

# GitHub Portfolio Presentation

**Oracle 19c | SQL Server | Java**

**Oracle Database 19c Enterprise Edition Production v. 19.3.0.0.0**

**Oracle SQL Developer v. 20.4.1.407, Build 407.0006**

**SQL \* Plus Release 19.0.0.0.0, v. 19.3.0.0.0**

**SQL Server 2019 v.15.0.2000.5, Build 19042**

**SQL Server Management Studio v.15.0.18390.0**

**Eclipse v. 2020-12 (4.18.0)**

Presented by



**David Watson (a.k.a Freestyle7)**



# Oracle | Java | GitHub Portfolio Presentation

## Contact Information

**David Watson (a.k.a Freestyle7)**  
**djnsmith7@yahoo.com**



**ORACLE®**



# Table of Contents

Introduction.....	1.1
Education.....	1.2
Project Objectives.....	1.3
PL/SQL Entity Tables List.....	2.1 – 2.2.1
SQL Server Entity Tables List.....	2.2.2
PL/SQL Entity Tables   SQL * Plus   Hospital_DB Database .....	2.3 – 2.5
PL/SQL Entity Tables   SQL * Plus.....	3 – 18
PL/SQL Regression Model   Sales_Data.....	19
PL/SQL Primary & Foreign Keys   SQL * Plus.....	20 – 21
PL/SQL Dynamic Stored Procedure   Dynamic Order Update   SQL Developer .....	22 - 23
PL/SQL Function   Net 30 Available Credit   SQL Developer.....	24
Java   JDBC   Call PL/SQL Function   Net 30 Available Credit   Eclipse .....	25
Java   JDBC   Call PL/SQL Stored Procedure   Java PL/SQL Order   Eclipse.....	26
Java   JDBC   Call PL/SQL Stored Procedure   Java PL/SQL Orders_Stat_Data   Eclipse.....	27
Java   JDBC   Create PL/SQL Entity Table.....	28
Java   JDBC   Insert PL/SQL Data.....	29
Java   JDBC   Alter Column   PL/SQL.....	30
Java   JDBC   Select PL/SQL Data   Output.....	31
Java   JDBC   Generate PL/SQL Foreign Keys.....	32
PL/SQL Dynamic Stored Procedure   Mean   Median   Standard Deviation   SQL Developer.....	33
PL/SQL AES192 Encryption   SQL Developer & SQL * Plus .....	34 - 35
PL/SQL DBMS_Scheduler   SQL Developer & SQL * Plus .....	36 - 38
PL/SQL Package   Dynamic Order Update   SQL Developer & SQL * Plus.....	39 – 40
SQL Server Entity Tables List .....	41
SQL Server Primary & Foreign Keys .....	42
SQL Server Statistical Data Sample   Mean   Median   Standard Deviation   SSMS .....	43



# **Database Slide Assignments**

## **Database I: Oracle 19c – GitHub Project**

Slide 2.1

Slides 8 – 27

Slides 33 – 40

## **Database II: Hospital\_DB**

Slides 2.2 – 2.5

Slides 28 – 32

## **Database III: Family\_Practice**

Slides 41 - 43



My path to programming began in high school in Salinas, California where I developed an interest in Red Hat Linux as a hobbyist. Fast-forward to 2019, where I decided to transition from a successful sales and marketing Account Manager with 10+ years of experience to an Oracle developer. I decided to make this transition because programming and working as a technologist is where my real passion resides.

I am building my Oracle, SQL Server, and Java portfolio on GitHub so that I can showcase my skill sets and experience. I intend to utilize this opportunity to build upon this platform as I progress and expand my knowledge and discoveries.

My primary objective is to create an opportunity that will result in landing a position as a Database Developer. I intend to create that opportunity by accurately presenting my Oracle 19c, SQL Developer, and Java projects where I have created three databases with fictitious data. I have designed and built queries and components that draw and execute data from those databases. The following is a list of queries and components that are presented and demonstrated within these projects: stored procedures, functions, data encryption, DBMS tools, a package, and Java JDBC components.

I graduated from Mission College in Spring 2021 with a an Associate of Arts in Communication Studies and a concentration in programming and a respectable GPA. My coursework included Oracle 12c, 19c, PL/SQL, SQL Server, Microsoft T-SQL, SSMS, and Python. As a result, I earned the Computer Information Systems Certificate of Proficiency.

Since I attended both Mission College and West Valley College, I was invited to join each of their Honors programs. If my full-time work schedule had permitted, I would have accepted their invitations. I also made the Dean's list.

I decided to continue my coursework and research by investing in the following programming literature: ‘Oracle PL/SQL Programming, 6<sup>th</sup> ed.’ by Steven Feuerstein, ‘Oracle Database 12c: The Complete Reference’ by Bob Bryla, ‘Oracle Database 12c: SQL’ by Jason Price, ‘Oracle PL/SQL by Example: 5<sup>th</sup> ed.’ by Benjamin Rosenzweig, ‘Java: A Beginner’s Guide: 8<sup>th</sup> ed.’, and ‘Java: The Complete Reference: 11<sup>th</sup> ed.’ by Herbert Schildt, each of which provides focused concentrations, emphasis, and extensive coverage on the components presented within this project.

My objectives for this project are as follows:

1. Demonstrate my PL/SQL and T-SQL abilities within Oracle 19c, SQL Developer, SQL \* Plus, SQL Server, SSMS, Eclipse, and Java environments.
2. Create a framework that showcases my skill sets within a structured layout that includes tables, primary and foreign keys, a dynamic stored procedure, a function, AES 192 encryption, a DBMS\_Scheduler, a package, and a statistical data snapshot with mean, median, and standard deviation data.
3. Create a professional portfolio on GitHub that will result in new Oracle and/or SQL Server development opportunities.
4. Build upon my professional portfolio so that others with similar interests and career aspirations may learn from my skill sets, endeavors, and success in the future.
5. Continue a career-long path of higher education and create a platform where I can share and exchange new discoveries, ideas, and technologies.

- I. Leads
- II. Catalog
- III. Catalog\_Class
- IV. Inventory
- V. Accounts
- VI. Orders
- VII. Dyn\_Order\_Upd
- VIII. Java\_PL\_SQL\_Ord
- IX. Payments
- X. Net\_30
- XI. Net\_30\_Info
- XII. Credit\_Cards
- XIII. Bank\_Accounts
- XIV. Shipments
- XV. Employees
- XVI. Sales Data

- I. Diagnoses
- II. Doctors
- III. Hospitals
- IV. Invoices
- V. Patients
- VI. Products
- VII. Transactions

- 
- I. Invoices
  - II. Patients
  - III. Payments
  - IV. Products
  - V. Transactions
  - VI. Vendor\_Orders
  - VII. Vendors

SQL Plus

SQL&gt; SELECT \* FROM diagnoses;

DIAG_ID	DIAG_DESC	DOC_ID	HOSP_ID
40004567	Common Cold	30003456	10001234
40004568	Pollen Allergies	30003457	10001235
40004569	Fever	30003458	10001236
40004570	Asthma	30003459	10001237
40004571	Migraine	30003460	10001238

SQL Plus

SQL&gt; SELECT \* FROM doctors;

DOC_ID	DOC_FIRST	DOC_LAST	DOC_PHONE	DOC_EMAIL	HOSP_ID
30003456	Vinit	Madhvani	4085592011	Vinit.Madhvani@yahoo.com	10001234
30003457	Shalin	Patel	4088855000	Shalin.Patel@yahoo.com	10001235
30003458	Phillip	Chan	4089472500	Phillip.Chan@yahoo.com	10001236
30003459	Shilpa	Gupta	4089726010	Shilpa.Gupta@yahoo.com	10001237
30003460	Richard	Newell	4082595000	Richard.Newell@yahoo.com	10001238

SQL Plus

SQL&gt; SELECT \* FROM hospitals;

HOSP_ID	HOSP_NAME	HOSP_S_NUM	HOSP_STREET	HOSP_CITY	HOSP_STATE	HOSP_ZIP	HOSP_PHONE
10001235	Santa Clara Valley Medical Center	751	South Bascom Avenue	San Jose	CA	95128	4088855000
10001234	Good Samaritan	2425	Samaritan Drive	San Jose	CA	95124	4085592011
10001236	O'Connor Hospital	2105	Forest Avenue	San Jose	CA	95128	4089472500
10001237	Kaiser Permanente	280	Hospital Parkway	San Jose	CA	95119	4089726010
10001238	Regional Medical Center	225	North Jackson Avenue	San Jose	CA	95116	4082595000

SQL Plus

```
SQL> SELECT inv_id, inv_status, trans_id, hosp_id, doc_id, pat_id, prod_id, prod_desc, prod_qty,
2 TO_CHAR(prod_price, '$99.99') prod_price,
3 TO_CHAR(inv_amt, '$999.99') inv_amt, trans_type, inv_date
4 FROM invoices;
```

INV_ID	INV_STATUS	TRANS_ID	HOSP_ID	DOC_ID	PAT_ID	PROD_ID	PROD_DESC	PROD_QTY	PROD_PR	INV_AMT	TRANS_TYPE	INV_DATE
70007890	Invoiced	60006789	10001234	30003456	20002345	50005678	Sudafed	2	\$16.00	\$32.00	Credit Card	18-JUL-21
70007891	Invoiced	60006790	10001235	30003457	20002346	50005679	Azelastine	2	\$35.00	\$70.00	Credit Card	18-JUL-21
70007892	Invoiced	60006791	10001236	30003458	20002347	50005680	Acetaminophen	2	\$30.00	\$60.00	Credit Card	18-JUL-21
70007893	Invoiced	60006792	10001237	30003459	20002348	50005681	Inhaler	2	\$45.00	\$90.00	Credit Card	18-JUL-21
70007894	Invoiced	60006793	10001238	30003460	20002349	50005682	Aleve	2	\$17.00	\$34.00	Credit Card	18-JUL-21
70007895	Invoiced	60006794	10001234	30003456	20002345	50005678	Sudafed	3	\$16.00	\$48.00	Credit Card	18-JUL-21
70007896	Invoiced	60006795	10001235	30003457	20002346	50005679	Azelastine	3	\$35.00	\$105.00	Credit Card	18-JUL-21
70007898	Invoiced	60006797	10001237	30003459	20002348	50005681	Inhaler	3	\$45.00	\$135.00	Credit Card	18-JUL-21
70007899	Invoiced	60006798	10001238	30003460	20002349	50005682	Aleve	3	\$17.00	\$51.00	Credit Card	18-JUL-21
70007897	Invoiced	60006796	10001236	30003458	20002347	50005680	Acetaminophen	3	\$30.00	\$90.00	Credit Card	18-JUL-21

SQL Plus

```
SQL> SELECT * FROM patients;
```

PAT_ID	PAT_FIRST	PAT_LAST	PAT_S_NUM	PAT_STREET	PAT_CITY	PAT_STATE	PAT_ZIP	PAT_PHONE	PAT_EMAIL
20002347	Joshua	Perez	33727	Daniel Locks	San Jose	CA	95135	5552428787	Joshua.Perez@yahoo.com
20002349	Martha	Martin	861	Richard Mills	San Jose	CA	95110	5552447878	Martha.Martin@yahoo.com
20002346	William	Parish	128	Jessica Park	San Jose	CA	95112	5552417979	William.Parish@yahoo.com
20002345	Peter	Smith	8561	Kennedy Alley	San Jose	CA	95136	5552431898	Peter.Smith@yahoo.com
20002348	Vanessa	Dickenson	794	Jennifer Skyway	San Jose	CA	95050	5552435858	Vanessa.Dickenson@yahoo.com

## Entity Tables | Hospital\_DB | Products | Transactions

2.5

SQL Plus

```
SQL> SELECT prod_id, prod_avail, prod_desc,
  2  TO_CHAR(prod_price, '$99.99') prod_price
  3  FROM products;
```

PROD_ID	PROD_AVAIL	PROD_DESC	PROD_PRICE
50005678	Yes	Sudafed	\$8.00
50005679	Yes	Azelastine	\$33.00
50005680	Yes	Acetaminophen	\$10.00
50005681	Yes	Inhaler	\$35.00
50005682	Yes	Aleve	\$12.00

SQL Plus

```
SQL> SELECT trans_id, hosp_id, diag_id, doc_id, pat_id, prod_id, prod_desc, prod_qty,
  2  TO_CHAR(prod_price, '$99.99') prod_price,
  3  TO_CHAR(trans_amt, '$999.99') trans_amt, trans_type, trans_date
  4  FROM transactions;
```

TRANS_ID	HOSP_ID	DIAG_ID	DOC_ID	PAT_ID	PROD_ID	PROD_DESC	PROD_QTY	PROD_PRICE	TRANS_AM	TRANS_TYPE	TRANS_DATE
60006789	10001234	40004567	30003456	20002345	50005678	Sudafed	2	\$16.00	\$32.00	Credit Card	18-JUL-21
60006790	10001235	40004568	30003457	20002346	50005679	Azelastine	2	\$35.00	\$70.00	Credit Card	18-JUL-21
60006791	10001236	40004569	30003458	20002347	50005680	Acetaminophen	2	\$30.00	\$60.00	Credit Card	18-JUL-21
60006792	10001237	40004570	30003459	20002348	50005681	Inhaler	2	\$45.00	\$90.00	Credit Card	18-JUL-21
60006793	10001238	40004571	30003460	20002349	50005682	Aleve	2	\$17.00	\$34.00	Credit Card	18-JUL-21
60006795	10001235	40004568	30003457	20002346	50005679	Azelastine	3	\$35.00	\$105.00	Credit Card	18-JUL-21
60006796	10001236	40004569	30003458	20002347	50005680	Acetaminophen	3	\$30.00	\$90.00	Credit Card	18-JUL-21
60006797	10001237	40004570	30003459	20002348	50005681	Inhaler	3	\$45.00	\$135.00	Credit Card	18-JUL-21
60006798	10001238	40004571	30003460	20002349	50005682	Aleve	3	\$17.00	\$51.00	Credit Card	18-JUL-21
60006794	10001234	40004567	30003456	20002345	50005678	Sudafed	3	\$16.00	\$48.00	Credit Card	18-JUL-21

## Entity Tables | Leads | Catalog | Catalog\_Class

3, 4, 5

SQL Plus

```
SQL> SELECT TO_CHAR(lead_id, '99999999') lead_id, lead_status, lead_name, lead_first, lead_last, lead_phone, lead_email, lead_s_num, lead_street, lead_city, lead_state, lead_zip,
  2 TO_CHAR(lead_created_date, 'YYYY-MM-DD') lead_created_date,
  3 TO_CHAR(lead_last_mod_date, 'YYYY-MM-DD') lead_last_mod_date
  4 FROM leads;
```

LEAD_ID	LEAD_STATUS	LEAD_NAME	LEAD_FIRST	LEAD_LAST	LEAD_PHONE	LEAD_EMAIL	LEAD_S_NUM	LEAD_STREET	LEAD_CITY	LEAD_STATE	LEAD_ZIP	LEAD_CREATED_DATE	LEAD_LAST_MOD_DATE
55550001	Open	Citigroup	Peter	Smith	9243799521	Peter.Smith@yahoo.com	8561	Kennedy Alley	Johnstown	SC	82834	2021-01-12	2021-01-13
55550002	Open	Marathon Petroleum	William	Parish	4710718185	William.Parish@yahoo.com	128	Jessica Park	West Edward	CO	84036	2021-01-12	2021-01-13
55550003	Open	Comcast	Joshua	Perez	5668042289	Joshua.Perez@hotmail.com	33727	Daniel Locks	East Kaitlyn	RI	46820	2021-01-12	2021-01-13
55550004	Closed	Anthem	Vanessa	Dicker	5748474529	Vanessa.Dicker@aol.com	794	Jennifer Skyway	Port Joshua	NY	75313	2021-01-12	2021-01-13
55550005	Closed	Dell Technologies	Martha	Martin	4047573224	Martha.Martin@aol.com	861	Richard Mills	Smithburgh	AK	51361	2021-01-12	2021-01-13
55550006	Closed	DuPont	Brandi	Ruby	9424480478	Brandi.Ruby@gmail.com	961	Roberts Garden	Trevorside	VA	71569	2021-01-12	2021-01-13

SQL Plus

```
SQL> SELECT cat_id, cat_status, ord_id,
  2 TO_CHAR(cat_price, '$99,999.99') cat_price, cat_desc,
  3 TO_CHAR(cat_created_date, 'YYYY-MM-DD') cat_created_date,
  4 TO_CHAR(cat_last_mod_date, 'YYYY-MM-DD') cat_last_mod_date
  5 FROM catalog;
```

CAT_ID	CAT_STATUS	ORD_ID	CAT_PRICE	CAT_DESC	CAT_CREATED_DATE	CAT_LAST_MOD_DATE
20000001	Active	50000001	\$350.00	COVID-19 Test Kit	2021-01-12	2021-01-13
20000002	Active	50000002	\$450.00	Bed Linen	2021-01-12	2021-01-13
20000003	Active	50000003	\$4,500.00	Adjustable Bed	2021-01-14	2021-01-14
20000004	Active	50000004	\$15,500.00	Hydrotherapy Fitness	2021-01-14	2021-01-14
20000005	Active	50000005	\$950.00	Glucose Monitor	2021-01-14	2021-01-14
20000006	Active	50000006	\$550.00	Wound Dressing	2021-01-14	2021-01-14

SQL Plus

```
SQL> SELECT TO_CHAR(cat_class_id, '99999999') cat_class_id, cat_class_status, cat_id, cat_class_desc,
  2 TO_CHAR(cat_class_created_date, 'YYYY-MM-DD') cat_class_created_date,
  3 TO_CHAR(cat_class_last_mod_date, 'YYYY-MM-DD') cat_class_last_mod_date
  4 FROM catalog_class;
```

CAT_CLASS_ID	CAT_CLASS_STATUS	CAT_ID	CAT_CLASS_DESC	CAT_CLASS_CREATED_DATE	CAT_CLASS_LAST_MOD_DATE
30000001	Active	20000001	COVID-19 Supplies	2021-01-14	2021-01-14
30000002	Active	20000002	Patient Care	2020-01-15	2021-01-15
30000003	Active	20000003	Patient Mobility	2020-01-15	2021-01-15
30000004	Active	20000004	Rehabilitation	2020-01-15	2021-01-15
30000005	Active	20000005	Diabetic Products	2020-01-15	2021-01-15
30000006	Active	20000006	Skin and Wound Care	2020-01-15	2021-01-15

## Entity Tables | Inventory | Accounts

6, 7

SQL Plus

```
SQL> SELECT inv_id, inv_status, cat_id, rt_inv_qty, ord_id, invoice_id,
  2 TO_CHAR(inv_created_date, 'YYYY-MM-DD') inv_created_date,
  3 TO_CHAR(inv_last_mod_date, 'YYYY-MM-DD') inv_last_mod_date
  4 FROM inventory;
```

INV_ID	INV_ST	CAT_ID	RT_INV_QTY	ORD_ID	INVOICE_ID	INV_CREATED_DATE	INV_LAST_MOD_DATE
40000001	Active	20000001	950	50000001	88880001	2021-01-15	2021-01-15
40000002	Active	20000002	850	50000002	88880002	2021-01-15	2021-01-15
40000003	Active	20000003	700	50000003	88880003	2021-01-15	2021-01-15
40000004	Active	20000004	650	50000004	88880004	2021-01-15	2021-01-15
40000005	Active	20000005	600	50000005	88880005	2021-01-15	2021-01-15
40000006	Active	20000006	500	50000006	88880006	2021-01-15	2021-01-15

SQL Plus

```
SQL> SELECT acct_id, acct_status, emp_id, parent_id, acct_name, acct_s_num, acct_street, acct_city, acct_state, acct_zip, pass_id,
  2 TO_CHAR(acct_created_date, 'YYYY-MM-DD') acct_created_date,
  3 TO_CHAR(acct_last_mod_date, 'YYYY-MM-DD') acct_last_mod_date
  4 FROM accounts;
```

ACCT_ID	ACCT_S	EMP_ID	PARENT_ID	ACCT_NAME	ACCT_S_NUM	ACCT_STREET	ACCT_CITY	ACCT_STATE	ACCT_ZIP	PASS_ID	ACCT_CREATED_DATE	ACCT_LAST_MOD_DATE
10000001	Open	11110001	22220001	Walmart	702	SW 8th Street	Bentonville	AR	72716	*****	2021-01-15	2021-01-15
10000002	Open	11110002	22220002	EXXON MOBIL	5959	Las Colinas Blvd	Irving	TX	75039	*****	2021-01-15	2021-01-15
10000003	Open	11110003	22220003	Apple	1	Apple Park Way	Cupertino	CA	95014	*****	2021-01-15	2021-01-15
10000004	Open	11110004	22220004	Amazon	410	Terry Avenue North	Seattle	WA	98109	*****	2021-01-15	2021-01-15
10000005	Open	11110005	22220005	UnitedHealth Group	9900	Bren Road East	Minnetonka	MN	55343	*****	2021-01-15	2021-01-15
10000006	Open	11110006	22220006	McKesson	6535	Texas State Highway	Irving	TX	75039	*****	2021-01-15	2021-01-15

# Entity Tables | Orders

SQL Plus										
	ACCT_ID	ACCT_NAME	CAT_ID	ORD_ID	ORD_QTY	ORD_PRICE	ORD_TOTAL	ORD_STATUS	ORD_CREATED_DATE	ORD_LAST_MOD_DATE
10000001	Walmart	20000001	50000001	5	\$325.00	\$1,625.00	Invoiced	2021-01-15	2021-01-15	
10000002	Exxon Mobil	20000002	50000002	10	\$425.00	\$4,250.00	Invoiced	2021-01-15	2021-01-15	
10000003	Apple	20000003	50000003	15	\$4,450.00	\$66,750.00	Invoiced	2021-01-15	2021-01-15	
10000004	Amazon	20000004	50000004	20	\$15,450.00	\$309,000.00	Invoiced	2021-01-15	2021-01-15	
10000005	UnitedHealth Group	20000005	50000005	25	\$925.00	\$23,125.00	Invoiced	2021-01-15	2021-01-15	
10000006	McKesson	20000006	50000006	30	\$325.00	\$15,750.00	Invoiced	2021-01-15	2021-01-15	
10000003	Apple	20000004	50000009	23	\$15,450.00	\$355,350.00	Invoiced	2020-12-15	2021-06-14	
10000003	Apple	20000005	50000030	16	\$925.00	\$14,800.00	Invoiced	2020-11-15	2021-06-14	
10000003	Apple	20000006	50000031	30	\$325.00	\$15,750.00	Invoiced	2020-10-15	2021-06-14	
10000003	Apple	20000001	50000032	27	\$325.00	\$8,775.00	Invoiced	2020-09-15	2021-06-14	
10000003	Apple	20000002	50000033	21	\$425.00	\$8,925.00	Invoiced	2020-08-15	2021-06-14	
10000003	Apple	20000003	50000034	29	\$4,450.00	\$129,050.00	Invoiced	2020-07-15	2021-06-14	
10000003	Apple	20000004	50000035	19	\$15,450.00	\$293,550.00	Invoiced	2020-06-15	2021-06-14	
10000003	Apple	20000005	50000036	29	\$925.00	\$26,825.00	Invoiced	2020-05-15	2021-06-14	
10000003	Apple	20000006	50000037	20	\$325.00	\$10,500.00	Invoiced	2020-04-15	2021-06-14	
10000003	Apple	20000001	50000038	26	\$325.00	\$8,450.00	Invoiced	2020-03-15	2021-06-14	
10000003	Apple	20000002	50000039	11	\$425.00	\$4,675.00	Invoiced	2020-02-15	2021-06-14	
10000004	Amazon	20000005	50000040	30	\$925.00	\$27,750.00	Invoiced	2020-12-15	2021-06-14	
10000006	Amazon	20000006	50000041	33	\$325.00	\$17,325.00	Invoiced	2020-11-15	2021-06-14	
10000004	Amazon	20000001	50000042	17	\$325.00	\$5,525.00	Invoiced	2020-10-15	2021-06-14	
10000004	Amazon	20000002	50000043	28	\$425.00	\$11,900.00	Invoiced	2020-09-15	2021-06-14	
10000004	Amazon	20000003	50000044	24	\$4,450.00	\$96,800.00	Invoiced	2020-08-15	2021-06-14	
10000004	Amazon	20000004	50000045	22	\$15,450.00	\$339,900.00	Invoiced	2020-07-15	2021-06-14	
10000004	Amazon	20000005	50000046	29	\$925.00	\$26,825.00	Invoiced	2020-06-15	2021-06-14	
10000004	Amazon	20000006	50000047	18	\$325.00	\$9,450.00	Invoiced	2020-05-15	2021-06-14	
10000004	Amazon	20000001	50000048	21	\$325.00	\$6,825.00	Invoiced	2020-04-15	2021-06-14	
10000004	Amazon	20000002	50000049	19	\$425.00	\$8,075.00	Invoiced	2020-03-15	2021-06-14	
10000004	Amazon	20000003	50000050	5	\$4,450.00	\$22,250.00	Invoiced	2020-02-15	2021-06-14	
10000005	UnitedHealth Group	20000006	50000051	23	\$325.00	\$12,075.00	Invoiced	2020-12-15	2021-06-14	
10000005	UnitedHealth Group	20000001	50000052	18	\$325.00	\$5,850.00	Invoiced	2020-11-15	2021-06-14	
10000005	UnitedHealth Group	20000002	50000053	15	\$425.00	\$6,375.00	Invoiced	2020-10-15	2021-06-14	
10000005	UnitedHealth Group	20000003	50000054	30	\$4,450.00	\$133,500.00	Invoiced	2020-09-15	2021-06-14	
10000005	UnitedHealth Group	20000004	50000055	21	\$15,450.00	\$324,450.00	Invoiced	2020-08-15	2021-06-14	
10000005	UnitedHealth Group	20000005	50000056	19	\$925.00	\$17,575.00	Invoiced	2020-07-15	2021-06-14	
10000005	UnitedHealth Group	20000006	50000057	27	\$325.00	\$14,175.00	Invoiced	2020-06-15	2021-06-14	
10000005	UnitedHealth Group	20000001	50000058	26	\$325.00	\$8,450.00	Invoiced	2020-05-15	2021-06-14	
10000005	UnitedHealth Group	20000002	50000059	7	\$425.00	\$3,297.50	Invoiced	2020-04-15	2021-06-14	
10000005	UnitedHealth Group	20000003	50000060	16	\$4,450.00	\$71,200.00	Invoiced	2020-03-15	2021-06-14	
10000005	UnitedHealth Group	20000004	50000061	29	\$15,450.00	\$448,050.00	Invoiced	2020-02-15	2021-06-14	
10000006	McKesson	20000001	50000062	20	\$325.00	\$6,500.00	Invoiced	2020-12-15	2021-06-14	
10000006	McKesson	20000002	50000063	22	\$425.00	\$9,350.00	Invoiced	2020-11-15	2021-06-14	
10000006	McKesson	20000003	50000064	24	\$4,450.00	\$106,800.00	Invoiced	2020-10-15	2021-06-14	
10000006	McKesson	20000004	50000065	29	\$15,450.00	\$448,050.00	Invoiced	2020-09-15	2021-06-14	
10000006	McKesson	20000005	50000066	11	\$925.00	\$10,175.00	Invoiced	2020-08-15	2021-06-14	
10000006	McKesson	20000006	50000067	12	\$325.00	\$6,300.00	Invoiced	2020-07-15	2021-06-14	
10000006	McKesson	20000001	50000068	14	\$325.00	\$4,550.00	Invoiced	2020-06-15	2021-06-14	
10000006	McKesson	20000002	50000069	16	\$425.00	\$6,800.00	Invoiced	2020-05-15	2021-06-14	
10000006	McKesson	20000003	50000070	18	\$4,450.00	\$80,100.00	Invoiced	2020-04-15	2021-06-14	
10000006	McKesson	20000004	50000071	19	\$15,450.00	\$293,550.00	Invoiced	2020-03-15	2021-06-14	
10000006	McKesson	20000005	50000072	21	\$925.00	\$19,425.00	Invoiced	2020-02-15	2021-06-14	
10000001	Walmart	20000002	50000007	9	\$425.00	\$3,825.00	Invoiced	2020-12-15	2021-06-14	
10000001	Walmart	20000003	50000008	12	\$4,450.00	\$53,400.00	Invoiced	2020-11-15	2021-06-14	
10000001	Walmart	20000004	50000009	15	\$15,450.00	\$231,750.00	Invoiced	2020-10-15	2021-06-14	
10000001	Walmart	20000005	50000010	18	\$925.00	\$16,650.00	Invoiced	2020-09-15	2021-06-14	
10000001	Walmart	20000006	50000011	23	\$325.00	\$12,075.00	Invoiced	2020-08-15	2021-06-14	
10000001	Walmart	20000001	50000012	27	\$325.00	\$8,775.00	Invoiced	2020-07-15	2021-06-14	
10000001	Walmart	20000002	50000013	30	\$425.00	\$12,750.00	Invoiced	2020-06-15	2021-06-14	
10000001	Walmart	20000003	50000014	7	\$4,450.00	\$31,150.00	Invoiced	2020-05-15	2021-06-14	
10000001	Walmart	20000004	50000015	11	\$15,450.00	\$169,950.00	Invoiced	2020-04-15	2021-06-14	
10000001	Walmart	20000005	50000016	10	\$925.00	\$9,250.00	Invoiced	2020-03-15	2021-06-14	
10000001	Walmart	20000006	50000017	21	\$325.00	\$11,025.00	Invoiced	2020-02-15	2021-06-14	
10000002	EXXON MOBIL	20000003	50000018	17	\$4,450.00	\$75,650.00	Invoiced	2020-12-15	2021-06-14	
10000002	EXXON MOBIL	20000004	50000019	12	\$15,450.00	\$185,400.00	Invoiced	2020-11-15	2021-06-14	
10000002	EXXON MOBIL	20000005	50000020	19	\$925.00	\$17,575.00	Invoiced	2020-10-15	2021-06-14	
10000002	EXXON MOBIL	20000006	50000021	24	\$325.00	\$12,600.00	Invoiced	2020-09-15	2021-06-14	
10000002	EXXON MOBIL	20000001	50000022	29	\$325.00	\$9,425.00	Invoiced	2020-08-15	2021-06-14	
10000002	EXXON MOBIL	20000002	50000023	27	\$425.00	\$11,475.00	Invoiced	2020-07-15	2021-06-14	
10000002	EXXON MOBIL	20000003	50000024	16	\$4,450.00	\$71,200.00	Invoiced	2020-06-15	2021-06-14	
10000002	EXXON MOBIL	20000004	50000025	18	\$15,450.00	\$278,100.00	Invoiced	2020-05-15	2021-06-14	
10000002	EXXON MOBIL	20000005	50000026	14	\$925.00	\$12,950.00	Invoiced	2020-04-15	2021-06-14	
10000002	EXXON MOBIL	20000006	50000027	22	\$325.00	\$11,550.00	Invoiced	2020-03-15	2021-06-14	
10000002	EXXON MOBIL	20000001	50000028	25	\$325.00	\$8,125.00	Invoiced	2020-02-15	2021-06-14	

## SQL Plus

```
SQL> SELECT * FROM dyn_order_upd
 2 ORDER BY dyn_ord_id;
DYN_ORD_ID DYN_ACCT_ID DYN_QUOTE_ID DYN_CAT_ID
----- ----- ----- -----
43210001 33330001 44440001 12340001
43210002 33330002 44440002 12340002
43210003 33330003 44440003 12340003
43210004 33330004 44440004 12340004
43210005 33330005 44440005 12340005
43210006 33330006 44440006 12340006
```

## SQL Plus

```
SQL> SELECT * FROM java_pl_sql_ord;
JAVA_ORD_ID JAVA_ACCT_ID JAVA_QUOTE_ID JAVA_CAT_ID
----- ----- ----- -----
1001      2001      3001      4001
1002      2002      3002      4002
1003      2003      3003      4003
1004      2004      3004      4004
1005      2005      3005      4005
```

## SQL Plus

```
SQL> SELECT pay_id, pay_status, trans_id,
 2 TO_CHAR(trans_amt, '$999,999.99') trans_amt,
 3 TO_CHAR(pay_created_date, 'YYYY-MM-DD') pay_created_date,
 4 TO_CHAR(pay_last_mod_date, 'YYYY-MM-DD') pay_last_mod_date
 5 FROM payments;
PAY_ID PAY_STATUS    TRANS_ID TRANS_AMT    PAY_CREATED_DATE    PAY_LAST_MOD_DATE
----- ----- ----- -----    ----- -----
60000001 Paid        77770001 $1,625.00 2021-01-15    2021-01-15
60000002 Paid        77770002 $4,250.00 2021-01-15    2021-01-15
60000003 Paid        77770003 $66,750.00 2021-01-15    2021-01-15
60000004 Paid        77770004 $309,000.00 2021-01-15    2021-01-15
60000005 Paid        77770005 $23,125.00 2021-01-15    2021-01-15
60000006 Paid        77770006 $15,750.00 2021-01-15    2021-01-15
```

## Entity Tables | Net\_30 | Net\_30\_Info \_Credit\_Cards

12, 13, 14

SQL Plus

```
SQL> SELECT net_30_id, net_30_status, acct_id, pay_id,
  2  TO_CHAR(net_30_credit_limit, '$999,999.99') net_30_credit_limit,
  3  TO_CHAR(trans_amt, '$99,999.99') trans_amt,
  4  TO_CHAR(net_30_avail_cred, '$99,999.99') net_30_avail_cred, bank_check_num,
  5  TO_CHAR(net_30_created_date, 'YYYY-MM-DD') net_30_created_date,
  6  TO_CHAR(net_30_last_mod_date, 'YYYY-MM-DD') net_30_last_mod_date
  7  FROM net_30;
```

NET_30_ID	NET_30_STATUS	ACCT_ID	PAY_ID	NET_30_CREDIT_LIMIT	TRANS_AMT	NET_30_AVAIL_CRED	BANK_CHECK_NUM	NET_30_CREATED_DATE	NET_30_LAST_MOD_DATE	
80000001	Open	10000001	60000001	\$10,000.00		\$0.00	\$10,000.00	0	2021-01-15	2021-01-15
80000002	Open	10000002	60000002	\$15,000.00		\$0.00	\$15,000.00	0	2021-01-15	2021-01-15
80000003	Open	10000003	60000003	\$75,000.00		\$66,750.00	\$8,250.00	1234	2021-01-15	2021-01-15
80000004	Open	10000004	60000004	\$350,000.00		\$309,000.00	\$41,000.00	1235	2021-01-15	2021-01-15
80000005	Open	10000005	60000005	\$30,000.00		\$0.00	\$30,000.00	0	2021-01-15	2021-01-15
80000006	Open	10000006	60000006	\$35,000.00		\$0.00	\$35,000.00	0	2021-01-15	2021-01-15

SQL Plus

```
SQL> SELECT net_30_id, net_30_status, acct_id,
  2  TO_CHAR(net_30_avail_cred, '$99,999.99') net_30_avail_cred
  3  FROM net_30_info;
```

NET_30_ID	NET_30_STATUS	ACCT_ID	NET_30_AVAIL_CRED
80000001	OPEN	10000001	\$10,000.00

SQL Plus

```
SQL> SELECT cc_id, pay_id, acct_id, cc_num,
  2  TO_CHAR(exp_date, 'YYYY-MM-DD') exp_date, cvv, cc_first, cc_last,
  3  TO_CHAR(cc_created_date, 'YYYY-MM-DD') cc_created_date,
  4  TO_CHAR(cc_last_mod_date, 'YYYY-MM-DD') cc_last_mod_date
  5  FROM credit_cards;
```

CC_ID	PAY_ID	ACCT_ID	CC_NUM	EXP_DATE	CVV	CC_FIRST	CC_LAST	CC_CREATED_DATE	CC_LAST_MOD_DATE
70000001	60000001	10000001	*****1234	2021-01-01	***	Benjamin	Nichols	2021-01-15	2021-01-15
70000002	60000002	10000002	*****1235	2021-01-01	***	Tyler	Brown	2021-01-15	2021-01-15
70000003	60000003	10000003	*****1236	2021-01-01	***	Richard	Robinson	2021-01-15	2021-01-15
70000004	60000004	10000004	*****1237	2021-01-01	***	Peter	Fernandez	2021-01-15	2021-01-15
70000005	60000005	10000005	*****1238	2021-01-01	***	Henry	Myers	2021-01-15	2021-01-15
70000006	60000006	10000006	*****1239	2021-01-01	***	Yessenia	Martin	2021-01-15	2021-01-15

# Entity Tables | Bank\_Accounts | Shipments | Employees

15, 16, 17

SQL Plus

```
SQL> SELECT TO_CHAR(bank_acct_id, '99999999') bank_acct_id, bank_acct_status, pay_id, trans_id, bank_name, routing_num, bank_acct_num, first_name, last_name,
  2 TO_CHAR(bank_created_date, 'YYYY-MM-DD') bank_created_date,
  3 TO_CHAR(bank_last_mod_date, 'YYYY-MM-DD') bank_last_mod_date
  4 FROM bank_accounts;
```

BANK_ACCT_ID	BANK_ACCT_STATUS	PAY_ID	TRANS_ID	BANK_NAME	ROUTING_NUM	BANK_ACCT_NUM	FIRST_NAME	LAST_NAME	BANK_CREATED_DATE	BANK_LAST_MOD_DATE
99990001	Open	60000001	77770001	Wells Fargo Bank	*****82	1214151820	Benjamin	Nichols	2021-01-15	2021-01-15
99990002	Open	60000002	77770002	Star One Credit Union	*****83	1214151821	Tyler	Brown	2021-01-15	2021-01-15
99990003	Open	60000003	77770003	Citibank	*****84	1214151822	Richard	Robinson	2021-01-15	2021-01-15
99990004	Open	60000004	77770004	JPMorgan Chase Bank	*****85	1214151823	Peter	Fernandez	2021-01-15	2021-01-15
99990005	Open	60000005	77770005	Technology Credit Union	*****86	1214151824	Henry	Myers	2021-01-15	2021-01-15
99990006	Open	60000006	77770006	First Republic Bank	*****87	1214151825	Yessenia	Martin	2021-01-15	2021-01-15

SQL Plus

```
SQL> SELECT ship_num_id, ship_status, ship_method_id, ship_track_id,
  2 TO_CHAR(ship_chrg_amt, '$999.99') ship_chrg_amt,
  3 TO_CHAR(acct_id, '99999999') acct_id,
  4 TO_CHAR(ord_id, '99999999') ord_id,
  5 TO_CHAR(ord_qty, '99') ord_qty,
  6 TO_CHAR(ship_created_date, 'YYYY-MM-DD') ship_created_date,
  7 TO_CHAR(ship_last_mod_date, 'YYYY-MM-DD') ship_last_mod_date
  8 FROM shipments;
```

SHIP_NUM_ID	SHIP_STATUS	SHIP_METHOD_ID	SHIP_TRACK_ID	SHIP_CHR	ACCT_ID	ORD_ID	ORD_QTY	SHIP_CREATED_DATE	SHIP_LAST_MOD_DATE
90000001	Shipped	UPS	1Z123AA10123456701	\$225.00	10000001	50000001	5	2021-01-15	2021-01-15
90000002	Shipped	FedEx	678901234567	\$175.00	10000002	50000002	10	2021-01-15	2021-01-15
90000003	Shipped	DHL	4567123789	\$275.00	10000003	50000003	15	2021-01-15	2021-01-15
90000004	Shipped	USPS	345690101234700011	\$195.00	10000004	50000004	20	2021-01-15	2021-01-15
90000005	Shipped	UPS	1Z123AA10123456702	\$185.00	10000005	50000005	25	2021-01-15	2021-01-15
90000006	Shipped	FedEx	678901234568	\$265.00	10000006	50000006	30	2021-01-15	2021-01-15

SQL Plus

```
SQL> SELECT emp_id, emp_status, emp_first, emp_last, emp_phone, emp_email, emp_s_num, emp_street, emp_city, emp_state, emp_zip,
  2 TO_CHAR(emp_created_date, 'YYYY-MM-DD') emp_created_date,
  3 TO_CHAR(emp_last_mod_date, 'YYYY-MM-DD') emp_last_mod_date
  4 FROM employees;
```

EMP_ID	EMP_STATUS	EMP_FIRST	EMP_LAST	EMP_PHONE	EMP_EMAIL	EMP_S_NUM	EMP_STREET	EMP_CITY	EMP_STATE	EMP_ZIP	EMP_CREATED_DATE	EMP_LAST_MOD_DATE
11110001	Active	Junior	Slover	4101435118	Junior.Slover@nexus.com	85763	Shawn Meadows	Port Xavierbury	MD	94532	2020-01-16	2021-01-16
11110002	Active	Kristine	Williams	8610195306	Kristine.Williams@nexus.com	609	Vaughan Estate	Lake Roberthaven	IA	65140	2020-01-16	2021-01-16
11110003	Active	Todd	Newcomb	8656611223	Todd.Newcomb@nexus.com	660	Jessica Forges	East Dannybury	ID	22252	2020-01-16	2021-01-16
11110004	Active	James	Neumayer	4412561388	James.Neumayer@nexus.com	28669	Robinson Orchard	South Matthew	KY	29977	2020-01-16	2021-01-16
11110005	Active	Homer	Glasper	4378491568	Homer.Glasper@nexus.com	471	Hebert Passage	Port Anthony	RI	82814	2020-01-16	2021-01-16
11110006	Active	Stephanie	Doutt	2696616752	Stephanie.Doutt@nexus.com	8601	Robert Burg	Chelseamouth	MS	26315	2020-01-16	2021-01-16

## Entity Tables | Sales\_Data

18

SQL Plus												
SELECT acct_id, acct_name, cat_id, cat_desc, cat_class_id, cat_class_desc, ord_id, ord_qty,												
2 TO CHAR(ord_price, '\$99,999.99') ord_price,												
3 TO CHAR(ord_total, '\$999,999.99') ord_total, ord_status,												
4 TO CHAR(ord_created_date, 'YYYY-MM-DD') ord_created_date,												
5 TO CHAR(ord_last_mod_date, 'YYYY-MM-DD') ord_last_mod_date												
6 FROM sales_data;												
ACCT_ID	ACCT_NAME	CAT_ID	CAT_DESC	CAT_CLASS_ID	CAT_CLASS_DESC	ORD_ID	ORD_QTY	ORD_PRICE	ORD_TOTAL	ORD_STATUS	ORD_CREATED_DATE	ORD_LAST_MOD_DATE
10000003	Apple	20000003	Adjustable Bed	30000003	Patient Mobility	50000003	15	\$4,450.00	\$66,750.00	Invoiced	2021-01-15	2021-01-15
10000003	Apple	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000029	23	\$15,450.00	\$355,350.00	Invoiced	2020-12-15	2021-06-14
10000003	Apple	20000005	Glucose Monitor	30000005	Diabetic Products	50000030	16	\$925.00	\$14,800.00	Invoiced	2020-11-15	2021-06-14
10000003	Apple	20000006	Wound Dressing	30000006	Skin and Wound Care	50000031	30	\$525.00	\$15,750.00	Invoiced	2020-10-15	2021-06-14
10000003	Apple	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000032	27	\$325.00	\$8,775.00	Invoiced	2020-09-15	2021-06-14
10000003	Apple	20000002	Bed Linen	30000002	Patient Care	50000033	21	\$425.00	\$8,925.00	Invoiced	2020-08-15	2021-06-14
10000003	Apple	20000003	Adjustable Bed	30000003	Patient Mobility	50000034	29	\$4,450.00	\$129,050.00	Invoiced	2020-07-15	2021-06-14
10000003	Apple	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000035	19	\$15,450.00	\$293,550.00	Invoiced	2020-06-15	2021-06-14
10000003	Apple	20000005	Glucose Monitor	30000005	Diabetic Products	50000036	29	\$925.00	\$26,825.00	Invoiced	2020-05-15	2021-06-14
10000003	Apple	20000006	Wound Dressing	30000006	Skin and Wound Care	50000037	20	\$525.00	\$10,500.00	Invoiced	2020-04-15	2021-06-14
10000003	Apple	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000038	26	\$325.00	\$8,450.00	Invoiced	2020-03-15	2021-06-14
10000003	Apple	20000002	Bed Linen	30000002	Patient Care	50000039	11	\$425.00	\$4,675.00	Invoiced	2020-02-15	2021-06-14
10000004	Amazon	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000040	20	\$15,450.00	\$309,000.00	Invoiced	2021-01-15	2021-01-15
10000004	Amazon	20000005	Glucose Monitor	30000005	Diabetic Products	50000040	30	\$925.00	\$27,750.00	Invoiced	2020-12-15	2021-06-14
10000004	Amazon	20000006	Wound Dressing	30000006	Skin and Wound Care	50000041	33	\$525.00	\$17,325.00	Invoiced	2020-11-15	2021-06-14
10000004	Amazon	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000042	17	\$325.00	\$5,525.00	Invoiced	2020-10-15	2021-06-14
10000004	Amazon	20000002	Bed Linen	30000002	Patient Care	50000043	28	\$425.00	\$11,900.00	Invoiced	2020-09-15	2021-06-14
10000004	Amazon	20000003	Adjustable Bed	30000003	Patient Mobility	50000044	24	\$4,450.00	\$106,800.00	Invoiced	2020-08-15	2021-06-14
10000004	Amazon	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000045	22	\$15,450.00	\$339,900.00	Invoiced	2020-07-15	2021-06-14
10000004	Amazon	20000005	Glucose Monitor	30000005	Diabetic Products	50000046	29	\$925.00	\$26,825.00	Invoiced	2020-06-15	2021-06-14
10000004	Amazon	20000006	Wound Dressing	30000006	Skin and Wound Care	50000047	18	\$525.00	\$9,450.00	Invoiced	2020-05-15	2021-06-14
10000004	Amazon	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000048	21	\$325.00	\$6,625.00	Invoiced	2020-04-15	2021-06-14
10000004	Amazon	20000002	Bed Linen	30000002	Patient Care	50000049	19	\$425.00	\$8,075.00	Invoiced	2020-03-15	2021-06-14
10000004	Amazon	20000003	Adjustable Bed	30000003	Patient Mobility	50000050	5	\$4,450.00	\$22,250.00	Invoiced	2020-02-15	2021-06-14
10000005	UnitedHealth Group	20000005	Glucose Monitor	30000005	Diabetic Products	50000005	25	\$925.00	\$23,125.00	Invoiced	2021-01-15	2021-01-15
10000005	UnitedHealth Group	20000006	Wound Dressing	30000006	Skin and Wound Care	50000051	23	\$525.00	\$12,175.00	Invoiced	2020-12-15	2021-06-14
10000005	UnitedHealth Group	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000052	18	\$325.00	\$5,850.00	Invoiced	2020-11-15	2021-06-14
10000005	UnitedHealth Group	20000002	Bed Linen	30000002	Patient Care	50000053	15	\$425.00	\$6,375.00	Invoiced	2020-10-15	2021-06-14
10000005	UnitedHealth Group	20000003	Adjustable Bed	30000003	Patient Mobility	50000054	30	\$4,450.00	\$133,500.00	Invoiced	2020-09-15	2021-06-14
10000005	UnitedHealth Group	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000055	21	\$15,450.00	\$324,450.00	Invoiced	2020-08-15	2021-06-14
10000005	UnitedHealth Group	20000005	Glucose Monitor	30000005	Diabetic Products	50000056	19	\$925.00	\$17,575.00	Invoiced	2020-07-15	2021-06-14
10000005	UnitedHealth Group	20000006	Wound Dressing	30000006	Skin and Wound Care	50000057	27	\$525.00	\$14,175.00	Invoiced	2020-06-15	2021-06-14
10000005	UnitedHealth Group	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000058	26	\$325.00	\$8,450.00	Invoiced	2020-05-15	2021-06-14
10000005	UnitedHealth Group	20000002	Bed Linen	30000002	Patient Care	50000059	7	\$425.00	\$2,975.00	Invoiced	2020-04-15	2021-06-14
10000005	UnitedHealth Group	20000003	Adjustable Bed	30000003	Patient Mobility	50000060	16	\$4,450.00	\$71,200.00	Invoiced	2020-03-15	2021-06-14
10000005	UnitedHealth Group	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000061	29	\$15,450.00	\$448,050.00	Invoiced	2020-02-15	2021-06-14
10000006	McKesson	20000006	Wound Dressing	30000006	Skin and Wound Care	50000060	30	\$525.00	\$15,750.00	Invoiced	2021-01-15	2021-01-15
10000006	McKesson	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000062	20	\$325.00	\$6,500.00	Invoiced	2020-12-15	2021-06-14
10000006	McKesson	20000002	Bed Linen	30000002	Patient Care	50000063	22	\$425.00	\$9,350.00	Invoiced	2020-11-15	2021-06-14
10000006	McKesson	20000003	Adjustable Bed	30000003	Patient Mobility	50000064	24	\$4,450.00	\$106,800.00	Invoiced	2020-10-15	2021-06-14
10000006	McKesson	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000065	29	\$15,450.00	\$448,050.00	Invoiced	2020-09-15	2021-06-14
10000006	McKesson	20000005	Glucose Monitor	30000005	Diabetic Products	50000066	11	\$925.00	\$10,175.00	Invoiced	2020-08-15	2021-06-14
10000006	McKesson	20000006	Wound Dressing	30000006	Skin and Wound Care	50000067	12	\$525.00	\$6,300.00	Invoiced	2020-07-15	2021-06-14
10000006	McKesson	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000068	14	\$325.00	\$4,550.00	Invoiced	2020-06-15	2021-06-14
10000006	McKesson	20000002	Bed Linen	30000002	Patient Care	50000069	16	\$425.00	\$6,800.00	Invoiced	2020-05-15	2021-06-14
10000006	McKesson	20000003	Adjustable Bed	30000003	Patient Mobility	50000070	18	\$4,450.00	\$80,100.00	Invoiced	2020-04-15	2021-06-14
10000006	McKesson	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000071	19	\$15,450.00	\$293,550.00	Invoiced	2020-03-15	2021-06-14
10000006	McKesson	20000005	Glucose Monitor	30000005	Diabetic Products	50000072	21	\$925.00	\$19,425.00	Invoiced	2020-02-15	2021-06-14
10000001	Walmart	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000001	5	\$325.00	\$1,625.00	Invoiced	2021-01-15	2021-01-15
10000001	Walmart	20000002	Bed Linen	30000002	Patient Care	50000007	9	\$425.00	\$3,250.00	Invoiced	2020-12-15	2021-06-14
10000001	Walmart	20000003	Adjustable Bed	30000003	Patient Mobility	50000008	12	\$4,450.00	\$53,400.00	Invoiced	2020-11-15	2021-06-14
10000001	Walmart	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000009	15	\$15,450.00	\$231,750.00	Invoiced	2020-10-15	2021-06-14
10000001	Walmart	20000005	Glucose Monitor	30000005	Diabetic Products	50000010	18	\$925.00	\$16,650.00	Invoiced	2020-09-15	2021-06-14
10000001	Walmart	20000006	Wound Dressing	30000006	Skin and Wound Care	50000011	23	\$525.00	\$12,075.00	Invoiced	2020-08-15	2021-06-14
10000001	Walmart	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	50000012	27	\$325.00	\$8,775.00	Invoiced	2020-07-15	2021-06-14
10000001	Walmart	20000002	Bed Linen	30000002	Patient Care	50000013	30	\$425.00	\$12,750.00	Invoiced	2020-06-15	2021-06-14
10000001	Walmart	20000003	Adjustable Bed	30000003	Patient Mobility	50000014	7	\$4,450.00	\$31,150.00	Invoiced	2020-05-15	2021-06-14
10000001	Walmart	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000015	11	\$15,450.00	\$169,950.00	Invoiced	2020-04-15	2021-06-14
10000001	Walmart	20000005	Glucose Monitor	30000005	Diabetic Products	50000016	10	\$925.00	\$9,250.00	Invoiced	2020-03-15	2021-06-14
10000001	Walmart	20000006	Wound Dressing	30000006	Skin and Wound Care	50000017	21	\$525.00	\$11,250.00	Invoiced	2020-02-15	2021-06-14
10000002	EXXON MOBIL	20000002	Bed Linen	30000002	Patient Care	50000002	10	\$425.00	\$4,250.00	Invoiced	2021-01-15	2021-01-15
10000002	EXXON MOBIL	20000003	Adjustable Bed	30000003	Patient Mobility	50000018	17	\$4,450.00	\$75,650.00	Invoiced	2020-12-15	2021-06-14
10000002	EXXON MOBIL	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	50000019	12	\$15,450.00	\$185,400.00	Invoiced	2020-11-15	2021-06-14
10000002	EXXON MOBIL	20000005	Glucose Monitor	30000005	Diabetic Products	50000020	19	\$925.00	\$17,575.00	Invoiced	2020-10-15	2021-06-14
10000002	EXXON MOBIL	20000006	Wound Dressing	30000006	Skin and Wound Care	5000002						

SQL Plus

```
SQL> SELECT TO_CHAR(acct_id, '99999999') acct_id, acct_name,
  2  TO_CHAR(cat_id, '99999999') cat_id, cat_desc,
  3  TO_CHAR(cat_class_id, '99999999') cat_class_id, cat_class_desc,
  4  REGR_SLOPE(SYSDATE-ord_created_date, ord_qty) ord_qty_slope,
  5  REGR_INTERCEPT(SYSDATE-ord_created_date, ord_qty) ord_qty_intercept,
  6  REGR_SLOPE(SYSDATE-ord_created_date, ord_total) ord_total_slope,
  7  REGR_INTERCEPT(SYSDATE-ord_created_date, ord_total) ord_total_intercept
  8  FROM sales_data
  9  WHERE acct_id BETWEEN 10000001 AND 10000006
 10 GROUP BY acct_id, acct_name, cat_id, cat_desc, cat_class_id, cat_class_desc
 11 ORDER BY acct_id;
```

ACCT_ID	ACCT_NAME	CAT_ID	CAT_DESC	CAT_CLASS_ID	CAT_CLASS_DESC	ORD_QTY_SLOPE	ORD_QTY_INTERCEPT	ORD_TOTAL_SLOPE	ORD_TOTAL_INTERCEPT
10000001	Walmart	20000002	Bed Linen	30000002	Patient Care	8.71428571	96.5287665	.020504202	96.5287665
10000001	Walmart	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	8.36363636	102.139156	.025734266	102.139156
10000001	Walmart	20000005	Glucose Monitor	30000005	Diabetic Products	-23	679.957338	-.02486486	679.957338
10000001	Walmart	20000006	Wound Dressing	30000006	Skin and Wound Care	-91	2389.95734	-.17333333	2389.95734
10000001	Walmart	20000003	Adjustable Bed	30000003	Patient Mobility	-36.8	646.557338	-.00826966	646.557338
10000001	Walmart	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	-45.75	922.207338	-.00296117	922.207338
10000002	EXXON MOBIL	20000003	Adjustable Bed	30000003	Patient Mobility	-183	3285.95734	-.0411236	3285.95734
10000002	EXXON MOBIL	20000006	Wound Dressing	30000006	Skin and Wound Care	-92	2473.95734	-.1752381	2473.95734
10000002	EXXON MOBIL	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	-45.5	1616.45734	-.14	1616.45734
10000002	EXXON MOBIL	20000002	Bed Linen	30000002	Patient Care	10.8235294	35.7220438	.025467128	35.7220438
10000002	EXXON MOBIL	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	30.6666667	-163.04266	.001984898	-163.04266
10000002	EXXON MOBIL	20000005	Glucose Monitor	30000005	Diabetic Products	-36.6	931.357338	-.03956757	931.357338
10000003	Apple	20000003	Adjustable Bed	30000003	Patient Mobility	13.1428571	-53.185519	.002953451	-53.185519
10000003	Apple	20000002	Bed Linen	30000002	Patient Care	-18.2	679.157338	-.04282353	679.157338
10000003	Apple	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	-45.75	1227.20734	-.00296117	1227.20734
10000003	Apple	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	-184	5233.95734	-.56615385	5233.95734
10000003	Apple	20000006	Wound Dressing	30000006	Skin and Wound Care	-18.3	784.957338	-.03485714	784.957338
10000003	Apple	20000005	Glucose Monitor	30000005	Diabetic Products	14.1538462	-21.5042	.015301455	-21.5042
10000004	Amazon	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	92	-1696.0427	.005954693	-1696.0427
10000004	Amazon	20000006	Wound Dressing	30000006	Skin and Wound Care	-12.266667	609.757338	-.02336508	609.757338
10000004	Amazon	20000002	Bed Linen	30000002	Patient Care	-20.444444	838.401782	-.04810458	838.401782
10000004	Amazon	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	45.75	-541.79266	.140769231	-541.79266
10000004	Amazon	20000005	Glucose Monitor	30000005	Diabetic Products	-183	5664.95734	-.19783784	5664.95734
10000004	Amazon	20000003	Adjustable Bed	30000003	Patient Mobility	-9.5789474	526.852075	-.00215257	526.852075
10000005	UnitedHealth Group	20000005	Glucose Monitor	30000005	Diabetic Products	-30.666667	910.624005	-.03315315	910.624005
10000005	UnitedHealth Group	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	23	-209.04266	.070769231	-209.04266
10000005	UnitedHealth Group	20000002	Bed Linen	30000002	Patient Care	-22.875	579.082338	-.05382353	579.082338
10000005	UnitedHealth Group	20000006	Wound Dressing	30000006	Skin and Wound Care	45.75	-877.29266	.087142857	-877.29266
10000005	UnitedHealth Group	20000003	Adjustable Bed	30000003	Patient Mobility	-13.142857	660.243052	-.00295345	660.243052
10000004	UnitedHealth Group	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	22.75	-180.79266	.001472492	-180.79266
10000006	McKesson	20000003	Adjustable Bed	30000003	Patient Mobility	-30.5	967.957338	-.00685393	967.957338
10000006	McKesson	20000002	Bed Linen	30000002	Patient Care	-30.666667	879.624005	-.07215686	879.624005
10000006	McKesson	20000004	Hydrotherapy Fitness	30000004	Rehabilitation	-18.4	799.557338	-.00119094	799.557338
10000006	McKesson	20000005	Glucose Monitor	30000005	Diabetic Products	18.2	96.757338	.019675676	96.757338
10000006	McKesson	20000001	COVID-19 Test Kit	30000001	COVID-19 Supplies	-30.5	784.957338	-.09384615	784.957338
10000006	McKesson	20000006	Wound Dressing	30000006	Skin and Wound Care	-10.222222	450.624005	-.0194709	450.624005

SQL Plus

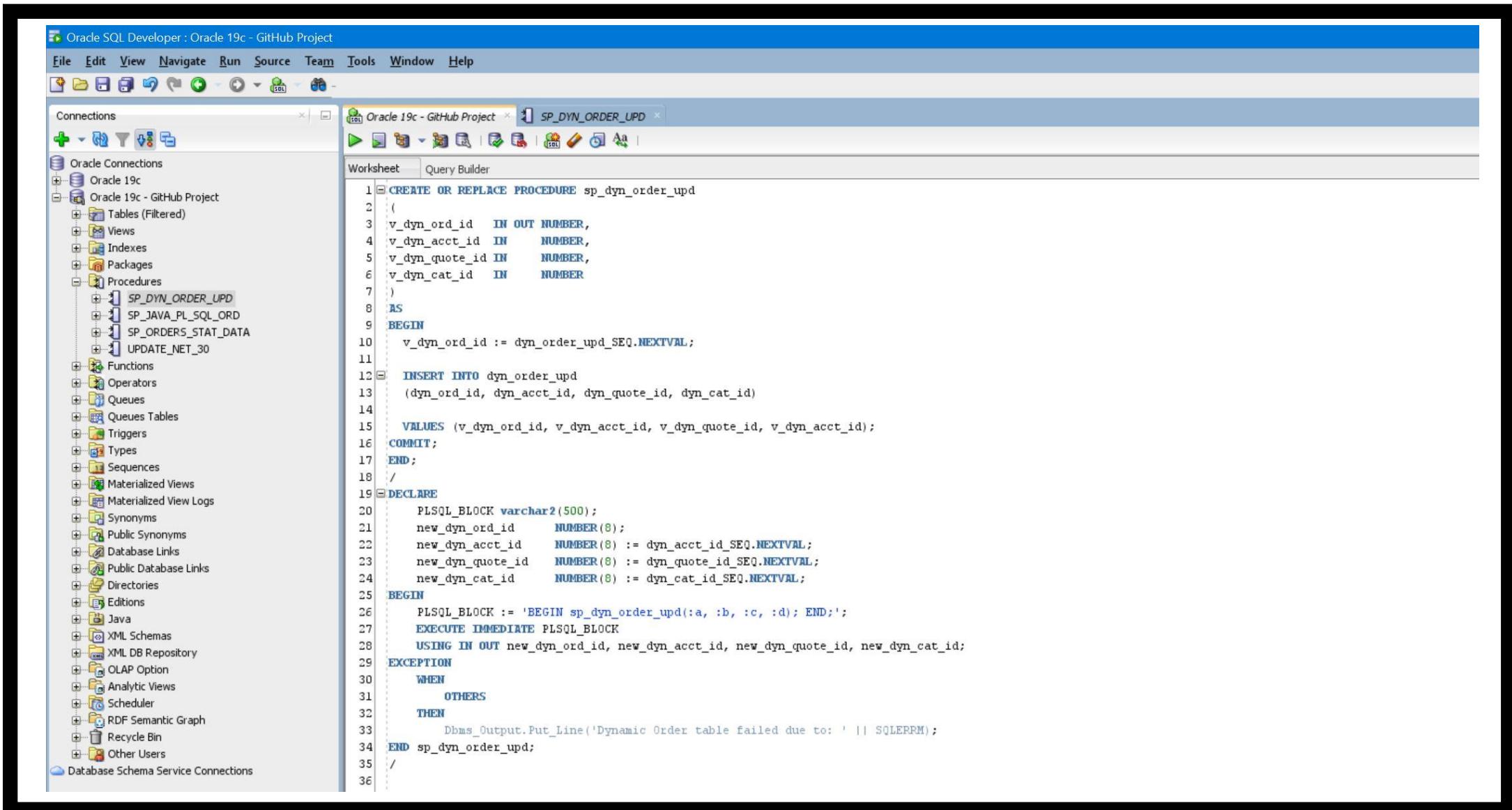
```
SQL> SELECT owner, constraint_name, constraint_type, table_name
  2  FROM user_constraints
  3 WHERE constraint_type = 'P'
  4 MINUS
  5 SELECT owner, constraint_name, constraint_type, table_name
  6 FROM user_constraints
  7 WHERE constraint_name LIKE 'bin%'
  8 AND owner = 'FREESTYLE7'
  9 ORDER BY table_name;
```

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
FREESTYLE7	ACCT_ID_PK	P	ACCOUNTS
FREESTYLE7	BANK_ACCT_ID_PK	P	BANK_ACCOUNTS
FREESTYLE7	CAT_ID_PK	P	CATALOG
FREESTYLE7	CAT_CLASS_ID_PK	P	CATALOG_CLASS
FREESTYLE7	CC_ID_PK	P	CREDIT_CARDS
FREESTYLE7	DYN_ORD_ID_PK	P	DYN_ORDER_UPD
FREESTYLE7	EMP_ID_PK	P	EMPLOYEES
FREESTYLE7	INV_ID_PK	P	INVENTORY
FREESTYLE7	INVOICE_ID_PK	P	INVOICES
FREESTYLE7	JAVA_ORD_ID_PK	P	JAVA_PL_SQL_ORD
FREESTYLE7	LEAD_ID_PK	P	LEADS
FREESTYLE7	NET_30_ID_PK	P	NET_30
FREESTYLE7	ORD_ID_PK	P	ORDERS
FREESTYLE7	MEAN_VAL_PK	P	ORDERS_STAT_DATA
FREESTYLE7	PAY_ID_PK	P	PAYMENTS
FREESTYLE7	SHIP_NUM_ID_PK	P	SHIPMENTS
FREESTYLE7	TRANS_ID_PK	P	TRANSACTIONS

SQL Plus

```
SQL> SELECT owner, constraint_name, constraint_type, table_name
  2  FROM user_constraints
  3 WHERE constraint_type = 'R'
  4 MINUS
  5 SELECT owner, constraint_name, constraint_type, table_name
  6  FROM user_constraints
  7 WHERE constraint_name LIKE 'bin%'
  8 AND owner = 'FREESTYLE7'
  9 ORDER BY table_name;
```

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
FREESTYLE7	EMP_ID_FK_1	R	ACCOUNTS
FREESTYLE7	PAY_ID_FK_1	R	BANK_ACCOUNTS
FREESTYLE7	TRANS_ID_FK_1	R	BANK_ACCOUNTS
FREESTYLE7	ORD_ID_FK_1	R	CATALOG
FREESTYLE7	CAT_ID_FK_1	R	CATALOG_CLASS
FREESTYLE7	ACCT_ID_FK_1	R	CREDIT_CARDS
FREESTYLE7	PAY_ID_FK_2	R	CREDIT_CARDS
FREESTYLE7	CAT_ID_FK_2	R	INVENTORY
FREESTYLE7	INVOICE_ID_FK_1	R	INVENTORY
FREESTYLE7	ORD_ID_FK_2	R	INVENTORY
FREESTYLE7	CAT_CLASS_ID_FK_1	R	INVOICES
FREESTYLE7	CAT_ID_FK_3	R	INVOICES
FREESTYLE7	PAY_ID_FK_3	R	NET_30
FREESTYLE7	ACCT_ID_FK_2	R	NET_30
FREESTYLE7	ACCT_ID_FK_3	R	ORDERS
FREESTYLE7	CAT_ID_FK_4	R	ORDERS
FREESTYLE7	TRANS_ID_FK_2	R	PAYMENTS
FREESTYLE7	CAT_ID_FK_5	R	SALES_DATA
FREESTYLE7	ACCT_ID_FK_4	R	SHIPMENTS
FREESTYLE7	ORD_ID_FK_3	R	SHIPMENTS
FREESTYLE7	ACCT_ID_FK_5	R	TRANSACTIONS



The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle 19c - GitHub Project
- Worksheet:** SP\_DYN\_ORDER\_UPD
- Code:** The code for the stored procedure SP\_DYN\_ORDER\_UPD is displayed in the worksheet.

```
CREATE OR REPLACE PROCEDURE sp_dyn_order_upd
(
  v_dyn_ord_id    IN OUT NUMBER,
  v_dyn_acct_id   IN      NUMBER,
  v_dyn_quote_id  IN      NUMBER,
  v_dyn_cat_id   IN      NUMBER
)
AS
BEGIN
  v_dyn_ord_id := dyn_order_upd_SEQ.NEXTVAL;
  INSERT INTO dyn_order_upd
  (dyn_ord_id, dyn_acct_id, dyn_quote_id, dyn_cat_id)
  VALUES (v_dyn_ord_id, v_dyn_acct_id, v_dyn_quote_id, v_dyn_acct_id);
  COMMIT;
END;
/
DECLARE
  PLSQL_BLOCK varchar2(500);
  new_dyn_ord_id      NUMBER(8);
  new_dyn_acct_id     NUMBER(8) := dyn_acct_id_SEQ.NEXTVAL;
  new_dyn_quote_id    NUMBER(8) := dyn_quote_id_SEQ.NEXTVAL;
  new_dyn_cat_id     NUMBER(8) := dyn_cat_id_SEQ.NEXTVAL;
BEGIN
  PLSQL_BLOCK := 'BEGIN sp_dyn_order_upd(:a, :b, :c, :d); END;';
  EXECUTE IMMEDIATE PLSQL_BLOCK
  USING IN OUT new_dyn_ord_id, new_dyn_acct_id, new_dyn_quote_id, new_dyn_cat_id;
EXCEPTION
  WHEN
    OTHERS
  THEN
    DBMS_OUTPUT.PUT_LINE('Dynamic Order table failed due to: ' || SQLERRM);
END sp_dyn_order_upd;
/

```

SQL> SELECT \* FROM dyn\_order\_upd;

DYN_ORD_ID	DYN_ACCT_ID	DYN_QUOTE_ID	DYN_CAT_ID
22220001	33330001	44440001	12340001
22220002	33330002	44440002	12340002
43210001	33330003	44440003	12340003
43210002	33330004	44440004	12340004
43210003	33330005	44440005	12340005
43210004	33330006	44440006	12340006
22220005	33330007	44440007	12340007
43210005	33330008	44440008	12340008

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema, including tables, views, indexes, packages, procedures, and functions. A specific function named `NET_30_AVAIL_CRED` is selected under the `Functions` category. The main workspace contains the PL/SQL code for this function:

```
1 CREATE OR REPLACE FUNCTION net_30_avail_cred
2   RETURN NUMBER
3 IS
4   v_net_30_avail_cred NUMBER;
5 BEGIN
6   SELECT net_30_avail_cred
7     INTO v_net_30_avail_cred
8     FROM net_30_info
9    WHERE acct_id = 1000001;
10   RETURN v_net_30_avail_cred;
11 END net_30_avail_cred;
12
13 DECLARE
14   v_n30 NUMBER;
15 BEGIN
16   v_n30 := net_30_avail_cred;
17   Dbms_Output.Put_Line('Net 30 Available Credit: ' || TO_CHAR(v_n30, '$99,999.99'));
18 END;
```

The bottom pane shows the output of the script execution:

```
Net 30 Available Credit: $10,000.00
PL/SQL procedure successfully completed.
```

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - Java\_PLSQL\_Net\_30/src/com/oracle/net30package/Java\_PLSQL\_Net\_30.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbars:** Standard toolbar with various icons.
- Left Sidebar (Package Explorer):** Lists Java files and packages related to a hospital database, including Java\_Hospital\_DB classes and Java\_PLSQL\_Net\_30.
- Central Editor:** Displays the Java code for `Java_PLSQL_Net_30.java`. The code connects to an Oracle database using JDBC and CallableStatement to execute a stored procedure that retrieves available credit for an account.

```
1 package com.oracle.net30package;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.SQLException;
5 import java.sql.Types;
6 import java.sql.CallableStatement;
7 import java.text.NumberFormat;
8 import org.slf4j.Logger;
9 import org.slf4j.LoggerFactory;
10
11 public class Java_PLSQL_Net_30 {
12
13    public static void main(String[] args) throws SQLException {
14
15        Logger logger = LoggerFactory.getLogger(Java_PLSQL_Net_30.class);
16
17        try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:username/password@//localhost:1521/pdb");
18             CallableStatement cstmt = conn.prepareCall (" {? = call NET_30_AVAIL_CRED() }")) {
19            cstmt.registerOutParameter(1, Types.INTEGER);
20            cstmt.setInt(1, 10000001);
21            cstmt.execute();
22
23            NumberFormat defaultFormat = NumberFormat.getCurrencyInstance();
24            System.out.println("Net 30 Available Credit for Account ID 10000001: " + defaultFormat.format(cstmt.getInt(1)));
25
26        } catch (SQLException e) {
27            logger.warn(e.getMessage(), e);
28        }
29    }
30 }
```

- Right Sidebar (Outline):** Shows the class structure with a main method.
- Bottom Status Bar:** Shows the output of the Java application: "Net 30 Available Credit for Account ID 10000001: \$10,000.00".

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java and database-related projects and files.
- Java\_PLSQL\_Order.java:** The main source file containing the Java code.
- Outline View:** Shows the class structure and the main method.
- Code Editor:** Displays the Java code for calling a PL/SQL stored procedure.
- Console Output:** Shows the output of the Java application, indicating successful connection to Oracle 19c.

```
package com.oracle.datapackage;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class Java_PLSQL_Order {
    public static void main(String[] args) throws SQLException {
        String query = "SELECT * FROM username.JAVA_PL_SQL_ORD";
        Logger logger = LoggerFactory.getLogger(Java_PLSQL_Order.class);
        try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:username/password@//localhost:1521/pdb");
             Statement stmt = conn.createStatement();
             ResultSet rset = stmt.executeQuery(query));
        while (rset.next()) {
            int javaOrdID   = rset.getInt("JAVA_ORD_ID");
            int javaAcctID = rset.getInt("JAVA_ACCT_ID");
            int javaQuoteID = rset.getInt("JAVA_QUOTE_ID");
            int javaCatID   = rset.getInt("JAVA_CAT_ID");
            System.out.print("JAVA_ORD_ID: " + javaOrdID);
            System.out.print(", JAVA_ACCT_ID: " + javaAcctID);
            System.out.print(", JAVA_QUOTE_ID: " + javaQuoteID);
            System.out.println(", JAVA_CAT_ID: " + javaCatID);
        }
        rset.close();
    } catch (SQLException e) {
        logger.warn(e.getMessage(), e);
    }
    System.out.println();
    System.out.println("Connected to Oracle 19c | PL/SQL database successfully.");
    System.out.println("No Oracle 19c, PL/SQL, Java, or connection errors to report.");
}
}
```

Console Output:

```
<terminated> Java_PLSQL_Order [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (Jul 19, 2021, 11:10:41 PM - 11:10:41 PM)
JAVA_ORD_ID: 1001, JAVA_ACCT_ID: 2001, JAVA_QUOTE_ID: 3001, JAVA_CAT_ID: 4001
JAVA_ORD_ID: 1002, JAVA_ACCT_ID: 2002, JAVA_QUOTE_ID: 3002, JAVA_CAT_ID: 4002
JAVA_ORD_ID: 1003, JAVA_ACCT_ID: 2003, JAVA_QUOTE_ID: 3003, JAVA_CAT_ID: 4003
JAVA_ORD_ID: 1004, JAVA_ACCT_ID: 2004, JAVA_QUOTE_ID: 3004, JAVA_CAT_ID: 4004
JAVA_ORD_ID: 1005, JAVA_ACCT_ID: 2005, JAVA_QUOTE_ID: 3005, JAVA_CAT_ID: 4005

Connected to Oracle 19c | PL/SQL database successfully.
No Oracle 19c, PL/SQL, Java, or connection errors to report.
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows various Java files and packages related to a hospital database.
- Java\_PLSQL\_OrdersStatData.java:** The main source file containing the Java code to call a PL/SQL stored procedure.
- Code Snippet:**

```
1 package com.oracle.ordersstatdata;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.SQLException;
5 import java.sql.Types;
6 import java.sql.CallableStatement;
7 import org.slf4j.Logger;
8 import org.slf4j.LoggerFactory;
9
10 public class Java_PLSQL_OrdersStatData {
11
12     private Java_PLSQL_OrdersStatData() {
13     }
14
15     public static void main(String[] args) throws SQLException {
16
17         Logger logger = LoggerFactory.getLogger(Java_PLSQL_OrdersStatData.class);
18
19         try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:username/password@//localhost:1521/pdb");
20              CallableStatement cstmt = conn.prepareCall ("{call SP_ORDERS_STAT_DATA(?, ?, ?)}")) {
21
22             cstmt.registerOutParameter(1, Types.FLOAT);
23             cstmt.registerOutParameter(2, Types.FLOAT);
24             cstmt.registerOutParameter(3, Types.FLOAT);
25
26             cstmt.execute();
27
28             cstmt.getFloat(1);
29             cstmt.getFloat(2);
30             cstmt.getFloat(3);
31
32             System.out.print("Mean Value: $%,.4f\n", cstmt.getFloat(1));
33             System.out.print("Median Value: $%,.4f\n", cstmt.getFloat(2));
34             System.out.print("Std. Dev. Value: $%,.4f\n", cstmt.getFloat(3));
35
36         } catch (SQLException e) {
37             logger.warn(e.getMessage(), e);
38         }
39     }
40 }
```

- Outline View:** Shows the class structure and methods.
- Console:** Displays the output of the program execution:

```
<terminated> Java_PLSQL_OrdersStatData [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (Jul 19, 2021, 11:14:32 PM - 11:14:32 PM)
Mean Value: $71,815.2813
Median Value: $14,487.5000
Std. Dev. Value: $114,798.2734
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "Java\_Hospital\_DB\_Diagnoses".
- Java\_Hospital\_DB\_Diagnoses.java:** The code is as follows:

```
1 package com.oracle.hospital_db_diagnoses;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.SQLException;
5 import java.sql.Statement;
6 import org.slf4j.Logger;
7 import org.slf4j.LoggerFactory;
8
9 public class Java_Hospital_DB_Diagnoses {
10     public static void main(String[] args) throws SQLException {
11         String query_1 = "CREATE TABLE diagnoses (" +
12             "diag_id int PRIMARY KEY NOT NULL," +
13             "diag_desc varchar2(50)," +
14             "doc_id int," +
15             "hosp_id int");
16
17         Logger logger = LoggerFactory.getLogger(Java_Hospital_DB_Diagnoses.class);
18
19         try (Connection conn = DriverManager.getConnection("jdbc:oracle:thin:username/password@//localhost:1521/pdb");
20              Statement stmt = conn.createStatement()) {
21             stmt.executeUpdate(query_1);
22
23         } catch (SQLException e) {
24             logger.warn(e.getMessage(), e);
25         }
26         System.out.println("Connected to Hospital_DB database successfully.");
27         System.out.println("Diagnoses table created.");
28     }
29 }
30
31 }
```

- Console:** Displays the output of the Java application.

```
Connected to Hospital_DB database successfully.
Diagnoses table created.
```

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - Java\_Hospital\_DB\_Hospitals\_Data/src/com/oracle/hospital/db\_hospitals\_data/Java\_Hospital\_DB\_Hospitals\_Data.java - Eclipse IDE
- Toolbar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Left Sidebar (Package Explorer):** Shows the project structure with packages like Java\_Hospital\_DB\_Hospitals\_Data and com.oracle.hospital.db.hospitals\_data.
- Central Area (Code Editor):** Displays Java code for updating a hospital database using JDBC. The code includes imports for java.sql.Connection, java.sql.DriverManager, java.sql.SQLException, and java.sql.PreparedStatement. It connects to a local Oracle database and inserts data into the 'hospitals' table. The code handles exceptions and prints confirmation messages to System.out.
- Right Sidebar (Outline):** Shows the class structure with a main method: public static void main(String[] args) throws SQLException { ... }.
- Bottom Status Bar:** Problems Declaration Console <terminated> Java\_Hospital\_DB\_Update.VarChar2 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (Jul 22, 2021, 11:10:03 PM – 11:10:07 PM)

```
1 package com.oracle.hospital.db_hospitals_data;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.SQLException;
5 import java.sql.PreparedStatement;
6
7 public class Java_Hospital_DB_Hospitals_Data {
8
9     public static void main(String[] args) throws SQLException {
10
11         Connection conn = null;
12         PreparedStatement pstmt = null;
13
14         try {
15
16             Class.forName("oracle.jdbc.driver.OracleDriver");
17             conn = DriverManager.getConnection("jdbc:oracle:thin:username/password@/localhost:1521/pdb", "username", "password");
18             System.out.println("Connected to Hospital_DB database successfully.");
19
20             String sql = "INSERT INTO hospitals (" +
21                     + "hosp_id," +
22                     + "hosp_name," +
23                     + "hosp_s_num," +
24                     + "hosp_street," +
25                     + "hosp_city," +
26                     + "hosp_state," +
27                     + "hosp_zip," +
28                     + "hosp_phone)" +
29                     + "VALUES (?, ?, ?, ?, ?, ?, ?, ?);";
30
31             pstmt = conn.prepareStatement(sql);
32
33             pstmt.setInt(1, 10001238);
34             pstmt.setString(2, "Regional Medical Center");
35             pstmt.setInt(3, 225);
36             pstmt.setString(4, "123 North Jackson Avenue");
37             pstmt.setString(5, "San Jose");
38             pstmt.setString(6, "CA");
39             pstmt.setInt(7, 95116);
40             pstmt.setLong(8, 4082595000L);
41             int result = pstmt.executeUpdate();
42             if (result == 0) {
43                 System.out.println("Hospitals data was not updated.");
44             } else {
45                 System.out.println("Hospitals data was updated.");
46             }
47
48         } catch (ClassNotFoundException cnfe) {
49             cnfe.printStackTrace();
50         } catch (SQLException sqle) {
51             sqle.printStackTrace();
52         } catch (Exception ex) {
53             ex.printStackTrace();
54         }
55
56         finally {
57             try {
58                 if(pstmt != null) {
59                     pstmt.close();
60                 }
61             } catch (SQLException sqle) {
62                 sqle.printStackTrace();
63             }
64             try {
65                 if (conn != null) {
66                     conn.close();
67                 }
68             } catch (SQLException sqle) {
69                 sqle.printStackTrace();
70             }
71         }
72     }
73 }
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "Java\_Hospital\_DB\_Update\_Varchar2".
- Code Editor:** Displays the Java code for `Java_Hospital_DB_Update_Varchar2.java`. The code uses JDBC to execute an `ALTER TABLE` statement on an Oracle database.
- Outline View:** Shows the class definition and the `main` method.
- Problems View:** Shows the output of the Java application, indicating successful connection to the database and alteration of the `PROD_DESC` column.

```
package com.oracle.hospital_db_update_varchar2;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class Java_Hospital_DB_Update_Varchar2 {
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        String query = "ALTER TABLE transactions MODIFY prod_desc varchar2(13)";
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521/pdb", "username", "password");
            System.out.println("Connected to Hospital_DB database successfully.");
            stmt = conn.createStatement();
            stmt.executeUpdate(query);
            System.out.println("PROD_DESC column altered.");
        } catch (ClassNotFoundException cnfe) {
            cnfe.printStackTrace();
        } catch (SQLException sqle) {
            sqle.printStackTrace();
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            try {
                if(stmt != null) {
                    stmt.close();
                }
            } catch (SQLException sqle) {
                sqle.printStackTrace();
            }
            try {
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException sqle) {
                sqle.printStackTrace();
            }
        }
    }
}
```

Connected to Hospital\_DB successfully.  
PROD\_DESC column altered.

**Java | JDBC | Select PL/SQL Data | Output**

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - Java\_Hospital\_DB\_Select\_Data[Data\src\com\oracle\hospital\db\select\_data\java\Hospital\_DB\_Select\_Data.java] - Eclipse IDE
- Left Side:** Package Explorer view showing the project structure. The `Java_Hospital_DB_Select_Data` package contains several files: `Hospitals.java`, `Update_Varchar2.java`, `Select_Data.java`, `Diagnoses.java`, `Doctors.java`, `Hospitals.java`, `Invoices.java`, `Invoices_Data.java`, `Patients.java`, `Patients_Data.java`, `Products.java`, `Products_Data.java`, and `Transactions.java`.
- Center:** The code editor displays the `Select_Data.java` file. The code implements a `main` method to query an Oracle database for transactions. It uses JDBC to establish a connection, prepare a statement, execute the query, and then loop through the results to print each transaction's details (trans\_id, hosp\_id, diag\_id, doc\_id, pat\_id, prod\_id, prod\_desc, prod\_qty, prod\_price, trans\_amt, trans\_type, trans\_date).
- Right Side:** The Outline view shows the class definition and the `main` method.
- Bottom:** The Problems view is empty. The Console view shows the output of the executed SQL query, which lists multiple rows of transaction data.

```
package com.oracle.hospital.db.select_data;

import oracle.jdbc.OracleConnection;
import oracle.jdbc.OracleDriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Date;
import java.sql.ResultSet;
import java.sql.Resultset;

public class Java_Hospital_DB_Select_Data {
    public static void main(String[] args) throws SQLException {
        Connection conn = null;
        Statement stmt = null;
        String query = "SELECT trans_id, hosp_id, diag_id, doc_id, pat_id, prod_id, prod_desc, prod_qty, prod_price, trans_amt, trans_type, trans_date FROM transactions WHERE trans_id >= 60006789";
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521/pdb", "username", "password");
            System.out.println("Connected to Hospital_DB database successfully.");
            stmt = conn.createStatement();
            ResultSet rset = stmt.executeQuery(query);
            while(rset.next()) {
                int transID = rset.getInt("TRANS_ID");
                int hospID = rset.getInt("HOSP_ID");
                int diagID = rset.getInt("DIAG_ID");
                int docID = rset.getInt("DOC_ID");
                int patID = rset.getInt("PAT_ID");
                int prodID = rset.getInt("PROD_ID");
                int prodQty = rset.getInt("PROD_QTY");
                float prodPrice = rset.getFloat("PROD_PRICE");
                float transAmt = rset.getFloat("TRANS_AMT");
                String transType = rset.getString("TRANS_TYPE");
                Date date = rset.getDate("TRANS_DATE");

                System.out.print("TRANS_ID: " + transID);
                System.out.print(", HOSP_ID: " + hospID);
                System.out.print(", DIAG_ID: " + diagID);
                System.out.print(", DOC_ID: " + docID);
                System.out.print(", PAT_ID: " + patID);
                System.out.print(", PROD_ID: " + prodID);
                System.out.print(", PROD_QTY: " + prodQty);
                System.out.print(", PROD_PRICE: " + prodPrice);
                System.out.print(", TRANS_AMT: " + transAmt);
                System.out.print(", TRANS_TYPE: " + transType);
                System.out.print(", TRANS_DATE: " + date);
            }
        } catch (ClassNotFoundException cnfe) {
            cnfe.printStackTrace();
        } catch (SQLException sole) {
            sole.printStackTrace();
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            try {
                if(stmt != null) {
                    stmt.close();
                }
            } catch (SQLException sole) {
                sole.printStackTrace();
            }
            try {
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException sole) {
                sole.printStackTrace();
            }
        }
    }
}
```

Connected to Hospital\_DB database successfully.  
Connected to Hospital\_DB database successfully.  
TRANS\_ID: 60006789, HOSP\_ID: 10001234, DIAG\_ID: 40004567, DOC\_ID: 30000245, PROD\_ID: 50005678 PROD\_QTY: 2, PROD\_PRICE: 16.0, TRANS\_AMT: 32.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006790, HOSP\_ID: 10001235, DOC\_ID: 40004568, PROD\_ID: 30000246, PROD\_QTY: 2, PROD\_PRICE: 35.0, TRANS\_AMT: 70.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006791, HOSP\_ID: 10001236, DOC\_ID: 40004569, PROD\_ID: 30000247, PROD\_QTY: 2, PROD\_PRICE: 30.0, TRANS\_AMT: 60.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006792, HOSP\_ID: 10001237, DOC\_ID: 40004570, PROD\_ID: 30000248, PROD\_QTY: 2, PROD\_PRICE: 45.0, TRANS\_AMT: 90.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006793, HOSP\_ID: 10001238, DOC\_ID: 40004571, PROD\_ID: 30000249, PROD\_QTY: 2, PROD\_PRICE: 17.0, TRANS\_AMT: 34.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006795, HOSP\_ID: 10001235, DOC\_ID: 40004568, PROD\_ID: 30000246, PROD\_QTY: 3, PROD\_PRICE: 35.0, TRANS\_AMT: 105.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006796, HOSP\_ID: 10001236, DOC\_ID: 40004569, PROD\_ID: 30000247, PROD\_QTY: 3, PROD\_PRICE: 30.0, TRANS\_AMT: 90.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006797, HOSP\_ID: 10001237, DOC\_ID: 40004570, PROD\_ID: 30000248, PROD\_QTY: 3, PROD\_PRICE: 45.0, TRANS\_AMT: 135.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006798, HOSP\_ID: 10001238, DOC\_ID: 40004571, PROD\_ID: 30000249, PROD\_QTY: 3, PROD\_PRICE: 17.0, TRANS\_AMT: 51.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18  
TRANS\_ID: 60006799, HOSP\_ID: 10001234, DIAG\_ID: 40004567, DOC\_ID: 30000245, PROD\_ID: 50005678 PROD\_QTY: 3, PROD\_PRICE: 16.0, TRANS\_AMT: 48.0, TRANS\_TYPE: Credit Card, TRANS\_DATE: 2021-07-18

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "Java\_Hospital\_DB\_Add\_Foreign\_Keys".
- Code Editor:** Displays the Java code for generating foreign keys. The code uses JDBC to connect to an Oracle database and execute 14 ALTER TABLE statements to add constraints.
- Outline View:** Shows the class definition and the main method.
- Problems View:** Shows a single error message: "Connected to Hospital\_DB successfully. Foreign Keys added successfully."

```
package com.oracle.hospital_db.add_foreign_keys;
import java.sql.Connection;
import java.sql.Statement;
public static void main(String[] args) throws SQLException {
    Connection conn = null;
    Statement stmt = null;
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        conn = DriverManager.getConnection("jdbc:oracle:thin:username/password@//localhost:1521/pdb", "username", "password");
        System.out.println("Connected to Hospital_DB database successfully.");
        stmt = conn.createStatement();
        String query_1 = "ALTER TABLE doctors ADD CONSTRAINT hosp_id_fk_1 FOREIGN KEY (hosp_id) REFERENCES hospitals (hosp_id)";
        String query_2 = "ALTER TABLE diagnoses ADD CONSTRAINT doc_id_fk_1 FOREIGN KEY (doc_id) REFERENCES doctors (doc_id)";
        String query_3 = "ALTER TABLE diagnoses ADD CONSTRAINT diag_id_fk_1 FOREIGN KEY (diag_id) REFERENCES hospitals (hosp_id)";
        String query_4 = "ALTER TABLE transactions ADD CONSTRAINT hosp_id_fk_3 FOREIGN KEY (hosp_id) REFERENCES hospitals (hosp_id)";
        String query_5 = "ALTER TABLE transactions ADD CONSTRAINT doc_id_fk_2 FOREIGN KEY (doc_id) REFERENCES doctors (doc_id)";
        String query_6 = "ALTER TABLE transactions ADD CONSTRAINT diag_id_fk_1 FOREIGN KEY (diag_id) REFERENCES diagnoses (diag_id)";
        String query_7 = "ALTER TABLE transactions ADD CONSTRAINT pat_id_fk_1 FOREIGN KEY (pat_id) REFERENCES patients (pat_id)";
        String query_8 = "ALTER TABLE transactions ADD CONSTRAINT prod_id_fk_1 FOREIGN KEY (prod_id) REFERENCES products (prod_id)";
        String query_9 = "ALTER TABLE invoices ADD CONSTRAINT hosp_id_fk_4 FOREIGN KEY (hosp_id) REFERENCES hospitals (hosp_id)";
        String query_10 = "ALTER TABLE invoices ADD CONSTRAINT doc_id_fk_3 FOREIGN KEY (doc_id) REFERENCES doctors (doc_id)";
        String query_11 = "ALTER TABLE invoices ADD CONSTRAINT diag_id_fk_2 FOREIGN KEY (diag_id) REFERENCES diagnoses (diag_id)";
        String query_12 = "ALTER TABLE invoices ADD CONSTRAINT pat_id_fk_2 FOREIGN KEY (pat_id) REFERENCES patients (pat_id)";
        String query_13 = "ALTER TABLE invoices ADD CONSTRAINT prod_id_fk_2 FOREIGN KEY (prod_id) REFERENCES products (prod_id)";
        String query_14 = "ALTER TABLE invoices ADD CONSTRAINT trans_id_fk_1 FOREIGN KEY (trans_id) REFERENCES transactions (trans_id)";
        stmt.executeUpdate(query_1);
        stmt.executeUpdate(query_2);
        stmt.executeUpdate(query_3);
        stmt.executeUpdate(query_4);
        stmt.executeUpdate(query_5);
        stmt.executeUpdate(query_6);
        stmt.executeUpdate(query_7);
        stmt.executeUpdate(query_8);
        stmt.executeUpdate(query_9);
        stmt.executeUpdate(query_10);
        stmt.executeUpdate(query_11);
        stmt.executeUpdate(query_12);
        stmt.executeUpdate(query_13);
        stmt.executeUpdate(query_14);
        System.out.println("Foreign Keys added successfully.");
    } catch (ClassNotFoundException cnfe) {
        cnfe.printStackTrace();
    } catch (SQLException se) {
        se.printStackTrace();
    } catch (Exception ex) {
        ex.printStackTrace();
    } finally {
        try {
            if(stmt != null) {
                stmt.close();
            }
        } catch (SQLException se) {
            se.printStackTrace();
        }
        try {
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException se) {
            se.printStackTrace();
        }
    }
}
```

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle 19c - GitHub Project~1
- Worksheet:** SP\_ORDERS\_STAT\_DATA
- Code:**

```
CREATE OR REPLACE PROCEDURE sp_orders_stat_data
(
    v_mean_val    OUT NUMBER,
    v_median_val  OUT NUMBER,
    v_stddev_val   OUT NUMBER
)
AS
BEGIN
    SELECT AVG(ord_total), MEDIAN(ord_total), STDDEV(ord_total)
    INTO v_mean_val, v_median_val, v_stddev_val
    FROM orders;
    Dbms_Output.Put_Line('mean_val: ' || TO_CHAR(v_mean_val, '$99,999.9999'));
    Dbms_Output.Put_Line('median_val ' || TO_CHAR(v_median_val, '$99,999.9999'));
    Dbms_Output.Put_Line('stddev_val ' || TO_CHAR(v_stddev_val, '$999,999.9999'));
END sp_orders_stat_data;
```

- Script Output:** Task completed in 0.063 seconds
- Output:** Procedure SP\_ORDERS\_STAT\_DATA compiled
- Dbms Output:**

```
Connecting to the database Oracle 19c - GitHub Project.
mean_val: $70,083.3333
median_val $19,437.5000
stddev_val $119,390.1605
Process exited.
```

The screenshot illustrates the creation of a PL/SQL procedure named SP\_ORDERS\_STAT\_DATA that calculates the mean, median, and standard deviation of the ord\_total column from the orders table. The results are then displayed using DBMS\_OUTPUT.PUT\_LINE statements.

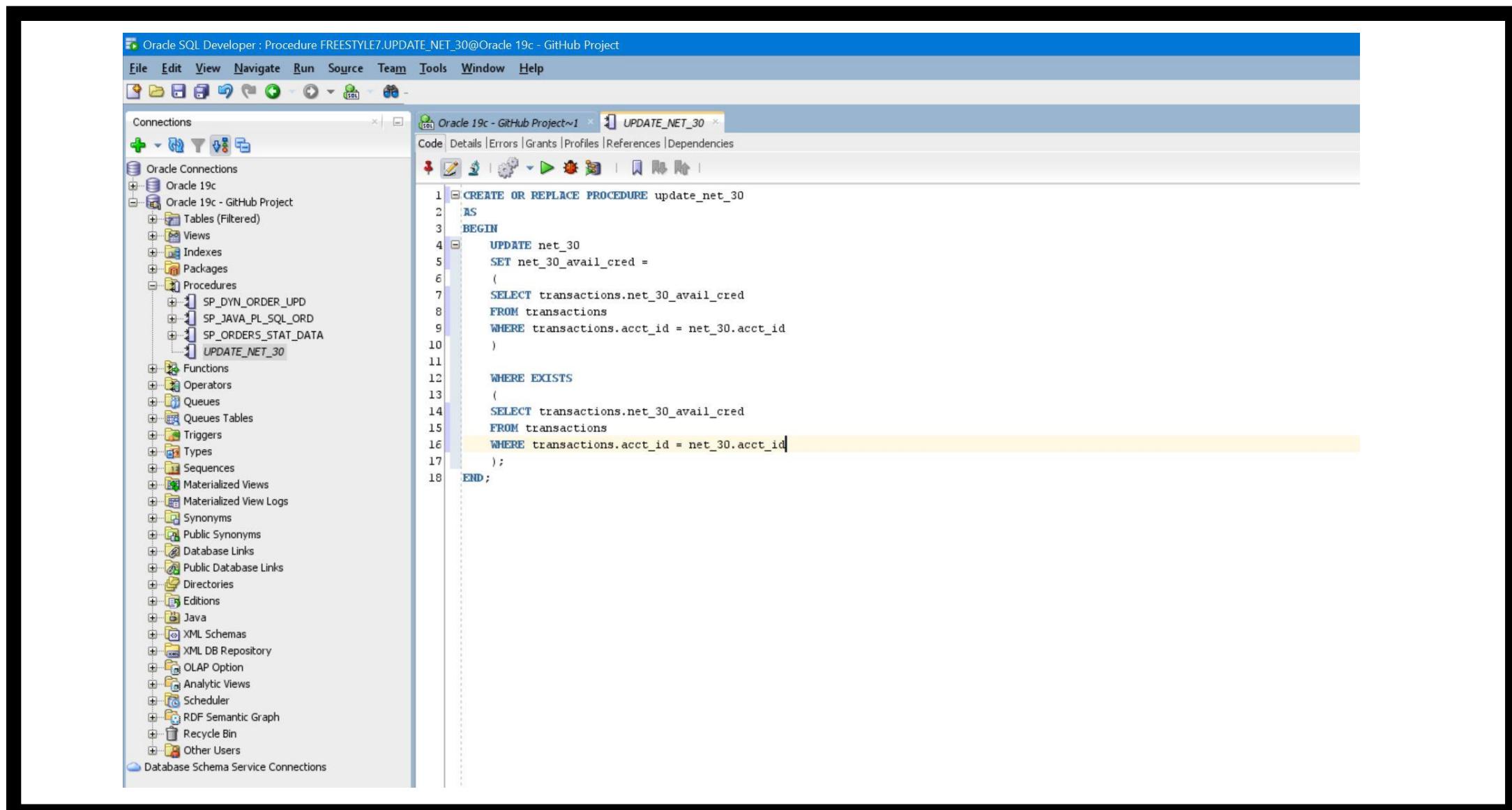
The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Oracle 19c - GitHub Project~1". The left pane displays the "Connections" tree, which includes "Oracle Connections", "Oracle 19c", and "Oracle 19c - GitHub Project", with "Tables (Filtered)" expanded to show numerous tables such as ACCOUNTS, BANK\_ACCOUNTS, CATALOG, CREDIT\_CARDS, EMPLOYEES, INVENTORY, INVOICES, LEADS, NET\_30, ORDERS, PAYMENTS, SALES\_DATA, SHIPMENTS, and TRANSACTIONS, each with a timestamp suffix like \_2\_150421.

The right pane contains a "Worksheet" tab where PL/SQL code is being written:

```
1 ALTER TABLE bank_accounts
2 MODIFY (bank_name ENCRYPT);
3
4 ALTER TABLE bank_accounts
5 MODIFY (routing_num ENCRYPT);
6
7 ALTER TABLE bank_accounts
8 MODIFY (bank_acct_num ENCRYPT);
9
10 ALTER TABLE credit_cards
11 MODIFY (cc_num ENCRYPT);
12
13 ALTER TABLE credit_cards
14 MODIFY (exp_date ENCRYPT);
15
16 ALTER TABLE credit_cards
17 MODIFY (cvv ENCRYPT);
18
19 ALTER TABLE credit_cards
20 MODIFY (cc_first ENCRYPT);
21
22 ALTER TABLE credit_cards
23 MODIFY (cc_last ENCRYPT);
24
25 ALTER TABLE net_30
26 MODIFY (bank_check_num ENCRYPT);
27
28 ALTER TABLE transactions
29 MODIFY (bank_name ENCRYPT);
```

SQL> SELECT \* FROM dba\_encrypted\_columns;

OWNER	TABLE_NAME	COLUMN_NAME	ENCRYPTION_ALG	SAL	INTEGRITY_ALG
FREESTYLE7	BANK_ACCOUNTS	BANK_NAME	AES 192 bits key	YES	SHA-1
FREESTYLE7	BANK_ACCOUNTS	ROUTING_NUM	AES 192 bits key	YES	SHA-1
FREESTYLE7	BANK_ACCOUNTS	BANK_ACCT_NUM	AES 192 bits key	YES	SHA-1
FREESTYLE7	CREDIT_CARDS	CC_NUM	AES 192 bits key	YES	SHA-1
FREESTYLE7	CREDIT_CARDS	EXP_DATE	AES 192 bits key	YES	SHA-1
FREESTYLE7	CREDIT_CARDS	CVV	AES 192 bits key	YES	SHA-1
FREESTYLE7	CREDIT_CARDS	CC_FIRST	AES 192 bits key	YES	SHA-1
FREESTYLE7	CREDIT_CARDS	CC_LAST	AES 192 bits key	YES	SHA-1
FREESTYLE7	NET_30	BANK_CHECK_NUM	AES 192 bits key	YES	SHA-1
FREESTYLE7	TRANSACTIONS	BANK_NAME	AES 192 bits key	YES	SHA-1



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema structure under 'Connections' for 'Oracle 19c - GitHub Project'. The 'Procedures' section is expanded, showing several procedures including 'SP\_DYN\_ORDER\_UPD', 'SP\_JAVA\_PL\_SQL\_ORD', 'SP\_ORDERS\_STAT\_DATA', and 'UPDATE\_NET\_30'. The right pane shows the code editor for the 'UPDATE\_NET\_30' procedure. The code is as follows:

```
1 CREATE OR REPLACE PROCEDURE update_net_30
2 AS
3 BEGIN
4   UPDATE net_30
5   SET net_30_avail_cred =
6   (
7     SELECT transactions.net_30_avail_cred
8     FROM transactions
9     WHERE transactions.acct_id = net_30.acct_id
10    )
11
12   WHERE EXISTS
13   (
14     SELECT transactions.net_30_avail_cred
15     FROM transactions
16     WHERE transactions.acct_id = net_30.acct_id
17   );
18 END;
```

The screenshot shows the Oracle SQL Developer interface. On the left is the Connections sidebar with 'Oracle 19c - GitHub Project' selected. In the center, the Worksheet pane displays the following PL/SQL code:

```
BEGIN
  Dbms_Scheduler.create_job
  (
    JOB_NAME      => 'update_net_30_avail_credit',
    JOB_TYPE       => 'PLSQL_BLOCK',
    JOB_ACTION     => 'transactions_net_30_avail_cred_update_net_30_avail_cred.sql',
    START_DATE     => '10-JUN-2021 09:00:00 AM',
    REPEAT_INTERVAL => 'FREQ=HOURLY',
    END_DATE       => NULL,
    ENABLED        => TRUE,
    AUTO_DROP      => FALSE,
    COMMENTS        => 'Update net_30 Available Credit from Transactions Table'
  );
END;
/
```

The Scheduler node under Oracle 19c - GitHub Project is expanded, showing the 'Jobs' node which contains the 'UPDATE\_NET\_30\_AVAIL\_CREDIT' job. The Script Output pane at the bottom right shows the message: "PL/SQL procedure successfully completed." and "Elapsed: 00:00:00.007".

## SQL Plus

```
SQL> SELECT TO_CHAR(log_id, '9999') log_id,
  2 TO_CHAR(log_date, 'YYYY-MM-DD HH12:MM:SS') timestamp, owner, job_name, status
  3 FROM all_scheduler_job_run_details
  4 WHERE owner = 'FREESTYLE7'
  5 ORDER BY log_date;
```

LOG_ID	TIMESTAMP	OWNER	JOB_NAME	STATUS
2854	2021-06-10 10:06:52	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2856	2021-06-10 10:06:46	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2858	2021-06-10 10:06:01	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2864	2021-06-10 11:06:12	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2874	2021-06-10 11:06:44	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2884	2021-06-11 11:06:22	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2904	2021-06-11 11:06:35	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2912	2021-06-11 12:06:52	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2922	2021-06-11 01:06:52	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED
2930	2021-06-11 02:06:52	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	SUCCEEDED

## SQL Plus

```
SQL> SELECT TO_CHAR(start_date, 'YYYY-MM-DD HH12:MM:SS') timestamp, owner, job_name, job_class, state, repeat_interval, job_type
  2 FROM all_scheduler_jobs
  3 WHERE owner = 'FREESTYLE7';
```

TIMESTAMP	OWNER	JOB_NAME	JOB_CLASS	STATE	REPEAT_INTERVAL	JOB_TYPE
2021-06-10 10:06:52	FREESTYLE7	UPDATE_NET_30_AVAIL_CRED	DEFAULT_JOB_CLASS	SCHEDULED	FREQ=HOURLY	PLSQL_BLOCK

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Oracle 19c - GitHub Project". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, Print, and Database navigation.

The Connections pane on the left shows "Oracle Connections" with "Oracle 19c" and "Oracle 19c - GitHub Project" selected. Under "Tables (Filtered)", there is a large list of tables including ACCOUNTS, ACCOUNTS\_2\_150421, BANK\_ACCOUNTS, BANK\_ACCOUNTS\_2\_150421, CATALOG, CATALOG\_2\_150421, CATALOG\_CLASS, CATALOG\_CLASS\_2\_150421, CREDIT\_CARDS, CREDIT\_CARDS\_2\_150421, DYN\_ORDER\_UPD, DYN\_ORD\_ID, DYN\_ACCT\_ID, DYN\_QUOTE\_ID, DYN\_CAT\_ID, DYN\_ORDER\_UPD\_2\_150421, EMPLOYEES, EMPLOYEES\_2\_150421, INVENTORY, INVENTORY\_2\_150421, INVOICES, INVOICES\_2\_150421, JAVA\_PL\_SQL\_ORD, JAVA\_PL\_SQL\_ORD\_2\_150421, LEADS, LEADS\_2\_150421, NET\_30, NET\_30\_2\_150421, NET\_30\_INFO, NET\_30\_INFO\_2\_150421, ORDERS, ORDERS\_2\_150421, ORDERS\_STAT\_DATA, ORDERS\_STAT\_DATA\_2\_150421, PAYMENTS, PAYMENTS\_2\_150421, SALES\_DATA, SHIPMENTS, SHIPMENTS\_2\_150421, TRANSACTIONS, and TRANSACTIONS\_2\_150421.

The central workspace displays the PL/SQL code for the package DYN\_ORDER\_UPD:

```
1 CREATE OR REPLACE PACKAGE pkg_dyn_order_upd
2 IS
3     FUNCTION prnt_string
4         RETURN varchar2;
5     PROCEDURE sp_dyn_order_upd;
6 END pkg_dyn_order_upd;
7
8 CREATE OR REPLACE PACKAGE BODY pkg_dyn_order_upd
9 IS
10    FUNCTION prnt_string
11        RETURN varchar2 IS
12        BEGIN
13            RETURN 'Dynamic Order Update data inserted successfully.';
14        END prnt_string;
15
16    PROCEDURE sp_dyn_order_upd IS
17        BEGIN
18            INSERT INTO dyn_order_upd (dyn_ord_id, dyn_acct_id, dyn_quote_id, dyn_cat_id)
19            VALUES (dyn_ord_id_SEQ.NEXTVAL, dyn_acct_id_SEQ.NEXTVAL, dyn_quote_id_SEQ.NEXTVAL, dyn_cat_id_SEQ.NEXTVAL);
20            COMMIT;
21        END;
22    END pkg_dyn_order_upd;
23
24 BEGIN
25     Dbms_Output.Put_Line(pkg_dyn_order_upd.sp_dyn_order_upd);
26 END;
27
28 BEGIN
29     pkg_dyn_order_upd.sp_dyn_order_upd;
30 END;
```

SQL Plus

```
SQL> SELECT * FROM dyn_order_upd
  2 ORDER BY dyn_ord_id;
DYN_ORD_ID DYN_ACCT_ID DYN_QUOTE_ID DYN_CAT_ID
----- ----- ----- -----
43210001 33330001 44440001 12340001
43210002 33330002 44440002 12340002
43210003 33330003 44440003 12340003
43210004 33330004 44440004 12340004
43210005 33330005 44440005 12340005
43210006 33330006 44440006 12340006
```

SQLQuery6.sql - DESKTOP-B33SJ7.Family\_Practice (djmsmith7 (51)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect... New Query Execute

Family\_Practice

```
1 1 --DECLARE @sqlText VARCHAR(MAX)
2 2 SET @sqlText = ''
3 3 SELECT @sqlText = @sqlText + ' SELECT * FROM ' + QUOTENAME(name) + CHAR(30) FROM sys.tables ORDER BY name
4 4 EXEC(@sqlText)
```

Results

InvID	TransID	PatID	ProdID	ProdDesc	ProdQty	Invoiced	InvDate
1	33330001	11110001	22220001	COVID-19 Test Kit	1	Invoiced	2021-12-12
2	33330002	11110002	22220002	Bed Linen	1	Invoiced	2021-12-12
3	33330003	11110003	22220003	Adjustable Bed	1	Invoiced	2021-12-12
4	33330004	11110004	22220004	Hydrotherapy Fitness	1	Invoiced	2021-12-12
5	33330005	11110005	22220005	Glucose Monitor	1	Invoiced	2021-12-12

PatID	FirstName	LastName	Patient_St_Num	PatientStreet	PatientCity	PatientState	PatientZip	PatientPhone	PatientEmail
1	Peter	Smith	8561	Kennedy Alley	Johnstown	SC	82834	9243799521	peter.smith@yahoo.com
2	William	Parish	128	Jessica Park	West Edward	CO	84036	4710718185	william.parish@yahoo.com
3	Joshua	Perez	33727	Daniel Locks	East Kaitlyn	RI	46820	5668042289	joshua.perez@hotmail.com
4	Vanessa	Dicker	794	Jennifer Parkway	Port Joshua	NY	75313	5748474529	vanessa.dicker@aol.com
5	Martha	Martin	861	Richard Mills	Smithburgh	AK	51361	4047573224	martha.martin@aol.com

PayID	PatID	TransID	InvID	PayAmt	PayStatus	PayDate
1	11110001	33330001	44440001	350.00	Paid	2021-12-12
2	11110002	33330002	44440002	450.00	Paid	2021-12-12
3	11110003	33330003	44440003	4500.00	Paid	2021-12-12
4	11110004	33330004	44440004	15500.00	Paid	2021-12-12
5	11110005	33330005	44440005	950.00	Paid	2021-12-12

ProdID	ProdDesc	ProdPrice	ProdClass
1	COVID-19 Test Kit	350.00	COVID-19 Supplies
2	Bed Linen	450.00	Patient Care
3	Adjustable Bed	4500.00	Patient Mobility
4	Hydrotherapy Fitness	15500.00	Rehabilitation
5	Glucose Monitor	950.00	Diabetic Products

TransID	PatID	ProdID	ProdDesc	ProdQty	ProdPrice	TransAmt	TransDate	
1	33330001	11110001	22220001	COVID-19 Test Kit	1	350.00	350.00	2021-12-12
2	33330002	11110002	22220002	Bed Linen	1	450.00	450.00	2021-12-12
3	33330003	11110003	22220003	Adjustable Bed	1	4500.00	4500.00	2021-12-12
4	33330004	11110004	22220004	Hydrotherapy Fitness	1	15500.00	15500.00	2021-12-12
5	33330005	11110005	22220005	Glucose Monitor	1	950.00	950.00	2021-12-12

VendorOrdID	VendorID	ProdID	VendorName	ProdQty	ProdPrice	VendorTransAmt	VendorTransDate	
1	88880001	77770001	22220001	Sammina	5	300.00	1500.00	2021-12-12
2	88880002	77770002	22220002	McKesson	10	400.00	4000.00	2021-12-12
3	88880003	77770003	22220003	Medtronic	15	4000.00	60000.00	2021-12-12
4	88880004	77770004	22220004	Cardinal Health	20	15000.00	300000.00	2021-12-12
5	88880005	77770005	22220005	3M Healthcare	25	900.00	22500.00	2021-12-12

VendorID	ProdID	VendorName	Vendor_St_Num	VendorStreet	VendorCity	VendorState	VendorZip	VendorPhone
1	77770001	22220001	Sammina	2700	North First Street	San Jose	CA	95134 4089643500
2	77770002	22220002	McKesson	6535	Texas State Highway	Irving	TX	75039 9724464800
3	77770003	22220003	Medtronic	710	Medtronic Parkway	Minneapolis	MN	55432 8086338766
4	77770004	22220004	Cardinal Health	7000	Cardinal Place	Dublin	OH	43017 6147575000
5	77770005	22220005	3M Healthcare	10351	West 78th Street	Eden Pra...	MN	55344 8002283957

Query executed successfully.

# SQL Server | Primary & Foreign Keys

42

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays a tree view of database objects for the 'Family\_Practice' database, including tables like 'Invoices', 'Payments', and 'Products'. The main window contains a query editor with the following SQL script:

```
1 SELECT TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE
2 FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
3 ORDER BY TABLE_NAME;
```

The results grid on the right shows the output of the query:

TABLE_NAME	CONSTRAINT_NAME	CONSTRAINT_TYPE
Invoices	FK_TransID_1	FOREIGN KEY
Invoices	FK_PatID_1	FOREIGN KEY
Invoices	FK_ProdID_1	FOREIGN KEY
Invoices	PK_InvID	PRIMARY KEY
Patients	PK_PatID	PRIMARY KEY
Payments	PK_PayID	PRIMARY KEY
Payments	FK_PatID_2	FOREIGN KEY
Payments	FK_TransID_2	FOREIGN KEY
Payments	FK_InvID_1	FOREIGN KEY
Products	PK_ProdID	PRIMARY KEY
Transactions	PK_TransID	PRIMARY KEY
Transactions	FK_PatID_3	FOREIGN KEY
Transactions	FK_ProdID_2	FOREIGN KEY
Vendor_Orders	FK_VendorID_1	FOREIGN KEY
Vendor_Orders	FK_ProdID_3	FOREIGN KEY
Vendor_Orders	PK_VendorOrdID	PRIMARY KEY
Vendors	PK_VendorID	PRIMARY KEY

At the bottom of the results grid, a message states "Query executed successfully."

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database 'Family\_Practice' and its objects. The central pane contains three T-SQL scripts labeled 1 through 19, which calculate the median, standard deviation, and mean of transaction amounts. The results pane at the bottom shows the output for each query.

```
1 DECLARE @c BIGINT = (SELECT COUNT(*) FROM Transactions);
2
3 SELECT CAST(ROUND(AVG(1.0 * TransAmt),4) AS DEC(10,4)) Trans_Amt_Mean
4 FROM
5 (SELECT TransAmt FROM Transactions
6 ORDER BY TransAmt
7 OFFSET (@c -1) / 2 ROWS
8 FETCH NEXT 1 + (1 - @c % 2) ROWS ONLY
9 )
10 AS x;
11
12
13
14 SELECT CAST(ROUND(STDEV(TransAmt),4) AS DEC(10,4)) Trans_Amt_STDDEV
15 FROM Transactions;
16
17
18 SELECT CAST(ROUND(AVG(DISTINCT TransAmt),4) AS DEC(10,4)) Trans_Amt_Mean
19 FROM Transactions;
```

	Results
Trans_Amt_Median	950.0000
Trans_Amt_STDDEV	6463.6484
Trans_Amt_Mean	4350.0000

Query executed successfully.