCHASE_ORDER_ID IN (SELECT PURCHASE_ORDER_ID
FROM PURCHASE_ORDERS WHERE TOTAL_AMOUNT > 10);
```

```sql
--display the name of patients from the patients table if the SIG='QID'
--from the unit dose orders table. Use Patient ID as connector for PK-FK
--SUBQUERY

SELECT PATIENT_ID FROM UNIT_DOSE_ORDERS WHERE SIG = 'QID';

SELECT FIRST_NAME FROM PATIENTS WHERE PATIENT_ID IN (SELECT PATIENT_ID FROM
UNIT_DOSE_ORDERS WHERE SIG = 'QID');
```

CREATING TABLE

VARCHAR2(50) the 50 is the amnt of chars in the stirng

NVARCHAR is always 25 chars in a string

--CREATE TABLE table_name (column_1 datatype, column_2 datatype, ...)

```
1  -- create a table studentID (PRIMAKY KEY), first name, last name,
2  --gender default 'M'
3  --CREATE TABLE table_name (column_1 datatype, column_2 datatype, ...)
4
5  CREATE TABLE STUDENTS_EVELYN (STUDENTID INT PRIMARY KEY, FIRSTNAME VARCHAR2(20
6  LASTNAME VARCHAR2(20), GENDER VARCHAR2(1) DEFAULT ('M') );
7
8
```

CREATING INDEX

```
10 --create an index based on first name from physician table
11 --CREATE INDEX INDEX_NAME ON TABLE_NAME(COLUMN_NAME);
12
13 CREATE INDEX FN_PHYSICIAN ON PHYSICIANS(FIRST_NAME);
14
15 --delete the index
16
17 DROP INDEX FN_PHYSICIAN;
```

ALTERING TABLE

```
21
22  ALTER TABLE EVELYNN_PHONES ADD COLUMN_NEW VARCHAR2(10);
23
24  --modify the size of the column into 250 and add constraint NOT NULL, unique
25
26  ALTER TABLE EVELYNN_PHONES MODIFY COLUMN_NEW VARCHAR2(250) NOT NULL;
27  ALTER TABLE EVELYNN_PHONES ADD CONSTRAINT EVELYN_COLUMN_NEW UNIQUE;
28
29  --rename the column name into your first_name_EXISTING_COLUMN_NAME
30
31  ALTER TABLE EVELYNN_PHONES RENAME COLUMN COLUMN_NEW TO EVELYN_COLUMN_NEW;
32
33  --drop the column
34
35  ALTER TABLE EVELYNN_PHONES DROP COLUMN EVELYN_COLUMN_NEW;
36
37
```

```
37  --Rename the existing table's name that you created into
38  --firstname_CHGF_EXISITING_TABLE_NAME
39
40  ALTER TABLE EVELYNN_PHONES RENAME TO EVELYN_CHGF_EVELYNN_PHONES;
41
42  --Add a constraint on the patients table based on the condition
43  --gender in ('M', 'F')
44
45  ALTER TABLE PATIENTS ADD CONSTRAINT GENDER CHECK('M', 'F');
46
47  --drop a constraint that you created from the table patients
48
49  ALTER TABLE PATIENTS DROP CONSTRAINT GENDER;
50
51  --create a backup of the table patients under the name backup_firstName_patients
52
53  CREATE TABLE backup_EVELYN_patients AS SELECT * FROM PATIENTS;
```

INSERT (not really used in real world)

```
8   --insert the data into the table '23', 'jacob', 'smith', 'm'
9   --INSERT INTO TABLE_NAME ( COLUMN1, COLUMN2, COLUMN3)
10  -- VALUES (VALUE1, VALUE2, VALUE3)
11
12  INSERT INTO STUDENTS_EVELYN (STUDENTID, FIRSTNAME, LASTNAME, GENDER) VALUES ('23',
13  'Jacob', 'Smith', 'M');
```

```
--insert the data into the table '23', 'jacob', 'smith', 'm'
--INSERT INTO TABLE_NAME ( COLUMN1, COLUMN2, COLUMN3)
-- VALUES (VALUE1, VALUE2, VALUE3)

INSERT INTO STUDENTS_EVELYN (STUDENTID, FIRSTNAME, LASTNAME, GENDER) VALUES ('2
'Jacob', 'Smith', 'M');


--Insert the data using multiple rows at the same time
INSERT INTO STUDENTS_EVELYN (STUDENTID, FIRSTNAME, LASTNAME, GENDER)
VALUES('1','owen','Maynard','M' ),('2','Shayne','La','M'),
('3','Diana','grace','F'),('4','Tiya','Grace','F');

--Insert the data, first name = 'john', last name = 'collins'
INSERT INTO STUDENTS_EVELYN (FIRSTNAME, LASTNAME) VALUES('john', 'Collins');

--insert data into studetns table from patients table using firstname, lastname
--gender from patients table
--INSERT INTO TABLE_NAME (COLUMN1, COLUMN2, COLUMN3) SELECT COLUMN1, COLUMN2,..
--FROM TABLE_NAEM WHERE CONDITION

DELETE FROM STUDENTS_EVELYN;

INSERT INTO STUDENTS_EVELYN (STUDENTID, FIRSTNAME, LASTNAME, GENDER)
SELECT PATIENT_ID, FIRST_NAME, LAST_NAME, GENDER FROM PATIENTS;
```

UPDATE

```
--add one more column into patients using your first name_dosage string

ALTER TABLE PATIENTS ADD EVELYN_DOSAGE VARCHAR2(50);

--update the newly added column from the dosage column from unit dose orders
--based on the patient id in both colums if medication id is not equal to 5


UPDATE PATIENTS SET EVELYN_DOSAGE=(SELECT DOSAGE FROM UNIT_DOSE_ORDERS
WHERE patients.PATIENT_ID=UNIT_DOSE_ORDERS.PATIENT_ID AND MEDICATION_ID != 5
FETCH FIRST 1 ROWS ONLY);
```

DELETE VS DROP

DROP deletes the WHOLE table PERMANENTLY.

DELETE  deletes the table DATA but keeps the table structure intact

DISPLAY LAST ROWS

```
1  --disply all the medication details of 5 last prescribed medications, (medications)
2
3  SELECT MEDICATION_DESCRIPTION, LAST_PRESCRIBED_DATE FROM MEDICATIONS
4  ORDER BY LAST_PRESCRIBED_DATE DESC FETCH FIRST 5 ROWS ONLY;
```

CREATING TABLE + ADDING ROWS PRACTICE

```
--create a table phones with the name firstname_phones
--phoneID: int PK
--make: varchar (apple, samsung, google, nothing, motorola)
--model: varchar
--year: int (2016-2024)
--price: decimal (500-2000)
--color varchar

--insert at least 5 records

CREATE TABLE EVELYN_PHONES (PHONEID INT PRIMARY KEY, MAKE VARCHAR2(50),
MODEL VARCHAR2(50), YEAR INT, PRICE INT, COLOR VARCHAR2(50) );

INSERT INTO EVELYN_PHONES (PHONEID, MAKE, MODEL, YEAR, PRICE, COLOR)
VALUES ('1', 'SAMSUNG', 'S24 PLUS', 2024, 700, 'VIOLET');

INSERT INTO EVELYN_PHONES (PHONEID, MAKE, MODEL, YEAR, PRICE, COLOR)
VALUES ('2', 'SAMSUNG', 'S24 ULTRA', 2024, 900, 'JET BLACK');

INSERT INTO EVELYN_PHONES (PHONEID, MAKE, MODEL, YEAR, PRICE, COLOR)
VALUES ('3', 'HONOR', 'X8B', 2023, 500, 'SKY BLUE');
```

DISPLAYING BASED ON STRINGS, LOGICAL OPERATORS, RANGE, (BETWEEN) AND IN

```
43  --DISPLAY THE NAME OF PHONES MADE BY 'APPLE'
44
45  SELECT MODEL FROM EVELYN_PHONES WHERE MAKE = 'APPLE';
46
47  --DISPLAY THE NAME OF PHONES MADE BY SAMSUNG OR GOOGLE
48
49  SELECT MODEL FROM EVELYN_PHONES WHERE MAKE = 'SAMSUNG' OR MAKE = 'GOOGLE';
50
51  --DISPLAY THE MODEL, MAKE, YEAR OF ALL PHONES PRODUCED IN 2020, 2021, 2022
52
53  SELECT MODEL, MAKE, YEAR FROM EVELYN_PHONES WHERE YEAR IN(2020, 2021, 2022);
54
55  --DISPLAY THE MAKE, MODEL, PRICE OF THE PNOES WHICH COST BETWEEN 700 AND 1000
56
57  SELECT MODEL, MAKE, YEAR FROM EVELYN_PHONES WHERE PRICE BETWEEN 700 AND 1000;
```

DISPLAY MIN MAX PRICE

```
--DISPLAY THE MAKE, MODEL, PRICE OF THE PHONES WHICH HAS MIN OR MAX PRICE

SELECT MODEL, MAKE, YEAR FROM EVELYN_PHONES WHERE PRICE IN(SELECT MAX(PRICE)
FROM EVELYN_PHONES);

SELECT MODEL, MAKE, YEAR FROM EVELYN_PHONES WHERE PRICE IN(SELECT MIN(PRICE)
FROM EVELYN_PHONES);
```

AVERAGE AND SUM DISPLAYED TOGETHER

```
--DISPLAY AVG AND SUM OF PRICE FROM PHONES TABLE

SELECT AVG(PRICE), SUM(PRICE) FROM EVELYN_PHONES;
```

LAPTOP TABLE

```
--Create a table firstname_laptop_inventory
--laptopID: int PK
--Brand: varchar appñe, dell, hp, lenovo, acer
--model: varchar
--releaseyear: int
--price: decimal
--RAM: int
--Storage: int

CREATE TABLE EVELYN_LAPTOP_INVENTORY (LAPTOPID INT PRIMARY KEY, BRAND VARCHAR2(50),
MODEL VARCHAR2(50), RELEASEYEAR INT, PRICE DECIMAL, RAM INT, STORAGE INT);

--INSERT AT LEAST 5 RECORDS

INSERT INTO EVELYN_LAPTOP_INVENTORY (LAPTOPID, BRAND, MODEL, RELEASEYEAR, PRICE,
RAM, STORAGE) VALUES(1, 'HP', 'ENVY X360', 2016, 1149.99, 16, 512);

INSERT INTO EVELYN_LAPTOP_INVENTORY (LAPTOPID, BRAND, MODEL, RELEASEYEAR, PRICE,
RAM, STORAGE) VALUES (2, 'DELL', 'INSPIRON 15', 2024, 980.50, 16, 1020);

INSERT INTO EVELYN_LAPTOP_INVENTORY (LAPTOPID, BRAND, MODEL, RELEASEYEAR, PRICE,
RAM, STORAGE) VALUES (3, 'APPLE', 'MACBOOK PRO', 2024, 5299.00, 48, 1020);

INSERT INTO EVELYN_LAPTOP_INVENTORY (LAPTOPID, BRAND, MODEL, RELEASEYEAR, PRICE,
RAM, STORAGE) VALUES (4, 'APPLE', 'MACBOOK AIR', 2024, 1299.00, 16, 256);

INSERT INTO EVELYN_LAPTOP_INVENTORY (LAPTOPID, BRAND, MODEL, RELEASEYEAR, PRICE,
RAM, STORAGE) VALUES (5, 'hP', 'LAPTOP 15', 2020, 299.99, 4, 128);
```

DISPLAY MAX, MIN, UPPER, LOWER, IN.BETWEEN, ORDER BY,  NOT IN, DISTINCT

```
--display the laptop brands which has the highst RAM or lowest storage
--and sort it based on its price

SELECT BRAND FROM EVELYN_LAPTOP_INVENTORY WHERE RAM IN(SELECT MAX(RAM)
FROM EVELYN_LAPTOP_INVENTORY) OR  RAM IN(SELECT MIN(RAM) FROM EVELYN_LAPTOP_INVENTORY)
ORDER BY PRICE;

--Display the name all laptop s from HP regardless of it is inserted as 'HP'
--'hp' 'Hp' 'hP'

SELECT MODEL FROM EVELYN_LAPTOP_INVENTORY WHERE LOWER(BRAND)='hp';
SELECT MODEL FROM EVELYN_LAPTOP_INVENTORY WHERE UPPER(BRAN)='HP';

--Display the price of laptops if the laptop is from apple, hp, acer
--or its storage is from 8 to 256 and sort it based on its ram from highste to lowest

SELECT PRICE FROM EVELYN_LAPTOP_INVENTORY WHERE BRAND IN('APPLE', 'HP', 'ACER')
OR STORAGE BETWEEN 8 AND 256 ORDER BY RAM ASC;

--display all the details if the releaseyear is 2020 or 2024 and ram is greater than 6

SELECT * FROM EVELYN_LAPTOP_INVENTORY WHERE RELEASEYEAR IN (2020, 2024) AND RAM > 6;

--display the  model, ram if the brand of the laptop is not from apple, hp

SELECT MODEL, RAM FROM EVELYN_LAPTOP_INVENTORY WHERE BRAND NOT IN('APPLE', 'HP');
```

```
--display total number of brands in laptop table

SELECT COUNT(DISTINCT(BRAND)) FROM EVELYN_LAPTOP_INVENTORY;
--display the model of brand who has the cheapest price

SELECT MODEL FROM EVELYN_LAPTOP_INVENTORY ORDER BY PRICE DESC FETCH FIRST 1
ROWS ONLY;

--UPDATE THE MODEL OF THE LAPTOP TO CH LAPTOP IF THE PRICE IS LESS THAN 500

UPDATE EVELYN_LAPTOP_INVENTORY SET MODEL = 'CH Laptop' WHERE PRICE < 500;
```

QUIZ PRACTICE

CREATE TABLE VACATION_PACKAGES (PACKAGE_ID INT, DESTINATION VARCHAR2(100), COUNTRY VARCHAR2(50), PRICE DECIMAL, DAYS_NUMBER INT);

INSERT INTO VACATION_PACKAGES (PACKAGE_ID, DESTINATION, COUNTRY, PRICE, DAYS_NUMBER) VALUES (1, 'NORTH BAY, ONTARIO', 'CANADA', 62.50, 7);

**Question 2** **0 / 1 point**

Write the query to add a new column 'Airline' to the table and add constraint, default 1000 to the price of package.

ALTER TABLE VACATION_PACKAGES ADD AIRLINE VARCHAR2(200);

ALTER TABLE VACATION_PACKAGES MODIFY PRICE DEFAULT 1000;

**This question has not been graded.**

**The correct answer is not displayed for Written Response type questions.**

▼ Hide question 2 feedback

**Feedback**

ALTER TABLE VACATION_PACKAGES
ADD AIRLINE VARCHAR(25)
ALTER TABLE VACATION_PACKAGES
MODIFY Price DEFAULT 1000

**Question 3** **0 / 1 point**

Write the query to display the number of countries the packages are?

SELECT SUM(COUNTRY) FROM VACATION_PACKAGES;

**This question has not been graded.**

**The correct answer is not displayed for Written Response type questions.**

▼ Hide question 3 feedback

**Feedback**

select count(distinct(country)) from vacation_packages

**Question 4** **0 / 1 point**

Write the query to display locations in Canada and the cost of the package is less than 1000

SELECT COUNTRY FROM VACATION_PACKAGES WHERE PRICE < 1000;

**This question has not been graded.**

**The correct answer is not displayed for Written Response type questions.**

▼ Hide question 4 feedback

**Feedback**

select DESTINATION from vacation_packages where price < 1000 and lower(COUNTRY) like '%canada%'

**Question 5** **0 / 1 point**

Write the query only to display the name of destinations in alphabetical order Z -> A

```
SELECT DESTINATION FROM VACATION_PACKAGES ORDER BY
DESTINATION DESC;
```

**This question has not been graded.**

**The correct answer is not displayed for Written Response type questions.**

▼ Hide question 5 feedback

**Feedback**

SELECT DESTINATION FROM VACATION_PACKAGES ORDER BY DESTINATION DESC

**Question 6** **0 / 1 point**

Write the query to display the locations that starts with letter A or letter E in alphabetical order

```
SELECT DESTIONATION FROM VACATION_PACKAGES WHERE
DESTINATION LIKE 'A%' OR DESTINATION LIKE 'E%' ORDER BY
DESTINATION ASC;
```

**This question has not been graded.**

**The correct answer is not displayed for Written Response type questions.**

▼ Hide question 6 feedback

**Feedback**

SELECT DESTINATION FROM VACATION_PACKAGES WHERE (LOWER(DESTINATION) LIKE 'a%' OR upper(destination) like 'E%') ORDER BY DESTINATION DESC

**Question 7** **0 / 1 point**

Write the query to display the destination name and country which has the min and max package cost

SELECT DESTINATION, COUNTRY FROM VACATION_PACKAGES WHERE PRICE IN (SELECT MIN(PRICE) FROM VACATION_PACKAGES) AND PRICE IN (SELECT MAX(PRICE) FROM VACATION:PACKAGES);

**This question has not been graded.**

**The correct answer is not displayed for Written Response type questions.**

▼ Hide question 7 feedback

**Feedback**

SELECT DESTINATION, COUNTRY FROM VACATION_PACKAGES WHERE PRICE = (SELECT MIN(PRICE) FROM VACATION_PACKAGES) or PRICE= (SELECT MAX(PRICE) FROM VACATION_PACKAGES)

or

SELECT DESTINATION, COUNTRY FROM VACATION_PACKAGES WHERE PRICE in ((SELECT MIN(PRICE) FROM VACATION_PACKAGES),(SELECT MAX(PRICE) FROM VACATION_PACKAGES))

**Question 8** **0 / 1 point**

Write the query to display all the locations in USA, Brazil, England, Australia

SELECT DESTINATION FROM VACATION_PACKAGES WHERE COUNTRY IN ('USA', 'Brazil', 'England', 'Australia');

**This question has not been graded.**

**The correct answer is not displayed for Written Response type questions.**

▼ Hide question 8 feedback

**Feedback**

SELECT DESTINATION, COUNTRY FROM VACATION_PACKAGES WHERE lower(country) in ('usa', 'brazil', 'england', 'australia')

**Question 9** **0 / 1 point**

Write the query to update all the airline into Air Canada for all the destinations in Canada

UPDATE VACATION_PACKAGES SET AIRLINE = 'Air Canada' WHERE DESTINATION = 'Canada';

**This question has not been graded.**

**The correct answer is not displayed for Written Response type questions.**

▼ Hide question 9 feedback

**Feedback**

UPDATE vacation_packages
SET AIRLINE = 'Air Canada'
WHERE Country = 'Canada'

WEEK 9 (after reading week)

```sql
--Display the name of all patients first name and birth date
--if they have dosage route "PQ" from unit dose orders

SELECT FIRST_NAME, BIRTH_DATE FROM PATIENTS WHERE PATIENT_ID IN(SELECT PATIENT_ID
FROM UNIT_DOSE_ORDERS WHERE DOSAGE_ROUTE = 'PQ');
```

```sql
--Display all the purchase orders lines details (*) if the unit cost
--is greater than the medication cost from medication table
--if the medication description = "Decadron"

SELECT * FROM PURCHASE_ORDER_LINES WHERE COST_PER_ITEM > (SELECT MEDICATION_COST FROM
MEDICATIONS WHERE MEDICATION_DESCRIPTION = 'Decadron');
```

```sql
--Display all the purchase orders lines details (*) if the unit cost is less than
--or equal to weight of all the patients from hamilton and lives in street adress
--that contains 27 Main E

SELECT * FROM PURCHASE_ORDER_LINES WHERE COST_PER_ITEM <= (SELECT PATIENTS.PATIENT_WEIGHT FROM
PATIENTS WHERE PATIENTS.CITY = 'Hamilton' AND PATIENTS.STREET_ADDRESS LIKE '%27 Main E%');
```

Remember to use single quote

```sql
--Display the name and street adress of vendor who has a order
--quantity  (items table) not equal to 56 and name of province "Ontario" (provinces)
SELECT QUANTITY_ON_HAND FROM ITEMS WHERE QUANTITY_ON_HAND != 56;
SELECT PROVINCE_ID FROM PROVINCES WHERE PROVINCE_NAME = 'Ontario';

SELECT VENDOR_NAME, STREET_ADDRESS FROM VENDORS WHERE VENDOR_ID IN
(SELECT PRIMARY_VENDOR_ID FROM ITEMS WHERE QUANTITY_ON_HAND != 56)
AND PROVINCE_ID IN(SELECT PROVINCE_ID FROM PROVINCES WHERE PROVINCE_NAME = 'Ontario');
```

For aliases always use 2 to 3 chars, never 1.

JOINS: always say what type of join

```
--Display the name, street adress, province id from vendors, item description order
--quantity and cost from items
--SELECT COLUMN NAMES FROM TABLE_NAME1 INNER JOIN TABLE_NAME2
--ON TABLE_NAME1.PK = TABLE_NAME2=FK

SELECT ven.VENDOR_NAME, ven.STREET_ADDRESS, ven.PROVINCE_ID FROM VENDORS ven
INNER JOIN ITEMS itm ON ven.VENDOR_ID = itm.PRIMARY_VENDOR_ID;

--Display the first name , last name, gender of patient and name of province
--they live in

SELECT pt.FIRST_NAME, pt.LAST_NAME, pt.GENDER FROM PATIENTS pt
INNER JOIN PROVINCES prov ON pt.PROVINCE_ID = prov.PROVINCE_ID;
```

JOINING MORE THAT 3 TABLES AND DISPLAY THE QUERIES FROM DIFFERENT TABLES IN ONE GO.

```
--Display the first name, last name, gender of patient and name of province
--they line in (provinces), dosage route, sig from unit dose orders table
--SELECT COLUMN NAMES FROM TABLE1 INNER JOIN TABLE2
--ON TABLE1.PK = TABLE2.FK INNER JOIN TABLE3 ON TABLE1.PK=TABLE3.FK

SELECT pt.FIRST_NAME, pt.LAST_NAME, pt.GENDER, prov.PROVINCE_NAME,
udo.DOSAGE_ROUTE, udo.SIG FROM PATIENTS pt
INNER JOIN PROVINCES prov ON pt.PROVINCE_ID = prov.PROVINCE_ID INNER JOIN
UNIT_DOSE_ORDERS udo ON pt.PATIENT_ID = udo.PATIENT_ID;
```

```
--Display first name, last name, gender of patient and name
--of province they live in, dosage rout, sig from unit dose orders,
--admission date nad discharge date from admission table

SELECT pt.FIRST_NAME, pt.LAST_NAME, pt.GENDER, prov.PROVINCE_NAME,
udo.DOSAGE_ROUTE, udo.SIG, adm.ADMISSION_DATE, adm.DISCHARGE_DATE FROM PATIENTS pt
INNER JOIN PROVINCES prov ON pt.PROVINCE_ID = prov.PROVINCE_ID INNER JOIN
UNIT_DOSE_ORDERS udo ON pt.PATIENT_ID = udo.PATIENT_ID INNER JOIN ADMISSIONS adm
ON pt.PATIENT_ID = adm.PATIENT_ID;
```

USING JOIN + WHERE + ORDERBY

Put where after all the joinings. Order by after the where.

```
45  --Display the first name of patient, dosage and sig of patient
46  --medication description and cost of the medication they used
47  --if the unit used ytd is not null
48
49  SELECT pt.FIRST_NAME, udo.DOSAGE, udo.SIG, med.MEDICATION_DESCRIPTION,
50  med.MEDICATION_COST FROM PATIENTS pt INNER JOIN UNIT_DOSE_ORDERS udo ON
51  pt.PATIENT_ID = udo.PATIENT_ID INNER JOIN MEDICATIONS med ON
52  udo.MEDICATION_ID = med.MEDICATION_ID WHERE med.UNITS_USED_YTD IS NOT NULL;
```

```
54  --Display the name of the department, total amount they purchase (purchase orders)
55  --order status if the total amount is less than or equal to 100000 and
56  --manager last name contains 'a' and sort the data based on total amount
57
58  SELECT dep.DEPARTMENT_NAME, po.TOTAL_AMOUNT, po.ORDER_STATUS FROM DEPARTMENTS dep
59  INNER JOIN PURCHASE_ORDERS po ON dep.DEPARTMENT_ID = po.DEPARTMENT_ID
60  WHERE po.TOTAL_AMOUNT <= 100000 AND dep.MANAGER_LAST_NAME LIKE '%a%'
61  ORDER BY TOTAL_AMOUNT ASC;
```

MORE JOINING (numbers, LIKE, ORDER BY, INNER JOIN, aliases)

```
1  --Display the first and last name of physicians and admission date of their
2  --admissions if the bed is not equal to 100 and phone number of physicians
3  --contains 0 also sort the data based on ohip registration
4
5  SELECT phys.FIRST_NAME, phys.LAST_NAME, adm.ADMISSION_DATE FROM PHYSICIANS phys
6  INNER JOIN ADMISSIONS adm ON phys.PHYSICIAN_ID = adm.ATTENDING_PHYSICIAN_ID
7  WHERE adm.BED != 100 AND phys.PHONE LIKE ('%0%') ORDER BY phys.OHIP_REGISTRATION;
```

```
8
9  --Display the name of department and order date of purchase order if the
10 --total amount is greater than or equal to 0 also sort data based on purch order id
11
12 SELECT dept.DEPARTMENT_NAME, po.ORDER_DATE FROM DEPARTMENTS dept INNER JOIN
13 PURCHASE_ORDERS po ON dept.DEPARTMENT_ID = po.DEPARTMENT_ID
14 WHERE po.TOTAL_AMOUNT >= 0 ORDER BY po.PURCHASE_ORDER_ID;
```

INNER JOIN: Has both IDs if they show up in all tables. IF there are no matching, the results just wont show for the no match.

LEFT JOIN: Left table content is retrieved with matching rows from right table. If no match, then results are null for the right column. First table (left) is the superior table (primary table). It will display the left table, but right table's info will be displayed as null.

```
15
16  --Display the name of all provinces and name of patients associated with
17  -- each provine regardless of if the province have a patient or not
18
19  SELECT prov.PROVINCE_NAME, pat.FIRST_NAME FROM PROVINCES prov LEFT JOIN
20  PATIENTS pat ON pat.PROVINCE_ID = prov.PROVINCE_ID;
21
22  --Display the name of all provinces and name of patients associated with
23  --if the province DOES NOT HAVE a patient
24
25  SELECT  prov.PROVINCE_NAME, pat.FIRST_NAME FROM PROVINCES prov
26  LEFT JOIN PATIENTS pat ON pat.PROVINCE_ID = prov.PROVINCE_ID
27  WHERE pat.FIRST_NAME IS NULL;
```

I would personally put prov.PROVINCE_ID = pat.PROVINCE_ID but wel, it also looks like that on the examples on the ppt

JOINING 3 TABLES WITH BOTH LEFT JOIN AND INNER JOIN

We add an inner join if having a null is not necessary. We use left join if having a null is necessary.

```
35
36  --Display the first name, gender, birthdate of all patients if they
37  --are not admitted in the hospital(admissions) and their name of province
38
39  SELECT pat.FIRST_NAME, pat.GENDER, pat.BIRTH_DATE, prov.prov.PROVINCE_NAME
40  FROM PATIENTS pat LEFT JOIN ADMISSIONS adm ON pat.PATIENT_ID = adm.PATIENT_ID
41  INNER JOIN PROVINCES prov ON prov.PROVINCE_ID = pat.PROVINCE_ID WHERE
42  adm.PATIENT_ID IS NULL;
43
44  --Display the first name, gender, birth date of all patients if they are OR
45  --not admitted in the hospital(admissions) and their dosage and sig of
46  --the unit dose orders if they have one
47
48  SELECT pat.FIRST_NAME, pat.GENDER, pat.BIRTH_DATE, udo.DOSAGE, udo.SIG
49  FROM PATIENTS pat LEFT JOIN ADMISSIONS adm ON pat.PATIENT_ID = adm.PATIENT_ID
50  INNER JOIN UNIT_DOSE_ORDERS udo ON pat.PATIENT_ID = udo.PATIENT_ID
51  WHERE udo.DOSAGE IS NOT NULL AND udo.SIG IS NOT NULL;
52
```

```sql
--Display the name of all vendors, name of province if they dont have
--a purchase order id and if they still have an item they can sell
--also sort the data based on their order quantity

SELECT ven.VENDOR_NAME, prov.PROVINCE_NAME FROM VENDORS ven
LEFT JOIN PURCHASE_ORDERS po ON ven.VENDOR_ID = po.VENDOR_ID
INNER JOIN ITEMS itm ON ven.VENDOR_ID = itm.PRIMARY_VENDOR_ID
INNER JOIN PROVINCES prov ON ven.PROVINCE_ID = prov.PROVINCE_ID
WHERE po.VENDOR_ID IS NULL OR po.ORDER_STATUS IS NULL;
```

Inner joining 3 tables

```sql
--Display the name of all nursing unit specialty (nursing units) if the attending
--nursing unit has beds (admissions) greater than 2 and if they have patient
--admitted with weight not equal to 55, sort by patient id

SELECT nur.SPECIALTY FROM NURSING_UNITS nur INNER JOIN ADMISSIONS adm ON
nur.NURSING_UNIT_ID = adm.NURSING_UNIT_ID INNER JOIN PATIENTS pat
ON adm.PATIENT_ID = pat.PATIENT_ID WHERE adm.BED > 2
AND pat.PATIENT_WEIGHT <> 55 ORDER BY pat.PATIENT_ID ASC;
```

DIFFERENCE BETWEEN INNER AND ÑEFT JOIN TYPE PROBLEMS

```sql
--Display the first name and street adress, province of all patients regardless
--of if they have an encounter notes or not. filter the patients if their
--postal contains L8/l8

SELECT PAT.FIRST_NAME, PAT.STREET_ADDRESS, PROV.PROVINCE_NAME FROM PATIENTS PAT
LEFT JOIN ENCOUNTERS ENC ON PAT.PATIENT_ID = ENC.PATIENT_ID
INNER JOIN PROVINCES PROV ON PAT.PROVINCE_ID = PROV.PROVINCE_ID
WHERE UPPER(PAT.POSTAL_CODE) LIKE '%L8%';


--Display the first name and street adress, province of all patients regardless
--of if they have an encounter notes. filter the patients if their
--postal contains L8/l8

SELECT PAT.FIRST_NAME, PAT.STREET_ADDRESS, PROV.PROVINCE_NAME FROM PATIENTS PAT
INNER JOIN ENCOUNTERS ENC ON PAT.PATIENT_ID = ENC.PATIENT_ID
INNER JOIN PROVINCES PROV ON PAT.PROVINCE_ID = PROV.PROVINCE_ID
WHERE UPPER(PAT.POSTAL_CODE) LIKE '%L8%';
```

FULL OUTER JOIN

Got an ambiguous error, fixed by adding AS to province id since it is named the same on both tables.

Primary table doesn't affect much in full outer join

```
29  --Display the first name of patients, province id from patients, province id
30  --from provinces, name of province using full outer sort (desc) if based on
31  --province id in provinces
32
33  SELECT PAT.FIRST_NAME, PAT.PROVINCE_ID AS PATIENT_PROV_ID, PRO.PROVINCE_ID,
34  PRO.PROVINCE_NAME FROM PATIENTS PAT FULL OUTER JOIN PROVINCES PRO
35  ON PAT.PROVINCE_ID = PRO.PROVINCE_ID ORDER BY PRO.PROVINCE_ID DESC;
```

```
49  --Display the name of vendor, name of provinces if that province has a vendor or not
50  --or vendor has a province or not, (full).
51  --Also along with it display the ORDER DATE of their purchase orders if vendor have a
52  --purchase order or not (left)
53  --along with it display the name of item if the vendor has an item to sell (inner)
54  --filter: name of province: ontario, alberta, quebec
55  --item cost from items greater than or equal to 5
56  --total amount from purchase order not equal to null
57
58  SELECT VEN.VENDOR_NAME, PRO.PROVINCE_NAME, PO.ORDER_DATE, ITM.ITEM_ID
59  FROM VENDORS VEN FULL OUTER JOIN PROVINCES PRO ON VEN.PROVINCE_ID = PRO.PROVINCE_ID
60  LEFT JOIN PURCHASE_ORDERS PO ON VEN.VENDOR_ID = PO.VENDOR_ID INNER JOIN ITEMS ITM
61  ON VEN.VENDOR_ID = ITM.PRIMARY_VENDOR_ID WHERE PRO.PROVINCE_NAME IN ('Ontario', 'Alberta',
62  'Quebec') AND ITM.ITEM_COST >= 5 AND PO.TOTAL_AMOUNT IS NOT NULL;
```

```
49  --Display the name of vendor, name of provinces if that province has a vendor or not
50  --or vendor has a province or not, (full).
51  --Also along with it display the ORDER DATE of their purchase orders if vendor have a
52  --purchase order or not (left)
53  --along with it display the name of item if the vendor has an item to sell (inner)
54  --filter: name of province: ontario, alberta, quebec
55  --item cost from items greater than or equal to 5
56  --total amount from purchase order not equal to null
57
58  SELECT VEN.VENDOR_NAME, PRO.PROVINCE_NAME, PO.ORDER_DATE, ITM.ITEM_DESCRIPTION
59  FROM VENDORS VEN FULL OUTER JOIN PROVINCES PRO ON VEN.PROVINCE_ID = PRO.PROVINCE_ID
60  LEFT JOIN PURCHASE_ORDERS PO ON VEN.VENDOR_ID = PO.VENDOR_ID INNER JOIN ITEMS ITM
61  ON VEN.VENDOR_ID = ITM.PRIMARY_VENDOR_ID WHERE PRO.PROVINCE_NAME IN ('Ontario', 'Alberta',
62  'Quebec') AND ITM.ITEM_COST >= 5 AND PO.TOTAL_AMOUNT IS NOT NULL;
```

RIGHT JOIN AND INNER JOIN

```sql
--Display the admission and discharge date (admissions) of the
--patients who is not in patient table

SELECT ADM.ADMISSION_DATE, ADM.DISCHARGE_DATE, PAT.FIRST_NAME FROM ADMISSIONS ADM
RIGHT JOIN PATIENTS PAT ON PAT.PATIENT_ID = ADM.PATIENT_ID
WHERE ADM.PATIENT_ID IS NULL;

--Display the name of the physicians and their admissions if
--they have a pateint admitted and display the notes from encounters
--table

SELECT PHYS.FIRST_NAME, ADM.ADMISSION_DATE, ENC.NOTES FROM PHYSICIANS PHYS
INNER JOIN ADMISSIONS ADM ON ADM.ATTENDING_PHYSICIAN_ID = PHYS.PHYSICIAN_ID
INNER JOIN ENCOUNTERS ENC ON ENC.PATIENT_ID = ADM.PATIENT_ID;

--Display the name of the vendors and their province name iF they
--dont have an item to sell

SELECT VEN.VENDOR_NAME, PROV.PROVINCE_NAME FROM VENDORS VEN
INNER JOIN PROVINCES PROV ON PROV.PROVINCE_ID = VEN.PROVINCE_ID
LEFT JOIN ITEMS ITM ON ITM.PRIMARY_VENDOR_ID = VEN.VENDOR_ID
WHERE ITM.PRIMARY_VENDOR_ID IS NULL;

--Display the first name of a patient if they are admitted in hopsital
--and they are assigned to a physician and not assigned to a nursing unit

SELECT PAT.FIRST_NAME FROM PATIENTS PAT INNER JOIN ADMISSIONS ADM ON
ADM.PATIENT_ID = PAT.PATIENT_ID WHERE ADM.ATTENDING_PHYSICIAN_ID IS NOT NULL
AND ADM.NURSING_UNIT_ID IS NULL;

--Display the total number of patients if (patient is from hamilton or
--if the they have province id QC) and units_used_ytd is greater than or equal
--to 2 and sort the data based on their medication cost

SELECT COUNT(UDO.PATIENT_ID) FROM UNIT_DOSE_ORDERS UDO FULL OUTER JOIN PATIENTS PAT
ON PAT.PATIENT_ID = UDO.PATIENT_ID INNER JOIN MEDICATIONS MED
ON MED.MEDICATION_ID = UDO.MEDICATION_ID
WHERE (UPPER(PAT.STREET_ADDRESS) = 'HAMILTON' AND UPPER(PAT.PROVINCE_ID) = 'QC')
AND MED.UNITS_USED_YTD >= 2 ORDER BY MED.MEDICATION_COST;

--Display the first name of and last name of patient if their primary or
--secondary diagnosis is 'Pregnancy' and if their dosage contains '4'

SELECT PAT.FIRST_NAME, PAT.LAST_NAME FROM PATIENTS PAT INNER JOIN ADMISSIONS ADM
ON ADM.PATIENT_ID = PAT.PATIENT_ID INNER JOIN UNIT_DOSE_ORDERS UDO
ON UDO.PATIENT_ID = PAT.PATIENT_ID WHERE (ADM.PRIMARY_DIAGNOSIS = 'Pregnancy' OR
ADM.SECONDARY_DIAGNOSES = 'Pregnancy') AND UDO.DOSAGE LIKE '%4%';
```

CROSS JOIN

THERE IS NO PRIMARY OR SECONDARY, NO PRIMARY KEY OR FOREIGN KEY RELATIONSHIP, JUST JOINS THE TABLES

```
--Do cross join between patients and provinces and display the total number of
--rows that you get as a result of it

SELECT COUNT(PAT.PATIENT_ID) FROM PATIENTS PAT CROSS JOIN PROVINCES PROV;

--Display the first name and last name of nursing unit managers of the
--patients who live in Hamilton or from province id = QC, SK, BC, AB

SELECT NUR.MANAGER_FIRST_NAME|| ' ' || NUR.MANAGER_LAST_NAME FROM
NURSING_UNITS NUR FULL OUTER JOIN ADMISSIONS ADM
ON ADM.NURSING_UNIT_ID = NUR.NURSING_UNIT_ID
FULL OUTER JOIN PATIENTS PAT ON PAT.PATIENT_ID = ADM.PATIENT_ID
WHERE UPPER(PAT.STREET_ADDRESS) = 'HAMILTON' OR PAT.PROVINCE_ID IN('QC', 'SK', 'BC', 'AB');

--Display the highest total amount from purchase order if the department name
--contains 'a' and vendor province id is ON

SELECT MAX(PO.TOTAL_AMOUNT) FROM PURCHASE_ORDERS PO INNER JOIN VENDORS VEN ON
VEN.VENDOR_ID = PO.VENDOR_ID INNER JOIN DEPARTMENTS DEPT ON DEPT.DEPARTMENT_ID = PO.VENDOR_ID
WHERE DEPT.DEPARTMENT_NAME LIKE '%a%' AND VEN.PROVINCE_ID = 'ON';
```